1 One Time Secure Signature Scheme

**Model** The scheme consists of the three efficient algorithms (*Setup, Sign, Verify*). *Setup* creates verification and signing keys. *Verify* returns 1 if the signature verifies. The scheme is named one time secure as the attacker is only allowed one query before they must produce a valid message and signature pair. The verification and signing keys are created with the message, the recipient does not know the verification key until they receive it with the message.

\[(Vk, Sk) \leftarrow \text{Setup}(1^k)\]
\[\alpha \leftarrow \text{Sign}(Sk, m)\]
\[\{1, 0\} \leftarrow \text{Verify}(Vk, m, \alpha)\]

**Correctness** With honest *Setup* and *Sign*, \[Pr[\text{Verify}(Vk, m, \text{Sign}(Sk, m)) = 1] = 1.\]

**Security** We base security on the following experiment between a challenger and a PPT adversary.

<table>
<thead>
<tr>
<th>Challenger</th>
<th>Adversary</th>
</tr>
</thead>
<tbody>
<tr>
<td>((Vk, Sk) = \text{Setup}(1^k))</td>
<td>Vk</td>
</tr>
<tr>
<td>(\alpha = \text{Sign}(m))</td>
<td>(m)</td>
</tr>
<tr>
<td>(\alpha \rightarrow m', \alpha')</td>
<td>Output 1 if (\text{Verify}(Vk, m', \alpha') = 1) and ((m, \alpha) \neq (m', \alpha'))</td>
</tr>
</tbody>
</table>

0 otherwise

We require that an adversary can not forge a signature for a different message. \[Pr[\text{Output} = 1] = \text{neg}(k).\]

2 CCA2

We now use the one-time signature system \((\text{Setup}_s, \text{Sign}_s, \text{Verify}_s)\), a CCA1 secure scheme \((\text{Gen}_1, \text{Enc}_1, \text{Dec}_1)\), and an adaptively secure NIZK proof system \((P, V)\) to create a CCA2 secure scheme \((\text{Setup}, \text{Enc}, \text{Dec})\) using the Dolev-Dwork-Naor (DDN) scheme [?].
2.1 Model

Setup  \(\text{Setup}(1^k)\) creates \(2k\) public keys, random bits \(\sigma\), and \(2k\) private keys.

\[
(Pk, Sk) \leftarrow \text{Setup}(1^k) \quad \text{where}
\]

\[
(Pk_{i,b}, Sk_{i,b}) \leftarrow \text{Gen}_1(1^k) \quad \forall i \in \mathbb{Z}, b \in \{0, 1\}
\]

\[
\sigma \leftarrow \{0, 1\}^{\text{poly}(k)}
\]

\[
P_k = \left( \begin{array}{cccc}
P_{k_1,0} & P_{k_2,0} & P_{k_3,0} & \cdots & P_{k_k,0} \\
P_{k_1,1} & P_{k_2,1} & P_{k_3,1} & \cdots & P_{k_k,1} \end{array} \right), \quad Sk = \left[ \begin{array}{c}
Sk_{1,0} \\
Sk_{1,1} \end{array} \right].
\]

Enc  \(\text{Enc}(Pk, m)\) uses our one-time signature and CCA1 schemes to encrypt the message \(k\) times.

1. Generate verification and signing keys

\[
(Vk, Sk) \leftarrow \text{Setup}_4(1^k).
\]

2. Select one key from each of the \(k\) columns of \(P_k\) based on the bits of \(V_k\)

\[
V_k = V_{k_1}|V_{k_2}|V_{k_3}|\cdots|V_{k_k}
\]

\[
\left[ P_{k_{1,V_k_1}} \quad P_{k_{2,V_k_2}} \quad P_{k_{3,V_k_3}} \quad \cdots \quad P_{k_{k,V_k_k}} \right].
\]

3. Use the \(k\) keys and a witness for each key to create \(k\) CCA1 encryptions of \(m\)

\[
w_i \leftarrow \{0, 1\}^* \\
C_i \leftarrow \text{Enc}_1(P_{k_{i,V_k_i}}, m; w_i).
\]

4. Sign the collection of \(k\) encryptions \(C_i\) along with a proof, \(\pi\), that each \(C_i\) encrypts the same message under the corresponding \(P_{k_{i,V_k_i}}\) public key

\[
\pi \leftarrow \mathcal{P}(\sigma, C_1, \cdots, C_k, m, w) \\
\alpha \leftarrow \text{Sign}_a(Sk, C_1, \cdots, C_k, \pi).
\]

5. Output

\[
C = (V_k, C_1, \cdots, C_k, \pi, \alpha).
\]

