Now UC It

A COMPOSABLE FRAMEWORK

JAN CZAJKOWSKI
ESTEBAN LANDERRECHE
CRYPTOLOGY STUDENT SEMINAR

APRIL 19, 2019
We start with an example\(^1\)

- Start with key \(k = (k_1, k_2)\)
- Let \(\{h_{k_1}\}\) be a family of strongly universal hash functions
- Let \(f_{h_{k_1}}(b) = \begin{cases} 
00 & \text{if } b = 0 \\
01 & \text{if } b = 1 \text{ with probability } 1/2 \\
10 & \text{if } b = 1 \text{ with probability } 1/2 
\end{cases}\)
- Let \(\{f_n\}_n : \{0, 1\}^n \rightarrow \{0, 1\}^{2n}\) be a family of functions where \(f_n\) applies bitwise \(f\) to a string of length \(n\)

\(^1\)Shamelessly stolen example from Christopher Portmann
**Authenticated Encryption**

### Encryption Function Enc*

- Append hash $u = m||h_{k_1}(m)$ *(secure authentication)*
- Apply $f_n$: $v = f_n(u)$
- Encrypt with OTP: $c = v \oplus k_2$ *(perfect secrecy)*

### Decryption Function Dec*

- Decrypt with OTP: $v' = c \oplus k_2$
- Invert $f_n$: $v' \rightarrow u' \lor \perp$
- Parse $u' = m'||t$ and check $t == h_{k_1}(m')$
Secure Communication?

If $m = 1$:

$$c = \text{Enc}^*(m, k, h, f_n, \text{OTP})$$

If $m = 0$:

$$c' = \text{Dec}^*(c', k, h, f_n^{-1}, \text{OTP})$$

$$c' = c \oplus 11||O|^{c|^{-2}}$$
Secure Communication?

If $m = 1|\|m'$

$$\text{Dec}^*(c') = f_n^{-1}(c' \oplus k_2)$$
If $m = 1 \parallel m'$

$$\text{Dec}^*(c') = f_n^{-1}(c \oplus 11||O|^{c|-2} \oplus k_2)$$
If $m = 1\|m'$

\[
\text{Dec}^*(c') = f_n^{-1}(f_n(m\|t) \oplus 11\|\overline{0}^{c-2})
\]
If $m = 1 \| m'$

$$\text{Dec}^*(c') = f_2^{-1}(f_2(1) \oplus 11) \| f_{n-2}^{-1}(f_{n-2}(m' \| t))$$
Secure Communication?

If \( m = 1 \| m' \)  
\[
\text{Dec}^*(c') = 1 \| f_{n-2}^{-1}(f_{n-2}(m' \| t))
\]
Secure Communication?

\[
\begin{align*}
\text{If } m = 1 \| m' & \quad \text{Dec}^*(c') = m \| t \\
\end{align*}
\]
**Secure Communication?**

If $m = 1||m'$ \[\text{Dec}^*(c') = m||t\]

If $m = 0||m'$ \[\text{Dec}^*(c') = f_2^{-1}(f_2(0) \oplus 11)||f_{n-2}^{-1}(f_{n-2}(m'||t))\]
**Secure Communication?**

If $m = 1||m'$, then $\text{Dec}^*(c') = m||t$

If $m = 0||m'$, then $\text{Dec}^*(c') = f_2^{-1}(11)||f_{n-2}^{-1}(f_{n-2}(m'||t))$

$c' = c \oplus 11||0|c|^{-2}$
Secure Communication?

If $m = 1|m'$, $\text{Dec}^*(c') = m|t$

If $m = 0|m'$, $\text{Dec}^*(c') = \perp$
<table>
<thead>
<tr>
<th></th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The UC Framework</td>
</tr>
<tr>
<td></td>
<td>- Efficient Computation</td>
</tr>
<tr>
<td></td>
<td>- Executions</td>
</tr>
<tr>
<td>2</td>
<td>Security</td>
</tr>
<tr>
<td>3</td>
<td>Composability</td>
</tr>
<tr>
<td>4</td>
<td>Building a Protocol</td>
</tr>
</tbody>
</table>
The UC Framework
What? We want *composable* security definitions: protocols maintain the security guarantees even when used in larger systems.
Our main goal

- **What?** We want composable security definitions: protocols maintain the security guarantees even when used in larger systems.

