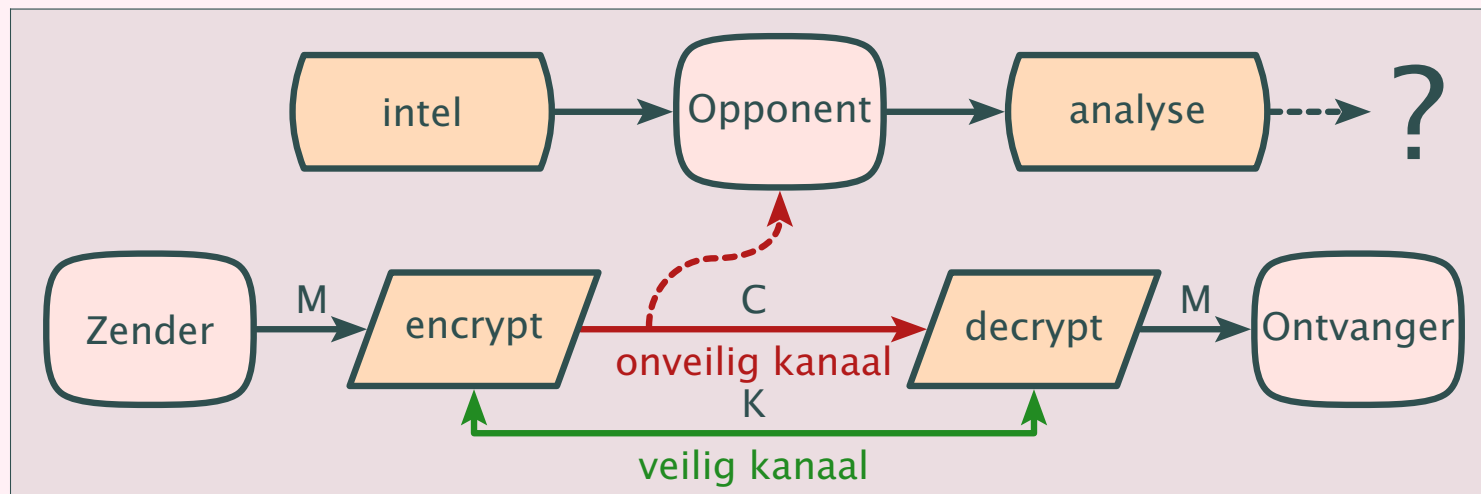


SYLLABUS CRYPTOGRAFIE

Beta studies



dr. H. van der Meer

Editie 2016

© dr. H. van der Meer, 2016. Niets uit deze uitgave mag worden verveelvoudigd en/of openbaar gemaakt door middel van druk, fotocopie, microfilm of op welke andere wijze dan ook, zonder voorafgaande schriftelijke toestemming.

Inhoud

1 Inleiding	1
2 Transpositie	59
3 Monoalfabetische systemen	78
4 Polyalfabetische systemen	117
5 Beroemde vercijfermachines	162
6 Cryptoanalyse	184
7 Informatietheorie	244
8 Moderne systemen	303
9 Publieke sleutel	405
10 Sleutelstroom	454
A Verklarende woordenlijst	541
B Index	548

1 Inleiding

1.1	Korte inhoud	3
1.2	Klassieke cryptografie	5
1.2.1	Cryptosysteem	5
1.3	COMSEC	7
1.3.1	Encryptie in netwerken	10
1.4	Publieke sleutel	12
1.5	Quantum cryptografie	14
1.5.1	Sleutелuitwisseling	16
1.5.2	Afluisterdetectie	18
1.6	Veiligheid en bruikbaarheid	21
1.6.1	Voorwaarden	22
1.6.2	Veilig	25
1.6.3	Methoden van cryptoanalyse	28
1.7	Eigenschappen	34
1.7.1	Gesloten systeem	34
1.7.2	Involutie	36
1.7.3	Endomorfie	36
1.7.4	Idempotent	36
1.7.5	Restklassen	37
1.7.6	MASTERSECTIE – Zuiver cryptosysteem	39
1.7.7	Commuteren	40

1.7.8	Gelijksoortigheid	42
1.8	Blokgeheimschrift	43
1.8.1	Implementatie aspecten	44
1.8.2	Beperkte range	48
1.9	Literatuurlijst	50
1.10	Referenties	57

1.1 Korte inhoud

Dit boek heeft de bedoeling een breed en grondig overzicht te geven van de cryptografie. Dat wil zeggen van het ontwerpen en gebruiken van geheimschrift. Naar zal blijken wordt in de cryptografie gebruik gemaakt van een veelheid aan technieken, die soms vrij uitgebreid en soms schetsmatig zullen worden behandeld. Niet alleen moderne cryptosystemen, maar ook oudere systemen worden gepresenteerd. De rol die deze laatste in de wereldgeschiedenis gespeeld hebben is op zich al genoeg rechtvaardiging daarvoor. Bovendien kunnen de grondbeginselen van het vak er eenvoudig mee worden uitgelegd, grondbeginselen die ook de basis voor de meeste moderne systemen uit het computertijdperk vormen.

Eerst een beknopte inhoudsbeschrijving van elk hoofdstuk.

1. In het eerste hoofdstuk wordt uitgelegd wat *cryptografie* is. Een aantal grondbegrippen wordt geïntroduceerd en het belang ervan aangegeven.
2. Het tweede t/m vijfde hoofdstuk behandelen de *klassieke cryptografie*. Historisch gezien omvat dit cryptografische technieken vanaf ongeveer de tijd van de Renaissance tot aan het begin van het computertijdperk. Uit de veelheid van cryptosystemen ooit bedacht, moest een keuze worden gemaakt. Gekozen is voor systemen waarmee de hoofdlijnen van de cryptografie en haar technieken het best kunnen worden geïllustreerd. Het is een bont gezelschap en illustreert zodoende de grote verscheidenheid aan cryptosystemen die ooit het licht hebben gezien. Enkele vercijfermachines zijn zo beroemd dat alleen al daarom een behandeling op zijn plaats is. De Japanse Purple, de Zweedse Hagelin en de Duitse Enigma verdienen die aandacht niet alleen vanwege hun historische betekenis, maar ook omdat ze drie verschillende principes belichamen.
3. Geheimen wekken onvermijdelijk de nieuwsgierigheid van anderen op. Het is dus niet verwonderlijk dat behalve technieken om geheimen te verbergen ook methoden zijn ontwikkeld om achter die geheimen te komen. De *cryptoanalyse* houdt zich bezig met het ontsluiten van geheimschrift. Het belang van een

kennismaking daarmee is tweeledig: (1) het geeft inzicht in de kracht en zwakte van cryptografische technieken, en (2) het bewerkstelligt een gezonde scepsis tegenover de claim van onbreekbaarheid waarmee ontwerpers en verkopers van geheimschriften hun producten soms plegen aan te prijzen. Dit hoofdstuk is gewijd aan statistische en andere technieken waarmee geheimschriften te lijf gegaan kunnen worden en wordt afgesloten met enkele voorbeeldanalyses.

4. Vervolgens wordt ingegaan op de theoretische achtergrond van het geheimschrift. In het bijzonder zal het gaan om de vraag onder welke omstandigheden een geheimschrift in principe kan worden gebroken. Centraal hierin staat de *informatietheorie* van de Amerikaan Shannon waarmee informatie en de overdracht van informatie kunnen worden gekwantificeerd.
5. In het dan volgende hoofdstuk wordt het moderne klassieke *blokgeheimschrift* behandeld. Dit zijn de systemen waarmee heden ten dage informatie wordt beschermd tegen ongewenste nieuwsgierige blikken. De Data Encryption Standard (DES) is het oudste en meest bekende van deze systemen. Het belang van dit systeem schuilt niet alleen in de gegevensbescherming die het vele jaren heeft geboden, maar de ontwikkeling ervan heeft ook geleid tot meer en diepgaand inzicht in de eigenschappen van moderne cryptosystemen. Na een vluchtige blik op enige andere systemen wordt dit hoofdstuk besloten met een bespreking van de thans geldende standaard voor de vercijfering van informatie, de Advanced Encryption Standard (AES).
6. Systemen met zogenaamde *publieke sleutel* verschillen wezenlijk van de systemen uit de klassieke cryptografie. Ze dienen onder andere als basis voor mechanismen om de authenticiteit van documenten en van gebruikers in computers en computernetwerken vast te stellen. Men zou zelfs kunnen stellen dat in het internet-tijdperk zonder deze systemen met publieke sleutel geen goede beveiliging valt te realiseren. Hun wortels liggen in de wiskunde, meer in het bijzonder in de getaltheorie.
7. De systemen uit de voorgaande hoofdstukken zetten alle tekens of groepjes bytes om in geheimschrift. Men kan een gegevensstroom echter ook opvatten als een rij individuele bits en de beveiliging op dit

niveau aanpakken. Een voor de hand liggend toepassingsgebied is dat van de mobiele telefonie. De techniek waarmee dit efficiënt kan worden gerealiseerd maakt gebruik van zogenaamde *schuifregisters*. In dit hoofdstuk zullen de principes alsmede sterkte en zwakte ervan worden behandeld.

1.2 Klassieke cryptografie

Het woord *cryptografie* is afgeleid van twee Griekse woorden: *κρυπτος* met de betekenis *verborgen*, *geheim* en *γραφω* dat *schrijven* betekent. Cryptografie is dus zoiets als *geheimschrijven*. Niet voor niets heette vroeger de speciaal met de geheime correspondentie belaste functionaris de geheimschrijver.

Naast cryptografie bestaat ook de zogenaamde *steganografie*. Het onderliggende Griekse woord is hier *στεγανος* of *bedekkend*. In de steganografie wordt een boodschap verscholen aangebracht in een drager. Enkele mogelijkheden: in een tekst worden letters gemarkeerd die samen het verborgen bericht vormen, in een gedigitaliseerde foto worden de minst significante bits gemodificeerd waarbij de veranderingen zo gering zijn dat het oog ze niet opmerkt. Het aantal variaties is legio, bijvoorbeeld een tekening waarin lange en korte grashalmen de punten en strepen van een bericht in morsecode voorstellen, of een groepsfoto van personen met het gezicht recht naar voren of naar opzij waarmee nullen en enen worden gecodeerd, of een aantal (niet zichtbare) spaties aan het eind van een regel, enzovoorts, enzovoorts.

1.2.1 Cryptosysteem

Het omzetten van een bericht in geheimschrift maakt er een *cryptogram* van (ciphertext in het Engels). Deze operatie wordt *encryptie* of *vercijferen* genoemd. Het woord vercijferen is terug te voeren op het feit dat in voorgaande eeuwen vooral in het diplomatiek berichtenverkeer een systeem werd gebruikt waarin een bericht werd omgezet in groepen cijfers. De oorspronkelijke tekst heet de *klaartekst* (plaintext in het Engels). Het cryptogram zal ook de *cijfertekst* genoemd worden. Het *ontcijferen* of de *decryptie* is

de omgekeerde operatie, waarin uit het cryptogram de klaartekst wordt teruggewonnen. Veelal ziet men de klaartekst aangeduid met de letter M van het Engelse woord *message* en het cryptogram met een C . Hiermee laten vercijferen en ontcijferen zich respectievelijk als volgt in formules vatten

$$C = E(M, K) \quad \text{en} \quad M = D(C, K) \quad (1.1)$$

Naast de klaartekst en de cijfertekst duikt nog een derde grootheid op, de *vercijfersleutel* K van het Engelse woord *key*. Deze vercijfersleutel kan – binnen de begrenzings van het systeem – vrij worden gekozen. Het gevolg is dat één enkel bericht door het kiezen van een wisselende sleutel in verschillende cryptogrammen kan worden omgezet. Zonder de door de sleutel geboden variatie zou bij elke klaartekst maar één cryptogram behoren. Een buitenstaander zou dan ook maar één keer de betekenis ervan behoeven te zien om het voortaan altijd te kunnen herkennen. Door het variëren van de sleutel kan een bericht meerdere keren worden vercijferd zonder dat dit direct uit de cryptogrammen blijkt*.

In het algemeen zal zowel bij het vercijferen als bij het ontcijferen dezelfde vercijfersleutel nodig zijn. Dat is niet het geval bij de hiervoor genoemde systemen met publieke sleutel. Deze systemen gebruiken afzonderlijke sleutels voor het vercijferen en voor het ontcijferen. Verderop in dit hoofdstuk wordt daarop teruggekomen.

De wijze van vercijferen en ontcijferen – de algoritme van het geheimschrift – samen met de daarvoor geldende verzameling sleutels heet een *cryptosysteem*. Een cryptosysteem waarin één en dezelfde sleutel zowel voor vercijferen als voor ontcijferen dient heet een *klassiek cryptosysteem* of ook wel een *symmetrisch cryptosysteem*. Als verschillende sleutels worden gebruikt spreekt men van een *asymmetrisch cryptosysteem*.

Bij het gebruik van cryptografie kan men zich natuurlijk afvragen hoe het met de veiligheid van het ge-

* Een waarschuwing is hier echter op z'n plaats. Het onveranderd hervercijferen van een bericht met een andere sleutel levert toch nog gevaar op voor de geheimhouding. Daarom moet men het bericht anders formuleren alvorens het opnieuw te vercijferen. Een regel waartegen in de loop der tijden nogal eens is gezondigd met soms desastreuze gevolgen.

bruikte cryptosysteem gesteld is. Weliswaar bestaat er een theoretisch onbreekbaar geheimschrift, maar dat is niet altijd praktisch in het gebruik en er zijn omstandigheden waarin toepassing ervan niet goed mogelijk is. Alle andere systemen kunnen in principe worden gebroken. Dat wil zeggen dat de vercijfersleutel kan worden achterhaald door analyse van onderschepte cryptogrammen. Het breken van geheimschriften wordt met de term *cryptoanalyse* aangeduid (cryptanalysis in het Engels). Wie een sterk geheimschrift wil ontwerpen doet er goed aan om zich eerst terdege te verdiepen in de cryptoanalyse. Pas dan leert men de sterke en zwakke punten van geheimschriften goed kennen. De beroemde twintigste-eeuwse Amerikaanse cryptograaf en cryptoanalist William Friedman gaf aan de combinatie van cryptografie en cryptoanalyse de naam *cryptologie* (cryptology in het Engels).

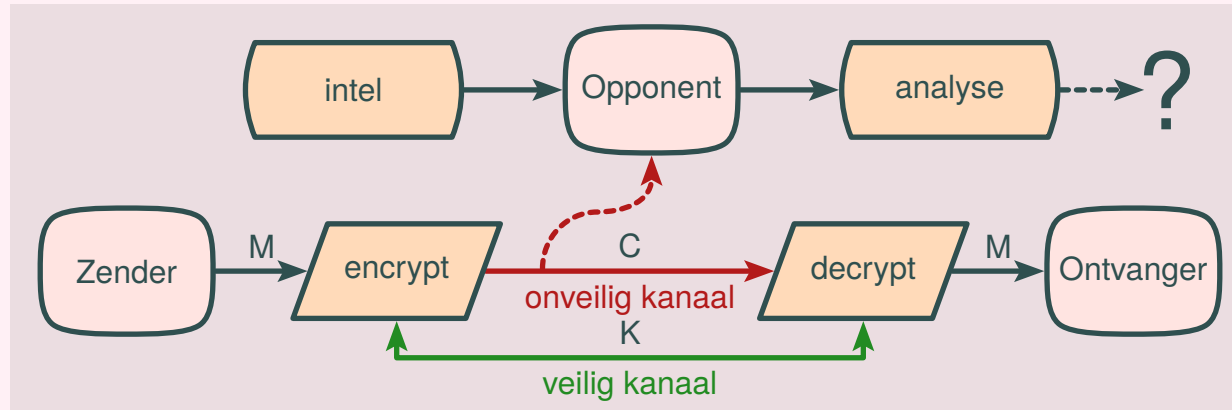
1.3 COMSEC

COMSEC is de afkorting van *communications security* ofwel beveiligde communicatie*. Cryptografie biedt de mogelijkheid om die te realiseren, maar het is niet zo eenvoudig als het simpelweg uitkiezen van een geheimschrift. Integendeel, een goed werkend en blijvend veilig communicatiesysteem opzetten is geen sinecure en er liggen veel voetangels en klemmen.

In [figuur 1.1](#) is de problematiek geschetst waarmee de deelnemers van door cryptografie beveiligde communicatie te maken hebben. Er is een bericht M dat moet worden verstuurd naar een ontvanger. De verbinding waarover de berichten worden verstuurd kan echter door een tegenstander worden afgeluisterd. Men zegt daarom dat transport van het bericht geschiedt over een *onveilig kanaal*. Omzetten van het bericht in geheimschrift verhindert weliswaar niet dat kan worden afgeluisterd, maar kan er wel voor zorgen dat de tegenstander geen kennis krijgt van de inhoud.

In een klassiek cryptosysteem geschiedt het vercijferen en ontcijferen met dezelfde berichtsleutel. Dat schept een probleem voor de zender en ontvanger, omdat zij uiteraard beiden over die sleutel moeten

* De verwante afkorting SIGINT staat voor signals intelligence.



Figuur 1.1 Communicatie beveiligd door cryptografie

kunnen beschikken. Daartoe moeten zij de sleutel – die vanzelfsprekend geheim moet blijven – uitwisselen. De standaardverbinding kan hiervoor niet worden gebruikt, want die is immers per definitie onveilig. Het lijkt een kip-en-ei probleem: zonder uitwisseling van de sleutel geen beveiligde communicatie en zonder beveiligde communicatie geen veilige sleuteluitwisseling.

Dit dilemma moet in het geval van het klassiek cryptosysteem worden opgelost door de introductie van een afzonderlijke, gegarandeerd veilige verbinding. Over dit tweede en *veilig kanaal* wordt een sleutel uitgewisseld, waarmee vervolgens berichten over de onveilige verbinding verstuurd kunnen worden. Verderop zal worden getoond hoe een cryptosysteem op basis van twee afzonderlijke sleutels zo'n veilige verbinding kan verschaffen. Een andere manier is om een koerier met de sleutel op pad te sturen. En sinds kort kan men ook quantumcryptografie gebruiken.

Is er veel berichtenverkeer dan is één enkele berichtssleutel niet goed genoeg. In het hoofdstuk over informatietheorie zal blijken dat het vercijferen van veel en/of lange berichten met slechts één enkele sleutel aangrijpingspunten biedt voor cryptanalyse. Om dit te voorkomen moet een communicatiesysteem worden geschapen waarin wisselende sleutels worden gebruikt en zelfs dat moet met beleid geschieden.

Het is verleidelijk om een systeem te scheppen waarin voor een zekere periode sleutels kunnen worden gemaakt uit één hoofdsleutel, die dan veel minder vaak hoeft te worden uitgewisseld. Hoe gevaarlijk dit is, ondervond Japan in de Tweede Wereldoorlog. Elke dag wisselde de sleutel, maar binnen een periode van tien dagen werd de dagelijkse sleutel uit de beginsleutel voor die periode afgeleid. Toen de Amerikaanse codebrekers dat systeem eenmaal door hadden, hoefden ze dus nog maar eens per tien dagen de sleutel te breken. Daarentegen kregen de Duitse onderzeeboten, die op de Atlantische Oceaan tegen geallieerde konvooien opereerden, sleutellijsten mee die ruimschoots voldoende waren voor de gehele tijd dat ze buitengaats zouden zijn.

De andere speler in het cryptografiespel is de tegenstander, die altijd op de loer kan liggen. Men moet er rekening mee houden dat een derde partij altijd wel op de een of andere manier toegang weet te krijgen tot het onveilig kanaal. In beginsel heeft deze tegenstrever twee tactieken tot zijn beschikking

- alleen passieve interactie, d.w.z. *afluisteren*;
- zich actief met de communicatie bemoeien en berichten spoorloos laten verdwijnen, de communicatie hinderen met stoorzenders, berichten verminken of valse berichten injecteren.

We mogen aannemen dat de tegenstander de bericht sleutel niet kent, maar dat betekent nog niet dat deze machteloos is. Cryptoanalyse wordt ingezet om onderschepte en waardevol geachte cryptogrammen te breken en zodoende achter de inhoud ervan te komen. Deze tak van sport wordt door bijna alle zichzelf serieus nemende naties bedreven. In de USA bijvoorbeeld is de National Security Agency (NSA) belast met het vergaren van informatie uit afgeluisterde en/of door cryptoanalyse ontcijferde berichten, en in de 18^e-eeuw waren er Nederlandse cryptoanalisten die zich met de besten in de toenmalige wereld konden meten.

Voor het uitvoeren van een cryptoanalyse kan de tegenstander gewoonlijk over meer informatie beschikken dan enkel het onderschepte cryptogram. Vaak is er een hele verzameling van onderschepte berichten, deels vercijferd met dezelfde sleutel en deels met verschillende sleutels. Soms hebben (delen van) berich-

ten dezelfde inhoud, maar zijn ze met verschillende sleutels gecijferd. Soms ook maakt de tegenpartij een fout die de cryptanalyse aan het rollen kan brengen. Meestal is veel aanvullende kennis over de tegenpartij beschikbaar en vaak zelfs zeer gedetailleerde informatie zowel over het gebruikte cryptosysteem als over de aard en inhoud van de gecijferde berichten*.

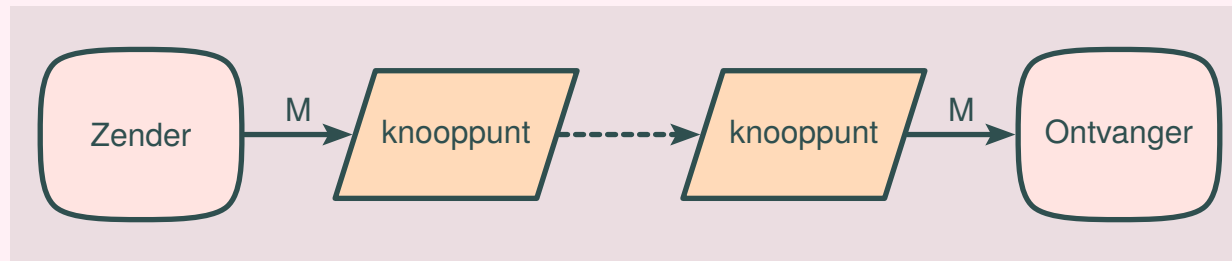
Zelfs als men er niet in slaagt om de afgeluisterde berichten te breken, dan nog valt er vaak wel wat mee te bereiken. In een militaire campagne worden talloze berichten verstuurd van hoofdkwartieren naar ondergeschikte eenheden en omgekeerd, alsook tussen die eenheden onderling. Om wijs te worden uit de wirwar van berichten zullen ze een standaardvorm hebben met allerlei administratieve gegevens in de aanhef, de zogenaamde *berichtkop* (message header in het Engels). In die berichtkop zullen afzender en bestemming vermeld staan, evenals andere belangrijke gegevens zoals de prioriteit van het bericht, de veiligheidsclassificatie, het tijdstip van aanbidding in de codekamer, enz. Analyse van de berichtkoppen, de zogenaamde *berichtenverkeersanalyse* (traffic analysis in het Engels), kan veel waardevolle informatie opleveren en vaak een belangrijk onderdeel van een SIGINT-operatie.

1.3.1 Encryptie in netwerken

In netwerken zoals het internet en netwerken voor mobiele telefonie kan encryptie op verschillende manieren worden geïmplementeerd. Een netwerk bestaat in het algemeen uit een aantal afzonderlijke netwerkknooppunten waarlangs het bericht van de zender naar de ontvanger gaat (figuur 1.2). Het bericht zal een bepaalde route nemen, afhankelijk van de plaats van de zender en ontvanger in het netwerk. Deze route hoeft niet altijd dezelfde te zijn, maar kan variëren met de conditie van het netwerk. Hieruit volgt direct dat een deel van het bericht een leesbare adressering moet bevatten – anders is geen dynamische

* Toen de Duitse generaal Rommel in Noord-Afrika optrok tegen de Engelse troepen, werden elke dag rapporten verstuurd over het aantal tanks in operationele staat, het aantal in de werkplaats en het aantal dat verloren ging. Deze berichten hadden een hoog stereotiep karakter en als gevolg daarvan een grotendeels voorspelbare inhoud.

routing mogelijk – en dus bestaat onafhankelijk van de mogelijkheid tot af luisteren altijd het gevaar van berichtenverkeersanalyse.



Figuur 1.2 Encryptie in netwerken

Belangrijk voor de beveiliging van de communicatie in een netwerk is de keuze voor de wijze van versleuteling. De veiligste manier is de zogenaamde *end-to-end encryptie*. Hierbij wisselen zender en ontvanger (over een veilig kanaal) een sleutel uit waarmee ze hun berichtenverkeer vercijferen. Na afloop van de communicatiesessie kan deze sleutel worden afgedankt en heet daarom een *sessiesleutel*. Afluisteren kan alleen als het gebruikte cryptosysteem dan wel het veilig kanaal voor de sleuteluitwisseling is gebroken*.

De veilige manier van end-to-end encryptie is echter niet altijd mogelijk. Neem als voorbeeld een gsm-netwerk. Niet iedereen heeft het laatste model mobiele telefoon, waardoor sommigen over het nieuwste cryptosysteem beschikken maar anderen het nog moeten doen met een ander, ouder systeem. Er zijn nu twee mogelijkheden, òf elk nieuw model mobiele telefoon kan alle bestaande cryptosystemen aan, òf er is een vertaalslag nodig tussen het mobieltje met het nieuwe en dat met het oude systeem.

In het eerste geval bestaat het gevaar van cryptoanalyse door de tegenstander als het oudere en waar-

* Sterke end-to-end-encryptie vormt een hoofdpijn punt voor diensten als de Amerikaanse FBI en NSA. Zoals blijkt uit de in 2015 door hen publiekelijk geuite onvrede met het door Apple ingebouwde geheimschrift, dat ook deze fabrikant zelf zegt niet te kunnen breken. Daardoor staan deze diensten met lege handen. De discussie over het toelaten van sterke encryptie speelt tegelijkertijd ook in Nederland.

schijnlijk zwakkere cryptosysteem is gebroken. Jammer voor degene met de nieuwste mobiele telefoon, maar diens veiligheid is gekoppeld aan de zwakste schakel in de communicatie. En in het tweede geval is end-to-end-encryptie niet mogelijk. Want tenminste één van de knooppunten moet kunnen beschikken over de encryptiesleutel om de provider in staat te stellen van het ene naar het andere systeem te vercijferen. Men spreekt in dit geval van *link-encryptie*. De veiligheid van de verbinding ligt dan dus volledig in de handen van de provider die het netwerk beheert*.

1.4 Publieke sleutel

In het klassiek- ofwel symmetrisch cryptosysteem wordt het vercijferen en ontcijferen gedaan met dezelfde sleutel. Op een gegeven moment is iemand op het idee gekomen om een splitsing aan te brengen tussen de vercijfer- en de ontcijfersleutel. Bij het vercijferen wordt een andere sleutel gebruikt dan bij het ontcijferen. Een cryptosysteem dat volgens dit principe werkt noemt men een systeem met *publieke sleutel* of *public-key cryptosysteem*[†]. De geschiedenis van de ontwikkeling van deze systemen door pioniers als Whitfield Diffie, Martin Hellman, Adi Shamir, Ronald Rivest en anderen, kan worden nagelezen in het boek *Crypto, secrecy and privacy in the new code war*^[1]. Het boek beschrijft ook de pogingen die zijn ondernomen om de ontwikkeling van deze krachtige cryptografische systemen tegen te gaan en ze uit de openbaarheid te houden. Tevergeefs, zo is gebleken, en vandaag de dag is beveiliging van computers en dataverkeer niet goed meer denkbaar zonder deze systemen.

* Waarschijnlijk is in dit geval dat de provider een sleutel uitgeeft. Handig voor opsporingsinstanties die deze sleutel dan bij de provider kunnen komen opeisen. Of voor buitenlandse inlichtingendiensten, die inbreken op de systemen van de provider; zoals uiteraard is gebeurd.

† De naamgeving van dit soort systemen is nog wel eens verwarrend. Naast *publieke sleutel* wordt ook wel *openbare sleutel* gebruikt, of zelfs gewoon het Engelse *public-key*. In deze syllabus zullen de aanduidingen publieke sleutel en public-key doorelkaar gebruikt worden.

Er kunnen dan wel twee verschillende sleutels zijn, maar natuurlijk moet toch een verband tussen die twee bestaan. Anders is niet goed in te zien waarom de ontcijfering zou kunnen lukken. Als het nu eenvoudig was om uit de ene sleutel de andere te berekenen, dan zou zo'n systeem niet veel waard zijn. Het is echter mogelijk om de sleutels op zo'n manier te kiezen dat dit ondoenlijk is. Met ondoenlijk wordt bedoeld dat het weliswaar in principe mogelijk is om die berekening uit te voeren, maar dat dit teveel tijd gaat kosten om van nut te zijn. De wetenschap dat een cryptogram kan worden ontcijferd brengt weinig vreugde als daar tien miljard jaar op moet worden gewacht.

Doordat dus in de praktijk de ene sleutel niet uit de andere kan worden afgeleid, kan men gerust één van de twee sleutels bekend maken. Zo ontstaat het gebruik waaraan het public-key cryptosysteem z'n naam ontleent. Eén van beide sleutels wordt bekend gemaakt en heet daarom de *publieke sleutel* (public key), de andere blijft verborgen en heet de *geheime sleutel* (secret key). In [vergelijking \(1.2\)](#) wordt de publieke sleutel aangeduid met K_p en de geheime sleutel met K_s . Door de asymmetrie in de sleutels wordt het public-key cryptosysteem een *asymmetrisch cryptosysteem* genoemd. Dit in tegenstelling tot het klassiek symmetrisch cryptosysteem met eenzelfde sleutel voor vercijferen en ontcijferen.

$$C = E(M, K_p) \quad \text{en} \quad M = D(C, K_s) \quad (1.2)$$

De gevolgen van het kunnen opdelen van de sleutel in een *publieke sleutel* voor het vercijferen en een *geheime sleutel* voor het ontcijferen zijn enorm. Een communicatiesysteem gebaseerd op het public-key cryptosysteem werkt in principe als volgt. Alle deelnemers krijgen of maken een sleutelpaar (een bij elkaar behorende geheime en publieke sleutel) en maken hun publieke sleutel bekend; bijvoorbeeld door plaatsing op een fileserver in een computernetwerk. De geheime sleutel houdt iedereen natuurlijk angstvallig voor zich. Wanneer iemand een bericht naar een andere deelnemer wil sturen zoekt zij de publieke sleutel van de betrokkene op en gebruikt deze om het bericht te vercijferen. Onder de veronderstelling dat het systeem werkelijk veilig is, kan nu niemand anders dan de bezitter van de bijbehorende geheime sleutel het bericht ontcijferen.

Zodoende is het in een asymmetrisch cryptosysteem mogelijk om met slechts 10 sleutelparen een groep van 10 deelnemers onderling veilig met elkaar te kunnen laten communiceren. In een symmetrisch cryptosysteem zijn daarvoor 45 sleutels nodig. Het verschil loopt snel op als het aantal deelnemers toeneemt. Om een groep van N deelnemers onderling veilig met elkaar te kunnen laten communiceren heeft een symmetrisch cryptosysteem $N(N-1)/2$ sleutels nodig. Met 10.000 deelnemers zijn in een symmetrisch systeem al bijna 50 miljoen sleutelparen nodig. Het is schier onmogelijk om op die manier alle internetgebruikers veilig met elkaar te laten communiceren.

Met public-key-systemen kan echter nog meer. Want als de geheime en publieke sleutel in het vercijferproces van rol kunnen wisselen, dan kan een bericht ook met de geheime sleutel vercijferd worden. Het zo ontstane bericht kan door iedereen worden ontcijferd, want de publieke sleutel past daarop en is aan iedereen bekend. Maar omdat de publieke sleutel van de afzender de enige is die tot ontcijfering leidt, is daarmee tegelijk ook het bericht gewaarmerkt als afkomstig van de bezitter van de bijbehorende geheime sleutel. Wat meer is, die kan nu ook niet meer ontkennen het bericht te hebben verstuurd. Het public-key-principe kan behalve voor geheimhouding daarom ook voor het *authenticiseren* (*digitale handtekening*) en *nonrepudiation* (afkomst niet kunnen ontkennen) worden benut.

1.5 Quantum cryptografie

Het ideale cryptosysteem is een onbreekbaar cryptosysteem dat niet afgeluisterd kan worden. Met de komst van quantumcryptografie wordt dat ideaal dicht benaderd. Quantumcryptografie kan goed worden gebruikt voor het veilig overbrengen van een cryptografische sleutel. Het bijzondere aan dit systeem is dat afluisteren niet onopgemerkt blijft. Een andere toepassing waarnaar bijvoorbeeld gezocht wordt, is om een bericht zodanig te kunnen vercijferen dat alleen iemand op een specifieke locatie het kan ontcijferen^[2].

De basis voor het systeem berust op de natuurkunde van elementaire deeltjes, de quantummechanica en wel speciaal die van fotonen (lichtdeeltjes). In de quantummechanica heeft een deeltje altijd twee aspecten:

een deeltjeskarakter en een golfkarakter. De massa en snelheid van een deeltje vertegenwoordigen een bepaalde hoeveelheid energie en het golfkarakter geeft een deeltje een zekere uitgebreidheid. Fotonen manifesteren zich aan ons vooral als lichtgolven, maar hun energie komt overeen met een – weliswaar zeer kleine – hoeveelheid massa. Bepaling van de impuls* van een foton levert informatie over de energie die het bevat, bepaling van de plaats beperkt de onzekerheid omtrent de met het golfkarakter samenhangende uitgebreidheid.

Het woord onzekerheid is gevallen. Onzekerheid is een kernbegrip in de quantummechanica. Metingen leveren nooit volledige zekerheid omtrent alle aspecten van een quantumdeeltje, maar veeleer een waarschijnlijkheid. En daarmee komen we bij het basisprincipe waarop quantumcryptografie berust, namelijk de *onzekerheidsrelatie van Heisenberg*^[3]. Deze zegt dat van een deeltje de plaats en de impuls niet tegelijkertijd met willekeurige precisie kunnen worden bepaald.

$$\Delta x \cdot \Delta p \gtrsim \hbar \quad (1.3)$$

De betekenis is als volgt: onzekerheid in positie en impuls kunnen niet beide *tegelijkertijd* willekeurig klein gemaakt worden. Het product van de spreiding en impuls kan niet kleiner zijn dan een van de constante van Planck afgeleide waarde.

Het gevolg van deze fundamentele onzekerheid is uiterst belangrijk. Het betekent namelijk dat meten aan quantumdeeltjes nooit kan geschieden zonder ze te beïnvloeden. Hoe nauwkeuriger de plaats van een deeltje gemeten wordt des te groter wordt de onzekerheid over de impuls. Omgekeerd, hoe preciezer de bepaling van de impuls des te groter de onzekerheid omtrent de plaats.

In de quantumcryptografie wordt deze onzekerheidseigenschap gebruikt om het *onopgemerkt* afluisteren onmogelijk te maken. Hiertoe worden fotonen – enerzijds lichtdeeltjes anderzijds lichtgolven – gepolariseerd. Licht is een zogenaamde transversale golf, d.w.z. het trilt in een vlak loodrecht op de voortplantingsrichting[†]. Als regel trilt het licht in alle vlakken die men zich kan denken. Door polarisatie kan die

* De impuls p is het product van massa en snelheid.

trilling worden beperkt tot één bepaald vlak. Het vlak van de polarisatie wordt aangeduid met een hoek: het aantal graden verdraaiing t.o.v. een vast referentievlak. De polarisatie kan waarden van 0 tot 180 graden aannemen.

Met een polarisatiefilter kan de hoek tussen het vlak van de polarisatie en het referentievlak worden gemeten^{*}. Het filter laat alleen golven door die trillen in hetzelfde vlak als waarop het filter is ingesteld. Stel het filter staat ingesteld op doorlaten van polarisatie 0-graad. Dan zal gepolariseerd licht met een polarisatie van 0-graad worden doorgelaten en licht met een polarisatie van 90-graden geheel worden tegengehouden. Licht dat trilt in een vlak tussen deze beide uitersten in zal niet volledig worden tegengehouden. Maar het wordt meer verzwakt naarmate de polarisatiehoek verder afwijkt van die van het filter. Men kan zich daartoe het trillend licht gesplitst denken in twee componenten: de ene component trilt in het vlak van het filter en de andere trilt in het vlak loodrecht daarop. De eerste component wordt doorgelaten, de tweede tegengehouden.

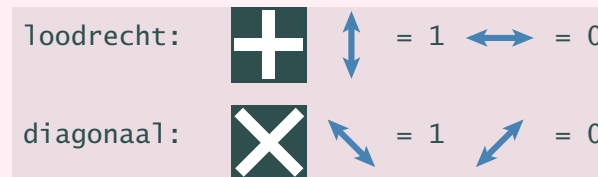
1.5.1 Sleuteluitwisseling

In quantumcryptografie wordt gekozen voor vier polarisatierichtingen met respectievelijk een polarisatiehoek van 0, 45, 90 en 135 graden. Aan deze polarisatierichtingen worden de bitwaarden 0 en 1 gekoppeld, bijvoorbeeld zoals in [figuur 1.3](#). Twee sets polarisatierichtingen – loodrecht en diagonaal –

[†] Een vergelijking met muziekinstrumenten kan het begrip *transversale* trilling verhelderen. De snaar van een viool komt door aanstrijken in een transversale trilling: de snaar wijkt uit in een richting loodrecht op de lengte van de snaar. De luchtkolom in een orgelpijp komt daarentegen in een *longitudinale* trilling: in de pijp treden verdunningen en verdichtingen op, de luchtdeeltjes bewegen zich in de lengterichting van de pijp. Vergelijk de longitudinale trilling met de verdichtingen en verdunningen in het verkeer op een snelweg, te zien wanneer men langs een file in de andere richting rijdt.

^{*} Licht dat door de atmosfeer verstrooid wordt is in zekere mate gepolariseerd. Een Polaroid-bril bevat een filter dat de intensiteit van licht trillend in een bepaald polarisatievlak vermindert. Daardoor neemt de schittering af en verbetert het zicht.

worden gebruikt, omdat daarmee nu juist kan worden ingespeeld op de quantumeigenschappen. Essentieel is dat slechts één enkel foton wordt verzonden.



Figuur 1.3 Definitie van 0 en 1 met polarisatierichtingen

Om een bit te versturen kiest de verzender een van de twee sets, bijvoorbeeld de loodrechte set. Voor een 1-bit wordt dan een foton gepolariseerd met hoek 0 en voor een 0-bit wordt het gepolariseerd met hoek 90. Als het inkomend foton door de ontvanger wordt gemeten met diezelfde set en de polarisatierichting van het filter staat op 0, dan manifesteert een 1-bit zich als licht en een 0-bit als duisternis en omgekeerd. Voor een volgend bit kiest de verzender opnieuw een van de twee sets uit.

Wat gebeurt er als de ontvanger de verkeerde meetset instelt? Dan komt bijvoorbeeld een loodrecht gepolariseerd foton op een diagonale detector. Zoals gezegd wordt gepolariseerd licht dan ontbonden in twee onderling loodrechte richtingen: een component parallel aan de polarisatiehoek van het filter en de ander loodrecht daarop. Maar omdat het onmogelijk is een foton te splitsen – één foton blijft één foton – kiest het foton willekeurig een van de twee mogelijkheden. Het heeft evenveel kans om een 0-bit als om een 1-bit te worden. Kan de ontvanger nu het foton nogmaals meten, maar dan met de goede set? Nee! Meten aan een quantumdeeltje beïnvloedt de eigenschappen van het deeltje onherroepelijk en de oorspronkelijke waarde is voorgoed verloren gegaan.

Allereerst wordt nu behandeld hoe een rij bits veilig kan worden verstuurd zonder dat zender en ontvanger tevoren over een veilig kanaal gegevens hebben behoeven uit te wisselen. Deze bits kunnen vervolgens gebruikt worden als sleutel voor het verscijferen van een bericht. Daarna wordt gekeken naar het effect

van af luisteren op de informatieoverdracht en zal blijken dat daarmee niet alle bits goed kunnen worden bepaald.

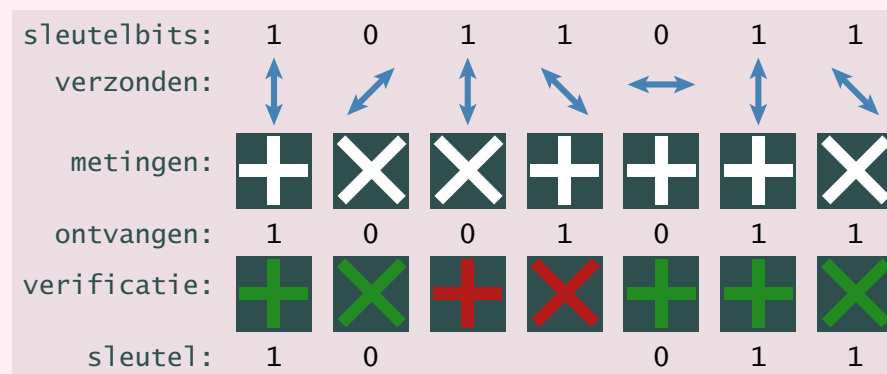
Om de bits te versturen kiest de verzender voor elk afzonderlijk bit willekeurig een van de twee polarisatiesets. Met die set wordt voor dat bit één enkel foton gepolariseerd en naar de ontvanger gestuurd. De ontvanger weet van tevoren niet welke set de verzender heeft gebruikt en meet het inkomend foton daarom met een van de twee sets die zij bij gebrek aan beter willekeurig kiest. Het verzonden bit komt zeker goed over als de ontvanger de set goed heeft gekozen. Heeft de ontvanger daarentegen verkeerd gekozen dan kan het ontvangen bit goed, maar ook verkeerd gelezen zijn.

Nadat de gehele bitrij is verstuurd, wisselen zender en ontvanger uit welke sets zij gekozen hebben en weten daardoor allebei welke bits goed zijn aangekomen. Zender en ontvanger kunnen zodoende de metingen met de verkeerd gekozen meetsets wegstrepen en houden dan de bruikbare bits over. In [figuur 1.4](#) is deze procedure weergegeven. Omdat de ontvanger een kans van 50% heeft om de goede meetset te kiezen, zullen gemiddeld twee bits moeten worden overgestuurd om er één goed te kunnen ontvangen. De veiligheid berust op het feit dat de metingen aan de fotonen plaatsvonden vóór de uitwisseling van de gekozen meetsets en dat deze metingen niet kunnen worden overgedaan.

[Tabel 1.1](#) resumeert wat met elk verzonden bit in [figuur 1.4](#) is gebeurd. De kolom uitkomst vermeldt resultaat goed wanneer het bit is aangekomen zoals het werd verzonden, ongeacht of het juiste masker wel werd gekozen. Sommige fotonen zijn gemeten zoals ze zijn verzonden, die metingen mogen worden geaccepteerd. Alle bits die met het verkeerde masker werden gemeten moeten worden verworpen.

1.5.2 Afluisterdetectie

Stel nu dat de verzonden fotonen worden onderschept door iemand die aan het af luisteren is en die ze vervolgens doorstuurt naar de ontvanger. Ook wie af luistert heeft ermee te maken dat de gekozen polarisatieset onbekend is en moet eveneens een willekeurige meetset kiezen. Gemiddeld in de helft van



Figuur 1.4 Overdracht van quantum bits

	zend	meet	uitkomst	actie
bit 1 :	⊥	⊥	goed	aannemen
bit 2 :	×	×	goed	aannemen
bit 3 :	⊥	×	fout	verwerpen
bit 4 :	×	⊥	goed	verwerpen
bit 5 :	⊥	⊥	goed	aannemen
bit 6 :	⊥	⊥	goed	aannemen
bit 7 :	×	×	goed	aannemen

Tabel 1.1 Overdracht van quantum bits in [figuur 1.4](#)

de gevallen zal de onderschepper goed kiezen en het correcte bit doorsturen. Bij een verkeerde keuze is de kans 50% dat het verkeerde bit wordt verzonden, waardoor dus gerekend naar de inkomende bits 25% kans bestaat op een foutief doorgezonden bit. Dit zou anders zijn wanneer de onderschepper het

oorspronkelijk ontvangen foton had kunnen doorsturen. Maar dat is nu juist onmogelijk met quantum-deeltjes en tevens de reden waarom slechts één enkel foton de informatie mag dragen. De meting aan het foton heeft de oorspronkelijke toestand onherroepelijk verloren doen gaan en die toestand kan niet worden teruggehaald.

sleutelbits:	1	0	1	1	0	1	1
verzonden:	↕	↗	↕	↗	↔	↕	↗
onderschept:	+	+	×	+	×	+	+
ontvangen:	1	0	0	1	1	1	0
metingen:	+	×	×	+	+	+	×
ontvangen:	1	0	0	1	0	1	0
verificatie:	+	×	+	×	+	+	×
sleutel:	1	0			0	1	0

Figuur 1.5 Onderschepping van quantum bits

In [figuur 1.5](#) is de situatie geschetst. Voor verzender en ontvanger gaat het om de bits die overblijven nadat de foutieve metingen zijn weggestreept. Degene die af luistert en de legitieme ontvanger hebben allebei *onafhankelijk* van elkaar een keuze gemaakt. Ze hebben beide met kans 50% voor de goede of de foutieve meetset gekozen. Doordat ze onafhankelijk van elkaar hebben gekozen zal gemiddeld slechts de helft van die keuzes overeenkomen.

Waar komt dit nu op neer? Voor ongeveer de helft van de verzonden bits zal de meetset tussen verzender en ontvanger overeenkomen; alleen die bits tellen mee. Van die helft is gemiddeld de helft – een kwart

van de verzonden bits – getroffen door een mismatch tussen de meetsets van de onderschepper en de ontvanger. En van dat kwart zal gemiddeld weer de helft een verkeerd foton hebben opgeleverd. Met als resultaat dat gemiddeld één op de acht verzonden fotonen bij de ontvanger een verkeerde waarde heeft gekregen. Als de zender nu de verstuurde bits beschermt met een zogenaamde checksum, die fouten in de ontvangen bitreeks aan het licht kan brengen, dan is daarmee een controle op de juiste ontvangst mogelijk. Klopt die checksum niet, dan weet de ontvanger dat er iets niet in de haak is.

Om niet door de mand te vallen had de onderschepper nu juist de meetsets met die één op acht foutieve bits wél goed moeten kiezen. Bij het in totaal versturen van 400 bits zijn dat er zo'n 50 en heeft de afliesteraar daarom maar een kans van 2^{-50} of ten naaste bij 1 op 1.000.000.000.000.000 om alle geaccepteerde bits goed te laten overkomen. Het is dus nagenoeg onmogelijk om onopgemerkt af te luisteren.

Tabel 1.2 resumeert wat met elk verzonden bit is gebeurd. De waarde goed of fout slaat op het gekozen masker uit de vorige stap. Voor de onderschepper is dat het masker gekozen door de oorspronkelijke afzender (kolom uitkomst-1) en voor de ontvanger het door de onderschepper gekozen masker (kolom uitkomst-2). De laatste uitkomst-kolom geeft de overeenstemming tussen het oorspronkelijk verzonden bit en dat bij de uiteindelijke ontvanger. De fotonen die door de ontvanger zijn gemeten zoals ze werden verzonden zullen worden geaccepteerd, ongeacht wat er tussendoor mee is gebeurd. Merk op dat het laatst verzonden bit weliswaar fout bij de ontvanger aankomt, maar toch door deze wordt geaccepteerd. Dit bit zal derhalve bijdragen tot het demasqué van de onderschepper.

1.6 Veiligheid en bruikbaarheid

De keuze van een cryptosysteem moet passen bij het gebruik waarvoor het is bedoeld. Enerzijds moet worden voorkomen dat het systeem te zwak is en door de eerste de beste tegenstander kan worden gebroken. Aan de andere kant is het ook weer niet nodig om met een kanon op een mug te schieten. Het systeem hoeft niet sterker en ingewikkelder te zijn dan de omstandigheden vereisen. Immers, hoe in-

	zend	meet-1	uitkomst-1	meet-2	uitkomst-2	uitkomst	actie
bit 1 :	⊥	⊥	goed	⊥	goed	goed	aannemen
bit 2 :	×	⊥	goed	×	goed	goed	aannemen
bit 3 :	⊥	×	fout	×	goed	fout	verwerpen
bit 4 :	×	⊥	goed	⊥	goed	goed	verwerpen
bit 5 :	⊥	×	fout	⊥	fout	goed	aannemen
bit 6 :	⊥	⊥	goed	⊥	goed	goed	aannemen
bit 7 :	×	⊥	fout	×	goed	fout	aannemen

Tabel 1.2 Onderschepping van quantum bits in *figuur 1.5*

gewikkelder het systeem is, des te meer kans dat er fouten mee gemaakt worden. En hoe meer fouten hoe meer kans op een succesvolle cryptanalyse. Het kan zelfs gevaarlijk zijn om een sterk systeem te gebruiken in omstandigheden die dat niet nodig maken. Als het systeem door dát gebruik gecompromitteerd raakt, is het ook niet meer bruikbaar in situaties waar een sterk systeem juist wel vereist is.

1.6.1 Voorwaarden

De eerste die zich systematisch heeft bezig gehouden met de vraag aan welke voorwaarden een veilig cryptosysteem voor communicatie moet voldoen, was onze in Frankrijk wonende landgenoot Auguste Kerckhoffs. De meeste van de door hem geformuleerde criteria zijn – met enige aanpassing aan de tegenwoordige tijd – nog steeds van belang. Kerckhoffs formuleerde deze als zijn *Desiderata de la cryptographie militaire*^[4].



Kerckhoffs



Kerckhoffs

1. *Le système doit être matériellement, sinon mathématiquement, indéchiffrable.*

Deze eerste regel zegt dat een cryptosysteem niet persé theoretisch onontcijferbaar behoeft te zijn om toch te kunnen voldoen. Het is voldoende als het in de praktijk maar een zwaar genoeg obstakel vormt. Wanneer de inspanning nodig voor en het tijdsbeslag gemoeid met het breken van het cryptogram groter zijn dan waarde of levensduur van het bericht zelf, dan is materieel gezien het systeem veilig genoeg. Deze regel heeft algemene geldigheid en vormt een goed criterium voor de keuze van een systeem bedoeld voor een specifieke situatie.

2. *Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi.*

De tweede regel is minstens zo belangrijk als de eerste. Deze stelt dat de veiligheid niet aangetast behoort te worden wanneer systeem en apparatuur in handen van de tegenstander vallen. De veiligheid van het systeem berust dan in laatste instantie op het geheimhouden van de sleutel van het cryptogram. Ook deze regel is algemeen geldig en het belang ervan kan niet genoeg worden onderstreept. Natuurlijk heeft het wel zin om de details van een cryptosysteem voor een tegenstander verborgen te houden, want dat vormt een extra belemmering. Men moet er echter altijd rekening mee houden dat de tegenstander op den duur wel achter de werking van het systeem zal komen.

3. *La clef doit pouvoir en être communiquée et retenue sans le secours de notes écrites, et être changée ou modifiée au gré des correspondants.*

De veiligheid is ermee gediend wanneer de bericht sleutel niet behoeft te worden opgeschreven, maar gemakkelijk kan worden onthouden. Bij computers vormt het onthouden geen probleem, zelfs niet voor de meest ingewikkelde sleutel. Aan de andere kant brengt het onthouden van sleutels in computers wel weer de noodzaak met zich mee om ze bijzonder goed te beschermen. Ze zijn immers in zekere zin nu juist wel neergeschreven en mogen dus op geen enkele manier voor onbevoegden toegankelijk zijn. Verder is het van belang dat de sleutel niet onveranderlijk is, maar naar behoefte eenvoudig kan

worden gewisseld. In computersystemen is dat vaak lastig, vooral als het gecijferde materiaal (ook) op allerlei backsystemen ligt opgeslagen.

4. *Il faut qu'il soit applicable à la correspondance télégraphique.*

De vierde regel zegt dat het systeem inpasbaar moet zijn in de gangbare vorm van telecommunicatie. In de huidige tijd komt dat praktisch gesproken neer op het kunnen verwerken van alle 256 bitcombinaties van een byte en het probleemloos kunnen omgaan met unicode en andere gangbare tekencoderingen.

5. *Il faut qu'il soit portatif, et que son maniement ou son fonctionnement n'exige pas le concours de plusieurs personnes.*

De vijfde regel is zeer direct gericht op systemen die 'in het veld' gebruikt moeten worden. Het systeem moet gemakkelijk meegenomen kunnen worden en door slechts een of enkele personen bediend kunnen worden. Dat geldt natuurlijk in het bijzonder voor spionnen, die niet betrappt mogen worden met opvallende apparatuur en die vaak geheel alleen moeten opereren.

6. *Enfin, il est nécessaire, vu les circonstances qui en commandent l'application, que le système soit d'un usage facile, ne demandant ni tension d'esprit, ni la connaissance d'une longue série de règles à observer.*

De procedure moet eenvoudig uitvoerbaar zijn. Ingewikkelde handelingsvoorschriften zijn uit den boze, omdat die gemakkelijk aanleiding geven tot fatale vergissingen, vooral onder stress. Ook deze regel is nog steeds van het grootste belang. Juist omdat beveiliging extra handelingen en alertheid vergt zijn mensen geneigd ze te vergeten of uit gemakzucht te omzeilen. Hoe eenvoudiger het uitvoeren van de beveiligingsvoorschriften is, hoe meer kans dat er ook de hand aan zal worden gehouden*.

* En natuurlijk doen de ontwerpers van veiligheidskritische computerprogrammatuur er verstandig aan om de beveiligingshandelingen dwingend in te programmeren, zodat omzeilen onmogelijk wordt gemaakt.

1.6.2 Veilig

In het voorgaande hebben de criteria van Auguste Kerckhoffs al iets aangegeven van de omstandigheden waaronder veilige cryptografie moet opereren. Hier zal worden gezien in hoeverre cryptoanalyse een gevaar vormt.

Ontwerpers van een geheimschrift zullen zich tegen cryptoanalyse willen verdedigen door het geheimschrift zo moeilijk mogelijk te maken. De vraag rijst wat in dit verband als *moeilijk* moet worden beschouwd. Dit probleem zal pragmatisch worden benaderd. Allereerst wordt als uitgangspunt genomen dat een geheimschrift wel voor langere tijd in gebruik zal blijven. Het is dan niet realistisch om te verwachten dat de werking ervan een tegenstander lang onbekend zal blijven. Op de een of andere manier – hetzij door cryptoanalyse, hetzij als gevolg van verraad, hetzij door een stommitieit – lekt het geheimschrift altijd wel uit. De veronderstelling dat bij de tegenstander het systeem van vercijferen bekend kan zijn, legt in overeenstemming met de tweede voorwaarde van Kerckhoffs de veiligheid geheel en al bij de sleutel. Het is dus zaak een geheimschrift zo te maken, dat die sleutel niet eenvoudig of goedkoop kan worden bepaald.

Uitgaande van het adagium dat elk geheimschrift in beginsel kan worden gebroken, is de cruciale vraag daarom hoeveel tijd en inspanning daarmee gemoeid zal zijn. In het algemeen moet men voor het breken van een geheimschrift veel kosten maken, want een team van bekwame cryptoanalisten op de been houden is beslist niet goedkoop. De motivatie moet daarom sterk genoeg zijn om de inspanning en investering te rechtvaardigen. Vertaald in economische termen wil dit zeggen, dat de verwachte opbrengst van de cryptoanalyse groter moet zijn dan de op te brengen kosten*. Dit is het eerste criterium waaraan de sterkte van een geheimschrift kan worden afgemeten: *hoeveel middelen en inspanningen zijn nodig om het te breken*.

* Amateurs en wetenschappers kunnen deze verwachting teniet doen omdat zij niet rekenen met een kosten/baten-analyse, maar worden gedreven door nieuwsgierigheid en ambitie.

Het tweede criterium heeft betrekking op het nuttig gebruik dat men van de ontcijferde berichten kan maken. Veel berichten zijn slechts kortstondig van belang. Wanneer het leger van de vijand de aanval eenmaal heeft ingezet, is cryptoanalyse van het aanvalsbevel niet meer zo relevant. Maar cryptoanalyse kost tijd. Als de tijd nodig om de cryptoanalyse uit te voeren groter is dan de levensduur van het geheim in het cryptogram, dan is dat geheim voor alle praktische doeleinden voldoende veilig. Het tweede criterium waaraan de sterkte van een geheimschrift kan worden afgemeten luidt daarom: *hoeveel tijd is nodig om het te breken*. Dit is een criterium dat ook van belang is voor de bescherming van gedigitaliseerde informatie die voor langere tijd geheim moeten blijven.

Welke methoden staan een cryptoanalist bij zijn analyse ter beschikking? Zijn er omstandigheden die cryptoanalyse gemakkelijker maken? De eigenschappen van een geheimschrift bieden vaak de mogelijkheid speciale methoden toe te passen om het te breken. Het is een misvatting – al door veel uitvinders van geheimschriften gemaakt – om te denken dat een ingewikkeld traject van vercijfering ook noodzakelijk door de cryptoanalist in omgekeerde richting moet worden afgelegd bij het breken ervan. Vergelijk dit met het zoeken naar de uitgang van een doolhof. De cryptoanalist is niet van plan om netjes op de paden te blijven, maar loopt desnoods dwars door de heggen heen.

Wanneer het cryptosysteem bekend is, is het in principe altijd mogelijk om systematisch alle sleutels te proberen: *uitputtend zoeken* (brute-force in het Engels). Hoe kan de cryptoanalist dan weten of de goede oplossing is gevonden? En niet iets wat toevallig best wel de oplossing had kunnen zijn, maar geen enkele relatie met de werkelijke oplossing heeft? Een antwoord daarop kan worden gegeven door de *informatietheorie*. Die theorie vertelt hoe groot het gebruikte cryptogramfragment ongeveer moet zijn om met voldoende zekerheid het vinden van de goede oplossing te kunnen vaststellen. Hoewel uitputtend zoeken altijd kan worden toegepast, bestaat er ook een betrekkelijk eenvoudig verweer tegen: *zorg dat er zoveel sleutels zijn, dat het zoeken te lang gaat duren*. Hoeveel sleutels genoeg is, hangt samen met de stand van de techniek. Men zal nu begrijpen waarom cryptoanalisten zich bij elke nieuwe generatie van nog snellere computers verlekkerd in de handen hebben gewreven. Het zal dus ook geen verbazing

wekken dat de Amerikaanse National Security Agency (NSA) actief participeerde in de ontwikkeling van de computer^[5].

De positie van een cryptoanalist kan als volgt worden gekarakteriseerd, gerangschikt van moeilijk naar steeds gunstiger omstandigheden voor de cryptanalyse.

- *ciphertext-only aanval*

De cryptoanalist beschikt slechts over cijfertekst en in het meest ongunstige geval is zelfs het systeem onbekend. Statistische methoden, aanvullende kennis, fouten bij het vercijferen, geluk, het zijn allemaal factoren die het analyseproces begunstigen. Hoewel moeilijk, is succes niet uitgesloten. Een voorbeeld daarvan is de analyse door een Duits team onder leiding van Hans Rohrbach, dat in de Tweede Wereldoorlog een Amerikaans diplomatiek systeem kraakte uitsluitend op basis van onderschepte codetelegrammen^[6].

- *known-plaintext aanval*

In dit geval beschikt de cryptoanalist over de klaartekst of een deel daarvan en kan deze leggen naast de overeenkomstige cijfertekst. Door vergelijking van beide kan de sleutel misschien worden gevonden. Zo'n klaartekst kan ter beschikking komen wanneer de tegenstander een bericht in verschillende cryptosystemen vercijfert en één daarvan reeds gebroken is. Een bericht dat op verschillende manieren is vercijferd noemt men een *isolog*. In de Tweede Wereldoorlog maakten de Engelsen op een slimme manier van deze techniek gebruik. Ze lieten mijnen uitstrooien in zee, "gardening" werd dit genoemd. Vervolgens luisterden ze naar Duitse berichten waarin voor deze mijnenvelden werd gewaarschuwd. Zulke berichten gingen naar verschillende eenheden, werkend met verschillende cryptosystemen. Eén van deze systemen (de zogenaamde Werftschlüssel (dockyard cipher in het Engels) kon vrij gemakkelijk gebroken worden. De ontcijferde cryptogrammen leverden dan brokjes klaartekst die in andere onderschepte cryptogrammen geprobeerd konden worden^[7].

- *chosen-plaintext aanval*

In deze situatie kan de cryptoanalist naar believen klaartekst omzetten in cijfertekst en dus experimenteren met het geheimschrift. Voor een succesvolle cryptoanalyse duidelijk de meest gunstige uitgangspositie. Het is niet persé noodzakelijk dat de cryptoanalist zelf de gekozen klaartekst in geheimschrift omzet. Soms kan men dit aan de tegenstander zelf overlaten. Een mooi voorbeeld hiervan is de manier waarop de Amerikanen in het begin van de Tweede Wereldoorlog het Japanse codewoord voor het eiland Midway in de Stille Oceaan te weten kwamen. Ze brachten het gerucht in omloop dat de waterzuiveringsinstallatie van Midway defect zou zijn en keken vervolgens hoe dat aan Tokyo werd gemeld^[8].

Om de sterkte van geheimschriften te karakteriseren kunnen bovenstaande aanvalstypen worden gebruikt. Een geheimschrift dat zelfs tegen een chosen-plaintext aanval bestand is, geldt als het sterkste van alle. Als een geheimschrift slechts bestand is tegen een ciphertext-only aanval dan is het zwakker dan een systeem dat ook een known-plaintext aanval kan weerstaan.

1.6.3 Methoden van cryptoanalyse

Nu volgt een illustratie van de stelling dat cryptoanalyse andere wegen kan inslaan dan zijn gevolgd bij het maken van het cryptogram. Aangetoond wordt dat er meerdere manieren zijn om een cryptogram te breken. Dat geeft een indruk van de soms zeer betrekkelijke veiligheid die een cryptosysteem biedt. **Hoofdstuk 6** is geheel gewijd aan de behandeling van enkele methoden waarmee cryptogrammen kunnen worden gebroken.

Het cryptosysteem hier als voorbeeld gebruikt is een van de eenvoudigst denkbare systemen. Het is de *monoalfabetische substitutie* en ontleent die naam aan het feit dat elk van de 26 letters van het alfabet systematisch wordt vervangen door een andere letter. De sleutel in dit systeem bestaat uit de correspondentie tussen het gewone alfabet en de voor vervanging gebruikte permutatie van het alfabet. Onder een

permutatie van het alfabet wordt verstaan een alfabet waarin de letters in een andere volgorde staan dan het gebruikelijke ABC.

$$K : A \cdots Z \leftrightarrow \Pi(A \cdots Z) \quad (1.4)$$

Uitputtend zoeken

Zoals in het vorenstaande al is opgemerkt, kan een cryptoanalist altijd een poging tot ontcijferen wagen door het proberen van alle sleutels. In dit geval is de sleutel een permutatie van het alfabet. Met 26 letters is het aantal mogelijke permutaties, en dus het aantal mogelijke sleutels, gelijk aan $26! \approx 4 \times 10^{26}$.

Veronderstel er is een computer beschikbaar, die per seconde 1000 miljoen sleutels kan testen. Het proberen van alle sleutels kost dan $4 \times 10^{26} / 10^9 \approx 4 \times 10^{17}$ seconden. Gegeven het feit dat er $60 \times 60 \times 24 \times 365 = 31.536.000$ secondes in een jaar gaan, komt dit overeen met een zoektijd van ongeveer 12×10^9 jaar. Het proberen van alle sleutels met deze computer kost meer dan 12.000 miljoen jaar!

Hieruit blijkt overduidelijk dat hoewel uitputtend zoeken steeds tot de mogelijkheden behoort, dit niet altijd praktisch uitvoerbaar is. Zelfs voor het buitengewoon eenvoudige systeem van deze monoalfabetische substitutie is het niet echt een optie. Toch zijn er ook systemen waartegen uitputtend zoeken wel met succes kan worden toegepast. Dat geldt bijvoorbeeld voor het in [hoofdstuk 8](#) te behandelen DES-cryptosysteem. Dit systeem telt ruwweg 7×10^{16} sleutels, die anno 1998 met een speciaal gebouwde zoekmachine in zo'n 9 dagen konden worden getest. Veel en veel minder sleutels dus dan die 4×10^{26} van de monoalfabetische substitutie. Desondanks is DES een veel sterker cryptosysteem dan de monoalfabetische substitutie. Het laat zien dat veel sleutels wel belangrijk is, maar niet het enige dat telt.

Opdeling zoekruimte

Het gaat er voor de cryptoanalist een stuk beter uitzien als het aantal mogelijkheden systematisch kan

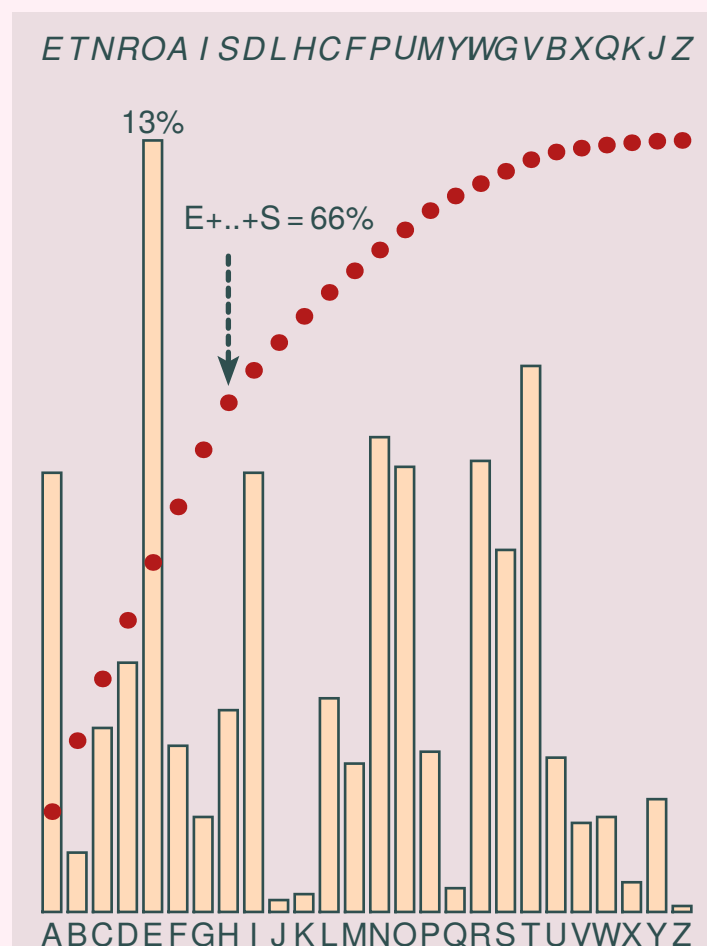
worden beperkt. Zijn er experimenten waarmee bepaalde, liefst grote groepen sleutels kunnen worden uitgesloten? In het geval van de monoalfabetische substitutie kan dit door gebruik te maken van de statistische eigenschappen van natuurlijke taal. In [figuur 1.6](#) staat een frequentiedistributie van de letters in het Engels. Direct valt op dat sommige letters veel en andere letters juist heel weinig voorkomen. De letter E komt met ongeveer 13% het meest voor, letters als J en Z vindt men weinig.

Deze nogal ongelijkmatige letterdistributie kan benut worden om de letters naar gelang hun voorkomen te verdelen in groepen. Veronderstel eens dat op grond van een telling van de letters in het cryptogram tot de volgende clustering van kandidaten kan worden gekomen: ET, AINORS, CDFHLMPUY en BGJKQVWXYZ. Nu behoeven niet alle mogelijke permutaties van het sleutelalfabet geprobeerd te worden, maar slechts die in elk van de afzonderlijke groepen. Het aantal te onderzoeken mogelijkheden daalt hierdoor met een factor 2.000.000.000.000. Het aantal mogelijkheden kan hier dus gigantisch worden verkleind eenvoudig door de letters te tellen.

Nog een saillant punt in de figuur is de cumulatieve frequentie. De stippen in de figuur geven het cumulatieve voorkomen van de letters, waarbij deze zijn gerangschikt naar dalende frequentie (de volgorde zoals weergegeven bovenaan de figuur). Met de acht meest voorkomende letters ETNROAIS heeft men al tweederde van de totale tekstmassa te pakken. Het zal zeker niet nodig zijn om alle letters te identificeren voordat de betekenis van het te ontcijferen cryptogram of delen daarvan zichtbaar wordt. Met een gering aantal gevonden letterssubstituties wordt al snel zoveel van het bericht leesbaar, dat het completeren een peuleschil wordt.

Er zijn nog wel meer eigenaardigheden die het oplossen vereenvoudigen. Uit de frequentiedistributie van [figuur 1.6](#) kan worden afgeleid dat ongeveer 40% van de tekst uit klinkers bestaat*. Deze zijn echter niet willekeurig verdeeld maar staan in afwisseling met medeklinkers. Dat kan worden gebruikt voor het onderscheid tussen klinkers en medeklinkers. Na een paar letters met lage frequentie, die hoogst waarschijnlijk

* De spaties in de tekst worden buiten beschouwing gelaten.



Figuur 1.6 Frequentie en cumulatieve frequentie letters Engels

alle medeklinkers zullen blijken te zijn, mag een klinker verwacht worden.

Het tellen van letters en het analyseren van hun gedrag kan worden uitgebreid naar groepen van twee letters (*digrammen*), drie letters (*trigrammen*) of meer, en zo helpen het aantal te onderzoeken mogelijk-

heden verder te verminderen.

Waarschijnlijke woorden

Wie een woordenboek opslaat denkt misschien dat er wel heel veel woorden zijn. Maar wanneer al die woorden worden vergeleken met alle mogelijke lettercombinaties van eenzelfde lengte, zal blijken dat slechts een kleine fractie van die lettercombinaties ook daadwerkelijk in het woordenboek voorkomt. Het kan daarom zinvol zijn om een woordenlijst te matchen met het cryptogram en te kijken welke combinaties goed passen.

Nog eenvoudiger wordt het als zoveel over de inhoud van het cryptogram bekend is, dat men kan raden naar bepaalde woorden. In een militair bericht zouden dat woorden kunnen zijn als: hoofdkwartier, compagnie, aanval, tankdivisie, namen van bekende commandanten, enz. Gezocht wordt dan naar plaatsen in het cryptogram waarop een dergelijk *waarschijnlijk woord* past. Als dat klopt kan men wellicht een stukje van de sleutel reconstrueren, waarna door toepassen hiervan op de rest van het cryptogram weer andere passages leesbaar worden, enz. In de Engelstalige literatuur wordt zo'n waarschijnlijk woord een *crib* genoemd.

Letterpatronen

In sommige woorden treft men specifieke letterpatronen aan, veroorzaakt door letterherhalingen of juist het ontbreken daarvan. In de monoalfabetische substitutie wordt elke letter systematisch vervangen door een andere en blijft het letterpatroon dus intact. Het betekent dat de cryptogramtekst GZGGTJ zou kunnen betekenen "kikker", "kokkel" of "pappot", maar zeker niet "makker" of "pispot". Op sommige patronen passen zo weinig werkelijk voorkomende lettercombinaties, dat de goede combinatie gemakkelijk kan worden bepaald. Zulke patronen zijn vaak ook een goed hulpmiddel bij het plaatsen van een waarschijnlijk woord.

Systeemspecifieke eigenschappen

Sommige systemen hebben specifieke eigenschappen die voor de cryptoanalyse kunnen worden benut. Een voorbeeld is de Duitse Enigma codeermachine uit de Tweede Wereldoorlog. Het vercijfermechanisme is zo ingericht dat een letter nooit zichzelf als vercijferd equivalent kan hebben. Voor een letter A in het bericht kan dus nooit op de overeenkomstige positie in het cryptogram een A staan. Op basis hiervan kan de cryptoanalyse worden gestuurd en kunnen testsleutels worden aanvaard of verworpen.

Tegenmaatregelen

Een goed cryptosysteem verhindert het gebruik van de frequentiedistributie en maakt opdelen van de zoekruimte voor de sleutel onmogelijk. Door de vercijfering moeten de statistische kenmerken van de klaartekst zo gelijkmatig mogelijk over de cryptogramtekst worden uitgesmeerd. Men noemt dit *diffusie* van de frequentiedistributie. Als in het cryptogram alle letters even vaak voorkomen, wordt het moeilijk om uit de letterdistributie conclusies te trekken omtrent de identiteit van individuele letters. Deze eis geldt ook voor deelverzamelingen uit het cryptografisch materiaal.

De andere belangrijke zwakke plek die moet worden afgedekt betreft de mogelijkheid tot het opdelen van de verzameling sleutels. In het voorbeeld van de monoalfabetische substitutie viel te zien hoe effectief dat kan zijn. Als voor elke sleutelletter het aantal mogelijkheden kan worden ingeperkt, daalt in dat systeem het totaal aantal plausibele sleutels geweldig. Een cryptosysteem moet dus als het even kan zo worden ontworpen dat voor elke component van de sleutel alle mogelijkheden open blijven.

Meer formeel kan dit als volgt worden verwoord. De verzameling sleutels wordt opgevat als een ruimte opgespannen door de coördinaten van het sleutelstelsel^{*}. Een cryptoanalist wordt verondersteld experimen-

* Een alfabetische sleutel van lengte 10 bijvoorbeeld, geeft een verzameling sleutels die wordt opgespannen door 10 onderling onafhankelijke coördinaten, elk met 26 mogelijke waarden.

ten uit te kunnen voeren waarvan het resultaat wordt beschreven als een functie met de sleutelcoördinaten als parameters.

$$\text{sleutel} = k_1, k_2, \dots \quad \text{waarneming} = f(k_1, k_2, \dots) \quad (1.5)$$

Men spreekt van een goede *menging* door het cryptosysteem als geen der waarnemingen van een (kleine) subset van de coördinaten $k_1, k_2, \dots$ afhankelijk is. Anders gezegd, in elke waarneming moet elke component van de sleutel een bijdrage tot de functiewaarde geven. Zoniet, dan kunnen wellicht door selectieve experimenten onderdelen van de sleutel uit het geheel worden geïsoleerd en afzonderlijk bepaald. Ook bestaat dan het gevaar dat de sleutelruimte in zulke kleine delen uiteen valt, dat ze met grote kans op succes uitputtend kunnen worden doorzocht.

1.7 Eigenschappen

Enkele algemene eigenschappen van cryptografische transformaties zullen nu de revue passeren. Ze beschrijven nagenoeg allemaal effecten die hun weerslag hebben op de kwaliteit van een cryptosysteem.

1.7.1 Gesloten systeem

Het vercijferen en ontcijferen is een transformatie die wordt uitgevoerd met de sleutel als parameter. Het ontcijferen is de inverse operatie van vercijferen.

$$C = E(M, K) \quad \text{en} \quad M = D(C, K) = E^{-1}(C, K) \quad (1.6)$$

Het lijkt op het eerste gezicht vanzelfsprekend dat bij elke klaartekst M één bepaalde cijfertekst C behoort en dat dit cryptogram zich op unieke wijze laat ontcijferen tot de oorspronkelijke klaartekst. Toch is dat niet zonder meer het geval. Er bestaan er cryptosystemen waarbij verschillende berichten eenzelfde

de cryptogram kunnen opleveren. Cryptogrammen in zo'n cryptosysteem zijn bijgevolg niet eenduidig ontcijferbaar*.

Er zijn veel berichten denkbaar en elk van deze kan met elke sleutel gecijferd worden. Het maximaal aantal te produceren cryptogrammen is het product van het aantal klaarteksten en het aantal sleutels. Meestal zullen het er echter minder blijken te zijn, omdat sommige combinaties van klaartekst en sleutel hetzelfde cryptogram opleveren. Met $|M|$ het aantal klaarteksten en $|K|$ het aantal sleutels

$$\text{maximaal aantal cryptogrammen} = |M| \cdot |K| \quad (1.7)$$

Het precieze aantal gerealiseerde verschillende gecijferingen verschilt per cryptosysteem. Verschillende klaartekst, elk met een andere sleutel gecijferd, kan toch heel goed eenzelfde cijfertekst opleveren. Eveneens kan het voorkomen dat verschillende berichtssleutels bij eenzelfde klaartekst maar één cijfertekst produceren. Een eenvoudig voorbeeld van dit laatste levert de monoalfabetische substitutie. Neem twee sleutels slechts verschillend in een verwisseling van de substituenten voor Q en Z. Nu zal elke klaartekst waarin deze twee letters niet voorkomen hetzelfde cryptogram opleveren ongeacht welke van deze twee sleutels is gebruikt.

In het meest ideale geval kan elke klaartekst worden gecijferd met elk van de sleutels tot cryptogrammen die met al die sleutels weer uniek kunnen worden ontcijferd. Een systeem dat hieraan voldoet wordt een *gesloten systeem* genoemd. Vooral wanneer ook nog elk cryptogram uit elke klaartekst kan zijn ontstaan, vormt dit een ideale combinatie. De reden daarvoor is dat nu elk cryptogram met elke sleutel kan worden ontcijferd en dan een in principe mogelijke oplossing geeft. Daardoor kan bij waarneming van een bepaald cryptogram geen enkele oplossing op voorhand worden uitgesloten en zo blijft de maximale onzekerheid omtrent de echte klaartekst gehandhaafd.

* De ambiguïteit die hiervan het gevolg is, leidt er toe dat men zulke cryptosystemen in de praktijk zal mijden.

1.7.2 Involutie

Bij het maken van machines voor cryptografie rees de wens om met één operatie zowel te kunnen verscijferen als te ontcijferen. Zodoende kan met slechts één enkel mechanisme worden volstaan, terwijl anders een aparte verscijfer- en ontcijfermachinerie nodig zou zijn. De cryptotransformatie E in onderstaande vergelijking die daaraan voldoet heet een *involutie*. In feite geldt dus $E = D$.

$$C = E(M, K) \quad \text{en} \quad M = E(C, K) \quad (1.8)$$

Voorbeelden van apparaten waarin dit principe is benut zijn de Hagelin cryptografen en de Enigma. Zowel de Beaufort substitutie in de Hagelin als de rotorsubstitutie van de Enigma zijn involuties (zie hoofdstuk ?? voor een beschrijving van deze machines).

1.7.3 Endomorfie

Men zou een verscijfering herhaald willen kunnen uitvoeren. Niet elk cryptosysteem geeft echter de gelegenheid tot willekeurige herhaling van de cryptotransformatie. Een voorwaarde voor herhalen is dat de verzameling symbolen van de klaartekst terugkeert in de cijfertekst. Het systeem heet dan een *endomorf cryptosysteem*. In een endomorf systeem geldt dus

$$M = s_a s_b \dots \text{ met } s_a, s_b, \dots \in \{S\} \rightarrow C = E(M, K) = s_x s_y \dots \text{ met } s_x, s_y, \dots \in \{S\} \quad (1.9)$$

Een voorbeeld van een niet-endomorf systeem is het gebruik van een cijfercode. Want daarin wordt alfabetische tekst omgezet in cijfergroepen, die zelfs onderling verschillend van lengte kunnen zijn.

1.7.4 Idempotent

In een endomorf cryptosysteem kan een verscijfering herhaald worden uitgevoerd, hetzij met dezelfde bericht sleutel hetzij met een steeds wisselende sleutel. De vraag die hierbij kan worden gesteld is of dit

nu een beter, dat wil zeggen sterker resultaat oplevert dan een enkelvoudige uitvoering. Zo ja, dan kan van een betrekkelijk eenvoudig en misschien niet zo sterk geheimschrift door herhaling toch een sterker geheimschrift worden gemaakt.

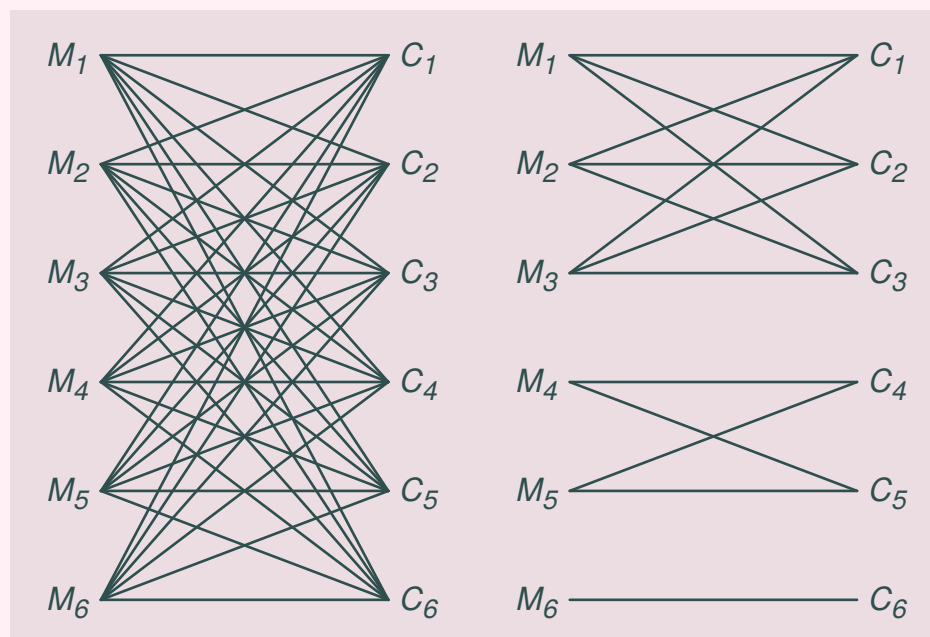
Dit blijkt voor sommige systemen op te gaan en voor andere niet. De monoalfabetische substitutie is een voorbeeld van een systeem waarin herhaling geen verbetering brengt. Gemakkelijk valt in te zien dat het effect van elke serie opeenvolgende monoalfabetische vercijferingen ook is te bewerkstelligen door middel van één vercijfering met een geschikt gekozen sleutel. Een cryptosysteem als dit heet een *idempotent cryptosysteem* omdat herhaalde vercijfering geen effect heeft.

$$\forall K_i, K_j \quad \exists K_m: \quad E(M, K_m) = E(E(M, K_j), K_i) \quad \text{of} \quad E_m = E_i E_j \quad (1.10)$$

Andere systemen bieden wel degelijk een verbetering door herhaling van een basisstap. Veel moderne systemen maken hiervan gebruik. Een goed voorbeeld is het Feistel-geheimschrift, dat in [hoofdstuk 8](#) wordt behandeld. Het is overigens verstandig om er voor te zorgen dat de lengte van het cryptogram bij de herhaalde vercijfering niet toeneemt, want anders dijt het cryptogram door de herhaling sterk uit.

1.7.5 Restklassen

Zojuist is verteld dat er systemen (zoals de monoalfabetische substitutie) zijn waarin voor elk paar achtereenvolgende vercijferingen een derde vercijfering kan worden gevonden die hetzelfde resultaat geeft. De vercijferoperaties kunnen dan een zogenaamde *algebraïsche groep* vormen. Dit heeft belangrijke consequenties voor de structuur en daarmee samenhangend, voor de sterkte van het systeem. Want als een cryptosysteem zo'n algebraïsche groep vormt, kan het alle eigenschappen van die groep vertonen en gelden allerlei regels uit de wiskundige groepentheorie. In het bijzonder van belang is dan het feit dat de verzameling van vercijferoperaties in afzonderlijke delen, de zogenaamde *restklassen* uiteen kan vallen.



Figuur 1.7 Restklassenstructuur

In [figuur 1.7](#) wordt hiervan een voorbeeld gegeven met twee verschillende cryptosystemen, elk met zes mogelijke klaarteksten, zes sleutels en zes mogelijke cijferteksten. Links staat een systeem waarin elke sleutel, toegepast op elke klaartekst, een andere cijfertekst geeft. Iedere cijfertekst kan uitgaande van elke klaartekst bereikt worden door een passende keuze van de sleutel. Het effect hiervan blijkt als een cijfertekst wordt aangetroffen en de vraag wordt gesteld: "Wat is de oorspronkelijke klaartekst?" Het antwoord is even eenvoudig als ontmoedigend: "Elke mogelijke klaartekst komt gelijkelijk in aanmerking." Maar als alle klaarteksten even waarschijnlijk zijn, valt de vraag "Wat is de goede ontcijfering?" niet te beantwoorden omdat ze het immers allemaal kunnen zijn. Deze structuur is dus prima voor een optimale beveiliging van de klaartekst.

Vervolgens wordt de aandacht gericht op het systeem aan de rechterkant van [figuur 1.7](#). Hier is de struc-

tuur zodanig dat het stelsel van zes klaarteksten, zes bericht sleutels en zes cijferteksten uiteenvalt in drie restklassen met van boven naar beneden het aantal van drie, twee en één elementen. Na het vinden van een cijfertekst wordt nu opnieuw de vraag gesteld: "Wat is de oorspronkelijke klaartekst?" Als de cijfertekst zich in de restklasse met drie elementen bevindt, komen slechts drie van de zes klaarteksten in aanmerking. En komt het uit de onderste van drie restklassen in de figuur, dan is er slechts één mogelijkheid. Blijkbaar vergroot het uiteenvallen van een cryptosysteem in restklassen de kans om de juiste bijbehorende klaartekst te vinden. De groepsstructuur kan dus een aanzienlijke verzwakking van het systeem tot gevolg hebben.

In de monoalfabetische substitutie is dit gedrag prominent aanwezig. De letterpatronen van de woorden vormen de restklassen van dit systeem. Hun optreden biedt een houvast voor de cryptoanalyse. Een houvast dat er niet zou zijn wanneer de letterpatronen van de woorden zouden zijn versluierd.

1.7.6 MASTERSECTIE – Zuiver cryptosysteem

list=Zuiver cryptosysteem,] Om een cryptotransformatie te kunnen itereren is het noodzakelijk dat het systeem een endomorf cryptosysteem volgens **vergelijking (1.9)** is. Anders kunnen immers de achtereenvolgende transformaties niet worden uitgevoerd. Nu zijn er ook cryptosystemen denkbaar die weliswaar niet endomorf zijn, maar toch een groep vormen. Om die systemen eveneens met behulp van de groepentheorie te kunnen benaderen, moet een geschikte groepsoperatie worden gedefinieerd. Waar in het geval van een endomorf cryptosysteem direct twee opeenvolgende operaties gekoppeld kunnen worden, komen er nu drie. Een van deze drie is een ontcijfering die de schakel vormt naar de volgende vercijfering.

Een cryptosysteem dat onder die gecombineerde operatie een groep vormt, heet een *zuiver cryptosysteem*. Een niet zuiver cryptosysteem heet een *gemengd cryptosysteem*. De definitie van een zuiver cryptosysteem luidt als volgt. Een cryptosysteem heet zuiver als voor elke drie transformaties E_i , E_j en E_k geldt dat voor het achtereenvolgens uitvoeren van vercijfering E_k , ontcijfering $D_j = E_j^{-1}$ en vercijfering E_i een equivalente

vercijfering E_m bestaat. Een aanvullende voorwaarde is nog dat alle sleutels even waarschijnlijk moeten zijn.

$$\forall i, j, k \quad \exists m: \quad E_m = E_i E_j^{-1} E_k = E_i D_j E_k \quad (1.11)$$

Eerst wordt aangetoond dat in een zuiver cryptosysteem de gecombineerde operatie $D_i E_j$ een groepsoperatie is. Met I de identiteit, volgt uit

$$(D_i E_j)(D_j E_i) = D_i (E_j D_j) E_i = D_i I E_i = D_i E_i = I \quad (1.12)$$

dat elke operatie $D_i E_j$ een inverse heeft. Als het cryptosysteem zuiver is, betekent dit volgens de definitie van zuiverheid dat elk product $E_i D_j E_k$ overeenkomt met een of andere E_m in het systeem. Hiermee kan worden aangetoond dat het product van twee operaties weer een operatie in het systeem is.

$$(D_i E_j)(D_p E_q) = D_i E_j D_p E_q = (E_p D_j E_i)^{-1} E_q = D_m E_q \quad (1.13)$$

Tezamen leveren deze twee conclusies het bewijs dat de operaties $D_i E_j$ een groep vormen. Opgemerkt kan nog worden dat elk endomorf cryptosysteem waarvan de operaties een groep vormen eveneens een zuiver cryptosysteem is. Immers direct uit de groepeigenschap dat voor alle paren i, j er een m is zodanig dat $E_m = E_i E_j$ volgt

$$E_i D_j E_k = E_i (D_j E_k) = E_i D_m = E_p \quad (1.14)$$

waarmee aan de voorwaarde voor zuiverheid is voldaan.

1.7.7 Commuteren

Een versterking van de cryptotransformatie kan men proberen te bewerkstelligen door het combineren van verschillende cryptosystemen. Om dit te kunnen doen is het een voorwaarde dat de cijfertekst die uit een vercijferstap komt, ingevoerd moet kunnen worden in de volgende stap. Dat wil zeggen, de uitvoer van de

eerste vercijfering moet compatibel zijn met de invoer van de tweede. Men spreekt van een *productvercijfering* als twee cryptosystemen kunnen worden gecombineerd door eerst met het ene cryptosysteem te vercijferen en de resulterende cijfertekst met het tweede.

$$E = E'E'' \quad \text{of} \quad C = E'(E''(M, K''), K') \quad (1.15)$$

Voor het maken van een echt sterk systeem blijkt het van belang dat deze twee operaties niet *commuteren*. Als twee operaties commuteren betekent dit dat de volgorde waarin ze worden uitgevoerd van geen belang is. De volgorde van de operaties kan dan zonder meer worden verwisseld. Als de twee vercijferoperaties niet commuteren geldt

$$E'E'' \neq E''E' \quad \text{of} \quad E'(E''(M, K''), K') \neq E''(E'(M, K'), K'') \quad (1.16)$$

Waarom is dit niet-commuteren belangrijk? Veronderstel dat een van beide systemen een idempotent cryptosysteem is, waardoor het herhalen van de vercijferoperatie geen wezenlijke versterking van het geheimschrift teweeg brengt. Als gevolg van het commuteren kan men alle operaties van het idempotente systeem samenbrengen, waardoor ze feitelijk tot slechts één enkele operatie worden gereduceerd. Met als resultaat dat het effect van de herhaling door het commuteren van de twee systemen volledig teniet gedaan wordt.

$$E'_1 E''_1 \dots E'_n E''_n = (E'_1 \dots E'_n) E''_1 \dots E''_n = E'_1 E''_1 \dots E''_n \quad (1.17)$$

MASTERSECTIE – Een andere reden voor het belang van niet-commuteren is gelegen in de volgende stelling: het product van twee commuterende zuivere cryptosystemen is weer een zuiver cryptosysteem. Met als consequentie dat ook het productsysteem een groep is die weer in restklassen uiteen kan vallen. Op die manier raakt men de restklassenstructuur niet kwijt.

Aantonen van de stelling is niet moeilijk. Wanneer twee systemen commuteren geldt voor elke combinatie $E'_i E''_j$ dat er specifieke vercijferoperaties zijn waarvan de uitkomst dezelfde is bij omdraaien van de bewerkingen

$$\forall i, j \quad \exists p, q: E'_i E''_j = E''_p E'_q \quad (1.18)$$

Nu moet worden aangetoond dat in het productsysteem is voldaan aan de zuiverheidseis, hetgeen wil zeggen dat

$$(E'_i E''_j)(E'_p E''_q)^{-1}(E'_x E''_y) = E'_u E''_v \quad (1.19)$$

en inderdaad volgt uit de commutatierelatie van E' en E'' door herhaaldelijk verwisselen van factoren, dat deze uitdrukking kan worden herleid tot

$$E'_i E''_j D''_q D'_p E'_x E''_y = (E'_i D'_a E'_b)(E''_c D''_d E''_y) = E'_u E''_v \quad (1.20)$$

De beste en sterkste combinaties ontstaan door het combineren van niet-commuterende systemen van geheel verschillende aard, zoals bijvoorbeeld een substitutie met een transpositie. Bij de behandeling in [hoofdstuk 8](#) van de moderne systemen zal dit uitgangspunt terugkeren.

1.7.8 Gelijksoortigheid

In de geschiedenis komt men vaak systemen tegen die door de uitvinder als nieuw worden gepresenteerd, maar die alleen ogenschijnlijk van een bestaand systeem verschillen. Twee systemen heten *gelijksoortig* als het ene systeem door een transformatie in het andere kan worden overgevoerd.

$$E' = f(E'') \quad \text{en} \quad E'' = g(E') \quad (1.21)$$

Als men de gelijksoortigheid tussen twee systemen heeft aangetoond, zijn ze natuurlijk beide gebroken zodra een van de twee gebroken is.

Een triviaal voorbeeld is een vercijfering waarin de letters systematisch zijn vervangen door een verzameling figuren of vreemde tekens. Substitueer een willekeurige letter voor ieder figuurtje en de bekende monoalfabetische substitutie is weer terug.

1.8 Blokgeheimsschrift

Met letters als samenstellende elementen van de klaartekst ligt het voor de hand om eveneens letter voor letter te vercijferen. Veel van de te behandelen klassieke geheimsschriften doen dit zo. Er zijn echter ook geheimsschriften waarin verscheidene elementen worden samengenomen om gezamenlijk te worden vercijferd. In het Playfair-geheimsschrift bijvoorbeeld, worden twee letters (een digram) tezamen vercijferd tot een ander digram. Het AES-geheimsschrift is een zogenaamd *blokgeheimsschrift* waarin een *blok* van zestien bytes wordt vercijferd.

In oude geheimsschriften – codes uitgezonderd – gaat het combineren meestal niet verder dan twee of drie letters. Hedendaagse blokgeheimsschriften combineren gewoonlijk 8 of 16 bytes (64 of 128 bits) in het vercijferblok. Niet toevallig vindt men daarin een weerspiegeling van de stand van de computertechniek. De breedte van de registers in de processor en de woordlengte van de geheugens bepalen mede hoe efficiënt en snel het vercijferen en ontcijferen kan worden uitgevoerd. Het DES-geheimsschrift stamt uit de zeventiger jaren van de 20^e-eeuw en kent een bloklengte van 64 bits, de opvolger AES kent blokken van 128 bits en heeft de mogelijkheid om tot 256 bits te gaan.

De lengte van het blok mag niet te klein gekozen worden met het oog op uitputtend zoeken door middel van een zogenaamde *message exhaustion aanval*. Hierbij worden zoveel mogelijk paren klaartekst en cijfertekst voor een bepaalde sleutel vooraf berekend en gesorteerd opgeslagen. Het is dan een klein kunstje om van een onderschept cryptogram met bekend veronderstelde klaartekst de bijbehorende sleutel te vinden. In de Tweede Wereldoorlog pasten de codebrekers van Bletchley Park deze methode toe op het in Duitse berichten veelvuldig voorkomende telwoord *eins*. Met een machine genaamd de Baby werd bij de eerder gevonden instelling van de Enigma een catalogus van de vercijfering van dit woord gemaakt voor alle letterposities^[9].

Een tweede reden om het blok niet te klein te nemen is om het verzamelen van statistische gegevens te bemoeilijken. Omdat een blok van 1 byte een afbeelding is van de verzameling getallen van 0...255

naar een permutatie hiervan, is in wezen sprake van een monoalfabetische substitutie met een alfabet van 256 tekens. Om die reden heet het afzonderlijk vercijferen van blokken klaartekst met eenzelfde sleutel *Electronic Code Book* (ECB-modus). Een monoalfabetische substitutie met een alfabet van 256 symbolen kan zonder problemen in kaart gebracht worden. Bij grotere bloklengte wordt dat moeilijker, want een 64-bit blok heeft al $2^{64} \approx 18 \times 10^{18}$ waarden en laat zich niet zomaar uittellen.

Bij een blokgeheimschrift is evident dat minder dan een compleet blok niet kan worden vercijferd. Maar wat als de te vercijferen tekst niet een geheel aantal malen de bloklengte is? De tekst moet dan worden aangevuld totdat ook het laatste blok geheel vol is. Dat heet *uitvullen* van het blok (padding in het Engels). Nu is bij letters de verleiding groot om met X'en aan te vullen en bij byte-blokken om dit met 0-bits te doen. Zo'n naïef opvulschema is natuurlijk fout omdat het de cryptanalist onnodig houvast geeft. Zulke informatie kan worden gebruikt om testen mee uit te voeren en bepaalde sleutels te kunnen uitsluiten. Uitvullen van de blokken moet dus zorgvuldig gebeuren en het beste is om willekeurige letters of bits te gebruiken.

1.8.1 Implementatie aspecten

In het voorgaande is gesuggereerd dat voor blokgeheimschriften de ECB-modus niet zo veilig is vanwege het inherent monoalfabetisch karakter ervan. Daarom zijn methoden bedacht om op betrekkelijk eenvoudige wijze die veiligheid te verbeteren.

Figuur 1.8 geeft een klaartekst met een zeer duidelijke structuur. Deze is vercijferd in blokken van 8 letters, elk blok met dezelfde sleutel. De structuur van de klaartekst is in de vercijfering in **figuur 1.9** onmiskenbaar aanwezig. Die structuur in de klaartekst kan onzichtbaar gemaakt worden door de vercijferde blokken *aan elkaar te knopen*. Deze methode heet *Cipher Block Chaining* (CBC-modus). Elk opvolgend klaartekstblok M_i wordt gecombineerd met het vercijferd vorig blok C_{i-1} .

$$C_i = E(M_i \oplus C_{i-1}, K) \quad (1.22)$$

* MYSTERY PROGRAM	00000010
* D.E.KNUTH	00000020
* THE ART OF COMPUTER PROGRAMMING, VOL I	00000030
* EXERCISE 1.3.2 # 8	00000040
PRINTER EQU 18	00000050
BUF ORIG *+3000	00000060
1H ENT1 1	00000070
ENT2 0	00000080
LDX 4F	00000090
2H ENT3 0,1	00000100

Figuur 1.8 Sterk gestructureerde klaartekst

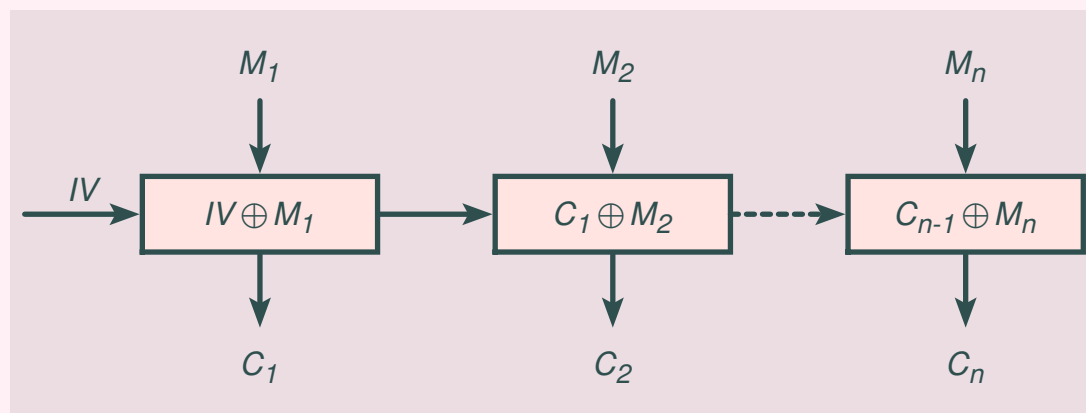
```
\BKF<&bm/qKI5"Uz}eKxbR.H\8KxbR.H\8KxbR.H\8KxbR.H\8KxbR.H\8KxbR.H\H[]rb?X  
\BK=pw<s+m AbR.H\8KxbR.H\8KxbR.H\8KxbR.H\8KxbR.H\8KxbR.H\8KxbR.H\H[]rb@  
\BKM+w.i/7KH)RQw*h!M(%x/grK$ [q+_wx9"ZH&8KxbR.H\8KxbR.H\8KxbR.H\H[]rbAX  
\BK>;w`k&kpxs`Avn8NxZR.H\8KxbR.H\8KxbR.H\8KxbR.H\8KxbR.H\8KxbR.H\H[]rbBX  
\h}B1'Sz\8K>4(.H\IcxbR.H\8KxbR.H\8KxbR.H\8KxbR.H\8KxbR.H\8KxbR.H\H[]rbCX  
\Z!?bR.H\8KH5{UH\BV,rb>H\8KxbR.H\8KxbR.H\8KxbR.H\8KxbR.H\8KxbR.H\H[]rbDX  
\IsxbR.H\8K>1'?H\IKxbR.H\8KxbR.H\8KxbR.H\8KxbR.H\8KxbR.H\8KxbR.H\H[]rbEX  
\8KxbR.H\8K>1'@H\HKxbR.H\8KxbR.H\8KxbR.H\8KxbR.H\8KxbR.H\8KxbR.H\H[]rbFX  
\8KxbR.H\8KE'+.H\LqxbR.H\8KxbR.H\8KxbR.H\8KxbR.H\8KxbR.H\8KxbR.H\H[]rbGX  
\JsxbR.H\8K>1'AH\HW*bR.H\8KxbR.H\8KxbR.H\8KxbR.H\8KxbR.H\8KxbR.H\H[]rc>X
```

Figuur 1.9 Vercijfering in ECB-modus

De keten wordt begonnen met IV , de *initiële vector*.

$$\begin{aligned} C_1 &= E(M_1 \oplus IV, K) \\ C_i &= E(M_i \oplus C_{i-1}, K) \quad i = 2 \dots n \end{aligned} \tag{1.23}$$

Figuur 1.10 toont het vercijferen in CBC-modus. Het is van belang om IV willekeurig te kiezen en zeker ook van bericht tot bericht te laten verschillen. Zo niet, dan lekt informatie over de klaartekst uit.



Figuur 1.10 Cipher Block Chaining vercijfering

Het aan elkaar rijgen van de opeenvolgende blokken maakt dat blokken met eenzelfde inhoud toch telkens anders de vercijfering met de ene sleutel K ingaan en dus een ander resultaat opleveren. De structuur van de klaartekst verdwijnt daardoor uit de cijfertekst, zoals in **figuur 1.11** duidelijk is te zien. Er is echter een maar bij deze manier van vercijferen. Wat gebeurt er als in de overbrenging van het bericht ergens een fout optreedt? Bij afzonderlijke vercijfering van elk blok wordt hoogstens één blok aangetast, maar in CBC-modus worden dan twee opvolgende blokken onleesbaar: het blok waarin de fout is opgetreden en het blok erna dat met dit foutieve blok wordt behandeld.


```
\BKF<&bm,T70qHX)JZ#It;'qG3nbw.UZdKZ{z!$CADF5}sR,>|2N!f!t;U}g$Y0]H>y17\ /V
E!enHtkjp/&PKg:SmgqiNZh<j@]#QM7%gxI<T@emdQ5UW34Va*!nZ&b?^b1(]x1(kKhQp{q!
h.T?<4@+8;@(e'2CbDau.L`\2LtaR1\n]L1z,/WW$X4/"&@!]DM2tT)}60f5g#q+~,0Hjddj
(awn$#evNm)(8$Gm]FwASvuVZ~cZViD?WW0sY\r(T0;- \OApQh'F_BoYNAr_b5>B[*n)u8!;
X3-kG_tVUkxJ{(C?RU|c~zq(O.h|"m@pLFT6%`nYI?@O(S=BFw,h+Fk+CPw".9:sP9sKA<}1
M45+D/LUJ1!syKB>G0w@-N!'D(cY0A0oA`Or34}X>9;;6'LA;q'E9yz*8Jr^<1IrE3n(Oo.k
B|#ARb\T?Un $*<=<?Z9'|j&9wFR*o9n6P2k-bgW3)}%0U6@0ai>3Hd)-:UW6;3q:#Q!I>xj
7[=:L1GS44)x}X(<1|t2!KV%.U`K$>%m+.Ld'1SV(f8}*"$"?%?$7-vP("woP0i~p/`kyC1ei
,9W3F_4R)qCxm+b;&^U2p}1$#7AKsp_1 o-dvc.U|Hx}yV\>y!d7|I+'vYPP <Yo$BLY3?Ah
!-`362oQ}eLqgYQ:zND<jL #w'0Um?Nkt_{np2|Tq8g(s%K=npSAvwy&kI?ZyjHnx2;$-n'g
```

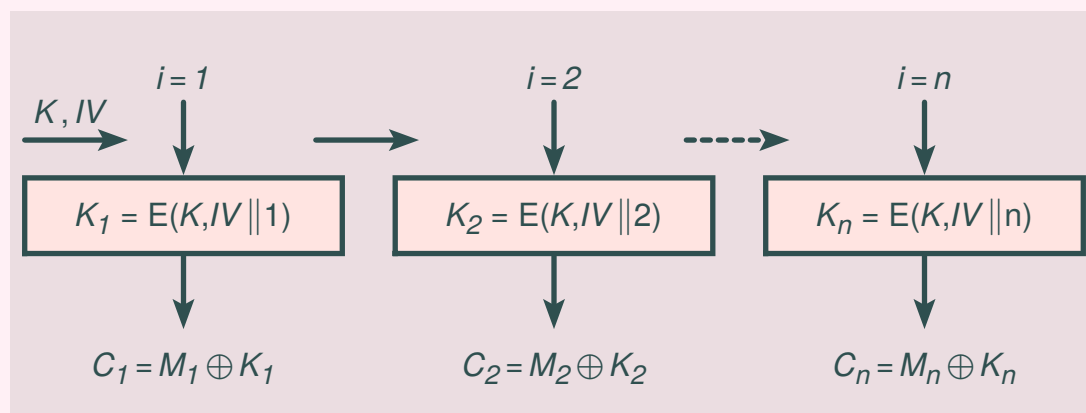
Figuur 1.11 Vercijfering in CBC-modus

Naast CBC-modus is er ook de mogelijkheid om niet de klaartekst te modifieren maar dit met de sleutel te doen. Er ontstaat dan een keten van elkaar opvolgende sleutels. Deze methode heeft als nadeel dat één enkele fout in een vercijferd blok vanaf dat moment alle volgende blokken eveneens onleesbaar maakt. Aan de andere kant kan dit verschijnsel ook nuttig gebruikt worden. Als aan het eind van het bericht een checksum meevercijferd wordt, kan als gevolg van dit effect eenvoudig worden vastgesteld of het bericht intact is.

Een populaire variant is de zogenaamde *counter mode* (CTR-modus). Samen met andere wordt deze aanbevolen door het Amerikaanse National Institute of Standards and Technology (NIST)^[10]. CTR-modus onderscheidt zich doordat het klaartekstblok niet rechtstreeks vercijferd wordt met de sleutel, maar met een voortdurend wisselende daarvan afgeleide sleutel. Het eenvoudigst doet men dit met de blokteller.

$$\begin{aligned} K_i &= E(K_i, IV \parallel i) \quad i = 1 \dots n \\ C_i &= M_i \oplus K_i \end{aligned} \tag{1.24}$$

Ook hier wordt gestart met een initiële vector IV , die om veiligheidsredenen weer voor elke te vercijferen tekst verschillend moet zijn. Daarnaast krijgt elk te vercijferen blok een eigen volgnummer i . De IV en het volgnummer worden geconcateneerd ($IV \parallel i$) en vormen samen een sleutel waarmee de oorspronkelijk ingegeven vercijfersleutel K wordt vercijferd. Het resultaat van die vercijfering is dan een voortdurend veranderende sleutel die middels een eenvoudige exclusive or de klaartekst beschermt.



Figuur 1.12 Counter modus vercijfering

1.8.2 Beperkte range

Een minder fraai aspect van blokgeheimschriften is dat ze vrijwel nooit alle mogelijke substituties benutten of zelfs maar een flink deel daarvan. Dit valt in te zien door de cryptotransformatie op te vatten als een afbeelding van een verzameling klaartekstblokken naar een verzameling vercijferde blokken. Voor een bloklengte van n bits moeten 2^n elementen afgebeeld worden naar eveneens 2^n vercijferde blokken. Een vercijfering met een bepaalde sleutel kan nu worden opgevat als de realisering van een afbeelding waarbij (willekeurig) aan elk klaartekstblok een bepaald cijfertekstblok wordt toegevoegd.

$$\forall i: E(M_i, K) \leftrightarrow C_i \quad (1.25)$$

De eis dat vercijferen en ontcijferen unieke operaties moeten zijn, leidt ertoe dat de vercijferingen een *permutatie* van de afgebeelde blokken moeten vormen, want anders kan de afbeelding geen *bijjectie* zijn*. Het is dus niet voldoende als de vercijfering slechts een pseudorandom resultaat oplevert. In andere bewoordingen: beschouw de toewijzing van de vercijferde blokken aan de klaartekst onder een zekere sleutel als een loterij, die moet geschieden door willekeurige trekking *zonder* teruglegging.

In het geval van een blok van n bits is het een afbeelding van $2^n \rightarrow 2^n$ elementen. Bijgevolg zijn er $2^n!$ permutaties voorhanden voor toewijzing aan elk van de sleutels. Als de sleutel n bits telt zijn er niet meer dan 2^n sleutels en wordt dus maar een kleine fractie van het totale aantal van $2^n!$ realiseringmogelijkheden benut want $2^n \ll 2^n!$.

Het feit dat een blokgeheimschrift een $2^n \rightarrow 2^n$ afbeelding bewerkstelligt heeft nog een ander gevolg. Bij herhaald uitvoeren van de vercijfering met één bepaalde sleutel moet onafwendbaar vroeg of laat een vercijferresultaat ontstaan dat al eerder in de reeks is voorgekomen

$$C_1 = E(M, K), C_2 = E(C_1, K), \dots, C_p = E(C_{p-1}, K) = M \quad (1.26)$$

Variabele p in deze formule heet de *periode* van de *cyclus* die wordt gevormd door de combinatie van M en K . Aangezien het blok n bits telt, is 2^n de langst mogelijke periode. Vaak zullen er cycli van verschillende lengte voorkomen, sommige lang en andere van kort. Is er meer dan één cyclus, dan wisselt die met de gekozen sleutel. In het extreme geval $p = 1$ levert vercijfering direct weer de klaartekst terug. Men spreekt dan van een *fixed point* en van een *anti-fixed point* als het complement resulteert.

$$E(M, K) = M \quad \text{of} \quad E(M, K) = \overline{M} \quad (1.27)$$

Bij $p = 2$ heet K een *zwakke sleutel* (weak key in het Engels).

$$E(E(M, K), K) = M \quad (1.28)$$

* Bijjectie: een één op één afbeelding; noodzakelijk om elk blok uniek te kunnen vercijferen en ontcijferen.

1.9 Literatuurlijst

Een nuttige en actuele bron van informatie vormt het ePrint archief van de International Association for Cryptologic Research (IACR) op www.iacr.org. Veel auteurs plaatsen hierop een preprint van artikelen die pas veel later in een tijdschrift of congresbundel zullen verschijnen. Dit archief is bereikbaar onder eprint.iacr.org en bevat artikelen van 1996 tot heden. De populariteit ervan is inmiddels sterk gegroeid. Eerst goed begonnen in het jaar 2000 met 69 artikelen, telt de lijst er al 1257 in 2015.

Hieronder volgt een beknopt literatuuroverzicht met titels relevant voor de behandelde onderwerpen. Deze kunnen worden gebruikt als beginpunt voor verdere studie. Sommige van de genoemde boeken zijn een begrip, sommige zijn van historisch belang en weer andere gelden als standaardwerk op hun terrein. De laatste decennia is op het vakgebied cryptografie zoveel verschenen dat het onmogelijk is om alle relevante titels in kort bestek te noemen. Het niet noemen van een bepaald boek houdt dus geen diskwalificatie in. De lijst is in alfabetische volgorde op auteur.

1. *Anon. (1942). A Course in Cryptanalysis, volume I and II. Aegean Park Press.*

Historisch interessant materiaal. Het betreft hier een cursus cryptoanalyse in Engeland tijdens de Tweede Wereldoorlog. Sommige opgaven, zoals de codeberichten uit de strijd in 1941 om Tobruk in Noord-Afrika, zijn nog steeds interessant om te doen en zeker niet gemakkelijk.

2. *J. Bamford (1983). The Puzzle Palace. Penguin Books.*

Eerste boek van deze schrijver over de machtige Amerikaanse NSA, de National Security Agency, aanvankelijk zo geheimzinnig dat ze ook wel "No Such Agency" is genoemd.

3. *J. Bamford (2001). Body of Secrets: Anatomy of the Ultra-secret National Security Agency - From the Cold War through the Dawn of a New Century. Doubleday.*

Vervolg op het vorige boek van deze schrijver over de wheelings and dealings van de NSA

4. *F. L. Bauer (2007). Decrypted Secrets, Methods and Maxims of Cryptology. Fourth edition Springer-Verlag.*

Algemeen overzicht van de cryptografie met nadruk op het klassieke cryptosysteem. Veel informatie over de cryptoanalyse van de behandelde systemen. Nogal compact geschreven.

5. *H. Beker and F. Piper (1982). Cipher Systems, The Protection of Communications. Northwood Books, London.*

Een wat gedateerd boek, maar een van de weinige waarin de analyse van de Hagelin cryptograaf wordt beschreven en veel aandacht voor schuifregisters.

6. *D. E. Boekee and J. C. A. van der Lubbe (1988). Informatietheorie. Delftse Uitgevers Maatschappij.*

In het Nederlands geschreven inleiding tot de informatietheorie. Van origine een collegedictaat van de TU Delft.

7. *T. M. Cover and J. A. Thomas (1991). Elements of Information Theory. John Wiley & Sons, Inc.*

Uitgave in de Wiley Series in Telecommunications, geheel gewijd aan de behandeling van de informatietheorie. Doorwrochte aanvulling op het behandelde [hoofdstuk 7](#).

8. *C. A. Deavours and L. Kruh (1985). Machine Cryptography and Modern Cryptanalysis. Artech House, Inc..*

Dit boek behandelt de geschiedenis en cryptoanalyse van een aantal bekende cryptografische machines. Zoals de Amerikaanse Hebern, de Duitse Enigma, de Zweedse Hagelins en de Japanse Red en Purple codeermachines.

9. *W. F. Friedman and L. D. Callimahos (1985). Military Cryptanalytics. Aegean Park Press.*

Dit is een heruitgave van een cursus die in de jaren 1956–1957 werd geschreven voor de cursus cryptoanalyse van het Amerikaanse leger. Oorspronkelijk geheim, maar begin jaren tachtig gedeclas-

sificeerd. De uit meerdere delen bestaande cursus gaat alleen over klassieke systemen. Feitelijk is het een bewerking van een oudere cursus, bestaande uit tenminste vier delen, geschreven door William Friedman. Laatstgenoemde cursus is bij weten van de schrijver dezes niet heruitgegeven. Hetgeen jammer is omdat bijvoorbeeld de er in voorkomende analyse van het ADFGVX-systeem niet terugkomt in de latere versie.

10. *H. F. Gaines (1956). Cryptanalysis, A Study of Ciphers and their Solution. Dover Publications, Inc., New York.*

Een klassieker en een van de aardigste boeken over het oplossen van klassieke geheimschriften met potlood en papier, dat deze schrijver kent. Nog steeds de moeite van het bestuderen waard, al was het alleen maar vanwege het handwerk. Heruitgave van het oorspronkelijke *Elementary Cryptanalysis* uit 1939.

11. *P. Garrett (2001). Making, Breaking Codes: An Introduction to Cryptology. Prentice-Hall.*

Een inleiding op de cryptografie, en wel voornamelijk van systemen met publieke sleutel. De behandeling is tamelijk wiskundig van aard en gaat op veel onderwerpen uit de onderliggende wiskunde in.

12. *R. Harris (1996). Enigma. Arrow Books.*

Een geromantiseerd verhaal over het breken van de Enigma in de slag om de Atlantische Oceaan. De personages en gebeurtenissen zijn grotendeels gefingeerd, maar de technische details kloppen wel. Naar dit boek is een film gemaakt.

13. *R. D. Haslach (1985). Nishi, No kaze, Hare. Van Kampen & Zn., Weesp.*

In dit boek kan men lezen hoe in de periode voorafgaande aan de Tweede Wereldoorlog de Nederlandse cryptoanalisten Van der Beek en Nuboer in het voormalige Nederlands-Indië Japanse codes kraakten.

14. *It. col. P. Hitt (1976). Manual for the Solution of Military Ciphers. Aegean Park Press.*

Een van de eerste handleidingen voor cryptanalyse geschreven ten behoeve van de opleiding in het Amerikaanse leger. Heruitgave van de oorspronkelijk uit 1916 en 1918 daterende tekst.

15. *D. Kahn (1967). The Codebreakers, The Story of Secret Writing. MacMillan Publ. Co., Inc.*

Het is de verdienste van David Kahn dat de cryptografie uit de schaduw is getreden. Dit is hét standaardwerk van de geschiedenis van de cryptografie en hoewel al meer dan dertig jaar oud, nog steeds de aanschaf ten volle waard. Er is inmiddels een nieuwe editie verschenen.

16. *D. Kahn (1991a). Seizing the Enigma, The Race to Break the German U-Boat Codes, 1939–1943. Houghton Mifflin Company, Boston.*

De geschiedenis van het breken van de Duitse Enigma, toegespitst op het berichtenverkeer van de Duitse Kriegsmarine en de strijd op de Atlantische Oceaan tussen de U-boten en de geallieerde konvooien. Ook onderwerp van een film getiteld U-571 (meer fictie dan historisch getrouw). Op de DVD van de film is David Kahn te zien in een bijgevoegde documentaire.

17. *A. I. Khinchin (1957). Mathematical Foundations of Information Theory. Dover Publications, Inc., New York.*

Uit het Russisch vertaalde artikelen over de grondslagen van Shannon's informatietheorie.

18. *W. Kozaczuk (1989). Geheimoperation Wicher. Bernard & Graefe Verlag, Koblenz.*

Een hommage aan de Poolse wiskundigen, in het bijzonder Marian Rejewski, die voor de oorspronkelijke doorbraak in de Duitse Enigma zorgden. De geschiedenis van hun recrutering en werk in Polen, de vlucht in de zomer van 1939 naar Frankrijk en de voortzetting van het werk aldaar worden beschreven.

19. *S. K. Kullback (1976). Statistical Methods in Cryptanalysis. Aegean Park Press.*

Solomon Kullback was een van de eerste recruten van William Friedman voor zijn team van cryptoanalisten. Tot de grootste prestatie van dat team mag worden gerekend het (ongezien) breken van de Japanse diplomatieke codeermachine, codenaam Purple. Dit boek behandelt statistische methoden die in de cryptoanalyse worden gebruikt.

20. *K. de Leeuw (2000). Cryptology and Statecraft in the Dutch Republic. PhD thesis, Universiteit van Amsterdam.*

Dissertatie over de geschiedenis van de cryptografie in Nederland, van de Tachtigjarige Oorlog tot het begin van de 19^e-eeuw. Laat zien dat op een aantal momenten in voorgaande eeuwen de Nederlandse cryptoanalisten voor niemand onderdeden. Mede de verklaring waarom hun codes in de tweede helft van de 18^e-eeuw tot de beste van die tijd behoorden.

21. *Karl de Leeuw and Jan Bergstra, editors (2007). The History of Information Security, A Comprehensive Handbook. Elsevier.*

Bevat op het gebied van de cryptografie o.a. bijdragen over de Europese Renaissance, de Enigma, de hoogniveau Duitse Lorenz Schlüsselzusatz uit de Tweede Wereldoorlog, de Hagelin en meer.

22. *S. Levy (2000). Crypto, secrecy and privacy in the new code war. Allen Lane The Penguin Press.*

De geschiedenis van de ontwikkeling van de moderne cryptografie, met name van systemen met publieke sleutel. Legt de nadruk op de erbij betrokken personages en de politieke ontwikkelingen in de Verenigde Staten rondom deze systemen.

23. *L. Marks (1998). Between Silk and Cyanide, A Codemaker's Story 1941–1945. HarperCollins Publ..*

Autobiografie van de man die in de Tweede Wereldoorlog codes maakte voor de agenten die in bezet Europa werden gearachuteerd. Vertelt over het zogenaamde Englandspiel, waarin de Duitsers het

radioverkeer van gevangen genomen gedropte agenten heel lang konden voortzetten zonder dat in Engeland acht geslagen werd op het ontbreken van de veiligheidskenmerken.

24. *K.M. Martin (2012). Everyday Cryptography. Oxford University Press.*

Principes van de cryptografie en het gebruik ervan in allerlei aspecten van informatiebeveiliging worden uitgelegd zonder veel wiskunde. Het boek is daardoor in het bijzonder geschikt voor studenten die een beta-achtergrond missen maar toch meer willen weten van de toepassingen van cryptografie.

25. *P. C. van Oorschot, A. J. Menezes and S. A. Vanstone (1997). Handbook of Applied Cryptography. CRC Press, Inc..*

Een uitputtende behandeling van de moderne cryptografie met veel aandacht voor de wiskundige details en een bijzonder uitgebreide bibliografie. Wel heel beknopt geschreven.

26. *R. K. Nichols (1996). Classical Cryptography Course, volume I and II. Aegean Park Press.*

Een uit twee delen bestaande cursus voor het oplossen van klassieke cryptosystemen. Vooral gericht op de cryptogrammen van de American Cryptogram Association (ACA).

27. *G. J. van Ojen jr. (1978). Leven en werken van Henri Koot (1883–1959). Sectie Krijgsgeschiedenis Koninklijke Landmacht, 's-Gravenhage.*

Een biografie van Henri Koot, die wordt beschouwd als de grootste Nederlandse geheimschriftkundige.

28. *J.A. Reeds, W. Diffie and J.V. Field, editors (2015). Breaking Teleprinter Ciphers at Bletchley Park - General Report on Tunny with Emphasis on Statistical Methods (1945). John Wiley & Sons, Inc.*

In het jaar 2000 werd bekend dat het oorspronkelijke rapport van I.J. Good, D. Michie en G. Timms, getiteld *General Report on Tunny with Emphasis on Statistical Methods (1945)* zou worden gedeclasificeerd. Tunny is de codenaam die in Bletchley Park was gegeven aan de Lorenz Schlüsselzusatz.

Voor de geschiedenis van die periode kan het verschijnen van deze editie worden beschouwd als een absoluut hoogtepunt.

29. *R. A. Rueppel (1986). Analysis and Design of Stream Ciphers. Springer-Verlag.*

Een van de weinige boeken dat volledig over schuifregisters gaat. Al tamelijk oud waardoor vele latere bedachte schuifregistercombinaties er niet in voorkomen.

30. *General Luigi Sacco (1977). Manual of Cryptography (reprint from: Manuale di Crittografia). Aegean Park Press.*

Om historische redenen van belang. Geschreven door de Italiaanse generaal Luigi Sacco die zelf actief cryptoanalist is geweest.

31. *B. Schneier (1996). Applied Cryptography: Protocols, Algorithms, and Source Code in C. second edition John Wiley & Sons, Inc.*

Naslagwerk voor een groot aantal cryptosystemen. Bevat voor veel systemen een implementatie in de computertaal C. De bijbehorende disk met programmatuur mocht lange tijd niet buiten de Verenigde Staten en Canada worden verspreid, maar is sindsdien op tal van servers daarbuiten te vinden.

32. *S. Singh (1999). The Code Book, The Science of Secrecy from Ancient Egypt to Quantum Cryptography. Fourth Estate, London.*

Een populair geworden boek. In een achttal items – van het door Walsingham gebroken geheimschrift van koningin Mary van Schotland tot de quantumcryptografie – wordt een aantal cryptosystemen beschreven. Sluit af met tien in moeilijkheid opklimmende opgaven. Hoe deze over een periode van bijna een jaar werden opgelost door een groep Zweden, die de beloning van £10.000 in de wacht sleepten, is te vinden op het internet: *How we Cracked the Code Book Ciphers*, Fredrik Almgren e.a.

33. *M. Smith (2000). The emperor's codes, The breaking of Japan's secret ciphers. Arcade Publishing, New York.*

De geschiedenis van het breken van de Japanse codes in de Tweede Wereldoorlog door de geallieerden.

34. *W. Stallings (1999). Cryptography and Network Security. second edition Prentice-Hall.*

Dit boek is toegespitst op de toepassing van cryptografie in de beveiliging van computernetwerken. Een hot topic dat nog lang belangrijk zal blijken.

35. *D. R. Stinson (1995). Cryptography, Theory and Practice. CRC Press, Inc..*

Een boek over de moderne cryptografie, geschreven vanuit een nogal wiskundige inslag.

36. *Wenbo Mao (2004). Modern Cryptography, Theory & Practice. Hewlet-Packard Company & Prentice-Hall, Inc..*

Dit boek wil zich meer op cryptografie in de praktijk dan op cryptografie in theorie richten. Behandeld worden cryptografische schemas, protocollen en systemen met hun sterkte en zwakte. Uit het voorwoord: "It reveals the general unfitness of *textbook crypto* for the real world by demonstrating numerous attacks on such schemes, protocols and systems under various real-world application scenarios."

37. *R. E. Weber (1979). United States Diplomatic Codes and Ciphers 1775–1938. Chicago: Precedent Publ. Inc..*

Dit boek geeft vele codelijsten van Amerikaanse diplomatieke codes, zodat men kan zien wat zulke codes zoal aan woorden en uitdrukkingen bevatten.

1.10 Referenties

- [1] S. Levy (2000). *Crypto, secrecy and privacy in the new code war*. Allen Lane The Penguin Press.

- [2] H. Buhrman, N. Chandran, S. Fehr, R. Gelles and V. Goyal et al. (2011). Position-Based Quantum Cryptography: Impossibility and Constructions. . <http://www.lanl.arxiv.org>
- [3] A. Messiah (1964). *Quantum Mechanics*, volume 1. North-Holland Publishing Company, Amsterdam.
- [4] Aug. Kerckhoffs (1883). *La Cryptographie Militaire ou Des Chiffres usités en temps de Guerre*. Librairie Militaire de L. Baudoin et Ce, Paris.
- [5] S. S. Snyder (1980). Computer Advances Pioneered by Cryptologic Organizations. *Annals of the History of Computing*, 2(1), 60–70.
- [6] H. Rohrbach (1979). Report on the Decipherment of the American Strip Cipher 0-2 by the German Foreign Office (Marburg 1945). *Cryptologia*, III(1), 16–26.
- [7] D. Kahn (1991a). *Seizing the Enigma, The Race to Break the German U-Boat Codes, 1939–1943*. Houghton Mifflin Company, Boston.
- [8] D. Kahn (1967). *The Codebreakers, The Story of Secret Writing*. MacMillan Publ. Co., Inc. In hoofdstuk 17. The Scrutable Orientals.
- [9] F. H. Hinsley and A. Stripp Eds. (1993). *Codebreakers, The inside story of Bletchley Park*. Oxford University Press. p. 114, Herinneringen van Joan Murray.
- [10] M. Dworkin (2001). Recommendation for Block Cipher Modes of Operation. NIST Special Publication 800-38A. Technical Report National Institute of Standards and Technology.

2 Transpositie

2.1	Transpositie en substitutie	60
2.2	Routetranspositie	61
2.3	Kolomtranspositie	64
2.3.1	Gevuld transpositieblok	64
2.3.2	Onthouden transpositiesleutel	65
2.3.3	Gedeeltelijk gevuld transpositieblok	66
2.3.4	Onregelmatig transpositieblok	68
2.4	Dubbele transpositie	70
2.4.1	Nihilist-transpositie	72
2.5	Roostertranspositie	72
2.5.1	Turning grille	73
2.6	Referenties	76

2.1 Transpositie en substitutie

In dit en de volgende twee hoofdstukken wordt een aantal van de meest bekende systemen uit de klassieke cryptografie besproken. Dat is overigens slechts een beperkte selectie uit de vele systemen die in de loop der tijden zijn bedacht. Voor de plaats in de geschiedenis van deze systemen kan het magistrale boek *The Codebreakers* van David Kahn van harte worden aanbevolen^[1]. Er zijn legio auteurs die hebben geschreven over klassieke geheimschriften. Een van de meest bekende is William Friedman, de *vader van de Amerikaanse cryptowetenschap*. Een willekeurige greep uit het leger van auteurs: Lange en Soudart, de Italiaanse generaal Luigi Sacco, het Amerikaanse Ministerie van Defensie en de Duitse auteur Friedrich Bauer^[2].

Klassieke geheimschriften kunnen worden ingedeeld in twee hoofdsoorten. In de *transpositie* worden de elementen van de klaartekst van plaats verwisseld, maar met de elementen zelf gebeurt niets. Daarentegen laat een *substitutie* de plaats in de tekst ongemoeid, maar worden de elementen systematisch vervangen door andere. Er zijn ook geheimschriften waarin transpositie en substitutie worden gecombineerd. In veel gevallen gebruikt men het gewone alfabet en dan nog alleen de hoofdletters A-Z*. Als daarvan wordt afgeweken zijn het veelal systemen die met cijfers werken.

De variatie in transpositiegeheimschriften is niet erg groot, veel minder in elk geval dan het aantal mogelijkheden voor substitutie. De *transpositiesleutel* is een rij getallen die de volgorde van de te verwisselen elementen aangeeft. Een praktijkprobleem is hoe die rij getallen op een gemakkelijke manier te onthouden en te communiceren. De verschillende transpositiesystemen zijn in feite slechts schema's waarmee een permutatie wordt vastgelegd. Omdat verwisselen twee of meer elementen vergt, is bij transposities altijd sprake van een zogenaamd *transpositieblok*. Dit blok bevat de te verwisselen elementen en is in klassieke geheimschriften doorgaans een rechthoek of vierkant, waarin de klaartekstelementen worden

* Voor oude geheimschriften moet rekening gehouden worden met het feit dat ons 26-letter alfabet betrekkelijk jong is. In vroeger tijden waren I, J en U, V nog niet gedifferentieerd en telde het alfabet 24 letters.

gerangschikt.

De mogelijkheden tot variatie zijn in substitutiegeheimschriften zeer veel groter. Zowel substitutie binnen het alfabet A–Z als substitutie naar een ander alfabet (cijfers, speciale tekens of combinaties daarvan) is mogelijk. De elementen kunnen stuk voor stuk worden vervangen of met twee of meer tegelijk en zelfs in wisselende aantallen. Handmatige systemen zijn gewoonlijk tamelijk eenvoudig van aard, maar er zijn ook behoorlijk ingewikkelde systemen bij. Met machines kunnen veel gemakkelijker ingewikkelde substituties worden gerealiseerd.

Behalve vaste vervanging kan met wisselende vervanging worden gewerkt. Dat kan een periodiek veranderend vervangingspatroon zijn, maar men kan ook verschillende vervangingspatronen elkaar laten afwisselen. Nog weer iets anders is het vervangen van een symbool door meerdere elementen om die vervolgens uit elkaar te halen. Zelfs kunnen die gescheiden elementen daarna weer in andere samenstelling worden gecombineerd. Het aantal mogelijkheden is legio en er is grote variatie in sterkte: van kinderlijk eenvoudig tot principieel onbreekbaar.

In de praktijk zal niet altijd worden volstaan met een enkelvoudige vercijfering, maar worden soms verschillende systemen na elkaar toegepast. Het achtereenvolgens toepassen van verschillende soorten geheimschrift moet worden onderscheiden van een combinatiesysteem, waarin de verschillende cryptoperaties samen één systeem vormen. Men spreekt van *oververcijfering* of *supervercijfering* als een cryptogram met een aanvullende vercijfering – vaak een eenvoudiger systeem – wordt versterkt. Het is dan zaak een zodanige combinatie van systemen te kiezen dat ook inderdaad een versterking wordt bereikt, want dit is niet zonder meer het geval. In dit verband mag worden verwezen naar de behandeling van het onderwerp commutatie in [hoofdstuk 1 \(pagina 40\)](#).

2.2 Routetranspositie

De routetranspositie is de eenvoudigste vorm van transpositie. Het transpositieblok is in de hier beschre-

ven routes steeds een vierkant, maar dat is geen must. De elementen van de klaartekst worden in het transpositieblok ingeschreven en daaruit langs een andere weg uitgenomen. In het voorbeeld van [figuur 2.1](#) is de tekst MAKKERS STAAKT UW WILD GERAAS van links naar rechts en van boven naar beneden geplaatst in een 5x5 transpositieblok. Het cryptogram is uit het transpositieblok genomen door de ingeschreven letters in kolommen van links naar rechts af te lezen en uit te schrijven in groepen van vijf letters*.

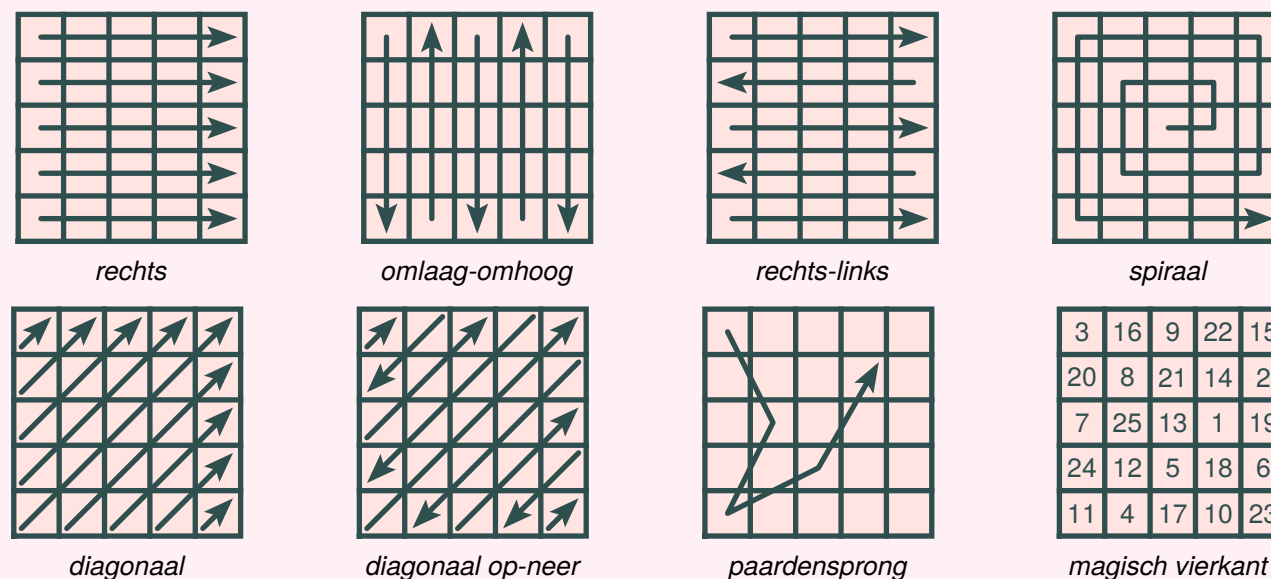
In het voorbeeld telt de tekst precies het benodigd aantal van 25 letters. Een tekst korter dan een compleet blok moet worden aangevuld met dummy letters, ook wel nullen genoemd omdat ze niets betekenen. De verleiding om dit met X'n te doen is groot, maar dat geeft aan een cryptoanalist meer houvast dan nodig. Opvullen met een willekeurige keuze uit de meest voorkomende letters is het beste. Wanneer de te vercijferen tekst langer is dan een blok, gebruikt men net zo veel blokken als nodig.



Figuur 2.1 Routetranspositie

* Cryptogrammen werden vroeger vaak via een publiek telegraafkantoor in Morse-code verstuurd. Om het de telegrafisten niet te moeilijk te maken en vergissingen te voorkomen werd het cryptogram in groepjes van 5 tekens verdeeld. De gründliche Duitsers verdeelden in de Tweede Wereldoorlog hun Enigma-telegrammen zelfs in groepen van 4 letters. Voor het gekozen aantal van 5 moet men bedenken dat het korte termijn geheugen van een mens veelal niet meer dan 6 tot 7 items kan bevatten.

Behalve de in het voorbeeld gekozen route zijn nog andere routes mogelijk. Zoals uitlezen van beneden naar boven of van rechts naar links, maar ook afwisselend van richting dan wel diagonaal in plaats van horizontaal/verticaal of zelfs spiraalsgewijs. Meer exotische routes kunnen worden gebaseerd op het magisch vierkant of de paardensprong uit het schaakspel. Het is niet nodig om de variatie in de routekeuze te beperken tot het uitlezen van het cryptogram, het inschrijven kan eveneens worden gevarieerd. **Figuur 2.2** geeft een paar van de mogelijkheden. William Friedman heeft een manier bedacht om routes systematisch te noteren en zodoende een wiskundige oplossing mogelijk te maken^[3].



Figuur 2.2 Variaties in routetranspositie

Cryptoanalyse van deze routetransposities berust op het betrekkelijk geringe aantal systematische routes. Het is daarom vrij eenvoudig alle in aanmerking komende routes te proberen. Zelfs een combinatie door inschrijven met de ene en uitlezen met een andere route vormt geen onoverkomelijke complicatie.

Ondanks – of misschien juist wel dankzij – de eenvoud is de routetranspositie overigens wel een vaak gebruikt middel om andere geheimschriften met een oververcijfering te beschermen.

2.3 Kolomtranspositie

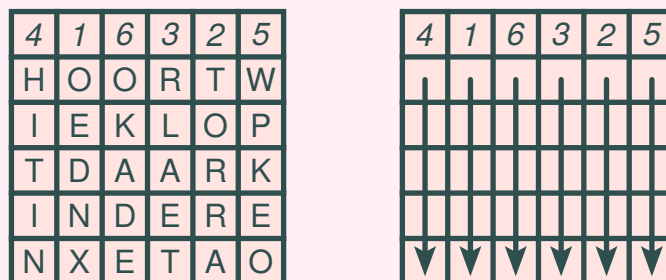
2.3.1 Gevuld transpositieblok

Waar de route bij de zojuist behandelde transpositie geheel vastgelegd is en het aantal routes (= sleutels) derhalve beperkt, introduceert de *kolomtranspositie* meer vrijheid in de sleutelkeuze. Het transpositieblok is nu in het algemeen een rechthoek, waarin de klaartekst op de gebruikelijke wijze wordt ingeschreven. Het cryptogram wordt gemaakt door de kolommen van de rechthoek in een door de sleutel bepaalde volgorde uit te lezen.

In de eenvoudigste vorm van de kolomtranspositie bestaat het transpositieblok uit geheel gevulde kolommen. **Figuur 2.3** geeft als voorbeeld de tekst HOORT WIE KLOPT DAAR KINDEREN ingeschreven in een transpositieblok van 6 kolommen en 5 rijen. Er zijn minder letters dan posities in het transpositieblok, daarom zijn in het voorbeeld de laatste vijf posities op de onderste rij met willekeurige letters opgevuld. Men spreekt van een *kolomtranspositie met kolommen van gelijke lengte* in tegenstelling tot de hierna te behandelen vorm met kolommen van ongelijke lengte.

De sleutel is de volgorde waarin het uitlezen van de kolommen geschiedt. In het voorbeeld zijn er 6 kolommen en derhalve $6! = 720$ sleutels. Hier is als volgorde en dus als sleutel de permutatie 416325 gekozen. Om te ontcijferen moet het cryptogram worden opgedeeld in hetzelfde aantal kolommen. Deze worden in de juiste volgorde in het transpositieblok geplaatst, zoals aangegeven door de sleutel, waarna het bericht kan worden gelezen.

klaartekst : HOORT WIE KLOPT DAAR KINDEREN
 cryptogram : OEDNX TORRA RLAET HITIN WPKEO OKADE



Figuur 2.3 Eenvoudige kolomtranspositie

2.3.2 Onthouden transpositiesleutel

Het onthouden van een transpositiesleutel komt neer op het onthouden van een tamelijk willekeurige rij getallen. Dat is niet eenvoudig* en het is daarom niet verwonderlijk dat men naar wegen heeft gezocht om zo'n rij gemakkelijker te kunnen memoriseren. Aangezien woorden en zinnen beter in het geheugen blijven hangen dan een willekeurige rij getallen, is een schema bedacht om zo'n rij af te leiden uit een stuk tekst.

Hiertoe gaat men als volgt te werk. De sleuteltekst wordt opgeschreven en de getallen 1, 2, 3 ... worden achtereenvolgens toegekend aan de letters in de tekst op volgorde van hun plaats in het alfabet. Twee voorbeelden om de procedure te verduidelijken. In het eerste voorbeeld wordt van het gekozen sleutelwoord MASKER stap voor stap elke letter vervangen door een getal. Hieruit ontstaat de in [figuur 2.3](#)

* Pincodes in het betalingsverkeer bestaan niet voor niets uit slechts vier cijfers: het maximum waarvoor men aanneemt dat iedereen die wel zal kunnen onthouden. En natuurlijk zijn dat er eigenlijk te weinig!

gekozen permutatie.

M	A	S	K	E	R
M	1	S	K	E	R
M	1	S	K	2	R
M	1	S	3	2	R
4	1	S	3	2	R
4	1	S	3	2	5
4	1	6	3	2	5

sleutel = MASKER

A	M	S	T	E	R	D	A	M
1	M	S	T	E	R	D	A	M
1	M	S	T	E	R	D	2	M
1	M	S	T	E	R	3	2	M
...								
...								
1	5	8	9	4	7	3	2	6

sleutel = AMTERDAM

Wat te doen als een letter meerdere keren in de sleutel voorkomt? In dat geval wordt deze van links naar rechts afgewerkt en pas daarna komt de volgende letter aan de beurt*. Een voorbeeld is de hierboven gegeven constructie met sleutel AMSTERDAM.

Voor de cryptoanalist is het de kunst om te proberen aan de hand van een dergelijke sleutel uit te vissen welk sleutelwoord gebruikt is. Dat kan zijn nut hebben als veel cryptogrammen moeten worden geanalyseerd en als de correspondenten een beperkte fantasie blijken te hebben in het bedenken van sleutels. Het raden van goede sleutels gaat dan tot de mogelijkheden behoren. De reconstructie is gemakkelijker naarmate de sleutel langer is^[4].

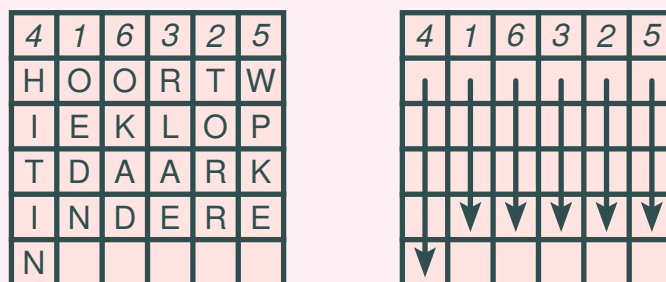
2.3.3 Gedeeltelijk gevuld transpositieblok

Het oplossen van een kolomtranspositie met kolommen van gelijke lengte is betrekkelijk eenvoudig, juist vanwege die gelijke lengte. Wanneer een cryptogram in kolommen is opgedeeld, moet men deze door

* Dit is de gebruikelijke procedure, maar er is natuurlijk niets op tegen om het anders af te spreken. Bijvoorbeeld werken van rechts naar links. Van belang is slechts dat de correspondenten van elkaar weten hoe de sleutel moet worden gemaakt.

anagrammeren in de goede volgorde plaatsen. Daartoe moet eerst het aantal rijen en kolommen van het transpositieblok worden bepaald. Het aantal mogelijkheden in geval van een kolomtranspositie met kolommen van gelijke lengte is niet groot, omdat het product van het aantal rijen en het aantal kolommen precies gelijk moet zijn aan het aantal letters. Daarom is het beter om het transpositieblok niet geheel uit te vullen waardoor kolommen van verschillende lengte ontstaan. Men spreekt van een *kolomtranspositie met kolommen van ongelijke lengte*. De *lange* kolommen tellen één letter meer dan de *korte* kolommen. De verdeling over kort en lang hangt af van het aantal open plaatsen aan het eind van het transpositieblok.

klaartekst : HOORT WIE KLOPT DAAR KINDEREN
cryptogram : OEDNT ORRRL AEHIT INWPK EOKAD



Figuur 2.4 Kolomtranspositie met ongelijke kolommen

Met hetzelfde voorbeeld als in [figuur 2.3](#) maar met achterwege laten van het uitvullen aan het eind, ontstaat het cryptogram van [figuur 2.4](#). Het bevat 1 lange en 5 korte kolommen. Als de cryptoanalist een opdeling van 6 kolommen en 5 rijen uitprobeert, moet hij ook nog de plaats van de lange kolom bepalen. In dit geval moeten 6 mogelijkheden worden onderzocht. De onzekerheid over de plaats van de lange kolommen maakt het opdelen van de cryptogrammen in kolommen een stuk lastiger. Bij n kolommen waarvan k lang geeft de binomiaalcoëfficiënt $\binom{n}{k}$ het aantal mogelijkheden*.

Het ontcijferen met behulp van de sleutel gaat op dezelfde manier als bij de transpositie met kolommen van gelijke lengte. Ook hier moet het cryptogram eerst in kolommen worden opgedeeld. Aangezien de sleutel bekend is, kan degene die ontcijfert zonder problemen vaststellen wat de lange en wat de korte kolommen zijn en zo het cryptogram op de goede plaatsen splitsen. Van de bovengenoemde onzekerheid is dan natuurlijk geen sprake. Na afsplitsing worden de kolommen in de juiste volgorde in het transpositiesblok geplaatst.

Deze vorm van transpositie is zelfs nog vrij recent in gebruik geweest. In de Eerste Wereldoorlog werkten de Duitsers er mee en toen de Abwehr (de Duitse spionagedienst) in de Tweede Wereldoorlog haastig spionagenetwerken in Zuid-Amerika moest opbouwen, greep ze aanvankelijk naar deze transposities^[5].

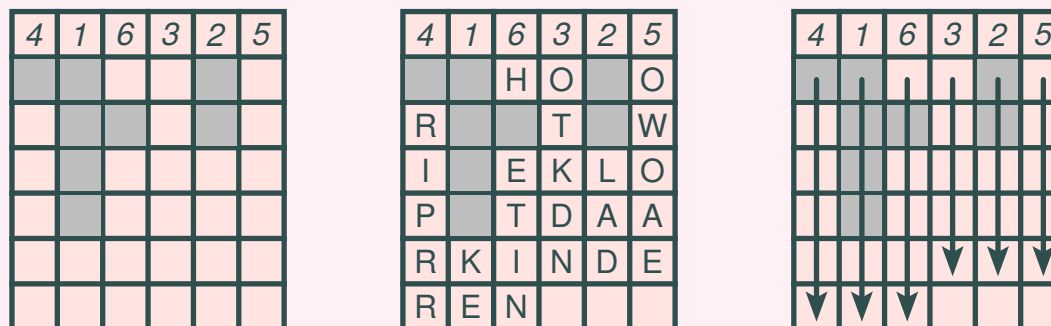
2.3.4 Onregelmatig transpositiesblok

Verdere versterking van de kolomtranspositie kan worden bereikt door de kolommen nog onregelmatiger van lengte te maken. Zo'n complicatie wordt gevonden in het Japanse K10-systeem uit de tijd rond 1940^[6]. Hier maakt men de lengte van de kolommen onregelmatig door in het transpositiesblok bepaalde plaatsen niet te gebruiken. De kolommen gaan daardoor in lengte verschillen en behoeven zelfs niet ononderbroken door te lopen. **Figuur 2.5** geeft een voorbeeld van deze variant.

Onze landgenoot Nuboer (1901–1984), gelegeerd in het toenmalige Nederlands Oost-Indië, brak in 1935 de variant van dit geheimschrift in gebruik bij de Keizerlijke Japanse Marine. Die gebruikte verscheidene codes welke in de telegrammen werden aangeduid met het laatste symbool op elke regel van het telegram. Zoals de Ni-code (Ni is een Japans kana-symbool) en de '.' code; de laatste naar alle waarschijnlijkheid dezelfde code die door de Amerikanen Blue Code was gedoopt. De codetelegrammen kregen een oververcijfering met een onregelmatige transpositie. In zijn bachelorscriptie beschrijft Joost Rijnveld

* Binomiaalcoëfficiënt $\binom{n}{k} = \frac{n!}{k!(n-k)!}$

klaartekst : HOORT WIE KLOPT DAAR KINDEREN
 cryptogram : KELAD OTKDN RIPRR OWOAE HETIN

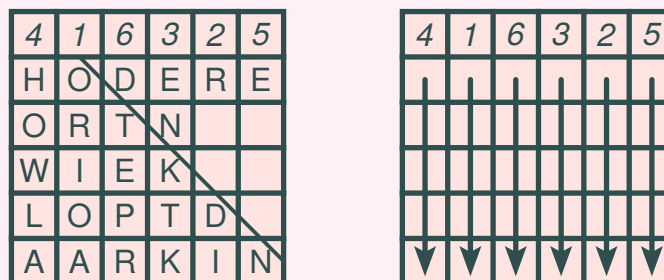


Figuur 2.5 Japanse K10 transpositie

aan de hand van de aantekeningen van Nuboer hoe deze dit geheimschrift wist te breken^[7]. Ontcijfering werd extra moeilijk gemaakt doordat een aanzienlijk aantal van de beschikbare posities in [figuur 2.5](#) met willekeurige symbolen werden gevuld.

Een andere mogelijkheid is om het transpositieblok te verdelen in verscheidene (onregelmatig gevormde) gebieden. De klartekst wordt per gebied ingeschreven, de afname geschiedt op de gebruikelijke manier van een kolomtranspositie. Een voorbeeld is gegeven in [figuur 2.6](#). Eerst wordt het gedeelte links van de streeplijn gevuld met klartekst, daarna gaan de resterende letters in het deel rechts, waarna de kolommen worden afgenomen in de aangegeven volgorde. Bij het ontcijferen kan er geen verwarring ontstaan omtrent de lege posities: de onge vulde hokjes zijn de laatste vijf van het laatst te vullen deel. Een systeem als dit maakte deel uit van het geheimschrift dat de Russische KGB in de vijftiger jaren van de 20^e-eeuw meegaf aan de spion Reino Hayhanen^[8].

klartaekst : HOORT WIE KLOPT DAAR KINDEREN
 cryptogram : ORIOA RDIEN KTKHO WLAEN DTEPR



Figuur 2.6 Kolomtranspositie met onregelmatigheden

2.4 Dubbele transpositie

De gewone kolomtransposities, met of zonder uitvullen, zijn niet bijster sterk. Aanzienlijk beter wordt het wanneer de transpositie tweemaal wordt uitgevoerd. Vanwege het gebruik als officieel geheimschrift door het Amerikaanse leger in het begin van de 19^e-eeuw ook wel de *US Army Double Transposition* genoemd. Overigens zijn de Amerikanen niet de enigen die met dit geheimschrift hebben gewerkt. De Duitsers maakten er in de Eerste Wereldoorlog eveneens gebruik van, maar toch – hoewel het als een sterk geheimschrift geldt – werden de berichten veelvuldig door Franse cryptoanalisten gebroken. In de Tweede Wereldoorlog komt men de dubbele transpositie weer tegen bij de radiocommunicatie door agenten van de Special Operations Executive (SOE) vanuit bezet gebied in Europa^[9]. Dat heeft het dramatisch verlopen Englandspeel opgeleverd, waardoor veel geparachuteerde Nederlandse agenten en verzetslieden het leven hebben gelaten*^[10].

* De dubbele transpositie werd gebruikt door Lt. H.M.G. Lauwers, een gedropte en gevangen genomen geheim agent. Schreieder

klaartekst : HOORT WIE KLOPT DAAR KINDEREN
 tussenstap : OEDNT ORRL AEHIT INWPK EOKAD
 cryptogram : OHKER IKORT OTEPD LNARI ENAWD

4	1	6	3	2	5
H	O	O	R	T	W
I	E	K	L	O	P
T	D	A	A	R	K
I	N	D	E	R	E
N					

klaartekst

4	1	6	3	2	5
H	O	O	R	T	W
I	E	K	L	O	P
T	D	A	A	R	K
I	N	D	E	R	E
N					

na 1e transpositie

4	1	6	3	2	5
H	O	O	R	T	W
I	E	K	L	O	P
T	D	A	A	R	K
I	N	D	E	R	E
N					

2e transpositie

Figuur 2.7 Dubbele transpositie

De vercijfering in dit systeem gaat als volgt. Het resultaat van de eerste kolomtranspositie wordt opnieuw in een transpositieblok geplaatst. Daarna wordt weer een kolomtranspositie uitgevoerd. Omdat de kolommen van de eerste transpositie horizontaal ingeschreven zijn en vervolgens verticaal zijn uitgenomen, is de verwisseling bijzonder grondig. Voor de tweede fase kan beter niet dezelfde sleutel gebruikt worden als voor de eerste fase. Het is veel veiliger om de afzonderlijke kolomtransposities elk een eigen sleutel te geven. In het voorbeeld van [figuur 2.7](#) is dit dan ook gedaan: de twee deeltransposities gebruiken zelfs sleutels van verschillende lengte.

