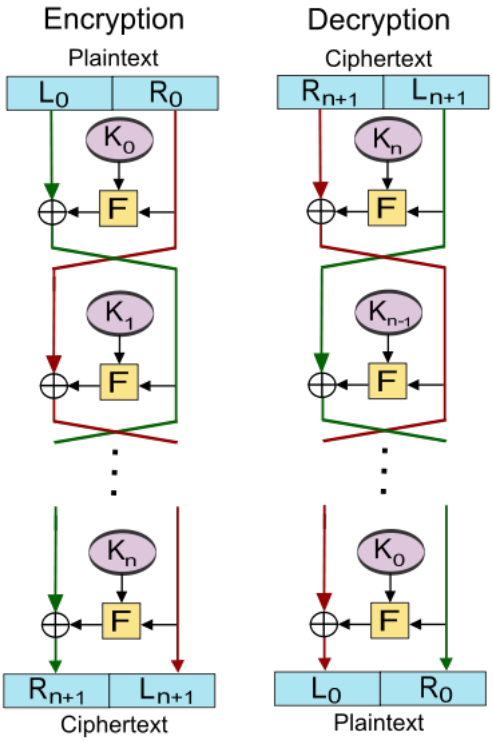
Quantum aanvallen op de 3-round Feistel Cipher

Sander Bet, Sebastian Zur

9 juli 2015

Projectverslag jaar 2

Begeleiding: dhr. dr. C. Schaffner



Korteweg-de Vries Instituut voor Wiskunde
Faculteit der Natuurwetenschappen, Wiskunde en Informatica
Universiteit van Amsterdam

Samenvatting

Cryptografie bestaat uit het beveiligen van informatie door middel van encryptie. De verzender van de informatie moet de informatie kunnen versleutelen (encrypten) en alleen de bedoelde ontvanger moet de oorspronkelijke informatie kunnen aflezen (decrypten). De Feistel cipher is een algoritme dat met behulp van willekeurige functies gebruikt wordt bij encryptie. Dit komt doordat wanneer het algoritme minstens drie ronden doorloopt, hij niet in polynomiale tijd te onderscheiden van een volledig willekeurige permutatie en willekeurige permutaties worden gebruikt in de cryptografie als encryptie methode. Met de nieuwe mogelijkheden van quantum computing is het echter niet meer vanzelfsprekend dat de Feistel cipher niet te onderscheiden is van een willekeurige permutatie. Wanneer dit het geval is, kan de Feistel cipher gekraakt worden en is de informatie versleuteld door de Feistel cipher niet meer veilig.

> What affected me most profoundly was the realization that the sciences of cryptography and mathematics are very elegant, pure sciences. I found that the ends for which these pure sciences are used are less elegant.
>
> James Sanborn

Titel: Quantum aanvallen op de 3-round Feistel Cipher
Auteurs:
Sander Bet, bet.sander@science.uva.nl, 10588728
Sebastian Zur, zur.sebastian@science.uva.nl, 10547800
Begeleiding: dhr. dr. C. Schaffner
Tweede beoordelaar: dhr. dr. C.G. Zaal
Einddatum: 9 juli 2015

Korteweg-de Vries Instituut voor Wiskunde
Universiteit van Amsterdam
Science Park 904, 1098 XH Amsterdam
`http://www.science.uva.nl/math`

inhoudsopgave

1 inleiding **4**

2 theorie **5**
2.1 qubits 5
2.1.1 superposition 5
2.1.2 gates 5
2.1.3 waarom qubits 6
2.2 willekeurige functies 6
2.2.1 distinguisher 6
2.3 verwaarloosbaarheid 6
2.3.1 pseudorandom functie familie 7
2.3.2 pseudorandom permutatie familie 7
2.4 Feistel cipher 7
2.4.1 klassieke sterkte 8
2.4.2 klassieke distinguisher 10
2.4.3 sterke pseudorandom permutatie 10
2.5 Simons algoritme 12
2.5.1 Simons probleem 12
2.5.2 quantum approach 12
2.6 quantum aanval op 3 round Feistel cipher 14

3 conclusie & discussie **17**

4 populaire samenvatting **18**

1 inleiding

Het veilig op kunnen sturen van informatie is erg nuttig. Niet alleen wordt het hedendaags gebruikt om ervoor te zorgen dat privé-informatie ook privé blijft, maar ook in tijden van oorlog is het een nodige vaardigheid. Het is dan ook niet raar dat cryptografie al gebruikt werd door de oude Romeinen. Doordat de kennis op het gebied van wiskunde in de jaren is toegenomen, zijn er ook betere encryptie methoden verzonnen. Zeker nadat de computers ontworpen waren, was het niet meer vanzelfsprekend dat bepaalde encryptiemethoden nog wel veilig waren.

Met het begin van de quantumcomputing in 1980 zijn er door de regels van de quantummechanica nieuwe algoritmen mogelijk gemaakt, die door computers snel uitgevoerd kunnen worden om bestaande encryptiemethoden te kraken. De huidige informatietheoretici die zich bezig houden met quantumcryptografie zijn dan ook niet alleen bezig met het creëren van nieuwe encryptie-methoden met behulp van quantummechanica, maar ook het bekijken welke 'veilige' encryptiemethoden niet meer veilig zijn onder aanval van een quantumcomputer.

In dit verslag zullen we het tweede zelf gaan doen. We werken naar een algoritme toe, dat vijf jaar geleden is verzonnen. Het kraakt een methode om pseudorandom permutaties te maken, wat voorheen als een veilige methode werd geacht. We zullen beginnen met een introductie in quantumcomputing. Vervolgens zullen we willekeurige en pseudorandom functies definiëren en beschrijven wat de Feistel cipher is. Tot slot zullen we het algoritme zelf bespreken.