- **How?** We describe an ideal functionality ($\mathcal{F}$) that describes what we expect from a computation and then show that $\pi$ behaves in the same way no matter the environment.
**Building Blocks**

**Environment** $\mathcal{E}$ The entity interacting with the protocol.

**Adversary** $\mathcal{A}$ The entity that attempts to interfere with the correct execution of the protocol.

**Parties** $\mathcal{P}_i$ The entities that participate in the protocol.

**Simulator** $\mathcal{S}$ The entity that simulates a different protocol.

**Ideal Functionality** $\mathcal{F}$ The entity that we are trying to achieve.
Building Blocks

**Environment** $\mathcal{E}$  The *entity* interacting with the protocol.

**Adversary** $\mathcal{A}$  The *entity* that attempts to interfere with the correct execution of the protocol.

**Parties** $P_i$  The *entities* that participate in the protocol.

**Simulator** $S$  The *entity* that simulates a different protocol.

**Ideal Functionality** $\mathcal{F}$  The *entity* that we are trying to achieve.
Interactive Turing Machines (ITMs)

Definition
An Interactive Turing Machine consists of a read/write head and a number of special tapes:

**Identity tape**  
*Read-only*, contains the *extended identity* of the ITM

**Outgoing message tape**  
Holds the outgoing message

**Externally-writeable tapes**  
*Read-only* and *read-once*
- Input tape
- Subroutine-output tape
- Backdoor tape

We know of the existence of the Universal Turing Machine
Definition

A protocol $\pi$ is a (single) ITM.

Definition

The extended identity of an ITM $M$ is a triple $(\mu, \text{sid}, \text{pid})$

- $\mu$ The code of the machine
- sid Determines which session of the protocol is being run
- pid Determines the identity of the party represented by the ITM

Definition

ITM instance or ITI of $M$ is the contents of the identity tape of $M$. 
A **system of ITMs** is a pair \( S = (I, C) \) where \( I \) is the *initial ITM* and \( C \) is a **control function** \( C : \{0, 1\}^* \rightarrow \{\text{allow, disallow}\} \).
Definition

Let $T : N \to N$. A Turing machine $M$ is said to be $T$-bounded if, given any input of length $n$, $M$ halts within at most $T(n)$ steps.
**Definition**

Let $T : N \rightarrow N$. A Turing machine $M$ is said to be *$T$-bounded* if, given any input of length $n$, $M$ halts within at most $T(n)$ steps.

How to apply to our setting?
Definition
Let $T : N \to N$. A Turing machine $M$ is said to be $T$-bounded if, given any input of length $n$, $M$ halts within at most $T(n)$ steps.

How to apply to our setting?

- Runtime dependent only on the security parameter
Let $T : N \to N$. A Turing machine $M$ is said to be $T$-bounded if, given any input of length $n$, $M$ halts within at most $T(n)$ steps.

How to apply to our setting?

- Runtime-dependent only on the security parameter
- Make the bound for the entirety of the protocol, not per activation
**Resource-bounded computation**

**Definition**

Let $T : N \to N$. A Turing machine $M$ is said to be **$T$-bounded** if, given any input of length $n$, $M$ halts within at most $T(n)$ steps.

How to apply to our setting?

- Runtime dependent only on the security parameter
- Make the bound for the entirety of the protocol, not per activation
- Bound each ITM locally
**Definition**

Let $T : N \rightarrow N$. A Turing machine $M$ is said to be **$T$-bounded** if, given any input of length $n$, $M$ halts within at most $T(n)$ steps.

**How to apply to our setting?**

- Runtime dependent only on the security parameter
- Make the bound for the entirety of the protocol, not per activation
- Bound each ITM locally
1. The initialization message contains a *runtime budget* depends on the security parameter and the input. Every message between ITIs will contain an *import field* which is a natural number.
1. The initialization message contains a **runtime budget**
depends on the security parameter and the input. Every
message between ITIs will contain an **import field** which is a
natural number

2. The import of an ITI $M$ is the overall imports of the messages
received by $M$ minus the overall imports of the messages
sent by $M$
1. The initialization message contains a **runtime budget** that depends on the security parameter and the input. Every message between ITIs will contain an **import field** which is a natural number.

2. The import of an ITI $M$ is the overall imports of the messages received by $M$ minus the overall imports of the messages sent by $M$.