Er zijn een paar dingen waarvoor men moet oppassen. Het eerste is het gevaar dat – in de hitte van de strijd – de tweede transpositie vergeten wordt. Het cryptogram is dan slechts een enkelvoudige kolom-

en Giskes, resp. hoofd van de Sicherheitsdienst en van de Abwehr hier te lande, waren aan de Duitse kant de hoofdrolspelers in het Englandspiel. Ook na de oorlog is een waas van geheimzinnigheid om het Englandspiel blijven hangen, temeer omdat delen van de archieven bleken te zijn vernietigd.

transpositie met ongelijke kolommen. Als die wordt opgelost beschikt de cryptoanalist over de eerste sleutel. Het tweede gevaar schuilt in het veelvuldig gebruik. Wanneer veel berichten met dezelfde twee sleutels zijn gemaakt, zullen er bijna altijd wel enkele met toevallig dezelfde lengte zijn. De cryptoanalist zoekt die bijeen en als er genoeg van verzameld zijn kunnen ze worden opgelost.

2.4.1 Nihilist-transpositie

Een sterk vereenvoudigde vorm van de dubbele transpositie is de zogenaamde *Nihilist transpositie*. Het transpositieblok in deze variant bestaat uit een geheel gevuld vierkant, waarvan de rijen en kolommen met dezelfde sleutel worden verwisseld. Als voorbeeld weer dezelfde klaartekst en het woord NIHIL als sleutel.

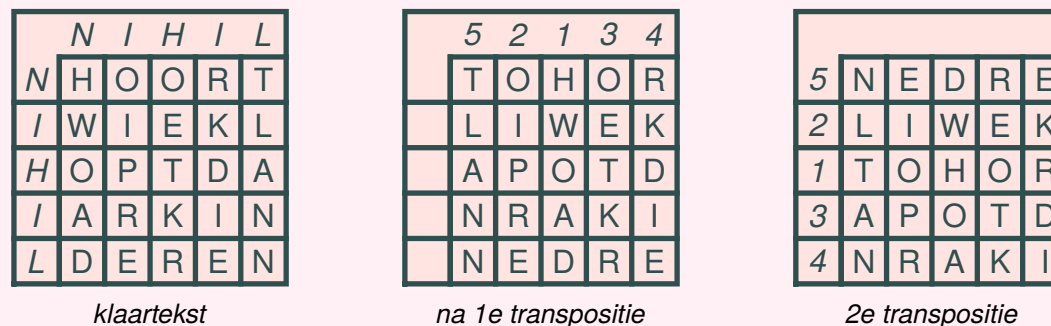
Anders dan in het vorige voorbeeld is hier een andere manier van inschrijven van de klaartekst en uitnemen van het cryptogram gebruikt. Hiervoor werden de letters van links naar rechts ingeschreven en de kolommen in de volgorde van de sleutel uitgenomen. Bij de Nihilist blijkt het handiger om de letters in de volgorde van de sleutel in te schrijven. Zo is het middelste vierkant van [figuur 2.8](#) gevuld. De eerste letter H komt op de eerste regel in de derde kolom onder sleutelgetal 1, de tweede letter O onder de 2 in de tweede kolom, enz. Het cryptogram wordt daarna gevormd door de rijen achtereenvolgens uit te nemen (rechts in de figuur).

Dit geheimschrift is genoemd naar de beweging van de Nihilisten, een groep anarchistische opposanten tegen het Russische tsaristische regime uit de tweede helft van de 19^e-eeuw. In tegenstelling tot de echte dubbele transpositie is dit een zwak geheimschrift.

2.5 Roostertranspositie

Al in een ver verleden maakte men gebruik van een geheimschrift dat eigenlijk meer als steganografie

klaartekst : HOORT WIE KLOPT DAAR KINDEREN
 cryptogram : NEDRE LIWEK TOHOR APOTD NRAKI



Figuur 2.8 Nihilist transpositie

dan als cryptografie moet worden betiteld. De vercijfering vindt plaats door middel van een rooster, dat wil zeggen een stuk papier of een plaat met uitsparingen. De tekst van het bericht wordt in de uitsparingen geschreven, waarna het rooster wordt weggehaald en de omringende ruimte zó wordt opgevuld dat een onschuldig lijkende tekst ontstaat. Dit geheimschrift is voor het eerst beschreven door de Italiaanse geleerde Girolamo Cardano (1501–1576)^[11]. Hij vermeldde er nog bij dat het bericht enkele malen moest worden overgeschreven "om onregelmatigheden in het handschrift te voorkomen, onregelmatigheden die de boodschap kunnen verraden." In [figuur 2.9](#) is de tekst *stuur meer soldaten naar Amsterdam* verborgen.

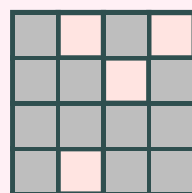
2.5.1 Turning grille

Een latere vorm van het rooster is de zogenaamde Fleissner-grille, genoemd naar de Oostenrijkse cryptoloog Fleissner von Wostrowitz^[12]. Dit zijn roosters met een vierkant transpositieblok. De uitsparingen moeten zo gekozen zijn dat na viermaal een kwartslag draaien alle posities in het transpositieblok precies

Morgen **stuur** ik Karel naar je toe. Hij heeft behoefte aan wat **meer** frisse lucht. Ik geef hem een stel tinnen **soldaten** mee voor je verzameling. Ga jij nog **naar** oom Hans in **Amsterdam** ?

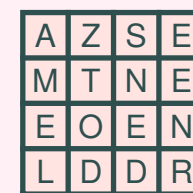
Figuur 2.9 Rooster volgens Cardano

één keer aan de beurt zijn gekomen. In elk van de vier posities van het rooster wordt in de uitsparingen een deel van het bericht geschreven. Na voltooiing wordt het cryptogram van links naar rechts en van boven naar beneden uit het transpositieblok gehaald. Voor het ontcijferen van het bericht wordt dit op de normale manier in het transpositieblok geschreven en door het opgelegde rooster heen wordt de loop van de tekst zichtbaar. Vanwege het draaien van het rooster wordt dit geheimschrift een *turning grille* genoemd. Het voorbeeld in **figuur 2.10** laat zien hoe het werkt.

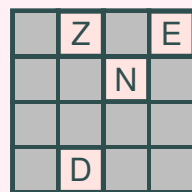


rooster

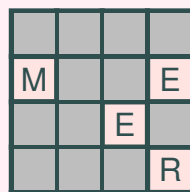
klaartekst : ZEND MEER SOLDATEN
cryptogram : AZSEM TNEEO ENLDD R



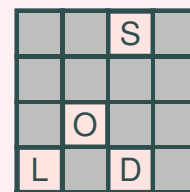
cryptogram



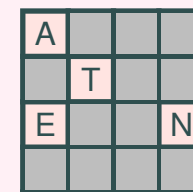
0°



90°



180°



270°

Figuur 2.10 Turning grille

Het is niet moeilijk om een turning grille te construeren. Daartoe wordt het transpositieblok in vier kwadranten verdeeld. Voor elk van de vakjes in een kwadrant wordt uit de vier kwadranten een van de overeenkomstige posities gekozen. Zo leidt men ook gemakkelijk af dat een $n \times n$ -rooster op $4^{n \times n/4}$ manieren kan worden gemaakt. Dit aantal wordt een factor 4 kleiner als niet vastgelegd hoeft te worden met welke van de vier standen moet worden begonnen. De eerste stand kan immers willekeurig worden gekozen.

Hoewel het boek van Fleissner von Wostrowitz uit 1881 altijd wordt aangehaald als bron voor de turning grille, is dit niet de vroegste plaats waar het wordt beschreven. Door ene Hindenburg werd er al bijna honderd jaar eerder in een wiskundig tijdschrift over gepubliceerd^[13]. Hindenburg laat daarin zien hoe een rooster kan worden geconstrueerd en berekent ook het aantal mogelijke roosters. De beschrijving wekt sterk de indruk dat het gaat om een systeem dat daadwerkelijk in gebruik is. Heel aardig is de manier waarop Hindenburg elk mogelijk rooster van een volgnummer voorziet en dan aangeeft hoe datzelfde volgnummer weer kan dienen voor de constructie van het bijbehorende rooster.

Hij redeneert ongeveer als volgt. Verdeel het rooster overlangs en overdwars in tweeën, waardoor vier kwadranten ontstaan. Elke positie in het rooster kan nu door draaien rond het middelpunt worden teruggebracht tot een plaats in het eerste kwadrant linksboven. Omgekeerd kan men ook zeggen: neem een positie in het eerste kwadrant en bepaal vervolgens welke van de vier overeenkomstige posities – na respectievelijk 0, 1, 2 of 3 rotatiestappen – voor de uitsparing in het rooster in aanmerking komt. Dit getal kenmerkt de roosterpositie.

Als voorbeeld een uitwerking van het rooster in [figuur 2.10](#). Het is een 4×4 rooster, elk kwadrant telt 4 vakjes. Het getal dat het rooster karakteriseert telt evenveel cijfers als er vakjes zijn in één kwadrant. Begin in de linkerbovenhoek waar in het volledig ingevulde transpositieblok op deze plaats een A staat. Na een kwartslag rechtsom komt dit vak in de rechterbovenhoek terecht. Omdat één kwartslag draaien genoeg is om het gekozen vak op z'n plaats te brengen wordt hiervoor het cijfer 1 genoteerd. Neem nu het tweede vak op de bovenste rij met de Z. Dit is al open – dus nul keer draaien – en daarom wordt een 0 opgeschreven. Dan het eerste vak van de tweede rij waar een M staat. Drie rotatiestappen zijn nodig

voordat dit op de gekozen plaats belandt (het open vakje op de onderste rij) en daarom wordt een 3 genoteerd. Tenslotte heeft de T in het vierde en laatste vak van het eerste kwadrant een kwartslag nodig, dus weer een 1.

Achter elkaar gezet leggen de gevonden cijfers het gatenpatroon eenduidig vast. Voor het rooster van **figuur 2.10** is dat dus 1031. Andersom redenerend kan uit deze cijfercombinatie het gatenpatroon van het rooster worden afgeleid. Eigenlijk gebeurt er niets anders dan dat de roosters in plaats van in het gewone 10-tallig stelsel nu in het 4-tallig stelsel worden geteld. De hoogste 4-tallige waarde voor dit rooster is $3333_4 = 255_{10}$ en omdat 0000 ook meetelt, zijn er dus 256 van deze roosters*.

De turning grille maakt op verschillende momenten in de geschiedenis z'n entree. Het oudst bekende voorkomen is in een bericht uit de correspondentie van stadhouder Willem IV. Dit met een 16x16 rooster vercijferd bericht stamt uit 1745^[14]. In de Eerste Wereldoorlog gebruikten de Duitsers van eind 1916 tot begin 1917 korte tijd het turning grille, tot vreugde van de Franse cryptoanalisten die er wel raad mee wisten. Jules Verne beschrijft in zijn boek *Mathias Sandorf* een bericht vercijferd met een 6x6 rooster^[15]. Een interessante variant is gevonden in de correspondentie uit het jaar 1800 van Baron van Spaen met de latere koning Willem I^[16]. Het rooster is hier rechthoekig in plaats van vierkant. De vier posities worden niet door roteren bereikt, maar door te spiegelen langs een horizontale en/of verticale as; in de praktijk zal men dit bereiken door het rooster op te tillen en om te keren.

2.6 Referenties

- [1] D. Kahn (1967). *The Codebreakers, The Story of Secret Writing*. MacMillan Publ. Co., Inc. Er bestaat een tweede, iets gewijzigde editie van.
- [2] A. Lange and E. A. Soudart (1935). *Traité de Cryptographie*. Librairie Félix Alcan, Paris.
General Luigi Sacco (1977). *Manual of Cryptography (reprint from: Manuale di Crittografia)*. Aegean Park Press.

* Strikt genomen zijn er maar 64 verschillende roosters, omdat de allereerste rotatie er niet toe doet.

- Anon. (1944). Advanced Military Cryptography. Technical Report Washington: US Government Printing Office.
- F. L. Bauer (1997). *Decrypted Secrets, Methods and Maxims of Cryptology*. Springer-Verlag.
- [3] Capt. C.L.R. Lohr and W. F. Friedman (1918). Formulae for the Solution of Geometrical Transposition Ciphers. Technical Report Geneva, Ill.: Riverbank Laboratories, Dept. of Ciphers.
- [4] W. F. Friedman (1941). Military Cryptanalysis Part IV, Transposition and Fractionating Systems. Technical Report Washington: War Department, Office of the Chief Signal Officer Washington. p. 15–17.
- [5] J. F. Bratzel and L. B. Rout jr. (1983). Abwehr Ciphers in Latin America. *Cryptologia*, VII(2), 132–144.
- [6] D. Kahn (1991b). Pearl Harbor and the Inadequacy of Cryptanalysis. *Cryptologia*, XV(4), 273–312.
- [7] J. Rijneveld (2013). *How the Dutch broke the Japanese Blue Code in the late 1930s*. PhD thesis, Radboud University Nijmegen.
- [8] D. S. Vogel (1990). Inside a KGB cipher. *Cryptologia*, XIV(1), 37–52.
- D. Kahn (1983). *Kahn on Codes*. MacMillan Publ. Co., Inc.
- [9] L. Marks (1998). *Between Silk and Cyanide, A Codemaker's Story 1941–1945*. HarperCollins Publ..
- [10] J. Schreieder (1950). *Das war das Englandspiel*. München: Walter Stutz Verlag.
- H. J. Giskes (1949). *Abwehr IIIF. De Duitse contraspionage in Nederland*. Amsterdam: De Bezige Bij.
- [11] Girolamo Cardano (1550). *De Subtilitate libri XXI*. Paris, Nuremberg: Anon..
- [12] E. B. Fleissner von Wostrowitz (1881). *Handbuch der Kryptographie*. Wenen: NN.
- [13] C. F. Hindenburg (1796). Fragen eines Ungenannten über die Art durch Gitter geheim zu schreiben. *Archiv der reinen und angewandten Mathematik*, pp. III 347–351, V 81–99.
- [14] K. de Leeuw and H. van der Meer (1995). A Turning Grille from the Ancestral Castle of the Dutch Stadtholders. *Cryptologia*, XIX(2), 153–165.
- [15] W. F. Friedman (1940). Jules Vernes as Cryptographer. *The Signal Corps Bulletin*, 108.
- [16] K. de Leeuw and H. van der Meer (1993a). Een Roostergeheimschrift van Alexander Baron van Spaen (1775–1811). *De Achttiende Eeuw*, 25(2), 187–210.

3 Monoalfabetische systemen

3.1	Monoalfabetische substitutie	79
3.1.1	Caesar-substitutie	79
3.1.2	Eenvoudige monoalfabetische substitutie	81
3.1.3	Onthouden substitutiesleutel	82
3.2	Monoalfabetische varianten	84
3.2.1	Homofone substitutie	84
3.2.2	Monome-dinomesubstitutie	88
3.3	Digramsubstitutie	90
3.3.1	Playfair-substitutie	94
3.3.2	Twosquare-substitutie	96
3.4	Fractionerende systemen	97
3.4.1	Bifid of Delastelle-substitutie	98
3.4.2	ADFGX en ADFGVX	100
3.5	Nomenclatuur en code	102
3.5.1	Een- en tweedelige code	108
3.5.2	Vercijferde code	110
3.5.3	Bijzondere codes	112
3.6	Referenties	115

3.1 Monoalfabetische substitutie

3.1.1 Caesar-substitutie

In op substitutie gebaseerde systemen worden de elementen van de klaartekst vervangen door elementen uit een vervangingsalfabet. In de eenvoudigste vorm is dit een letter-voor-lettervervanging. Het vroegst bekende schema voor deze vervanging wordt toegeschreven aan Julius Caesar, vlak voor het begin van onze jaartelling. Volgens de Romeinse schrijver Suetonius verving Julius Caesar elke letter in het bericht door de letter drie plaatsen verder in het alfabet^{*}. Om deze reden wordt dit geheimschrift *Caesar-substitutie* genoemd. Een korte tekst is op deze wijze gecijferd.

```
klaartekst : ORDE LEIDT TOT ALLE DEUGDEN  
cryptogram : RUGH OHLGW WRW DOOH GHXJGHQ
```

Meer algemeen staat de Caesar-substitutie voor een vervanging door een substitutieletter die een vast aantal plaatsen verder in het alfabet staat. De vaste verschuiving kan voor een 26-letteralfabet variëren tussen 1 en 25 stappen[†]. De verschuiving geschiedt modulo 26, dat wil zeggen, bij het doortellen gaat men van de Z over op de A. Ontcijferen geschiedt door hetzelfde aantal letter terug te tellen. In formule, met P voor de klaartekstletter, K de verschuiving (sleutel), C de cryptogramletter en E , D respectievelijk de vercijfer- en ontcijferfunctie

$$C = E(P, K) = C^K(P) = P + K \bmod 26 \quad \{A...Z\} \leftrightarrow \{0...25\} \quad (3.1)$$

- ^{*} Zo wordt het tenminste altijd gebracht, maar uit de Latijnse tekst zou blijken dat de richting omgekeerd is: de gecijferde letter is dan degene die drie plaatsen voor de klaartekstletter staat. Niet onlogisch, omdat dit het ontcijferen eenvoudiger maakt. Doortellen in het alfabet is makkelijker dan terugtellen. Probeer zelf maar eens het alfabet van achter naar voren op te zeggen.
- [†] Een verschuiving van 0 is ook mogelijk, maar levert gewoon de oorspronkelijke tekst. Hier is dat niet erg zinvol, maar in later te behandelen systemen is het beter deze mogelijkheid niet weg te laten.

$$P = D(C, K) = C^{-K}(C) = C - K \bmod 26 \quad (3.2)$$

Alfabetsubstituties, zoals in bovenstaand voorbeeld, kunnen worden weergegeven door twee alfabetten onder elkaar te schrijven. In het als *klaaralfabet* gemarkeerde alfabet zoekt men de te vercijferen letter op en in het daaronder geplaatste *cijferalfabet* staat dan op de overeenkomstige plaats de vervangende letter.

```
klaaralfabet : ABCDEFGHIJKLMNOPQRSTUVWXYZ
cijferalfabet : DEFGHIJKLMNOPQRSTUVWXYZABC
```

In een *variant* van de oorspronkelijke Caesar-substitutie laat men het substitutie-alfabet in tegengestelde richting lopen. Vercijferen en ontcijferen gaat op dezelfde wijze, namelijk bij de klaartekstletter de overeenkomstige cryptogramletter opzoeken. De sleutel is wederom het aantal letters dat deze beide alfabetten ten opzichte van elkaar verschoven zijn. Dezelfde tekst wordt vercijferd met sleutel 22. Een paar voorbeelden: Q = 16 vercijfert naar $22 - 16 = 6 = G$ en Y = 24 naar $22 - 24 = -2 = 24 = Y$, volgens onderstaande vergelijking.

$$C = K - P \bmod 26 \text{ en } P = K - C \bmod 26 \quad (3.3)$$

```
klaaralfabet : ABCDEFGHIJKLMNOPQRSTUVWXYZ
cijferalfabet : WVUTSRQPONMLKJIHGFEDCBAZYX

klaartekst : ORDE LEIDT TOT ALLE DEUGDEN
cryptogram : IFTS LSOTD DID WLLS TSCQTSJ
```

Merk op dat nu alle encrypties hun eigen inverse zijn, waardoor $I \rightarrow O$ impliceert dat ook $O \rightarrow I$. Dit substitutieschema is dus een *involutie* (sectie 1.7.2). Een bijzonder geval zijn de letters L en Y die zichzelf als vercijfering hebben*.

Een meer algemene variant van de Caesar-substitutie is de *affine transformatie*. Hierin wordt de vercijfering uitgevoerd door een lineaire vergelijking toe te passen

$$C = a \times P + b \bmod 26 \quad 0 \leq a, b \leq 25 \quad \text{ggd}(a, 26) = 1 \quad (3.4)$$

De voorwaarde voor de grootste gemene deler betekent dat a en 26 relatief priem moeten zijn. Als gevolg daarvan wordt de inverteerbaarheid – de mogelijkheid tot ontcijferen – gegarandeerd. De gewone Caesar-substitutie is het bijzondere geval met $a = 1$.

De Caesar-substitutie is buitengewoon eenvoudig, vooral door de vaste verschuiving tussen de cryptogramletters en die van de klaartekst. Geef voor één letter de substitutie en alle andere zijn eveneens bekend. Het aantal bruikbare sleutels is ook maar klein, namelijk 25.

3.1.2 Eenvoudige monoalfabetische substitutie

Een stapje ingewikkelder dan de Caesar-substitutie is het om de vaste, normale volgorde van het alfabet los te laten bij de keuze van het cijferalfabet. Door een willekeurige permutatie van de letters van het alfabet te kiezen, neemt het aantal sleutels in een klap enorm toe. Namelijk van 26 naar $26! \approx 4 \times 10^{26}$. Een enorme toename! Hieronder is de tekst van het vorige voorbeeld opnieuw vercijferd, maar nu met een willekeurig gepermuteerd cijferalfabet.

```
klaaralfabet : ABCDEFGHIJKLMNOPQRSTUVWXYZ
cijferalfabet : YSNZMJXRQIFVDGWEPHULTOCAK

klaartekst : ORDE LEIDT TOT ALLE DEUGDEN
cryptogram : WHZM VMQZU UWU YVVM ZMLXZMG
```

- * Dit is geen zwakte in een systeem, integendeel. Het aantal mogelijke vercijferingen van een letter vermindert van 26 naar 25 door uit te sluiten dat één bepaalde letter tot zichzelf vercijferd worden; dat is in het voordeel van de cryptoanalist.

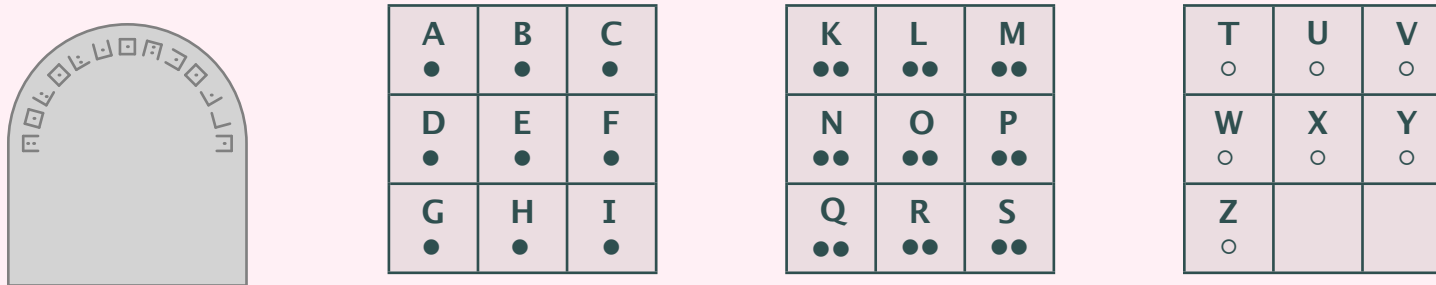
Door in het klaaralfabet van het substitutieschema de letters van A naar Z te laten lopen is het vercijferproces gemakkelijk uit te voeren. Om dit ook voor het ontcijferen te bewerkstelligen kan een invers paar gemaakt worden, waarin het cijferalfabet in de normale volgorde staat. In de navolgende substitutieschemas dient het bovenste paar voor het vercijferen en het onderste voor het ontcijferen (let op de positiewisseling tussen de beide alfabetten in het onderste schema).

```
klaara1fabet : ABCDEFGHIJKLMNOPQRSTUVWXYZ  
cijfera1fabet : YSNZMJXRQIFVDGWEPHBULTOCAK  
cijfera1fabet : ABCDEFGHIJKLMNOPQRSTUVWXYZ  
klaara1fabet : YSXMPKNRJFZUECWQIHBVTLOGAD
```

In de loop van de geschiedenis zijn veel monoalfabetische substituties gemaakt met een fantasiealfabet. De letters worden daarin door speciale symbolen voorgesteld. De gebruikte symbolen komen uit de astrologie, rune-tekenen, enz. Een van de bekendste voorbeelden is hier afgebeeld. De tekens zijn afgeleid van een spelletje dat bij ons bekend staat als melk-boter-kaas; in het Engels noemt men het wel tic-tac-toe-cipher. Vanwege het gebruik in vrijmetselaarskringen is het ook wel Vrijmetselaars- of Rozenkruisersgeheimsschrift genoemd. In het voorbeeld van [figuur 3.1](#) staat een tekst die kan worden aangetroffen op de grafsteen van vrijmetselaar James Leeson. Zijn graf bevindt zich op het Trinity Churchyard in New York, vlakbij Wall Street^[1].

3.1.3 Onthouden substitutiesleutel

Net als voor de transpositiesleutels op [pagina 65](#) is het ook bij substituties handig om een geheugensteuntje te hebben voor de sleutelpermutatie. Weer wordt een sleutelwoord of sleutelzin gebruikt voor de constructie. Elke letter van het sleutelwoord wordt bij eerste voorkomen neergeschreven, latere voorkomens moeten natuurlijk worden genegeerd. Dit staat in tegenstelling tot de procedure bij transpositie-



Figuur 3.1 REMEMBER DEATH – Vrijmetselaarsgeheimschrift op grafsteen

sleutels waar alle letters mee mogen doen. Als nog niet alle letters van het alfabet gebruikt zijn, wordt de sleutelpermutatie daarmee aangevuld. Uitgaande van de sleutelzin ZONDER GELUK VAART NIEMAND WEL loopt het proces als volgt. Eerst zijn alle letters eenmaal opgeschreven in de volgorde van optreden en dan volgt aanvulling met de resterende letters.

sleutelzin : ZONDER GELUK VAART NIEMAND WEL
 unieke letters : ZONDERGLUKVATIMW
 cijferalfabet : ZONDERGLUKVATIMWBCFHJPQSXY

Als permutatie is het resultaat vaak nog niet zo geweldig. Meestal komt het merendeel van de letters UVWXYZ achteraan te staan en een cryptoanalist maakt daar dan dankbaar gebruik van. Daarom worden gewoonlijk nog extra bewerkingen uitgevoerd. Populair is het uitvoeren van een aanvullende kolomtranspositie (met ongelijke kolommen) met behulp van een tweede sleutelwoord. De in het eerste monoalfabetische voorbeeld gebruikte alfabetpermutatie* werd gemaakt met sleutelwoord MASTER aan het begin van het alfabet en sleutelwoord CHILD aan het eind ervan, waarna de tussenliggende posities met de ontbrekende letters zijn opgevuld. Vervolgens is een kolomtranspositie met sleutel ARROGANCE uitgevoerd

* Dat was sleutel YSNZMJXRQIFVDGWEPHBULTOCAK op pagina 81.

en als laatste is het zo ontstane alfabet nog eens vier posities rechtsom gedraaid.

Bij het feit dat er zo veel gepermuteerde alfabetsleutels zijn, past een kritische kanttekening. In [hoofdstuk 1](#) is daar al iets over gezegd. Veel permutaties zijn onbruikbaar. Er mogen bijvoorbeeld maar weinig letters op hun plaats blijven staan. Voor een enkele letter kan dat overigens geen kwaad. Het is zelfs wel goed om dit af en toe te doen, want anders gaat de cryptoanalist er op rekenen dat alle letters zonder uitzondering van plaats zullen wisselen. Maar als in het andere uiterste niet meer dan twee letters van plaats gewisseld zijn, dan zal het cryptogram niet veel van de klaartekst verschillen. Idem dito wanneer de permutatie voornamelijk gelokaliseerd is in betrekkelijk weinig voorkomende letters zoals BJKQV en dergelijke.

3.2 Monoalfabetische varianten

3.2.1 Homofone substitutie

Al vroeg in het bestaan van de cryptografie werd ingezien dat de ongelijkmatige verdeling van de letters in de taal een zwakte vormt. Vooral ten aanzien van de klinkers had men dat snel in de gaten. De eenvoudigste remedie is het introduceren van verschillende substituenten voor de klinkers, in het bijzonder de AEIO. Door de substituenten voor deze letters te laten variëren wordt de frequentie waarmee ze in het cryptogram voorkomen gedrukt en onderscheiden ze zich minder van de andere letters. De verschillende waarden voor een bepaalde letter noemt men *homofonen* en men spreekt van een *homofone substitutie*.

Een voorbeeld van een homofoon geheimschrift is de (gereconstrueerde) sleutel in [tabel 3.1](#), die voorkomt in cryptogrammen uit de correspondentie van raadspensionaris Van der Spiegel met het stadhouderslijk hof in Nijmegen in het jaar 1787^[2]. Men ziet dat voor de klinkers homofonen zijn gebruikt.

Beter dan homofonen voor slechts enkele letters, is het om twee of meer volledige alfabetsubstituties door elkaar te gebruiken. Een voorbeeld daarvan is te vinden in een brief verzonden vanuit Coevorden in

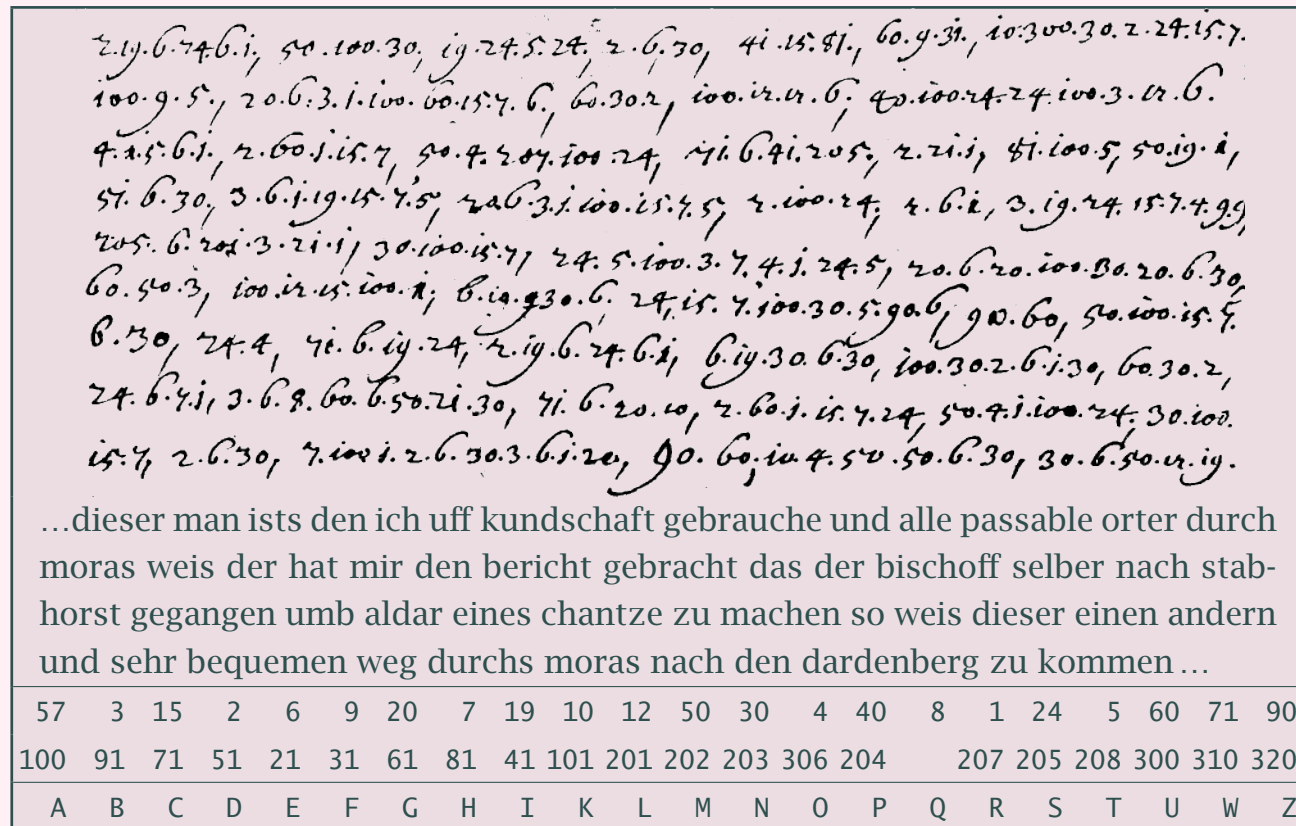
600												
606				604								
608				607				613				
610		602	603	609		611	612	615	617	614	616	618
A	B	C	D	E	F	G	H	I	J	K	L	M
	620							628				
	622							630				
619	625	621		624	626	627	632	629	631		634	635
N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Tabel 3.1 Sleutel homofoon geheimschrift Van der Spiegel

het jaar 1673. Het voorgaande jaar was de stad veroverd door Christoph Bernhard van Galen, de bisschop van Munster bijgenaamd Bommen-Berend, echter kort daarna weer heroverd. [Figuur 3.2](#) laat een stukje van de gecijferde tekst zien, alsmede de beide alfabetten die – op een enkele Q na – uit het beschikbare materiaal konden worden gereconstrueerd.

Degene die een bericht met een homofoon systeem codeert moet er natuurlijk steeds op bedacht zijn de geboden variatie goed te benutten, anders gaat het voordeel weer verloren. Dat gebeurt lang niet altijd en bij veel uit de archieven opgedoken stukken kan men waarnemen dat de codeklerk op dit punt onzorgvuldig is geweest.

Een meer systematische constructie van homofone sleutels is in [figuur 3.3](#) te zien. Het alfabet is in deze voorbeelden gezet in een 5x5 vierkant maar dat is niet persé noodzakelijk. Men kan ook een groter vierkant of een rechthoek met meer dan 25 velden nemen en de extra ruimte benutten voor cijfers en leestekens. Zo'n 5x5 vierkant voor de constructie van een sleutel wordt wel een *Polybius-vierkant* genoemd. In de



Figuur 3.2 Homophoon geheimschrift Coevorden 21 juli 1673

Engelstalige literatuur vindt men ook de term checkerboard*.

In schema I van **figuur 3.3** zijn de coördinaten van de velden aangegeven met letters. Om een M te coderen schrijft men ET, ten minste, als afgesproken is om eerst de rij- en dan de kolomcoördinaat te geven. Het

* Polybius (200–120 BC) was een Griekse geschiedschrijver.

S T R O P					
K	A	B	C	D	E
L	F	G	H	I	K
E	L	M	N	O	P
I	Q	R	S	T	U
N	V	W	X	Y	Z

schema I

1 2 3 4 5					
6	Q	U	A	L	I
7	F	Y	B	C	D
8	E	G	H	K	M
9	N	O	P	R	S
0	T	V	W	X	Z

schema II

1 2 3 4 5					
16	Q	U	A	L	I
27	F	Y	B	C	D
38	E	G	H	K	M
49	N	O	P	R	S
50	T	V	W	X	Z

schema III

Figuur 3.3 Drie schema's voor homofone substitutie

mag ook andersom met eerst de kolom en dan de rij. Aangezien elke letter door een tweetal is vervangen, wordt het cryptogram natuurlijk wel tweemaal zo lang als de klaartekst.

klaartekst : O R D E L E I D T T O T A L L E D E U G D E N
rij-kolom : EOITKOKP ESKPLOKIO IOEOIO KSESESKP KOKPIPLTKOKPER
kolom-rij : OETIOKPK SEPKOLOKIO IOEOIO SKSESEPK OKPKPITLOKPKRE

Omdat in dit voorbeeld geen gemeenschappelijke letters voorkomen in de rij- en de kolomcoördinaten van het sleutelvierkant, kan men die zelfs willekeurig door elkaar gebruiken.

klaartekst : O R D E L E I D T T O T A L L E D E U G D E N
gemengd : OEITKOPK ESPKOLKIO OIEOIO KSSEESPK KOKPIPTLOKPKER

In schema II zijn de coördinaatletters vervangen door cijfers en is het vierkant gevuld met een permutatie van het alfabet. Het kan vanzelfsprekend ook anders, namelijk de cijfers langs de kanten in willekeurige volgorde zetten, of zelfs allebei doen. In dit schema mogen rijen en kolommen weer door elkaar gebruikt worden, omdat er geen gemeenschappelijke coördinaten zijn. In schema III is nog een andere variant bedacht, namelijk meerkeuze in de rijcoördinaat.

```
klaartekst : O R D E L E I D T T O T A L L E D E U G D E N  
schema II : 92495781 6481657501 012910 36464618 57182682758191  
schema III : 42447531 1481157551 014251 13641431 25811232758191
```

Het idee is duidelijk. Met meer homofonen voor veelvoorkomende letters neemt hun frequentie af en wordt het moeilijker om hun identiteit te achterhalen.

3.2.2 Monome-dinomesubstitutie

Een nadeel van een systeem met een vast aantal vervangende symbolen per cryptografisch element is het gemak waarmee de cryptoanalist het cryptogram kan opsplitsen in afzonderlijke elementen. Ze kunnen dan worden uitgeteld en zullen wellicht door hun frequentie en andere eigenaardigheden hun identiteit verraden. Dit wordt veel moeilijker als de lengte van een gecodeerd symbool niet vastligt maar kan variëren. Het navolgende systeem maakt vercijfering met in het cryptogram een verschillend aantal symbolen per letter van de klaartekst mogelijk. Het meest voorkomend is een omzetting naar cijfers, maar hetzelfde principe kan ook gebruikt worden om te substitueren naar lettergroepen van wisselende lengte.

Een letter wordt vervangen door één of door twee cijfers, respectievelijk *monome* en *dinome* genoemd. Vandaar dat het systeem monome-dinome substitutie heet. **Figuur 3.4** geeft twee sleutels voor dit systeem. De sleutel in schema I bestaat uit een tabel van drie rijen. De bovenste rij is bestemd voor de monomes, de onderste twee zijn voor de dinomes. Links vooraan staan de zogenaamde dinomecoördinaten. Op de eerste rij is de dinomecoördinaat leeg, voor de andere twee rijen is deze respectievelijk 2 en 4.

De tweede coördinaat, die ook voor de monomes geldt, bevindt zich bovenaan de kolommen. De cijfers bovenaan zijn willekeurig gepermuteerd. Het sleutelwoord STRUIKGEWAS is gebruikt om de tabel te vullen met een alfabetpermutatie. Merk op dat in de bovenste rij de waarden van de dinomecoördinaten zijn overgeslagen, omdat deze niet als monome gebruikt kunnen worden. De tabel heeft twee ongebruikte posities aan het eind, die bijvoorbeeld voor leestekens zouden kunnen worden benut. De sleutel in schema II

heeft drie dinomecoördinaten en dus ook drie lege posities op de bovenste rij. De drie dinomecoördinaten zorgen voor zoveel posities in de tabel, dat de cijfers er ook in passen. Het sleutelwoord is hier BLADNERF.

	2	5	0	6	9	4	8	1	7	3
		S	T	R	U		I	K	G	E
2	W	A	B	C	D	F	H	J	L	M
4	N	O	P	Q	V	X	Y	Z		

schema I – sleutel STRUIKGEWAS

	2	5	0	6	9	4	8	1	7	3
		B	L		A		D	N	E	R
2	F	C	G	H	I	J	K	M	O	P
4	Q	S	T	U	V	W	X	Y	Z	.
6	0	1	2	3	4	5	6	7	8	9

schema II – sleutel BLADNERF

Figuur 3.4 Monome-dinome sleutel

Het vercijferen gaat als volgt. Zoek de letter op in de tabel en als deze in de bovenste rij staat wordt alleen het kolomnummer ingevuld: dit is een monome. Staat de letter in een van de andere rijen dan komt eerst het nummer vooraan de rij en daarna het nummer bovenaan de kolom: dit is een dinome. Het uitsluiten van de rijcoördinaten bij het toewijzen van monomes zorgt voor een eenduidige codering. Het voorbeeld luidt vercijferd als volgt

klaartekst		:	ORDE	LEIDT	TOT	ALLE	DEUGDEN
cryptogram	schema I	:	456293	2738290	0450	2527273	2939729342
cryptogram	schema II	:	27387	0729840	402740	9007	874620871

Op het als voorbeeld gegeven schema zijn vele variaties mogelijk. Het aantal rijen mag anders gekozen worden, waardoor ook het aantal beschikbare posities verandert. In plaats van cijfers kan men letters als coördinaten nemen. Omdat er meer dan twee keer zoveel letters als cijfers zijn, kunnen homofonen dan weer zorgen voor meerkeuze in de coördinaatwaarden. Vermeldenswaard is nog dat de vercijferde berichten het kortst worden wanneer de meest voorkomende letters als monome gecodeerd worden.

..	39	50	6	76	8	2	0	6	72	39	2	0	9	79	6	58	6	50	39	31	65	8	78	58	9	0	6
..	I	V	A	N	E	S	T	A	L	I	S	T	O	P	A	R	A	V	I	J	A	R	K	R	O	T	A
76	39	6	8	2	0	6	6	39	2	72	6	4	6	6	71	52	39	37	79	52	8	2	50	39	76	9	50
N	I	A	E	S	T	A	A	I	S	L	A	D	A	A	Q	U	I	;	P	U	E	S	V	I	N	O	V
39	9	72	6	76	4	9	39	76	2	0	58	52	32	32	39	9	76	8	2	1	30	52	8	2	9	58	79
I	O	L	A	N	D	O	I	N	S	T	R	U	C	C	I	O	N	E	S	Y	F	U	E	S	O	R	P
58	8	76	4	39	4	6	79	9	58	79	2	6	32	9	76	0	8	32	39	70	39	8	76	0	9	2	..
R	E	N	D	I	D	A	P	O	R	P	S	A	C	O	N	T	E	C	I	M	I	E	N	T	O	S	..
... Ivan is ready to travel but Tania is isolated here; for she came in violation of instructions and was trapped by events ...																											

Figuur 3.5 Ché Guevara aan Fidel Castro 18 mei 1967

Een realistisch voorbeeld van een monome-dinome vercijfering is te zien in [figuur 3.5](#). Het is een deel van een bericht uit 1967 van Ché Guevara aan Fidel Castro^[3]. Ché zat toen als guerillaleider in Bolivia. Nadat het bericht van de oververcijfering ([pagina 110](#)) is ontdaan, resteert een monome-dinome met sleutelwoord ESTADYOY (dinomecoördinaten 375, alfabet plus ñ en enige aanvullende tekens). Het korte stukje in het voorbeeld geeft een alinea uit het bericht. De Tania waarvan sprake is, is Ché's belangrijkste undercover agente in Bolivia en naar wordt verondersteld zijn minnares. Ze werkte echter tegelijkertijd voor zowel de Oostduitse geheime dienst als de Russische KGB en het was zij die hem uiteindelijk zou verraden.

3.3 Digramsubstitutie

De geringe omvang van het alfabet is de voornaamste reden waarom de eenvoudige monoalfabetische substitutie zo'n zwak geheimschrift is. Met 26 letters zijn weliswaar 26! sleutels mogelijk, maar – zoals al eerder is betoogd – dat grote aantal houdt geen stand tegen eenvoudige cryptoanalyse. Heel vaak geven

bedenkers van een geheimschrift hoog op van het grote aantal sleutels in hun systeem en verbinden daaraan dan de conclusie dat het vinden van de oplossing een onmogelijke zaak is. Ergo, hun systeem is veilig! Ze vergeten dat deze redenering slechts geldig is wanneer uitproberen van alle sleutels de enige mogelijkheid tot cryptoanalyse is. Maar menig systeem heeft zwakke plekken waardoor veel effectievere aanvalsmethoden kunnen worden ingezet. Het aantal sleutels op zich biedt geen veiligheidsgarantie.

Versterking kan worden gezocht in vergroting van het aantal substituenten door invoering van homofone substituties, zoals in het voorgaande is gebeurd. Een andere manier is om het aantal elementen in het alfabet te vergroten. In plaats van individuele letters kunnen *digrammen*, combinaties van twee letters, als substitutie-elementen worden genomen. Het oudst bekende voorbeeld van een digramsubstitutie stamt van Giambattista della Porta (1535-1615) en is te vinden in zijn boek *De Furtivis Literarum Notis*, verschenen in jaar 1563. **Figuur 3.6** toont zijn tabel waarmee combinaties van twee letters kunnen worden vervangen door speciale symbolen. Opvallend is dat deze tabel slechts 20 symbolen telt. Dat blijkt niet zo verwonderlijk als men bedenkt dat in die tijd de letters I/J en U/V nog niet werden onderscheiden, terwijl in het Italiaans de overige letters K, W, X en Y niet voorkomen^[4].

De verzameling AA-ZZ telt met 676 symbolen veel meer elementen dan de magere 26 van het eenvoudige A-Z alfabet. Het aantal sleutels, dat wil zeggen het aantal permutaties van de 676 digrammen, bedraagt maar liefst 676! (een getal van 1238 cijfers). Een bijna onvoorstelbaar groot aantal. En hoewel ook hier het grote aantal sleutels niet alles zegt, is toch sprake van een aanzienlijke verbetering ten opzichte van de eenvoudige monoalfabetische substitutie. **Tabel 3.2** geeft een voorbeeld van een sleutel voor digramvercijfering.

Met die tabel als sleutel is de voorbeeldtekst vercijferd. Omdat de klaartekst een oneven aantal letters telt, moest daaraan een letter worden toegevoegd. Het cryptogram is zodoende één letter langer geworden dan de klaartekst. Voor de duidelijkheid is in het voorbeeld de tekst opgesplitst in digrammen, waardoor de woordverdeling geen zin meer heeft en deze is weggelaten. In de onderste regel van het voorbeeld is het cryptogram weer opgedeeld in de gebruikelijke vijflettergroepen. Let op het herhaalde digram DE

A	T	Q	G	I	M	Z	F	R	L	B	O	E	S	V	P	D	H	N	C	
⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	T
⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙
⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	V
⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	M
⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	P
⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	E
⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	B
⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	N
⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	C
⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	L
⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	F
⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	R
⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	I
⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	Z
⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	D
⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	Q
⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	G
⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	S
⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	H
⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	A

Figuur 3.6 Digramtabel van Giambattista della Porta

in de klaartekst. Ook hier verraden veelvoorkomende combinaties zich door hun frequentie, zij het in mindere mate als gevolg van de spreiding over het grotere aantal digrammen en ook omdat herhalingen beginnend op even of oneven posities nu in het algemeen verschillende substituenten krijgen.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	LZ	SW	BH	YJ	YR	WP	BC	FB	FW	XH	DY	MV	KC	UL	CJ	FJ	XW	BR	AD	JP	BJ	PM	JW	IU	OU	DE
B	AJ	GE	KT	AP	TN	VO	GY	CT	JS	OB	YM	MH	WJ	PF	PA	TA	IF	NR	GG	PV	LH	NX	KX	ST	UT	RP
C	LW	KW	DO	QF	JN	LX	DI	TL	DR	RM	SS	HF	RB	QU	UJ	KR	MY	GO	WF	TG	RS	YQ	SC	FI	CR	HK
D	UV	FP	PS	XZ	EV	GR	SV	KF	ZX	WL	RU	WO	YZ	JJ	NJ	VJ	IT	QT	XG	AS	KE	WE	ND	HS	YC	UH
E	AO	CZ	CI	SI	BV	OM	ZO	LE	GD	LB	OI	UK	RC	DK	PZ	YX	KJ	ZT	EM	BS	IZ	XL	RF	WA	YW	EL
F	OC	RI	SP	FY	VH	QE	SE	FC	IK	NZ	RG	LN	TX	NM	SD	JB	UQ	XY	ZG	ML	AV	JC	QM	PQ	AB	ZF
G	MD	VE	FX	MW	OD	PJ	XX	HT	IC	LC	NH	ZD	GC	YY	VP	YA	PC	BE	JF	DS	QK	SX	EQ	ET	YD	JH
H	BT	TK	PR	KY	EC	AN	HZ	SO	YV	MF	ES	YP	FU	AK	NI	SJ	YT	LY	TF	KV	NV	XV	DJ	WX	OO	QB
I	WR	GK	IE	QH	EZ	OY	MU	MT	LA	BP	HA	NM	TJ	QJ	AL	EE	SU	GA	HI	MG	YO	GW	KS	AY	JE	NO
J	VG	ZY	UE	FM	EH	FR	ZW	CA	DN	WD	KD	AU	GP	YS	XM	MR	NC	BQ	HC	NS	NN	ZJ	GJ	VB	RA	TH
K	KQ	UR	VQ	AT	OA	YI	FS	RJ	LT	JD	KI	PG	AC	MI	CD	BG	TZ	PH	OT	WQ	IH	LK	OK	XE	HY	CX
L	YE	VX	GS	VY	IM	HW	HB	JX	NE	ZI	IB	HL	BI	QO	VK	AH	LL	VT	YB	DL	ZC	QI	JA	DH	UY	ZH
M	HU	EW	UC	IJ	UO	SQ	OR	EP	ZE	MX	KL	IQ	TS	QZ	BM	TI	JV	VD	XS	OH	IX	TV	TB	QN	UW	KN
N	LM	CB	SK	EY	PO	FQ	LG	MS	RK	VS	RW	CL	II	RO	ZR	NP	HX	RN	BF	IV	DX	XI	UG	BX	JM	AQ
O	TQ	XN	SH	ZS	WK	OX	WU	HH	MQ	PT	GL	QA	EX	PX	ZB	HJ	VW	SB	PL	DB	NA	CM	UX	IA	JK	LU
P	XD	GM	TC	FG	EJ	FN	WT	NF	OG	QY	DZ	NB	NU	IN	ZV	HM	CS	JU	WV	QG	FH	RQ	TE	DA	GH	AF
Q	YG	DV	EF	HV	TU	HR	LJ	CQ	FK	VC	GF	FZ	ER	XK	NW	XU	VA	ED	MN	UI	RL	GX	WH	WS	TM	OW
R	OS	XR	ID	SG	CY	TY	KG	ZN	YL	KZ	OJ	GU	VF	VR	BD	JO	GV	ZU	FF	WG	XF	GZ	KP	KU	QD	JT
S	RY	GQ	ZZ	HP	CC	HQ	UF	AD	PK	DW	XQ	DU	RH	DC	GN	QR	DM	MK	SF	RZ	MC	FT	BZ	LQ	IO	LO
T	YF	BA	UU	YN	TR	LD	WB	NQ	TW	VN	RD	FA	YU	OP	OQ	LR	FL	JI	JZ	HO	QQ	QC	GI	QW	KH	MA
U	JQ	XO	CH	EA	SY	XJ	IG	PD	ZL	LF	LP	KO	JY	ZP	UD	KA	TD	NG	ZQ	CF	AI	XT	HD	XB	UB	CW
V	XC	EI	BU	VV	AX	DF	MZ	VU	VM	RV	UP	PN	WC	FE	DT	IL	ZM	CU	EK	WZ	OF	LS	BL	IS	XA	BB
W	LI	FO	KM	JR	CV	QP	EG	WN	UA	NT	AG	UN	KK	US	WY	MP	SL	MB	BK	KB	AR	YH	DD	OE	DG	VI
X	AE	FD	ZK	SA	QX	SM	HE	CE	ZA	QV	IY	CN	PY	HN	JG	XP	AZ	UZ	BN	BW	PI	MO	AW	QL	DP	HG
Y	RX	NY	TO	MJ	SR	PE	BO	TT	BY	OV	WM	VZ	GT	CO	JL	GB	SN	NK	OL	PU	EU	RE	PP	RT	AM	CG
Z	ON	ME	IP	PB	WI	EB	LV	PW	EN	VL	NL	AA	QS	WW	RR	SZ	DQ	UM	CP	TP	IW	YK	CK	OZ	FV	IR

Tabel 3.2 26x26 Digram substitutie tabel

```
klaartekst : OR DE LE ID TT OT AL LE DE UG DE NQ  
substitutie : SB EV IM QH HO DB MV IM EV IG EV HX  
cryptogram : SBEVI MQHHO DBMVI MEVIG EVHX
```



Wheatstone



Baron Playfair

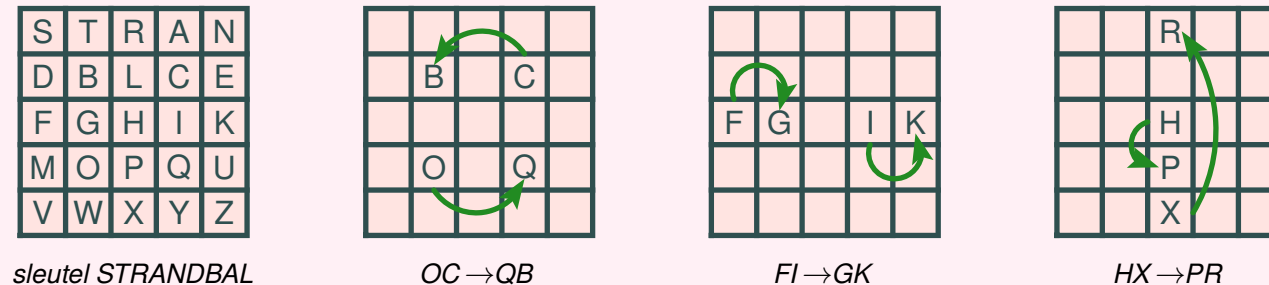
3.3.1 Playfair-substitutie

Het mooie van de substitutie in tabel 3.2 is dat een volstrekt willekeurige verdeling van de digrammen over de tabel mogelijk is. Het nadeel is een direct gevolg daarvan, namelijk de onmogelijkheid om zo'n substitutie te onthouden of eenvoudig aan de hand van een sleutel te construeren. Vandaar dat men naarstig heeft gezocht naar manieren om de digramsubstitutiesleutel op een eenvoudige manier te maken. De keerzijde is dat men als gevolg daarvan niet het volle aantal permutaties benut, waardoor het effectief aantal sleutels veel minder is dan maximaal mogelijk. Het systeem wordt daardoor weer navenant zwakker.

De meest bekende digramsubstitutie is de *Playfair*. Dit is een bijzonder populair geheimschrift geworden, tot in detective romans toe*. Bedacht door Charles Wheatstone (de eerste beschrijving dateert van 1854), is het door baron Playfair bij het British Foreign Office aan de man gebracht en heeft daaraan z'n naam te danken. In de meest algemene vorm wordt een permutatie van het alfabet ingeschreven in een rechthoek, maar in de praktijk komt men hoofdzakelijk het 5x5 vierkant tegen. Eén van de 26 letters moet daarbij worden weggelaten (gewoonlijk is dit de J)[†]. Figuur 3.7 geeft een sleutelvierkant gebaseerd op het sleutelwoord STRANDBAL.

* Een mooi voorbeeld van een Playfair in een detectiveroman – ook goed oefenmateriaal voor beginnende cryptoanalisten – is te vinden in Dorothy Sayer's *Have His Carcase* (1932), in het Nederlands vertaald onder de titel *Lijk in zicht*.

[†] In het Nederlands kan dit ook heel goed de Q (schrijf KW) of de Y (schrijf IJ) zijn.



Figuur 3.7 Playfair sleutelvierkant en vercijferoperaties

Het vercijferen gaat als volgt in zijn werk. Van digram OC staan de twee letters op de overstaande hoekpunten van een rechthoek en worden vervangen door de letters QB op de andere hoekpunten. Daarbij is de volgorde van de rijen aangehouden: de eerste letters van klaartekstdigram en cijferdigram bevinden zich op dezelfde rij in het sleutelvierkant. Dit is de gangbare afspraak voor Playfair-vercijfering. Maar niets verhindert om het anders te doen en de beide eerste letters uit dezelfde kolom te nemen. In dat geval wordt BQ het substitutieresultaat.

Voor digram FI valt geen rechthoek te construeren, omdat de beide letters in dezelfde rij staan. Daarom worden de letters direct rechts ervan als vervangers genomen en dus gaat $F \rightarrow G$ en $I \rightarrow K$ en samen $FI \rightarrow GK$. Bij de K van FK ligt het voor de hand om als naastliggende letter de rij cyclisch te doorlopen en $K \rightarrow F$ te nemen, geeft $FK \rightarrow GF$. Hetzelfde, maar dan in verticale richting, gebeurt als de te substitueren letters in dezelfde kolom staan. Hier worden de letters eronder genomen waarbij de letters in de onderste rij worden opgevolgd door de letter bovenaan de kolom: $HX \rightarrow PR$.

Ten slotte kan het voorkomen dat een verdubbelde letter in de tekst voorkomt. Dan is er geen rechthoek en ook geen rij of kolom om als vervangersrichting te gebruiken. De meest gangbare oplossing is om het digram op te splitsen door tussenvoegen van een letter. In het voorbeeld is X gebruikt om TT op te splitsen tot TX-T. Het cryptogram wordt hierdoor langer dan de klaartekst. Is het resulterend aantal letters oneven, dan moet zelfs nog een extra letter worden toegevoegd. Een andere oplossing is het vervangen

door de letter er naast. In dat geval blijft TT staan in de klaartekst en wordt door RR in het cryptogram vervangen.

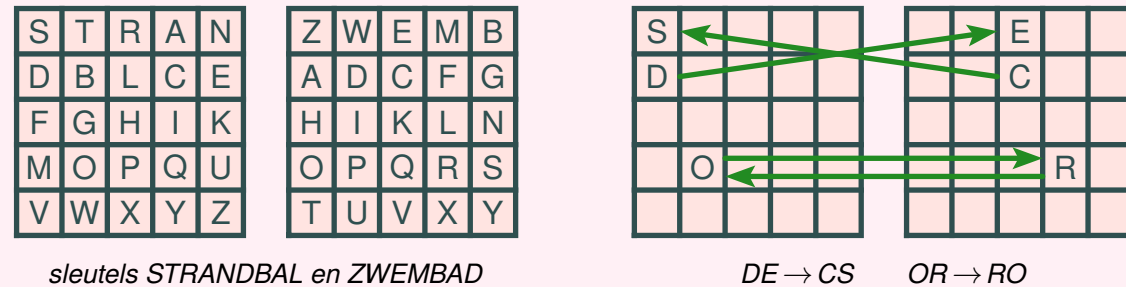
De regels toepassend op het voorbeeld wordt het cryptogram als volgt

```
klaartekst : OR DE LE ID TT OT AL LE DE UG DE N  
aanpassing : OR DE LE ID TX TO TA LX LE DE UG DE NQ  
substitutie : PT BD CD FC RW BW RN HR CD BD OK BD AU  
cryptogram : PTBDC DFCRW BWRNH RCDBD OKBDA U
```

3.3.2 Twosquare-substitutie

Iets meer flexibiliteit dan de Playfair wordt verkregen door het gebruik van twee vierkanten in plaats van een. In het Duits heet dit systeem Doppelkasten (twosquare in het Engels). In de Tweede Wereldoorlog werd dit systeem aan Duitse kant gebruikt voor de laagste legeronderdelen. Engelse soldaten maakten tijdens de campagne in Noord-Afrika het systeem buit bij een overval op een Duits communicatiecentrum^[5]. Over het algemeen gebruikten de Duitsers bij het maken van dit soort sleutelvierkanten geen sleutelwoord, maar namen willekeurige permutaties. Dat dit veel beter is kan men direct zien aan de sleutelvierkanten in het voorbeeld. Gebruik van een sleutelwoord brengt vaak het merendeel van de letters UVWXYZ op de onderste rij, iets wat beter kan worden vermeden.

Voor de vercijfering wordt de eerste letter van het te vercijferen digram uit het vierkant links genomen, de tweede letter uit het rechtervierkant. Deze twee bepalen in het algemeen een rechthoek met hoekpunten in elk van beide vierkanten. De overstaande hoekpunten geven het vercijferd digram. Als de rechthoek gedegenereerd is tot een lijn, doordat de beide letters in dezelfde rij liggen, neemt men de letters in omgekeerde volgorde. Merk op dat een digram met een verdubbelde letter hier geen speciaal probleem vormt.



Figuur 3.8 Twosquare sleutelvierkanten en vercijferoperatie

klaartekst : OR DE LE ID TT OT AL LE DE UG DE NQ
 substitutie : RO CS CR IC ZW OW MI CR CS SE CS EU
 cryptogram : ROCSC RICZW OWMIC RCSSE CSEU

3.4 Fractionerende systemen

In een *fractionerend* systeem worden de letters opgesplitst en de delen op een andere manier weer samengevoegd tot de letters van het cryptogram. In het algemeen kan men in een fractionerend systeem drie fasen onderscheiden

1. een substitutie naar een representatie met twee of meer componenten
2. de componenten ondergaan afzonderlijk een transpositie
3. de componenten worden in hun nieuwe volgorde samengevoegd

Soms wordt de laatste combinatiestap achterwege gelaten en blijven de gesplitste componenten staan. Het cryptogram wordt dan twee keer zo lang als de klaartekst. Twee voorbeelden van fractionerende systemen worden hier behandeld, de bifid en de ADFGVX.

3.4.1 Bifid of Delastelle-substitutie

De *bifid* is bedacht door de Fransman Delastelle (1840–1902). Voor de eerste stap, een substitutie naar een vorm die opsplitsing van de letters mogelijk maakt, wordt als voorbeeld sleutel-1 van [figuur 3.9](#) gebruikt. De letters worden daarmee vervangen door hun rij- en kolomcoördinaten*.

	1	2	3	4	5
1	B	U	L	D	O
2	G	A	C	E	F
3	H	I	K	M	N
4	P	Q	R	S	T
5	V	W	X	Y	Z

sleutel BULDOG

	1	2	3	4	5
1	H	O	N	D	E
2	K	A	R	B	C
3	F	G	I	L	M
4	P	Q	S	T	U
5	V	W	X	Y	Z

sleutel HONDENKAR

Figuur 3.9 Bifid sleutelvierkanten

klaartekst :	M	A	A	K	G	E	E	N	B	E	E	S	T	E	N	B	O	E	L
cijfers :	34	22	22	33	21	24	24	35	11	24	24	44	45	24	35	11	15	24	13

Om de verdere verwerking te vergemakkelijken wordt dit een beetje anders opgeschreven. De tekst wordt verdeeld in stukken, de *periode* van de bifid, en bovendien komen de rij- en kolomcoördinaten *onder elkaar* te staan. Het blijkt verstandig om oneven periodes te nemen. In het voorbeeld is periode 7 gekozen.

klaartekst :	MAAKGEE		NBEESTE		NBOEL
rij :	3223222		3122442		31121
kolom :	4223144		5144454		51543

* Indien de opsplitsing geschiedt in drie, in plaats van in twee coördinaten, spreekt men van een *trifid*.

Na de splitsing komt de transpositie aan de beurt en daarna de recombinitie. Voor de samenvoeging is hier dezelfde sleutel-1 gebruikt als bij de splitsing. In de oorspronkelijke volgorde behoren de cijfers in de verticale richting bijeen, voor de transpositie nemen we ze daarom juist horizontaal samen. De eerste twee te combineren cijfers zijn 3 en 2. De 3 is de rijcoördinaat van de M en de 2 is de rijcoördinaat van de A. Samen geeft dit 32 en via het substitutievierkant van sleutel-1 wordt dit omgezet naar de letter I. De volgende twee cijfers 2 en 3 worden op dezelfde manier een C, enz. Aan het eind van de periode gekomen, wordt de 2 afkomstig van de tweede E gecombineerd met de eerste kolomcoördinaat, de 4 afkomstig van de M. Dit spel wordt herhaald voor alle groepen, waardoor het volgende cryptogram ontstaat.

klaartekst :	M	A	A	K	G	E	E		N	B	E	E	S	T	E		N	B	O	E	L
cijfers :	34	22	22	33	21	24	24		35	11	24	24	44	45	24		35	11	15	24	13
rij :	3	2	2	3	2	2	2		3	1	2	2	4	4	2		3	1	1	2	1
kolom :	4	2	2	3	1	4	4		5	1	4	4	4	5	4		5	1	5	4	3
transpositie :	32	23	22	24	22	31	44		31	22	44	25	14	44	54		31	12	15	15	43
cryptogram :	I	C	A	E	A	H	S		H	A	S	F	D	S	Y		N	U	O	O	R

Als variatie kan men afzonderlijke lettervierkanten nemen voor de splitsing en voor de recombinitie. Met sleutel-1 in [figuur 3.9](#) voor de eerste en sleutel-2 voor de tweede operatie resulteert onderstaand cryptogram. Merk op dat dit en het vorige cryptogram door middel van een monoalfabetische substitutie in elkaar overgevoerd kunnen worden: ze zijn isomorf, waarover meer in [hoofdstuk 6](#).

klaartekst :	M	A	A	K	G	E	E		N	B	E	E	S	T	E		N	B	O	E	L
cijfers :	34	22	22	33	21	24	24		35	11	24	24	44	45	24		35	11	15	24	13
transpositie :	32	23	22	24	22	31	44		31	22	44	25	14	44	54		31	12	15	15	43
cryptogram :	G	R	A	B	A	F	T		F	A	T	C	D	T	Y		F	O	E	E	S

De recombinitie kan ook op nog andere wijze gebeuren. Bijvoorbeeld door het kolomnummer van de eerste letter samen te nemen met het rijnummer van de volgende, enz. De overblijvende coördinaten van

het laatste kolom- en het eerste rijnummer worden samengevoegd en al naar gelang de afspraak voor- of achteraan de rij gezet. De eerste cryptogramletter wordt zo 42 is Q, gevolgd door 22 is A, enz. De afsluiting is gedaan met de laatst overblijvende 3 aan het eind en de eerste 3 van de M aan het begin. Tenslotte is in deze variant ook nog eens afgezien van de opdeling in periodes en is de gehele klaartekst van begin tot eind in één ruk door vercijferd. Hieronder het resultaat van die vercijfering.

klaartekst :	M	A	A	K	G	E	E	N	B	E	E	S	T	E	N	B	O	E	L	
cijfers :	34	22	22	33	21	24	24	35	11	24	24	44	45	24	35	11	15	24	13	
rij :	3	2	2	3	2	2	2	3	1	2	2	4	4	2	3	1	1	2	1	
kolom 1 :		4	2	2	3	1	4	4	5	1	4	4	4	5	4	5	1	5	4	3
transpositie :		42	22	23	32	12	42	43	51	12	42	44	44	52	43	51	11	52	41	33
cryptogram :		Q	A	R	G	O	Q	S	V	O	Q	T	T	W	S	V	H	W	P	I

3.4.2 ADFGX en ADFGVX

Het ongetwijfeld beroemdste onder de fractionerende geheimschriften is wel de ADFGX en diens opvolger de ADFGVX. Dit geheimschrift verscheen voor het eerst op 1 maart 1918 in de communicatie tussen de hogere hoofdkwartieren van de Duitse legers. Op dat ogenblik in de Eerste Wereldoorlog stonden de Duitsers op het punt om met een laatste, min of meer wanhopig slotoffensief te proberen de patstelling in de loopgravenoorlog aan het westelijk front in Vlaanderen en Noord-Frankrijk te doorbreken*. De cryptogrammen bestonden uit slechts de vijf letters A, D, F, G en X, waaraan de naam van het geheimschrift is ontleend. Op 1 juni daaropvolgend werd het systeem uitgebreid met de letter V. Deze letters zijn zo gekozen dat ze in morsecode sterk van elkaar verschillen, waardoor de kans op seinfouten kleiner wordt†.

* Het is een goede tactiek om aan het begin van een belangrijke campagne van geheimschrift te veranderen. Het zal ongetwijfeld enige tijd duren voordat de tegenstander er in slaagt om dit te breken. In de tussentijd kan die tegenstander zijn acties veel minder baseren op gebroken berichten dan in de voorafgaande periode.

Als voorbeeld van dit geheimschrift is een van de op 1 april 1918 onderschepte telegrammen genomen. In **figuur 3.10** zijn het telegram, de sleutel en de ontcijfering ervan weergegeven. De eerste stap is de bepaling van het aantal lange en korte kolommen aan de hand van de lengte van het cryptogram. De transpositiesleutel telt 20 kolommen en gegeven dat het cryptogram 46 letters telt, zijn er dus 6 lange en 14 korte kolommen van respectievelijk 3 en 2 letters. De letters worden ingeschreven in de volgorde van de sleutel, die in de tweede rij van de figuur staat. FX gaat in de kolom gelabeld 1, FX in de kolom met 2, AAD in 3, AA in 4, enz. De letters zijn vervolgens twee aan twee samengenomen en met het substitutievierkant vertaald naar de klaartekst.



Painvin

FXFXA ADAAD GDDDD DGGDG AXADD GDDDG FXADF GGDFX AFGGA F																			
13	3	18	7	6	11	4	19	9	16	8	5	17	2	10	15	1	14	12	20
D	A	X	D	D	D	A	G	G	G	G	D	D	F	X	D	F	X	D	A
G	A	A	D	D	D	A	G	A	G	D	G	F	X	A	F	X	A	D	F
F	D	F	G	D	G														
S		T		A		N		D		O		R		T		X		S	
U		Z		A		N		N		E		X		K		I		R	
C		H		E															
STANDORT X SUZANNE X KIRCHE																			

	A	D	F	G	X
A	G	Z	K	N	B
D	S	A	R	E	Y
F	L	C	P	H	X
G	U	O	W	D	Q
X	I	T	F	V	M

Figuur 3.10 ADFGX bericht van 1 april 1918

Aanvankelijk werden te weinig berichten onderschept om iets mee te kunnen beginnen, maar op 1 april konden 18 berichten uit de ether worden geplukt. Dit bleek voor de Fransman Georges Jean Painvin voldoende om het systeem te breken^[6]. In de voor het verloop van de laatste fase in de oorlog zo belangrijke periode van maart tot eind juni 1918 is slechts een tiental dagsleutels gebroken. Maar dat waren juist de dagen met veel berichten en belangrijke informatie over komende acties. Eén van de op 1 juni 1918 onderschepte telegrammen bleek zo belangrijk, dat de Fransen dit *Le radiogramme de la victoire* hebben genoemd.

† A = ·- D = -·· F = ···- G = --- V = ···- X = -··-

Het ADFGVX-systeem is een fractionerend systeem waarin de recombinatiestap achterwege blijft, dit in tegenstelling tot de bifid. In de eerste verschijningsvorm bestond de substitutie uit een 5x5-vierkant gelabeld met de vijf letters ADFGX. De in juni volgende uitbreiding met de letter V tot 6x6 maakte het mogelijk om ook de tien cijfers een plaats te geven. Handig, omdat in de berichten veelvuldig getallen voorkwamen. Bij de vercijfering worden de letters gesplitst in afzonderlijke coördinaten, gevolgd door een kolomtranspositie. Zoals gezegd blijft recombinatie achterwege.



Pierre Lyonet

3.5 Nomenclatuur en code

Wat begon als een monoalfabetische substitutie met homofonen voor de klinkers werd reeds lang geleden verder uitontwikkeld. Er kwamen homofonen voor medeklinkers bij, elementen voor groepen van twee of drie letters en substituenten voor complete persoons- en plaatsnamen. Het resultaat wordt *nomenclatuur* genoemd. De vroegste voorbeelden van dit geheimschrift stammen uit het begin van de 15^e-eeuw en worden gevonden in het diplomatiek verkeer van het Vaticaan en van de Italiaanse stadstaten^[7]. In [figuur 3.11](#) is een van de oudste voorbeelden, een Florentijnse nomenclatuur uit 1424 afgebeeld.

In de gehele 16^e en 17^e-eeuw speelde deze nomenclatuur een hoofdrol in het diplomatiek berichtenverkeer. Door geheel Europa liep weliswaar een netwerk van postroutes, maar zonder briefgeheim. Pakketten post werden systematisch opengemaakt en de inhoud afgeschreven. Grote mogendheden zoals Engeland en Frankrijk hadden een zogenaamde *zwarte kamers* (Black Chamber in het Engels) in bedrijf, waarin onderschepte nomenclaturen werden gebruikt om berichten te oncijferen en waarin zelfs onbekende nomenclaturen werden gereconstrueerd uit de stroom van berichten. Er was dus voortdurend behoefte aan het verversen van de nomenclaturen ter vervanging van gecompromitteerde versies. Ook in onze eigen Republiek van de Verenigde Nederlanden opereerden in de loop van de 18^e-eeuw enkele zeer succesvolle zwarte kamers. In de tweede helft van die eeuw bemand door Pierre Lyonet en later door diens neef Samuel Egbert Croiset*.

a	b	c	d	e	f	g	h	i	l	m	n	o	p	q	r	s	t	u	x	z
∴	9	9	u	S	π	=		∴	∴	6	>	8	4	∴	∴	∴	∴	∴	∴	∴
H	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴
ε	E		∴										∴							∴
Nulle: ∴																				
Ambasciatore	∴	Commune di Firenze	∴	Gente d'arme	∴															
Agnalo	∴	Conte d'Urbino	∴	Lance	∴															
Bernardo	∴	Con	∴	Malatesta	∴															
Bernardino	∴	Dieci (i)	∴	Mantova	∴															
Carlo di Malatesta	∴	Duca di Milano	∴	Mille	∴															
Carlo (Signore)	∴	Faenzi (Signore di)	∴	Non	∴															
cento	∴	Fanti	∴	Pandolfo	∴															
che	∴	Fiorini, danari	∴	Pace	∴															
castello	∴	Galeazzo	∴	Per	∴															
Città	∴	Galeotto	∴	Tregua	∴															

Parole:	
Andrea	= Galeotto
Che	= Cavalli
con	= Guerra
Giovanni	= Dieci di Balia
Tagliato	= Scapito
Trombetta	= Ambasciatore

Figuur 3.11 Florence 1424: Cifra di Galiotto Fibindacci da Ricasoli

Een voorbeeld van een missive in nomenclatuur staat in [figuur 3.12](#). Dit bericht is gedagtekend Madrid 27 juli 1658 en behelst een instructie van de Spaanse koning Philips IV aan zijn ambassadeur bij de Nederlanden. De figuur geeft slechts het eind van de missive, de gehele instructie beslaat zeven vellen. De codering is in het officiële regeringsgeheimschrift van dat moment, dat wil zeggen het cifra general

- * Men hield kennis en vaardigheden van cryptografie en cryptoanalyse graag in de familie, vandaar dat Lyonet zijn neef Croiset opleidde als zijn opvolger.

243 17 g po 16 ya go 48 ge 22 15 8 25 61 gi ma so 25 229 22 25 5 179 22 25
 84 53 80 ac de 17 25 84 22 22 22 nu ma 16 212 ge 16 50 19 15 mi ya 60 g 16g
 ma g 23 60 xe o 56 63 fi 84 de 3 8 ge so 25 70 g gi ri 8 la 25 ge 22 25 ma
 153 53 59 gi 22 61 ge 25 70 g 2 54 17 16 58 22 61 17 25 22 27 22 nu ma 16 53
 61 gi mi 22 g go ge 22 15 toc 213 53 60 mi fi 80 o 22 g 80 22 25 56 63 8
 25 no xe 16 la 61 17 25 70 g 50 g me g ri 8 g 5 g ri 22 16 mo 80 g fi
 nu 61 22 15 xu 22 g 237 60 fe 84 fe g o 74 *Cv M 227 de Julio 1658*
 yo Weyf

Figuur 3.12 Instructie van Philips IV aan zijn ambassadeur d.d. 27 juli 1658

ingaaude oktober 1653.

Alle elementen van een nomenclatuur zijn in het voorbeeld aanwezig. In de bovenste regel staat bijvoorbeeld 16 voor de letter *r*, de daaropvolgende groep *ya* staat voor de twee letters *ma* en 153 aan het begin van de vierde regel betekent *instrucion*. Andere groepen in de lijst, zoals *paz* voor *Don Juan de Austria* en 193 voor *Principe de Oranje*, verraden waar de belangrijkste interesses lagen ten tijde van het opstellen van deze nomenclatuur.

Een nomenclatuur kon lang in gebruik blijven, het samenstellen ervan was immers een heel werk. De nomenclatuur van [figuur 3.13](#) heeft het ruim twintig jaar volgehouden van 1653 tot 1675, extreem lang. Tegen het eind van die periode moet heel Europa hebben kunnen meelesen! Soms echter bleek er aanleiding om ze snel te vervangen, zoals wordt gesuggereerd door twee elkaar in korte tijd opvolgende nomenclaturen van koning Philips II, namelijk het cifra general de Su Magestad van respectievelijk 26-5-1568 en 1-7-1568. Een vervanging kon het gevolg zijn van het verloren gaan of in verkeerde handen vallen

a	b	c	d	e	f	g	h	i	l	m	n
1	u	t	s	α	z	c	m	11	d	12	g
q	x	2	n	22	y	23	b	17	15	p	f
r			o				e				
o	p	q	r	s	t	v	x	y	z	j	
6	20	9	4	h	18	19	27	5	3	7	
8	21	13	16	25	26	2	28	1	14	24	
i					5						
Finales						Nulas					
α	e	i	o	u		η	o	ƒ			
÷	+	I	//	/							
Numeros						Duplices					
20	21	22				7	θ	ƒ			

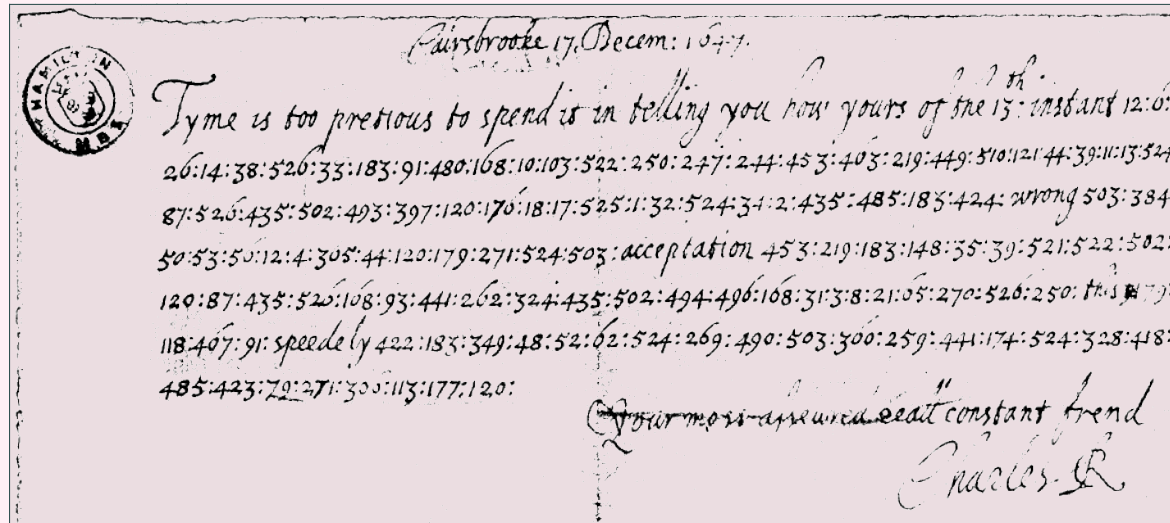
emperador	pez
emperatriz	piz
España	poz
Español	puz
embaxador	7
embaxada	θ
Elector	ƒ
esquizaro	100
execucion	101
exercito	102
empresa	103
experiencia	104
esperanza	105

Figuur 3.13 Spanje: Cifra real á principio de Otubre del año 1653

van de sleutel, maar kon ook komen door een reorganisatie in het secretariaat van de koning. Naast de nomenclatuur voor algemeen gebruik bestonden ook versies voor de correspondentie met afzonderlijke personen. Een voorbeeld daarvan is de nomenclatuur bestemd voor een in onze geschiedenis niet onbekende hertog, het cifra particular de Su Magestad con el duque de Alva van 1-7-1568. De sleutels van al deze genoemde Spaanse geheimschriften zijn in de archieven van Simancas (Valladolid) te vinden en gecompileerd in een uitgave van de Belgische Koninklijke Academie^[8].

In andere gevallen is de sleutel niet bewaard gebleven, maar kan deze met behulp van gedecodeerde, wel bewaard gebleven afschriften van de in geheimschrift gestelde correspondentie worden gereconstrueerd. Een voorbeeld daarvan is te zien in [figuur 3.14](#), een gecodeerde brief van de Engelse koning Karel I*.

* Scottish Record Office: GD406/1/2186. Karel I (1600–1649) kwam tijdens zijn regeerperiode voortdurend met het Engelse parlement in aanvaring. Begin 1647 werd hij door de Schotten aan het parlement uitgeleverd en twee jaar later door toedoen van Cromwell terechtgesteld.



Figuur 3.14 Brief van Karel I aan Earl of Lanerick 17 dec 1647

Een klein stukje van de sleutellijst staat in [tabel 3.3](#). De lijst telt een kleine 600 groepen en is daarmee van bescheiden omvang. Deze lijsten konden soms wel duizenden tot tienduizenden groepen bevatten. De vraagtekens in de sleutel staan op de plaats van groepen die niet zijn aangetroffen, of die wel in gecodeerde brieven voorkomen maar waarvan de vertaling niet uit de context kon worden opgemaakt. Alle elementen van het geheimschrift zijn hier van eenzelfde soort (namelijk cijfergroepen) en daarom wordt dit een *code* genoemd in plaats van een nomenclatuur. Dat het cijfers zijn is niet wezenlijk, meer dat alle elementen gelijksoortig zijn. Er bestaan ook codes die geheel uit lettergroepen bestaan, zoals bijvoorbeeld een Engelse marinecode uit het begin van de 20^e-eeuw.

Het voorbeeld van [tabel 3.3](#) laat groepen van één en drie cijfers zien, in de gehele code komen ook groepen van twee cijfers voor. Later zal men in codes alle groepen dezelfde lengte gaan geven. Daardoor is het niet meer nodig om met scheidingstekens tussen de groepen te werken. Zo kan men de tegenstander ook nog in onzekerheid laten omtrent de grootte van de afzonderlijke codegroepen.

0 ?	130 Cromwell	280 keep	370 ?
1 null	131 Culpepper	281 King	371 Parliament
2 e	132 e	282 kingdom	372 Pembroke
3 a	133 can	283 know	373 Pendennis
4 i	134 cannot	284 Lancashire	374 Plymouth
5 o	135 ?	285 L...? (naam)	375 ?
6 u,v	136 care	286 ?	376 party
7 w	137 ?	287 ?	377 pass
8 y	138 cause	288 ?	378 ?
9 a	139 city	289 London	379 per

Tabel 3.3 Deel sleutel van geheimschrift Karel I

Dutch Oil Trading Company Koningsplein Batavia We think it advisable to do it promptly, otherwise you may not be able to compete. Before we can come to any conclusion in the matter however requires further information regarding present conditions your market as well as how soon a change will likely take place. If through any unforeseen circumstances it will be necessary to make a counteroffer you may make a cut in price of 2¼ to 2½% if you can land the business.	Dutoiltrad Batavia bitatomufy widibewedi fugabyluge rabiwezuwo tegynariwe nugudemisi tasesipidu voresofyby xavodegify ekibikuduk
--	--

Figuur 3.15 Bericht in A1-handelscode (Londen 1888)

Niet alleen in het diplomatiek berichtenverkeer bewezen codes hun diensten. Voor het handelsverkeer werden speciale codes te koop aangeboden. De ontwikkeling van deze codes komt zo rond 1870 op

gang, ongeveer in de tijd waarin alle delen van de wereld door kabels worden verbonden*. In een tijd waarin berichten over zakentransacties per telegram via het postkantoor werden verzonden, gaf dat enige bescherming tegen onbescheiden blikken op de inhoud.

Evenzeer van belang was het feit dat een goed ontworpen code de berichten flink in omvang kon reduceren. Dat scheelde aanmerkelijk in de kosten. Zo berekende de PTT in het jaar 1932 voor verzending van Amsterdam naar Batavia (nu Jakarta, Indonesië) van het telegram links in [figuur 3.15](#) een tarief van f 2 per woord ofwel f 170 voor het gehele telegram. De in code gestelde tekst rechts kostte maar f 24^[9]. De telegraafdiensten berekenden meestal een apart tarief voor codegroepen, soms meer voor cijfergroepen dan voor lettergroepen. Lettergroepen gaf men vaak een afwisseling van klinkers en medeklinkers. Dat maakt dat ze nog enigszins uitgesproken kunnen worden en zodoende gemakkelijker te verwerken door de telegrafisten. Een codegroep als OMPOT valt nu eenmaal beter te onthouden dan KXFRM en zulke telegrammen waren daarom goedkoper.

3.5.1 Een- en tweedelige code

De code van [tabel 3.3](#) is een zogenaamde *eendelige code*. Voor het coderen en decoderen van berichten wordt dezelfde lijst gebruikt. Dat kan goed omdat in de sleutellijst de klaartekst en de codecijfers gelijk oplopen. Bij het coderen heeft men het gemak van de alfabetische volgorde van de klaartekst elementen, bij het decoderen van de oplopende reeks cijfercodes.

Hoewel gemakkelijk in het gebruik heeft dit systeem een bezwaar. Het kan eenvoudiger gebroken worden dan een code waarvan de groepen in willekeurige volgorde staan. Kijk in genoemde tabel naar de naam die zich onder groep 285 verschuilt. Uit de plaats in de lijst valt af te leiden dat 285 alfabetisch thuishoort tussen 284 = Lancashire en 289 = London. Voor een kenner van de lokale en familiale connecties van de correspondenten valt die naam waarschijnlijk wel te raden[†].

* In 1850 de eerste kabel tussen Dover en Calais, in 1866 de eerste transatlantische kabel.

Om voornoemde reden is een *tweedelige code* beter. In zo'n code wordt gebruik gemaakt van afzonderlijke lijsten voor het coderen en het decoderen. De codeerlijst geeft de klaartekstgroepen in alfabetische volgorde, de corresponderende codegroepen staan wat betreft nummering door elkaar. De decodeerlijst is daarentegen gesorteerd naar de codegroepen, de bijbehorende klaarteksten zullen een willekeurige alfabetische volgorde vertonen. Het nadeel van twee aparte lijsten weegt op tegen de grotere mate van veiligheid die een dergelijke code biedt. Het stukje van zo'n code in [figuur 3.16](#) is een voorbeeld van een kleine tweedelige code. Deze werd in de Eerste Wereldoorlog in de loopgraven gebruikt om snel en veilig de situatie te rapporteren en orders door te geven. Door de geringe omvang kon deze overigens gemakkelijk genoeg gekraakt worden en moest daarom voortdurend worden ververs.

About to advance	BY	AF	Enemy machine gun fire
Ammunition exhausted	FB	AG	Gas is being released
Are advancing	PX	AP	Stretcher bearers needed
At	SX	AV	Recall working party
Attack failed	BM	AW	Casualties heavy
Attack successful	PF	AX	Using gas shells
Barrage wanted	XF	AZ	Relief completed
Be ready to attack	ZF	BJ	Situation serious

Figuur 3.16 Loopgraafcode uit de Eerste Wereldoorlog

† Merk op dat in de codetabel het alfabetisch oplopen van de klaartekst niet helemaal consequent gedaan is. Bij elke letter komen namelijk eerst de persoons- en plaatsnamen, daarna de overige woorden beginnend met deze letter. Dat maak het tegelijk ook makkelijker om de codegroepen voor namen en plaatsen van elkaar te onderscheiden.

3.5.2 Vercijferde code

Hoe men het ook wendt of keert, een code blijft in wezen een monoalfabetische substitutie. Weliswaar met een groot alfabet, maar toch in essentie monoalfabetisch en dus kwetsbaar voor het herkennen van veelvoorkomende groepen. Immers, stel het lidwoord *de* wordt vercijferd tot codegroep 37125. Omdat *de* nogal eens voorkomt in onze taal, zal van de weeromstuit 37125 ook veel voorkomen. Zoals de letter E zich in monoalfabetisch vercijferde tekst verradt door z'n hoge frequentie, zo zal datzelfde in een code gebeuren met *de*.

Om een code hiertegen te beveiligen past men een *oververcijfering* toe en spreekt dan van *encicode* (enciphered code in het Engels). Met deze eenvoudige operatie wordt het monoalfabetisch karakter van de code verhuld. Het gemakkelijkst gaat dit met cijfercodes, waar de meest gangbare oververcijfering bestaat uit het optellen of aftrekken van een rij willekeurige cijfers, de sleutelrij*. Deze rekenoperatie geschiedt onder verwaarlozing van de carry, dat wil zeggen cijfer voor cijfer zonder overdracht of lenen van tientallen. Bij optellen van een sleutelrij spreekt men van een *additief* of *optelgetallen* en bij aftrekken van een *subtractor*. Als voorbeeld een stukje (fantasie) code, oververcijferd met respectievelijk een subtractor en een additief. Uiteraard moet het ontcijferen met dezelfde cijferreeks geschieden.

codegroepen :	5060	1798	8311	1908	4945	7640	9885	4749	3728	5984
subtractor :	4978	1489	3880	0503	8658	8433	5999	2863	3808	4801
encicode :	1192	0319	5531	1405	6397	9217	4996	2986	0920	1183
codegroepen :	5060	1798	8311	1908	4945	7640	9885	4749	3728	5984
additief :	4978	1489	3880	0503	8658	8433	5999	2863	3808	4801
encicode :	9938	2177	1191	1401	2593	5073	4774	6502	6526	9785

Verscheidene bronnen kunnen voor de benodigde cijferreeksen worden aangewend. Het beste is om ze

* De Duitsers hebben een aansprekende naam voor zo'n reeks willekeurige cijfers: Zahlenwurm.

speciaal voor dit doel te vervaardigen. Men moet dan wel zorgen dat de reeksen zo goed mogelijk random (willekeurig) zijn. Als de reeks echt volkomen willekeurig is, spreekt men van een *one-time pad*. Oververcijfering hiermee maakt het cryptogram ook werkelijk onbreekbaar. Maar, zoals de benaming al aangeeft, dan mag de reeks uitsluitend eenmalig voor vercijfering worden benut. Bij herhaald gebruik vervalt de theoretische onbreekbaarheid, zoals de Russische KGB heeft gemerkt. Toen tijdens de Tweede Wereldoorlog door fouten in de productie van de sleutelboeken delen van de sleutelreeks herhaald in het gedistribueerde sleutelmateriaal voorkwamen, konden Amerikaanse codebrekers als gevolg hiervan delen van de onderliggende code blootleggen in een operatie die VENONA is gedoopt^[10]*. Met grote gevolgen, want zo kwam men onder meer de spionnen op hoge posities in de Engelse geheime dienst op het spoor, de zogenaamde Cambridge Five[†].

Het nadeel van het echte one-time pad is de grote hoeveelheid sleutelmateriaal die hiermee is gemoeid. Er zijn omstandigheden waaronder dit niet gemakkelijk uitgewisseld en aangevuld kan worden. In die gevallen kunnen openbaar verkrijgbare bronnen voor de cijferreeksen worden gebruikt, zoals een logaritmentafel of statistische publicaties. Richard Sorge was in de jaren rond het begin van de Tweede Wereldoorlog een van de belangrijkste spionnen voor Moskou, met het Verre Oosten als werkterrein. Tot aan zijn ontmaskering in 1941 leverde hij buitengewoon waardevolle inlichtingen, die bijvoorbeeld een rol hebben gespeeld bij de totstandkoming van het Hitler-Stalin-pact in 1939. Als bron voor oververcijfering gebruikte hij het Statistisches Jahrbuch für das Deutsche Reich, editie 1935. Een speciale indicatorgroep, die in het bericht werd verstopt, gaf aan op welke plaats in het boek de cijferreeks begon^[11].

* Zie ook de internet site van de NSA op <http://www.nsa.gov> waar het een en ander uit de VENONA-operatie kan worden bekeken.

† De als eerste ontmaskerde Kim Philby heeft de spion Bill Haydon in de roman *Tinker, Tailor, Soldier, Spy* van John Le Carré als pendant. De gelijknamige filmserie van de BBC is ook een aanrader voor geïnteresseerden in de 'tradecraft' van de spionnenwereld.

3.5.3 Bijzondere codes

Tot slot in het kort nog enkele bijzondere vormen van codes. Het is slechts een greep uit de talloze mogelijkheden.

Boekcode

In de *boekcode* vertijft men woorden en/of letters van de tekst met een boek als sleutel. De correspondenten moeten allen over een exemplaar van hetzelfde boek beschikken, zelfde druk, zelfde editie. Een van de gehanteerde methoden is om de letters van het bericht te coderen met bladzijde-, regel- en letternummer. In het voorbeeld van [figuur 3.17](#) is Jules Verne's *De Reis om de Wereld in 80 Dagen* hiervoor genomen*.

O 122-7-14	L 88-5-2	T 75-3-7	A 37-1-3	D 34-7-7
R 95-19-16	E 161-5-10	O 79-5-2	L 43-6-6	E 50-1-18
D 110-2-9	I 161-17-20	T 72-14-16	L 49-2-23	U 65-2-3
E 80-25-37	D 132-2-1		E 22-6-2	G 65-22-23
	T 133-37-17			D 204-3-5
				E 217-11-7
				N 158-12-30

Figuur 3.17 Boekcode met bladzijde, regel en letterpositie

Een andere mogelijkheid is het gebruik van een woordenboek. Bladzijde, kolom en regel geven aan welk woord is bedoeld. Om de cryptoanalyse te bemoeilijken kan men de bladzijden willekeurig hernoemen,

* Nederlandse vertaling van Uitgeversmaatschappij Elsevier, 16^e druk, 1950. Wie de codering wil nagaan zal een exemplaar dienen op te sporen.

hetgeen het teruglezen natuurlijk wel bemoeilijkt. Zulke codes brengen bij het vercijferen en ontcijferen een hoop telwerk met zich mee en dus veel kans op fouten. Ze zijn daarom in het metier niet geliefd. Het voorbeeld maakt denkelijk wel duidelijk hoe moeizaam dit systeem werkt. Ook bestaat het gevaar dat een (vermoeide) codist reeds eerder gebruikte coderingen gaat herhalen, waardoor het systeem sterk aan kracht inboet. Boekcodes zijn zeker niet onbreekbaar gebleken en in een enkel geval kon zelfs het als sleutel gebruikte boek worden achterhaald.

Metereologische code

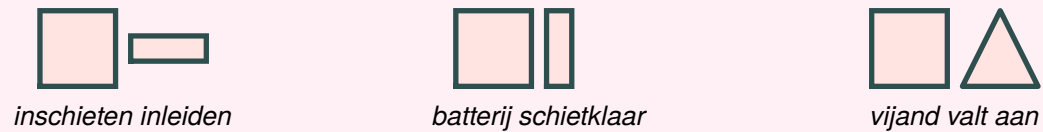
Metereologische codes zijn ontwikkeld voor het efficiënt coderen van de weersgesteldheid. De Baudot code bevat een reeks weercodes (zie de afbeelding [op pagina 459](#)). In de Tweede Wereldoorlog moesten de op de Atlantische Oceaan opererende Duitse onderzeeboten op gezette tijden de weersgesteldheid melden. Die gegevens waren van belang voor op handen zijnde militaire operaties*. Omdat de radiocommunicatie van de onderzeeërs kon worden uitgepeild, moesten de weersignalen kort gehouden worden. Met de *Wetterkurzschlüssel* comprimeerde men gegevens als windsnelheid, bewolking, zeewater- en luchttemperatuur, enz., tot slechts enkele letters. Daardoor kon het versturen van een weerbericht tot een luttel aantal seconden worden teruggebracht.

Bodemsignalen

Een wat ongebruikelijke code zijn de bodemsignalen zoals die bijvoorbeeld aan het front in de Eerste Wereldoorlog werden gebruikt om berichten door te geven aan verkenningsvliegtuigen. Lappen ter grootte

* Het weer in West-Europa komt nu eenmaal vaak van de Atlantische Oceaan in de vorm van depressies. Door gebrek aan actuele gegevens misten de Duitsers begin juni 1944 de korte opklaring in een reeks van depressies, die door de geallieerden wel gezien werd en zodoende voor de landingen in Normandië kon worden benut.

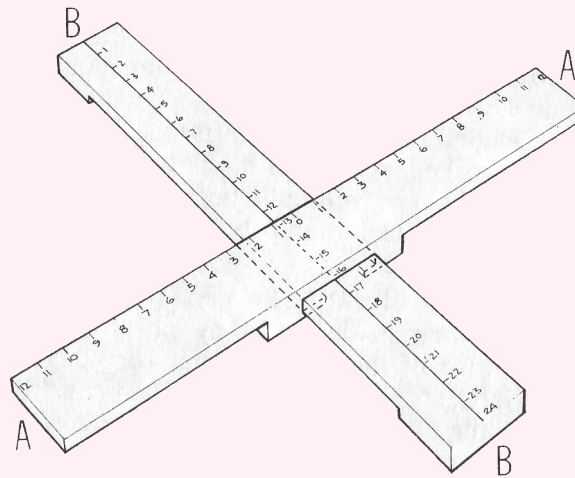
van enkele meters werden op de grond uitgelegd. **Figuur 3.18** geeft een paar voorbeelden van door de Duitse verkenningsdienst waargenomen signalen in gebruik bij Franse geschutsstellingen^[12].



Figuur 3.18 Bodemsignalen Franse artillerie uit 1917

Stosslinie

Posities op landkaarten werden in het algemeen niet op de normale wijze in graden noorderbreedte en oosterlengte in de berichten doorgegeven. De Duitse generaal Erwin Rommel codeerde voor zijn tankbrigades kaartposities met behulp van de zogenaamde *Stosslinie* en het apparaat in **figuur 3.19**^[13].



Figuur 3.19 Stosslinie methode voor kaartcoördinaten

Schaal B-B werd op een vooraf bepaalde lijn op de kaart gelegd en de verschuifbare schaal A-A zodanig gepositioneerd dat deze aan de linker- of rechterkant langs de gewenste plaats op de kaart viel. De langs de schalen A en B afgelezen waarden zijn dan de gecodeerde kaartcoördinaten. Omdat de *Stosslinie* voor elke belangrijke operatie opnieuw wordt vastgesteld, kan de tegenstander uit de onderschepte berichten de opgegeven posities niet zonder meer bepalen.

3.6 Referenties

- [1] L. Kruh (1977). The Churchyard Ciphers. *Cryptologia*, I, 372–375.
- [2] K. de Leeuw and H. van der Meer (1993b). A Homophonic Substitution in the Archives of the Last Great Pensionary of Holland. *Cryptologia*, XVII(3), 225–236.
- [3] D. James (1969). *Ché Guevara*. Stein and Day Publ..
- [4] S. K. Kullback (1976). *Statistical Methods in Cryptanalysis*. Aegean Park Press.
- [5] C. David (1996). A World War II German Army Field Cipher and How We Broke It. *Cryptologia*, XX(1), 55–75. De tekst werd opgesplitst door verdeling over twee regels, de digrammen verticaal samengenomen en tweemaal achtereen gesubstitueerd.
- [6] G. J. Painvin (1961). Decryptement des radiogrammes Allemands 1914–1918, Exposé des methodes de recherche. .
- [7] A. Meister (1902). *Die Anfänge der Modernen Diplomatischen Geheimschrift, Beitrage zur Geschichte der Italienischen Kryptographie des XV. Jahrhunderts*. Verlag Ferdinand Schöningh.
A. Meister (1906). *Die Geheimschrift im Dienste der Päpstlichen Kurie, Von ihren Anfängen bis zum Ende des XVI. Jahrhunderts*. Verlag Ferdinand Schöningh.
- [8] J. P. Devos (1950). *Les Chiffres de Philippe II (1555–1598) et du Despacho Universal durant le XVIIe Siècle*. Koninklijke Belgische Academie, Koninklijke Commissie voor Geschiedenis.
- [9] C. B. Barto (1933). *Economie en Techniek van Codes en Code-condensors*. PhD thesis, Technische Universiteit Delft.
- [10] R. L. Benson and M. Warner Eds. (1996). *Venona, Soviet Espionage and the American Response 1939–1957*. Aegean Park Press.

- [11] Anon. (?). The Sorge Spy Ring. Fort Holabird, Maryland.
- [12] Funkerdienst I (1917). Erfahrungen im Flieger-Stör- und -Warndienst an der Westfront. Hauptquartier Mézières-Charleville.
- [13] H. Skillen (1989). *Spies of the Airwaves - A History of Y Sections During the Second World War*. ASTRA Design and Print, Middlesex. p.58

4 Polyalfabetische systemen

4.1	Polyalfabetische substitutie	118
4.1.1	Porta-substitutie	120
4.1.2	Vigenère-substitutie	121
4.1.3	Beaufort-substitutie	124
4.1.4	Vigenère, Variant en Beaufort	125
4.1.5	Gepermuteerd alfabet	125
4.1.6	Nihilist-substitutie	130
4.2	Progressieve polyalfabetische substitutie	132
4.2.1	Alfabetprogressie	133
4.2.2	Periodieke progressie	134
4.3	Autoclaafstelsel	139
4.3.1	Autoclaaf met cijfertext	140
4.3.2	Autoclaaf met plaintext	141
4.3.3	Wheatstone-cryptograaf	142
4.4	Multiplexstelsel	144
4.5	Cryptografische rotor	149
4.6	Lineaircongruente reeks	153
4.6.1	MacLaren-Marsaglia-generator	155
4.7	One-time pad	157
4.8	Referenties	160

4.1 Polyalfabetische substitutie

In dezelfde periode waarin de nomenclatuur tot bloei kwam, werd ook de *polyalfabetische substitutie* ontwikkeld. De zwakte van alle monoalfabetische systemen is de onveranderlijkheid van de vervanging: eens E wordt P dan altijd E wordt P. De frequentiedistributie van de letters in de klaartekst komt onveranderd terug in het cryptogram en vormt het voornaamste aanvalspunt in de cryptoanalyse. Voor n -gram substituties en voor codes geldt mutatis mutandis hetzelfde, zij het dat door het grotere alfabet (het aantal symbolen c.q. codegroepen) de effecten geringer zijn en dus voor de cryptoanalyse meer materiaal nodig zal zijn.

Deze zwakte wordt een stuk minder wanneer de substitutie wisselt. Met meerdere, van elkaar verschillende substitutiealfabetten kan de letterdistributie beter worden verdoezeld. Hierin ligt de kracht van de polyalfabetische substitutie. De vercijfering verschilt per letter van de klaartekst op een manier die gekoppeld is aan de plaats in de tekst.

Als zo'n polyalfabetisch geheimschrift zoveel beter is dan de (in wezen monoalfabetische) nomenclatuur en ook nog eens in dezelfde periode tot ontwikkeling kwam, waarom heeft deze de nomenclatuur dan niet verdrongen? De belangrijkste oorzaken zijn van praktische aard. Vercijferen met een polyalfabetisch systeem is bewerkelijk. Men moet het hoofd er goed bij houden, want een fout in het vercijferschema brengt de ontcijferaar snel van de wijs. Het vercijferwerk gaat niet alleen langzamer dan bij een nomenclatuur doordat het proces ingewikkelder is, het gaat ook langzamer omdat met de nomenclatuur woorden en uitdrukkingen in een keer in hun geheel worden omgezet. Het cryptogram zal in de nomenclatuur daarom als regel compacter zijn.

Het eenvoudigste polyalfabetisch geheimschrift is de periodieke polyalfabetische substitutie, waarin een beperkte serie alfabetsubstituties herhaaldelijk voor vercijfering wordt gebruikt. Begonnen zal worden met systemen waarin de verschillende substituties met behulp van sleutelletters worden afgeleid van het gewone alfabet, dat wil zeggen met de letters in de normale volgorde. Drie varianten komen aan bod, te

weten de Porta, de Vigenère en de Beaufort. Als voorbeeld nemen we de tekst

VYAND NADERT WATERLINIE STOP FORT ASPEREN PARAAT

Het sleutelwoord zal zijn PENTAGON. Het substitutieschema wordt verkregen door het sleutelwoord en de klaartekst onder elkaar te schrijven en het sleutelwoord net zo vaak te herhalen als nodig.

```
klaartekst : VYAND NADERT WATERLINIE STOP FORT ASPEREN PARAAT
sleutel   : PENTA GONPEN TAGONPENTA GONP ENTA GONPENT AGONPE
```

De periode wordt bepaald door de lengte van het sleutelwoord. Hier bedraagt de periode 8, nog eens gedemonstreerd door een andere opstelling.

```
klaartekst : VYANDNAD ERTWATER LINIESTO PFORTASP ERENPARA AT
sleutel   : PENTAGON PENTAGON PENTAGON PENTAGON PENTAGON PE
```

Een handig alternatief is om meteen in de gebruikelijke groepen van vijf letters te werken.

```
klaartekst : VYAND NADER TWATE RLINI ESTOP FORTA SPERE NPARA AT
sleutel   : PENTA GONPE NTAGO NPENT AGONP ENTAG ONPEN TAGON PE
```

Onderscheid moet worden gemaakt tussen de periode, dat wil zeggen het zich herhalend patroon van substituties, en het aantal verschillende substitutiealfabetten. In het gegeven voorbeeld bedraagt de periode 8, doch er zijn slechts 7 verschillende substituties in het spel. Dit verschil ontstaat doordat in het sleutelwoord PENTAGON de letter N tweemaal voorkomt. Gelijke sleutelletters bewerkstelligen gelijke substituties. Omdat het woord PENTAGON slechts 7 verschillende letters bevat, zijn er ook maar 7 verschillende substituties gerangschikt in een zich herhalend patroon van lengte 8.

4.1.1 Porta-substitutie

De Porta substitutie is genoemd naar Giambattista della Porta, een Italiaan die leefde van 1535–1615. In dit systeem wordt het alfabet in twee gelijke stukken geknipt, de ene helft bevat de letters A–M en de andere de letters N–Z. Deze helften worden gepaard, zodat een letter uit de eerste helft steeds een letter uit de tweede helft van het alfabet tegenover zich vindt. Substitutie bestaat uit het vervangen van een klaartekstletter door de overstaande letter in de andere helft. Met 26 letters in het alfabet telt elke helft 13 letters, waaruit 13 verschillende paren te maken zijn. De paren worden aangeduid met de letters van het alfabet, twee opeenvolgende letters bepalen telkens één substitutie.

In [tabel 4.1](#) is dit vercijferschema uitgewerkt. Voor het vercijferen van een letter uit de reeks A–M zoekt men deze op in de rij aan de top van het schema en vervangt deze door de letter op de overeenkomstige positie in de rij die vooraan de gekozen sleutelletter heeft. Bij het vercijferen van een letter uit N–Z zoekt men in diezelfde rij de letter op en vervangt deze door de letter op de overeenkomstige positie aan de top. Voor ontcijferen gaat men op dezelfde wijze te werk. Uitgaande van de cryptogramletter geeft eenzelfde bewerking de klaartekstletter terug.

Porta-vercijfering van het voorbeeld leidt tot het volgende resultaat

```
klaartekst : VYANDNAD ERTWATER LINIESTO PFORTASP ERENPARA AT  
sleutel : PENTAGON PENTAGON PENTAGON PENTAGON PENTAGON PE  
cryptogram : BJTEQKUW YCAANDYL SXHRRMI IUIIGQLJ YCXECQKT UE
```

In het voorgaande is opgemerkt dat als gevolg van de twee N'en in sleutel PENTAGON slechts 7 i.p.v. 8 substituties gebruikt worden. De combinatie van twee sleutelletters tot één substitutie in de Porta maakt dat O en P in de sleutel dezelfde substitutie bewerkstelligen. Deze sleutel gebruikt dus maar 6 van 13 mogelijke substituties.



Porta

	A	B	C	D	E	F	G	H	I	J	K	L	M	
AB	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AB
CD	O	P	Q	R	S	T	U	V	W	X	Y	Z	N	CD
EF	P	Q	R	S	T	U	V	W	X	Y	Z	N	O	EF
GH	Q	R	S	T	U	V	W	X	Y	Z	N	O	P	GH
IJ	R	S	T	U	V	W	X	Y	Z	N	O	P	Q	IJ
KL	S	T	U	V	W	X	Y	Z	N	O	P	Q	R	KL
MN	T	U	V	W	X	Y	Z	N	O	P	Q	R	S	MN
OP	U	V	W	X	Y	Z	N	O	P	Q	R	S	T	OP
QR	V	W	X	Y	Z	N	O	P	Q	R	S	T	U	QR
ST	W	X	Y	Z	N	O	P	Q	R	S	T	U	V	ST
UV	X	Y	Z	N	O	P	Q	R	S	T	U	V	W	UV
WX	Y	Z	N	O	P	Q	R	S	T	U	V	W	X	WX
YZ	Z	N	O	P	Q	R	S	T	U	V	W	X	Y	YZ
	A	B	C	D	E	F	G	H	I	J	K	L	M	

Tabel 4.1 Porta vercijferschema

4.1.2 Vigenère-substitutie

In het Porta-systeem zijn slechts 13 vervangende substituties, terwijl het er 26 hadden kunnen zijn wanneer alle letters van het alfabet als vervanger mogen optreden. De Vigenère-substitutie waaraan de Fransman Blaise de Vigenère (1523–1596) zijn naam heeft gegeven, kent deze beperking niet*. Elke letter kan door elk van de 26 letters van het alfabet worden vervangen. Het gemakkelijkst valt de werking uit te leggen aan de hand van de Vigenère-tabel, afgebeeld in [tabel 4.2](#).

Vercijferen gaat als volgt in z'n werk. De letter die moet worden vercijferd wordt opgezocht in de rij aan

* Het door Vigenère in zijn *Traicté des Chiffres* (1585) beschreven systeem is overigens ingewikkelder – en sterker – dan de vereenvoudigde vorm waaraan het etiket Vigenère is blijven hangen.



Vigenère

sleutel	letters van de klaartekst																										
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	

de top van de tabel. Neem de eerste letter van het voorbeeld, de V. Deze krijgt de sleutelletter P, die aan de linkerkant wordt opgezocht. In de kolom V en in de rij P wordt op het snijpunt de cryptogramletter K gevonden. Voor het ontcijferen wordt de rij met de sleutelletter opgezocht, vervolgens loopt men daarin naar de cryptogramletter en de ontcijferde letter staat in die kolom aan de top van de tabel.

Het cryptogram luidt in Vigenère-vercijfering als volgt

```
klaartekst : VYANDNAD ERTWATER LINIESTO PFORTASP ERENPARA AT
sleutel : PENTAGON PENTAGON PENTAGON PENTAGON PENTAGON PE
cryptogram : KCNGDTOQ TVGPAZSE AMABEYHB EJBKTGGC TVRGPGFN PX
```

Kijkend naar de Vigenère-tabel ziet men dat het systeem in feite uit een reeks Caesar-substituties bestaat. De sleutelletter bepaalt over hoeveel posities de letter wordt verschoven: sleutelletter A schuift over 0 posities, sleutelletter B over 1 positie, sleutelletter C over 2 posities, enz. De eerste letter van elk verschoven alfabet uit de body van de tabel is tevens de sleutelletter voor die rij. Merk op dat Caesar-sleutel A hier wel van nut is, dit in tegenstelling tot de op zichzelf staande Caesar-vercijfering uit [hoofdstuk 3](#).

Met periode p , sleutel $K = k_0 k_1 \dots k_{p-1}$, klartekst $P = p_0 p_1 \dots$, cryptogram $C = c_0 c_1 \dots$ en Caesar-verschuiving C^k uit [vergelijking \(3.1\)](#) kunnen deze bewerkingen als volgt geformuleerd worden. Let op, in deze en soortgelijke berekeningen geldt A=0, B=1 ... Z=25 en wordt het resultaat module 26 gereduceerd.

$$c_i = E(p_i) = C^{k_i \bmod p} p_i \quad (4.1)$$

Behalve in tabelvorm kan de Vigenère ook compact worden geïmplementeerd door middel van twee langs elkaar schuivende linealen. Wanneer op een van beide het alfabet tweemaal wordt gezet, vindt men altijd twee tegenoverstaande posities. In deze vorm spreekt men wel van de Saint Cyr-lineaal, naar de Franse militaire academie van Saint Cyr waar deze omstreeks 1880 in gebruik was. Hieronder staan de twee linealen zo opgesteld dat ze corresponderen met sleutelletter E. De klartekstletter op de bovenste lineaal wordt vercijferd tot de tegenoverstaande letter op de onderste.

ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZ

Figuur 4.1 Saint Cyr-lineaal



Beaufort

In de Vigenère zijn vercijfering en ontcijfering niet symmetrisch, vercijfering gaat een andere route dan ontcijfering. Het is daarom mogelijk om ze van rol te laten wisselen. Vercijferen op de manier die bij de Vigenère ontcijfert en ontcijferen op de manier die bij Vigenère vercijfert. Deze werkwijze wordt aangeduid als de *Variant Vigenère*. Het zelfde cryptogram nu in Variant-vercijfering

```
klaartekst : VYANDNAD ERTWATER LINIESTO PFORTASP ERENPARA AT
sleutel   : PENTAGON PENTAGON PENTAGON PENTAGON PENTAGON PE
cryptogram : GUNUDHMQ PNGDANQE WEAPEMFB ABBYTUEC PNRUPUDN LP
```

4.1.3 Beaufort-substitutie

Als laatste van de eenvoudige polyalfabeten een behandeling van de *Beaufort-substitutie*. Dit geheimschrift is in 1857 bedacht door de Engelse admiraal Sir Francis Beaufort (ook bekend van de gelijknamige schaal voor de windsnelheid)*.

De Beaufort maakt gebruik van dezelfde soort tabel als de Vigenère, maar met een andere vulling zoals te zien is in [tabel 4.3](#). In de Beaufort staat het alfabet in teruglopende volgorde op de rijen. Het gevolg hiervan is dat vercijferen en ontcijferen dezelfde weg volgen. Voor vercijferen zoekt men de rij met de sleutelletter op en loopt daarin naar de kolom met de klartekstletter, de cryptogramletter staat in die kolom bovenaan de tabel. Voor ontcijferen volgt men dezelfde procedure maar loopt nu in de rij met de

* Zoals veel geheimschriften, is dit een heruitvinding van een systeem dat al in 1710 door Giovanni Sestri is beschreven.

sleutelletter naar de cryptogramletter en vindt de ontcijferde letter bovenaan. Deze overeenkomst tussen vercijferen en ontcijferen heeft tot gevolg dat een op de Beaufort gebaseerde machine met slechts één mechanisme beide operaties kan uitvoeren en dus geen aparte circuits voor vercijferen en ontcijferen nodig heeft.

In formule levert dit dezelfde uitdrukking als [vergelijking \(4.1\)](#) maar nu met de Caesar-verschuiving voor een teruglopend alfabet, dat wil zeggen die van [vergelijking \(3.3\)](#). Daardoor geldt voor klaartekstletter P , sleutel K en resultaat C

$$P + C = K \quad (4.2)$$

Het cryptogram in Beaufort-vercijfering luidt

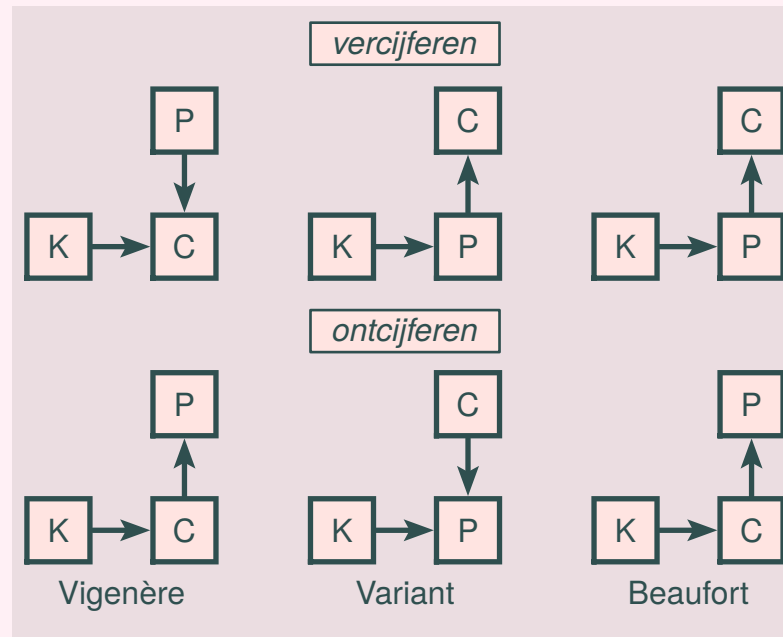
```
klaartekst : VYANDNAD ERTWATER LINIESTO PFORTASP ERENPARA AT  
sleutel : PENTAGON PENTAGON PENTAGON PENTAGON PENTAGON PE  
cryptogram : UNGXTOK LNUXANKW EWALWOVZ AZZCHGWY LNJGLGXN PL
```

4.1.4 Vigenère, Variant en Beaufort

Om duidelijk te maken in welk opzicht de operaties in de drie verwante systemen Vigenère, Variant Vigenère en Beaufort van elkaar verschillen zijn ze in [figuur 4.2](#) naast elkaar gezet.

4.1.5 Gepermuteerd alfabet

In plaats van het alfabet in de normale volgorde A-Z kan het in deze systemen worden vervangen door een gepermuteerd alfabet. Een gunstige complicatie is dat het verschil maakt welk alfabet gepermuteerd wordt. Het gepermuteerde alfabet kan het *klaaralfabet* zijn, waarin de letters van de klaartekst worden



Figuur 4.2 Vigenère, Variant en Beaufort vergeleken

sleutel	letters van de klaartekst respectievelijk cijfertekst																										
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	
A	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A
B	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B
C	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C
D	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D
E	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E
F	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F
G	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G
H	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H
I	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I
J	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J
K	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K
L	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L
M	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M
N	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N
O	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O
P	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P
Q	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q
R	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R
S	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S
T	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T
U	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U
V	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V
W	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W
X	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X
Y	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y
Z	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	

opgezocht. Het kan ook het *cijferalfabet* zijn, waarin de overeenkomstige letters van het cryptogram zich bevinden. En zelfs kunnen allebei worden gepermuteerd, al dan niet op dezelfde manier. Om niet steeds een volledige tabel te hoeven geven, wordt de voorgaande in [figuur 4.1](#) geïntroduceerde opstelling van de Saint-Cyr-lineaal gebruikt. In de volgende voorbeelden staat het klaaralfabet steeds bovenaan en het cijferalfabet onderaan.

Er zijn dus twee alfabetten die onafhankelijk van elkaar gepermuteerd kunnen worden en het is zeker niet om het even welke van de twee die bewerking heeft ondergaan. Bij de behandeling van de cryptanalyse van deze systemen wordt daar nog uitvoerig op teruggekomen. In volgorde van toenemende sterkte van het geheimschrift zijn de mogelijkheden als volgt.

- *Klaaralfabet gepermuteerd*

```
klaaralfabet : TRAPEZIUMBCDFGHJKLNOQSVWXY
cijferalfabet : ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

Deze vorm komt overeen met een Vigenère-tabel waarvan de topregel het gepermuteerde alfabet vertoont. De body van de tabel blijft onveranderd. De eerste letter van elke rij is sleutelletter en die staan van boven naar beneden in de normale alfabetische volgorde.

```
klaartekst : VYANDNAD ERTWATER LINIESTO PFORTASP ERENPARA AT
sleutel   : PENTAGON PENTAGON PENTAGON PENTAGON PENTAGON PE
cryptogram : LDPLLYQY TFNQCGSO GKFZEBOG SQGUAIJQ TFRLDIPP RE
```

- *Cijferalfabet gepermuteerd*

```
klaaralfabet : ABCDEFGHIJKLMNOPQRSTUVWXYZ
cijferalfabet : TRAPEZIUMBCDFGHJKLNOQSVWXY
```

Deze vorm komt overeen met een Vigenère-tabel waarvan de topregel het normale alfabet vertoont en de body de verschoven gepermuteerde alfabetten bevat. De eerste letter van elke rij is wederom de sleutelletter, maar deze sleutelletters staan nu van boven naar beneden in de gepermuteerde volgorde. Als men voor het gemak de sleutelletters in de normale volgorde zet door sorteren van de rijen, raakt men de regelmaat in de verschuiving tussen de opeenvolgende rijen natuurlijk kwijt. Maar het is volstrekt legitiem om de tabel op die manier in te richten. Wel moet daarbij worden opgemerkt dat de keuze in de volgorde van de sleutelletters bij de hierna te behandelen progressieve systemen invloed heeft op het resultaat.

```
klaartekst : VYANDNAD ERTWATER LINIESTO PFORTASP ERENPARA AT  
sleutel : PENTAGON PENTAGON PENTAGON PENTAGON PENTAGON PE  
cryptogram : XANGZTOS USDVAIWB HFZMIZFI NBILSGDU USVGLGCN PW
```

- *Klaartekst- en cijferalfabet hetzelfde gepermuteerd*

```
klaaralfabet : TRAPEZIUMBCDFGHJKLNOQSVWXY  
cijferalfabet : TRAPEZIUMBCDFGHJKLNOQSVWXY
```

Dit is in wezen dezelfde Vigenère-tabel als in [tabel 4.2](#), met dien verstande dat deze gebaseerd is op een alfabet in de gepermuteerde volgorde in plaats van de normale.

```
klaartekst : VYANDNAD ERTWATER LINIESTO PFORTASP ERENPARA AT  
sleutel : PENTAGON PENTAGON PENTAGON PENTAGON PENTAGON PE  
cryptogram : YPQNGZSP UZNWEGWO QCCIIMOD IKDRAJHS UZVNZJQQ ZE
```

- *Klaar- en cijferalfabet afzonderlijk gepermuteerd*

```
klaaralfabet : TRAPEZIUMBCDFGHJKLNOQSVWXY  
cijferalfabet : ZESKANTIGBCDFHJLMOPQRUVWXY
```

Deze variant is vergelijkbaar met de vorige, maar nu is het alfabet aan de top voorzien van een andere permutatie dan in de body.

```
klaartekst : VYANDNAD ERTWATER LINIESTO PFORTASP ERENPARA AT  
sleutel : PENTAGON PENTAGON PENTAGON PENTAGON PENTAGON PE  
cryptogram : JZIXLZQM VSNKTGUT BIWFGKX UHXIACFG VSBXICPI RE
```

Vanzelfsprekend behoeft men al deze variaties niet tot de Vigenère te beperken. Echter voor de Beaufort brengt op dezelfde manier variëren het nadeel met zich mee dat de eenvoud van de vercijfer-ontcijferoperatie verloren gaat.

4.1.6 Nihilist-substitutie

De Nihilist-substitutie is een polyalfabeet met cijfers in plaats van letters. Evenals de gelijknamige transpositie ([pagina 72](#)) is de naam van dit geheimschrift ontleend aan de beweging van de Nihilisten aan het eind van de 19^e-eeuw. Door de beperkingen op de parameters van het systeem is het veel zwakker dan een gewone Vigenère.

Voor het coderen worden de letters van het bericht en van de sleutel omgezet in groepen van twee cijfers. Deze dinomes worden ontleend aan de coördinaten van een vierkant waarin het alfabet is ingeschreven, al of niet gemodificeerd met behulp van een sleutelwoord. In [figuur 4.3](#) staat links een ongemodificeerd alfabetvierkant en rechts een vierkant gebaseerd op een sleutelwoord.

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

sleutel normaal

	1	2	3	4	5
1	K	R	E	M	L
2	I	N	A	B	C
3	D	F	G	H	O
4	P	Q	S	T	U
5	V	W	X	Y	Z

sleutel KREMLIN

Figuur 4.3 Nihilist substitutie alfabet

Met de omzetting van letters naar dinomes – de coördinaten in de volgorde (rij, kolom) genomen – wordt de M bijvoorbeeld vertaald naar 32 (vierkant links) en naar 14 (vierkant rechts). Na vertaling van zowel de klaartekst als de periodieke sleutel in dinomes, worden deze twee aan twee opgeteld. Is de sleutelletter een N dan wordt $M + N$ vercijferd als $32 + 33 = 65$ (vierkant links) en als $14 + 22 = 36$ (vierkant rechts). Het optellen geschiedt met overdracht voor de tientallen, maar een eventueel honderdtal wordt weggelaten. Dus $U + U$ levert $45 + 45 = 90$, maar let op $Z + Z$ wordt $55 + 55 = 10$ in plaats van 110.

Hieronder is het voorbeeld vercijferd in de Nihilist-substitutie, zowel met de linker- als met de rechter alfabet sleutel.

ABCDEF–sleutelvierkant														
klaartekst :	V	Y	A	N	D	N	A	D	E	R	T	W	A	T
	:	51	54	11	33	14	33	11	14	15	42	44	52	11
sleutel :	P	E	N	T	A	G	O	N	P	E	N	T	A	G
	:	35	15	33	44	11	22	34	33	35	15	33	44	11
cryptogram :	86	69	44	77	25	55	45	47	50	57	77	96	22	66
groepen :	86694	47725	55454	75057	77962	26649	75							

KREMLIN-sleutelvierkant																	
klaartekst :	V	Y	A	N	D	N	A	D	E	R	T	W	A	T	E	R	
	:	51	54	23	22	31	22	23	31	13	12	44	52	23	44	13	12
sleutel :	P	E	N	T	A	G	O	N	P	E	N	T	A	G	O	N	
	:	41	13	22	44	23	33	35	22	41	13	22	44	23	33	35	22
cryptogram :	92	67	45	66	54	55	58	53	54	25	66	96	46	77	48	34	
groepen :	92674	56654	55585	35425	66964	67748	34										

4.2 Progressieve polyalfabetische substitutie

In de voorbeelden is voor de vercijfering steeds een betrekkelijk kort sleutelwoord gebruikt. Bij de behandeling van de cryptanalyse van polyalfabetische systemen zal blijken dat dit niet zo gunstig is. Lange sleutels bieden meer bescherming dan korte. Een mogelijkheid is het nemen van een sleutelzin in plaats van een enkel woord. Een sleutelzin kan tamelijk veel letters tellen en toch gemakkelijk te onthouden zijn. Geliefd waren eertijds bekende bijbelteksten of versregels uit Vergilius' *Anæas*. In de tijd van de Renaissance kon men zo sleutelzinnen tegenkomen als

VIRTUTI OMNIA PARENT *of* FUNDAMENTA EIUS IN MONTIBUS SANCTUS

Hoe goed lange zinnen ook lijken, ze zijn kwetsbaar juist omdat ze uit normale tekst bestaan. Die kwetsbaarheid kan door een cryptanalist worden geëxploiteerd door te proberen gelijktijdig klaartekst en sleutel te vinden. De één fungeert dan als een controle op de ander. Onzinnige, willekeurige sleuteltekst valt daarom te preferen maar is nu juist weer moeilijk te onthouden.

Een manier om aan het boven geschetste dilemma te ontsnappen en met eenvoudige middelen toch een lange, willekeurig ogende sleutel te genereren, is het combineren van twee of meer korte sleutels. In het navolgende voorbeeld is sleutel PENTAGON gecombineerd met een tweede sleutel CIA. Daar deze twee

sleutelwoorden zo gekozen zijn dat hun lengte (8 en 3) geen factor gemeen hebben, wordt door combinatie een periode van 24 letters verkregen. De gecombineerde sleutel is in feite een Vigenère-vercijfering van sleutel PENTAGON met sleutel CIA, waarna het resulterend cryptogram gebruikt wordt als sleutel voor de klaartekst.

```
deelsleutel 1 : PENTAGONPENTAGONPENTAGON
deelsleutel 2 : CIACIACIACIACIACIACIACIA
combinatiesleutel : RMNVIGQVPGVTCOOPXEPBAIWN
```

```
klaartekst : VYANDNADERTWATERLINIESTO PFORTASPERENPARAAT
sleutel : RMNVIGQVPGVTCOOPXEPBAIWN RMNVIGQVPGVTCOOPXE
cryptogram : MKNILTQYTXOPCHSGIMCJEAPB GRBMBGIKTXZGROFPXX
```

Ook hier moet men beseffen dat dit gebruik van deelsleutels in combinatie minder sterk is dan wanneer een compleet willekeurige sleutel van dezelfde lengte zou zijn toegepast. Voor betere methodes om lange sleutels te maken staan de navolgende progressieve en autoclaafsystemen ter beschikking.

4.2.1 Alfabetprogressie

Het eenvoudigste progressieve systeem gebruikt als sleutel de achtereenvolgende letters van het alfabet. Er zijn 26 variaties omdat de beginletter vrijelijk mag worden gekozen. Het zal duidelijk zijn dat dit een polyalfabetische substitutie met periode 26 geeft. De progressie in het navolgende voorbeeld begint bij het F-alfabet. Na elke vercijferstap wordt het volgende sleutelalfabet genomen.

```
klaartekst : VYANDNADERTWATERLINIESTOPF ORTASPERENPARAAT
sleutel : FGHIJKLMNOPQRSTUVWXYZABCDE FGHIJKLMNOPQRSTU
cryptogram : AEHVMXLPRFIMRLXLGEKGDSUQSJ TXAIBZPDRBEQISTN
```

Een beperking tot alleen het direct opvolgend sleutelalfabet is niet nodig, men kan ook telkens een paar sleutelletters overslaan en bijvoorbeeld elk derde sleutelalfabet nemen. Opnieuw wordt zo een periode 26 verkregen, maar nu met een *decimatie* van het alfabet. Een decimatie van het alfabet verkrijgt men door steeds hetzelfde aantal letters verder te tellen en net zo vaak rond te gaan totdat alle letters aan de beurt geweest zijn. Het gehele alfabet wordt alleen door de sleutel doorlopen als het aantal letters van het alfabet en de stapgrootte geen factor gemeen hebben, d.w.z. $\text{ggd}(\text{stapgrootte}, 26) = 1$, want anders ontstaan kortere cycli. Bijvoorbeeld met stapgrootte 2 ontstaan 2 cycli van elk 13 letters en met stapgrootte 13 cycli van 2 letters. Elke volgende cyclus moet dan steeds met een nog niet voorgekomen letter gestart worden. Een uitgewerkt voorbeeld van een decimatie staat in [tabel 6.11](#). Het volgende voorbeeld vercijfert met decimatie 3 en slaat dus bij elke vercijferstap drie sleutelletters over. De start ligt wederom bij het F-alfabet.

```
klaartekst : VYANDNADERTWATERLINIESTOPF  ORTASPERENPARAAT  
sleutel   : FILORUXADGJMPSVYBEHKNQWZC  FILORUXADGJMPSVY  
cryptogram : AGLBUHXDHXCIPLZPMMUSRIMKOH  TZE0JJBRHTYMGSVR
```

Het effect van de progressie in bovenstaande voorbeelden valt goed te zien in de opstelling van tabel ??, waar het cryptogram schuin door de kolomsgewijs opgestelde alfabetten heenloopt. De punten in deze tabel zijn bedoeld om het contrast met de vercijferde letters te vergroten; de letters op die plaatsen zijn dus weggelaten. In de voorbeelden is het alfabet steeds in normale volgorde van A tot Z gebruikt, maar natuurlijk kunnen in plaats daarvan ook gemengde alfabetvarianten gebruikt worden.

4.2.2 Periodieke progressie

De regelmatige progressie van de sleutelletters is erg voorspelbaar en daarom niet bijzonder sterk. Het valt dan ook aan te raden om de regelmaat op gezette momenten te doorbreken. Bijvoorbeeld door om de zoveel vercijferstappen een of of meer sleutelletters over te slaan, de sleutel bij de eerste sleutelletter

	V	Y	A	N	D	N	A	D	E		V	Y	A	N	D	N	A	D	E
A	V	Y	A	N	D	N	A	D	E		V	Y	A	N	D	N	A	D	E
B	W	Z	B	O	E	O	B	E	F		W	Z	B	O	E	O	B	E	F
C	X	A	C	P	F	P	C	F	G		X	A	C	P	F	P	C	F	G
D	Y	B	D	Q	G	Q	D	G	H		Y	B	D	Q	G	Q	D	G	H
E	Z	C	E	R	H	R	E	H	I		Z	C	E	R	H	R	E	H	I
F	A	D	F	S	I	S	F	I	J		A	D	F	S	I	S	F	I	J
G	B	E	G	T	J	T	G	J	K		B	E	G	T	J	T	G	J	K
H	C	F	H	U	K	U	H	K	L		C	F	H	U	K	U	H	K	L
I	D	G	I	V	L	V	I	L	M		D	G	I	V	L	V	I	L	M
J	E	H	J	W	M	W	J	M	N		E	H	J	W	M	W	J	M	N
K	F	I	K	X	N	X	K	N	O		F	I	K	X	N	X	K	N	O
L	G	J	L	Y	O	Y	L	O	P		G	J	L	Y	O	Y	L	O	P
M	H	K	M	Z	P	Z	M	P	Q		H	K	M	Z	P	Z	M	P	Q
N	I	L	N	A	Q	A	N	Q	R		I	L	N	A	Q	A	N	Q	R
O	J	M	O	B	R	B	O	R	S		J	M	O	B	R	B	O	R	S
P	K	N	P	C	S	C	P	S	T		K	N	P	C	S	C	P	S	T
Q	L	O	Q	D	T	D	Q	T	U		L	O	Q	D	T	D	Q	T	U
R	M	P	R	E	U	E	R	U	V		M	P	R	E	U	E	R	U	V
S	N	Q	S	F	V	F	S	V	W		N	Q	S	F	V	F	S	V	W
T	O	R	T	G	W	G	T	W	X		O	R	T	G	W	G	T	W	X
U	P	S	U	H	X	H	U	X	Y		P	S	U	H	X	H	U	X	Y
V	Q	T	V	I	Y	I	V	Y	Z		Q	T	V	I	Y	I	V	Y	Z
W	R	U	W	J	Z	J	W	Z	A		R	U	W	J	Z	J	W	Z	A
X	S	V	X	K	A	K	X	A	B		S	V	X	K	A	K	X	A	B
Y	T	W	Y	L	B	L	Y	B	C		T	W	Y	L	B	L	Y	B	C
Z	U	X	Z	M	C	M	Z	C	D		U	X	Z	M	C	M	Z	C	D
	A	E	H	V	M	X	L	P	R		A	G	L	B	U	H	X	D	H

Tabel 4.4 Progressie vanaf F met patroon 1-3-2

te herstarten, enz. Er zijn legio varianten denkbaar. Twee voorbeelden van zulke variaties op de regelmatige progressie. De serie sleutels wordt weer gestart bij het F-alfabet. In het eerste voorbeeld wordt de regelmaat onderbroken door na elke vijf vercijferingen een sleutelletter over te slaan.

```
klaartekst : VYAND NADER TWATE RLINI ESTOP FORTA SPERE NPARA AT
patroon   : 11112 11112 11112 11112 11112 11112 11112 11112 11
sleutel   : FGHIJ LMNOP RSTUV XYZAB DEF GH JKLMN PQRST VWXYZ BC
cryptogram : AEHVM YMQSG KOTNZ OJHNJ HWYUW OYCFN HFVJX ILXPZ BV
```

In het tweede voorbeeld is het sleutelwoord PENTAGON op de manier voor transpositiesleutels omgezet in de getalreeks 7-2-4-8-1-3-6-5. Deze getalreeks wordt gebruikt als het patroon dat de onderbreking van de progressie stuurt. Telkens na 7, 2, ... vercijferstappen wordt een sleutelletter overgeslagen.

```
klaartekst : VYANDNA DE RTWA TERLINIE S TOP FORTAS PEREN PARAAT
patroon   : 1111112 12 1112 11111112 2 112 111112 11112 111111
sleutel   : FGHIJKL NO QRST VWXYZABC E GHI KLMNOP RSTUV XYZABC
cryptogram : AEHVMXL QS HKOT OAOJHNJG W ZVX PZDGOH GWKYI MYQABV
```

Een andere manier om een wisselende progressie te krijgen wordt gevonden in het variëren van het stappenpatroon. In plaats van een regelmatige progressie met steeds dezelfde stap, herhaalt men een stappenpatroon. In het volgende voorbeeld is dit een patroon van achtereenvolgens 1, 3 en 2 stappen waarna dit patroon zich herhaalt. **Tabel 4.4** laat zien hoe in dit geval de vercijfering door de verschillende sleutelalfabetten heen loopt; de letters op de plaats van de puntjes zijn weer weggelaten om de diagonaal met vercijferde letters beter te doen uitkomen.

```
klaartekst : VYA NDN ADE RTW ATE RLI NIE STO PFO RTA SPE REN PAR AAT
patroon   : 132 132 132 132 132 132 132 132 132 132 132 132 132 132
sleutel   : FGJ LMP RSV XYB DEH JKN PQT VWZ BCF HIL NOR TUX ZAD FGJ
cryptogram : AEJ YPC RVZ ORX DXL AVV CYX NPN QHT YBL FDV KYK OAU FGC
```

Het voorbeeld laat zien dat deze eenvoudige 3-stapscyclus (1,3,2) al een periode van lengte 39 bewerkstelligt (de drie laatste sleutelletters in het voorbeeld komen overeen met de eerste drie). Dat dit zo is kan als volgt worden beredeneerd. Elke cyclus (1,3,2) brengt de sleutel 6 sleutelletters verder. De som van het aantal stappen over een gehele cyclus, modulo 26 genomen, wordt de *progressie-index PGI* genoemd. Hier heeft deze *PGI* dus de waarde 6.

Voor het berekenen van de periode wordt als volgt geredeneerd. Na elke cyclus heeft de sleutelletter de som van het aantal stappen in de cyclus doorlopen (6 in het voorgaande voorbeeld). Na zoveel cycli als nodig is om het aantal stappen een 26-voud te laten zijn, is één volledige periode doorlopen. Met een alfabet van 26-letters kunnen vier gevallen worden onderscheiden; de lengte van de cyclus noemen we *lengte*.

- Het aantal stappen in de cyclus heeft geen factor gemeen met 26. Er zijn dan *lengte* cycli nodig voordat de periode is voltooid. In dit geval is $\text{ggd}(\text{som stappen in periode}, 26) = 1$, maar omdat gereduceerd moet worden modulo 26 (wrap-around van het alfabet) is dat equivalent met $\text{ggd}(PGI, 26) = 1$. De maximale periode van $26 \times \text{lengte}$ is het resultaat.
- De *PGI* heeft de factor 2 gemeen met 26 ($\text{ggd}(PGI, 26) = 2$) en daardoor halveert het aantal benodigde cycli, dus ook de periode. De periode is $13 \times \text{lengte}$.
- De *PGI* heeft de factor 13 gemeen met 26 ($\text{ggd}(PGI, 26) = 13$) en daardoor daalt het aantal benodigde cycli met een factor 13. De periode is $2 \times \text{lengte}$.
- De *PGI* heeft een factor 26, zodat slechts één cyclus hoeft te worden doorlopen. De periode is *lengte*.

	V	Y	A	N	D	N	A	D	E	R	T	W	A	T	E	R
A	V	Y	A	N	D	N	A	D	E	R	T	W	A	T	E	R
B	W	Z	B	O	E	O	B	E	F	S	U	X	B	U	F	S
C	X	A	C	P	F	P	C	F	G	T	V	Y	C	V	G	T
D	Y	B	D	Q	G	Q	D	G	H	U	W	Z	D	W	H	U
E	Z	C	E	R	H	R	E	H	I	V	X	A	E	X	I	V
F	A	D	F	S	I	S	F	I	J	W	Y	B	F	Y	J	W
G	B	E	G	T	J	T	G	J	K	X	Z	C	G	Z	K	X
H	C	F	H	U	K	U	H	K	L	Y	A	D	H	A	L	Y
I	D	G	I	V	L	V	I	L	M	Z	B	E	I	B	M	Z
J	E	H	J	W	M	W	J	M	N	A	C	F	J	C	N	A
K	F	I	K	X	N	X	K	N	O	B	D	G	K	D	O	B
L	G	J	L	Y	O	Y	L	O	P	C	E	H	L	E	P	C
M	H	K	M	Z	P	Z	M	P	Q	D	F	I	M	F	Q	D
N	I	L	N	A	Q	A	N	Q	R	E	G	J	N	G	R	E
O	J	M	O	B	R	B	O	R	S	F	H	K	O	H	S	F
P	K	N	P	C	S	C	P	R	T	G	I	L	P	I	T	G
Q	L	O	Q	D	T	D	Q	S	U	H	J	M	Q	J	U	H
R	M	P	R	E	U	E	R	T	V	I	K	N	R	K	V	J
S	N	Q	S	F	V	F	S	V	W	J	L	O	S	L	W	J
T	O	R	T	G	W	G	T	W	X	K	M	P	T	M	X	K
U	P	S	U	H	X	H	U	X	Y	L	N	Q	U	N	Y	L
V	Q	T	V	I	Y	I	V	Y	Z	M	O	R	V	O	Z	M
W	R	U	W	J	Z	J	W	Z	A	N	P	S	W	P	A	N
X	S	V	X	K	A	K	X	A	B	O	Q	T	X	Q	B	O
Y	T	W	Y	L	B	L	Y	B	C	P	R	U	Y	R	C	P
Z	U	X	Z	M	C	M	Z	C	D	Q	S	V	Z	S	D	Q
	A	E	J	Y	P	C	R	V	Z	O	R	X	D	X	L	A

Tabel 4.5 Progressie vanaf F met patroon 1-3-2

Hiermee valt gemakkelijk in te zien dat het gegeven voorbeeld met $lengte=2$ en $PGI=6$ een periode $13 \times 3 = 39$ heeft. Na 13 cycli is de opschuiving in het alfabet $13 \times 6 = 78$ stappen, precies het eerste 26-voud in de reeks. Die 13 cycli zijn voltooid in $13 \times 3 = 39$ vercijferstappen en dat is dus de periode.

De complete formule voor de periode van de periodieke progressie luidt

$$p = \frac{26 \times lengte}{\text{ggd}(26, PGI)} \quad (4.3)$$

Een voorbeeld van een progressief systeem in een vercijfermachine is de cryptograaf van Alexander von Kryha uit 1924. De cyclus van deze machine wordt gegenereerd door een draaiende schijf bestaande uit 17 sectoren. Bij elke vercijfering gaat de sleutelletter een in deze sectoren vastgelegd aantal stappen verder in het cijferalfabet. Het klaaralfabet en het cijferalfabet kunnen onafhankelijk van elkaar worden gepermuteerd. De standaard sectorindeling is de volgende.

7 6 7 5 6 7 6 8 6 10 5 6 5 7 6 5 9

Het totaal aantal stappen in één rondgang van de sector bedraagt 111. Daarmee krijgt de PGI de waarde 111 modulo 26 = 7. Toepassing van [vergelijking \(4.3\)](#) geeft een totale periode van $26 \times 17 = 442$. De machine is overigens veel minder sterk dan door de periode van 442 en de claims van de uitvinder worden gesuggereerd. Een team van cryptoanalisten van het Amerikaanse leger onder leiding van William Friedman brak in 1933 (met handmatige methoden) een bericht van 1135 letters in precies 2 uur en 41 minuten^[1]. In [figuur 4.4](#) is een Kryha vercijfermachine afgebeeld.

4.3 Autoclaafstelsel

In de voorgaande polyalfabetische systemen was steeds sprake van periodiciteit. Na een bepaald aantal vercijferstappen herhaalt de vercijfersleutel zich. Men kan zich voorstellen dat daarin een zwakte schuilt. Zeker wanneer de periode klein is ten opzichte van de lengte van het bericht, zou men informatie kunnen putten uit letters die een geheel aantal periodes uit elkaar staan. Daarom is gezocht naar mechanismen



Figuur 4.4 Kryha vercijfermachine

die een aperiodieke, dat wil zeggen voortdurend veranderende sleutel kunnen genereren. Zo'n sleutel kan worden gerealiseerd met de zogenaamde *autokey* of *autoclaaf**.

De doorlopende sleutel moet worden gerealiseerd door middel van een voortdurend veranderend gegeven. Hiervoor zijn twee bronnen direct voorhanden: (1) de cijfertekst die tijdens het vercijferproces allengs groeit, en (2) de klaartekst zelf. Hoewel de navolgende voorbeelden gebruik maken van Vigenère-vercijfering met standaard alfabet, is dat geen must. Zo is ook het gebruik van gepermuteerde alfabetten in alle mogelijke variaties hier bruikbaar.

4.3.1 Autoclaaf met cijfertekst

In de cijfertekstautoclaaf dient de tekst van het cryptogram, na een langere of kortere aanloop met een sleutelwoord, als vervolg van de sleutel. In de navolgende vercijfering is eerst het sleutelwoord PENTAGON gebruikt, daarna de ontstane cijfertekst. Door de opdeling in groepen van 8 (de lengte van het sleutelwoord) is goed te zien dat de cijfertekst als sleutel wordt gebruikt.

* Het latijnse woord *clavis* betekent *sleutel*.

```
klaartekst : VYANDNAD ERTWATER LINIESTO PFORTASP ERENPARA AT  
sleutel : PENTAGON KCNGDTOQ OTGCDMSH ZBTKHELV OGHBAEDK SX  
cryptogram : KCNGDTOQ OTGCDMSH ZBTKHELV OGHBAEDK SXLOPEUK SQ
```

De inleidende sleutel PENTAGON zorgt voor cijfertekst KCNGDTOQ die vervolgens sleutel wordt, enz. In een variant van deze procedure wordt niet de gehele cijfertekst vanaf het begin gebruikt, maar zodra de sleutel uitgeput is neemt men de laatst gecijferde letter en gaat daarmee verder.

```
klaartekst : VYANDNAD ERTWATERLINIESTOPFORTASPERENPARAAT  
sleutel : PENTAGON QULEAATXOZHUCGYRFUZNEXXPEIZDQFFWWW  
cryptogram : KCNGDTOQ ULEAATXOZHUCGYRFUZNEXXPEIZDQFFWWWP
```

4.3.2 Autoclaaf met klartekst

Het nadeel van de cijfertekstautoclaaf is dat in feite de sleutel bijna geheel in het zicht is. De cryptoanalist hoeft slechts het punt te vinden waar de cijfertekst de rol van de sleutel overneemt en kan dan de rest van het bericht lezen. De klartekstautoclaaf, de tweede vorm van het autoclaaf systeem, heeft dat nadeel niet. Hier wordt de klartekst als vervolgsleutel gebruikt. Eerst wordt weer gecijferd met de sleutel PENTAGON, daarna neemt de klartekst die taak over.

```
klaartekst : VYANDNAD ERTWA TERLIN IESTOPFORT ASPE RENP ARAAT  
sleutel : PENTAGON VYAND NADERT WATERLINIE STOP FORT ASPER  
cryptogram : KCNGDTOQ ZPTJD GEUPZG EELXFANBZX SLDT WSEI AJPEK
```

Wat betreft de sterkte van het geheimschrift geldt de klartekstautoclaaf als sterker dan de cijfertekstautoclaaf, juist vanwege hetgeen hiervoor is opgemerkt. De klartekstautoclaaf heeft echter ook een zwakke

plek. Omdat klaartekst wordt gebruikt voor de sleutel, kan de cryptoanalist cryptogram en sleutel tegelijkertijd proberen te ontcijferen tot goede klaartekst. In termen van de informatietheorie uit **hoofdstuk 7**: de sleutelentropie wordt door het gebruik van klaartekst als sleutel verlaagd tot de entropie van de klaartekst.



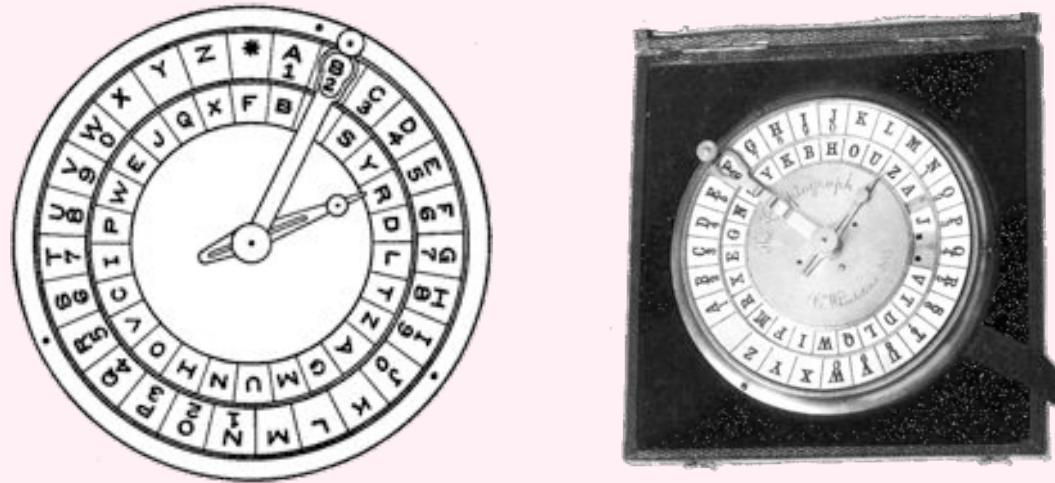
Wheatstone

4.3.3 Wheatstone-cryptograaf

Een nadeel van autoclaafcryptosystemen is het feit dat fouten cumulatief werken. Wanneer met de hand vercijferd wordt op spannende momenten, kunnen gemakkelijk fouten worden gemaakt die het ontcijferen bemoeilijken. Een uitweg hiervoor biedt mechanisatie van de vercijfering. De in **figuur 4.5** getoonde horlogeachtige constructie maakt een klaartekstautoclaaf. Deze Wheatstone-cryptograaf is geconstrueerd door Charles Wheatstone en voor het eerst door hem getoond op de Wereldtentoonstelling van 1867 te Parijs.

Het apparaat bestaat aan de bovenkant uit twee concentrische cirkels, op beide daarvan bevindt zich een rondlopend alfabet. De buitenste cirkel in de figuur draagt het normale alfabet, tussen Z en A aangevuld met een spatie of een sterretje. Deze cirkel heeft 27 posities. De binnenste cirkel moet het met slechts 26 posities doen. In de afbeeldingen is de binnenring bezet met een permutatie van het alfabet. Op de foto rechts ontbreken daaruit een paar letters, zodat te zien is dat het losse letters zijn. De permutatie kan dus worden veranderd. Hetzelfde geldt voor het buitenste alfabet. De twee groepen cijfers 1-0 bij de letters A-J en N-W dienen om gemakkelijk getallen te kunnen vercijferen.

Een grote wijzer bepaalt een positie op de buitenste cirkel, een kleine wijzer een op de binnenste. Gewoonlijk laat men de grote wijzer de klaartekst aangeven en de kleine wijzer de overeenkomstige cijfertekst. Niets verhindert echter om dat juist andersom te doen. De twee wijzers voeren een gekoppelde beweging uit. De constructie is zodanig dat voor elke positie die de grote wijzer opschuift, de kleine wijzer eveneens een positie verder gaat. Aangezien de buitenste cirkel 27 en de binnenste 26 posities telt, zal bij één



Figuur 4.5 Wheatstone cryptograaf

rondgang van de grote wijzer de kleine één omloop plus één positie verder gelopen zijn. In de afbeelding links staat de grote wijzer op B en de kleine op R, na 27 stappen staat de grote weer op B maar de kleine wijst dan D aan.

Om te vercijferen legt men de gekozen alfabetten op beide cirkels uit en zet de wijzers in de afgesproken beginstand. Dan wordt de grote wijzer gedraaid (klok of antiklok, net wat is afgesproken) totdat deze de eerste letter van de klaartekst aanwijst, de kleine wijzer geeft dan de overeenkomstige letter van het cryptogram. Vervolgens wordt doorgedraaid naar de tweede letter van de klaartekst en de vercijferde letter weer afgelezen, enz. Komt men een verdubbelde letter tegen dan wordt de tweede vervangen door een andere letter, Q en X zijn hiervoor populair. Voor het ontcijferen wordt het apparaat op dezelfde manier ingesteld, maar het draaien wordt nu gestuurd door de letters op het binnenste alfabet.

Verscheidene varianten van het apparaat zijn bekend. Zoals een exemplaar van Deense origine met 29 posities op de binnenste cirkel en 30 op de buitenste. Het grotere aantal posities is bedoeld om extra

tekens als ø, ü te kunnen gebruiken. Blijkens de *Vejledning i Anvendelse af Urkryptografen* van het Deense Kriegsministeriet is dit apparaat nog betrekkelijk recent in gebruik geweest bij het Deense leger, want deze handleiding draagt 23 juni 1936 als datum^[2].

In het volgende voorbeeld is de instelling van de Wheatstone op de foto rechts gebruikt. Het klaaralfabet bestaat uit de 26 letters in de normale volgorde plus de spatie. Weliswaar is het cijferalfabet in de afbeelding niet geheel compleet en is de letter onder de grote wijzer onleesbaar, maar het kan gecompleteerd worden omdat voor de constructie ervan een sleutelwoord blijkt te zijn gebruikt*. De grote wijzer is aan het begin van de vercijfering op de blanco gezet, de kleine wijzer op de letter F. Vanaf daar gaat de grote wijzer naar de eerste letter V van de klaartekst en loopt de kleine mee naar de eerste cryptogramletter L, enz. In de klaartekst is de laatste dubbele AA vervangen door AQ.