Ook willen wij onze begeleider dhr. dr. Christian Schaffner hierbij hartelijk bedanken voor zijn begeleiding bij dit project.

2 theorie

2.1 qubits

2.1.1 superposition

In de klassieke informatica heeft een bit twee mogelijke toestanden: 0 of 1. Wanneer we echter de regels van de quantummechanica gebruiken, kunnen we een qubit (quantumbit) construeren, die zich in een superpositie van deze twee toestanden bevindt:

$$|\phi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle$$

waarbij $\alpha_0, \alpha_1 \in \mathbb{C}$ de amplitude van de respectievelijke toestand geven.

Deze toestanden vormen een orthonormale basis voor een vectorruimte, waarop een inproduct gedefinieerd is (een Hilbertruimte) en $|\phi\rangle$ is hierin een vector.

We kunnen twee dingen doen met een qubit, één is het meten ervan. Door het meten van een qubit vervalt de qubit in een klassieke bit, dus een 1 of een 0. De kans dat onze qubit $|\phi\rangle$ bij het meten een 0 wordt is gelijk aan $|\alpha_0|^2$ en de kans dat het een 1 wordt is $|\alpha_1|^2$. Hieruit volgt dat $|\alpha_0|^2 + |\alpha_1|^2 = 1$, een toestand heeft dus een norm van 1.

2.1.2 gates

Het andere dat we kunnen doen met qubits is gates toepassen. Gates zijn unitaire matrices en dus normbehoudend en inverteerbaar. Dit betekent dat wanneer er een gate op een qubit wordt uitgevoerd, de uitkomst nog steeds een vector met norm 1 is en de operatie 'ongedaan te maken' is, door de inverse van de gate op de uitkomst uit te voeren. Een voorbeeld van deze gates is de Hadamard gate:

$$H_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Deze Hadamard gate werkt alleen op 1 qubit, de Hadamard gate die werkt op n qubits ziet er als volgt uit:

$$(H_n)_{ij} = \frac{1}{\sqrt{2^n}}(-1)^{i \cdot j} \text{ met } i, j \in \{1, ..., n\}$$

waarbij $i \cdot j$ het binaire inproduct is. In binaire getallen is bijvoorbeeld $4 = 100$ en $3 = 011$, dus $4 \cdot 3 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = 0$

2.1.3 waarom qubits

Binnen de informatica wordt een vorm van encryptie als veilig beschouwd, zodra er geen algoritme in een polynomiaal aantal query's de encryptie kan kraken. Dit betekent dat voor een inputgrootte n het aantal query's dat het algoritme uitvoert gelijk is aan een polynoom $\in \mathbb{R}[X]$ met input n. Een query is wanneer er iets met de input van het algoritme wordt gedaan, bijvoorbeeld het lezen van de $5e$ bit. Er zijn echter encryptie methoden die niet meer veilig zijn als we gebruik maken van qubits, omdat er dan wel algoritmen bestaan die in een polynomiaal aantal query's de encryptie kraken. Een quantumcomputer met 100 qubits kan met behulp van een Hadamard gate deze 100 qubits omzetten naar een superpositie van 2^{100} states. Op deze manier wordt er met maar 100 qubits gerekend op 2^{100} verschillende toestanden. Door vervolgens unitaire matrices toe te passen, zoals de Hadamard gate, kunnen de amplitudes van deze toestanden worden aangepast, zodat bepaalde amplitudes bijvoorbeeld 0 worden. Dit noemen we interferentie en op deze manier hebben we invloed op welke toestanden we kunnen verwachten wanneer de superpositie vervalt bij het meten.

2.2 willekeurige functies

Een willekeurige deterministische functie is een willekeurig gekozen functie uit een verzameling met mogelijke functies. In ons geval is dat een familie van functies met $f : \{0,1\}^n \to \{0,1\}^n$. Dit geeft ons een verzameling van functies ter grootte van $|\{0,1\}^n|^{|\{0,1\}^n|} = 2^{n2^n}$.

2.2.1 distinguisher

D is een polynomiale tijd distinguisher dan en slechts dan als D een algoritme is dat met toegang tot de functie, dus mogelijkheid om query's te doen, in polynomiale tijd onderscheidt maakt tussen tussen objecten. In ons geval wordt er onderscheidt gemaakt tussen pseudorandom permutaties en willekeurige permutaties

2.3 verwaarloosbaarheid

Verwaarloosbaarheid is een begrip uit de cryptografie. Het zegt dat een functie klein genoeg wordt om niet meer mee te nemen in de benadering, omdat de kans dat het gebeurt er bijna niet is.

Definition 2.3.1. Een functie $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ is verwaarloosbaar in n, ook wel $negl(n)$ dan en slechts dan als voor alle positieve polynomen p $\exists N \in \mathbb{N}$ zodanig dat $\forall n > N \in \mathbb{N}$ geldt $f(n) < \frac{1}{p(n)}$.

Lemma 2.3.2. Een functie $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ is verwaarloosbaar als $\forall c \in \mathbb{R}$ $\exists N \in \mathbb{N}$ zodanig dat $\forall n > N \in \mathbb{N}$ geldt $f(n) < n^{-c}$.

Bewijs. Als er voor alle c er een N_1 bestaat zodat voor alle $n > N_1$ geldt $f(n) < n^{-c}$ dan geldt er voor elk polynoom met graad c dat $\exists N$ zodanig dat $\forall n > N$ geldt $f(n) < n^{-c}$. Namelijk $N = max\{N_1, N_2\}$ met N_2 twee keer de kleinste n zodat geldt $p(n) - n^c \leq n^c$. □

Remark 2.3.2.1. De functie $f(n) = 2^{-n}$ in verwaarloosbaar. Zij $c > 0$, kies N zodat geldt $\frac{N}{\log_2(N)} > c$. Dit kan omdat $\frac{N}{\log_2(N)}$ strikt stijgend is op (e, ∞). Dus nu geldt $\forall c \exists N$ zodanig dat $\forall n > N$ geldt $f(n) < n^{-c}$ en volgens lemma (2.3.2) geldt ook dat $f(n)$ verwaarloosbaar is.