3. Import acts effectively as *computation tokens* where ITI’s send and receive tokens.
1. The initialization message contains a **runtime budget** depends on the security parameter and the input. Every message between ITIs will contain an **import field** which is a natural number.

2. The import of an ITI $M$ is the overall imports of the messages received by $M$ minus the overall imports of the messages sent by $M$.

3. Import acts effectively as *computation tokens* where ITI’s send and receive tokens.

4. An ITM $M$ is **locally $T$-bounded** if at any point, the overall number of computational steps taken by $M$ so far is at most $T(n)$ where $n$ is the number of tokens it holds at such point.
1. The initialization message contains a runtime budget that depends on the security parameter and the input. Every message between ITIs will contain an import field which is a natural number.

2. The import of an ITI $M$ is the overall imports of the messages received by $M$ minus the overall imports of the messages sent by $M$.

3. Import acts effectively as computation tokens where ITI’s send and receive tokens.

4. An ITM $M$ is locally $T$-bounded if at any point, the overall number of computational steps taken by $M$ so far is at most $T(n)$ where $n$ is the number of tokens it holds at such point.

5. We say $\mu$ is $T$-bounded if the ITM $M = (\mu, id)$ is locally $T$-bounded and it only makes calls to locally $T$-bounded ITMs.
### Claim

If the initial ITM in a system $(I, C)$ is $T$-bounded (and $C$ is computable in $T'$), then the system can be simulated by a single TM in time $O(T(n)T'(T(n)))$.

### Corollary

If the initial ITM in a system $(I, C)$ is PPT (as well as $C$) according to this definition, then it is PPT according to the classical definition.
In a system of ITMs, only one ITM is active at any point in time.

The system is expressive enough to describe concurrent operations.
Communication between ITMs is handled by the control function $C$, this is not the same as communication between parties.

Communication is modelled through ideal functionalities.

Different modes of communication (synchronous, asynchronous) and different security notions (noisy, secure).
Different corruption models can coexist in the same protocol
- Parties or functionalities might be corrupted
- Assumptions are encoded in the functionalities themselves
A **configuration** of an ITM $\mu$ consists of the contents of all tapes, the current state and the location of the head.

An **activation** of a ITI $M = (\mu, id)$ is a sequence of configurations that correspond to a computation of $\mu$ starting from some active configuration of $M$. 

**Execution**

Definition

- A **configuration** of an ITM $\mu$ consists of the contents of all tapes, the current state and the location of the head.
- An **activation** of a ITI $M = (\mu, id)$ is a sequence of configurations that correspond to a computation of $\mu$ starting from some active configuration of $M$. 

### Executions

#### Definition

- A **configuration** of an ITM $\mu$ consists of the contents of all tapes, the current state and the location of the head.

- An **activation** of a ITI $M = (\mu, id)$ is a sequence of configurations that correspond to a computation of $\mu$ starting from some active configuration of $M$.

#### Definition

An **execution** of a system $S = (I, C)$ on input $z$ consists of a sequence of activations of ITIs. The **output** of an execution is the contents of first cell in the outgoing message tape of the initial ITI when it halts.
The Execution of a Protocol

We want to talk about the execution of a protocol $\pi$ given an environment $\mathcal{E}$ and an adversary $\mathcal{A}$. We represent this by the system $\left(\mathcal{E}, C_{\text{exec}}^{\pi,\mathcal{A}}\right)$.

The Execution of $\pi$

Let $\text{exec}_{\pi,\mathcal{A},\mathcal{E}}(z, k)$ denote the random variable (over local random tapes) describing the output of execution $\left(\mathcal{E}, C_{\text{exec}}^{\pi,\mathcal{A}}\right)$ on input $z$ and security parameter $k$.

Let $\text{exec}_{\pi,\mathcal{A},\mathcal{E}}$ denote the ensemble $\left\{\text{exec}_{\pi,\mathcal{A},\mathcal{E}}(z, k)\right\}_{z \in \{0,1\}^*, k \in \mathbb{N}}$. 
**Indistinguishability**