```
klaartekst : VYAND NADERT WATERLINIE STOP FORT ASPEREN PARAQT  
cryptogram : LIRAN CESYDQ FGWBQPCDJZ RXIF JFXG UGXVNTX GJYPYH
```

4.4 Multiplexsysteem

Variatie in de afwisseling van het cijferalfabet behoeft niet uitsluitend van eenzelfde alfabet te worden afgeleid, zoals in de voorgaande polyalfabetische systemen wel steeds is gebeurd. Het *multiplexsysteem* is gebaseerd op meerdere, totaal van elkaar verschillende cijferalfabetten.

Een multiplexgeheimschrift bestaat uit een verzameling verschillend gepermuteerde versies van het alfabet. Hieruit wordt een subset gekozen voor de vercijfering van het bericht. In het voorbeeld van [tabel 4.6](#) zijn dat 5 horizontaal op strips uitgeschreven alfabetpermutaties, gekozen uit een totaal van 25. De verkozen strips met de nummers 3, 7, 11, 12, 23 zijn onder elkaar gelegd in de volgorde 11, 3, 23, 12, 7. De nummers en hun volgorde bepalen de sleutel voor het cryptogram. De 5 strips vormen samen een

* Ook het cijferalfabet in de schets links in [figuur 4.5](#) berust op een sleutelwoord. Kan de lezer beide vinden?

11	A	L	T	M	S	X	V	Q	P	N	O	H	U	W	D	I	Z	Y	C	G	K	R	F	B	E	J	
3	C	Z	I	N	X	F	Y	Q	R	T	V	W	L	A	D	K	O	M	J	U	B	G	E	P	H	S	
23	J	C	P	G	B	Z	A	X	K	W	R	E	V	D	T	U	F	O	Y	H	M	L	S	I	Q	N	
12	V	E	W	O	A	M	N	F	L	H	Q	G	C	U	J	T	B	Y	P	Z	K	X	I	S	R	D	
7	V	R	O	G	S	Y	D	U	L	C	F	M	Q	T	W	A	H	X	J	E	Z	B	N	I	K	P	
11	M	S	X	V	Q	P	N	O	H	U	W	D	I	Z	Y	C	G	K	R	F	B	E	J	A	L	T	
3	Q	R	T	V	W	L	A	D	K	O	M	J	U	B	G	E	P	H	S	C	Z	I	N	X	F	Y	
23	X	K	W	R	E	V	D	T	U	F	O	Y	H	M	L	S	I	Q	N	J	C	P	G	B	Z	A	
12	X	I	S	R	D	V	E	W	O	A	M	N	F	L	H	Q	G	C	U	J	T	B	Y	P	Z	K	
7	B	N	I	K	P	V	R	O	G	S	Y	D	U	L	C	F	M	Q	T	W	A	H	X	J	E	Z	
11	F	B	E	J	A	L	T	M	S	X	V	Q	P	N	O	H	U	W	D	I	Z	Y	C	G	K	R	
3	F	Y	Q	R	T	V	W	L	A	D	K	O	M	J	U	B	G	E	P	H	S	C	Z	I	N	X	
23	J	C	P	G	B	Z	A	X	K	W	R	E	V	D	T	U	F	O	Y	H	M	L	S	I	Q	N	
12	H	Q	G	C	U	J	T	B	Y	P	Z	K	X	I	S	R	D	V	E	W	O	A	M	N	F	L	
7	T	W	A	H	X	J	E	Z	B	N	I	K	P	V	R	O	G	S	Y	D	U	L	C	F	M	Q	
11	I	Z	Y	C	G	K	R	F	B	E	J	A	L	T	M	S	X	V	Q	P	N	O	H	U	W	D	
cryptogram: DDTJW XTWSI VKRZI X																											

Tabel 4.6 De 5 gekozen alfabet strips en de vercijfering

vercijferblok, waarin door onderlinge verschuivingen tussen de strips de klaartekst verticaal kan worden gelezen.

Naast de verticale kolom met klaartekst staan ter weerszijden 25 andere kolommen, die elk een andere vercijfering van de klaartekst geven. Men noemt zo'n kolom een *generatrix*, omdat deze een vercijfering genereert. Elk van deze generatrices mag worden gekozen als cijfertekst. Voor de ontcijfering geeft dat nauwelijks problemen, want meestal zal niet meer dan één kolom bruikbare tekst opleveren. Dat is zeker het geval wanneer het blok meer strips telt dan hier. Gewoonlijk wordt het gepermuteerde alfabet tweemaal op de strip uitgeschreven, zodat bij het uitkiezen van een bepaalde generatrix niet hoeft te

worden doorgeschoven. Omwille van de ruimte is dit in het voorbeeld achterwege gelaten.

Een bericht langer dan het aantal gekozen alfabetten wordt opgedeeld in blokken en per blok vercijferd. Men doet er goed aan om voor elk blok ook steeds een andere generatrix te kiezen. Merk op dat in dit systeem een klaartekstletter altijd door een andere letter wordt vervangen, *nooit* door zichzelf.



Figuur 4.6 Jefferson cylinder in M-94 uitvoering

Zoals zoveel cryptosystemen is ook dit principe herhaaldelijk opnieuw uitgevonden. De oudst bekende versie is die van de Amerikaan Thomas Jefferson, rond het jaar 1790. Zijn versie bestond uit 36 schijven met een alfabetpermutatie op de omtrek. In 1891 probeerde de Franse majoor Etienne Bazeries vergeefs zijn soortgelijke Bazeries-cylinder met 20 alfabetschijven aan de man te brengen. De Amerikaanse kolonel Parker Hitt bedacht het systeem opnieuw in 1914, nu in de vorm van strips.

In 1923 werd de Jefferson-cylinder door het US Army Signal Corps (Verbindingsdienst), in de vorm van de befaamde M-94 officieel in gebruik genomen ([figuur 4.6](#)). Het apparaat telt 25 schijven, die in willekeurige volgorde op een as kunnen worden vastgeschroefd. Een horizontale lineaal fungeert als leeslijn bij het instellen van de klaartekst. De alfabetpermutaties zijn weergegeven in [tabel 4.7](#). Alleen strip 17 vertoont duidelijke sporen van een sleutelconstructie, de overige zijn grondig gemengde alfabetten en

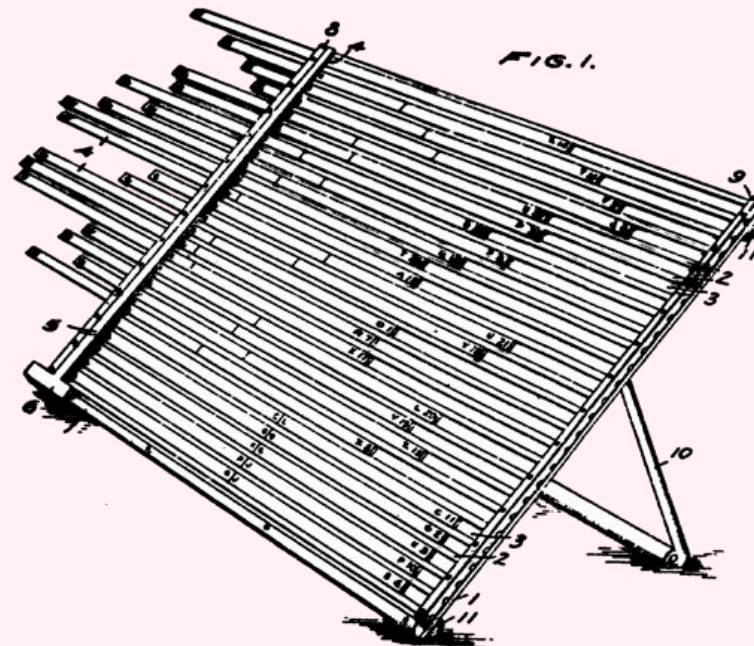
1	A	B	C	E	I	G	D	J	F	V	U	Y	M	H	T	Q	K	Z	O	L	R	X	S	P	W	N
2	A	C	D	E	H	F	I	J	K	T	L	M	O	U	V	Y	G	Z	N	P	Q	X	R	W	S	B
3	A	D	K	O	M	J	U	B	G	E	P	H	S	C	Z	I	N	X	F	Y	Q	R	T	V	W	L
4	A	E	D	C	B	I	F	G	J	H	L	K	M	R	U	O	Q	V	P	T	N	W	Y	X	Z	S
5	A	F	N	Q	U	K	D	O	P	I	T	J	B	R	H	C	Y	S	L	W	E	M	Z	V	X	G
6	A	G	P	O	C	I	X	L	U	R	N	D	Y	Z	H	W	B	J	S	Q	F	K	V	M	E	T
7	A	H	X	J	E	Z	B	N	I	K	P	V	R	O	G	S	Y	D	U	L	C	F	M	Q	T	W
8	A	I	H	P	J	O	B	W	K	C	V	F	Z	L	Q	E	R	Y	N	S	U	M	G	T	D	X
9	A	J	D	S	K	Q	O	I	V	T	Z	E	F	H	G	Y	U	N	L	P	M	B	X	W	C	R
10	A	K	E	L	B	D	F	J	G	H	O	N	M	T	P	R	Q	S	V	Z	U	X	Y	W	I	C
11	A	L	T	M	S	X	V	Q	P	N	O	H	U	W	D	I	Z	Y	C	G	K	R	F	B	E	J
12	A	M	N	F	L	H	Q	G	C	U	J	T	B	Y	P	Z	K	X	I	S	R	D	V	E	W	O
13	A	N	C	J	I	L	D	H	B	M	K	G	X	U	Z	T	S	W	Q	Y	V	O	R	P	F	E
14	A	O	D	W	P	K	J	V	I	U	Q	H	Z	C	T	X	B	L	E	G	N	Y	R	S	M	F
15	A	P	B	V	H	I	Y	K	S	G	U	E	N	T	C	X	O	W	F	Q	D	R	L	J	Z	M
16	A	Q	J	N	U	B	T	G	I	M	W	Z	R	V	L	X	C	S	H	D	E	O	K	F	P	Y
17	A	R	M	Y	O	F	T	H	E	U	S	Z	J	X	D	P	C	W	G	Q	I	B	K	L	N	V
18	A	S	D	M	C	N	E	Q	B	O	Z	P	L	G	V	J	R	K	Y	T	F	U	I	W	X	H
19	A	T	O	J	Y	L	F	X	N	G	W	H	V	C	M	I	R	B	S	E	K	U	P	D	Z	Q
20	A	U	T	R	Z	X	Q	L	Y	I	O	V	B	P	E	S	N	H	J	W	M	D	G	F	C	K
21	A	V	N	K	H	R	G	O	X	E	Y	B	F	S	J	M	U	D	Q	C	L	Z	W	T	I	P
22	A	W	V	S	F	D	L	I	E	B	H	K	N	R	J	Q	Z	G	M	X	P	U	C	O	T	Y
23	A	X	K	W	R	E	V	D	T	U	F	O	Y	H	M	L	S	I	Q	N	J	C	P	G	B	Z
24	A	Y	J	P	X	M	V	K	B	Q	W	U	G	L	O	S	T	E	C	H	N	Z	F	R	I	D
25	A	Z	D	N	B	U	H	Y	F	W	J	L	V	G	R	C	Q	M	P	S	O	E	X	T	K	I

Figuur 4.7 De 25 alfabet strips van M-94

zijn natuurlijk weer het best^{[3]*}. De alfabetten van de Bazeries-cylinder blijken voor minstens een deel op

* De volgorde van de schijven in de foto van de M-94 volgens de nummering van tabel van links naar rechts is: 24, 23, 22, 25,

teksten gebaseerd te zijn. Gereconstrueerd konden worden: "Allons enfants de la patrie, le jour de gloire est arrivé", "Dieu protège la France", "Evitez les courants d'air" en "J'aime l'oignon frit à l'huile"^[4].



Figuur 4.8 Multiplex strips, patent van William Friedman

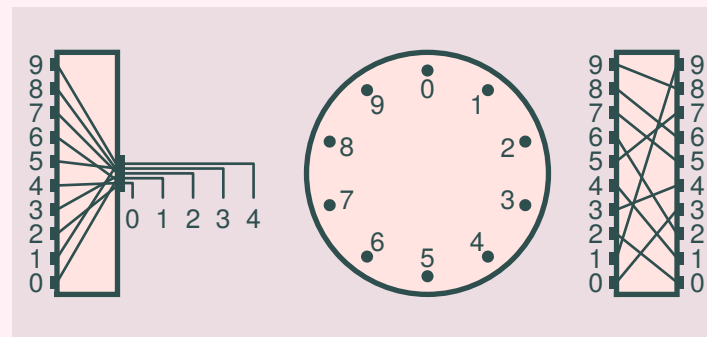
De M-94 heeft tot 1942 dienst gedaan. Andere varianten van het stripsysteem hebben het tot aan de jaren 1950 volgehouden. Genoemd kunnen worden de met strips uitgevoerde CSP-845 van de Amerikaanse marine en de M-138 van het Amerikaanse leger, sinds 1945 samengevoegd tot één systeem met 100 alfabetstrips. Elke dag werden 30 strips uitgekozen, waaruit voor elk bericht dan weer 25 stuks genomen werden. Ook het US State Department (Buitenlandse Zaken) gebruikte een stripsysteem, getuige de analyse daarvan in Duitsland in de periode 1942–44 door Pers-Z (speciale afdeling van Buitenlandse Zaken)^[5]. Een

18, 14, 4, 17, 3, 2, 15, 13, 12, 1, 10, 8, 9, 11, 7, 21, 6, 20, 19, 16, 5.

schets van een uitvoering met in een raamwerk schuivende strips staat in [figuur 4.8](#); deze tekening is genomen uit een door William Friedman aangevraagd patent.

4.5 Cryptografische rotor

Een machinale manier om een voortschrijdende reeks van sleutelletters te verkrijgen is gebaseerd op de *cryptografische rotor*. De rotor is uitgevonden aan het begin van de 20^e-eeuw en sedert de eerste decennia ervan in verschillende varianten gebruikt in cryptografen. In [figuur 4.9](#) zijn twee rotorvarianten afgebeeld. Links een eenzijdige of halfrotor en rechts een tweezijdige rotor. Een rotor bewerkstelligt in feite niet meer dan een monoalfabetische substitutie. Maar omdat de rotor kan draaien – zoals de naam al sugereert – wisselt het effect van deze substitutie met elke stap.

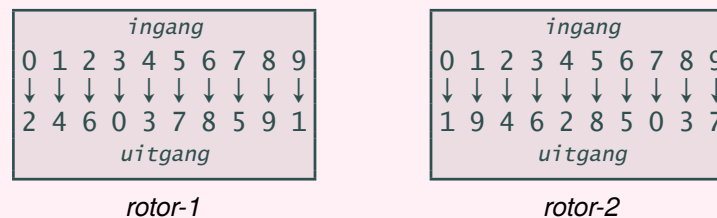


Figuur 4.9 Rotor varianten, halfrotor (links), tweezijdige rotor (rechts)

Kijk eerst naar de linkerrotor van [figuur 4.9](#), een door Arvid Damm ontworpen zogenaamde halfrotor en gebruikt in de Hagelin B-211. De ingangscontacten van deze rotor bevinden zich aan de rechterkant. Dit zijn vaste aansluitpunten op de as van de rotor, die door middel van sleepcontacten aangesloten kunnen worden. De uitgangscontacten zitten in de figuur aan de linkerkant van deze rotor en draaien met deze mee. In het zijaanzicht in het midden kijkt men op de in een concentrische cirkel geplaatste uit-

gangscontacten. Deze Damm-rotor heeft vijf invoercontacten die worden uitgesplitst naar tien uitgaande contacten.

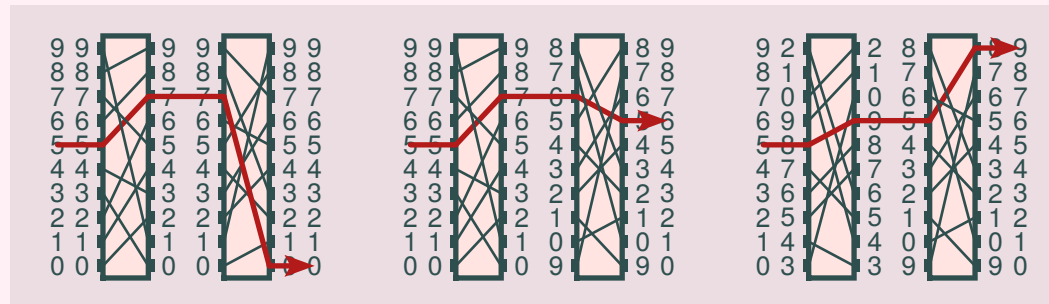
Rechts in [figuur 4.9](#) staat de meer gangbare tweezijdige rotor, die o.a. voorkomt in de Hebern-cryptograaf en in de Duitse Enigma's. Aan beide kanten van de rotor bevindt zich een serie contacten, in dit voorbeeld 10. Meestal zijn het er 26 en worden hun posities op de omtrek van de rotor gemarkeerd met de letters van het alfabet. De contacten zijn door draden kruislings met elkaar verbonden, en wel zodanig dat ze allemaal een verbinding hebben met een contact aan de andere kant. Daardoor kunnen de contacten aan de ene zijde worden gezien als een permutatie van die aan de overkant. Twee voorbeelden van een rotorpermutatie staan in [figuur 4.10](#). Als aan de ingangskant van rotor-1 op positie 0 een elektrische spanning wordt aangelegd, zal deze worden doorgegeven naar uitgangskontakt 2, in het geval van rotor-2 is dat uitgangskontakt 1.



Figuur 4.10 Rotorbedrading

In een cryptograaf zitten de rotors gemonteerd op een as zodat ze kunnen draaien. Door middel van vaste, uitwendige sleepcontacten wordt de rotor aangesloten op de elektrische installatie van het apparaat. Ook deze contacten worden van 0–9 genummerd. In [figuur 4.11](#) staan twee gekoppelde rotors in drie onderling verschillende posities. Links ziet men de 10 ingangskontakten en rechts de 10 uitgangskontakten met daartussen de twee rotors. In de linkerfiguur van de drie staan de beide rotors zó, dat de contacten van de rotors en de uitwendige contacten qua nummering op elkaar aansluiten, 0 tegenover 0, 1 tegenover 1, enz. Op uitwendig contact nummer 5 wordt links een spanningsbron aangesloten. De spanning steekt

over naar ingangscontact 5 van rotor-1 en zal conform diens bedrading deze aan de andere zijde via contact 7 verlaten. Rotor-2 wordt dan binnengegaan op contact 7 en weer verlaten bij 0. Het uitwendig uitgangssignaal verschijnt op contact 0 aan de rechterkant.



Figuur 4.11 Rotorkoppeling en draaiing

Nu wordt de rotor rechts één stap verdraaid (middelste figuur). Tegenover contact 7 van de rotor links staat door de verdraaiing nu contact 6 van rotor-2. De uitvoer verschijnt zodoende op contact 5 van die rotor, maar door de draaiing ervan staat dit contact nu tegenover uitgangssignaal 6. Vervolgens is in de rechterfiguur de rotor links drie stappen naar de andere kant gedraaid. Hierdoor betreedt het ingangssignaal deze rotor via diens contact 8, verlaat die via 9, steekt over naar 5 van de rotor rechts, verlaat die weer op 8, welk contact tenslotte aansluit op uitgangssignaal 9.

Voor een systematische beschrijving van deze draaiende rotorsubstitutie moet als volgt geredeneerd worden. Om te beginnen moet worden bedacht dat het effect van een rotor wordt beschreven ten opzichte van een vast referentiepunt voor het ingaande en het uitgaande signaal, zoals is gedaan in het bedradingschema in [figuur 4.10](#). Begin weer links in [figuur 4.11](#), waar de uitgaande contacten van de stilstaande rotor links de referentie vormen voor de ingangskant van de rotor rechts. De substitutietabel van een rotor is slechts geldig voor één bepaalde uitgangspositie. In de figuur is die positie in het plaatje links wel gerealiseerd, omdat daar de uitwendige contacten matchen met die op de rotor. Door een rotor een stap

te draaien – zoals in het middelste plaatje – gaat de referentiepositie verloren en kan de substitutietabel niet zonder meer toegepast worden. Immers 7 komt niet meer binnen op 7 maar op 6.

Om de referentiepositie weer te bereiken moet het ingaande signaal daarom één positie worden teruggedraaid. Dit teruggedraaien van hetingangssignaal is in feite een Caesar-substitutie C^{-1} . Nu kan de rotorsubstitutie wel volgens de tabel worden uitgevoerd. Vervolgens moet de verdraaiing van hetingangssignaal ongedaan gemaakt worden door een Caesar-verschuiving C^1 de andere kant uit. Het signaal dat op 7 uit de rotor links komt, is naar $C^{-1}(7) = 6$ teruggedraaid, gesubstitueerd van $6 \rightarrow 5$ en tenslotte door de tweede Caesar-verschuiving $C^1(5) = 6$ overgevoerd naar uitgang 6. Evenzo ondergaat het signaal in het plaatje rechts achtereenvolgens een verdraaiing naar $C^{-(-3)}(5) = 8$, substitutie $8 \rightarrow 9$ en tot slot een draai naar $C^{-3}(9) = 6$ vanwaar het op die positie oversteekt naar het inmiddels voorgedraaide contact 5.

Deze redenering kan worden omgezet in een handzame formule. Noem de substitutie geïmplementeerd in de rotor R , hetingangssignaal p en het resultaat c . De substitutie door een rotor verdraaid tot positie i wordt dan gegeven door de van rechts naar links uitgevoerde transformaties

$$c = C^i R C^{-i} p \quad (4.4)$$

Noem de rotor rechts A en de rotor links B , hun actuele positie respectievelijk a en b . De twee gekoppelde rotors uit het voorbeeld kunnen 100 verschillende standen innemen. Het voortgangsmechanisme bepaalt hoe de waarden van a en b achtereenvolgens veranderen. Vercijferen van klaartekst $p_1, p_2, p_3, \dots$ met deze twee draaiende rotors wordt gegeven door

$$c_i = C^b R_B C^{-b} C^a R_A C^{-a} p_i \quad (4.5)$$

Aldus bewerkstelligt de combinatie van rotors met gelijkmatige of onregelmatige voortbeweging een voortgaande reeks vercijferingen.

4.6 Lineaircongruente reeks

Nagenoeg elk computersysteem beschikt over een routine voor het maken van reeksen willekeurige getallen. Een van de meest bekende recurrente betrekkingen voor het maken van zo'n reeks is de lineaircongruente reeks.

$$x_{n+1} = (a \cdot x_n + b) \bmod m \quad (4.6)$$

De reeks wordt gestart met het kiezen van een beginwaarde voor x_0 . De drie constanten a , b en m moeten met zorg gekozen worden. Het hoe en waarom wordt uit de doeken gedaan in het tweede deel van Donald Knuth's standaardwerk *The Art of Computer Programming*^[6]. Een lineaircongruente reeks is altijd periodiek, hoe langer de periode des te beter. Met $b = 0$ wordt de periode korter dan met $b \neq 0$, maar het berekenen van de volgende waarde gaat dan iets sneller. Vanzelfsprekend is m de langst mogelijke periode.

Aangezien computers werken met woordlengtes die veelvouden van 8 zijn, is het niet verwonderlijk dat (in de wat oudere literatuur) vaak $m = 2^{16}$ of $m = 2^{32}$ wordt aangetroffen; het reduceren modulo m gaat dan praktisch vanzelf. Deze keuze van m brengt echter een nadeel met zich mee. Namelijk de minst significante bits van de reeks zullen korte periodes vertonen en minder random zijn dan de meest significante bits. Kijk bijvoorbeeld naar het meest rechterbit, dat geïsoleerd wordt met

$$y_n = x_n \bmod 2 \quad (4.7)$$

Het daaropvolgende rechterbit wordt

$$\begin{aligned} y_{n+1} &= ((a \cdot x_n + b) \bmod m) \bmod 2 \\ &= (a \cdot (x_n \bmod 2) + b) \bmod m \\ &= (a \cdot y_n + b) \bmod m \end{aligned} \quad (4.8)$$

Afhankelijk van de pariteit van x_0 , a en b zal in de reeks het rechterbit òf constant zijn òf afwisselend 0

en 1. Het is om deze reden dat men vaak modulus 2^n vervangt door $(2^n + 1)$ of $(2^n - 1)$. Een alternatief is het kiezen van een zo groot mogelijk priemgetal p als modulus^{*}. Het nadeel is dan een meer tijdrovende berekening van de reductie modulo p .

Voor a en b gelden restricties als men de maximale periode wil halen. Zo moeten b en m relatief priem zijn. Dus met bijvoorbeeld $m = 2^{32}$ moet b oneven gekozen worden. Voor a zijn er twee voorwaarden. Ten eerste moet $(a - 1)$ een veelvoud zijn van elk priemgetal p dat voorkomt in m . En ten tweede, als m een 4-voud is dan moet $(a - 1)$ ook een 4-voud zijn.

m en de beginwaarde van x_0 moeten relatief priem zijn, a een primitief element modulo m . De maximale periode wordt niet gehaald als voor de eenvoud $b = 0$ wordt genomen. Neemt men daarbij $m = 2^{32}$ en $a = 3$ of 5 modulo 8 dan is de periode met 2^{30} het best mogelijke. Men heeft wel genomen $a = (2^k + 1)$ en $b = 1$ omdat dan de vermenigvuldiging kan worden vermeden (schuiven in een register en optellen volstaan). Echter de gegenereerde getallen zijn dan niet willekeurig genoeg. Een klein voorbeeld

$$x_{n+1} = 5x_n + 3 \bmod 8 \rightarrow 0, 3, 2, 5, 4, 7, 6, 1 \quad (4.9)$$

Merk op dat de coëfficiënten $a = 5$ en $b = 3$ aan de gestelde voorwaarden voldoen. Zie ook hoe het rechterbit wisselt tussen even en oneven. Men kan zelfs opmerken dat de twee minst significante bits een subperiode $\{00, 11, 10, 01\}$ vormen.

Een voorbeeld van de toepassing van deze generator voor encryptie kan worden gevonden in de Postscript Type 1 fonts[†]. Het font-programma bestaat uit een reeks instructies waarmee in een printer een afbeelding van een letterteken wordt opgebouwd. Deze instructies zijn met een lineair congruente rij vercijferd.

De lineair congruente generator is op zich niet sterk genoeg voor cryptografische doeleinden. De constanten a , b en m kunnen te gemakkelijk berekend worden^[7]. Het kernpunt is het bepalen van de modulus

* Vergelijk met hetgeen in [hoofdstuk 9](#) voor systemen met publieke sleutel wordt gedaan

† *Adobe Type 1 Font Format*, Adobe Systems Inc.

m . Is die eenmaal bekend dan kunnen a en b berekend worden uit drie opeenvolgende waarden van de reeks. Immers uit het verschil van twee opvolgende recurrente betrekkingen valt b weg en volgt a uit

$$(x_2 - x_1) = a(x_1 - x_0) \bmod m \quad (4.10)$$

Als a gevonden is volgt b als vanzelf. Marsaglia geeft een eenvoudige methode om de modulus te benaderen^[8]. Laat telkens twee opeenvolgende waarden uit de reeks een punt (x, y) bepalen. Vier opeenvolgende waarden leggen zo drie punten vast. Deze drie punten bepalen samen een parallelopipedum waarvan de oppervlakte zich laat berekenen als

$$O_1 = |(x_3 - x_1)(x_3 - x_2) - (x_2 - x_1)(x_4 - x_2)| \quad (4.11)$$

Herhaal dit voor O_2 uit x_2, x_3, x_4, x_5 enz. De reeks waarden O_i zal in de grootste gemene delers van O_i en O_{i+1} een bovengrens voor m geven. In het voorbeeld $(5x_n + 3 \bmod 8)$ komen waarden $O = 8, 16$ en 24 tevoorschijn.

Omdat voor generatoren met een modulus zoals 2^n de minst significante bits onvoldoende willekeurig zijn, laat men die soms weg en gebruikt alleen de hogere bits. Dat maakt de cryptanalyse wel lastiger maar niet onmogelijk.

4.6.1 MacLaren-Marsaglia-generator

MacLaren en Marsaglia bedachten een manier om twee generatoren te combineren, naar hen de MacLaren-Marsaglia generator genoemd^[9]. Het komt erop neer dat de ene generator de waarden produceert en dat de andere de volgorde daarvan wijzigt. Een buffer wordt geïnitieerd met de eerste generator waarna de tweede generator de plaats in de buffer bepaalt vanwaaruit het volgende resultaat komt. De uitgekozen plaats in de buffer wordt meteen weer opgevuld door middel van de eerste generator. De random reeks uit de eerste generator ondergaat dus een permutatie, hoewel er toch een zekere lokaliteit blijft bestaan. Dat is het gevolg van het feit dat de verblijfsduur van een waarde in de buffer beperkt is. Knuth rapporteert

een naar zijn zeggen betere manier om die permutatie tot stand te brengen, waarvoor maar één generator nodig is in plaats van twee^[10].

Op dit principe baseerde de firma Cogent Systems Inc. in de jaren-80 van de vorige eeuw haar CSI-10 cryptogenerator. In feite een Hewlett-Packard HP41C programmeerbare rekenmachine met een cryptografisch programma in de vorm van een MacLaren-Marsaglia generator^[11]. Deze toevalsreeks vercijfert 16-bit waarden met een exclusive or, eventueel in autokeymodus. De gebruikte generatoren zijn

$$\begin{aligned}x_{n+1} &= 4353x_n + 62633 \bmod 32768 (= 2^{15}) \\y_{n+1} &= 46876y_n + 32749 \bmod 59049 (= 3^9)\end{aligned}\tag{4.12}$$

De eerste generator levert de waarden x_i en de tweede haalt ze met $(y_i \bmod 257)$ uit de 257 plaatsen tellende buffer.

In een artikel in het tijdschrift Cryptologia liet Charles Retter zien hoe dit systeem met een korte hoeveelheid klaartekst kan worden gebroken^[12]. Dat artikel leidde tot een ietwat vinnige correspondentie van de kant van de directeur van Cogent Systems, die vanzelfsprekend niet blij was met de manier waarop zijn product werd afgebrand.

De cryptoanalyse kon uitgaan van bekende waarden voor de parameters van de generatoren. Die kwamen ter beschikking door uitlezen van het vercijferprogramma. Gegeven de korte periode van de tweede generator (zelfs in 1984 was 59049 niet teveel om een volledige periode op te slaan in een computer) wordt een tabel gemaakt van de verblijfstijden in de buffer. Een bekend of geraden woord wordt op alle mogelijke posities geplaatst, de bijbehorende sleutelstroom wordt bepaald en dan na enkele berekeningen getest op tegenstijdingen in genoemde verblijfstijd. Meestal was met 4 en maximaal 11 waarden wel duidelijk of de klaartekst past of niet. Daarna is berekening van de startwaarden van de generatoren eenvoudig.

4.7 One-time pad

Het zogenaamde one-time pad is het ultieme geheimschrift. Een volstrekt willekeurige reeks sleutel-elementen wordt voor vercijfering gebruikt. Die reeks moet tenminste even lang zijn als het te vercijferen bericht. Men kan het one-time pad zien als een periodiek systeem met een oneindig lange sleutel. Het allergrootste voordeel van dit systeem: *het is werkelijk onbreekbaar*.

Er is wel een voorwaarde verbonden aan die onbreekbaarheid: zo'n sleutel of een deel ervan mag nooit en te nimmer meer dan één keer worden gebruikt. Dat laatste overkwam de Russische KGB in de tijd van de Tweede Wereldoorlog. Door herhaald gebruik van stukken van de sleutelreeksen konden Amerikaanse cryptanalisten delen van de cryptogrammen breken. Codenaam VENONA is een project dat tot 1980 is doorgegaan. Ongeveer 25.000 onderschepte telegrammen konden worden verzameld, waarvan er zo'n 3.000 (gedeeltelijk) zijn opgelost.

Al eerder is het gebruik van het one-time pad als oververcijfering bij een code genoemd. Oververcijfering met een willekeurige reeks versluiert de herhaling van codegroepen. Vooral bij een numerieke code gaat dit eenvoudig: in het navolgende voorbeeld wordt een willekeurige reeks cijfers zonder overdracht of lenen opgeteld bij de codegetallen (uit een fictief codeboek).

klaartekst :	VYAND	NADERT	WATER	LINIE	STOP	FORT	ASPEREN	PARAAT
code :	3248	1002	9954	4850	0023	7377	1192	5528
sleutel :	4223	3504	6233	9230	1887	0818	1764	7490
cryptogram :	7461	4506	5187	3080	1800	7185	2856	2918

Nu zal men vragen: "Waarom gebruikt niet iedereen altijd en overal dit geweldige systeem?" Daar zijn verschillende redenen voor.

- Het maken van een goede, willekeurige sleutelreeks is niet eenvoudig. Weliswaar bestaan algoritmen die goede willekeurige reeksen voortbrengen. Maar de (wiskundige) fabricageprocessen zijn onont-



Figuur 4.12 Pagina uit het one-time pad van een spion

koombaar *deterministisch* en dus niet echt willekeurig. Wie de algoritme en de parameters kent, kan de reeks probleemloos reproduceren. Het best en eenvoudigst is nog om gebruik te maken van willekeurige fysische processen, zoals radioactief verval, elektronische ruis. Recent is daar nog bijgekomen het benutten van quantumprocessen: fotonen die een 0 of een 1 produceren.

- De sleutelreeks moet zowel bij de afzender als bij de ontvanger van het bericht bekend zijn. Dat kan een probleem vormen als de communicatie tussen afzender en ontvanger is verbroken. Op een gegeven moment is misschien de sleutelvoorraad uitgeput. Wat dan? Herhaald gebruik van de sleutel is immers uit den boze.
- Onthouden van een sleutel is er vanzelfsprekend niet bij. Deze moet dus worden opgeschreven, maar dat brengt meteen het gevaar van diefstal of ontdekking met zich mee. Daar moeten dan weer voorzorgen tegen getroffen worden. Spionnen hebben bijvoorbeeld de sleutel op een dun folie dat gemakkelijk

en snel kan worden verbrand.

Een one-time pad kan verschillende vormen aannemen. Met een uit letters bestaande sleutel kan bijvoorbeeld een Vigenère gestuurd worden, zoals in het volgende voorbeeld.

```
klaartekst : VYAND NADER TWATE RLINI ESTOP FORTA SPERE NPARA AT  
sleutel : DBQLX UVAPX FUGGN KTBAL KMKLW UFKGY ABE0Z JOGLS PC  
cryptogram : YZQYA HVDT0 YQGZR BEJNT OEDZL ZTBZY SQIFD WDGCS PV
```

Het gebruik van sleutelzinnen – zoals bijvoorbeeld zinnen uit een boek – is geheel uit den boze. Omdat in dat geval zowel de klaartekst als de sleutel een normale betekenis hebben, kan de cryptoanalist proberen om in beide tegelijkertijd woorden te laten verschijnen. Dat is niet zo moeilijk als het lijkt, omdat er een beperkt aantal woorden is dat relatief vaak in de sleuteltekst zal voorkomen. Denk aan veelvoorkomende werkwoordsvormen, voegwoorden, lidwoorden, enz. Voor elke positie in het cryptogram kan met die woorden worden ontcijferd, totdat in de ontcijferde tekst een (deel van een) woord herkend wordt. Vaak kan dan naar voren en naar achteren de sleutel worden uitgebouwd door afwisselend in de ontcijferde klaartekst en in de sleutel woorden te completeren of te raden.

Vanuit het gezichtspunt van de informatietheorie werkt deze ontcijfermethode omdat de sleutel onvolledige entropie bevat. In plaats van de maximale entropie van een volstrekt willekeurige reeks, wordt gewerkt met een sleutelentropie gelijk aan die van natuurlijke taal. In [hoofdstuk 7](#) blijkt dat dit een aanzienlijke reductie van het aantal mogelijkheden teweeg brengt.

klaartekst :	VYAND	NADER	TWATE	RLINI	ESTOP	FORTA	SPERE	NPARA	AT
sleutel :	DEINB	OORLI	NGENW	ERDEN	METHA	RDEHA	NDVER	DREVE	NE
cryptogram :	YCIAE	BOUPZ	GCEGA	VCLRV	QWMVP	WRVAA	FSZVV	QGEME	NX
crib :			...WE	RDEN.					..
klaar :			...KW	EZHE.					..
crib :			W	ERDEN					..
klaar :			E	RLINI					..
klaar :			.WATE	RLINI	E....				..
crib :			.GENW	ERDEN	M....				..
crib :			.GENW	ERDEN	MET..				..
klaar :			.WATE	RLINI	EST..				..

4.8 Referenties

- [1] C. A. Deavours and L. Kruh (1985). *Machine Cryptography and Modern Cryptanalysis*. Artech House, Inc..
- [2] G. Mellen (1984). Cryptanalysts' Corner. *Cryptologia*, VIII(1), 55–57.
G. Mellen (1985). Cryptanalysts' Corner. *Cryptologia*, IX(1), 70.
- [3] G. Mellen and L. Greenwood (1977). The Cryptology of Multiplex Systems. *Cryptologia*, I(1), 4–16.
- [4] L. D. Callimahos and W. F. Friedmann (1985). *Military Cryptanalytics*. Aegean Park Press. Part 2, volume 2, p. 444.
- [5] H. Rohrbach (1979). Report on the Decipherment of the American Strip Cipher 0-2 by the German Foreign Office (Marburg 1945). *Cryptologia*, III(1), 16–26. Cryptoanalyse uitsluitend gebaseerd op afgeluisterd materiaal van Amerikaanse diplomatiek berichtenverkeer.
- [6] D. E. Knuth (1998b). *The Art of Computer Programming, Volume 2, Seminumerical Algorithms*. third edition Addison-Wesley.

- [7] J. Boyar (1989). Inferring Sequences Produced by Pseudo-Random Number Generators. *Journal of the ACM*, 36(1), 129–141. De methode doet denken aan de Berlekamp-Massey algoritme voor de bepaling van het lineair equivalent van schuifregisterrijen. Gaandeweg wordt een steeds betere uitdrukking voor de generator gevonden.
- [8] G. Marsaglia (2003). Random Number Generators. *Journal of Modern Applied Statistical Methods*, 2(1), 2–13.
- [9] M. D. MacLaren and G. Marsaglia (1965). Uniform Random Number Generators. *Journal of the ACM*, 12, 83–89.
- [10] D. E. Knuth (1998b). *The Art of Computer Programming, Volume 2, Seminumerical Algorithms*. third edition Addison-Wesley.
- [11] L. Kruh (1983). Cipher Equipment, The Cryptographic Unit CSI-10. *Cryptologia*, VII(1), 83–88.
- [12] C. T. Retter (1984). Cryptanalysis of a MacLaren-Marsaglia System. *Cryptologia*, VIII(2), 97–108.

5

Beroemde vercijfermachines

5.1	Hagelin	164
5.2	Enigma	173
5.3	Purple	178
5.4	Referenties	182

Het midden van de 20^e-eeuw, te beginnen zo rond 1920, is de glorie tijd van de cryptografische machines. In Amerika ontwikkelde Edward Hebern een vercijfermachine. De door hem opgerichte firma Hebern Electric Code Inc. haalde bijna een belangrijk contract met het Amerikaanse leger binnen, maar miste dat op de valreep toen William Friedman een serie van tien proefcryptogrammen wist te breken.

In Zweden begint een lijn met Arvid Damm die door Boris Hagelin zal worden voortgezet in een reeks buitengewoon succesvolle machines. Deze zouden Hagelin tot miljonair maken – een van de weinige uitvinders van cryptoapparatuur die dat gelukt is – en maakte van zijn firma Crypto A.G. een van de belangrijkste producenten van deze apparatuur.

In Duitsland benutte Arthur Scherbius het via de Nederlander Hugo Koch verkregen patent voor de ontwikkeling van een reeks rotormachines, de Enigma's. Uiteindelijk zouden deze machines in de Tweede Wereldoorlog in alle onderdelen van de Duitse legers gebruikt worden.

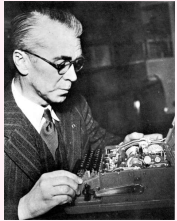
In Japan werd aan de vooravond van de Tweede Wereldoorlog een geheel eigensoortige cryptograaf ontwikkeld, de Angooki Taipu B. Deze is dienst gaan doen voor de vercijfering van de diplomatieke correspondentie op het allerhoogste niveau. Het breken van Purple, zoals deze machine door de Amerikaanse cryptoanalisten werd genoemd, zou de geallieerden een schat aan hoogwaardige informatie brengen.

Vele machines volgden. Om er enkele te noemen: de Russische Fialka, de Amerikaanse Sigaba en de op z'n minst tot midden jaren 1980 door de NAVO gebruikte KL7 codenaam Adonis*. Van de cryptografen zijn de Hagelin, de Enigma en Purple interessant genoeg om hun werking te behandelen. Niet alleen vanwege het grote belang dat deze machines hebben gehad, maar ook omdat ze alledrie een verschillend principe voor vercijfering benutten en zo een aardig inkijkje in het spectrum van mogelijkheden bieden.

* Pas toen de Walker spionage groep in 1984 werd opgerold, bleek dat de Russen 17 jaar lang konden beschikken over sleutel-materiaal en details van deze machine. Alle KL7's zijn toen uit dienst genomen.

5.1 Hagelin

Boris Caesar Wilhelm Hagelin (1892–1983) was de zoon van een Zweedse diplomaat en werd in de Kaukasus geboren. In Sint Petersburg en Stockholm studeerde hij voor ingenieur en behaalde in 1914 het diploma. In 1922 werd hij als vertegenwoordiger van de familiebelangen opgenomen in de firma Aktiebolaget Cryptograph. Deze firma, later verhuisd naar Zug in Zwitserland onder de naam Crypto A.G., is vooral bekend geworden door een tweetal machinefamilies^[1].



Hagelin



Figuur 5.1 Hagelin Model B-211

De bekendste representant van de eerste familie is de Hagelin B-211. Deze werd in 1925 ontwikkeld als concurrent van de Enigma met het oog op een order van het Zweedse leger. De B-machines werken volgens het principe van de fractionerende systemen (sectie 3.4).

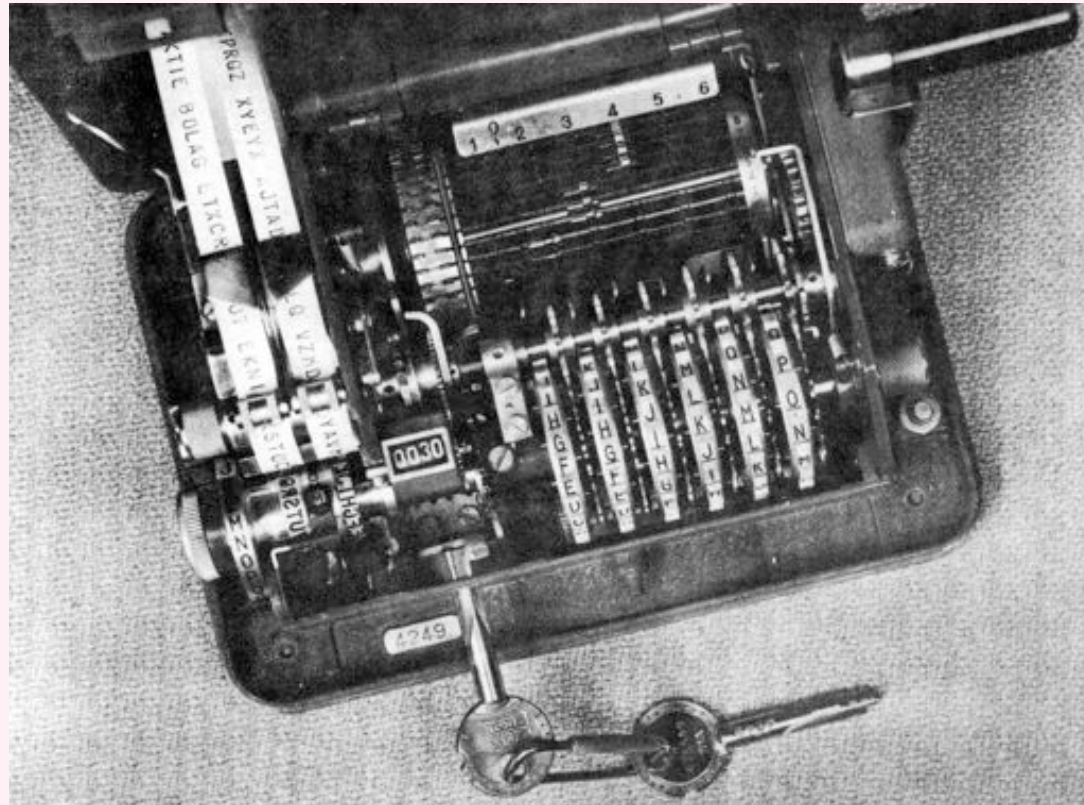
In dit verband is het interessant om op te merken dat Nederland ooit een *mol* in de Amerikaanse NSA had. Joseph Sidney Petersen was een cryptoanalist van de NSA en een goede kennis van zijn Nederlandse collega

J.A. Verkuyl. Verkuyl was ten tijde van de Tweede Wereldoorlog in de Verenigde Staten en heeft samen met Petersen gewerkt aan de cryptoanalyse van Japanse diplomatieke berichten. Petersen waarschuwde in 1948 de Nederlanders door het overhandigen van fotokopieën van *Analysis of the Hagelin Cryptograph Hagelin B-211* aan een medewerker van de Nederlandse ambassade, dat deze machine al voor de oorlog door William Friedman en consorten was gebroken^[2]. In Nederland was dit apparaat nog in gebruik voor diplomatiek berichtenverkeer. Petersen werd hiervoor in 1954 gearresteerd en veroordeeld tot zeven jaar waarvan hij er vier heeft uitgezeten^[3]. In een door NSA in 2010 in het kader van de *Cryptologic Almanac 50th Anniversary Series* vrijgegeven document wordt Petersen gerekend tot de "Betrayers of the Trust".

De meest bekende en beroemdste machines zijn die uit de C-serie. Hun ontwikkeling volgde uit de vraag van de Franse regering naar een draagbaar apparaat^[4]. Het eerste model was de Hagelin C-35 (het getal 35 slaat op het jaar waarin de machine is ontwikkeld), later gevolgd door de Hagelin C-38. De serie is een lang leven beschoren geweest, getuige modelnummers als de Hagelin C-52 en Hagelin C-57. Het zijn deze machines die de Hagelin tot een begrip hebben gemaakt.

De allerberoemdste en meest bekende van de Hagelin machines is ongetwijfeld de Hagelin M-209. In 1940 slaagde Hagelin er op het nippertje in om via Duitsland en Genua per passagierschip Europa te verlaten en naar Amerika te gaan. In zijn koffer zaten de blauwdrukken en enkele gedemonteerde cryptografen van het model C-38. Deze werd wat vereenvoudigd en ging als M-209 een gouden toekomst tegemoet. Het handzame en draagbare apparaat is in grote getale geproduceerd voor de lagere echelons van het Amerikaanse leger. Er zijn er meer dan honderdduizend van gemaakt. Het eerste operationele gebruik zag de machine in 1942 bij Operation Torch, de landingen in Noord-Afrika. Tot in de Koreaanse oorlog (1950–1953) heeft de machine dienst gedaan. Het Nederlandse leger heeft eveneens een verwante versie gebruikt. De M-209 is onderwerp geweest van vele cryptoanalytische studies. Een van de eerste artikelen over de cryptoanalyse van de Hagelin in de open literatuur verscheen in 1978, ongetwijfeld niet tot ieders vreugde^[5].

Het cryptografisch systeem in deze Hagelin is de Beaufort-substitutie (sectie 4.1.3). Het voordeel van de



Figuur 5.2 Hagelin Model C

Beaufort is dat vercijferen en ontcijferen op dezelfde manier gaat, zodat kan worden volstaan met een enkel mechanisme in plaats van een aparte vercijfer- en ontcijfereenheid. In essentie maakt de Hagelin M-209 een polyalfabetische substitutie met normaal alfabet maar met een zeer lange sleutel. De machine bevat een paar handigheidjes voor het verwerken van berichten waarvan de tekst in woorden is verdeeld. Bij het vercijferen wordt de cijfertekst op de printstrook automatisch verdeeld in groepen van vijf en bij het ontcijferen wordt de letter Z niet afgedrukt maar geeft een spatie^{*}. Voor het vercijferen van code is

er een aparte stand waarin dit mechanisme is uitgeschakeld.

Figuur 5.2 toont een Hagelin, in dit geval een model C-38. Rechtsvooraan is een zetal wielen met lettermarkeringen op de omtrek zichtbaar. Deze wielen zitten samen op een as gemonteerd en doen bij elke vercijfering alle tegelijk een stap*. Het aantal posities verschilt per wiel† en is in dit model van rechts naar links respectievelijk 17, 19, 21, 23, 25 en 26. Doordat deze aantallen geen factor gemeen hebben ontstaat een periode van lengte $17 \times 19 \times 21 \times 23 \times 25 \times 26 = 101.405.848$. Op iedere wielpositie bevindt zich een *codeerpin* die in twee standen gezet kan worden en de wielen heten daarom pinwielen. Het codeermechanisme wordt door deze pinnen gestuurd.

Tegenover de pinwielen, rechtsachter in de machine, bevindt zich een halfronde kooi met 27 staafjes, de *linealen* genoemd‡. Op elke lineaal ziet men twee verschuifbare lipjes, de zogenaamde *ruiters*** . Deze ruiters kunnen op acht posities staan, een tegenover elk pinwiel plus twee neutrale posities aan de uiteinden. Een lineaal kan zodoende 0, 1 of 2 actieve ruiters hebben. Op de foto zijn duidelijk twee rijtjes ruiters op de posities 3 en 4 te zien. Samen met de beginstand van de zes wielen en de status van de 131 pinnen vormen de 54 ruiters de sleutel tot de vercijfering.

Tussen de pinwielen en de kooi met linealen bevindt zich een zetal nokjes. Sommige nokjes (zoals die op positie 4 en 5 in **figuur 5.3**) bevinden zich dichterbij de kooi dan andere (zoals dat op positie 6). De plaats van de nokjes is het gevolg van de stand van de pinnen op de bijbehorende pinwielen. Staat een pin in de *actieve* stand dan duwt deze het bijbehorende nokje naar voren, een *inactieve* pin laat het nokje op z'n plaats. Telkens wanneer bij een vercijfering de pinwielen een stap verder draaien, verandert het patroon van actieve en inactieve pinnen en daarmee de stand van de nokjes. De zes pinposities zorgen voor 2^6

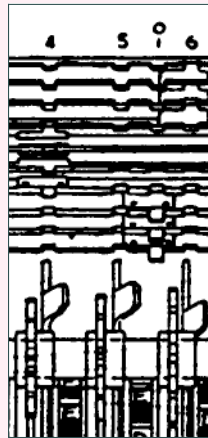
* In sommige uitvoeringen is dit de letter X.

* In latere modellen is deze voortbeweging onregelmatiger gemaakt en afhankelijk van de pinsetting op een naastliggend wiel.

† In het oudste model C-35 ontbreekt het 26-wiel, in latere modellen is het aantal posities opgevoerd tot een maximum van 47.

‡ De oudere C-35 heeft 25 linealen, de latere C-57 heeft er 40.

** De C-35 en de C-57 hebben slechts één ruiters per lineaal.

*Figuur 5.3*

= 64 verschillende combinaties in de stand van de nokjes. Deze 64 standen zullen door de interactie tussen de kooi en de nokjes vertaald worden naar de 26 sleutelletters van de Beaufort-vercijfering.

Voordat het vercijferen en ontcijferen kan beginnen moeten de pinnen en de ruiters worden ingesteld. Operationeel kan dit zijn voorbehouden aan een ander dan de operator, bijvoorbeeld aan een bevelvoerende officier. In dit verband is het aardig om te weten dat de twee sleutels op de voorgrond in [figuur 5.2](#) niet gelijk zijn. Een van de twee geeft slechts toegang tot het uitwendig vercijfermechanisme, met de andere kan men ook in het inwendige komen voor het veranderen van de instellingen*.

Met 64 verschillende pinconfiguraties en 26 substitutiealfabetten is het niet mogelijk om de pinconfiguraties gelijkelijk over de substitutiealfabetten te verdelen. Daarnaast is de keuze van het aantal actieve linealen en de verdeling tussen actieve en inactieve pinnen van belang voor de kwaliteit van de sleutelstroom. Sommige instellingen maken de machine kwetsbaarder voor cryptanalyse dan andere. De

* Na gebruik van de machine behoort men alle pinnen op inactief en alle ruiters op nul te zetten. Dit voor het geval het apparaat onverhoopt in handen van de vijand valt.

instellingsparameters moeten dus zorgvuldig geselecteerd worden.

Het vercijfermechanisme wordt aangedreven door de hendel aan de rechterzijde. Na het verdraaien van de pinwielen naar de beginpositie stelt de operateur de te vercijferen letter in op de letterschijf geheel linksvooraan (links van de twee gekoppelde letterwielen) en haalt de hendel over. Het overhalen van de hendel geeft de kooi een draaiende beweging, waarna deze aan het eind van de slag weer terugspringt. Bij de draaiing lopen de ruiters langs de nokjes. Wanneer een ruiter langs een vooruitgeschoven nokje komt wordt het nokje (dat een beetje scheef staat) opzij geduwd. Deze zijdelingse beweging wordt omgezet in een draaiende beweging van het letterwielduo linksvooraan tegenover de twee papierstroken. Iedere lineaal met een ruiter op een plaats waar het nokje door een actieve pin vooruitgeschoven is, doet dit duo een stap draaien. De twee wieltes vormen samen een Beaufort-klaar- en cijferalfabet. Elke verdraaiing ervan doet de vercijfersleutel één letter opschuiven. Aan de linkerkant van de speciale Beaufort-tabel (zie [tabel 5.1](#)) staan getallen in plaats van de gebruikelijke sleutelletters. Deze getallen beduiden het aantal linealen dat door de draaiende kooi is aangestuurd.

Bij vergelijking van deze tabel met [tabel 4.3](#) ziet men dat de sleutelregels 25 plaatsen naar beneden verschoven zijn, waardoor de eerste sleutelletter niet de A maar de Z is. De verschuiving van A naar Z heet de *slide*. Bij de M-209 heeft deze slide de vaste waarde 25, maar op de vrijwel identieke C-38 is de slide instelbaar op alle 26 mogelijke waarden.

Met P de klaartekstletter $A=0, B=1, \dots, K$ de sleutelletter en C het vercijferresultaat kan de gewone Beaufort-substitutie in formule worden weergegeven als

$$P + C = K \bmod 26 \quad (5.1)$$

Deze formule valt het best te begrijpen door K te zien als het aantal stappen van een Caesar-substitutie en dan te bedenken dat het alfabet in de tabel van Z naar A terugloopt. Introduceer een extra variabele S voor de slide in de sleutelletter en de vercijfering van de Hagelin wordt nu gegeven door

$$P + C = K + S \bmod 26 \quad (5.2)$$

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0,26	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A
1,27	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B
2	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C
3	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D
4	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E
5	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F
6	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G
7	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H
8	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I
9	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J
10	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K
11	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L
12	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M
13	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N
14	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O
15	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P
16	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q
17	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R
18	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S
19	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T
20	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U
21	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V
22	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W
23	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X
24	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y
25	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Tabel 5.1 Beaufort-tabel ingericht voor M-209 vercijfering

Met 27 linealen kan het voorkomen dat de codeerwielen meer dan 25 stappen maken. Dat geeft in de codeertabel aanleiding tot *wraparound* en heeft de bedoeling het breken van het geheimschrift te bemoeilijken. Doordat 0 en 26 of 1 en 27 stappen eenzelfde vercijferresultaat geven, ontstaat extra onzekerheid bij de reconstructie van de vercijfersleutel. In de C-57 met 40 linealen is dit effect nog sterker.

Als op een gegeven lineaal twee ruiters zijn geactiveerd, dat wil zeggen opgesteld staan tegenover een pinwiel, dan zijn er drie mogelijkheden. Eerstens kan geen van de beide pinwielen actief zijn en dan heeft deze lineaal geen effect. Staat tegenover een van de twee een actief pinwiel dan brengt dit de sleutel een stap verder. Tenslotte kunnen beide ruiters een actief pinwiel tegenover zich vinden, maar ook dan resulteert dit in slechts één stap in de sleutel: de twee ruiters vertonen *overlap* en tellen samen slechts voor één. Zonder deze mogelijkheid tot overlap zou het aantal sleutelstappen altijd gelijk zijn aan de som van het aantal pinnen op de actieve pinwielen, met overlap wordt dat ten hoogste gelijk aan die som. Extra onzekerheid dus om de analyse te bemoeilijken*. Men moet natuurlijk de machine wel zodanig instellen dat overlap daadwerkelijk in het spel komt.

Een paar voorbeelden. Eerst van de draaiing van de de letterwielen om te laten zien hoe de Beaufort-substitutie met en zonder slide tot stand komt. Kies slide A en vercijfer de letter C. De twee gekoppelde wielen, waarvan het ene het indicatorwiel en het andere het printwiel is, staan in de aangegeven stand met het sterretje op de plaats van de afdrukpositie. Let op het teruglopend alfabet op het printwiel.

	*
indicatorwiel :	HIJKLMNOPQRSTUVWXYZABCDEFG
printwiel :	TSRQPONMLKJIHGFEDCBAZYXWVU
	*

Laat de machine werken en veronderstel dat deze 12 stappen doet, of in het jargon een "*kick* van 12" krijgt. Nadat de twee wielen gezamenlijk gedraaid zijn, is de situatie als volgt geworden en is de vercijferde letter

* Een bepaalde aanvalsmethode op de C-35, die drijft op het ontbreken van overlap, zal zeker hebben bijgedragen tot de ontwikkeling van deze complicatie.

de K.

```

*
indicatorwiel : VWXYZABCDEFGHIJKLMNQRSTU
printwiel : FEDCBZYXWVUTSRQPONMLKJIHG
*
```

Met een slide van 7 in het volgende voorbeeld zal dit als vercijfering de R geven. Let op de H op het printwiel tegenover de A op het indicatorwiel als gevolg van deze slide.

```

*
indicatorwiel : VWXYZABCDEFGHIJKLMNQRSTU
printwiel : MLKJIHGFEDCBZYXWVUTSRQPON
*
```

Nu wordt getoond hoe de kick wordt opgebouwd uit de instelling van de pinwielen en de ruiters. Voor het gemak zijn slechts het 17, 19 en 21 wiel meegenomen en staan daartegenover respectievelijk 4, 3 en 2 ruiters; de ruiters voor het 17 en 19 wiel vertonen een overlap. Een 0 staat voor een inactieve pin op de betreffende positie, een 1 voor een actieve pin. De vercijfering volgens de M-209 en [tabel 5.1](#) gaat nu als volgt.

```

klaartekst : GEENZNOODZHETZLOOPTZALSZEENZTREIN
17-wiel : 4 000110110001011000001101100010110
19-wiel : 3 110010100001110000111001010000111
21-wiel : 2 010000100001000010001010000100001
kick : 350460840008364020338426430240665
cijfertekst : WAVQGMTPWASDJGSLNKJDHSJGZYMCKIBXR
ontcijfering : GEEN NOOD HET LOOPT ALS EEN TREIN
```

De vercijfering van de Z op de vijfde positie laat het effect van de overlap op het 17 en het 19 wiel zien. De 4 respectievelijk 3 ruiters tellen niet op tot een kick van 7 maar slechts tot een kick van 6.

5.2 Enigma

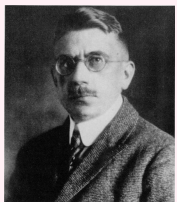
Wie precies de eerste aanzet tot de ontwikkeling van de Enigma-cryptograaf heeft gegeven, is niet voor de volle 100% zeker. Maar het is wel heel waarschijnlijk dat het oorspronkelijke idee afkomstig is van twee Nederlandse marineofficieren, R.P.C. Spengler en Theo A. van Hengel, die ten tijde van de Eerste Wereldoorlog het zogenaamde *proeftoestel* ontwierpen, maar daar van de marineleiding aanvankelijk geen patent op mochten nemen^[6]. Zeker is dat kort daarna de Nederlander Hugo Koch en de Duitser Arthur Scherbius in hun beider landen sterk op elkaar en op het ontwerp van de twee marineofficieren gelijkende patenten op een cryptograaf lieten registreren. Verder staat vast dat Scherbius op 15 april 1918 in een brief aan het Reichs Marineamt een Chiffriermachine aanbiedt. Hoewel er van officiële zijde weinig belangstelling voor bleek te bestaan, ging Scherbius door met het ontwikkelen van een cryptograaf voor commercieel gebruik. Zo ontstond wat de *Enigma* is geworden, een serie verwante machines gebaseerd op vercijfering door een stel rotors (sectie 4.5).

De grote vlucht van de machine begon aan het eind van de jaren twintig, toen de Duitse Generale Staf bij de opbouw van de Wehrmacht zocht naar een geschikt apparaat om de communicatie tussen de legeronderdelen op te baseren. De Enigma werd uitgekozen en verder ontwikkeld. Het apparaat werd in verscheidene varianten in de Tweede Wereldoorlog gebruikt o.a. door de Wehrmacht, Luftwaffe, Kriegsmarine, Abwehr en zelfs de spoorwegen.

Het behoeft geen betoog dat dit overvloedig gebruik van de Enigma door de Duitse strijdkrachten tot grote cryptoanalytische inspanningen bij de geallieerden heeft geleid. En hoewel de Britten in de jaren 1939–1945 in het tussen Oxford en Cambridge gelegen Bletchley Park een geweldige cryptoanalytische machinerie tegen de Enigma hebben opgezet, had de eerste en belangrijkste doorbraak ergens anders plaats. Het heeft lang geduurd voordat de sluier van geheimzinnigheid werd opgelicht. De strikte geheimhoudingsdiscipline uit de oorlogsjaren is vele jaren gehandhaafd gebleven. Pas in 1974 werd met de verschijning van het boek *The Ultra Secret* van F.W. Winterbotham een tipje van de sluier opgelicht^{[7]*}.



Van Hengel



Scherbius

De eerste die technische details over het breken van Enigma prijs gaf was de Brit Gordon Welchman in 1982^[8]. Hij was naar Amerika geëmigreerd en kon zo onder de Britse Secrecy Act uitkomen*. Inmiddels is er over het breken van Enigma een zeer omvangrijke hoeveelheid literatuur verschenen.

De eerste en belangrijkste doorbraak in de Enigma had plaats in Polen. De Poolse wiskundige Marian Rejewski slaagde in 1932–33 waar vele anderen faalden. Hij brak een aantal Enigma-berichten en reconstrueerde de bedrading van de rotors^[9]. De ontcijfering van Enigma-berichten is in het bijzonder van grote betekenis geweest voor de strijd tegen de Duitse onderzeeboten, die de konvooien op de Atlantische Oceaan bedreigden om zo de bevoorrading van Engeland vanuit Amerika af te snijden^[10].

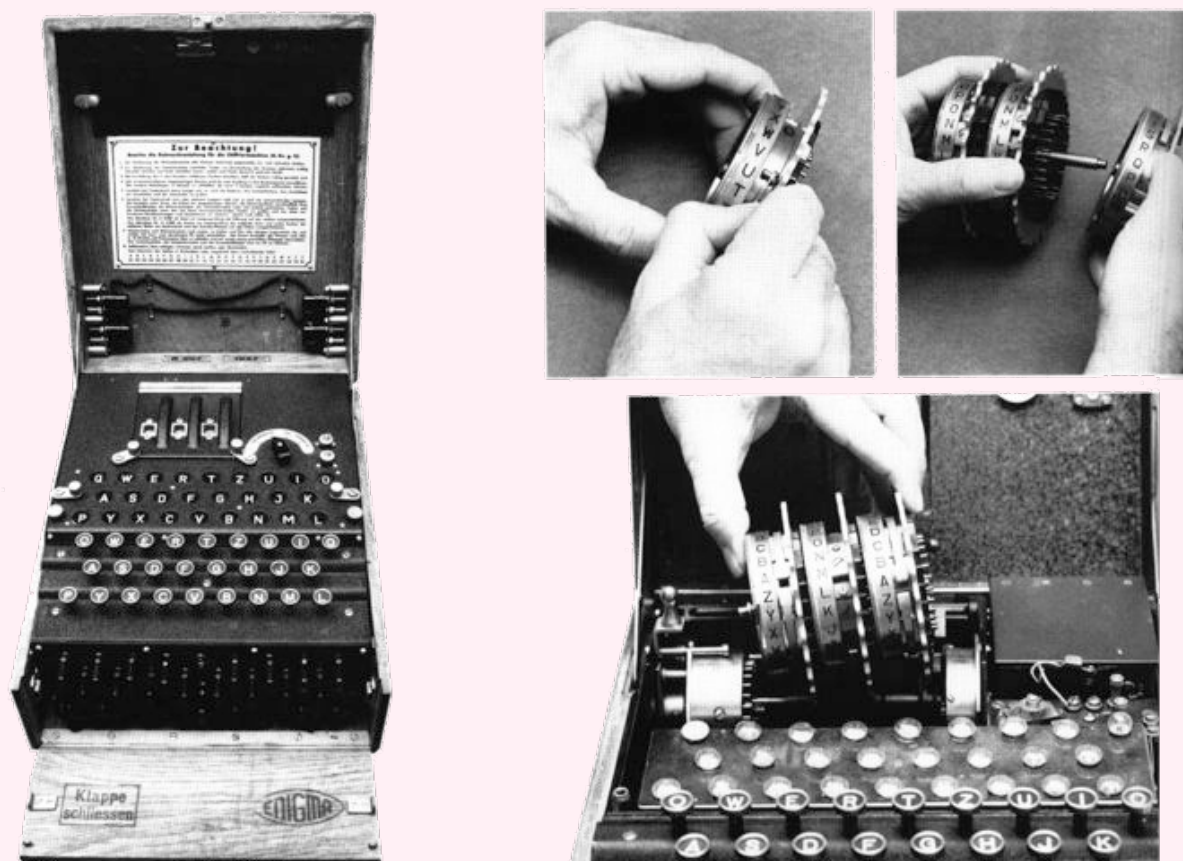
Het hier beschreven model van de Enigma is de door Wehrmacht gebruikte versie, het meest gangbare model. **Figuur 5.4** links is een foto van het apparaat en **figuur 5.5** geeft het schema van de bedrading. De machine is zo ingericht dat vercijferen en ontcijferen op eenzelfde manier gaat. De cryptotransformatie van de Enigma is derhalve een involutie (**sectie 1.7.2**).

Onderaan het toestel in de foto ziet men een QWERTY-toetsenbord en daarboven drie rijen letters met onder elk een gloeilampje. Vercijferen geschiedt door het indrukken van een toets, de vercijferde letter licht dan op. Boven de lampjes bevinden zich de rotors onder drie vensters die de momentane stand laten zien, en naast elk van de vensters de er bovenuitstekende kartelrand van een rotor waarmee deze handmatig kan worden verdraaid.

Aan de voorkant, vlak boven de op de foto zichtbare neergeslagen klap, zitten 26 paar stekkerbusjes, eveneens met letters gemerkt. Dit is het zogenaamde *stekkerbord* (plugboard in het Engels). De Enigma-operator beschikt over een aantal verbindingsstekkers, waarmee twee lettercontacten met elkaar kunnen

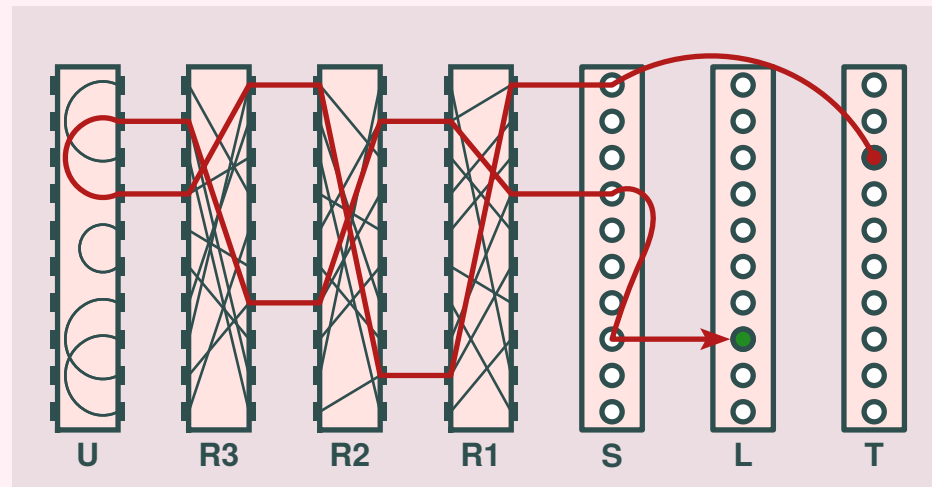
* Winterbotham was in de Tweede Wereldoorlog officier van de RAF en belast met de supervisie over de distributie van de zogeheten Ultra intelligence, de resultaten van o.a. het breken van de Enigma. Hoewel zijn boek niet het eerste was dat ging over het breken van Enigma – en onnauwkeurigheden bevat – heeft het bij verschijnen wel de meeste impact gehad.

* Veel vreugde heeft het hem desondanks niet gebracht, want men heeft hem zijn onthullingen zeer kwalijk genomen. In 2015 is een BBC-documentaire over hem uitgebracht onder de titel "Gordon Welchman – Code-breaking's Forgotten Genius".



Figuur 5.4 links Wehrmacht Enigma, rechts instelling van de rotors

worden verbonden. Dit bewerkstelligt een verwisseling van de beide letters. Is bijvoorbeeld op deze manier de E met de H verbonden en tikt de operateur een E in, dan zal een H naar het rotorhuis gaan. Het stekkerbord bewerkstelligt dus een monoalfabetische substitutie die paren letters verwisselt. Hoe simpel deze monoalfabetische substitutie ook is, ze is gebleken een zeer belangrijk element in de vercijfering te zijn en een aanzienlijke complicatie voor de cryptanalyse.



Figuur 5.5 Schema van Enigmavercijfering

Het hart van de machine wordt gevormd door het stelsel rotors bovenaan op de foto. In [figuur 5.5](#) is de werking van de Enigma te volgen. Bij het indrukken van een toets wordt een stroomkring gesloten die vanaf het toetsenbord T, door het stekkerbord S naar het rotorhuis loopt. Vandaar gaat de stroom door de drie rotors R1-R2-R3 en komt dan in een speciale vierde rotor U – de Umkehrwalze – die de stroom in omgekeerde richting R3-R2-R1 door de drie rotors stuurt. Via de aansluitplaat wordt de rotorset verlaten, gaat de stroom opnieuw door het stekkerbord S en tenslotte naar een van de lampjes L waardoor de vercijferde (ontcijferde) letter oplicht. Het stekkerbord doet dus tweemaal mee, op de heenweg en op de

terugweg. Uit deze gang van zaken is direct duidelijk waarom de Enigma-vercijfering een involutie is en waarom een letter nooit tot zichzelf kan worden vercijferd.

Telkens aan het begin van een vercijfering doet het rotormechanisme een stap. De rotor aan de rechterkant loopt het snelst en neemt eens per omwenteling de rotor links ervan mee, die op zijn beurt weer de derde rotor aanstuurt; de keerrotor beweegt niet. Op het eerste gezicht lijkt zo een periode van $26 \times 26 \times 26$ te ontstaan, maar door een eigenaardigheid in de rotorkoppeling is dat iets minder. Op een gegeven moment doet de middelste rotor tweemaal achtereen een stap, waardoor de periode iets wordt verkort.

Voordat kan worden vercijferd moet de sleutel worden ingesteld. Deze sleutel bestaat uit de volgende elementen:

1. de drie draaiende rotors worden genomen uit een stel van vijf
2. van elke rotor moet het nokje voor het meenemen van de naastliggende rotor naar de juiste positie worden gedraaid
3. de drie gekozen rotors worden in de juiste volgorde aan elkaar gezet en in de machine geplaatst; **figuur 5.4** rechts laat de eerste drie stappen zien.
4. een variërend aantal stekkers wordt in het stekkerbord gestoken

De machine staat nu ingesteld voor de lopende cryptoperiode.

5. voorafgaand aan de vercijfering van elk bericht worden de drie rotors op een willekeurige beginpositie gezet, de zogenaamde Spruchschlüssel*

Sleutels werden centraal aangemaakt en gedistribueerd over de verschillende eenheden. Alle machines met een bepaalde sleutelinstelling vormen een communicatienetwerk. Aanvankelijk waren dat er slechts een paar, maar naarmate de oorlog vorderde kwamen er steeds meer netwerken bij. Deze communica-

* Met een speciale procedure werd deze beginstand vercijferd en als indicator in de kop van het bericht geplaatst, zodat de ontvanger de machine voor de ontcijfering kan instellen.

tiennetwerken worden aangeduid met een codewoord. Het netwerk voor de Duitse onderzeeboten op de Atlantische Oceaan werd door de Kriegsmarine *Triton* genoemd en door de Engelse codebrekers *Shark*.

5.3 Purple

Tijdens de vlootconferentie van 1921 genoten de Amerikanen het immense voordeel dat ze de geheime instructies aan de Japanse delegatie kenden. Daardoor konden ze de afgesproken omvang van de Japanse vloot veel verder inperken dan de Japanse regering zich ten doel gesteld had. Dit voordeel dankten de Amerikanen aan Herbert O. Yardley, die tezamen met een kleine staf erin was geslaagd om de Japanse diplomatieke codes te breken.

Toen echter in 1929 de Amerikaanse regering Yardley's 'zwarte kamer' sloot, kwam deze zonder veel middelen van bestaan op straat te staan*. Desperaat op zoek naar een inkomstenbron schreef hij in korte tijd het boek *The American Black Chamber* over zijn werk als codebreker^[11]. Dit boek werd een groot success en verscheen zelfs als feuilleton. Anderen waren er minder blij mee, of zelfs verontwaardigd, omdat Yardley in zijn boek teveel geheimen verklapte. Maar de grootste impact had het boek in Japan, waar het insloeg als een bom. Het verhaal over de vlootconferentie van 1921 en hoe Japan daar naar de zijlijn was gemanoevreerd, betekende gezichtsverlies voor het oog van de gehele wereld. Het mag geen wonder heten dat de Japanse diplomatieke communicatie vanaf dat moment grondig werd herzien en de codes werden vervangen door machines.

Een aantal opeenvolgende machines verscheen. De belangrijkste daarvan is de zogenaamde 97-shiki-obun In-ji-ki (Codeermachine 97) ook genaamd Angooki Taipu B en door de Amerikanen aangeduid met de codenaam *Purple*. Een voorganger was de *Red* gedoopte machine. Deze voorganger is van belang omdat er een eigenaardige constructie in voorkwam. Namelijk een splitsing van de 26 letters van het alfabet in



Yardley

* De politicus in kwestie was Henry Stimson, die zijn besluit zou hebben gemotiveerd met de gedenkwaardige woorden "Gentlemen do not read each other's mail".

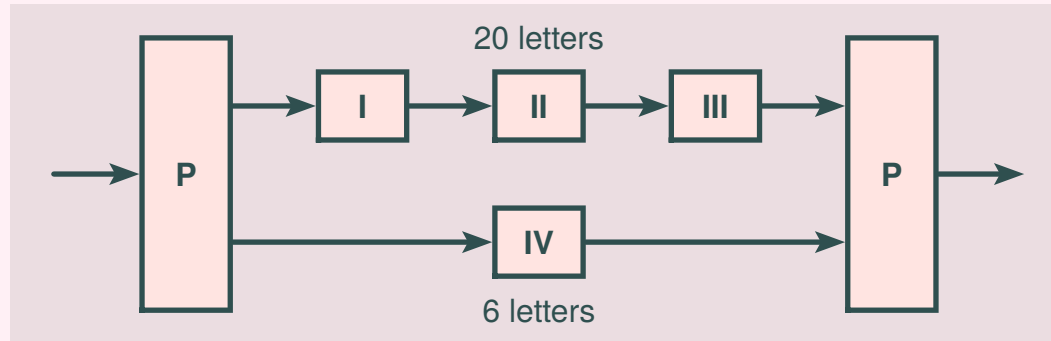
twee groepen, de 6 klinkers en de 20 medeklinkers. Die groepen werden in Red afzonderlijk vercijferd. De reden voor die opsplitsing ligt voor de hand. Om telegrammen te kunnen verzenden via openbare telegraafkantoren werd de Japanse tekst in het Romaji systeem – dat veel klinkers oplevert – getranscribeerd naar ons gewone alfabet. Door bij het coderen de klinkers onderelkaar te permuteren ontstaat een cijfertekst die min of meer valt uit te spreken. Met als gevolg daarvan het lagere telegramtarief dat voor zulke codetelegrammen gold. In Purple is de 6-20 splitsing gehandhaafd, zij het dat hier de klinkers niet meer uitsluitend onder elkaar worden gepermuteerd. De splitsing vormt echter een zwakke stee die in de cryptoanalyse kon worden geëxploiteerd.

In de loop van 1939 werd de Purple-codeermachine operationeel in de belangrijkste ambassades, waaronder die in Washington, USA. De Amerikaanse marine en het leger werkten samen bij de cryptoanalyse van de Japanse communicatie. Een team onder leiding van William Friedman heeft lang gezwoegd om tenslotte eind 1940 zijn pogingen tot cryptoanalyse te zien beloond. In het memo *Preliminary Historical Report on the Solution of the "B" Machine* van 14 oktober 1940 doet William Friedman daarvan kond. De impact van deze geslaagde cryptoanalyse is groot geweest. De Amerikanen gaven aan het product de naam *Magic*. Om een voorbeeld te geven van het belang: de Japanse ambassadeur in Berlijn werd rondgeleid langs de fortificaties die werden aangelegd langs de Atlantische kust en bracht daarvan uitvoerig verslag uit aan Tokyo – uiteraard in de Purple-code.

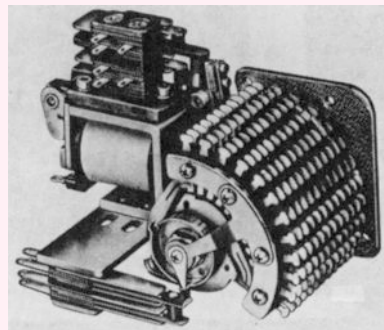
De hoofdstructuur van de machine is afgebeeld in [figuur 5.6](#). De invoer afkomstig van het toetsenbord is aan de linkerkant. De letters ondergaan een permutatie en worden gesplitst in een groep van 6 en een groep van 20 letters. De 6 letters ondergaan één van 25 verschillende monoalfabetische substituties (IV in de figuur) en verschijnen na een permutatie aan de uitgang rechts. De 20 overige letters ondergaan drie achtereenvolgende substituties (I, II en III in de figuur). Deze drie substituties bestaan elk ook weer uit 25 verschillende substituties waaruit er telkens een gekozen wordt. Van deze drie wisselt er een per vercijferde letter (de snelle), een tweede eenmaal voor elke cyclus van de eerste (de matige) en de derde wisselt als de tweede een cyclus heeft voltooid (de langzame).



Friedman



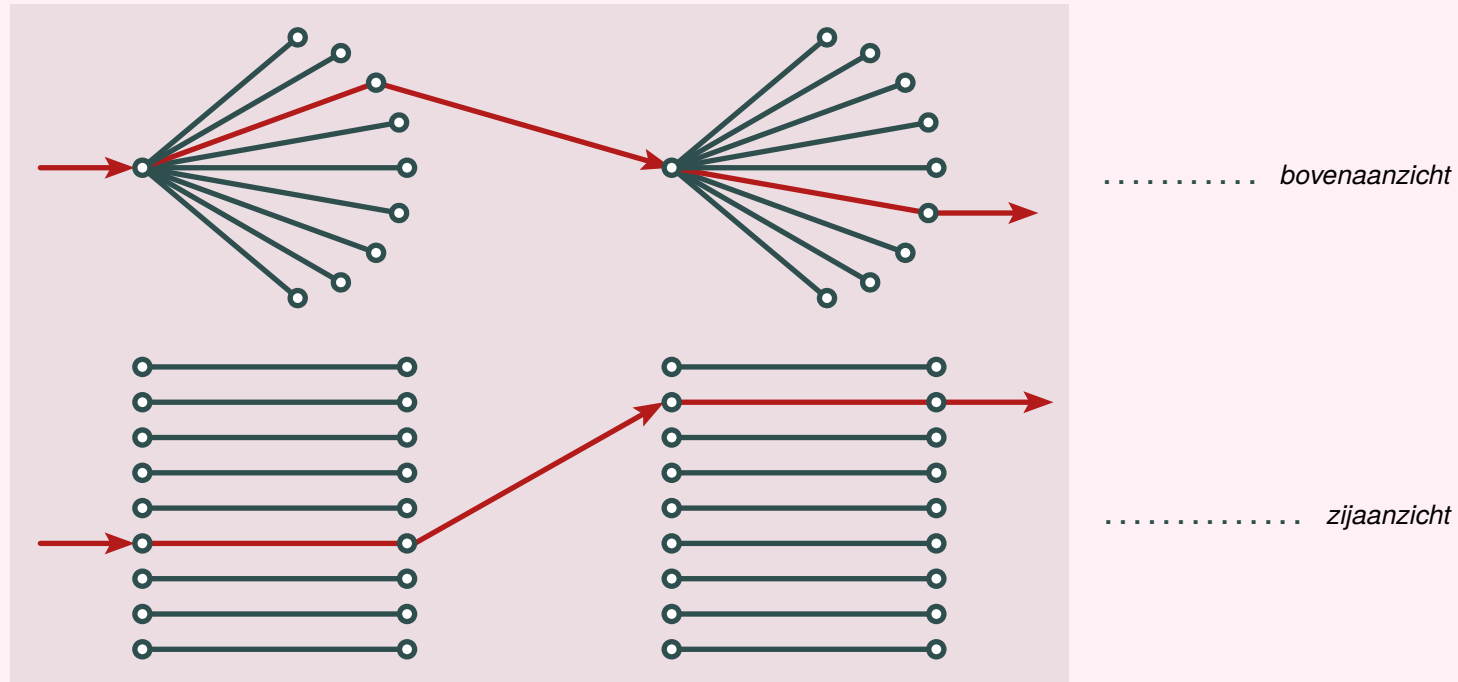
Figuur 5.6 Purple mechanisme



Figuur 5.7 Stappenschakelaar

Het stappenmechanisme wordt gestuurd vanuit blok IV en is zo ingericht dat nooit meer dan één van de substituties I, II en III tegelijk verandert. Bovendien kan de volgorde van de langzame, de matige en de snelle beweging naar believen worden toegekend aan de drie substitutie-elementen. Voor de details kan worden verwezen naar de uitvoerige beschrijving in een artikel dat in 2003 in het tijdschrift *Cryptologia* is verschenen^[12].

De basiselementen van het substitutiemechanisme worden gevormd door stappenschakelaars zoals deze voorkwamen in de vroegere, niet digitale telefooncentrales. [Figuur 5.7](#) toont een voorbeeld hiervan. Zo'n



Figuur 5.8 Achtereenvolgende substituties

schakelaar bestaat uit een aantal lagen op een gemeenschappelijke as. Elke laag is een waaier van contacten waarlangs een beweegbare contactarm loopt. Op elke laag kan de stroom door de contactarm worden afgenomen van het contactpunt. In Purple hebben deze schakelaars 25 contacten op de waaier en zijn er in eenheden I, II en III zoveel opelkaar gestapeld dat de benodigde substituties tussen de inkomende 20 letters kunnen worden gemaakt.

Het bovenste schema in [figuur 5.8](#) geeft twee opeenvolgende substitutieschakelaars in bovenaanzicht, het onderste schema toont ze in zijaanzicht. Van links komt de te vercijferen letter binnen op een van de lagen (te zien in het onderste schema) en verlaat deze via de contactarm op het gekozen waaiercontact (te

zien in het bovenste schema). De substitutie gaat verder op een andere laag in de volgende schakelaar (te zien in het onderste schema). Elk van de 25 kolommen met contacten is door draadjes in gepermuteerde volgorde verbonden met de invoer van de volgende substitutieschakelaar. Met 25 contacten per waaier bewerkstelligt elke stappenschakelaar dus 25 onafhankelijke monoalfabetische substituties.

5.4 Referenties

- [1] B. C. W. Hagelin (1994). The Story of the Hagelin Cryptos. *Cryptologia*, XVIII(3), 204–242. Posthuum gepubliceerd door David Kahn.
- [2] D. Kahn (1967). *The Codebreakers, The Story of Secret Writing*. MacMillan Publ. Co., Inc.
- [3] C. Wiebes (2008). Operation Piet: The Joseph Sidney Petersen Jr. Spy Case, a Dutch Mole inside the National Security Agency. *Intelligence and National Security*, 23(4), 488–535.
- [4] C. A. Deavours and L. Kruh (1985). *Machine Cryptography and Modern Cryptanalysis*. Artech House, Inc..
- [5] R. Morris (1978). The Hagelin Cipher Machine (M-209), Reconstruction of the Internal Settings. *Cryptologia*, II(3), 267–289.
- [6] K. de Leeuw (2003). The Dutch Invention of the Rotor Machine, 1915–1923. *Cryptologia*, 27(2), 73–94.
- [7] F.W. Winterbotham (1974). *The Ultra Secret*. HarperCollins Publ.
F.W. Winterbotham (1999). *The Ultra Secret*. Weidenfeld & Nicolson.
- [8] G. Welchman (1982). *The Hut Six Story, Breaking the Enigma Codes*. Allen Lane, London.
- [9] J. Garliński (1979). *Intercept, The Enigma War*. J.M. Dent & Sons Ltd, Magnum Edition 1981.
W. Kozaczuk (1989). *Geheimoperation Wicher*. Bernard & Graefe Verlag, Koblenz.
M. Rejewski (1981). How Polish Mathematicians Deciphered the Enigma. *Annals of the History of Computing*, 3(3), 213–234.
M. Rejewski (1982). Mathematical Solution of the Enigma Cipher. *Cryptologia*, VI(1), 1–18.
- [10] D. Kahn (1991a). *Seizing the Enigma, The Race to Break the German U-Boat Codes, 1939–1943*. Houghton Mifflin Company, Boston.
J. Brennecke (1984). *Die Wende im U-Boot Krieg, Ursachen und Folgen 1939–1943*. Koehlers Verlagsgesellschaft mbH, Herford.

Ministry of Defence (Navy) (1989). *German Naval History, The U-Boat War in the Atlantic, 1939–1945*. Her Majesty's Stationery Office, London.

[11] H.O. Yardley (1931). *The American Black Chamber*. Bobbs-Merrill Cie, Indianapolis.

[12] W. Freeman, G. Sullivan and F. Weierud (2003). Purple Revealed: Simulation and Computer-Aided Cryptanalysis of Angooki Taipu B. *Cryptologia*, XXVII(1), 1–43.

6 Cryptoanalyse

6.1	Korte inhoud	185
6.2	Letterfrequentie	185
6.3	Tweede moment	187
6.3.1	Benadering tweede moment	190
6.3.2	Bepaling monoalfabeticiteit	193
6.3.3	Bepaling periode	195
6.3.4	Bepaling aantal letters	198
6.4	Coïncidenties	200
6.4.1	Poisson verdeling	202
6.4.2	Chi-test	204
6.5	Klinkers en medeklinkers	209
6.5.1	Consonantlijn	209
6.5.2	Sukhotin	217
6.6	Symmetrie in positie	219
6.7	Isomorfie	227
6.7.1	Reconstructie alfabet	231
6.7.2	U.S. Army Cipher Disk	237
6.8	Referenties	242

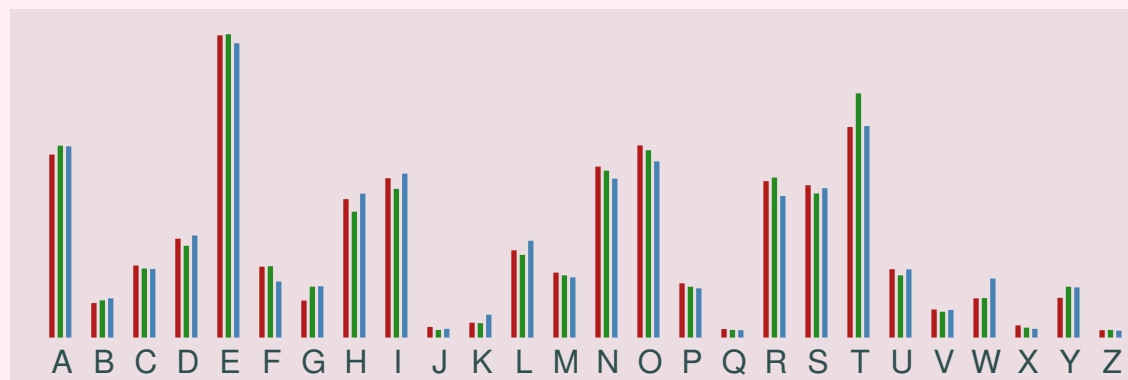
6.1 Korte inhoud

In dit hoofdstuk worden enige technieken en hulpmiddelen uit de cryptoanalyse behandeld. Het is in het bestek van dit werk onmogelijk om alle methoden van aanpak voor alle mogelijke cryptosystemen te behandelen, zelfs met beperking tot de klassieke systemen. Maar er zijn een paar methoden en gereedschappen van algemene aard, die tegen veel systemen kunnen worden ingezet. Enkele uitgewerkte cryptoanalyses illustreren deze technieken.

- Statistische methoden, in het bijzonder het benutten van de letterdistributie. Deze methoden kunnen worden gebruikt voor verschillende doeleinden, zoals het bepalen van de aard van een geheimschrift, het bepalen van de periode en het matchen van de sleutelletters van periodieke geheimschriften.
- Onderscheid tussen klinkers en medeklinkers. Het oplossen van monoalfabetische substituties wordt eenvoudiger als tevoren onderscheid kan worden gemaakt tussen de substituenten voor klinkers en die voor medeklinkers. Twee methoden worden behandeld, de consonantlijn en de Sukhotin algoritme.
- De reconstructie van het alfabet in periodieke polyalfabeten wordt vergemakkelijkt als substituenten op onderscheiden posities in de sleutel aan elkaar gekoppeld kunnen worden. Een techniek die bekend staat als symmetrie in positie systematiseert dit.
- Isomorfie is een generalisatie van de herhaling van elementen in het cryptogram. In het geval van een periodieke polyalfabeet met gemengd cijferalfabet kan, door gebruik te maken van optredende isomorfen, dit alfabet zelfs worden bepaald zonder ook maar een letter te hebben ontcijferd.

6.2 Letterfrequentie

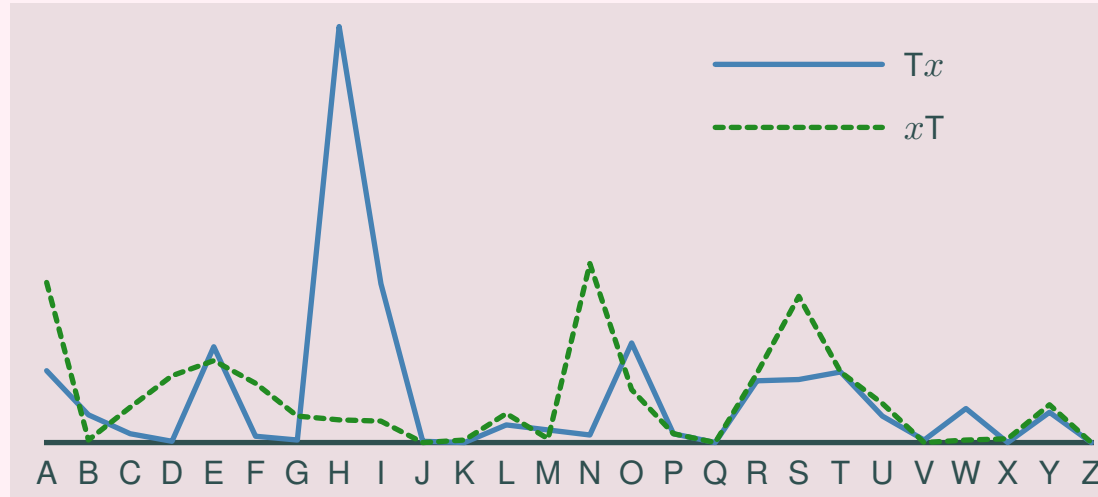
In het eerste hoofdstuk is al gewezen op het fenomeen letterdistributie en het belang hiervan voor het oplossen van cryptogrammen. Hier zal daaraan een kwantitatieve, statistische basis worden gegeven.



Figuur 6.1 Letterfrequenties Engels uit resp. 1933, 1942 en 1982

Waarom is het mogelijk letterfrequenties te gebruiken als instrument in de cryptoanalyse? De reden is dat voor elke taal de frequentiedistributie van de letters in het alfabet in hoge mate constant blijkt. Verschillen tussen de diverse talen zijn er natuurlijk wel. En ook als men individuele, relatief korte stukjes tekst neemt, kunnen vrij grote verschillen in de letterfrequenties optreden. Maar zodra sprake is van flinke hoeveelheden tekst in eenzelfde taal zal opvallen hoe sterk de frequentietellingen op elkaar lijken. In het Engels zal de letter E vrijwel altijd het meest voorkomen en de letter Z zal altijd zeldzaam zijn. Dat geldt zelfs voor teksten die in de tijd betrekkelijk ver uiteen liggen. Om dit te onderstrepen geeft [figuur 6.1](#) voor elke letter drie tellingen. De telling aan de linkerkant stamt uit het jaar 1933, de middelste komt uit een Engelse cursus voor cryptoanalisten van 1942 en die rechts uit tellingen van krantenartikelen en romans uit 1982^[1].

Het tellen hoeft niet te worden beperkt tot de enkele letters, want digram- en trigramfrequenties zijn zeker zo nuttig. Bij de contacteigenschappen van letterparen in het Engels springt bijvoorbeeld het opvallend verschil in frequentie tussen TH en HT in het oog. [Figuur 6.2](#) laat het verschil in contactfrequentie zien tussen letters voorafgaand en volgend op de T. Een ander algemeen verschijnsel is de uitgesproken voorkeur voor een klinker links van de letter N. En zo zijn er nog meer kenmerken die de cryptoanalist



Figuur 6.2 Digrammen Tx en xT in Engels

helpen. Een complete telling van alle 676 digrammen voor de Engelse taal staat in [tabel 6.1](#). Deze tabel is geschaald op een totaal van 10.000 digrammen.

6.3 Tweede moment

In de statistiek gebruikt men voor het kwantificeren van een frequentiedistributie de *momenten* van die verdeling. Deze momenten worden hier aangeduid met de letter S . Zo staat S_1 voor het eerste moment, S_2 voor het tweede moment, enz. Het eerste moment is het gemiddelde en levert weinig op. Buitengewoon nuttig daarentegen blijkt het tweede moment. In de cryptoanalyse wordt het tweede moment ook wel de *Index of Coincidence* (IC) genoemd, een benaming afkomstig van William Friedman. Het tweede moment is gerelateerd aan de variatie in de letterfrequenties. Noem p_A de frequentie van letter A, p_B die van de B, enz. Deze letterfrequenties kunnen worden gezien als de *kans* om de letter aan te treffen. Het tweede moment van de letterfrequenties is de som van de kwadraten van die 26 kansen.

	-A	-B	-C	-D	-E	-F	-G	-H	-I	-J	-K	-L	-M	-N	-O	-P	-Q	-R	-S	-T	-U	-V	-W	-X	-Y	-Z
A-	4	20	28	52	2	11	28	4	32	4	6	62	23	167	2	14		83	76	127	7	25	8	1	9	1
B-	13				55				8	2		22			11			15	4	2	13				15	
C-	32		7	1	69			33	17		10	9	1		50	3		10		28	11				3	
D-	40	16	9	5	65	18	3	9	56		1	4	15	6	16	4		21	18	53	19	5	15		3	
E-	84	20	55	125	51	40	19	16	50	1	4	55	54	146	35	37	6	191	149	65	9	26	31	12	5	
F-	19	3	5	1	19	21	1	3	30	2		11	1		51			26	8	47	6	3	3		2	
G-	20	4	3	2	35	1	3	15	18			5	1	4	21	1	1	20	9	21	9		5		1	
H-	101	1	3		270	5	1	6	57				3	2	44	1		3	10	18	6		5		3	
I-	40	7	51	23	25	9	11	3			2	38	25	202	56	12	1	46	79	117	1	22		4		3
J-	3				5				1						4						3					
K-	1				11				13					2	2				6	2	1		2		1	
L-	44	2	5	12	62	7	5	2	42	1	1	53	2	2	25	1	1	2	16	23	9		1		33	
M-	52	14	1		64			3	37				7	1	17	18	1	2	12	3	8		1		2	
N-	42	10	47	122	63	19	106	12	30	1	6	6	9	7	54	7	1	7	44	124	6	1	15		12	
O-	7	12	14	17	5	95	3	5	14		19	41		134	13	23		91	23	42	55	16	28		4	1
P-	19	1			37			4	8			15	1		27	9		33	14	7	6					
Q-																					17					
R-	83	8	16	23	169	4	8	8	77	1	10	5	26	16	60	4		24	37	55	6	11	4		28	
S-	65	9	17	9	73	13	1	47	75	3		7	11	12	56	17	6	9	48	116	35	1	28		4	
T-	57	22	7	1	76	5	2	330	126	1		14	10	6	79	7		49	50	56	21	2	27		24	
U-	11	5	9	6	9	1	6		9		1	19	5	31	1	15		47	39	31		3				
V-	7				72				28						5										3	
W-	36	1	1		38			33	36			4	1	8	15				4	2			1			
X-	1		2			1			3						1	5				3			1			
Y-	14	5	4	2	7	12	2	6	10			3	7	5	17	3		4	16	30			5			
Z-	1				4																					
	-A	-B	-C	-D	-E	-F	-G	-H	-I	-J	-K	-L	-M	-N	-O	-P	-Q	-R	-S	-T	-U	-V	-W	-X	-Y	-Z

Tabel 6.1 Digramtelling Engels

$$S_2 = \sum_{i=A}^Z p_i^2 = p_A^2 + p_B^2 + \dots + p_Z^2 \quad (6.1)$$

Meteen kan worden opgemerkt dat het tweede moment onveranderlijk moet zijn onder verwisseling van de letters, zoals in een monoalfabetische substitutie. Immers, een permutatie van het alfabet verwisselt slechts de aan de kansen p opgehangen etiketten (de identiteit van de letters), maar niet hun waarde.

Substitueer bijvoorbeeld een X voor de E en omgekeerd. Dan krijgt p_X de frequentie van E en p_E die van X. De waarde van de som zal door deze verwisseling niet veranderen. Een monoalfabetische substitutie kan zodoende het tweede moment van de frequentiedistributie niet maskeren en daarom laat een monoalfabetische substitutie zich herkennen aan de waarde van het tweede moment.

In willekeurige (random) tekst komen alle letters gemiddeld even vaak voor en heeft elke letter een kans op voorkomen van $1/26$. De waarde van het tweede moment voor een 26-letter alfabet is in dit geval de volgende.

$$S_2^{random} = \sum_{i=A}^Z p_i^2 = \sum_{i=A}^Z \left(\frac{1}{26}\right)^2 = 26 \left(\frac{1}{26}\right)^2 = \frac{1}{26} = 0.0385 \quad (6.2)$$

Verschillende talen blijken elk een verschillend eigen karakteristiek frequentiepatroon te hebben met bijbehorende karakteristieke waarde van het tweede moment. Hoe onevenwichtiger de verdeling van de letterfrequenties, hoe hoger de waarde van het tweede moment*.

In [tabel 6.2](#) vindt men het tweede moment van enkele talen. Een paar dingen vallen op in deze tabel. De hoge waarde voor het Duits ten opzichte van het Engels wordt veroorzaakt door de dominantie van de E en de N in het Duits. Hetzelfde geldt voor het Nederlands, dat ongeveer hetzelfde tweede moment heeft. Het Frans heeft eveneens een hoge frequentie voor E, in het Spaans komt de A daar nog bij en in het Italiaans bovendien de I. De hoge waarde van het tweede moment voor de Romaji-schrijfvorm van het Japans vindt zijn oorsprong in het buitengewoon groot aantal klinkers. De lage waarde voor het Russisch wordt verklaard door het feit dat het cyrillisch schrift niet 26 maar 32 letters heeft. Als lettervoorkomens over meer letters worden uitgesmeerd leidt dit vanzelf tot een lagere waarde van het tweede moment. Ook is duidelijk dat er een flink verschil bestaat tussen de waarden voor normale tekst en die van een willekeurige verzameling van letters. Een monoalfabetische distributie laat zich daarom in het algemeen goed daarvan onderscheiden.

* In het extreme geval dat slechts één enkele letter voorkomt, zal het tweede moment de waarde 1 hebben.

Taal	S_2	Taal	S_2
Engels	0,0661	Duits	0,0762
Frans	0,0778	Italiaans	0,0738
Spaans	0,0775	Portugees	0,0791
Japans-Romaji	0,0819	Russisch	0,0529
random	0,0385		

Tabel 6.2 Tweede moment S_2 voor diverse talen

Het gebruik van het tweede moment als instrument voor de cryptoanalyse mag worden toegeschreven aan William Friedman en komt het eerst voor in een van zijn befaamde Riverbank publicaties^[2].

Men hoeft zich bij het tweede moment niet te beperken tot de enkele letters. Ook voor digrammen, trigrammen en hogere combinaties kan het tweede moment worden uitgerekend en gebruikt. In het Engels is het tweede moment 0.0069 voor digrammen, het is 0.0015 voor willekeurige digrammen. Een relatief groter verschil dan voor de losse letters. Wel dient men te bedenken dat door het grote aantal mogelijke digrammen en trigrammen veel meer cijfertext nodig zal zijn voor het trekken van betrouwbare conclusies dan meestal beschikbaar is. Zodoende is het tweede moment van de letterfrequenties toch weer het meest nuttig.

6.3.1 Benadering tweede moment

Voor de berekening van het tweede moment van een letterdistributie zijn de waarden van de kansen p_A enz. voor alle letters A tot en met Z nodig. Om die te bepalen wordt het aantal voorkomens f_A t/m f_Z van alle letters geteld en gedeeld door de som van het totaal aantal letters. Hoe meer letters in de totaaltelling, hoe nauwkeuriger het resultaat.

$$p_i = \lim_{N \rightarrow \infty} \frac{f_i}{N} \quad \forall i = A, \dots, Z \quad \text{met} \quad N = \sum_{i=A}^Z f_i \quad (6.3)$$

In de tellingen van **figuur 6.1** is genoeg tekst geteld om tot een min of meer redelijke benadering van de werkelijke waarden van die kansen te komen. De standaardverdeling van de letters gaat soms niet op, met name niet in de getrukte teksten die voorkomen in bepaalde puzzlecryptogrammen*. Hetzelfde geldt voor berichten in een specifiek jargon. Een ander extreem geval vormen teksten waarin de schrijver een bepaalde letter systematisch heeft vermeden, zoals o.a. in de roman Gadsby (zonder e) of Die Zwillinge (zonder r).

Bedenk dat er niet één, op alle gevallen toepasbare frequentiedistributie bestaat. Altijd zal men te maken hebben met kleinere of grotere afwijkingen in de waarden van de letterfrequenties. Al naar gelang aard en taalveld van de teksten die worden onderzocht en ook nog eens beïnvloed door de lengte van de tekst. In de praktijk van de cryptoanalyse moet men het doorgaans met veel minder materiaal stellen dan waarmee de standaardfrequentietabellen zijn opgesteld. Vaak heeft men slechts een of enkele cryptogrammen van korte lengte en dus onvoldoende statistische basis voor een goede benadering van de kansen.

Het is dus van belang om uit het beschikbare schaarse materiaal toch een zo betrouwbaar mogelijke schatting van de werkelijke waarde van het tweede moment te halen. Hier komen statistiek en kansrekening de cryptoanalist te hulp met het begrip *zuivere schatter*. Uitgaande van de veronderstelling dat de gevonden waarden deel uitmaken van een kansverdeling met de werkelijke waarde als *verwachtingswaarde*, is een zuivere schatter een functie die deze waarde zo goed mogelijk benadert. Zo goed mogelijk wil zeggen *zonder systematische afwijking*.

Het hier volgende rekenvoorschrift wordt geponeerd als een zuivere schatter voor de gezochte waarde van het tweede moment. Het is gebruikelijk om deze schatter aan te duiden met de Griekse letter Φ (phi). Hieraan ontleent de op het tweede moment gebaseerde test op monoalfabetischeit de naam *phi-test*. De

* Een tekst als "yucky yack drank yoghurt, young yogi yawned" geeft een veel te hoge frequentie voor de y.

wijze waarop het tweede moment het best kan worden berekend uit de waargenomen aantallen letters f_A t/m f_Z is de volgende

$$\Phi = \frac{\sum_{i=A}^Z f_i (f_i - 1)}{N(N - 1)} \quad \text{met } N = \sum_{i=A}^Z f_i \quad (6.4)$$

Men neemt dus niet de som van de kwadraten van de getelde aantallen letters maar net iets minder. Aangevoerd kan worden dat de statistische verwachtingswaarde van deze uitdrukking gelijk is aan de werkelijke waarde van het tweede moment, zijnde de som van de kwadraten van de werkelijke letterkansen. Hetgeen wil zeggen dat deze Φ inderdaad een zuivere schatter – de best mogelijke benadering – voor het tweede moment is. Dat dit klopt, kan worden aangetoond door de verwachtingswaarde van het gegeven voorschrift uit te rekenen en te constateren dat de uitkomst inderdaad de som van de kwadraten van de kansen is.

Het is eenvoudiger om eerst het voorschrift te geven en vervolgens aan te tonen dat dit werkelijk een zuivere schatter is, dan andersom. De verificatie van die bewering loopt als volgt. Begin met de voor een verzameling letters uit een 26-letter alfabet toepasselijke kansverdeling, de *multinomiale verdeling*

$$p(f_A, f_B, \dots, f_Z) = \frac{N! p_A^{f_A} p_B^{f_B} \dots p_Z^{f_Z}}{f_A! f_B! \dots f_Z!} \quad \text{met } N = \sum_{i=A}^Z f_i \quad (6.5)$$

Kans $p(f_A, f_B, \dots, f_Z)$ is de kans dat bij willekeurige trekking van N letters precies f_A letters A, f_B letters B, etc. zullen worden gevonden. Als N groot genoeg is mag worden aangenomen dat de verwachtingswaarde $E(f_i)$ de waarde np_i zal hebben en dat de variantie van f_i gegeven wordt door de uitdrukking $\text{var}(f_i) = np_i(1-p_i)$. Dit wordt zo dadelijk gebruikt.

De verwachtingswaarde $E(\Phi)$ van functie Φ wordt nu uitgewerkt. Voor het gemak blijft $N(N - 1)$, de noemer uit [vergelijking \(6.4\)](#) achterwege, zoals aan het eind van de afleiding zal blijken.

$$E(\Phi) = E\left(\sum_{i=A}^Z f_i(f_i - 1)\right) = \sum_{i=A}^Z E(f_i(f_i - 1)) = \sum_{i=A}^Z (E(f_i^2) - E(f_i)) \quad (6.6)$$

De lastige term $E(f_i^2)$ kan worden geloosd door gebruik te maken van diens relatie met de variantie

$$\text{var}(f_i) = E(f_i^2) - E(f_i)^2 \quad (6.7)$$

Substitueer dit vervolgens

$$E(\Phi) = \sum_{i=A}^Z (\text{var}(f_i) + E(f_i)^2 - E(f_i)) \quad (6.8)$$

Nu worden de bovengenoemde relaties $E(f_i) = np_i$ en $\text{var}(f_i) = Np_i(1-p_i)$ benut

$$E(\Phi) = \sum_{i=A}^Z (Np_i(1-p_i) + (Np_i)^2 - Np_i) \quad (6.9)$$

$$E(\Phi) = N(N-1) \sum_{i=A}^Z p_i^2 = N(N-1)S_2 \quad (6.10)$$

Hieruit volgt direct de juistheid van **vergelijking (6.4)**.

6.3.2 Bepaling monoalfabeticiteit

Een paar voorbeelden laten het gebruik van het tweede moment voor de bepaling van de monoalfabeticiteit van een cryptogram zien. Voor het gemak is in de berekening de deling door $N(N-1)$ achterwege gelaten, waardoor de verwachtingswaarde $\Phi = N(N-1)S_2$ wordt. De volgende monoalfabetische vercijfering telt 23 letters en heeft de bijgaande verdeling over het alfabet.

WHZM VMQZU UWU YVVM ZMLXZMG																									
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
						1	1				1	5				1				3	3	2	1	1	4

Uit deze telling wordt $\Phi = 46$ berekend. Voor het Nederlands is het tweede moment bij benadering 0,076. De verwachtingswaarde van het tweede moment voor een tekst van 23 letters is voor Nederlandse en willekeurige tekst respectievelijk

$$\text{Nederlands: } \Phi_m = 23 \times 22 \times 0,076 = 38,4 \quad \text{Random: } \Phi_r = 23 \times 22 \times 0,038 = 19,2$$

De gevonden waarde 46 duidt dus eerder op een monoalfabetische tekst dan op een willekeurige verzameling letters. Een meer kwantitatief gefundeerde conclusie kan worden getrokken door de standaardafwijking van Φ erbij te betrekken. Deze kan uitgerekend worden met behulp van S_2 en S_3 , respectievelijk het tweede en derde moment van de frequentiedistributie. Voor een tekst van N letters volgt de variantie – het kwadraat van de standaardafwijking – uit de volgende formule^[3]

$$\text{var}(\phi) = 4N^3(S_3 - S_2^2) + 2N^2(5S_2^2 + S_2 - 6S_3) + 2N(4S_3 - S_2 - 3S_2^2) \quad (6.11)$$

Voor een uniforme verdeling van de letters geldt in het bijzonder $S_2^2 = S_3$, zodat in dat geval de variantie kan worden vereenvoudigd tot

$$\text{var}(\phi) = 2N(N - 1)(S_2 - S_3) \quad (6.12)$$

Met voor Nederlandse tekst $S_2 = 0,076$ en $S_3 = 0,00869$, geven respectievelijk de monoalfabetische en de random hypothese de volgende uitkomsten

$$\text{Nederlands: } \Phi_m = 38,4 \pm 14,0 \quad \text{Random: } \Phi_r = 19,2 \pm 6,1$$

De gevonden waarde $\Phi = 46$ verschilt meer dan vier standaardafwijkingen van Φ_r . De hypothese dat hier van een willekeurige verzameling letters sprake is, kan dus gevoeglijk worden verworpen. Daarentegen is het verschil met de monoalfabetische verwachtingswaarde Φ_m niet meer dan een halve standaardafwijking. De hypothese dat het hier een monoalfabetische distributie betreft kan beslist niet worden verworpen.

Een voorbeeld van een niet-monoalfabetische tekst is de navolgende klaartekstautoclaaf van 42 letters.

KCNGD TOQZP TJDGE UPZGE ELXFA NBZXS LDTWS EIAJP EK																									
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
2	1	1	3	5	1	3		1	2	2	2		2	1	3	1		2	3	1		1	2		3

Hieruit wordt $\Phi = 64$ gevonden, te vergelijken met de volgende verwachtingswaarden voor monoalfabetische en random tekst. De hypothese dat dit een randomverdeling is kan zeker niet worden verworpen, terwijl de monoalfabetische waarde twee standaardafwijkingen afwijkt.

$$\text{Nederlands: } \Phi_m = 42 \times 41 \times 0,076 = 131 \pm 32 \quad \text{Random: } \Phi_r = 42 \times 41 \times 0,038 = 65 \pm 11$$

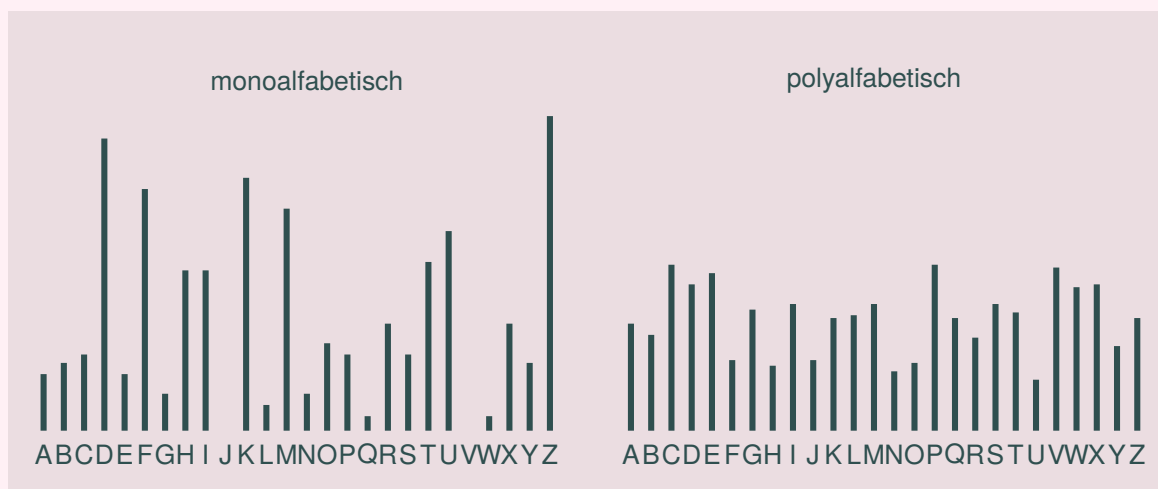
6.3.3 Bepaling periode

Nu zal worden behandeld hoe de zojuist geïntroduceerde phi-test kan worden benut om de periode van een periodieke polyalfabetische substitutie te bepalen. [Figuur 6.3](#) laat het effect van polyalfabetische substitutie op de letterfrequenties zien. Het histogram links is afkomstig van een monoalfabetische vercijfering, het histogram rechts van een Vigenère met periode 9. Waar het patroon van letterfrequenties links duidelijke pieken en dalen te zien geeft, is dat van de polyalfabeet opvallend veel vlakker.

Veronderstel de gezochte periode heeft de waarde p . Verdeel de cryptogramtekst in p groepen met in elke groep de letters die met eenzelfde sleutelletter zijn vercijferd. Voor de positie i van de cryptogramletters in groep k geldt

$$c_i \in k \text{ iff } i \bmod p \equiv k \quad (6.13)$$

Zonneklaar is dat alle p groepen een monoalfabetische distributie zullen hebben als p de juiste periode is. Want in dat geval heeft de opdeling precies de goede letters bijeengebracht. Voor elke groep valt dan te verwachten dat de phi-test een goede schatting van het tweede moment voor de betreffende taal zal geven. In elke groep zal de frequentiedistributie lijken op die aan de linkerkant van [figuur 6.3](#).

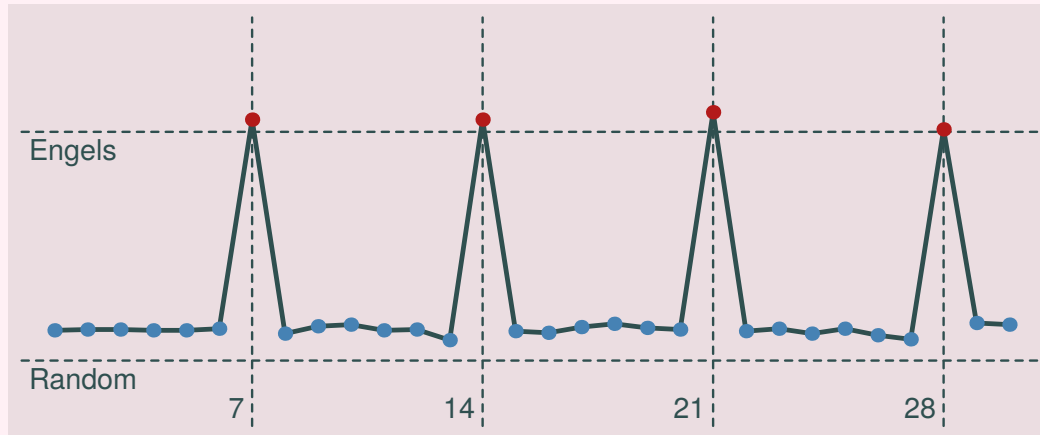


Figuur 6.3 Effect polyalfabetische substitutie op letterdistributie

Nu wordt het cryptogram opnieuw opgedeeld met een andere waarde voor p dan de juiste periode. In elke groep zullen in dat geval cryptogramletters afkomstig van vercijfering met verschillende sleutelletters bijeenkomen. Elke groep heeft nu een willekeurige samenstelling en naar verwachting zal de phi-test de waarde 0,0385 voor een willekeurige verdeling te zien geven. In elke groep zal de frequentiedistributie lijken op die aan de rechterkant van [figuur 6.3](#).

De periode van een periodieke polyalfabetische substitutie kan worden gevonden door alle mogelijke waarden voor de periode te proberen en te kijken welke goede waarden voor de phi-test opleveren. Wanneer een bepaalde waarde p voldoet, zullen natuurlijk ook de veelvouden $2p$, $3p$, ... goede waarden geven. Als voorbeeld is een polyalfabetisch vercijferde tekst (Vigenère, sleutel PICKLES, Engels) van 1673 letters genomen ([figuur 6.4](#)). Dit cryptogram heeft zonder twijfel periode 7.

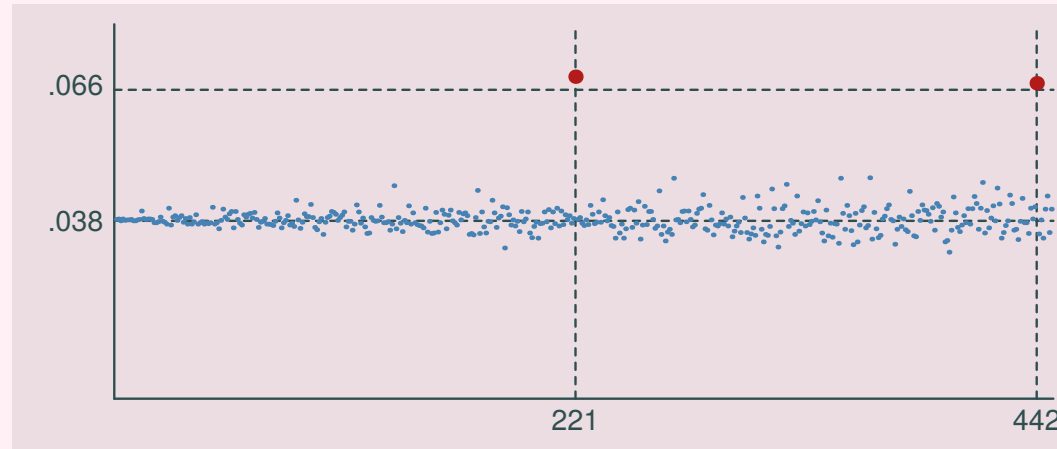
De statistiek komt nog verder te hulp. Bij elke veronderstelde periode worden evenzoveel waarden van de phi-test berekend. Eén voor ieder van de p opdelingsgroepen. Aangezien deze alle zullen tenderen naar 0 of



Figuur 6.4 Phi-test op Vigenère met sleutel PICKLES, 1673 letters, Engels

de monoalfabetische of de randomwaarde, mogen ze worden gemiddeld. Net als hierboven kan worden berekend welke standaardafwijkingen te verwachten zijn. De statistiek leert dat voor het gemiddelde de standaardafwijking moet worden gedeeld door de vierkantswortel van het aantal waarnemingen.

Voor een kort cryptogram is het resultaat niet altijd zo duidelijk als in [figuur 6.4](#) en zal wellicht meer dan één mogelijke waarde voor de periode in aanmerking komen. Anderzijds illustreert [figuur 6.5](#) hoe gevoelig deze test ook kan zijn. Het betreft een Engelse tekst van 2272 letters die polyalfabetisch is gecijferd met periode 221^[4]. De grafiek laat zelfs bij de dubbele periode 442 nog een duidelijk monoalfabetisch resultaat zien. Bedenk dat met deze periode het aantal letters waarover de frequentiedistributie geteld wordt niet meer dan 5 à 6 is. Het grote aantal van 442 benaderingen van het tweede moment dat hier wordt gemiddeld, zorgt ervoor dat desondanks een redelijk nauwkeurig gemiddelde wordt gevonden.



Figuur 6.5 Phi-test op 2272 letters met periode 221

6.3.4 Bepaling aantal letters

Een willekeurige verdeling is op te vatten als het gemiddelde van een groot aantal onderling verschillende monoalfabetische substituties. Zo'n situatie wordt in geval van een polyalfabetische substitutie benaderd met veel verschillende letters in de sleutel*. De verdeling gaat dan lijken op een randomverdeling. Het andere uiterste is een sleutel met slechts één letter, waarbij de polyalfabeet degenereert tot een monoalfabeet.

Men kan zich nu afvragen welke verwachtingswaarde voor Φ behoort bij 1, 2 ... 26 verschillende letters in de sleutel. Let op, dat is niet hetzelfde als de lengte van de sleutel, want letters kunnen meerdere keren voorkomen in een sleutel. Door dit uit te rekenen kan een – voorzichtige – schatting gemaakt worden van het aantal verschillende letters in de sleutel. Stel het cryptogram telt in totaal N letters en er zitten k verschillende letters in de sleutel. Het aantal letters in het alfabet wordt gesteld op n en veelal zal dus

* In het geval van een 26-letteralfabet kunnen dat er vanzelfsprekend niet meer dan 26 zijn, hoe lang de sleutel ook is.

$n = 26$. N_j is het aantal voorkomens van de letter met index j , de sommatie loopt over alle k letters. De verwachtingswaarde van Φ volgt uit de formule

$$E(\Phi) = \frac{(N-1)}{N} \left(S_2 \sum_{j=1}^k N_j^2 + \frac{2}{n} \sum_{j=1}^k \sum_{j'=j+1}^k N_j N_{j'} \right) \quad (6.14)$$

Als iedere letter even vaak in de sleutel voorkomt, kan deze uitdrukking worden vereenvoudigd tot

$$\forall j: N_j = \frac{N}{k} \rightarrow E(\Phi) = \frac{N(N-1)}{k} \left(S_2 + \frac{k-1}{n} \right) \quad (6.15)$$

Engels				Nederlands			
k	Φ	k	Φ	k	Φ	k	Φ
1	0,0661	14	0,0404	1	0,0760	14	0,0411
2	0,0523	15	0,0403	2	0,0572	15	0,0410
3	0,0477	16	0,0402	3	0,0510	16	0,0408
4	0,0454	17	0,0401	4	0,0479	17	0,0407
5	0,0440	18	0,0400	5	0,0460	18	0,0405
6	0,0431	19	0,0399	6	0,0447	19	0,0404
7	0,0424	20	0,0398	7	0,0438	20	0,0403
8	0,0419	21	0,0398	8	0,0432	21	0,0403
9	0,0415	22	0,0397	9	0,0426	22	0,0402
10	0,0412	23	0,0397	10	0,0422	23	0,0401
11	0,0410	24	0,0396	11	0,0419	24	0,0400
12	0,0408	25	0,0396	12	0,0416	25	0,0400
13	0,0406	26	0,0395	13	0,0414	26	0,0399

Tabel 6.3 Tweede moment en aantal sleutelletters

Tabel 6.3 geeft de waarde van bovenstaande formule voor het Engels en voor het Nederlands. De in het cryptogram gevonden Φ kan worden gebruikt om een indicatie te verkrijgen voor het aantal verschillende

letters in de sleutel. Herhaalde letters in de sleutel zullen in deze verwachtingswaarde het zwaarst meetellen. De gevonden waarde zal daardoor hoger uitvallen en het aantal verschillende sleutelletters lijkt kleiner dan het in werkelijkheid is.

Het cryptogram uit [figuur 6.4](#), waarvan is gegeven dat het de sleutel PICKLES heeft, geeft $\phi = 0,0417$. In de linkerkolom van de tabel (het cryptogram is in het Engels) is te zien dat deze waarde behoort bij 8 à 9 sleutelletters. Dat is duidelijk slechts een benadering, want de sleutellengte is maar 7. Toch mag nu met redelijke zekerheid worden aangenomen dat die 7 sleutelletters alle verschillend zullen zijn.

6.4 Coïncidenties

```
tekst1 : FLQHCTPLREFOUPMGJJJBVQKYKOVGEYSMRPGDZBWQOVQGEXWVVQB
tekst2 : LVUYTQIDONGDDEGSXTVJEETQTKSHOGZYFSIIXRRIJSRBLBTNTI
-----
tekst1 : RTDDGZMBDEQPFMPMTUZQFVATFCKYZPJBMLFEPXZDODIUBILOAM
tekst2 : FYOIVTOZWHEBTMTDTINVOVSANHWLQQDUFHXMIOUFFGMGXTQGON
-----+--+--+-----
```

Hierboven staan twee geheel willekeurige reeksen van 100 letters die met elkaar zijn vergeleken (voor de duidelijkheid opgesplitst in twee groepen van 50). De plustekens laten zien waar de twee teksten een overeenkomstige letter hebben; dat zijn er drie.

Vervolgens is hetzelfde gedaan met twee willekeurig gekozen teksten uit Nederlandse boeken^{*}; nu komen de teksten op acht posities overeen. De voor de hand liggende vraag is natuurlijk of dit verschil iets zegt of binnen de marge van het toeval blijft.

^{*} Jules Verne *Het geheimzinnige eiland* en W.F. Hermans *Nooit meer slapen*.