2.3.1 pseudorandom functie familie

Een pseudorandom functie familie, PRF, is een verzameling van willekeurige deterministische functies, die in polynomiale tijd uit te reken zijn en niet in polynomiale tijd te onderscheiden zijn van willekeurige functies.

Formele Definitie: Laat D een polynomiale tijd distinguisher zijn en RF de verzameling van willekeurige functies. Dan is F PRF dan en slechts dan als

$$|Pr_{f \in F}[D^f(1^n) = 1] - Pr_{f \in RF}[D^f(1^n) = 1]| = negl(n)$$

2.3.2 pseudorandom permutatie familie

Een pseudorandom permutatie familie, PRP, is een verzameling van bijectieve functies, die in polynomiale tijd uit te rekenen zijn en niet in polynomiale tijd te onderscheiden van een willekeurige permutatie. Hierdoor is het mogelijk om een terugkeerbare encryptie te maken, die nagenoeg veilig is. De Feistel Cipher is een manier om van een PRF een PRP te maken.

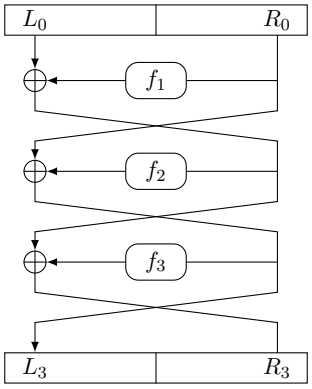
2.4 Feistel cipher

Feistel Cipher, $F^{(n)}$, is een algoritme om van een pseudorandom functie een pseudorandom permutatie te maken. Door de symmetrie in het algoritme kan je de versleutelde tekst decoderen door het algoritme achteruit te doorlopen. Drie iteraties van het algoritme zijn genoeg om een pseudorandom functie tot een pseudorandom permutatie te maken.

Laat $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ een pseudorandom functie en $K_1, K_2, \ldots, K_m \in \{0,1\}^n$ de sleutels voor respectievelijk ronde $1, 2, \ldots, m$. Splits de te versleutelen tekst van $2n$ bits in twee strings met een lengte van n bits, (L_0, R_0). Dan geldt er voor iteraties $i = 0, 1, \ldots, m-1$.

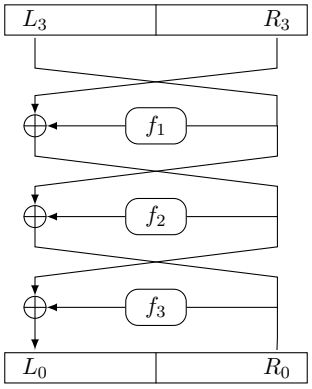
$$\begin{aligned} L_{i+1} &= R_i \\ R_{i+1} &= L_i \oplus F(R_i, K_{i+1}) \end{aligned}$$

en dan is (L_{n+1}, R_{n+1}) je versleutelde tekst.



De decryptie gaat als volgt je hebt (L_{n+1}, R_{n+1}) dan;

$$R_i = L_{i+1}$$
$$L_i = R_{i+1} \oplus F(L_{i+1}, K_{i+1})$$



2.4.1 klassieke sterkte

Als F een pseudorandom functie is, dan is $F^{(3)}$ een pseudorandom permutatie.

We willen bewijzen dat voor alle polynomiale distinguishers D geldt

$$|Pr_{f\in F}[D^{Feistel_{f_1,f_2,f_3}}(1^n) = 1] - Pr_{f\in RP}[D^f(1^n) = 1]| = negl(n)$$

met $f_i(x) = F(x, K_i)$.

Omdat F een pseudorandom functie is, is hij in polynomiale tijd niet te onderscheiden van een volledig willekeurige functie. Dus kunnen we aannemen dat F volledig willekeurig is. We laten $q = q(n)$ de polynomiale bovengrens zijn van het aantal orakel queries die D maakt. We mogen aannemen zonder verlies van algemeenheid dat D nooit twee keer dezelfde qeury maakt. We bekijken $D^{Feistel_{f_1,f_2,f_3}}$. Laat (L_0^i, R_0^i) de i-de query van D zijn. En laat (L_1^i, R_1^i), (L_2^i, R_2^i) en (L_3^i, R_3^i) de toestanden van de query zijn na respectievelijk ronde 1, 2 en 3 zijn. De invoer van de query van D is (L_0^i, R_0^i) en de uitvoer (L_3^i, R_3^i).