**Definition**

Two binary probability distribution ensembles $X$ and $Y$ are indistinguishable (*written* $X \approx Y$) if for any polynomial $p$, all $k$ big enough, and all $z \in \{0, 1\}^{p(k)}$, we have:

$$|\Pr[X(k, z) = 1] - \Pr[Y(k, z) = 1]| < \text{negl}(k).$$
Security
Definition

Let $\pi$ and $\phi$ be subroutine-respecting PPT protocols. We say that $\pi$ UC-emulates $\phi$ if for any PPT adversary $A$ there exists a PPT adversary $S$ such that for any balanced PPT environment $E$ we have:

$$\text{exec}_{\phi,S,E} \approx \text{exec}_{\pi,A,E}. \quad (1)$$
Definition

Let $\pi$ and $\phi$ be subroutine-respecting PPT protocols. We say that $\pi$ UC-emulates $\phi$ if for any PPT adversary $A$ there exists a PPT adversary $S$ such that for any balanced PPT environment $E$ we have:

$$\text{exec}_{\phi,S,E} \approx \text{exec}_{\pi,A,E}.$$  \hfill (1)
**UC-emulation**

**Definition**

Let \( \pi \) and \( \phi \) be **subroutine-respecting** PPT protocols. We say that \( \pi \) **UC-emulates** \( \phi \) if for any PPT adversary \( A \) there exists a PPT adversary \( S \) such that for any balanced PPT environment \( E \) we have:

\[
\text{exec}_{\phi,S,E} \approx \text{exec}_{\pi,A,E}. 
\]  

(1)

**Subroutine-respecting** All subsidiaries of the protocol do not send or accept messages from outside their session.
**Definition**

Let $\pi$ and $\phi$ be subroutine-respecting PPT protocols. We say that $\pi$ **UC-emulates** $\phi$ if for any PPT adversary $A$ there exists a PPT adversary $S$ such that for any balanced PPT environment $E$ we have:

$$\text{exec}_{\phi,S,E} \approx \text{exec}_{\pi,A,E}.$$  \hspace{1cm} (1)

**Subroutine-respecting** All subsidiaries of the protocol do not send or accept messages from outside their session.

**Balanced $E$** If at any point in time $E$ provided import $n_1, \ldots, n_k$ to $k$ ITMs overall, then the overall import to the adversary is at least $n_1 + \cdots + n_k$. 


£: Side effects, $: API
What does this have to do with security?
What does this have to do with security?

- We define ideal functionalities that have the properties and API behaviour that we expect from the protocol
- The side effects of the functionality do not need to be realistic
Let $\mathcal{F}$ be an ideal functionality. Let $\pi$ and $\text{ideal}_{\mathcal{F}}$ be subroutine-respecting PPT protocols. We say that $\pi$ realizes $\mathcal{F}$ iff $\pi$ UC-emulates $\text{ideal}_{\mathcal{F}}$. 

\begin{center}
\begin{tikzpicture}
  \node (F) {$\mathcal{F}$};
  \node (DF1) [left of=F] {$D^1_{\mathcal{F}}$};
  \node (DF2) [below of=F] {$D^2_{\mathcal{F}}$};
  \node (DF3) [right of=F] {$D^3_{\mathcal{F}}$};
  \node (IF) [below of=DF1, yshift=-1cm] {$\text{ideal}_{\mathcal{F}}$};

  \draw [->] (F) -- (DF1);
  \draw [->] (F) -- (DF2);
  \draw [->] (F) -- (DF3);
  \draw [->] (DF1) -- (IF);
  \draw [->] (DF2) -- (IF);
  \draw [->] (DF3) -- (IF);
\end{tikzpicture}
\end{center}
Let $\pi$ and $\phi$ be subroutine-respecting PPT protocols. We say that $\pi$ UC-emulates $\phi$ with respect to the dummy adversary if there exists a PPT adversary $S$ such that for any balanced PPT environment $E$ we have:

$$\text{exec}_{\phi,S,E} \approx \text{exec}_{\pi,D,E}. \tag{2}$$

Dummy adversary just relays the messages it gets.
Definition

Let $\pi$ and $\phi$ be subroutine-respecting PPT protocols. We say that $\pi$ UC-emulates $\phi$ with respect to the dummy adversary if there exists a PPT adversary $S$ such that for any balanced PPT environment $E$ we have:

$$\text{exec}_{\phi,S,E} \approx \text{exec}_{\pi,D,E}.$$ (2)

Dummy adversary just relays the messages it gets.