Dec  If the signature and proof verify, decrypt \(C_1\) using \(Sk_{1,V_k_1}\)

\[
\text{Dec}(V_k, C_1, \cdots, C_k, \pi, \alpha) = \bot \text{ if } \text{Verify}_s(V_k, (C_1, \cdots, C_k, \pi), \alpha) = 0, \text{ else} \\
\text{Dec}(Sk_{1,V_k_1}, C_1).
\]
Table 1: CCA2 Indistinguishability Experiment

<table>
<thead>
<tr>
<th>Phase</th>
<th>Challenger</th>
<th>Adversary $A^{Dec_{Sk}(), Enc_{Pk}()}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setup</td>
<td>$(Pk, Sk) \leftarrow \text{Setup}(1^k)$</td>
<td>$P_k \leftarrow$</td>
</tr>
<tr>
<td>Oracle Phase 1</td>
<td>$C_i \leftarrow i \in [2, q]$</td>
<td>$m_i = \text{Dec}(Sk, C_i)$</td>
</tr>
<tr>
<td>Challenge</td>
<td>$m_0, m_1 \leftarrow A^{Dec_{Sk}()}(P_k)$</td>
<td>$b \leftarrow {0, 1}$</td>
</tr>
<tr>
<td>Oracle Phase 2</td>
<td>$C_i \leftarrow i \in [q+1, n]$</td>
<td>$m_i = \text{Dec}(Sk, C_i)$</td>
</tr>
<tr>
<td>Response</td>
<td>$b' \leftarrow A^{Dec_{Sk}()}(P_k, C^*)$</td>
<td>Output 1 if $b = b'$, 0 otherwise.</td>
</tr>
</tbody>
</table>

2.2 Security

The proof is similar to that for CCA1. We start with encrypting $m_0$, simulate the proof, change all the $C^*_i$ from $m_0$ to $m_1$, then return to the honest proof. All while still answering before and after decryption oracle queries. We use the experiment in Table 1. For the proof we define two events that might occur.

**Forge** Event Forge concerns the signature and is defined as the adversary submitting a decryption query $d_i$ that uses our $V_k$ and $\text{Verify}(d_i) = 1$, but is different from the cipher text we sent.

$$\exists i \in [q'] : d_i = (V_k, C_1, \ldots, C_k, \pi, \alpha) \neq C^*$$

were $V_k = V_k'$ and $\text{Verify}(d_i) = 1$.

**Fake** Event Fake concerns the proof and is defined as the adversary submitting a decryption query $d_i : \mathcal{V}(d_i) = 1$ but two of its cipher texts decrypt to different messages.

$$\exists i \in [q'] : d_i = (V_k, C_1, \ldots, C_k, \pi, \alpha)$$

where $\mathcal{V} = \{0, 1\}$ but $\exists j, l \in [k] : \text{Dec}(Sk_{V_k, i}, C_j) \neq \text{Dec}(Sk_{V_k, i}, C_l)$

We want to move from $C^* = (V_k^*, C_1^*, \ldots, C_k^*, \pi^*, \alpha^*)$.
encrypting $m_0$ to encrypting $m_1$ in a manner undetectable to the adversary while continuing to be able to answer our adversaries decryption queries, $C_i$. We examine a series of experiments with the first encrypting $m_0$ to the last encrypting $m_1$ showing that each successive pair is indistinguishable.

**H0**  H0 is the normal experiment above that encrypts $m_0$.

$Pr[Forge]$ is negligible in H0. If it were not, we could use our adversary, $A$, to break the security guarantee of a challenger, $C_{1-time}$, for the one-time secure signature scheme. We construct a signature scheme adversary, $A'$, that receives $Vk$ from $C_{1-time}$ and passes it to $A$ who returns a forgery attempt, $C_i$. As $Pr[Forge]$ is non-negligible, by passing $C_i$ to $C_{1-time}$ we have a non-negligible chance of verification, thus breaking the security guarantee of $C_{1-time}$.

$Pr[Fake]$ is negligible in H0. If it were not, we could with non-negligible probability cause the NIZK $Verify$ to falsely accept a cypher text, $C_i$, with proof that all $C_{j,i}$ encrypt the same message, when two of them did not.

**H1**  H1 replaces $\sigma, \mathcal{P}$ from the NIZK proof system with simulations

$$\sigma \leftarrow Sim_1$$
$$\pi \leftarrow Sim_2.$$

If H0, H1 could be distinguished by a CCA2 adversary $A$, we could construct an adversary, $A'$, of a NIZK system challenger $C'$. Sometimes $A'$ creates and sends to $A$ a cipher text $C_i = (Vk, C_{1,i}, \cdots, C_{k,i}, \pi = \mathcal{P}(i, \alpha)$, as in H0. Other times $A'$ creates and sends to $A$ a cipher text $C_i = (Vk_i, C_{1,i}, \cdots, C_{k,i}, \pi = Sim_2, \alpha)$, as in H1. If the distribution of $A'$'s responses $b$ differ between these two situations then $A'$ has learned something extra from the NIZK challenger $C'$ thus breaking the NIZK system.