De distinguisher krijgt dus nooit te zien wat (L_1^i, R_1^i) en (L_2^i, R_2^i) zijn. We definieren een botsing in R_m wanneer er geldt $R_m^i = R_m^j$ en $i \neq j$. Voor vaste verschillende i, j. Als $R_0^i = R_0^j$ en $L_0^i \neq L_0^j$, dan geldt

$$R_1^i = L_0^i \oplus f_1(R_0^i) \neq L_0^j \oplus f_1(R_0^j) = R_1^j$$

. Wanneer $R_0^i \neq R_0^j$ dan zijn $f_1(R_0^i)$ en $f_1(R_0^j)$ uniform en onafhankelijk verdeeld, dus

$$Pr[L_0^i \oplus f_1(R_0^i) = L_0^j \oplus f_1(R_0^j)] = Pr[f_1(R_0^i) = L_0^j \oplus f_1(R_0^j) \oplus L_0^i] = 2^{-n}$$

. De kans dat we dus een botsing in R_1 is dus

$$Pr[\text{botsing in } R_1] \leq \frac{q^2}{2^n}$$

We bewijzen nu dat wanneer er geen botsing is in R_1, dat de kans op een botsing in R_2 verwaarloosbaar is. Voor vaste i, j nemen we aan dat er geen botsing is in R_1 dus $R_1^i \neq R_1^j \ \forall i, j : i \neq j$. Hierdoor zijn $f_2(R_1^i)$ en $f_2(R_1^j)$ uniform en onafhankelijk verdeeld. Daarom geldt;

$$Pr[L_1^i \oplus f_2(R_1^i) = L_1^j \oplus f_2(R_1^j) | \text{ geen botsing in } R_1] = 2^{-n} \tag{2.1}$$

De kans dat we een botsing hebben in R_2, als er geen botsing is in R_1 is;

$$Pr[\text{botsing in } R_2 | \text{geen botsing in } R_1] \leq \frac{q^2}{2^n} \tag{2.2}$$

. Aangezien q polynomiaal is, is deze kans verwaarloosbaar.

$L_3^i = R_2^i = L_1^i \oplus f_2(R_1^i)$, dus onder de aanname dat er geen botsingen zijn in R_1, zijn $L_3^1, L_3^2, \ldots, L_3^q$ uniform en onafhankelijk verdeeld in $\{0,1\}^n$. Met de extra aanname dat er ook geen botsing is in R_2, zijn $L_3^1, L_3^2, \ldots, L_3^q$ uniform verdeeld tussen alle string met q verschillende toestanden in $\{0,1\}^n$. Op gelijke wijze geldt voor $R_3^i = L_2^i \oplus f_3(R_2^i)$, als er geen botsing is in R_2 zijn $R_3^1, R_3^2, \ldots, R_3^q$ uniform verdeeld in $\{0,1\}^n$ en onafhankelijk van elkaar en van $L_3^1, L_3^2, \ldots, L_3^q$.

Dus wanneer je een query doet op $F^{(3)}$ met pseudorandom functies op q verschillende inputs, dan zijn de output toestanden met verwaarloosbare kleine onwaarschijnlijkheid $(L_3^1, R_3^1), (L_3^2, R_3^2), \ldots, (L_3^q, R_3^q)$ zo verdeeld dat de $\{L_3^i\}$ uniform en onafhankelijk verdeelde, verschillende n-bit strings zijn. En de $\{R_3^i\}$ uniform en onafhankelijke n-bit strings zijn.

Wanneer je een willekeurige permutatie hebt dan zijn voor q verschillende inputs, zijn de output toestanden $(L_3^1, R_3^1), (L_3^2, R_3^2), \ldots, (L_3^q, R_3^q)$ uniform en onafhankelijk verdeelde, verschillende 2n-bit strings. Dus de beste manier voor D om te onderscheiden, is te checken of $L_3^i = L_3^j$ voor $i \neq j$. Maar die kans is exponentieel klein. Dus $F^{(3)}$ is een pseudorandom permutatie.[1]

[1] Jonathan Katz en Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition.* Chapman&Hall/CRC Cryptography en Network Security Series, 2014. DOI: `https://www.crcpress.com/browse/series/chcrynetsec`.

2.4.2 klassieke distinguisher

We hebben hierboven bewezen dat er geen polynomiale tijd distinguisher is voor de Feistel cipher. Maar dat betekent niet dat er geen manier is om erachter te komen of je te maken hebt met een Feistel cipher of niet. Het is mogelijk om een exponentiële tijd distinguisher te maken voor de Feistel cipher.

Distinguisher D:

D krijgt als input string 1^{2n} en toegang tot de te onderscheiden functie $P : \{0,1\}^{2n} \to \{0,1\}^{2n}$ en de tabel van alle mogelijke 3 round Feistel ciphers en hun input met corresponderende output.

1. Maak een tabel van alle mogelijke inputs en de corresponderende outputs.
2. Vergelijk verkregen tabel met de tabel van mogelijke Feistel ciphers
   - Wanneer er een Feistel cipher in de tabel zit waarvoor alle outputs hetzelfde zijn, dan zeggen we dat P een 3 round Feistel cipher is.
   - Anders beslissen we dat het een willekeurige permutatie is.

Dit algoritme heeft minstens $O(2^n)$ query's nodig om een tabel te maken en moet hij het met maximaal 2^{3n} Feistel ciphers vergelijken om te onderscheiden. Dus het algoritme is zeker geen polynomiale tijd algoritme. Maar wanneer hij beslist dat het P een Feistel cipher is, is de kans dat het toch een willekeurige permutatie is $\frac{2^{3n}}{(2^n)!}$. Wat verwaarloosbaar is en dus een bij benadering deterministische distinguisher geeft. In de praktijk is deze aanpak echter niet bruikbaar aangezien het uitzonderlijk lang zou duren.

2.4.3 sterke pseudorandom permutatie

De 3 round Feistel cipher is echter geen sterke pseudorandom permutatie. Dat betekent dat wanneer we een distinguisher hebben die de inverse van de te onderscheiden permutatie P kan gebruiken. Er hij in polynomiale tijd kan onderscheiden of P een Feistel cipher of een willekeurige permutatie is.

Distinguisher D:

D krijgt als input string 1^{2n} en toegang tot de te onderscheiden functie $P : \{0,1\}^{2n} \to \{0,1\}^{2n}$ en zijn inverse $P^{-1} : \{0,1\}^{2n} \to \{0,1\}^{2n}$.