Claim

Let $\pi$ and $\phi$ be subroutine-respecting PPT protocols. $\pi$ UC-emulates $\phi$ if and only if $\pi$ UC-emulates $\phi$ with respect to the dummy adversary.
Alternative definition, with ideal \( \mathcal{F} \) instead of \( \phi \)

\( \leftrightarrow \): Side effects, \( \uparrow \): API
Composability
Universal Composition Theorem

Let $\rho$, $\pi$ be PPT protocols, $\mathcal{F}$ be a PPT ideal functionality, and $\xi$ be a PPT predicate, such that $\rho$ is $(\pi, \text{ideal}_\mathcal{F}, \xi)$-compliant, $\pi$ is subroutine exposing, and $\pi$ UC-realizes $\mathcal{F}$ with respect to $\xi$-identity-bounded environments. Then $\rho^{\mathcal{F}\rightarrow\pi}$ UC-emulates protocol $\rho$. 

Subroutine exposing

All subsidiaries of the protocol are known

Calls only identities satisfying $(\pi; \text{ideal}_\mathcal{F}; \xi)$-compliant

Doesn’t call ideal $\mathcal{F}$ and $\xi$ in a single session and identities of subroutines satisfy


Universal Composition Theorem

Let \( \rho, \pi \) be PPT protocols, \( \mathcal{F} \) be a PPT ideal functionality, and \( \xi \) be a PPT predicate, such that \( \rho \) is \((\pi, \text{ideal}_{\mathcal{F}}, \xi)\)-compliant, \( \pi \) is subroutine exposing, and \( \pi \) UC-realizes \( \mathcal{F} \) with respect to \( \xi \)-identity-bounded environments. Then \( \rho^{\mathcal{F} \rightarrow \pi} \) UC-emulates protocol \( \rho \).

Subroutine exposing All subsidiaries of the protocol are known
Universal Composition Theorem

Let $\rho$, $\pi$ be PPT protocols, $\mathcal{F}$ be a PPT ideal functionality, and $\xi$ be a PPT predicate, such that $\rho$ is $(\pi, \text{ideal}_F, \xi)$-compliant, $\pi$ is subroutine exposing, and $\pi$ UC-realizes $\mathcal{F}$ \textbf{with respect to} $\xi$-identity-bounded environments. Then $\rho^{\mathcal{F} \rightarrow \pi}$ UC-emulates protocol $\rho$.

**Subroutine exposing** All subsidiaries of the protocol are known

**$\xi$-identity-bounded environments** Calls only identities satisfying $\xi$
Universal Composition Theorem

Let $\rho$, $\pi$ be PPT protocols, $\mathcal{F}$ be a PPT ideal functionality, and $\xi$ be a PPT predicate, such that $\rho$ is $(\pi, \text{ideal}_F, \xi)$-compliant, $\pi$ is subroutine exposing, and $\pi$ UC-realizes $\mathcal{F}$ with respect to $\xi$-identity-bounded environments. Then $\rho^{\mathcal{F} \rightarrow \pi}$ UC-emulates protocol $\rho$.

**Subroutine exposing** All subsidiaries of the protocol are known

**$\xi$-identity-bounded environments** Calls only identities satisfying $\xi$

**$(\pi, \text{ideal}_F, \xi)$-compliant** Doesn’t call ideal$_F$ and $\pi$ in a single session and identities of subroutines satisfy $\xi$
\( (\pi, \text{ideal}_F, \xi) \)-COMPLIANCE

- \( \rho \) is \((\pi, \text{ideal}_F, \xi)\)-compliant, doesn’t call \text{ideal}_F and \( \pi \) in a single session and identities of subroutines satisfy \( \xi \)
Proof

Outline:

1. Assume that
   \[ \exists S_\pi \forall E_\pi \quad \text{exec}_{\pi,D,E_\pi} \approx \text{exec}_{\text{ideal}_F,S_\pi,E_\pi} \]

2. Construct \( S \) out of \( S_\pi \) that would work for protocol \( \rho \)

3. Prove that \( S \) works, namely that
   \[ \forall E \quad \text{exec}_{\rho,F\rightarrow\pi,D,E} \approx \text{exec}_{\rho,S,E} \]

4. To do that show that
   \[ \exists E \quad \text{exec}_{\rho,F\rightarrow\pi,D,E} \not\equiv \text{exec}_{\rho,S,E} \Rightarrow \exists E_\pi \quad \text{exec}_{\pi,D,E_\pi} \not\equiv \text{exec}_{\text{ideal}_F,S_\pi,E_\pi} \]