$Pr[Forge]$ is negligible in H1. If it were not, we could break the signature scheme just as in H0.

$Pr[Fake]$ is negligible in H1. On every query $C_i$ by $A$, we can check that each $C_{j,i}$ decrypts to the same message. If $Pr[Fake]$ increased between H0 and H1, we could use this to distinguish between $\mathcal{P}$ and the simulator. As $\mathcal{P}$ is a NIZK proof system, this is a contradiction, thus there is no detectable change in $Pr[Fake]$ between H0 and H1, and so it is still negligible.

**H1.5**  Samples $Vk^*$ for the encryption challenge in the beginning as part of Setup. Note that the $Vk^*$ we create for $C^*$ is separate from the $Vk$ sampled by the adversary when they create $C_i$ decryption requests. $Vk^*$ is generated randomly so generating it early or just before we need it will not affect the distribution of its values. $Vk^*$ is only used in creating our challenge cipher, so changing from creating it just prior to giving the challenge cipher to the adversary, to creating it as some point earlier has no effect on the interaction with the adversary. Thus the distribution the adversary sees is unchanged.

$Pr[Forge]$ and $Pr[Fake]$ do not change as no change is visible to the adversary. Thus they are still negligible.

**H1.75**  Changes which secret key components we keep so that we know none of the challenge cipher text’s secret key components, while still having the secret key components to answer decryption oracle queries. As we now know $Vk^*$ from the start, we can use the secret key components not used by $C^*$ to decrypt the decryption oracle queries.
Consider the complement of $V k^*$, $\overline{V k}^*$, i.e.

$$V k^* = 011 \cdots 1$$

$$\overline{V k}^* = 100 \cdots 0.$$ 

Highlighting the corresponding parts of our existing public key and an expanded secret key, we have

$$P k = \left( \begin{array}{cccc} P k_{1,0} & P k_{2,0} & P k_{3,0} & \cdots & P k_{k,0} \\ P k_{1,1} & P k_{2,1} & P k_{3,1} & \cdots & P k_{k,1} \end{array} \right), \sigma$$

$$S k = \left( \begin{array}{cccc} S k_{1,0} & S k_{2,0} & S k_{3,0} & \cdots & S k_{k,0} \\ S k_{1,1} & S k_{2,1} & S k_{3,1} & \cdots & S k_{k,1} \end{array} \right).$$

For the challenge cipher, we will use only the $V k^*$ components of the $P k$ and we do not need the $V k^*$ components of $S k$, as we know the value of $b$. If we were to construct and publish the $P k$ as before, and change our private $S k$ to keep only the $\overline{V k}^*$ components we can still provide and grade the challenge cipher. We now have

$$P k = \left( \begin{array}{cccc} P k_{1,0} & P k_{2,0} & P k_{3,0} & \cdots & P k_{k,0} \\ P k_{1,1} & P k_{2,1} & P k_{3,1} & \cdots & P k_{k,1} \end{array} \right), \sigma$$

$$S k = \left( \begin{array}{cccc} S k_{1,0} & S k_{2,0} & S k_{3,0} & \cdots & S k_{k,0} \\ S k_{1,1} & S k_{2,1} & S k_{3,1} & \cdots & S k_{k,1} \end{array} \right).$$

The adversary queries to the decryption oracle will have a $V k$ different from $V k^*$. As $Pr[Forge]$ is negligible, at least one bit of the adversaries $V k$ will have the same value as the corresponding bit in $\overline{V k}^*$. Thus there is always one $\overline{V k}^*$ component of $S k$ that we can use to decrypt the query. As $Pr[Fake]$ is negligible, this provides the same message as in H1.5.

Thus the changes from H1.5 to H1.75 are undetectable to the adversary. With H1.75 we have ensured that when we create and present the challenge cipher we know none of the secret keys necessary to decrypt it.

**H2** Changes the challenge cipher from $m_0$ to $m_1$. As we do not know the secret keys to decrypt $m_0$, we can use semantic security to move from $m_0$ to $m_1$.

**H3** Returns to the real $\sigma, \pi$. As in changing from H0 to H1, if the adversary, $A$, could detect the change, we could use $A$ to break the underlying NIZK proof system.

### 2.3 Use and Efficiency

This is an example of non-malleable cryptography. An example of its use is in preventing one bidder in an auction from slightly increasing an opponent’s bid without knowledge of the opponent’s unencrypted bid. It is not an efficient construction, but an efficient construction does exist in RSA.

### References