1. Doe een query (L_0^i, R_0) en krijg (L_3^i, R_3^i) terug.
2. Doe een query (L_0^j, R_0) met $L_0^i \neq L_0^j$ en krijg terug (L_3^j, R_3^j).
3. Doe een inverse query $(L_3^j, R_3^j \oplus L_0^i \oplus L_0^j)$ en krijg (L_0^q, R_0^q).
   - Wanneer $R_0^q = R_0 \oplus L_3^i \oplus L_3^j$ dan is het een Feistel cipher.
   - Anders is het een random permutatie

Neem aan dat P een Feistel cipher is, dan geldt voor het versleutelen:

L_0	R_0
$L_1 := R_0$	$R_1 := L_0 \oplus f_1(R_0)$
$L_2 := L_0 \oplus f_1(R_0)$	$R_2 := R_0 \oplus f_2(L_0 \oplus f_1(R_0))$
$L_3 := R_0 \oplus f_2(L_0 \oplus f_1(R_0))$	$R_3 := L_0 \oplus f_1(R_0) \oplus f_3(R_0 \oplus f_2(L_0 \oplus f_1(R_0)))$

en voor het ontsleutelen gebruiken we de volgende tabel:

L_3	R_3
$L_2 := R_3 \oplus f_3(L_3)$	$R_2 := L_3$
$L_1 := L_3 \oplus f_2(R_3 \oplus f_3(L_3))$	$R_1 := R_3 \oplus f_3(L_3)$
$L_0 := R_3 \oplus f_3(L_3) \oplus f_1(L_3 \oplus f_2(R_3 \oplus f_3(L_3)))$	$R_0 := L_3 \oplus f_2(R_3 \oplus f_3(L_3))$

Zo kunnen we eenvoudig zien dat onze query's de volgende resultaten hebben:

$$L_3^i = R_0 \oplus f_2(L_0^i \oplus f_1(R_0))$$
$$R_3^i = L_0^i \oplus f_1(R_0) \oplus f_3(R_0 \oplus f_2(L_0^i \oplus f_1(R_0)))$$

$$L_3^j = R_0 \oplus f_2(L_0^j \oplus f_1(R_0))$$
$$R_3^j = L_0^j \oplus f_1(R_0) \oplus f_3(R_0 \oplus f_2(L_0^j \oplus f_1(R_0)))$$

$$R_0^q = L_3^j \oplus f_2(R_3^j \oplus L_0^i \oplus L_0^j \oplus f_3(R_2^j))$$

We weten dat $R_3^j = L_0^j \oplus f_1(R_0) \oplus f_3(R_2^j)$ dus $R_3^j \oplus L_0^i \oplus L_0^j \oplus f_3(R_2^j) = L_0^i \oplus f_1(R_0)$

$$R_0^q = L_3^j \oplus f_2(L_0^i \oplus f_1(R_0))$$

Nu vervangen we $f_2(L_0^i \oplus f_1(R_0)) = R_0 \oplus L_3^i$.

$$R_0^q = L_0 \oplus L_3^i \oplus L_3^j$$

We zien dus dat D voor elke Feistel cipher vindt dat $R_0^q = L_3^j \oplus f_2(L_0^i \oplus f_1(R_0))$. Het is echter mogelijk om een willekeurige permutatie te vinden waar dit ook voor geldt. De kans dat dit geldt voor een willekeurige permutatie is $\frac{1}{2^n}$. Hierdoor is de kans op een fout-positief van de distinguisher weer verwaarloosbaar maakt. En is dus een goede manier om een 3 round Feistel cipher te onderscheiden van een willekeurige permutatie.

2.5 Simons algoritme

2.5.1 Simons probleem

Simons probleem beschrijft een functie $f : \{0,1\}^n \to \{0,1\}^n$, waarbij geldt dat $\exists s \in \{0,1\}^n \backslash \{0\}^n$ zodat $\forall y, z \in \{0,1\}^n \; f(y) = f(z) \Leftrightarrow y = z$ of $y = z \oplus s$. Het probleem is het vinden van s.

2.5.2 quantum approach

Er is echter wel een quantum algoritme, dat s vindt in polynomiale tijd:

1. We beginnen met $2n$ qubits met de toestand $|0\rangle$;

$$|\phi_0\rangle = |0^n\rangle \, |0^n\rangle \, .$$

2. Een Hadamard gate toepassen op de eerste n qubits geeft ons:

$$|\phi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{i \in \{0,1\}^n} |i\rangle \, |0^n\rangle \, .$$

3. Dan passen we hierop de gate $U_f \, |x\rangle \, |y\rangle = |x\rangle \, |f(x) \oplus y\rangle$ toe:

$$|\phi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{i \in \{0,1\}^n} |i\rangle \, |f(i)\rangle \, .$$

4. Door nu de laatste n bits te meten, vervallen deze bits tot een zekere $y \in \{0,1\}^n$. We weten dat er twee toestands voor x zijn zodat $f(x) = y$, namelijk $x = i$ en $x = i \oplus s$, voor een enkele $i \in \{0,1\}^n$;

$$|\phi_3\rangle = \frac{1}{\sqrt{2}} (|i\rangle + |i \oplus s\rangle) \, |y\rangle \, .$$

5. Een Hadamard gate toepassen op de eerste n qubits geeft dan:

$$\begin{aligned} |\phi_4\rangle &= \frac{1}{\sqrt{2^{n+1}}} \Big(\sum_{v \in \{0,1\}^n} ((-1)^{i \cdot v} + (-1)^{(i \oplus s) \cdot v}) \, |v\rangle \, |y\rangle \\ &= \frac{1}{\sqrt{2^{n+1}}} \Big(\sum_{v \in \{0,1\}^n} ((-1)^{i \cdot v} (1 + (-1)^{s \cdot v}) \, |v\rangle \, |y\rangle \end{aligned}$$