5. by constructing \( E_\pi \) out of such \( E \)
We want $\forall \mathcal{E} \; \text{exec}_{\rho \mathcal{F} \to \pi, \mathcal{D}, \mathcal{E}} \approx \text{exec}_{\rho, \mathcal{S}, \mathcal{E}}$. 
Environment $\mathcal{E}_\pi$

\[ S \text{ or } D \rightarrow \text{ideal}_\mathcal{F} \text{ or } \pi \]

\[ \mathcal{E} \]

\[ \rho \]

\[ \text{ideal}_\mathcal{F} \]

\[ \text{ideal}^{\text{sid}_1}_\mathcal{F} \]

\[ \text{ideal}^{\text{sid}_{\ell-1}}_\mathcal{F} \]

\[ \text{ideal}^{\text{sid}_\ell}_\mathcal{F} \]

\[ \mathcal{E}_\pi \]

\[ \pi^{\text{sid}_m} \]

\[ \pi^{\text{sid}_{\ell+1}} \]
BUILDING A PROTOCOL
Ideal Functionality $\mathcal{F}_{\text{SMT}}^l$

Parametrized by leakage function $l : \{0, 1\}^* \rightarrow \{0, 1\}^*$

**Communication** On input $(\text{Send}, \text{sid}, R, m)$ from party $S$, do:
- If this is the first $(\text{Send} \ldots)$ input, then record $R, m$ and send $(\text{Sent}, \text{sid}, S, R, l(m))$ to $A$
- Else: do nothing.

When receiving $(\text{ok})$ from $A$ for the first time, output $(\text{Sent}, \text{sid}, S, R, m)$ to $R$.

**Corruption** On input $(\text{Corrupt}, \text{sid}, P, m', R')$ from $A$, where $P \in \{S, R\}$, disclose $m$ to $A$.
Next if $P = S$ and have not yet received $(\text{ok})$ from $A$, then record $R', m'$ instead of $R, m$. 
Communication between parties is done only via the ideal functionality.

The adversary decides to corrupt parties and only then she gets information about the messages, beyond the leaked information.

Corruption of an instance of $\mathcal{F}_{\text{SMT}}$ is "one-time", the parties are in general not corrupted beyond what is described.
Ideal functionality $F_{\text{SMT}}$ can be realized using a CPA-secure encryption scheme and the ideal functionality for authentication of messages $F_{\text{AUTH}}$.

Which in turn can be realized if the computational model is expanded to include *pre-shared keys*, a trusted *bulletin board*, or a *public ledger* holding public keys of the parties.
NOW YOU SAW IT!

THANK YOU
Ideal Functionality for Digital Signatures $\mathcal{F}_{\text{SIG}}$

$\mathcal{F}_{\text{SIG}}$

**Key Generation**  Upon receiving $(\text{KeyGen}, \text{sid})$ from some party $P$, hand $(\text{KeyGen}, \text{sid})$ to $A$. Upon receiving $(\text{VerificationKey}, \text{sid}, v)$ from $A$, output it to $P$, and record the pair $(P, v)$.

**Signature Generation**  Upon receiving $(\text{Sign}, \text{sid}, m)$ from $P$, send $(\text{Sign}, \text{sid}, m)$ to $A$. Upon receiving $(\text{Signature}, \text{sid}, m, \sigma)$ from $A$, verify that no entry $(m, \sigma, v, 0)$ is recorded. If it is, then halt. Else, output $(\text{Signature}, \text{sid}, m, \sigma)$ to $P$, and record the entry $(m, \sigma, v, 1)$. 
**Signature Verification**  Upon receiving \((Verify, sid, m, \sigma, v')\) from some party \(P\), hand it to \(A\). Upon receiving \((Verified, sid, m, \varphi)\) from \(A\) do:

- If \(v = v\) and the entry \((m, \sigma, v, 1)\) is recorded, then set \(f = 1\). *(Completeness)*
- Else, if \(v = v\), the signer is not corrupted, and no entry \((m, \sigma', v, 1)\) for any \(\sigma'\) is recorded, then set \(f = 0\) and record the entry \((m, \sigma, v, 0)\). *(Unforgeability)*
- Else, if there is an entry \((m, \sigma, v', f')\) recorded, then let \(f = f'\). *(Consistency)*
- Else, let \(f = \varphi\) and record the entry \((m, \sigma, v', \varphi)\).

Output \((Verified, sid, m, f)\) to \(P\)