```

tekst1 : AANDEPLANNENVANDEZEEROVERSVELDUSNIETMEERTETWYFELE
tekst2 : IKGAZITTENVOUWDEKAARTOPENENLEGHEMMETDEBOVENKANTIND
-----+-----+-----+-----+-----+
tekst1 : NZYHADDENHETANKEROPKORTEAFSTANDVANHETEILANDGEWORPE
tekst2 : ERICHTINGWAARDEZONSTAATHETNOORDENMISSCHIENRONDKYKE
-----+-----+-----+-----+

```

De overeenkomende letters uit de beide voorbeelden noemt men *coïncidenties*. De juxtapositie van de twee randomteksten telt 3 coïncidenties, die van de Nederlandse teksten 8. Het is niet moeilijk om te bedenken waarom juist dit verschil.

Neem eerst de randomtekst. Hierin hebben alle letters precies evenveel kans op voorkomen, namelijk 1 op 26. De kans op een bepaalde letter op een bepaalde positie is dus $1/26$ en de kans dat diezelfde letter er ook tegenover staat is $1/26 \times 1/26$. Op elke positie mogen alle letters staan, zodat de totale kans op een coïncidentie in een willekeurige combinatie van letters uitkomt op de volgende waarde

$$\sum_{A \dots Z} \left(\frac{1}{26} \right)^2 = \frac{1}{26} = 0.0384 \quad (6.16)$$

Dat is weer precies de waarde van het tweede moment voor een randomverdeling. Voor de 100 letters uit het eerste voorbeeld mag men daarom gemiddeld 4 coïncidenties verwachten.

Analoog wordt geredeneerd voor de Nederlandse teksten. Elke letter heeft z'n eigen individuele kans op voorkomen. De kans op een EE-coïncidentie is daarom het kwadraat van de kans op een enkele E. Voor de kans op een willekeurige coïncidentie moeten de kwadraten gesommeerd worden

$$\sum_{A \dots Z} p_i^2 \quad (6.17)$$

De totale kans op een coïncidentie is ook weer gelijk aan het tweede moment, maar nu voor het Nederlands zijnde 0,076. De verwachting is dat in de 100 letters van het tweede voorbeeld gemiddeld 7 à 8

coïncidenties zullen worden aangetroffen.

In principe kan men met coïncidentietellingen in bijvoorbeeld een Vigenère uitzoeken wat de Caesar-verschuivingen tussen de letters van de sleutel onderling zijn. Dat gaat op de volgende manier.

Verdeel het cryptogram in groepen waarin de letters alle met dezelfde letter van de sleutel zijn vercijferd. Onderzoek alle combinaties van twee van deze groepen door tussen de eerste 100 letters van elk een coïncidentietelling te maken. Herhaal dit 25 keer met alle letters uit de tweede groep onderworpen aan Caesar-verschuiving van respectievelijk B, C,

Zo krijgt men 26 tellingen waarvan er 25 niet zullen matchen met de relatie tussen de twee betrokken sleutelletters en 1 wel. De verwachting is dat de 25 tellingen die niet matchen gemiddeld 4 coïncidenties zullen bevatten en degene die wel matcht 7.

Helaas werkt deze methode in de praktijk niet goed, of misschien moet zelfs gezegd worden, buitengewoon slecht. De oorzaak is gelegen in de statistische fluctuaties die in de coïncidentietellingen zullen voorkomen. Uitschieters naar boven en naar beneden zullen geen uitzondering zijn en daar hebben vooral de niet-matchende tellingen last van: het zijn er immers met 525 veel meer dan de 21 die wel matchen. Uitschieters naar boven vallen ruimschoots te verwachten en zullen zich voordoen als een match. Omgekeerd zullen matchende tellingen die naar beneden afwijken niet als zodanig herkend worden. Gelukkig bestaat er een methode, de chi-toets, die wel goed onderscheid kan maken tussen matchende en niet-matchende distributies.

6.4.1 Poisson verdeling

Hoe betrouwbaar is het verwachte aantal coïncidenties? Een waarschuwing is op z'n plaats, omdat grote afwijkingen niet zeldzaam zijn. Met twee cryptogrammen van 100 letters in eenzelfde monoalfabetische substitutie hoeft men niet vreemd op te kijken als het aantal coïncidenties eens een keer 3 is. Terwijl men

er tussen de 6 en 8 zou verwachten. Het is daarom gevaarlijk om op tellingen van zo geringe omvang af te gaan. Conclusies worden pas redelijk betrouwbaar als verscheidene waarnemingen kunnen worden gemiddeld of als de steekproef veel groter is.

Om wat preciezer te kunnen zijn over de betrouwbaarheid van deze verwachtingen moet worden gekeken naar de kansverdeling die hier van toepassing is. Het gaat om gebeurtenissen met een betrekkelijk kleine kans op voorkomen (in de orde van 0,04 tot 0,07) en het gaat om uitkomsten die geheeltallig zijn (het aantal coïncidenties is een geheel getal). Daarmee is de *Poisson-verdeling* de aangewezen kansverdeling. De Poisson-verdeling geeft de kans op 0, 1, 2, ... gebeurtenissen voor een verdeling met verwachtingswaarde m

$$P(0) = e^{-m}, P(1) = me^{-m}, P(2) = \frac{m^2 e^{-m}}{2!} \dots \quad (6.18)$$

De bijbehorende standaardafwijking heeft de waarde $\sigma = \sqrt{m}$. Voor 100 letters in Engelse monoalfabetische distributie met $m = 6,6$ is de waarde van $\sigma = 2,6$. Een voorkomen van 4 coïncidenties is daarom maar 1 σ verwijderd van de verwachtingswaarde en valt niet meer te onderscheiden van de verwachtingswaarde voor een randomdistributie.

Het cryptogram waarvoor de in het voorgaande beschreven coïncidentietellingen zijn uitgevoerd is een Vigenère met periode 7. In de rechterkolommen van [tabel 6.4](#) zijn de resultaten samengevat. Van de 525 niet-matchende combinaties zijn er 43 die een aantal van 7 of meer coïncidenties vertonen, waar er gemiddeld 47 worden verwacht. De 9 gevonden combinaties van 7 of meer coïncidenties die wel matchen verdrinken als het ware in een zee van niet-matchenden.

De gecumuleerde kansen op n of meer coïncidenties in [tabel 6.4](#) laten dit verschijnsel goed zien. De kolommen onder N geven het verwachte aantal tellingen met n of meer coïncidenties. Deze kolommen corresponderen met de kansen $p_{\geq n}$ zijnde de kans op n of meer gebeurtenissen. Naarmate de lat voor het aantal coïncidenties hoger gelegd wordt zullen het er steeds minder zijn. In de kolommen onder *fout* staan de aantallen voor de niet-matchende tellingen, in de kolommen onder *goed* die voor de matchende

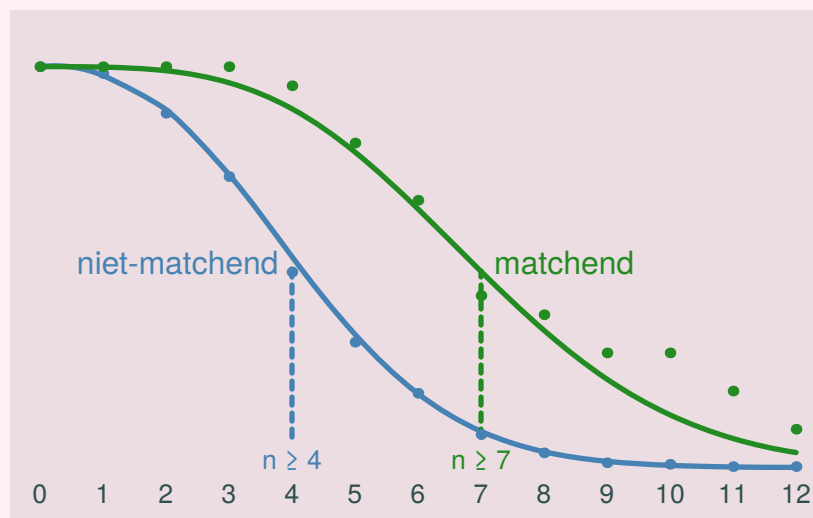
random $m = 0,038$			Engels $m = 0,066$		alfabetmatch			
n	p_n	$p_{\geq n}$	p_n	$p_{\geq n}$	fout		goed	
					N	exp	N	exp
0	0,022	1,000	0,001	1,000	525	525	21	21
1	0,085	0,978	0,009	0,999	513	516	20	21
2	0,162	0,893	0,030	0,990	468	464	20	21
3	0,205	0,731	0,065	0,960	383	381	20	21
4	0,194	0,526	0,108	0,895	276	256	18	20
5	0,148	0,332	0,142	0,787	174	164	16	17
6	0,094	0,184	0,156	0,645	96	97	13	14
7	0,051	0,090	0,147	0,489	47	43	10	9
8	0,024	0,039	0,122	0,342	21	19	7	8
9	0,010	0,015	0,089	0,220	8	6	4	6
10	0,004	0,005	0,059	0,131	3	4	2	6
11	0,001	0,001	0,035	0,072	0	1	1	4
12	0,000	0,000	0,019	0,037	0	1	0	2

Tabel 6.4 Kans op coïncidenties met Poisson verdeling

tellingen. De kolommen onder exp geven de resultaten zoals gevonden in het onderzochte cryptogram. **Figuur 6.6** illustreert de toevallige hoge aantallen coïncidenties bij de niet-matchende combinaties nog eens in een grafiek.

6.4.2 Chi-test

In het voorafgaande is aangeduid hoe de relatie tussen de sleutelletters van een Vigenère valt te bepalen door matching via coïncidentietellingen, maar is tegelijk aangegeven dat deze hiervoor geen bijzonder bruikbaar criterium opleveren. Een doelmatiger instrument is om de letterdistributies van de sleutelletters te vergelijken met de zogenaamde chi-test^[5]. Met deze chi-test wordt de overeenkomst tussen twee letter-



Figuur 6.6 Distributie van aantallen coïncidenties

tellingen gemeten als de som van de kruisproducten tussen de aantallen overeenkomstige letters. Stel ter beschikking staan twee series tellingen f en f' van respectievelijk N en N' letters. De verwachtingswaarde van de chi-test als som van de kruisproducten is

$$\chi = \frac{\sum_{i=A}^Z f_i f'_i}{NN'} \text{ met } N = \sum_{i=A}^Z f_i \text{ en } N' = \sum_{i=A}^Z f'_i \quad (6.19)$$

Twee gevallen kunnen worden onderscheiden, namelijk vergelijken van overeenkomstige en vergelijken van verschillende distributies.

1. De distributies komen overeen

De tellingen komen dan allebei uit dezelfde verdeling zodat de verwachtingswaarden voor de frequentietellingen aan elkaar gelijk zijn. Uitwerking van de uitdrukking voor de chi-test geeft als resultaat

$$E(\chi) = NN' \sum_{i=A}^Z \left(\frac{f_i}{N} \right) \left(\frac{f'_i}{N'} \right) = NN' \sum_{i=A}^Z p_i^2 = NN' S_2 \quad (6.20)$$

De variantie kan worden geschat met de volgende formule

$$\sigma^2 = NN' \{ (N + N') (s_3 - s_2^2) + s_2^2 + s_2 - 2s_3 \}$$

2. De distributies zijn verschillend

Bijvoorbeeld omdat ze weliswaar allebei monoalfabetisch zijn maar onderling verschoven. Nu geldt voor de verwachtingswaarde van de kruisproductensom

$$E(\chi) = \frac{NN'}{n} \quad \text{waarin } n \text{ het aantal symbolen} \quad (6.21)$$

En als variantie^[6]

$$\sigma^2 = NN' \frac{\{N(S_2 - \frac{1}{n}) + (1 - S_2)\} \{N'(S_2 - \frac{1}{n}) + (1 - S_2)\}}{n - 1} \quad (6.22)$$

Deze test wordt toegepast op de eerste 700 letters van een Vigenère met periode 7 (Engelse tekst) onderverdeeld in groepen van elk 100 letters. Voor respectievelijk de 21 combinaties met de goede onderlinge relatie (overeenkomende monoalfabetische distributies) en de 525 combinaties die niet matchen (onderling verschoven monoalfabetische distributies) zijn de volgende verwachtingswaarden berekend

$$\chi_{\text{goed}} = 100 \times 100 \times 0.066 = 660 \pm 96 \quad \text{en} \quad \chi_{\text{fout}} = 100 \times 100 / 25 = 384 \pm 74$$

Uitvoering van de benodigde tellingen geeft de volgende gemiddelden en standaardafwijkingen in de twee groepen als resultaat. Tussen de haakjes staan de laagste en hoogste waarde in elk van de berekeningen.

$$\chi_{21} = 669 \pm 42 \text{ (594-748)} \quad \text{en} \quad \chi_{525} = 373 \pm 65 \text{ (189-574)}$$

Tabel 6.5 geeft de uitkomsten voor alle Caesar-verschuivingen van alle 21 combinaties. Het maximum van 748 in kolom 12 (de overeenkomst tussen de 1^e en de 2^e sleutelletter) signaleert een overeenstemming in

	12	13	14	15	16	17	23	24	25	26	27	34	35	36	37	45	46	47	56	57	67
A	369	434	485	574	357	260	410	375	276	439	298	307	396	374	347	443	357	306	363	372	520
B	362	355	499	312	341	373	374	302	316	315	430	370	416	296	281	304	327	286	337	332	444
C	350	508	318	322	305	415	411	318	335	348	350	381	498	380	372	298	362	365	336	335	323
D	368	331	291	441	307	443	298	348	310	388	398	500	389	374	355	438	325	479	382	395	252
E	275	314	400	731	411	453	434	372	353	646	454	392	364	396	481	393	381	348	324	503	382
F	329	236	650	418	420	401	374	342	327	370	372	366	367	387	390	311	397	301	355	365	408
G	420	344	434	381	383	369	666	348	514	391	353	353	448	259	511	342	625	438	367	269	408
H	748	387	396	211	430	304	409	492	428	300	290	479	463	295	213	251	404	408	690	440	404
I	379	415	249	356	280	445	332	360	409	348	407	401	301	351	274	302	325	315	403	360	399
J	402	398	349	359	312	317	260	423	417	354	371	302	254	488	442	413	309	313	365	348	226
K	242	274	365	338	431	377	409	422	459	373	523	255	277	426	630	446	445	409	336	302	343
L	383	360	387	323	702	462	313	441	401	383	370	304	437	474	398	392	428	417	482	377	376
M	345	426	345	336	409	405	403	385	367	293	430	401	410	352	366	394	362	378	384	418	663
N	466	723	383	471	410	312	351	428	407	383	254	431	569	411	321	455	387	377	396	342	352
O	300	439	450	409	223	219	280	355	281	343	318	528	273	412	402	450	275	426	320	368	345
P	344	330	360	412	351	364	448	281	294	452	346	305	325	355	451	413	249	273	255	449	311
Q	382	225	447	392	364	390	385	288	269	446	630	301	464	310	380	343	370	339	238	189	381
R	320	431	402	391	418	487	438	321	448	531	388	438	677	359	447	279	471	447	400	362	362
S	398	365	388	391	378	398	358	415	371	410	396	630	453	408	302	306	425	663	440	380	440
T	424	474	398	424	308	426	496	436	546	335	317	424	315	354	317	408	388	318	457	689	501
U	533	312	460	386	332	201	351	502	341	358	324	362	275	485	380	386	407	283	415	353	379
V	416	305	341	241	310	259	475	345	352	336	418	288	351	290	485	433	392	338	388	301	387
W	434	400	248	260	447	377	346	346	414	296	379	351	365	329	304	311	402	442	371	323	324
X	307	396	247	260	473	727	354	389	656	372	402	398	346	403	332	341	453	459	438	517	378
Y	413	444	311	424	500	411	295	594	371	453	362	388	326	608	391	436	354	428	433	408	345
Z	291	374	397	437	398	405	330	372	338	337	420	345	241	424	347	712	380	444	325	503	347

Tabel 6.5 Chi-test op periode 7 Vigenère

letterfrequentie tussen de groep letters vercijferd met de eerste sleutelletter en die met de tweede, als op die laatste groep letters een Caesar-verschuiving H wordt toegepast. Dit impliceert het volgende voor de relatie tussen de eerste en de tweede sleutelletter: als de eerste sleutelletter een A is, dan wordt de tweede sleutelletter gevonden door een Caesar-verschuiving H vanaf de A in de tegengestelde richting (equivalent met een standaard Caesar-verschuiving T op de A).

Eén stel relaties 12, 13, 14, 15, 16, 17 is voldoende om het sleutelwoord te bepalen. De overige kolommen kunnen dienen ter controle of zijn van nut wanneer de uitkomsten van de chi-berekening niet zo duidelijk zijn. Zie hier bijvoorbeeld dat het maximum in kolom 13 = N overeenstemt met 12 = H plus 23 = G, samen precies weer N. Kies voor de eerste sleutelletter de A, dan geeft de tabel voor de maxima in de verschuivingen de relaties AHNFELX. Zoals gezegd, moet voor de werkelijke sleutelletters een overeenkomstige Caesar-verschuiving van de letter A in tegengestelde richting worden uitgevoerd. De sleutel wordt zodoende ATNVWPD. Voor het gekozen sleutelwoord behoeft nu nog slechts de juiste Caesar-verschuiving te worden bepaald (tabel 6.6).

A T N V W P D	H A U C D W K	O H B J K D R	V O I Q R K Y
B U O W X Q E	I B V D E X L	P I C K L E S	W P J R S L Z
C V P X Y R F	J C W E F Y M	Q J D L M F T	X Q K S T M A
D W Q Y Z S G	K D X F G Z N	R K E M N G U	Y R L T U N B
E X R Z A T H	L E Y G H A O	S L F N O H V	Z S M U V O C
F Y S A B U I	M F Z H I B P	T M G O P I W	
G Z T B C V J	N G A I J C Q	U N H P Q J X	

Tabel 6.6 Periode 7 Vigenère-sleutel

6.5 Klinkers en medeklinkers

Allerlei methoden zijn bedacht om in een monoalfabetische substitutie onderscheid te kunnen maken tussen klinkers en medeklinkers. Geen van deze methoden is overigens onfeilbaar. Soms verstopt een medeklinker zich onder de klinkers of worden een of meer klinkers gemist.

De methode van de consonantlijn is een van de oudste methodieken en wordt beschreven in het boek van Helen Fouché Gaines uit 1939^[7]. Ze berust op de contacteigenschappen van de minder voorkomende medeklinkers, zoals de B, P en dergelijke. Deze medeklinkers worden gewoonlijk vrijelijk voorafgegaan en gevolgd door klinkers. Ten aanzien van medeklinkers zijn ze echter kieskeuriger. Medeklinkers komen als regel òf ervoor òf erachter, maar niet allebei tegelijk. Zie bijvoorbeeld in de digram-frequenties van [tabel 6.1](#) de frequenties van BL = 22 en LB = 2, BT = 2 en TB = 22.

Een andere methode berust op het feit dat klinkers er toe neigen elkaar te vermijden^[8]. Een statistiek van de afstanden binnen de groep van klinkers onderling zal een piek vertonen bij een afstand van slechts enkele letters en niet continue afvallen zoals bij een willekeurige verdeling. Gezocht wordt naar groepen letters waarvoor een dergelijke piek prominent optreedt. Deze methode zal niet verder worden toegelicht. De algoritme van Sukhotin baseert zich op het feit dat klinkers meer variatie in hun contacten vertonen dan medeklinkers.

6.5.1 Consonantlijn

De consonantlijn brengt de contacten van de minst voorkomende medeklinkers in kaart. Dat zijn letters als B, G, J, K. De constructie van de lijn is eenvoudig. Voor elk trigram ABC waarin cryptogramletter B een van de weinig voorkomende medeklinkers als klaartekstequivalent heeft – B zal dan in het cryptogram weinig voorkomen – wordt de A links en de C rechts van een verticale lijn genoteerd. De letters B worden terwille van het overzicht bovenaan op een horizontale lijn geplaatst. Als voorbeeld is een tekst geconstrueerd.

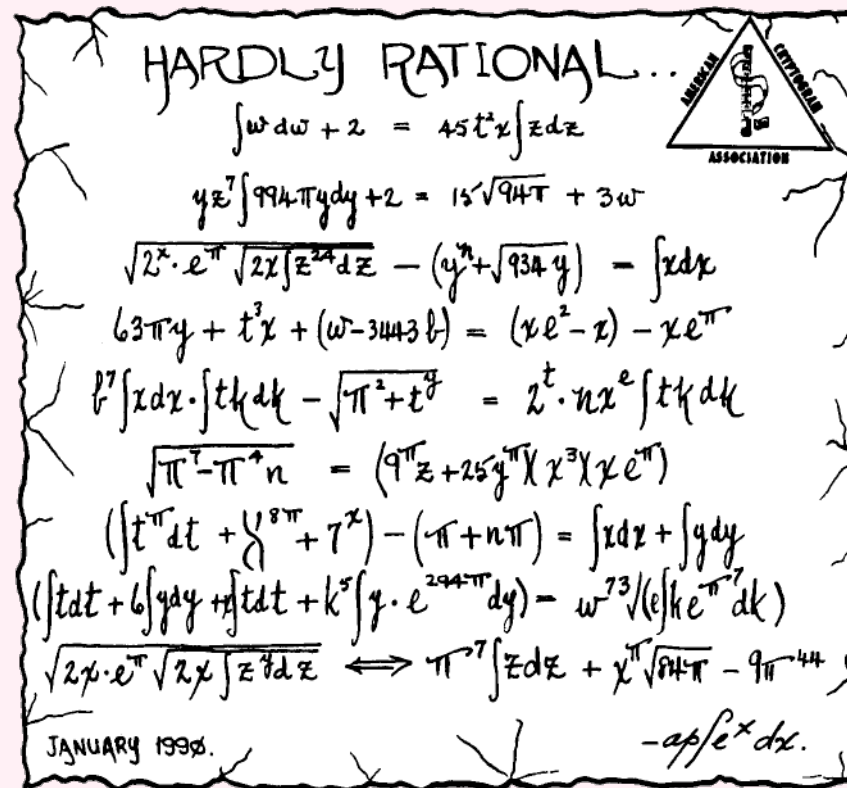
Deze is tamelijk lang om de consonantlijn een beetje duidelijk te doen uitkomen. In de praktijk zal er niet zoveel tekst voorhanden zijn en is het resultaat magerder en moeilijker te duiden. Ook is de tekst onvercijferd gelaten om de duiding van klinkers en medeklinkers te vergemakkelijken.

TWEE ZWERVENDE FLAMBOYANTE FLAMINGOS PAKTEN EEN BELGISCH PINTJE. EXQUISE ZAAK HIER, ZEI DE BLEEKSTE VAN HET PAAR. LOGISCH WANT HET TOCHT NAUWELYKS EN HET BIER IS BEST PUIK. VLEUGELLAM DOOR EEN JEUKAANVAL ALS GEVOLG VAN EEN UITBRAAKPOGING UIT EEN VOORBYKOMEND VLOOIENCIRCUS LEGDEN ZE NOG EEN BILJARTJE ALVORENS ZWIERIG HET LUCHTRUIM TE KIEZEN OP WEG NAAR VERRE GEWESTEN.

B	C	F	G	J	K	M	P	Q	V	W	X	Z
AAAAAA												
	D											
EEEEEEEEEEEEEEEE												
	HH											
	III											
	LLLL											
NNNNNNNNN												
	OOOOO											
	RRRRR											
	SSSSS											
	TTTTTT											
	UUUU											
	YY											

Spaties en leestekens zijn bij de opstelling van de consonantlijn genegeerd. De A, E, O en U zijn duidelijk klinkers met voorkomens aan beide zijden van de lijn. Medeklinkers zijn ongetwijfeld H (rechts) en N, R, S, T (links). Onduidelijker of zelfs misleidend zijn de medeklinker L en de klinker I.

Het is belangrijk de goede letters uit het cryptogram mee te nemen in de constructie van de consonantlijn. Dat is niet altijd gemakkelijk, omdat een verkeerde letter toevallig weinig in het cryptogram kan voorkomen en daardoor de indruk wekt een van de bruikbare medeklinkers te zijn. Zo'n abusievelijk meegenomen verkeerde letter kan het beeld genoeg verstoren om tot verkeerde conclusie te komen.



Figuur 6.7 Reproductie van het APEXDX cryptogram

De navolgende cryptoanalyse van het in [figuur 6.7](#) afgebeelde cryptogram laat iets zien van de praktijk van de consonantlijn. Het op te lossen probleem bevindt zich op de omslag van het Januari-Februari nummer van *The Cryptogram*, jaargang 1991. The Cryptogram is het blad van de American Cryptogram Association (ACA)*. Het is geconstrueerd door APEX DX, de nom de plume van Don Allen uit Montreal, Canada. Hij

* De ACA is een kleine vereniging van liefhebbers van de cryptografie. Ze bestaat al sedert 1932 en veel van haar leden hebben de cryptografie in de praktijk beoefend.

Het hakenpaar kan overigens niet zonder meer en in z'n geheel overboord gegooid worden als een nul. Het is best mogelijk dat de openingshaak een letter voorstelt, terwijl de sluihaak wel een echte nul is en naar believen op een geschikte plaats is ingevoegd. Omgekeerd kan natuurlijk ook de sluihaak de informatiedrager zijn en de openingshaak de nul. Wel kan voor zeker worden aangenomen dat het hakenpaar op zijn hoogst als één teken telt.

In het geval van de integralen is de situatie analoog aan die van het hakenpaar. Het is hetzij het integraalteken, hetzij de dx -groep die informatiedrager kan zijn. Aan het specifieke voorkomen van dk , dt , dx , dy , dz , behoeft niet teveel waarde gehecht worden. De keuze voor x , y , ... wordt min of meer gedicteerd door het materiaal in de integrand. Ook moet serieus rekening gehouden worden met de mogelijkheid dat het hele integraal-gedoe maar een rookgordijn is. Is de ondertekening $-ap \int e^x dx$ rechtsonder in [figuur 6.7](#) misschien een hint in die richting? Het lijkt er immers verdacht veel op dat het minteken en de integraal moeten worden weggelaten om "apexdx" te verkrijgen, de schuilnaam van de maker van deze opgave. Om hiermee rekening te kunnen houden zijn in [tabel 6.7](#) de voorkomens van dk , dt , dx , dy en dz afzonderlijk opgenomen aan de rechterkant van de eerste kolom.

De aandacht wordt vervolgens gericht + – en = de overige layoutsymbolen. Met deze tekens wordt aan de symboolreeksen het aanzien van formules en vergelijkingen gegeven. Er valt daarom veel te zeggen voor de gedachte dat dit een serie nullen is. Voor de $\sqrt{}$ kan hetzelfde gelden, maar dat is toch iets minder zeker. Een kleine moeilijkheid wordt nog geschapen door de variërende schrijfwijze voor de x . Soms is het duidelijk een x , en soms lijkt het weer meer op een χ . Omdat een continue gradering tussen de beide uitersten valt waar te nemen, is het niet erg waarschijnlijk dat het twee verschillende tekens zijn en worden ze maar allemaal als x genomen.

Het wordt tijd om het cryptogram bij de horens te vatten. De eerste indruk van de frequentietellingen in [tabel 6.7](#) is dat de π – die met 22 voorkomens het talrijkste symbool is – wel voor een cryptogramletter zal staan. Het moeten per slot van rekening 174 letters worden. Beperk de aandacht eerst eens tot de letters

(exclusief degene die tot de integraal-combinaties behoren), de cijfers, en π . Dat zijn dan tezamen al 145 tekens. Het lijkt een solide begin en omvat het leeuwendeel van het aantal te verantwoorden tekens.

De letter π bezit de hoogste frequentie, namelijk 22. Wanneer die vergeleken wordt met hetgeen voor de letter E verwacht kan worden, dan is dit precies de natuurlijke frequentie van de E in het Engels. Immers 13% van 174 is 22. Bezie nu de contacten van π . Er blijkt nogal wat variatie in de voorafgaande en opvolgende tekens te zijn, net als het de E betaamt. Dus is de voorlopige conclusie dat E in het cryptogram door π wordt voorgesteld.

In de lijst met contacten van π springt digram $e\pi$ met vijf voorkomens er nogal uit. Reden waarom de e maar eens aan een nader onderzoek wordt onderworpen. Van de negen e's worden er zes voorafgegaan door een x, terwijl daaronder het trigram $xe\pi$ wel viermaal voorkomt. Omdat x de op een na hoogste frequentie bezit, ligt het zeer voor de hand om te concluderen dat $xe\pi$ is THE. Het Engels telt ongeveer 9,7% T's, 9,7% van 174 is 17, mooi overeenkomend met de 18 getelde x'en. Voor de H is de verwachting 5,4% van 174 is 9 voorkomens, en die zijn er ook*.

Invullen van de substituties x, e, π = T, H, E in het cryptogram brengt in de vierde formuleregels in het stuk $(xe^2-x)-xe^\pi$ de veel belovende klaartekst TH-T THE aan het licht. De onmiddellijke suggestie is, dat de 2 voor de A staat. Ook hier lijkt de frequentietelling die suggestie te bevestigen: 7,8% van 174 is 13 en er zijn 13 cijfers 2. Omdat het cryptogram hier een hakenpaar vertoont, versterkt dit de idee dat de hakenparen en de min-tekens nullen zijn. En als het min-teken een nul is zal het plus-teken het ook wel zijn. Het slot van het cryptogram – met het verdubbeld symbool 4 – samen met het in de tweede formuleregels voorkomend verdubbeld symbool 9 van middelmatige frequentie, wekt de suggestie om hier te lezen

$$+x^\pi\sqrt{84\pi} - 9\pi^{44} = \text{TH USELESS}$$

Vervolgens zijn zonder resultaat allerlei substituties geprobeerd. Dat gaf weliswaar vertrouwde met

* De nauwkeurige overeenstemming tussen de standaardfrequenties voor de Engelse taal en de gevonden aantallen is bijna beangstigend. Maar misschien heeft de constructeur van het cryptogram hier bewust naar gestreefd.

het cryptogram, maar bracht de oplossing niet veel dichterbij. In Tolkien's boek *The Lord of the Rings* wordt van Barliman Butterbur, de waard van de Prancing Pony, hoewel traag van begrip gezegd "yet he can see through a brick wall in time"*. Zo voelt het hier ook. Na enige tijd krijgt de cryptoanalist steeds sterker het gevoel als het ware door het cryptogram heen te kunnen kijken. De oplossing ligt dan bij wijze van spreken voor het grijpen. Een van de dingen, die tijdens het zoekend en tastend omgaan met het cryptogram helder worden, is de structuur ervan. Want plotseling valt de incongruentie tussen de = tekens in de eerste acht formuleregels en het $\Leftrightarrow$ teken in de laatste regel op. Het geheel ziet er ineens uit als een citaat met aan het slot de naam van de geciteerde persoon.

Na deze excursie terug naar het uitgangspunt. De splitsing van de verzameling tekens in cryptogramletters en nullen. Zoals is opgemerkt zijn er met de letters, de cijfers en π in totaal 145 tekens verdeeld over 19 symbolen. Het aantal van 19 symbolen is aan de kleine kant. Bij 174 letters in het cryptogram kan men het voorkomen van 22–23 verschillende letters verwachten^[9]. Er missen waarschijnlijk nog een paar symbolen, die samen 29 letters moeten bijdragen. Kijkend naar de frequentietelling blijkt dat de integraal en de vierkantswortel daarvan met z'n tweeën 28 voor hun rekening kunnen nemen. Wordt de interpretatie van $\Leftrightarrow$ als scheidingsteken voor juist aangenomen – en dus niet meetelt als letter – dan kan met toevoeging van de $\aleph$ het totaal precies op de gezochte 29 uitkomen. Dat is een gezonde basis om $\sqrt{}$ en $\int$ serieus te nemen als substituut voor letters. En ten slotte is het bijzonder bemoedigend wanneer een berekening van het tweede moment wordt uitgevoerd. Dat blijkt de waarde 0,064 te hebben en dat is een heel mooie waarde voor een stuk Engelse tekst.

De volgende stap is de verdere identificatie van de belangrijke klinkers. Al gevonden zijn A en E, dus I en O moeten nog worden opgespoord. Tabel 6.8 laat de meest saillante delen van de hiertoe opgestelde consonantlijn zien. Conclusies die uit deze consonantlijn kunnen worden getrokken

- uitgesproken medeklinkers zijn 5, 7, $\sqrt{}$, t en 4

* J.J.R. Tolkien, *The Lord of the Rings*, Part I, The Fellowship of the Ring, Book One, 'Many meetings'.

	55
	777
33	333
ttttt	t
ssssss	ssssss
	√√√√√
44	

Tabel 6.8 Consonantlijn voor APEXDX

- de ontbrekende klinkers I en O verschuilen zich ongetwijfeld achter 3 en j
- de t afficheert zich nogal opzichtig als de letter N

De gevolgtrekkingen uit de consonantlijn blijken voldoende om het cryptogram nu compleet te breken. Aan het slot van de zevende en het begin van de achtste formuleregel wordt gevonden

$$(e \int k e^{\pi^7} dk) \sqrt{2x} \cdot e^{\pi} \sqrt{2x} \int z^y dz = \text{HI-HE--ATHE-ATI-- of HO-HE--ATHE-ATO--}$$

Zelfs zonder "Hardly Rational" als opschrift boven het cryptogram en het wiskundig imago ervan, is het niet moeilijk om hierin HIGHER MATHEMATICS te lezen. De gevonden waarden kunnen worden ingevuld en brengen vaart in de oplossing. Zo bevestigt ANYTHING in de vijfde formule de toeschrijving van N aan t, LUNATIC in de beginregel voegt de L toe, SCRIBBLES A JUMBLE OF in de tweede brengt B, J en F, enz. Tot slot de complete oplossing.

if a lunatic
scribbles a jumble of
mathematical symbols it
does not follow that the
writing means anything
merely because to the
inexpert eye it is
indistinguishable from higher
mathematics $\Leftrightarrow$ eric temple bell

6.5.2 Sukhotin

Door de Rus Sukhotin is een eenvoudig maar doeltreffend algoritme bedacht voor het opsporen van de klinkers^[10]. Niet alleen voor het Engels, maar zelfs voor andere dan de gebruikelijke westerse talen wordt Sukhotin bruikbaar geacht. Voor de talen Georgisch, Kroatisch, Gaelic, Hongaars en Hebreeuws is dit nagegaan^[11]. Hier volgt de eenvoudigste van de Sukhotin-varianten voor de klinkerbepaling.

Sukhotin algoritme voor klinkers

- Tel de digrammen in een 26x26-matrix zonder rekening te houden met woordverdeling, tel AB zowel onder AB als onder BA.
- Verwijder de contacten AA, BB, ... door de diagonaal nul te maken.
- Bepaal in elke rij som S van de contacten.
- De hoogste S indiceert een klinker.
- Trek de contacten van de gevonden klinker *tweemaal* af van elke S van een nog niet toegekende letter waarin die klinker voorkomt.
- Herhaal totdat geen enkele som meer groter is dan nul.

De algoritme baseert zich op de grotere contactvariatie van de klinkers ten opzichte van de medeklinkers. Net als andere algoritmen is ook deze niet 100% betrouwbaar. In het Engels bijvoorbeeld zal de letter T nogal eens onder de klinkers figureren. Natuurlijk is de lengte van de tekst waarover geteld wordt ook van belang. Hoe langer hoe beter.

Een klein (en daarom onvolledig) voorbeeld levert het zinnetje:

VYAND NADER WATERLINIE STOP FORT ASPEREN PARAAT

	A	D	E	F	I	L	N	O	P	R	S	T	V	W	Y	Σ	Σ	Σ	Σ
A	0	1					2	1	2	1	3		1	1		12			
D	1	0	1				2						1			5	3	1	1
E		1	0		1		1		1	4	1	1				10	10		
F				0				1	1							2	2	2	0
I			1		0	1	2									4	4	2	2
L					1	0				1						2	2	2	2
N	2	2	1		2		0		1							8	4	2	2
O				1				0	1	1		1				4	4	4	
P	1		1	1			1	1	0		1					6	4	2	0
R	2		4			1		1		0		2				10	6	-2	-4
S	1		1						1		0	1				4	2	0	0
T	3		1					1		2	1	0				8	2	0	-2
V													0	1		1	1	1	1
W	1	1												0		2	0	0	0
Y	1												1	0		2	0	0	0

Tabel 6.9 Bepaling van klinkers volgens Sukhotin

In tabel 6.9 is dit voorbeeld uitgewerkt. Als klinkers worden achtereenvolgens gevonden de A, E en O. Daarna is de berekening gestopt omdat door het geringe aantal letters voortgaan niet zinvol is.

6.6 Symmetrie in positie

Voor het oplossen van de eenvoudige periodieke polyalfabetische substitutie zoals de Vigenère of de Beaufort is het bepalen van de voor de vercijfering gebezigde sleutel voldoende. De rest volgt uit het feit dat in deze systemen het klaartekst- en het cijferalfabet de standaardvolgorde hebben. Anders ligt het wanneer hetzij het klaartekst- dan wel het cijferalfabet gepermuteerd is^{*}. Dan moet ook die volgorde worden bepaald. Elke sleutelletter gebruikt dezelfde, maar verschoven versie van dit alfabet. Daarom kunnen bij de ene sleutelletter gevonden delen van de alfabetpermutatie worden benut om het gevonden deel bij een andere sleutelletter te completeren. Het systematiseren hiervan is door William Friedman *symmetrie in positie* genoemd^[12]. Deze techniek zal worden uitgelegd aan de hand van het volgende voorbeeld.

Het is zomer 1759. De Zevenjarige Oorlog tussen Frankrijk en Engeland strekt zich ook uit tot de koloniën in Noord-Amerika. Het Franse gebied loopt op dat moment van Quebec in Canada via de grote meren en Niagara tot aan de vallei van de Ohio en Louisiana. Het is het doel van de Britten om door dit Franse gebied heen te breken en aldus het verre westen voor kolonisatie open te leggen. Het krijgsplan van de Britse generaal Amherst behelst een gelijktijdige aanval op Niagara, Montreal en Quebec. Fort Duquesne is al veroverd en omgedoopt tot Pittsburg, naar de Engelse eerste minister Sir William Pitt. Generaal Prideaux is op weg naar het door de Fransen in 1726 gebouwde en recent versterkte Fort Niagara. Generaal Amherst, die in het volgende jaar Montreal zal veroveren, stuurt hem de volgende missive^[13].

S KDNQNYT QAKY XQA 21ST FTXQ OBP&OP EUTRU TGFFOWMT MKZMFRTZNT UOM&A QCXY:T. TD M&M&EDNT
BPT FX&Y WF CKUBWSONUEHP EZEDSPK EHD WP& OXQX&PMYAY ZNUR&STW S&Z MUT& DZ WW SAMAM& ZHY
OXNIAQWQBWUXS DWM GTLX &WFCMU ZT WPWT ZEDPA DWM ZNOX YXIY XXZNTR W FI DWNKBLTWSH HDYZXCBS
KNCF<CAS TUUM&T&:CAS && P&OX GPWQN T XQXOZ HWH D PZCAK YQU ZNU U EN BQRCSXS U SRBEF QF&YDGH &&
UR TSM TCUP CQU OXCTT ZT&DST UOZ GFYY HQXMO U IXDE TX ZNY EWOPW LXNXPX W QBP.

* Het geval met klaartekst- en cijferalfabet beide gepermuteerd wordt hierna behandeld.

D PFZWDWS PWWKY BPT SZXAZRT FOW QF&Y I&AO &MWTP ZNT BW GCAGYO MFRY XQASL ZSQB&Z FOTK XQAONZ
YRX D UNTU UE MFG GUG[Y]R UOM&A IUM& CKTRXSXTR CHD GUG[Y]R BLLS XO C OC[E]SOUZ EGUG[Y]M SA&YQ
UFQQ RCK B DF&AU UE FY&LX&AO QPWCP GPWQN KBSZ GY XQA PEL& LTAGCFSPBWHY OENX GUG[Y]R UOZT
ZTU& E TDUWB FOS RPRTUQ&O BPT LYA[U] QDFA UU FXDSQICOH WP[F]&A[U] RYOGFKY XQA XPZMA FCQYAR
&MXDSQUZ[X]SP KFOPR UOZWO FOWGA CUMPA FOR YRXP LEOMOZU ZT QDWSA TWQSP WW KA XFBHDR E[C]XO
E[C]AZKM KWRZKYDG FOSXWYOSZNQ[Y]HE IT ZNTR YXIQ[Y] XQAA GUGH GY DST&LZU ZNY RWAHY US ZT HZ
PFKLS&W TP DRW ULSYY RGE&SR HWYR ZEDPAP WPWT FTFG TUUA YRXK ENQ[Y]I QZRWSPK ENBP SXDK GDT FOS
HWHK KDAA &Q LETUYMS X NUA& YC[E] FG QBPDUF.

Bovenstaande tekst is een transcriptie van het handschrift. Niet van alle letters is de identiteit zeker. Bij twijfel is de meest waarschijnlijke letter genomen en staat het beste alternatief er tussen rechte haken achter. Verder dient men behalve op fouten door slechte leesbaarheid van het handschrift ook bedacht te zijn op fouten gemaakt bij het coderen. Het veelvuldig voorkomen van het letterteken & valt op. Maar bij nadere beschouwing blijkt dit te worden gecompenseerd door het ontbreken van de letters J en V. Daarom zal & wel een letter voorstellen en zijn er 25 letters.

Waarschijnlijk is het een of andere polyalfabetische substitutie. Het bericht stamt uit het midden van de 18^e-eeuw en dan is een Vigenère geen slechte gok. Er blijven daarbij nog vragen genoeg over. Hoeveel substitutie-alfabetten zijn in het spel? Zijn deze door verschuiving van één alfabet ontstaan of is sprake van verschillende alfabetten? Is de sleutelprogressie regelmatig of onregelmatig?

Om te beginnen wordt een berekening van de periode uitgevoerd volgens de methode van [sectie 6.3.3](#). Van alle mogelijkheden is periode 5 (en veelvouden daarvan) de beste met een gemiddelde waarde voor de phi-test van 0,0502. De overige mogelijkheden voor de periode scoren tussen 0,0418 en 0,0434. Dit resultaat valt een beetje laag uit – iets in de buurt van 0,06 had er betrouwbaarder uitgezien. Omdat het bericht duidelijk uit twee afzonderlijke blokken bestaat, worden deze ook nog eens apart bekeken. In beide gevallen komt 5 er weer als beste uit. In het eerste blok geeft de phi-test de wat teleurstellende waarde 0,0492, maar in het tweede blok een meer bemoedigende 0,0566. Periode 5 wordt als werkhypothese aanvaard.

Het eerste aangrijpingspunt is de gelukkige omstandigheid dat bij de vercijfering het woordbeeld intact gebleven is. Direct van nut zijn natuurlijk de éénletter-woorden. In het Engels zijn dit *a* en *i*. Er zijn negen verschillende van deze éénletter-woorden in het cryptogram, te weten S, T, U, W in het eerste blok en B, C, D, E, X in het tweede. Nog niet duidelijk is of er verschil bestaat tussen de vercijfering van het eerste en het tweede blok. Het verschil bij de bepaling van de periode geeft te denken. Het meest voor de hand liggend lijkt voorshands toch dat dit negen verschillende vercijferingen zijn van *a* en *i*. Het aaneensluiten van de cryptogramletters in de twee groepen is bemoedigend: BCDE en STUWX (de V komt niet voor in het cryptogram). Deze twee groepen zouden best kunnen staan voor *a* en *i* dan wel omgekeerd. De veronderstelling dat 5 de lengte van de periode is ondervindt nu steun, terwijl het ook plausibel wordt dat slechts één alfabetpermutatie is gebruikt.

Vervolgens valt bij de woorden van drie letters de betrekkelijk hoge frequentie (driemaal) van XQA en BPT op. Een goede gok lijkt hier een vertaling met *and* en *the*. Waar XQA ook nog voorkomt in de combinaties XQAQ, XQASL, XQAONZ suggereert dit woorden als *the*, *them*, *then* of *they*, enz. BPT zal dan wel het voegwoord *and* zijn.

Kan op basis van deze eerste veronderstellingen nog iets meer gezegd worden? Kijk hiervoor naar het slot van het eerste blok waar het groepje W QBP wordt aangetroffen. Als uit de toekenning van *and* aan BPT afgeleid mag worden dat *a* als substituenten BCDE heeft, dan moet W wel de *i* voorstellen en eindigt dit blok dus op *i ?an*. Dit suggereert klaartekst *I can*.

Weer een stapje verder in het nog tere weefsel van veronderstellingen levert het woord OXNIAQWQBWUXS. De groep QB is waarschijnlijk *ca* en staat op een positie passend bij het voor het Engels zeer plausibele wordeinde *cation*. En wat meer is, het spoort met de conclusie dat U de *i* kan zijn. In dit beginstadium van de oplossing is iedere aanwijzing dat het goede spoor gevonden meer dan welkom. Vooral in het begin van de cryptoanalyse schaatst men gewoonlijk over flinterdun ijs.

De combinatie BW wordt *at* en komt tot grote vreugde aan het begin van het tweede blok voor als een

woord van twee letters: *at* is een plausibel woord*. Een ander goed teken is de waarde *n* voor P, afgeleid uit QBP = *can*. Over de P vallen twee dingen te zeggen. In de eerste plaats dat 9 woorden op een P eindigen. Een plaats op het eind van een woord past goed bij letter n. In de tweede plaats de letters die aan P voorafgaan, zijnde BCDESTUWX. In deze groep zijn inmiddels de klinkers a en i geplaatst. Van de n is bekend dat deze zich zeer graag laat voorafgaan door een klinker. Hieronder een compilatie van de gevonden fragmenten in onderlinge relatie op basis van de veronderstelde sleutellengte 5.

1	2	3	4	5	1	2	1	2	3	4	5	1	2
W	Q	B	P	.	.	.	=	i	c	a	n	.	.
.	.	B	P	T	.	.	=	.	.	a	n	d	.
.	.	B	W	.	.	.	=	.	.	a	t	.	.
.	Q	B	W	U	X	S	=	.	c	a	t	i	o

Het wordt tijd om te zien of deze gevolgtrekkingen zich in een substitutieschema laten onderbrengen. De gevonden letters worden in hun respectievelijke 5 alfabetten geplaatst. Links vooraan in onderstaande tabel staat het nummer van het substitutiealfabet, bovenaan staan de letters van het cijferalfabet (& arbitrair achteraan) en daaronder de tot nu toe gevonden overeenkomstige letters van de klaartekst.

	A	B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U	W	X	Y	Z	&	
1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
2	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
3	.	a	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
4	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
5	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

* Het is zelfs mogelijk om hier te gokken op at Quebec voor BW GCAGY0, maar dat is in dit stadium wel wat gewaagd.

Het mechanisme van *symmetrie in positie* laat zich met dit schema meteen benutten. Omdat er maar één alfabet is – in vijf onderling verschoven posities gebruikt – zullen de rijen in het schema vijf keer hetzelfde alfabet vertonen. Als in alfabet 1 de letters io naast elkaar staan en in 5 de letters di, dan kan het niet anders of in beide lopen de reeksen door tot dio. Hetzelfde geldt voor c.n in alfabet 2 en n in 4, waardoor in die laatste rij de letter c kan worden geplaatst. De letter t in alfabet 4 kan een plaatsje in alfabet nummer 2 krijgen.

	A	B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U	W	X	Y	Z	&
1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	d	i	o	.	.
2	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	c	.	n	.	.	.	.	.	t	.
3	.	a	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
4	.	.	.	.	.	.	.	.	.	.	.	.	.	c	.	n	.	.	.	.	.	t	.	.	.
5	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	d	i	o	.	.

Nu hierdoor meer letters kunnen worden ontcijferd is de volgende stap het opsporen van woorden waarin genoeg bekende letters staan om op gevoel de rest van het woord te kunnen aanvullen. Aan het slot van het tweede blok staat als laatste woord QBPDUF met sleutelletter 2 als beginpositie. Dat geeft de klaartekst *can?d?*. Dat moet wel *Canada* zijn en levert daarmee de positie van letter a in alfabet 2 en 5. Ingevuld in het schema

	A	B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U	W	X	Y	Z	&	
1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
2	.	.	.	.	a	.	.	.	.	.	.	.	.	.	.	.	c	.	n	.	.	.	.	.	.	t
3	.	a	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
4	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	c	.	n	.	.	.	.	.	.	t
5	.	.	.	a	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

Door het vinden van a in alfabet 2 kan door de afstand tot c.n te nemen, de a in alfabet nummer 4 worden geplaatst onder de C. En met dio kan a in 1 onder de E komen.

	A	B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U	W	X	Y	Z	&	
1	.	.	.	.	a	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	d	i	o	.	.	.
2	.	.	.	.	.	a	.	.	.	.	.	.	.	.	.	.	c	.	n	.	.	.	.	.	t	.
3	.	a	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
4	.	.	a	.	.	.	.	.	.	.	.	.	.	c	.	n	.	.	.	.	.	.	t	.	.	.
5	.	.	.	a	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	d	i	o	.	.	.

Daarmee zijn de mogelijkheden echter nog niet uitgeput. Als gevolg van de *symmetrie in positie* kunnen met de a als linking pin alle sleutelposities aan elkaar gekoppeld worden.

	A	B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U	W	X	Y	Z	&
1	.	.	.	.	a	.	.	.	.	.	.	.	.	c	.	n	.	.	d	i	o	t	.	.	.
2	.	.	.	.	.	a	.	.	.	.	.	.	.	c	.	n	.	.	d	i	o	t	.	.	.
3	.	a	.	.	.	.	.	.	.	.	.	.	.	c	.	n	.	.	d	i	o	t	.	.	.
4	.	.	a	.	.	.	.	.	.	.	.	.	.	c	.	n	.	.	d	i	o	t	.	.	.
5	.	.	.	a	.	.	.	.	.	.	.	.	.	c	.	n	.	.	d	i	o	t	.	.	.

Het begin van het bericht luidt: S KDNQNYT QAKY XQA 21ST. Aangenomen dat alfabet 3 het beginpunt is, kan dit worden ontcijferd tot: *I ?a?c??d ???? t?? 21st*. Nu bestond al het vermoeden dat XQA = *the*, hetgeen dit zinnetje al direct begrijpelijker maakt: *I ?a?c??d he?? the 21st*. Na een c komt vaak een h. Dan moet de op een na laatste letter van dit woord wel een klinker zijn en e is hiervoor een goede kandidaat. Nu valt te lezen *I ?a?ched he?? the 21st*. Het woord *marched* valt te herkennen, een woord dat goed past in de context van een militaire campagne. Het begin van het bericht kan nu worden gelezen als *I marched he?e*

the 21st en meteen blijkt dat de ontbrekende letter in *he.e* een *r* moet zijn. Op basis deze gevolgtrekkingen wordt het substitutieschema verder aangevuld tot

	A	B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U	W	X	Y	Z	&
1	.	.	.	.	a	.	.	.	.	.	.	.	.	r	.	c	h	n	.	.	d	i	o	t	.
2	e	.	.	.	a	.	.	.	.	.	.	.	.	.	.	c	.	n	.	.	d	i	o	t	.
3	.	a	.	.	.	.	.	.	.	.	.	.	.	r	.	c	h	n	.	.	d	i	o	t	.
4	.	.	a	.	.	.	.	.	.	.	m	.	.	c	.	n	.	.	d	i	o	t	.	e	.
5	.	.	.	a	.	.	.	.	.	.	.	.	.	.	.	c	.	n	.	.	d	i	o	t	.

En weer met behulp van de *symmetrie in positie* verder gecompleteerd

	A	B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U	W	X	Y	Z	&
1	.	.	.	.	a	.	.	.	.	.	.	m	r	.	c	h	n	.	.	d	i	o	t	.	e
2	e	.	.	.	a	.	.	.	.	.	.	m	r	.	c	h	n	.	.	d	i	o	t	.	.
3	.	a	.	.	.	.	.	.	.	.	.	m	r	.	c	h	n	.	.	d	i	o	t	.	e
4	.	.	a	.	.	.	.	.	.	.	.	m	r	.	c	h	n	.	.	d	i	o	t	.	e
5	.	.	.	a	.	.	.	.	.	.	.	m	r	.	c	h	n	.	.	d	i	o	t	.	e

De structuur van het substitutieschema is hiermee duidelijk geworden. Het is inderdaad één enkele permutatie van het alfabet, steeds over één plaats verschoven. Voor de completering van het alfabet wordt gezocht naar woorden waarvan nu een groot deel bekend is. Neem het al eens gebruikte OXNIAQWBWUXS. Dit laat zich thans lezen als *comm.nication*. De vierde A is ongetwijfeld de *u*. Met ontcijfering van dit woord valt tevens alle twijfel weg – zo die overgebleven mocht zijn – of de goede weg naar de oplossing is ingeslagen. Het is nog slechts een kwestie van geduldig invullen totdat alle letters geduid zijn. Het eindresultaat is een alfabetpermutatie waaraan slechts vier letters ontbreken.

	A	B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U	W	X	Y	Z	&
1	k	p	u	.	a	f	l	q	w	.	.	m	r	x	c	h	n	s	y	d	i	o	t	.	e
2	e	k	p	u	.	a	f	l	q	w	.	.	m	r	x	c	h	n	s	y	d	i	o	t	.
3	.	a	f	l	q	w	.	.	m	r	x	c	h	n	s	y	d	i	o	t	.	e	k	p	u
4	u	.	a	f	l	q	w	.	.	m	r	x	c	h	n	s	y	d	i	o	t	.	e	k	p
5	p	u	.	a	f	l	q	w	.	.	m	r	x	c	h	n	s	y	d	i	o	t	.	e	k

Bij nadere beschouwing blijkt ook de constructie van het alfabet. De regelmatige afstand van 5 letters tussen de QRST en UWXY verraadt dit. De basis is een 5x5-tabel waar de letters kolomsgewijs zijn uitgenomen.

A	B	C	D	E
F	G	H	I	K
L	M	N	O	P
Q	R	S	T	U
W	X	Y	Z	&

Wanneer men de opeenvolging van de substituties nagaat, blijkt dat degene die het cryptogram heeft gemaakt op een aantal plaatsen fouten heeft begaan. De substitutie springt dan bijvoorbeeld van alfabet 3 naar alfabet 5. Deze fouten verstoren de regelmaat in de opeenvolging van de sleutelletters en verklaren waarom de periodebepaling vooral in het eerste blok minder duidelijk was. In de ontcijferde tekst is steeds de beste letter gekozen op die plaatsen waar een alternatief in de transcriptie van de oorspronkelijke tekst staat. Op andere plaatsen is waarschijnlijk een codeerfout gemaakt. De letter u moet meestal worden gelezen als een v.

I marched here the 21st with rangers liyt infantry gremadiers three batt:s of regulars and fiue of prouinnicals leauing all the restexcept thosenot yet code up to secure the communication fom fort edward to this place fom thht time tothis I am forwarding Watteaus whaleboaps storesu:&kn up here which I think will neuer end tho I am assumed I shall haueall up in six duys and ready topass the lake which I will do the first moment I can.

A captain Stobo and Steuens who haue been prison ers at Quebec made their escape from therec the f irst of mab with three more prisoneis ull with army in a cannoe after seuen days suw a fleet of fourteen ships which must be the slme Bouqainuille came with they took a sloop and schooner and got safe to Louisburg they declare the enemy wanted prouisions m?cch their whole force whs then marched to Saint Iohns to be equally for Quebc Montreal andtionderogl by this time they will be assured the siege is to be carryed on and tryto defend both places this will stop them from sending aman your way and will make us masters I hope of al Canada.

6.7 Isomorfie

Isomorfie kan worden opgevat als de veralgemenisering van de herhaling. Twee letterreeksen zijn *isomorf* als ze door een monoalfabetische substitutie in elkaar kunnen worden overgevoerd. Isomorfie is in de cryptoanalyse een krachtig hulpmiddel gebleken, dat tegen veel cryptosystemen kan worden ingezet.

Het is duidelijk dat isomorfen veel vaker kunnen voorkomen dan herhalingen. Bij periodieke polyalfabeten zijn herhalingen in het cryptogram het gevolg van herhalingen in de klaartekst die precies een geheel aantal malen de sleutellengte uit elkaar liggen. Bij lange sleutels zal dit niet vaak het geval zijn. Isomorfen zijn daarentegen niet afhankelijk van zulke positieverschillen. Er is echter een 'maar' bij. Allereerst het feit dat isomorfen moeilijker te spotten zijn dan herhalingen, zeker wanneer men dat met de hand wil doen. Bij isomorfen moet gezocht worden naar *gelijke letterpatronen* in plaats van naar *gelijke letters*.

Een tweede 'maar' heeft betrekking op de plaatsen waar de isomorf begint en eindigt. Bij herhalingen is de kans dat de eerste en laatste letter(s) slechts bij toeval overeenstemmen niet zo heel groot. Er bestaat een kans van 1 op 26 dat de eerste letter toevallig dezelfde is, 1 op 676 dat dit voor de twee voorgaande letters geldt. Bij isomorfen is de kans dat de eerste en laatste letter(s) slechts toevallig voldoen aan het

isomorfiepatroon veel groter, namelijk $25/26^e$ per letter. Het is daarom moeilijker om het begin- en het eindpunt van een isomorf met zekerheid vast te leggen. Letten op herhaalde letters in het letterpatroon biedt nog de meeste houvast.

Maar nu eerst een uiteenzetting over het verschijnsel isomorfie zelf en het optreden ervan in polyalfabetische substituties, gevolgd door het bepalen van het cijferalfabet in polyalfabetische systemen met behulp van isomorfen. Als voorbeeld is de Variant Vigenère gebruikt om te vercijferen*. Het vercijferen geschiedt dus door in de sleutelrij te gaan naar de klaartekstletter en de vercijferde letter af te lezen bovenaan de tabel. Links met sleutel FEEST (periode 5) en rechts met een eenvoudige progressieve sleutel ABC...Z (periode 26).

periodieke sleutel vercijfering patroon	progressieve sleutel vercijfering patroon
KAARSENMAKER ABBCDEFGBAEC FEESTFEESTFE	KAARSENMAKER ABBCDEFGBAEC ABCDEFGHIJKL
FWWZZZJIIRZN ABBCCCEEFCEG	KZYOOZHFSBUG ABCDDBEFGHIJ
KAARSENMAKER ABBCDEFGBAEC EESTFEESTFEE	KAARSENMAKER ABBCDEFGBAEC BCDEFGHIJKLM
GWIYNAJUHFAN ABCDEFGHIJFE	JYXNNYGERATF ABCDDBEFGHIJ

Vercijferd is het woord KAARSENMAKER dat letterpatroon ABBCDEFGBAEC heeft†. Daaronder in de linkerkolom staan twee vercijferingen met sleutel FEEST, die slechts daarin verschillen dat de beginpositie van de sleutel een plaats verder ligt. De letterpatronen van deze twee vercijferingen lijken niet op elkaar en evenmin op dat van de klaartekst. Hier treedt geen isomorfie op. Dat is wel het geval bij de progressieve

* Zie ook [figuur 4.2](#) voor het verschil tussen de Variant Vigenère en de gewone Vigenère.

† Het letterpatroon is gemaakt door elke nieuw optredende letter te vervangen door de eerstvolgende nog niet gebruikte letter uit het alfabet, te beginnen met A.

sleutel aan de rechterkant. Ook hier is tussen de eerste en tweede vercijfering de sleutel een plaats opgeschoven. Overeenkomst tussen het letterpatroon van de klaartekst en de cijfertekst is er niet, maar wel zijn de patronen van de twee vercijferingen gelijk. De cijferteksten KZY00ZHFSBUG en JYXNNYGERATF zijn isomorf*.

Het volgende voorbeeld laat zien dat dit ook zo blijft als de progressie in het alfabet meerdere stappen tegelijk neemt. Aan de rechterkant is de progressie in de sleutel twee stappen per vercijfering. Ook dan leidt verschuiving van de sleutel tot isomorfen, zij het dat dit een andere isomorf is dan in het vorige geval.

progressie stap = 1 vercijfering patroon	progressie stap = 2 vercijfering patroon
KAARSENMAKER ABBCDEFGBAEC ABCDEF GHIJKL KZY00ZHFSBUG ABCDDBEFGHIJ	KAARSENMAKER ABBCDEFGBAEC ACEGIKMOQS UW KYWLKUBYKSKV ABCDAEFBAGAH
KAARSENMAKER ABBCDEFGBAEC BCDEF GHIJKLM JYXNNYGERATF ABCDDBEFGHIJ	KAARSENMAKER ABBCDEFGBAEC BDFHJLNPRTVX JXVKJTAXJRJU ABCDAEFBAGAH

Het progressiemechanisme kan zelfs een periodiek patroon van progressiestappen hebben. Ook dan ontstaan isomorfen, en wel als de herhalingen in de tekst een geheel aantal perioden van het stappenpatroon uit elkaar staan. Isomorfie vertoont daarin een gedrag analoog aan dat van herhalingen in de gewone periodieke substitutie.

Wat is het effect van een permutatie in het klaartekst- en/of cijferalfabet? Tabel 6.10 heeft een gemengd klaaralfabet en een verschillend gemengd cijferalfabet. De opstelling van de opeenvolgende sleutelletters volgt het sleutelwoord BEDOUINENKAMP van het cijferalfabet. Het klaaralfabet is afgeleid van de sleutel

* De relatie is zelfs een eenvoudige Caesar-verschuiving, omdat de tabel het alfabet in standaardvolgorde bevat.

	W	O	E	S	T	Y	N	A	B	C	D	F	G	H	I	J	K	L	M	P	Q	R	U	V	X	Z
B	B	Z	Y	X	W	V	T	S	R	Q	L	J	H	G	F	C	P	M	A	K	N	I	U	O	D	E
E	E	B	Z	Y	X	W	V	T	S	R	Q	L	J	H	G	F	C	P	M	A	K	N	I	U	O	D
D	D	E	B	Z	Y	X	W	V	T	S	R	Q	L	J	H	G	F	C	P	M	A	K	N	I	U	O
O	O	D	E	B	Z	Y	X	W	V	T	S	R	Q	L	J	H	G	F	C	P	M	A	K	N	I	U
U	U	O	D	E	B	Z	Y	X	W	V	T	S	R	Q	L	J	H	G	F	C	P	M	A	K	N	I
I	I	U	O	D	E	B	Z	Y	X	W	V	T	S	R	Q	L	J	H	G	F	C	P	M	A	K	N
N	N	I	U	O	D	E	B	Z	Y	X	W	V	T	S	R	Q	L	J	H	G	F	C	P	M	A	K
K	K	N	I	U	O	D	E	B	Z	Y	X	W	V	T	S	R	Q	L	J	H	G	F	C	P	M	A
A	A	K	N	I	U	O	D	E	B	Z	Y	X	W	V	T	S	R	Q	L	J	H	G	F	C	P	M
M	M	A	K	N	I	U	O	D	E	B	Z	Y	X	W	V	T	S	R	Q	L	J	H	G	F	C	P
P	P	M	A	K	N	I	U	O	D	E	B	Z	Y	X	W	V	T	S	R	Q	L	J	H	G	F	C
C	C	P	M	A	K	N	I	U	O	D	E	B	Z	Y	X	W	V	T	S	R	Q	L	J	H	G	F
F	F	C	P	M	A	K	N	I	U	O	D	E	B	Z	Y	X	W	V	T	S	R	Q	L	J	H	G
G	G	F	C	P	M	A	K	N	I	U	O	D	E	B	Z	Y	X	W	V	T	S	R	Q	L	J	H
H	H	G	F	C	P	M	A	K	N	I	U	O	D	E	B	Z	Y	X	W	V	T	S	R	Q	L	J
J	J	H	G	F	C	P	M	A	K	N	I	U	O	D	E	B	Z	Y	X	W	V	T	S	R	Q	L
L	L	J	H	G	F	C	P	M	A	K	N	I	U	O	D	E	B	Z	Y	X	W	V	T	S	R	Q
Q	Q	L	J	H	G	F	C	P	M	A	K	N	I	U	O	D	E	B	Z	Y	X	W	V	T	S	R
R	R	Q	L	J	H	G	F	C	P	M	A	K	N	I	U	O	D	E	B	Z	Y	X	W	V	T	S
S	S	R	Q	L	J	H	G	F	C	P	M	A	K	N	I	U	O	D	E	B	Z	Y	X	W	V	T
T	T	S	R	Q	L	J	H	G	F	C	P	M	A	K	N	I	U	O	D	E	B	Z	Y	X	W	V
V	V	T	S	R	Q	L	J	H	G	F	C	P	M	A	K	N	I	U	O	D	E	B	Z	Y	X	W
W	W	V	T	S	R	Q	L	J	H	G	F	C	P	M	A	K	N	I	U	O	D	E	B	Z	Y	X
X	X	W	V	T	S	R	Q	L	J	H	G	F	C	P	M	A	K	N	I	U	O	D	E	B	Z	Y
Y	Y	X	W	V	T	S	R	Q	L	J	H	G	F	C	P	M	A	K	N	I	U	O	D	E	B	Z
Z	Z	Y	X	W	V	T	S	R	Q	L	J	H	G	F	C	P	M	A	K	N	I	U	O	D	E	B

Tabel 6.10 Vigenère-tabel met sleutels WOESTYN en BEDOUINENKAMP

WOESTYN. De vercijfering is weer gedaan met een Variant Vigenère. Ook hier blijken met een progressieve sleutel herhalingen in de klaartekst isomorf op te leveren in de cijfertekst*.

klaartekst	sleutel	vercijfering	patroon
KAARSENMAKER	BEDOUINKAMPC	PPQFFTWXWEC	AABCCDEFEGHA
KAARSENMAKER	EDOUINKAMPCF	QQRGGYOZOSDQ	AABCCDEFEGHA
...			
KAARSENMAKER	PCFGHJLQRSTV	SSTRRIDBDGPS	AABCCDEFEGHA

6.7.1 Reconstructie alfabet

Het cijferalfabet kan worden gereconstrueerd als voldoende isomorfe relaties ter beschikking staan. Vercijfer met Variant Vigenère, gebruikmakend van de gewone ongepermuteerde [tabel 4.2](#) uit [hoofdstuk 4](#). Het woord KAARSENMAKER wordt vercijferd met de progressieve sleutel ABCD... en vervolgens met BCDE... tot het isomorf paar

KAARSENMAKER → KZY00ZHFSBUG en JYXNNYGERATF

De vaste relatie tussen de letters op overeenkomstige posities is duidelijk zichtbaar. De eerste letters van de isomorf zijn respectievelijk K en J. Van K naar J is het één stap naar links in het cijferalfabet, dat hier gewoon een standaard alfabet is. Maar het is ook één stap naar links voor de tweede letters van de isomorf, namelijk van Z naar Y en eveneens één stap voor de derde van Y naar X, enz. De paren letters afkomstig van de beide isomorfe letterreeksen staan in een constante relatie ten opzichte van elkaar. De isomorf verklapt als het ware dat in het cijferalfabet K en J even ver van elkaar staan als Z en Y, Y en X ...

Bovendien kunnen zulke relaties aan elkaar gekoppeld worden. Wanneer het één stap naar rechts is van B naar C en ook één stap naar rechts van C naar D, dan is er dus een keten BCD met steeds één stap naar

* **Let op!** In zo'n gepermuteerde tabel gaat het Vigenère-vercijfer/ontcijferschema wel, maar het Beaufort-schema niet op.

rechts tussen de opeenvolgende letters

$$BC + CD \rightarrow BCD$$

Als gevolg van deze constante relaties zullen de letters bij de reconstructie van het alfabet op de juiste positie ten opzichte van elkaar geplaatst kunnen worden. Nu is dat bij een Vigenère met standaardalfabet triviaal, omdat de volgorde van dat alfabet bekend is. Maar in het geval van een gemengd alfabet met onbekende lettervolgorde geldt de constante relatie ook. Alleen is dan de waarde van de constante, het aantal tussenliggende stappen, een onbekende. Een voorbeeld van die situatie leveren eerder gedane vercijferingen. Het woord KAARSENMAKER vercijferd met de gepermuteerde [tabel 6.10](#) werd

$$KAARSENMAKER \rightarrow PPQFFTWXWEC \text{ en } SSTRRIDBDGPS$$

Ook hier is er een vaste relatie tussen letters op overeenkomstige posities. Dat wil zeggen: als het x stappen is van P naar S, dan is het ook x stappen van Q naar T, van F naar R, enz. Alle relaties tussen de letters van deze isomorf op een rij gezet

$$(PS) (QT) (FR) (TI) (WD) (XB) (EG) (CP)$$

En net zoals BC en CD aan elkaar geschakeld konden worden tot BCD, zo kan dat ook met deze relaties.

$$(CP) + (PS) \rightarrow (CPS) \text{ en voorts } (QTI) (FR) (WD) (XB) (EG)$$

Vergelijk dit nu eens met het (bekende) cijferalfabet van [tabel 6.10](#)

$$W O E S T Y N A B C D F G H I J K L M P Q R U V X Z$$

De vaste relatie is er inderdaad want C, P en S liggen steeds 10 plaatsen uiteen, idem voor de reeks QTI en al de andere relaties uit de isomorf. Ook als het alfabet onbekend is geldt dit, zij het dat in dat geval de onderlinge afstand onbekend is. Veronderstel dat nog een andere isomorf gevonden is. Bijvoorbeeld de volgende twee vercijferingen van het woord OVERWEGWACHTER

$$OVERWEGWACHTER \rightarrow THBMJFWMAYSXME \text{ en } ZBSHDNRHEWXPHV$$

Dit levert de volgende relaties

$$(TZ) (MHBSXP) (JD) (FN) (YWR) (AEV)$$

De afstand tussen de opeenvolgende letters van M naar H naar B is nu 21 stappen. Een andere afstand dan bij de vorige isomorf, maar dat is ook logisch. Het is immers een isomorf veroorzaakt door een andere klaartekst. Het zal ook duidelijk zijn dat door ineenvoegen van de twee stellen relaties meer informatie over het onderliggende cijferalfabet kan worden verkregen. Maar hoe?

Noem de afstand tussen de opeenvolgende letters in het eerste stel x en die in het tweede stel y . De relatie tussen x en y is volkomen onbekend en daarom is het beter om x en y als twee onafhankelijke variabelen te behandelen. Een mooie manier om twee onafhankelijke relaties in kaart te brengen is om de een horizontaal en de ander verticaal uit te zetten. Het zijn dan de twee onafhankelijke richtingen in het platte vlak.

Overblijft de vraag wat aan te vangen met de onbekende waarde van x respectievelijk y . De relatie CPS kan immers elk van de volgende mogelijkheden zijn, waarbij ? staat voor een vooralsnog onbekende letter

CPS

?C?P?S?

??C??P??S??

???C???P???S???

...

Bij gebrek aan beter mag verrassend genoeg een willekeurige keuze worden gemaakt! Het aardige is dat deze keuze voor sommige cryptosystemen geen enkel verschil blijkt te maken. Een andere keuze levert een equivalente oplossing die evengoed voldoet. Maar er zijn ook systemen waar de bepaling van de juiste mogelijkheid wel degelijk van belang is. In dat geval moet het gevonden alfabet nog een vervolgbewerking ondergaan*.

* De Wheatstone cryptograaf is zo'n systeem waarbij de keuze van de juiste oplossing cruciaal is.

In de tweede serie zit de lange reeks MHBSXP van 6 letters. Het is handig om deze horizontaal te leggen. Het wit tussen de afzonderlijke fragmenten stelt hun nog onbekende onderlinge afstand en volgorde voor.

TZ	MHBSXP	JD	FN	YWR	AEV
----	--------	----	----	-----	-----

Overal waar een letter gemeenschappelijk is met de eerstgevonden reeks (CPS)(QTI)(FR)(WD)(XB)(EG) kan de betreffende relatie in verticale richting (van boven naar beneden) worden aangekoppeld.

	C				
Q	XP?C	W		F	
TZ	MHBSXP	JD	FN	YWR	AEV
I	BS		R	D	G

Direct valt op dat als gevolg van overlappende letters de fragmenten met JD, FN en YWR in elkaar gepast kunnen worden.

	C		
Q	XP?C	FN	
TZ	MHBSXP	YWR	AEV
I	BS	JD	G

Op dit moment is het aantal koppelingsmogelijkheden uitgeput en moeten nieuwe isomorfen in stelling gebracht worden om verder te komen. Stel dat de reeks PCZ is gevonden. Hoe kan die in het diagram worden ingepast? Vrije keuze is er niet meer, want met de twee gekozen relaties zijn alle vrijheidsgraden opgebruikt. Er zit dus niets anders op dan in het diagram te zoeken naar aanknopingspunten die PCZ relateren aan de reeds gevonden betrekkingen. Zo'n aanknopingspunt bevindt zich boven in het tweede fragment bij XP?C. De reeks PCZ past hierin als deze met één extra tussenpositie wordt vergroot tot P?C?Z.

	C		
Q	XP?C	FN	
TZ	MHBSXP?C?Z	YWR	AEV
I	BS	JD	G

En nu kan via de Z het eerste fragment met het tweede worden verbonden.

	C		
	XP?C??Q	FN	
	MHBSXP?CTZ	YWR	AEV
	BS??I	JD	G

Zojuist werd een verband in horizontale richting gebruikt, maar dat kan ook in verticale richting of langs een of andere diagonaal. De H en P bevinden zich een paardensprong uit elkaar in het grootste fragment. Levert een andere isomorf nu $QW = HP$ dan is bekend hoe Q en W zich ten opzichte van elkaar verhouden en wordt middels deze letters het tweede fragment aan het eerste gelinkt. Aangezien het krijgen van een compleet alfabet op een horizontale rij het uiteindelijke doel is, is het profijtelijk om het diagram niet teveel in verticale richting te laten uitdijen. Daarom wordt eerst de Q een rij lager geplaatst met behulp van de C. Het tweede fragment wordt vervolgens aan die Q geplakt.

	C????????FN	
	XP?C??Q??YWR	
	MHBSXP?CTZQJD	AEV
	BS??I	G

De I komt tussen de P en C, de F en N kunnen naar beneden gehaald worden, enz. De reconstructie zal uiteindelijk het volgende alfabet opleveren

M H B S X P I C T Z Q J D Y W R K F N O U L G A E V

Alleen is dit niet het oorspronkelijke van WOESTYN afgeleide alfabet. Dat komt door de aanname voor de afstand tussen de letters in horizontale richting. In het werkelijke cijferalfabet staan MH niet naast elkaar, maar zitten 21 plaatsen tussen M en H. De bij de start van de analyse gemaakte keuze leidt tot dit resultaat. Een andere keuze zou een ander volgorde in het alfabet hebben opgeleverd.

1:	M	H	B	S	X	P	I	C	T	Z	Q	J	D	Y	W	R	K	F	N	O	U	L	G	A	E	V
3:	M	S	I	Z	D	R	N	L	E	H	X	C	Q	Y	K	O	G	V	B	P	T	J	W	F	U	A
5:	M	P	Q	R	U	V	X	Z	W	O	E	S	T	Y	N	A	B	C	D	F	G	H	I	J	K	L
7:	M	C	W	L	B	Z	K	A	X	J	N	V	I	Y	U	H	T	R	G	S	Q	F	E	P	D	O
9:	M	Z	N	H	Q	O	B	J	U	S	D	L	X	Y	G	P	W	A	I	R	E	C	K	V	T	F
11:	M	J	G	C	N	S	W	V	Q	L	I	F	B	Y	E	Z	U	P	K	H	D	A	T	O	X	R
15:	M	R	X	O	T	A	D	H	K	P	U	Z	E	Y	B	F	I	L	Q	V	W	S	N	C	G	J
17:	M	F	T	V	K	C	E	R	I	A	W	P	G	Y	X	L	D	S	U	J	B	O	Q	H	N	Z
19:	M	O	D	P	E	F	Q	S	G	R	T	H	U	Y	I	V	N	J	X	A	K	Z	B	L	W	C
21:	M	L	K	J	I	H	G	F	D	C	B	A	N	Y	T	S	E	O	W	Z	X	V	U	R	Q	P
23:	M	A	U	F	W	J	T	P	B	V	G	O	K	Y	Q	C	X	H	E	L	N	R	D	Z	I	S
25:	M	V	E	A	G	L	U	O	N	F	K	R	W	Y	D	J	Q	Z	T	C	I	P	X	S	B	H

Tabel 6.11 Decimaties van het gereconstrueerd alfabet

Het is niet nodig om de analyse nog eens over te doen met een andere beginkeuze. De letters van het gevonden alfabet hebben de goede relatie ten opzichte van elkaar *afgezien van hun onderlinge afstand*. Alle mogelijkheden kunnen eenvoudig gegenereerd worden door het alfabet uit te schrijven met steeds 1

letter verder te nemen, 3 letters verder te nemen, enz. Men verkrijgt op die manier de verschillende decimaties van het alfabet. Tabel 6.11 zet ze op een rij. Aangezien het alfabet $26 = 2 \times 13$ letters telt, behoeven alleen de intervallen zonder gemeenschappelijke factor met 26 te worden genomen*. Het oorspronkelijke WOESTYN-alfabet wordt gevonden als decimatie 5.

6.7.2 U.S. Army Cipher Disk

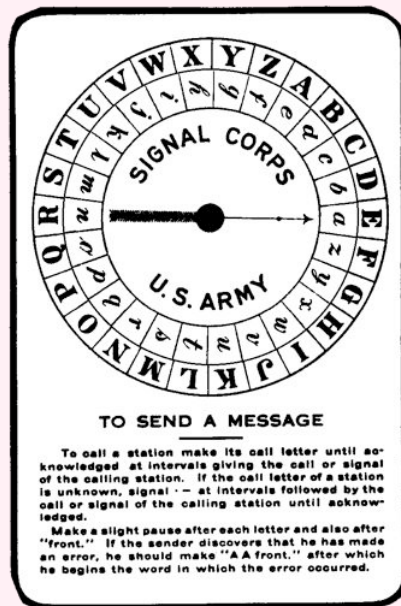
Omdat isomorfie lastige materie is en het reconstrueren van het alfabet niet altijd even glad verloopt, wordt een moeilijker voorbeeld uitgewerkt. Het cryptogram is gemaakt met de zogenaamde US-Army Cipher Disk van figuur 6.8[†]. Het probleem is afkomstig uit een cursus cryptoanalyse van het Amerikaanse leger[‡].

De US-Army Cipher Disk bestaat uit twee ringen (schijven), elk voorzien van een alfabet. Op de buitenste loopt het alfabet met de klok mee, op de binnenste tegen de klok in. De binnenste kan ten opzichte van de buitenste worden verdraaid. Gemakkelijk valt in te zien dat het apparaat op de afbeelding een gewone Beaufort-substitutie aflevert. De binnenste ring wordt zo gedraaid, dat de eerste letter van de sleutel tegenover de A op de buitenste staat. De te vercijferen of te ontcijferen letter wordt opgezocht op de binnenste ring, de pendant staat op de buitenste. Daarna wordt de volgende sleutelletter tegenover de A gezet. Geheel analoog aan de procedure voor het gebruik van de standaard Beaufort-tabel 4.3. Voor een progressieve sleutel wordt de binnenste ring bij elke vercijferoperatie het aantal stappen van de progressie verdraaid.

* Merk op dat men maar tot de helft hoeft te gaan, want daarna komen de alfabetten er in omgekeerde volgorde uit.

[†] Volgens kolonel Parker Hitt was deze Signal Corps disk aan het begin van de Eerste Wereldoorlog in 1914 nog bij het Amerikaanse leger in gebruik als veldcijfer.

[‡] *Army Extension Courses*, een verzameling opgaven bij William Friedman's *Military Cryptanalysis*.



OBDZR DRUAI PODB J RYYDR AOXYX WBMAZ OQNEY FJVRM KHRLQ PCQUO SLRXN
 EWWTE OUVZH NKAOA QFNQO USJIF IDEGM BYMLU KGZFQ ZZWCB OZCQE RNUXD
 TRANT YDVPW CMLYG IVBSD DTNXH HPAON SQAQJ OIZPR MEWAS QBLJU MXVTS
 HMLKH QMSAL VBZQL OMMLH ISWHP NKLKG QEUCK JKHOG THPMS PZVDH UDNLW
 HDHVI CITUX JZBKR OISES ZQORL HDDHV DSKIM BNRBU TFNDE GMBYM LUKGL
 IZLKV XDZZX JQBOO UOMUX CSENN BFZBZ SPMUD SORDJ CEWTL EDZYO EGUUE
 XWMFX VQJYA NZTKQ AKAXX JCXVU MVTDB XRIQE FXICA UQEZR YKTZK VZDVD
 QXYFP MMAOT IIMAW UZQNT XTHAV BINRI PWTBP LQPTK UAQQU ZXNYY WYINV
 KPVEJ SESZG TGUBC BLNFZ YNMPZ DKLEX HZYAD XRJPO PHDXF CKPDI DRLQU
 SIVEQ SJTQP ZGTPX LRQRM LICWI WGVPL QVEAA IKGCF AOAQO HJUYB UTFNS
 ESZZK EUQFP AVRPE VBKTR GDWSJ DBHPC TQUTA CANSL DTDBV YCJYY HTSRP
 AERNU KPAYR DQPAK UOCIT HGUYF ROOHL HFGZF IBFWG BWTBP OBDZR DRUEV
 PLRHT KBKLU RMJCK LELHV Y

Figuur 6.8 US-Army Cipher Disk en het op te lossen cryptogram

De substitutie kan worden gecompliceerd door niet met normale, maar met gepermuteerde alfabetten te werken. In het onderhavige probleem zijn *beide ringen* van een onderling verschillend gemengd alfabet voorzien. Verder is bekend dat niet met een korte periodieke sleutel wordt gewerkt, maar dat na elke vercijfering de binnenste ring één positie naar rechts wordt gedraaid. Als ook nog de beginpositie gegeven is, kan een bericht worden gecodeerd respectievelijk gedecodeerd. Met niet meer dan deze algemene gegevens moet het gegeven cryptogram worden opgelost.

De eerste hindernis is het opzoeken van betrouwbare isomorfen. Moet men met de hand isomorfen zoeken, dan is de beste methode om uit te gaan van een opvallend herhalingspatroon. Gezocht wordt vervolgens naar eenzelfde patroon, maar met andere letters. Met de computer is het zoeken natuurlijk veel

comfortabeler. De isomorfen moeten met zorg worden gekozen, waarbij vooral de onzekerheid omtrent het precieze begin en eind parten kan spelen.

Er zijn wel een paar vuistregels te geven voor het inschatten van de betrouwbaarheid van isomorfen. Hoe langer de isomorf, hoe minder kans dat deze vals is. De uiteinden van een isomorf zijn doorgaans twijfelachtig, het middenstuk is redelijk safe. Aan de uiteinden voegen zich gemakkelijk letters, die niet afkomstig zijn van dezelfde klaartekst, bij het isomorfiepatroon. Een bruikbaar criterium is het aantal herhaalde letters in het patroon: de kans op toevallige herhalingen is niet zo groot. Zoek dus vooral isomorfen met zoveel mogelijk herhaalde letters. Die herhaalde letters zijn vaak ook betrouwbare indicatoren voor het begin en eind van een isomorf. Meer mogelijkheden tot controle komen beschikbaar naarmate de reconstructie vordert. Met de al gevonden relaties kan een voorzichtig gekozen isomorf soms aan de uiteinden worden uitgebreid. Komen er daarentegen tegenstrijdigheden tevoorschijn, dan worden de gewantrouwde relaties weggelaten. Een valse relatie als gevolg van een fout in de codering of van een vermindering bij de transmissie van het bericht kan overal in de isomorfen opduiken.

```
positie 263: NDEGMBYMLUKGLIZLKVDZZXJQB00UOM
positie 408: HAVBINRIPWTBPLQPTKUAQQUZXNYYWYI
relaties    (DA) (EVKT) (GBNH) (MILP) (OYR) (JZQXUW)

positie 174: ALVBZQLOMMLHISWHPNK
positie 320: LEDZY0EGUUEXWMFXVQJ
relaties    (ALE) (PVD) (BZY) (NQOG) (SMU) (HX) (IWF) (KJ)
```

Hier staan de isomorfen waarmee de reconstructie van het cijferalfabet wordt begonnen. De eerste stap wordt gezet met de reeks JZQXUW, waarna de analyse – afgezien van enkele valse relaties* – probleemloos van start gaat. Drie grote brokstukken GBNHMILP, JZQXUWEVKT en OYRFSDA van het alfabet worden gevonden. Het eindstadium van de reconstructie is hieronder afgebeeld.

* De relaties BZY en KJ uit de tweede isomorf blijken later niet te kloppen.

[	SDA	]
[	GBNHMILP	]
[	JZQXUVEVKT.....	]
[	OYRFSDA	]
[	G	]

Nu alle beschikbare relaties uitgeput zijn, staan twee wegen open om het alfabet te completeren. De eerste is zoeken naar andere isomorfen, waarmee de brokstukken aan elkaar gelijmd kunnen worden. Ook korte isomorfen zijn nu bruikbaar, omdat ze kunnen worden gecontroleerd aan de hand van het diagram. Er is echter een makkelijker route naar de oplossing. Omdat op de C na alle letters gevonden zijn, is er maar een beperkt aantal mogelijkheden waarop de brokstukken kunnen worden gecombineerd. Ze moeten per slot van rekening passen in een alfabet van 26 letters. De volgende 6 mogelijkheden komen in aanmerking

1. GBNHMILPJZQXUVEVKT OYRFSDAC
2. GBNHMILPJZQXUVEVKT COYRFSDA
3. GBNHMILPCJZQXUVEVKT OYRFSDA
4. OYRFSDACJZQXUVEVKT GBNHMILP
5. COYRFSDAJZQXUVEVKT GBNHMILP
6. POYRFSDAJZQXUVEVKT CGBNHMIL

Onder deze is er slechts één, nummer 1, die de verticale relaties NQOG, SMU respecteert. Die variant moet dus het gezochte cijferalfabet zijn.

G B N H M I L P J Z Q X U W E V K T O Y R F S D A C

Het alfabet hoeft niet gebaseerd te zijn op een sleutelwoord, maar die mogelijkheid bestaat natuurlijk wel. Omdat in de oplossing niet direct een sleutelwoord valt te herkennen, worden de decimaties ervan

onderzocht. Succes wordt geboekt met deze decimatie

G P E F N Z K D M X O C L W R B J V S H Q T A I U Y

Nu zal niet iedereen daarin direct een sleutelwoord zijn opgevallen. Men bedenke echter dat dit probleem bedoeld was voor meer gevorderde cryptoanalisten in spé, zodat een complicatie valt te verwachten. Het zijn de regelmatige afstanden tussen de letters Z, X, W, V, die doen denken aan een kolomtranspositie. En inderdaad, het alfabet is gemaakt van sleutelwoord KEYSTROKE en met een kolomtranspositie gepermuteerd. De transpositiesleutel is 6432175, die eveneens is gebaseerd op dit sleutelwoord. Want het equivalent van KEYSTRO is 2175643 (zie [sectie 2.3.2](#)).

6	4	3	2	1	7	5
T	R	O	K	E	Y	S
A	B	C	D	F	G	H
I	J	L	M	N	P	Q
U	V	W	X	Z		

Cijferalfabet

EGPVL FNUMCJSODKTYBIRXZAHQW						
	6	4	3	4	2	1
	Z	Y	M	O	L	E
	A	B	C	D	F	G
	H	I	J	K	N	P
	Q	R	S	T	U	V
	W	X				

Klaaralfabet

Alhoewel nog geen letter van het cryptogram gelezen is, zit het werk er bijna op. Ontcijfering met behulp van het gevonden alfabet lijkt niet mogelijk zolang het klaaralfabet niet gevonden is, doch dat is maar schijn. Aan de top van de Beaufort-tabel wordt eenvoudigweg het gewone standaardalfabet gezet. Daardoor wordt niet de werkelijke klare letter gevonden bij de ontcijfering, maar een monoalfabetische substituent daarvan. Na ontcijferen met het standaardalfabet moet dus alleen nog een monoalfabetische substitutie opgelost te worden. Gegeven de lengte van het cryptogram is dat een fluitje van een cent. Het klaaralfabet blijkt eveneens gebaseerd op een kolomtranspositie, namelijk het woord MOLE voorafgegaan door de letters ZY die ter complicatie daar zijn neergezet. De oplossing luidt als volgt, links zoals deze uit het cryptogram komt en rechts wat gefatsoeneerd.

movement of three hundred second field artillery parenthesis one hundred fifty five mm howitzer parenthesis to positions reserved for corps artillery in field order number four of third corps comma will be via road junctions six zero four dash d parentheses three five five point four dash seven five two point three parenthesis and road junction five twenty two dash f semicolon and via road junction six three two dash a parenthesis three fifty six point two dash seven fifty two point five parenthesis and road junction five nine two dash b stop heads of columns enter twelfth division sector at road junction six zero two dash d and crossroad five three three dash a about one am march ten stop corps commander desires that your division make necessary arrangements to facilitate movement in twelfth division sector

Movement of 302 field artillery (155mm howitzer) to positions reserved for corps artillery in field order #4 of 3rd corps, will be via road junctions 604-D (355.4-752.3) and road junction 522-F; and via road junction 632-A (356.2-752.5) and road junction 592-B. Heads of columns enter 12th division sector at road junction 602-D and crossroad 533-A about 1:00am March 10. Corps commander desires that your division make necessary arrangements to facilitate movement in 12th division sector.

6.8 Referenties

- [1] H. F. Gaines (1956). *Cryptanalysis, A Study of Ciphers and their Solution*. Dover Publications, Inc., New York.
 Anon. (1942). *A Course in Cryptanalysis*, volume I and II. Aegean Park Press.
 H. Beker and F. Piper (1982). *Cipher Systems, The Protection of Communications*. Northwood Books, London.
- [2] R. W. Clark (1977). *The Man Who Broke Purple, The Life of the World's Greatest Cryptologist Colonel William F. Friedman*. Frome, London: Butler & Tanner Ltd.
 W. F. Friedman (1922). The Index of Coincidence and Its Applications in Cryptography. Technical Report Geneva, Ill.: Riverbank Laboratories, Dept. of Ciphers.
- [3] S. K. Kullback (1976). *Statistical Methods in Cryptanalysis*. Aegean Park Press.
- [4] A. G. Konheim (1981). *Cryptography, A Primer*. John Wiley & Sons, Inc.
- [5] S. K. Kullback (1976). *Statistical Methods in Cryptanalysis*. Aegean Park Press.
- [6] B. Tilt (1984). On Kullback's χ -Tests for Matching and Non-Matching Multinomial Distributions. *Cryptologia*, VIII(2), 132–141.

- [7] H. F. Gaines (1956). *Cryptanalysis, A Study of Ciphers and their Solution*. Dover Publications, Inc., New York. Hoofdstuk X, The consonant-line short cut, a method for attacking difficult cases, een bijdrage van George C. Lamb.
- [8] R. Anderson (1984). Finding vowels in simple substitution ciphers by computer. *Cryptologia*, 8(4), 348–358.
- [9] S. K. Kullback (1976). *Statistical Methods in Cryptanalysis*. Aegean Park Press.
- [10] J. B. M. Guy (1991). Vowel identification: an old (but good) algorithm. *Cryptologia*, XV(3), 258–262.
- [11] G. T. Sassoon (1992). The application of Sukhotin's algorithm to certain non-english languages. *Cryptologia*, XVI(2), 165–173.
- [12] L. D. Callimahos and W. F. Friedman (1985). *Military Cryptanalytics*. Aegean Park Press. Part II
- [13] M. Barlow (1990). Historical and unsolved ciphers. *ACA, The Cryptogram*, MA, 5. Afkomstig uit de archieven van de Fort Niagara Historical Association.

7

Informatietheorie

7.1	Veiligheid	246
7.1.1	Cryptosysteem	246
7.2	Optimale veiligheid	248
7.2.1	Ontcijferkans	249
7.2.2	Regel van Bayes	252
7.3	Informatie	255
7.3.1	Entropie	257
7.3.2	Entropie Engelse taal	261
7.4	Uitwisseling van Informatie	262
7.4.1	Communicatie	267
7.4.2	Waarneming	269
7.5	Informatie in taal	272
7.5.1	Entropie Markov-keten	274
7.5.2	Entropie Caesar-substitutie	277
7.6	Informatie in cryptografie	279
7.6.1	Uniciteitsafstand	282
7.6.2	Voorbeelden uniciteitsafstand	286
7.6.3	Relativering Shannon-entropie	289
7.7	Informatieoverdracht	291
7.7.1	Binair symmetrisch kanaal	294
7.7.2	Kanaalentropie	295

7.8	Slaagkans cryptoanalyse	297
7.8.1	Kansverdeling oplossingen	299
7.8.2	Voorbeeld slaagkans	300
7.9	Referenties	302

7.1 Veiligheid

Bij het gebruiken van een geheimschrift moet steeds de vraag gesteld worden: Is het wel sterk genoeg om de informatie in het cryptogram verborgen te houden? In dit hoofdstuk wordt gepoogd die vraag te beantwoorden. Het zal daarbij blijken dat een altijd en overal geldig antwoord niet kan worden gegeven. Steeds zullen de omstandigheden bepalen wat in een gegeven situatie een adequaat antwoord is.

Begonnen wordt met het verkennen van hetgeen *sterk genoeg* en *veilig genoeg* betekenen. Sterk en veilig zijn betrekkelijke begrippen en moeten worden gezien tegen de achtergrond van de context waarin de cryptografie wordt gebruikt.

Het hoogst bereikbare in veiligheid is absolute veiligheid en het mooiste geheimschrift is natuurlijk het onbreekbare geheimschrift. Het begrip absolute veiligheid kan vanuit de kansrekening worden benaderd en afgeleid zal worden wat dit voor een geheimschrift betekent. Het onbreekbaar geheimschrift bestaat echt, maar is niet altijd goed toepasbaar. In verreweg de meeste gevallen geeft men daarom de voorkeur aan een geheimschrift dat praktisch bruikbaar is en neemt op de koop toe dat het in principe kan worden gebroken.

Met behulp van de *informatietheorie* kan worden bepaald wanneer een cryptogram in beginsel kan worden gebroken en of er kans is op een succesvolle cryptanalyse. Het sleutelbegrip hier is *uniciteitsafstand*. Deze uniceitsafstand kan in redelijke mate worden bepaald uitgaande van de hoeveelheid informatie die ligt besloten in taal en communicatie.

7.1.1 Cryptosysteem

Een geheimschrift dat gemakkelijk kan worden gebroken geeft niet veel veiligheid. Aan de andere kant is niet voor elk beveiligingsprobleem een supersterk geheimschrift nodig. Wanneer is een geheimschrift dan sterk genoeg? Die vraag kan alleen beantwoord worden wanneer alle relevante factoren worden afge-

wogen. Daartoe behoren het belang van de te beschermen informatie, de omstandigheden waarin deze wordt gegenereerd, bewaard en/of getransporteerd. Maar ook van belang is over welke mogelijkheden de tegenstander beschikt: tijd, menskracht, goedgepaste cryptoanalisten, computerkracht.

Het is verleidelijk om het geheimhouden van het gebruikte geheimschrift als een onderdeel van de beveiliging te beschouwen. En inderdaad, toegegeven moet worden dat onbekendheid met het gebezigde geheimschrift een formidabele hindernis kan zijn bij het volvoeren van een succesvolle cryptoanalyse. Maar al is het dan een hindernis, toch kan deze vaak worden overwonnen. Er zijn in de geschiedenis legio voorbeelden waarin een geheimschrift werd gebroken zonder voorkennis over de aard ervan en uitsluitend uit onderschepte cryptogrammen.

Het is vanzelfsprekend zaak om het de tegenstander niet makkelijker te maken dan nodig en daarin past het geheimhouden van het geheimschrift wel. Maar wanneer geheimhouding van de geheimschriftalgoritme cruciaal wordt voor de veiligheid, dan moet toch geconcludeerd worden dat zo'n geheimschrift ondeugdelijk is. Want uiteindelijk zullen de details van het geheimschrift altijd wel bekend raken. Of dit nu door verraad, een fatale vergissing of een geslaagde cryptoanalyse gebeurt, vroeg of laat beschikt de cryptoanalist van de tegenstander over de benodigde gegevens.

Als de details van het gebruikte geheimschrift bekend zijn, staat alleen de vercijfersleutel tussen het cryptogram en de inhoud. Het is dan in principe altijd mogelijk om alle vercijfersleutels te proberen. Of daarmee uitsluitsel wordt verkregen over de juistheid van een geprobeerde sleutel? Dat is nu juist de vraag die vanuit de informatietheorie moet worden beantwoord.

Aangenomen dat het mogelijk is om vast te stellen of een gekozen sleutel de juiste is, is het slechts een kwestie van tijd en geld om ze allemaal te proberen. Tijd, omdat het proberen van een sleutel een zekere hoeveelheid rekenwerk vergt en veel tijd wanneer er veel sleutels zijn. Geld, omdat apparatuur en cryptoanalisten nodig zijn om het werk te doen. Tussen deze factoren tijd en geld bestaat een zekere samenhang: in veel gevallen is het mogelijk om door extra investeringen in bijvoorbeeld geheugen de

benodigde verwerkingstijd te verminderen*.

Wat zijn hiervan nu de gevolgen voor de veiligheid van een cryptogram? Door de tijd voor het testen van één sleutel te vermenigvuldigen met het totaal aantal sleutels kan worden berekend hoe lang een bericht gemiddeld veilig is. Het geheimschrift moet zodanig gedimensioneerd zijn dat die tijd lang is ten opzichte van de levensduur van het geheim. Soms is dat een paar uur, maar er zijn ook geheimen die vele tientallen jaren moeten worden beschermd.

Uitproberen van alle mogelijke sleutels mag dan de meest tijdrovende manier zijn om een geheimschrift te breken, het is in principe altijd mogelijk. Daarom fungeert dit als een maatstaf waarmee andere methoden om een geheimschrift te kraken kunnen worden vergeleken. Men noemt een geheimschrift *gebroken* wanneer een betere methode is bedacht dan het proberen van alle sleutels. Hoe sterk de reductie in benodigde tijd en inspanning is, dat bepaalt in welke mate het breken van het geheimschrift een werkelijke bedreiging voor de geheimhouding vormt. Het kan immers altijd nog meevallen. Want zelfs de forse teruggang in de gemiddeld benodigde tijd van 100 miljard naar 1 miljoen jaar zou men nog een aanvaardbaar risico kunnen vinden.

7.2 Optimale veiligheid

Absoluut veilig, dat klinkt goed. Maar wanneer is eigenlijk een cryptografisch systeem absoluut veilig? Bestaat absolute veiligheid eigenlijk wel? Is het in de praktijk te realiseren of blijft het een theoretische mogelijkheid? Als absolute veiligheid niet haalbaar is, wat is dan de *optimale* veiligheid?

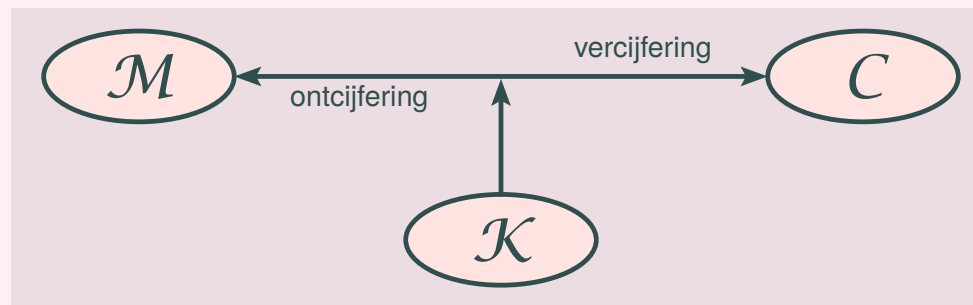
Voor het beantwoorden van deze vragen moet eerst een bruikbare definitie van optimale veiligheid opgesteld worden. Aan een definitie in de trant van "een cryptografisch systeem waarin de vercijferde berichten zonder de sleutel onontcijferbaar zijn" is niet veel eer te behalen. Want die helpt niet bij het zoeken naar

* Een bekend gegeven in de informatica. Zie als voorbeeld de time-memory tradeoff in het hoofdstuk over moderne systemen.

de voorwaarden voor het bereiken van de best mogelijke veiligheid. Om iets bruikbaar te kunnen zeggen wordt een andere richting ingeslagen, een die naar de kansrekening en de regel van Bayes voert*.

7.2.1 Ontcijferkans

Veronderstel dat alle mogelijke klaartekstberichten zijn bijeengebracht in de berichtenverzameling $\mathcal{M}$ van [figuur 7.1](#). De vercijfering van een klaartekst is een cryptogram uit een andere verzameling, namelijk die van de cryptogrammen $\mathcal{C}$. Die vercijfering geschiedt met behulp van een sleutel uit nog weer een derde verzameling $\mathcal{K}$ met daarin alle mogelijke sleutels.



Figuur 7.1 Verzamelingen berichten, cryptogrammen en sleutels

Het is niet nodig drukte te maken over het feit of berichten alleen meetellen als ze zogenaamd zinvol zijn. Alles wat met de gegeven verzameling symbolen (*het alfabet*) kan worden opgeschreven is als bericht toegestaan. Natuurlijk is een bericht in de trant van "kom morgen even langs" waarschijnlijker dan "gloep woebah hojo habakuk". Het verschil tussen die twee zit in de kans dat het gegeven bericht zal optreden. Noem P_1 de kans op "kom morgen even langs" en P_2 de kans op "gloep woebah hojo habakuk". Het zal niemand verbazen dat in de praktijk P_1 veel groter is dan P_2 .

* Rev. Thomas Bayes, 1702–1761, Engels predikant en wiskundige. De eerste die de waarschijnlijkheidsrekening toepaste op de mogelijke oorzaken in plaats van op de mogelijke gevolgen van een waargenomen gebeurtenis.

Het is dus niet erg als bepaalde berichten uit $\mathcal{M}$ onwaarschijnlijk zijn: ze hebben gewoon een verwaarloosbare kans op voorkomen. Hiermee wordt een belangrijk punt aangeraakt. De kans dat een bepaald bericht überhaupt zal optreden is een *à priori kans*. Dat wil zeggen, een kans losstaand van eventuele voorafgaande gebeurtenissen. Naast de *à priori kans* is er ook een *à posteriori kans*. Die beschrijft de situatie waarin wel sprake is van een voorafgaande gebeurtenis. Kennis van die gebeurtenis kan de verdere gevolgtrekkingen beïnvloeden.

Een voorbeeld van een *à priori kans* is het trekken van een gekleurde bal uit een vaas met een goed gemengde inhoud waarin 25 rode, 25 gele en 50 witte ballen. De *à priori kans* op het trekken van een rode bal bedraagt 25%. Stel nu dat van de getrokken bal de kleur weliswaar niet precies bekend is, maar dat wel kan worden waargenomen dat het een primaire kleur is*. Door die kennis is er nu een kans van 50% dat het een rode bal is.

Zo dadelijk zal ook een *à posteriori kans* opduiken, maar eerst verder met het cryptografisch systeem. In [figuur 7.1](#) staat een verzameling $\mathcal{M}$ met allerlei berichten M_j . Met elk van deze berichten is een *à priori kans* op voorkomen $P(M_j)$ geassocieerd. Daarnaast is er een verzameling sleutels $\mathcal{K}$. Hierin bevinden zich alle K mogelijke sleutels van het cryptosysteem. Met elke sleutel K_j is ook weer een kans $P(K_j)$ op diens voorkomen geassocieerd.

Direct valt in te zien dat het verstandig is alle sleutels dezelfde kans op gebruik te geven. Anders zal de cryptoanalist eerst de meest waarschijnlijke sleutels proberen en misschien vlugger succes boeken. Zodoende krijgt de cryptoanalist een onverwacht voordeeltje. Een voordeel dat er niet is wanneer alle sleutels evenveel kans maken. In de praktijk komt het gebruik van ongelijkwaardige sleutels maar al te vaak voor. Bijvoorbeeld als alleen bestaande woorden als sleutel worden gekozen in plaats van willekeurige letterreeksen.

Alle mogelijke cryptogrammen worden bijeengebracht in de verzameling C . Die is dus het resultaat van

* De primaire kleuren zijn rood, geel en blauw.

de vercijfering van alle voorkomende klaarteksten met alle mogelijke sleutels. Net als bij de verzameling klaarteksten is ook de kans op het aantreffen van een bepaald cryptogram niet voor alle cryptogrammen gelijk. De grootte van deze kans volgt uit het effect van de vercijferoperatie op alle mogelijke combinaties van berichten en sleutels.

Is het nu zo dat bij ontcijfering elk cryptogram in C overeenkomt met één specifiek bericht uit $\mathcal{M}$ en omgekeerd? Nee, dat is in het algemeen niet zo. Het komt zelfs heel veel voor dat ontcijfering van een cryptogram met verschillende sleutels tot verschillende klaartekst leidt. Wel geldt – althans in verreweg de meeste cryptosystemen – dat na de vercijfering het oorspronkelijke bericht eenduidig terugkomt bij ontcijfering met de goede sleutel.

Het is mogelijk dat vercijfering van eenzelfde bericht met verschillende sleutels hetzelfde cryptogram oplevert. Ook kan het gebeuren dat verschillende berichten met bepaalde sleutels toch hetzelfde cryptogram als uitkomst geven. Een eenvoudig voorbeeld laat zien dat dit laatste niet zo exotisch is als het misschien lijkt. In de Caesar-substitutie wordt elke letter vervangen door een letter die een bepaald aantal plaatsen verder in het alfabet staat. Neem voor de ene sleutel de vervanging $A \rightarrow D$, $B \rightarrow E$ en voor de andere de vervanging $A \rightarrow X$, $B \rightarrow Y$. Vercijferd worden de klaarteksten TOM en ZUS. Het resultaat is in beide gevallen het cryptogram WRP, zoals gemakkelijk valt na te gaan.

De totale kans $P(C)$ op een gegeven cryptogram C is natuurlijk gelijk aan de som van de à priori kansen op alle klaartekstberichten M_i , waarvoor een sleutel K_j te vinden is die dat cryptogram C als resultaat heeft.*.

$$P(C) = \sum_i P(M_i) \quad \text{waarvoor} \quad \exists j : E(M_i, K_j) = C \quad (7.1)$$

De cryptoanalist is echter niet zozeer geïnteresseerd in de à priori kans op de klaartekst M_i , maar wel in de à posteriori kans $P(M|C)$ bij waarneming van het cryptogram. Dat wil zeggen in de *voorwaardelijke kans* kans op de werkelijke klaartekst gegeven het waargenomen cryptogram. In nog andere woorden:

* Steeds wordt hier aangenomen dat alle sleutels eenzelfde kans op voorkomen hebben.

dit is de waarschijnlijkheid dat een bepaalde klaartekst de oorspronkelijke klaartekst van het cryptogram in kwestie is.

7.2.2 Regel van Bayes

Tussen de à priori kans op een bericht en de à posteriori kans na waarneming van het cryptogram, bestaat een verband dat wordt uitgedrukt door de regel van *Bayes*. In woorden luidt dit verband als volgt. Neem de kans dat het cryptogram een bepaald gecijferd bericht is. Die kans kan worden berekend door het product te nemen van de à priori kans op het bericht met de à posteriori kans op het cryptogram in kwestie. Deel dit product vervolgens door de totale kans dat het bewuste cryptogram wordt aangetroffen en het quotiënt is de kans dat precies dit bericht behoort bij het waargenomen cryptogram. Preciezer geformuleerd

$$P(M|C) = \frac{P(M) P(C|M)}{P(C)} \quad (7.2)$$

Dankzij deze regel van Bayes kan nu iets worden gezegd over de voorwaarde waaronder van optimale veiligheid sprake kan zijn. *Van optimale veiligheid wordt gesproken wanneer de à priori kans op een bericht gelijk is aan diens à posteriori kans.* Dit kan ook als volgt worden geformuleerd. Wanneer bij alle gecijferingen de kans op het vinden van het juiste bericht niet verandert door waarnemen van het cryptogram, dan heeft de cryptoanalist niets aan die waarneming. Ondanks de onderschepping van het cryptogram is de kans op een bepaalde inhoud van het bericht niet groter geworden dan voordien al het geval was. Nog weer anders gezegd: waarneming van het cryptogram vermindert de onzekerheid omtrent de inhoud van het bericht niet.

Kijk naar bovenstaande uitdrukking en probeer te bedenken hoe kan worden voldaan aan de eis dat waarneming van het cryptogram niets toevoegt, namelijk de eis dat $P(M|C) = P(M)$. Het triviale geval waarin de kans op een bericht nul is, d.w.z. $P(M) = 0$, levert niets op omdat optimale veiligheid gewenst



Bayes

is voor *alle berichten* en niet alleen voor berichten die zelfs niet eens voorkomen. Aan de à priori kans op een bericht kan niet worden getornd, omdat die een vaststaand gegeven is. Blijft over dat voor alle berichten en cryptogrammen het quotiënt van de kans op het cryptogram bij gegeven klaartekst en de totale kans op juist dat cryptogram precies 1 moet zijn. Want dan is de kans op de juiste klaartekst ook precies gelijk aan de kans op het cryptogram.

$$\frac{P(C|M)}{P(C)} = 1 \quad \rightarrow \quad P(C|M) = P(C) \quad (7.3)$$

Met andere woorden, noodzakelijk en voldoende voor optimale veiligheid is dat de à posteriori kans $P(C|M)$ op het cryptogram onafhankelijk is van de aangeboden klaartekst. De som van de kansen op alle sleutels die een bericht overvoeren in een bepaald cryptogram, moet gelijk zijn aan eenzelfde som over de sleutels waarmee een ander bericht in hetzelfde cryptogram wordt overgevoerd. De keuze van het te vercijferen bericht heeft zodoende geen invloed op de kans waarmee cryptogrammen voorkomen.

$$\sum_p P(K_p) = \sum_q P(K_q) \quad \text{met} \quad E(M_i, K_p) = E(M_j, K_q) = C \quad (7.4)$$

Wat betekent dit nu? Bedenk dat in een cryptografisch systeem waarin elk cryptogram eenduidig kan worden ontcijferd, tenminste evenveel cryptogrammen moeten voorkomen als er berichten zijn. Immers, veronderstel er zijn minder cryptogrammen dan berichten. Dan zullen er berichten zijn die tot hetzelfde cryptogram vercijferd worden. Dat cryptogram is dan niet meer eenduidig te ontcijferen. Ergo, tegenspraak met het uitgangspunt dat alle cryptogrammen eenduidig te ontcijferen moeten zijn. Dus moet wel het aantal mogelijke cryptogrammen tenminste gelijk zijn aan het aantal berichten.

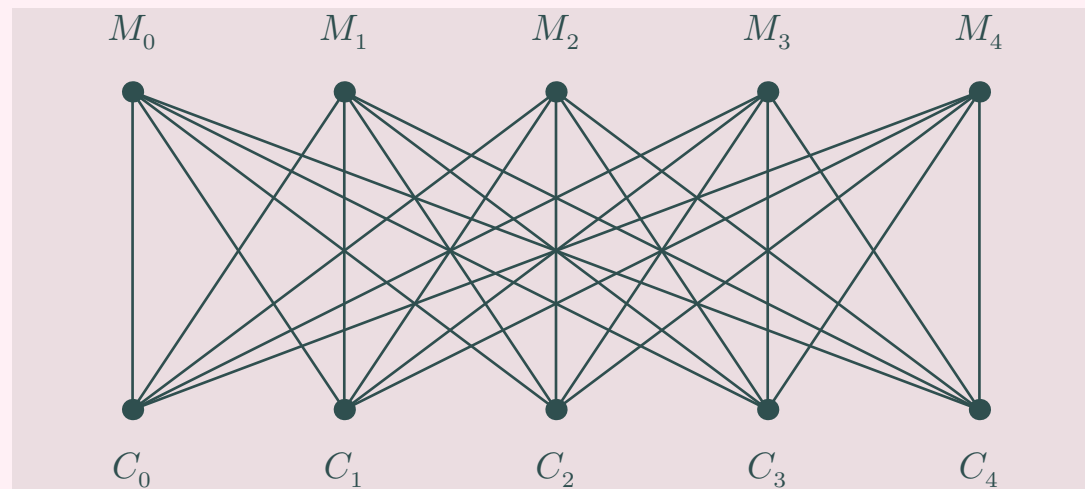
$$|\mathcal{C}| \geq |\mathcal{M}| \quad (7.5)$$

Na te hebben vastgesteld hoeveel cryptogrammen er tenminste moeten zijn, wordt vervolgens gekeken naar het aantal sleutels nodig voor een optimaal veilig systeem. Toepassing van de regel van Bayes leert dat voor alle voorkomende cryptogrammen en berichten moet gelden dat de à posteriori kans op een bepaald cryptogram gegeven de klaartekst, precies gelijk moet zijn aan de à priori kans op dat cryptogram.

$$P(C|M) = P(C) \text{ waarbij ook } P(C) > 0 \quad (7.6)$$

Dat kan alleen worden bereikt als elk cryptogram het resultaat kan zijn van elk bericht. Maar dan moet er voor elk klaartekstbericht en elk cryptogram tenminste één sleutel zijn die de klaartekst daarin omzet. Eén sleutel kan een bericht slechts in één cryptogram overvoeren en voor het maken van verschillende cryptogrammen uit één bericht zijn dus meerdere sleutels nodig. Daarom zijn minstens zoveel verschillende sleutels nodig als er cryptogrammen zijn. Omdat al aangetoond is dat er minstens zoveel cryptogrammen als berichten moeten zijn, volgt hieruit dat het aantal sleutels tenminste zo groot moet zijn als het aantal berichten én tenminste zo groot als het aantal cryptogrammen dat kan worden geproduceerd.

$$|\mathcal{K}| \geq |\mathcal{C}| \geq |\mathcal{M}| \quad (7.7)$$



Figuur 7.2 Optimaal veilig cryptosysteem

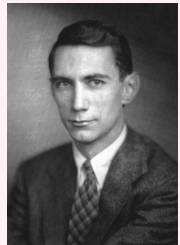
Voor optimale veiligheid zijn dus tenminste zoveel sleutels nodig als er berichten zijn. **Figuur 7.2** geeft een minimaal voorbeeld van een optimaal veilig systeem met precies evenveel sleutels als berichten. Er

zijn vijf berichten, vijf sleutels en vijf cryptogrammen. Elk bericht kan met goed gekozen sleutel worden vercijferd tot elk gewenst cryptogram en omgekeerd kan elk cryptogram tot elk bericht ontcijferd worden.

Gemakkelijk valt te controleren dat de voorwaardelijke kans $P(M|C)$ op het juiste bericht gelijk is aan $1:5 = 20\%$. Dus elk cryptogram heeft een kans van één op vijf om geproduceerd te worden, afhankelijk van de gekozen sleutel. Ook is te zien dat de kans $P(C)$ op elk cryptogram eveneens $1:5 = 20\%$ is. Want bij elk van de vijf berichten is er een sleutel die dat bepaalde cryptogram oplevert. Zodat is voldaan aan de voorwaarde $P(M|C) = P(C)$ voor optimale veiligheid.

Aan de hand van deze figuur valt te begrijpen hoe waarneming van een cryptogram niet méér over het oorspronkelijke bericht zegt dan dat het één van alle mogelijkheden is. Hetgeen tevoren ook al bekend was en dus niets nieuws oplevert. Het is het gevolg van het feit dat alle berichten tot alle cryptogrammen leiden en omgekeerd alle cryptogrammen alle berichten tot oorsprong hebben.

In de praktijk wordt lang niet altijd aan deze voorwaarde voldaan en is dit ook niet persé noodzakelijk. Een bekend geheimschrift als de Data Encryption Standard (DES) heeft slechts 2^{56} sleutels voor het vercijferen van 2^{64} berichten. Toch is dit geheimschrift lange tijd veilig genoeg geweest.



Shannon

7.3 Informatie

Informatie is alles *wat het weten waard is*. Informatie heeft echter niet genoeg aan zichzelf. Ze moet worden gecommuniceerd en daartoe wordt ze uitgesproken of neergeschreven en aldus toegankelijk gemaakt. In de cryptografie gaat het dan bovendien nog om gecontroleerde toegankelijkheid.

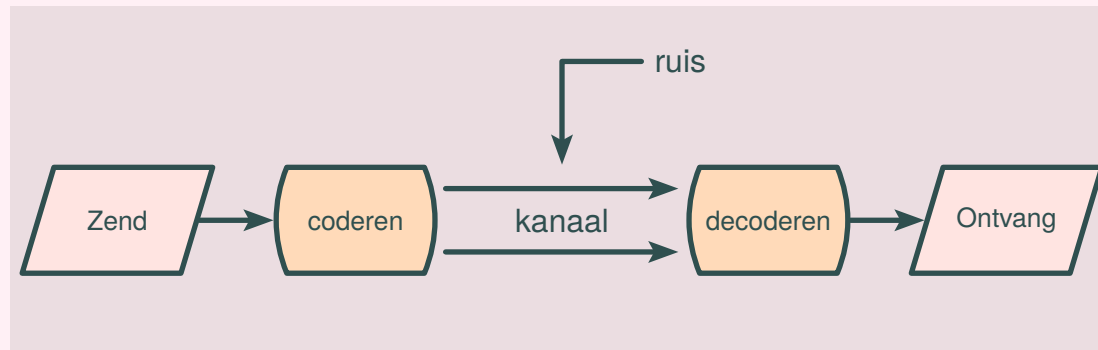
Het kwantificeren en in een wiskundige beschrijving vatten van informatie is de prestatie van Claude Shannon. Hij legde deze theorie neer in een aantal befaamde artikelen waarvan *Communication Theory of Secrecy Systems* uit 1949 het meest relevant is voor de cryptografie^[1].

Het belangrijkste aspect aan informatie is het overdragen ervan. Voor die overdracht is een medium nodig, dat *informatiekanaal* wordt genoemd (zie [figuur 7.3](#)). Het overdrachtsmedium kan van alles zijn: lucht die in trilling gebracht wordt, gemoduleerd licht, radiogolven, rooksignalen, inkt op papier, ga zo maar door. Ook het vercijferen van informatie door middel van geheimschrift behoort daarbij. Een geheimschrift kan worden opgevat als een informatiekanaal met de speciale eigenschap dat het door middel van een sleutel toegang geeft tot de informatie.

De belangrijkste probleemstellingen die optreden bij het overbrengen van informatie zijn

- hoe kan informatie volledig overgedragen worden ondanks optreden van storende ruis?
- hoe kan het meeluisteren door anderen worden verhinderd?
- hoe kan de informatie zo beknopt mogelijk worden vastgelegd?

Informatie wordt gegenereerd door een informatiebron en moet over een informatiekanaal worden overgebracht naar haar bestemming. Maar het is ook mogelijk dit om te keren en vanuit de bestemming via het informatiekanaal terug te redeneren naar de informatiebron.



Figuur 7.3 Informatieoverdracht

Het toegankelijk maken of vastleggen van informatie wordt bewerkstelligd door het bronmateriaal te *coderen*. Aan de ontvangkant wordt het gecodeerde materiaal door *decoderen* weer omgezet tot informatie.

Het coderen van informatie geschiedt met behulp van een *symboolverzameling* of *alfabet*.

$$s_1 \cdots s_n \in \mathcal{S} \quad (7.8)$$

Voorbeelden van symboolverzamelingen zijn het binaire alfabet 0,1 dat intern in computers gebruikt wordt, het Morse alfabet, het Latijnse alfabet A–Z, de ASCII-tekenen, de cijfers 0–9, enz. Het aantal symbolen van een verzameling S wordt gewoonlijk aangeduid met $|S|$ en heet de *cardinaliteit* van de verzameling. In veel klassieke cryptosystemen worden slechts de 26 letters A–Z gebruikt en dus is de cardinaliteit van die letterverzameling 26^* .

Een bericht M van lengte n kan worden geschreven als de rij symbolen

$$M = M^n = s_1 \dots s_n \quad \text{met} \quad s_i \in \mathcal{S} \quad (7.9)$$

7.3.1 Entropie

De symbolen van de symboolverzameling zijn de dragers van informatie. In het algemeen dragen de verschillende symbolen van het alfabet niet allemaal evenveel informatie. Het is nu de bedoeling om informatie te kwantificeren en daartoe wordt begonnen met het definiëren van de hoeveelheid informatie die in elk symbool begrepen is.

Wanneer het voorkomen van een symbool wordt opgevat als het optreden van een bepaalde gebeurtenis, dan kan verschil gemaakt worden tussen zeldzame en veel voorkomende gebeurtenissen. Men kan stellen dat een veel voorkomende gebeurtenis min of meer volgens verwachting is en daarom weinig informatie geeft. Een zeldzame gebeurtenis levert vanwege het onverwachte karakter juist veel informatie. Aan de andere kant zullen de symbolen met een lage informatiewaarde dankzij hun grote aantal toch samen weer

* In vroeger tijden telde het alfabet minder letters. De I en J, U en V werden niet onderscheiden. Een feit waarmee men bij de analyse van oude geheimschriften rekening moet houden.

veel informatiemassa opleveren. Terwijl zeldzame symbolen per stuk wel veel informatie geven, maar door hun geringe aantal samen toch weer betrekkelijk weinig.

De hoofdgedachte achter het kwantificeren van informatie is het kansbegrip: *hoe waarschijnlijker een gebeurtenis hoe lager de informatiewaarde en hoe onwaarschijnlijker een gebeurtenis hoe hoger de informatiewaarde*. De hoeveelheid informatie is zodoende omgekeerd evenredig met de kans op voorkomen. De informatiewaarde van een symbool s_i is in die redenering gekoppeld aan p_i de kans op het symbool. Door niet de kans zelf te nemen, maar 1 gedeeld door die kans bereikt men dat kleine kansen een grote informatiewaarde geven en grote kansen een kleine.

Kansen moeten niet worden opgeteld maar met elkaar vermenigvuldigd. Kijk bijvoorbeeld naar het opgooien van een munt. Het wordt kop of munt en als eerlijk gespeeld wordt is de kans op allebei precies gelijk. Eén keer opgooien en de kans op munt is $1/2$. Nu twee keer achtereen opgooien, wat is de kans dat het allebei munt zal worden? De eerste keer is de kans $1/2$ en als het munt, is het de tweede keer weer een kans $1/2$. En dus is de kans op twee keer achterelkaar munt $1/2 \times 1/2 = 1/4$.

In zekere zin is het jammer dat kansen met elkaar moeten worden vermenigvuldigd in plaats van opgeteld. Dat maakt niet alleen het rekenwerk ingewikkelder, maar het is ook gevoelsmatig vreemd. Informatie is nu eenmaal iets dat wij graag optellen. In het spraakgebruik zeggen we toch vaak genoeg "één en één is twee" om aan te geven dat we de feiten gecombineerd hebben? Dit wordt goedgemaakt door niet de kans zelf te gebruiken, maar de logaritme van de kans. De logaritme van een product is de som van de logaritmes van de factoren: $\log ab = \log a + \log b$. En nog een handigheidje: omdat kansen een waarde hebben die tussen 0 en 1 ligt, wordt een minteken voor de logaritme gezet waardoor de informatiewaarde een positief getal wordt. De informatiewaarde van symbool s_i dat met kans p_i wordt aangetroffen is dan

$$I(s_i) = \log \frac{1}{p_i} = -\log p_i \quad (7.10)$$

Men neemt 2 als basis voor de logaritme, waardoor de eenheid van informatie een *bit* wordt. Dat doet prettig aan in het computertijdperk. Een symbool dat niet voorkomt heeft kans $p = 0$ en dientengevolge

een informatiewaarde $0 \log 0$ die eveneens gelijk is aan 0. Zo'n symbool draagt zodoende niets bij tot de totale informatie. Een interpretatie die overeenkomt met hetgeen men zou verwachten. Elk symbool uit het alfabet draagt haar steentje bij tot de totale informatiewaarde van een tekst. De informatiewaarde van een complete tekst kan dus worden bepaald door de informatiewaarde van alle symbolen stuk voor stuk bij elkaar op te tellen. Ieder s_i symbool van S draagt $I(s_i)$ bij, zodat de totale informatie van een bericht M van lengte n wordt gegeven door

$$I(M) = \sum_{i=1}^n I(s_i) = - \sum_{i=1}^n \log_2 p_i \quad \text{met} \quad M = s_1 \dots s_n \quad (7.11)$$

De gemiddelde hoeveelheid informatie per symbool wordt gevonden door in een groot aantal teksten de aantallen symbolen te tellen. Hun relatieve voorkomens leveren een benadering voor de kans per symbool en daarmee een waarde voor de informatie van dat symbool. De gemiddeld gevonden informatie per symbool is karakteristiek voor zulke teksten wanneer maar genoeg representatieve tekst wordt genomen. Voor veel talen is de frequentiedistributie van de letters redelijk bekend en daarmee de informatiewaarde van elk symbool in de betreffende taal.

Door de gemiddelde hoeveelheid informatie van elk symbool uit het alfabet te wegen met het relatieve aantal voorkomens ervan en deze gewogen waarden over alle symbolen samen te tellen, wordt de *gemiddelde hoeveelheid informatie per symbool* berekend. Die gesommeerde, gewogen hoeveelheid informatie heet de *entropie* van de symboolverzameling. Entropie is de centrale grootheid in de navolgende behandeling. De gebruikelijke aanduiding voor entropie in de literatuur is de letter H . Uit de definitie volgt de formule voor de entropie van een symboolverzameling S met kansen p_i voor de afzonderlijke symbolen s_i

$$H(S) = - \sum_{i=1}^{|S|} p_i \log_2 p_i \text{ bits/symbool} \quad (7.12)$$

De grenzen waartussen de waarde van de entropie zich beweegt zijn eenvoudig aan te geven. Doordat kansen een waarde tussen 0 en 1 hebben, is de logaritme van een kans negatief of ten hoogste nul. Het

minteken in de **uitdrukking (7.10)** zorgt er dan voor dat alle informatiewaarden groter of gelijk nul zijn – negatieve informatie is er niet. Entropie H is de gewogen som van een aantal informatiewaarden en kan zodoende nooit kleiner dan nul zijn.

$$0 \leq p_i \leq 1 \rightarrow H_{\min}(S) = 0 \quad (7.13)$$

Hoewel het op het eerste gezicht misschien merkwaardig lijkt, heeft de entropie ook de waarde nul voor een symboolverzameling waarin slechts één van de symbolen voorkomt. Dan geldt immers $p_i = 1$ en $p_j = 0$ voor alle $j \neq i$ en bestaat de som uit één term $1 \log 1 = 0$ en verder allemaal termen $0 \log 0 = 0$. Waarom draagt dit ene symbool dan geen informatie? Het antwoord hierop luidt: informatie ontstaat juist door de aanwezigheid van verschillen en die zijn er niet wanneer slechts één enkel symbool voorkomt.

Men kan laten zien dat de entropie z'n hoogste waarde bereikt als alle symbolen even vaak voorkomen, d.w.z. allemaal dezelfde kans op voorkomen hebben. Die maximale waarde blijkt gelijk aan de logaritme van het aantal symbolen. Voor een alfabet van 26 letters is dit $\log_2 26 = 4,7$ bits per symbool. Het betekent dat in een 26-letter alfabet een symbool hoogstens 4,7 bits informatie kan dragen, twee symbolen samen hoogstens 9,4 bits, enz. In de praktijk is dit voor teksten minder, maar daarover dadelijk meer.

Voor het bepalen van de bovengrens aan de entropiewaarde wordt gebruik gemaakt van het feit dat de som van alle $p_i \log p_i$ maximaal is wanneer alle kansen aan elkaar gelijk zijn^[2]

$$p_i = \frac{1}{|S|} \quad \forall i \quad (7.14)$$

Zodat voor H dan volgt

$$H_{\max}(S) = - \sum_{i=1}^{|S|} \frac{1}{|S|} \log_2 \frac{1}{|S|} = \log_2 |S| \quad (7.15)$$

De entropie beweegt zich dus tussen de grenzen

$$0 \leq H(S) \leq \log_2 |S| \quad (7.16)$$

Entropie is in het voorgaande opgevat als de *informatiecapaciteit* van een symboolverzameling. Er is nog een andere interpretatie mogelijk, een interpretatie die verwant is aan de entropie in de thermodynamica (warmteleer)^[3]. Entropie is daar een maat voor het aantal realiseringsmogelijkheden van een systeem. Een hoge entropie betekent veel mogelijke realisaties en dat brengt ook veel onzekerheid omtrent de daadwerkelijk aan te treffen toestand met zich mee (één uit een veelheid van mogelijkheden). Bij lage entropie zijn er weinig mogelijkheden en is er bijgevolg minder onzekerheid omtrent de realiteit. Als de entropie de minimumwaarde nul heeft is maar één realisatie mogelijk, waarover dan ook geen enkele onzekerheid bestaat.

Dit laat zich vertalen naar symboolverzamelingen. Zijn de symbolen willekeurig verdeeld (random tekst) dan zijn er heel veel teksten mogelijk en bestaat dus grote onzekerheid over de juiste: dat is er één uit vele. Dit correspondeert met een hoge waarde voor de entropie. In het andere uiterste zijn er weinig alternatieve teksten en dus veel minder tot weinig onzekerheid over de juiste interpretatie. In die situatie is de entropie laag.

7.3.2 Entropie Engelse taal

Als voorbeeld wordt de informatiewaarde en de daaruit af te leiden entropie voor de 26 letters A–Z van de Engelse taal gegeven. In het Engels heeft de letter E de hoogste frequentie van ongeveer 13%. De bijbehorende informatiewaarde per letter E is met 2,944 bits dan ook het laagst. Maar omdat de E zo vaak voorkomt draagt deze letter met 0,383 bits/letter gemiddeld toch nog het meeste bij tot de totale informatiemassa. De Z komt met 1% het minst voor en heeft van alle letters daarom de hoogste informatiewaarde. Maar zelfs samen hebben alle Z's in een tekst nog weinig in de melk te brokkelen (zie [tabel 7.1](#)).

	p	$-\log_2 p$	$-p \log_2 p$		p	$-\log_2 p$	$-p \log_2 p$
A	0,074	3.763	0,277	N	0,080	3.653	0,290
B	0,010	6.682	0,065	O	0,075	3.732	0,281
C	0,031	5.027	0,154	P	0,027	5.227	0,140
D	0,042	4.558	0,193	Q	0,004	8.158	0,029
E	0,130	2.944	0,383	R	0,076	3.722	0,282
F	0,028	5.142	0,146	S	0,061	4.031	0,247
G	0,016	5.932	0,097	T	0,092	3.444	0,316
H	0,034	4.883	0,165	U	0,026	5.265	0,137
I	0,074	3.766	0,277	V	0,015	6.028	0,092
J	0,002	9.252	0,015	W	0,016	6.002	0,094
K	0,003	8.400	0,025	X	0,005	7.758	0,036
L	0,036	4.799	0,174	Y	0,019	5.692	0,110
M	0,025	5.337	0,132	Z	0,001	9.995	0,010
Totale gemiddelde entropie = 4,167 bits/symbool							

Tabel 7.1 Entropie van letters in het Engels

De hoogst mogelijke entropie wordt bereikt in het geval van een willekeurige verdeling waarin alle 26 letters even vaak voorkomen. De verdeling van de letterfrequenties in het Engels is echter veel schever en daarom zal de entropie van het Engels lager uitvallen dan het maximum. Dat klopt ook bij narekenen. De som van de bedragen in de kolom $-p \log_2 p$ van [tabel 7.1](#) is met 4,167 lager dan de maximale 4,7 bits per letter. In het vervolg zal blijken dat de werkelijke entropie van de Engelse taal nog aanzienlijk lager ligt. Dat komt omdat tekst niet uit afzonderlijke letters bestaat, maar uit letters die een grote onderlinge samenhang vertonen.

7.4 Uitwisseling van Informatie

Taal bestaat niet uit losse letters, maar deze hangen onderling samen. Het gevolg hiervan is dat voor

de berekening van de informatiewaarde van een tekst de letters niet los van elkaar mogen worden bezien, maar alleen in hun onderling verband. Vandaar dat eerst moet worden nagaan wat samenhang in gebeurtenissen betekent voor de informatiewaarde en dus voor de entropie.

Als twee gebeurtenissen met elkaar samenhangen is er een zekere kans dat een bepaalde gebeurtenis volgt op een andere. De ene gebeurtenis kan worden opgevat als oorzaak en de andere als gevolg. Een voorbeeld kan dit duidelijk maken. Als de ene auto de andere geen voorrang geeft is dat een gebeurtenis met mogelijke gevolgen. Er is een kans dat het goed afloopt, maar er is ook kans op een botsing met blikshade tot gevolg. En zelfs bestaat de kans dat slachtoffers vallen. Al deze gebeurtenissen kunnen het gevolg zijn van de inleidende gebeurtenis: het geen voorrang verlenen. Met elk mogelijk gevolg kan een kans op voorkomen geassocieerd worden. A priori bestaat er onzekerheid over de afloop. Nemen we ook andere verkeersproblemen mee, zoals rijden door rood licht, te hard rijden, enz. dan kan dit gehele systeem van oorzaak en gevolg in het verkeer aan een entropiebeschouwing worden onderworpen. De entropie waar het hier in het bijzonder om draait is de entropie die de gevolgen van een bepaalde gebeurtenis verdisconteert.

De zaak kan ook worden omgedraaid. Waarneming van een gevolg maakt speculeren over de oorzaak mogelijk. Wie twee beschadigde autos midden op een kruispunt met verkeerslichten ziet staan, zal al snel concluderen dat waarschijnlijk een van de twee door rood is gereden.

De relevantie voor de cryptografie is evident. Wie een geheim aan een geheimschrift toevertrouwt is een bron van informatie en moet rekening houden met hetgeen daarvan kan uitlekken. En omgekeerd neemt de cryptoanalist het onderschepte cryptogram ter hand en vraagt zich af wat dit leert over de klaartekst en de gebruikte sleutel. In beide gevallen hebben we te maken met oorzaak en gevolg en de informatiestroom die daarmee gepaard gaat.

In een systeem met twee onafhankelijke gebeurtenissen kan met elk van beide een kans op voorkomen geassocieerd worden. Maar indien ze met elkaar samenhangen als oorzaak en gevolg wordt dit anders.

De kans het vervolg aan te treffen hangt nu af van twee afzonderlijke kansen: (1) de kans dat de gegeven oorzaak überhaupt optreedt, en (2) de kans dat deze oorzaak juist dat specifieke gevolg heeft. De kans op de vervolggebeurtenis is het product van deze twee kansen. De afhankelijke kans voor het optreden van het vervolg heet een *voorwaardelijke kans*. De kans op beide gebeurtenissen – oorzaak en gevolg – tezamen optredend heet een *simultane kans*.

Al deze kansen kunnen worden gebruikt om de bijbehorende waarde van de entropie van het systeem te berekenen. De simultane kansen geven de totale entropie (informatie) van het systeem. Maar ook met de voorwaardelijke kansen is een entropie te berekenen en die entropie kwantificeert specifiek de hoeveelheid informatie betrokken bij het proces van oorzaak en gevolg. Deze laatste vorm van entropie heeft een aparte naam gekregen en wordt de *equivocatie* (equivocation in het Engels) genoemd. Een Nederlandse term is ook wel de *twijfel*. Dat laatste valt te begrijpen door te bedenken dat een hoge entropie veel onzekerheid representeert. Een hoge waarde voor de equivocatie laat dus veel mogelijke gevolgen open en schept aldus grote twijfel aangaande het vervolg.

Deze kwalitatieve beschouwing zal nu kwantitatief worden uitgewerkt. Noem de kans op de afzonderlijke gebeurtenissen i en j respectievelijk p_i en p_j . Voor twee samenhangende gebeurtenissen is p_{ij} de kans op de simultane gebeurtenis (i, j) . Hetgeen wil zeggen: het is de kans dat i optreedt en wordt gevolgd door j , of omgekeerd geredeneerd: j is het gevolg gegeven het optreden van i . I is op te vatten als een bron van gebeurtenissen (informatie) en J als een waarnemer. De bron genereert gebeurtenissen waarop de waarnemer met een zekere waarschijnlijkheid reageert.

Om de *simultane kans* p_{ij} te kunnen benaderen wordt de *voorwaardelijke kans* $p_{j|i}$ geïntroduceerd. Dit is de kans dat het optreden van i zal worden gevolgd door een optreden van j . Impliciet wordt hierbij verondersteld, dat gebeurtenis i op meerdere wijze een vervolg kan hebben; elk mogelijk vervolg heeft een zekere kans. Als altijd sprake is van enig vervolg geldt dat de som van alle $p_{j|i}$ gelijk is aan 1. Een uitdrukking die als een behoudswet kan worden opgevat.

De kans op de simultane gebeurtenis (i, j) wordt gegeven door de kans dat i zal optreden maal de kans dat i daadwerkelijk wordt gevolgd door j

$$p_{ij} = p_i \cdot p_{j|i} \quad (7.17)$$

De vraag is nu wat de entropie $H(I, J)$ van zo'n systeem met simultane gebeurtenissen zal zijn. De waarde van $H(I, J)$ wordt berekend door het invullen van de juiste kansen in de uitdrukking voor de entropie en het sommeren daarvan over alle mogelijkheden. De som zal dus dienen te lopen over alle paren (i, j)

$$H(I, J) = - \sum_{i,j} p_{ij} \log p_{ij} \quad (7.18)$$

Door invullen van p_{ij} en uitwerken met

$$\log p_{ij} = \log p_i p_{j|i} = \log p_i + \log p_{j|i}$$

komt er

$$\begin{aligned} H(I, J) &= - \sum_i \sum_j p_i p_{j|i} \log p_i p_{j|i} \\ &= - \sum_i p_i \log p_i \sum_j p_{j|i} - \sum_i p_i \sum_j p_{j|i} \log p_{j|i} \\ &= - \sum_i p_i \log p_i - \sum_i p_i \sum_j p_{j|i} \log p_{j|i} \end{aligned} \quad (7.19)$$

De eerste term is juist de entropie $H(I)$ van I en de tweede krijgt de naam $H(J|I)$

$$H(J|I) = - \sum_i p_i \sum_j p_{j|i} \log p_{j|i} \quad (7.20)$$

Er resulteert

$$H(I, J) = H(I) + H(J|I) \quad (7.21)$$

De uitdrukking $H(J|I)$ is de *equivocatie* en representeert de afhankelijkheid tussen de ensembles I en J . Hierin staan I en J voor de verzameling van respectievelijk alle mogelijke afzonderlijke gebeurtenissen i en j .

Het bovenstaand resultaat heeft de volgende interpretatie. Het complete systeem bestaat uit twee deelsystemen die een onderlinge afhankelijkheid vertonen. Die afhankelijkheid is niet totaal omdat uit het optreden van i niet met volledige zekerheid tot het optreden van j geconcludeerd kan worden. Daaruit volgt dat de informatie van het totale systeem moet worden benaderd als $H(I)$, de informatie van het bronstelsel I , waaraan wordt toegevoegd een component $H(J|I)$ die het gevolg is van de onzekerheid omtrent de precieze reactie j van de waarnemer.

Het is nuttig om nog even naar de grensgevallen te kijken. Wanneer i en j onafhankelijk van elkaar zijn, geldt natuurlijk dat p_{ij} gelijk is aan het product van p_i en p_j . Zodat dan $p_{j|i} = p_j$ en uitwerking van het equivocatie-deel laat zien dat

$$H(J|I) = - \underbrace{\sum_i p_i}_1 \underbrace{\sum_j p_j \log p_j}_{H(J)} \quad (7.22)$$

en dus

$$i \text{ en } j \text{ onafhankelijk} \rightarrow H(I, J) = H(I) + H(J) \quad (7.23)$$

Het complete systeem gedraagt zich onder deze conditie als twee volledig onafhankelijke subsystemen en de totale entropie is gelijk aan de som van de entropieën van de subsystemen. De hoeveelheid informatie in het totale systeem is nu maximaal.

In het andere uiterste zijn i en j zo vast aan elkaar gekoppeld dat op i altijd j volgt. Dan is $p_{ij} = p_i$ en $p_{j|i} = 1$, waardoor $H(J|I) = 0$. Nu is

$$i \text{ en } j \text{ afhankelijk} \rightarrow H(I, J) = H(I) \quad (7.24)$$

Het systeem wordt in dit geval volledig gedomineerd door gebeurtenissen in I en de gebeurtenissen in J volgen daaruit automatisch. Met de entropie $H(I)$ is zodoende ook alle informatie gegeven, subsysteem J kan daaraan niets meer toevoegen.

Omdat $H(J|I)$ in waarde hoogstens gelijk is aan $H(J)$ (en zoals elke entropie minstens nul) geldt

$$0 \leq H(J|I) \leq H(J) \quad (7.25)$$

In combinatie met [vergelijking \(7.21\)](#) betekent dit dat een oorzakelijk verband tussen gebeurtenissen – waardoor $H(J|I) < H(J)$ – altijd zal leiden tot vermindering van de hoeveelheid onzekerheid in het totale systeem.

Met het gegeven model van simultane gebeurtenissen zullen de twee voor toepassing in de cryptografie belangrijkste issues bekeken worden. Eerst wordt nagegaan wat ze betekenen voor communicatie, dat wil zeggen voor een gewilde overdracht van informatie. Vervolgens wisselt het standpunt, zodat niet de bron als uitgangspunt wordt genomen maar de bestemming. Anders gezegd, de bestemming fungeert in dat geval als een waarnemer die uit de overgekomen informatie gevolgtrekkingen maakt omtrent de gebeurtenissen aan de bron.

7.4.1 Communicatie

De in [figuur 7.3](#) getekende schets van het informatiekanaal bevat een component *ruis* die de kwaliteit van de informatieoverdracht beïnvloedt. Zonder de aanwezigheid van ruis is sprake van *perfecte communicatie* en is er een situatie waarin de waarneming volstrekt samenhangt met de gebeurtenis aan de bron.

$$p_{j|i} = \begin{cases} 1 & \text{als } j = i \\ 0 & \text{als } j \neq i \end{cases} \rightarrow \begin{cases} H(J|I) = 0 \\ H(I, J) = H(I) \end{cases} \quad (7.26)$$

De equivocatie $H(J|I)$ heeft in dit geval de waarde nul. Dat wil zeggen, er bestaat geen enkele onzekerheid met betrekking tot de overgedragen informatie. Aan beide kanten van het informatiekanaal is precies

dezelfde hoeveelheid informatie (entropie) $H(I) = H(J)$ voorhanden. De informatie uitgezonden door de bron wordt onaangetast overgebracht naar de bestemming. Bij *imperfecte communicatie* is de equivocatie $H(J|I)$ wel groter dan nul.

$$0 < p_{j|i} \leq 1 \rightarrow H(I) < H(I) + H(J|I) \leq H(I) + H(J) \quad (7.27)$$

Het zorgt ervoor dat daardoor de totale hoeveelheid entropie in gehele systeem met het bedrag $H(I, J)$ groter wordt en stijgt van $H(I)$ naar $H(I) + H(J|I)$, hetgeen duidt op meer onzekerheid. Want – het kan niet vaak genoeg benadrukt worden – hoe hoger de waarde van een entropie, hoe meer realiseringmogelijkheden er in verdisconteerd zitten en dus hoe meer onzekerheid bestaat over welke daarvan actueel gerealiseerd wordt. De equivocatie representeert de hoeveelheid informatie die door de imperfecte overdracht wordt opgeslokt.

In de communicatietechniek is het doel een volledig correcte overdracht van het signaal (perfecte communicatie). De aanwezigheid van ruis en andere storingen verhindert dit. Ter compensatie moet daarom redundantie worden toegevoegd. Hoeveel extra nodig is vertelt de entropieberekening. Namelijk precies de hoeveelheid equivocatie $H(J|I)$ die nodig is om voor de ruis te compenseren. Wanneer de bron de uitgezonden $H(I)$ aan informatie met juist dat bedrag $H(J|I)$ verhoogt, komt er precies genoeg door het kanaal om de bedoelde informatie over te houden.

De toevoeging van redundantie kan geschieden door het aanvullen van de berichten met extra symbolen die afhankelijk zijn van hetgeen er al staat. Storingen en ruis zorgen er immers voor dat de uitgezonden symbolen in meerdere of mindere mate niet goed worden ontvangen. De toegevoegde symbolen zorgen door hun overvloedigheid voor vermindering van de onzekerheid. Ze geven namelijk dankzij de onderlinge afhankelijkheid de gelegenheid om het oorspronkelijke signaal door berekening te herstellen. Dit is de reden dat foutherstellende codes meer bits gebruiken dan strikt noodzakelijk is voor het vastleggen van de informatie.

De equivocatie $H(J|I)$ is dus de hoeveelheid informatie die moet worden toegevoegd om aan de ontvanger

de kant van de communicatie precies zoveel informatie over te houden als aan de bron werd gegenereerd. Deze hoeveelheid informatie is blijkbaar in het informatiekanaal door ruis verdwenen en daarom heet de equivocatie in de Engelse literatuur ook wel de noise entropy.

Gaat het in de communicatietechniek om een goede signaaloverdracht en dus om een minimale equivocatie, in de cryptografie daarentegen is de ideale situatie nu juist het maximaliseren van de equivocatie. Immers als men er in slaagt om de onzekerheid zover op te blazen dat de equivocatie $H(J|I)$ gelijk wordt aan de entropie $H(I)$ van de bron, dan is een situatie bereikt waarin de gebeurtenissen i en j onafhankelijk van elkaar lijken op te treden. Anders gezegd, dan is een situatie bereikt waarin het waarnemen van een cryptogram geen enkele informatie geeft over de oorspronkelijke klaartekst. Klaartekst en cryptogram hebben ogenschijnlijk niets met elkaar van doen.

Voor het ontcijferen van een cryptogram is de sleutel nodig. Wie daarover beschikt verkeert juist weer wel in de situatie van perfecte communicatie. Elke waargenomen gebeurtenis j valt in dat geval eenduidig te herleiden tot een specifieke gebeurtenis i aan de bron vanwege $p_{j|i} = 1$ voor $j = i$ en $p_{j|i} = 0$ voor $j \neq i$.

7.4.2 Waarneming

Hiervoor was communicatie het gezichtspunt: het overbrengen van informatie van bron naar bestemming over een informatiekanaal. Door een wisseling van standpunt wordt de waarneming de bekende en de bron de onbekende grootte. Deze waarneming heeft over het algemeen plaats over een imperfect informatiekanaal. Dit gezichtspunt legt de verbinding naar de cryptanalyse, waar het cryptosysteem fungeert als de ruis op het informatiekanaal. Het sleutelbegrip is hier de *wederzijdse informatie*. Die wordt gedefinieerd als de informatie (entropie) verkregen door waarneming minus de equivocatie gegenereerd door het informatiekanaal.

$$W(I, J) = H(J) - H(J|I) \quad (7.28)$$

Hoe groter de equivocatie hoe minder informatie er voor de waarnemer overblijft. De wederzijdse informatie $W(I, J)$ representeert de mate waarin de ontvangen informatie J (nog) iets zegt over de bron I . Ze kan omgekeerd ook worden opgevat als de hoeveelheid informatie die van de bron naar de waarnemer getransporteerd wordt, rekening houdend met het informatieverlies veroorzaakt door het transportkanaal. Aan de kant van de waarnemer is de hoeveelheid informatie $H(J)$ de entropie van de waargenomen symbolen. Is er perfecte communicatie dan is aan beide zijden van het informatiekanaal dezelfde informatie aanwezig. Al eerder bleek dat in dat geval sprake is van perfecte communicatie met $H(I) = H(J)$, omdat $H(J|I) = 0$, zie [vergelijking \(7.24\)](#). De informatie bij bron en waarnemer stemmen dan met elkaar overeen. Dit geval is aan de orde wanneer klaartekst over een perfect informatiekanaal wordt gezonden of wanneer een cryptogram met de juiste sleutel wordt ontcijferd.

$$p_{j|i} = 0/1 \rightarrow H(J|I) = 0 \rightarrow W(I, J) = H(J) = H(I) \quad (7.29)$$

Komt er storing in de informatiedracht dan wordt de relatie tussen de uitgezonden informatie I en de ontvangen informatie J minder sterk. Storingen geven stukjes informatie die niet direct meer in verband staan met de bron. Bij storing mag worden gedacht aan ruis en stoorzenders, maar ook aan het vercijferen met een geheimschrift. Voor een waarnemer die niet over de juiste berichtssleutel beschikt werkt het geheimschrift als een storing die het verband met de originele tekst lijkt te verbreken.

Als gevolg van de storing neemt de equivocatie toe en bijgevolg de wederzijdse informatie af, want nu geldt voor verschillende combinaties i en j dat $p_{j|i} \neq 0$. De afname aan zekerheid bedraagt precies zoveel als de equivocatie van het informatiekanaal, een bedrag $H(J|I)$. Bij maximale storing blijft er geen afhankelijkheid over en is de wederzijdse informatie tot nul gereduceerd. In de cryptografie wordt dit gerealiseerd met een onbreekbaar cryptosysteem. De waarde van de equivocatie $H(J|I)$ is in dat geval gelijk aan de informatiewaarde van de ontvangen symbolen (wat die dan ook nog waard mogen zijn). De extra informatie nodig om iets over de bron te kunnen zeggen ontbreekt in dat geval geheel.

$$\left. \begin{aligned} H(J|I) = - \sum_i p_i \sum_j p_{j|i} \log p_{j|i} \\ p_{j|i} = p_j \end{aligned} \right\} \rightarrow H(J|I) = H(J) \quad (7.30)$$

Het is nu duidelijk waarom de wederzijdse informatie tussen J en I is gedefinieerd als in [vergelijking \(7.28\)](#). Men kan zeggen dat $H(J|I)$ de onzekerheid is over de informatiebron I bij waarnemer J .

De cryptoanalist die een geheimschrift wil breken probeert uit de imperfecte waarneming (het cryptogram) de bron (de klaartekst) te reconstrueren. Om de cryptoanalyse te doen slagen is het nodig dat een zekere hoeveelheid informatie over de bron is overgebleven. In termen van de wederzijdse informatie betekent dit dat de equivocatie klein genoeg moet zijn om voldoende van de broninformatie over te houden.

$$p_{j|i} \neq 0 \rightarrow 0 < H(J|I) < H(J) \rightarrow 0 < W(I, J) < H(I) \quad (7.31)$$

Als een onbreekbaar geheimschrift is gebruikt, zal geen enkele informatie door het informatiekanaal gekomen zijn. De cryptoanalist heeft dan niets om mee te werken. Een onbreekbaar geheimschrift laat geen wederzijdse informatie over.

$$p_{j|i} = p_j \rightarrow H(J|I) = H(J) \rightarrow W(I, J) = 0 \quad (7.32)$$

Eenvoudig valt ook in te zien dat de hoeveelheid informatie in een systeem niet afhankelijk kan zijn van het ingenomen standpunt. Of nu naar de communicatiekant dan wel de waarnemingskant wordt gekeken maakt voor het systeem op zich geen verschil. Er zal daarom geen verschil mogen bestaan tussen $H(I, J)$ en $H(J, I)$ en dus

$$H(I, J) = H(I) + H(J|I) = H(J) + H(I|J) \quad (7.33)$$

ofwel I plus wat I over J zegt is gelijk aan J plus hetgeen J over I zegt. Het is eveneens duidelijk (en niet moeilijk aan te tonen) dat $W(I, J) = W(J, I)$.

Tot slot nog een andere uitdrukking voor $W(I, J)$ die volgt uit combinatie van de uitdrukkingen voor $H(I, J)$ en $W(I, J)$

$$H(I, J) = H(I) + H(J|I) \quad \text{vergelijking (7.21)}$$

$$W(I, J) = H(J) - H(J|I) \quad \text{vergelijking (7.28)}$$

Deze geven door eliminatie van $H(J|I)$ de uitdrukking

$$W(I, J) = H(I) + H(J) - H(I, J) \quad (7.34)$$

7.5 Informatie in taal

Natuurlijke taal is geen willekeurige aaneenschakeling van letters. Niet alle lettercombinaties zijn even waarschijnlijk en niet alle letters volgen even vaak op elkaar. Een opvallend voorbeeld is de letter Q die uitsluitend gevolgd wordt door de letter U, waarachter dan weer een klinker komt. Ook bedraagt het aantal achtereenvolgende medeklinkers zelden meer dan drie, hooguit vier. Klinkers vertonen – afgezien van enkele vaste klankcombinaties – weinig affiniteit met elkaar. Klinkers en medeklinkers wisselen elkaar voortdurend af.

Dit alles laat zich samenvatten in de conclusie dat de combinaties tussen de letters in taal een bepaalde structuur vertonen. Die structuur kan in de cryptanalyse worden geëxploiteerd. Taal modelmatig benaderen maakt het mogelijk om deze structuur te kwantificeren. Een model voor de opeenvolging van letters bestaat daaruit dat elke volgende letter wordt gekoppeld aan de reeds verschenen letters. Het optreden van een letter wordt opgevat als een gebeurtenis, een rij letters als een keten van gebeurtenissen. Langs deze weg kan, met gebruik van de entropie-beschouwingen uit het voorgaande, een berekening worden opgezet voor de informatiewaarde van taal.

Zo'n keten van gebeurtenissen, waarbij opvolgende gebeurtenissen worden beïnvloed door hun voorgangers, heet een *Markov-keten*. Met behulp van het Markov-model voor taal kan worden vastgelegd dat een tekst niet op willekeurige wijze is samengesteld, maar dat het aantal mogelijkheden beperkt is.

Verder is het nuttig om te kunnen bepalen of een cryptoanalyse met succes kan worden uitgevoerd. Met het oog daarop wordt uit dit Markov-model een criterium afgeleid waarmee bepaald kan worden of de juiste sleutel en/of ontcijfering eigenlijk wel kan worden gevonden: de *uniciteitsafstand* (unicity distance in het Engels). Deze afstand geeft de minimale lengte die een cryptogram moet hebben om te kunnen concluderen tot de juiste ontcijfering. Omgekeerd kan langs die weg ook worden gewerkt aan het versterken van een geheimschrift. Als cryptogrammen korter dan deze uniciteitsafstand veilig zijn voor cryptoanalyse, dan loont het de moeite om de uniciteitsafstand zo groot mogelijk te maken.

Dat er zoiets als een uniciteitsafstand moet zijn, kan met de eenvoudige monoalfabetische substitutie als voorbeeld aannemelijk gemaakt worden. Als sleutel is gekozen*

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ  
YXWVRQLJFSPZUBNOKGITHCAMED
```

Enkele vertalingen

```
KAT = PYT  
BAKER = XYPRG  
LEPEL = ZRORZ
```

Nu wordt de sleutel weggenomen en gekeken naar de mogelijke ontcijferingen

```
PYT = (veel mogelijkheden)  
XYPRG = (veel mogelijkheden)  
ZRORZ = kajak, lepel, nemen, radar, reder, rotor
```

De vraag om vast te stellen welke ontcijfering de juiste is, hangt af van factoren als de lengte van het woord en het letterpatroon. Voor het derde woord uit het laatste voorbeeld is dat veel kansrijker dan voor

* De sleutel voor deze substitutie is gemaakt met sleutelzin DE MACHTIGE KONING BUIGT ZICH DIEP IN HET STOF achterstevoren opgeschreven.

de eerste twee. Hier volgt een extreem voorbeeld, een lang woord met een zo specifiek patroon dat er eigenlijk maar één kandidaat voor ontcijfering is.

UYIIYUNNGVRBYYG = massamoordenaar

De nu volgende behandeling van Markov-ketens zal daarom uitmonden in een discussie over de vraag: hoe lang een fragment moet zijn om eenduidig tot de juiste ontcijfering te kunnen besluiten. Om tot een uitspraak hierover te kunnen komen wordt de entropie van Markov-ketens berekend.



Markov

7.5.1 Entropie Markov-keten

Een Markov-keten van lengte 1 bestaat uit losstaande gebeurtenissen, verlenging tot een keten van lengte 2 houdt rekening met de interactie tussen twee elkaar direct opvolgende gebeurtenissen. In een keten van lengte 2 is er na het optreden van de eerste gebeurtenis i een voorwaardelijke kans $p_{j|i}$ op de vervolgstap j .

Een digram (twee-lettergroep) als bijvoorbeeld DE, is een keten van twee opeenvolgende met elkaar samenhangende letters en derhalve een Markov-keten van lengte 2. In het geval van de Q – in gewone tekst zonder uitzondering gevolgd door de U – geldt dat de voorwaardelijke kans voor de letter na de Q gelijk is aan 1 voor de U en 0 voor de overige letters. De waarde van de voorwaardelijke kansen kan experimenteel worden bepaald door het tellen van letters en lettercombinaties in representatieve stukken tekst.

$$p_{j|i} = \frac{\#ij}{\#i} \quad (7.35)$$

De kans op de verschillende combinaties van twee letters wordt weergegeven in de matrix

$$P = \begin{pmatrix} p(A|A) & p(B|A) & \cdots & p(Z|A) \\ p(A|B) & p(B|B) & \cdots & p(Z|B) \\ \cdots & \cdots & \cdots & \cdots \\ p(A|Z) & p(B|Z) & \cdots & p(Z|Z) \end{pmatrix} \quad (7.36)$$

Dit Markov-model kan uitgebreid worden tot ketens van twee of meer voorafgaande gebeurtenissen. Bijvoorbeeld de letter die volgt op QU is nagenoeg zonder uitzondering een van de klinkers A, E, I of O. In termen van Markov-ketens dus een gebeurtenis X die volgt op de simultane gebeurtenis QU met $p_{X|QU}$ als de bijbehorende voorwaardelijke kans.

Hoeveel informatie verbonden is met ensembles waarin zulke gebeurtenissen optreden, is al aangegeven bij de behandeling van de begrippen equivocatie en wederzijdse informatie. Pas de berekening van deze grootheden toe op Markov-ketens om hun informatiewaarde te bepalen in termen van de *hoeveelheid informatie die een volgend symbool toevoegt aan een bestaande keten*. Die waarde kan dan vervolgens worden gebruikt om voor tekst de gemiddelde waarde per letter (bij benadering) te berekenen. De K^e letter als verlenging van de keten $I \dots J$ naar $I \dots JK$ voegt aan de keten $I \dots J$ een hoeveelheid informatie toe die gelijk is aan de equivocatie van deze toegevoegde letter gegeven die voorafgaande keten.

$$\begin{aligned}
 H(I \dots JK) &= - \sum_{i \dots j, k} p_{i \dots jk} \log p_{i \dots jk} \\
 &= - \sum_{i \dots j, k} p_{i \dots j} p_{k|i \dots j} \log p_{i \dots j} p_{k|i \dots j} \\
 &= - \sum_{i \dots j} p_{i \dots j} \log p_{i \dots j} \sum_k p_{k|i \dots j} - \sum_{i \dots j, k} p_{i \dots j} p_{k|i \dots j} \log p_{k|i \dots j} \\
 &= H(I \dots J) + H(K|I \dots J)
 \end{aligned} \tag{7.37}$$

Aangetoond is eerder dat de equivocatie kleiner (of hoogstens gelijk) is aan die van de symbooldistributie in kwestie, zie [vergelijking \(7.25\)](#). Daarom wordt per verlengende stap over het algemeen steeds minder dan de maximaal mogelijke hoeveelheid informatie toegevoegd. Aldus valt in te zien dat met elk symbool waarmee de Markov-keten wordt verlengd (tot een zekere grens) steeds minder informatie wordt toegevoegd.

$$H(L|I \dots JK) \leq H(K|I \dots J) \leq \dots \leq H(I) \tag{7.38}$$

Omdat een entropie altijd groter of hoogstens gelijk aan nul is, zal de toegevoegde waarde per symbool bij toenemende lengte van de keten een grenswaarde naderen. Hiermee is een eerste belangrijk tussenresultaat op weg naar de bepaling van de uniciteitsafstand bereikt, namelijk de hoeveelheid informatie in taal. Voor berichten van lengte n letters blijkt de totale informatiewaarde tenminste gelijk aan n keer de informatie die per letter wordt toegevoegd aan een bestaande keten. Die waarde per letter wordt aangeduid met $H_1, H_2, H_3 \dots$ voor ketens van lengte 1, 2, 3 enz. Met H_0 bedoelt men de informatiewaarde van de verzameling afzonderlijke letters en met H_∞ die voor een letter toegevoegd aan een in principe oneindig lange keten.

$$\begin{aligned} H_n &= H_1(S_1) + H_2(S_2|S_1) + \dots + H_n(S_n|S_1 \dots S_{n-1}) \\ &\geq n \cdot H_n(S_n|S_1 \dots S_{n-1}) \end{aligned} \quad (7.39)$$

In de laatste regel is het feit dat per letter verlenging steeds minder informatie wordt toegevoegd gebruikt om een ondergrens voor de entropie H_n te bepalen.

Uit tellingen kan H_n voor verschillende waarden van n worden bepaald. Voor Markov-ketens van lengte 1 (losse letters) in de Engelse taal zijn de kansen op de afzonderlijke letters weergegeven in [tabel 7.1](#). Daaruit wordt berekend dat $H_1 = 4,167$. Enkele overige waarden voor het Engels staan in [tabel 7.2](#).

H_i = entropie voor lengte i
$H_0 = 4,7$
$H_1 = 4,2$
$H_2 = 3,6$
$H_3 = 3,2$
...
$H_\infty = 1,2$

Tabel 7.2 Entropiewaarden Engels

De aandacht wordt in het bijzonder gevestigd op H_∞ , de laatste waarde in deze tabel. Die representeert als gezegd de informatiewaarde per letter voor lange teksten. In feite kan men H_∞ opvatten als de gemiddelde waarde voor een letter in de Engelse taal. In de Engelstalige literatuur wordt deze waarde de *rate of the language* genoemd.

Het verschil tussen de maximale entropie per letter H_0 (voor een willekeurige verzameling letters) en de gemiddelde waarde H_∞ voor natuurlijke taal kan worden geïnterpreteerd als een *ongebruikte codeercapaciteit*. Ons taalgebruik is kennelijk inefficiënt waar het de hoeveelheid symbolen betreft waarmee informatie wordt neergeschreven. Met 26 letters zijn in theorie heel wat meer berichten mogelijk dan in (begrijpelijke) taal voorkomen. De ongebruikte codeercapaciteit wordt de *redundantie* in de taal genoemd.

$$D = H_0 - H_\infty \quad (7.40)$$

Computerprogramma's waarmee teksten tot geringere omvang worden gecomprimeerd maken gebruik van dit fenomeen. Ze hercoderen de tekst zodanig dat zo goed mogelijk H_0 wordt benaderd. Dit is ook meteen een ondergrens aan de mogelijkheid tot compressie, omdat bij minder dan H_0 bits per teken informatie verloren zal gaan.

7.5.2 Entropie Caesar-substitutie

Als illustratie van het effect van de toenemende lengte van een tekstfragment op de waarde van de entropie wordt in [tabel 7.3](#) een cryptogram in de Caesar-substitutie getoond. De tekst van het cryptogram ABIGPMZM wordt met alle sleutels ontcijferd. Vervolgens is voor elk van deze mogelijkheden (het cryptogram zelf inbegrepen) uitgerekend wat de kans op voorkomen is. Om te beginnen voor het eerste digram, vervolgens voor het eerste tetragram, enz.

De kansen in de eerste kolom komen uit een tabel van digramfrequenties voor het Engels. In de andere kolommen is de kans op voorkomen benaderd door de kansen op de opeenvolgende digrammen met

L=2	L=4	L=6	L=8
AB 0,044	ABIG 0,139	ABIGPM 0,001	ABIGPMZM 0,000
BC 0,000	BCJH 0,000	BCJHQN 0,000	BCJHQNaN 0,000
CD 0,002	CDKI 0,008	CDKIRO 0,003	CDKIROBO 0,000
DE 0,142	DELJ 0,041	DELJSP 0,004	DELJSPCP 0,000
EF 0,088	EFMK 0,000	EFMKTQ 0,000	EFMKTQDQ 0,000
FG 0,002	FGNL 0,004	FGNLUR 0,001	FGNLURER 0,001
GH 0,033	GHOM 0,000	GHOMVS 0,000	GHOMVSFS 0,000
HI 0,125	HIPN 0,000	HIPNWT 0,000	HIPNWTGT 0,000
IJ 0,000	IJQO 0,000	IJQOXU 0,000	IJQOXUHU 0,000
JK 0,000	JKRP 0,000	JKRPYV 0,000	JKRPYVIV 0,000
KL 0,000	KLSQ 0,000	KLSQZW 0,000	KLSQZWJW 0,000
LM 0,004	LMTR 0,062	LMTRAX 0,000	LMTRAXKX 0,000
MN 0,002	MNUS 0,025	MNUSBY 0,002	MNUSBYLY 0,000
NO 0,118	NOVT 0,000	NOVTCZ 0,000	NOVTCZMZ 0,000
OP 0,050	OPWU 0,000	OPWUDA 0,000	OPWUDANA 0,000
PQ 0,000	PQXV 0,000	PQXVEB 0,000	PQXVEBOB 0,000
QR 0,000	QRYW 0,000	QRYWFC 0,000	QRYWFCPC 0,000
RS 0,081	RSZX 0,000	RSZXGD 0,000	RSZXGDQD 0,000
ST 0,254	STAY 0,658	STAYHE 0,989	STAYHERE 0,998
TU 0,046	TUBZ 0,000	TUBZIF 0,000	TUBZIFSf 0,000
UV 0,007	UVCA 0,061	UVCAJG 0,000	UVCAJGTG 0,000
VW 0,000	VWDB 0,000	VWDBKH 0,000	VWDBKHUH 0,000
WX 0,000	Wxec 0,000	WXECLI 0,000	WXECLIVI 0,000
XY 0,000	XYFD 0,000	XYFDMJ 0,000	XYFDMJWJ 0,000
YZ 0,000	YZGE 0,000	YZGENK 0,000	YZGENKXK 0,000
ZA 0,002	ZAHF 0,003	ZAHFOL 0,001	ZAHFOLYL 0,000
H = 3,182	H = 1,719	H = 0,118	H = 0,022

Tabel 7.3 Entropie Caesar-substitutie van STAYHERE

elkaar te vermenigvuldigen^{*}. De kansen uit een volledige digram-frequentietabel kunnen overigens niet zonder meer worden gebruikt, omdat die ervan uitgaan dat alle 676 digrammen kunnen voorkomen. Dat is voor een Caesar-substitutie een onjuiste aanname: elk digram heeft slechts 25 in plaats van 675 mogelijke vertijferingen. Daarom zijn de frequenties in elke kolom zo geschaald dat hun totaal op 1 uitkomt.

Goed valt te zien dat de entropie afneemt met toenemende lengte van het ontcijferde fragment. Met de eerste twee letters is een tiental digrammen in de race, met vier letters is een deel daarvan al als serieuze kandidaat voor de ontcijfering afgefallen. Bij acht letters is de entropie van de verzameling van 26 mogelijke oplossingen nagenoeg nul. Vanaf zes letters bestaat er al weinig twijfel over de juiste ontcijfering.

7.6 Informatie in cryptografie

De informatietheorie kan het een en ander vertellen over de kansen op het bepalen van de sleutel van een geheimschrift. Hiertoe wordt teruggerepen op het eerder geïntroduceerde begrip wederzijdse informatie. Uit bijvoorbeeld de probleemstelling van de known-plaintext-analyse (cryptogram met bekende klaartekst) kan een uitdrukking worden afgeleid voor de hoeveelheid informatie aangaande de sleutel die in zo'n analyse ter beschikking komt.

Het systeem waarin de cryptanalyse zich beweegt bevat drie grootheden: het cryptogram C , de klaartekst van het bericht M en de sleutel K waarmee het bericht vertijferd is. In feite heeft men te maken met een simultane gebeurtenis uit het ensemble MCK van berichten, cryptogrammen en sleutels (zie [figuur 7.1](#)). Als de klaartekst van een (onderschept) cryptogram bekend is zijn M en C beide tegelijk beschikbaar voor de analyse.

De vraag is nu hoeveel informatie over de vertijfersleutel daarmee bekend kan zijn geworden. Deze hoeveelheid wordt gegeven door de voor deze situatie toepasselijke wederzijdse informatie $W((C, M), K)$. Die

^{*} Een ruwe benadering, maar voldoende om het gewenste effect te laten zien.

representeert de informatie die ter beschikking komt uit de simultane gebeurtenis cryptogram plus klaretekst (C, M) . Ze wordt in de vakliteratuur aangeduid met *key appearance*, een uitdrukking die zich slecht in het Nederlands laat vertalen: zo ongeveer *datgene wat omtrent de sleutel geopenbaard wordt*. Uit de definitie van [vergelijking \(7.28\)](#) blijkt dat deze wederzijdse informatie geschreven kan worden als

$$W((C, M), K) = H(K) - H(K|(C, M)) \quad (7.41)$$

Dat wil zeggen, als het verschil van de *sleutelentropie* $H(K)$ (het aantal realiseringsmogelijkheden van de sleutel) en de equivocatie $H(K|(C, M))$.

De wederzijdse informatie heeft de waarde nul als geen enkele informatie over de sleutel uit de waarneming van M en C kan worden afgeleid. Dit is het geval wanneer de key appearance $H(K|(C, M))$ de maximale waarde ter grootte van $H(K)$ aanneemt. Dat betekent: dan is bij elk paar (M, C) elke sleutel K nog een reële mogelijkheid. Maximale onzekerheid over de identiteit van de sleutel is hiervan het gevolg.

Anderzijds betekent een minimale waarde $H(K|(C, M)) = 0$ van de key appearance dat er geen onzekerheid meer bestaat omtrent de sleutel. Met andere woorden: gegeven het paar (C, M) ligt de sleutel eenduidig vast. Een perfecte deductie van de sleutel is in principe mogelijk.

Om iets te kunnen zeggen over de waarde van $H(K|(C, M))$ wordt deze grootheid uitgewerkt. Schrijf hiertoe de totale informatie in het systeem van berichten, cryptogrammen en sleutels $\mathcal{M}, C, \mathcal{K}$ op, en splits deze vervolgens op twee manieren in afzonderlijke componenten (zoals in [vergelijking \(7.21\)](#)). Omdat beide splitsingen op eenzelfde situatie slaan hebben ze vanzelfsprekend dezelfde waarde

$$\begin{aligned} H(M, C, K) &= H(M|(C, K)) + H(C, K) \\ H(M, C, K) &= H(K|(C, M)) + H(M, C) \end{aligned} \quad (7.42)$$

Nu is direct duidelijk dat

$$H(M|(C, K)) = 0 \quad (7.43)$$

Want die grootheid representeert de informatie die over het bericht ter beschikking komt, gegeven het cryptogram en de sleutel. Hier bestaat geen enkele onzekerheid, omdat in het bezit gesteld van cryptogram en sleutel de ontcijfering kan worden uitgevoerd. Dus moet de equivocatie voor die situatie wel de waarde nul hebben. Vul $H(M|(C, K)) = 0$ in en splits de verkregen grootheden $H(C|K)$ en $H(C|M)$ op dezelfde manier

$$\begin{aligned} H(C, K) &= H(C) + H(K|C) \\ H(C, M) &= H(C) + H(M|C) \end{aligned} \quad (7.44)$$

Na invullen hiervan in [vergelijking \(7.42\)](#) luidt het resultaat

$$H(K|(C, M)) = H(K|C) - H(M|C) \quad (7.45)$$

Hieruit kan de volgende interessante gevolgtrekking gemaakt worden. In het geval van een known-plaintext-analyse wordt de waarde van de key appearance $H(K|C)$ (eerste term in het rechterlid) geldend voor een ciphertext-only-analyse, verminderd met een bedrag $H(M|C)$. Deze verminderde waarde leidt uiteraard der zaak tot minder onzekerheid omtrent de sleutel. Het bekend worden van een known-plaintext paar doet dus meer informatie uitlekken over de sleutel dan een ciphertext-only analyse.

Op soortgelijke wijze kan met de key appearance iets gezegd worden over de mogelijkheid tot ciphertext-only analyse waar de cryptoanalist alleen over cryptogrammen beschikt. Dit geval wordt beschreven door de wederzijdse informatie $W(C, M)$. Schrijf $W(C, M)$ als

$$W(C, M) = H(M) - H(M|C) \quad (7.46)$$

En gebruik de uitdrukking

$$H(K|(C, M)) = H(K|C) - H(M|C) \quad (7.47)$$

Om met de constatering dat $H(K|(M, C)) \geq 0$ (geldt nu eenmaal voor elke entropie) te concluderen dat

$$H(M|C) \leq H(K|C) \quad (7.48)$$

Waaruit dan vanwege $H(K|C) \leq H(K)$ weer volgt

$$W(C, M) = H(M) - H(M|C) \geq H(M) - H(K) \quad (7.49)$$

De zojuist afgeleide betrekking kan als volgt worden geïnterpreteerd. De wederzijdse informatie tussen de klaartekst en het cryptogram is tenminste zo groot als het verschil van de berichtentropie en de sleutelentropie. De entropie van de sleutel drukt dus als het ware de informatie van het bericht weg. Is de sleutelentropie groot genoeg, dan resteert er geen wederzijdse informatie en is een situatie van absolute veiligheid bereikt. De sleutelentropie groeit met het aantal sleutels

$$H(K) = \log_2 |K| \quad (7.50)$$

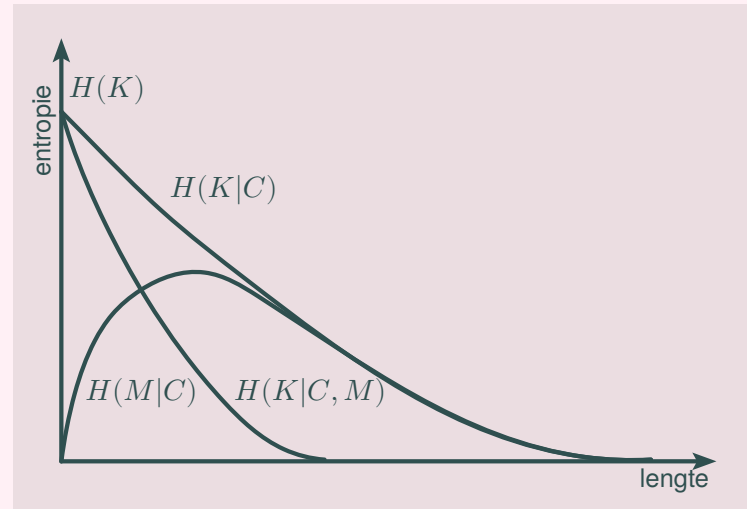
In de praktijk betekent dit: hoe meer sleutels (sleutelbits) hoe beter.

7.6.1 Uniciteitsafstand

De grafiek in [figuur 7.4](#) geeft het verloop weer van de entropie-grootheden $H(K|C)$, $H(K|(C, M))$ en $H(M|C)$ als functie van de lengte van het bericht^[4]. $H(K|C)$ en $H(K|(C, M))$ representeren de equivocatie bij respectievelijk ciphertext-only- en known-plaintext-analyse. Het punt waar deze krommen de horizontale as bereiken (beide naderen daartoe) komt overeen met equivocatie nul. Dat wil zeggen er bestaat vanaf dat punt geen onzekerheid meer over de sleutel. De horizontale coördinaat geeft aan voor welke berichtlengte dit het geval is: de uniciteitsafstand.

Uit het verloop van $H(K|(C, M))$ en $H(K|C)$ blijkt dat known-plaintext eerder de horizontale as vindt dan ciphertext-only. Dit is intuïtief juist, omdat naar verwachting meer informatie (bekend paar klaartekst en cryptogram) eerder zekerheid zal geven. Dit kan ook uit de betreffende grootheden worden afgeleid. Eerder was voor een known-plaintext-analyse al gevonden

$$H(K|(C, M)) = H(K|C) - H(M|C) \quad (7.51)$$



Figuur 7.4 Uniciteitsafstand en equivocatie

Wetend dat een entropiewaarde altijd groter of gelijk nul is, mag worden geconcludeerd dat

$$H(K|(C, M)) \leq H(K|C) \quad (7.52)$$

Dus ligt $H(K|(C, M))$ beneden $H(K|C)$.

De grootte $H(M|C)$ representeert de cryptoanalyse van een bericht uit het cryptogram zonder referentie aan de sleutel. Te zien is dat $H(M|C)$ – na een afwijkende start – naar $H(K|C)$ toe gaat. Dat wil zeggen, ciphertext-only-analyse van bericht en sleutel hebben allebei ongeveer evenveel tekst nodig om zekerheid te bereiken. De lage waarde van $H(M|C)$ aan het begin is het gevolg van het feit dat het aantal berichten van korte lengte gering is; bijgevolg kan er dan ook weinig onzekerheid over bestaan. Maar zodra de lengte van het bericht toeneemt, neemt ook het aantal mogelijke berichten sterk toe en dus groeit de onzekerheid aanvankelijk snel.

Merk ook op dat bij een bericht van lengte nul (nog voordat van een cryptogram sprake is) de onzekerheid

over de sleutel begint met een waarde gelijk aan de sleutelentropie $H(K)$. Ook dit klopt met het gevoel. Zolang er geen cryptogram is kan immers nog elke sleutel worden gekozen. Door het maken van een cryptogram is pas vanaf dat moment sprake van een gekozen sleutel. Informatie over deze sleutel lekt uit via het cryptogram. Hoe langer de gecijferde tekst, hoe meer informatie over deze sleutel ter beschikking komt. Uiteindelijk zoveel dat deze eenduidig kan worden bepaald.

Een benadering van de uniciteitsafstand kan worden gevonden door na te gaan wat de limietwaarde voor $H(K|C)$ is.

$$H(C, K) = H(C) + H(K|C) \rightarrow H(K|C) = H(C, K) - H(C) \quad (7.53)$$

Gebruik het feit dat er een 1 op 1 relatie bestaat tussen de klaartekst en het cryptogram, gegeven een bepaalde sleutelkeuze, zodat

$$H(C, K) = H(M, K) \rightarrow H(K|C) = H(M, K) - H(C) \quad (7.54)$$

Splits $H(M, K)$ in de som van $H(K)$ en $H(M)$, een gevolg van het feit dat sleutel en klaartekst onafhankelijk van elkaar gekozen worden. Het zijn twee onafhankelijk van elkaar optredende gebeurtenissen en dus is de totale entropie de som van beide

$$H(M, K) = H(M) + H(K) \rightarrow H(K|C) = H(K) + H(M) - H(C) \quad (7.55)$$

De waarde van $H(C)$ voor een cryptogram van lengte n kan worden benaderd uit de entropie voor een Markov-keten (zie [vergelijking \(7.39\)](#))

$$\begin{aligned} H(C) &\leq n \cdot H_0 \\ H(M) &\geq n \cdot H_\infty \end{aligned} \quad (7.56)$$

Daarmee wordt in [vergelijking \(7.55\)](#) de term $H(C)$ vervangen door $n \cdot H_0$ en $H(M)$ door $n \cdot H_\infty$. Als gevolg van bovenstaande ongelijkheden wordt nu de volgende afschatting verkregen.

$$H(K|C) \geq H(K) + n \cdot H_\infty - n \cdot H_0 \quad (7.57)$$

Noem $D = H_0 - H_\infty$ de in [vergelijking \(7.40\)](#) geïntroduceerde redundantie en dit wordt

$$H(K|C) \geq H(K) - n \cdot D \quad (7.58)$$

De onzekerheid is verdwenen zodra de equivocatie $H(K|C)$ de waarde nul bereikt heeft, dus ongeveer bij

$$H(K|C) \approx 0 \rightarrow H(K) - n \cdot D \approx 0 \quad (7.59)$$

Hieruit tenslotte volgt de waarde van n voor de uniciteitsafstand ud .

$$ud \approx \frac{H(K)}{D} = \frac{H(K)}{H_0 - H_\infty} \quad (7.60)$$

De lengte van de sleutel is in veel systemen een vaststaand aantal bits, zodat verlenging van de uniciteitsafstand moet worden gevonden in vermindering van de redundantie: een kleinere D in de noemer leidt tot een grotere ud . Dit betekent dat het zin heeft om teksten die moeten worden versleuteld vóóraf door een compressie-algoritme van zoveel mogelijk redundantie te ontdoen. Omgekeerd blijkt uit deze formule de drijvende kracht van de redundantie als motor achter de cryptanalyse. Tot slot kan de conclusie worden getrokken dat onzekerheid over de sleutel blijft bestaan zolang maar

$$H(K) \geq ud \cdot D \quad (7.61)$$

Tot slot nog een opmerking over de praktische betekenis van de uniciteitsafstand behorend bij een cryptosysteem. Het is verleidelijk om te concluderen dat een cryptogram langer dan de uniciteitsafstand *dus* altijd kan worden opgelost. Niets is minder waar. De uniciteitsafstand beweert slechts dat een cryptogram van tenminste die lengte prijsgeeft of de gekozen sleutel de juiste is. Omdat in principe alle sleutels zouden kunnen worden geprobeerd, is in die zin de oplossing wel gegarandeerd. Maar misschien zijn er zoveel sleutels dat het praktisch ondoenlijk is om ze alle te proberen. De uniciteitsafstand garandeert niet dat het cryptogram op een andere wijze kan worden opgelost of hoe gemakkelijk dat zal zijn. Het is best mogelijk dat een of andere aanpak tot success zal leiden, maar waarschijnlijk is daarvoor vele malen meer materiaal nodig dan de uniciteitsafstand.