6. De amplitude van $|v\rangle \, |y\rangle$ is dus ongelijk aan 0 als $s \cdot v = 0 \bmod 2$.

   Wanneer we dus een meting doen, dan zijn onze eerste n bits een vector v uit een $n-1$ dimensionale vectorruimte ($v \in \{0,1\}^n$ en $v \cdot s = 0 \bmod 2$) met 2^{n-1}

elementen. Als we $n-1$ lineaire onafhankelijke v_j vectoren kunnen vinden, dan weten we dat:

$$\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{n-1} \end{bmatrix} s = 0^{n-1} \bmod 2$$

Omdat al onze v_j lineair onafhankelijk moeten zijn, kan v_1 dus niet gelijk zijn aan de nulvector. Na v_1 gemeten te hebben (en deze is ongelijk aan 0^n), is de kans dat v_2 lineair onafhankelijk is aan v_1 gelijk aan de kans dat $v_2 \notin span\{v_1\}$. We werken met bits, dus $|span\{v_1, ..., v_k\}| = 2^k$, wanneer deze vectoren allemaal lineair onafhankelijk zijn. Hieruit volgt dat:

$$\begin{aligned} P[\text{twee lineair onafhankelijke vectoren}] &= P[v_2 \notin span\{v_1\} | v_1 \neq 0^n] * P[v_1 \neq 0^n] \\ &= (1 - \frac{1}{2^{n-2}})(1 - \frac{1}{2^{n-1}}) \end{aligned}$$

Er vanuit gaande dat v_1 en v_2 lineair onafhankelijk zijn, dan is de kans dat v_3 geen lineaire combinatie is van deze twee vectoren gelijk aan:

$$\begin{aligned} &P[\text{drie lineair onafhankelijke vectoren}] \\ &= P[v_3 \notin span\{v_1, v_2\} | v_2 \notin span\{v_1\} \wedge v_1 \neq 0^n] \ * \ P[v_2 \notin span\{v_1\} | v_1 \neq 0^n] \ * \ P[v_1 \neq 0^n] \\ &= (1 - \frac{1}{2^{n-3}})(1 - \frac{1}{2^{n-2}})(1 - \frac{1}{2^{n-1}}) \end{aligned}$$

De kans dat we $n-1$ lineaire onafhankelijke vectoren krijgen is dus gelijk aan:

$$\begin{aligned} \prod_{i=1}^{n-1}(1 - \frac{1}{2^{n-i}}) &= (1 - \frac{1}{2}) \prod_{i=2}^{n-1}(1 - \frac{1}{2^i}) \\ &\geq (1 - \frac{1}{2})(1 - \sum_{i=2}^{n-1} \frac{1}{2^i}) \\ &\geq \frac{1}{2}(\frac{5}{2} - \sum_{i=0}^{\infty} \frac{1}{2^i}) \\ &= \frac{1}{2}(\frac{5}{2} - \frac{1}{1 - \frac{1}{2}}) = \frac{1}{4} \end{aligned}$$

Hierbij hebben we gebruik gemaakt van het feit dat $(1-a_1)(1-a_2)...(1-a_n) \geq 1 - (a_1 + a_2 + ... + a_n)$ als $a_1, ..., a_n \in [0, 1]$.

7. Wanneer we dus $n-1$ vectoren verkrijgen via bovenstaande methode, dan is er een kans van $\frac{1}{4}$ dat deze allemaal lineair onafhankelijk zijn. We kunnen dus verwachten

dat bij $4(n-1)$ uitvoeringen, we $n-1$ lineair onafhankelijke vectoren krijgen. Nu we onze lineair onafhankelijke vectoren hebben, kan het lineaire stelstel opgelost worden. s heeft echter n (en niet $n-1$) onbekenden, dus de oplossing van het stelsel zal dimensie 1 hebben en komen de oplossingen uit $span\{s\} = \{s, 0^n\}$. Gezien $s \neq 0^n$ weten we nu s. Elke keer dat we een vector v_j vekrijgen, wordt er maar één query uitgevoerd(namelijk als we U_f toepassen) en is de complexiteit van het verkrijgen van $n-1$ lineair onafhankelijke vectoren $O(n)$.[2]

2.6 quantum aanval op 3 round Feistel cipher

Met behulp van een quantum computer is het mogelijk om een Feistel cipher onveilig te maken, mits deze voldoet aan een aantal eisen:

1. De Feistel cipher bestaat uit drie iteraties.

2. De gebruikte functies zijn pseudorandom permutaties.

Als aan deze voortoestanden is voldaan, dan kan er in polynomiale tijd over een pseudorandom functions, gemaakt door deze Feistel cipher, gezegd worden of deze permutatie daadwerkelijk random is of dat deze gemaakt is door een 3 round Feistel cipher bestaande uit enkel permutaties, met het gevolg dat deze permutatie dus niet meer pseudorandom is.