En als laatste nog dit. Uit de aard van de berekeningen volgt dat de uniciteitsafstand niet zoiets als een natuurconstante is, die tot vele decimalen achter de komma nauwkeurig kan worden bepaald. Het is eerder een wat vaag getal dat meer indicatief dan precies gehanteerd moet worden. Dat de uniciteitsafstand desondanks bestaansrecht heeft en toch ook weer niet zo vreselijk vaag is, is met behulp van wat kansrekening duidelijk te maken.

7.6.2 Voorbeelden uniciteitsafstand

Als illustratie van de uniciteitsafstand zal die worden afgeleid voor enkele eenvoudige geheimschriften. Om een indruk te krijgen eerst enkele resultaten ontleend aan de literatuur^{[5]*}.

- Vigenère met periode p : $1,3p$ letters
- Vigenère met gemengd cijferalfabet: $24 + 1,3p$ letters
- Vigenère met gemengd klaar- en cijferalfabet: $48 + 1,3p$ letters
- Vigenère met n onafhankelijke alfabetten in successie: $24n + 1,3p$ letters
- Playfair: 23 letters
- Willekeurige digramsubstitutie: 1460 letters

Monoalfabetische substitutie

In de monoalfabetische substitutie worden de letters van het alfabet vervangen volgens een willekeurige alfabetpermutatie van het standaard 26-letter alfabet A-Z. Een cryptogram in monoalfabetische substitutie doet zich in eerste aanleg voor als een willekeurig assortiment van letters. Benader de informatiewaarde van monoalfabetische cryptogrammen daarom in eerste aanleg met H_0 (waarde 4,7 voor 26-letter alfabet).

* Let op, hier is niet in *bits* maar in *letters* gerekend.

Weliswaar gaat deze aanname niet op voor één enkel cryptogram met één enkele sleutel (de letterdistributie is en blijft ook na substitutie H_1), maar in een complete verzameling cryptogrammen met meerdere sleutels wordt de randomdistributie wel benaderd en zo erg veel scheelt het niet. Voor de klaartekst mag H_∞ (waarde bij benadering 1,2) worden gebruikt. De redundantie komt aldus uit op $4,7 - 1,2 = 3,5$.

Omdat de uniciteitsafstand bij benadering gelijk is aan de entropie van de sleutel gedeeld door de redundantie, hebben we de entropie van de verzameling sleutels nodig. In het geval van de monoalfabetische substitutie zijn dat alle verwisselingen binnen een 26-letteralfabet. Er zijn $26! \approx 4 \times 10^{26}$ alfabetpermutaties en de entropie van 88,4 bits is de 2-logaritme daarvan. Aldus resulteert een uniciteitsafstand van ruwweg 25 letters.

$$ud = \frac{H(K)}{H_0 - H_\infty} \approx \frac{88.4}{4.7 - 1.2} = 25 \text{ letters}$$

In deze benadering zitten enkele aannames die nog wat kunnen worden bijgesteld. Zo is het gebruik van de waarde 1,2 voor H_∞ aan de optimistische kant, omdat die waarde eigenlijk pas bij grotere lengte wordt bereikt. Het getal 1,8 is een betere schatting voor cryptogramfragmenten van rond de 25 letters.

Aan de andere kant wordt in bovenstaande berekening het aantal sleutels overschat. Dat komt omdat in een tekstfragment van ongeveer 25 letters niet alle 26 letters van het alfabet zullen voorkomen. De verwisselingen tussen de afwezige letters tellen natuurlijk niet mee voor de bepaling van het aantal sleutels. Die hebben immers geen invloed op de uitkomst van de vercijfering.

lengte	5	10	20	30	50	100	500	1000
aantal	5	8	12	14	17	20	24	25

Tabel 7.4 Verwacht aantal verschillende letters

In [tabel 7.4](#) treft men het resultaat aan van een berekening voor het aantal verschillende letters dat gemiddeld in een fragment van zekere lengte zal voorkomen. Deze tabel kan worden berekend uit de

letterfrequenties. Men ziet dat in fragmenten van rond de 25 letters doorgaans niet meer dan zo'n 14 verschillende letters present zullen zijn. Daarom moeten de verwisselingen tussen de gemiddeld 12 afwezige letters weggedeeld worden uit het aantal sleutels. Hierdoor daalt de sleutelentropie tot ongeveer de 2-logaritme van $26!/12!$. Tezamen met de voorgaande opmerking over H_∞ leidt dit tot een uniciteitsafstand van om en nabij 22 letters.

Transpositie

Een transpositie cryptosysteem bestaat in de meest algemene vorm uit een willekeurige verwisseling van de N elementen van het transpositieblok. In het algemeen heeft een transpositie dus $N!$ sleutels. Behalve de uniciteitsafstand voor een algemene transpositie wordt deze ook voor een $N \times N$ rooster (turning grille) berekend, een systeem dat aanmerkelijk minder sleutels telt. Een vierkant rooster met N posities kent slechts $4^{N/4-1}$ mogelijkheden*. Zodoende heeft een algemene 64-letter transpositie een sleutelentropie van $\log_2 64! \approx 170$ bits en een 8×8 rooster van 64 letters een sleutelentropie van slechts $\log_2 4^{15} = 30$ bits.

Voor transposities kan H_0 zeker niet gebruikt worden als benadering voor de entropie van de cryptogrammen. Immers in alle transposities is en blijft de verdeling van de letters dezelfde als die in de klaartekst. Dus moet H_1 in plaats van H_0 worden gebruikt bij de berekening van de redundantie.

Tabel 7.5 geeft wat resultaten van de uniciteitsberekening voor een algemene transpositie van 20 letters en voor het 8×8 turning grille. De opvolgende rijen laten zien hoe de analysetechniek van invloed is op de effectieve uniciteitsafstand. Door langere n -grammen in de analyse te betrekken, wordt de uniciteitsafstand verkort en is met kortere fragmenten succes mogelijk.

* De -1 in de exponent is het gevolg van het feit dat de eerste positie willekeurig is. D.w.z. de beginrotatie doet er niet toe. Met als gevolg dat het aantal mogelijke roosters met een factor 4 vermindert.

n	H_n	N=20	8x8
1	4,2	∞	∞
2	3,6	111	55
3	3,2	65	32
7	2,8	45	22
∞	1,2	20	10

Tabel 7.5 Uniciteitsafstand voor transposities

Wanneer slechts de letterfrequenties worden ingezet is cryptanalyse van een transpositie onmogelijk (uniciteitsafstand oneindig). Dat is logisch omdat in transposities niets met de letters zelf gebeurt. Pas wanneer langere Markov-ketens in de strijd geworpen worden, kan men iets bereiken. Men ziet dat met het gebruik van digramfrequenties de algemene transpositie nog niet te breken valt. De uniciteitsafstand is met een waarde van 100 groter dan de beschikbare 20 letters van het transpositieblok*. In het geval van het 8x8 rooster bevindt men zich met een uniciteitsafstand van 55 letters al iets onder de grens van een mogelijke oplossing. Naarmate meer samenhang in de tekst voor de cryptanalyse wordt ingezet (langere Markov-ketens, dus hogere n in H_n), ziet men inderdaad de effectieve uniciteitsafstand navenant afnemen.

7.6.3 Relativering Shannon-entropie

De in het voorafgaande gehanteerde definitie van entropie noemt men de *Shannon-entropie* naar de grondlegger van de informatietheorie. De onderliggende veronderstelling bij het gebruik van deze definitie is, dat het te coderen bronmateriaal een uniforme verdeling heeft. D.w.z. alle mogelijke letter/bit patronen

* Dat verbetert natuurlijk wanneer men over meer transpositieblokken in parallel beschikt en meervoudig kan anagrammeren.

komen naar verhouding even vaak voor. In de praktijk is dat veelal niet zo, denk maar aan de uitgesproken letterverdeling in literaire teksten. Uniformiteit wordt nog het best verkregen door op het bronmateriaal tekstcompressie toe te passen, waarna in binaire representatie het aantal nullen en enen ten naaste bij gelijk is en bovendien 'keurig' verdeeld.

Voor toepassingen als datacompressie en foutcorrectie is de Shannon-entropie goed genoeg, maar voor cryptografie ligt dat een beetje anders*. Cryptografie en in het bijzonder cryptoanalyse, hebben niet zozeer te maken met het gemiddelde dat voortspruit uit uniformiteit, maar meer met een 'worst case' scenario. De cryptoanalyst heeft in veel gevallen genoeg aan een enkele gelukkige greep om een begin te kunnen maken met het breken van het cryptogram. Denk bijvoorbeeld aan een monoalfabeet waarin een groep letters met een opvallende herhalingskarakteristiek de eerste stap in de ontcijfering mogelijk maakt, waarna de rest van het cryptogram een fluitje van een cent is. Ieder die zulke cryptogrammen heeft opgelost, begrijpt onmiddellijk wat wordt bedoeld.

Vanuit de informatietheorie bezien, betekent dit dat in de definitie van entropie een meer dan gemiddeld gewicht moet worden toegekend aan de minder waarschijnlijke mogelijkheden. Het gevolg hiervan is, dat de Shannon-entropie een iets te rooskleurig beeld geeft van de veiligheid van het systeem.

De Shannon-entropie is eerder in dit hoofdstuk gedefinieerd als

$$H_{Shannon} = - \sum_i p_i \log_2 p_i \quad (7.62)$$

Dit is echter slechts een van de vele definities die voor het begrip entropie zijn te geven. Een andere, meer op de cryptografie toegesneden definitie, neemt de kleinste waarde van de logaritme van p_i

$$H_{min} = \min_i (-\log_2 p_i) \quad (7.63)$$

Deze vorm van entropie staat bekend als de *min-entropie*. Evenals bij de Shannon-entropie heeft een uniforme verzameling van 2^K sleutels in deze definitie een entropie van K bits.

* MIT-news: <http://web.mit.edu>, 14 augustus 2013.

Een meer algemene definitie is die van de *Renyi-entropie*. Deze is gedefinieerd als

$$H_t = \frac{1}{1-t} \log_2 \sum_i p_i^t \quad t > 0, t \neq 1 \quad (7.64)$$

De t is een parameter die de specifieke vorm van de Renyi-entropie bepaalt. Als $t \rightarrow \infty$ dan nadert de Renyi-entropie de min-entropie, en bij $t \rightarrow 1$ de Shannon-entropie. Shannon- en min-entropie zijn dus beide bijzondere gevallen van de Renyi-entropie. Men kan aantonen dat de waarde van de Renyi-entropie daalt wanneer t groter wordt, waaruit kan worden afgeleid dat de min-entropie kleiner of hoogstens gelijk is aan de Shannon-entropie. Een conclusie die in lijn ligt met de in het voorgaande geponeerde stelling dat de Shannon-entropie een overschatting geeft van de voor cryptografie relevante entropie en daarmee een te optimistisch beeld van de cryptografische veiligheid.

Voor nog andere benaderingen van de entropie zie het artikel *Formally Assessing Cryptographic Entropy* van Daniel Brown in het ePrint-archief van de IACR^[6].

7.7 Informatieoverdracht

Een *informatiekanaal* is een medium waarover informatie kan worden overgedragen. Van bijzondere interesse is de beschrijving van zo'n kanaal vanuit het gezichtspunt van de informatietheorie, omdat die kan worden gekoppeld aan de cryptografische beveiliging van de informatieoverdracht. Als voorbeeld wordt een eenvoudig model behandeld, het *binair symmetrisch kanaal*. Getoond wordt hoe hiermee een absoluut veilig geheimschrift kan worden gerealiseerd. In termen van uniciteitsafstand wil dit zeggen: een geheimschrift met een oneindig grote uniciteitsafstand.

Bij een informatiekanaal, zoals geschetst in [figuur 7.3](#), gaan aan de kant van de bron symbolen erin en komen ze aan de bestemmingskant er weer uit. De ingaande symbolen behoeven niet noodzakelijk van dezelfde soort te zijn als de uitgaande symbolen. Het is bijvoorbeeld denkbaar, dat signalen in Morse-code worden verzonden, die door de ontvangstapparatuur als letters worden afgeleverd. Als regel echter

zal men aan beide kanten van het kanaal dezelfde symboolverzameling aantreffen. Om met genoemde mogelijkheid rekening te houden worden voor het kanaal twee symboolverzamelingen gedefinieerd: verzameling $\mathcal{A}$ aan de ingang en $\mathcal{B}$ aan de uitgang van het informatiekanaal. De symbolen van beide verzamelingen hebben alle een bepaalde kans op voorkomen. Voor de symbolen die het kanaal ingaan is deze het gevolg van de kansverdeling van die symbolen in de informatiebron. Aan de uitgang wordt de kans bepaald door zowel de kansverdeling van de bronsymbolen als door de eigenschappen van het kanaal

$$\begin{aligned}\mathcal{A} &= \{a_1, a_2, \dots, a_n\} \quad \text{met } p(a_1), p(a_2) \dots \\ \mathcal{B} &= \{b_1, b_2, \dots, b_m\} \quad \text{met } p(b_1), p(b_2) \dots\end{aligned}\tag{7.65}$$

Als een symbool a_i het kanaal wordt ingestuurd zal een of ander symbool b_j aan de uitgang van het kanaal verschijnen. Het statistisch karakter van het kanaal heeft tot gevolg dat bij herhaald verzenden van a_i in principe alle mogelijke symbolen $b_1 \dots b_m$ kunnen worden ontvangen. De kans op ontvangst van een bepaald symbool b_j als gevolg van het verzenden van a_i kan worden vastgelegd in de voorwaardelijke kans $p(b_j|a_i)$. Elk ingaand symbool a_i kan bijdragen tot ontvangst van b_j , zodat de kans op het verschijnen van b_j in de uitvoer wordt gegeven door de volgende som

$$p(b_j) = \sum_{i=1}^n p(b_j|a_i)p(a_i)\tag{7.66}$$

De verzameling voorwaardelijke kansen voor alle invoer- en uitvoercombinaties kan worden gerepresenteerd als een $m \times n$ -matrix. Daarmee kan de werking van het kanaal worden geschreven als de volgende matrixvermenigvuldiging

$$\begin{pmatrix} p(b_1) \\ p(b_2) \\ \vdots \\ p(b_m) \end{pmatrix} = \begin{pmatrix} p(b_1|a_1) & p(b_1|a_2) & \dots & p(b_1|a_n) \\ p(b_2|a_1) & p(b_2|a_2) & \dots & p(b_2|a_n) \\ \dots & \dots & \dots & \dots \\ p(b_m|a_1) & p(b_m|a_2) & \dots & p(b_m|a_n) \end{pmatrix} \begin{pmatrix} p(a_1) \\ p(a_2) \\ \vdots \\ p(a_n) \end{pmatrix}\tag{7.67}$$

Afgekort tot

$$\vec{p}_b = P \cdot \vec{p}_a \quad (7.68)$$

Als de kansen onveranderlijk vastliggen representeert matrix P een zogenaamd *stationnair kanaal*. Dat is in de praktijk lang niet voor elk communicatiekanaal het geval. Radiocommunicatie is een typisch voorbeeld van een kanaal waarvan de kanaaleigenschappen met de tijd kunnen variëren als gevolg van atmosferische storingen, de wisselende invloed van de zonnenvlekkencyclus of ingeschakelde stoorzenders. Het is zelfs denkbaar dat informatie in het kanaal verloren gaat doordat (een deel van) de verzonden symbolen niet aankomt.

In het geval van een stationnair kanaal waarin *geen* informatie verloren gaat, zal elk symbool dat het kanaal wordt ingestuurd ook leiden tot een uitvoersymbool. De volgende uitdrukkingen geven dan behoudswetten over rijen en kolommen van de kanaalmatrix. De eerste betekent dat een bepaald invoersymbool altijd leidt tot een of ander uitvoersymbool en de tweede dat een uitvoersymbool altijd het gevolg is van een of ander invoersymbool.

$$\forall i: \sum_{j=1}^m p(b_j|a_i) = 1 \quad (7.69)$$

$$\forall j: \sum_{i=1}^n p(b_j|a_i) = 1 \quad (7.70)$$

Er vallen nog meer verbanden aan te wijzen. De kans op het invoer-uitvoerpaar (a_i, b_j) is $p(a_i, b_j)$. Deze kans kan worden berekend als het product van de kans op a_i maal de voorwaardelijke kans $p(b_j|a_i)$ dat het kanaal daaruit b_j produceert. Maar het mag ook andersom bekeken worden. Dan is het de kans op uitvoer b_j maal $p(a_i|b_j)$, de kans dat daaraan invoer a_i ten oorzaak heeft gelegen. Beide gezichtspunten op deze simultane gebeurtenis moeten vanzelfsprekend leiden tot eenzelfde waarde van de simultane kans, zodat

$$p(a_i, b_j) = p(a_i)p(b_j|a_i) = p(b_j)p(a_i|b_j) \quad (7.71)$$

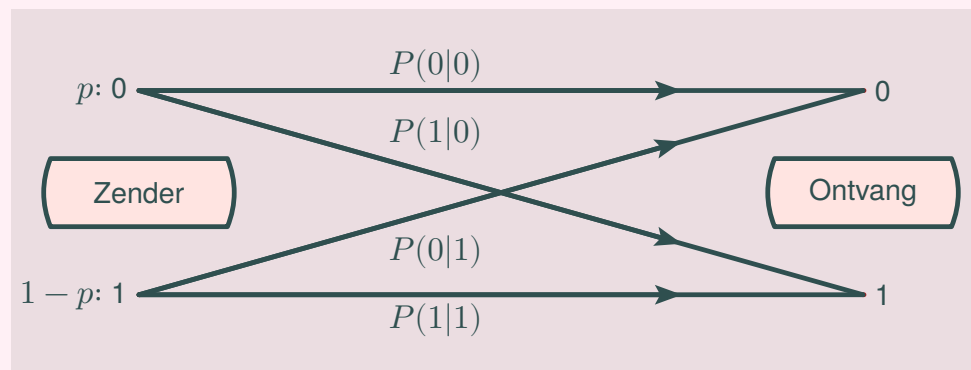
Met behulp van deze uitdrukking kunnen de $p(a_i|b_j)$ worden berekend uit de $p(b_j|a_i)$ en vice versa.

$$p(a_i|b_j) = \frac{p(a_i)p(b_j|a_i)}{p(b_j)} = \frac{p(a_i)p(b_j|a_i)}{\sum_{i=1}^n p(a_i)p(b_j|a_i)} \quad (7.72)$$

7.7.1 Binair symmetrisch kanaal

Een eenvoudig voorbeeld van een informatiekanaal is het *binair symmetrisch kanaal*. In dit kanaalmodel bestaan de invoer- en uitvoerverzamelingen uit het binair alfabet $\{0,1\}$. Een overgezonden 0-bit heeft kans $p_{0|0}$ om als een 0 en kans $p_{1|0}$ om als een 1 te worden ontvangen; voor een 1-bit is $p_{1|1}$ de kans op ontvangst van een 1 en $p_{0|1}$ de kans op een 0. Men noemt het kanaal symmetrisch omdat de kansen $P = p_{0|0} = p_{1|1}$ op onveranderde ontvangst aan elkaar gelijk zijn, evenals de kansen $p_{1|0} = p_{0|1} = (1-P)$ op ontvangst van de tegengestelde waarde. In [figuur 7.5](#) is dit kanaal afgebeeld. Het binair symmetrisch kanaal kan worden gekarakteriseerd door een kanaalmatrix

$$\mathbf{P} = \begin{pmatrix} P & 1-P \\ 1-P & P \end{pmatrix} \quad (7.73)$$



Figuur 7.5 Binair symmetrisch kanaal

Gegeven de invoerdistributie van nullen en enen $(p_0, p_1) = (p, 1-p)$, kan de uitvoerdistributie (q_0, q_1) door uitvoering van de matrixvermenigvuldiging in [vergelijking \(7.67\)](#) worden berekend. Het resultaat luidt

$$\begin{pmatrix} q_0 \\ q_1 \end{pmatrix} = \begin{pmatrix} P & 1-P \\ 1-P & P \end{pmatrix} \begin{pmatrix} p \\ 1-p \end{pmatrix} = \begin{pmatrix} 1-p-P+2pP \\ p+P-2pP \end{pmatrix} \quad (7.74)$$

De lezer kan zelf narekenen dat de geformuleerde behoudswetten op dit model van toepassing zijn. Voor het terugrekenen van de waargenomen distributie naar de verdeling van symbolen aan de bron kan [formule \(7.72\)](#) worden benut; ook dit wordt als oefening aan de lezer overgelaten.

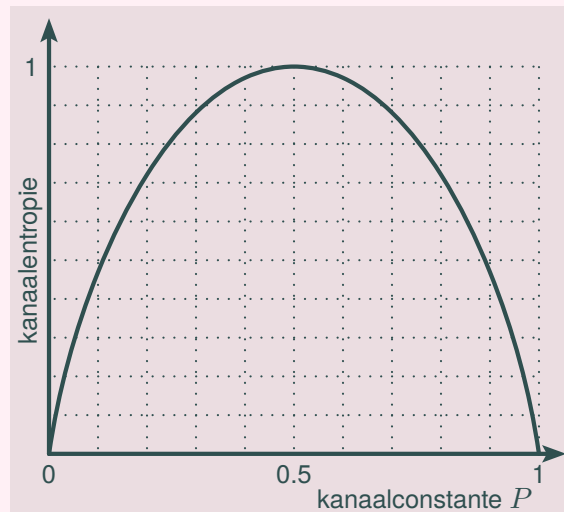
Voor $P = 1$ is er perfecte communicatie: elke 0 en elke 1 worden als zodanig ontvangen, de overgangswaarschijnskheden $p_{1|0}$ en $p_{0|1}$ hebben allebei de waarde nul. In het andere uiterste, waarin $P = 0$, is eveneens sprake van een perfecte communicatie. Dit kanaal levert dan weliswaar geïnverteerde waarden af, maar elke 0 wordt systematisch ontvangen als 1 en omgekeerd. Bij $0 < P < 1$ is er imperfecte communicatie, omdat sommige enen en nullen op onvoorspelbare wijze zullen zijn omgezet in hun tegengestelde. Dit brengt bij de ontvanger onzekerheid omtrent het bronsignaal teweeg. Hoe dichter P tot 0,5 nadert, hoe meer de onzekerheid toeneemt. Voor precies $P = 0,5$ is de onzekerheid volkomen, want dan komt maar de helft van de uitgezonden symbolen goed over zonder dat bekend is welke symbolen goed en welke fout zijn aangekomen.

7.7.2 Kanaalentropie

De *kanaalentropie* (equivocatie) $H(\text{ontvanger} | \text{bron})$ geeft precies aan hoeveel onzekerheid ontstaat, d.w.z. hoeveel informatie in het kanaal verloren gaat. Deze wordt als volgt berekend

$$\begin{aligned}
H(J|I) &= - \sum_{i=0,1} p_i \sum_{j=0,1} p_{j|i} \log p_{j|i} \\
&= -p_0 p_{0|0} \log p_{0|0} - p_0 p_{1|0} \log p_{1|0} - p_1 p_{0|1} \log p_{0|1} - p_1 p_{1|1} \log p_{1|1} \\
&= -pP \log P - p(1-P) \log(1-P) \\
&\quad - (1-p)(1-P) \log(1-P) - (1-p)P \log P \\
&= -P \log P - (1-P) \log(1-P)
\end{aligned} \tag{7.75}$$

In dit resultaat valt op dat de kanaalentropie $H(J|I)$ slechts van de kanaalconstante P afhangt en niet van de invoerdistributie $(p_0, p_1) = (p, 1-p)$. De hoeveelheid onzekerheid in de overgebrachte informatie blijft steeds dezelfde, ongeacht de verdeling van nullen en enen aan de bron.



Figuur 7.6 Kanaalentropie binair symmetrisch kanaal

Het verloop van de waarde van $H(J|I)$ is geschetst in [figuur 7.6](#). Aan de uiteinden bij $P = 0$ en $P = 1$ is de equivocatie nul, overeenkomend met perfecte communicatie. Bij $P = 0,5$ wordt $H(J|I) = 1$ en bereikt daar z'n maximale waarde. Voor een alfabet $\{0,1\}$ is de maximale entropiewaarde gelijk aan $\log_2 2 = 1$. Omdat

bij $P = 0,5$ de kanaalentropie eveneens de waarde 1 heeft, verdwijnt op dat moment dus alle informatie uit het kanaal. Anders gezegd, de *wederzijdse informatie* voor een waarnemer is dan nul.

Voor een cryptografisch systeem betekent deze constatering het bereiken van het hoogste doel: absolute veiligheid omdat de inhoud van getransporteerde berichten in het geheel niet meer kan worden waargenomen. Om dit te bereiken is het nodig de kanaalconstante P zó te manipuleren dat elk bit gemiddeld even vaak onveranderd blijft of omklapt: dat wil zeggen effectief $P = 0,5$. Dit kan worden gedaan door de bits van het bericht met een (pseudo)random bitrij van goede kwaliteit te maskeren. Elk over het cryptografisch binair symmetrisch kanaal te verzenden bit wordt door middel van een *exclusive or* met het volgende bit van de randomrij gecombineerd. Zo ontstaat voor wie de modificerende bitrij niet kent een kanaal met $P = 0,5$ en voor degene die de rij wel kent een perfect kanaal. De bitrij fungeert dus als de sleutel van een one-time pad. De kanaalentropie van $H(J|I) = 1$ garandeert dat een cryptoanalist geen enkele informatie krijgt om een analyse van het geheimschrift mee op te zetten.

7.8 MASTERSECTIE – Slaagkans cryptoanalyse

In het voorgaande is de uniciteitsafstand benaderd vanuit het entropiebegrip. Nu zal een meer op de kansrekening gebaseerde behandeling volgen. Hiermee wordt inzicht verkregen in het verloop van de kans op een oplossing als functie van de lengte van het bericht. Het zal blijken dat deze kans een vrij scherp verloop heeft, zodat het inderdaad zin heeft om van een uniciteits*afstand* te spreken.

De eerste stap is dat bij ontcijfering van alle mogelijke cryptogrammen met alle mogelijke sleutels meerdere zinvolle berichten het resultaat van die ontcijfering zouden kunnen zijn. De stochastische variabele $\underline{M}$ staat voor het aantal mogelijke ontcijferingen dat zo gevonden kan worden.

$$\underline{M} = |k \in K : D(C, k) \in M| \quad \underline{M} \in \{1, \dots, |K|\} \quad (7.76)$$

Als er K sleutels zijn kan $\underline{M}$ alle waarden van $1 \dots K$ aannemen. Dat wil zeggen dat het aantal goed uitziende ontcijferingen alle mogelijke waarden zou kunnen aannemen. Natuurlijk geldt $\underline{M} \geq 1$, omdat de oorspron-

kelijke vercijfersleutel ook altijd een deugdelijke ontcijfering oplevert. Veronderstel verder dat alle goed uitziende oplossingen even waarschijnlijk zijn*. Gegeven een specifieke waarde m voor $\underline{M}$ is de kans p_m op het pakken van de goede oplossing

$$\underline{M} = m \rightarrow p_m = \frac{1}{m} \quad (7.77)$$

Om de totale kans op de goede oplossing te vinden moeten alle mogelijke waarden van m gesommeerd worden, met andere woorden

$$P = \sum_{m=1}^{|K|} \frac{1}{m} p(\underline{M} = m) \quad (7.78)$$

Hierin is $p(\underline{M} = m)$ de kansverdeling van het aantal plausibele oplossingen m . De volgende stap moet dan zijn het uitrekenen van de distributie van $\underline{M}$, dus van het verloop van $p(\underline{M} = m)$.

In principe is de kans p_k dat een willekeurige sleutel $k \in K$ een goede ontcijfering geeft gelijk aan de fractie echte berichten in het totaal aan mogelijke berichten. Met behulp van de grootheden H_0 en H_∞ , de lengte van het bericht L en de redundantie D is de fractie echte berichten te stellen op

$$p_k = \frac{2^{L.H_\infty}}{2^{L.H_0}} = 2^{-L.D} \quad (7.79)$$

Hierbij valt echter nog een complicatie te noteren. De goede sleutel vormt een uitzondering op de zojuist berekende p_k . Die heeft per definitie een kans $p = 1$, terwijl alle andere een kans 2^{-LD} hebben. Ontsnapping aan deze moeilijkheid is mogelijk door de goede oplossing voorlopig uit te sluiten. Vandaar dat eerst met de gemodificeerde stochast $\underline{M}' = \underline{M} - 1$ wordt gewerkt. Van deze grootheid zal nu de kansverdeling worden bepaald.

* Deze aanname geldt indien er geen verdere aanwijzingen zijn op grond waarvan oplossingen kunnen worden uitgesloten. Bekendheid met de bron kan extra informatie opleveren om te onderscheiden tussen de verschillende oplossingen.

7.8.1 Kansverdeling oplossingen

Gezocht wordt de kansverdeling van $p(\underline{M}' = m')$ om daarna door substitutie van $m' = m - 1$ de gewenste verdeling van $p(\underline{M} = m)$ te vinden. De stochast $\underline{M}'$ volgt een binomiale verdeling met kans p_k voor m' successen bij trekking van sleutels uit $|K| - 1$ (bedenk: de juiste sleutel is weggelaten).

$$p(\underline{M}' = m') = \binom{|K| - 1}{m'} p_k^{m'} (1 - p_k)^{|K| - 1 - m'} \quad (7.80)$$

Voeg nu de in eerste instantie weggelaten goede oplossing weer toe door substitutie van $m' = m - 1$. Dit leidt tot de volgende uitdrukking

$$p(\underline{M} = m) = \binom{|K| - 1}{m - 1} p_k^{m-1} (1 - p_k)^{|K| - m} \quad (7.81)$$

Uitvoering van de sommatie over m

$$\begin{aligned} P &= \sum_{m=1}^{|K|} \frac{1}{m} p(\underline{M} = m) \\ &= \sum_{m=1}^{|K|} \frac{1}{m} \binom{|K| - 1}{m - 1} p_k^{m-1} (1 - p_k)^{|K| - m} \end{aligned} \quad (7.82)$$

Deze uitdrukking kan worden vereenvoudigd door toepassing van de volgende eigenschap van binomiale coëfficiënten

$$\frac{1}{n} \binom{k - 1}{n - 1} = \frac{1}{k} \binom{k}{n} \quad (7.83)$$

Vervolgens wordt opgemerkt dat na invulling hiervan bijna het Binomium van Newton is ontstaan. Om dit compleet te maken moet nog slechts een factor $1/p_k$ buiten de som worden gebracht en een term met $m = 0$ toegevoegd, waarbij die laatste natuurlijk ter compensatie meteen ook weer wordt afgetrokken.

$$P = \frac{1}{|K| \cdot p_k} \left\{ \sum_{m=0}^{|K|} \binom{|K|}{m} p_k^m (1 - p_k)^{|K|-m} - \binom{|K|}{0} p_k^0 (1 - p_k)^{|K|} \right\} \quad (7.84)$$

Na toepassing van het Binomium van Newton resteert

$$P = \frac{1}{p_k |K|} \left(1^{|K|} - (1 - p_k)^{|K|} \right) \quad (7.85)$$

Voor het gemak wordt de afkorting $p_k |K| = \mu$ geïntroduceerd

$$P = \frac{1 - \left(1 - \frac{\mu}{|K|} \right)^{|K|}}{\mu} \quad (7.86)$$

Omdat als regel het aantal mogelijke berichten veel groter is dan het aantal werkelijke berichten, mag veilig worden aangenomen dat p_k klein is. Dan kan met de Taylor benadering van $\ln(1 - x) \approx -x$ voor $-1 \leq x \leq 1$ worden vereenvoudigd en komt er

$$\ln \left(1 - \frac{\mu}{|K|} \right)^{|K|} = |K| \ln \left(1 - \frac{\mu}{|K|} \right) \approx -\mu \quad (7.87)$$

Zodat

$$\left(1 - \frac{\mu}{|K|} \right)^{|K|} \approx e^{-\mu} \quad (7.88)$$

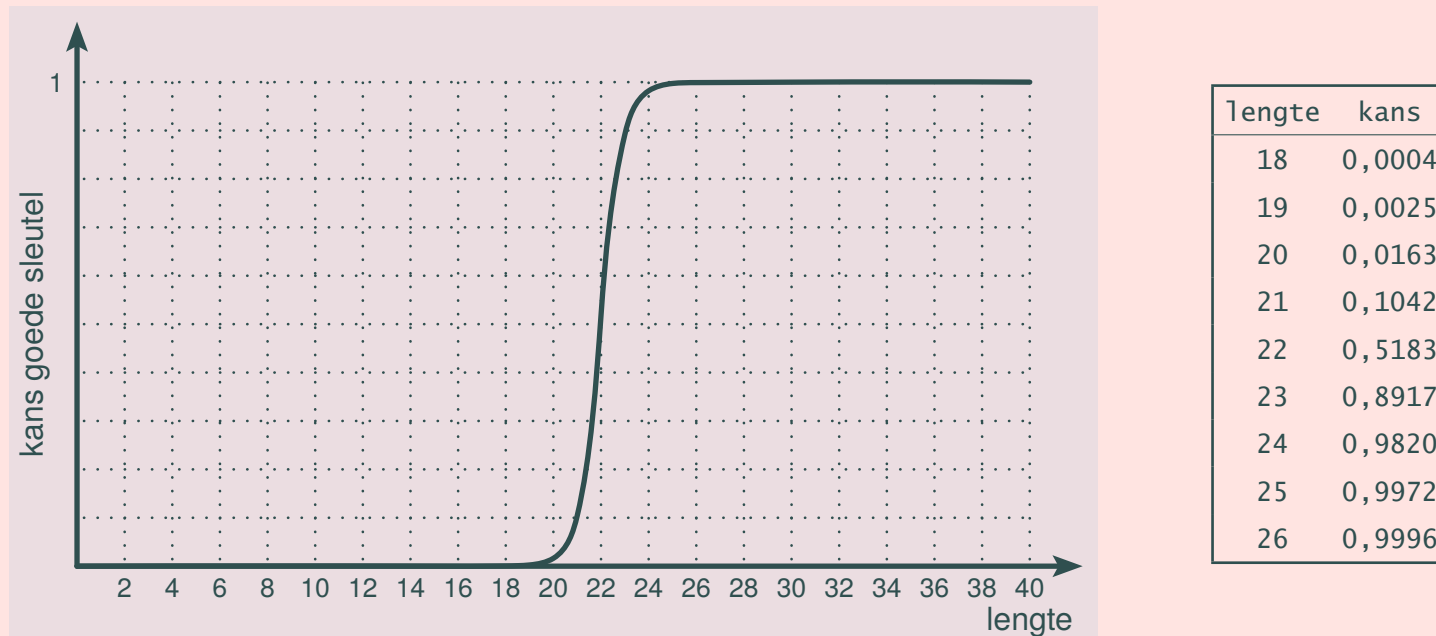
En tenslotte

$$P \approx \frac{1 - e^{-\mu}}{\mu} \quad \text{met} \quad \mu = p_k |K| \quad (7.89)$$

7.8.2 Voorbeeld slaagkans

In [figuur 7.7](#) is het verloop van de kans op de goede sleutel uit [formule \(7.89\)](#) uitgezet voor $H_0 = 4,7$ en $H_\infty = 2,0$ en $|K| = 26!/12! = 8,4 \times 10^{17}$. Tezamen zijn dit ongeveer de karakteristieken voor de monoalfabetische

substitutie van een korte tekst. Let op de scherpe overgang tussen $p = 0$ en $p = 1$ die plaatsvindt in het gebied tussen 20-25 letters. In de tabel staan de berekende waarden in dit overgangsgebied. De uniciteitsafstand blijkt inderdaad een redelijk begrensde grootte.



Figuur 7.7 Verloop van $P = (1 - e^{-\mu})/\mu$

Door de uniciteitsafstand te definiëren als die waarde waarbij meer dan 50% kans bestaat op het vinden van de goede sleutel, kan deze grens bij $\mu = 1$ gelegd worden. Immers dan is $P = 1 - e^{-1} = 0,63$.

$$\lim_{\mu \rightarrow 0} \frac{1 - e^{-\mu}}{\mu} = 1 \quad (7.90)$$

Cryptoanalyse wordt kansrijker naarmate μ meer naar 0 toegaat. Vanwege $p_k \cdot 2^{-LD}$ is dat het geval wanneer L maar groot genoeg is. Anderzijds kan cryptoanalyse lastiger gemaakt worden door het kleiner worden

van μ te vertragen. Dit laatste kan worden bewerkstelligd door $|K|$ groot te maken, dus door een systeem met meer sleutels. Bovendien helpt het om p_k (het gedeelte zinvolle berichten in het totaal aantal mogelijke berichten) te vergroten en gezien formule (7.79) kan het verminderen van de redundantie $D = H_0 - H_\infty$ door compressie hierbij helpen.

Het is eveneens mogelijk om in de berekening niet de kansverdeling voor het aantal goede sleutels $p(\underline{M}=m)$ mee te nemen, maar de overeenkomstige kansverdeling voor het aantal betrouwbaar lijkende klaartekstoplossingen. Het resultaat is een soortgelijke uitdrukking als voor het vinden van de sleutel. Deze verschilt slechts in een vermenigvuldingsfactor met de reeds gevonden uitdrukking. Een factor die bovendien voor toenemende berichtlengte de waarde 1 nadert. Dit gedrag is overigens consistent is met hetgeen al eerder gezien is, namelijk dat $H(M^L | C^L)$ tot $H(K^L | C^L)$ nadert voor grote L zoals in figuur 7.4.

7.9 Referenties

- [1] C. E. Shannon (1948). A Mathematical Theory of Communication. *Bell Technical Journal*, 27, 379–423, 623–656.
C. E. Shannon (1949). Communications Theory of Secrecy Systems. *Bell Technical Journal*, 28, 656–715.
- [2] A. I. Khinchin (1957). *Mathematical Foundations of Information Theory*. Dover Publications, Inc., New York.
- [3] T. M. Cover and J. A. Thomas (1991). *Elements of Information Theory*. John Wiley & Sons, Inc. p.31, 33.
- [4] J. C. A. van der Lubbe (1998). *Basic Methods of Cryptography*. Cambridge University Press.
- [5] C. A. Deavours (1977). Unicity Points in Cryptanalysis. *Cryptologia*, 1(1).
- [6] D.R.L. Brown (2011). Formally Assessing Cryptographic Entropy. <http://eprint.iacr.org/>, 2011/659.

8

Moderne systemen

8.1	Feistel geheimschrift	305
8.2	Data Encryption Standaard	310
8.2.1	Voorgeschiedenis	310
8.2.2	Data Encryption Algoritme	313
8.2.3	Drievoudig DES	320
8.3	Cryptoanalyse van DES	322
8.3.1	Uitputtend zoeken	325
8.3.2	Tijd-geheugen compromis	328
8.3.3	Differentiële cryptoanalyse	331
8.3.4	Differentiële cryptoanalyse verklaard	332
8.3.5	Modificaties van DES	348
8.3.6	Ontwerpcriteria van DES	350
8.3.7	Lineaire cryptoanalyse	354
8.3.8	Lineaire cryptoanalyse verklaard	355
8.3.9	Side-Channel-analyse	362
8.4	Skipjack	364
8.5	IDEA	372
8.6	RC5	374
8.7	Andere geheimschriften	377
8.8	Advanced Encryption Standaard	378
8.8.1	Evaluatie door NIST	381

8.8.2	Overige finalisten	386
8.8.3	Winnaar Rijndael	389
8.9	Referenties	400

In dit hoofdstuk wordt om te beginnen aandacht geschonken aan een algemeen principe, het zogenaamde *Feistel-geheimschrift*, dat aan veel moderne geheimschriften ten grondslag ligt. Vervolgens wordt de *Data Encryption Standard (DES)* vrij uitvoerig behandeld, zowel wat betreft werking alsmede de (vele) pogingen tot cryptanalyse ervan en de controverse over de betrouwbaarheid waartoe de ontwikkelingsgeschiedenis aanleiding heeft gegeven. De Data Encryption Standard is inmiddels geen standaard meer, maar blijft van interesse vanwege z'n structuur en de ontwikkelingen op het gebied van de moderne cryptografie en cryptanalyse waartoe dit algoritme aanleiding heeft gegeven. Na beknopte informatie over enkele andere geheimschriften, wordt besloten met diens opvolger, de *Advanced Encryption Standard (AES)*.

8.1 Feistel geheimschrift

Het ontwerpen van een sterk en goed bruikbaar geheimschrift is geen sinecure. Vooral niet wanneer versleutelen ook nog eens snel moet kunnen geschieden en dan niet alleen op supercomputers, maar ook op betrekkelijk trage smartcards. Dat is de voornaamste reden waarom moderne geheimschriften worden opgebouwd uit eenvoudige basisoperaties zoals de exclusive or.

Hoewel eenvoudige operaties door computers razendsnel kunnen worden uitgevoerd, schuilt er bij een aantal van deze operaties voor een cryptografische toepassing een venijnig addertje onder het gras. De reden hiervan is het feit dat een operator zoals de exclusive or een *lineaire operator* is. En met alleen maar lineaire operatoren kan geen sterk cryptosysteem worden gemaakt. Van stelsels lineaire vergelijkingen is bekend is dat ze eenvoudig op te lossen zijn en als zulke stelsels voor de cryptanalyse kunnen worden ingezet, is breken van het geheimschrift bij wijze van spreken kinderspel. Om tot een sterk geheimschrift te komen moet daarom *niet-lineariteit* worden ingebouwd. Het is de reden waarom de moderne cryptosystemen in dit hoofdstuk bijna allemaal zo ingewikkeld in elkaar zitten.

De zwakte van de exclusive or is vooral evident in het licht van de eigenschap

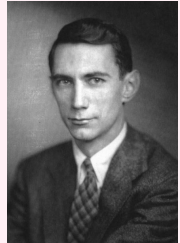
$$a \oplus b \oplus a = b \quad (8.1)$$

Het effect van het maskeren van b door een exclusive or met a , wordt teniet gedaan door a nogmaals toe te passen.

Hoe kan met zulke eenvoudige bouwstenen dan toch een sterk geheimschrift worden gemaakt? Een eerste poging zou kunnen zijn: herhaal een eenvoudige en wellicht niet zo sterke basisvercijfering een aantal malen. Men mag dan hopen dat met genoeg iteraties van een basisvercijfering uiteindelijk een sterker geheimschrift ontstaat. Maar dat herhaald vercijferen niet zonder meer een verbetering geeft, blijkt uit het tegenvoorbeeld van de monoalfabetische substitutie. Hoe vaak ook herhaald, deze levert nooit iets beters dan een enkelvoudige monoalfabetische vercijfering*.

In zijn befaamde artikel over geheime communicatie heeft Claude Shannon de weg gewezen naar een goed principe voor geheimschriftconstructie^[1]. Het gaat bij een geheimschrift om twee dingen, zegt hij. Het ene is het wegpoetsen van de statistische kenmerken van de klaartekst. Hoe minder verschil in voorkomen tussen de letters, hoe minder houvast voor statistische analyse. Het tweede oogmerk is voorkomen dat een eenvoudige relatie bestaat tussen klaartekst, sleutel en cijfertekst. Dat gebeurt vooral als de verzameling sleutels in deelverzamelingen uiteenvalt; een voorbeeld hiervan is sectie ?? gegeven bij de inleidende behandeling van Methoden van cryptoanalyse.

Dit gevaar kan worden bezworen door andersoortige, met name *niet-commuterende* operaties te combineren[†], bijvoorbeeld een substitutie met een transpositie. Het in de Eerste Wereldoorlog in 1918 door de Duitsers gebruikte ADFGVX is een voorbeeld van een klassiek systeem waarin een substitutie en een transpositie gecombineerd zijn. Het niet-commuteren is een essentiële vereiste. Immers, in het geval van commuterende operaties zou men een aanval zo kunnen arrangeren, dat de voor cryptoanalyse meest kwetsbare onderdelen het eerst worden aangevallen. De combinatie van een substitutie met een transpositie vormt een goede bouwsteen voor een sterk geheimschrift en wordt in nagenoeg alle moderne



Shannon

* Monoalfabetische vercijfering is immers een idempotent cryptosysteem, zie [sectie 1.7.4](#).

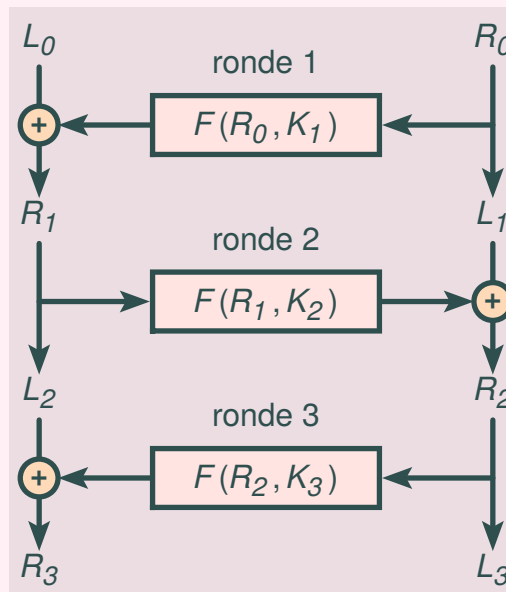
† Commuteren van cryptosystemen is behandeld in [sectie 1.7.7](#).

geheimschriften aangetroffen. Herhaald uitvoeren van zo'n basiscombinatie blijkt een wezenlijk sterker resultaat te kunnen opleveren.

De volgende vraag ligt nu voor de hand: hoe moeten de operaties worden gecombineerd om behalve een eenvoudige manier van vercijferen ook een eenvoudige manier van ontcijferen te verkrijgen? Meestal vergt ontcijfering een andere gang van zaken dan vercijfering, zoals bijvoorbeeld bij de Vigènere het geval is. Het is de verdienste van Horst Feistel dat hij een schema heeft bedacht waarbij een aparte operatie voor de ontcijfering niet nodig is en de vercijferoperatie, zij het met een geringe aanpassing, ook kan worden gebruikt voor het ontcijferen. Geheimschriften van deze soort heten daarom *Feistel-geheimschrift*^{*}.



Feistel



Figuur 8.1 Feistel netwerk

* Horst Feistel was lid van het IBM-team dat de DES heeft ontworpen. Het Feistel-principe is voor het eerst gebruikt in het door hem daarvoor reeds ontworpen Lucifer geheimschrift.

In **figuur 8.1** wordt het principe uit de doeken gedaan met een uit drie herhalingen (rondes) van de basisstap bestaande vercijfering als voorbeeld. Het te vercijferen blok wordt om te beginnen in twee gelijke delen gesplitst, een linker- en een rechterhelft, respectievelijk aangeduid met L_0 en R_0 .

Functie F heeft als argumenten de rechterhelft en een voor elke ronde van de vercijfering steeds wisselende sleutel: een zogenaamde *subsleutel*. Zo'n subsleutel wordt afgeleid van de ingevoerde vercijfersleutel. Het resultaat van $F(R_i, K_{i+1})$ wordt middels een exclusive or gecombineerd met de linkerhelft: $L_i \oplus F(R_i, K_{i+1})$. Dit wordt de nieuwe rechterhelft R_{i+1} . De oude rechterhelft is onveranderd doorgeschoven naar de volgende ronde als nieuwe linkerhelft: $L_{i+1} = R_i$.

In de figuur is de hieruit voortvloeiende voortdurende links/rechts-verwisseling te volgen in de verticale stromen. De figuur laat drie zogenaamde *rondetransformaties* zien. Voor het ontcijferen gelden precies dezelfde operaties, maar de sleutels lopen dan in omgekeerde volgorde.

In formule luiden de drie achtereenvolgende vercijfer- en ontcijferoperaties

$L_1, R_1 = R_0, L_0 \oplus F(R_0, K_1)$	$L_2, R_2 = R_3, L_3 \oplus F(R_3, K_3)$
$L_2, R_2 = R_1, L_1 \oplus F(R_1, K_2)$	$L_1, R_1 = R_2, L_2 \oplus F(R_2, K_2)$
$L_3, R_3 = R_2, L_2 \oplus F(R_2, K_3)$	$L_0, R_0 = R_1, L_1 \oplus F(R_1, K_1)$
<i>vercijfering</i>	<i>ontcijfering</i>

Met de verwisseling van de linker- en rechterhelft, die in elke stap plaats heeft, moet rekening worden gehouden bij de aansluiting van de ontcijfering op de vercijfering. Daartoe kan zonodig aan het eind van de vercijfering de laatste verwisseling weggelaten worden, dan wel voorafgaand aan de ontcijfering worden ingevoegd.

Om aan te tonen dat deze procedure inderdaad de juiste ontcijfering oplevert, wordt in de eerste stap van de ontcijfering het resultaat van de laatste vercijferstap gesubstitueerd. De ontcijferde helften zijn met

een accent gemarkeerd. L'_0 is dus de linkerhelft in de eerste ontcijferstap en gelijk aan de rechterhelft van de laatste vercijferstap R_3 , enz. De ontcijfering van de laatste vercijferronde gaat als volgt

$$\begin{aligned} L'_0 &= R_3 \rightarrow L_2 \oplus F(R_2, K_3) \\ R'_0 &= L_3 \rightarrow R_2 \\ L'_1 &= R'_0 \\ &= R_2 \\ R'_1 &= L'_0 \oplus F(R'_0, K_3) \\ &= L_2 \oplus F(R_2, K_3) \oplus F(R_2, K_3) \\ &= L_2 \end{aligned} \tag{8.2}$$

In de allerlaatste stap is gebruik gemaakt van $x \oplus x = 0$ voor alle x . Aldus is de laatste vercijferronde ongedaan gemaakt en dit proces kan worden herhaald totdat de gehele ontcijfering een feit is.

Het klinkt wellicht verrassend, maar elke willekeurige vercijferfunctie kan worden gebruikt voor het maken van een Feistel-geheimschrift. De operatie behoeft zelfs niet eens omkeerbaar te zijn. Het Feistel-mechanisme garandeert dat het resultaat van de vercijfering weer kan worden ontcijferd. Wel moet men zich afvragen of het resultaat random genoeg is. Aangetoond is dat een Feistel-vercijfering van tenminste drie ronden daaraan al kan voldoen^[2]. Door meer vercijferstappen of ronden in te bouwen kan de sterkte van het geheimschrift tot op zekere hoogte worden vergroot en vaak neemt men meer stappen dan strikt nodig is. De hierdoor gecreëerde veiligheidsmarge fungeert als *overwaarde*, die bescherming moet bieden tegen toekomstige ontwikkelingen in de cryptanalyse.

De invoer is hier gesplitst in een gelijk linker- en rechterdeel en daarom spreekt men van een *gebalanceerd Feistel-geheimschrift*. Het feit dat in elke ronde van een gebalanceerd Feistel-geheimschrift niet meer dan de helft van de bits wordt bewerkt, betekent dat een schema van n ronden voor het totale klaartekstblok effectief neerkomt op slechts de helft ervan. In andere varianten wordt de invoer in ongelijke delen verdeeld

en spreekt men van een ongebalanceerd Feistel-geheimschrift. Een voorbeeld daarvan is het door de Amerikaanse National Security Agency ontworpen Skipjack-algoritme (sectie 8.4). In de meeste literatuur over deze soort geheimschriften ligt het accent echter op gebalanceerde systemen.

Feistel-geheimschriften zijn niet immuun voor cryptoanalyse. Er zijn methoden ontwikkeld waarmee speciale effecten in de klaartekst kunnen worden getraceerd over een groot aantal rondetransformaties. Het aan het gezicht onttrekken van de data die het rondeschema ingaan en uitkomen versterkt daarom zo'n geheimschrift in niet geringe mate. Men doet dit door de klaartekst op een speciale manier met (een deel van) de sleutel te bewerken voordat deze het rondeschema ingaat en evenzo aan het eind ervan. Deze techniek wordt het *witten* (whitening in het Engels) genoemd. Ze kan worden vergeleken met het ondoorzichtig maken van vensterruiten met een laagje witkalk. Het witten onttrekt de invoer en uitvoer van het rondeschema aan het gezicht van de cryptoanalist. Een ingewikkelde operatie is als regel niet nodig, meestal een exclusieve or met een subsleutel.

$$M \oplus K_{in} = M' \rightarrow M' \dots \text{rondeschema} \dots C' \rightarrow C' \oplus K_{uit} = C \quad (8.3)$$

8.2 Data Encryption Standaard

8.2.1 Voorgeschiedenis

In het begin van de zeventiger jaren, midden in de Koude Oorlog, begon men zich in de Verenigde Staten ernstig zorgen te maken over de beveiliging van (computer)gegevens. Niet alleen met betrekking tot militaire gegevens en andere staatsgeheimen, maar vooral ten aanzien van de gegevensverwerking in het bedrijfsleven. Met de toegenomen mogelijkheden voor het aftappen van informatie namen ook de mogelijkheden voor economische spionage toe. Het zal geen verbazing wekken dat men de Russische ambassade in Washington ervan verdacht op dit terrein actief te zijn. Bij economische spionage gaat het

niet slechts om het inlopen van een technologische achterstand. Ook het bepalen van de industriële capaciteit van de tegenstander is buitengewoon belangrijk met het oog op een potentieel militair conflict. Die capaciteit is immers medebepalend voor het in het veld kunnen houden van een effectieve strijdmacht^[3]. Tevens begonnen banken en grote financiële instellingen, waaronder creditcard-maatschappijen, aandacht te schenken aan de noodzaak hun transacties afdoende te beschermen.

Het is daarom niet verwonderlijk dat in kringen van de Amerikaanse regering de gedachte rees om een goed en in het bijzonder op gebruik door het bedrijfsleven toegesneden geheimschrift te laten ontwerpen. Logischerwijze zou deze taak hebben moeten toevallen aan de National Security Agency (NSA), maar om niet geheel duidelijke redenen zag men daar vanaf*. Wat toen gebeurde is nogal ongebruikelijk voor het tot op dat moment sterk gesloten wereldje waarin cryptografie werd bedreven. Er verscheen op 15 mei 1973 een algemeen gestelde, openbare uitnodiging van het National Bureau of Standards (NBS). Een afdeling van het U.S. Department of Commerce, die thans bekend staat onder de naam National Institute of Standards and Technology (NIST)[†]. Hierin werd gevraagd om voorstellen te doen voor "cryptographic algorithms for protection of computer data during transition and dormant storage." Uit de motivering worden twee alinea's aangehaald die duidelijk maken wat voor ogen stond.

"Over the last decade, there has been an accelerating increase in the accumulations and communication of digital data by government, industry and by other organizations in the private sector. The contents of these communicated and stored data often have very significant value and/or sensitivity. It is now common to find data transmissions which constitute funds transfers of several million dollars, purchase or sale of securities, warrants for arrest and conviction records being communicated between law enforcement agencies, airline reservations and ticketing representing investment and

* De NSA houdt zich zowel bezig met de beveiliging van de eigen geheimen door middel van cryptografie, als met het ontcijferen van de cryptogrammen van andere naties. Het is niet ondenkbaar dat de NSA bevreesd was om met een door haar te leveren geheimschrift impliciet teveel informatie over het eigen kunnen en kennen te onthullen.

† Op het internet te vinden op de NIST-site <http://csrc.nist.gov/>.

value both to the airline and passengers, and health and patient care records transmitted among physicians and treatment centers.

The increasing volume, value and confidentiality of these records regularly transmitted and stored by commercial and government agencies has led to heightened recognition and concern over their exposure to unauthorized access and use. This misuse can be in the form of theft or defalcations of data records representing money, malicious modification of business inventories or the interception and misuse of confidential information about people. The need for protection is then apparent and urgent.”

In augustus 1974 herhaalde het National Bureau of Standards de uitnodiging. Uiteindelijk bleek slechts één voorstel, afkomstig van IBM, te voldoen aan de gestelde eisen. Na een uitgebreid onderzoek, waarin ook de National Security Agency een rol heeft gespeeld, werd de uiteindelijke algoritme op 17 maart 1975 door het NBS gepubliceerd als voorgenomen standaard en vervolgens op 15 januari 1977 vastgesteld. Waarna het American National Standards Institute (ANSI) eind 1980 volgde. Zo werd de *Data Encryption Standard* (DES) geboren. Men vindt ook wel de aanduiding *Data Encryption Algorithm* (DEA). Officieel staat deze standaard beschreven in de Federal Information Processing Standards Publication nr. 46 (FIPS PUB 46). Omdat zo'n standaard slechts een geldigheid van vijf jaar heeft, is deze respectievelijk in 1983, 1988, 1993 en 1999 verlengd. De laatste publicatie draagt het nummer FIPS PUB 46-3.

Dat de Data Encryption Standard algemeen gebruikt kon worden is omdat IBM weliswaar de patentrechten behield (om te voorkomen dat onder dezelfde naam allerlei varianten zouden verschijnen), maar het gebruik niet-exclusief en vrij van royalties beschikbaar heeft gesteld. Althans, dat gold voor de Verenigde Staten. Aangezien cryptografie aldaar valt onder de werking van wetten die de vrije export verhinderen, mochten producten met de DES niet zonder vergunning worden geëxporteerd. Maar de algoritme is openbaar gemaakt en dus hebben producenten buiten de USA, waar ze niet aan die wetten zijn onderworpen, voor alternatieve implementaties gezorgd*. Inmiddels is de DES aan het eind van zijn bruikbare levens-

duur gekomen en zijn de restricties verzacht, maar daarover later meer.

8.2.2 Data Encryption Algoritme

De Data Encryption Standard is een blokgeheimschrift van het Feistel-type met een blok grootte van 8 bytes ofwel 64 bits. Bovenaan in de linkerhelft van [figuur 8.2](#) ziet men het 64-bit invoerblok. Encryptie van kleinere hoeveelheden informatie verlangen bijgevolg een aanvulling (padding) tot 64 bits*.

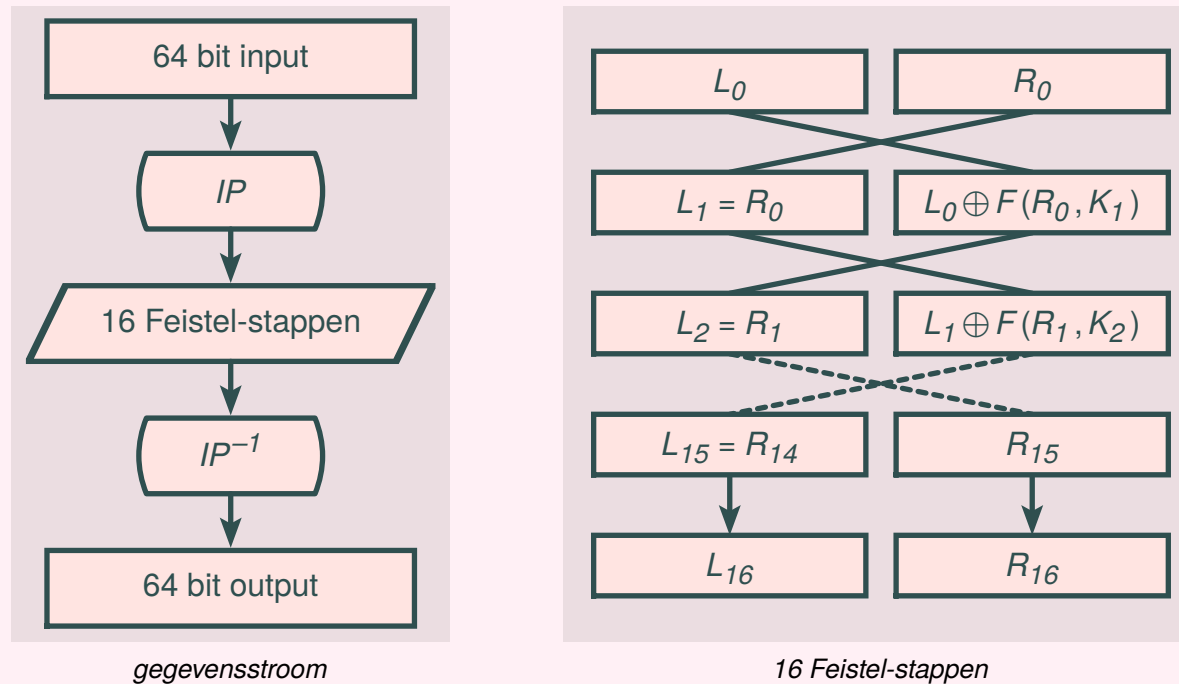
Direct aan het begin worden de ingevoerde bits aan een transpositie onderworpen, de invoerpermutatie *IP*. De rol van deze permutatie is niet geheel duidelijk. Ze is niet geheim en ook onveranderlijk. Er is weleens gesuggereerd dat deze dient om structuur verderop in de algoritme te verhullen. Waarschijnlijker is dat men hiermee de constructie van een DES-chip heeft willen vergemakkelijken. Deze permutatie heeft tot gevolg dat de pariteitsbits[†] van de invoer bijeen gebracht worden in het eerste octet (achtal bits) van de gepermuteerde invoer.

Conform het Feistel-mechanisme wordt de invoer gesplitst in een linker- en een rechterhelft van elk 32 bits. DES is dus een gebalanceerd Feistel-geheimschrift. De helften doorlopen 16 maal achtereen een

* Met *standaard* werd aanvankelijk bedoeld op een hardware implementatie. Dat had kunnen betekenen dat chipfabrikanten door ondertekening van een non-disclosure overeenkomst de beschikking krijgen over de details nodig om de chips te maken (zoals later bij de Clipper chip en ook bij de A5-algoritmen voor gsm-telefonie, wel is gebeurd). De DES-algoritme is in de gepubliceerde standaard echter zo gedetailleerd beschreven, dat implementatie in software eveneens mogelijk werd. Men heeft wel de veronderstelling geuit dat die mate van openheid NSA niet voor ogen heeft gestaan. Het resultaat werd nu een door de professionals van NSA gekwalificeerd algoritme, dat alleen daarom al vele creatieve cryptoanalisten uit de gehele wereld heeft aangetrokken – als motten naar een kaarsvlam. De DES heeft dan ook – wellicht ongewild – tot veel voortuitgang op het gebied van de cryptoanalyse geïnspireerd.

* Naïef aanvullen met 0-bits is fout. Als een cryptoanalist het geluk heeft een blok tegen te komen dat met 7 bytes nullen is uitgevuld dan zijn er niet meer dan 256 mogelijkheden voor de klaartekst van het blok.

† Het pariteitsbit in een byte van 8 bits is standaard het meest linker en hoogste bit. Het dient als een controle op de 7 andere bits en was van belang in de tijd dat de overdracht van informatie veel meer onderhevig was aan fouten dan nu.



Figuur 8.2 Hoofdschema van de DES

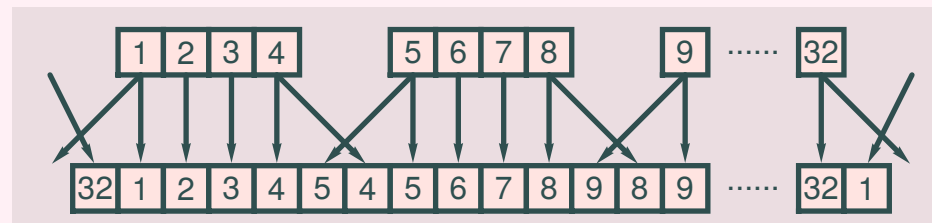
rondetransformatie, de 16 Feistel-ronden van de DES-vercijfering in de rechterhelft van [figuur 8.2](#). Bij deze rondetransformaties wordt voor elke ronde een andere sleutel gebruikt. Deze 16 subsleutels worden alle afgeleid uit de eigenlijke sleutel, waarover straks meer.

In elke ronde worden de linker- en rechterhelft verwisseld. De rechterhelft blijft ongewijzigd en wordt de linkerhelft van de volgende ronde. De nieuwe rechterhelft wordt de exclusive or van de linkerhelft uit de vorige ronde met de vercijferde rechterhelft $F(R_{i-1}, K_i)$. In de eerste 15 ronden wordt de verwisseling uitgevoerd, maar in de 16^e blijft deze achterwege. Het achterwege laten van de verwisseling in de laatste ronde betekent dat de linker- en rechterhelft in de juiste volgorde klaarstaan voor ontcijfering.

De laatste stap is weer een transpositie en wel de inverse van de ingangspemutatie. Als gevolg van die keuze valt deze bij de ontcijfering weg tegen de ingangspemutatie $IP^{-1}IP = I$. Het enige wat bij het ontcijferen nog wel veranderen moet is de volgorde van de subsleutels. Bij het vercijferen lopen deze van K_1 naar K_{16} en bij het ontcijferen terug van K_{16} naar K_1 .

In elke ronde wordt de rechterhelft gecombineerd met een uit de gehele sleutel gemaakte subsleutel en vervolgens gesubstitueerd en gepermuteerd. De uitwendige sleutel wordt opgegeven als een rij van 8 bytes, maar van de 64 bits daarin worden de pariteitsbits niet gebruikt. De werkelijk benutte sleutel heeft dus een omvang van 56 bits. DES heeft daarom $2^{56} = 72.057.594.037.927.936$ sleutels.

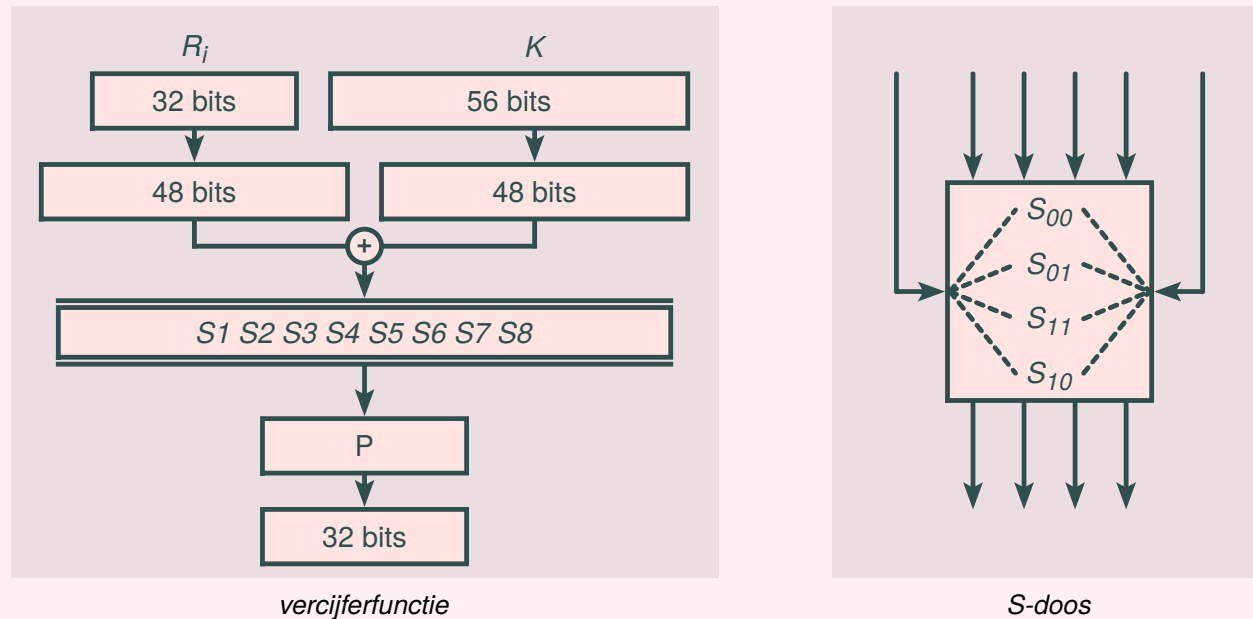
In elke ronde wordt uit de 56 sleutelbits (64 oorspronkelijke sleutelbits minus de 8 pariteitsbits) een subsleutel van 48 bits gekozen. Deze 48-bits subsleutel moet nu worden gecombineerd met de 32-bits rechterhelft. Daarom wordt deze rechterhelft van 32 naar 48 bits uitgebreid. Die expansie is afgebeeld in [figuur 8.3](#). Denk hiervoor de 32 bits opgedeeld in acht groepjes van vier bits. Tussen deze groepjes worden steeds twee bits door verdubbeling ingevoegd. De gedupliceerde bits komen van de eerste en de laatste positie in elk groepje. Ze worden zo doorelkaar gevlochten dat het laatste bit van het vorig groepje wordt voorafgegaan door het eerste bit van het volgend groepje. Bit 1 en 32 komen respectievelijk achteraan en vooraan.



Figuur 8.3 Expansie van 32→48 bits

Het tot 48 bits geëxpandeerde blok wordt middels een exclusive or gecombineerd met de subsleutel; dat is de $\oplus$ in het linkerdeel van [figuur 8.4](#). De nu 48 bits moeten vervolgens weer teruggebracht worden

tot 32 bits, omdat anders het blok in elke ronde met 16 bits zou groeien. Deze reductie wordt op een handige manier gecombineerd met een substitutie. Er volgt daartoe een verdeling van de 48 bits in acht groepjes van zes. Elk groepje wordt toegevoerd aan een substitutie-eenheid, de zogenaamde *S*-doos. Er gaan zes bits een *S*-doos binnen en er komen vier bits uit.



Figuur 8.4 Rondeformatie

Elke *S*-doos (rechterkant van [figuur 8.4](#)) bevat vier verschillende substituties waaruit kan worden gekozen. De acht *S*-dozen zijn allemaal van elkaar verschillend. De keuze voor één van de vier substituties in de *S*-doos wordt gemaakt door twee van de zes invoerbits te bestempelen tot adresbits voor de selectie. Men heeft hiervoor het eerste en laatste bit van het groepje van zes genomen, juist de extra bits uit de expansie. De vier middelste bits worden door de geselecteerde substitutie vervangen door vier andere bits.

De 32 bits die uit de gezamenlijke S -dozen komen worden vervolgens door permutatie P verwisseld. Deze verwisseling heeft vooral ten doel te zorgen dat de bits in de volgende ronde in andere S -dozen terecht komen. Bovendien worden ze daardoor de volgende keer met andere bitposities in de subsleutel gecombineerd. Deze werkwijze maakt dat elk bit snel naar alle andere plaatsen diffundeert. Het schema is zo efficiënt dat na drie keer een verandering in de invoer van één S -doos zich al over alle acht S -dozen heeft verspreid. Omdat in het rondeschema linker- en rechterhelft elkaar afwisselen, is dat dus na zes ronden het geval.

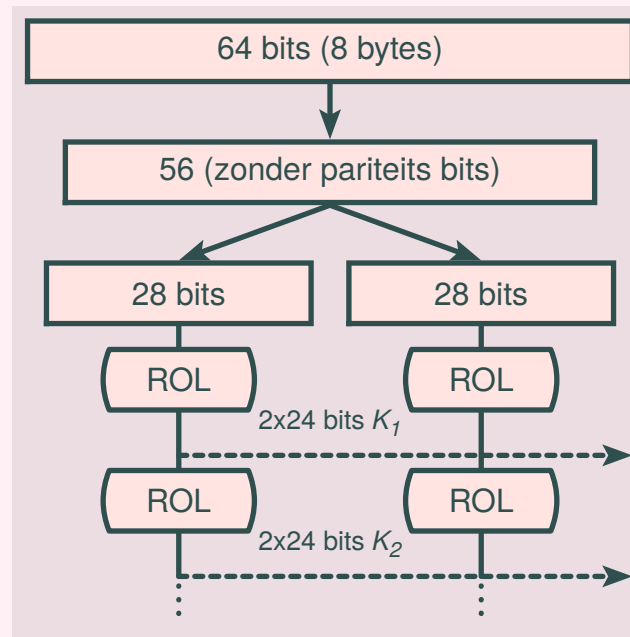
Uit het feit dat voor de combinatieoperator in het algemeen schema en in de rondetransformaties alleen de exclusive or wordt gebruikt, volgt dat de DES-vercijfering de volgende eigenschap heeft. Vercijfer een blok met een bepaalde sleutel. Neem nu het complement van alle bits in het oorspronkelijke blok en in de sleutel (alle 0'en worden een 1 en omgekeerd) en voer daarmee opnieuw de vercijfering uit. Het blijkt dat dan de twee vercijferde resultaten precies elkaars complement zijn.

$$E(M, K) = \overline{E(\overline{M}, \overline{K})} \quad (8.4)$$

Vercijferen van klaartekstblok M met sleutel K resulteert in het complement van vercijferen van het complement $\overline{M}$ met sleutelcomplement $\overline{K}$.

Dit heet de *complementariteitseigenschap* van de DES. Ze heeft tot gevolg dat het afzoeken van alle sleutels niet nodig is, maar dat met de helft kan worden volstaan. De andere helft volgt eenvoudig uit het verwisselen van de 0'en en de 1'en. Zo wordt het in een brute-force-aanval uit voeren aantal ontcijferoperaties tot de helft terug gebracht.

De werkelijke complicatie in dit geheimschrift zetelt in de constructie van de S -dozen. De substituties in de S -dozen blijken cruciaal voor de veiligheid van de DES. Er mag zeker geen lineaire relatie tussen invoer- en uitvoerwaarde van de S -doos bestaan. Bij de discussie over de veiligheid en cryptoanalyse van de DES wordt op dit punt nog teruggekomen.



Figuur 8.5 Substitutieschema

Tot slot van de beschrijving van de Data Encryption Standard volgt de manier waarop de 16 subsleutels worden gekozen uit de bits van de sleutel (zie [figuur 8.5](#)). De eerste stap is het weglaten van de 8 pariteitsbits uit de 8 bytes van de ingevoerde sleutel. De geselecteerde 56 bits worden gesplitst in twee groepjes van elk 28 bits, die worden geplaatst in twee registers. In de navolgende 16 ronden wordt de inhoud van die registers iedere keer linksom geroteerd (ROL in de figuur). Telkens worden uit elk van die beide registers 24 bits afgetapt, die samen genomen de 48-bits subsleutel vormen. De rotatie is niet helemaal regelmatig. Deze onregelmatigheid in het rotatieschema voor het genereren van de subsleutels pareert een aanval gebaseerd op een groep van verwante sleutels (een zogenaamde *related key attack*^[4]). Het stappenschema doet eerst tweemaal een 1-bit rotatie naar links, dan zesmaal een 2-bit rotatie, enz. Het complete rotatieschema luidt

1 1 2 2 2 2 2 2 1 2 2 2 2 2 1

Dit is het schema voor het vercijferen. Bij het ontcijferen moeten de subsleutels in omgekeerde volgorde gemaakt worden. Om dit te bereiken worden de linksrotaties vervangen door rotaties naar rechts en wordt het stappenschema met een kleine aanpassing in omgekeerde volgorde doorlopen.

Er zijn nog enkele bijzonderheden aan dit schema voor sleutelproductie. Bij bepaalde bitpatronen in de sleutel ontstaan periodieke patronen in de 16 subsleutels. Neem de sleutels die in de beide roterende registers òf alleen nullen òf alleen enen geven. Het is duidelijk dat alle 16 subsleutels dan hetzelfde zullen zijn. De vier sleutels die dit verschijnsel vertonen noemt men de *zwakke sleutels* van de DES. Deze zwakke sleutels staan in [tabel 8.1](#) (in hexadecimale notatie). Het is onverstandig om ze te gebruiken, omdat zo'n sleutel bij tweemaal achtereen vercijferen of ontcijferen voor alle mogelijke invoer de oorspronkelijke klaartekst teruggeeft.

$$M = E(E(M, K), K) \text{ en } M = D(D(M, K), K) \quad \forall M \quad (8.5)$$

1 :	01	01	01	01	01	01	01	01
2 :	1F	1F	1F	1F	0E	0E	0E	0E
3 :	E0	E0	E0	E0	F1	F1	F1	F1
4 :	FE	FE	FE	FE	FE	FE	FE	FE

Tabel 8.1 Zwakke DES-sleutels (hexadecimaal)

Behalve de al genoemde zwakke sleutels met alleen nullen of enen in de roterende registers, kunnen ook patronen van periode 2 voorkomen (afwisselend nullen en enen). Deze sleutels heten de *halfzwakke sleutels*. Ze zijn ontdekt door D.W. Davies en W.L. Price van het National Physical Laboratory in Engeland. Eén van beide registers bevat dan een herhaling van het patroon 01 of 10 en het andere 0...0 of 1...1 of

herhaald 01 of herhaald 10. Het gevolg van deze structuur is dat slechts 2 onderscheiden subsleutels geproduceerd worden in plaats van 16. Daardoor ontstaan paren (K_1, K_2) van halfzwakke sleutels met de eigenschap dat achtereenvolgens vercijferen met de sleutels van een paar de oorspronkelijke klaartekst teruggeeft. Er zijn twaalf halfzwakke sleutels, die met elkaar zes paren vormen. Deze twaalf sleutels worden gegeven in [tabel 8.2^{\[5\]}](#). Ook het gebruik van deze sleutels moet worden ontraden. Gegeven het geringe aantal zwakke en halfzwakke sleutels is dat niet zo moeilijk.

$$M = E(E(M, K_2), K_1) \text{ en } M = E(E(M, K_1), K_2) \quad \forall M \quad (8.6)$$

1 : E0 1F E0 1F F1 0E F1 0E	7 : FE 1F FE 1F FE 0E FE 0E
2 : FE E0 FE E0 FE F1 FE F1	8 : E0 1F E0 1F F1 0E F1 0E
3 : 1F FE 1F FE 0E FE 0E FE	9 : FE 01 FE 01 FE 01 FE 01
4 : 01 FE 01 FE 01 FE 01 FE	10 : E0 01 E0 01 F1 01 F1 01
5 : 1F E0 1F E0 0E F1 0E F1	11 : 01 1F 01 1F 01 0E 01 0E
6 : 01 E0 01 E0 01 F1 01 F1	12 : 1F 01 1F 01 0E 01 0E 01

Tabel 8.2 Halfzwakke DES-sleutels (hexadecimaal)

Op dezelfde manier kan men ook herhaalde patronen van 4, 7 en 14 bits in een of beide van de sub-sleutelregisters hebben. Ooit heeft eens iemand geclaimed dat dit leidt tot een zwakte die kan worden geëxploiteerd voor cryptoanalyse, maar nooit is het bewijs geleverd dat dit ook werkelijk het geval is^[6]. Vier van de zwakke sleutels bezitten elk 2^{32} fixed points en vier van de halfzwakke sleutels hebben elk 2^{32} anti-fixed points ([vergelijking \(1.27\)](#)).

8.2.3 Drievoudig DES

In de loop der jaren heeft de chiptechniek flinke vorderingen gemaakt en is de Data Encryption Standard

niet meer voldoende bestand tegen uitputtend afzoeken van alle sleutels. Daarom is *TripleDES* – een drievoudige DES-vercijfering – in de meest recente versie van de DES-standaard als aanbevolen implementatie toegevoegd^{[7]*}.

Wanneer één encryptie de bescherming van een 56-bits sleutel geeft[†], mag men hopen dat herhaalde encryptie met verschillende sleutels een betere bescherming biedt. Dat werkt in dit geval omdat de Data Encryption Standard geen idempotent cryptosysteem is (sectie 1.7.4). Eveneens van belang is een eigenschap uit de wiskunde, namelijk uit de groepentheorie. Als de DES een zogenaamde algebraïsche groep was geweest, dan zou men voor elk paar vercijfersleutels een derde moeten kunnen vinden waarmee dezelfde vercijfering tot stand te brengen is. Herhaald vercijferen, al dan niet met verschillende sleutels, kan in dat geval geen verbetering geven. Onderzoek heeft echter uitgewezen dat de DES geen groep is, zodat TripleDES inderdaad een versterking van de veiligheid betekent^{[8]‡}.

$$\forall K_1, K_2, \exists K_3: E(E(M, K_1), K_2) = E(M, K_3) \quad (8.7)$$

In tegenstelling tot hetgeen men op het eerste gezicht wellicht zou verwachten, levert het achtereenvolgens met twee verschillende sleutels vercijferen toch niet de gewenste versterking. In plaats dat tweemaal achtereenvolgens vercijferen met verschillende sleutels van 56 bits de veiligheid opvoert tot 112 bits, blijkt dit zwaar tegen te vallen. Tweemaal vercijferen biedt niet meer veiligheid dan een sleutel van 57 bits: een nauwelijks lonende verbetering.

De oorzaak van dit tegenvallend resultaat komt doordat men met een bekend klaartekst/cijfertekstpaar een zogenaamde *meet-in-the-middle* aanval kan opzetten. Vercijfer bekende klaartekst met alle

* De organisatie die in Nederland het betalingsverkeer verzorgde – het vroegere Interpay dat inmiddels samengegaan is met een Duitse zustermaatschappij tot het bedrijf Equens – is daarom in het begin van de jaren 2000 gestart met het ombouwen van de pinautomaten in de winkels naar TripleDES. Of een oud pinkastje omgebouwd is, kon worden geconstateerd door dit om te keren: aan de onderzijde zat dan een oranje sticker met **3-DES**

† Effectief eigenlijk maar een 55-bits sleutel als men rekening houdt met de complementariteitseigenschap van de DES.

‡ Zie ook het zuiver cryptosysteem uit sectie 1.7.6.

2^{56} sleutels en ontcijfer de cijfertekst eveneens met alle 2^{56} sleutels. De oplossing is daar waar een overeenkomend tussenresultaat wordt gevonden. Afgezien van de geheugenvereisten is slechts tweemaal zoveel werk nodig als voor een enkele DES-vercijfering en telt de sleutel dus geen 112 maar effectief slechts 57-bits. Een derde encryptie is nodig om deze aanval te verhinderen.

Voor deze drievoudige TripleDES-vercijfering wordt een speciale vorm gekozen. Niet drie achtereenvolgende vercijferingen, maar een ontcijfering ingeklemd tussen twee vercijferingen.

$$C = E_{K_1} D_{K_2} E_{K_3} \text{ en } M = D_{K_3} E_{K_2} D_{K_1} \quad (8.8)$$

De standaard geeft drie mogelijkheden voor het kiezen van de drie sleutels.

1. drie onafhankelijke sleutels
2. één sleutel voor de vercijferingen en een andere sleutel voor de ontcijfering
3. één sleutel voor alle drie de operaties

De keuze voor afwisselend vercijferen en ontcijferen zorgt ervoor dat in het geval van de laatstgenoemde optie compatibiliteit met enkelvoudig DES ontstaat. Zodoende behoeft voor de verwerking van bestaande gegevens geen apart programma te worden gebruikt en kunnen alle operaties geschieden met TripleDES. De speciale sleutelkeuze in de derde optie reduceert dan tot enkelvoudig DES. Een nadeel van TripleDES is dat alle crypto-operaties nu driemaal zoveel tijd vergen en mede om die reden is het werk aan een nieuwe standaard gestart.

8.3 Cryptoanalyse van DES

Nog voordat de DES tot standaard werd verheven, barstte er al een heftige discussie over los. Twijfels werden geuit omtrent de lengte van de sleutel, die men met 56 bits veel te kort vond^[9]. De Data Encryption Standard was immers voortgekomen uit het werk van Horst Feistel aan de Lucifer-algoritme, die een sleutellengte van 128 bits heeft en dat is aanzienlijk meer dan de 56 bits van de Data Encryption Standard.

De argwaan groeide toen – na eerdere ontkenningen – bekend werd dat deze sleutellengte mede op advies van de National Security Agency tot stand gekomen was. Het vermoeden rees dat die 56 bits wel eens net binnen de cryptoanalytische mogelijkheden van de National Security Agency zouden kunnen vallen. Immers, waarom zou zo'n machtige organisatie van codebrekers een algoritme toestaan dat zij niet zelf de baas kunnen? Dat is pas echt jezelf in de voet schieten.

De verdenkingen werden versterkt toen onderzoekers de substituties in de *S*-dozen aan de tand gingen voelen. Het bleek dat dit beslist geen willekeurig gekozen substituties konden zijn. Integendeel, er moest een systematiek achter zitten. Aangezien de enige niet-lineariteit van de *S*-dozen moet komen, mag de constructie van deze substituties cruciaal geacht worden voor de bestendigheid van de algoritme tegen cryptoanalyse. Ook hier bleek de National Security Agency achter de schermen aanwezig – en wat nog meer voedsel aan de verdenkingen van vals spel gaf – de criteria voor het ontwerp van de *S*-dozen waren tot staatsgeheim verklaard en de leden van het ontwerpteam van IBM hielden zorgvuldig hun mond.

Zou er een achterdeur in de *S*-dozen zitten? Een verborgen structuur waardoor cryptoanalyse wordt vergemakkelijkt? Een manier om zoveel bits van de sleutel te vinden, dat de rest binnen de mogelijkheden van uitputtend zoeken valt? Die verdenking kon echter niet hard gemaakt worden. Geconcludeerd werd bijvoorbeeld dat de *S*-dozen gekozen leken te zijn om bij vasthouden van één bit in de invoer een zo gelijk mogelijke verdeling van enen en nullen in de uitvoer te bewerkstelligen. Een conclusie die overigens eerder leek te wijzen op kracht dan op zwakte van de algoritme^[10].

Om het vertrouwen te versterken, werd een hearing gehouden door de U.S. Senaatscommissie voor de Veiligheidsdiensten. Deze commissie zuiverde de National Security Agency van de blaam met het ontwerp van de Data Encryption Standard te hebben geknoeid, hoewel dit de kritiek niet echt deed verstommen^[11].

Samenvatting conclusies Senaatscommissie

- The DES is more than adequate for its intended applications
- IBM invented and designed the DES
- The National Security Agency did not tamper with the design
- The National Security Agency certified the DES free of any known statistical/mathematical weakness
- The National Security Agency recommended that the Federal Reserve Board use the DES for electronic funds transfer applications

De DES werd tot standaard verheven ondanks de twijfels die bleven knagen, in het bijzonder vanwege de geheim gehouden criteria voor de S-dozen. Om die twijfel wat weg te nemen werden enkele van deze criteria vrijgegeven. Maar niet allemaal en dus bleef het wantrouwen. Ironisch genoeg is uiteindelijk gebleken dat die achtergehouden criteria niet op een zwakte van de algoritme slaan, maar juist op de kracht ervan. Bijna 20 jaar lang heeft de Data Encryption Standard alle praktisch uitvoerbare cryptoanalyse weerstaan. Pas aan het eind van de 90'er jaren van de vorige eeuw zijn chip- en computertechniek zover voortgeschreden dat uitputtend zoeken een reële bedreiging is gaan vormen.

Alle twijfels en verdenkingen rond de veiligheid van de DES hebben – ongetwijfeld onbedoeld – de stoot gegeven tot veel nieuwe ontwikkelingen in de cryptoanalyse. In het vervolg worden enkele van zulke cryptoanalytische methoden gepresenteerd met hun effect op de DES. Ze zijn echter niet specifiek voor dit geheimschrift en kunnen ook op andere, soortgelijke geheimschriften worden losgelaten. Vanwege hun belang voor een beter begrip aangaande de kwaliteit van moderne geheimschriften kan een ieder worden aanbevolen zich wat meer in deze methoden te verdiepen; het wiskundig gehalte is soms echter vrij hoog. Enkele van deze methoden zijn

- uitputtend doorzoeken van de sleutelverzameling
- benutten van een time-memory tradeoff

- differentiële cryptoanalyse
- lineaire cryptoanalyse
- metingen aan de fysieke (chip)implementatie, de zgn. side-channel-analyse

Met bovenstaande verzameling is het arsenaal van de cryptoanalist allerm minst uitgeput. Varianten op differentiële cryptoanalyse, combinatie met lineaire cryptoanalyse, aanvallen via verwante paren sleutels, groepen sleutels met minder weerstand tegen een bepaalde methode van cryptoanalyse, enz. Een overzicht uit 1999 (en daarom alweer behoorlijk gedateerd) kan men vinden in een *zelfstudieprogramma cryptoanalyse* geschreven door Bruce Schneier en te vinden op de internetsite van Counterpane Internet Security, Inc.^[12]

8.3.1 Uitputtend zoeken

Vanaf het begin is kritiek geuit op de lengte van de sleutel. Uitputtend afzoeken van een 56-bits sleutel betekent in principe het proberen van 2^{56} sleutels*. In 1974, ten tijde van het ontwerp van de Data Encryption Standard, was dit nog heel veel. Maar ook toen kon al worden voorzien dat de technologische mogelijkheden betrekkelijk snel zouden toenemen. Men moet daarbij bedenken dat het enkele jaren kan duren voordat het ontwerp van een nieuw cryptosysteem voldoende is gevalideerd en tot standaard verheven. Voor de DES heeft die fase geduurd van 1974–1977. Pas daarna komt een geleidelijke invoering in de praktijk van de gegevensverwerking tot stand. Vervolgens moet een standaardsysteem toch minstens een paar decennia kunnen meegaan en ook nadat het is uitgediend zullen de ermee gecijferde gegevens wellicht nog vele jaren ontoegankelijk moeten blijven. Al met al betekent dit dat een aanzienlijke veiligheidsmarge moet worden ingebouwd.

Reeds in 1976 werd door Whitfield Diffie en Martin Hellman de claim van het National Bureau of Standards (NBS) dat "trying all possible keys is not economically feasible" betwist^[13]. Zij beweerden dat met

* Strict genomen is het eigenlijke werk de helft minder vanwege de complementariteitseigenschap van de DES.

de middelen van dat moment voor ongeveer \$20 miljoen een machine kon worden gebouwd waarmee binnen 24 uur een sleutel zou worden gevonden. Veel geld voor die tijd en niet voor iedere organisatie weggelegd. Maar naar algemene overtuiging wel binnen de mogelijkheden van de NSA en haar zusterorganisaties elders in de wereld. In 1993 werd een gedetailleerde beschrijving van een chip met daarop een zoekmachine voor DES-sleutels gepubliceerd^[14].

Hoe snel moet een zoekmachine zijn en hoe snel is de ontwikkeling van de chiptechniek gegaan? Aan een in 1991 verschenen artikel kan een tussenstand worden ontleend^[15]. Het aantal benodigde vercijferingen per seconde kan worden afgeleid uit de combinatie van verlangde zoektijd met het aantal processoren dat parallel kan worden ingezet. De cijfers uit genoemd artikel staan in [tabel 8.3](#). De totale zoekruimte vergt bij benadering 72×10^{15} ontcijferingen, maar dit aantal kan worden gehalveerd door gebruik te maken van de complementariteitseigenschap (zie [vergelijking \(8.4\)](#)). In 1984 haalden DES-chips een snelheid van 256.000 encrypties per seconde en in 1987 is dat ongeveer verdubbeld. Uit de tabel kan men afleiden dat breken van een sleutel binnen een week dan nog altijd een machine met zo'n 250.000 processoren vergt.

tests/sec	aantal processoren nodig			
	1 jaar	1 maand	1 week	1 dag
1.000.000	2300	27400	119200	834000
2.000.000	1150	13700	59600	417000
4.000.000	600	6850	29800	208500
32.000.000	75	850	3700	26100
256.000.000	9	107	500	3300

Tabel 8.3 Zoektijd, snelheid en aantal processoren

Het heeft tot 1998 geduurd voordat op particulier initiatief een effectieve zoekmachine – Deep Crack genaamd – werd gebouwd en beproefd^[16]. Op 13 juli 1998 schreef de firma RSA Data Security een wedstrijd

uit voor het breken van een DES-bericht en loofde \$10.000 uit voor de eerste die dit zou ontcijferen. Na 56 uur zoeken, ongeveer een kwart van de totale zoekruimte, kwam de machine met de oplossing. Omdat deze machine binnen een budget van \$250.000 kon worden gebouwd en zeker nog niet optimaal is, bezegelde die prestatie het definitieve einde van de enkelvoudig Data Encryption Standard als veilige vercijfermethode*.

Het is illustratief om machine en zoektechniek meer in detail te bekijken. Een Application Specific Integrated Circuit (ASIC) is een chip die men op basis van een specificatie vrij goedkoop kan laten maken bij daarin gespecialiseerde bedrijven. In dit geval maakte men een chip met 24 identieke DES-processoren en een kloksnelheid van 40 MHz. Het ontwerp was zodanig dat in elke klokcyclus een rondetransformatie wordt uitgevoerd en elke processor dus 2.500.000 sleutels per seconde kan testen. Uit de tabel kan worden afgelezen dat voor het breken van een sleutel binnen een week bijna 60.000 processoren ofwel 2500 chips nodig zijn. Uiteindelijk kreeg men op 27 circuitborden met elk 64 chips 37.050 goede zoekmachines aan het draaien†. In een etmaal gaan 86.400 seconden en dus valt eenvoudig te berekenen dat een complete zoektocht 9 dagen duurt. Een personal computer houdt alle chips in de gaten, stuurt ze aan en zorgt voor de nacontrole wanneer een chip meent een goede sleutel te hebben gevonden.

De machine kan een cijfertekstaantal uitvoeren door elke kandidaatsleutel als volgt te testen. Een 64-bits blok levert bij goede ontcijfering acht bytes, waarvan mag worden verondersteld dat ze leesbare tekst vormen. Omdat van de 256 bytewaarden slechts een stuk of 70 daarvoor in aanmerking komen (letters, cijfers en wat leestekens), heeft een willekeurige ontcijfering een kans van ongeveer $70/256$ per byte dat een acceptabele waarde verschijnt. Voor het gehele 8-byte blok derhalve een kans van $(70/256)^8$ en een kans van $(70/256)^{16}$ voor twee opeenvolgende blokken met 16 bytes‡. Er moeten 2^{56} sleutels in aanmerking genomen worden, waardoor men ruim 70 miljoen sleutels met een plausibele ontcijfering kan

* Dat geldt niet voor TripleDES, daar komt deze machine zelfs niet bij in de buurt.

† Door uitval konden niet alle 1728 gemonteerde chips meedoen.

‡ Voor een cijfertekstaantal is de uniciteitsafstand van DES ongeveer 16 bytes, voor klaartekst/cijfertekst is dit 8 bytes.

verwachten. Dit aantal is klein genoeg om door de besturende personal computer verder te laten testen. De chip bevat een tabel waarin voor alle waarden 0...255 kan worden aangegeven of deze acceptabel is als ontcijfering of niet. Voor een klaartekst/cijfertekst-aanval komen slechts de acht bytes van de klaartekst als plausibele ontcijfering in aanmerking. In dat geval verwacht men bij het testen van een 64-bits blok ongeveer $2^{56} \times (8/256)^8 = 65.536$ valse oplossingen aan te treffen.

Op het internet kan men informatie vinden over een 'aantrekkelijk' geprijsde machine gebaseerd op FPGA chips, de zogenaamde COPACOBANA*. De naam is afgeleid van *Cost-Optimized Parallel Code Breaker*. Een uitputtende scan van alle DES-sleutels kost hiermee minder dan een week.

In software is het breken van de Data Encryption Standard een ander verhaal. Gespecialiseerde hardware laat software-ontcijfering ver achter zich in snelheid. Desondanks is het ook langs die weg gelukt om de DES in een betrekkelijk beperkt tijdsbestek te breken. Sedert de komst van het internet is het namelijk vrij eenvoudig geworden om een groot aantal machines in te schakelen voor rekenwerk dat goed geparalleliseerd kan worden†. In 1997 lukte het programmeur Rocke Verser om met in vrije tijd ontwikkelde programmatuur vele vrijwilligers te mobiliseren. Ze doorzochten allemaal een stukje van de sleutelruimte en mailden de resultaten naar een centraal punt. Ruim 78.000 vrijwilligers deden mee en op topdagen waren wel 14.000 machines tegelijk bezig. Op 18 februari 1997 werd het project gelanceerd en op 17 juni kon de oplossing worden ingeleverd. Ook deze gebeurtenis maakte duidelijk dat de enkelvoudige Data Encryption Standard z'n tijd heeft gehad.

8.3.2 Tijd-geheugen compromis

De eerste serieuze aanval op de Data Encryption Standard kwam van Martin Hellman, criticus van de DES

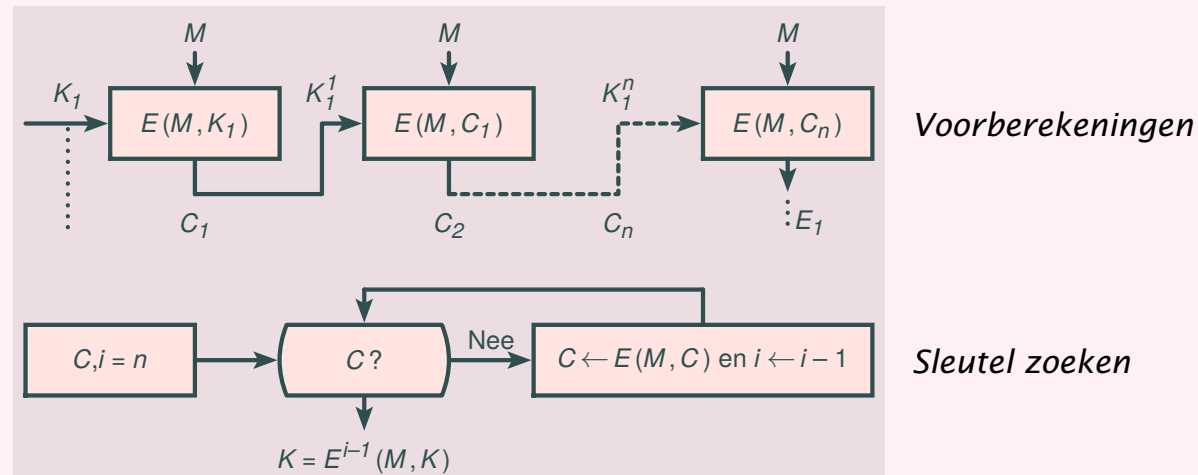
* Zie <http://www.sciengines.com>

† Het wellicht meeste bekende voorbeeld hiervan is het (inmiddels gestaakte) SETI@home programma voor het zoeken naar buitenaards leven.

vanaf het eerste uur^[17]. Het idee is niet specifiek voor cryptanalyse, maar eigenlijk gemeengoed in de informatica. Namelijk dat door het inzetten van geheugen (opslaan van voorberekeningen en tussenresultaten) een vermindering van het rekenwerk en dus van de benodigde rektijd kan worden bewerkstelligd. De benadering is van het type klaartekst/cijfertekst-aanval. Gegeven zulk een paar wordt gevraagd naar de bijbehorende sleutel waarmee de rest van de gecijferde gegevens kan worden ontcijferd.



Hellman



Figuur 8.6 Hellman time-memory tradeoff

Bovenaan in **figuur 8.6** staan de voorberekeningen van Martin Hellman's *time-memory tradeoff*. Kies willekeurig een sleutel K_1 en vercijfer daarmee het klaartekstblok M . Het resultaat van deze eerste vercijfering is C_1 . Gebruik vervolgens C_1 als sleutel om M opnieuw mee te vercijferen, herhaal dit voor het resultaat C_2 , enz. De cijfertekst E_1 , aan het eind van deze keten van vercijferingen, wordt tezamen met K_1 in een tabel opgeslagen. Kies een nieuwe sleutel K_2 , herhaal de operaties en berg K_2 en E_2 weer samen op, enz. Het aantal malen dat deze procedure wordt herhaald is arbitrair, maar hoe meer hoe beter. Tot slot is het verstandig om de lijst op de E 's te sorteren, omdat dan gemakkelijk een cijfertekst kan worden opgezocht.

Voor het gebruik van deze voorberekeningen gaat men als volgt te werk. De te onderzoeken cijfertekst C , waarvan wordt verondersteld dat M de bijbehorende klaartekst is, wordt gezocht in de E -lijst. Stel deze C wordt inderdaad aangetroffen als bijvoorbeeld E_{13492} de 13492^e uit misschien wel 100.000 voorberekeningen. Dit betekent dat de gezochte sleutel kan worden gevonden in de laatste stap van de keten die is begonnen met K_{13492} . Deze keten moet dus nogmaals worden doorlopen en één stap voor het einde gestopt. Het vercijferresultaat geeft dan de gezochte sleutel.

Wat als cijfertekst C niet in de E -lijst voorkomt? De kans bestaat dat dit wél het geval geweest zou zijn als de ketens één stap korter waren gemaakt. Het is niet nodig om alle ketens opnieuw uit te rekenen. Want stel er is een keten waarin C in de voorlaatste stap staat. Als de vercijfering van M met C als sleutel wordt uitgevoerd, dan verlengt dit die keten effectief met één stap en moet zodoende op een van de opgeslagen E 's uitkomen. En dus wordt M met C vercijferd, in de E -lijst opgezocht en bij succes de betreffende keten doorlopen tot de op twee na de laatste vercijfering. Herhaal dit proces totdat of de sleutel is gevonden of de juiste sleutel niet in de voorberekende ketens terecht gekomen is.

De time-memory tradeoff condenseert in elke keten een reeks sleutels tot een enkel waardenpaar beginsleutel-eindresultaat en realiseert daarmee een efficiënte opslag van vele voorberekeningen. Na het kiezen van de beginsleutel bestaat er uiteraard geen controle meer over welke tussensleutels wel en niet zullen voorkomen, maar volgen deze automatisch uit de reeks vercijferingen. Of en in welke keten bepaalde sleutels zullen voorkomen is in feite een kwestie van geluk. Naarmate de ketens langer gemaakt worden en naarmate meer ketens worden gestart, zal een groter aantal sleutels zijn opgenomen. Dus hoe meer voorberekeningen hoe minder kans dat een sleutel ontsnapt.

Hellman's idee is verfijnd tot de zogenaamde *regenboogtabel* (rainbow table in het Engels). Deze tabellen worden in het bijzonder gebruikt voor het achterhalen van wachtwoorden. Wachtwoorden worden gewoonlijk niet als zodanig in de computer bewaard, maar gebruikt als sleutel om een vaste klaartekst mee te vercijferen; het resultaat van deze vercijfering wordt opgeslagen. Bij inloggen wordt dezelfde vercijfering

uitgevoerd en vergeleken met de opgeslagen vercijfering. Zonder verdere veiligheidsmaatregelen is deze procedure kwetsbaar voor een aanval met een regenboogtabel.*

Een voorbeeld waarin deze aanvalstechniek benut wordt, is de hack van bepaalde simkaarten voor mobiele telefonie. Dit voorval is meteen ook een mooie illustratie van de gevaren van onoordeelkundig gebruik van geheimschrift. De hack werkt als volgt. Aanbieders van mobiele diensten gebruiken sms-jes om opdrachten naar een simkaart te sturen. Komt er een ongeldige opdracht binnen, dan stuurt de simkaart een foutmelding terug. De clou is dat dit foutbericht een cryptografische handtekening krijgt, in oudere simkaarten met enkelvoudig DES. Het is nutteloos om een standaardfoutmelding een encryptie mee te geven, maar het is desastreus omdat daarmee de aanvaller een knownplaintext in handen gespeeld krijgt. Met behulp van een regenboogtabel kan de sleutel van de enkelvoudige DES-vercijfering vlot gevonden worden†.



Biham



Shamir

8.3.3 Differentiële cryptoanalyse

Jarenlang maakte de cryptoanalyse van de Data Encryption Standard weinig voortgang. Af en toe bedacht iemand een methode die gewerkt zou hebben wanneer DES maar minder ronden had geteld, bijvoorbeeld 6 in plaats van 16. Dit dan nog afgezien van de praktische bruikbaarheid van zo'n aanval. Want wanneer met 2^{42} gekozen paren klaartekst-cijfertekst de algoritme gebroken kan worden, telt dit weliswaar theoretisch als een succesvolle cryptoanalyse maar vormt het nog geen werkelijke bedreiging. Concrete voortuitgang werd pas in 1990 geboekt door Eli Biham en Adi Shamir. Eerst gepubliceerd in een tijdschrift en later uitgebreider in boekvorm^[18]. Deze beide heren bedachten een geheel nieuwe aanpak voor de cryptoanalyse van blokgeheimschriften, een aanpak die goed kan worden ingezet tegen Feistel-

* Meer informatie over regenboogtabellen kan in Wikipedia worden gevonden.

† J. Schellevis op tweakers.net, 22-07-2013.

geheimschriften in het algemeen^[19]. Ze noemden deze aanpak *differential cryptanalysis*, hier vertaald met differentiële cryptoanalyse.

In deze vorm van cryptoanalyse kijkt men niet naar het effect van de vercijferalgoritme op individuele klaartekst, maar naar het effect op speciale combinaties van klaartekst. Paren klaartekst worden zo gekozen dat voor elk paar de exclusive or-combinatie steeds dezelfde waarde heeft. Uit het vercijferen van zulke paren kan worden geconcludeerd welke sleutels waarschijnlijker zijn dan andere. Het is dus een *chosen-plaintext-aanval*. Dit betekent dat de invoer en de uitvoer van de vercijfering bekend zijn, maar niet de sleutel. De cryptoanalist kan de algoritme niet stap voor stap volgen, doch slechts invoer en uitvoer met elkaar vergelijken. De opzet van deze aanval is tamelijk ingewikkeld en daarom onmogelijk in kort bestek uit te leggen. Voer voor de echte liefhebbers die niet bang zijn voor een paar formules meer of minder.

8.3.4 MASTERSECTIE – Differentiële cryptoanalyse verklaard

Om het mechanisme van differentiële cryptoanalyse zo eenvoudig mogelijk te beschrijven wordt een minder ingewikkelde transformatiefunctie dan in de Data Encryption Standard als voorbeeld gebruikt. Met name de expansiestap van 32 naar 48 bits maakt het lastig om de operaties te volgen. Voor de rondetransformatie $f(R_{i-1}, K_i)$ van het hier als voorbeeld gebruikte Feistel-geheimschrift wordt de volgende parametrisering genomen

- Een 16-bits blok bestaat uit twee 8-bits helften L en R waarin de bits worden genummerd van rechts naar links, te beginnen met 0. De klaartekst $M = (L_0, R_0)$ komt na n ronden als $C = (L_n, R_n)$ de vercijfering uit (aan het eind worden linker- en rechterhelft verwisseld).
- Een 8-bits subsleutel wordt aan het begin van elke rondetransformatie gecombineerd met de rechterhelft uit de vorige ronde

$$R_{i-1} \oplus K_i \tag{8.9}$$

De sleutel wordt daarna drie bits linksom gerooteerd, zodat de subsleutels een cyclus van 16 stappen doorlopen.

- De met de sleutel gecombineerde invoer wordt voorafgaand aan de navolgende substitutie gesplitst in tweemaal een 4-bits invoer voor twee S -dozen, uit elk waarvan een 4-bits uitvoer resulteert

$$S(R_{i-1} \oplus K_i) = S_1(r_0 r_1 r_2 r_3 \oplus k_0 k_1 k_2 k_3) S_2(\dots) \quad (8.10)$$

- De substituties van deze S -dozen zijn*

i	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S_1(i)$	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7
$S_2(i)$	F	1	8	E	6	B	3	4	9	7	2	D	C	0	5	A

(8.11)

- Na de substitutie worden de 4-bits helften weer samengenomen en ondergaan de navolgende permutatie. Deze is zo gekozen dat een cyclus van 8 stappen wordt doorlopen en elk bit één keer op elke positie terechtkomt

$$P(76543210) = (53160742) \quad (8.12)$$

Zoals gezegd, is differentiële cryptoanalyse een chosen-plaintext aanval en wel op paren klaartekst en de daaruit resulterende paren cijfertekst. De sleutel is onbekend en diens bepaling is het doel van de analyse[†]. De paren klaartekst (M, M^*) worden gekenmerkt door hun *verschil* ΔM

$$\Delta M = M \oplus M^* \quad (8.13)$$

Straks zal blijken dat een bepaalde keuze van ΔM het mogelijk maakt om uit de vercijferde paren (C, C^*) conclusies te trekken over (een deel van) de sleutel. Hiertoe zal eerst naar de basis van de analyse worden

* Dit is de eerste substitutie uit respectievelijk de eerste en tweede S -doos van de DES.

† Een known-plaintext aanval kan worden geconstrueerd door het bijeenzoeken van 'geschikte' paren klaartekst. Omdat slechts een deel van het bekende materiaal aan het geschiktheids criterium zal voldoen, zullen veel meer paren nodig zijn om het analyseresultaat te bereiken.

gekeken. Te weten het gedrag van zulke verschillen onder substitutie door de S -dozen. Om vervolgens in enkele uitgewerkte voorbeelden de gevolgen van het doorlopen van een aantal ronden van het Feistel-netwerk te analyseren.

S -doos invoer

De raison d'être van het werken met verschillen $M \oplus M^*$ is gelegen in de wijze waarop de (sub)sleutel in de rondetransformatie wordt gecombineerd met de invoer, zie [vergelijking \(8.9\)](#). De subsleutel valt daardoor weg uit de verschillen

$$\begin{aligned}(R_{i-1} \oplus K_i) \oplus (R_{i-1}^* \oplus K_i) &= (R_{i-1} \oplus R_{i-1}^*) \oplus (K_i \oplus K_i) \\ &= R_{i-1} \oplus R_{i-1}^* \\ &= \Delta R_i\end{aligned}\tag{8.14}$$

Verschillende paren $R \oplus R^*$ met constant verschil ΔR zullen zodoende alle met hetzelfde verschil de substitutie van de S -dozen ingaan. En hoewel de afzonderlijke substitutieresultaten $S(R)$ en $S(R^*)$ zullen variëren, is er toch een merkbaar effect op het verschil $S(R) \oplus S(R^*)$. Dit effect wordt zichtbaar wanneer alle mogelijke invoercombinaties van een S -doos worden gerubriceerd naar het verschil van hun uitvoerwaarden. Voor de S -dozen van [vergelijking \(8.11\)](#) is voor alle 256 invoercombinaties (i, j) met $i, j = 0 \dots F$ het uitvoerpaar (s_i, s_j) bepaald en zijn de verschilcombinaties $(i \oplus j, s_i \oplus s_j)$ geteld. Het resultaat hiervan voor S_1 staat in [tabel 8.4](#).

Elke positie in de tabel geeft weer hoe vaak bij een zeker invoerverschil een bepaald uitvoerverschil wordt aangetroffen. De specifieke waarden zijn vanzelfsprekend afhankelijk van de substitutie in de S -doos. Direct valt op dat de verdeling van de waarden niet homogeen is. Sommige combinaties komen betrekkelijk vaak voor, zoals 8 keer ($B \rightarrow 2$). Andere daarentegen in het geheel niet, zoals ($B \rightarrow 3$). Dat de combinatie $(0 \rightarrow 0)$ 16 keer voorkomt is niet zo verwonderlijk. Een invoerverschil van nul betekent immers dat gelijke

$s_i \oplus s_j$	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	2	0	0	0	2	0	2	4	0	4	2	0	0
2	0	0	0	2	0	6	2	2	0	2	0	0	0	0	2	0
3	0	0	2	0	2	0	0	0	0	4	2	0	2	0	0	4
4	0	0	0	2	0	0	6	0	0	2	0	4	2	0	0	0
5	0	4	0	0	0	2	2	0	0	0	4	0	2	0	0	2
6	0	0	0	4	0	4	0	0	0	0	0	0	2	2	2	2
7	0	0	2	2	2	0	2	0	0	2	2	0	0	0	0	4
8	0	0	0	0	0	0	2	2	0	0	0	4	0	4	2	2
9	0	2	0	0	2	0	0	4	2	0	2	2	2	0	0	0
A	0	2	2	0	0	0	0	0	6	0	0	2	0	0	4	0
B	0	0	8	0	0	2	0	2	0	0	0	0	0	2	0	2
C	0	2	0	0	2	2	2	0	0	0	0	2	0	6	0	0
D	0	4	0	0	0	0	0	4	2	0	2	0	2	0	2	0
E	0	0	2	4	2	0	0	0	6	0	0	0	0	0	2	0
F	0	2	0	0	6	0	0	0	0	4	0	2	0	0	2	0

Tabel 8.4 Distributie S_1 invoer/uitvoer-paren

waarden worden gesubstitueerd en dan stemmen de substitutieresultaten uiteraard ook overeen. Overigens zal deze triviale ($0 \rightarrow 0$) combinatie nog buitengewoon nuttig blijken.

Richt als voorbeeld de aandacht op invoerpaar $(i, j) = (2, A)$ dat in S_1 uitvoerpaar $(D, 6)$ geeft

$$2 \xrightarrow{S_1} D \quad A \xrightarrow{S_1} 6 \quad (2 \oplus A) = 8 \xrightarrow{S_1} (D \oplus 6) = B$$

Het invoerverschil is $2 \oplus A = 8$ en het uitvoerverschil $D \oplus 6 = B$. Er blijken nog drie andere invoercombinaties met ditzelfde invoer/uitvoer-verschil, waardoor in tabel 8.4 in rij 8, kolom B een 4 verschijnt. De andere zijn de invoerparen $(A, 2)$ $(6, E)$ en $(E, 6)^*$. Ook laat de tabel zien dat bij een invoerverschil van 8 alleen uitvoerverschillen 6, 7, B, D, E, F kunnen voorkomen. Tevens blijkt dat een bepaald uitvoerverschil niet

uit elk willekeurig invoerverschil kan zijn ontstaan.

Wissel nu van gezichtspunt en kijk alleen naar de invoer/uitvoer-verschillen. Wat kan op grond hiervan worden geconcludeerd over de invoer van de S -doos? In het geval van de combinatie (8,B) behoren uitsluitend de zojuist genoemde invoerparen (2,A) (A,2) (6,E) en (E,6) tot de kandidaten. Zo leidt waarneming van een verschilpaar dus tot inperking van de mogelijke invoer die aan de S -doos werd gepresenteerd. Uit diverse verschilcombinaties met andere invoerparen kan het aantal mogelijkheden door het combineren van dit soort waarnemingen worden gereduceerd.

Indien nu ook de werkelijke invoerwaarden R_{i-1} en R_{i-1}^* bekend zijn, dan is het mogelijk om de overeenkomstige bits van de subsleutel te bepalen. Stel de invoer van de S -doos is beperkt tot respectievelijk 2 en A. Als nu $R_{i-1} = B$ en $R_{i-1}^* = 3$ volgt eenduidig: $K_i = B \oplus 2 = 3 \oplus A = 9$. Hieronder is een en ander nog eens schematisch weergegeven.

$$\begin{array}{ccccccc}
 R_{i-1} \oplus K_i & & R_{i-1}^* \oplus K_i & & R_{i-1} \oplus R_{i-1}^* & & K_i \\
 \downarrow & & \downarrow & & \downarrow & & \uparrow \\
 S_1(r_1) S_2(r_2) \dots & & S_1(r_1^*) S_2(r_2^*) \dots & & r_1, r_2 \dots & \rightarrow & k_1, k_2 \dots \\
 \downarrow & & \downarrow & & \uparrow & & \\
 P(S_1(r_1) S_2(r_2) \dots) & & P(S_1(r_1^*) S_2(r_2^*) \dots) & & P^{-1}(R_i \oplus R_i^*) & & \\
 \downarrow & & \downarrow & & \uparrow & & \\
 R_i & & R_i^* & & R_i \oplus R_i^* & &
 \end{array} \quad (8.15)$$

Recapitulerend: met de inverse van de bekende permutatie P wordt het eindresultaat van de rondetransformatie teruggerekend naar het uitvoerverschil van de S -dozen. In combinatie met het invoerverschil wordt de invoer van de S -dozen bepaald en daaruit met behulp van de bekende invoer R_{i-1} het overeenkomstige deel van de subsleutel K_i .

* Gegeven de symmetrie $(i \oplus j) = (j \oplus i)$ is het niet verbazingwekkend dat alle combinaties paarsgewijs voorkomen en dat de tellingen in de tabel zodoende alle een even waarde hebben.

Feistel 4 ronden

In het voorafgaande is één enkele ronde van het Feistel-netwerk geanalyseerd. Nu moet de stap naar meerdere ronden worden gemaakt en het is handig om dat eerst tot vier te beperken. De opzet verandert niet, namelijk het bepalen van de invoer van een S -doos door combinatie van invoer/uitvoer-verschilparen. Het verloop van de vercijfering over vier ronden is als volgt

$$\begin{array}{ll}
 & L_0 \qquad \qquad R_0 \\
 1 : & L_1 = R_0 \qquad R_1 = L_0 \oplus f(R_0, K_1) \\
 2 : & L_2 = R_1 \qquad R_2 = L_1 \oplus f(R_1, K_2) \\
 3 : & L_3 = R_2 \qquad R_3 = L_2 \oplus f(R_2, K_3) \\
 4 : & L_4 = R_3 \qquad R_4 = L_3 \oplus f(R_3, K_4)
 \end{array} \tag{8.16}$$

Uitwerken van L_3 met de erboven staande vergelijkingen geeft voor het vercijferd eindresultaat

$$L_4, R_4 = R_3, \underbrace{L_1 \oplus f(R_1, K_2) \oplus f(R_3, K_4)}_{L_3=R_2} \tag{8.17}$$

Voor het bepalen van subsleutel K_4 is een aantal gegevens nodig. Eerstens de invoer (respectievelijk R_3 en R_3^*) van de S -dozen in de laatste rondetransformatie. Deze invoergegevens zijn direct terug te vinden in het vercijferd resultaat, want $R_3 = L_4$. Het invoerverschil levert hier dus geen probleem op. Het uitvoerverschil van de S -dozen in de vierde en laatste ronde is, terugrekenend over de laatste permutatie

$$P^{-1}(f(R_3, K_4) \oplus f(R_3^*, K_4)) \tag{8.18}$$

Dit uitvoerverschil is moeilijker te vinden. Het kan met behulp van [vergelijking \(8.17\)](#) bepaald worden

$$\Delta R_4 = \Delta R_2 \oplus f(R_3, K_4) \oplus f(R_3^*, K_4) \tag{8.19}$$

Door omwerken volgt het uitvoerverschil van de substitutie in de laatste ronde

$$f(R_3, K_4) \oplus f(R_3^*, K_4) = \Delta L_1 \oplus f(R_1, K_2) \oplus f(R_1^*, K_2) \oplus \Delta R_4 \tag{8.20}$$

ΔL_1 en ΔR_4 zijn direct te vinden, die komen uit respectievelijk de klaartekst (vanwege $L_1 = R_0$) en de cijfertekst. Het probleem schuilt in $f(R_1, K_2) \oplus f(R_1^*, K_2)$, het uitvoerverskil twee ronden voor het eind. Om iets over dit uitvoerverskil te kunnen zeggen moet worden nagegaan hoe een invoerverskil $\Delta M = M \oplus M^*$ zich voortplant. In het algemene geval is dat niet doenlijk, omdat de opeenvolgende substituties en permutaties dit aan het zicht onttrekken. Elke S -doos volvoert een andere substitutie en de permutatie verspreidt die veranderingen naar andere S -dozen. Na een paar ronden is het resultaat volkomen ondoorzichtig.

Gelukkig zijn er *bijzondere verschilwaarden*. Waarden waarbij na een aantal ronden nog iets gezegd kan worden over het eindresultaat. Zo'n speciale waarde en diens vervolgstappen noemt men een *karakteristiek*. Het zoëven gesignaleerde probleem zetelt hoofdzakelijk in de interactie tussen de verschillende S -dozen, waarvan de uitvoer in de volgende ronden doorelkaar gaat lopen. Er is echter een manier om een S -doos uit te schakelen, namelijk door deze een invoerverskil van nul aan te bieden. Daarop kan maar één reactie volgen, namelijk een uitvoerverskil van nul. Door nu het invoerverskil ΔM zó te kiezen dat op een na alle S -dozen een verschil van nul toegevoerd krijgen, blijft het effect van de transformatiefunctie aanvankelijk beperkt tot de ene overgebleven S -doos.

In het als voorbeeld gebruikte Feistel-netwerk wordt de 8-bits invoer van de rondetransformatie in stukken van 4 bits verdeeld over de beide S -dozen. Startend met invoerverskil $\Delta M = 0F00$ ofwel $\Delta L_0 = 0F$, $\Delta R_0 = 00$, valt nu het volgende te zien*. In de eerste ronde krijgen de beide S -dozen een invoerverskil 0 en antwoorden daarop met een uitvoerverskil van eveneens 0. De permutatie die op de substitutie volgt verandert daaraan niets, zodat

$$f(R_0, K_1) \oplus f(R_0^*, K_1) = 00 \quad (8.21)$$

Dit resultaat moet nog worden gecombineerd met ΔL_0 , het verschil van de linkerhelften en zodoende luidt de karakteristiek voor deze analyse

* Feitelijk zijn alle waarden 0100, 1000, 1100,..., 0F00 hier als verschil bruikbaar.

$$\begin{aligned}\Delta L_0 &= 0F & \Delta R_0 &= 00 \\ \Delta L_1 &= \Delta R_0 = 00 & \Delta R_1 &= \Delta L_0 \oplus 00 = 0F\end{aligned}\tag{8.22}$$

Maar er moet nog een ronde gegaan worden, want om de linkerzijde van [vergelijking \(8.20\)](#) te berekenen is $f(R_1, K_2) \oplus f(R_1^*, K_2)$ nodig. In de tweede ronde gaat een verschil $\Delta R_1 = 0F$ de substitutie in. S -doos S_1 krijgt zodoende weer een invoerverschil 0 en produceert opnieuw een uitvoerverschil van 0. Daarentegen krijgt S_2 een van nul verschillende invoer en zal dus een of ander uitvoerverschil opleveren. Het verschilresultaat van deze substitutie uit S -doos S_2 wordt door de permutatie over alle 8 bits van de rechterhelft verdeeld en zal dus pas in de daaropvolgende ronde de substitutie door beide S -dozen gaan beïnvloeden.

Wat wordt het resultaat? In [vergelijking \(8.20\)](#) voor de berekening van de uitvoer van de laatste rondetransformatie zijn als gevolg van de genomen karakteristiek nu $\Delta L_1 = 00$ en $P^{-1}(f(R_1, K_2) \oplus f(R_1^*, K_2)) = 0X$. De X hierin staat voor de onbekende uitvoer van S_2 . Zodoende kan samen met ΔR_4 (zie weer [vergelijking \(8.20\)](#)) de uitvoer van S_1 in de laatste ronde worden berekend en met ΔL_4 de invoer. Hieruit komen mogelijke waarden voor de invoer van S_1 tevoorschijn en deze leveren net als in de voorgaande analyse de vier sleutelbits van S_1 . Herhaling met verschillende paren zal naar men mag hopen een eenduidig resultaat opleveren.

[Tabel 8.5](#) geeft de uitkomst van een experiment waarbij verschillende waarden voor M' , maar wel allemaal passend binnen de karakteristiek, zijn gebruikt. Voor elk vercijferd paar is het invoer/uitvoer-verschil gegeven alsook de waarden die dit voor de sleutelbits van S_1 suggereert. De juiste waarde moet er elke keer bij zitten als gevolg van de gebruikte karakteristiek. Turven van de gesuggereerde waarden doet de goede oplossing er onmiddellijk uitspringen, want uit zestien vercijferde paren wordt de waarde A zestien keer gevonden.

Door het experiment te herhalen met de andere karakteristiek $\Delta M = F000$ kunnen de sleutelbits tegenover S_2 op dezelfde manier worden bepaald. Essentieel is het ingangsverschil 0 voor de S -doos waarvan men de ermee overeenkomende sleutelbits wil vinden.

M	M^*	ΔM	C	C^*	ΔC	$i \rightarrow S_1(i)$	$K_4(S_1)$								
3EF5	3FF5	0100	B41C	0328	B734	3 $\rightarrow$ F	1 2 9 A								
2F79	2E79	0100	30DD	A40C	94D1	D $\rightarrow$ E	A 7								
4E7E	4F7E	0100	BD54	7FBA	C2EE	E $\rightarrow$ 3	4 0 E A								
4AD4	48D4	0200	B456	1085	A4D3	D $\rightarrow$ A	7 A								
3360	3060	0300	591E	5B89	0297	9 $\rightarrow$ 1	3 A								
2F5D	2B5D	0400	EFD8	24C6	CB1E	1 $\rightarrow$ 3	B A								
092A	0D2A	0400	E902	6E84	8786	8 $\rightarrow$ B	2 6 A E								
1B16	1C16	0700	B6A1	A636	1097	9 $\rightarrow$ 4	A 3								
0D37	0737	0A00	CC19	8B6F	4776	7 $\rightarrow$ 9	A D								
7DBB	77BB	0A00	DF34	5A4D	8579	7 $\rightarrow$ A	A D								
1CCE	16CE	0A00	34DD	5296	664B	4 $\rightarrow$ 9	E A								
3602	3D02	0B00	F435	D1E7	25D2	D $\rightarrow$ 8	7 A								
3D00	3600	0B00	4B93	7260	39F3	F $\rightarrow$ 4	8 B A 5 4 7								
5AB3	54B3	0E00	8CF8	77DF	FB27	2 $\rightarrow$ 7	A 8								
7B16	7516	0E00	0E8F	A003	AE8C	8 $\rightarrow$ B	A E 2 6								
7983	7683	0F00	078F	D539	D2B6	B $\rightarrow$ 7	A 1								
Telling voor S_1 sleutelbits															
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	2	3	2	2	1	2	4	2	1	16	2	0	2	4	0

Tabel 8.5 Bepaling van de vier S_1 -sleutelbits uit 4-ronde-analyse

Feistel 6 ronden

De voorgaande analyse over vier ronden kon beschikken over een karakteristiek die voor elk verschilpaar werkt. Dat wordt anders indien meer ronden moeten worden geanalyseerd, omdat dan niet meer zo'n mooie karakteristiek kan worden gevonden. Om het voorbeeldprobleem over zes ronden van het Feistel-netwerk aan te pakken, wordt gezocht naar een karakteristiek die over drie ronden doorloopt. De daaropvolgende laatste drie vercijferronden kunnen worden opgevangen door vanaf het eind van de verci-

jfering terug te rekenen naar de staart van de karakteristiek.

De karakteristiek die hier benut wordt is de volgende

$$\begin{aligned}\Delta L_0 &= 80 & \Delta R_0 &= B0 \\ \Delta L_1 &= B0 & \Delta R_1 &= 00 \\ \Delta L_2 &= 00 & \Delta R_2 &= B0 \\ \Delta L_3 &= B0 & \Delta R_3 &= 80\end{aligned}\tag{8.23}$$

En nu komt het probleem, want kijkend naar [tabel 8.4](#) blijkt in S_1 het invoerverschil B tot verscheidene uitvoerverschillen aanleiding te geven: acht keer tot 2 en elk twee keer tot 5, 7, D en F. S -doos S_2 geeft geen moeite, want invoerverschil 0 komt er altijd als 0 uit. Indien in een vercijfering door S_1 nu inderdaad verschil $B \rightarrow 2$ dan is het substitutieverchil dus 20. Hetgeen vervolgens wordt gepermuteerd tot 80 en daarna gecombineerd met ΔL_0

$$\Delta R_1 = \Delta L_0 \oplus P(S_1(B)S_2(0)) = 80 \oplus 80 = 00\tag{8.24}$$

Men zegt dat een *goed paar* klaartekst/cijfertekst door de vercijferstap heenloopt als hiervoor ook werkelijk het verschil $B \rightarrow 2$ optreedt en dus de karakteristiek $B0 \rightarrow 80$ in deze stap opgaat. Het kan ook anders lopen, namelijk wanneer de invoer van de afzonderlijke vercijferingen van het paar in S_1 leidt tot een van de andere mogelijke verschillen, namelijk $B \rightarrow 5, 7, D, F$. Hoe vaak dit precies zal gebeuren is onbekend, maar gevoeglijk mag men aannemen dat de aantallen wel statistisch verdeeld zullen zijn naar rato van de aantallen in [tabel 8.4](#). En dus is de veronderstelling dat acht van de zestien keer $B \rightarrow 2$, waarmee deze overgang een kans van $p = 0,5$ krijgt. Hetzelfde geldt voor de laatste stap, waarin (wederom) verschil $B0 \rightarrow 80$ wordt verondersteld. De kans daarop is dus ook weer $p = 0,5$. Gegeven de willekeurige keuze van het paar klaartekstblokken met verschil $80B0$, is er dus in totaal een kans $p = 0,25$ dat dit paar de vercijferingen $\Delta L_0 \Delta R_0 = 80B0 \rightarrow \Delta L_3 \Delta R_3 = B080$ van deze karakteristiek zal doorlopen.

De verdere analyse verloopt in principe hetzelfde als in het vorige geval en ook hier kunnen in de laatste ronde de subsleutelbits worden bepaald voor de S -doos die een invoerverschil van 0 in de laatste ΔR van

de karakteristiek bezit. Analooq aan **vergelijking (8.18)** en **(8.20)** komt er nu

$$\begin{aligned}
 \Delta in &= \Delta L_6 \\
 S_1(\Delta in)S_2(\Delta in) &= P^{-1}(f(R_5, K_6) \oplus f(R_5^*, K_6)) \\
 &= P^{-1}\left(\underbrace{\Delta L_3}_{B0} \oplus \underbrace{f(R_3, K_4) \oplus f(R_3^*, K_4)}_{S_2(\Delta in=0)=0} \oplus \underbrace{\Delta R_6}_{\Delta C}\right)
 \end{aligned} \tag{8.25}$$

De invoer in de laatste ronde wordt geleverd door $\Delta L_6 = \Delta R_5$ en de uitvoer voor de berekenbare bits volgt uit bovenstaande vergelijking. De factor ΔL_3 komt uit de karakteristiek. Weer worden alle gesuggereerde oplossingen bepaald en gekeken welke sleutel er uit springt.

Natuurlijk geeft deze berekening slechts dan een betrouwbaar resultaat als het een *goed paar* betreft, want in dat geval wordt het patroon van verschillen keurig gevolgd en zijn de veronderstellingen waaruit de sleutelbits berekend worden juist. Vercijferde paren die niet aan het patroon van de karakteristiek voldoen zijn foutieve paren en kunnen niet worden gebruikt om bits van de sleutel te bepalen. Het probleem is echter dat tijdens de analyse niet bekend is welke paren goed en welke fout zijn. Zo komen er dus ook foute suggesties in de tellingen terecht. Men kan slechts hopen dat de foute oplossingen willekeurig verdeeld zijn over alle mogelijkheden en dat de goede oplossing na genoeg pogingen kan worden onderscheiden van de foute.

De verhouding van het aantal goede tot het aantal fouten paren bepaalt hoeveel *signaal* er is en hoeveel *ruis*. Relatief meer goede paren betekent een betere signaal/ruis-verhouding en dus een sneller uitlopen van de goede oplossing. De kans geassocieerd met de gekozen karakteristiek heeft grote invloed op deze verhouding. Bij een karakteristiek met een lage kans op voorkomen zal de goede oplossing gemakkelijk in de ruis kunnen verdrinken.

Een lage signaal/ruis-verhouding kan worden gecompenseerd door meer paren te onderzoeken. Dat lijkt een uitweg te bieden, maar ook hiervoor geldt "aan alles komt een eind". Op een gegeven moment zijn zoveel paren nodig dat uitputtend afzoeken voordeliger wordt en toepassen van differentiële analyse

weinig nut meer heeft. En als de karakteristiek zo'n kleine kans op voorkomen bezit dat meer paren nodig zijn dan er überhaupt bestaan, dan is differentiële cryptanalyse zelfs niet mogelijk*.

In het gegeven voorbeeld heeft de karakteristiek een kans $p = 0,25$, hetgeen betekent dat waarschijnlijk driekwart van de paren een fout paar zal zijn. Gelukkig heeft een deel van de foute paren geen effect. Van S_2 is bekend welke in- en uitvoerverschillen in de laatste ronde de transformatiefunctie ingaan. Voor een goed paar levert dat een bruikbare suggestie op, maar wat doet een fout paar? Sommige foute paren hebben een combinatie van in- en uitvoerverschil dat niet kan voorkomen! In tabel 8.4 zijn dit de combinaties met een 0 als tabelwaarde. Die kunnen geen bruikbare suggestie voor de sleutel leveren en men kan ze daarom zonder meer laten vallen. Voor invoerverschil B zijn dit de paren met uitvoerverschil 0, 1, 3, 4, 6, 8, 9, A, B, C, F. Zo raakt men hier dus eenvoudig ongeveer 11/16 van de foute paren kwijt. Soms ook is er een invoer/uitvoer-paar (0,0) voor S_2 , dat niets oplevert omdat dit alle 16 invoerwaarden 0...F suggereert en ook die paren kunnen worden weggelaten.

Al met al leidt dit tot het uitgewerkte voorbeeld in tabel 8.6. Deze laat de bepaling zien van de sleutelbits van S_2 met de karakteristiek van vergelijking (8.23) uit 64 paren. Van deze 64 paren blijven er op grond van de hiervoor gegeven overwegingen 42 over. De goede oplossing verwacht men ongeveer $64/4 = 16$ keer te zien, vermeerderd met een aandeel uit de (willekeurige) resultaten van de overgebleven foute paren.

Onderaan in dezelfde tabel is de telling voor de sleutelbits van S_1 uit een andere karakteristiek (8.26) gegeven. Deze karakteristiek heeft een lagere kans op voorkomen en bij eenzelfde aantal paren is het resultaat daarom minder sprekend. Bij 64 paren (een na onderste regel in de tabel) kan de goede oplossing – te weten $K = 1$ – nauwelijks worden onderscheiden, maar na 128 pogingen (onderste regel in de tabel) valt deze niet te missen.

* Een systeem met n -bits blokken telt hoogstens 2^n verscijferde blokken, meer zijn er niet.

M	M^*	ΔM	C	C^*	ΔC	$K_6(S_2)$
7A90	FA20	80B0	DF65	041E	DB7B	7 C
236B	A3DB	80B0	A63D	4C49	EA74	D 9
6D74	EDC4	80B0	F2B7	F6BD	040A	fout
1186	9136	80B0	CA9B	7243	B8D8	fout
408D	C03D	80B0	8301	FFD4	7CD5	9 C
...	...	...	...	...	...	
7126	F196	80B0	0ADF	2627	2CF8	fout
180E	98BE	80B0	A8F0	E285	4A75	9 C
403F	C08F	80B0	1CD5	24A2	3877	E 9
3A4B	BAFB	80B0	4623	06DD	40FE	3 5 B D
76A1	F611	80B0	0A3B	74DA	7EE1	fout
4F74	CFC4	80B0	64CC	40E5	2429	fout
57BE	D70E	80B0	BF54	181A	A74E	6 0 E 8
Telling voor sleutelbits S_2 (64 paren)						
0	1	2	3	4	5	6 7 8 9 A B C D E F
4	6	3	7	8	3	3 3 6 20 1 8 8 8 11 3
Telling voor sleutelbits S_1 (64/128 paren)						
0	1	2	3	4	5	6 7 8 9 A B C D E F
4	9	3	3	1	3	3 2 5 2 8 4 0 1 1 1
11	22	5	10	3	8	4 4 9 4 14 11 1 6 7 3

Tabel 8.6 Bepaling van 8 sleutelbits uit 6-ronde-analyse

$$\begin{array}{lll}
 \Delta L_0 = 15 & \Delta R_0 = 20 & p = 3/8 \\
 \Delta L_1 = 20 & \Delta R_1 = 07 & p = 3/8 \\
 \Delta L_2 = 07 & \Delta R_2 = 00 & p = 1 \\
 \Delta L_3 = 00 & \Delta R_3 = 07 & p = 3/8
 \end{array}
 \tag{8.26}$$

$$p_{\text{totaal}} = 27/512 \approx 0.05$$

In sommige gevallen bestaat de mogelijkheid om de controle op foute paren nog een stapje uit te breiden. In dit geval kan dat door **karakteristiek (8.26)** met één stap te verlengen

$$\begin{array}{lll}
 \Delta L_0 = 15 & \Delta R_0 = 20 & p = 3/8 \\
 \Delta L_1 = 20 & \Delta R_1 = 07 & p = 3/8 \\
 \Delta L_2 = 07 & \Delta R_2 = 00 & p = 1 \\
 \Delta L_3 = 00 & \Delta R_3 = 07 & p = 3/8 \\
 \Delta L_4 = 07 & \Delta R_4 = 20 & p = 3/8
 \end{array} \tag{8.27}$$

$$p_{\text{totaal}} = 81/4096 \approx 0.02$$

Hiermee kan worden gecontroleerd of de invoer/uitvoer-combinatie in de derde ronde inderdaad de door de karakteristiek verlangde waarden heeft en zo vallen meer foute paren door de mand. Tegenover dit voordeel staan ook nadelen, want goede paren met een ander toegestaan uitvoerverschil blijven nu ongebruikt en de verminderde kans op deze langere karakteristiek doet aan de signaal/ruis-verhouding geen goed. Om dezelfde reden valt het gebruik van karakteristieken zonder een 0 af te raden. Bij die karakteristieken moeten voor één stap de kansen over twee S-dozen met elkaar worden vermenigvuldigd. Maar zo kan wel **karakteristiek (8.27)** aan de voorkant worden uitgebreid

$$\begin{array}{lll}
 \Delta L_0 = E4 & \Delta R_0 = 15 & p = 1/16 \\
 \Delta L_1 = 15 & \Delta R_1 = 20 & p = 3/8 \\
 \Delta L_2 = 20 & \Delta R_2 = 07 & p = 3/8 \\
 \Delta L_3 = 07 & \Delta R_3 = 00 & p = 1 \\
 \Delta L_4 = 00 & \Delta R_4 = 07 & p = 3/8 \\
 \Delta L_5 = 07 & \Delta R_5 = 20 & p = 3/8
 \end{array} \tag{8.28}$$

$$p_{\text{totaal}} = 81/65536 \approx 0.001$$

Deze laatste karakteristiek zou in principe kunnen worden benut voor een analyse over 8 ronden. Maar door de lage kans zullen er slechts een stuk of 80 goede paren voorhanden zijn.

Differentiële cryptoanalyse van DES

Na deze tamelijk uitvoerige aanloop, kan worden geschetst hoe de aanval op de Data Encryption Standard met differentiële cryptoanalyse door Eli Biham en Adi Shamir is opgezet^[20]. De aanval staat of valt met het vinden van een *kansrijke karakteristiek*, een die minstens over 13 ronden moet doorlopen om de volle 16 ronden van de DES de baas te kunnen*.

Het vinden van een karakteristiek met een redelijk hoge kans is om de volgende reden niet eenvoudig. Er is geen enkele *S*-doos die een karakteristiek met kans $p = 1/4$ of meer geeft. Om precies te zijn, op z'n best komen 14 van de 64 invoerverschillen in één uitvoerverschil terecht, zodat $p \leq 14/64$ [†]. Als dan ook nog twee of meer *S*-dozen tegelijk een bepaalde invoer/uitvoer-verschilcombinatie moeten vertonen, worden de afzonderlijke kansen per *S*-doos met elkaar vermenigvuldigd en wordt de gecombineerde kans kleiner. Men moet verder ook nog bedenken dat – als gevolg van het ontwerp van de *S*-dozen – een invoerverschil in één bit een verschil in tenminste twee uitvoerbits bewerkstelligt. Deze twee verschillen worden vervolgens door de permutatie en de expansie in de volgende ronde over verschillende *S*-dozen verdeeld. Een karakteristiek over twee ronden kan dus op z'n best een kans $p = (14/64)^3 \approx 0,01$ hebben. Een karakteristiek over 13 ronden heeft op grond van deze analyse een kans van ten hoogste $p \approx 2^{-41.7}$.

De best bruikbare karakteristiek berust op een combinatie van 3 *S*-dozen en heeft een kans die nog beduidend lager ligt dan het zojuist afgeleide maximum

$$\begin{array}{lll}
 \Delta L_0 = 19\ 60\ 00\ 00 & \Delta R_0 = 00\ 00\ 00\ 00 & p = 1 \\
 \Delta L_1 = 00\ 00\ 00\ 00 & \Delta R_1 = 19\ 60\ 00\ 00 & p = (14/64) \times (8/64) \times (10/64) \\
 \Delta L_2 = 19\ 60\ 00\ 00 & \Delta R_2 = 00\ 00\ 00\ 00 & p = 1 \\
 \Delta L_3 = 00\ 00\ 00\ 00 & \Delta R_3 = 19\ 60\ 00\ 00 & p = (14/64) \times (8/64) \times (10/64) \\
 \dots & \dots &
 \end{array} \tag{8.29}$$

* De behandeling hier laat omwille van de beknoptheid vele fijnere puntjes van die analyse onbesproken.

† Bedenk dat een *S*-doos een 6-bits invoer en 4-bits uitvoer heeft.

Het mooie aan deze karakteristiek is wel het iteratief gedrag. Na elke twee ronden is de uitgangstoestand weer bereikt. Men kan haar zo vaak herhalen als men wil, maar met het aantal iteraties neemt de kans op een goed paar natuurlijk wel snel af. Een van de twee ronden heeft weliswaar vanwege alle nullen een kans $p = 1$, maar die van de andere is ≈ 0.004274 .

Om de DES hiermee te breken zou men deze gecombineerde karakteristiek zeven keer (over 14 ronden) moeten itereren, wat over het geheel een kans oplevert van

$$p = \left(\frac{14}{64} \times \frac{8}{64} \times \frac{10}{64} \right)^7 = 2^{-55.1}$$

Met uitputtend zoeken in 2^{-56} operaties kan dat geen indrukwekkende prestatie genoemd worden. Bovendien worden zo slechts de 48 bits van de laatste subsleutel gevonden en zijn er dus nog 256 operaties nodig om de gehele 56-bits sleutel te completeren. Het is duidelijk dat nog enkele slimme trucs bedacht moeten worden, iets wat Eli Biham en Adi Shamir wel is toevertrouwd.

Zij itereren [karakteristiek \(8.29\)](#) over 13 ronden (= 6,5 iteraties), waarbij de laatste (die met $p = 1$) eigenlijk cadeau gegeven wordt, zodat

$$p = \left(\frac{14}{64} \times \frac{8}{64} \times \frac{10}{64} \right)^6 = 2^{-47.1}$$

Aan de staart blijven nog twee ronden over waardoor, zoals bij [karakteristiek \(8.27\)](#) is betoogd, meer mogelijkheden tot het verwerpen van foute paren bestaan dan bij een karakteristiek die drie ronden voor het eind stopt. Er mist dan nog één stap en die wordt aan het begin ingevoegd, zó gekozen dat deze eindigt met de eerste ronde van de iteratieve [karakteristiek \(8.29\)](#).

Het slimme zit 'm er in dat de toevoeging van de eerste ronde zonder vermindering van de totale kans kan geschieden. In een voorbereidingsfase worden 2^{48} gekozen 64-bits klaartekstblokken vercijferd, waaruit door een selectieproces op basis van de gewenste karakteristiek $2^{35.25}$ paren als kandidaat naar de uiteindelijke analyse doorstromen. De kans op tenminste één goed paar daarin is ongeveer 0,58

en zo'n goed paar geeft gegarandeerd de juiste sleutel. Doordat verder geanalyseerd wordt dan in de voorgaande voorbeelden, kan ook nog het tellen van alle gesuggereerde subsleutels worden vermeden. In de voorgaande voorbeelden waren dat er slechts 16, maar bij de DES moeten vanwege de 48 subsleutelbits 2^{48} mogelijke sleutels geturfd worden. Dankzij een nog verdere verfijning kan het aantal benodigde gekozen klaartekstblokken met nog eens factor twee teruggebracht worden. De tijdcomplexiteit van de gehele analyse blijkt tenslotte equivalent met 2^{37} DES-vercijferingen.

8.3.5 Modificaties van DES

Na hun succesvolle aanval op de Data Encryption Standard hebben Eli Biham en Adi Shamir onderzocht wat het weerstandsvermogen van een aantal varianten van de DES tegen differentiële cryptanalyse is. Hieronder volgen hun resultaten.

- *De invloed van de permutatie P.*

Deze permutatie blijkt een zeer belangrijke invloed te hebben en biedt goede aangrijpingspunten voor de analyse. Verwijdering ervan maakt dat een S-doos in de volgende ronde slechts zichzelf en zijn twee naaste burens beïnvloedt (dit laatste als gevolg van de expansiestap). Bijgevolg verspreiden veranderingen zich langzamer over de andere posities dan mét de permutatie het geval is. Dit resulteert in een karakteristiek met een invoer/uitvoer-verschil van nul voor vijf S-dozen in ronde 14 met waarschijnlijkheid $2^{-16.5}$. Aangezien deze karakteristiek twee ronden voor het einde stopt, valt een goed onderscheid tussen een fout en een goed paar te maken. Het feit dat nu in de cijfertekst 18 bits van ΔC een 0 moeten zijn, laat veel foute paren uitfilteren. Achterwege laten van de permutatie verzwakt de algoritme aanzienlijk en de cryptanalyse vergt dan nog maar 2^{20} paren. De Data Encryption Standard is dus veel zwakker zonder deze permutatie. Aan de andere kant blijkt de precieze keuze van de permutatie niet veel invloed te hebben op de aard van de aanval, omdat die is gebaseerd

op verschillen tussen de uitvoer van de S -dozen. Een andere permutatie kan in dat opzicht de Data Encryption Standard evenmin verder versterken.

- *Veranderen volgorde van de S -dozen.*

De onderlinge volgorde van de S -dozen blijkt buitengewoon kritisch. Als bijvoorbeeld de S -dozen $S_1 S_7 S_4$ naast elkaar hadden gestaan, zou de differentiële cryptoanalyse veel efficiënter hebben kunnen worden uitgevoerd. Er was dan een analoge iteratieve karakteristiek over twee ronden mogelijk geweest met een kans van $1/73$ in plaats van $1/234$, ongeveer driemaal zo hoog dus. Hiermee had een bruikbare karakteristiek met waarschijnlijkheid 2^{-43} over 15 ronden gemaakt kunnen worden. Op dit punt wordt zo dadelijk nog teruggekomen.

- Veranderingen in de S -doos substituties.

Er is in de voorbije jaren wel gesuggereerd dat beter willekeurig gekozen S -dozen gebruikt hadden kunnen worden dan de S -dozen waarin een achterdeur werd vermoed. In een willekeurig gekozen S -doos heeft men echter meer dan 99% kans dat twee invoeren die slechts verschillen in de middelste bits van de S -doos, dezelfde uitvoer hebben (de middelste bits zijn degene die niet via de expansie andere S -dozen beïnvloeden). Dat schept een kansrijke karakteristiek, namelijk een met kans $p = 1/8$ of meer. Er is dan een karakteristiek over 13 ronden met $p \geq (1/8)^6 = 2^{-18}$ en een aanval heeft slechts 2^{20} paren nodig om te slagen. Zelfs één enkele verandering in een S -doos kan al tot gevolg hebben dat een karakteristiek met $p = 1/32$ ter beschikking komt. Willekeurige S -dozen nemen geeft een extra boost aan de differentiële cryptoanalyse. Zelfs één enkele verandering in een S -doos kan daar al toe leiden. De substituties in de S -dozen luisteren buitengewoon nauw.

- *Weglaten van de expansie.*

Zonder de expansie zijn er geen twee extra bits om per S -doos uit vier substituties te kiezen. Dan resteren voor de rondetransformatie slechts acht $4 \rightarrow 4$ substituties en dat maakt de Data Encryption

Standard erg zwak. Reductie van het aantal substituties in de *S*-doos is dus buitengewoon onverstandig. Het nemen van alleen de eerste substitutie van de acht *S*-dozen levert een iteratieve karakteristiek over 4 ronden met $p = 1/256$ op.

- *Verwisselen van de expansie en het combineren met de sleutel.*

In dit geval zijn er 32 bits subsleutels die eerst met de klaartekst gecombineerd worden en pas daarna wordt geëxpandeerd tot 48 bits. Dat maakt de Data Encryption Standard iets zwakker. Namelijk de iteratieve karakteristiek over twee ronden krijgt dan $p = 1/146$

- *Onafhankelijke subsleutels.*

Door onafhankelijke subsleutels te nemen, krijgt men een sleutel van $16 \times 48 = 768$ bits. Dat levert de gigantische hoeveelheid van 2^{768} sleutels op en uitputtend zoeken kan dan wel worden vergeten. Eenzelfde effect weerspiegelt zich echter niet in de weerstand tegen differentiële cryptoanalyse. Die neemt in dit geval toe tot ongeveer 2^{60} stappen en vergt circa 2^{59} paren chosen-plaintext. Dat is maar een betrekkelijk geringe verbetering en het grotere aantal sleutelbits nauwelijks waard.

De conclusie van al deze experimenten is eigenlijk verrassend. De Data Encryption Standard blijkt binnen de gekozen opzet behoorlijk optimaal ontworpen en is uitstekend bestand tegen differentiële cryptoanalyse. In dat opzicht blijft weinig te wensen over.

8.3.6 Ontwerpcriteria van DES

Niet lang na de presentatie van differentiële cryptoanalyse als een nieuwe, krachtige aanvalstechniek tegen iteratieve blokgeheimsschriften, kwam de aap uit de mouw. In een bericht op het nieuwsnet en later in een artikel in IBM's research tijdschrift, maakt Don Coppersmith bekend dat men dit in 1974 bij IBM allemaal al wist en er ook duchtig rekening mee heeft gehouden bij het ontwerp van de Data Encryption Standard^[21]. Waarom niet de DES-controverse vermeden en alle ontwerpcriteria voor de *S*-dozen bekend gemaakt, in

plaats van – zoals in 1976 is gebeurd – slechts een tipje van de sluier op te lichten? Coppersmith, een van de leden van het oorspronkelijke ontwerpteam, schrijft daarover het volgende.

“After discussions with NSA, it was decided that disclosure of the design considerations would reveal the technique of differential cryptanalysis, a powerful technique that can be used against many ciphers. This in turn would weaken the competitive advantage the United States enjoyed over other countries in the field of cryptography.”

Men nam dus de twijfels op de koop toe in ruil voor het geheimhouden van differentiële cryptoanalyse. Nu die techniek onafhankelijk opnieuw was uitgevonden, kon de sluier van geheimzinnigheid worden opgelicht en onthult Coppersmith welke criteria zijn gehanteerd bij het ontwerp van de DES.

In de volgende opsomming van deze criteria is met een * aangegeven welke de criteria zijn die tot het laatst toe geheim gehouden werden. De nummering van de criteria wisselt in de literatuur, hier worden nummering en volgorde uit het artikel van Coppersmith zelf aangehouden.

- S-1 Elke S-doos heeft zes invoer- en vier uitvoerbits. Meer ging met de techniek van 1974 niet op een enkele chip.
- S-2 Geen enkel uitvoerbit van een S-doos mag te dicht bij een lineaire functie van de invoerbits liggen. Dat wil zeggen alle deelverzamelingen van de invoerbits moeten gemiddeld voor de helft nullen en enen genereren. Alle niet-lineariteit in de DES moet immers van de S-dozen komen.
- S-3 Alle vier substituties in een S-doos moeten elk van de zestien hexadecimale waarden 0...F precies één keer opleveren. In elke substitutie zijn alle 16 substitutiewaarden dus een mogelijke uitkomst.
- S-4 Als twee invoeren van een S-doos in precies één bit verschillen, moet hun respectievelijke uitvoer in tenminste twee bits verschillen.
- S-5 Als twee invoeren van een S-doos in precies de twee middelste bits verschillen – d.w.z. hun exclusive or is 001100 – moet hun respectievelijke uitvoer in tenminste twee bits verschillen. Ten koste van



Coppersmith

alles moet worden vermeden dat op enigerlei moment alle S -dozen een verschil nul als resultaat hebben en dat dit niet meer verandert, want dat is het walhalla van de differentiële cryptanalyse! Als dus een situatie ontstaat waarin een geïsoleerde S -doos een van nul verschillende invoer krijgt (omringende S -dozen wel verschil nul), mag hieruit geen verschil nul ontstaan. Samen met criterium $P-3$ zorgen dit en het vorig criterium ervoor dat verschillen ontstaan die naar alle richtingen worden verspreid.

- $S-6$ Als twee invoeren van een S -doos in precies de twee eerste bits verschillen en juist niet in de twee laatste – exclusive or is $11xy00$, de twee middelste bits doen er niet toe – moeten de respectievelijke uitvoeren verschillend zijn. Het gaat hier om het effect van twee aanliggende S -dozen die een verschillende invoer krijgen, terwijl de naast gelegen S -dozen juist een verschil van nul hebben. Doorrekenend naar de mogelijkheid dat die twee aanliggende S -dozen een uitvoerverschil van nul hebben, zorgt de hier geformuleerde voorwaarde dat dit niet kan optreden: minstens een van de twee krijgt een uitvoerverschil ongelijk nul.
- $S-7^*$ Voor elk niet-nul invoerverschil mogen niet meer dan zestien gelijke uitvoerverschillen ontstaan waardoor de aanvalsmogelijkheden beperkt worden. De reden is dat dan over alle ronden $(1/4)^n$ de maximale waarschijnlijkheid voor een karakteristiek is.
- $S-8^*$ Dit criterium is ontwikkeld voor het patroon van drie S -dozen met verschil direct naast elkaar, afwisselend gevolgd door een ronde waarin ze alle drie een verschil nul hebben. Precies het idee waarop Eli Biham en Adi Shamir hun aanval hebben gebaseerd. De op dat idee gebaseerde aanval kan niet helemaal worden voorkomen, maar wel zoveel mogelijk onschadelijk gemaakt. Daartoe zijn de S -dozen zo gunstig mogelijk gerangschikt.

Biham en Shamir's karakteristiek heeft de waarde 19600000. Wanneer dit verschil gevolgd wordt over de expansie naar de acht S -dozen komt er voor de verschillen

1	9	6	0	...	
0001	1001	0110	0000	...	binair
↓	↓	↓	↓		expansie
000011	110010	101100	000000	...	
00ab11	11cd10	10ef00	000000	...	
S_1	S_2	S_3	S_4	...	

Het optreden van deze karakteristiek kan niet worden voorkomen. Gezocht is naar die waarden voor $a, b, \dots$ waarbij de kansen op uitvoerverschillen van nul maximaal zijn. Het resultaat is het gegeven bitpatroon. Omdat het gaat om drie gecombineerde uitvoerverschillen, moeten de kansen op de individuele verschillen met elkaar vermenigvuldigd worden. Deze kansen zijn per S -doos verschillend (te zien in tabellen zoals [tabel 8.4](#)). Vervolgens zijn de S -dozen zo gerangschikt dat deze gecombineerde kans zo gering mogelijk is.

- P-1* Van de vier uitvoerbits van een S -doos gaan er in de volgende ronde twee naar middelste bits (niet gedeeld met aanliggende S -dozen) en twee naar bits aan de uiteinden (gedeeld met aanliggende S -dozen).
- P-2* De vier uitvoerbits van een S -doos beïnvloeden zes verschillende S -dozen in de volgende ronde. Twee vanuit de middelste bits, tweemaal twee van de bits aan de uiteinden via de expansie.
- P-3* Als de uitvoer van een bepaalde S -doos een middelste bit van een andere S -doos beïnvloedt, kan een uitvoerbit van die laatste niet op zijn beurt een middelste bit van de eerste beïnvloeden. Dit criterium zorgt ervoor dat zich niet de situatie van slechts één enkele S -doos met minimaal invoerverschil kan gaan voordoen. Bedenk dat de middelste bits niet met aanliggende S -dozen worden gedeeld, zodat het gevaar van gelokaliseerde verschillen die elkaar weer opheffen alleen via de middelste bits zou kunnen optreden.

8.3.7 Lineaire cryptoanalyse

Al in 1985 vestigde Adi Shamir de aandacht op het feit dat in de vijfde *S*-doos van de Data Encryption Standard een bijna-lineariteit optreedt: één van de invoerbits heeft vaak dezelfde waarde als de som van de vier uitvoerbits^[22]. Deze eigenaardigheid werd verklaard met een beroep op de (onbekende) ontwerpcriteria, die dit kennelijk tot gevolg hadden. Op dat moment was er niemand die hier iets mee aankon. Totdat in 1993 de Japanse onderzoeker Mitsuru Matsui met de opzienbarende nieuwe techniek van de *lineaire cryptoanalyse* op de proppen kwam^[23]. Niet alleen was dit een nieuwe techniek waartegen de DES niet bestendig gemaakt was, maar ook een *known-plaintext aanval* in plaats van een chosen-plaintext zoals de differentiële cryptoanalyse. En dus veel krachtiger is dan wat Eli Biham en Adi Shamir hadden laten zien.

Het principe van lineaire cryptoanalyse is betrekkelijk eenvoudig. Geprobeerd wordt een lineaire of bij benadering lineaire betrekking te vinden tussen een aantal bits uit de klaartekst, de cijfertekst en de sleutel. Ruwweg gezegd komt het er op neer dat de gekozen bits bij elkaar opgeteld vaker een 0 dan een 1 opleveren (of omgekeerd natuurlijk, vaker een 1 dan een 0). Matsui baseerde zijn aanvalsmethode op het feit dat voor antwoord 0 en 1 de kansen niet precies gelijk liggen, maar een klein beetje daarvan afwijken. Dat kleine verschil brengt een klein beetje lineariteit met zich mee. Het bij elkaar optellen van de uitgekozen bits tezamen met het meest voorkomende antwoord, geeft lineaire vergelijkingen die in net iets meer dan de helft van de gevallen kloppen. Als men er maar genoeg van kan vinden kunnen ze worden opgelost, precies wat Matsui beoogde.

$$M[m_1, m_2, \dots, m_a] \oplus C[c_1, c_2, \dots, c_b] = K[k_1, k_2, \dots, k_c] \quad (8.30)$$

De m_i , c_i en k_i zijn bitposities in respectievelijk M , C en K . De uitdrukking $M[\dots]$ staat voor

$$M[m_1, m_2, \dots, m_a] = M_{m_1} \oplus M_{m_2} \oplus \dots \oplus M_{m_a} \quad (8.31)$$

Men kan deze uitdrukking ook anders formuleren, namelijk met een vector die bits selecteert en door

het inwendig product van een bitrij met deze vector te nemen. In de volgende formule is optelling in de sommatie de exclusive or operator, de vermenigvuldiging de logische and-operator en $X \cdot M$ het inwendig product van X en M

$$\begin{aligned} X \cdot M &= \sum_{i=0}^{n-1} x_i \cdot m_i \\ X &= \{x_0, x_1, \dots, x_{n-1}\} \\ M &= \{m_0, m_1, \dots, m_{n-1}\} \end{aligned} \quad (8.32)$$

Een goed geheimschrift is juist niet-lineair. Maar het ideaal van volkomen niet-lineariteit – hetgeen wil zeggen gemiddeld precies even vaak een kloppende als een niet-kloppende vergelijking – wordt ook niet gehaald. Meestal is de kans dat de vergelijking opgaat enigszins verschillend van $1/2$. Voor kans groter dan $1/2$ is er meer kans dat de vergelijking opgaat, terwijl deze voor kans kleiner dan $1/2$ minder vaak opgaat. Hoe meer die kans van $1/2$ verschilt des te uitgesprokener de lineariteit en des te meer kans dat op deze manier bits van de sleutel gevonden kunnen worden.

Het principe dat Matsui voor zijn aanval op de Data Encryption Standard gebruikt is dus vrij eenvoudig. Om er een bruikbare aanval mee op te bouwen blijkt minder simpel. Het eerste probleem is het vinden van geschikte bitcombinaties met een zo groot mogelijke lineariteit, daarna komt nog de praktische uitvoering die ook niet direct eenvoudig is.

8.3.8 MASTERSECTIE – Lineaire cryptoanalyse verklaard

Het voorbeeld-cryptosysteem uit de behandeling van de differentiële cryptoanalyse wordt ook hier gebruikt. De eerste stap is te bepalen welke combinaties van invoer- en uitvoerbits van de S -dozen de meeste lineariteit vertonen. Daartoe wordt voor alle bitkeuzes van de invoer en uitvoer de lineariteit van de combinatie doorgerekend. **Tabel 8.7** geeft het resultaat voor de hier gebruikte S -doos S_1 . De coördinaten stellen hier wel iets anders voor dan in de vergelijkbare **tabel 8.4**. Daar zijn het bitwaarden voor invoer en uitvoer, hier zijn het de bitpatronen X uit **vergelijking (8.32)**. De waarden in de tabel zijn berek-

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	-2	-2	0	0	-2	6	2	2	0	0	2	2	0	0
2	0	0	-2	-2	0	0	-2	-2	0	0	2	2	0	0	-6	2
3	0	0	0	0	0	0	0	0	2	-6	-2	-2	2	2	-2	-2
4	0	2	0	-2	-2	-4	-2	0	0	-2	0	2	2	-4	2	0
5	0	-2	-2	0	-2	0	4	2	-2	0	-4	2	0	-2	-2	0
6	0	2	-2	4	2	0	0	2	0	-2	2	4	-2	0	0	-2
7	0	-2	0	2	2	-4	2	0	-2	0	2	0	4	2	0	2
8	0	0	0	0	0	0	0	0	-2	2	2	-2	2	-2	-2	-6
9	0	0	-2	-2	0	0	-2	-2	-4	0	-2	2	0	4	2	-2
A	0	4	-2	2	-4	0	2	-2	2	2	0	0	2	2	0	0
B	0	4	0	-4	4	0	4	0	0	0	0	0	0	0	0	0
C	0	-2	4	-2	-2	0	2	0	2	0	2	4	0	2	0	-2
D	0	2	2	0	-2	4	0	2	-4	-2	2	0	2	0	0	2
E	0	2	2	0	-2	-4	0	2	-2	0	0	-2	-4	2	-2	0
F	0	-2	-4	-2	-2	0	2	0	0	-2	4	-2	-2	0	2	0

Tabel 8.7 Lineariteit van S-doos S_1 voor alle bitcombinaties

end door te tellen hoe vaak de som van de gekozen invoerbits (kolom-coördinaat) gelijk is aan de som van de gekozen uitvoerbits (rij-coördinaat)

$$IN[i_1i_2i_3i_4] = UIT[j_1j_2j_3j_4], \quad i_1i_2i_3i_4, j_1j_2j_3j_4 = [0000], \dots, [1111] \quad (8.33)$$

Er zijn 16 invoerwaarden zodat een telling van 8 overeenkomt met een neutraal, dat wil zeggen volkomen niet-lineair gedrag. Van alle waarden is 8 afgetrokken, waardoor dit neutrale gedrag een tabelwaarde 0 krijgt. De maximale waarde +8 staat nu voor volledige lineariteit (voor alle relaties geldt het = teken), terwijl het minimum van -8 het tegenovergestelde aangeeft. (0,0) is het triviale geval waarin geen enkel bit wordt meegenomen. De tabelwaarde van bijvoorbeeld (1,7) bedraagt 6, waaruit kan worden afgeleid dat met $p = (6+8)/16 = 14/16$ zal gelden dat $IN[0001] = UIT[0111]$ en de -4 van (F,2) laat zien dat $p =$

4/16 voor $IN[1111] = UIT[0010]$.

Om te zien hoe deze betrekkingen gebruikt kunnen worden voor het bepalen van sleutelbits, wordt een vercijfering over drie ronden uitgewerkt en daarbij de $S_1(1,7)$ lineariteit benut. Aangezien S -doos S_1 als invoer de bits op de plaatsen 7654 heeft, is bit 4 het invoerbit overeenkomend met het bitpatroon 1 van de (1,7)-combinatie. De meegetelde uitvoerbits (de 7) staan op de plaatsen 654, maar door **permutatie (8.12)** komen deze in de uitvoer van de rondetransformatie op de plaatsen 471 terecht. In ronde 1 is R_0 de invoer en dus is $R_0[4]$ de invoer van S_1 . Dit bit wordt gecombineerd met het overeenkomstige sleutelbit $K_1[4]$ en tenslotte worden de uit de transformatie komende bits $F(R_0, K_1)[1,4,7]$ daarbij opgeteld. Zodat de complete vergelijking voor de eerste ronde luidt

$$\underbrace{R_0[4] \oplus K_1[4]}_{IN} \oplus \underbrace{F(R_0, K_1)[1,4,7]}_{UIT} = 0 \quad (8.34)$$

Voor de tweede ronde wordt niets ingevuld, voor de derde ronde is dezelfde betrekking opnieuw gebruikt. Denk er echter aan dat het de *subsleutel* is waarvan bit 4 wordt gebruikt. Als gevolg van de sleutelrotatie (per ronde drie posities naar links) is dit een *ander sleutelbit* dan in de eerste ronde.

$$R_2[4] \oplus K_3[4] \oplus F(R_2, K_3)[1,4,7] = 0 \quad (8.35)$$

De combinatie van **vergelijking (8.34)** met **(8.35)** kan als volgt worden vereenvoudigd

$$\begin{aligned} R_0[4] \oplus K_1[4] \oplus F(R_0, K_1)[1,4,7] &= 0 \\ &\rightarrow L_0[1,4,7] \oplus R_1[1,4,7] \\ R_2[4] \oplus K_3[4] \oplus F(R_2, K_3)[1,4,7] &= 0 \\ &\rightarrow L_2[1,4,7] \oplus R_3[1,4,7] \\ &\rightarrow R_1[1,4,7] \oplus R_3[1,4,7] \end{aligned} \quad (8.36)$$

Gebruik gemaakt is van het feit dat $F(R_{i-1}, K_i) = L_{i-1} \oplus R_i$, waarna door optellen van beide vergelijkingen de termen $R_1[1,4,7]$ er uit vallen

$$L_0[1, 4, 7] \oplus R_0[4] \oplus R_2[4] \oplus R_3[1, 4, 7] = K_1[4] \oplus K_3[4] \quad (8.37)$$

De overblijvende termen kunnen respectievelijk in de klaartekst en in de cijfertekst worden gevonden (R_2 komt terug in L_3) en met elkaar gecombineerd (aan het eind van de vercijfering zijn L en R verwisseld).

$$M[4, 9, 12, 15] \oplus C[4, 9, 12, 15] = K[4, 10] \quad (8.38)$$

Hoeveel kans is er dat deze vergelijking klopt? Ze komt in twee gevallen uit. Namelijk wanneer zowel de eerste als de tweede samenstellende vergelijking opgaat én wanneer ze geen van beide opgaan. Met twee deelnemende vergelijkingen p_1 en p_2 is er zodoende een totale kans van

$$p = p_1 p_2 + (1 - p_1)(1 - p_2) \quad (8.39)$$

Aangenomen dat de betrekking in [vergelijking \(8.36\)](#) een kans $p = 14/16$ heeft (blijkens $S_1(1,7) = 6$ in [tabel 8.7](#)), is de totale kans over drie ronden $p \approx 0,78$ voor het voldoen aan [vergelijking \(8.38\)](#). In een experiment met willekeurige vercijferingen kan dit eenvoudig geverifieerd worden. In het algemeen geldt voor n gekoppelde betrekkingen met respectievelijke kansen p_i een totale kans van

$$p = 1/2 + 2^{n-1} \prod_{i=1}^n (p_i - 1/2) \quad (8.40)$$

Aangetoond is nu dat een bij benadering lineaire betrekking kan worden gevonden tussen bits van de klaartekst, de cijfertekst en de sleutel. Zo'n benadering is bruikbaar als de kans daarop voldoende van 0,5 verschilt en in samenhang daarmee genoeg bekende klaartekst/cijfertekst-paren ter beschikking staan. Het aantal benodigde paren is afhankelijk van de gewenste betrouwbaarheid in de conclusie. Voor zijn DES-analyse berekent Matsui een slaagkans van 96,7% of beter bij een aantal van $8 \times |p - 1/2|^{-2}$ paren.

Tot hier kan het resultaat van de analyse nog niet bijster indrukwekkend worden genoemd. Het vinden van één sleutelbit c.q. som van sleutelbits, is voor de 56 bits van de DES-sleutel nog ver van een complete oplossing verwijderd*. Aan de hand van een analyse over zes ronden wordt nu getoond hoe deze ene betrekking kan worden benut om een groter deel van de sleutel te bepalen.

* In een aantal gevallen kan op grond van symmetriebeschouwingen nog een tweede bit worden bepaald.

Eerst is uit een tweetal betrekkingen een karakteristiek opgesteld – gebaseerd op $S_1(2,E)$ en $S_2(F,2)$. Een over vijf ronden doorlopende combinatie kan hieruit gevormd worden, omdat ze middels een gemeenschappelijk bit aan elkaar kunnen worden gekoppeld

$$\begin{aligned}
 1: & R_0[0, 1, 2, 3] \oplus F(R_0, K_1)[5] = K_1[0, 1, 2, 3] \\
 2: & R_1[5] \oplus F(R_1, K_2)[2, 4, 7] = K_2[5] \\
 3: & - \\
 4: & R_3[5] \oplus F(R_3, K_4)[2, 4, 7] = K_4[5] \\
 5: & R_4[0, 1, 2, 3] \oplus F(R_4, K_5)[5] = K_5[0, 1, 2, 3]
 \end{aligned} \tag{8.41}$$

Omdat beide betrekkingen een kans van 14/16 hebben, kan met **vergelijking (8.40)** de totale kans voor deze karakteristiek worden berekend op $p = 0,658$. Na substitutie van F zoals in **vergelijking (8.36)** en eliminatie van tussentermen wordt de volgende betrekking verkregen

$$\begin{aligned}
 & R_0[0, 1, 2, 3] \oplus L_0[5] \oplus L_1[2, 4, 7] \oplus R_4[0, 1, 3, 4, 7] \oplus R_5[5] \\
 & = K_1[0, 1, 2, 3] \oplus K_2[5] \oplus K_4[5] \oplus K_5[0, 1, 2, 3]
 \end{aligned} \tag{8.42}$$

Om iets met deze betrekking te kunnen aanvangen is kennis van L_1 , R_4 en R_5 nodig. Met L_1 en R_5 is geen probleem vanwege de gelijkheden $L_1 = R_0$ en $L_6 = R_5$, welke direct te vinden zijn in respectievelijk de klaartekst en de cijfertekst. R_4 echter moet worden gevonden uit L_5 en om die te kunnen bepalen is een ontcijferstap met de laatste subsleutel nodig

$$R_4 = L_5 = R_6 \oplus F(L_6, K_6) \tag{8.43}$$

Daarmee wordt **vergelijking (8.42)**

$$\begin{aligned}
 & R_0[0, 1, 3, 4, 7] \oplus L_0[5] \oplus \{R_6 \oplus F(L_6, K_6)\}[0, 1, 3, 4, 7] \oplus L_6[5] \\
 & = K_1[0, 1, 2, 3] \oplus K_2[5] \oplus K_4[5] \oplus K_5[0, 1, 2, 3]
 \end{aligned} \tag{8.44}$$

Invullen van $M = (L_0, R_0)$, $C = (L_6, R_6)$ en terugrekenen van de subsleutelbits naar posities in de complete sleutel, levert nu de betrekking*

* De bits in M en C verschillen 8 in plaatsnummer als gevolg van de verwisseling van links en rechts in de laatste vercijferstap.

$$\begin{aligned}
 M[0, 1, 3, 4, 7, 13] \oplus C[5, 8, 9, 11, 12, 15] \oplus F(L_6, K_6)[0, 1, 3, 4, 7] \\
 = K[0, 1, 3, 4, 5, 6, 7, 12]
 \end{aligned}
 \tag{8.45}$$

Aangezien de subsleutel onbekend is, kan $F(L_6, K_6)$ slechts door proberen worden uitgerekend. Gelukkig telt de subsleutel minder bits dan de gehele sleutel en zelfs dat is vaak nog teveel. Alleen de subsleutelbits die direct verantwoordelijk zijn voor het bepalen van de benodigde bitselectie tellen mee, de andere bits van de subsleutel doen er niet toe. In het voorbeeld doen de twee S -dozen mee in de berekening van bits $[0, 1, 3, 4, 7]$ en zijn er 8 subsleutelbits bij betrokken.

Als de subsleutelbits onjuist gekozen zijn, zal [vergelijking \(8.45\)](#) bij invullen gemiddeld een willekeurig resultaat opleveren: 0 en 1 in ongeveer gelijke aantallen. Hebben echter de gebruikte subsleutelbits de goede waarde, dan is het resultaat niet willekeurig maar geeft met kans p een overmaat van 0 of 1. Zo kan dus onderscheid gemaakt worden tussen goede en foute sleutels. Uit een kansberekening moet blijken hoeveel paren klaartekst-cijfertekst nodig zijn om met voldoende statistische betrouwbaarheid een conclusie te kunnen trekken. Gegeven de kans van ongeveer 0,675 voor het opgaan van [vergelijking \(8.45\)](#) bij een goede subsleutelkeuze, blijken in het onderhavige voorbeeld 160 paren genoeg voor het onderscheiden van de oplossing.

Alles is nu gereed om te zien hoe Matsui de DES heeft kunnen breken. De karakteristiek die hij gebruikt is in hoofdzaak gebaseerd op een reeks van drie betrekkingen, die geïtereerd kunnen worden omdat de tussenliggende termen tegen elkaar wegvallen. Deze drie zijn*.

$$\begin{aligned}
 R[15] \oplus F(R, K)[7, 18, 24] &= K[22] & p &= 42/64 \\
 R[29] \oplus F(R, K)[15] &= K[44] & p &= 30/64 \\
 R[15] \oplus F(R, K)[7, 18, 24, 29] &= K[22] & p &= 12/64
 \end{aligned}
 \tag{8.46}$$

Net als in [vergelijking \(8.41\)](#) kunnen deze groepen worden geconcateneerd door ze in tegengestelde vol-

* De bitnummering is dezelfde als in het voorbeeld, namelijk 0...63 van rechts naar links; anders dan de bitnummering in de officiële DES-publicatie en de meeste literatuur. Bij het narekenen van de bitposities moet rekening worden gehouden met de diverse permutaties, de expansie en het subsleutelschema van DES.

gorde aan elkaar te koppelen met een lege tussenliggende ronde. De laatste van de drie is de al eerder genoemde meest lineaire benadering uit S -doos S_5 . De eerste betrekking vertoont ook een behoorlijke kans, maar voor de middelste is er geen keuze en moet een lage waarschijnlijkheid op de koop toe genomen worden (30/64 verschilt weinig van $0,5 = 32/64$). Met een extra betrekking aan het begin en aan het eind wordt een karakteristiek over 15 ronden gecompleteerd. De gehele betrekking tussen de onderscheiden bits luidt

$$\begin{aligned} M[12, 16, 39, 40, 56] \oplus C[7, 18, 24, 29, 47] \oplus F_{16}(C_R, K_{16})[15] = \\ K_1[19, 23] \oplus K_3[22] \oplus K_4[22] \oplus K_5[22] \oplus K_7[22] \oplus \\ K_8[44] \oplus K_9[22] \oplus K_{11}[22] \oplus K_{12}[44] \oplus K_{13}[22] \oplus K_{15}[22] \end{aligned} \quad (8.47)$$

De totale kans van Matsui's schema bedraagt $0,5 + 1,19 \times 2^{-22}$, leidend tot een benodigd aantal van $8 \times |1/2 - p|^{-2} \approx 2^{47}$ paren klaartekst-cijfertekst voor voldoende betrouwbaarheid in de conclusie. De zestiende en laatste ronde wordt vervangen door een ontcijferstap waarvoor alleen de invoer van S_5 nodig is. Daardoor behoeven slechts de 64 combinaties van de met de invoer van S_5 overeenkomende sub-sleutelbits te worden geteld. Tezamen met het rechterlid van [vergelijking \(8.47\)](#) levert dit 7 bits van de sleutel. Omdat een aan de ontcijferstap analoge operatie ook aan het begin kan worden uitgevoerd, worden zo nog eens 7 bits gevonden. Met in totaal 14 bekende sleutelbits kunnen de 42 resterende bits door uitputtend zoeken bepaald worden.

Tenslotte is Matsui nog nagegaan of een andere keuze voor de volgorde in de S -dozen beter geweest zou zijn. Hij komt tot de conclusie dat in plaats van de volgorde 12345678 de volgorde 56412738 tegen lineaire cryptanalyse de beste van alle 40320 mogelijkheden is. De kans voor de gehele karakteristiek is ongeveer een factor 2^{-9} kleiner. Die combinatie is echter weer slechter met betrekking tot differentiële cryptanalyse. Een betere volgorde tegen zowel differentiële als lineaire cryptanalyse blijkt 24673158. Ook deze combinatie degradeert de hier beschreven aanval tot slechter dan uitputtend zoeken^[24].

8.3.9 Side-Channel-analyse

Men kan zich blind te staren op alle mogelijke technieken voor het breken van een geheimschrift om dan te vergeten dat het ook nog eens concreet geïmplementeerd moet worden op een computer of smartcard. Niet alleen de vercijferalgoritme zelf maar ook die implementatie biedt mogelijkheden voor cryptoanalyse. Men noemt dit soort aanvallen *side-channel-analyse*. Zo'n aanval kan in bepaalde gevallen tegen geringe kosten en met bescheiden apparatuur worden opgezet en kan vooral tegen smartcards zeer effectief zijn. Er is weinig reden om zich te verheugen in een superieur geheimschrift, wanneer de gebruikte sleutel zonder veel moeite aan de smartcard kan worden ontfutseld.

Eind 1995 deed Paul Kocher de wereld kond van deze nieuwe manier om een geheimschrift aan te vallen. Hij mat hoeveel tijd met het vercijferen is gemoeid^[25]. De achterliggende gedachte is dat de verwerkingstijd niet voor elke vercijferoperatie constant zal zijn, maar zal variëren met de verwerkte gegevens inclusief de sleutel. Dit kan veroorzaakt worden door vertakkingen in de programmacode waardoor meer of andere processorinstructies worden uitgevoerd, lussen die meer of minder vaak worden doorlopen, of het kan het gevolg zijn van operaties die de processor niet voor alle invoerwaarden in een constante tijd uitvoert.

Hoeveel kansen dit de cryptoanalist biedt hangt af van de mate waarin deze effecten doorwerken, iets wat per geheimschrift nogal wisselt. De Data Encryption Standard biedt bijvoorbeeld minder houvast voor deze soort analyse dan RSA. Maar toch is men er in geslaagd om uit het totale tijdsverloop van de vercijfering ten naaste bij het aantal enen in een DES-sleutel te bepalen^[26]. Zulke aanvallen zijn kansrijker als de subsleutels voor elke vercijfering opnieuw moeten worden gegenereerd dan wanneer ze tevoren berekend en in een geheugen opgeslagen zijn.

Een andere methode is de in 1997 door Eli Biham en Adi Shamir geïntroduceerde differential fault analysis^[27]. Deze aanvalstechniek berust op het injecteren van fouten in de verwerking van individuele vercijferoperaties. Zulke fouten kunnen worden veroorzaakt door een vercijferchip bloot te stellen aan ionis-



Kocher

erende straling. Vooral fouten die in de laatste ronde(n) worden opgewekt laten informatie over de sleutel uitlekken. Een staalkaart van hetgeen langs deze lijnen mogelijk is kan in de literatuur worden gevonden, evenals hetgeen allemaal kan na het oplossen van chipcard-omhulsels en het inzetten van instrumenten als elektronenmicroscopen en UV-lasers^[28].

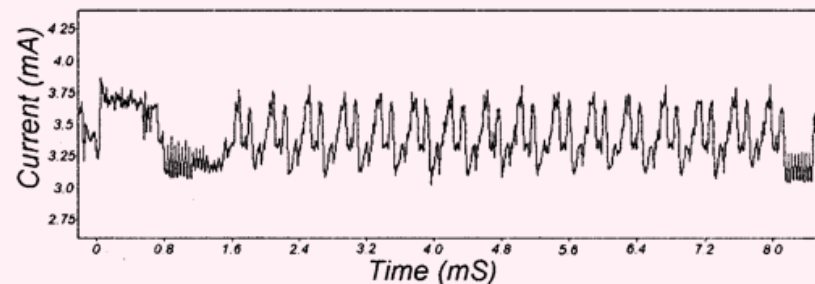
Een derde en zeer effectief gebleken methode is het meten van het stroomverbruik van een chipkaart in werking^[29]. Dat blijkt zeer eenvoudig. Een weerstand van 50 Ohm wordt in serie gezet met de spanningsbron of de aarde en de door de chip getrokken stroom wordt gemeten met de spanningsval over deze weerstand. Zulke metingen kunnen met hoge nauwkeurigheid en zeer hoge meetfrequentie worden uitgevoerd en ook nog met vrij goedkope apparatuur. De mate van detail die hiermee omtrent de vercijferoperatie kan worden verkregen is fenomenaal. In [figuur 8.7](#) ziet men het stroomverbruik van een chipkaart tijdens een DES-vercijfering. De 16 rondes door het Feistel-netwerk kunnen uitstekend worden waargenomen. Maar er kan met nog veel hogere resolutie worden gemeten. Gedetailleerde metingen laten zien in welke rondes het subsleutelschema 1 of 2 stappen neemt en bij nog groter oplossend vermogen wordt zichtbaar wanneer in de programmacode een sprong al dan niet wordt genomen.



Biham



Shamir



Figuur 8.7 Stroomverbruik tijdens DES vercijfering

De effecten veroorzaakt door het verwerken van de programma-instructies zijn duidelijker dan die door verschillen in de waarde van de verwerkte data. Deze laatste effecten bevinden zich op de grens van de

meetnauwkeurigheid en verdrinken bijna in de ruis. Toch is het mogelijk gebleken om met behulp van stroommeting direct sleutelbits te bepalen door een flink aantal metingen te middelen.

De aanvallen middels de executietijd en de stroomconsumptie moeten niet worden onderschat, vooral niet bij toepassingen in smartcards en dergelijke. Uit het feit dat hieraan expliciet aandacht geschonken is in het programma van eisen voor de ontwikkeling van de Advanced Encryption Standard, de opvolger van de DES, kan men afleiden hoe serieus deze bedreiging wordt genomen^[30]. Aanvallen kunnen zowel gericht zijn op de encryptie als geheel of speciaal op het mechanisme voor het genereren van de subsleutels^[31]. Verdedigingsmechanismen variëren van het vermijden van bepaalde instructies (vooral spronginstructies) tot het maskeren van de effecten door operaties tweemaal uit te voeren, de tweede maal met het complement van de eerste waarde (software balancing).

Men kan de computerinstructies in dit verband als volgt classificeren

- opzoeken in tabellen (adresseren), vaste verschuivingen, rotaties en logische (booleaanse) operaties zijn ongevoelig voor timing analyse, stroomverbruik kan worden gemaskeerd met complement berekeningen
- optellen, aftrekken zijn enigermate lastig te maskeren
- vermenigvuldigen, delen, kwadrateren of variabele verschuivingen en rotaties zijn zeer moeilijk te maskeren

Een extra verdedigingsmechanisme kan worden gevonden in het aanbrengen van aanvullende hardware die ruis toevoegt en daardoor de metingen bemoeilijkt.

8.4 Skipjack

Onder de alternatieven voor de Data Encryption Standard als geheimschrift voor algemeen gebruik neemt de Skipjack-algoritme een bijzondere plaats in. Er zijn twee redenen om dit hier te behandelen. De eerste

is dat Skipjack een product van de National Security Agency is. Dit in tegenstelling tot de Data Encryption Standard waarmee NSA wel bemoeienis heeft gehad maar geen leidende rol als ontwikkelaar speelde. De tweede reden is de speciale setting waarin dit algoritme is gelanceerd en het daaropvolgende politieke debat.

In het begin van de jaren negentig raakten opsporingsinstanties als de FBI bezorgd over hun mogelijkheden om informatie te vergaren, in het bijzonder uit afgeluisterde telefoongesprekken*. Een van de zorgpunten vormde de toenemende verkrijgbaarheid van goedkope apparatuur voor encryptie van telefoongesprekken en ander telefonisch dataverkeer. De Amerikaanse justitiële autoriteiten achtten de tijd rijp voor een wettelijke regeling van digitale telefonie, waarmee tevens het gebruik van cryptografie kon worden gereguleerd. Een wetsvoorstel moest voorkomen dat de nieuwe technologieën de af luistermogelijkheden zouden beperken en verplichtte de providers van communicatiediensten tot het treffen van voorzieningen daarvoor^[32].

Als mogelijkheden voor ongehinderd af luisteren komen enkele alternatieven in aanmerking

1. *voorschrijven van zwakke cryptografie*, waardoor de berichten door de opsporingsautoriteiten naar behoefte (lieft in real time) kunnen worden gebroken
2. *escrow van sleutels*, waarbij iedere gebruiker zijn eigen sleutel deponeert bij een onafhankelijke bewaarder[†]
3. *directe toegang tot de sessiesleutels*, de provider heeft slechts de sessiesleutel aan de autoriteiten te overhandigen om af luisteren van het gesprek mogelijk te maken[‡]

* Die bezorgdheid is met het verstrijken van de jaren niet verdwenen, integendeel, want in 2015 is het debat weer flink opgelaaid mede als gevolg van de Snowden-onthullingen en diverse terroristische aanslagen.

† Escrow kan worden omschreven als het deponeren van iets bij een derde persoon, die dit bij het vervullen van een bepaalde voorwaarde overhandigt aan een begunstigde partij.

‡ Voorkomen moet worden dat teveel materiaal onder een en dezelfde sleutel wordt gecijferd, omdat dit de kans op een succesvolle cryptanalyse vergroot. Daarom wordt elke conversatie gecijferd met een aparte sleutel, de sessiesleutel. De hoofd-

Het dilemma "als encryptie gecriminaliseerd wordt, zullen alleen criminelen nog over encryptie beschikken", wordt afgedaan met het argument dat afluisteren standaard in de verkrijgbare telefoons zelf zal zijn aangebracht*.

Een concreet initiatief werd op 16 april 1993 gepresenteerd op een persconferentie in het Witte Huis. Op basis van vrijwillige samenwerking tussen regering en industrie werd een programma aangekondigd voor het vergroten van de veiligheid en privacy van telefonische communicatie onder behoud van de afluistermogelijkheden door de opsporingsautoriteiten.

Het voorgestelde systeem bestaat uit twee componenten. De eerste is een in telefoons, faxen, en dergelijke in te bouwen vercijfereenheid, de zogenaamde *Clipper Chip*. Het vercijferalgoritme in deze chip is door de NSA ontworpen en wordt Skipjack genoemd. De details van Skipjack worden geclassificeerd als "Secret, not releasable to foreign nationals" en productie van de chips blijft daarom voorbehouden aan bedrijven die over de benodigde *clearance* beschikken. Deze geheimhouding wordt gemotiveerd met de noodzaak om te voorkomen dat ongeautoriseerde implementaties het hierna te beschrijven mechanisme kunnen omzeilen en om geen informatie te laten uitlekken over verwante algoritmen die voor geclassificeerde informatie worden gebruikt. Een select gezelschap van deskundigen mocht zich bij de NSA een maand lang buigen over het ontwerp en zich van de kwaliteit ervan vergewissen^[33].

De tweede component van het Clipper Chip systeem is een escrow-mechanisme voor de sessiesleutels "to ensure that the Clipper Chip is used to protect the privacy of law-abiding Americans." Elke chip bevat daartoe een ingebouwde, geheime sleutel waarmee de sessiesleutels van telefoongesprekken kunnen

sleutel hoeft slechts gebruikt te worden voor het beveiligen van de sessiesleutels.

- * Ook in Nederland heeft men een poging gedaan om "wilde cryptografie" uit te bannen. Begin 1994 deed een voorontwerp van wet de ronde, inhoudende een "Wijziging van de Wet op de telecommunicatievoorzieningen in verband met de regeling van cryptografie,... in het belang van de bestrijding van criminaliteit en de veiligheid van de staat". Met als een van de hoogtepunten een artikel luidende: "Het is verboden om cryptografie voorhanden te hebben of te gebruiken." Dit door critici als nogal naïef bestempelde ontwerp, is een stille dood gestorven.

worden geëxtraheerd. Deze sleutel wordt – in tweeën opgedeeld – in afzonderlijke escrow-databases bewaard, waaruit ze door de autoriteiten met toestemming van de rechter kunnen worden opgevraagd.

MASTERSECTIE – Implementatie van de Key Escrow

Deze sleutel-escrow is als volgt geïmplementeerd. Bij de start van een verbinding wisselen de betrokken telefoons een sessiesleutel uit; hoe ze dat doen is niet aan de Clipper Chip gebonden. Echter deze chip voegt aan de gecijferde datastroom een belangrijk gegeven toe, het zogenaamde Law Enforcement Access Field (LEAF), waarin de sessiesleutel in gecijferde vorm is opgenomen. Het LEAF telt 128 bits en bevat

$$E((E(KS, KU) + UID + EA), KF) \quad (8.48)$$

met de volgende componenten:

- KS* Session Key, opgeslagen gecijferd onder *KU*
- KU* Device Unique Key, een sleutel speciaal voor de chip
- UID* Device Unique Identifier, het serienummer van de chip
- EA* Escrow Authenticator, een checksum voor het LEAF
- KF* Family Key, gemeenschappelijke sleutel voor alle chips waarmee de data in het LEAF zijn gecijferd

Bij de productie wordt een batch chips voorzien van de geheime en per chip unieke sleutel *KU*. Daartoe komen twee vertegenwoordigers van twee afzonderlijke escrow-organisaties opdraven. Zij dragen elk 80 bits aan voor de startwaarde van een toevalsgenerator. Deze produceert voor elke chip een randomgetal waaraan het serienummer wordt toegevoegd, waarna dit zodanig in twee delen wordt gesplitst dat $KU_1 \oplus KU_2 = KU$. De beide vertegenwoordigers krijgen elk een floppy disk mee met daarop een sleutelheft en de corresponderende lijst van serienummers. Een derde floppy disk met de complete sleutels gaat de fabriek in voor de fabricage van de chips en behoort direct daarna te worden vernietigd.

Tezamen met een procedure voor het authentifieren vormt de Clipper Chip de *Escrowed Encryption Standard*, ingebouwd in de Capstone chip en onder meer geleverd op PCMCIA kaarten voor personal comput-

ers^[34]. Het authenticeren dient om ervoor te zorgen dat vercijferoperaties alleen plaatsvinden wanneer een geldig LEAF wordt meegezonden en de *EA* checksum is bedoeld om dit te garanderen.

Wanneer een conversatie moet worden afgeluisterd door de FBI gaat deze op de volgende manier te werk. Allereerst wordt de *KF*-sleutel gebruikt om het LEAF te ontcijferen. Deze sleutel is dus eigenlijk niet meer dan een voorzorg om al te gemakkelijke toegang tot het LEAF te verhinderen en om de daarin opgenomen checksum te beschermen. Met het aldus leesbaar gemaakte serienummer *UID* gaat de FBI naar de rechter en vraagt om toestemming tot afluisteren. Met het verkregen toegangsbewijs tot de beide escrow-databases worden KU_1 en KU_2 aan de hand van het serienummer opgevraagd. Samen vormen ze *KU* en vanaf dat moment kan de FBI alle met deze chip vercijferde conversaties ontcijferen. Wanneer alles volgens het boekje gaat vervalt op een gegeven moment de toestemming tot afluisteren en houdt de FBI daarmee op, vernietigt ze haar versie van *KU* en wordt de eigenaar van het apparaat op de hoogte gebracht. Deze eigenaar moet nu verder maar vertrouwen op de integriteit van de FBI of een nieuw apparaat kopen.

Het Clipper Chip-initiatief kreeg geen bijzonder hartelijk onthaal en er kwam nogal wat kritiek met vragen naar de geheime Skipjack-algoritme, de onduidelijkheid over welke instanties de escrow-databases zouden gaan beheren, de traceerbaarheid van elke chip door het ingebakken serienummer, de beperkte exportmogelijkheden (nog afgezien van de kwetsieuze bereidwilligheid in het buitenland om apparaten te kopen die de Amerikaanse regering laat meeluisteren), de verhouding tot de wettelijke grondrechten vastgelegd in de Constitutie, gebruik op basis van vrijwilligheid (dus waarom zou de crimineel om wie het allemaal begonnen was er überhaupt gebruik van maken), enz.

Een succes is de Clipper Chip nooit geworden. Al was het alleen al omdat er snel aanvallen verschenen die het LEAF-veld waardeloos maakten. Een groep bekende cryptografen en computerwetenschappers publiceerde in 1998 een uiterst kritisch rapport over Key Escrow en de Clipper Chip in het bijzonder^[35].

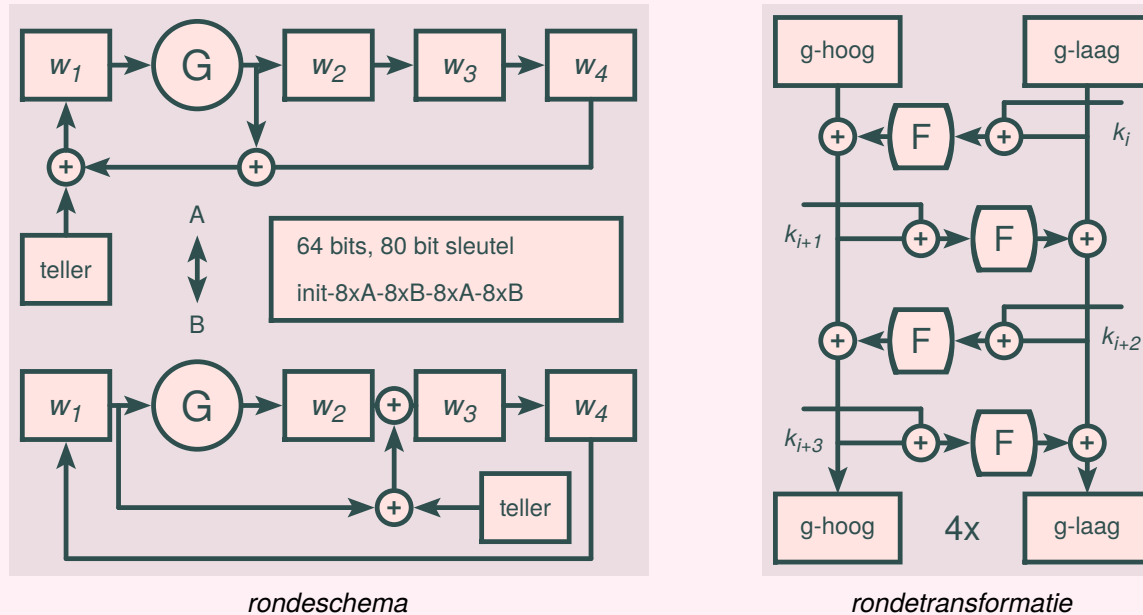
“This is not an abstract concern. Most of the key recovery or key escrow proposals made to date, including those designed by the National Security Agency, have had weaknesses discovered after

their initial implementation. For example, since the system's introduction in 1993, several failures have been discovered in the U.S. Escrowed Encryption Standard, the system on which the "Clipper Chip" is based. These problems are not a result of incompetence on the part of the system's designers. [...] The reason the Escrowed Encryption Standard had flaws is that good security is an extremely difficult technical problem to start with, and key recovery adds enormous complications with requirements unlike anything previously encountered."

Op 24 juni 1998 maakte de NSA in een persbericht de declassificatie van Skipjack bekend, alsmede de tot op dat moment eveneens geheim gehouden algoritme voor de sleuteluitwisseling, de Key Exchange Algorithm (KEA). Deze bekendmaking werd gemotiveerd met de behoefte om alternatieve implementaties in hardware en software mogelijk te maken. Het was niet de bedoeling om Skipjack naar voren te schuiven als de beoogde opvolger van de Data Encryption Standard*.

Skipjack is een blokgeheimschrift met een 80-bits sleutel en blokken van 64 bits. Het is maar ten dele een Feistel-geheimschrift. De structuur vertoont ook enige verwantschap met de in [hoofdstuk 10](#) te behandelen teruggekoppelde schuifregisters. De werking is verdeeld over twee, enigszins van elkaar verschillende vercijferoperaties A en B . Te beginnen met stap A wisselen deze elkaar telkens na acht keer af, dus $8 \times A$, $8 \times B$, $8 \times A$ en $8 \times B$; de gehele vercijfering telt 32 ronden. Omdat echter in elke stap slechts een kwart van de bits wordt behandeld – in tegenstelling tot de DES waar per stap het halve blok wordt behandeld – kunnen 32 ronden Skipjack gevoeglijk worden vergeleken met 16 ronden DES. Links in [figuur 8.8](#) zijn de operaties in de beide stappen geschetst. De invoer van 64 bits wordt verdeeld over vier 16-bits registers w_1 , w_2 , w_3 en w_4 , een vijfde register bevat een rondeteller, die achtereenvolgens de waarden 1–32 doorloopt. Na de laatste ronde worden de vier w 's geconcateneerd tot een 64-bits uitvoerblok.

* Het ligt voor de hand om te veronderstellen dat het gebruik van de Clipper chip achterbleef bij de verwachtingen en daarom een aanvullende impuls wel kon gebruiken.



Figuur 8.8 Skipjack algoritme

De bits in w_1 worden door de functie G bewerkt. Deze functie vormt een Feistel-netwerk van vier ronden met als transformatiefunctie een vaste byte-substitutie F . In elk van de vier ronden wordt het linker- dan wel het rechterbyte van w_1 gecombineerd met een byte van de sleutel en door F gesubstitueerd. De 10 bytes van de 80-bits sleutel worden achtereenvolgens gebruikt, zodat telkens na vijf ronden een volledige sleutelcyclus is voltooid. Het Feistel-netwerk is afgebeeld rechts in [figuur 8.8](#).

In stap A vervangt $G(w_1)$ de inhoud van w_2 , w_2 schuift door naar w_3 en die weer naar w_4 . De inhoud van w_1 wordt ververst met som $G(w_1) \oplus w_4 \oplus \text{teller}$. In stap B is het vervangingsschema anders. Hier gaat $G(w_1)$ naar w_2 , krijgt w_3 de som $w_1 \oplus w_2 \oplus \text{teller}$, gaat w_3 naar w_4 en w_4 naar w_1 . In formule luiden deze operaties

Stap – A

$$w_1^{k+1} = G^k(w_1^k) \oplus w_4^k \oplus \text{teller}^k$$

$$w_2^{k+1} = G^k(w_1^k)$$

$$w_3^{k+1} = w_2^k$$

$$w_4^{k+1} = w_3^k$$

Stap – B

$$w_1^{k+1} = w_4^k$$

$$w_2^{k+1} = G^k(w_1^k) \quad (8.49)$$

$$w_3^{k+1} = w_1^k \oplus w_2^k \oplus \text{teller}^k$$

$$w_4^{k+1} = w_3^k$$

Ontcijfering kan op dezelfde manier worden beschreven, maar blijkt een rapport van Eli Biham e.a. uit 1998 kan dit ook door het omdraaien van de loop van de teller en een herschikking van de bytes van sleutel K , klaartekst M en cijfertekst C ^[36].

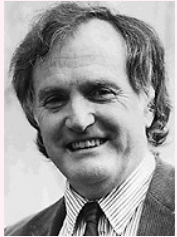
$$\begin{array}{lcl} K & \rightarrow & k_7 \quad k_6 \quad k_5 \quad k_4 \quad k_3 \quad k_2 \quad k_1 \quad k_0 \quad k_9 \quad k_8 \\ M & \rightarrow & m_3 \quad m_2 \quad m_1 \quad m_0 \quad m_7 \quad m_6 \quad m_5 \quad m_4 \\ C & \rightarrow & c_3 \quad c_2 \quad c_1 \quad c_0 \quad c_7 \quad c_6 \quad c_5 \quad c_4 \end{array} \quad (8.50)$$

In datzelfde rapport wordt een alternatieve beschrijving van Skipjack gegeven, waarin de lus wordt uitgerold en niet de inhoud van de registers wordt doorgeschoven, maar waarin de operaties G en $\oplus$ op wisselende registers worden uitgevoerd.

Een cryptanalyse van Skipjack is gepubliceerd in 1998 en geeft het volgende resultaat met behulp van een variant op differentiële cryptanalyse^[37]. In de oorspronkelijke differentiële cryptanalyse worden goede van foute sleutels onderscheiden door in paren cijfertekst te kijken naar het optreden van verwachte verschillen. Hoe meer daarvan gevonden worden, hoe waarschijnlijker de veronderstelde sleutel. In de hier gebezigde variant wordt het tegengestelde gedaan: van bepaalde verschillen wordt verwacht dat ze er juist *niet* kunnen zijn (impossible differentials). Dit verschijnsel kan worden gebruikt als een zeef, doordat sleutels die zo'n verschil wél geven niet de juiste kunnen zijn.

Deze analyse drijft verder op de volgende waarneming: een paar dat ronde 4 uitkomt met verschil $(w_1, w_2, w_3, w_4) = (0, w_2, 0, 0)$, $w_2 \neq 0$ kan nimmer ronde 28 verlaten met $(w_1, w_2, w_3, w_4) = (w_1, 0, 0, 0)$,

$w_1 \neq 0$. Dit is een karakteristiek over 24 ronden, die kan worden gebruikt om mogelijke sleutels uit te sluiten*. Om beter uit te komen dan het afzoeken van 2^{80} sleutels lukt het niet om alle 32 ronden in de analyse te betrekken. Bij 31 ronden kost het beste resultaat nog altijd 2^{34} gekozen klaartekst blokken en 2^{78} vercijferoperaties; slechts een marginaal verschil met uitputtend zoeken. Hieruit blijkt – evenals bij de DES het geval was – dat Skipjack qua veiligheidsmarge krap bemeten is, want de parameters zijn maar juist voldoende om differentiële cryptanalyse niet beter te maken dan uitputtend zoeken.



Massey

8.5 IDEA

Als een alternatief voor de Data Encryption Standard werd de Proposed Encryption Standard (PES) ontwikkeld door Lai en Massey, medewerkers van de ETH in Zürich. Na de bekendmaking van de techniek van differentiële cryptanalyse werd PES versterkt tot IPES (Improved PES) en vervolgens omgedoopt tot *International Data Encryption Algorithm* (IDEA)^[38]. IDEA is gepatenteerd en mag daarom niet vrijelijk worden toegepast in commerciële programmatuur, maar maakt wel onderdeel uit van de in het public domain verkrijgbare Pretty Good Privacy (PGP) software.

IDEA is een blokgeheimsschrift met 64 bits in het blok en 128 bits in de sleutel. Er zijn 8 ronden gevolgd door een afsluitende operatie; men spreekt daarom wel van 8,5 ronden in IDEA. Tegen de achtergrond van het feit dat ten tijde van de ontwikkeling van IDEA de 16-bit processoren domineerden, is het niet verwonderlijk dat op 16 bits eenheden wordt geopereerd. IDEA kan efficiënt worden geïmplementeerd.

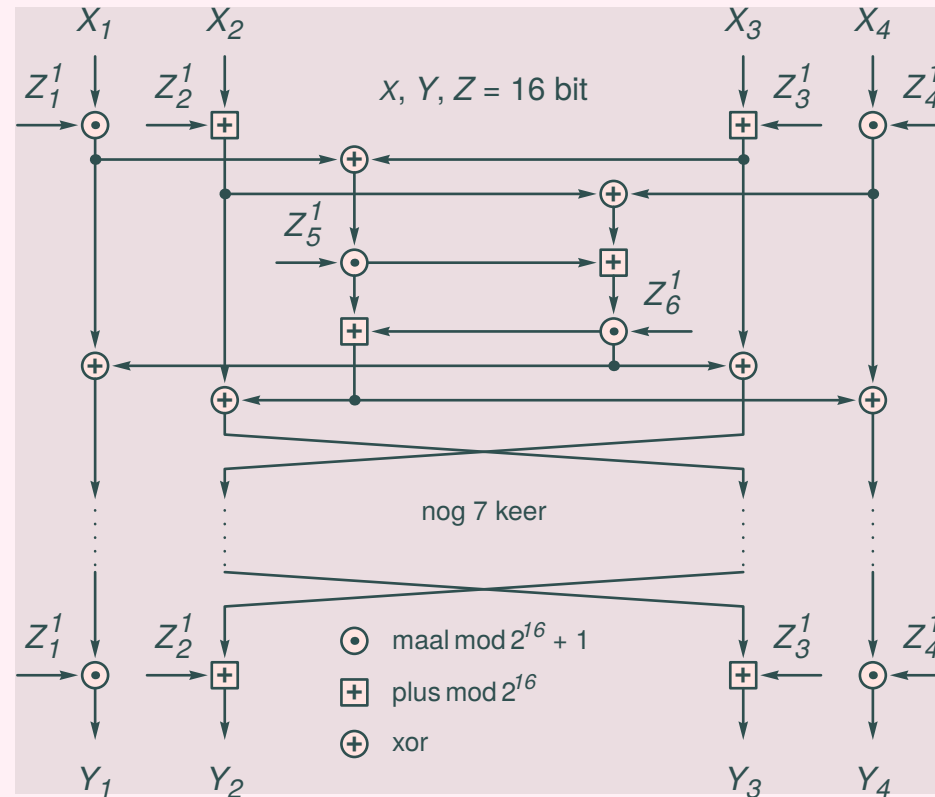
In tegenstelling tot de DES is IDEA niet gebaseerd op een substitutie-permutatie netwerk, maar op een combinatie van een drietal algebraïsche operaties, te weten

$x \oplus y$: exclusive or

* Als bijzonderheid wordt nog genoteerd dat de sleutelcyclus van 5 ronden niet moet overeenkomen met A en B -cycli van eveneens 5 ronden. In dat geval zou een *onmogelijk verschil* over zelfs 27 ronden beschikbaar zijn geweest en daarmee een effectievere cryptanalyse.

$x \boxplus y$: optelling modulo 2^{16}

$x \odot y$: vermenigvuldiging modulo $2^{16} + 1$ met $0 \equiv -1 \pmod{2^{16} + 1}$



Figuur 8.9 IDEA geheimschrift

Het rondeschema is afgebeeld in [figuur 8.9](#). De invoer wordt verdeeld in vier parten van 16 bits en door $\boxplus$ en $\odot$ met vier 16-bits subsleutels gecombineerd. De rondetransformatie wordt vervolgd met een mengoperatie, waarbij twee subsleutels worden gebruikt en waarin de zojuist genoemde operatoren eveneens een prominente rol spelen. Aan het eind van de achtste ronde volgt een afsluitende operatie met

nog eens vier subsleutels. Per ronde consumeren de zes subsleutels 96 bits van de 128-bits sleutel; na elke acht subsleutels wordt de sleutel over 25 bitposities naar links geroteerd. In totaal zijn 52 subsleutels nodig. Voor het ontcijferen wordt dezelfde algoritme gebruikt, maar moet als gevolg van de toegepaste aritmetische operaties meer werk worden verzet voor het genereren van de subsleutels*.

IDEA geldt nog steeds als een sterk geheimschrift. De beste aanvallen zijn niet verder gekomen dan 4,5 ronden^[39]. Wel heeft IDEA last van groepen zwakke sleutels. Joan Daemen toonde in 1993 aan dat van twee groepen van respectievelijk 2^{23} en 2^{51} sleutels vrij eenvoudig kan worden vastgesteld of de sleutel tot de groep behoort en dan kan de sleutel relatief gemakkelijk worden bepaald. Op zich is het aantal zwakke sleutels gering te achten op het totaal van 2^{128} sleutels.

De zwakte is gerelateerd aan de $\odot$ -operatie met een factor 1 of -1. Overigens blijkt het door middel van een eenvoudige ingreep mogelijk om deze zwakte te elimineren: modificeer de subsleutels met $K_i \oplus 0DAE_{16}$ ^[40]. Recent is aangetoond dat een grotere groep sleutels slecht bestand is tegen een combinatie van differentiële en lineaire cryptoanalyse; hierbij gaat het om 2^{63} sleutels. Als vastgesteld kan worden dat een sleutel zwak is, levert dit 72 sleutelbits op.^[41]

8.6 RC5

De RC5-algoritme is een aardig voorbeeld van een geheel andere aanpak dan met de Feistel-algoritme. Het is klein, eenvoudig, efficiënt, elegant en flexibel. De bedenker van deze algoritme is Ronald Rivest van RSA Laboratories. Rivest is de R uit RSA^[42]. Naar verluid staat RC voor Rivest Cipher of ook wel Ron's Code.

De flexibiliteit stamt van de variatie die mogelijk is in bloklengte, sleutellengte en aantal ronden. De bloklengte kan worden gekozen overeenkomstig de meest gunstige waarde voor de betreffende processor.



Rivest

* Multiplicatieve inverses met behulp van de uitgebreide Euclides algoritme.

Omdat RC5 het te vercijferen blok in twee helften splitst, wil dat voor 16, 32 en 64-bit processoren zeggen een bloklengte van respectievelijk 32, 64 of 128 bits. De lengte van de sleutel wordt gegeven in bytes en kan naar behoefte variëren van 0...255 bytes. Hoe meer ronden hoe veiliger is de gedachte, ook hier een keuze uit de range 0...255.

RC5 is dus in hoge mate geparametriseerd. Om de gebezigde variant aan te duiden noteert men RC5- $w/r/b$ voor een vercijfering met een bloklengte van $2w$ bits, aantal ronden r en sleutellengte b bytes. Rivest suggereerde aanvankelijk RC5-32/12/16. Later is het aanbevolen aantal ronden tot 16 verhoogd. Een overzicht van de cryptanalyse van RC5 tot 1998 kan worden gevonden in een rapport van RSA Laboratories^[43]. Anno 2000 werd de veiligheid van RC5 nog voldoende geacht, als tenminste het aantal ronden groot genoeg genomen wordt^[44].

RC5 maakt gebruik van de volgende operaties.

1. exclusive or : $x \oplus y$
2. optellen en aftrekken modulo 2^{2w} : $\boxplus xy$ en $\boxminus xy$
3. gegevensafhankelijke bitrotatie : $x \lll y$ en $x \ggg y$

De niet-lineariteit steunt geheel op de gegevensafhankelijke en dus wisselende rotaties, de overige operatoren zijn lineair. Voor de rotatie $x \lll y$ zijn slechts $\log_2 w$ rechterbits van y nodig (5 bits voor $w = 32$).

De sleutel $K[0...b-1]$ wordt geëxpandeerd tot een rij van $2r + 2$ intermediaire sleutels $S[0,...,2r+1]$ elk van w bits. In deze expansie wordt gebruik gemaakt van twee vaste constanten, afgeleid uit $e = 2,718...$ (de basis van de natuurlijke logaritmen) en $\phi = 1,618...$ (de gulden snede). De precieze keuze van deze constanten is op zich niet cruciaal, andere constanten zijn ook bruikbaar. Door te kiezen voor zulke vaste wiskundige constanten kan de schijn worden vermeden dat de gekozen waarde een verborgen achterdeur bevat.

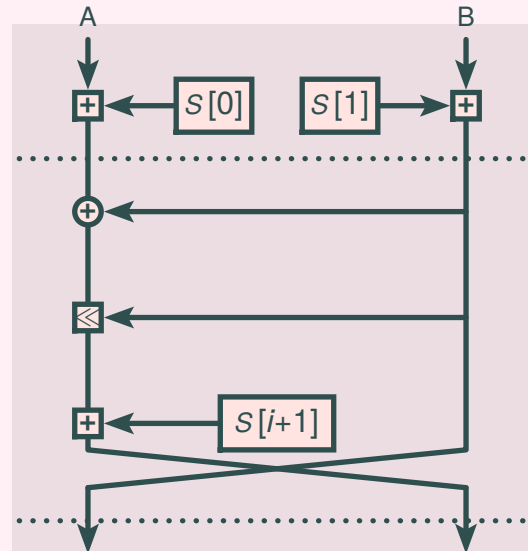
Vercijferen in RC5 geschiedt als volgt:

- $A = A \boxplus S[0]$
- $B = B \boxplus S[1]$
- for $i = 1$ to r do
- $A = ((A \oplus B) \lll B) \boxplus S[2i]$
- $B = ((B \oplus A) \lll A) \boxplus S[2i+1]$

Voor het ontcijferen worden de operaties in omgekeerde volgorde uitgevoerd en de subsleutels in omgekeerde volgorde gebruikt. De optelling $\boxplus$ en linksrotatie $\lll$ worden vervangen door respectievelijk aftrekken $\boxminus$ en naar rechts roteren $\ggg$. In een alternatieve representatie van RC5 wordt de tweede operatie in de lus weggelaten, de eerste vormt dan een *halve ronde*. Wanneer men na deze halve ronde A en B verwisselt en het aantal ronden verdubbelt, krijgt men weer hetzelfde resultaat. In [figuur 8.10](#) is het schema van de bewerkingen in zo'n halve ronde afgebeeld.

RSA Laboratories (een onderdeel van RSA Security Inc.) heeft, evenals voor andere geheimschriften, een prijs uitgelooft voor het breken van een aantal RC5 sleutels van verschillende lengte. In september 2002 brak een team van vrijwilligers, verenigd onder de naam `distributed.net`, een RC5 64 bits sleutel en sleepte daarmee de uitgelooft beloning van \$10.000 in de wacht. Om dit resultaat te bereiken hebben 331.252 vrijwillige deelnemers gedurende ruim vier jaar de ongebruikte tijd van hun PC's besteed aan het uitputtend zoeken naar deze sleutel*. De volgende uitdaging is een 72 bits sleutel. Als het in hetzelfde tempo gaat dan duurt het breken naar verwachting minstens een paar eeuwen.

* De willekeurig gekozen sleutel was 63DE 7DC1 54F4 D039 en de klaartekst luidde "The unknown message is: some things are better left unread." Bron: internet.



Figuur 8.10 RC5 halve ronde

8.7 Andere geheimschriften

Na het verschijnen van de Data Encryption Standard zijn er nog vele andere dan de hier behandelde (blok)geheimschriften bedacht; teveel om op te noemen. Volstaan wordt met het noemen van enkele namen. Meer voorbeelden kunnen worden gevonden in het boek *Applied Cryptography* van Bruce Schneier^[45].

GOST stamt uit de voormalige Sovjet Unie. Dit algoritme is een Feistel-geheimschrift met 32 rondes en vercijfert 64-bits blokken met een 256-bits sleutel. In de rondetransformatie wordt een 32-bits helft gecombineerd met een subsleutel en per vier bits door een *S-doos* gesubstitueerd, gevolgd door een linksrotatie van 11 bits. De samenstelling van de *S*-dozen is in principe geheim en kan per toepassing worden gevarieerd. Aangezien de weerstand tegen differentiële en lineaire cryptoanalyse sterk afhankelijk is van de precieze substituties, is wel eens gespeculeerd dat de uitgifte ervan mede werd bepaald door

de behoefte om de cryptogrammen van bepaalde organisaties zonodig te kunnen breken^[46].

Blowfish is een algoritme bedacht door Bruce Schneier. Het is niet gepatenteerd en C-code is verkrijgbaar via het Internet. Het werd ontworpen met de volgende uitgangspunten

1. snelle verwerking op 32-bit processoren
2. compacte code, minder dan 5K geheugen
3. slechts eenvoudige operatoren optellen, exclusive or en opzoeken in tabel
4. variabele sleutellengte

Blowfish heeft een relatief trage opstarttijd door het opzetten van de subsleutels, waardoor het minder geschikt is bij snelle sleutelwisselingen. Tegelijkertijd echter biedt dit weer meer bescherming tegen sleutelzoekprogramma's^[47].

FEAL was bedoeld als rechtstreekse concurrent van de DES met een openbaar algoritme en bovendien een sterkere rondetransformatie^[48]. Het heeft niet zo mogen zijn. Aanvankelijk telde FEAL vier ronden: FEAL4. Na te zijn gebroken door onze landgenoot Bert den Boer en anderen, werd in FEAL8 het aantal ronden verdubbeld tot acht^[49]. Daarna kwamen er steeds weer (succesvolle) aanvallen. Ondanks verdere uitbreiding van het aantal ronden en andere aanpassingen geldt het niet als een sterk geheimschrift. FEAL bleek niet slechts buitengewoon gevoelig voor differentiële cryptoanalyse, ook lineaire cryptoanalyse breekt FEAL4 al met 5 bekende klaarteksten. Sindsdien lijkt het wel of iedere nieuwe soort cryptoanalyse eerst op het arme FEAL wordt uitgeprobeerd.

8.8 Advanced Encryption Standaard

Met het verstrijken van de jaren na de verheffing van de Data Encryption Standard tot algemene standaard (januari 1977), heeft het gaandeweg aan veiligheid ingeboet. Zoals het een Federal Information Processing Standard (FIPS) betaamt, is de DES steeds om de vijf jaar geëvalueerd met het oog op verlenging van de

geldigheid voor een volgende periode van vijf jaar. Laatstelijk is dat gebeurd op 25 oktober 1999^[50]. Toen was al een paar jaar duidelijk dat de levensduur van de DES snel teneinde liep, ondanks de kunstgreep middels TripleDES. De succesvolle bouw van de DES-sleutelzoekmachine Deep Crack in 1998 bezegelde dit lot slechts.

Ook nu weer koos het National Institute of Standards and Technology NIST voor een publieke aanbesteding. En ditmaal met meer succes dan in 1973. Want leverde de vorige oproep slechts één enkele bruikbare inzending van IBM op, thans strijden ontwerpers uit alle windstreken om de eer het standaardgeheim-schrift voor de komende decennia te mogen leveren. Op 2 januari 1997 start NIST de race met een vooraankondiging die op 12 september van dat jaar wordt gevolgd door een officiële oproep voor kandidaten voor de *Advanced Encryption Standard* (AES), zoals de beoogde opvolger van de DES wordt genoemd. De vorderingen kunnen vanaf dat moment worden gevolgd op een speciale Advanced Encryption Standard-internetpagina op de NIST-site*. De kandidaat-algoritmen zullen uiteindelijk door NIST worden beoordeeld en geselecteerd, maar de internationale cryptogemeenschap wordt aangemoedigd daaraan mee te doen door onderzoek en cryptanalyse. Een programma van eisen vermeldt waaraan inzendingen moeten voldoen, zoals

1. ongeclassificeerd, publiekelijk bekend algoritme
2. overal ter wereld vrij van royalties beschikbaar
3. symmetrische algoritme, blokgrootte tenminste 128 bits
4. meerdere sleutellengten van 128, 192 en 256 bits mogelijk
5. efficiënt te implementeren op allerlei platforms (smartcards, werkstations)
6. documentatie, testvectoren, voorbeeld implementaties (C, Java)

* URL <http://www.nist.gov/aes/>

CAST-256	Entrust Technologies, Inc.	Canada
CRYPTON	Future Systems, Inc.	Korea
DEAL	Outerbridge, Knudsen	Canada
DFC	Centre National Recherche Scientifique	Frankrijk
E2	Nippon Telegraph and Telephone Corp.	Japan
FROG	TecApro Internacional S.A.	Costa Rica
HPC	Schroeppel	USA
LOKI97	Brown, Pieprzyk, Seberry	Australië
MAGENTA	Deutsche Telekom AG	Duitsland
▶ MARS	IBM	USA
▶ RC6	RSA Laboratories	USA
▶ RIJNDAEL	Daemen, Rijmen	België
SAFER+	Cylink Corp.	USA
▶ SERPENT	Anderson, Biham, Knudsen	Internationaal
▶ TWOFISH	Schneier, Kelsey, e.a.	USA

Tabel 8.8 AES-kandidaten uit de eerste ronde ▶ finalist

De eerste fase van het selectieproces is in augustus 1998 tijdens een speciale AES-conferentie afgesloten met de bekendmaking van de vijftien tot de volgende ronde toegelaten kandidaten. Alle geaccepteerde kandidaten kregen de gelegenheid om hun inzending te presenteren^[51]. Om een indruk te krijgen van de wereldwijde belangstelling voor de AES-competitie worden ze in **tabel 8.8** allemaal genoemd.

In een tweede conferentie in maart 1999 worden commentaren vanuit de gehele wereld bediscussieerd en op 15 april 1999 sluit de eerste fase. NIST selecteert dan vijf finalisten voor nader onderzoek in de tweede ronde. Deze finalisten zijn MARS, RC6, Rijndael, Serpent en Twofish. Een vervolgconferentie wordt gehouden op 13–14 april 2000. De finalisten waren uitgenodigd om met de deelnemers van de conferentie van gedachten te wisselen.

Op 15 mei 2000 eindigt de tweede ronde en gaat NIST zich bezinnen op de winnaar. Hoewel aanvankelijk niet uitgesloten werd dat meer dan een algoritme als toekomstige standaard kon worden verkozen, wordt uiteindelijk Rijndael op 2 oktober 2000 tot winnaar uitgeroepen.

8.8.1 Evaluatie door NIST

Het rapport waarin NIST z'n keuze heeft verantwoord is beschikbaar op het internet^[52]. Bovenaan in de evaluatie staat het criterium *veiligheid*. Dit omvat zaken als de weerstand tegen cryptanalyse, degelijkheid van de wiskundige onderbouwing, goede random eigenschappen van de uitvoer en de veiligheid in vergelijking tot de andere kandidaten voor de Advanced Encryption Standard. Daarna komt het efficiëncy criterium in de zin van snelheid en geheugenvereisten. En op de derde plaats aspecten als flexibiliteit met betrekking tot hardware en software implementatie: variatie in blok- en sleutelgrootte, variatie in platform waarop de algoritme moet worden geïmplementeerd, toepassing in stroomvercijfering, als hash-algoritme en dergelijke.

Veiligheid

Ten aanzien van de veiligheid kon worden vastgesteld dat tegen geen van de finalisten een afdoende aanval of het bestaan van diskwalificerende eigenschappen is gerapporteerd. Slechts varianten van bekende aanvalsmethoden tegen in aantal ronden gereduceerde of anderszins vereenvoudigde versies van de algoritmen zijn verschenen. Hoeveel waarde moet worden gehecht aan het success van een aanval op een gereduceerde variant? In het algemeen kan men slechts met zekerheid stellen dat "aanvallen mettertijd nooit minder krachtig worden". Veel belang bij de evaluatie is in dit opzicht gehecht aan de afkomst van de inzendingen, zoals die van RC6 uit RC5 of Rijndael uit diens voorloper Square. Alsook aan de mate waarin door analyse van de algoritme iets over de sterkte kan worden geconcludeerd, zoals bij Twofish en MARS moeilijk is omdat die zo ingewikkeld zijn.

Geconcludeerd werd dat MARS de grootste veiligheidsmarge lijkt te vertonen, maar dat de ingewikkelde constructie een goede analyse tot nu toe in de weg gestaan heeft. RC6 wordt de geringste veiligheidsmarge toegedacht, maar is aan de andere kant eenvoudig genoeg voor goede analyse en bovendien gebaseerd op het uitgebreid aan de tand gevoelde RC5. Rijndael acht men op dit punt voldoende bemeten, al bestaat er enige zorg dat de sterk wiskundige grondslag de weg voor aanvallen zal openen. Maar aan de andere kant had men die dan al mogen verwachten, juist op grond van diezelfde eenvoud en structuur. Serpent heeft een hoge veiligheidsmarge, is eenvoudig van structuur en scoort derhalve op dit punt een voldoende. Twofish lijkt op dit punt goed, maar er zijn twijfels aangaande de sleutelafhankelijke *S*-dozen en de kans dat er nog eens een verdeel-en-heers aanval op wordt gevonden; de ingewikkeldheid bemoeilijkt de analyse in de hiervoor toegemeten tijd.

Snelheid

Een vergelijking van de snelheid waarmee de finalisten in verschillende implementaties hun gegevens verwerken levert de volgende conclusies op. MARS vertoont een gemiddelde performance in vercijfering, ontcijfering en het opzetten van de sleutel. RC6 scoort op dat punt gemiddeld tot hoog. Rijndael heeft een consistent hoge performance, al vermindert deze bij het kiezen van sleutels langer dan 128 bits. Serpent presteert consequent laag en gedraagt zich op de verschillende onderzochte platforms nogal wisselend ten aanzien van de sleutelinitialisatie. Twofish tenslotte wisselt in gedrag per platform in de snelheid van de vercijferoperaties en presteert laag bij het opzetten van de sleutels. Eveneens is onderzocht in hoeverre de prestaties variëren bij de verschillende toepassingsmethodieken als Electronic Code Book (ECB-modus), Cipher Block Chaining (CBC-modus) en zo meer.

Hardware

De aandacht is ook uitgegaan naar het gedrag op smartcards, uiteraard omdat men daar nu juist veel

toepassingen verwacht en omdat smartcards typisch beperkt zijn in processorkracht en geheugenomvang. Twee case studies zijn uitgevoerd en de resultaten zijn in de evaluatie betrokken. Vooral MARS ondervond in zo'n omgeving last van de gekozen rondestructuur. RC6 presteert weliswaar goed, maar de subsleutels kunnen niet on-the-fly worden gegenereerd en dat kost geheugen, terwijl ook het opzetten van de sleutel relatief traag bleek. Rijndael was hier het meest efficiënt, subsleutels kunnen gedurende de vercijfering met weinig geheugenoverhead worden gegenereerd. Nadelig is het feit dat vercijfering en ontcijfering enigszins van elkaar verschillen, waardoor ontcijfering langzamer wordt en extra programmeergeheugen kost. Aan de andere kant heeft een smartcard vaak maar een van beide operaties nodig. Overigens hangt bij dit soort toepassingen veel af van de vraag of men naar snelheid of naar geheugengebruik wil optimaliseren en of de smartcard al dan niet over een coprocessor beschikt.

Een ander platform van interesse is dat van de hoog presterende Field Programmable Gate Array (FPGA) en het Application Specific Integrated Circuit (ASIC). Een aantal verschillende case studies is uitgevoerd, vier voor FPGA's en twee voor ASIC's. Naast de te behalen verwerkingssnelheid is hier een van de belangrijkste parameters de benodigde oppervlakte op de chip op grond van het aantal gates. Ook hier scoort Rijndael verrassend goed met Serpent een geduchte concurrent.

Speciale aanvallen

Eveneens is gekeken naar de mogelijkheden voor aanvallen gericht op de implementatie. Aan de effecten van tijdsduur en stroomverbruik van de operaties is al eerder in dit hoofdstuk (bij het beschrijven van de aanvallen op de Data Encryption Standard aandacht geschonken. Rijndael en Serpent lijken hier in het voordeel omdat de gebruikte logische operaties, substitutie middels een tabel en vaste rotaties/verschuivingen het gemakkelijkst tegen zulke aanvallen te verdedigen zijn. Vooral MARS en RC6 zijn hier door hun vermenigvuldiging/deling/kwadratering en variabele rotatie in het nadeel. Het blijkt lastig hi-

erover algemene conclusies te trekken, anders dan dat veel afhangt van de afweermaatregelen die in de hardware worden ingebouwd.

Overige aspecten

Rijndael en Serpent hebben afwijkende algoritmen voor vercijfering en ontcijfering, hetgeen de efficiëntie wat beïnvloedt, zowel voor wat betreft de verwerkingssnelheid onder alle omstandigheden als voor de geheugenvereisten.

De snelheid en flexibiliteit waarmee van sleutel gewisseld kan worden is van belang. In sommige toepassingen wordt vaak van sleutel gewisseld, in andere gevallen, zoals bulkencryptie, gaat een sleutel lang mee. Rijndael, Serpent en Twofish maken het mogelijk om subsleutels on-the-fly te produceren, RC6 kan dit alleen voor vercijfering. MARS hinkt als het ware op twee gedachten, omdat per keer 10 van 40 benodigde subsleutels kunnen worden gemaakt. Rijndael is hier in het nadeel, aangezien voor het berekenen van de eerste ontcijfersleutel de gehele reeks subsleutels allereerst in voorwaartse richting moet zijn uitgevoerd en pas daarna met de laatste subsleutel de ontcijfering kan worden ingegaan. Als de berekening van de subsleutels recursief en inverteerbaar is, kan men al naar gelang vercijfering of ontcijfering gewenst is, de eerste of de laatste subsleutel bewaren en die voor elke vercijfer/ontcijferoperatie opnieuw benutten. Bij het vaak wisselen van sleutel is het voordelig als slechts de eerste of de laatste subsleutel hoeft te worden gewisseld in plaats van een swap van het gehele ensemble subsleutels.

De flexibiliteit in blok grootte en sleutellengte blijkt het grootst voor RC6 en Rijndael. MARS, Serpent en Twofish bieden weliswaar andere sleutellengtes dan standaard 128, 192 en 256 bits, maar zijn ten opzichte van RC6 en Rijndael weer minder wendbaar in de blok grootte en het aantal ronden.

Toekomstige processoren zullen naar verwachting meer gebruik gaan maken van de mogelijkheden om instructies parallel uit te voeren. Het blijkt dat Rijndael potentieel het meest kan profiteren van zulk parallelisme.

Eindconclusie

Aan het eind van deze opsomming van overwegingen en meer of minder subtiele vergelijkingen tussen de finalisten, volgt hier de slotconclusie uit het NIST-rapport waarin de keuze voor Rijndael wordt gemotiveerd*.

“Rijndael appears to be consistently a very good performer in both hardware and software across a wide range of computing environments regardless of its use in feedback or non-feedback modes. Its key setup time is excellent, and its key agility is good. Rijndael’s very low memory requirements make it very well suited for restricted-space environments, in which it also demonstrates excellent performance. Rijndael’s operations are among the easiest to defend against power and timing attacks. Additionally, it appears that some defence can be provided against such attacks without significantly impacting Rijndael’s performance. Rijndael is designed with some flexibility in terms of block and key sizes, and the algorithm can accommodate alterations in the number of rounds, although these features would require further study and are not being considered at this time. Finally, Rijndael’s internal round structure appears to have good potential to benefit from instruction-level parallelism.

There are many unknowns regarding future computing platforms and the wide range of environments in which the AES will be implemented. However, when considered together, Rijndael’s combination of security, performance, efficiency, implementability, and flexibility make it an appropriate selection for the AES for use in the technology of today and in the future.”

Wie mocht menen dat het aantal nieuwe geheimschriften met de finale van de AES-competitie wel zou afnemen, komt bedrogen uit. Nieuwe geheimschriften verschijnen bij wijze van spreken dagelijks. Een

* Op 31 december 2000 kregen Vincent Rijmen en Joan Daemen van hun landgenoten de zojuist ingestelde Castar-prijs. Deze Belgische prijs gaat naar iemand die het afgelopen jaar op positieve manier in het nieuws is gekomen en wordt toegekend door Radio 1, CANVAS (het 2e Belgische TV-net) en de krant De Standaard.

greep uit de tombola van fantasievolle namen en acroniemen*: ARIA, FOX, KLEIN, LEA, MICKEY, PICARO, PRESENT, PRINCE, SNOW, enzovoorts; de kroon spant voor het moment wellicht het systeem genaamd Zorro. Werk genoeg aan de winkel dus voor wie zich in de moderne cryptografie en cryptanalyse heeft bekwaamd.



Coppersmith

8.8.2 Overige finalisten

MARS

MARS is het product van een IBM onderzoeksteam met daarin Don Coppersmith, een van de ontwerpers van de Data Encryption Standard. In dit algoritme worden de data vercijferd in blokken van 128 bits, opgedeeld in vier woorden van 32 bits en met een sleutel van 128–448 bits. Bij de variabele sleutellengte tekenen de ontwerpers aan dat naar hun verwachting de extra bits boven 256 relatief minder tot de sterkte zullen bijdragen dan hun aantal suggereert. Alle bewerkingen worden op 32 bits woorden uitgevoerd. Deze bewerkingen zijn optellen, aftrekken en vermenigvuldigen (alledrie modulo 2^{32}), exclusive or, vaste en variabele cyclische rotaties, alsmede substitutie met een vaste S-doos van 512 woorden die kan worden gebruikt voor een 9-bits of voor twee 8-bits substituties.

MARS begint met een sleutelexpansie in een array van 40 woorden. De encryptie start met een whitening-stap. De kern van de transformatie bestaat uit een Feistel-netwerk van 14 ronden. In elke ronde wordt een van de vier 32-bits woorden van het te vercijferen blok gebruikt om de drie andere te modificeren. Deze kern wordt voorafgegaan en gevolgd door 8 ronden zonder sleutel, bedoeld om de bits goed dooreen te mengen. Tenslotte volgt een afsluitende whitening-stap.

* URL <http://eprint.iacr.org/>

RC6

Dit geheimschrift, ingediend door RSA Laboratories, is een door Rivest verder ontwikkelde versie van RC5 (eerder beschreven op [pagina 374](#)). Net als RC5 is RC6 een geparametriseerd geheimschrift: RC6- $w/r/b$.

Gekozen is voor een 20 ronden Feistel-netwerk op vier 32-bit woorden. De sleutel kan 16, 24 of 32 bytes tellen. De sleutelexpansie is nagenoeg dezelfde als in RC5. Echter aan de vercijfering is in de rondetransformatie een complicatie toegevoegd. Het belangrijkste verschil met de rondetransformatie van RC5 is dat het aantal te roteren bits niet slechts afhankelijk is gemaakt van de minst significante bits – was $\lll A, B$ in RC5 – maar dat dankzij de transformatie $F(x) = x(2x + 1)$ op B en D alle bits van het betreffend woord aan de bepaling ervan meedoen.

Vercijferen in RC6 geschiedt als volgt

- $B = B + S[0], D = D + S[1]$ (whitening stap)
- for $i = 1$ to r do
 - $t = B(2B + 1) \lll \log w$
 - $u = D(2D + 1) \lll \log w$
 - $A = ((A \oplus t) \lll u) + S[2i]$
 - $C = ((C \oplus u) \lll t) + S[2i + 1]$
 - $A, B, C, D = B, C, D, A$
- $A = A + S[2r + 2], C = C + S[2r + 3]$ (whitening stap)



Rivest



Anderson



Biham



Knudsen

SERPENT

Deze inzending is afkomstig van Ross Anderson (Cambridge, Engeland), Eli Biham (Technion, Israël) en Lars Knudsen (San Diego, Californië), drie bekende namen in de cryptowereld.

Serpent is een substitutie-permutatie netwerk (SP-netwerk in het jargon) met 32 rondes. Het grote aantal rondes is bedoeld om het geheimschrift een hoge veiligheidsmarge te bezorgen*. Aan het begin en eind van de vercijfering wordt net als bij de DES een initiële permutatie (IP respectievelijk IP^{-1}) uitgevoerd. In elke ronde worden de data met de sleutel gecombineerd en door een S -doos gesubstitueerd. Er zijn in totaal acht verschillende 4x4-bit S -dozen, die elkaar per ronde afwisselen. De S -dozen zijn vast en door de ontwerpers met behulp van de DES gecreëerd. Het substitutieresultaat ondergaat tot slot een lineaire transformatie waarin een reeks van exclusive or's en vaste rotaties en shifts wordt uitgevoerd.



Schneier

TWOFISH

Twofish is ingezonden door het bedrijf Counterpane Internet Security, Inc. van Bruce Schneier, die als schrijver bekend is door een omvangrijke compilatie van geheimschriften in zijn boek *Applied Cryptography*^[53]. Een uitvoerige beschrijving van Twofish kan worden gevonden in een door de ontwerpers geschreven boekwerk^[54].

Twofish bestaat uit een Feistel-netwerk van 16 rondes en onderscheidt door het feit dat de vier 8x8 S -dozen met behulp van de sleutel gemaakt worden; het zijn daarmee dus variabele substituties. Deze dynamische constructie van S -dozen draagt het gevaar in zich dat zo nu en dan zwakke exemplaren zullen meedoen. Omdat de S -dozen echter per sleutel verschillen, wordt het gevaar van de kant van differentiële en lineaire cryptoanalyse verwaarloosbaar geacht. Na de substitutie volgen onder meer een matrixvermenigvuldiging en een combinatie met de sleutel. Rotaties over 1 bit verhinderen opsplitsing naar de afmeting van de S -doos-substitutie.

* Met veiligheidsmarge wordt bedoeld het verschil tussen het aantal rondes nodig voor succesvolle cryptoanalyse en het werkelijk uitgevoerd aantal rondes. De gedachte is dat een flinke overdosis aan rondetransformaties de sterkte van het geheimschrift opvoert, iets dat tot op zekere hoogte ook het geval is.

8.8.3 Winnaar Rijndael

Rijndael is het geesteskind van twee Belgische onderzoekers, Vincent Rijmen en Joan Daemen. Zij hebben hierin voortgebouwd op hun Square geheimschrift^[55]. De naam Rijndael is een samentrekking van de namen van de bedenkers (voor mensen buiten het Nederlandstalig gebied overigens een crime om uit te spreken). Een uitvoerige beschrijving van Rijndael met de onderliggende uitgangspunten kan worden gevonden in het document dat behoort bij de inzending als kandidaat voor de Advanced Encryption Standard*. Gegeven het grote belang van dit algoritme, volgt een tamelijk gedetailleerde beschrijving.

Efficiënte implementatie op een reeks van processoren, van smartcards tot grote werkstations, is duidelijk een van de hoofduitgangspunten van het ontwerp geweest. De vercijfering is opgesplitst in enkele basis-transformaties die kunnen worden uitgevoerd middels een combinatie van substitutietabellen en exclusive or's. De efficiëncy wordt bereikt door zich te beperken tot het manipuleren van bytes, hetgeen vooral voor de kleinere processoren in bijvoorbeeld chipkaarten van belang is. De bytes van het te vercijferen blok worden in matrixvorm opgesteld, waardoor met mengoperaties in twee richtingen (horizontaal en verticaal) toch eenvoudig voor een goede diffusie gezorgd kan worden.

De flexibiliteit van Rijndael komt tot uiting in het feit dat blok- en sleutellengte onafhankelijk van elkaar mogen variëren; de waarden zijn 128, 192 en 256 bits (16, 24 en 32 bytes). De ontwerpers verwachten zelf dat alleen de 128-128 variant van praktisch belang zal blijken. Zij zien daarom de extra complicaties, die voor de langste sleutels zijn ingebouwd, nauwelijks als een nadeel[†]. Blok en sleutel worden in een reeks vercijferronden met elkaar gecombineerd. Het aantal ronden is afhankelijk van de combinatie blok- en sleutellengte.



Rijmen



Daemen

* *AES Proposal: Rijndael*. Te vinden via de AES pagina van de NIST-internetsite. Hier staan ook implementaties in de vorm van C en Java-code.

† Mededeling van Rijmen tijdens een lezing op 9 februari 2001 voor de EIDMA Werkgroep voor Cryptografie.

Sleutel- en bloklengte in Rijndael

- 10 ronden voor 128 bit blok en 128 bit sleutel
- 12 ronden voor 192 blok en/of sleutel, beide kleiner dan 256
- 14 ronden voor 256 bit blok en/of sleutel

In elke ronde wordt het gehele blok bewerkt, waardoor het aantal ronden vergelijkbaar is met het dubbele aantal in een Feistel-geheimschrift. Rijndael is geen Feistel-geheimschrift, waarin zoals bekend per ronde slechts een deel van het blok bewerkt wordt. Omdat het geen Feistel is, moet er expliciet voor gezorgd worden dat alle operaties inverteerbaar zijn, omdat anders ontcijferen niet mogelijk is. De algoritme is daarom niet volledig symmetrisch, waardoor vercijferen en ontcijferen op enkele punten van elkaar verschillen.

Rijndael-algoritme

Voorafgaand aan het rondeschema ondergaat het te vercijferen blok een whitening-stap. Vervolgens wordt het materiaal in elke ronde aan diverse transformaties onderworpen, door de ontwerpers lagen (layers) genoemd.

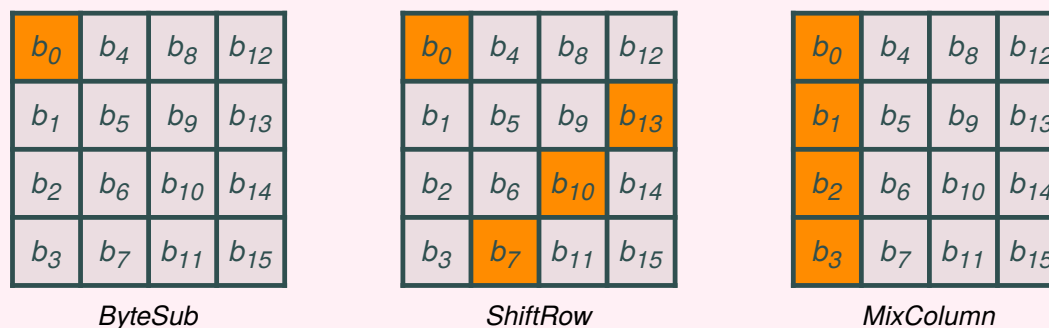
Operaties in Rijndael

- S-doos substitutie voor de niet-lineaire effecten (ByteSub)
- lineaire menglaag ter bevordering van diffusie (ShiftRow en MixColumn)
- exclusive or combinatie met subsleutels (AddRoundKey)

AddRoundKey - whitening		
stap	ronde	slotronde
1	ByteSub	ByteSub
2	ShiftRow	ShiftRow
3	MixColumn	AddRoundKey
4	AddRoundKey	

Tabel 8.9 Vercijferschema van Rijndael

Het hoofdschema voor de vercijfering staat in tabel 8.9. Het weglaten van MixColumn in de laatste ronde is bedoeld om vercijferen en ontcijferen beter met elkaar in overeenstemming te brengen en valt te vergelijken met het achterwege laten van de verwisseling in de laatste ronde van de DES. De deeloperaties ByteSub, ShiftRow en MixColumn zijn in figuur 8.11 afgebeeld. De werking van deze deeloperaties zal achtereenvolgens de revue passeren.



Figuur 8.11 Basisoperaties in Rijndael

Alle data in Rijndael, dat wil zeggen het te vercijferen blok, de sleutel en de tussenresultaten, worden gerepresenteerd door een matrix van 4 rijen en een van de bloklengte afhankelijk aantal kolommen. De elementen van deze matrix zijn bytes. Een 128-bit blok wordt dus behandeld als een matrix van $4 \times 4 =$

16 bytes, terwijl de 32 bytes van een 256-bits blok worden verdeeld over 8 kolommen. De bytes vullen de matrix kolomsgewijs, zodat de data kunnen worden opgevat als een aaneenschakeling van kolommen met een lengte van 4 bytes.

$$(b_0, b_1, b_2, b_3, b_4 \dots b_{n-1}) \rightarrow \begin{pmatrix} (b_{0,0} = b_0) & (b_{0,1} = b_4) & \dots \\ (b_{1,0} = b_1) & (b_{1,1} = b_5) & \dots \\ (b_{2,0} = b_2) & (b_{2,1} = b_6) & \dots \\ (b_{3,0} = b_3) & (b_{3,1} = b_7) & \dots \end{pmatrix} \quad (8.51)$$

Het rijnummer i geeft de index van het byte in de kolomvector en het kolomnummer j het volgnummer van de 4-byte vector. De betrekkingen tussen de indices in de beide respectievelijke representaties luiden dan

$$\begin{aligned} b_k &\rightarrow b_{i,j} & i &= k \bmod 4, j = \lfloor k/4 \rfloor \\ b_k &\leftarrow b_{i,j} & k &= i + 4 \times j \end{aligned} \quad (8.52)$$

Byte-substitutie

De ByteSub-operatie is een substitutie die zowel in het rondeschema als voor het genereren van subsleutels wordt gebruikt. Deze substitutie kan rechttoe rechtaan worden geïmplementeerd met een vaste substitutietabel van 256 bytes, zie [tabel 8.10](#). Over de constructie ervan valt echter heel wat meer te zeggen, omdat de ontwerpers hier hun wiskundige achtergrond hebben doen gelden. Voor rekenoperaties op bytes maakt Rijndael gebruik van het Galois Field $\text{GF}(2^8)$. Daartoe wordt elk bit in een byte opgevat als ware het de coëfficiënt van de overeenkomstige macht in een polynoom $f(x)$ met $x \in \{0,1\}$.

$$b_7b_6b_5b_4b_3b_2b_1b_0 \leftrightarrow b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0 \quad (8.53)$$

Optellen en vermenigvuldigen van bytes wordt bewerkstelligd door deze operaties uit te voeren met de overeenkomstige polynomen. Om bij vermenigvuldiging het resultaat niet in graad te doen uitstijgen

boven x^7 volgt een reductie* modulo een vaste polynoom $m(x)$. Voor $m(x)$ moet een irreducibele polynoom worden genomen en gekozen is

$$m(x) = x^8 + x^4 + x^3 + x + 1 \quad (8.54)$$

Aan deze keuze van $m(x)$ liggen geen bijzondere overwegingen ten grondslag. Voor het berekenen van de inverse van vermenigvuldiging kan gebruik gemaakt worden van de algoritme van Euclides†.

Twee voorbeelden die ook door Daemen en Rijmen zelf worden gebruikt. Optellen van $57_{16} = 01010111$ en $83_{16} = 10000011$ geeft

$$(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2$$

Hieruit volgt als antwoord $11010100 = D4_{16}$. Vermenigvuldiging van dezelfde getallen levert als resultaat $C1_{16} = 11000001$, zoals blijkt uit

$$(x^6 + x^4 + x^2 + x + 1)(x^7 + x + 1) = x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$$

Hetgeen modulo $m(x)$ wordt gereduceerd tot

$$x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \bmod m(x) = x^7 + x^6 + 1$$

Het voorwerk om de ByteSub operatie te kunnen definiëren is nu gedaan. De substitutie bestaat uit twee achtereenvolgende stappen g en f zodanig dat $S(x) = f(g(x))$. De eerste stap is het nemen van de multiplicatieve inverse in $GF(2^8)$ van het te substitueren byte. Het bijzondere geval 00 geldt als z'n eigen inverse. Op de bits van deze inverse wordt de affine transformatie $f(x)$ uitgevoerd, gedefinieerd door de matrixvermenigvuldiging en vectorsom $y = Mx + C$.

* Dit is vergelijkbaar met de reductie $p \bmod q$, waarin de modulus q dezelfde rol speelt als $m(x)$.

† In de uitgebreide variant van Euclides wordt de inverse van $f(x) \bmod m(x)$ gevonden middels de twee uit de reductie resulterende functies $a(x)$ en $b(x)$ met als eigenschap $a(x)f(x) + b(x)m(x) = 1$, waaruit dan weer volgt $f^{-1}(x) = a(x) \bmod m(x)$.

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \quad (8.55)$$

Elk byte van het blok wordt vervangen door het resultaat van deze bewerkingen. Het is handig om de berekening tevoren uit te voeren voor alle 256 bytewaarden en de uitkomsten in een tabel te bewaren. Er geldt dan eenvoudig $\text{ByteSub}(x) = y[x]$. Voor het ontcijferen heeft deze tabel slechts in de inverse tabel te worden omgezet. Ook aan efficiënte implementatie in hardware is gedacht. Want doordat $g(x)$ z'n eigen inverse is geldt

$$S^{-1}(x) = g^{-1}(f^{-1}(x)) = g(f^{-1}(x)) \quad (8.56)$$

Daarom behoeven slechts g , f en f^{-1} geïmplementeerd te worden, waarvan de beide laatste eenvoudige bitoperaties zijn.

Rij-verschuiving

De data worden binnen de rijen linksom cyclisch geroteerd over een geheel aantal bytes. De rotatie verschilt per rij en is bovendien afhankelijk van het aantal aanwezige kolommen; [tabel 8.11](#) geeft de waarden. De inverse van deze operatie is eenzelfde rotatie rechtsom ofwel het complementaire aantal linksom. [Vergelijking \(8.57\)](#) geeft de rotaties voor een 128-bits blok als voorbeeld

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
10	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
20	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
30	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
40	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
50	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
60	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
70	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
80	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
90	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A0	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B0	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C0	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D0	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E0	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F0	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Tabel 8.10 Numerieke voorwaartse ByteSub substitutietabel

<i>rij</i>	<i>4-kolom</i>	<i>6-kolom</i>	<i>8-kolom</i>
0	0	0	0
1	1	1	1
2	2	2	3
3	3	3	4

Tabel 8.11 Rijndael rotatieschema

$$\begin{pmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \end{pmatrix} \rightarrow \begin{pmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,1} & b_{1,2} & b_{1,3} & b_{1,0} \\ b_{2,2} & b_{2,3} & b_{2,0} & b_{2,1} \\ b_{3,3} & b_{3,0} & b_{3,1} & b_{3,2} \end{pmatrix} \quad (8.57)$$

Mengen van kolommen

Voor deze operatie worden de 4-bytes kolommen eveneens aangemerkt als polynomen. Zoals eerder de bits van een byte figureerden als de coëfficiënten van een polynoom, zo corresponderen nu de bytes b_i van de kolomvector met de coëfficiënten van de polynoom

$$b_3x^3 + b_2x^2 + b_1x + b_0 \quad (8.58)$$

De coëfficiënten – de bytes b_i – zijn weer op te vatten als elementen van $\text{GF}(2^8)$. Optellen geschiedt eenvoudig door optellen van de coëfficiënten. Vermenigvuldigen is ingewikkelder, want ook hier is een reductie modulo een polynoom nodig. Rijndael kiest als reductiepolynoom

$$x^4 + 1 \quad (8.59)$$

De transformatie in MixColumn is een vermenigvuldiging met een gekozen polynoom modulo de reductiepolynoom. Aangevoerd kan worden dat deze vermenigvuldiging wordt bewerkstelligd door te vermenigvuldigen met een circulante matrix waarvan de elementen de coëfficiënten van de polynoom zijn. Als vermenigvuldiger is gekozen

$$03x^3 + 01x^2 + 01x + 02 \quad (8.60)$$

De coëfficiënten zijn bytes (genoteerd als tweecijferige hexadecimale getallen). Elke kolom b in het tussenresultaat wordt zodoende door het navolgende product a vervangen

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} \quad (8.61)$$

De polynoom waarmee wordt vermenigvuldigd is relatief priem ten opzichte van $x^4 + 1$. Daardoor heeft de transformatie een inverse, hetgeen nodig is om te kunnen ontcijferen. Deze inverse is de polynoom

$$0Bx^3 + 0Dx^2 + 09x + 0E \quad (8.62)$$

Sleutel-operatie

Het in elke ronde bewerken van de data met de betreffende subsleutel geschiedt eenvoudig met de exclusive or-operator. Elke databyte wordt gecombineerd met een overeenkomstige byte van de subsleutel.

$$a_{i,j} = b_{i,j} \oplus k_{i,j} \quad (8.63)$$

Vanzelfsprekend is van belang hoe het berekenen van de subsleutels in zijn werk gaat. Per ronde zijn evenzoveel sleutelbytes nodig als het aantal databytes in het blok bedraagt. De vercijfering van een 128-bits blok ofwel 16 bytes doorloopt 10 ronden en vraagt dus om $10 \times 16 = 160$ subsleutelbytes. De aanvullende AddRoundKey operatie aan het begin van de vercijfering (de whitening stap) komt daar nog extra bij. In totaal moeten dus 176 bytes worden geproduceerd. Om deze te maken wordt de ingevoerde sleutel geëxpandeerd tot de benodigde lengte. De subsleutels worden dan achtereenvolgens uit deze geëxpandeerde sleutel genomen.

De geëxpandeerde sleutel kan worden gezien als een rij met groepen van telkens vier bytes, die alleen door hun vier directe voorgangers in de rij worden bepaald. Dit maakt het mogelijk om subsleutels on-the-fly te produceren zonder tussenresultaten te moeten opslaan. Dat is weer nuttig in situaties waarin de mogelijkheden voor geheugenopslag uiterst beperkt zijn. Men behoeft dan nooit meer aan subsleutelbytes de grootste van het aantal bytes in blok- en sleutellengte te onthouden. De expansie-algoritme is

afhankelijk van de lengte van de sleutel. Bij een 256-bits sleutel wordt een iets ingewikkelder berekening uitgevoerd dan voor de kortere sleutels.

Rijndael subsleutelexpansie

- $W[]$ is array van 4-byte woorden dat wordt gevuld
- N_k is aantal 4-byte kolommen in de sleutel
- $RotByte(w)$ roteert woord $w = (a,b,c,d) \rightarrow (b,c,d,a)$
- $R[n]$ is constante $[r(n), 00, 00, 00]$
- vul posities $0 \dots N_k - 1$ van W met sleutelbytes
- voor alle overige benodigde posities i in W geldt
 - $temp = W[i - 1]$
 - als $i \bmod N_k = 0$ dan $temp = ByteSub(RotByte(temp)) \oplus R[i/N_k]$
 - als $N_k > 6$ en $i \bmod N_k = 4$ dan $temp = ByteSub(temp)$
 - $W[i] = W[i - N_k] \oplus temp$

De geëxpandeerde sleutel wordt opgebouwd uit groepjes van vier bytes, waarbij elk volgend groepje wordt gevormd door de exclusive or van een tweetal uiteenliggende voorgangers. Te weten het woord op positie i als combinatie van dat op positie $i - 1$ met dat op positie $i - N_k$, N_k is het aantal kolommen in de sleutel. Na een aantal stappen N_k wordt een aanvullende bewerking uitgevoerd met behulp van onder meer de ByteSub operatie. Begrijpen van deze subsleutelexpansie gaat het gemakkelijkst aan de hand van bovenstaande pseudocode, ontleend aan het Rijndael-rapport. De constanten r_i zijn bytes die moeten worden opgevat als elementen van $GF(2^8)$: $r_1 = x^0 = 01_{16}$, $r_2 = x^1 = 02_{16} \dots r_n = x^{n-1} = x \cdot r_{n-1}^*$.

* Vermenigvuldigen met x wordt uitgevoerd door het byte een stap naar links te roteren $b \ll 1$. Was het linkerbit een 0 dan is dit meteen het resultaat, anders moet modulo $m(x)$ gereduceerd worden. Deze reductie kan worden uitgevoerd door middel van een enkele xor operatie $b \oplus 1B$.

Ontcijfering

Om te zien hoe vercijfering en ontcijfering op elkaar aansluiten, kan worden gekeken naar een gekortwielte procedure met slechts twee opeenvolgende rondes. Het vercijferen begint met de whitening-stap, vervolgens een volle ronde en dan de ingekorte slotronde, zoals gegeven in [tabel 8.9](#). Dit rondeschema is opnieuw weergegeven aan de linkerkant van [tabel 8.12](#), de afzonderlijke rondes zijn door horizontale lijnen van elkaar gescheiden.

<i>vercijfering</i>	<i>ontcijfering</i>	<i>ontcijfering</i>
AddRoundKey	InvAddRoundKey	InvAddRoundKey
ByteSub	InvShiftRow	InvByteSub
ShiftRow	InvByteSub	InvShiftRow
MixColumn	InvAddRoundKey	InvMixColumn
AddRoundKey	InvMixColumn	InvAddRoundKey
ByteSub	InvShiftRow	InvByteSub
ShiftRow	InvByteSub	InvShiftrow
AddRoundKey	InvAddRoundKey	InvAddRoundKey

Tabel 8.12 Ontcijfering in Rijndael

In de middelste kolom staan de operaties voor het ontcijferen door de vercijferstappen in omgekeerde volgorde te nemen en ze te vervangen door hun inverse. Het oorspronkelijk rondeschema uit de vercijfering wordt aangegeven met de horizontale scheidslijnen.

In het voorgaande is al opgemerkt dat voor ontcijfering de subsleutels slechts in omgekeerde volgorde behoeven te worden aangeboden. Verder is evident dat ByteSub en ShiftRow naar believen kunnen worden omgewisseld. Dat is het gevolg van het feit dat de laatste van deze twee operaties de bytes van plaats

verwisselt, maar ze overigens intact laat. De volgorde waarin ze door ByteSub voor substitutie gepakt worden is niet van belang.

Tenslotte mag men in de ontcijfering AddRoundKey en InvMixColumn verwisselen, mits de subsleutel vóór de ontcijfering eveneens de InvMixColumn ondergaat. InvMixColumn is een lineaire transformatie en zulke transformaties hebben de eigenschap dat $A(x + k) = A(x) + A(k)$.

Gebruikmakend van deze operaties resulteert het schema in de rechterkolom van [tabel 8.12](#). De horizontale lijnen zijn zo getrokken dat de overeenkomsten in het rekenschema van vercijfering en ontcijfering duidelijk worden. De opeenvolging van functieaanroepen kan dus voor ontcijfering dezelfde zijn als voor vercijfering, waarbij de subsleutels in de omgekeerde volgorde worden aangereikt en ze waar nodig met een voorafgaande InvMixColumn zijn geprepareerd. Uit dit schema wordt nu ook duidelijk waarom in de laatste ronde van de vercijfering een MixColumn ontbreekt.

8.9 Referenties

- [1] C. E. Shannon (1949). Communications Theory of Secrecy Systems. *Bell Technical Journal*, 28, 656–715.
- [2] M. Luby and C. Rackoff (1988). How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17, 373–386.
M. Naor and O. Reingold (1999). On the construction of Pseudorandom Permutations: Luby-Rackoff Revisited. *Journal of Cryptology*, 12, 29–66.
- [3] F. H. Hinsley (1979). *British Intelligence in the Second World War, Volume I*. London: Her Majesty's Stationery Office. In hoofdstuk 7, *Intelligence on the German Economy, September 1939 to the Autumn of 1940*. Laat zien hoe grondig deze informatieverzameling en -verwerking door de Engelsen in het begin van de Tweede Wereldoorlog werd georganiseerd.
- [4] E. Biham (1992). New Types of Cryptanalytic Attacks using Related Keys. Technical Report Technion – Israel Institute of Technology.
- [5] C. H. Meyer and S. M. Matyas (1982). *Cryptography: A New Dimension in Computer Data Security*. John Wiley & Sons, Inc. p.152.

- [6] D. W. Davies and W. L. Price (1989). *Security for Computer Networks*. second edition John Wiley & Sons, Inc.
- [7] National Institute of Standards and Technology (1999). *Data Encryption Standard (DES) - FIPS PUB 46-3*. .U.S. Dept. of Commerce
- [8] K. W. Campbell and M. J. Wiener (1992). DES is not a group. In E. F. Brickell, editor (1992). *Advances in Cryptology*, number 740 in Lecture Notes in Computer Science ., pages 512–517.
- [9] W. Diffie and M. E. Hellman (1977). Exhaustive cryptanalysis of the NBS Data Encryption Standard. *IEEE Computer*, 10, 74–84.
- [10] M. E. Hellman and others (1976). Results of an initial attempt to cryptanalyze the NBS Data Encryption Standard. Technical Report Stanford University.
- [11] U. S. Senate Select Committee on Intelligence (1978). Unclassified Summary: Involvement of NSA in the Development of the Data Encryption Standard. *IEEE Communications Magazine*, 16(6), 53–55.
- [12] B. Schneier (1999). A Self-Study Course in Block-Cipher Cryptanalysis. . URL = <http://www.counterpane.com/>.
- [13] W. Diffie and M. E. Hellman (1977). Exhaustive cryptanalysis of the NBS Data Encryption Standard. *IEEE Computer*, 10, 74–84. Verschenen in 1977 maar al in 1976 ingezonden.
- [14] M. J. Wiener (1994). Efficient DES Key Search. Technical Report School of Computer Science, Carleton University, Ottawa. Gepresenteerd tijdens de Crypto'93 conferentie.
- [15] G. Garon and R. Outerbridge (1991). DES watch: An Examination of the Sufficiency of the Data Encryption Standard for Financial Institution Information Security in the 1990's. *Cryptologia*, XV, 177–193.
- [16] P. Kocher (1999). Breaking DES. *RSA Laboratories CryptoBytes*, 4.
- [17] M. E. Hellman (1980). A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory*, IT-26, 401–406.
- [18] E. Biham and A. Shamir (1991). Differential Cryptanalysis of DES-like Cryptosystems. *Journal of Cryptology*, 4, 3–72.
E. Biham and A. Shamir (1993). *Differential Cryptanalysis of Data Encryption Standard*. Springer-Verlag.
- [19] B. den Boer (1988). Cryptanalysis of FEAL. In C. Günther, editor (1988). *Advances in Cryptology*, number 330 in Lecture Notes in Computer Science ., pages 293–299. De aanval van onze landgenoot Bert den Boer op FEAL-4 (een als alternatief voor de DES aangeprezen geheimschrift) geldt als een van de voorlopers van de differentiële analyse.

- [20] E. Biham and A. Shamir (1993). *Differential Cryptanalysis of Data Encryption Standard*. Springer-Verlag.
- [21] D. Coppersmith (1994). The Data Encryption Standard (DES) and its strength against attacks. *IBM. J. Res. Develop.*, 38(3), 243–250.
- [22] A. Shamir (1986). On the Security of DES. In H. C. Williams, editor (1986). *Advances in Cryptology*, number 218 in Lecture Notes in Computer Science, pages 280–281.
- [23] M. Matsui (1993). Linear Cryptanalysis Method for DES Cipher. In T. Hellese, editor (1993). *Advances in Cryptology*, number 765 in Lecture Notes in Computer Science, pages 386–397.
- [24] M. Matsui (1994). On Correlation Between the Order of S-boxes and the Strength of DES. In A. de Santis, editor (1994). *Advances in Cryptology*, number 950 in Lecture Notes in Computer Science, pages 366–375.
- [25] P. Kocher (1996). Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In N. Koblitz, editor (1996). *Advances in Cryptology*, number 1109 in Lecture Notes in Computer Science, pages 104–113. D.m.v. een preliminary extended abstract al eerder verspreid in december 1995.
- [26] A. Hevia and M. Kiwi (1999). Strength of Two Data Encryption Standard Implementations Under Timing Attacks. *ACM Transactions on Information and System Security*, 2, 416–437.
- [27] E. Biham and A. Shamir (1997b). Differential Fault Analysis of Secret Key Cryptosystems. In B. S. Kaliski Jr., editor (1997). *Advances in Cryptology*, number 1294 in Lecture Notes in Computer Science, pages 513–525.
- [28] R. J. Anderson and M. G. Kuhn (1997). Low Cost Attacks on Tamper Resistant Devices. In *Security Protocols, 5th International Workshop*, number 1361 in Lecture Notes in Computer Science, pages 125–136.
O. Kömmerling and M. G. Kuhn (1999). Design Principles for Tamper-Resistant Smartcard Processors. In *Proceedings of the USENIX Workshop on Smartcard Technology — Smartcard '99*, pages 9–20.
- [29] P. Kocher, J. Jaffe and B. Jun (1999). Differential Power Analysis. In M. Wiener, editor (1999). *Advances in Cryptology*, number 1666 in Lecture Notes in Computer Science, pages 388–397.
- [30] J. Nechvatal and others (2000a). Status Report on the First Round of the Development of the Advanced Encryption Standard. Technical Report Nist Institute of Standards and Technology.
- [31] E. Biham and A. Shamir (1999). Power Analysis of the Key Scheduling of the AES Candidates. Technical Report Technion – Israel Institute of Technology.

- [32] D. E. Denning and others (1993). To Tap Or Not To Tap. *Communications of the Association for Computing Machinery*, 36, 25–44.
- [33] E. F. Brickell and others (1993). SKIPJACK Review, Interim Report The SKIPJACK Algorithm. Technical Report National Institute of Standards and Technology.
- [34] National Institute of Standards and Technology (1994). *Escrowed Encryption Standard (EES) - FIPS PUB 185*. .U.S. Dept. of Commerce
- [35] Ad Hoc Group of Cryptographers and Computer Scientists (1998). The Risks of Key Recovery, Key Escrow, and Trusted Third Party Encryption. Technical Report. URL = <http://www.crypto.com/papers/escrowrisks98.pdf>
- [36] E. Biham and others (1998). Initial Observations on Skipjack: Cryptanalysis of Skipjack-3XOR. Technical Report Technion – Israel Institute of Technology.
- [37] A. Shamir, E. Biham and A. Biryukov (1999). Cryptanalysis of Skipjack Reduced to 31 Rounds using Impossible Differentials. InJ. Stern, editor (1999). *Advances in Cryptology*, number 1592 in Lecture Notes in Computer Science ., pages 12–33.
- [38] X. Lai and J. Massey (1991). A Proposal for a New Block Encryption Standard. InI.B. Damgård, editor (1991). *Advances in Cryptology*, number 473 in Lecture Notes in Computer Science ., pages 389–404.
X. Lai, J. Massey and S. Murphy (1992). Markov Ciphers and Differential Cryptanalysis. InJ. Feigenbaum, editor (1992). *Advances in Cryptology*, number 576 in Lecture Notes in Computer Science ., pages 17–38.
- [39] J. Kelsey, N. Ferguson, B. Schneier and M. Stay (2000). Cryptanalytic Progress: Lessons for AES. Technical Report Counterpane Internet Security, Inc..
- [40] J. Daemen, R. Govaerts and J. Vandewalle (1994). Weak Keys for IDEA. InD. R. Stinson, editor (1994). *Advances in Cryptology*, number 773 in Lecture Notes in Computer Science ., pages 224–231.
- [41] P. Hawkes (1998). Differential-Linear Weak Key Classes of IDEA. InK. Nyberg, editor (1998). *Advances in Cryptology*, number 1403 in Lecture Notes in Computer Science ., pages 112–126.
- [42] R. L. Rivest (1995). The RC5 Encryption Algorithm. InB. Preneel, editor (1995). *Advances in Cryptology*, number 1008 in Lecture Notes in Computer Science ., pages 86–96.
- [43] B. S. Kaliski Jr. and Y. L. Yin (1998). On the Security of the RC5 Encryption Algorithm. Technical Report RSA Laboratories.

- [44] J. Kelsey, N. Ferguson, B. Schneier and M. Stay (2000). Cryptanalytic Progress: Lessons for AES. Technical Report Counterpane Internet Security, Inc..
- [45] B. Schneier (1996). *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. second edition John Wiley & Sons, Inc.
- [46] C. Charnes, L. O'Connor, J. Pieprzyk, R. Safavi-Naini and Y. Zheng (1994). Comments on the Soviet Encryption Algorithm. In A. de Santis, editor (1994). *Advances in Cryptology*, number 950 in Lecture Notes in Computer Science ., pages 433–438.
B. Schneier (1995). The GOST Encryption Algorithm. *Dr. Dobbs's Journal*.
- [47] B. Schneier (1994). Description of a New Variable"-Length Key, 64-Bit Block Cipher (Blowfish). In R. Anderson, editor (1994). *Advances in Cryptology*, number 809 in Lecture Notes in Computer Science ., pages 191–204.
- [48] A. Shimizu and S. Miyaguchi (1988). Fast Data Encipherment Algorithm FEAL. In D. Chaum and W. L. Price, editor (1988). *Advances in Cryptology*, number 304 in Lecture Notes in Computer Science ., pages 267–278.
- [49] B. den Boer (1988). Cryptanalysis of FEAL. In C. Günther, editor (1988). *Advances in Cryptology*, number 330 in Lecture Notes in Computer Science ., pages 293–299.
- [50] National Institute of Standards and Technology (1999). *Data Encryption Standard (DES) - FIPS PUB 46-3*. .U.S. Dept. of Commerce
- [51] J. Nechvatal and others (2000a). Status Report on the First Round of the Development of the Advanced Encryption Standard. Technical Report Nist Institute of Standards and Technology.
- [52] J. Nechvatal and others (2000b). Report on the the Development of the Advanced Encryption Standard (AES). Technical Report Nist Institute of Standards and Technology.
- [53] B. Schneier (1996). *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. second edition John Wiley & Sons, Inc.
- [54] B. Schneier, J. Kelsey, D. Whiting, D. Wagner and C. Hall et al. (1999). *The Twofish Encryption Algorithm, A 128-Bit Block Cipher*. John Wiley & Sons, Inc.
- [55] J. Daemen, L. Knudsen and V. Rijmen (1997). The Block Cipher Square. In E. Biham, editor (1997). *Advances in Cryptology*, number 1267 in Lecture Notes in Computer Science ., pages 149–165.

9

Publieke sleutel

9.1	Inleiding	406
9.1.1	Functie met achterdeur	407
9.1.2	Getaltheorie	408
9.1.3	Euler totiënt functie	410
9.1.4	Kwadratische rest	411
9.1.5	Legendre-symbool en Jacobi-symbool	411
9.2	Priemgetallen	412
9.2.1	Sterke priemgetallen	414
9.2.2	Priemgetallen vinden	415
9.3	Diffie-Hellman	417
9.4	RSA	420
9.4.1	Voorbeeld RSA-vercijfering	423
9.4.2	Veiligheid van RSA	425
9.5	ElGamal systeem	433
9.6	Elliptische krommen	435
9.7	Knapzaksysteem	442
9.7.1	Voorbeeld knapzak-vercijfering	445
9.7.2	Cryptoanalyse knapzak	447
9.8	Quantum computer	449
9.9	Referenties	451

9.1 Inleiding

In hoofdstuk 1 is uiteengezet dat een systeem met *publieke sleutel* (public-key cryptosysteem) z'n specifieke eigenschappen en toepassingen ontleent aan het werken met twee verschillende sleutels, één voor het vercijferen en één voor het ontcijferen. Dit in tegenstelling tot het klassiek symmetrisch cryptosysteem, waar één sleutel voor beide operaties wordt gebruikt. **Figuur 9.1** laat het verschil tussen klassieke (symmetrische) en public-key (asymmetrische) vercijfering zien.



Figuur 9.1 Symmetrische en asymmetrische vercijfering

De bijbehorende formules luiden

$$\begin{array}{ll} \text{symmetrisch systeem} & C = E(M, K) \quad \text{en} \quad M = D(C, K) \\ \text{asymmetrisch systeem} & C = E(M, K_p) \quad \text{en} \quad M = D(C, K_s) \end{array} \quad (9.1)$$

Het spreekt welhaast vanzelf dat tussen de twee sleutels K_p en K_s van het public-key cryptosysteem een verband moet bestaan. Het is bijkans ondenkbaar dat tussen twee willekeurige, ongerelateerde sleutels zo'n vercijfer/ontcijfer-relatie zal bestaan. Vanwege dat verband zal het in principe altijd mogelijk zijn om met een van de twee sleutels de andere te berekenen.

$$K_p = f(K_s) \quad \text{en} \quad K_s = g(K_p) \quad (9.2)$$

Maar, zal men zeggen, wat belet iemand dan om die berekening daadwerkelijk uit te voeren? Maakt dat het idee van het systeem met publieke sleutel dan niet bij voorbaat waardeloos? Gelukkig voor de cryptografie is het antwoord op die laatste vraag (vooralsnog) ontkennend*. Er zijn systemen bedacht, waarvoor het

uitvoeren van die berekening zó tijdrovend is (bijvoorbeeld een paar miljard jaar op de snelste computers), dat de veiligheid van het geheimschrift voor alle praktische doeleinden gegarandeerd is.

9.1.1 Functie met achterdeur

De eerste stap naar een public-key systeem wordt gezet met het begrip *oneway functie*. Zo'n functie heeft als kenmerk dat het berekenen van de waarde van de functie eenvoudig uit te voeren is, maar de omgekeerde bewerking niet. Net zoals het wel gemakkelijk is om een getal te kwadrateren, maar omgekeerd worteltrekken niet zo eenvoudig is.

$$y = f(x) \text{ is eenvoudig maar } x = f^{-1}(y) \text{ is moeilijk} \quad (9.3)$$

Voor de toepassing in een public-key cryptosysteem is moeilijk niet goed genoeg. Het moet in de praktijk *ondoenlijk* zijn. Met dit ondoenlijk wordt hier bedoeld dat het zo lastig en tijdrovend is, dat menselijkerwijs gesproken geen oplossing kan worden gevonden.

Een cryptosysteem waarin vercijferen eenvoudig maar ontcijferen ondoenlijk is, heeft nog geen praktisch nut. Wanneer het ontcijferen ook voor de legitieme ontvanger niet is te doen, wat baat het dan als communicatiemiddel? Niet dat er geen toepassingen voor zulke oneway functies zijn. Een ervan is de cryptografische checksum die een document tot een niet zomaar te vervalsen klein aantal bytes condenseert. Uitvoeren van de inverse operatie is dan niet nodig. De echtheid van het document kan worden getoetst door de vercijfering opnieuw uit te voeren en het condensaat te vergelijken met de geclaimde waarde.

Voor het ontwikkelen van een bruikbare bouwsteen voor een public-key cryptosysteem moet een variatie op de oneway functie gekozen worden, namelijk de *trapdoor-oneway-functie*. Dat is een functie met een valluik, een achterdeur. Zo'n functie is gewoonlijk net zo moeilijk te inverteren als een volbloed oneway

- * Het meest waarschijnlijke gevaar hebben deze systemen te duchten van de komst van de quantumcomputer. Die is op dit moment echter nog geen bestaande bedreiging.

functie, behalve wanneer men over extra informatie beschikt. Die extra informatie maakt het inverteren gemakkelijk, en dat wil zeggen *doenlijk*.

$$y = f(x) \text{ eenvoudig} \mid x = f^{-1}(y) \text{ ondoenlijk} \mid x = f^{-1}(y, z) \text{ eenvoudig dankzij } z \quad (9.4)$$

De kunst is nu om valluikfuncties te vinden waarvoor het achterhalen van de extra informatie niet triviaal is. De leverancier bij uitstek van zulke functies is het deel van de wiskunde dat *getaltheorie* wordt genoemd. Eens beschouwd als de koningin van de wiskunde, zo zuiver dat er geen toepassingen voor bestonden anders dan het strelen van het wiskundig wetenschappelijk vernuft, is de getaltheorie inmiddels uitgegroeid tot de hoeksteen van de moderne computerbeveiliging en een onmisbare factor in public-key-systemen.

9.1.2 Getaltheorie

In uiterst kort bestek volgt hier iets over getallen, voorzover van belang voor het begrijpen van public-key cryptografie. Wie belangstelling heeft voor de theorie en achtergronden van de gepresenteerde begrippen kan terecht bij een van de vele boeken die over getaltheorie geschreven zijn.

De verzameling gehele getallen $\mathbb{Z}$ wordt ingeperkt tot de verzameling $\mathbb{Z}_n$ door de getallen modulo n te nemen

$$\mathbb{Z}_n = \{0, 1, 2, 3, \dots, n-1\} \quad (9.5)$$

De waarde van n voldoet als regel aan $n > 1$. Door aan deze verzameling een optelling, aftrekking en vermenigvuldiging modulo n toe te voegen, ontstaat een *algebraïsche ring*, dat wil zeggen een systeem waarin kan worden gerekend.

Een beperktere versie van $\mathbb{Z}_n$ is $\mathbb{Z}_n^*$, een verzameling die uit $\mathbb{Z}_n$ ontstaat door daaruit alle getallen weg te laten die met n een factor gemeenschappelijk hebben.

$$\mathbb{Z}_n^* = \{x \in \mathbb{Z}_n : x > 0 \wedge \text{ggd}(x, n) = 1\} \quad (9.6)$$

Een voorbeeld

$$\mathbb{Z}_{10} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \text{ en } \mathbb{Z}_{10}^* = \{1, 3, 7, 9\}$$

Merk op dat $\mathbb{Z}_n^*$ alle elementen – behalve de 0 – van $\mathbb{Z}_n$ bevat indien n een priemgetal is. Een voorbeeld

$$\mathbb{Z}_{11} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\} \quad \mathbb{Z}_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

Een *primitief element* van de verzameling $\mathbb{Z}_n$ is een element waaruit door herhaald vermenigvuldigen de gehele verzameling (exclusief de 0) wordt gegenereerd*. Zo is 2 een primitief element van $\mathbb{Z}_{11}$, want de opeenvolgende machten $2^1 \bmod 11$, $2^2 \bmod 11$, enz. geven de rij

$$2, 4, 8, 5, 10, 9, 7, 3, 6, 1, 2, \dots$$

Maar 3 is geen primitief element, omdat $3^1 \bmod 11$, $3^2 \bmod 11$, enz. niet alle elementen genereren

$$3, 9, 5, 4, 1, 3, \dots$$

De verzameling $\mathbb{Z}_n^$ heeft de volgende eigenschappen mits $\text{ggd}(a, n) = 1$*

1. voor alle paren x, y geldt $ax \equiv ay \bmod n$ als $x \equiv y \bmod n$
2. noem $a\mathbb{Z}_n^* = \{ax \bmod n \text{ voor alle } x \text{ in } \mathbb{Z}_n^*\}$ dan $a\mathbb{Z}_n^* = \mathbb{Z}_n^*$
3. er is een b in $\mathbb{Z}_n^*$ zodanig dat $ba \equiv 1 \bmod n$, dus de multiplicatieve inverse modulo n bestaat

Uit bovenstaande eigenschappen volgt dat $\mathbb{Z}_n^*$ een Abelse groep is onder vermenigvuldiging modulo n . $\mathbb{Z}_n^*$ wordt de multiplicatieve groep van $\mathbb{Z}_n$ genoemd. In het bijzonder als voor n een priemgetal p wordt gekozen is de *ring* $\mathbb{Z}_p = \{0, 1, 2, \dots, p-1\}$ een *eindig veld* met p elementen (de verzameling $\mathbb{Z}_p^*$ aangevuld met de 0) en wordt aangeduid met $\text{GF}(p)$. GF is een afkorting voor Galois Field[†]. Het is vooral om de eigenschappen van $\text{GF}(p)$ dat in de public-key-cryptografie zoveel met verzamelingen modulo een priemgetal wordt gewerkt.

* Niet in elke verzameling bestaat zo'n element, bijvoorbeeld niet in $\mathbb{Z}_8^*$

† Vernoemd naar de veelbelovende, maar in een duel jonggestorven Franse wiskundige Évariste Galois (1811–1832).

In de praktijk zal men werken met grote tot zeer grote getallen. Getallen zo groot, dat ze niet meer passen in de rekeneenheid van de computer. Dit heeft tot gevolg dat voor alle rekenoperaties als optellen, vermenigvuldigen en delen het rekenwerk moet worden opgesplitst: gerekend wordt in wat men noemt multilengte arithmetiek. Er zijn algoritmen ontwikkeld die het rekenwerk sneller doen verlopen dan met de standaard pen-en-papiermethode. Bijvoorbeeld om de veelvoorkomende modulo-operatie te kunnen uitvoeren zonder te hoeven delen^[1], want dat is een notoire lastpak*.

Ook met slimme algoritmen blijft de benodigde multilengte arithmetiek een bottleneck, in het bijzonder voor de relatief trage processoren zoals in chipkaarten. Reden waarom systemen met publieke sleutel in het algemeen meer gebruikt worden voor het versleutelen van kleine hoeveelheden gegevens (zoals wachtwoorden) dan voor versleuteling in bulk (zoals videoconferencing).



Euler

9.1.3 Euler totiënt functie

Een belangrijk getal is het aantal elementen in $\mathbb{Z}_n^*$. Dit getal wordt aangeduid met $\phi(n)$ en heet de *Euler totiënt functie*. De waarde van $\phi(n)$ wordt gevonden door te tellen hoeveel getallen van 1 tot n relatief priem zijn ten opzichte van n .

$$\phi(n) = \#i \text{ met } \text{ggd}(i, n) = 1, \quad i = 1, 2, \dots, n-1 \quad (9.7)$$

In het bijzondere geval van n een priemgetal geldt $\phi(n) = (n-1)$, hetgeen direct volgt uit de definitie van een priemgetal (slechts deelbaar door 1 en zichzelf). Bij een product volgt de waarde van ϕ uit de ontbinding in factoren. Zodat voor bijvoorbeeld het product van twee priemgetallen p en q wordt berekend

$$\phi(p \cdot q) = \phi(p)\phi(q) = (p-1)(q-1) \quad (9.8)$$

* Delen door een willekeurig getal is de meest kostbare operatie. Delen is dragelijk te maken als kan worden volstaan met delen door machten van 2.

9.1.4 Kwadratische rest

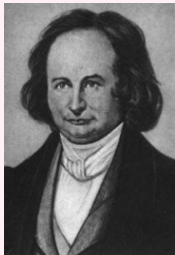
Stel x en p zijn onderling ondeelbaar, d.w.z. $\text{ggd}(x, p) = 1$. Dan is x dus een element van $\mathbb{Z}_p^*$. Het getal x heet een *kwadratische rest* modulo p als er een getal y is waarvoor het volgende geldt

$$\exists y \in \mathbb{Z}_p^*: y^2 \equiv x \pmod{p} \quad (9.9)$$

De door bovenstaande vergelijking gedefinieerde verzameling van kwadratische resten in $\mathbb{Z}_p^*$ heet $\mathbb{Q}_p$. In het voorbeeld van $\mathbb{Z}_{11}^*$ ziet men de kwadratische resten 1, 3, 4, 5 en 9 optreden, want $(\pm 1)^2 = 1$, $(\pm 2)^2 = 4$, $(\pm 3)^2 = 9$, $(\pm 4)^2 = 5$ en $(\pm 5)^2 = 3$.



Legendre



Jacobi

9.1.5 Legendre-symbool en Jacobi-symbool

Het zogenaamde *Legendre-symbool* wordt gebruikt om een getal te karakteriseren als kwadratische rest. Voor een priemgetal p groter dan 2 en een geheel getal a heeft dit Legendre-symbool de volgende waarde

$$L(a, p) = \left(\frac{a}{p} \right) = \begin{cases} 0 & \text{als } p \mid a \\ +1 & \text{als } a \in \mathbb{Q}_p \\ -1 & \text{als } a \notin \mathbb{Q}_p \end{cases} \quad (9.10)$$

Het *Jacobi-symbool* is een generalisatie van het Legendre-symbool voor een oneven samengesteld getal n . De waarde van het Jacobi-symbool volgt de ontbinding van n in factoren

$$n = p_1^{e_1} p_2^{e_2} \dots \rightarrow J(a, n) = \left(\frac{a}{n} \right) = \left(\frac{a}{p_1} \right)^{e_1} \left(\frac{a}{p_2} \right)^{e_2} \dots \quad (9.11)$$

Voor de berekening van Jacobi-symbolen kan gebruik gemaakt worden van een recursief algoritme^[2]. Dit algoritme heeft als prettige bijkomstigheid dat het niet nodig is om de ontbinding van n te kennen.

Berekening Jacobi-symbool

- als $a = 0$ of $a = 1$ dan $J(a, n) = a$
- haal alle m factoren 2 uit a zodat $a = 2^m \cdot a'$ en a' oneven is
- geef s de volgende waarde

$$s = \begin{cases} 1 & m \text{ is even of } n \equiv 1 \text{ of } 7 \pmod{8} \\ -1 & n \equiv 3 \text{ of } 5 \pmod{8} \end{cases}$$

- als $n \equiv 3 \pmod{4}$ en $a' \equiv 3 \text{ of } 4 \pmod{8}$ dan $s = -s$
- geef n' de waarde $n \pmod{a'}$
- het resultaat is de recursie $J(a, n) = s \cdot J(a', n')$

9.2 Priemgetallen

Om met public-key-systemen zoals RSA te kunnen werken heeft men priemgetallen nodig. Bovendien moeten het grote priemgetallen zijn, in de orde van 100–200 of zelfs meer decimale cijfers. Deze priemgetallen moeten op een efficiënte manier worden gevonden. Efficiënter in ieder geval dan door te proberen kandidaatpriemgetallen in factoren te ontbinden, want dat zou het paard achter de wagen spannen zijn. Immers, we zullen zien dat dit hetgene is wat nu juist breken van een bekend public-key systeem als RSA ondoenlijk maakt. Gelukkig bestaan er niet op factoriseren gebaseerde methoden om vast te stellen of een getal al dan niet een priemgetal is.

De eerste vraag die kan worden gesteld luidt als volgt. Zijn er eigenlijk wel genoeg priemgetallen? Het is duidelijk dat priemgetallen steeds schaarser worden naarmate de getallen groter zijn. Dat valt gemakkelijk in te zien aan de hand van de zeef van Eratosthenes, een algoritme waarmee priemgetallen kunnen worden gevonden. Men begint met het eerste priemgetal 2 en verwijdert uit alle volgende getallen de veelvoud

van 2. Het eerstvolgend overgebleven getal is weer een priemgetal namelijk 3 en hiermee wordt hetzelfde gedaan, enzovoorts. Hoe verder in de rij getallen, hoe meer veelvouden er al uit gezeefd zijn.

Op zich is het niet erg dat priemgetallen steeds schaarser worden, mits er maar genoeg overblijven. Te weinig priemgetallen van een bepaalde grootte zou betekenen dat een cryptoanalist slechts een beperkt aantal waarden hoeft te testen. Om te zien of genoeg priemgetallen overblijven, wordt gekeken naar aantal en verdeling van de priemgetallen op de getallenrechte. Het is een probleem waar de wiskundige Carl Friedrich Gauss (1777–1855) zich al mee bezig heeft gehouden. Hij vond een uitdrukking die bij benadering het aantal priemgetallen geeft dat zich bevindt tussen 2 en een getal x . Een latere en betere benadering is afkomstig van Riemann, maar die van Gauss is hier goed genoeg.

$$\pi(x) \approx \frac{x}{\ln x} \quad (9.12)$$

Voor getallen tot honderd decimale cijfers is het resultaat ten naaste bij $4,34 \times 10^{97}$ priemgetallen. Trek daarvan af de $0,44 \times 10^{97}$ priemgetallen tot 99 cijfers. Wat overblijft is een schatting van het aantal priemgetallen van 100 cijfers. Dat zijn er 4×10^{97} , een 4 gevolgd door 97 nullen! Dit resultaat haalt alle kou wel uit de lucht. Met meer dan 10^{97} priemgetallen van honderd cijfers om uit te kiezen zal men niet snel nee behoeven te verkopen. Zelfs als de tegenstander weet dat het om een getal van 100 cijfers gaat, dan nog is het onbegonnen werk ze allemaal te proberen.

Een volgende vraag is hoever de priemgetallen uit elkaar zullen liggen. De beantwoording hiervan is van belang voor het zoeken naar priemgetallen. Bij het testen van een aantal opeenvolgende kandidaten loopt men het risico heel vaak mis te grijpen. Of zal het als regel wel meevallen? Het antwoord ligt eveneens besloten in de door Gauss gevonden uitdrukking voor het aantal priemgetallen. Daaruit valt af te leiden wanneer gemiddeld het volgende priemgetal zich zal aandienen, omdat ze gemiddeld $\ln x$ getallen uitelkaar liggen. Voor honderd cijfers is dat na ruwweg 230 getallen. Het valt dus nogal mee, temeer omdat het zoeken beperkt kan blijven tot getallen eindigend op een 1, 3, 7 of 9. Maar let op, het



Gauss

kan ook tegenzitten want er bestaan willekeurig grote intervallen zonder priemgetal*.

9.2.1 Sterke priemgetallen

Zomaar een willekeurig priemgetal is eigenlijk niet goed genoeg gebleken. Dat komt door de wisselende efficiëntie van methoden voor het factoriseren bij bepaalde samenstellingen van het te factoriseren getal. Een samengesteld getal kan bijvoorbeeld alleen uit kleine factoren bestaan, of veel kleine factoren en een enkele grote, of juist veel grote priemfactoren hebben. Men spreekt van een *sterk priemgetal* wanneer dit een naar verhouding snel succes van de (bekende) methoden van factoriseren verhindert. Om priemgetal p een sterk priemgetal te mogen noemen moet het voldoen aan een aantal eisen

1. $p - 1$ heeft een grote priemfactor q
2. $p + 1$ heeft een grote priemfactor r
3. $q - 1$ heeft eveneens een grote priemfactor

Soms verzwaart men deze eisen en verlangt ook nog dat $r - 1$ een grote priemfactor heeft. Met een grote priemfactor wordt bedoeld dat bij het factoriseren tenminste een van de factoren groot moet zijn. Wat precies groot genoeg is blijft enigszins in het vage, maar hangt samen met de specifieke methode van factoriseren waartegen men zich wil wapenen. Bij een willekeurige keuze van het priemgetal blijkt de kans op een niet-sterk priemgetal overigens niet zo groot, zodat men daarop zou kunnen vertrouwen. Aan de andere kant kost het ook weer niet veel extra werk om een sterk priemgetal te vinden. In sommige gevallen worden nog andere eisen opgelegd aan priemgetallen, zoals bijvoorbeeld voor de digitale handtekeningstandaard, de Digital Signature Standard (DSS)^[3]. Door John Gordon is een manier aangegeven om systematisch een sterk priemgetal te vinden^[4].

* De zogenaamde tweelingpriemgetallen, zoals 17-19 en 29-31, staan veel dichter bijeen dan de gemiddelde afstand. Dus zullen er ook opeenvolgende priemgetallen zijn die verder dan het gemiddelde uiteen liggen.

9.2.2 Priemgetallen vinden

Om te bepalen of een getal een priemgetal is kunnen in principe twee wegen bewandeld worden. De eerste is het toepassen van een *deterministische* algoritme, d.w.z. een methode die met zekerheid kan vaststellen of een getal een priemgetal is. Een eenvoudig maar onpraktisch voorbeeld is het testen van alle getallen tot aan de vierkantswortel om te zien of bij deling een rest nul overblijft. Het beste deterministische algoritme om te bepalen of een getal een priemgetal is, is het naar de ontdekkers Agrawal, Kayal en Saxena genoemde AKS algoritme^[5]; later door anderen zoals Lenstra c.s. verfijnd en versneld^[6]. Hoewel dit een polynomiaal algoritme is, d.w.z. de rekentijd groeit in beperkte mate mee met de grootte van de te onderzoeken getallen, kost het desondanks zoveel rekentijd dat de hierna te behandelen algoritmen nog niet uit de markt geprijsd zijn.

Het alternatief voor een deterministische algoritme is een *probabilistische* algoritme. Dat is een methode die een marge van onzekerheid laat. Er bestaat dan bijvoorbeeld een kans van 50% dat het antwoord goed is. Dit lijkt niet erg aantrekkelijk, maar dat verbetert snel als zulke testen onafhankelijk van elkaar een willekeurig aantal keren herhaald kunnen worden. Bij twee onafhankelijke testen met elk 50% kans op een foutief antwoord is de totale kans op een misser immers nog maar 25%. Herhaal zo'n test 20 keer en de kans op een misser is kleiner dan één op de miljoen. De probabilistische methoden hebben het voordeel dat ze minder rekenintensief zijn dan de deterministische methoden, terwijl men de kans op mislukking door keuze van het aantal herhalingen willekeurig klein kan maken.

Strategie voor het zoeken naar een priemgetal

- kies een beginwaarde
- doorloop de opeenvolgende oneven getallen niet eindigend op een 5
- gebruik een tabel van priemgetallen tot een bepaalde grenswaarde om door deling snel uit te maken of het getal een kleine priemfactor heeft want daarop is grote kans
- overleeft de kandidaat die test doe dan een probabilistische test

Bij het testen op priemgetallen is nog een opmerking van belang. Gegeven de afmetingen van de getallen moeten alle berekeningen met speciale rekenprogramma's worden uitgevoerd. De registers van de processor zijn bij lange na niet groot genoeg om de getallen te kunnen bevatten. De verhouding tussen de tijd nodig voor het uitvoeren van een volledige machtsverheffing en de uitvoering van een deling bepaalt de meest gunstige grenswaarde. Een alternatief voor de deling is om producten van kleine priemgetallen te nemen en met de algoritme van Euclides de grootste gemene deler te bepalen.

Solovay-Strassen-test

De Solovay-Strassen-test is de oudste populaire priemtest en berust op een kenmerk afkomstig van Euler. De functie $J(a, n)$ is het Jacobi-symbool van [vergelijking \(9.11\)](#).

$$n > 2 \text{ en priem} \rightarrow J(a, n) \equiv a^{(n-1)/2} \pmod{n}, \forall a: \text{ggd}(a, n) = 1 \quad (9.13)$$

Er hoeft slechts naar oneven getallen gekeken te worden. Als het onderzochte getal n samengesteld is, dan kan het zijn dat voor een willekeurig gekozen getal $1 < a < n$ wordt gevonden dat

$$\text{ggd}(a, n) > 1 \text{ en/of } J(a, n) \not\equiv a^{(n-1)/2} \pmod{n} \quad (9.14)$$

Het getal a heet een *Euler-getuige* voor de primaliteit van n . Indien niet is voldaan aan het Euler-kenmerk kan het geteste getal geen priemgetal zijn; de getuige laat daarover dan geen twijfel bestaan. Het is ook

mogelijk dat wel aan het kenmerk wordt voldaan terwijl n toch een samengesteld getal is. Getal a heet dan een *Euler-leugenaar* en n is een pseudopriemgetal voor deze a . Ten hoogste $\phi(n)/2$ van alle a 's kunnen een Euler-leugenaar zijn en bijgevolg is er na k keer testen met verschillende a een kans van minder dan 2^{-k} dat een product onopgemerkt door alle testen komt*.

Miller-Rabin-test

De Miller-Rabin-test is een verfijning van de Solovay-Strassen-test. Ook hier kiest men een getal a als getuige voor de primaliteit van het kandidaatpriemgetal. Als de kandidaat faalt in de test heet a daarvoor een sterke getuige. Slaagt de kandidaat dan kan dat net als bij Solovay-Strassen ten onrechte zijn. In dit geval heet a een sterke leugenaar. Ten hoogste 25% van alle a 's geeft niet het goede antwoord in het geval n een product is. Verder is het zo dat elke sterke leugenaar eveneens een Euler-leugenaar is, terwijl het omgekeerde niet geldt. Miller-Rabin scoort daarom nooit slechter dan Solovay-Strassen en vaak beter. Hierom geniet het gebruik van Miller-Rabin de voorkeur boven Solovay-Strassen.

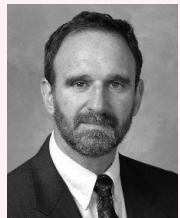
De getallen a kunnen willekeurig worden gekozen, maar men kan ook een serie kleine priemgetallen nemen. Daar zit achter dat het product $a_1 a_2$ van twee sterke leugenaars vaak ook weer een sterke leugenaar is. Testen met dat product levert dan bijna zeker niets nieuws op.

9.3 Diffie-Hellman

De eersten die een implementatie van een public-key-systeem realiseerden, waren Whitfield Diffie en Martin Hellman^[7]. Eigenlijk is hun systeem niet zozeer een cryptosysteem alswel een manier om een geheime sleutel te delen. Die geheime sleutel kan vervolgens worden gebruikt voor het versleutelen van berichten.



Diffie



Hellman

* Functie $\phi(n)$ is de Euler totiënt functie. De hoogste waarde ervan is $n-1$, namelijk als n een priemgetal is.

In het algemeen zal men voor een groep van n personen $n(n - 1)/2$ sleutels moeten vastleggen, wanneer men elk willekeurig paar een eigen veilige communicatie wil geven. Het aantal sleutels groeit zodoende kwadratisch met de omvang van de groep. Een onmogelijke toestand als iedere internetgebruiker veilig met elke andere internetgebruiker moet kunnen communiceren. Er is dus zeker empoel voor een methode waarmee het aantal benodigde sleutels slechts evenredig met het aantal deelnemers groeit.

De grondslag voor het Diffie-Hellman-systeem voor sleuteluitwisseling is gelegen in het feit dat het bepalen van logaritmen in $GF(p)$ tot de groep van moeilijke c.q. ondoenlijke problemen behoort. De basisgegevens van het Diffie-Hellman-systeem zijn een priemgetal p en een primitief element a van $GF(p)$. Kies nu een getal x kleiner dan p en verhef a tot de x -de macht modulo p

$$y = a^x \bmod p \quad (9.15)$$

De omgekeerde bewerking heet de *discrete logaritme* en valt voor voldoende groot gekozen p onder de ondoenlijke problemen

$$x = \log_a y \bmod p \quad (9.16)$$

De sleuteldistributie wordt nu als volgt opgezet. De getallen p en a worden geschikt gekozen en zijn aan alle deelnemers bekend. Elke deelnemer kiest bovendien een x en berekent daaruit y met **vergelijking (9.15)**. Alle deelnemers publiceren hun waarde van y , maar houden de bijbehorende x zorgvuldig geheim.

De openbare Diffie-Hellman sleutel bestaat uit

- een priemgetal p
- een getal a uit $0 \dots p - 1$
- voor elke deelnemer een getal $y = a^x \bmod p$

De geheime Diffie-Hellman sleutel bestaat uit

- voor elke deelnemer een getal x uit $0 \dots p - 1$

Twee deelnemers, i en j kunnen dan op de volgende wijze een gezamenlijke sleutel vastleggen. Stel i wil een bericht naar j sturen. Voor het berekenen van de sleutel neemt i de door j gepubliceerde y_j en voert met zijn eigen geheime x_i een machtsverheffing uit

$$K_{ij} = y_j^{x_i} \bmod p \quad (9.17)$$

Met de verkregen sleutel K_{ij} kan i nu het bericht vercijferen en zal daar als regel een symmetrisch geheimschrift voor kiezen, bijvoorbeeld de AES.

Hoe kan ontvanger j het bericht ontcijferen? Weliswaar beschikt j niet over de geheime x_i , maar j kan wel de openbare y_i opzoeken. Samen met de eigen geheime x_j rekt hij uit

$$K_{ij} = y_i^{x_j} \bmod p \quad (9.18)$$

en kan nu het bericht met K_{ij} ontcijferen.

Het systeem werkt omdat de gezamenlijke sleutel langs twee wegen kan worden berekend

$$K_{ij} = y_i^{x_j} \bmod p = (a^{x_i})^{x_j} = a^{x_i x_j} = (a^{x_j})^{x_i} = y_j^{x_i} \bmod p \quad (9.19)$$

Elk van de deelnemers combineert de eigen geheime informatie met de publieke sleutel van de ander.

Hoe zit het nu met de mogelijkheden tot 'onbevoegd' ontcijferen? Een derde beschikt noch over x_i noch over x_j , maar zal een van de twee moeten vinden alvorens K_{ij} te kunnen berekenen om daarmee het bericht te kunnen ontcijferen. Voor het vinden van een van deze twee moet de discrete logaritme van **vergelijking (9.16)** worden opgelost. De vooronderstelling is dat dit ondoenlijk is en daarmee is het geheim veilig.

Zoals voor alle public-key-systemen geldt ook voor Diffie-Hellman dat een zorgvuldige parameterkeuze van het grootste belang is. Indien verschillende organisaties allemaal hetzelfde basispriemgetal gebruiken, dan is het na (weliswaar tijdrovende) voorberekeningen mogelijk ze alle tegelijk aan te vallen. Praktijkstudies hebben uitgewezen dat dit een reëel gevaar vormt, bijvoorbeeld waar gebruik gemaakt wordt

van openbaar beschikbare priemgetallenverzamelingen^[8]. In het aangehaalde artikel wordt het niet ondenkbaar genoemd dat de National Security Agency van dit mechanisme gebruik heeft gemaakt om veilig geachte VPN-verbindingen te ontcijferen.

9.4 RSA

De eerste en nog steeds betrouwbare implementatie van een volwaardig public-key cryptosysteem is afkomstig van de drie onderzoekers Ronald Rivest, Adi Shamir en Leonard Adleman. De aanduiding RSA staat voor hun initialen. Het systeem werd in 1978 gepubliceerd in de Communications of the ACM^[9]. Basis voor het systeem is de problematiek van het factoriseren, de ontbinding van getallen in factoren. Zo eenvoudig als dat lijkt voor kleine getallen zo moeilijk blijkt het voor producten van grote getallen, zoals van meer dan 100 decimale cijfers. Toen in 1999 een product van 140 decimale cijfers werd gefactoriseerd, gold dit als een prestatie van belang die onmiddellijk over de gehele wereld bekend gemaakt werd^[10].

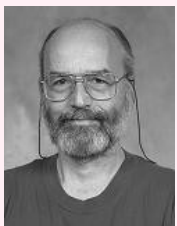
Ook al gaat de snelheid van computers vooruit en worden de algoritmen voor factoriseren nog steeds verbeterd, door het kiezen van steeds grotere getallen kan het cryptosysteem het nog altijd winnen. Op dit moment geldt een getal van 512 bits als onvoldoende veilig, maar lijken 1024 of 2048 bits voorlopig toereikend (ook al wordt aan die conclusie geknabbeld)*. Gelukkig voor RSA wordt het ontbinden in factoren al met een paar cijfers extra onevenredig veel moeilijker. Dat is vergelijkbaar met het vooruitkijken bij een spelletje schaak: met elke zet die men verder vooruit wil kijken neemt het aantal mogelijkheden sterk toe.

De oorzaak is gelegen in de complexiteit van het factoriseren als functie van de lengte van het te ontbinden getal. Naar eerder gangbare inzichten bedraagt deze complexiteit

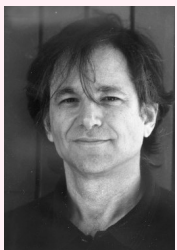
* Voor de omrekening naar het aantal decimale cijfers kan men gebruik maken van $2^{10} = 1024 \approx 10^3$. Hanteer als vuistregel dat elke 10 bits met 3 decimale cijfers overeenkomen.



Rivest



Shamir



Adleman

$$\mathcal{O}(e^{\sqrt{\log n \log \log n}}) \quad (9.20)$$

Sedertdien is de Generalized Number Field Sieve voor het factoriseren van grote getallen ontwikkeld en is deze complexiteit verder verscherpt tot

$$\mathcal{O}(e^{\sqrt{\frac{64}{9} \log n (\log \log n)^2}}) \quad (9.21)$$

Hoe men het ook wendt of keert, het is een exponentiële groeifactor waardoor een kleine vergroting van n toch een flinke verzwaring van het rekenprobleem veroorzaakt.

Voor het opzetten van een RSA-vercijfering gaat men als volgt te werk. Er zijn twee openbare gegevens, een modulus n en een exponent e . Zo dadelijk zal worden uitgelegd hoe deze getallen moeten worden gekozen. Omdat modulo n gerekend gaat worden, moet het te vercijferen bericht opgesplitst worden in blokken van overeenkomstige lengte. Anders zou immers door de reductie modulo n een deel van de informatie verloren gaan. Vercijferen met RSA is niets anders dan machtsverheffen met exponent e en reductie modulo n . Deze e heet de *openbare exponent* (vercijfersleutel K_1), omdat daarmee iedereen een vercijfering kan uitvoeren

$$C = E(M, K_1) = M^e \bmod n \quad (9.22)$$

Voor het ontcijferen heeft men de *geheime exponent* nodig, die gewoonlijk met d wordt aangeduid (de ontcijfersleutel K_2)*. Ontcijfering geschiedt wederom door machtsverheffen met reductie modulo n

$$M = D(C, K_2) = C^d \bmod n \quad (9.23)$$

Waarom werkt dit en hoe moeten de parameters van het systeem gekozen worden? Het getal n is het product van twee priemgetallen p en q . Deze beide priemgetallen moeten overigens wel met zorg uitgezocht worden, anders zijn er methoden die toch nog redelijk vlot voor een ontbinding in factoren kunnen zorgen. Dat is van belang omdat de veiligheid van RSA geheel afhangt van de moeilijkheid om de factoren p en q te bepalen uit een bekende n . Eén van de criteria is bijvoorbeeld dat deze priemgetallen in lengte

* Lees e als de eerste letter van encryption en d als die van decryption.

niet teveel van elkaar mogen verschillen. Voor een 300-cijferige 1024-bit RSA zal elk priemgetal dus zo rond de 150 decimale cijfers tellen.

Nadat p en q – en daarmee ook n – zijn vastgelegd wordt van het paar exponenten e en d één van de twee gekozen. Die keuze is niet geheel willekeurig, maar moet aan bepaalde eisen voldoen. De gekozen exponent en $\phi(n) = (p - 1)(q - 1)$ moeten relatief priem zijn. D.w.z. de grootste gemene deler met $\phi(n)$ is 1. In de praktijk wil men nog wel eens kiezen voor een kleine waarde van de vercijferexponent e (bijvoorbeeld 3), omdat dan degene die een bericht moet vercijferen het minste werk heeft*. Verderop zal worden aangetoond waarom dat misschien toch niet zo'n goede keuze is. Wanneer e gekozen is wordt d berekend zodanig dat aan de gestelde eisen is voldaan. Hetgeen betekent dat e en d elkaars inverse modulo $\phi(n)$ moeten zijn. Interessant is dat die berekening gebruikt maakt van een oude algoritme, namelijk de algoritme van Euclides voor de berekening van de grootste gemene deler.