Zonder verlies van algemeenheid stellen we dat de inputlengte van onze Feistelcipher $2n$ is. We delen de input string x door de helft: $x = a||c$ en dan bekijken we de functie W, die de input van de permutatie afbeeldt op de eerste n bits van de output:

$$W(x) = W(a||c) = c \oplus f_2(a \oplus f_1(c))$$

Neem $a, \alpha, \beta \in \{0,1\}^n$, met $\alpha \neq \beta$ en $b \in \{0,1\}$, dan kunnen we nu een functie f definiëren:

$$f(b||a) = \begin{cases} W(a||\alpha) \oplus \beta & \text{als } b = 0 \\ W(a||\beta) \oplus \alpha & \text{als } b = 1 \end{cases}$$

We kunnen het volgende over deze functie zeggen als de permutatie afkomstig is van de Feistel cipher:

Lemma 2.6.1. $f(b||a) = f(b'||a') \Leftrightarrow b' = b \oplus 1 \ \wedge \ a' = a \oplus z$, met $z = f_1(\alpha) \oplus f_1(\beta)$

Bewijs. $(\rightarrow)$

1. $b = b' = 0$
   $f(b||a) = W(a||\alpha) \oplus \beta = \alpha \oplus f_2(a \oplus f_1(\alpha)) \oplus \beta$
   $f(b'||a') = W(a'||\alpha) \oplus \beta = \alpha \oplus f_2(a' \oplus f_1(\alpha)) \oplus \beta$
   Omdat f_2 een permutatie (en dus injectief) is, geldt $a = a'$ en dus $b||a = b'||a'$

[2] Ronald de Wolf. *Quantum Computing:Lecture Notes.* CWI, Amsterdam, 2015. DOI: `http://homepages.cwi.nl/~rdewolf/qcnotes.pdf`.

2. $b = 1, b' = 0$
   $f(b||a) = W(a||\alpha) \oplus \beta = \alpha \oplus f_2(a \oplus f_1(\alpha)) \oplus \beta$
   $f(b'||a') = W(a'||\beta) \oplus \alpha = \beta \oplus f_2(a' \oplus f_1(\beta)) \oplus \alpha$
   Omdat f_2 een permutatie (en dus injectief) is, geldt $a' = f_1(\alpha) \oplus f_1(\beta) \oplus a = z \oplus a$

3. $b = 0, b' = 1$

   $f(b'||a') = W(a'||\alpha) \oplus \beta = \alpha \oplus f_2(a' \oplus f_1(\alpha)) \oplus \beta$
   Omdat f_2 een permutatie (en dus injectief) is, geldt $a' = f_1(\alpha) \oplus f_1(\beta) \oplus a = z \oplus a$

4. $b = b' = 1$
   $f(b||a) = W(a||\alpha) \oplus \beta = \alpha \oplus f_2(a \oplus f_1(\alpha)) \oplus \beta$
   $f(b'||a') = W(a'||\alpha) \oplus \beta = \alpha \oplus f_2(a' \oplus f_1(\alpha)) \oplus \beta$
   Omdat f_2 een permutatie (en dus injectief) is, geldt $a = a'$ en dus $b||a = b'||a'$

($\leftarrow$)

1. $b = 0$
   $f(b||a) = W(a||\beta) \oplus \alpha = \beta \oplus f_2(a \oplus f_1(\beta)) \oplus \alpha$
   $f(b'||a') = W((\alpha \oplus z)||\alpha) \oplus \beta = \alpha \oplus f_2(\alpha \oplus z \oplus f_1(\alpha)) \oplus \beta$
   $= \alpha \oplus f_2(\alpha \oplus f_1(\alpha) \oplus f_1(\beta) \oplus f_1(\alpha)) \oplus \beta = \alpha \oplus f_2(a \oplus f_1(\beta)) \oplus \beta$
   Dus $f(b||a) = f(b'||a')$.

2. $b = 1$
   $f(b||a) = W(a||\alpha) \oplus \beta = \alpha \oplus f_2(a \oplus f_1(\alpha)) \oplus \beta$
   $f(b'||a') = W((\alpha \oplus z)||\beta) \oplus \alpha = \beta \oplus f_2(\alpha \oplus z \oplus f_1(\beta)) \oplus \alpha$
   $= \beta \oplus f_2(\alpha \oplus f_1(\alpha) \oplus f_1(\beta) \oplus f_1(\beta)) \oplus \alpha = \beta \oplus f_2(a \oplus f_1(\alpha)) \oplus \alpha$
   Dus $f(b||a) = f(b'||a')$.

Als we $(b||a) = x$ nemen, dan is dit lemma equivalent met:

$$f(x) = f(x') \Leftrightarrow x' = x \oplus (1||z)$$

[3] □

Met behulp van dit lemma kan het algoritme gemaakt worden:

1. Begin met een lege set V en de de $2n + 1$ 0 qubits:

$$|\phi_0\rangle = \left|0^{n+1}\right\rangle |0^n\rangle$$

2. Een Hadamard gate toepassen op de eerste $n + 1$ qubits geeft ons:

$$|\phi_1\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{i \in \{0,1\}^{n+1}} |i\rangle |0^n\rangle$$

[3] Hidenori Kuwakado en Masakatu Morii. „Quantum Distinguisher Between the 3-Round Feistel Cipher and the Random Permutation". In: *ISIT 2010, Austin, Texas, U.S.A., June 13 - 18* (2010), p. 2682–2685. DOI: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5513654.

3. Dan passen we hierop de gate $U_f |x\rangle |y\rangle = |x\rangle |f(x) \oplus y\rangle$ toe, waarbij $x \in \{0,1\}^{n+1}$ en $y \in \{0,1\}^n$:

$$|\phi_2\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{i \in \{0,1\}^{n+1}} |i\rangle |f(i)\rangle$$

4. Door nu de laatste n bits te meten, vervallen deze bits tot een zekere $y \in \{0,1\}^n$. Alle $x \in \{0,1\}^{n+1}$ waarvoor geldt dat $f(x) = y$ stoppen we in X_y (In het geval dat f afkomstig is van onze Feistel cipher geldt er dat $X_y = \{i, i \oplus s\}$ voor enkele $i, s \in \{0,1\}^{n+1}$):

$$|\phi_3\rangle = \frac{1}{\sqrt{|X_y|}} \sum_{i \in X_y} |i\rangle |y\rangle$$

We sommeren niet over alle y, omdat door het meten van $f(i)$ de superpositie is vervallen en er dus nog maar één $f(i) = y$ over is.