De openbare RSA sleutel bestaat uit

- een getal n , het product van twee grote priemgetallen p en q
- een getal e , de openbare encryptie-exponent

De geheime RSA sleutel bestaat uit

- de twee priemgetallen p en q
- een getal d met $ed \equiv 1 \pmod{\phi(n)}$, de geheime decryptie-exponent

Rest de vraag, waarom die speciale rol van $\phi(n) = (p - 1)(q - 1)$? Het antwoord zal blijken als wordt aangetoond hoe het komt dat vercijferen en ontcijferen op de RSA-manier werken. Het product $(p - 1)(q - 1) = \phi(pq)$ is overigens de waarde van de Euler totiënt functie voor $n = pq$. Nu is alles in gereedheid om te laten zien waarom RSA werkt. Uitgevoerd wordt een vercijfering gevolgd door een ontcijfering

* Gezien de grote getallen waarmee gewerkt wordt, is het nog niet zo eenvoudig om dat snel en efficiënt te doen. Er zijn speciale manieren bedacht om de benodigde vermenigvuldigen te doen.

$$D(E(M, e), d) = (M^e \bmod n)^d \bmod n = M^{ed} \bmod n \quad (9.24)$$

In herinnering wordt gebracht dat e en d zodanig gekozen zijn dat ze modulo $\phi(n)$ elkaars inverse zijn, hetgeen betekent dat

$$ed = 1 \bmod \phi(n) = k \cdot \phi(n) + 1 \quad \text{voor zekere gehele } k \quad (9.25)$$

Merk op dat gebruik gemaakt wordt van de definitie van de modulo operator om ed te herschrijven als 1 plus een geheel aantal malen de modulus. De laatste vergelijking invullend in de encryptie-decryptie uitdrukking levert

$$M^{ed} \bmod n = M^{k \cdot \phi(n) + 1} \bmod n = M^{k \cdot \phi(n)} M \bmod n \quad (9.26)$$

Nu wordt nog een resultaat uit de getaltheorie te hulp geroepen, namelijk de stelling van Euler die zegt*

$$M^{\phi(n)} = 1 \bmod n \quad (9.27)$$

Zodat er komt

$$M^{k \cdot \phi(n)} M \bmod n = (M^{\phi(n)})^k M = 1^k M = M \bmod n = M \quad (9.28)$$

Aldus blijkt het resultaat van de ontcijferoperatie op het vercijferresultaat weer het oorspronkelijke bericht op te leveren.

9.4.1 Voorbeeld RSA-vercijfering

Het voorbeeld van een vercijfering in het RSA-systeem is ontleend aan het oorspronkelijke artikel van Rivest, Shamir en Adleman zelf. De gekozen priemgetallen zijn $p = 47$, $q = 59$ en dus is $n = 47 \cdot 59 = 2773$. Gekozen is $d = 157$, waaruit kan worden berekend dat $e = 17$ moet zijn.

* Deze stelling van Euler verlangt dat $\text{ggd}(M, n) = 1$. Maar ook als dat niet het geval is, gaat RSA toch goed. De kans dat het zich voordoet is overigens bijzonder klein.

De waarde van de Euler totiënt functie is nodig om d en e te kunnen kiezen en bedraagt $\phi(n) = 46.58 = 2668$. De gekozen $d = 157$ is relatief priem t.o.v. $\phi(n)$, zoals meteen zal blijken bij de berekening van de bijbehorende e . Die berekening kan worden gedaan volgens het aangevulde schema van Euclides voor de grootste gemene deler

$$\begin{array}{rcl} 1 \cdot 2668 + & 0 \cdot 157 & = 2668 \\ 0 \cdot 2668 + & 1 \cdot 157 & = 157 \\ 1 \cdot 2668 + & -16 \cdot 157 & = 156 \\ -1 \cdot 2668 + & 17 \cdot 157 & = 1 \end{array}$$

Aldus blijkt $\text{ggd}(157, \phi) = 1$ en volgt 17 als waarde voor e . We kunnen nog controleren dat inderdaad $ed \bmod \phi = 17 \cdot 157 \bmod 2668 = 1$.

Met deze parameters kan nu een vercijfering worden opgezet. De te vercijferen blokken moeten klein genoeg blijven om te voorkomen dat door de reductie modulo 2773 informatie verloren gaat. Het hoogste getal dat aldus kan worden gecodeerd is 2772. Voor het gemak wordt de verzameling symbolen beperkt tot de 26 hoofdletters aangevuld met de spatie (aangeduid door een $\square$), te coderen als respectievelijk 01–26 en 00. De waarde van n laat toe dat twee, in één blok naast elkaar staande lettercoderingen, probleemloos kunnen worden vercijferd want de grootste combinatie $ZZ = 2626$ is kleiner dan $n = 2773$.

Rivest, Shamir en Adleman vercijferden de volgende tekst:

ITS $\square$ ALL $\square$ GREEK $\square$ TO $\square$ ME $\square$ = 0920190001121200071805051100201500130500

De twee-lettercombinatie IT = 0920 wordt bij de vercijfering verheven tot de 17^e macht en levert 0948. Dat resultaat wordt weer ontcijferd door verheffing tot de macht 157.

$$IT = 0920^{17} \bmod 2773 = 0948 \quad \text{en} \quad 0948^{157} \bmod 2773 = 0920 = IT$$

	I	T	S	□	A	L	L	□	G	R	E	E	K	□	T	O	□	M	E	□
M =	0920	1900	0112	1200	0718	0505	1100	2015	0013	0500										
C =	0948	2342	1084	1444	2663	2390	0778	0774	0219	1655										

Tot slot een opmerking over het uitvoeren van de machtsverheffing. Een handige methode hiervoor is om gebruik te maken van de tweetallige representatie van de exponent omdat $M^{17}_{10} = M^{10001}_2$. Hiermee kan de machtsverheffing worden uitgevoerd door vijf keer te kwadrateren en voor de 1'en een extra vermenigvuldiging met M in te lassen. Voor het nog efficiënter uitvoeren van de vermenigvuldiging zijn diverse snelle algoritmen ontwikkeld. Behandeling daarvan zou te ver voeren.

9.4.2 Veiligheid van RSA

Gegeven het feit dat RSA momenteel het belangrijkste public-key-geheimschrift genoemd mag worden, is de vraag naar de veiligheid ervan buitengewoon belangrijk.

Factoriseren van n

Het meest directe aangrijpingspunt voor een aanval op RSA is factoriseren van n . Met de toenemende populariteit van RSA is ook de interesse voor het ontbinden in factoren gegroeid. Aan het al bestaande instrumentarium zijn de laatste decennia enkele krachtige nieuwe methoden toegevoegd. Omdat het benodigde rekenwerk echter exponentieel toeneemt met de grootte van de exponent in de machtsverheffing, is het steeds mogelijk om de techniek van het factoriseren een stap voor te blijven.

#	schatting 1985	realisatie
100	1 jaar	
125	100 jaar	1994: RSA-129
150	10.000 jaar	1999: RSA-155
175	700.000 jaar	2003: RSA-176
200	30 miljoen jaar	2005: RSA-200
225	1 miljard jaar	2009: RSA-232
250	60 miljard jaar	

Tabel 9.1 Factoriseren rond 1985 en realisering

De middelste kolom van [tabel 9.1](#) geeft een indruk van de stand van de techniek zoals men die in 1985 zag. In die tijd kon men de grens tussen veilig en onveilig ergens bij de 150 decimale cijfers ofwel 500 bits leggen. Vandaar dat implementaties met een RSA-modulus van 512 bits in zwang kwamen. Inmiddels zijn de computers sneller geworden en zijn betere factorisatiemethoden ontwikkeld. Begin 1999 kon in enkele maanden een getal van 140 cijfers worden gefactoriseerd. Enkele maanden later bepaalde een internationaal team van wetenschappers onder leiding van Herman te Riele van het Centrum voor Wiskunde en Informatica (CWI) in Amsterdam de priemfactoren van RSA-155, een getal van 512 bits^{*}. In 2005 werd RSA-200 gefactoriseerd en in 2009 een product van 232 cijfers[†].

Vandaar dat toen een RSA-modulus van 1024 bits werd aanbevolen, met ongeveer 300 cijfers weer even boven de gevarengrens. Op dat moment werd geschat dat zo'n modulus het tot ongeveer het jaar 2020 zou uithouden. Maar het is bijna geen vraag meer of dit voor gegevens die voor langere tijd ontoegankelijk moeten blijven wel genoeg is en of niet een nog grotere modulus moet worden genomen. Na de

^{*} Persbericht 26 augustus 1999 van het CWI. Benodigde computertijd ongeveer 3 1/2 maand op 300 PC's plus 11 dagen afrondende bewerkingen op een supercomputer.

[†] De actuele stand van zaken kan men vinden op de website www.keylength.com.

onthullingen van Snowden in 2013 over de af luister capaciteit en cryptanalytische programma's van de Amerikaanse NSA is het wel duidelijk: een RSA-modulus van 1024 is niet meer 'state of the art'. Een uitvoerige uiteenzetting over het bepalen van sleutellengtes kan worden gevonden in een artikel van Lenstra en Verheul uit 2001^[11].

RSA Laboratories heeft een aantal getallen uitstaan als uitdaging. Wie voor het eerst zo'n getal factoriseert kan een prijs in ontvangst nemen. Als voorbeeld hier RSA-640, een getal van 193 decimale cijfers, met als beloning US\$20.000 (tabel 9.2). De getallen en de wedstrijdvoorwaarden zijn op het Internet te vinden. Het hier gegeven voorbeeld is overigens al in 2005 opgelost, dus wie met deze wedstrijd geld wil verdienen moet van goeden huize komen*.

RSA-640	31074	18240	49004	37213	50750	03588	85679	30037	34602	28427
	27545	72016	19488	23206	44051	80815	04556	34682	96717	23286
	78243	79162	72838	03341	54710	73108	50191	95485	29007	33772
	48227	83525	74238	64540	14691	73660	24776	52346	609	

Tabel 9.2 Het getal RSA-640

Net als bij het factoriseren van grote getallen is men bij het bepalen van de discrete logaritme modulo een priemgetal eveneens druk bezig de grenzen te verleggen. Bijvoorbeeld werd op 19 januari 2001 de bepaling van een discrete logaritme van 110 cijfers wereldkundig gemaakt.

Theoretisch werk is verricht om aan te tonen dat het vinden van de geheime sleutel niet eenvoudiger kan dan met factoriseren. Met andere woorden, of er geen achterdeur naar ontcijfering van RSA is. Ander werk is er op gericht om te bewijzen dat het ontcijferen van één bit net zo moeilijk is als de complete ontcijfering. Dat is een resultaat van buitengewoon groot belang, omdat het laat zien dat men RSA niet

* Meer informatie is op het internet te vinden. Bijvoorbeeld via Wikipedia RSA-numbers.

stukje bij beetje kan breken. Het is alles of niets. Ook kan men zich de vraag stellen of d de enige sleutel is waarmee kan worden ontcijferd. Het zijn er wel meer, maar die komen in wezen allemaal op hetzelfde neer. Dat levert de cryptoanalist dus niet werkelijk iets op. Het antwoord op de vraag luidt daarom: nee.

Herhaald vercijferen

Als men een cryptogram maar herhaald blijft vercijferen, komt er onafwendbaar een moment waarop de cyclus is voltooid en de vercijfering opnieuw het cryptogram oplevert. En dus verscheen in de voorlaatste stap de klaartekst. Dat dit zo moet zijn volgt uit het feit dat de verzameling $0 \dots n - 1$ een eindig aantal getallen telt. Hoe vaak er ook vercijferd wordt, nooit kunnen meer dan n verschillende cryptogrammen gemaakt worden en dus zal eens weer de oorspronkelijke klaartekst opduiken. Theoretisch is het aardig bedacht, maar de lengte van de cyclus is dusdanig groot dat de cryptoanalyse aan deze methode niet veel plezier zal beleven.

$$M^e, (M^e)^2, (M^e)^3, \dots (M^e)^{k-1}, (M^e)^k = M \quad (9.29)$$

Toevallig factoriseren

Het kan bij hoge uitzondering voorkomen dat de tweetallige representatie van de klaartekst een van de priemgetallen p en q bevat. Omdat gereduceerd wordt modulo $n = pq$ zal dat priemgetal zich dan in het cryptogram bevinden. Een hierop gebouwde aanval zou als volgt kunnen worden opgezet.

Genereer willekeurige klaartekst en vercijfer deze. Met behulp van de algoritme van Euclides (een behoorlijk efficiënt algoritme) wordt van het paar klaartekst-cijfertekst de grootste gemene deler bepaald. Meestal zal die wel 1 zijn (geen gemeenschappelijke factor) of een kleine gemeenschappelijke factor*.

* De kans dat twee willekeurig gekozen getallen geen factor gemeenschappelijk hebben is tamelijk groot, namelijk $6/\pi^2 \approx 0,60793$ ofwel ruim 60%.

Komt er echter een grote gemeenschappelijke factor tevoorschijn dan is dit misschien wel p of q , te controleren door te kijken of deze een deler is van n . Ook bij veelvuldig herhalen van dit experiment blijkt de kans op succes van deze methode echter verwaarloosbaar klein.

Tot hier zijn de mogelijkheden tot cryptanalyse van RSA niet indrukwekkend, maar nu volgen enkele methoden die wel degelijk gevaarlijk kunnen zijn, tenminste als er niet op de juiste wijze rekening mee wordt gehouden. Ze maken bovendien duidelijk dat het nog niet zo eenvoudig is om RSA goed en veilig te gebruiken. Zoals de geschiedenis herhaaldelijk heeft laten zien, het gebruik van een sterk geheimschrift is wel een noodzakelijke maar niet een voldoende voorwaarde voor een veilig communicatiesysteem. Zo zijn er omstandigheden waarin het gebruik van een kleine exponent (veelal wordt dan de waarde 3 genomen) prettig is. Een voorbeeld vinden we bij de smartcards, die slechts een beperkte hoeveelheid geheugen en een beperkte rekencapaciteit hebben. Het zal echter blijken dat dit in sommige situaties meteen ook de mogelijkheid tot een succesvolle cryptanalyse opent.

Gemeenschappelijke priemfactor

Is het toegestaan om één van de twee priemfactoren herhaald te gebruiken en alleen de andere te variëren? Dat vermindert de behoefte aan priemgetallen.

$$n_i = p \cdot q_i \quad p = \text{constant} \quad i = 1, 2, \dots \quad (9.30)$$

Misplaatste zuinigheid, want een kind kan zien dat dit fout moet gaan. Er wordt dan immers gewerkt met verschillende producten n die eenzelfde p gemeenschappelijk hebben. Die kan dan direct met Euclides voor de grootste gemene deler worden bepaald. Daarna vallen de andere priemfactoren de cryptanalist als rijpe appelen in de schoot. Het is dus van het grootste belang om steeds andere priemgetallen te nemen.

Een probleem is dat dit niet altijd kan worden voorkomen. Immers, op allerlei plaatsen in de wereld worden priemfactoren gemaakt om met RSA te kunnen vercijferen en in veel gevallen zal men daarvoor dezelfde

programmatuur gebruiken*. Het is dan onvermijdelijk dat af en toe eenzelfde priemfactor opduikt of zelfs hetzelfde product van priemgetallen wordt gevonden. Een recent onderzoek bekeek ruim 6 miljoen RSA-sleutels en vond in meer dan 12.000 sleutels een gemeenschappelijke factor. Bovendien kwamen de onderzoekers ruim 70.000 duplicaten tegen, die onafhankelijk van elkaar waren gemaakt. Al met al voor de gebruikers ervan een behoorlijk risico^[12]. Het lijkt geen twijfel of geheime diensten hebben deze achterdeur in hun af luisterprogramma's opgenomen.

Gemeenschappelijke modulus

De verleiding is groot om eenmalig een n te kiezen en vervolgens daarbij verschillende combinaties van de exponenten e en d uit te geven. Dat heeft het voordeel dat men slechts één keer de priemgetallen hoeft te bepalen en daaraan veel paren e en d kan koppelen. De berekening van zo'n paar berust op Euclides en is eenvoudiger dan het vinden van goede priemgetallen. Helaas, het blijkt betrekkelijk eenvoudig om n te factoriseren als men één paar e en d gegeven krijgt. Daarna is het een koud kunstje om voor alle openbare exponenten e de bijbehorende ontcijferexponent d uit te rekenen.

Identieke berichten

Veronderstel dat eenzelfde bericht naar verschillende ontvangers wordt gestuurd. Elke ontvanger heeft een eigen sleutel n , maar allemaal gebruiken ze dezelfde vercijferexponent e . We mogen aannemen dat alle paren p en q voor die n 'en verschillend zijn, want anders is zelfs een nog eenvoudiger aanval mogelijk. Daar alle berichten met dezelfde e zijn vercijferd geldt

$$C_1 = M^e \bmod n_1 \quad C_2 = M^e \bmod n_2 \quad C_3 = M^e \bmod n_3 \quad \dots \quad (9.31)$$

* Een veelvoorkomende oorzaak is het herstarten van de voor het maken van priemgetallen gebruikte randomgenerator op dezelfde beginwaarde; bijvoorbeeld bij het opstarten van een virtuele machine.

Met de zogenaamde Chinese reststelling berekent men uit $C_1, C_2, C_3 \dots$ een C' element van $\mathbb{Z}_{n_1 n_2 n_3 \dots}$. Is nu het vercijferde blok M kleiner dan elk der n_i dan geldt voor kleine e (zoals de waarde 3)

$$C' = M^e \quad \text{in plaats van} \quad C' = M^e \bmod (n_1 n_2 n_3 \dots) \quad (9.32)$$

De vergelijking links speelt zich af in het rijk van de gehele getallen $\mathbb{Z}$ in plaats van in $\mathbb{Z}_n$ en is daarom eenvoudiger op te lossen. Aldus wordt kan de inhoud M van het bericht worden gevonden.

Omdat deze aanvalsmethode slaagt als het aantal berichten groter is dan e , vormt ze een sterk argument tegen het gebruik van het lage getal 3 als vercijferexponent. Verder kan nog worden aangetoond dat tenminste het telkens *verschillend* invullen van elk van de klaartekstblokken noodzakelijk is om deze aanval te verhinderen en dan nog kan dit onvoldoende zijn.

Lage openbare exponent

Een bijzonder gevaarlijke aanval op de vercijfering met een lage openbare exponent zoals de al meer genoemde 3, is het gevolg van een door Coppersmith bedacht algoritme^[13]. Het blijkt dat exponent $e = 3$ kwetsbaar is wanneer men $2/3$ van het berichtblok kent of wanneer men beschikt over twee berichten die voor $8/9$ overeenkomen. Het eerste geval kan zich gemakkelijk voordoen als men een 128 bits AES-sleutel voor verzending vercijfert met een 1024 bits RSA-modulus en de resterende 896 bits standaard opvult met nullen. De algoritme van Coppersmith vindt oplossingen van vergelijkingen waarmee het bericht kan worden gebroken.

Lage geheime exponent

Het gebruik van een kleine geheime exponent is evenmin veilig. Aangenomen dat de twee priemgetallen p en q zoals gebruikelijk niet te ver uiteen liggen, is cryptanalyse mogelijk wanneer de ontcijferexponent kleiner is dan een bepaalde van n afhankende waarde. Vanwege deze aanvalsmethode moet de geheime

exponent in het geval van 1024 bits RSA tenminste 256 bits tellen. De details voeren hier te ver, maar het is aardig om te vermelden dat er zogenaamde kettingbreuken aan te pas komen. Dat is een techniek die al in de Gouden Eeuw door Christiaan Huygens werd gebruikt om voor klokken de overbrenging tussen de verschillende tandraderen te berekenen.

In het geval p en q relatief dicht bij elkaar liggen, kan een aanval succes hebben wanneer de ontcijferexponent $d < n^{0.292}$ is, en naar wordt vermoed zelfs voor $d < n^{0.5}$. Deze aanval berust op het herschrijven van $ed \equiv 1 \pmod{\phi(n)}$ tot

$$\left| \frac{e}{\phi(n)} - \frac{k}{d} \right| = \frac{1}{d \cdot \phi(n)} \text{ met } k = 0, 1, 2, \dots \quad (9.33)$$

Daar $1/d\phi(n)$ klein is, kan men concluderen dat $k/d \approx e/\phi(n)$. Gecombineerd met nog een paar zaken (zoals de aanname dat $\phi(n)$ mag worden benaderd met n en de veronderstelling dat p en q ongeveer gelijke bitlengte hebben) blijken tenslotte hoogstens $\log_2 n$ stappen in de kettingbreukontwikkeling van e/n voldoende om k/d te vinden en daaruit weer d . Een nadere analyse toont een verband aan tussen het verschil $\Delta = |p - q|$ en de minimaal veilige d . Een Δ aanzienlijk kleiner dan $\sqrt{n}$ maakt, in combinatie met een te kleine d , een aanval als hierboven beschreven bijzonder effectief^[14].

Zwakke priemgetallen

Het gebruik van zwakke priemgetallen (zie op [sectie 9.2](#)) moet eveneens worden afgeraden, geïllustreerd bijvoorbeeld in een artikel getiteld *Factoring RSA moduli with weak prime factors*^[15].

Side-Channel-aanval

Evenmin als moderne systemen zijn ook de implementaties van systemen met publieke sleutel niet immuun voor *side-channel-analyse*. Recent verscheen hierover een artikel van Shamir e.a.^[16] Uit de inleiding:

"Many computers emit a high-pitched noise during operation, due to vibration in some of their electronic components. These acoustic emanations are more than a nuisance: they can convey information about the software running on the computer, and in particular leak sensitive information about security-related computations." Gerapporteerd wordt dat voor de GnuPG-implementatie van RSA bij decryptie op een laptop van bepaalde gekozen cijfertekst binnen het uur een 4096-bit RSA-sleutel kan worden gevonden. Een gevoelige microfoon is in staat de benodigde signalen tot op vier meter afstand op te vangen.

9.5 ElGamal systeem

Een nadeel van RSA is het feit dat voor encryptie slechts één enkele publieke sleutel ter beschikking staat. Wanneer hetzelfde bericht herhaaldelijk vercijferd wordt, ontstaat daardoor steeds hetzelfde cryptogram. Dat is niet onwaarschijnlijk wanneer veel berichten een stereotype inhoud hebben*. Is eenmaal de link gelegd tussen een bericht en de vercijfering ervan, dan is een blik op het cryptogram voldoende om te constateren of dat specifieke bericht verzonden is. Het ElGamal-systeem vermijdt dit nadeel al gaat dit ten koste van de compactheid van het bericht, want een cryptogram met ElGamal is wat langer dan de oorspronkelijke klaartekst^[17].

De ElGamal methode berust evenals de Diffie-Hellman sleuteluitwisseling op de moeilijkheid van de discrete logaritme. Neem een eindig veld $\mathbb{Z}_p$ met p een priemgetal en kies een primitief element a uit $\mathbb{Z}_p$. Kies een s volgens onderstaande vergelijking en bereken daaruit b door herhaald vermenigvuldigen.

$$0 \leq s < p - 1: a^s \equiv b \pmod{p} \quad \text{ofwel} \quad s = \log_a b \quad (9.34)$$

De probleemstelling waarop de moeilijkheid van cryptoanalyse berust is het vinden van een uniek geheel getal s uit b . Deze exponent s fungeert namelijk als de geheime sleutel. De keuze van het priemgetal p

* Het is zelfs niet nodig dat de berichten volledig overeenstemmen. Omdat in blokken wordt vercijferd is het herhaaldelijk voorkomen van een stereotyp gedeelte op eenzelfde plaats genoeg om de overeenkomst te signaleren.

moet met zorg geschieden, met name om de effectiviteit van bekende algoritmen voor de berekening van de discrete logaritme te frustreren.

Noem C de vercijfering van de klaartekst M en kies voor de vercijfering als extra parameter een willekeurig geheel getal k . Deze laatste keuze kan per vercijfering verschillen en maakt het zodoende mogelijk om eenzelfde bericht op verschillende manieren te vercijferen. Het systeem moet er nu voor zorgen dat de ontvanger impliciet k krijgt toegezonden, zodanig dat een af luisteraar deze k niet kan vinden.

Informeel kunnen we ElGamal als volgt beschrijven. De klaartekst M wordt geblindeerd (gemaskeerd) door deze te vermenigvuldigen met b^k . Om k bij de ontvanger te kunnen gebruiken wordt als extra gegeven de waarde van a^k meegezonden. Men ziet dat het vercijferde bericht hierdoor langer wordt, terwijl bovendien een extra machtsverheffing moet worden gedaan. De ontvanger van het bericht kan ontcijferen dankzij diens kennis van de geheime exponent s , omdat hiermee b^k kan worden berekend en daarmee vervolgens de blinding verwijderd.

De openbare ElGamal-systeemsleutel bestaat uit

- een priemgetal p dat de verzameling getallen $\mathbb{Z}_p$ vastlegt
- een primitief element a uit $\mathbb{Z}_p$ als basis voor de machtsverheffing
- het getal $b = a^s \bmod p$, berekend uit a en de geheime exponent s

De geheime ElGamal-deelnemersleutel bestaat uit

- de geheime exponent s

De geheime ElGamal-berichtsleutel bestaat uit

- de per bericht willekeurig te kiezen k met $0 \leq k < p - 1$

De klaartekst wordt zodanig in blokken M verdeeld dat $0 < M < p$. Het vercijferen gaat als volgt

$$C = (c_1, c_2) \text{ met } c_1 = M \cdot b^k \bmod p \text{ en } c_2 = a^k \bmod p \quad (9.35)$$

Voor het ontcijferen berekent de ontvanger van het bericht

$$M = c_1 c_2^{-s} = (M \cdot b^k)(a^k)^{-s} = (M \cdot b^k)(a^s)^{-k} = M \cdot b^k b^{-k} \bmod p = M \quad (9.36)$$

Direct valt in te zien dat expliciete kennis van k niet nodig is om tot ontcijfering te kunnen overgaan.

Het is overigens verstandig om voor elk bericht een andere waarde k te nemen. Doet men dit niet en heeft iemand van een bericht de klaartekst weten te bemachtigen, dan kan deze ook de andere berichten lezen. Dat gaat als volgt, waarbij we ons concentreren op de delen c_1 waarin de klaartekst M zit. Stel we beschikken over twee berichten met respectievelijk $c_1 = M \cdot b^k$ en $c'_1 = M' \cdot b^k$ en veronderstellen dat de M van het eerste bericht bekend is. Nu wordt de inhoud M' van het tweede bericht gevonden uit

$$M \cdot c_1^{-1} c'_1 = M \cdot (M^{-1} b^{-k})(M' b^k) = M \cdot M^{-1} \cdot M' = M' \quad (9.37)$$

Voorbeeld ElGamal vercijfering

Een getallenvoorbeeld om desgewenst na te kunnen rekenen. Voor het priemgetal wordt genomen $p = 2579$, voor de basis $a = 2$ en voor de geheime sleutel $s = 765$. De openbare b wordt zodoende $b = 2^{765} \bmod 2579 = 949$.

Kies als klaartekst $x = 1299$ en als berichtssleutel $k = 853$. Nu luidt het cryptogram

$$y = (y_1, y_2) = (1299 \cdot 949^{853} \bmod 2579, 2^{853} \bmod 2579) = (2396, 435)$$

Om te ontcijferen luidt de berekening

$$x = 2396 \cdot 435^{-765} \bmod 2579 = 1299$$

9.6 Elliptische krommen

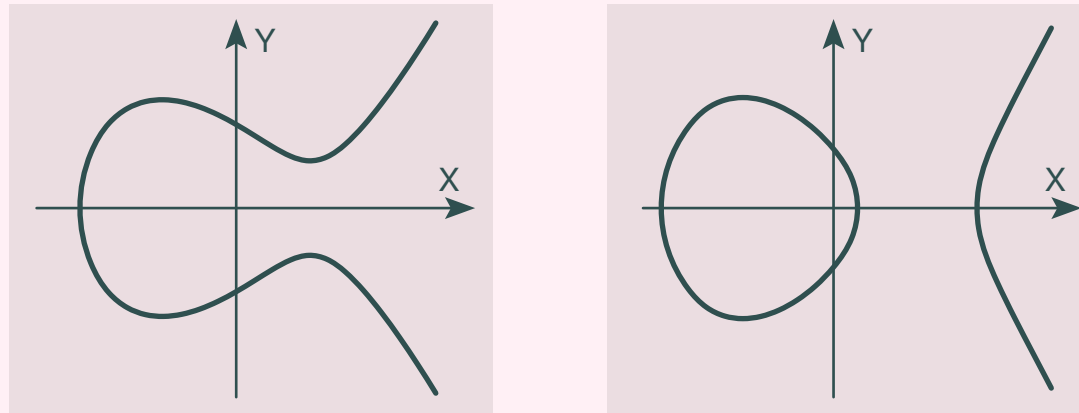
De toenemende rekenkracht van computers en de steeds grotere effectiviteit van methoden om producten in factoren te ontbinden, dwingen tot het gebruik van steeds grotere priemgetallen. Als gevolg daarvan

neemt het benodigde rekenwerk voor het vercijferen en ontcijferen toe, terwijl de langere parameters ook meer geheugen vergen. Publieke sleutelsystemen verkeren wat dit betreft toch al in het nadeel ten opzichte van symmetrische systemen en dat verschil neemt hierdoor toe.

In antwoord op deze ontwikkeling is men op zoek gegaan naar methoden waarin met minder grote getallen toch een vergelijkbare weerstand tegen cryptoanalyse wordt verkregen. Dat kan door een andere, meer ingewikkelde bewerking als basisoperatie te nemen. In plaats van machtsverheffen (een herhaalde vermenigvuldiging) kiest men een andere operatie die kan worden herhaald. Als door overgaan op deze operatie de cryptoanalyse sterk wordt bemoeilijkt, terwijl de berekening er relatief minder last van heeft dan kan met minder grote getallen worden volstaan. Dat wil zeggen, men kan met kleinere getallen werken en toch dezelfde weerstand tegen cryptoanalyse bereiken. Wanneer nu de vervangende operatie in een algoritme als RSA zonder meer voor de standaardvermenigvuldiging in de plaats kan komen, verandert het systeem zelf niet.

Het meest populaire alternatief wordt geboden door de zogenaamde *elliptische krommen* en wordt Elliptic Curve Cryptography (ECC) genoemd. Deze methode is inmiddels zo ingeburgerd dat ze een plaats heeft gekregen in de standaard voor digitale handtekeningen, de Digital Signature Standard (DSS), van het Amerikaanse National Institute of Standards and Technology (NIST)^[18].

In het begin van dit hoofdstuk is een verzameling getallen geïntroduceerd, die werd afgeleid van een groot priemgetal en die voor systemen met publieke sleutel een geschikt rekensysteem oplevert. Echter elk ander systeem met de juiste algebraïsche eigenschappen is in principe geschikt om mee te werken. In ECC nu, wordt gerekend met punten op een bepaalde vlakke kromme. Het systeem ontleent zijn naam aan de gebruikte functies, namelijk de familie van de elliptische krommen. Zoals de vanouds bekende parabool wordt bepaald door de vergelijking $y = ax^2 + bx + c$, zo wordt een elliptische kromme bepaald door de vergelijking $y^2 = x^3 + ax + b$. **Figuur 9.2** geeft twee voorbeelden van zulke elliptische krommen. Afhankelijk van de waarde van de parameters vormt de curve een geheel (figuur links) of bestaat uit twee afzonderlijke delen (figuur rechts).

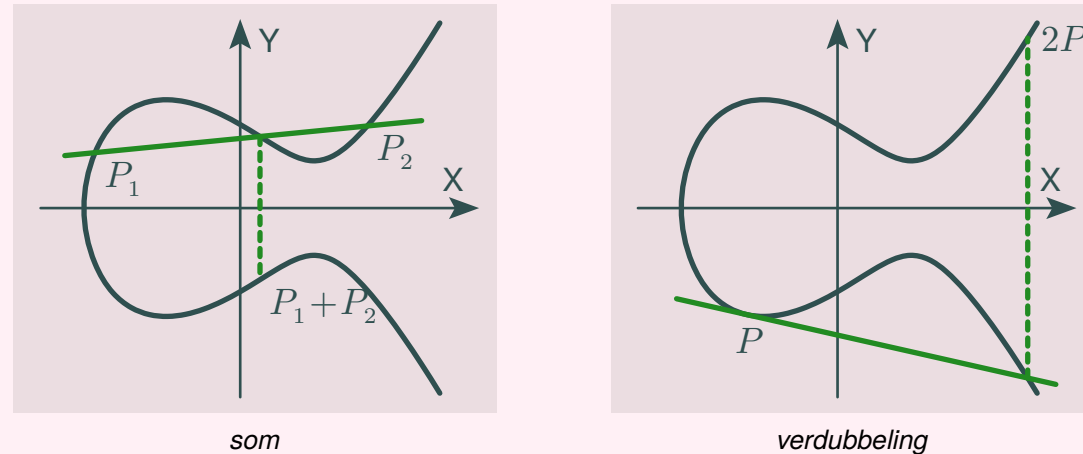


Figuur 9.2 Twee voorbeelden van een elliptische kromme

Het ECC-rekensysteem heeft als basisoperatie het optellen van punten: de som van twee punten op de kromme is weer een punt op de kromme. Anders gezegd, de gewone vermenigvuldiging als basisoperatie wordt vervangen door een optelling c.q. verdubbeling van punten op de kromme. In [figuur 9.3](#) wordt dat toegelicht. Links de optelling van twee verschillende punten $P_1 + P_2$. Men trekt de rechte door beide punten en bepaalt het snijpunt met de kromme. Uit dit snijpunt wordt een loodlijn op de X-as neergelaten en waar deze lijn de kromme opnieuw snijdt is het sompunt. Als de som $P + P = 2P$ moet worden berekend, dan trekt men de raaklijn aan P en gaat op dezelfde wijze verder (figuur rechts). In plaats van grafisch zoals in de figuur, kan men deze berekeningen ook analytisch uitvoeren door het oplossen van de betreffende vergelijkingen. Om de rekenregels te completeren moet nog een 0 worden toegevoegd. Daarvoor dient hier het punt in het oneindige, aangeduid met het symbool O .

Oplossing van de vergelijkingen heeft het volgende resultaat*.

* Met uitzondering van de gevallen waarin beide punten dezelfde x-coördinaat hebben of een van beide het nulpunt is.



Figuur 9.3 Optellen van punten op de kromme

$$P_s = (x_s, y_s) = (x_1, y_1) + (x_2, y_2) \quad \text{met} \quad \begin{cases} x_s &= \lambda^2 - x_1 - x_2 \\ y_s &= \lambda(x_1 - x_s) - y_1 \end{cases} \quad (9.38)$$

Voor de berekening van de hulpvariabele λ moet men twee gevallen onderscheiden

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{als } P_1 \neq P_2 \\ \frac{3x_1^2 + a}{2y_1} & \text{als } P_1 = P_2 \end{cases} \quad (9.39)$$

Tot zover zijn het reële variabelen x en y . Dat is ongelukkig, want in computers kunnen reële variabelen maar met beperkte precisie worden gehanteerd. De oplossing is om niet alle punten op de curve mee te nemen, maar zich te beperken tot een eindige deelverzameling ervan. Deze overgang op een discrete verzameling punten geschiedt door het maken van een afbeelding tussen de getallen in $\text{GF}(p^n)$ en de punten op de elliptische kromme. In zwang zijn twee mogelijkheden, namelijk middels $\text{GF}(p)$ of $\text{GF}(2^n)^*$. In het geval van $\text{GF}(p)$ worden de gekozen punten op de kromme gegeven door

* Vergelijk voor deze laatste mogelijkheid de beschrijving van de rekenoperaties in $\text{GF}(2^8)$ bij Rijndael [hoofdstuk 8](#).

$$y^2 \equiv x^3 + ax + b \pmod{p} \quad \text{met} \quad 4a^3 + 27b^2 \not\equiv 0 \pmod{p} \quad (9.40)$$

Behalve de parameters van de kromme heeft men nog een beginpunt (aftelpunt) nodig. In principe kan dit beginpunt willekeurig worden gekozen uit de puntenverzameling. Vergelijk het met de rij gehele getallen 1, 2, ... waarvan 1 het beginpunt vormt. Door bij het vercijferen verschillende punten als beginpunt te kiezen kan het systeem worden gevarieerd.

Van belang is voorts het aantal punten op de curve. Men kan er niet zonder meer van uitgaan dat alle getallen $p - 1$ één op één zullen corresponderen met overeenkomstige punten op de elliptische kromme. **Figuur 9.2** laat al zien dat niet voor alle waarden van x een oplossing van $y^2 = x^3 + ax + b$ bestaat. Is er een oplossing dan zijn dat op enkele uitzonderingen na steeds twee punten, zoals uit de figuur blijkt.

Als voorbeeld de kromme $y^2 = x^3 + 2x + 4$ met $p = 11$ ^[19]. Deze curve telt de volgende 17 punten:

$P_1 = (0, 2)$	$P_2 = (3, 2)$	$P_3 = (8, 9)$	$P_4 = (6, 1)$	$P_5 = (9, 5)$
$P_6 = (7, 3)$	$P_7 = (2, 4)$	$P_8 = (10, 10)$	$P_9 = (10, 1)$	$P_{10} = (2, 7)$
$P_{11} = (7, 8)$	$P_{12} = (9, 6)$	$P_{13} = (6, 10)$	$P_{14} = (8, 2)$	$P_{15} = (3, 9)$
$P_{16} = (0, 9)$	$P_{17} = O$			

Figuur 9.4 geeft de punten op de kromme. Merk op dat die punten de echte waarden van y volgend uit de vergelijking $y^2 = x^3 + 2x + 4$ hebben. In de lijst zijn die waarden gereduceerd modulo 11. Hieronder drie rekenvoorbeelden voor de som van twee punten en de verdubbeling van een punt.

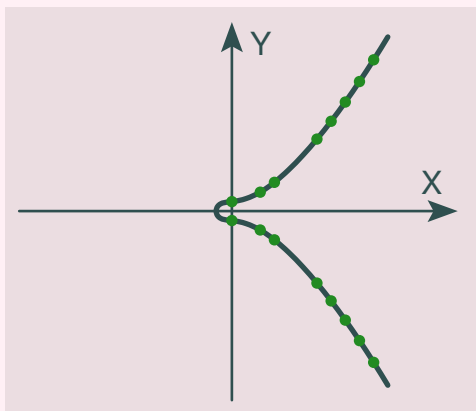
- $P_1(0,2) + P_2(3,2)$ moet $P_3 = (8,9)$ worden.

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} = \frac{2 - 2}{3 - 0} = 0$$

$$x_3 = \lambda^2 - x_1 - x_2 = 0^2 - 0 - 3 = -3 \pmod{11} = 8$$

$$y_3 = \lambda(x_1 - x_3) - y_1 = 0(0 - 8) - 2 = -2 \pmod{11} = 9$$

$$P_3 = (8, 9)$$



Figuur 9.4 Punten op de kromme $y^2 = x^3 + 2x + 4$

- $2P_1(0,2)$ moet $P_2 = (3,2)$ worden.

$$\lambda = \frac{3x_1^2 + a}{2y_1} = \frac{3 \cdot 0^2 + 2}{2 \cdot 2} = \frac{1}{2} = \frac{12}{2} \bmod 11 = 6$$

$$x_2 = \lambda^2 - x_1 - x_1 = 6^2 - 0 - 0 = 36 \bmod 11 = 3$$

$$y_2 = \lambda(x_1 - x_2) - y_1 = 6(0 - 3) - 2 = -20 \bmod 11 = 2$$

$$P_2 = (3, 2)$$

- $P_2(3,2) + P_{15}(3,9)$ moet $P_{17} = O$ worden.

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} = \frac{9 - 2}{3 - 3} = \frac{7}{0} = \infty$$

$$P_{17} = (\infty, \infty) = O$$

Het precies tellen van het aantal punten op de kromme is nog niet zo eenvoudig. Voor kleine oneven p kan men gebruik maken van de volgende formule

$$\# = p + 1 + \sum_{x=0}^{p-1} \left(\frac{x^3 + ax + b}{p} \right) \quad (9.41)$$

Hierin staat de term tussen haakjes voor het Legendre-symbool. De waarden van p voor cryptografische toepassingen zijn echter te groot voor deze formule. Een schatting van het aantal punten kan worden gemaakt met behulp van de volgende ongelijkheid (stelling van Hasse)*

$$p + 1 - 2\sqrt{p} \leq \# \leq p + 1 + 2\sqrt{p} \quad (9.42)$$

Een efficiënte algoritme voor het precies berekenen van het aantal punten op de kromme bestaat ook en heet Schoof's algoritme; op de details ervan wordt hier niet ingegaan^[20].

Het blijkt dat 173 bits ECC ruwweg overeenkomt met 1024 bits conventioneel en 313 bits ECC met 4096 bits conventioneel. In vergelijking met het conventionele rekenwerk groeit de effectieve sleutellengte in ECC bij benadering met de derde macht van het aantal bits. Overigens moet, voordat men efficiënt kan gaan rekenen met dit systeem, nog wel het een en ander aan werk worden verzet. Opgemerkt moet nog worden dat de keuze van de parameters zorgvuldig moet geschieden. Net als het onderscheid tussen sterke en zwakke priemgetallen van belang is, zo zijn er ook gradaties in sterkte onder de elliptische krommen.

Het Amerikaanse NIST heeft een aantal elliptische krommen uitgekozen voor gebruik in de standaard voor digitale handtekeningen. Al deze krommen hebben de vorm $y^2 = x^3 - 3x + b$. Voor de beide soorten afbeelding $GF(p)$ en $GF(2^n)$ zijn er vijf varianten met verschillende bitlengte. Deze staan in [tabel 9.3](#) tezamen met hiermee in sterkte overeenkomende symmetrisch cryptosystemen. Voor elke curve wordt in de standaard tevens een geschikt beginpunt opgegeven.

* Dat het aantal punten bij $GF(p)$ gemiddeld op $p + 1$ uitkomt valt vrij eenvoudig te beredeneren. Bij ongeveer de helft van de punten heeft x een oplossing en dat zijn er dan meteen ook twee, de positieve en de negatieve wortel (behalve voor maximaal drie punten, namelijk die met $y = 0$). Men verwacht dus $2(p/2)$ oplossingen, samen met het nulpunt totaal gemiddeld $p + 1$.

GF(p)	GF(2 ⁿ)	equivalent met
$p = 192$ bits	$n = 163$ bits	80 bits Skipjack
$p = 224$ bits	$n = 233$ bits	112 bits TripleDES
$p = 256$ bits	$n = 283$ bits	128 bits AES
$p = 384$ bits	$n = 409$ bits	192 bits AES
$p = 521$ bits	$n = 571$ bits	256 bits AES

Tabel 9.3 ECC bitlengtes in de NIST standaard

De ECC-krommen kunnen voor wat betreft hun sterkte ook worden vergeleken met de conventionele public-key-systemen in GF(p). In de literatuur kan men voorbeelden vinden voor respectievelijk het rekenen in GF(p) en in GF(2ⁿ)^[21]. Interessant is dat de National Security Agency (NSA), in een statement uitgegeven in 2015, voor overheidsgebruik wel de curve met $p = 384$ uit tabel 9.3 toestaat maar niet (meer) die met $p = 256$. Eveneens interessant is dat bij het gebruik van RSA een modulus van 3072 bits aanbevolen wordt*.

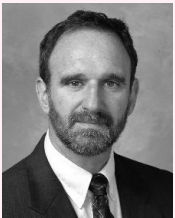
Net als voor RSA is ook voor de elliptische krommen een aantal uitdagingen gepubliceerd, in dit geval in 1997 door het bedrijf Certicom, <http://www.certicom.com/chal/>. Van genoemde uitdagingen werd ECC p-97, een curve met $p = 97$ bits, in 1999 opgelost in 53 dagen onder gebruikmaking van zo'n 1200 computers^[22]. In 2002 volgde een 109 bit exemplaar en in 2009 een van 112 bits. De progressie is hier beduidend minder sterk dan bij het factoriseren (zie tabel 9.1).

9.7 Knapzakstelsel

De *knapzak* werd in 1978 geïntroduceerd door Merkle en Hellman^[23]. Het zag er aanvankelijk veelbelovend



Merkle



Hellman

* Committee on National Security Systems, Use of public standards for the secure sharing of information among national security systems, Advisory Memorandum 02-15, July 2015.

uit. Temeer omdat dit systeem goede mogelijkheden leek te bieden voor een snelle implementatie in hardware en dus een flinke encryptiebandbreedte beloofde. Dat laatste is het zwakke punt van algoritmen als RSA. De snelheid waarmee verscijferen en ontcijferen in public-key geheimschriften kan worden uitgevoerd blijft sterk achter bij de snelheden die met blokgeheimschriften als des mogelijk bleken.

Merkle en Hellman loofden een beloning van \$ 100 uit voor de eerste persoon die hun systeem zou weten te kraken. Deze beloning kon door Adi Shamir in 1982 worden opgeëist op de Crypto-conferentie in Santa Barbara, Californië. Nadien heeft men nog wel geprobeerd om het systeem door een aantal verfijningen te redden, maar deze hebben alle schipbreuk geleden en het knapzakstelsel geldt nu als onveilig. Desondanks is behandeling ervan nuttig, vooral omdat dit duidelijk maakt waarom een public-key cryptosysteem moet worden gebaseerd op een werkelijk hard wiskundig probleem en niet op een probleem dat alleen maar schijnbaar moeilijk is.

Het knapzakstelsel is gebaseerd op het gelijknamige NP-complete knapzak-probleem uit de complexiteitstheorie. In een notendop beschreven is er een vector $\vec{a}$ met componenten $(a_1, a_2 \dots a_n)$ van positieve getallen, die wordt gecombineerd met een binaire oplosvector $\vec{x} = (x_1, x_2 \dots x_n)$ met x_i uit $\{0, 1\}$. De som S wordt gevormd uit die componenten a_i waarvoor de overeenkomstige coëfficiënt x_i de waarde 1 heeft.

$$S = \vec{x} \cdot \vec{a} = \sum_{i=1}^n x_i a_i \quad (9.43)$$

Gemakkelijk valt in te zien hoe hiermee een bitvector kan worden gecodeerd. De knapzakssom S vormt het cryptogram en is een blokverscijfering waarvan de bloklengthe wordt bepaald door het aantal componenten van $\vec{a}$ en een daarmee verband houdende nog te introduceren parameter. Er zit echter een addertje onder gras, want decoderen blijkt niet zo eenvoudig. De essentie van het knapzakprobleem is het complexiteitskarakter NP. Hetgeen betekent dat oplossen ervan – het vinden van de coëfficiëntenvector $\vec{x}$ met slechts S en $\vec{a}$ gegeven – een exponentiële inspanning vergt. Begrijpelijk, omdat voor het systematisch uitproberen van alle bitcombinaties van $\vec{x}$ (de deterministische algoritme) in totaal 2^n maal de som $S =$

$\vec{x} \cdot \vec{a}$ moet worden uitgerekend; dit geeft overigens wel gegarandeerd de oplossing. Het duale karakter van NP is daarin gelegen dat de probabilistische oplosmethode (het gissen van een oplosvector $\vec{x}$) wel in polynomiale tijd tot een conclusie aangaande die gissing voert. Dan echter is het vinden van de oplossing niet meer gegarandeerd: men kan pech of geluk hebben.

Om uit deze klem te komen en een snelle, gegarandeerde ontcijfering te bewerkstelligen, namen Merkle en Hellman geen algemene knapzakvector, maar een die aan bijzondere eisen voldoet. Zij namen een knapzakvector waarvan de componenten een *superstijgende rij* vormen.

$$\vec{a} = (a_1, a_2, \dots, a_n) \text{ met } \forall i : a_i > \sum_{j=1}^{i-1} a_j \quad (9.44)$$

Als gevolg van deze bijzondere keuze is het oplossen van de knapzak nu kinderspel geworden. Men vergelijkt S met het grootste element a_n en als $S \geq a_n$ dan moet a_n in de som zitten, want alle voorgaande $a_1 \dots a_{n-1}$ tezamen kunnen wegens *vergelijking (9.44)* niet tot a_n reiken. Aldus kan x_n eenduidig worden bepaald. Men trekt a_n van S af en het spel herhaalt zich met a_{n-1} .

Nu kan weliswaar worden vercijferd en ontcijferd, maar het vervelende is dat iedereen de *openbare knapzakvector* $\vec{a}$ kan gebruiken voor ontcijfering. De geheimhouding is weg en technisch gesproken is het geen cryptosysteem meer. De truc die Merkle en Hellman hierop bedachten is de volgende: verberg het superstijgende karakter van de knapzakvector zodat het een gewone knapzakvector lijkt. Vercijfer met de vermomde knapzakvector $\vec{b}$ en maak ontcijfering mogelijk door een bewerking op S die de vermomming teniet doet. Deze vermomming geschiedt met behulp van twee gekozen getallen w en m , relatief priem, waarmee $\vec{b}$ wordt berekend uit $\vec{a}$

$$m > \sum_{j=1}^n a_j \quad \text{ggd}(w, m) = 1 \quad b_i = w \cdot a_i \bmod m \quad (9.45)$$

Om de zaak extra te compliceren, worden de componenten van $\vec{b}$ doorelkaar gehusseld tot permutatie $\pi(\vec{b})$. De bloklength van het cryptogram moet zodanig zijn dat de waarde van m niet wordt overschreden.

De openbare knapzaksleutel bestaat uit

- $\pi(\vec{b}) = (b_1, b_2 \dots b_n)$
- b_i berekend met $b_i = w \cdot a_i \bmod m$

De geheime knapzaksleutel bestaat uit

- $\vec{a} = (a_1, a_2 \dots a_n)$
- π, w, m met $\text{ggd}(w, m) = 1$

Nu wordt aangetoond waarom ontcijferen van een met $\vec{b}$ tot $S = \vec{x} \cdot \vec{b}$ vercijferde klaartekst mogelijk is. Door vermenigvuldiging van S met inverse w^{-1} wordt $\vec{x} \cdot \vec{b}$ gereduceerd tot $\vec{x} \cdot \vec{a}$

$$\begin{aligned}
 w^{-1}S &= w^{-1}(\vec{x} \cdot \vec{b}) \bmod m = w^{-1} \sum_{i=1}^n x_i b_i \bmod m \\
 &= w^{-1} \sum_{i=1}^n x_i (w \cdot a_i \bmod m) \bmod m \\
 &= w^{-1}w \sum_{i=1}^n x_i a_i \bmod m = \sum_{i=1}^n x_i a_i \bmod m = \vec{x} \cdot \vec{a}
 \end{aligned} \tag{9.46}$$

Door inverteren van de eveneens geheime permutatie π worden tenslotte de coëfficiënten x_i in de oorspronkelijke volgorde teruggeplaatst.

9.7.1 Voorbeeld knapzak-vercijfering

Een superstijgende rij van lengte acht wordt gekozen om een byte te vercijferen

		a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
a	=	3	5	12	22	57	102	240	499
$\sum_1^i a_i$	=	3	8	20	42	99	201	441	940

Kies $m = 1000$ (groter dan 940) en $w = 231$, w en m zijn relatief priem. Daarna wordt met de algoritme van Euclides w^{-1} bepaald op 671. De parametrisering wordt afgerond met het opstellen van de openbare vector $\pi(b)$

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
$a =$	3	5	12	22	57	102	240	499
$b = 231 a_i \bmod 1000 =$	693	155	772	82	167	562	440	269
$\pi(b) =$	82	155	167	269	440	562	693	772

De permutatie van de elementen van de openbare vector bepaalt in welke volgorde de bits bij de vercijfering worden meegenomen, voor de ontcijfering moeten ze dan in de omgekeerde volgorde worden geplaatst

$$\pi = (4, 2, 5, 8, 7, 6, 1, 3)$$

De klaartekst bitrij $\vec{x} = (01100110)$ wordt vercijferd met

$$0 \cdot 82 + 1 \cdot 155 + 1 \cdot 167 + 0 \cdot 269 + 0 \cdot 440 + 1 \cdot 562 + 1 \cdot 693 + 0 \cdot 772 \bmod 1000$$

Het cryptogram is het getal 577. Dit zojuist verkregen cryptogram 577 wordt nu weer ontcijferd. Vermenigvuldiging met $w^{-1} = 671$ verwijdert de vermomming

$$w^{-1}S = 671 \cdot 577 \bmod 1000 = 167$$

Dan worden de coëfficiënten van hoog naar laag bepaald door reductie van de knapzaksom met $a_n \dots a_1$. In [tabel 9.4](#) is deze reductie stap voor stap weergegeven.

Dit is natuurlijk maar een speelgoedvoorbeeld met slechts een zeer gering aantal componenten in de knapzakvector. De makers van het systeem voorzagen een knapzakvector van een honderdtal elementen met een beginwaarde in de orde van 2^{100} , per element in de superstijgende rij een factor twee groter wordend, waardoor tenslotte m een getal in de orde van 2^{200} zal zijn.

bit	S	a_i	x_i	$S - x_i \cdot a_i$
8	167	499	0	167
7	167	240	0	167
6	167	102	1	65
5	65	57	1	8
4	8	22	0	8
3	8	12	0	8
2	8	5	1	3
1	3	3	1	0
4, 2, 5, 8, 7, 6, 1, 3 → 01100110				

Tabel 9.4 Reductie van de knapzaksom

9.7.2 MASTERSECTIE – Cryptoanalyse knapzak

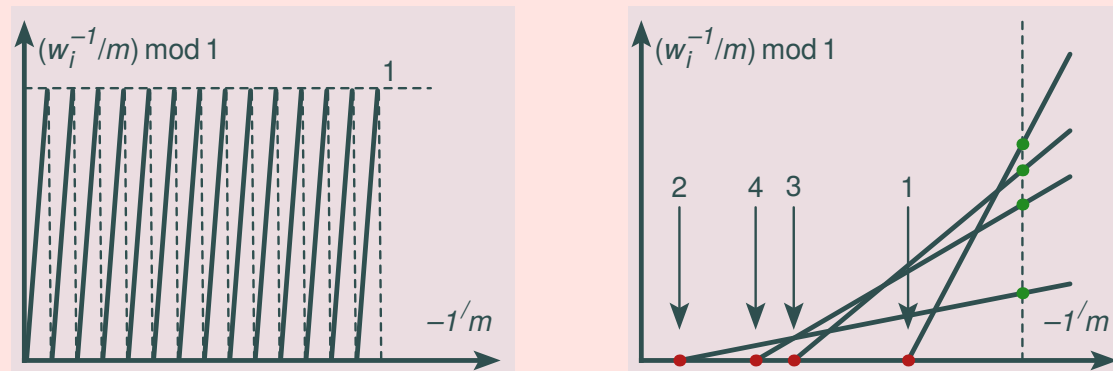
Het breken van de knapzak door Shamir in 1982 wordt slechts schetsmatig behandeld^[24]. In de eerste, cruciale stap van de cryptoanalyse wordt de vermomming van de openbare knapzakvector doorzien.

De achilleshiel van de knapzak is het feit dat het onderliggende probleem niet werkelijk het NP-complete knapzakprobleem is. Door de keuze voor een superstijgende rij is een wezenlijk eenvoudiger knapzak gebruikt. De vermomming kan dat niet werkelijk veranderen. En dus vroeg Shamir zich af of er niet een manier was om een paar getallen (w, m) te vinden waarmee de reductie tot de superstijgende rij zou kunnen worden bewerkstelligd. Het bleek zelfs niet nodig om precies dezelfde (w, m) van de vercijfering te vinden, want er kunnen meer getallenparen (w, m) zijn die tot oplossing van de knapzak voeren.

De eerste stap is zich te realiseren dat – gegeven de parametrisatie van de knapzak – de waarde van m van de orde 2^{200} zal zijn en zodoende ook de b_i 's in die grootteklasse zullen vallen. Ten opzichte daarvan

zijn de laagste knapzakcomponenten $a_1, a_2 \dots$ veel en veel kleiner, namelijk van de grootteorde $2^{100}, 2^{101}$ enz. Bij de reductie van b_1 naar a_1 met $w^{-1}b_1 \bmod m$ vindt dus een flinke verkleining plaats.

Genoemde verkleining kan aanschouwelijk gemaakt worden door met w^{-1}/m over te gaan op een continue, reële variabele en deze te beschouwen als de vermenigvuldigingsfactor voor de reductie van b_i naar a_i . De modulo m operatie wordt daarbij omgezet naar modulo 1. **Figuur 9.5** laat in het plaatje links het verloop van de reductiefactor $(w^{-1}m) \bmod 1$ als functie van w^{-1}/m zien: een zaagtandfunctie waarvan de waarde telkens van 0 aangroeit naar 1 om dan weer terug te vallen tot 0. De getalswaarden van de knapzak zorgen ervoor dat de lijnen zeer steil verlopen, veel steiler zelfs dan in het plaatje kon worden getekend. Er zullen daarom zeer veel lijnstukken zijn. Voor elke knapzakcomponent is er zo'n functie voor de reductiefactor, onderling verschillend in de helling van de lijnen.



Figuur 9.5 Reductie van de knapzakvector

Om $b_1 \approx 2^{200}$ te reduceren naar $a_1 \approx 2^{100}$ moet de waarde van de vermenigvuldigingsfactor w^{-1}/m heel klein zijn, in de orde van 2^{-100} . In de figuur is dat dus nagenoeg onderaan op een van de zaagtanden en heel dicht bij de horizontale as. Slechts een klein stukje rechts van de plaats waar de zaagtand een nulpunt heeft. Hetzelfde zal zeker ook gelden voor de reductie van andere laaggenummerde knapzak-

componenten, zoals $a_2, a_3, \dots$

Nu komt de belangrijkste stap. Bij de juiste waarde van (w, m) is deze goed voor alle reducties tegelijk. Anders gezegd, voor die combinatie van w en m liggen de nulpunten van de respectievelijke zaagtanden zeer dicht bijeen. Aan de rechterkant in [figuur 9.5](#) is die situatie sterk uitvergroot getekend. De verticale streeplijn stelt de reducerende combinatie (w, m) voor. Deze lijn snijdt de afzonderlijke lijnstukken voor de reductie van a_1, a_2, a_3, a_4 . De snijpunten geven de respectievelijke reductiefactoren. Al die factoren zijn zeer klein, dus ligt het nulpunt van elk der lijnstukken niet ver links. Er tekent zich een cluster van nulpunten af.

Twee vragen zijn nu van belang. De eerste is: "Kan zo'n nulpuntencluster op een efficiënte manier worden gevonden?" En de tweede: "Is zo'n cluster tamelijk uniek of verdrinkt de goede cluster in een zee van toevallige clusters?" De eerste vraag kan direct bevestigend worden beantwoord. Er is een polynomiaal – en voor de cryptanalyse dan per definitie goed genoeg – algoritme waarmee de cluster kan worden gevonden. Dit zogenaamde L3-algoritme van Lenstra, Lenstra en Lovász vindt basisvectoren voor een rooster in een n -dimensionale ruimte^[25]. Voor de tweede vraag moet de kans op een toevallige nulpuntencluster worden uitgerekend. Voor een systeem zoals hier, met 100 componenten en 4 nulpunten blijkt de kans op het toevallig voorkomen van zo'n cluster ongeveer 2^{-94} , verwaarloosbaar klein dus. Het vinden van vier bijeenliggende nulpunten blijkt also voldoende om de knapzak te breken. Wanneer de nulpuntencluster gevonden is, moet nog wel naar de afzonderlijke w en m gezocht worden. De details daarvan blijven hier achterwege.

9.8 Quantum computer

Een hot item in de ontwikkeling van computers is de mogelijkheid dat binnen afzienbare tijd een quantum-computer beschikbaar zal komen. Hoe lang dit duurt kan nog niemand zeggen, de meest optimistische schatting mikt op een jaar of vijftien. Komt er een quantumcomputer, dan is de public-key cryptografie

in groot gevaar. Dat is gevolg van het feit dat er efficiënte algoritmen op de quantumcomputer mogelijk zijn voor het factorisatie- en het discrete logaritme probleem. Een directe bedreiging dus voor systemen als Diffie-Hellman, RSA en ECC-cryptografie.

In dit verband is curieus dat de Amerikaanse National Security Agency (NSA) in de tweede helft van 2015 een statement liet uitgaan waarin op bovenstaande ontwikkelingen wordt vooruitgelopen:

“For those partners and vendors that have not yet made the transition to Suite B algorithms, we recommend not making a significant expenditure to do so at this point but instead to prepare for the upcoming quantum resistant algorithm transition... Unfortunately, the growth of elliptic curve use has bumped up against the fact of continued progress in the research on quantum computing, necessitating a re-evaluation of our cryptographic strategy.”

en

“...elliptic curve cryptography is not the long term solution many once hoped it would be. Thus, we have been obligated to update our strategy.”

In het lezenswaardige artikel *A Riddle Wrapped in an Enigma*^[26] proberen twee gerenommeerde onderzoekers (Koblitz en Menezes) deze uitspraken te duiden. In het bijzonder vragen als: Wat zijn de 'wheelings en dealings' van de NSA met ECC-cryptografie? Hoe ver gevorderd is de NSA met het breken van ECC-cryptografie? Hoe groot is het gevaar van de toekomstige quantumcomputer? De onthullingen van Snowden hebben geopenbaard dat de aanvankelijke promotie van ECC-cryptografie door de NSA wel eens gemotiveerd kon zijn door het bestaan van een achterdeur in de gestandaardiseerde versie van de zogenaamde Dual Elliptic Curve Deterministic Random Bit Generator (Dual_EC_DRBG)* waardoor de NSA het geheimschrift kon breken[†].

* Meer informatie over deze en andere aanbevolen random generatoren is te vinden in NIST publicatie 800-90A van januari 2012 getiteld *Recommendation for Random Number Generation Using Deterministic Random Bit Generators* en te vinden op de NIST-site <http://www.nist.gov>, Computer Security Resource Center.

Wat is meer, ook aan het licht kwam dat de NSA in het geheim \$10.000.000 heeft betaald aan RSA Company, vermoedelijk om die versie met de achterdeur als default in te bouwen in hun cryptografische toolkit. Begin 2016 maakt Juniper Networks, een Amerikaanse producent van firewalls bekend de Dual_EC_DRBG random generator uit een van zijn productlijnen te zullen verwijderen omdat deze "contained unauthorized code that can surreptitiously decrypt traffic sent through virtual private networks." Het is zelfs zo erg dat in 2014 een verandering in de programmacode was aangebracht, waardoor iemand die een hard ingecodeerd wachtwoord kent de communicatie kan ontcijferen*.

Duidelijk is in ieder geval dat de ontwikkeling van quantumcomputer-bestendige algoritmen voor systemen met publieke sleutel de komende jaren een hot research topic zal zijn†.

9.9 Referenties

- [1] Mark A. Will and Ryan K.L. Ko (2014). Computing Mod Without Mod. <http://eprint.iacr.org/, 2014/755>.
- [2] P. C. van Oorschot, A. J. Menezes and S. A. Vanstone (1997). *Handbook of Applied Cryptography*. CRC Press, Inc.. In hoofdstuk 2, algoritme 2.149.
- [3] National Institute of Standards and Technology (2000). *Digital Signature Standard (DSS) - FIPS PUB 186-2*. U.S. Dept. of Commerce
- [4] J. Gordon (1985). Strong primes are easy to find. In N. Cot, T. Beth and I. Ingemarsson, editors (1985). *Advances in Cryptology*, number 209 in Lecture Notes in Computer Science., pages 216–223.

† Kennis van de waarden die de random generator aflevert, geeft cruciale informatie over de startwaarden waarmee de cryptooritmen gestart worden.

* Credit voor deze ontdekking op <https://community.rapid7.com/community/infosec/blog/2015/12/20/cve-2015-7755-juniper-screeno>

† Van bestaande symmetrische cryptosystemen zoals AES wordt verondersteld dat een quantumcomputer de sleutelentropie met een factor 2 verlaagt. Dit betekent dat AES-256 een veiligheidsmarge van 128 bits overhoudt tegen een aanval met een quantumcomputer.

- [5] M. Agrawal, N. Kayal and N. Saxena (2004). PRIMES is in P. *Annals of Mathematics*, 160(2), 781–793.
- [6] H. W. Lenstra jr and C. Pomerance (2011). Primality testing with Gaussian periods. <https://math.dartmouth.edu/carlp/aks041411.pdf>.
- [7] W. Diffie and M. E. Hellman (1976). New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22, 644–654.
- [8] D. Adrian e.a. (2015). Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice. In *22nd ACM Conference on Computer and Communications Security 2015, Denver, Colorado, USA* . URL = www.sigsac.org/ccs/ccs2015
- [9] L. M. Adleman, R. L. Rivest and A. Shamir (1978). A Method for Obtaining Digital Signatures and PublicKey CryptoSystems. *Communications of the Association for Computing Machinery*, 21(2), 120–126.
- [10] S. Contini (1999). The Factorization of RSA-140. Technical Report RSA Laboratories.
- [11] A. K. Lenstra and E. R. Verheul (2001). Selecting Cryptographic Key Sizes. *Journal of Cryptology*, 14, 255–293.
- [12] A. K. Lenstra, J. P. Hughes, M. Augier, J. W. Bos and T. Kleinjung et al. (2012). Ron was wrong, Whit is right. *Cryptology ePrint Archive* <http://eprint.iacr.org>.
- [13] D. Coppersmith (1997). Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *Journal of Cryptology*, 10, 233–260.
- [14] B. de Weger (2002). Cryptanalysis of RSA with Small Prime Difference. *AAECC*, 13, 17–28.
- [15] Abderrahmane Nitaj and Tajjeeddine Rachidi (2015). Factoring RSA moduli with weak prime factors. <http://eprint.iacr.org/2015/389>.
- [16] D. Genkin, A. Shamir and E. Tromer (2013). RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis. <http://eprint.iacr.org/>.
- [17] T. ElGamal (1985). A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In G. R. Blakley and D. Chaum, editors (1985). *Advances in Cryptology*, number 196 in Lecture Notes in Computer Science ., pages 10–18.
- [18] National Institute of Standards and Technology (2000). *Digital Signature Standard (DSS) - FIPS PUB 186-2*. .U.S. Dept. of Commerce
- [19] N. Koblitz (2004). A Survey of Public-Key Cryptosystems. *SIAM Review*, 46(4), 599–634.
- [20] I. Blake, G. Seroussi and N. Smart (1999). *Elliptic Curves in Cryptography* Number 265 in London Mathematical Society Lecture Note Series. Cambridge University Press.

- [21] D. Hankerson M. Brown, J. López and A. Menezes (2001). Software Implementation of the NIST Elliptic Curves over Prime Fields. InD. Naccache, editor (2001). *Topics in Cryptology*, number 2020 in Lecture Notes in Computer Science ..
R. Schroepel, H. Orman, S. O'Malley and O. Spatscheck (1995). Fast Key Exchange with Elliptic Curve Systems. InD. Copper-smith, editor (1995). *Advances in Cryptology*, number 963 in Lecture Notes in Computer Science ., pages 43–56.
J. López and R. Dahab (1999). Fast Multiplication on Elliptic Curves over $GF(2^m)$ without Precomputation. InL. Goubin and J. Patarin, editors (1999). *Advances in Cryptology*, number 1717 in Lecture Notes in Computer Science ..
- [22] A. E. Escott, J. C. Sager, A. P. L. Selkirk and D. Tsapakidis (1999). Attacking Elliptic Curve Cryptosystems Using the Parallel Pollard rho Method. *RSA Laboratories CryptoBytes*, winter.
- [23] R. C. Merkle and M. E. Hellman (1978). Hiding Information and Signatures in Trap-door Knapsacks. *IEEE Transactions on Information Theory*, IT-24(5), 525–530.
- [24] A. Shamir (1983). A Polynomial Time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem. InA. T. Sherman D. Chaum, R.L.Rivest, editor (1983). *Advances in Cryptology* ., pages 279–288.
- [25] A. K. Lenstra, H. W. Lenstra jr. and L. Lovász (1982). Factoring Polynomials with Rational Coefficients. *Mathematische Annalen*, 261, 515–534.
- [26] Neal Koblitz and Alfred J. Menezes (2015). A Riddle Wrapped in an Enigma. <http://eprint.iacr.org/>, 2015/1018.

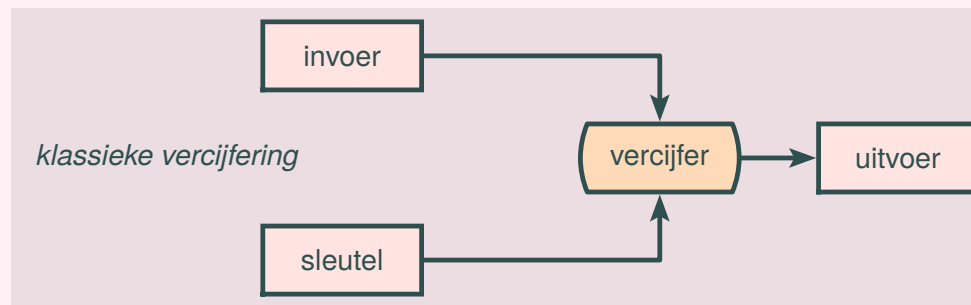
10 Sleutelstroom

10.1	Stroomvercijfering	456
10.1.1	Vernam-systeem	459
10.1.2	Golomb-criteria	460
10.2	Teruggekoppeld schuifregister	463
10.2.1	Terugkoppelfunctie	467
10.2.2	Vertakkingspunten	471
10.3	Lineair schuifregister	473
10.3.1	Periodiciteit	477
10.3.2	Kwaliteit	484
10.3.3	Cryptografische sterkte	487
10.4	Niet-lineair schuifregister	488
10.4.1	Lineaire complexiteit	489
10.5	Registerselectie	491
10.5.1	Multiplex generator	492
10.5.2	Filterfuncties	497
10.6	Registercombinatie	501
10.6.1	Pless generator	503
10.6.2	Geffe generator	506
10.6.3	Bruer generator	507
10.7	Correlatieaanval	508
10.7.1	Cijfertekstcorrelatie	510

10.7.2	Siegenthaler-aanval	512
10.7.3	Meier-Staffelbach-aanval	514
10.7.4	Correlatie-immuniteit	520
10.7.5	Somgenerator	522
10.8	Klokgestuurde registers	525
10.8.1	Stop-and-go generator	526
10.8.2	Krimpgenerator	528
10.8.3	GSM A5/1-generator	530
10.9	Andere generatoren	534
10.9.1	RC4	534
10.9.2	KASUMI	535
10.10	Referenties	537

10.1 Stroomvercijfering

In geheimschriften als transposities, Vigenère, en dergelijke, wordt uit de klaartekst en de vercijfersleutel direct het cryptogram gemaakt. Klaartekst en sleutel worden in één vercijferstap tot het eindresultaat omgezet, zoals valt te zien **figuur 10.1**. Het is dus de klaartekst die direct de bewerkingen van de cryptaalgoritme ondergaat.

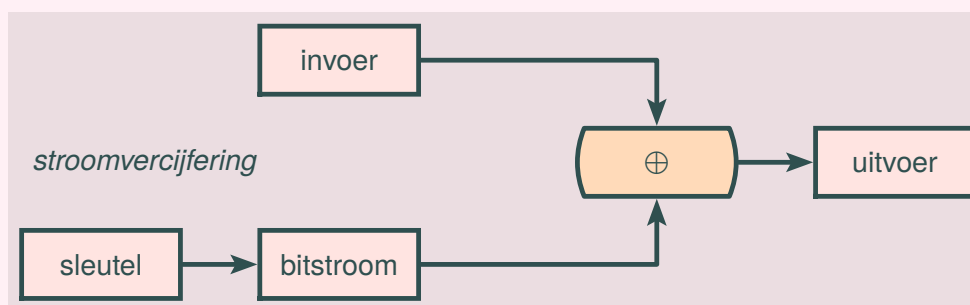


Figuur 10.1 Rechtstreekse vercijfering van de klaartekst

In de praktijk heeft deze werkwijze enige nadelen die vooral optreden bij het gebruik van blokgeheimschriften. De oorzaak is gelegen in het feit dat bij zulke geheimschriften de tekst moet worden onderverdeeld in blokken van vaste lengte, die stuk voor stuk vercijferd worden. Voor tekstdocumenten vormt dit niet zo'n probleem. Het is daarentegen wel lastig als bijvoorbeeld een telefoongesprek moet worden vercijferd, omdat er dan immers een doorlopende *gegevensstroom* is*. Weliswaar kan een methode worden bedacht om een blokgeheimschrift te benutten voor de vercijfering van zo'n stroom, maar in dat geval moet ingeleverd worden op de snelheid van vercijfering.

* Deze uitspraak geldt wel voor de ouderwetse draadtelefoon, maar moet worden genuanceerd voor GSM-telefonie. Daar immers is wel sprake van opdeling van de gegevensstroom in bitblokken. Dat tast overigens het principe van de stroomvercijfering ook in dat geval, niet aan.

Het bewerken van gegevensstromen kan ook anders worden ingericht. In plaats van de vercijferoperaties uit te voeren op de klaartekst, voorziet men zich van een *sleutelstroom*. Deze sleutelstroom in de vorm van een rij bits of bytes, wordt gecombineerd met die van de gegevensstroom zoals weergegeven in [figuur 10.2](#). Als combinatie-operator is de *exclusive or* (xor, modulo 2 optelling) zeer geschikt, want die bewerking gaat snel en eenvoudig. De vercijfersleutel creëert hier dus niet het cryptogram maar creëert de sleutelstroom.



Figuur 10.2 Vercijfering met intermediaire sleutelstroom

In [tabel 10.1](#) staat een voorbeeld waarin de ASCII-tekst "Hallo" met een gekozen sleutelstroom is vercijferd en ontcijferd. Het zware werk is in de *stroomvercijfering* verschoven van de gegevensvercijfering naar de productie van de sleutelstroom. Voor het maken van zo'n sleutelstroom staan goede en efficiënte methoden ter beschikking. In formule, met P de klaartekststroom, S de sleutelstroom en C de resulterende vercijferde stroom, luiden vercijfering en ontcijfering

$$C = E(P, S) = P \oplus S \quad \text{en} \quad P = D(C, S) = C \oplus S \quad (10.1)$$

vercijfering					
ASCII	H	a	l	l	o
P	01001000	01100001	01101100	01101100	01101111
S	01100111	10101000	01000110	11010011	01000010
$C = P \oplus S$	00101111	11001001	00101010	10111111	00101101
ontcijfering					
C	00101111	11001001	00101010	10111111	00101101
S	01100111	10101000	01000110	11010011	01000010
$P = C \oplus S$	01001000	01100001	01101100	01101100	01101111
ASCII	H	a	l	l	o

Tabel 10.1 Stroomvercijfering en -ontcijfering

In dit hoofdstuk zal de productie van hoogwaardige, voor vercijfering geschikte sleutelstromen worden behandeld. De meest gangbare methoden zijn gebaseerd op combinaties van zogenaamde teruggekoppelde schuifregisters. Als andere mogelijke bronnen kunnen worden genoemd: het gebruik van cijferreeksen uit statistische jaarboeken, logaritme-tabellen, reeksen gegeneerd door pseudorandom-algoritmen, enz. Stukken tekst uit een boek zijn natuurlijk niet geschikt, omdat dan een sleutelstroom met te weinig entropie wordt gebruikt*.

Het zal in dit hoofdstuk hoofdzakelijk gaan over de productie van een *bitstroom*. Daar waar gesproken wordt van een sleutelstroom is dat dus als regel een stroom bits: vercijfering en ontcijfering geschieden bit voor bit. Maar alvorens daaraan te beginnen eerst een uitstapje naar de oorsprong van de stroomvercijfering, gevolgd door een beschouwing over de kwaliteit van sleutelstromen.

* Dit is al aan de orde gekomen bij het one-time pad en bij de behandeling van de sleutelentropie.

10.1.1 Vernam-systeem

De uitvinding van de stroomvercijfering wordt toegeschreven aan de Amerikaan Gilbert S. Vernam^[1]. Deze werkte in 1917 bij de American Telephone & Telegraph Company (AT&T) aan de veiligheid van telexcommunicatie. De Verenigde Staten waren toen net aan de Eerste Wereldoorlog gaan deelnemen. In een telex worden berichten op een ponsband gezet in de zogenaamde Baudot code, een vijfbits code afgebeeld in [figuur 10.3](#).



Vernam

[illegible]

Figuur 10.3 Baudot telexcode

Vernam bedacht dat het vercijferen kan geschieden door een tweede ponsband met een vercijfersleutel gelijk op te laten lopen met het bericht en de ponsingen in beide te combineren met de exclusieve or. Op deze manier is bovendien het vercijferen geautomatiseerd, een flinke verbetering ten opzichte van handmatig werken. Naar de uitvinder is dit geheimschrift *Vernam-vercijfering* gaan heten.

Aanvankelijk werd gebruik gemaakt van een enkele rondlopende ponsband. Het resultaat is dan een periodieke substitutie met een 32-letter alfabet. Later werd de periode groter gemaakt door twee banden van verschillende lengte met elkaar te combineren. De totale periode wordt dan het kleinste gemene veelvoud van de lengte van de afzonderlijke banden. Het eenmalig gebruik van een lange, willekeurig geconstrueerde sleuteltape maakt er een one-time pad-systeem van.

10.1.2 Golomb-criteria

Toepassen van stroomvercijfering staat of valt met de productie van een goede sleutelstroom. Het moet vanzelfsprekend niet eenvoudig zijn om uit een klein stukje van de stroom zomaar de gehele stroom te kunnen reconstrueren. Want een deel van de sleutelstroom kan bekend raken door overeenkomstige klaartekst en cryptogram te combineren. Dat is eenvoudig in te zien door de [vergelijkingen \(10.1\)](#) te combineren. De sleutelstroom kan worden gereconstrueerd met

$$C \oplus P = P \oplus S \oplus P = S \quad (10.2)$$

[Tabel 10.2](#) gebruikt het voorbeeld van [tabel 10.1](#) en laat zien hoe door combinatie van klaartekst en cijfertekst eenvoudig de bijbehorende sleutelstroom wordt gevonden. Een goede sleutelstroomgenerator is daarom zo ontworpen dat de ingestelde sleutel niet gemakkelijk valt af te leiden uit kleine stukjes sleutelstroom. In dit verband wordt een *complexiteitsmaatstaf* gehanteerd voor het kwantificeren van de kwaliteit van de generator. Daar wordt later nog op teruggekomen bij de behandeling van het onderwerp *lineaire complexiteit*.

	sleutelextractie				
P	01001000	01100001	01101100	01101100	01101111
C	00101111	11001001	00101010	10111111	00101101
$S = P \oplus C$	01100111	10101000	01000110	11010011	01000010

Tabel 10.2 Reconstructie sleutelstroom uit cryptogram en klaartekst

De mooiste sleutelstroom – mooi uit het oogpunt van veiligheid – is de oneindig lange, volstrekt willekeurige sleutelstroom van het one-time pad. Zo'n stroom kan gemaakt worden uit natuurlijke processen met willekeurig gedrag, zoals radioactief verval, elektronische ruis, en dergelijke. In alle andere gevallen is sprake van een deterministisch proces (een computerprogramma bijvoorbeeld). Zoiets

kan nimmer een echte willekeurige sleutelstroom voortbrengen, maar slechts een *pseudorandom* sleutelstroom.

Het wezenlijke verschil tussen een echte en een pseudorandom sleutelstroomgenerator kan als volgt worden verwoord. Door het deterministische karakter van de sleutelstroomgenerator (een *automaat*) zal een bepaalde toestand altijd door dezelfde volgende toestand worden opgevolgd. Nu kan een eindige machine (en dat zijn alle apparaten die wij maken) zich slechts in een eindig aantal verschillende toestanden bevinden. Op den duur zal daarom altijd een eerdere toestand opnieuw bereikt worden. Ergo, een sleutelstroom komend uit zo'n automaat heeft altijd een *periode* en komt vroeg of laat in een cyclus terecht. De lengte van die cyclus (de periode) hangt af van de constructie van de automaat en soms ook van de toestand waarin deze wordt gestart. Sommige automaten produceren precies één complete cyclus, andere meerdere onafhankelijke of zelfs een cyclus die via een aanlooptraject wordt bereikt. Een bitstroomgenerator is efficiënt als met een kleine instelsleutel een lange cyclus kan worden gemaakt.

Een lange periode is op zich geen garantie voor een hoogwaardige sleutelstroom. Een andere vereiste is onvoorspelbaarheid. Daartoe moeten de nullen en enen goed verdeeld zitten over de periode. Een sleutelstroom waarin alle 0'en bij elkaar zitten is te voorspelbaar en voor de cryptografie waardeloos. Door Golomb is in zijn standaardwerk over schuifregisters aangegeven aan welke eisen een kwalitatief goede sleutelstroom moet voldoen^{[2]*}. Hij geeft een drietal criteria

1. Het aantal nullen en enen moet zo goed mogelijk aan elkaar gelijk zijn. De reden is wel duidelijk. Stel er zijn driemaal zoveel nullen als enen, dan heeft de cryptoanalist bij het raden van een sleutelbit driemaal zoveel kans op een 0 als op een 1. Dat geeft minder onzekerheid dan in het geval van gelijke aantallen, want dat biedt slechts een fifty-fifty kans op goed raden.
2. Een rij opeenvolgende dezelfde bits heet een *run*. Een run van enen noemt men een *blok* en een rij

* Golomb borduurt in dat boek voort op werk uit 1965 van Ernst Selmer, een Noorse cryptograaf in regeringsdienst. Golomb zelf werkte voor de Amerikaanse NSA.



Golomb

opeenvolgende nullen een *gat*. Voor een goede onvoorspelbaarheid moeten er evenveel blokken als gaten van bepaalde lengte zijn en hoe langer de run hoe zeldzamer. Om precies te zijn, de helft van alle rijtjes mag lengte 1 hebben, een kwart lengte 2, een achtste lengte 3, enz. Op die manier wordt het moeilijk om op grond van een waargenomen rij bits een voorspelling te doen over het volgende bit.

3. Het derde criterium is de *autocorrelatie*, waarbij de rij nullen en enen met een verschoven copie van zichzelf wordt vergeleken. De autocorrelatie meet als het ware de overeenkomst tussen rij en verschoven rij. De rij geldt van goede kwaliteit wanneer deze grootte voor alle waarden van de verschuiving constant blijft*. Zo niet, dan kan het verloop van deze grootte informatie verschaffen over de structuur van de rij.

Voor een periodieke bitrij $s_0, s_1, \dots, s_{p-1}$ met periode p en met faseverschil δ is de waarde van de autocorrelatie $C(\delta)$ gelijk aan het gewogen verschil tussen de aantallen gelijke en ongelijke overstaande bits. Deze grootte kan worden berekend met de volgende formule†

$$C(\delta) = 1 - \frac{2}{p} \sum_{i=0}^{p-1} s_i \oplus s_{i+\delta} \quad \delta = 0, 1, \dots, p-1 \quad (10.3)$$

Vanwege de eigenschappen van $\oplus$: $(0 + 0) = (1 + 1) = 0$ en $(0 + 1) = (1 + 0) = 1$, valt direct in te zien dat de sommatie het aantal bits met verschillende waarde over een periode telt (en dus indirect ook het aantal gelijke). Eenvoudig volgt daaruit dat de formule inderdaad overeenkomt met de bovengegeven interpretatie als verschil van genoemde aantallen bits. Een hoge absolute waarde van $C(\delta)$ duidt op veel overeenkomst (hetzij rechtstreeks, hetzij na inverteren van een der rijen), een lage waarde duidt op grote verschillen. Voor $\delta = 0$ bevinden de rijen zich precies in fase en is $C(0) = 1$ maximaal.

Verderop, bij de behandeling van de lineaire schuifregisters, volgt een uitwerking van deze Golomb-

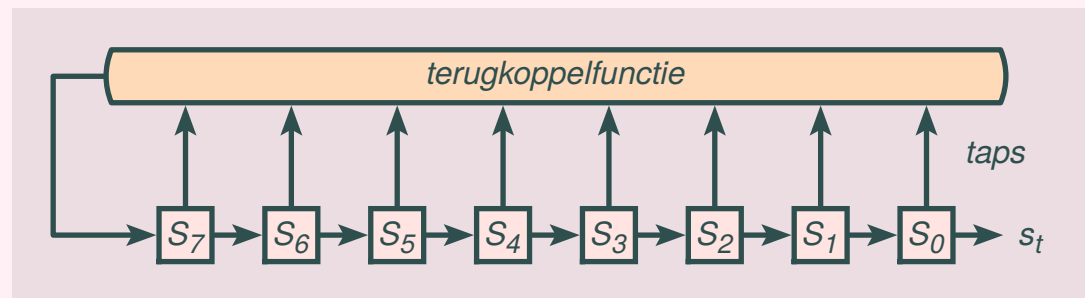
* Behalve voor het ene geval waarin de rij precies met zichzelf wordt vergeleken want dan vallen alle bits natuurlijk over elkaar.

† In het vervolg zullen $\oplus$ en $+$ doorelkaar gebruikt worden als er geen verwarring mogelijk is.

criteria. Men moet overigens bedenken dat het tamelijk zware eisen zijn. Ook als ze wat worden verzacht kan een bitstream goed bruikbaar zijn. Er zijn nog meer criteria om de kwaliteit van pseudorandomreeksen te meten dan de drie genoemde, maar die worden hier niet behandeld. Een overzicht kan worden gevonden in het tweede deel van Knuth's standaardwerk *The Art of Computer Programming*^[3].

10.2 Teruggekoppeld schuifregister

In [figuur 10.4](#) staat een voorbeeld van een veel gebruikt systeem om een pseudorandom bitstream te genereren, het *teruggekoppeld schuifregister* (Feedback Shift Register in het Engels), afgekort tot fsr. Na de introductie van het lineair teruggekoppeld schuifregister (Linear Feedback Shift Register in het Engels) zal dat de afkorting lfsr krijgen. Schuifregisters laten zich eenvoudig implementeren in hardware en zijn dan zeer veel sneller dan cryptosystemen in software.



Figuur 10.4 Teruggekoppeld 8-bits schuifregister

Het schuifregister van [figuur 10.4](#) bevat 8 hokjes gemerkt S_0 tot en met S_7 . Deze hokjes zijn geheueeenheden die elk 1 bit kunnen bevatten; ze heten de *registertrappen*. Bij n trappen $S_0 \dots S_{n-1}$ heet het register een n -traps schuifregister. Alle registertrappen tesamen bepalen de *toestand* van het register. Op elk tijdstip t wordt de uitvoer van het fsr gerepresenteerd door s_t , het uitvoerbit van dat moment. Dit

uitvoerbit komt uit registertrap S_0 en de uitvoerrij wordt ermee verlengd op het moment dat de generator een bit produceert.

$$\cdots S_{t-2} S_{t-1} \rightarrow \cdots S_{t-2} S_{t-1} S_t = \cdots S_{t-2} S_{t-1} S_0(t) \quad (10.4)$$

Het schuifregister gaat vervolgens over in de volgende toestand doordat alle bits één positie naar rechts opschuiven en helemaal links een nieuw gemaakt bit de opengevallen plaats inneemt.

$$S_1 \rightarrow S_0 \quad S_2 \rightarrow S_1 \quad \dots \quad S_{n-1} \rightarrow S_{n-2} \quad \text{nieuw bit} \rightarrow S_{n-1} \quad (10.5)$$

Omdat de operaties in hardware worden gestuurd door een elektronische klok, heeft de overgang van het fsr naar de volgende toestand plaats in een volgende *klokcyclus* van het register. Het nieuwe bit voor de meest linker registertrap (S_7 in het voorbeeld) wordt gemaakt uit de bestaande toestand (inhoud van de registertrappen) door daarop de zogenaamde *terugkoppelfunctie* te laten werken.

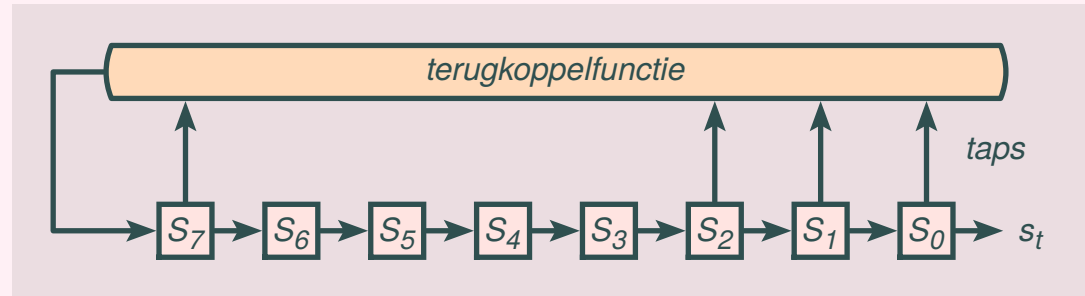
$$S_{n-1} = f(S_0, \dots, S_{n-1}) \quad (10.6)$$

Deze terugkoppelfunctie hoeft niet noodzakelijk alle registerbits te gebruiken. Het kan bijvoorbeeld zijn dat alleen de bits in de trappen met de nummers 0, 1, 2 en 7 een directe rol spelen. De gebruikte registertrappen heten de *tappunten* (taps in de Engelstalige literatuur). De waarde van de terugkoppelfunctie is alleen afhankelijk van bits in registertrappen met een tap. **Figuur 10.5** laat het teruggekoppeld schuifregister nog eens zien, maar nu zijn alleen de tappunten 0, 1, 2 en 7 verbonden.

Men kan de opeenvolgende toestanden van het schuifregister op verschillende tijdstippen weergeven met opvolgende registertoestanden.

$$S(t) = S_{n-1,n-2,\dots,0}(t), \quad S(t+1) = S_{n-1,n-2,\dots,0}(t+1) \quad \dots \quad (10.7)$$

Hetzelfde kan echter ook bereikt worden door naar de uitvoerrij te kijken, want daarin ziet men de *historie* van het register aan zich voorbijtrekken. Immers, als op tijdstip t het uitvoerbit in S_0 zit, dan bevond zich dit op tijdstip $t-1$ nog in S_1 , enz. Uit tenminste n opeenvolgende voorgaande bits van de bitrij zijn daarom alle voorgaande (en opvolgende) toestanden van het fsr te bepalen.



Figuur 10.5 Teruggekoppeld 8-bits schuifregister met taps 0, 1, 2, 7

Het moge duidelijk zijn dat de langste periode die met een n -taps register behaald kan worden van lengte 2^n is. Dat komt omdat een register van n -bits nooit meer dan 2^n verschillende waarden kan aannemen. Zodra een bepaalde registerwaarde opnieuw voorkomt, zal de geproduceerde bitrij zich vanaf dat moment exact herhalen. Of de maximale waarde 2^n ook daadwerkelijk gehaald wordt, of dat verschillende kleinere cycli worden gevormd, hangt geheel af van de keuze van de terugkoppelfunctie. Een goede keuze van de terugkoppelfunctie is cruciaal voor zowel de lengte van de periode als voor de kwaliteit van de geproduceerde bitrij. Het effect van verschillende terugkoppelfuncties zal worden gedemonstreerd aan twee 4-taps fsr's die ogenschijnlijk weinig van elkaar verschillen.

$$f_1 = 1 + S_2 + S_0 S_1 S_2 S_3 \quad \text{en} \quad f_2 = 1 + S_0 + S_2 + S_0 S_1 S_2 S_3 \quad (10.8)$$

In [tabel 10.3](#) is van deze beide terugkoppelfuncties voor alle 16 mogelijke toestanden van de registers uitgerekend welk nieuw bit gemaakt wordt en wat dus de opvolger van elke toestand is. Startend vanuit een gekozen begintoestand kan een cyclus worden opgebouwd. Herhaling van de keuze van een starttoestand is nodig voor nog niet voorgekomen toestanden en dat levert uiteindelijk de complete verzameling cycli op die het register produceert. Deze cycli zijn in de [figuur 10.6](#) in beeld gebracht door met pijlen de nummers van elkaar opvolgende toestanden te verbinden. Goed is te zien hoe verschillend de cyclusstructuur wordt door variatie van de terugkoppelfunctie. De ene functie vormt bijna alleen maar losstaande cycli, de andere

kent veel punten waar losse takken bijeenkomen.

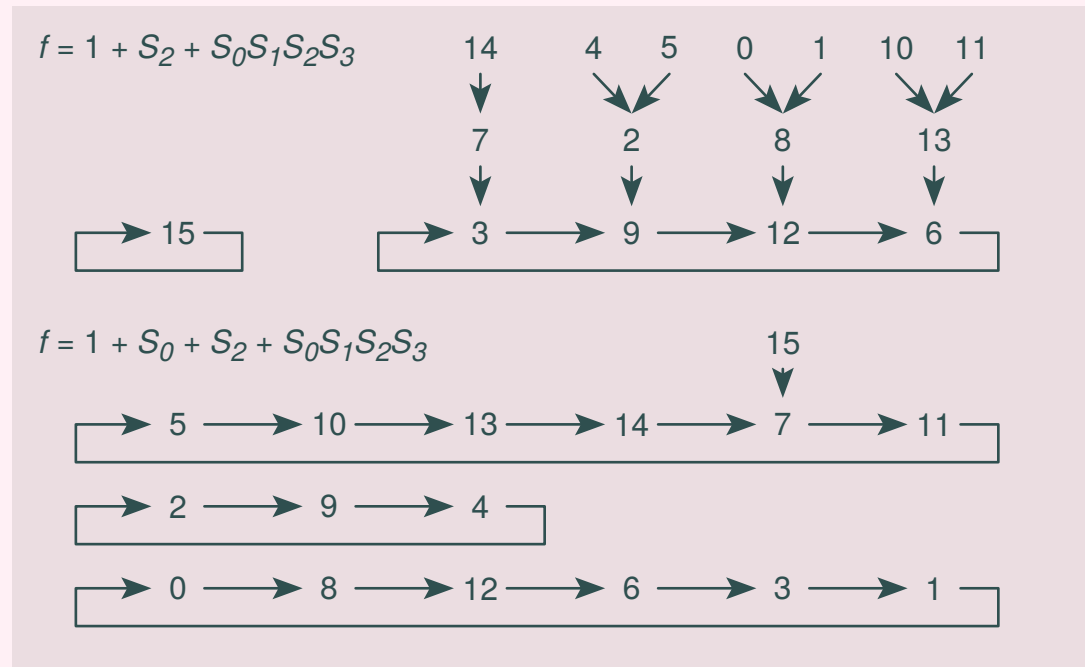
#	$S(t)$	f_1	$S(t+1)$	#	f_2	$S(t+1)$	#
0	= 0000	1	1000 =	8	1	1000 =	8
1	= 0001	1	1000 =	8	0	0000 =	0
2	= 0010	1	1001 =	9	1	1001 =	9
3	= 0011	1	1001 =	9	0	0001 =	1
4	= 0100	0	0010 =	2	0	0010 =	2
5	= 0101	0	0010 =	2	1	1010 =	10
6	= 0110	0	0011 =	3	0	0011 =	3
7	= 0111	0	0011 =	3	1	1011 =	11
8	= 1000	1	1100 =	12	1	1100 =	12
9	= 1001	1	1100 =	12	0	0100 =	4
10	= 1010	1	1101 =	13	1	1101 =	13
11	= 1011	1	1101 =	13	0	0101 =	5
12	= 1100	0	0110 =	6	0	0110 =	6
13	= 1101	0	0110 =	6	1	1110 =	14
14	= 1110	0	0111 =	7	0	0111 =	7
15	= 1111	1	1111 =	15	0	0111 =	7

Tabel 10.3 $f_1 = 1 + S_2 + S_0 S_1 S_2 S_3$ $f_2 = 1 + S_0 + S_2 + S_0 S_1 S_2 S_3$

De volgende drie vragen zijn van belang voor de keuze van geschikte terugkoppelfuncties

- Hoe kan de terugkoppelfunctie worden bepaald uit waargenomen sleutelbits?
- Hoe kan worden bereikt dat alle toestanden zich in een cyclus bevinden zonder losse takken?
- Welke terugkoppelfuncties zijn geschikt als bouwsteen voor meer ingewikkelde systemen?

Ter beantwoording van de eerste vraag zal een eenvoudig rekenschema worden gepresenteerd, waarmee voor elke willekeurige bitrij de bijbehorende terugkoppelfunctie kan worden gevonden. De beantwoording



Figuur 10.6 Opeenvolgende toestanden voor f_1 en f_2

van de tweede vraag sluit daar direct op aan. De derde vraag zal leiden naar de eenvoudigste teruggekoppelde schuifregisters, te weten de lineair teruggekoppelde schuifregisters.

10.2.1 Terugkoppelfunctie

Uit een waargenomen bitstroom en het gegeven dat een n -traps register in het spel is, kan de onderliggende terugkoppelfunctie bepaald worden. Hiervoor valt een algemeen rekenschema op te stellen. Deze techniek is niet alleen geschikt voor het bepalen van de terugkoppelfunctie van een enkel fsr, maar zal in het vervolg ook bruikbaar blijken bij combinaties van fsr's.

De eerste stap in het werkschema is het construeren van alle voorkomende registertoestanden, waardoor duidelijk wordt welke toestand welk uitvoerbit oplevert. In de tweede stap wordt met behulp daarvan de terugkoppelfunctie uitgerekend. Wanneer bit s_t op tijdstip t uit registertrap S_0 komt, zit op datzelfde moment het volgende te produceren bit s_{t+1} nog in S_1 . Het bit dat daar weer na komt is s_{t+2} en dat zit nu nog in S_2 , enz.

Onderstaande bitstroom fungeert als voorbeeld voor de uitwerking.

$s = 01100011110100011110100011110100011110$

Deze bitstroom heeft een aanloop van 2 bits, gevolgd door een cyclus met periode 9.

$s = (01)(100011110)(100011110)(100011110)(100011110)...$

Bekend is dat deze bitstroom uit een 4-traps fsr komt en dit betekent dat de volgende toestanden elkaar hebben opgevolgd

0110-0011-0001-1000-1100-1110-1111-0111-1011-0101-0010-0001-

Met deze gegevens kan van 11 van de 16 registertoestanden de opvolger worden bepaald (niet die van 0000, 0100, 1001, 1010, 1101). Aangenomen dat uit andere waarnemingen de overige opvolgers gevonden zijn, is het resultaat zoals gepresenteerd in [tabel 10.4](#)*.

In de kolom links in de tabel staan alle 16 mogelijke uitgangstoestanden voor een register met 4 trappen. In de kolom daarnaast staat de toestand die er blijkens vorenstaande reconstructie op volgt. Een uitgangstoestand zoals $S_t = 0010$ betekent dat op dat moment het register als volgt was gevuld $S_3 = 0$, $S_2 = 0$, $S_1 = 1$, $S_0 = 0$. Dit heeft in de uitvoer een 0 geproduceerd (het bit in S_0), waarna alle bits één plaats zijn opgeschoven en S_3 met de berekende nieuwe waarde is gevuld. Dat heeft geleid tot de nieuwe registertoestand in de kolom ernaast: $S_3 = 0$, $S_2 = 0$, $S_1 = 0$, $S_0 = 1$. De nieuwe waarde van S_3 blijkt een 0 en dat moet dus het resultaat geweest zijn van de terugkoppelfunctie $f(S)$ met de waarden van de

* Elke toestand heeft slechts twee mogelijke opvolgende toestanden. Deze zijn te vinden door een 0 of een 1 ervoor te plaatsen en het rechterbit weg te strepen.

#	S_t	S_{t+1}	$f(S)$	selector voor S_t
0	0000	→ 0000	0	$(S_3+1)(S_2+1)(S_1+1)(S_0+1)$
1	0001	→ 1000	1	$(S_3+1)(S_2+1)(S_1+1)S_0$
2	0010	→ 0001	0	$(S_3+1)(S_2+1)S_1(S_0+1)$
3	0011	→ 0001	0	$(S_3+1)(S_2+1)S_1S_0$
4	0100	→ 0010	0	$(S_3+1)S_2(S_1+1)(S_0+1)$
5	0101	→ 0010	0	$(S_3+1)S_2(S_1+1)S_0$
6	0110	→ 0011	0	$(S_3+1)S_2S_1(S_0+1)$
7	0111	→ 1011	1	$(S_3+1)S_2S_1S_0$
8	1000	→ 1100	1	$S_3(S_2+1)(S_1+1)(S_0+1)$
9	1001	→ 0100	0	$S_3(S_2+1)(S_1+1)S_0$
10	1010	→ 0101	0	$S_3(S_2+1)S_1(S_0+1)$
11	1011	→ 0101	0	$S_3(S_2+1)S_1S_0$
12	1100	→ 1110	1	$S_3S_2(S_1+1)(S_0+1)$
13	1101	→ 1110	1	$S_3S_2(S_1+1)S_0$
14	1110	→ 1111	1	$S_3S_2S_1(S_0+1)$
15	1111	→ 0111	0	$S_3S_2S_1S_0$

Tabel 10.4 Reconstructie terugkoppelfunctie uit toestandstabel

uitgangstoestand daarin ingevuld.

In de meest rechterkolom staan functies van registertrappen. Deze functies dienen om uniek de uitgangstoestand van de betreffende regel te selecteren. Daatoe wordt de waarde van de registertrappen S_3, S_2, S_1, S_0 van de uitgangstoestand ingevuld. Een 0-bit in de uitgangstoestand van het schuifregister wordt geselecteerd door een factor $S_i + 1$ en een 1-bit door een factor S_i . Als deze factoren voor elke

registertrap met elkaar vermenigvuldigd worden komt er slechts dan een 1 uit, als geen van de factoren van de selector een 0 is. En dat is bij elk van de functies in de rechterkolom slechts voor één van alle mogelijke registertoestanden het geval. Elke selectorfunctie filtert dus één bepaalde registertoestand uit. Door nu bij elke toestand de selector te vermenigvuldigen met het bijbehorende functiebit $f(S)$ en de producten op te tellen, wordt de gezochte terugkoppelfunctie gevonden. Dat is dan weliswaar de som van 16 verschillende berekeningen, maar door de bijzondere constructie is die telkens maar voor één enkele uitgangstoestand actief. Voor elke afzonderlijke toestand is er altijd maar één enkel deel van de functie dat de juiste waarde van $f(S)$ aflevert. De gecombineerde functie maakt dus voor alle 16 toestanden van dit register het goede nieuwe bit.

Gemakkelijk is nog dat bij de uitwerking van dit rekenschema de regels met uitvoerbit $f(S) = 0$ kunnen worden weggelaten. Immers 0 keer de selectiefunctie inplannen komt op hetzelfde neer als deze weglaten. Alleen de regels met $f(S) = 1$ worden samengevoegd en uitgewerkt, waarna het resultaat moet worden vereenvoudigd. Let op! Het uitwerken en vereenvoudigen moet zorgvuldig gebeuren want een vergissing is gauw gemaakt. De rekenregels van de exclusive or helpen hierbij, want n maal $S_i \cdots S_j$ is 0 voor n is even en $S_i \cdots S_j$ voor n oneven. Op deze wijze uitgewerkt geeft het voorbeeld het volgende resultaat.

$$\begin{aligned} f(S_0, S_1, S_2, S_3) &= (S_3 + 1)(S_2 + 1)(S_1 + 1)S_0 + (S_3 + 1)S_2S_1S_0 + \\ &\quad S_3(S_2 + 1)(S_1 + 1)(S_0 + 1) + S_3S_2(S_1 + 1)(S_0 + 1) + \\ &\quad S_3S_2(S_1 + 1)S_0 + S_3S_2S_1(S_0 + 1) \\ &= S_0 + S_3 + S_0S_1 + S_0S_2 + S_1S_3 + S_1S_2S_3 \end{aligned}$$

Het werkschema waarmee deze terugkoppelfunctie is bepaald, laat ook beredeneren hoeveel terugkoppelfuncties voor een n -traps register mogelijk zijn. In [tabel 10.4](#) heeft elke regel de betekenis van een registertoestand waaruit de terugkoppelfunctie het volgende bit produceert. In een n -bits register telt deze tabel 2^n regels die met elkaar en met de specifieke reeks bitwaarden in kolom $f(S)$ één terugkoppelfunctie bepalen. Elke specifieke invulling van kolom $f(S)$ geeft een andere terugkoppelfunctie. Omdat elk

bit in kolom $f(S)$ twee waarden kan hebben, zijn er in totaal 2^{2^n} mogelijkheden om de tabel in te vullen en dus ook 2^{2^n} daarmee corresponderende terugkoppelfuncties.

Dit getal kan nog op een andere manier worden berekend. Een terugkoppelfunctie heeft de volgende algemene vorm, de *algebraïsche normaalvorm*

$$f(S) = 1 + S_0 + \dots + S_{n-1} + S_0S_1 + \dots + S_{n-2}S_{n-1} + \dots + S_0S_1\dots S_{n-1} \quad (10.9)$$

Ze bestaat uit een term 1, n termen S_0 t/m S_n , termen bestaande uit twee factoren S_i zoals S_0S_1 , enz. Het aantal termen $S_i\dots S_j$ bestaande uit k factoren is precies gelijk aan het aantal manieren waarop de k indices $i\dots j$ kunnen worden gekozen uit de gehele verzameling $0\dots n-1$, namelijk de waarde van de binomiaalcoëfficiënt van n en k . Het totaal aantal termen is derhalve^[4]

$$\sum_{k=0}^n \binom{n}{k} = 2^n \quad (10.10)$$

Elk van deze termen kan aan- of afwezig zijn, wat het aantal mogelijke functies brengt op 2^{2^n} .

10.2.2 Vertakkingspunten

Vertakkingspunten – zoals in f_1 van [figuur 10.6](#) zo prominent aanwezig – vormen een bron van verspilling. Dat komt omdat de toestanden die zich in een loshangende tak (en dus buiten een cyclus) bevinden ten hoogste éénmaal worden aangedaan. Namelijk alleen dan wanneer het register aldaar of eerder in de vertakking is begonnen. Hoe meer toestanden zich in vertakkingen bevinden, des te minder blijven er over voor de cycli. Omdat opeenvolgende toestanden slechts in het bit zich bevindend in S_0 kunnen verschillen op het moment dat ze gaan samenkomen, kunnen nooit meer dan twee takken tegelijk convergeren. En omdat elke toestand slechts één opvolger heeft, kunnen takken wel samenkomen maar niet meer splitsen. Alle toestanden komen in de een of andere cyclus terecht als men maar zorgt dat er geen vertakkingspunten zijn. De vraag of de cyclus een lange periode heeft kan hier wel gesteld worden, maar het antwoord

wordt uitgesteld tot de behandeling van de lineaire schuifregisters. Functies zonder vertakkingspunten en met de maximale periode van 2^n genereren een zogenaamde DeBruijn-reeks, genoemd naar de Nederlandse wiskundige N.G. de Bruijn^[5]. Er zijn $2^{2^{n-1}-n}$ van zulke functies. Zonder vertakkingen gaan in ieder geval geen toestanden verloren in losse aanloopstukjes. Het is daarom van belang om te zien of er een algemene regel bestaat voor het vermijden van vertakkingen.

Een vertakking is een plaats waar twee verschillende voorgangers eenzelfde opvolger hebben. Het betekent dat ze op dat moment allebei hetzelfde nieuwe bit voor de volgende registertoestand genereren. Voor het overige kunnen deze voorgangers slechts in het ene bit S_0 verschillen, omdat alleen dit in de overgang naar de volgende toestand uit het register verdwijnt. In een vertakkingspunt geldt daarom

$$\left. \begin{array}{l} S_{n-1} \cdots S_1 S_0(t) \\ S_{n-1} \cdots S_1 S'_0(t) \end{array} \right\} \rightarrow S_{n-1} \cdots S_0(t+1) \quad (10.11)$$

De stelling is dat er geen vertakkingen zullen zijn als in de terugkoppelfunctie registertrap S_0 slechts als afzonderlijke term voorkomt.

$$f(S_0 S_1 \cdots S_{n-1}) = S_0 + f'(S_1 \cdots S_{n-1}) \quad (10.12)$$

Het bewijs valt in twee delen uiteen. Eerstens is eenvoudig in te zien dat bij een dergelijke vorm van de functie twee toestanden die alleen in de waarde van S_0 verschillen, wel tot een verschillend functieresultaat moeten leiden. Immers, het deel $f'(n-1 \dots S_1)$ is onafhankelijk van S_0 en heeft daarmee een waarde die niet van S_0 afhangt. Hierbij wordt hetzij $S_0 = 0$, hetzij $S_0 = 1$ opgeteld. Met als gevolg een verschillend resultaat voor het bit dat in S_{n-1} terechtkomt. De twee toestanden die slechts in bit S_0 van elkaar verschillen krijgen aldus een verschillende opvolger zodat ze hier niet kunnen samenlopen, d.w.z. hier kan geen vertakkingspunt zitten. Om het omgekeerde aan te tonen wordt gesteld dat er geen vertakkingspunt is. Dan is dus kennelijk altijd

$$f(S_0 S_1 \cdots S_{n-1}) \neq f(S'_0 S_1 \cdots S_{n-1}), S_0 \neq S'_0, \forall S_1 \cdots S_{n-1} \quad (10.13)$$

Vanwege de ongelijkheid voor alle waarden van $S_{n-1} \cdots S_1$ moet S_0 wel lineair onafhankelijk zijn van $S_{n-1} \cdots S_1$. En dus is f van de vorm als in [uitdrukking \(10.12\)](#).

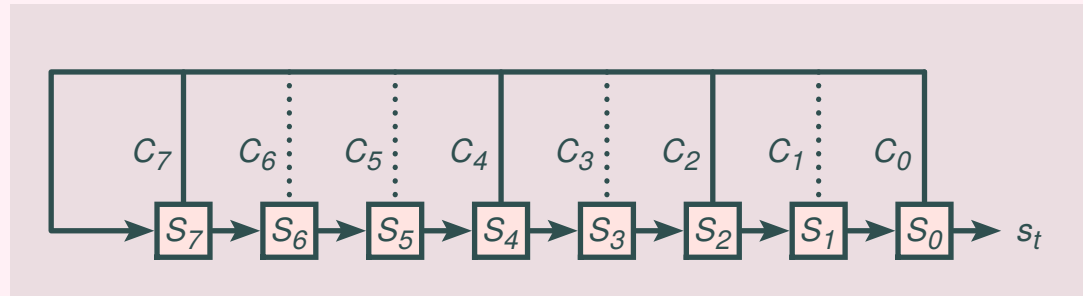
10.3 Lineair schuifregister

De in het voorgaande geïntroduceerde teruggekoppelde schuifregisters kampen met het probleem dat hun eigenschappen over het algemeen moeilijk te voorspellen zijn. Het liefst zou men willen dat alle toestanden zich in één lange cyclus bevinden. Maar hoe valt dat te bereiken? En hoe staat het dan met de kwaliteit van de bitstroom? Zitten de nullen en enen wel goed verdeeld?

Omdat die vragen voor terugkoppelfuncties van algemene vorm lastig of niet te beantwoorden zijn, is uitgekeken naar eenvoudige terugkoppelfuncties met goed te voorspellen eigenschappen welke bovendien bitstromen van goede kwaliteit afleveren. Het zogenaamde *lineair teruggekoppeld schuifregister* (Linear Feedback Shift Register in het Engels) (lfsr) voldoet in dit opzicht uitstekend. Door de lineariteit van de terugkoppelfunctie – d.w.z. de waarden van de deelnemende registertrappen S_i worden alleen bij elkaar opgeteld – kunnen deze schuifregisters goed geanalyseerd en "op maat" gemaakt worden.

Ongelukkigerwijs zal straks blijken dat deze bitstromen niet bestand zijn tegen cryptanalyse. Een klein stukje van de sleutelstroom blijkt al voldoende om het register te reconstrueren. Maar door ze te combineren en door variaties in de constructie is het mogelijk gebleken om goede, wel beter tegen cryptanalyse bestendige systemen te bouwen. Hun toepassing vinden deze systemen onder andere in draagbare telefoons.

In [figuur 10.7](#) is een lineair teruggekoppeld lfsr getekend. Het register bestaat uit 8 trappen $S_7 \cdots S_0$. Met doorgetrokken opstaande lijnen is aangegeven welke registertrappen als aftappunt fungeren en met gebroken lijnen welke niet meedoen. In de figuur doen dus alleen S_0 , S_2 , S_4 en S_7 mee aan de berekening van het nieuwe bit voor S_7 . Vanwege het lineaire karakter van de terugkoppelfunctie kan het nieuwe bit worden uitgerekend door de inhoud van de meegenomen aftappunten met de exclusive or (xor) bij elkaar



Figuur 10.7 8-bits lineair schuifregister

op te tellen. In het tweetalig stelsel waarin hier wordt gewerkt wordt de + in de formules gezien als een xor. Voor het gemak worden daarom al naar het uitkomt deze operatoren doorelkaar gebruikt, vooral in sommaties. De terugkoppelfunctie in dit voorbeeld is dus $f(S) = S_0 \oplus S_2 \oplus S_4 \oplus S_7$. Stel de inhoud van S_0 , S_2 , S_4 , S_7 is respectievelijk 1, 0, 1, 1 dan is het resultaat van de optelling een 1^* .

Om de opbouw van het lineaire schuifregister eenvoudig te kunnen beschrijven, krijgt elk aftappunt S_i een coëfficiënt C_i . Een waarde $C_i = 1$ betekent dat het betreffende aftappunt meedoet in de terugkoppelfunctie, een waarde $C_i = 0$ betekent dat het niet meedoet. In het voorbeeld zijn de aftappunten die niet meedoen degene met een opgaande onderbroken lijn. De coëfficiënten voor de verbindingen in [figuur 10.7](#) zijn dus $C_0 = C_2 = C_4 = C_7 = 1$ en $C_1 = C_3 = C_5 = C_6 = 0$. Het is nu gemakkelijk om dit lfsr in algemene termen te formuleren.

$$f(S) = \sum_{i=0}^{n-1} C_i S_i \quad (10.14)$$

In het vervolg zal worden uitgelegd hoe zo'n register met een algebraïsche functie kan worden beschreven. Uit die functie kunnen de eigenschappen van het register worden afgeleid. Direct duidelijk is dat in de geproduceerde toestandsreeksen geen vertakkingen zullen zitten en dat ze dus uitsluitend complete cycli

* Het gemakkelijkst gaat de berekening door het aantal 1'en te tellen. Is dit oneven dan resulteert een 1, is het even dan een 0.

vormen. Dit kan worden geconstateerd aan het afzonderlijk voorkomen van registertrap S_0 in $f(S)$. Of alle toestanden één cyclus zullen vormen en hoe dit eventueel valt af te dwingen komt in het vervolg aan de orde.

Gemakkelijk valt in te zien dat een lineair schuifregister altijd tenminste twee cycli zal produceren. Stel alle registertrappen bevatten een 0-bit. Wat levert de berekening van het volgende bit op? Ongeacht het aantal registertrappen dat meedoet is dit natuurlijk altijd een 0. Met andere woorden, begintoestand $S = 0 \dots 0$ heeft in een lfsr altijd zichzelf als opvolger*. In een lfsr produceert de nul-toestand altijd een aparte cyclus met periode 1. De overige begintoestanden produceren een of meer cycli.

Van de 2^n mogelijke toestanden van het register staat de nul-toestand dus apart, maar alle overige toestanden kunnen bij geschikt gekozen terugkoppelfunctie wel één grote cyclus vormen. De langst mogelijke cyclus die een n -bits lineair teruggekoppeld schuifregister kan maken heeft zodoende $2^n - 1$ als periode. Zo'n rij heet daarom een *maximaalrij* (m-sequence in de engelstalige literatuur). Men komt ook wel de benaming PN-sequence tegen (PN staat voor Pseudo Noise).

Tenslotte wordt de aandacht nog gevestigd op het feit dat het over het algemeen weinig nut heeft om C_0 een andere waarde te geven dan 1. Immers voor $C_0 = 0$ doet S_0 in het geheel niet mee en neemt S_1 effectief de rol van S_0 over. In feite komt dit neer op het verkorten van het register van n naar $n - 1$ bits (of nog verder wanneer ook $C_1 = 0$, enz.). Het betekent slechts het inbouwen van een vertragingstraject voordat het bit van de sleutelstroom naar buiten komt. In de praktijk een verspilling van hardware die nergens toe dient. Daarom vinden we voor C_0 altijd de waarde 1.

Met de algemene uitdrukking voor $f(S)$ uit [vergelijking \(10.14\)](#) kan uit de geproduceerde bitstroom voor elk gewenst tijdstip t de waarde van een daarin voorkomend bit uitgerekend worden. Op tijdstip t is de bitstroom

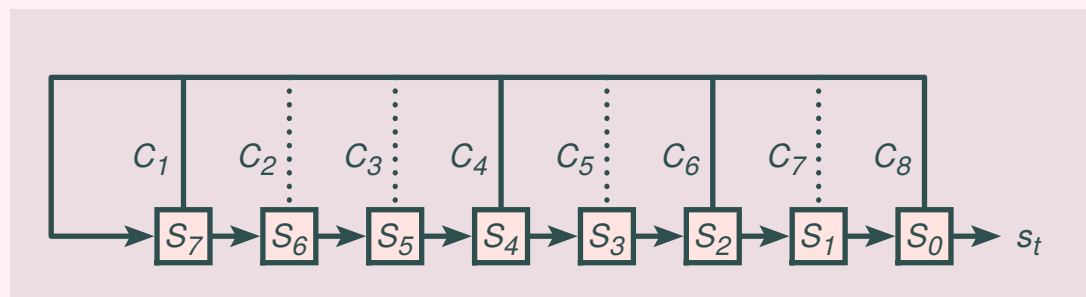
$$\dots S_{t-n} S_{t-n+1} S_{t-n+2} \dots S_{t-2} S_{t-1} S_t \dots \quad (10.15)$$

* In het algemene niet-lineaire schuifregister ligt dit anders en kan de nul-toestand wel degelijk een andere opvolger hebben.

Daarin valt s_t te berekenen uit

$$s_t = C_{n-1}s_{t-1} + C_{n-2}s_{t-2} + \dots + C_0s_{t-n} = \sum_{i=1}^n C_{n-i}s_{t-i} \quad (10.16)$$

Let op, hoewel dit veel lijkt op de uitdrukking in [vergelijking \(10.14\)](#) staan hier s 'en, dat wil zeggen bits uit de geproduceerde stroom en *niet* registertrappen S .



Figuur 10.8 Lineair schuifregister met alternatieve nummering

In de literatuur komt nog een andere nummering van de registercoëfficiënten voor. Dat kan gemakkelijk tot verwarring aanleiding geven. In [figuur 10.8](#) is deze alternatieve toekenning gegeven voor het voorbeeld van [figuur 10.7](#). De C 's zijn in omgekeerde richting genummerd en lopen van 1 tot en met n . Hiermee wordt de met [vergelijking \(10.14\)](#) overeenkomstige berekening van het nieuwe bit

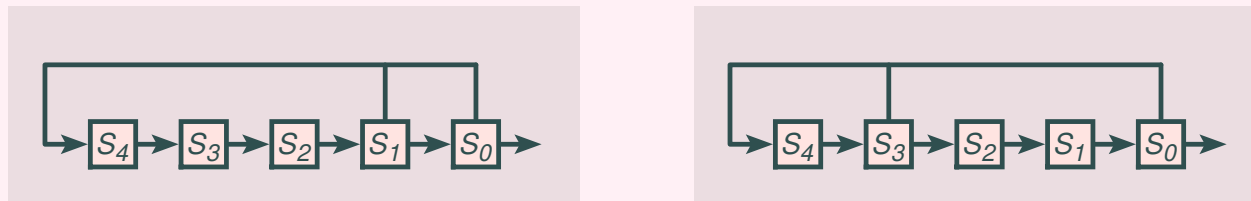
$$f(S) = \sum_{i=1}^n C_i S_{n-i} \quad (10.17)$$

De berekening van s_t loopt analoog aan [vergelijking \(10.16\)](#) en luidt dan

$$s_t = C_1s_{t-1} + C_2s_{t-2} + \dots + C_ns_{t-n} = \sum_{i=1}^n C_is_{t-i} \quad (10.18)$$

10.3.1 Periodiciteit

Alvorens over te gaan tot het afleiden van de cyclusstructuur van het lineaire schuifregister, eerst een paar voorbeelden. Een 5-trapsregister is klein genoeg als voorbeeld en groot genoeg om een indruk te geven. In [figuur 10.9](#) staan twee exemplaren met in [tabel 10.5](#) de achtereenvolgende toestanden en de uitvoerbits. Het register links in de figuur heeft behalve de altijd aanwezige nul-cyclus drie afzonderlijke cycli met respectievelijk periode 3, 7 en 21, het andere register bezit de voor een 5-bits lfsr langst mogelijke periode van 31 bits. In beide registers bedraagt de som van de lengtes van alle niet-nul periodes vanzelfsprekend $2^5 - 1 = 31$.



Figuur 10.9 $f = S_0 \oplus S_1$ (links) en $f = S_0 \oplus S_3$ (rechts)

Waar toe kan het verschil in gedrag tussen beide terugkoppelfuncties herleid worden? Wat maakt van het ene register een maximaalrij, terwijl in het andere verscheidene korte perioden ontstaan? Om dit in te zien wordt de *genererende functie* $G(x)$ te hulp geroepen. Deze is een tot in het oneindige doorlopende machtreeks van termen x^t met x uit $\{0,1\}$ en met de waarden van de bitstroom als coëfficiënten

$$G(x) = \sum_{t=0}^{\infty} s_t x^t \quad (10.19)$$

Elke term in de sommatie van $G(x)$ kan worden uitgerekend met behulp van [vergelijking \(10.16\)](#). Invullen voor s_t en uitwerken met verwisselen van de sommatievolgorde geeft dan

t	$f_1 = S_0 \oplus S_1$				$f_2 = S_0 \oplus S_3$	
1	00000	01101	00101	00001	00000	00001
2		10110	10010	10000		10000
3		11011	11001	01000		01000
4			11100	00100		10100
5			01110	00010		01010
6			10111	10001		10101
7			01011	11000		11010
8				01100		11101
9				00110		01110
10				10011		10111
11				01001		11011
12				10100		01101
13				01010		00110
14				10101		00011
15				11010		10001
16				11101		11000
17				11110		11100
18				11111		11110
19				01111		11111
20				00111		01111
21				00011		00111
22						10011
23						11001
24						01100
25						10110
26						01011
27						00101
28						10010
29						01001
30						00100
31						00010

Tabel 10.5 Cyclusstructuur van $f_1 = S_0 \oplus S_1$ en $f_2 = S_0 \oplus S_3$

$$\begin{aligned}
G(x) &= \sum_{t=0}^{\infty} x^t \sum_{i=1}^n C_{n-i} s_{t-i} \\
&= \sum_{i=1}^n C_{n-i} x^i \sum_{t=0}^{\infty} s_{t-i} x^{t-i} \\
&= \sum_{i=1}^n C_{n-i} x^i \{s_{-i} x^{-i} + \dots + s_{-1} x^{-1} + \dots + s_t x^t + \dots\}
\end{aligned} \tag{10.20}$$

Verrassend duiken binnen de accolades de in $G(x)$ gesommeerde termen weer op. Voor het deel $(s_0 x^0 + \dots)$ wordt $G(x)$ teruggesubstitueerd, zodat een vergelijking in $G(x)$ ontstaat waaruit deze kan worden opgelost

$$\begin{aligned}
G(x) &= \sum_{i=1}^n C_{n-i} x^i \{s_{-i} x^{-i} + \dots + s_{-1} x^{-1} + G(x)\} \\
&= \frac{\sum_{i=1}^n C_{n-i} x^i \{s_{-i} x^{-i} + \dots + s_{-1} x^{-1}\}}{1 - \sum_{i=1}^n C_{n-i} x^i}
\end{aligned} \tag{10.21}$$

Omdat* $1 = C_n x^0$ en omdat plus en min vanwege x uit $\{0,1\}$ vrijelijk mogen worden verwisseld kan de noemer worden vereenvoudigd en komt er tenslotte

$$G(x) = \frac{\sum_{i=1}^n C_{n-i} x^i \{s_{-i} x^{-i} + \dots + s_{-1} x^{-1}\}}{\sum_{i=0}^n C_{n-i} x^i} \tag{10.22}$$

In de gevonden uitdrukking voor $G(x)$ staat in de teller van de breuk een som waarin de registertoestand op tijdstip $t = 0$ voorkomt (de toestand van het register wanneer de bitstroom gaat lopen). Op het moment dat s_0 door de terugkoppelfunctie gemaakt wordt, bevinden $s_{-n} \dots s_{-1}$ – de bits binnen de accolades die aan s_0 voorafgaan – zich als volgt in het register: $S_{n-1} = s_{-1}$, $S_{n-2} = s_{-2}$, ... tot en met $S_0 = s_{-n}$. Het is daarmee duidelijk dat $G(x)$ voor elke gekozen begintoestand van het lfsr de bitstroom representeert, die

* Deze C_n hier correspondeert met de verbinding tussen de terugkoppelfunctie en de meest linker registertrap in de afgebeelde schuifregisters.

vervolgens daaruit ontstaat.

De meest eenvoudige vorm krijgt de teller bij starten met $S_0 = 1$, $S_1 = \dots = S_{n-1} = 0$, anders gezegd met begintoestand $0\dots 01$. Bij het uitwerken van de teller vallen dan alle termen weg, behalve de eerste term tussen accolades en die alleen als $i = n$. Aldus resteert van de gehele sommatie nog slechts $C_0 = 1$

$$S_{n-1}\dots S_0 = s_{-1}\dots s_{-n} = 00\dots 01 \rightarrow \sum_{i=1}^n C_{n-i}x^i \{s_{-i}x^{-i} + \dots + s_{-1}x^{-1}\} = C_0x^n s_{-n}x^{-n} \rightarrow C_0 = 1 \quad (10.23)$$

Na de uitwerking van de teller vervolgens aandacht voor de noemer. Opgemerkt kan worden dat hierin geen s_i 's voorkomen, zodat de noemer onafhankelijk is van de inhoud van de registertrappen en ze is derhalve een *constante* voor het lfsr. De noemer is een polynoom in x van graad n blijkens de grenzen van de sommatie-index gecombineerd met de eerdere vaststelling dat altijd $C_0 = 1$. Merk ook op dat de teller hoogstens van de graad $n - 1$ kan zijn (vanwege het maximale product $x^n x^{-1}$ bij $i = n$). Het heeft daarom zin $G(x)$ te schrijven als het quotiënt van twee polynomen.

$$G(x) = \frac{s(x)}{f(x)} \text{ en voor } s(x) = 00\dots 01 \text{ wordt dit } \frac{1}{f(x)} \quad (10.24)$$

De functie $f(x) = \sum_{i=0}^n C_{n-i}x^i$ in de noemer is een belangrijk kenmerk voor het lfsr. Deze polynoom is *karakteristiek* voor de geproduceerde bitstream. Om het verband tussen $f(x)$ en de aftappunten van het lfsr te zien, wordt de sommatie in $f(x)$ geschreven met de index naar beneden lopend

$$f(x) = \sum_{i=0}^n C_{n-i}x^i = C_0x^n + C_1x^{n-1} + \dots + C_{n-1}x^1 + C_nx^0 \quad (10.25)$$

Als de producten $C_{n-i}x^i$ in de sommatie vergeleken worden met de coëfficiënten en registertrappen in de alternatieve nummering van **figuur 10.8**, valt direct de bijna volledige overeenkomst op: de machten van x corresponderen met de indices van de registertrappen. Toevoegen van C_0 aan de linkerkant in de figuur bij de toevoer naar registertrap S_{n-1} maakt de overeenkomst compleet.

Ook met de nummering van **figuur 10.7** bestaat overeenstemming. Deze wordt gevonden in de *reciproke polynoom* van $f(x)$. Deze reciproke polynoom is gedefinieerd door

$$f^*(x) = x^n f\left(\frac{1}{x}\right) = C_0 x^0 + C_1 x^1 + \dots + C_{n-1} x^{n-1} + C_n x^n \quad (10.26)$$

Nu moet aan de linkerkant een C_n toevoegd worden aan de toevoer naar S_{n-1} . Vanwege $C_0 = C_n = 1$ en $x^0 = 1$ wordt de functie gewoonlijk gepresenteerd als

$$f(x) = 1 + C_1 x + C_2 x^2 + \dots + C_{n-1} x^{n-1} + x^n \quad (10.27)$$

De twee lfsr's uit **figuur 10.9** kunnen zodoende worden beschreven met respectievelijk $f_1 = 1 + x + x^5$ voor $f_1 = S_0 \oplus S_1$ en $f_2 = 1 + x^3 + x^5$ voor $f_2 = S_0 \oplus S_3$.

Polynoom $f(x)$ en diens reciproke polynoom $f^*(x)$ vervullen in feite dezelfde rol bij het karakteriseren van een lfsr, want een verschil in benaming kan daarop geen effect hebben. Afhankelijk van het gekozen nummerschema wordt daarom de ene of de andere polynoom de *karakteristieke functie* van het schuifregister genoemd. Er ligt een direct verband tussen de aard van deze twee verwante polynomen en de cyclusstructuur.

Voor het leggen van verband tussen de periode van het lfsr en de karakteristieke polynoom wordt opnieuw uitgegaan van $G(x)$ maar de termen van **vergelijking (10.19)** worden nu anders uitgewerkt. Deze worden zodanig gegroepeerd dat daaruit een periode blijkt. Periodiek met periode p betekent $s_i = s_{i+k.p}$ voor alle gehele k en $0 \leq i < p$. Hiervan gebruikmakend

$$G(x) = \sum_{t=0}^{\infty} s_t x^t = \sum_{k=0}^{\infty} \sum_{i=0}^{p-1} s_{i+k.p} x^{i+k.p} \quad (10.28)$$

Door uitschrijven en de termen anders te groeperen kan een complete periode uitgefactoriseerd worden

$$G(x) = (s_0 + s_1 x + s_2 x^2 + \dots + s_{p-1} x^{p-1})(1 + x^p + x^{2p} + x^{3p} + \dots) \quad (10.29)$$

De tot in het oneindige doorlopende som van machten rechts kan worden vervangen door het quotiënt $1/(1 - x^p)$. Dat blijkt bij uitdelen van $1/(1 - x^p)$ in een staartdeling, waarbij de som al snel te voorschijn komt. Zodat tenslotte

$$G(x) = \frac{s_0 + s_1x + s_2x^2 + \dots s_{p-1}x^{p-1}}{1 - x^p} \quad (10.30)$$

Dit resultaat wordt gelegd naast de vorige uitwerking van $G(x)$, gepresenteerd in [vergelijking \(10.24\)](#). Gebruikmaken van $1/f(x)$ in plaats van de meer algemene uitdrukking $s(x)/f(x)$ kan zonder bezwaar, omdat toch gezocht wordt naar een register met maximale periode. Dan is de vooronderstelling van begintoestand $0\dots 01$ valide (is altijd inbegrepen en mag dus ook altijd als startpunt worden genomen). Gelijkestelling en overbrengen van de noemers naar de andere kant geeft

$$f(x)(s_0 + s_1x + s_2x^2 + \dots s_{p-1}x^{p-1}) = 1 - x^p \quad (10.31)$$

Als de bitstroom een periode p heeft betekent dit dat $f(x)$ een deler is van functie $(1 - x^p)$. Het zegt ook dat het kleinste gehele getal p , waarvoor geldt dat $f(x)$ een deler is van $(1 - x^p)$, een periode is van de bitstroom gegenereerd door het lfsr met karakteristieke functie $f(x)$. Men noemt in dit verband de waarde van p ook wel de *exponent* van $f(x)$. Als p de waarde $2^n - 1$ heeft, genereert $f(x)$ een maximaalrij.

Gezocht wordt naar lfsr's met maximale periode en hier zal worden aangetoond dat in dat geval $f(x)$ aan een zeer bepaalde eis moet voldoen: $f(x)$ moet een zogenaamde *primitief polynoom* zijn. Dit blijkt zowel een noodzakelijke als een voldoende voorwaarde. Een polynoom is primitief wanneer deze aan de volgende eisen voldoet

- *irreducibel*, hetgeen wil zeggen niet geschreven kunnen worden als een product van functies, en
- zijnde van graad n , exponent $2^n - 1$ hebben en daarmee voor waarden $p < 2^n - 1$ geen deler van $(1 - x^p)$ zijn.

Om aan te tonen dat een maximaalrij irreducibiliteit van $f(x)$ impliceert wordt het tegendeel gesteld. Stel functie $f(x)$ is een product maar genereert toch een maximaalrij. Dan kan $f(x)$ dus minstens geschreven

worden als het product $f(x) = g(x)h(x)$. Nu is in [vergelijking \(10.24\)](#) al gebleken dat $1/f(x)$ mag worden opgevat als de genererende functie van een bitstream. Als gevolg van de veronderstelde samengesteldheid van $f(x)$ kan deze functie als volgt opgesplitst worden

$$\frac{1}{f(x)} = \frac{a(x)}{g(x)} + \frac{b(x)}{h(x)} \quad (10.32)$$

Hoe moet die splitsing worden geïnterpreteerd? Wel het is natuurlijk een combinatie van twee bitstromen, elk met een periode afgeleid van respectievelijk $g(x)$ en $h(x)$. In het gunstigste geval produceren die allebei een maximaalrij. Neem dat aan, dan zijn voor $g(x)$ met graad n_g en $h(x)$ met graad n_h de maximale periodes respectievelijk $2^{n_g} - 1$ en $2^{n_h} - 1$. Voor graad n_f van $f(x)$ moet natuurlijk gelden $n_f = n_g + n_h$. De maximaal bereikbare periode bij combinatie van de twee bitstromen is het kleinste gemene veelvoud van de samenstellende periodes. Zodat de samengesteldheid van $f(x)$ leidt tot de conclusie dat diens periode hoogstens gelijk kan zijn aan het product van de deelperiodes

$$\begin{aligned} p &\leq \text{kgv}(2^{n_g} - 1, 2^{n_h} - 1) \leq 2^{n_g+n_h} - 2^{n_g} - 2^{n_h} + 1 \\ p &\leq 2^{n_g+n_h} - 3 = 2^n - 3 \quad \text{wegens } n_g, n_h \geq 1 \\ p &< 2^n - 1 \end{aligned} \quad (10.33)$$

Hieruit blijkt dat samengesteldheid van $f(x)$ altijd leidt tot een periode kleiner dan $2^n - 1$. Waardoor een tegenspraak ontstaat met het gestelde dat $f(x)$ een maximaalrij zou voortbrengen. Ergo $f(x)$ kan niet samengesteld zijn en is dus irreducibel. Omdat periode $2^n - 1$ tevens inhoudt dat $f(x)$ exponent $2^n - 1$ zal hebben, is aldus aangetoond dat een maximaalrij een primitieve functie verlangt. Tot slot nog het volgende. Als $f(x)$ een primitief polynoom is, houdt dit in dat de reciproke polynoom $f^*(x)$ eveneens primitief is. Zodat met elk maximaal lfsr een tweede, corresponderend maximaal lfsr gegeven is*.

Twee vervolgvragen dringen zich nog op. Zijn er genoeg primitieve polynomen en hoe kunnen ze worden gevonden? Beantwoording van die vragen valt buiten het bestek van deze tekst, maar kan worden gevon-

* Behalve natuurlijk als $f^*(x) = f(x)$ zoals in het geval van bijvoorbeeld $1 + x + x^2$.

den in boeken over algebra en getaltheorie. Hier wordt volstaan met het geven van het antwoord. Het aantal primitieve functies van graad n is $\phi(2^n - 1)/n$ met ϕ de Euler totiënt functie.

$1 + x + x^2$	$1 + x^2 + x^5$	$1 + x + x^6$
$1 + x + x^3$	$1 + x^3 + x^5$	$1 + x^5 + x^6$
$1 + x + x^4$	$1 + x^2 + x^3 + x^4 + x^5$	$1 + x + x^2 + x^5 + x^6$
$1 + x + x^6$	$1 + x + x^2 + x^3 + x^5$	$1 + x + x^4 + x^5 + x^6$
$1 + x + x^7$	$1 + x + x^2 + x^4 + x^5$	$1 + x^2 + x^3 + x^5 + x^6$
$1 + x + x^{15}$	$1 + x + x^3 + x^4 + x^5$	$1 + x + x^3 + x^4 + x^6$

Tabel 10.6 Enkele primitieve polynomen

In [tabel 10.6](#) is als voorbeeld een aantal primitieve polynomen van lage graad bijeengebracht. Te klein voor praktisch gebruik maar wel geschikt om mee te experimenteren. Meer primitieve functies kunnen gevonden worden in boeken van Golomb en van Schneier^{[6],[7]}. Een enkele waarschuwing is nog op z'n plaats. Vanwege het geringe rekenwerk lijken polynomen met weinig tappunten, zoals $1 + x + x^{127}$ misschien wel handig. Maar gebleken is dat die nu juist het meest kwetsbaar zijn voor cryptoanalyse en daarom beter vermeden kunnen worden.

10.3.2 Kwaliteit

Nu behandeld is hoe met behulp van een lineair schuifregister langperiodieke bitstromen gemaakt kunnen worden, is het zaak na te gaan of deze stromen kwalitatief voldoende zijn om voor stroomvercijfering te dienen. Eerst zal gekeken worden naar de mate waarin deze rijen voldoen aan de criteria van Golomb en vervolgens naar hun bestendigheid tegen cryptoanalyse. De door Golomb geformuleerde criteria zijn in het voorgaande geïntroduceerd ([sectie 10.1.2](#)). Vereist zijn gelijke aantallen nullen en enen, gelijkmatige

verdeling van rijtjes enen en nullen en constante autocorrelatie. Deze eisen moeten stuk voor stuk worden gecontroleerd voor het lfsr. Aangezien vrijwel altijd gewerkt wordt met lfsr's die een maximaalrij produceren, zal de behandeling zich tot die gevallen beperken.

1. *Aantal nullen en enen*

In een maximaalrij doorloopt het lfsr $2^n - 1$ toestanden, slechts de toestand met alleen maar nullen in het register ontbreekt. Dit betekent dat het register in een maximaalrij precies één keer alle bitpatronen behalve 00...0 aanneemt. Elk bit in de stroom komt uit S_0 , het rechterbit van elke registertoestand. Van alle 2^n bitpatronen heeft de helft een 0 in het rechterbit, de andere helft een 1. Bij de nullen doet er één niet mee vanwege de nulperiode en zodoende telt de stroom dus $2^{n-1} - 1$ nullen en 2^{n-1} enen. Een minimaal verschil, maar beter is niet mogelijk bij een periode van oneven lengte. De maximaalrij voldoet dus zo goed als maar kan aan het eerste criterium.

2. *Verdeling rijtjes enen en nullen*

Opnieuw wordt het feit gebruikt dat op één na alle bitpatronen bij de productie van de rij één keer voorkomen. Nu ligt de focus op runs van enen en nullen. Een blok 111 van drie enen is dat slechts bij de gratie van het feit dat in 01110 een nul voorafgaat en een nul volgt. Anders zouden er immers meer enen staan. Hieruit volgt dat met een run van k bits meteen $k + 2$ bits vastliggen. Omdat het register n bits telt zijn er zodoende 2^{n-k-2} verschillende runs van enen (blokken) en ook 2^{n-k-2} runs van nullen (gaten). Dit verhaal gaat op tot en met $k = n - 2$. Daarenboven is er nog één gat van $n - 1$ nullen (de nultoestand doet niet mee) en één blok van n enen. Er is geen blok van $n - 1$ enen, maar de argumentatie daarvoor is een beetje tricky*.

Er kan dus al worden vastgesteld dat het aantal gaten en blokken aan elkaar gelijk is. Nu valt nog aan te tonen dat het aantal runs afneemt met toenemende lengte. Om in te zien dat ook hieraan is voldaan wordt het aandeel van de runs van lengte k op het totaal aantal runs berekend. Voor de

* Wie het naadje van de kous wil weten kan terecht bij het in de inleiding genoemde boek van Beker en Piper uit 1982, p.196.

berekening van dit totaal moeten 2^{n-k-2} gaten en 2^{n-k-2} blokken worden gesommeerd voor elke k van 1 tot en met $n-2$ plus het ene lange gat en het dito blok. Het aandeel van de blokken en gaten van lengte k blijkt dan inderdaad omgekeerd evenredig met de lengte, namelijk $1/2^k$

$$\frac{2 \cdot 2^{n-k-2}}{2 + 2 \sum_{i=1}^{n-2} 2^{n-k-2}} = \frac{2^{n-k-1}}{2^{n-1}} = \frac{1}{2^k} \quad (10.34)$$

3. Uitfase-autocorrelatie

De uitfase-autocorrelatie wordt berekend met **vergelijking (10.3)**. De sommatie in die uitdrukking kan worden opgevat als het bitsgewijs samennemen van twee over een afstand δ verschoven bitrijen, waarna de samengenomen bitparen worden getotaliseerd. Maar wat gebeurt er eigenlijk in die bitsgewijze optelling? Bekijk dit met de maximaalrij van register $f = 1 + x + x^4$ als voorbeeld. Rij en verschoven rij zijn onder elkaar gelegd en bit voor bit samengenomen. De somrij blijkt een verschoven kopie van de oorspronkelijke rij te zijn.

$$\begin{aligned} 100010011010111 &= a \\ 100110101111000 &= b \\ 000100110101111 &= a \oplus b \end{aligned} \quad (10.35)$$

Dit geldt algemeen voor een maximaalrij. Het kan ook anders worden gezien: elke begintoestand van een maximaal lfsr produceert in beginsel dezelfde rij, maar wel steeds in een andere fase. Door combineren van twee rijen met verschillende begintoestand ontstaat steeds een rij die ook uit een andere begintoestand gemaakt had kunnen worden. Gewapend met dit inzicht is het niet moeilijk meer om de sommatie uit te voeren. De uitkomst is natuurlijk gelijk aan het aantal enen in de rij en al aangetoond is dat dit er 2^{n-1} zijn. Bovendien is dit resultaat overduidelijk voor elke $\delta \neq 0$ hetzelfde. Zodoende is de uitfase-autocorrelatie inderdaad een constante met als waarde

$$C(\delta) = 1 - \frac{2}{2^n - 1} 2^{n-1} = \frac{-1}{2^n - 1} \quad (10.36)$$

10.3.3 Cryptografische sterkte

De producten van een lfsr met welgekozen karakteristieke functie voldoen aan de kwaliteitseisen van Golomb, maar de bestendigheid tegen cryptoanalyse is een ander chapter. Wanneer de karakteristieke functie bekend is, blijkt het kennen van n opeenvolgende bits voldoende om de gehele periode te kunnen uitrekenen. Logisch, want die n opeenvolgende bits geven precies de inhoud van de registertoestand op het moment dat ze allemaal in het lfsr zaten. Daaruit kan met behulp van de karakteristieke functie het volgende bit worden berekend, enz.

Maar ook als de karakteristieke functie van het register onbekend is, kan deze toch uit heel weinig bits worden bepaald. Slechts $2n$ opeenvolgende bits zijn al voldoende, zoals ook eenvoudig kan worden ingezien. De eerste n bits worden gecombineerd met de n onbekende coëfficiënten $C_0 \dots C_{n-1}$ van de karakteristieke functie*. Die moeten het volgende bit opleveren, maar zijn nu onbekend zodat een lineaire vergelijking in de n onbekende C 's wordt verkregen. De uitkomst van die vergelijking is bekend, namelijk het volgende bit. Door dit kunstje te herhalen met steeds een bit verder, kan men uit $2n$ opeenvolgende bits een stelsel van n lineaire vergelijkingen in n onbekenden vormen. Dat stelsel is beslist oplosbaar, want het gezochte – en dus bestaande – lfsr moet eruit komen. Oplossing van het stelsel vergelijkingen levert de karakteristieke functie van het lfsr, waarna alle voorgaande en volgende bits kunnen worden uitgerekend.

In de vorm van een matrixvergelijking wordt het lfsr gebroken door het oplossen van het stelsel

$$\begin{pmatrix} s_1 & s_2 & \dots & s_n \\ s_2 & s_3 & \dots & s_{n+1} \\ \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ s_n & s_{n+1} & \dots & s_{2n-1} \end{pmatrix} \begin{pmatrix} C_0 \\ C_1 \\ \cdot \\ \cdot \\ C_{n-1} \end{pmatrix} = \begin{pmatrix} s_{n+1} \\ s_{n+2} \\ \cdot \\ \cdot \\ s_{2n} \end{pmatrix} \quad (10.37)$$

* Omdat als regel $C_0 = 1$ is het zelfs nog eentje minder.

10.4 Niet-lineair schuifregister

Lineaire schuifregisters zijn in het voorgaande uitstekende leveranciers van bitrijen gebleken, maar helaas cryptografisch zwak. Introductie van niet-lineariteit is noodzakelijk om cryptografisch aanvaardbare bitrijen te maken. De verdere behandeling van schuifregisters zal draaien om de manier waarop dit kan worden bereikt met lineaire schuifregisters als bouwstenen.

Er is nogal wat variatie in de methoden waarmee niet-lineariteit kan worden geïntroduceerd en de belangrijkste ervan zullen de revue passeren

- randomselectie van een bit uit een van de registertrappen
- combinatie van bits uit een register
- combinatie van de uitvoer van verschillende registers
- combinatie van registers met voorgaande uitvoer
- onregelmatige loop van registers

De lineaire schuifregisters die als componenten van het niet-lineaire systeem de bitstroom genereren, worden met S_i aangeduid. In een niet-lineair systeem staat S_i voor een door lfsr- i geproduceerde deelstroom. Dat kan verwarring wekken met het gebruik van S_i als aanduiding voor registertrap- i binnen een lfsr; houdt die twee dus goed uit elkaar.

De issues bij het maken van bruikbare niet-lineaire bitstromen liggen in het verlengde van het zojuist behandelde: periode zo lang als mogelijk gegeven de afmetingen van de samenstellende lfsr's, goede kwaliteit in termen van de Golomb criteria, en natuurlijk een behoorlijke weerstand tegen cryptoanalyse.

Ten aanzien van de periode zal het begrip *lineaire complexiteit* geïntroduceerd worden. Daarmee beoogt men een kwaliteitscriterium voor de geproduceerde bitstroom te geven. De lineaire complexiteit kan in beperkte mate gebruikt worden als een maatstaf voor de veiligheid.

Weerstand tegen cryptoanalyse brengt ook een nieuwe aanvalstechniek in het spel: de *correlatieaanval*. Bij dat soort aanvallen is het doel om uit een complexe sleutelbitstroom de bijdragen van de afzonderlijke lfsr's te bepalen. Verderop in de behandeling zullen daarvan enkele voorbeelden worden gegeven.

10.4.1 Lineaire complexiteit

Nu de lineair teruggekoppelde schuifregisters zijn geïntroduceerd is er de gelegenheid om iets te zeggen over de complexiteit van bitrijen. Het blijkt dat voor elke bitrij van eindige lengte een kleinste lfsr kan worden gevonden dat vanuit een bepaalde begintoestand precies die rij als beginstuk van z'n eigen bitstroom genereert. Intuïtief is een bitrij *complexer* naarmate dat kleinste lfsr groter is. De lengte van dit lfsr wordt de *lineaire complexiteit* van de rij genoemd en het register zelf heet wel het *lineair equivalent*. De lineaire complexiteit van een echte random bitrij met n bits blijkt gemiddeld iets groter dan $n/2$. Voor een uitvoeriger behandeling van dit complexiteitsbegrip kan worden verwezen naar het boek van Rueppel^[8].

Tussen de lineaire complexiteit en de wijze waarop de samenstellende deelstromen gecombineerd worden bestaat een verband. Door meer deelstromen samen te nemen valt een hoger lineair equivalent (en dus een complexere bitstroom) te bereiken. Helaas is de keerzijde daarvan een verminderde weerstand tegen de zogenaamde correlatieaanval, zoals in het vervolg zal worden gedemonstreerd. Daarmee wordt een goed ontworpen niet-lineair systeem een compromis tussen enerzijds een hoog lineair equivalent en anderzijds een goede weerstand tegen cryptoanalyse.

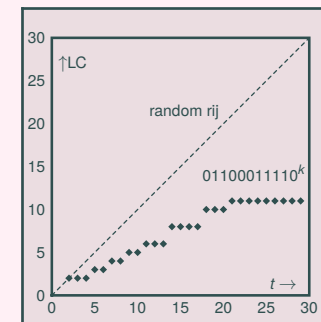
Voor de bepaling van de lineaire complexiteit moet het equivalente lfsr voor de geproduceerde bitrij worden berekend. Een bekend algoritme hiervoor is de *Berlekamp-Massey algoritme*. Het idee achter deze algoritme is om een register dat het begin van de rij reproduceert te laten werken totdat het gefabriceerde bit niet meer deugt. Het register wordt dan zodanig aangepast dat ook dit bit er weer goed uitkomt. Omdat de voorgaande bits allemaal wel goed waren, zal het aangepaste register meer trappen tellen. Men gaat hiermee door totdat de gehele rij kan worden gereproduceerd. Het aantal trappen van het verkregen lfsr

is een *bovengrens* voor de lineaire complexiteit van de bitrij. Het kan zijn dat de lineaire complexiteit in werkelijkheid lager is, maar ze is zeker niet groter. Opstarten kan met een eenvoudig klein lfsr en na evenzoveel stappen als er bits in de rij staan is de algoritme klaar en is het gezochte lfsr gevonden. Wie het allemaal precies wil nazien of het zelf ook kunnen uitvoeren wordt verwezen naar de literatuur^[9].

Als voorbeeld wordt de lineaire complexiteit van een bitrij bepaald. Deze bitrij bestaat uit een voorloop van de twee bits 01 gevolgd door een periodieke herhaling van de negen bits 100011110. Deze rij is geproduceerd door een 4-bits (nietlineair) schuifregister met terugkoppelfunctie

$$f(S_0, S_1, S_2, S_3) = S_0 + S_3 + S_0S_1 + S_0S_2 + S_1S_3 + S_1S_2S_3$$

bitrij	LC = $f(x)$
100011110	$1 + x^3 + x^4$
$01\{100011110\}^1$	$1 + x^2 + x^3 + x^6$
$01\{100011110\}^2$	$1 + x + x^2 + x^3 + x^8 + x^9 + x^{10}$
$01\{100011110\}^3$	$x^2 + x^{11}$
$01\{100011110\}^4$	$x^2 + x^{11}$
...	...
$01\{100011110\}^n$	$x^2 + x^{11}$



Tabel 10.7 Lineaire complexiteit van $01\{100011110\}^n$

De tabel links in **figuur 10.7** is het resultaat van de Berlekamp-Massey algoritme toegepast op verschillende uit dit voorbeeld afgeleide bitrijen. Het periodieke deel blijkt door een 4-bits lfsr $f(x) = 1 + x^3 + x^4$ te kunnen worden gemaakt. Vervolgens is de voorloop 01 gecombineerd met achtereenvolgens 1, 2, 3, ... cycli van het periodieke stuk. Na een paar perioden blijkt steeds hetzelfde lfsr in staat om de volledige rij te produceren. Dat is een 11-bits register met $C_0 = C_1 = 0$, waardoor S_1S_0 de voorloop 01 kunnen bevatten, terwijl de overige negen bits van het register steeds dezelfde cyclus 100011110 herhalen. Uit het convergeren van het lineair equivalent op de waarde 11 blijkt de lage lineaire complexiteit van deze



Berlekamp



Massey

bitrij. In de bijbehorende grafiek rechts wordt dat zichtbaar in het horizontaal lopen van het *lineaire complexiteitsprofiel* vanaf $LC = 11$. Dit complexiteitsprofiel bestaat uit de opeenvolgende waarden van de lineaire complexiteit tijdens de uitvoering van de Berlekamp-Massey algoritme. Een echte random rij (de streeplijn) bereikt met toenemende lengte een steeds hogere waarde. Deze bitrij blijft daar over de gehele linie duidelijk onder.

Het blijkt overigens dat de Berlekamp-Massey algoritme hier het kortst mogelijke lfsr niet vindt. De beste oplossing is een constructie met twee losse registertrappen S_0 en S_1 voor de voorloop en $f(x) = 1 + x^3 + x^4$ voor het cyclische deel; dat geeft als functie $f(x) = x^2 + x^5 + x^6$. Het is niet verwonderlijk dat de algoritme deze oplossing mist, omdat al vanaf de eerste stap rekening gehouden moet worden met de twee bits van de voorloop. De algoritme start echter met het inpassen van de bitstroom in een lfsr met $S_0 = 1$ en raakt dat niet meer kwijt.

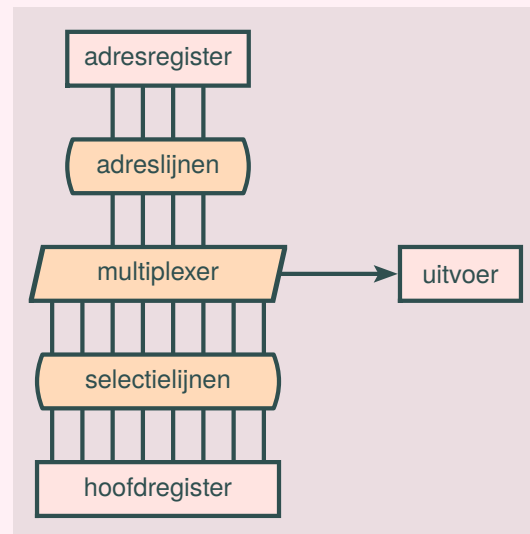
Tot slot nog een eenvoudig voorbeeld dat aantoont waarom een hoge lineaire complexiteit voor cryptografische doeleinden op zich onvoldoende is. De bitrij 000...01 heeft een lineaire complexiteit zo hoog als de lengte van de rij, maar mist het evenwicht tussen het aantal nullen en enen en is bovendien bijzonder voorspelbaar.

10.5 Registerselectie

Langs een omweg is het mogelijk om uit een lfsr toch niet-lineaire uitvoer te krijgen. Dat kan kan door het uitvoerbit niet uitsluitend van S_0 af te tappen, maar door ook de andere registertrappen in te schakelen. De lineaire terugkoppelfunctie van het schuifregister blijft daarbij onveranderd het nieuwe bit produceren. Echter het uitvoerbit wordt op een andere manier uit de registertrappen verkregen.

10.5.1 Multiplex generator

In de *multiplex generator* wordt het uitvoerbit willekeurig uit de registertrappen van het hoofdregister gekozen. In vergelijking met de rechtstreeks uit S_0 komende rij bits verwisselen de uitvoerbits nu van plaats, worden soms herhaald of juist overgeslagen. De keuze van de registertrap die het eerstvolgende uitvoerbit mag leveren geschiedt met behulp van een afzonderlijk schuifregister. Dit tweede register kiest (adresseert) in het eerste register de registertrap voor het uitvoerbit. Daarna doen beide schuifregisters de gebruikelijke stap naar de volgende toestand. Het systeem is afgebeeld in [figuur 10.10](#).



Figuur 10.10 Multiplex-schakeling

Veronderstel het hoofdregister telt 16 registertrappen. Dan zijn 4 bits (een 4-bits adres) nodig voor het aanwijzen van een van de registertrappen $S_0 \dots S_{15}$. Daartoe worden 4 registertrappen uit het hulpregister afgetapt. De 4 bits hierin worden gebruikt om een registertrap in het hoofdregister te adresseren. Het bit in de door dit adres aangewezen registertrap wordt het uitvoerbit van de multiplexschakeling. Het

hulpregister is te zien als een willekeurig heen en weer bewegend aanwijsstokje, dat telkens een bit in het hoofdregister uitkiest. In elke klokcyclus doen beide registers een stap en wordt een bit uitgevoerd. Door de lengte van de registers zo te kiezen dat deze geen gemeenschappelijke factor hebben, realiseert men de maximale periode en blijft de autocorrelatie ook acceptabel klein.

De geïnteresseerde lezer kan een analyse van de eigenschappen (waaronder de bepaling van het lineair equivalent) in de literatuur vinden^[10].

Een eenvoudig voorbeeld bestaande uit een 8-bits hoofdregister en een 7-bits hulpregister is

$$f_1(x) = 1 + x^4 + x^5 + x^6 + x^8 \quad \text{en} \quad f_2(x) = 1 + x^6 + x^7$$

	f_2				f_1								
t	65	432	10		7	6	5	4	3	2	1	0	z
1	11	001	01		1	0	1	0	1	1	0	0	0
2	01	100	10		1	1	0	1	0	1	1	0	1
3	00	110	01		0	1	1	0	1	0	1	1	1
4	10	011	00		1	0	1	1	0	1	0	1	0
5	11	001	10		1	1	0	1	1	0	1	0	1
6	11	100	11		0	1	1	0	1	1	0	1	0
7	01	110	01		1	0	1	1	0	1	1	0	0
8	10	111	00		0	1	0	1	1	0	1	1	0
9	11	011	10		1	0	1	0	1	1	0	1	1
10	11	101	11		0	1	0	1	0	1	1	0	0
11	01	110	11		0	0	1	0	1	0	1	1	0
12	10	111	01		0	0	0	1	0	1	0	1	0

Tabel 10.8 Voorbeeld uitvoer van multiplex-schakeling

Tabel 10.8 toont een aantal stappen van deze registercombinatie. Adresseren van een registertrap in het hoofdregister kan met een 3-bits adres. Het hulpregister gebruikt daarvoor de drie trappen $S_2S_3S_4$ en

het adres wordt berekend met $S_2 + 2S_3 + 4S_4$. De inhoud van de twee registers wordt van een passende beginwaarde voorzien. In de tabel staan van links naar rechts het volgnummer van het uitgevoerde bit, de inhoud van het hulregister, het daaruit gewonnen adres, de inhoud van het hoofdregister en het gekozen uitvoerbit.

Wie mocht denken dat deze generator goed bestand is tegen cryptanalyse komt bedrogen uit. In een artikel in het tijdschrift Cryptologia heeft Ross Anderson laten zien hoe gemakkelijk de begintoestand van het hulregister en de bitstroom uit het hoofdregister kunnen worden bepaald^[11].



Anderson

MASTERSECTIE – Cryptanalyse multiplex register

Om te begrijpen hoe deze generator kan worden geanalyseerd, wordt de productie van de sleutelstroom uit het voorbeeld van tabel 10.8 gevolgd, te beginnen op de eerste regel waar de klok start op tijdstip $t = 1$. Bij de analyse moet de door de combinatie geproduceerde *sleutelstroom* goed worden onderscheiden van de *bitrij van het lfsr* van het hoofdregister.

Het eerste sleutelbit is een 0 en is afkomstig uit S_1 van het hoofdregister. Als de klok op $t = 1$ start dan schuift het bit in S_1 op dat moment door naar S_0 en het zal bij de volgende klokcyclus op $t = 2$ uit het hoofdregister zijn verdwenen. In de geproduceerde sleutelstroom zien we als eerste het tweede bit uit de cyclus van het hoofdregister komen, een ander bit dan nu uit S_0 verdwijnt. Op $t = 2$ komt de 1 van S_4 als tweede bit in de sleutelstroom terecht. Dit bit zal op $t = 3$ in S_3 zitten en verdwijnen op $t = 6$. Men kan dus zeggen dat het adres uit het hulregister aangeeft over hoeveel stappen het aangewezen bit het hoofdregister zal verlaten. Na de eerste twee stappen zijn twee bits van het lfsr van het hoofdregister gezien, namelijk het tweede bit $s_2 = 0$ en het zesde bit $s_6 = 1$. Van die rij weten we op dit moment het volgende (bits lopen van links naar rechts)

$$s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8 s_9 s_{10} \dots = ?0???1????\dots$$

Op tijdstip $t = 3$ komt het bit in S_6 in de sleutelstroom terecht, dat is een 1. Dit bit zal 6 stappen later het hoofdregister verlaten en komt in de bitrij van het hoofdregister op plaats 6. Vervolgens komt er een 0

en zijn vier bits van de bitrij in het hoofdregister geplaatst.

$$s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8 s_9 s_{10} = ?0???10?1?$$

t	$S_6 \cdots S_0$	a	$t + a$	$s(t)$	12345678901234567890123456
1	0000001	0	1	0	0
2	1000000	0	2	1	01
3	1100000	0	3	1	011
4	1110000	4	8	0	011????0
5	1111000	6	11	1	011????0??1
6	1111100	7	13	0	011????0??1?0
7	1111110	7	14	0	011????0??1?00
8	1111111	7	15	0	011????0??1?000
9	0111111	7	16	1	011????0??1?0001
10	1011111	7	17	0	011????0??1?00010
11	0101111	3	14	0	011????0??1?00010
12	1010111	5	17	0	011????0??1?00010
13	0101011	2	15	1	011????0??1?00110
1	0001000	2	3	0	??0
2	0000100	1	3	1	??1
1	1100101	1	2	0	?0
	...				...
19	0110100	5	24	0	?0???10?10?10?010?000??0
20	0011010	6	26	1	?0???10?10?10?010?000??0?1
21	0001101	3	24	0	?0???10?10?10?010?000??0?1

Tabel 10.9 Cryptoanalyse van multiplex generator

In de volgende stap gebeurt iets belangrijks. Op $t = 5$ komt namelijk de 1 uit S_1 in de sleutelstroom en moet worden geplaatst op positie 6 in de lfsr-bitrij. Dit bit is echter al eerder gebruikt en wel toen het zich op $t = 2$ nog in S_4 bevond. Hetzelfde bit komt dus tweemaal in de sleutelstroom terecht. Conclusie: de twee

bits op deze plaatsen in de sleutelstroom moeten aan elkaar gelijk zijn! Hierop kan een test gebaseerd worden. Probeer een begintoestand van het hulpregister en controleer een bekende sleutelstroom op zulke bits.

Voor elke begintoestand van het hulpregister worden de achtereenvolgens geproduceerde adressen gegenereerd en gebruikt om het bit uit de bekende sleutelstroom in een reconstructie van de hoofdregisterstroom te plaatsen. Telkens wanneer een bit tweemaal als sleutelbit gebruikt is wordt gecontroleerd of deze sleutelbits aan elkaar gelijk zijn. Als niet de goede begintoestand is gekozen dan is het zeer waarschijnlijk dat dit op enig moment spaak loopt: de adressen van het hulpregister corresponderen dan immers niet met de werkelijke selectie uit de hoofdregisterstroom.

In het gekozen voorbeeld wordt deze analyse gedemonstreerd in [tabel 10.9](#). Bovenaan is 0000001 gekozen als begintoestand van het hulpregister. Twaalf stappen lang gaat het goed maar in de dertiende stap klopt het sleutelbit niet meer met wat het register aangeeft. Lager in de tabel valt te zien dat een foutieve positie soms al heel vlug door de mand valt, want voor begintoestand 0001000 treedt al bij de tweede stap een contradictie op. Onderaan tenslotte is de juiste begintoestand gekozen en gevolgd tot en met stap 21.

Bepaling van de begintoestand van het hoofdregister is mogelijk op het moment dat evenveel opeenvolgende bits bekend zijn als het aantal registertrappen. In het voorbeeld is dat na stap 21 nog niet het geval en mogelijk komt het ook niet zover. Er zijn echter al zes van de benodigde acht opeenvolgende bits bekend, zodat maar vier mogelijkheden overblijven. Die kunnen snel worden getest tegen de rest van de bitstroom. Als alle 127 mogelijke begintoestanden van het hulpregister worden doorgerekend, blijkt dat geen van de foutieve toestanden het langer dan 27 stappen volhoudt. De juiste begintoestand laat zich dus probleemloos vaststellen. In een andere test is $f(x) = 1 + x + x^{15}$ als hulpregister gebruikt naast hetzelfde hoofdregister. Alle foutieve 32766 begintoestanden werden in hoogstens 63 stappen ontmaskerd.

Het doorrekenen van alle begintoestanden wordt bewerkelijker naarmate het hulpregister groter is. Het aantal te onderzoeken waarden kan echter worden ingeperkt. Hiervoor wordt in het voorbeeld gekeken naar de tweede test uit de tabel, die waarin de eerste twee sleutelbits al direct tot verwerping leiden. De oorzaak van dit snelle falen is het samenlopen van twee omstandigheden

1. het bitpatroon 0010 in registertrappen $S_5S_4S_3S_2$ produceert achtereenvolgens 2 en 1 als adres, zodat hetzelfde bit in het hoofdregister tweemaal achtereen gebruikt wordt, en
2. de corresponderende waargenomen sleutelbits zijn verschillend.

Steeds wanneer die samenloop zich voordoet kan de betreffende begintoestand worden uitgeschakeld. Dus als de twee voorste bits van de sleutelstroom verschillen dan vallen alle bitpatronen ?0010?? af. Hetzelfde geldt voor ?0001??, terwijl ?0100?? een overeenkomstig effect geeft op sleutelbits die twee posities uiteen liggen, enz. De beginwaarden van het adresregister kunnen al onafhankelijk van de begintoestand van het hoofdregister worden getest. Het adres- en hoofdregister worden dus (deels) ontkoppeld.

10.5.2 Filterfuncties

In de multiplex-schakeling wordt steeds één bit van het hoofdregister benut. Een andere mogelijkheid is om naast de terugkoppelfunctie een tweede functie in te schakelen voor het berekenen van de sleutelstroom. Men noemt deze tweede functie een *filterfunctie*. Zo worden kool en geit gespaard: het schuifregister doorloopt de bekende maximaalrij en de (niet-lineaire) filterfunctie verhindert dat eenvoudige cryptanalyse op de uitvoerstroom kan worden gepleegd. De kwaliteit van de uitvoerstroom is afhankelijk van de gekozen filterfunctie. Uit het aantal registertrappen en de orde van de filterfunctie kan men berekenen wat de hoogste lineaire complexiteit is die hiermee gerealiseerd kan worden.

Wanneer de lengte van het schuifregister groot genomen wordt, zullen relatief veel van alle mogelijke filterfuncties die waarde voor de maximale lineaire complexiteit benaderen. Het blijkt wel van belang

om van elke orde genoeg termen in de filterfunctie mee te nemen. D.w.z. als een filterfunctie de vier registertrappen S_1, S_2, S_5, S_7 gebruikt, dan moeten in die filterfunctie allerlei termen staan zoals $S_2, S_1S_7, S_2S_5S_7$ en $S_1S_2S_5S_7$.

Hier enkele voorbeelden van zulke functies.

- $f_1 = S_1 \oplus S_2 \oplus (S_1 \oplus S_3)(S_2 \oplus S_4 \oplus S_5) \oplus (S_1 \oplus S_4)(S_2 \oplus S_3)S_5$
- $f_2 = S_1S_2 \oplus S_3S_4 \oplus S_5$
- $f_3 = S_1S_2 \oplus S_3S_4 \oplus S_5S_6 \oplus S_7$
- $f_4 = S_1(1 \oplus S_4) \oplus S_2(1 \oplus S_5) \oplus S_3(1 \oplus S_6) \oplus S_1S_2S_3$

In het geval de filterfunctie statistische onvolkomenheden heeft kan deze generator met succes worden aangevallen^[12]. Zulke onvolkomenheden treden op wanneer niet alle bitcombinaties met eenzelfde kans worden geproduceerd.

MASTERSECTIE – Cryptoanalyse filterfunctie

Veronderstel de filterfunctie heeft m registertrappen $S_1 \dots S_m$ zodat het uitvoerbit $z = f(S_1, \dots, S_m)$ is. Voor het gemak wordt aangenomen dat deze trappen zich aaneengesloten in het register bevinden. Elk uitvoerbit is dan afhankelijk van een blokje van m opeenvolgende registertrappen S , dus ook van evenveel opeenvolgende bits s in de lfsr-stroom. Gezocht wordt echter juist het omgekeerde, namelijk een manier om van de bitstroom z van de filtergenerator terug te rekenen naar de bitstroom s van het lfsr. Volg daartoe eens de productie van een groepje van m uitvoerbits z . Op tijdstip t geldt

$$z_t = f(S_1(t), \dots, S_m(t)) \quad (10.38)$$

Omdat de bits uit $S_1(t), S_2(t)$, enz. achtereenvolgens als de bits s_1, s_2 , enz. uit het lfsr weglopen, kan dit ook geschreven worden als

$$z_t = f(s_1, \dots, s_m) \quad (10.39)$$

Op $t = t + 1$ zijn de bits in het lfsr alle een plaats opgeschoven en komt er dus

$$z_{t+1} = f(s_1(t+1), \dots, s_m(t+1)) = f(s_2, \dots, s_{m+1}) \quad (10.40)$$

Zo kan voor m opeenvolgende bits $z_t \dots z_{t+m-1}$ de volgende reeks productievergelijkingen opgeschreven worden

$$\begin{aligned} z_t &= f(s_1, \dots, s_m) \\ z_{t+1} &= f(s_2, \dots, s_{m+1}) \\ &\dots \\ z_{t+m-1} &= f(s_m, \dots, s_{2m-1}) \end{aligned} \quad (10.41)$$

Uit de bovenstaande vergelijkingen kan de conclusie getrokken worden dat een blokje van m opeenvolgende bits uit z slechts afhankelijk is van een blokje van $(2m - 1)$ opeenvolgende bits uit het lfsr. Met andere woorden, alle mogelijke groepjes van m opeenvolgende uitvoerbits ontstaan uit de 2^{2m-1} waarden van $(2m - 1)$ bits. Bereken nu voor alle argumenten het uitvoerbit

$$z(x) = f(x), \quad x = 0, \dots, 2^{2m-1} - 1, \quad z(x) \in [0: 2^m - 1] \quad (10.42)$$

Kijk vervolgens naar de distributie van de gevonden waardenverzameling. Sommige filterfuncties zullen een mooie, gelijkmatige verdeling te zien geven, maar andere geven de ongelijkmatigheden waaraan de cryptoanalist houvast vindt.

Om de bedoelde aanval op de filtergenerator te illustreren is als filterfunctie gekozen $f = S_2 \oplus S_1 S_3$, zodat $m = 3$ en $2m - 1 = 5$. Het aantal producties is zodoende gering genoeg om te kunnen overzien. In [tabel 10.10](#) zijn ze alle 32 opgenomen, over de kolommen verdeeld naar elk van de acht mogelijke uitkomsten. De ongelijkmatige verdeling van de uitkomsten is goed te zien. De + tekens in de onderste rij van de tabel geven aan dat het bit in de kolom daarboven ondubbelzinnig kan worden bepaald.

Uitkomst $z = 101$ komt slechts één keer voor, hetgeen betekent dat daar de rij 01110 in het schuifregister moet hebben gestaan. Voor andere waarden komt eveneens direct informatie over s ter beschikking:

0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
00000	01000	00100	01100	00010	01110	00101	01010
00001	01001	00111	10100	00011		00110	01011
10000	01111	11100	10111	10010		11101	01101
10001	11000			10011			10101
11111	11001			11110			10110
							11010
							11011
?????	?+???	??+??	??+??	???+?	+++++	+	?????

Tabel 10.10 Alle producties van $f = S_2 \oplus S_1 S_3$

uit het voorkomen van 001 kan één bit van s worden geconcludeerd, want een bitpatroon ?1??? is er de generator van. In andere gevallen kan met grote kans een bit worden vastgelegd, zoals bij 000 wanneer er 80% kans is op ?000?. Soms zijn twee kolommen in de productietabel gelijk of juist elkaars tegengestelde, waardoor relaties tussen bits uit s worden gevonden. Wanneer op deze manier een voldoende deel van s gereconstrueerd is, kan daarmee de begintoestand van het register worden bepaald.

Mifare Classic

Een voorbeeld van het gebruik van een filterfunctie is te vinden in de Mifare Classic, een cryptosysteem bedoeld voor smartcards en o.a. te vinden in de OV-chipkaart. Deze bevat een 48-bits LFSR met taps op de posities 43, 39, 38, 36, 34, 33, 31, 29, 24, 23, 21, 19, 13, 9, 7, 6, 5 en 0. Elke klokcyclus gaan twintig bits van het register door een filterfunctie en produceren dan één uitvoerbit. Voor de filtering worden deze twintig in vijf groepjes van vier afgetapt, elk groepje levert een functiewaarde. De vijf functiewaarden worden dan verder gecombineerd.

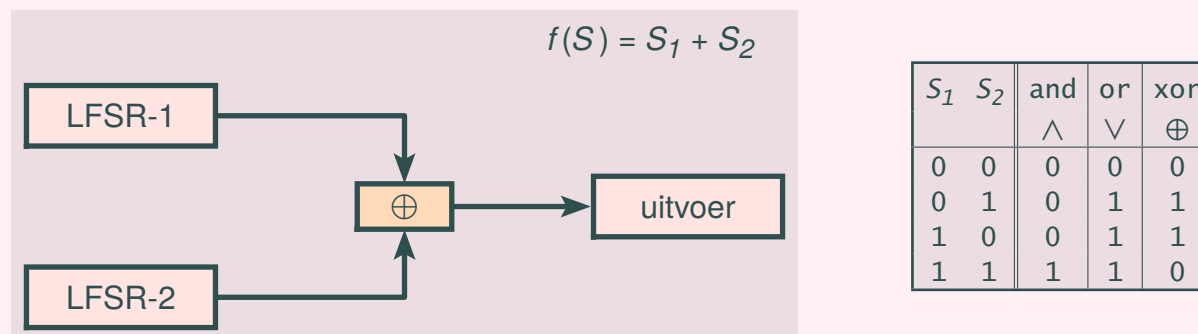
Het ontwerp van de Mifare Classic bevat nogal wat zwakheden. Om er eentje te noemen: de chip bevat een toevalsgenerator met een periode van slechts 65535, die zich elke 618 milliseconden herhaalt. De chip kan dan ook op verschillende manieren worden gekraakt. Onderzoekers van de Nijmeegse Radboud Universiteit geven in hun artikel over het draadloos kraken van de chip een viertal scenarios^[13]. Ondanks deze zwakheden blijft het gebruik ervan gehandhaafd met als voornaamste reden dat de kaart goedkoop is en dat een duurdere versie qua kosten niet opweegt tegen de feitelijk geleden schade.

10.6 Registercombinatie

Bij de constructie van niet-lineaire schuifregisters door het combineren van de uitvoer van lfsr's wordt ernaar gestreefd om met zo gering mogelijke middelen een zo hoog mogelijke lineaire complexiteit te bereiken. Een hoge lineaire complexiteit maakt het immers moeilijk of zelfs ondoenlijk om een equivalent lfsr te bouwen waarmee de cryptografische effectiviteit teniet gedaan kan worden. Het is zaak mechanismen te vinden waarmee het teniet doen van de lineariteit in de samenstellende lfsr's effectief kan gebeuren. Tegelijk echter moet men er op bedacht zijn dat de goede random eigenschappen van het lfsr intact blijven. Na de behandeling van enkele van deze soort generatoren zal de gevaarlijkste aanval op hen, de correlatieaanval, worden gedemonstreerd. De behandelde voorbeelden zijn de

- Pless generator
- Geffe generator
- Bruer generator

Maar eerst zal worden aangetoond dat het direct combineren van de uitvoer van lfsr's met een eenvoudige operatie niet voldoet. In [figuur 10.11](#) zijn de uitvoerbits van twee lfsr's met respectievelijk de logische *AND*, de *OR* en de *XOR* gecombineerd. De *AND*-operator voldoet niet omdat dan driekwart van alle uitvoerbits een 0 wordt, waarmee het eerste kwaliteitscriterium van Golomb geweld wordt aangedaan. De *OR*-operator geeft een even beroerd resultaat, want dan wordt driekwart van de bits een 1.



Figuur 10.11 Combinatie van twee lfsr's met eenvoudige operator

Blijft over de exclusive or (xor) als combinerende operator. Maar ook deze biedt weinig perspectief, omdat het een lineaire operator is en als gevolg daarvan ontbeert de combinatie de nu juist zo gewenste niet-lineariteit. De exclusive or som van twee lfsr's blijkt gelijk aan het register gebaseerd op het product van de karakteristieke functies van de samenstellende lfsr's: $f(x) = f_1(x)f_2(x)$. De periode ervan is het kleinste gemene veelvoud van de periodes van de deelstromen.

De volgende redenering doet dit inzien. Het effect van deze combinatie blijkt uit de volgende berekening. Gebruikmakend van [vergelijking \(10.24\)](#) kan de som van twee lfsr's geschreven worden als

$$\text{LFSR-1} \oplus \text{LFSR-2} = \frac{1}{f_1(x)} + \frac{1}{f_2(x)} = \frac{f_1(x) + f_2(x)}{f_1(x)f_2(x)} = \frac{1}{f(x)} \quad (10.43)$$

Ook bij primitieve $f_1(x)$ en $f_2(x)$ kan het geen maximaalrij zijn omdat de genererende functie $f(x)$ samengesteld is, zoals in het voorgaande is aangetoond, zie [vergelijking \(10.33\)](#).

Het lineair equivalent van de gecombineerde stroom is met $n_1 + n_2$ maar klein. Deze combinatie heeft dus weinig te bieden. Ter illustratie een voorbeeld. Met $f_1(x) = 1 + x + x^3$ en $f_2(x) = 1 + x^3 + x^5$ zijn de componenten twee maximaalrijen met respectievelijke periode 7 en 31 die samen een periode van 217 geven. Startend met een klein beginstukje van de bitrij, vindt de Berlekamp-Massey algoritme het

productregister $f(x) = 1 + x + x^4 + x^5 + x^8$ al na 15 bits.

Tot slot wordt gekeken naar combinaties van bitrijen afkomstig van hetzelfde register. Wanneer de lfsr's vanuit identieke begintoestand starten lopen beide stromen synchroon en is het effect gelijk aan starten in de nultoestand: een stroom van nullen. Voor starten in verschillende begintoestand mag worden verwezen naar hetgeen is opgemerkt bij de introductie van het derde criterium van Golomb en de berekening van de autocorrelatiefunctie uit [vergelijking \(10.3\)](#): het resultaat is dezelfde, onveranderde bitreeks gestart vanuit een andere begintoestand. Deze combinatie biedt daarom geen enkele meerwaarde.

Voor hogere machten $f(x)^n$ van de karakteristieke functie geldt hetzelfde, de bitstroom verandert er niet door. Bij het machtsverheffen ontstaat in het algemeen een karakteristieke functie met meer termen (meer tappunten) dan in $f(x)$ zelf, behalve wanneer n een macht van twee is. Dan geldt

$$f(x)^n = f(x^n) \text{ als } n = 2^k \quad (10.44)$$

De karakteristieke functie wordt nu eenvoudig gevonden door alle termen tot de macht n te verheffen, het aantal termen blijft ongewijzigd. Er wordt dus een register verkregen met hetzelfde aantal tappunten als $f(x)$. Bijvoorbeeld uit $f(x) = 1 + x + x^3$ met tappunten op de coëfficiënten $C_0 = C_1 = 1$ ontstaat $f(x)^4 = 1 + x^4 + x^{12}$, een lfsr voor dezelfde bitstroom, van lengte $n = 12$ en eveneens met slechts twee tappunten, nu op $C_0 = C_4 = 1$. Van deze eigenschap kan handig gebruik gemaakt worden bij de cryptanalyse van gecombineerde bitstromen, omdat hiermee extra relaties tussen uitvoerbits gelegd kunnen worden.

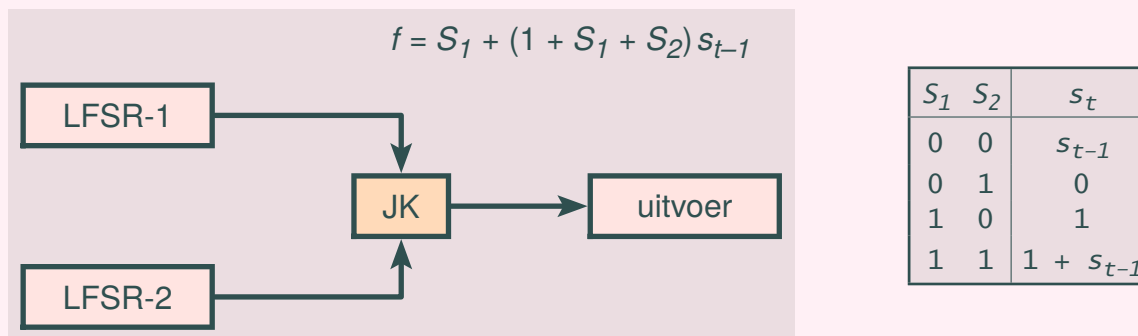
10.6.1 Pless generator

De naar Vera Pless genoemde Pless generator is een van de vroegst bruikbare combinaties^[14]. De combinerende operator voor deze generator is de zogenaamde JK-flipflop. Dit is een uit de electronica bekende geheugenschakeling voor opslag van één bit. Het bijzondere aan deze schakeling is dat de uitvoer niet alleen van de twee invoerbits afhangt, maar soms ook van het vorige bit. De productietabel in [figuur](#)

10.12 laat dit zien*. Als de invoerbits van de flipflop verschillend zijn is het resultaat alleen van die twee afhankelijk. Maar als ze aan elkaar gelijk zijn wordt òf het voorgaande bit òf het tegengestelde daarvan afgeleverd. De functie die behoort bij de JK-flipflop valt te berekenen met het algemeen schema gegeven in tabel 10.4. Gevonden wordt

$$\begin{aligned} f(x) &= (1 + S_1)(1 + S_2)s_{t-1} + S_1(1 + S_2) + S_1S_2(1 + s_{t-1}) \\ &= S_1 + (S_1 + S_2 + 1)s_{t-1} \end{aligned} \quad (10.45)$$

Aan deze functie valt direct te zien dat het vorige bit (of diens complement) in de uitvoer terechtkomt wanneer S_1 en S_2 dezelfde waarde hebben omdat in dat geval $S_1 + S_2 = 0$. Als S_1 en S_2 niet aan elkaar gelijk zijn wordt de term tussen de haakjes 0 en is bit S_1 de uitvoer.



Figuur 10.12 Combinatiestroom en productietabel JK-flipflop

Uit het afgeleverde bit kan het een en ander over S_1 en S_2 geconcludeerd worden.

uitvoer 0 $\rightarrow (S_1, S_2) = (0, 1)$ of $(0, 0)$ of $(1, 1)$

uitvoer 1 $\rightarrow (S_1, S_2) = (1, 0)$ of $(0, 0)$ of $(1, 1)$

* Het complement van een bit S is $1 + S$ als het gevolg van het rekenen met bits. Een ander manier om het complement aan te duiden is er een streep boven te plaatsen $\bar{S}$.

Dit laat zien dat als een uitvoerbit bekend is, slechts drie van de vier combinaties van invoerbits als voorganger in aanmerking komen. Is de uitvoer een 0 dan was het bit uit LFSR-1 geen 1 en tegelijk het bit uit LFSR-2 geen 0. Bij een 1 als uitvoerbit is dat omgekeerd. Is nu ook nog het daar aan voorafgaande bit bekend, dan kan òf het bit uit LFSR-1, òf het bit uit LFSR-2 met zekerheid worden bepaald. Want met eerst een 0 gevolgd door opnieuw een 0 kan de huidige toestand niet $(S_1, S_2) = (1,1)$ zijn, omdat in dat geval juist het vorige bit een 1 had moeten zijn. Daardoor blijven twee van de drie mogelijkheden over, te weten $(0,1)$ en $(0,0)$; deze twee hebben $S_1 = 0$ gemeenschappelijk. Zodat de conclusie is dat uit LFSR-1 een 0 kwam, terwijl de uitvoer van LFSR-2 onbekend blijft. In tabel 10.11 zijn alle mogelijkheden uitgeplozen.

2 uitvoerbits	LFSR-1	LFSR-2
00	0	?
01	1	?
10	?	1
11	?	0

Tabel 10.11 Reconstructie JK-flipflop invoer

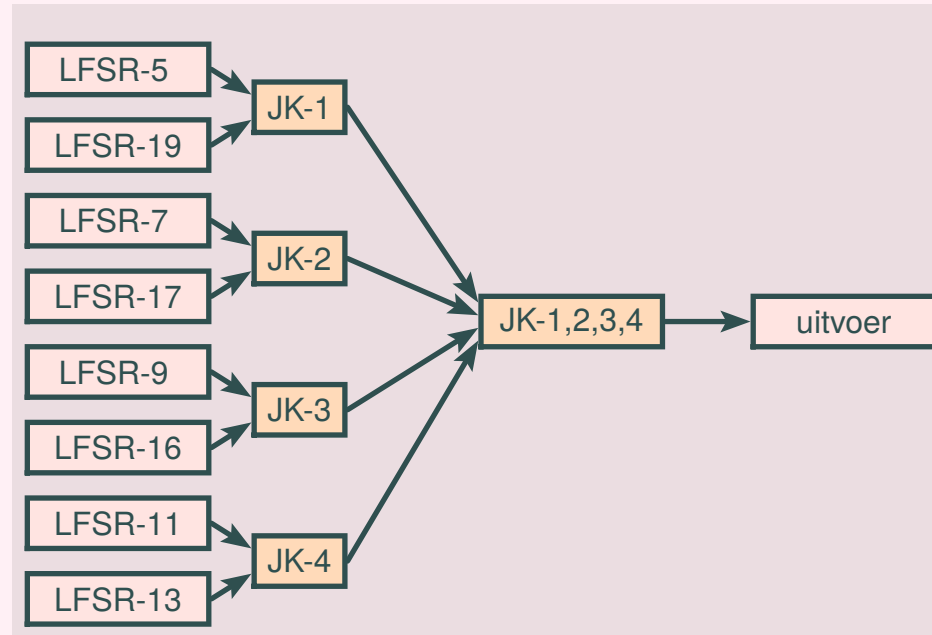
Deze halve reconstructie heeft gevolgen voor de reconstructie van de lfsr's aan de ingangen van de JK-flipflop. Vanaf het tweede bit kan steeds in één van beide invoerstromen een bit gevonden worden.

$$1001110011110100110010001 \rightarrow \begin{cases} ??01???01????1?01??01?001 \\ ?1??001??0001?1??01??1??? \end{cases}$$

De combinatiestroom met de JK-flipflop kan tegen deze reconstructie beschermd worden door het onmogelijk te maken dat opeenvolgende bits uit dezelfde JK-flipflop komen. Daarom construeerde Pless haar pseudorandom generator niet uit één maar uit vier van zulke JK-flipflop's. Om en om komt elk vierde bit uit een bepaalde deelstroom. De Pless generator staat in figuur 10.13. Hierin is LFSR-5 een 5-trapsregister, LFSR-19 een 19-traps, enz. Reconstructie van deze generator vergt het bepalen van acht lfsr's. Dat lijkt geen kleinigheid, zelfs al zijn de karakteristieke functies van de lfsr's bekend. Met de



Pless



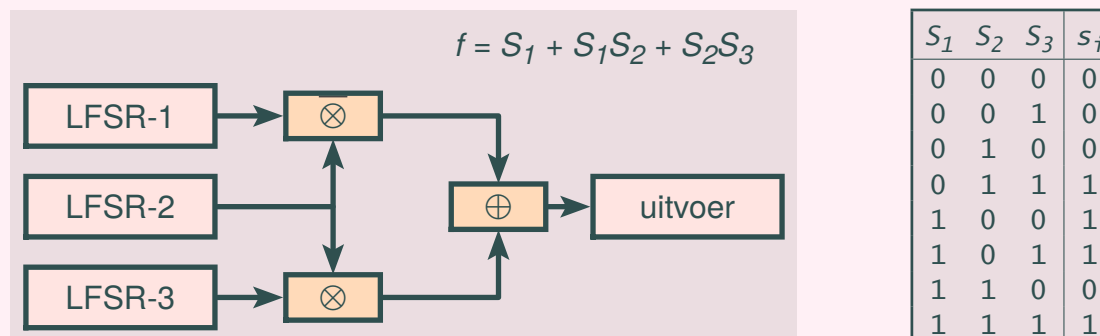
Figuur 10.13 Pless generator

parametrisering door Pless moeten immers bijna 2^{97} begintoestanden bekeken worden – voorwaar een formidabel aantal. De hierna te behandelen methode van de correlatieaanval biedt echter de mogelijkheid om de lfsr's van deze generator een voor een aan te pakken en dan blijkt dat een reconstructie heel goed tot de mogelijkheden behoort.

10.6.2 Geffe generator

Een aardige manier om twee bitstromen te combineren werd bedacht door Geffe^[15]. Het idee is om telkens willekeurig te kiezen tussen twee lfsr's met behulp van een selectiebit uit een derde lfsr. Heel ingenieus wordt het selectiebit vermenigvuldigd met het bit uit het ene lfsr en het complement ervan met het bit uit

het andere lfsr. Daarna worden die twee producten opgeteld en dan is altijd een van beide 0 en de ander gelijk aan het bit van het geselecteerde lfsr (figuur 10.14).



Figuur 10.14 Geffe generator met productietabel

In de figuur staat behalve de registercombinatie van de Geffe generator ook de productietabel. Met behulp van de bekende standaardmethode is de terugkoppelfunctie berekend. De periode van deze combinatie is gelijk aan het kleinste gemene veelvoud van de afzonderlijke lfsr's. Kiest men de lengte van de afzonderlijke registers zo dat ze geen gemeenschappelijke factor hebben, dan wordt de maximale periode gehaald. Uit de terugkoppelfunctie kan het lineair equivalent berekend worden.

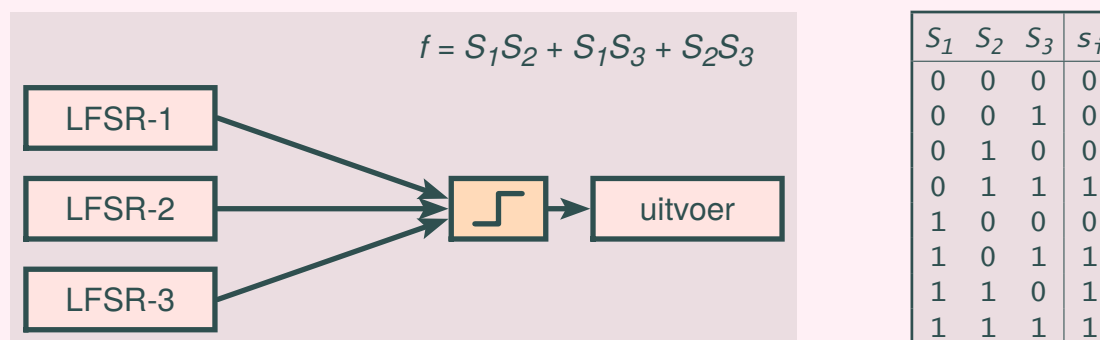
$$\begin{aligned}
 f(S_1, S_2, S_3) &= S_1 + S_1S_2 + S_2S_3 \\
 p &= \text{kgv}((2^{n_1} - 1), (2^{n_2} - 1), (2^{n_3} - 1)) \\
 LC &= n_1 + n_1n_2 + n_2n_3
 \end{aligned}
 \tag{10.46}$$

10.6.3 Bruer generator

Als derde voorbeeld van dit type schuifregister de Bruer generator^[16]. Het uitvoerbit wordt *bij meerderheid van stemmen* bepaald uit de deelnemende lfsr's. Een oneven aantal lfsr's garandeert dat er altijd een

winnaar is. In **figuur 10.15** vormen drie lfsr's de generator. De terugkoppelfunctie, periode en lineaire complexiteit zijn respectievelijk

$$\begin{aligned} f(S_1, S_2, S_3) &= S_1S_2 + S_1S_3 + S_2S_3 \\ p &= \text{kgv}((2^{n_1} - 1), (2^{n_2} - 1), (2^{n_3} - 1)) \\ LC &= n_1n_2 + n_1n_3 + n_2n_3 \end{aligned} \quad (10.47)$$



Figuur 10.15 Bruer generator met productietabel

De terugkoppelfunctie is hier symmetrisch in de deelnemende lfsr's, getuige de drie kruistermen S_iS_j . De lineaire complexiteit is iets groter dan bij de Geffe generator, maar deze twee ontlopen elkaar niet veel. Vanzelfsprekend kunnen meer samenstellende registers gebruikt worden, mits hun aantal maar oneven is.

10.7 Correlatieaanval

Hoewel de in het voorgaande behandelde generatoren er op papier indrukwekkend uitzien, valt het resultaat op het gebied van veiligheid tegen. De reden is gelegen in het feit dat ze met een zogenaamde

correlatieaanval gemakkelijk kunnen worden gebroken. Deze aanvalstechniek kent twee varianten. De eerste en meest eenvoudige is die van Siegenthaler, waarmee de correlatieaanval werd geïntroduceerd. Een meer geavanceerde variant is afkomstig van Meier en Staffelbach en kan tegen schuifregisters van behoorlijk grote afmetingen worden ingezet.

Kijk voor het principe van de correlatieaanval eerst naar [tabel 10.12](#), de productietabel van de Geffe generator uit [figuur 10.14](#). Voor alle toestanden van de drie deelnemende lfsr's is het uitvoerbit vergeleken met dat van de generator als geheel. Een ☒ achter het lfsr-bit signaleert dat het uitvoerbit van het lfsr overeenkomt met de sleutelstroom van de generator. Het blijkt dat in zes van de acht gevallen het uitvoerbit hetzelfde is als het bit uit LFSR-1, en evenzo zes van de acht keer hetzelfde als het bit uit LFSR-3. Er bestaat dus een *correlatie* van $6/8 = 0,75$ ofwel van 75% tussen deze twee registers en de uitvoer van de generator. Is de sleutelstroom bekend (bijvoorbeeld vanwege een bekende klaartekst), dan komt bij vergelijking met respectievelijk LFSR-1 en LFSR-3 gemiddeld driekwart van de bits overeen. Voor het keuzeregister LFSR-2 gaat die vlieger niet op, want daarvoor bedraagt de correlatie 50% hetgeen zeggen wil: een fiftyfifty kans dat de bits overeenkomen.

LFSR-1	LFSR-2	LFSR-3	s_i
0 ☒	0 ☒	0 ☒	0
0 ☒	0 ☒	1 ☐	0
0 ☒	1 ☐	0 ☒	0
0 ☐	1 ☒	1 ☒	1
1 ☒	0 ☐	0 ☐	1
1 ☒	0 ☐	1 ☒	1
1 ☐	1 ☐	0 ☒	0
1 ☒	1 ☒	1 ☒	1

Tabel 10.12 Productietabel Geffe generator met overeenkomsten

De correlatie tussen de lfsr's 1 en 3 en de sleutelstroom maakt het mogelijk om uit de sleutelstroom meer dan de helft van de uitvoerbits van die lfsr's goed te raden. Als meer dan de helft van die bits goed is, kan dit aantal door berekening worden vergroot. Zodra tenminste $2n$ opeenvolgende bits zijn gevonden kan het betreffend lfsr c.q. de begintoestand ervan worden bepaald. Voor correlaties kleiner dan 0,5 spreekt men van *anticorrelatie*. Die is net zo bruikbaar, omdat in dat geval de kans op het complement van het lfsr-bit gelijk aan het sleutelbit groter is dan 50%.

10.7.1 Cijfertekstcorrelatie

De sleutelstroom is te bepalen als de klaartekst van het cryptogram bekend is. Maar ook met alleen cryptogramtekst kan een correlatieaanval uit de voeten. Dat komt omdat in klaartekst over het algemeen de verhouding tussen 0'en en 1'en niet precies één op één zal zijn. Als men bedenkt dat in de gangbare 8-bits ASCII-code het linkerbit meestal een 0 is*, wordt dat duidelijk. De ongelijke verdeling tussen 0 en 1 in de klaartekst werkt door in het cryptogram en maakt een correlatieaanval op de sleutelstroom ook met cijfertekst mogelijk. Het ligt anders wanneer in de klaartekst de verhouding tussen 0 en 1 ongeveer gelijk gemaakt is, bijvoorbeeld door toepassing van tekstcompressie. In dat geval faalt deze methode. Men heeft dan als het ware te maken met een omgekeerd one-time pad: de randomverdeling van de bits in de klaartekst maskeert de structuur in de sleutelstroom.

De invloed op de correlatie van de verdeling tussen nullen en enen in de klaartekst kan als volgt worden verdisconteerd. Waar bij bekende sleutelstroom wordt gekeken naar de kans op gelijke bits uit lfsr en sleutelstroom, komt daarvoor in de plaats de kans op gelijke bits uit lfsr en cryptogram. Indien bit s_i van de sleutelstroom uit het schuifregister en bit c_i van het cryptogram dezelfde waarde hebben is hun som 0. Het omgekeerde geldt eveneens

* Alle meest gebruikte tekens zoals letters cijfers, leestekens hebben in ASCII een waarde kleiner dan 128 en dus een linkerbit van 0. De tekens met een 1 in het linkerbit zijn die met accenten, umlaut, e.d. en komen veel minder voor.

$$s_i = c_i \iff s_i \oplus c_i = 0 \quad (10.48)$$

Bereken vervolgens de kans op zo'n gebeurtenis, dat wil zeggen zoek de kans $p(s_i \oplus c_i = 0)$. De modulo 2 som in deze uitdrukking heeft in twee gevallen de waarde nul, namelijk voor $s_i = c_i = 0$ en voor $s_i = c_i = 1$. De kans op tegelijkertijd $s_i = 0$ en $c_i = 0$ wordt gegeven door het product van de kansen op de afzonderlijke gebeurtenissen

$$p(s_i \oplus c_i = 0) = p(s_i = 0) \cdot p(c_i = 0) + p(s_i = 1) \cdot p(c_i = 1) \quad (10.49)$$

Omdat c_i gelijk is aan $P \oplus S$ ([vergelijking \(10.1\)](#)) met P het bit van de klaartekst en S het sleutelbit van de gehele generator, kunnen $p(c_i = 0)$ en $p(c_i = 1)$ worden uitgeschreven

$$(S, P) = (0, 0) \text{ of } (1, 1) \rightarrow c_i = 0$$

$$(S, P) = (0, 1) \text{ of } (1, 0) \rightarrow c_i = 1$$

$$\begin{aligned} p(s_i \oplus c_i = 0) &= p(s_i = 0) \cdot p(S = 0) \cdot p(P = 0) + p(s_i = 1) \cdot p(S = 1) \cdot p(P = 0) \\ &\quad + p(s_i = 0) \cdot p(S = 1) \cdot p(P = 1) + p(s_i = 1) \cdot p(S = 0) \cdot p(P = 1) \\ &= p(s_i = S) \cdot p(P = 0) + p(s_i \neq S) \cdot p(P = 1) \end{aligned} \quad (10.50)$$

In de laatste stap zijn de kansen voor de verschillende waardecombinaties van s_i en S samengenomen tot de kans dat de bits van het lfsr en van de sleutel al of niet gelijk zijn. Kort $p(P = 0)$ af tot p_0 (de kans op een 0 in de klaartekst) en $p(s_i = S)$ tot q_{lfsr} (de correlatie tussen lfsr en sleutelstroom). De correlatie p_{lfsr} tussen de bitstroom uit een van samenstellende lfsr's en de cryptotekst, wordt dan

$$p_{lfsr} = p(s_i = c_i) = 1 - p_0 - q_{lfsr} + 2p_0q_{lfsr} \quad (10.51)$$

Ga na dat $p_0 = 1$ overeenkomt met bekende sleutelstroom waardoor de correlatie uitkomt op $p_{lfsr} = q_{lfsr}$ en verifieer ook dat $p_0 = 0,5$ niets van de correlatie overlaat. Voor reële klaartekst zal de correlatie ergens tussen deze twee uitersten liggen.

Uit enkele experimenten met bestanden in de printertaal Postscript bleek een gemiddelde kans van 63%

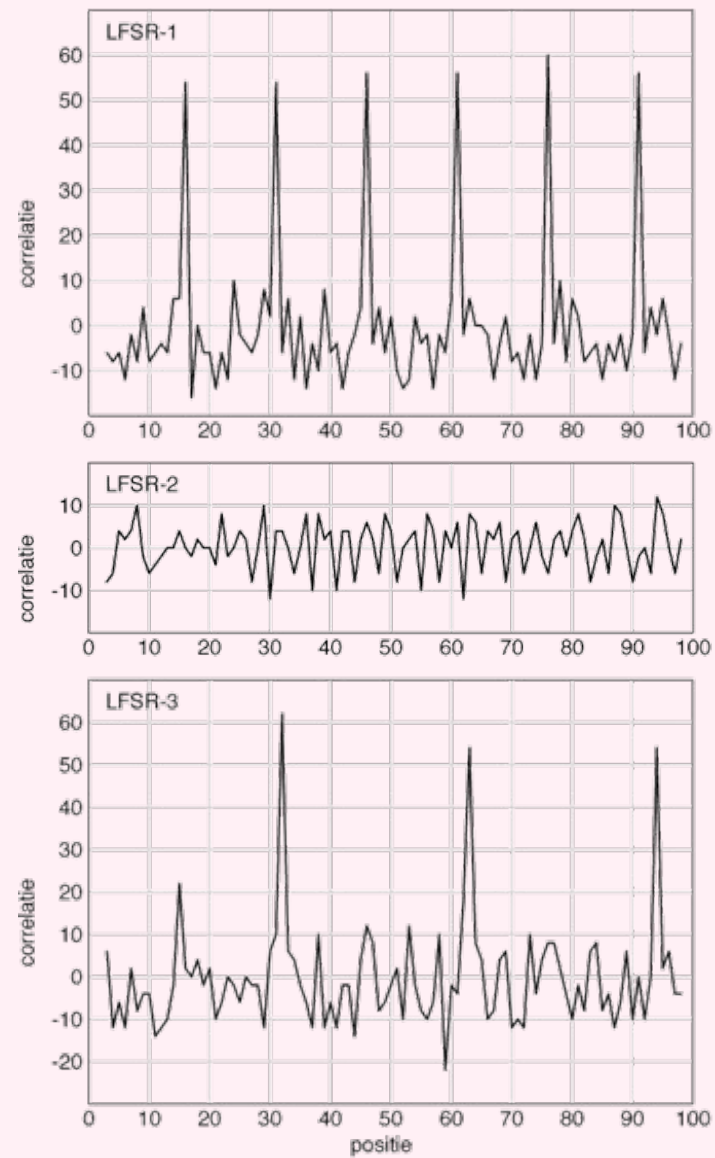
op een 0. Daarbij moet worden opgemerkt dat het linkerbit in postscript altijd een 0 is. In pdf-bestanden bleek de kans op een 0 met ongeveer 54% wat kleiner. Een correlatieaanval op een pdf is dus minder kansrijk dan een op postscript. Maar net als bij postscript lijkt het ook bij pdf's de moeite waard om de aanval op het linkerbit te openen; dat is in gemiddeld 66% van de gevallen een 0.

10.7.2 Siegenthaler-aanval

Siegenthaler bedacht de volgende methode om dankzij een correlatie die verschilt van 50% de begintoestand van de afzonderlijke registers in een sleutelstroomgenerator te achterhalen^[17].

Stel dat van de Geffe generator de karakteristieke functie van LFSR-1 bekend is, of is gekozen uit een aantal waarschijnlijke kandidaten. Neem een willekeurige begintoestand en genereer vandaaruit de gehele bitstroom die dit lfsr produceert. Die stroom kan worden gelegd naast de uitvoer van de generator en als gevolg van de 6/8 correlatie zal gemiddeld 75% van de bits overeenkomen. Aangenomen natuurlijk dat de juiste begintoestand is gebruikt. Met een andere begintoestand is die overeenkomst er niet. De beide bitstromen zijn dan uit fase en gemiddeld zal slechts de helft van de bits overeenstemmen: de overeenstemming berust geheel op toeval. Tel dus het aantal overeenstemmende bits, d.w.z. bereken de correlatie. Een uitdrukking analoog aan die voor de autocorrelatie in [vergelijking \(10.3\)](#). De juiste begininstelling verradt zich door een hoge waarde van die correlatie. In het ongunstigste geval moet men op deze manier voor een lfsr alle begintoestanden doorrekenen.

Is het beslist nodig om de correlatie te berekenen over een gehele periode van het lfsr? Meestal niet. Een kleiner stuk geeft vaak al genoeg zekerheid omtrent de juiste positionering van de bitstroom. Siegenthaler geeft in zijn artikel een berekening van het geschatte aantal bits nodig om met een bepaalde mate van onzekerheid een conclusie te kunnen trekken. Die berekening verdisconteert zowel de kans dat een foutieve positie wordt geaccepteerd (vals alarm), als de kans dat een goede positie wordt gemist. Afhankelijk van de lengte van het lfsr en het aantal te onderzoeken lfsr's, kan men bereken hoeveel bits in



Figuur 10.16 Correlatiesommen Geffe generator

de analyse moeten worden meegenomen om bijvoorbeeld gemiddeld hoogstens één vals alarm te genereren.

Als voorbeeld is een kleine Geffe generator genomen. LFSR-1 en LFSR-3 genereren maximaalrijen met respectievelijk periode 15 en 31. Het LFSR-2 heeft een periode van 7. Uitgerekend kan worden dat met ongeveer 100 bits in de correlatiesom niet meer dan 0,1% kans bestaat op missen van de goede positie of op meer dan één vals alarm. In [figuur 10.16](#) is voor een honderdtal posities uit de generatorstroom de correlatieberekening uitgevoerd. De periodes van LFSR-1 en LFSR-3 komen er zonder mankeren uit.

Bij LFSR-2 is alleen ruis te zien. Dat mag ook van dat register worden verwacht, omdat de correlatie hier precies 50% is. Maar na het vinden van LFSR-1 en LFSR-3 kan de bitstroom uit LFSR-2 toch voor een deel gereconstrueerd worden. Bedenk dat LFSR-2 tussen LFSR-1 en LFSR-3 kiest. Steeds als de uitvoerbits van die twee laatst verschillend zijn, kan zonder mankeren worden vastgesteld welk lfsr gekozen is. Daarmee is ook bekend welk bit LFSR-2 verlaten heeft. Van de vier mogelijke bitcombinaties uit LFSR-1 en LFSR-3 hebben twee een van elkaar verschillend bit. Men zal dus gemiddeld de helft van de bitstroom uit LFSR-2 kunnen reconstrueren.

De Pless generator en de Bruer generator zijn eveneens gevoelig voor een correlatieaanval. Bij de laatste valt in de productietabel ([figuur 10.15](#)) direct de correlatie van 75% tussen de uitvoerstroom en elk van de drie samenstellende lfsr's op. Meer lfsr's opnemen vermindert weliswaar de correlatie, maar toch niet zo veel als gewenst. Zelfs bij negen lfsr's is de correlatie nog ongeveer 64%. Het LFSR-19 uit de Pless generator ([figuur 10.13](#)) met alle 27594 daarvoor te kiezen primitieve functies, vergt slechts een kleine 350 bits voor het bepalen van terugkoppelfunctie en begintoestand.

10.7.3 MASTERSECTIE – Meier-Staffelbach-aanval

In de methode van Siegenthaler moet men voor elke begintoestand de correlatiesom uitrekenen en bij registers met veel registertrappen (bijvoorbeeld meer dan 100) brengt dat veel werk met zich mee. Door Meier en Staffelbach is een manier bedacht om het aantal te onderzoeken begintoestanden drastisch te

beperken^[18]. Deze aanval werkt wanneer het aantal taps (aantal termen in de terugkoppelfunctie) gering is, een situatie die veelal vanwege de efficiency is gekozen (eenvoud in implementatie). Om te zien hoe het werkt wordt **vergelijking (10.16)** te hulp geroepen. Deze vergelijking legt het verband tussen de bits in de uitvoerstream van een lfsr. Een voorbeeld maakt de uitleg het eenvoudigst. Veronderstel daarom dat lfsr $f(x) = 1 + x + x^{15}$ onderdeel is van de generator. Uit **vergelijking (10.16)** volgt voor het bit s_t dat op tijdstip t uit dit register komt (door de gekozen $f(x)$ doen alleen C_0 en C_1 mee)

$$s_t = s_{t-14} + s_{t-15} \quad (10.52)$$

Bit s_t werd geproduceerd uit de bits in S_0 en S_1 , neergezet in S_{14} en 15 stappen later uitgevoerd. Dit is echter niet de enige relatie waarin s_t voorkomt. Toen het namelijk zelf in S_0 en in S_1 zat nam het deel aan de productie van nieuwe bits. Dus ook

$$s_{t+14} = s_t + s_{t-1} \quad \text{en} \quad s_{t+15} = s_{t+1} + s_t \quad (10.53)$$

Voor de overzichtelijkheid gaan alle bits naar de linkerkant.

$$\begin{array}{rclclcl} s_t & = & s_{t-14} & + & s_{t-15} & & s_t + s_{t-14} & + & s_{t-15} & = & 0 \\ s_{t+14} & = & s_t & + & s_{t-1} & \rightarrow & s_t + s_{t-1} & + & s_{t+14} & = & 0 \\ s_{t+15} & = & s_{t+1} & + & s_t & & s_t + s_{t+1} & + & s_{t+15} & = & 0 \end{array} \quad (10.54)$$

Er zijn drie van deze uitdrukkingen omdat de terugkoppelfunctie twee taps telt. Algemeen geldt: k taps, dan $k + 1$ vergelijkingen waaraan s_t deelneemt. In deze drie (noem ze L_i) staat steeds hetzelfde bit s_t . Voor het gemak wordt daarom de index t weggelaten en wordt de som van de andere bits (zoals $s_{t-14} + s_{t-15}$) afgekort tot een variabele a_i

$$\begin{array}{rcl} L_1 & = & s + a_1 = 0 \\ L_2 & = & s + a_2 = 0 \\ L_3 & = & s + a_3 = 0 \end{array} \quad (10.55)$$

De vergelijkingen L_i geven de pariteit van een combinatie van uitvoerbits, in de literatuur wordt daarom wel de benaming *pariteitsfunctie* gebruikt. Ze gelden vanzelfsprekend exact omdat het allemaal bits uit

één lfsr betreft. Nu is echter niet de uitstroom s van het lfsr bekend, maar alleen de sleutelstroom z uit de generator. Daarom zijn het niet precies deze vergelijkingen, maar andere waarin de bits s uit het lfsr zijn vervangen door de overeenkomstige bits z uit de sleutelstroom van de generator. De sommen a_i worden overeenkomstig vervangen door de b_i waarin eveneens s_i is vervangen door z_i . En tenslotte verdwijnt het rechterdeel met $=0$ omdat de vergelijkingen niet meer voor de volle 100% opgaan, maar slechts voorzover volgt uit de correlatie. Alleen de vergelijkingen met alle $z_i = s_i$ voldoen echt.

$$\begin{aligned} L_1 &= z + b_1 \\ L_2 &= z + b_2 \\ L_3 &= z + b_3 \end{aligned} \tag{10.56}$$

Het ware mooi geweest, wanneer alle vergelijkingen met uitkomst $L_i = 0$ gebruikt konden worden om vast te stellen of op die plaats $z = s$. Maar helaas ligt het ingewikkelder. Want ook als niet alle betrokken bits overeenkomen kan toch $L_i = 0$ zijn. De vergelijkingen L_i hebben immers in twee gevallen de waarde nul

1. als inderdaad $z = s$ en $a_i = b_i$ zodat $L = z + b = s + a = 0$
voor $a_i = b_i$ is het voldoende dat een even aantal bits uit b_i verschilt
2. als weliswaar $z \neq s$ maar ook $a_i \neq b_i$ zodat toch $L = z + b = s + a = 0$
in b_i mag een oneven aantal bits verschillend zijn

Het kan dus voorkomen dat $L = 0$ niet betekent dat ook $z = s$. Gelukkig is er een uitweg. Deze wordt gevonden in het feit dat er verschillende vergelijkingen L_i zijn. Als de meerderheid van die vergelijkingen $L_i = 0$ laat zien dan is er een behoorlijke kans dat inderdaad $z = s$ zal zijn. Het is dus zaak uit te rekenen met welke kans $p(a = b)$ de som van alle bits in b gelijk zal zijn aan a . Gegeven is daarbij dat voor elk bit afzonderlijk die kans gelijk is aan de correlatie. Daarmee valt uit te rekenen met welke kans $L = 0$ geldt (beide bovenstaande gevallen moeten worden meegenomen) en vervolgens voor welk gedeelte dit mag worden toegeschreven aan de gezochte vaststelling $z = s$. Tenslotte wordt de kans $p(z = s)$ berekend, gegeven het werkelijk aangetroffen aantal vergelijkingen met $L_i = 0$.

Het resultaat van deze berekening luidt als volgt. Hierin is

- q de berekende kans dat $a_i = b_i$,
- h het aantal vergelijkingen waarvoor $L_i = 0$ wordt geconstateerd,
- m het totaal aantal vergelijkingen, en
- p de correlatie tussen lfsr en sleutelstroom.

De berekening van $q = q(1 \cdots k)$ volgt uit een recursie over de k tappunten van het lfsr

$$q_k = pq_{k-1} + (1-p)(1-q_{k-1}) \text{ met } q_1 = p \quad (10.57)$$

De gezochte kans wordt gegeven door de binomiale verdeling voor willekeurige keuze van h uit m vergelijkingen

$$\begin{aligned} p(z = s) &= p(z = s \mid L_1 = \cdots = L_h = 0, L_{h+1} = \cdots = L_m = 1) \\ &= \frac{pq^h(1-q)^{m-h}}{pq^h(1-q)^{m-h} + (1-p)(1-q)^h q^{m-h}} \end{aligned} \quad (10.58)$$

Met het uitrekenen van deze kans staat een criterium ter beschikking voor de beslissing of $z = s$ al dan niet kan worden aanvaard. Stel daartoe een grenswaarde vast waarboven die aanvaarding geschiedt. Op deze manier kan van een aantal bits uit het lfsr de waarde worden bepaald. Misschien zijn er wel enkele onjuist en misschien ook ontbreken een paar van de benodigde bits. Dan toch behoeft, in vergelijking met de in het voorgaande behandelde methode Siegenthaler, nog maar een beperkt aantal mogelijkheden te worden getest.

Het is echter te vroeg om al te juichen. In het gegeven voorbeeld van $f(x) = 1 + x + x^{15}$ zijn er slechts drie vergelijkingen en dat is te weinig voor een betrouwbare conclusie*. Het is zou prettig zijn als het aantal vergelijkingen kon worden vergroot en daartoe biedt machtsverheffing van $f(x)$ (vergelijking (10.44)) de

* Weliswaar geven terugkoppelfuncties met meer taps meer vergelijkingen, maar het wordt daar niet beter van als gevolg van de negatieve effecten van meer taps op de kans dat $L_i = 0$.

kans. Uitgelegd is dat $f(x)^n$ terugkoppelfuncties oplevert die dezelfde stroom produceren als $f(x)$. Als n ook nog eens een macht van 2 is dan leidt dit tot een terugkoppelfunctie met ongewijzigd aantal taps. Hiervan wordt nu goed gebruik gemaakt. Het levert extra vergelijkingen op, terwijl het gelijkblijvend aantal taps de kansen niet ongunstig beïnvloedt. In het voorbeeld van $f(x) = 1 + x + x^{15}$ worden daarom ook $f(x)^2 = 1 + x^2 + x^{30}$, $f(x)^4 = 1 + x^4 + x^{60}$, enz. benut. Men kan zo ver doorgaan als de lengte van de bekende sleutelstroom toestaat

$$\begin{aligned}
 s_t &= s_{t-28} + s_{t-30} \\
 s_{t+28} &= s_t + s_{t-2} \\
 s_{t+30} &= s_{t+2} + s_t \\
 s_t &= s_{t-56} + s_{t-60} \\
 &\dots
 \end{aligned}
 \tag{10.59}$$

Een verdere verfijning van de Meier-Staffelbach-aanval is mogelijk. Eerst wordt berekend welke kans elk bit in de sleutelstroom heeft om overeen te komen met het bit uit het gezochte lfsr. Uitgangspunt voor deze berekening zijn de vergelijkingen (10.57) en (10.58). Vervolgens wordt de berekening herhaald met voor elk bit de afzonderlijk berekende waarde van $p(z_i = s_i) = p_i$. Daartoe moet (10.58) worden aangepast. In de berekening van q uit (10.57) zullen de verschillende p_i van de onderscheiden bits een rol spelen en in plaats van de binomiale verdeling in (10.58) komt het product over $L_i = 0$ en $L_i = 1$ met afzonderlijke berekening voor elke i .

$$\begin{aligned}
 p'(z = s) &= p'(z = s | L_i \dots = 0, L_j \dots = 1) \\
 &= \frac{p_i \prod_{i:L=0} q_i \prod_{j:L=1} (1 - q_j)}{p_i \prod_{i:L=0} q_i \prod_{j:L=1} (1 - q_j) + (1 - p_i) \prod_{i:L=0} (1 - q_i) \prod_{j:L=1} q_j}
 \end{aligned}
 \tag{10.60}$$

Na enkele iteraties zullen de berekende individuele p_i 's niet veel meer veranderen en kunnen de bits met een lage p_i van waarde wisselen. Als het grootste deel van deze veranderingen terecht is geschied, zal daardoor het percentage bits met $z = s$ zijn toegenomen. De individuele correlaties p_i worden teruggezet naar p en het proces wordt opnieuw gestart. Dit gaat door totdat de bitstroom van het lfsr geheel of

nagenoeg geheel is bepaald.

Hoe effectief is deze aanval? Meier en Staffelbach geven voor de eenvoudige aanvalsmethode een voorbeeld van een lfsr met 500 registertrappen, 2 tappunten, correlatiefactor 0,75 en 50000 bits uit de sleutelstroom. Volgens hun kansberekening zullen 547 bits van die 50000 voor minstens 11 van de 12 opgestelde vergelijkingen $L = 0$ te zien geven en is voor hooguit 1 van die 547 de conclusie $z = s$ niet gewettigd. In een tweede voorbeeld laten ze zien hoe goed de verder verfijnde aanval werkt. Bij een correlatie van 0,75, een register van 100 trappen met 2 tappunten en 5000 waargenomen bits komen in de eerste ronde 793 bits in aanmerking voor waardeverandering; daardoor daalt naar verwachting het aantal verkeerde bits van 1250 naar 758.

Deze aanval wordt zelfs voor zeer grote lfsr's effectief geacht, tot wel lengte 1000 toe. De effectiviteit neemt echter snel af als het aantal tappunten in de terugkoppelfunctie toeneemt; bij 8 of meer tappunten is deze methode al nagenoeg hulpeloos. Dit komt omdat bij meer tappunten de sommen b_i meer termen bevatten, waardoor de kans op $p(b_i = 0) \approx 0,5$ snel toeneemt. Bijgevolg komt er meer onzekerheid over de voorspellende waarde van de vergelijkingen L . Veel tappunten maken het nagenoeg onmogelijk om er nog iets uit af te leiden en verhinderen dus conclusies over $z = s$. Een overzicht en vergelijking van een aantal op correlatie gebaseerde aanvalsmethoden in de recente literatuur kan worden gevonden in een artikel van Clark en andere^[19].

	ronde 1	ronde 2	ronde 3	ronde 4
goed:	48	58	62	64
veranderd:	16	6	2	0
ten onrechte veranderd:	3	1	0	0
ten onrechte niet veranderd:	3	1	0	0

Tabel 10.13 Voorbeeld Meier-Staffelbach-aanval op Geffe-generator

ronde 1					ronde 2					ronde 3			ronde 4		
z	h	p_1	p_2	p_3	s	h	p_2	p_2	p_3	s	h	p_3	s	h	p_3
1	0	0.28	0.96	0.01	0	2	0.75	0.52	0.52	0	0	0.0	1	4	1.0
0	2	0.64	0.94	0.89	0	5	0.98	1.00	1.00	0	4	1.0	0	5	1.0
0	3	0.75	0.86	0.80	0	4	0.90	1.00	1.00	0	5	1.0	0	6	1.0
0	2	0.52	0.73	0.32	0	2	0.52	0.97	0.30	0	4	1.0	0	6	1.0
0	3	0.52	0.89	0.34	0	2	0.28	1.00	0.00	1	5	1.0	1	8	1.0
0	4	0.75	0.91	0.99	0	6	0.96	1.00	1.00	0	7	1.0	0	8	1.0
0	4	0.75	0.91	0.98	0	7	0.98	1.00	1.00	0	7	1.0	0	8	1.0
1	6	0.96	1.00	1.00	1	2	0.28	1.00	0.02	0	0	0.0	1	8	1.0
0	4	0.52	0.84	0.07	1	5	0.75	1.00	1.00	1	7	1.0	1	10	1.0
1	3	0.28	0.70	0.03	0	8	0.98	1.00	1.00	0	9	1.0	0	10	1.0
1	4	0.52	0.97	0.34	1	7	0.96	1.00	1.00	1	9	1.0	1	10	1.0
1	6	0.89	1.00	1.00	1	4	0.52	0.79	0.00	0	8	1.0	0	10	1.0
1	7	0.96	1.00	1.00	1	8	0.98	1.00	1.00	1	10	1.0	1	10	1.0
1	4	0.52	0.96	0.88	1	9	0.99	1.00	1.00	1	9	1.0	1	10	1.0
1	8	0.98	1.00	1.00	1	7	0.96	1.00	1.00	1	10	1.0	1	10	1.0
0	6	0.89	1.00	1.00	0	4	0.52	0.70	0.00	1	8	1.0	1	10	1.0

Tabel 10.14 Verloop Meier-Staffelbach-analyse voor 16 bits

Toegepast op dezelfde Geffe generator als in het voorbeeld van de Siegenthaler-correlatieaanval toont [tabel 10.13](#) het effect van de geperfectioneerde aanval op LFSR-1 met slechts 64 bits sleutelstroom. In elke ronde is driemaal geïtereerd op de kansen p_i , waarna van de bits met $p_i < 0,1$ het complement werd genomen. Na drie zulke rondes blijken alle 64 bits juist bepaald. De andere tabel laat van de bits 1–16 zien hoe in opeenvolgende iteraties in de eerste twee rondes de bits en kansen (p_1 , p_2 , p_3) veranderen.

10.7.4 Correlatie-immuniteit

Een functie kan tot op zekere hoogte ongevoelig zijn voor correlatie. Dit wordt *correlatie-immuniteit*

genoemd. Het begrip is door Siegenthaler geïntroduceerd en gedefinieerd^[20].

Uit n onderling onafhankelijke, uniform verdeelde random variabelen $S_1 \dots S_n$ wordt een sleutelstroom gemaakt

$$z = f(S_1, \dots, S_n) \quad (10.61)$$

Als bron voor deze variabelen kan men lfsr's nemen. De functie f heet orde m correlatie-immuun met $1 \leq m < n$ als elk willekeurig m -tupel $S_1, \dots, S_m$ uit $S_1, \dots, S_n$ statistisch onafhankelijk is van z . Men kan dit ook beredeneren vanuit de informatietheorie. Correlatie-immuniteit van orde m is dan equivalent met de eis dat de wederzijdse informatie tussen z en elk willekeurig m -tupel nul is. Het betekent in feite dat geen enkel m -tupel informatie levert over z en omgekeerd.

$$W(S_1, \dots, S_m, z) = 0, \quad \forall S_1, \dots, S_m \in \{S_1, \dots, S_n\} \quad (10.62)$$

Wanneer is een functie correlatie-immuun? Daartoe wordt de functie in algebraïsche normaalvorm geschreven; zie [vergelijking \(10.9\)](#). Dat wil zeggen als een som van termen $1, S_i, S_i S_j, S_i S_j S_k$, enz. Een functie z van n variabelen blijkt correlatie immuun van orde m als in de algebraïsche normaalvorm geen termen met meer dan $n - m$ factoren voorkomen.

Deze voorwaarde is voor de opwekking van bruikbare sleutelstromen echter nog onvoldoende. In het algemeen zal men een *gebalanceerde stroom* willen produceren, waarmee wordt bedoeld dat nullen en enen even vaak voorkomen. Anders gezegd, $P(z = 0) = P(z = 1) = 0,5$ moet gelden.

Drie gevallen worden onderscheiden

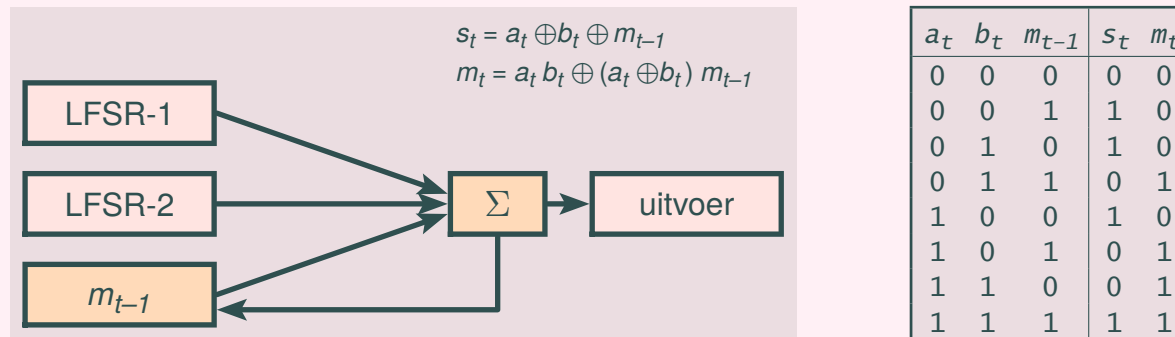
1. $m = 0$: Er is geen sprake van correlatie-immuniteit, alle termen tot en met $S_1 \dots S_n$ kunnen voorkomen.
2. $m = n - 1$: De hoogst mogelijke vorm van correlatie-immuniteit, waaraan slechts de som van de afzonderlijke bitstromen voldoet $f = S_1 + \dots + S_n$ en $f = 1 + S_1 + \dots + S_n$
3. $1 \leq m < n - 1$: Voor gebalanceerde bitstromen wordt de eis verzwakt tot het voorkomen van termen met ten hoogste $n - m - 1$ factoren, één minder dus dan voor ongebalanceerde bitstromen.

Aan de hand van bovenstaande eisen valt in te zien dat voor de Geffe generator – een gebalanceerde generator met $z = S_1 + S_1S_2 + S_2S_3$ – de aanwezigheid van de twee producten S_1S_2 en S_2S_3 de correlatie-immuniteit verhindert. Want zelfs voor $m = 1$ is het hoogst toegestane product al overschreden. Een gebalanceerde generator van n variabelen met minimale correlatie immuniteit van 1 kan geen termen van n of $n - 1$ factoren bevatten.

Bij de volgende generator die wordt behandeld, kan men zien hoe geprobeerd is het probleem van de correlatie-immuniteit te omzeilen. Dat lukt tot op zekere hoogte, de correlatie tussen de sleutelstroom en de deelnemende lfsr's kan worden onderdrukt. Jammer genoeg blijkt dan een andere vorm van correlatie diens plaats in te nemen.

10.7.5 Somgenerator

Het verlangen naar correlatie-immuniteit – en dus meer bestendigheid tegen een correlatieaanval – brengt met zich mee dat in de generatorfunctie termen met groot aantal S -factoren moeten worden vermeden, al gaat dit ten koste van een hoog lineair equivalent. Er blijkt een uitweg uit het daardoor opgeroepen dilemma. De correlatie-immuniteit kan worden verbeterd wanneer de generator wordt voorzien van een geheugenfunctie^[21]. Eén geheugenbit blijkt al voldoende om de koppeling tussen correlatie-immuniteit en het hoog lineair equivalent op te heffen. Let op, dit betekent niet dat een op dit geheugenprincipe gebaseerde generator bestand zal zijn tegen elke soort van correlatieaanval. Correlatie met de onderliggende deelstromen is niet de enig denkbare vorm van correlatie, terwijl ook het aantal in de generator opgenomen geheugenbits van invloed is op de mate van correlatiebestendigheid^[22].



Figuur 10.17 Somgenerator met productietabel

De eerder behandelde Pless generator maakt door middel van de JK-flipflop gebruik van geheugen. Als gevolg van het feit dat die geheugenfunctie niet expliciet wordt benut, is daar desondanks sprake van een correlatie met de samenstellende lfsr's. De *somgenerator* is een voorbeeld waarin wel expliciet geheugen wordt gebruikt. In zijn eenvoudigste vorm is deze generator afgebeeld in [figuur 10.17](#). Twee lfsr's, respectievelijk LFSR-1 met uitvoer a en LFSR-2 met uitvoer b , worden gekoppeld met een geheugenbit m . Bij elke stap wordt het geheugenbit meegenomen in de berekening van de uitvoer en krijgt het geheugenbit een nieuwe waarde. De tabel laat voor alle bitcombinaties de resulterende sleutel- en geheugenbits zien. Deze generator is weliswaar correlatie-immuun in voorgaande zin, maar in plaats daarvan duikt wel een andere correlatie op. Er is namelijk een anticorrelatie zichtbaar tussen het gegenereerde sleutelbit s en het nieuwe geheugenbit m . Men ziet in de rechterkolom van de productietabel dat die twee bits zes van de acht keer elkaars complement zijn.

MASTERSECTIE – Cryptoanalyse somgenerator

Geschetst wordt een door Dawson beschreven aanval op deze somgenerator^[23]. Veronderstel dat een deel van de sleutelstroom van de generator is bepaald en dat bovendien de lfsr's bekend zijn. Probeer alle mogelijke combinaties van beginwaarden van het geheugenbit en het kleinste van de twee lfsr's. Voor

deze combinaties wordt uit de bekende sleutelstroom een deel van de uitvoer van LFSR-2 berekend met $b_t = s_t \oplus a_t \oplus \sigma_{t-1}$. Zodra n_2 bits zijn gevonden kan de rest van de uitvoer van LFSR-2 ook worden berekend. Daarop kan het volgende bit van s worden berekend en vervolgens vergeleken met de bekende waarde. Als de gekozen begintoestand de juiste was zal deze vergelijking steeds kloppen, zoniet dan bestaat er in elke stap een kans 0,5 op een tegenspraak. Men gaat zolang door met proberen totdat uit een lange rij goede uitkomsten met aan zekerheid grenzende waarschijnlijkheid de juiste oplossing blijkt.

f_1	1010111100010011010111100010011010111100010011010111
f_2	1101001000010101110110001111100110100100001010111011
s	1000001010001110011010011100000011000110011000010011
σ	11111111100010001110111100011111110111100000011111111

beginwaarde $\sigma=0$														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0		0			0		0		0		0	0	
			1									1		
			0											

beginwaarde $\sigma=1$														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		0		0	0		0	0	0		0			0
				1	1		1	1	1					
				0					0					
				1										
				0										

Tabel 10.15 Cryptoanalyse somgenerator

Deze analyse vergt het afzoeken van maximaal $2(2^{n_1}-1)$ begintoestanden (de periode van deze generator),

voor elk waarvan tenminste $n_2 + 1$ bits berekend moeten worden. Het aantal bits nodig om de goede oplossing te verifiëren blijkt gering, ongeveer $n_1 + n_2 + 1$ voldoet. Wanneer de tappunten van LFSR-2 onbekend zijn, kan men $2n_2$ bits bepalen, daaruit een terugkoppelfunctie berekenen en hiermee de verificatie op de volgende bits van s uitvoeren.

In tabel 10.15 is deze analyse uitgevoerd op een systeem met de twee lfsr's $f_1 = 1 + x + x^4$ en $f_2 = 1 + x^3 + x^5$. Ondubbelzinnig kan de juiste begininstelling $\sigma = 1$, $f_1 = 0101$ en (daaruit) $f_2 = 01011$ worden bepaald. De helft (15 beginwaarden) valt meteen uit, een kwart (8) gaat 1 stap goed, een achtste gaat 2 stappen mee en tenslotte zijn er twee beginwaarden die het 3 stappen volhouden; daarna blijft slechts de goede oplossing over.

10.8 Klokgestuurde registers

Als laatste serie van de op lfsr's gebaseerde generatoren volgen nu constructies waarin de niet-lineariteit wordt bewerkstelligd door het manipuleren van de kloksturing. Zowel de toestandsverandering van een of meer van de samenstellende lfsr's, als de productie van diens uitvoerbit kunnen worden onderbroken. Met als gevolg dat niet meer elk register in elke cyclus een stap doet of een bit uitvoert. De besturing van deze onregelmatige registerloop geschiedt door een (ander) lfsr. Het gestuurde lfsr gaat onregelmatig lopen en kan gedurende één of meer cycli van de generator stil blijven staan of geen uitvoerbit geven. In het laatste geval moet worden gezorgd dat het volgend uitvoerbit niet te lang op zich laten wachten, want anders kan het proces dat gebruik maakt van de generator niet snel genoeg verder. Dit probleem kan bijvoorbeeld worden ondervangen door buffering of door het verhogen van de kloksnelheid.

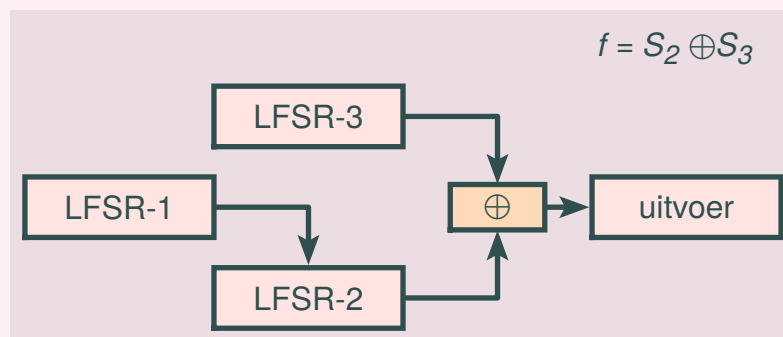
Klokgestuurde registercombinaties hebben wijd verspreide toepassing gevonden in telefoontoestellen. In mobiele telefoons vooral in het A5/1-mechanisme dat verderop in dit hoofdstuk wordt beschreven. Draadloze huistelefoons werken meest met het DECT-systeem. Evenals A5/1 aanvankelijk een geheimgehouden

algoritme voor encryptie en authenticatie, maar net als bij A5/1 uiteindelijk d.m.v. reverse engineering ontraadseld^[24].

10.8.1 Stop-and-go generator

De eerste uitwerking van dit idee van een onregelmatig lopend register is de *stop-and-go generator* van Beth en Piper, afgebeeld in [figuur 10.18](#)^[25]. In deze generator wordt de klok van LFSR-2 gestuurd door de uitvoer van LFSR-1. Is het uitvoerbit van LFSR-1 een 1 dan zal LFSR-2 in de volgende cyclus een stap doen, is het een 0 dan blijft LFSR-2 in de volgende cyclus stilstaan. In dit laatste geval zijn twee opvolgende bits van LFSR-2 gelijk. Het derde register LFSR-3 wordt normaal geklokt en diens uitvoerbit wordt opgeteld bij dat van LFSR-2. De uitvoer is statistisch van voldoende kwaliteit en met een geschikte keuze van de samenstellende registers worden de maximale periode en een hoge lineaire complexiteit bereikt.

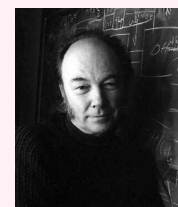
$$\begin{aligned} p &= (2^{n_1} - 1)(2^{n_2} - 1)(2^{n_3} - 1) \\ LC &= (2^{n_1} - 1)n_2 + n_3 \end{aligned} \quad (10.63)$$



Figuur 10.18 Beth-Piper stop-and-go generator



Beth



Piper

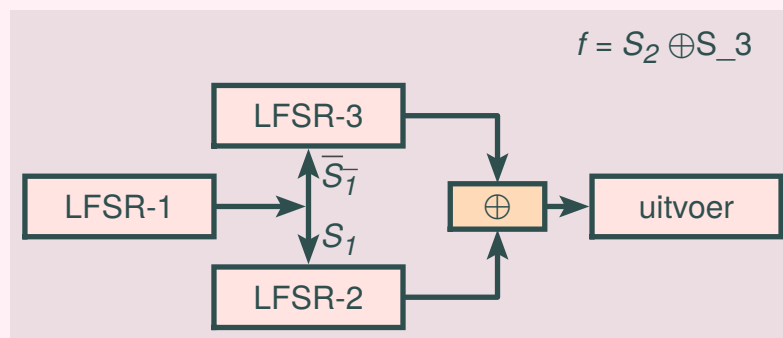
De bestendigheid tegen cryptoanalyse is echter niet groot als gevolg van (opnieuw) een correlatie-effect^[26]. Deze correlatie treedt op tussen de som van de opvolgende uitvoerbits $s_t \oplus s_{t+1}$ van de generator en de som van de opvolgende uitvoerbits van LFSR-3: $c_t \oplus c_{t+1}$. Die correlatie is als volgt te berekenen

$$p(s_t \oplus s_{t+1} = c_t \oplus c_{t+1}) = p(s_t \oplus s_{t+1} = b_t \oplus c_t \oplus b_{t+1} \oplus c_{t+1}) \quad (10.64)$$

Als $b_t = b_{t+1}$, d.w.z. als twee opvolgende bits van LFSR-2 gelijk zijn, dan vallen de b 's tegen elkaar weg

$$s_t \oplus s_{t+1} = c_t \oplus c_{t+1} \quad (10.65)$$

De grootte van deze correlatie is te vinden door de kans op twee gelijke opvolgende bits in LFSR-2 te berekenen. Dit is het geval wanneer het voorafgaande bit van LFSR-1 een 0 is, want dan staat LFSR-2 stil. Deze kans is $1/2$ omdat LFSR-1 praktisch evenveel 0'en als 1'en heeft. Maar ook als uit LFSR-1 een 1 komt bestaat de mogelijkheid dat LFSR-2 achtereenvolgens twee gelijke bits produceert. De kans op zo'n gebeurtenis is $1/4$: namelijk $1/2$ voor een 1 uit LFSR-1 maal $1/2$ voor twee gelijke bits uit LFSR-2. In totaal is er dus kans $3/4$ dat LFSR-2 tweemaal achtereenvolgens hetzelfde bit levert en de gezochte correlatie is derhalve 75%. Die hoge waarde van 75% maakt reconstructie van de begintoestand van LFSR-3 mogelijk en daaruit volgt weer die van LFSR-1. Het is niet eens nodig om de karakteristieke functies van alledrie de lfsr's te kennen. Die van LFSR-2 kan worden bepaald uit de gereconstrueerde uitvoer van dit register.



Figuur 10.19 Alternierende stop-and-go generator

Een variatie op de Beth-Piper constructie is de *alternerende stop-and-go generator* van [figuur 10.19](#). Hierin worden de klokken van LFSR-2 en LFSR-3 allebei gestuurd door de uitvoer van LFSR-1. Bij een 1 loopt LFSR-2 en staat LFSR-3 stil, bij een 0 is het net andersom. Het beste resultaat wordt bereikt door de lengtes van de drie lfsr's ongeveer evengroot te nemen en te zorgen dat ze onderling ondeelbaar zijn. De best bekende aanvalsmethode tegen deze generator loopt via een correlatie met de uitvoer van LFSR-1^[27]. Deze generator vertoont in opzet een treffende gelijkenis met de eerder behandelde Geffe-generator ([sectie 10.6.2](#)).

10.8.2 Krimpgenerator

Een van de meer recente varianten van dit soort generatoren is de *krimpgenerator* (shrinking generator in het Engels). De niet-lineariteit is hier het gevolg van het slechts gedeeltelijk benutten van de uitvoer van een of meer samenstellende lfsr's. In de eenvoudigste vorm bestaat deze generator uit twee lfsr's^[28]. Beide worden gelijk geklokt, maar als de uitvoer van LFSR-1 een 0 is wordt de uitvoer van LFSR-2 niet gebruikt. In het andere geval geeft LFSR-1 een 1 en is de uitvoer van de generator het uitvoerbit van LFSR-2. Een nadeel is de onregelmatige uitvoerproductie die de bitstroom telkens even doet stokken. De maximale stilstand treedt op wanneer de langst mogelijke rij nullen uit het register komt, namelijk 1 minder dan het aantal registertrappen. In formule luidt de selectie van bits uit LFSR-2 door de enen uit LFSR-1

$$s_t = b_t \wedge a_t = 1 \quad (10.66)$$

Een klein voorbeeld met een volledige periode van de krimpgenerator staat in [tabel 10.16](#). Dit voorbeeld laat van deze generator de complete cyclus van 21 stappen zien. De samenstellende registers hebben hier een periode van respectievelijk 3 en 7. De uitgevoerde sleutelstroom heeft een periode van lengte 14. Met behulp van de Berlekamp-Massey algoritme kan de lineaire complexiteit daarvan op 6 worden bepaald.

$1 + x^2$	101101101101101101...
$1 + x + x^3$	100101110010111001011...
uitvoer	1 01 11 00 01 10 10 1...

Tabel 10.16 Voorbeeld krimpgenerator

In het algemeen geldt voor de periode en de lineaire complexiteit van de krimpgenerator bij welgekozen lfsr's

$$p = 2^{n_1-1}(2^{n_2} - 1) \quad \text{en} \quad LC = 2^{n_1-2} < LC \leq n_2 \cdot 2^{n_1-1} \quad (10.67)$$

De *self-shrinking generator* is een variant waarin nog maar één lfsr nodig is^[29]. Steeds worden twee opeenvolgende bits bekeken. Is het eerste bit een 0 dan blijft het tweede bit ongebruikt, is het eerste bit een 1 dan wordt het tweede bit de uitvoer: dus 00 en 01 worden niet gebruikt, 10 en 11 leveren 0 respectievelijk 1. Deze generator loopt dus gemiddeld op halve snelheid ten opzichte van het genererend lfsr. De beide vormen van krimpgenerator zijn nauw aan elkaar verwant. Aangevoerd kan worden dat een krimpgenerator kan worden geïmplementeerd met een zelfkrimpende generator en omgekeerd.

$1 + x + x^3$	100101110010111001011...
uitvoer	0 1 0 1 0 ...

Tabel 10.17 Voorbeeld zelfkrimpende-generator

De bits van de uitvoerstream voldoen aan

$$s_t = a_{2t+1} \wedge a_{2t} = 1 \quad (10.68)$$

Voor de periode en de lineaire complexiteit vindt men

$$p \geq 2^{\frac{n}{2}} \quad \text{en} \quad LC > 2^{\frac{n}{2}-1} \quad (10.69)$$

10.8.3 GSM A5/1-generator

In de GSM-telefonie worden verscheidene crypto-algoritmen gebruikt, een daarvan is de *A5-algoritme* voor de bescherming van het spraaksignaal. Van dit algoritme bestonden oorspronkelijk de varianten A5/0, A5/1 en A5/2. Later is daar de KASUMI-algoritme als A5/3 bijgekomen. De beide varianten A5/1 en A5/2 zijn echte geheimschriften, A5/0 wordt wel gebezigd in het geval het spraaksignaal zonder vercijfering wordt overgezonden. A5/1 is de sterkste van de twee en wordt in Europa gebruikt*. De A5/1-algoritme is niet publiekelijk bekend gemaakt, maar wordt onder voorwaarde van geheimhouding verstrekt aan fabrikanten van GSM-SIM kaarten. Details ervan zijn echter in 1995 uitgelekt en vervolgens in 1999 geverifieerd door reverse engineering aan de hand van SIM-kaarten. Vanwege het onofficiële karakter van de beschrijving spreken veel artikelen van de *veronderstelde* (alleged in het Engels) A5/1.

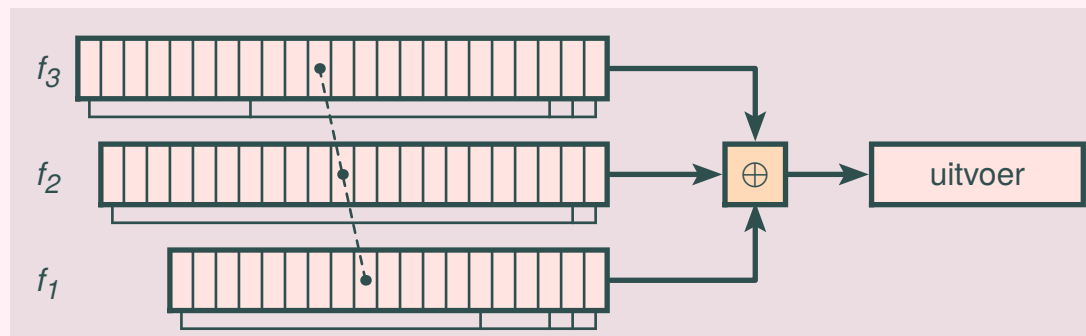
Een gesprek over een GSM-telefoon begint met een authenticatiefase en met het overdragen van de sleutel waarmee de vercijfering wordt opgestart. De conversatie zelf is opgedeeld in stukjes (frames) van 228 bits, tijdsduur 4,6 milliseconden, met in ieder frame 114 bits voor elk van de beide verkeersrichtingen. Voorafgaand aan ieder frame worden de lfsr's opnieuw ingesteld met behulp van een in de Subscriber Identity Module (SIM kaart) gegenereerde sleutel in combinatie met een doorlopende frameteller. De frameteller wordt in een 22 bits register bijgehouden, hetgeen voldoende is om zonder sleutelherhaling een gesprek van ruim vijf uur mogelijk te maken.

* Een gevreesde luistervink was de voormalige DDR met grote installaties op de Brocken in de Harz, juist op de grens met de Bondsrepubliek. Het is niet onwaarschijnlijk dat dit een belangrijke reden was om voor een redelijk sterk algoritme te kiezen. Begin 2014 verscheen een bericht (afkomstig van de Noorse krant Afterposten) dat oud-medewerkers van KPN hebben toegegeven dat waarschijnlijk onder druk van het Britse GCHQ de encryptie van A5/1 zwakker is gemaakt dan oorspronkelijk in de bedoeling lag. In plaats van de voorgestelde 128-bits sleutel wilde die dienst niet meer dan 48 bits. Bij wijze van compromis zijn het er tenslotte 54 geworden (van de 64 bits van de sleutel zijn de laatste 10 bits altijd nul).

A5/1 bestaat uit een combinatie van drie lfsr's met lengtes van 19, 22 en 23 bits. De karakteristieke functies zijn

$$f_1 = 1 + x + x^2 + x^5 + x^{19} \quad f_2 = 1 + x + x^{22} \quad f_3 = 1 + x + x^2 + x^{15} + x^{23} \quad (10.70)$$

De klok van elk register volgt het stop-and-go principe en wordt bepaald door een combinatie van drie registertrappen, de *klok-bits*, één in elk register. In f_1 is dit het bit in registertrap S_{10} , in f_2 is het S_{11} en in f_3 is het S_{12} . De registers waarvan het klokbit overeenkomt met de meerderheid doen een stap. Dus als het klokbit in f_1 , f_2 , f_3 achtereenvolgens 1, 1 en 0 is, doen de f_1 en f_2 registers een stap en blijft dat van f_3 stilstaan. Van de acht mogelijke bitcombinaties zijn er twee waarin elk register een stap doet en zes waarin een van de drie blijft staan. Ieder register heeft zodoende een kans van 75% om door te lopen. De uitvoerbits van de drie lfsr's worden bij elkaar opgeteld (uiteraard modulo 2), zie [figuur 10.20](#). De uitvoer van de generator valt in verscheidene cycli uiteen, de kortste daarvan heeft een lengte van $\approx 2^{23}$. De statistische kwaliteit van de generator is goed.



Figuur 10.20 GSM A5/1-generator

Na een eerste poging tot cryptanalyse door Golić, is deze door Biryukov en Shamir vervolmaakt tot een praktisch uitvoerbare aanval die eind 1999 gepubliceerd is^[30]. De cryptanalyse berust op een time-memory tradeoff. Uit een ongeveer twee minuten durende conversatie kon de instelsleutel wor-

den bepaald. De voor de analyse gebruikte apparatuur bestond uit een personal computer met 128 MB intern geheugen en ongeveer 150 GB opslag voor de voorberekeningen. Daarmee waren de onderzoekers in staat om een sleutel in minder dan een seconde te vinden.

Nadien zijn nog andere aanvalsmethoden bedacht. Zo kan gebruik gemaakt worden van het feit dat A5/1 en A5/2 voor vercijfering dezelfde sleutel gebruiken^[31]. Nu is A5/2 verreweg de zwakste van de twee en kan met minder dan een seconde conversatie in een ciphertext-only aanval worden gebroken. Deze aanval berust op het gebruik van foutcorrigerende codes voorafgaand aan de vercijferstap. Als de aangevallen GSM ertoe gebracht kan worden om A5/2 encryptie te genereren (veelal aanwezig voor compatibiliteit met netwerken die A5/1 niet hebben), wordt langs deze weg de sessiesleutel gevonden en kan vervolgens na omschakeling op A5/1 de conversatie verder gevolgd worden.

Een meer rechtstreekse, known-plaintext aanval, benut de manier waarop de vercijfersleutel wordt ingesteld^[32]. Per gesprek is er een verschillende 64-bits sleutel, die tijdens het gesprek wordt gecombineerd met de 22-bits frameteller. Bij het vercijferen van het steeds 228 bits tellend conversatieframe worden de volgende stappen doorlopen

1. alle lfsr's worden op 0 geïnitieerd
2. elk register doet 64 keer een stap waarbij telkens een exclusive or van het feedbackbit en het volgend bit van de sessiesleutel wordt ingevoerd
3. elk register doet 22 een keer een stap waarbij ditmaal een bit van de frameteller meedoet
4. vanaf dit punt wordt het stop-go mechanisme ingeschakeld en loopt de gehele registercombinatie eerst 100 stappen zonder het uitvoerbit te gebruiken
5. de volgende 228 uitvoerbits worden voor de vercijfering van het conversatieframe gebruikt

De aanval maakt gebruik van het feit dat het laden van de sessiesleutel en de frameteller apart geschiedt. Omdat elk register in elke stap 75% kans heeft om door te lopen, zullen de drie registers bij het begin van de framevercijfering gemiddeld 75 stappen achter de rug hebben. Er is bijvoorbeeld een kans van

0,066% dat bij het 6^e bit van de vercijfering de drie lfsr's na de instelling van de sleutel allemaal precies 80 stappen hebben gedaan. Vanaf het moment waarop de vercijfering van het gesprek begint kan nu een correlatieaanval worden opgezet.

Een meer verfijnde aanval op A5/1 wordt gerapporteerd in een artikel van Shamir e.a. over de cryptanalyse van A5/3^[33]. Met een door de Duitse 'hacker' Karsten Nohl in 2009 geproduceerde regenboogtabel van twee Terabyte zou het terugvinden van een A5/1 sessiesleutel zeer vergemakkelijkt worden.*

Een van de laatste ontwikkelingen in de voortgaande saga van het breken van A5/1 is een aanval die minder dan 6 Gigabyte aan gegevensopslag vergt en 100% kans op success belooft. Evenals andere aanvalsmethoden heeft deze genoeg aan het bekend zijn van 64 bits sleutelstroom. Voor alle toestanden van het kleinste 19-traps register is een voorberekening gemaakt voor de bepaling van de toestand van de beide andere registers. Vergelijk nu alle toestanden van dit 19-bit register en de er uit af te leiden toestanden van de beide andere registers één voor één met de werkelijk gerealiseerde bits van de sleutelstroom. Dan blijkt snel welke toestanden afvallen. Na verwerking van hoogstens 16 van de benodigde 64 sleutelstroombits bevindt zich de goede combinatie van alle drie registers gegarandeerd onder de resterende mogelijkheden en is het nog slechts een kwestie van controleren welke de juiste is.^[34]

Uit een van de in 2013 door Snowden gelekte documenten kan de conclusie getrokken worden dat de NSA geen moeite heeft met A5/1-encryptie, ook als het de sleutel niet kent. Dat overigens de interesse van geheime diensten voor schuifregisters niet van vandaag of gisteren dateert, kan worden afgeleid uit de herinneringen van een Canadese codebreker "... One such course, run in February 1985 *as a result of direct operational requirements* was shift-register cryptology, which required 3 weeks of full-time study"^[35].



Karsten Nohl

* Het webadres genoemd in het artikel bestaat niet meer, waarschijnlijk vanwege dreiging met gerechtelijke stappen. Andere wapenfeiten van deze Karsten Nohl zijn het breken van de o.a. voor de OV-kaart gebruikte Mifare-chip, de onveiligheid van het in telefoonhandsets gebruikte DECT-protocol, de kwetsbaarheid van SIM-kaarten en recent de onveiligheid van de elektronische startonderbreker in autos.

10.9 Andere generatoren

Het genereren van een sleutelstroom is niet persé voorbehouden aan de hiervoor beschreven schuifregistercombinaties. Ook andere systemen kunnen zo opgezet worden, dat ze een voor vercijfering geschikte bytestroom afleveren. De opzet is hetzelfde als in [figuur 10.2](#) geschetst, met dien verstande dat geen bitstroom maar een bytestroom gemaakt wordt.

Een manier om een sleutelstroom uit een blokgeheimschrift als bijvoorbeeld DES te genereren is de volgende. Kies een sleutel en kies een klaartekst ter lengte van het vercijferd blok (initial vector) om mee te beginnen. Vercijfer het klaartekstblok en gebruik het vercijferd resultaat (of het eerste byte daarvan) in de sleutelstroom. Hetzelfde vercijferd resultaat wordt opnieuw met de sleutel vercijferd als ware het klaartekst, enzovoorts. De 'klaartekst' in deze pseudovercijfering is dus de voortschrijdende cijfertekst. Een variatie hierop is om niet de vercijferde uitstroom te hervercijferen, maar om deze te gebruiken als een voortdurend veranderende sleutel waardoor het in feite een soort autokey is ([sectie 4.3](#)).

10.9.1 RC4

De RC4-algoritme is een van de eerste uit de RCx-family van Ronald Rivest en al in 1987 ontwikkeld voor het bedrijf RSA Data Security. Voor het eerst is het gebruikt in het programma Lotus Notes. Aanvankelijk werd de algoritme geheim gehouden, maar in 1994 verscheen een door reverse-engineering verkregen versie. Deze bleek een identiek resultaat op te leveren in vergelijking met een officiële implementatie. Gezien de inmiddels gevonden statistische en andere zwakheden verscheen in 2014 een artikel met een verbeterde versie genaamd Spritz^[36]. In die publicatie erkent Rivest impliciet de juistheid van de door reverse-engineering verkregen versie en geeft zelf een beschrijving van RC4 in pseudocode.

Initialisatie met sleutel K (max 256 bytes), $N = 256$ (één byte), $+$ is mod N

- for $i = 0$ to $N-1$ do $S[i] = i$
- $j = 0$
- for $i = 0$ to $N-1$ do begin $j = j + S[i] + K[i \bmod \text{lengte_van_K}]$; verwissel $S[i]$ en $S[j]$ endfor
- $i = j = 0$

Productie van de sleutelstroom

- $i = i + 1$
- $j = j + S[i]$
- verwissel $S[i]$ en $S[j]$
- *resultaat* = $S[S[i] + S[j]]$

RC4 heeft op verschillende plaatsen toepassing gevonden. Echter niet altijd op de juiste manier. Een goed voorbeeld van een slechte implementatie werd in 2005 gevonden in de toenmalige Microsoft programma's Word en Excel^[37]. De documenten werden versleuteld met een sleutel (tot 128 bits) en de versleuteling gestart met een initialisatievector. Als een document gewijzigd is wordt het opnieuw versleuteld, uiteraard onder dezelfde versleutelsleutel. Maar aangezien ook de initialisatievector onveranderd blijft, kan het niet anders of RC4 genereert precies dezelfde sleutelstroom. En dus zijn er dan twee of meer, meest weinig van elkaar verschillende, documenten elk met eenzelfde sleutelstroom versleuteld. Het behoeft geen betoog dat een cryptoanalist met de resulterende Kerckhoffs-superpositie dan wel een stagger aan de slag kan gaan. Andere voorbeelden zijn aanvallen op RC4 in het Wi-Fi WEP en WPA-TKIP protocol^[38].

10.9.2 KASUMI

De in dit hoofdstuk behandelde schuifregisters hebben een belangrijk toepassingsgebied gevonden in de versleuteling van draadloze telefonie. De eenvoud en snelheid van de daarin verwerkte componenten

maakten de schuifregisters daarvoor bij uitstek geschikt. Inmiddels heeft de ontwikkeling van de computertechniek niet stilgestaan en heden ten dage zijn de processoren in mobiele telefoons krachtig genoeg om moderne blokgeheimschriften eveneens te kunnen toepassen. De jongste loot aan de stam van algoritmen in gebruik voor het gsm-verkeer heet *KASUMI*. Dit geheimschrift wordt ingezet als opvolger van de op schuifregisters gebaseerde A5/1 en A5/2, die inmiddels vergaand gebroken kunnen worden. KASUMI heeft voor toepassing in het telefoonverkeer de aanduiding A5/3 gekregen.

KASUMI is een geheimschrift met 64-bit blokken, een 128-bit sleutel en een uit 8 ronden bestaande Feistelstructuur. Voor een ronde zijn er twee afzonderlijke operaties, waarvan de volgorde per ronde afwisselt. Een van die twee zorgt voor de niet-lineariteit en bestaat daartoe zelf ook weer uit drie Feistel-stappen met *S*-dozen voor respectievelijk een 7- en een 9-bits substitutie. Dit geheimschrift heeft het zogenaamde MISTY-geheimschrift als voorloper. Het is daaruit ontstaan door enkele vereenvoudigingen die het sneller en gemakkelijker te implementeren hebben gemaakt. De artikelen in de hieronder geciteerde literatuur bevatten een uitvoerige beschrijving van de KASUMI algoritme.

Cryptoanalyse in de open literatuur is anno 2013 amper beter dan uitputtend zoeken en vergt op z'n best ongeveer 2^{125} encrypties plus 2^{32} aan chosen-plaintext.^[39] Een tot zes i.p.v. acht ronden gereduceerde KASUMI met aanval op een enkele sleutel brengt het tot 2^{61} benodigde data en ruim 2^{65} encrypties.^[40] Er bestaat wel een meer efficiënte aanval met 2^{32} encrypties, maar die verlangt de beschikbaarheid van 2^{26} aan chosen-plaintext/ciphertext met vier aan elkaar gerelateerde sleutels. Opvallend genoeg werkt die methode niet op het oorspronkelijke MISTY en moet dus het gevolg zijn van de daarin aangebrachte vereenvoudigingen^[41].

Alhoewel uit het voorgaande kan worden geconcludeerd dat KASUMI behoorlijk veilig is, wil dit nog niet zeggen dat het gebruik ervan in mobiele telefoons een garantie biedt tegen afluisteren. Indien een telefoongesprek wordt opgezet met een zogenaamde end-to-end encryptie, dan wordt een voor het gesprek unieke sleutel uitgewisseld tussen de twee telefoons. Nu kan het voorkomen dat een van de twee tele-

foons van een ouder type is en gebruik maakt van A5/1 of A5/2. Dan behoeft slechts de zwakste schakel gebroken te worden om het gehele gesprek te kunnen volgen.

Voor een veilig gesprek moeten dus minstens beide telefoons A5/3 gebruiken en moet bovendien een end-to-end sleutel veilig worden uitgewisseld. Gebeurt dat laatste niet en kan de af luisterende organisatie het protocol waarin die sleutel wordt uitgewisseld breken of deze geforceerd afdwingen van de telecomaanbieder, dan helpt geen enkel algoritme. Werken met aparte sleutels voor elk van beide telefoons is natuurlijk evenzeer dodelijk voor de veiligheid, omdat in dat geval in een basisstation een ontcijfering gevolgd door een hervercijfering nodig is. De tegenstander behoeft zich dan nog slechts toegang tot dat basisstation te verschaffen. Kortom de aard van mobiele telefonie biedt tegenover professionele tegenstanders en overheden weinig bescherming. Alleen een private end-to-end verbinding met een krachtig encryptiealgoritme kan zoden aan de dijk zetten.

10.10 Referenties

- [1] D. Kahn (1967). *The Codebreakers, The Story of Secret Writing*. MacMillan Publ. Co., Inc. p. 394–397.
- [2] S. W. Golomb (1967). *Shift Register Sequences*. Holden-Day, Inc., reprinted by Aegean Park Press, 1982.
- [3] D. E. Knuth (1998b). *The Art of Computer Programming, Volume 2, Seminumerical Algorithms*. third edition Addison-Wesley.
- [4] D. E. Knuth, R. L. Graham and O. Patashnik (1989). *Concrete Mathematics*. Addison-Wesley. p. 163.
- [5] N.G. de Bruijn (1946). A Combinatorial Problem. *Nederl. Akad. Wetensch. Proc.*, 49, 756–758.
- [6] S. W. Golomb (1967). *Shift Register Sequences*. Holden-Day, Inc., reprinted by Aegean Park Press, 1982. Table III-5, p. 62–65.
- [7] B. Schneier (1996). *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. second edition John Wiley & Sons, Inc. p. 376–377.
- [8] R. A. Rueppel (1986). *Analysis and Design of Stream Ciphers*. Springer-Verlag. Hoofdstuk 4.

- [9] P. C. van Oorschot, A. J. Menezes and S. A. Vanstone (1997). *Handbook of Applied Cryptography*. CRC Press, Inc.. Hoofdstuk 6. De in dat boek gehanteerde alternatieve nummering van **figuur 10.8** hoeft geen probleem op te leveren, omdat de gevonden $f(x)$ in de andere vorm (de reciprook polynoom) kan worden gebracht kan met behulp van **vergelijking (10.26)**.
- [10] S. M. Jennings (1980). *A special class of binary sequences*. PhD thesis, University of London.
S. M. Jennings (1983). Multiplexed sequences: some properties of the minimum polynomial. In *Cryptography Proc. Burg Feuerstein 1982 Lecture Notes in Computer Science* .
- [11] R. J. Anderson (1990). Solving a class of stream ciphers. *Cryptologia*, XIV(3), 285–288.
- [12] R. J. Anderson (1995). Searching for the Optimum Correlation Attack. In B. Preneel, editor (1995). *Advances in Cryptology*, number 1008 in Lecture Notes in Computer Science ., pages 137–143.
- [13] F.D. Garcia, P. van Rossum, R. Verdult and R.W. Schreur (2009). Wirelessly Pickpocketing a Mifare Classic Card. *IEEE Symposium on Security and Privacy*, pp. 3–15.
- [14] V. S. Pless (1977). Encryption schemes for computer confidentiality. *IEEE Transactions on Computers*, C-26, 1133–1136.
- [15] P. R. Geffe (1973). How to protect data with ciphers that are really hard to break. *Electronics*, pp. 99–101.
- [16] J. O. Bruer (1982). On nonlinear combinations of linear shift register sequences. In *Proc. IEEE Int. Symp. Inform. Theory, les Arcs, France* .
- [17] T. Siegenthaler (1985). Decrypting a Class of Stream Ciphers Using Ciphertext Only. *IEEE Transactions on Computers*, C-34, 81–85.
- [18] W. Meier and O. Staffelbach (1989). Fast Correlation Attacks on Certain Stream Ciphers. *Journal of Cryptology*, 1, 159–176.
- [19] A. Clark, J. D. Golić and E. Dawson (1996). A Comparison of Fast Correlation Attacks. In D. Gollmann, editor (1996). *Advances in Cryptology*, number 1039 in Lecture Notes in Computer Science ., pages 145–157.
- [20] T. Siegenthaler (1984). Correlation-Immunity of Nonlinear Combining Functions for Cryptographic Applications. *IEEE Transactions on Information Theory*, IT-30, 776–780.
- [21] R. A. Rueppel (1986). *Analysis and Design of Stream Ciphers*. Springer-Verlag. p. 209 e.v.
- [22] W. Meier and O. Staffelbach (1992). Correlation Properties of Combiners with Memory in Stream Ciphers. *Journal of Cryptology*, 5, 67–86.

- [23] E. Dawson (1992). Cryptanalysis of Summation Generator. InJ. Seberry and Yuliang Zheng, editor (1992). *Advances in Cryptology*, number 718 in Lecture Notes in Computer Science ., pages 209–215.
E. Dawson and A. Clark (1994). Divide and Conquer Attacks on Certain Classes of Stream Ciphers. *Cryptologia*, XVIII(1), 25–40.
- [24] Erik Tews (2012). DECT Security Analysis. Technical Report TU Darmstadt. URL = <http://tuprints.ulb.tu-darmstadt.de/2932>.
- [25] T. Beth and F. C. Piper (1985). The Stop-and-Go Generator. InN. Cot, T. Beth and I. Ingemarsson, editors (1985). *Advances in Cryptology*, number 209 in Lecture Notes in Computer Science ., pages 88–92.
- [26] D. Y. Wei, K. Zeng, C. H. Yang and T. R. N. Rao (1991). Pseudorandom Bit Generators in Stream-Cipher Cryptography. *IEEE Computer*, pp. 8–17.
- [27] C. G. Günther (1988). Alternating Step Generators Controlled by de Bruijn Sequences. InD. Chaum and W. L. Price, editor (1988). *Advances in Cryptology*, number 304 in Lecture Notes in Computer Science ., pages 5–14.
- [28] D. Coppersmith, H. Krawczyk and Y. Mansour (1994). The Shrinking Generator. InD. R. Stinson, editor (1994). *Advances in Cryptology*, number 773 in Lecture Notes in Computer Science ., pages 22–39.
- [29] W. Meier and O. Staffelbach (1994). The Self-Shrinking Generator. InA. de Santis, editor (1994). *Advances in Cryptology*, number 950 in Lecture Notes in Computer Science ., pages 205–214.
- [30] J. D. Golić (1997). Cryptanalysis of Alleged A5 Stream Cipher. InW. Fumy, editor (1997). *Advances in Cryptology*, number 1233 in Lecture Notes in Computer Science ., pages 239–255.
A. Biryukov and A. Shamir (1999). Real Time Cryptanalysis of the Alleged A5/1 on a PC (preliminary draft). . Te vinden via <http://cryptome.org/a5.ps>
- [31] E. Barkan, E. Biham and N. Keller (2003). Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication. InD. Boneh, editor (2003). *Advances in Cryptology*, number 2729 in Lecture Notes in Computer Science ., pages 600–616.
- [32] P. Ekdahl and T. Johansson (1976). Another Attack on A5/1. *IEEE Transactions on Information Theory*, IT-49, 284–289.
- [33] O. Dunkelman, N. Keller and A. Shamir (2010). A Practical-Time Related-Key Attack on the KASUMI Cryptosystem Used in GSM and 3G Telephony. InT. Rabin, editor (2010). *Advances in Cryptology*, number 6223 in Lecture Notes in Computer Science ., pp. 393–410.
- [34] J. Shah and A. Mahalanobis (2012). A New Guess-and-Determine Attack on the A5/1. <http://eprint.iacr.org/, 2012/208>.

- [35] B. Robinson (1992). The Fall and Rise of Cryptanalysis in Canada. *Cryptologia*, XVI(1), 23–38.
- [36] Ronald L. Rivest and Jacob C.N. Schuldt (2014). Spritz - a spongy RC4-like stream cipher and hash function. , people.csall.mit.edu/rivest/pubs/RS14.pdf.
- [37] Hongjun Wu (2005). The Misuse of RC4 in Microsoft Word and Excel. <http://eprint.iacr.org/>, 2005/007.
- [38] Scott Fluhrer, Itsik Mantin and Adi Shamir (2001). Weaknesses in the Key Scheduling Algorithm of RC4. .
Pouyan Sepehrdad, Pet Susil, Serge Vaudenay and Martin Vuagnoux (2015). Tornado Attack on RC4 with Applications to WEP and WPA. <http://eprint.iacr.org/>, 2015/254.
Mathy Vanhoef and Frank Piessens (2015). All Your Biases Belong To Us: Breaking RC4 in WPA-TKIP and TLS. .
- [39] Keting Jia, Honbo Yu and Xiaoyun Wang (2011). A Meet-in-the-Middle Attack on the Full Kasumi. <http://eprint.iacr.org/>, 2011/466.
- [40] Teruo Saito (2011). A Single-Key Attack on 6-Round Kasumi. <http://eprint.iacr.org/>, 2011/584.
- [41] O. Dunkelman, N. Keller and A. Shamir (2010). A Practical-Time Related-Key Attack on the KASUMI Cryptosystem Used in GSM and 3G Telephony. In T. Rabin, editor (2010). *Advances in Cryptology*, number 6223 in Lecture Notes in Computer Science ., pp. 393–410.

A Verklarende woordenlijst

algoritme De algoritme is een recept voor een bepaalde berekening. Bijvoorbeeld de algoritme van Euclides is een recept waarmee de grootste gemene deler van twee getallen kan worden berekend. Een computerprogramma voor die berekening is een implementatie van de algoritme. Het is van belang te bedenken dat de algoritme een theoretisch voorschrift is dat geen last heeft van de beperkingen die de constructie van een computer wel oplegt aan de implementatie ervan.

and-operator De and-operator is een vermenigvuldiging(!) van bits. De mogelijke waarden zijn $(0 \text{ and } 0) = (0 \text{ and } 1) = (1 \text{ and } 0) = 0$ en $(1 \text{ and } 1) = 1$.

ANSI Afkorting van American National Standards Institute. Deze organisatie legt o.a. standaarden voor computers vast, zoals de bekende ASCII code, de American Standard Code for Information Interchange.

aperiodiek cryptosysteem Geheimschrift waarin de sleutel een zich niet herhalende reeks substituties is.

asymmetrisch cryptosysteem Geheimschrift waarin afzonderlijke sleutels voor respectievelijk het verscijferen en het ontcijferen dienen.

authenticiseren Het waarmerken van een document door het plaatsen van een (digitale) handtekening.

autoclaaf Een aperiodiek geheimschrift waarin de sleutel van opeenvolgende substituties wordt afgeleid van een vorige verscijfering.

bigram Ander woord voor digram, een combinatie van twee letters.

bijectie Wiskundige term voor een zogenaamde één-op-één afbeelding. D.w.z. een unieke correspondentie tussen twee verzamelingen: elk element van de ene verzameling heeft één en slechts één pendant in de andere. Dit staat in tegenstelling tot bijvoorbeeld een woordenboek Nederlands-Engels: er zijn veel woorden die meerdere vertalingen hebben.

blokgeheimschrift Een geheimschrift waarbij meerdere symbolen tegelijk moeten worden verscijferd. In computers wordt de lengte van het blok uitgedrukt in bits of in bytes.

cardinaliteit De cardinaliteit van een verzameling (verschillende) symbolen is het aantal symbolen daarin aanwezig; de cardinaliteit van het gewone alfabet van A-Z is 26.

carryloze optelling Als bij optelling een overloop (carry) ontstaat, wordt deze verwaarloosd; in de normale optelling $7 + 8 = 15$ is de carry 10 en het resultaat van de carryloze optelling is dus 5. Analoog kunnen twee getallen zonder carry van elkaar worden afgetrokken. Zie ook de xor.

cipher-block-chaining Procedure waarbij de klaartekstblokken niet afzonderlijk worden vercijferd, maar aan elkaar worden gekoppeld om de nadelen van electronic-code-book te vermijden. Ook aangeduid met de term: CBC-modus.

checksum Een aantal controlebits berekend uit een gegeven bitrij. Door de checksum opnieuw te berekenen en te vergelijken met de opgegeven waarde, kan worden geconstateerd of in de bitrij geen fouten zijn geslopen. Goede checksumalgoritmen kunnen met betrekkelijk weinig controlebits volstaan. Een voorbeeld is de in de communicatietechniek veel gebruikte Cyclic Redundancy Check (CRC).

chosen-plaintext aanval Een aanval op een geheimschrift waarbij de cryptoanalist de klaartekst geheel of gedeeltelijk kan beïnvloeden.

ciphertext-only aanval Een aanval op een geheimschrift waarbij de cryptoanalist slechts beschikt over een of meer cryptogrammen.

coïncidentie Een positie van onder elkaar geplaatste symboolreeksen waar deze hetzelfde symbool vertonen.

commuteren Operaties commuteren wanneer ze in willekeurige volgorde kunnen worden uitgevoerd zonder daardoor het resultaat te beïnvloeden.

complexiteit Het uitvoeren van een algoritme kost een zekere hoeveelheid rekentijd. Soms blijft die constant, ongeacht de grootte van het op te lossen probleem. Maar vaak groeit de benodigde rekentijd mee met de omvang van het probleem. Als een twee keer zo groot probleem viermaal zoveel rekentijd vergt groeit de rekentijd kwadratisch met de grootte van het probleem: grootte tot de macht 2. Zolang die macht maar een constant getal is spreken informatici van *polynomiale complexiteit*, d.w.z. de benodigde rekentijd als functie van de grootte van het probleem wordt gegeven door een polynoom. Informatici vinden zo'n probleem *doenlijk*, al kan dat in de praktijk nogal tegenvallen als die constante groot is. Ondoenlijk noemt men problemen waarvan de rekentijd *exponentieel* groeit, ongeveer zoals het aantal mogelijke zetten in het schaakspel wanneer men steeds verder vooruit wil rekenen.

crib Een stukje klaartekst waarvan vermoed wordt dat het in een cryptogram voorkomt. Als dit inderdaad het geval is, levert dit een mogelijke entree voor de ontcijfering.

decimatie Het verkrijgen van een permutatie van een rij symbolen door steeds het symbool een vast aantal plaatsen verder uit te nemen, dan wel door de symbolen achtereenvolgens een vast aantal posities uiteen te plaatsen; (1,3,5,2,4) is een decimatie van (1,2,3,4,5) waaruit telkens

het tweede cijfer is genomen.

deterministisch Een proces heet deterministisch wanneer elke uitgangspositie steeds tot hetzelfde resultaat leidt. Dit staat in tegenstelling tot een nondeterministisch proces, waarvan de uitkomst niet te voorspellen valt. Een computerprogramma is noodzakelijkerwijs een deterministisch proces.

digram Een combinatie van twee letters, ook wel aangeduid als bigram.

dinome Een combinatie van twee cijfers.

electronic-code-book Procedure waarbij de klaartekstblokken afzonderlijk worden vercijferd. Leidt in wezen tot een monoalfabetische substitutie met evenveel symbolen als er verschillende klaartekstblokken mogelijk zijn. Ook aangeduid met de term: ECB-modus.

encode Een cryptogram in code dat een oververcijfering heeft ondergaan.

end-to-end encryptie Berichten worden in het systeem van de verzender vercijferd en in het systeem van de ontvanger ontcijferd. In het tussenliggend traject in het transportnetwerk vindt geen ontcijfering plaats. Hiertoe moeten verzender en ontvanger beide over dezelfde sleutel beschikken of een systeem met publieke sleutel gebruiken.

endomorf cryptosysteem Een geheimschrift waarbij klaartekst en cryptogram zijn samengesteld uit dezelfde verzameling symbolen.

entropie Grootheid die een hoeveelheid informatie aanduidt; er bestaat een inhoudelijke overeenkomst met de gelijknamige grootheid uit de thermodynamica (warmteleer).

equivocatie De hoeveelheid informatie (entropie) gepaard gaande met samenhangende gebeurtenissen; het kanaalverlies is de equivocatie in een imperfect informatiekanaal.

exponentieel algoritme Zie complexiteit.

factoriseren Het ontbinden van een getal in een product van priemfactoren; de ontbinding van het getal 20 is $2 \times 2 \times 5 = 2^2 \times 5$.

faculteit Met $N!$ (spreek uit N faculteit) wordt bedoeld het product $N \times (N-1) \times (N-2) \times \dots \times 2 \times 1$. De waarde hiervan neemt snel toe met de waarde van N : $3! = 6$, $4! = 24$... $10!$ is al 3.628.800 en $26! = 4,0310^{26}$, het aantal mogelijke verwisselingen in een 26 letter alfabet. Let op, $0!$ krijgt de waarde 1.

firewall Een beschermingsconstructie die tussen het internet en een lokaal netwerk wordt geplaatst. De firewall filtert de data die van het internet komen en houdt ongewenste datapakketten tegen. Een firewall kan een apart apa-

raat met bijbehorende software zijn, maar ook software in het besturingssysteem van een computer.

fpga Een FPGA (field-programmable gate array) is een chip die door de gebruiker kan worden geprogrammeerd tot een special purpose chip.

gelijksoortige cryptosystemen Geheimschriften die ogenschijnlijk verschillend zijn, maar in wezen hetzelfde doen en daarom zonder meer in elkaar kunnen worden overgevoerd.

homofone substitutie Voor een symbool van de klaartekst kan naar believen worden gekozen uit een aantal substituenten.

IACR De International Association for Cryptologic Research (IACR) is een nonprofit organisatie ter bevordering van onderzoek in de cryptologie en aanverwante onderwerpen. Organiseert drie jaarlijkse internationale conferenties (Asiacrypt, Eurocrypt en Crypto), workshops en geeft het Journal of Cryptology uit. Internet <http://www.iacr.org/>.

idempotent cryptosysteem Een cryptosysteem waarbij herhaald uitvoeren van een vercijfering geen versterkend effect heeft, omdat in het systeem altijd een vercijfering kan worden gevonden die hetzelfde resultaat geeft als de gecombineerde vercijferingen.

index of coincidence Door William Friedman ingevoerde benaming voor het (statistisch) tweede moment van een frequentiedistributie.

informatiekanaal Een medium waarlangs informatie kan worden overgebracht.

involutie Een operatie die z'n eigen inverse is; d.w.z. als uitvoeren van de operatie op element A resulteert in element B, dan geeft uitvoeren op B het oorspronkelijke element A terug.

isolog Een cryptogram met dezelfde klaartekst als een overeenkomstig cryptogram, maar vercijferend in een ander geheimschrift.

isomorf Een isomorf is de generalisatie van de herhaling; twee isomorfe symboolreeksen kunnen d.m.v. een monoalfabetische substitutie in elkaar worden overgevoerd.

kanaalverlies Het verlies aan informatie dat optreedt bij informatieoverdracht over een imperfect informatiekanaal.

Kerckhoffs-superpositie Men spreekt van een Kerckhoffs-superpositie wanneer de cryptoanalist kan beschikken over verschillende documenten die alle met dezelfde sleutelstroom zijn vercijferd. Op elke positie kunnen de overeenkomstige letters bijeen genomen worden en vertonen een monoalfabetische distributie.

key appearance De hoeveelheid informatie die uit de beschikbare gegevens omtrent de sleutel wordt geopenbaard.

known-plaintext aanval Een aanval op een geheimschrift waarbij de cryptanalist beschikt over een of meer cryptogrammen en de daarmee corresponderende klaartekst.

link encryptie Berichten verzonden over een netwerk worden in een knooppunt ontcijferd en vervolgens hervercijferd naar het volgende knooppunt gestuurd. De beide betrokken knooppunten moeten hiertoe over dezelfde sleutel beschikken.

Markov-keten Een serie opeenvolgende gebeurtenissen waarin elke volgende gebeurtenis afhankelijk is van diens voorgeschiedenis.

modulo rekenen Een rekenkundige bewerking waarbij elk resultaat wordt gedeeld door een vast getal (modulus geheten), waarna slechts de rest van de deling wordt behouden. Stel het quotiënt van a gedeeld door b is q met rest r . Dan is $a = qb + r$ en $a \bmod b = r$. Men kort "a modulo b" ook wel af tot " $a \bmod b$ ".

monome Een enkel cijfer.

NIST Afkorting van National Institute of Standards and Technology, voorheen het National Bureau of Standards. Vanuit deze organisatie stamt het initiatief voor de DES en de AES. Internet site <http://www.nist.gov/>, site van de

Computer Security Division met links naar cryptografische technologie <http://csrc.nist.gov/>.

nonrepudiation Het niet kunnen ontkennen van een handeling, zoals bijvoorbeeld het verzenden van een bericht.

NSA Afkorting van National Security Agency. Dit is de belangrijkste Amerikaanse organisatie voor het verzamelen van inlichtingen d.m.v. spionagesatellieten, af luisteren van communicatieverbindingen, cryptanalyse, enz. Bij het ontstaan zo geheim dat ze wel schertsend No Such Agency werd genoemd. Internet site <http://www.nsa.gov/>.

or-operator De or-operator is de tegengestelde van de and-operator. De mogelijke waarden zijn $(0 \text{ or } 0) = 0$ en $(0 \text{ or } 1) = (1 \text{ or } 0) = (1 \text{ or } 1) = 1$.

oververcijfering Hervercijferen van een cryptogram in een ander geheimschrift.

padding Het uitvullen van het blok in een blokgeheimschrift, wanneer de klaartekst niet het gehele blok vult.

pariteitsbit Bit dat een waarde krijgt afhankelijk van het aantal 1-bits in een 8-bits byte; maakt bijvoorbeeld het aantal 1-bits even. Bedoeld als controle bij onvolkomen signaaloverdracht.

periodiek cryptosysteem Een geheimschrift waarin de sleutel een zich herhalende reeks substituties is; de lengte

van de herhaling heet de periode van het geheimschrift.

permutatie De wiskundige term voor een verwisseling. Een verzameling van N elementen kan op $N!$ (zie faculteit) manieren worden gepermuteerd. Bijvoorbeeld is (3,2,1) een permutatie van (1,2,3); deze drie getallen kunnen op $3! = 6$ manieren worden gerangschikt.

Polybius rechthoek Een rechthoek (van origine een vierkant) met rij- en kolomcoördinaten en in de vakjes symbolen die worden vervangen door een combinatie van deze coördinaten. Veelgebruikte Engelse term hiervoor: checkerboard.

polynomiaal algoritme Zie complexiteit.

priemgetal Een getal dat niet in factoren kan worden ontbonden, d.w.z. het is geen product van kleinere getallen. Formeel: een getal dat slechts deelbaar is door 1 en door zichzelf. Dat er oneindig veel priemgetallen zijn blijkt uit de volgende redenering. Veronderstel er zijn eindig veel priemgetallen, dan bestaat een grootste priemgetal. Neem nu het product van alle priemgetallen van 2 tot en met het grootste en tel daarbij 1 op. Dit getal is door geen van de priemgetallen deelbaar, want altijd zal een rest van 1 overblijven. Dus is het of zelf een priemgetal groter dan het grootste priemgetal, of het heeft een priemfactor groter dan het grootste priemgetal. Dit maakt duidelijk dat er geen grootste priemgetal kan zijn: na elk priemgetal komt altijd een nog groter priemgetal.

productvercijfering Een vercijfering die ontstaat door achtereenvolgens verschillende geheimschriften toe te passen.

quantummechanica Het deel van de natuurkunde dat gaat over de beschrijving en eigenschappen van elementaire deeltjes, zoals fotonen, elektronen, enz.

randomgenerator Zie toevalsgenerator.

regenboogtabel Een methode om door middel van voorberekeningen het vinden van vercijfersleutels te vergemakkelijken. Hoe meer voorberekeningen des te meer kans dat uit een knownplaintext de gebruikte sleutel kan worden achterhaald.

sessiesleutel Een vercijfersleutel bedoeld voor eenmalig gebruik in een communicatiesessie.

side-channel-analyse Aanvalsmethode die zich niet op de vercijferalgoritme zelf richt, maar gebruik maakt van de fysieke implementatie ervan.

stagger Men spreekt van een stagger als twee documenten nagenoeg gelijk zijn behoudens een of enkele invoegingen/weglatingen en bovendien met dezelfde sleutelstroom zijn vercijferd. Het gevaar hiervan is dat de sleutelstroom kan worden gereconstrueerd vanaf het punt waar de vercijferde documenten van elkaar zijn gaan verschillen.

substitutie Het (systematisch) vervangen van elementen van de klaartekst door andere.

symmetrisch cryptosysteem Geheimschrift waarin eenzelfde sleutel zowel voor het vercijferen als voor het ontcijferen dient.

time-memory-tradeoff Principe waarbij in de cryptoanalyse door inzet van meer (computer)geheugen een reductie van het aantal rekenstappen kan worden bewerkstelligd.

toevalsgenerator Een mechanisme dat een willekeurige reeks getallen maakt. De kwaliteit van de toevalsreeks wordt bepaald met allerlei statistische tests. Het maken van een toevalsreeks van hoge kwaliteit is niet eenvoudig. Wordt ook wel randomgenerator genoemd.

traffic analysis Het uit de berichtkoppen afleiden van informatie over de verzender en afzender, de aard van het bericht e.d. (Nederlands: berichtenverkeersanalyse)

transpositie Het van plaats laten verwisselen van elementen van de klaartekst.

trigram Een combinatie van drie letters.

trinome Een combinatie van drie cijfers.

uniciteitsafstand De minimale lengte van een crypto-

gram waarvan mag worden aangenomen dat het eenduidig kan worden ontcijferd zonder te beschikken over de sleutel. (Engels: unicity distance)

veilig kanaal Een verbinding die niet kan worden afgevuist.

VPN De afkorting VPN staat voor Virtual Private Network. Dit is een manier om middels end-to-end encryptie over het internet een verbinding te creëren die niet kan worden afgevuist.

wederzijdse informatie De hoeveelheid informatie over een informatiebron die kan worden afgeleid wanneer deze bron over een (imperfect) informatiekanaal wordt waargenomen.

whitening Technische term waarmee de vercijfering wordt aangeduid die wordt toegepast voorafgaand en/of volgend op de eigenlijke vercijferalgoritme.

xor-operator De xor-operator (exclusive or) is een carryloze optelling van bits. Aanduiding in formules met $\oplus$, maar ook wel met een eenvoudige $+$. De mogelijke waarden zijn $(0 \text{ xor } 0) = (1 \text{ xor } 1) = 0$ en $(0 \text{ xor } 1) = (1 \text{ xor } 0) = 1$.

zwarte kamer Organisatie die het breken van geheimschriften tot doel heeft en waarvan het bestaan geheim moet blijven.



Index

a

AKS algoritme 415

A5/3 533

Abwehr 68, 173

ACA 55, 211

achterdeur 323, 407

additief 110

AddRoundKey 390, 397, 400

ADFGVX

52, 97, 100, 306

ADFGX

100

Adleman, Leonard Max 420, 423, 424

Adonis 163

AES 4, 43, 305, 364, 379, 380, 381, 385, 389, 419, 431

affine transformatie 81

afluisteren 9

alfabet 249, 257

algebraïsche groep 37, 321

algebraïsche normaalvorm 471

algoritme 157

anagrammeren 67

AND 501

Anderson, Ross 387, 494

Angooki Taipu B 163, 178

ANSI 312

anticorrelatie 510

anti-fixed point 49

aperiodiek 140

ARIA 386

ASCII 510

ASIC 327, 383

AT&T 459

authentiseren 14

autocorrelatie 462

automaat 461

A5

A5/0 530

A5/1 530, 531, 532, 533, 536, 537

A5/2 530, 532, 536, 537

A5/3 530, 536, 537

b

Binomium van Newton 299
Black Chamber 102
Bletchley Park 55
Baby 43
Baudot 113, 459
Bauer, Friedrich 60
Bayes, Rev. Thomas 249, 252, 253
Bazeries
 cylinder 146, 147
 Etienne 146
Beaufort 36, 119, 124, 125, 130, 165, 166, 168, 169,
 171, 219, 231, 237, 241
 Sir Francis 124
Beek, H.A. van der 52
berichtenverkeersanalyse 10, 11
berichtkop 10
Berlekamp-Massey 489, 490, 491, 502, 528
Beth, T. 526
bifid 97, 98
Biham, Eli 331, 346, 347, 348, 352, 354, 362, 371, 387
binair symmetrisch kanaal 294
Biryukov 531
bit 258
Bletchley Park 173

geblindeerd 434
blok 43, 461
blokgeheimschrift 4, 43, 49, 313
Blowfish 378
Blue Code 68
bodemsignalen 113
boekcode 112
Boer, Bert den 378
Bommen-Berend 85
briefgeheim 102
Bruer generator 501, 507, 514
brute-force 26
Bruin, N.G. de 472
bijectie 49
ByteSub 390, 391, 392, 393, 398, 399, 400

c

Capstone chip 367
Chinese reststelling 431
COPACOBANA 328
CSI-10 cryptogenerator 156
C-38 169
Caesar
 Julius 79
 substitutie 79, 80, 81, 123, 152, 169, 251, 277, 279
 verschuiving 123, 125, 152, 229

- Cambridge Five 111
- Cardano, Girolamo 73
- cardinaliteit 257
- carry 110
- Castro, Fidel 90
- CBC-modus 44, 46, 47, 382
- checkerboard 86
- checksum 21, 47
- chi-test 204
- chosen-plaintext 28, 332, 354
- cifra
 - general 103, 104
 - particular 105
- ciphertext 5
- ciphertext-only 27, 281, 282
- Clipper Chip 366, 367, 368
- Clipper chip 369
- code 106
- coderen 256
- coïncidentie 201
- coïncidentietelling 202
- communicatie 269
- communications security 7
- commuteren 41, 306
- complementariteitseigenschap 317
- computernetwerk 57
- COMSEC 7
- consonantlijn 185, 209
- constante van Planck 15
- Coppersmith, Don 350, 351, 386, 431
- correlatieaanval 509
- correlatie-immuniteit 520
- crib 32
- Croiset, Samuel Egbert 102
- cryptanalysis 7
- Crypto A.G. 163, 164
- cryptoanalyse 3, 7, 8, 9, 9, 11, 22, 25, 27, 28, 33, 39, 118
- cryptografie 5
- cryptografische checksum 407
- cryptogram 5
- cryptologie 7
- cryptology 7
- cryptosysteem 6, 21
 - asymmetrisch 6, 13
 - autoclaaf 140
 - autokey 140
 - endomorf 36, 39
 - gemengd 39
 - idempotent 37, 306, 321
 - klassiek 6, 7, 8

- multiplex 144
- public-key 12, 13, 406
- publieke sleutel 52, 54
- symmetrisch 6
- zuiver 39, 41
- CSP-845 148
- CTR-modus 47
- cursus cryptoanalyse 50, 51, 237
- CWI 426
- cyclus 49, 461
- cijferalfabet 128, 144
- cijfercode 36
- cijfertekst 5
- cijfertekstautoclaaf 140
- cyrillisch 189
- d**
- Digital Signature Standard 414
- Dual_EC_DRBG 450, 451
- Daemen, Joan 374, 389
- Damm, Arvid 149, 163
- DDR 530
- DEA 312
- DeBruijn-reeks 472
- decimatie 134, 237, 240
- decoderen 256
- decryptie 5
- DECT 525
- deeltjeskarakter 15
- Deep Crack 326, 379
- Delastelle, F.M. 98
- DES 4, 29, 43, 255, 305, 312, 313, 314, 315, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 331, 332, 346, 347, 348, 349, 350, 351, 354, 355, 358, 362, 363, 364, 365, 369, 372, 377, 378, 379, 383, 386, 388, 391, 534
- deterministisch 158
- deterministische algoritme 443
- differential fault analysis 362
- differentiële cryptoanalyse 325, 332
- Diffie, Whitfield 12, 325, 417
- Diffie-Hellman 418, 433, 450
- diffusie 33
- digitale handtekening 14, 441
- digram 31, 91, 186, 190, 274
- digramsubstitutie 91, 286
- dinome 88, 89, 130
- diplomatieke code 57
- discrete logaritme 418, 433
- dockyard cipher 27
- Doppelkasten 96

DSS 414, 436

e

Enigma 54

Escrowed Encryption Standard 367

Euclides 416, 422, 424, 428, 429, 430, 446

Euler 416, 423

Euler totiënt functie 410, 422

Euler-getuige 416

Euler-leugenaar 417

ECB-modus 44, 382

ECC 436, 437, 441, 442, 450

eendelige code 108

electronische ruis 158

ElGamal 433, 434

elliptische kromme 436

Generalized Number Field Sieve 421

Juniper Networks 451

message 6

encicode 110

enciphered code 110

encryptie 5

end-to-end encryptie 11, 536

energie 15

Englandspiel 54, 70

Enigma 3, 33, 36, 51, 52, 53, 150, 163, 173

entropie 159, 259, 458

ePrint 50

equivocatie 264, 266, 295

Eratosthenes 412

escrow 365

Euclides 374, 393

exponent 482

f

FBI 365, 368

FPGA 328

factoriseren 420

FEAL 378

Feistel 374

gebalanceerd 309, 313

geheimsschrift 37, 305, 307, 309, 310, 313, 331, 332, 369, 377, 390,
536

Horst 307, 322

netwerk 334, 337, 338, 340, 363, 370, 386, 387, 388

ongebalanceerd 310

Fialka 163

filterfunctie 497

FIPS 378

firewall 451

fixed point 49

Fleissner

grille 73

von Wostrowitz, E.F. 73, 75

foton 14

fourterstellende code 268

FOX 386

FPGA 383

fractionerend systeem 97, 102

frequentiedistributie 30, 33, 118, 186, 187, 189, 191,
194, 195, 196, 197, 259Friedman, William F. 7, 52, 54, 60, 63, 139, 149, 163,
165, 179, 187, 190, 219, 237

fsr 463

g

Gaelic 217

Georgisch 217

Galen, Christoph Bernhard van 85

Galois, Évariste 409

gat 462

Gauss, Carl Friedrich 413

gebalanceerde stroom 521

Geffe generator 501, 506, 507, 508, 509, 512, 514, 520,
522

geheime exponent 421

geheime sleutel 13

geheimschrijver 5

gelijksoortig systeem 42

gemiddelde 187

generator

Bruer 501, 507, 514

Geffe 501, 506, 507, 508, 509, 512, 514, 520, 522

MacLaren-Marsaglia 155

multiplex 492

Pless 501, 503, 505, 514, 523

som 523

generatrix 145

genererende functie 477

gesloten systeem 35

getaltheorie 4, 408

golfkarakter 15

Golomb, Solomon W. 461

Gordon, John 414

GOST 377

groepentheorie 37

gsm 11

Guevara, Ché 90

h

Hagelin 54

Hebreeuws 217

Hewlett-Packard HP41C 156

Hongaars 217
Hagelin 3, 36, 51, 163, 165, 167
 Boris C.W. 163, 164, 165
 B-211 149, 164, 165
 C-35 165
 C-38 165
 C-52 165
 C-57 165
 C35 167, 171
 C38 167
 C57 167
 M-209 165, 166
halfrotor 149
halfzwakke sleutel 319
hash-algoritme 381
Hayhanen, Reino 69
Hebern 51, 150
 Edward 163
Hellman, Martin 12, 325, 328, 329, 417
Hengel, Theo A. van 173
herhaling 227
hertog van Alva 105
Hindenburg, C.F. 75
Hitt, Parker 146
homofone substitutie 84

homofonen 102
Huygens, Christiaan 432

i
IACR 50, 291
IDEA 372
International Data Encryption Algorithm 372
IBM 307, 312, 323, 324, 350, 379, 386
IDEA 372, 374
idempotent 41
imperfecte communicatie 268
impossible differential 371
impuls 15
index of coincidence 187
indicatorgroep 111
Informatie 255
informatie
 bron 256
 capaciteit 261
 kanaal 256, 291
 theorie 4, 8, 26, 51, 53
 waarde 258
informatietheorie 246, 247
ingangspermutatie 315
initial vector 534
initiële vector 46

- internet 10
- InvMixColumn 400
- invoerpermutatie 313
- involutie 36, 174
- irreducibel 482
- isolog 27
- isomorf 99, 227
- isomorfie 185, 237
- j**
- Jacobi-symbool 411, 416
- Jefferson
 - cylinder 146
 - Thomas 146
- JK-flipflop 503, 504, 505, 523
- k**
- Kroatisch 217
- Kahn, David 53, 60
- kana 68
- kanaalentropie 295
- kans 187
- kansrekening 249
- kansverdeling 203
- karacteristiek 338
- karacteristieke functie 481
- KASUMI 530, 536
- Kerckhoffs, Auguste 22, 25, 535
- kettingbreuk 432
- key 6
- key appearance 280
- KGB 69, 90, 111, 157
- kick 171
- klaaralfabet 125
- klaartekst 5
- klaartekstautoclaaf 141, 142
- KLEIN 386
- klinker 30, 209
- klokcyclus 464
- KL7 163
- knapzak-probleem 443
- known-plaintext 27, 279, 281, 282, 354
- Knudsen, Lars 387
- Knuth, Donald E. 153
- Koch, Hugo 163, 173
- Kocher, Paul 362
- kolomtranspositie 64, 83, 241
- koning
 - Karel I 105
 - Philips II 104
 - Philips IV 103

Willem I 76
koningin Mary van Schotland 56
Koot, Henri 55
Koreaanse oorlog 165
Kriegsmarine 53, 173
krimpgenerator 528, 529
Kryha, Alexander von 139
Kullback, Solomon 54
kwadratische rest 411

I

Law Enforcement Access Field 367
Legendre-symbool 411, 441
Lorenz Schlüsselzusatz 54, 55
Lange, A. 60
LEA 386
letterdistributie 30, 33, 118, 185, 190, 287
letterpatroon 32, 39
Ifsr 463, 473
lichtgolven 15
linealen 167
lineair equivalent 489
lineaire complexiteit 489
lineaire cryptanalyse 325, 354
lineaire operator 305
link-encryptie 12

logaritme 258
loterij 49
Lotus Notes 534
Lucifer 307, 322
Luftwaffe 173
Lyonet, Pierre 102

m

Mifare Classic 500
M-209 169
MacLaren-Marsaglia generator 155
MacLaren, M.D. 155
Magic 179
magisch vierkant 63
Markov-keten 272, 274, 275, 276, 284, 289
MARS 380, 381, 382, 383, 384, 386
Marsaglia, G. 155
massa 15
Matsui, Mitsuru 354
maximaalrij 475, 482
medeklinker 30, 209
meet-in-the-middle 321
menging 34
message exhaustion 43
message header 10
metereologische code 113

MICKEY 386
Microsoft 535
Midway 28
Miller-Rabin 417
min-entropie 290, 291
MISTY 536
MixColumn 390, 391, 396, 400
mobiele telefonie 10
mol 164
moment 187
 derde 194
 eerste 187
 tweede 187, 190, 191, 193, 201
monoalfabetische substitutie 193
monoalfabetische substitutie 28, 35, 37, 39, 42, 44, 102, 110, 176, 188
monome 88, 89
monome-dinome substitutie 88
morsecode 100
multilengte arithmetiek 410
multinomiale verdeling 192
multiplex generator 492
m-sequence 475
M-94 146, 147, 148
M-138 148

n

Nederlands 194
NIST 381
Nihilisten 72, 130
Ni-code 68
NP-compleet 443
NSA 50
natuurkunde 14
natuurlijke taal 30
NAVO 163
NBS 311, 312, 325
niet-lineariteit 305, 323
Nihilist
 substitutie 130
 transpositie 72
NIST 47, 311, 379, 436
Nohl, Karsten 533
noise entropy 269
nomenclatuur 102, 106, 118
nonrepudiation 14
NSA 9, 27, 50, 111, 164, 165, 310, 311, 312, 323, 324, 326, 351, 365, 366, 369, 420, 427, 442, 450, 451, 461, 533
Nuboer, J.F.W. 52, 68
nul 62

O

ondoenlijk 13
oneway functie 407
one-time pad 111, 157, 459, 460
ontcijferen 5
onveilig kanaal 7, 9
Onzekerheid 15
onzekerheid 261
onzekerheidsrelatie van Heisenberg 15
openbare exponent 421
openbare knapzakvector 444
Operation Torch 165
optelgetallen 110
optimale veiligheid 248, 252
OR 501
overlap 171
oververcijfering 61, 64, 110, 157

P

Pers-Z 148
Postscript Type 1 154
Pretty Good Privacy 372
paardensprong 63
padding 44, 313
Painvin, Georges Jean 101
patentrechten 312

pdf-bestand 512
perfecte communicatie 267
periode 49, 98, 119, 139, 153, 195, 461, 465, 481
periodiciteit 139
permutatie 29, 44, 49, 60, 81, 94
Petersen, Joseph Sidney 164
phi 410
phi-test 191, 195
PICARO 386
pinwiel 167
Piper, Fred 526
plaintext 5
Playfair 43, 94, 95, 96, 286
 baron Lyon 94
Pless generator 501, 503, 505, 514, 523
plugboard 174
PN-sequence 475
Poisson-verdeling 203
polarisatie 15, 16
polyalfabetische substitutie 118, 195, 198
Polybius-vierkant 85
polynomiaal algoritme 415
polynomiale tijd 444
Porta 119, 120, 121
 Giambattista della 91, 120

Postscript 511
potlood en papier 52
PRESENT 386
priemgetal 412
primitief polynoom 482
PRINCE 386
probabilistische oplosmethode 444
productvercijfering 41
progressie 133, 136
progressieve systemen 129
progressie-index 137
pseudorandom 49, 461
publieke sleutel 4, 6, 12, 13, 406
Purple 3, 51, 54, 163, 178, 179, 181

q

quantumcomputer 449
quantumcryptografie 8, 14
quantummechanica 14

r

Renaissance 54
RSA 412
RSA Company 451
raadspensionaris Van der Spiegel 84
radioactief verval 158

rate of the language 277
RC4 534, 535
RC5 374, 375, 376, 381, 382, 387
RC6 380, 381, 382, 383, 384, 387
realisatie 261
reciproke polynoom 481
recurrente betrekking 153
Red 51, 178, 179
redundantie 268, 277, 285
regenboogtabel 330
registertrap 463
Rejewski, Marian 53, 174
related key attack 318
Renaissance 3, 132
Renyi-entropie 291
Republiek van de Verenigde Nederlanden 102
restklasse 37, 41
Retter Charles 156
Riele, Herman te 426
Riemann, Bernhard 413
Rivest, Ronald 12, 374, 375, 387, 420, 423, 424, 534
Rohrbach, Hans 27
Romaji 179, 189
Rommel, Erwin 10, 114
ronde 308

rondetransformatie 308, 314, 332
rooster 73
rotor 36, 149
routetranspositie 61
Rozenkruisers 82
RSA 362, 374, 412, 420, 421, 422, 423, 425, 426, 427,
429, 430, 431, 432, 433, 436, 442, 443, 450
RSA Data Security 326, 534
ruis 267
ruiter 167
run 461
Rijmen, Vincent 389
Rijndael 380, 381, 382, 383, 384, 385, 389, 390, 391,
392, 396, 398, 438

S

Schoof's algoritme 441
Shark 178
Sukhotin algoritme 185
Sacco, Luigi 56, 60
Saint Cyr-lineaal 123
Saint-Cyr-lineaal 128
Scherbius, Arthur 163, 173
Schneier, Bruce 377, 378, 388
schuifregister 5, 51, 56
self-shrinking generator 529

Serpent 380, 382, 383, 384, 388
sessiesleutel 11, 365
Shamir, Adi 12, 331, 346, 347, 348, 352, 354, 362, 420,
423, 424, 443, 447, 531, 533
Shannon, Claude Elwood 4, 53, 255, 306
Shannon-entropie 289, 290, 291
ShiftRow 390, 391, 399
side-channel-analyse 325, 362, 432
Sigaba 163
SIGINT 7, 10
signaaloverdracht 269
signals intelligence 7
simultane kans 264
Skipjack 310, 364, 365, 366, 368, 369, 371, 372
sleutelalfabet 30
sleutelentropie 280
sleutelpaar 13
sleutelrij 110
sleutelstroom 457
sleuteluitwisseling 418
slide 169
SNOW 386
Snowden, Edward Joseph 365, 427, 450, 533
SOE 70
software balancing 364

- Solovay-Strassen 416, 417
somgenerator 523
Sorge, Richard 111
Soudart, E.A. 60
Spaen, Alexander Baron van 76
Spengler, R.P.C. 173
Spritz 534
Spruchschlüssel 177
Square 381, 389
stadhouder Willem IV 76
stagger 535
standaard 312
standaardafwijking 194
stappenpatroon 136
stationnair kanaal 293
statistische kenmerken 306
statistische methoden 54
steganografie 5, 72
stekkerbord 174
sterk priemgetal 414
sterke getuige 417
sterke leugenaar 417
stochast 297
stoorzender 9
stop-and-go generator 526
Stosslinie 114, 115
stroomverbruik 363
stroomvercijfering 381, 457
subsleutel 308, 314, 318
substitutie 42, 60
subtractor 110
Sukhotin, B.V. 217
superstijgende rij 444, 447
supervercijfering 61
symboolverzameling 257
symmetrie in positie 185, 219
S-doos 316, 317, 323, 324, 333, 334, 336, 337, 338, 339, 341, 345, 346, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 360, 361, 377, 382, 386, 388, 390, 536
- t**
Triton 178
Tunny 55
taal
Duits 189
Engels 189
Frans 189
italiaans 189
Japans 189
Nederlands 189
Russisch 189

Spaans 189
tap 464, 473
tekstcompressie 290
telex 459
terugkoppelfunctie 464, 473
thermodynamica 261
time-memory tradeoff 248, 324, 329, 330, 531
timing analyse 364
toestand 463
traffic analysis 10
transpositie 42, 60, 456
transpositieblok 60
transpositiesleutel 60, 65
transversale golf 15
trapdoor-oneway-functie 407
trifid 98
trigram 31, 186, 190
TripleDES 321, 322, 327, 379
turning grille 74, 288
tweedelige code 109
Twofish 380, 381, 382, 384, 388
twosquare 96
twijfel 264

u

Ultra Secret 173

uitputtend zoeken 26
uitvullen 44
Umkehrwalze 176
uniciteitsafstand 246, 273, 285, 291
unicode 24
US Army Double Transposition 70
US-Army Cipher Disk 237

v

Verkuyl, J.A. 165
valluik 407
vals alarm 512
Variant Vigenère 124, 125, 228, 231
variantie 192, 194
Vaticaan 102
veilig kanaal 8, 11
veiligheid 6
VENONA 111, 157
vercijferen 5
vercijfersleutel 6
Vernam 459
Vernam, Gilbert S. 459
Verne, Jules 76, 112
Vertakkingspunten 471
verwachtingswaarde 191, 192
Vigenère 119, 121, 123, 124, 125, 130, 133, 140, 159,

- 195, 196, 197, 203, 204, 219, 220, 231, 232, 286, 456
Blaise de 121
tabel 121, 123, 128, 129
voorwaardelijke kans 251, 264, 274
vrijmetselaar 82
- W**
- Walsingham 56
waarneming 269
waarschijnlijk woord 32
waarschijnlijkheid 15
warmteleer 261
weak key 49
wederzijdse informatie 269
Wehrmacht 173
Welchman, Gordon 174
WEP 535
Werftschlüssel 27
Wetterkurzschlüssel 113
Wheatstone 142, 233
Charles 94, 142
- whitening 310
witten 310
Wi-Fi 535
woordenboek 32, 112
WPA-TKIP 535
- X**
- XOR 501
- Y**
- Yardley, Herbert O. 178
- Z**
- Zahlenwurm 110
Zorro 386
zuivere schatter 191
zwakke sleutel 49, 319
zwarte kamer 102
97-shiki-obun In-ji-ki 178