5. Een Hadamard gate toepassen op de eerste $n+1$ qubits geeft dan:

$$|\phi_4\rangle = \frac{1}{\sqrt{|X_y| 2^{n+1}}} \sum_{v \in \{0,1\}^n} (-1)^{i \cdot v} |v\rangle |y\rangle$$

6. Nu gaan we net als bij Simons algoritme lineair onafhankelijke vectoren v_j zoeken om het volgende lineaire stelsel op te lossen:

$$\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \begin{bmatrix} 1 \\ s \end{bmatrix} = 0^n \bmod 2$$

We weten volgens Simons algoritme dat dit in $O(n)$ kan

7. Nu hebben we een niet triviale oplossing voor s en nu kan er gekeken worden of het volgende geldt voor een willekeurige $u \in \{0,1\}^{n+1}$:

$$f(u) = f(u \oplus (1||s))$$

Als de gelijkheid opgaat, dan is f afkomstig van de Feistel cipher. Zo niet, dan is f een willekeurige permutatie. Dit komt omdat lemma 2.6.1 alleen geldt voor wanneer de functie f werkt op de Feistel cipher. Willekeurige permutaties hebben deze eigenschap niet.

8. Als de permutatie niet afkomstig is van de Feistel cipher, dan is de verkregen s willekeurig. De kans dat het algoritme zegt dat de permutatie dan wel van de Feistel cipher afkomstig is, is dus $\frac{1}{2^n}$. De output van f is dan immers uniform verdeeld en we willen dat 2^n bits overeenkomen. Deze kans is exponentieel klein en dus verwaarloosbaar.

3 conclusie & discussie

We hebben eerst naar de quantummechanica gekeken om te zien hoe we met qubits kunnen werken. Vervolgens hebben we willekeurige functies gedefinieerd en gedefinieerd wanneer een Feistel cipher veilig is. Ook hebben we de Feistel cipher zelf besproken en laten zien waarom deze veilig is in het klassieke geval. Met deze kennis en Simons algoritme konden we de quantum distinguisher behandelen die de 3 round Feistel cipher met inwendige permutaties kan onderscheiden van een willekeurige functie.

We hebben gezien dat door de eigenschappen van de quantummechanica er een algoritme bestaat wat in polynomiale tijd van een permutatie kan zeggen of het van een 3 round Feistel cipher afkomt of dat het een willekeurige permutatie is. We eisen echter wel dat de Feistel cipher bestaat uit (pseudorandom) permutaties. De paper die dit algoritme heeft beschreven is pas vijf jaar geleden gepubliceerd, er is dus nog niet zoveel bekend over quantumaanvallen op Feistel ciphers. Algoritmen die de Feistel cipher kraken, wanneer deze uit meer dan 3 ronden bestaat of als de functies in elke ronde geen permutaties, zijn nog niet bekend. Het is echter wel bewezen dat als de Feistel cipher uit een polynomiaal aantal ronden bestaat, deze dan veilig is tegen quantumaanvallen.

4 populaire samenvatting

Informatie wordt door computers opgeslagen als bits. Deze bits hebben de waarde 0 of 1. Met n bits kun je dan dus 2^n verschillende toestanden beschrijven. Je kan informatie ook sturen naar iemand anders, dit doe je dan door de bits te sturen. Soms wil je echter dat niemand anders deze informatie leest, als deze bijvoorbeeld je wachtwoorden bevat. Je kan je informatie dan versleutelen. Dan verander je jouw bits voordat je ze verstuurt en dan kunnen alleen mensen de orignele bits achterhalen als ze weten hoe je jouw bits hebt verandert. Een manier van versleutelen is het omwisselen van bits, dit noemen we dan een permutatie. Dan stuur je in plaats van 1001 bijvoorbeeld 1010 door de derde en vierde bit om te wisselen. Nu kan iedereen je bericht aflezen als ze weten wat voor permutatie je hebt gebruikt. Daarom worden er pseudorandom permutaties gebruikt. Dit zijn permutaties, waarbij mensen en computers niet kunnen zeggen welke permutatie het is en daarom maar het beste een willekeurige permutatie kunnen pakken en kijken of het werkt. Een manier om zo een pseudorandom permutatie te maken is door middel van een Feistel cipher. Tegenwoordig moeten encryptie methoden niet alleen rekening houden met normale computers, maar ook met quantumcomputers. Quantumcomputers gebruiken geen bits, maar qubits. Qubits hebben niet de waarde 0 of 1, maar alle twee tegelijk. Pas als je de qubit meet, dan wordt het een van de twee waardes. Deze handige eigenschap zorgt voor allemaal nieuwe algoritmen, deze noemen we quantumalgoritme. In dit verslag kijken we naar zo een quantumalgoritme, wat bepaalde Feistel ciphers onveilig maakt. Dit komt doordat het algoritme wel verschil ziet tussen de Feistel cipher en een willekeurige permutatie.

Bibliografie

[1] Jonathan Katz en Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. Chapman&Hall/CRC Cryptography en Network Security Series, 2014. DOI: https://www.crcpress.com/browse/series/chcrynetsec.

[2] Ronald de Wolf. *Quantum Computing:Lecture Notes*. CWI, Amsterdam, 2015. DOI: http://homepages.cwi.nl/~rdewolf/qcnotes.pdf.

[3] Hidenori Kuwakado en Masakatu Morii. „Quantum Distinguisher Between the 3-Round Feistel Cipher and the Random Permutation". In: *ISIT 2010, Austin, Texas, U.S.A., June 13 - 18* (2010), p. 2682–2685. DOI: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5513654.