SK-ID schemes (both enc-based and MAC-based); Public-key ID schemes; Schnorr’s protocol as an example of a Σ-protocol; Two properties of Σ-protocols: HVZK and PoK; Special Soundness; Proof that special soundness implies PoK.

1 SK-ID schemes (continued from last week)

Last week we started talking about Symmetric-key Identification schemes. A reminder of the setting: *alice* and *bob* share a secret key $s \leftarrow \{0, 1\}^\lambda$. *eve* listens to the insecure channel between *alice* and *bob*: $alice(s) \leftarrow eve \rightarrow bob(s)$. *alice* is trying to prove to *bob* that she is indeed *alice*. This game has two phases:

1. Leakage Phase: This phase may be defined by two kinds of adversaries:
   
   (a) Passive: *eve* can only observe the traffic between *alice* and *bob*.
   
   (b) Active: *eve* can do whatever she wants, including changing messages.

2. Impersonation Phase: In this phase *alice* disappears and *eve* tries to convince *bob* that she is really *alice*.

In both cases we are interested in the probability that *eve* succeeds in tricking *bob* to accept her. For a protocol to be secure against a passive (active) adversary, we require $\Pr(bob\,	ext{accepts }\,eve) \leq \text{negl}(\lambda)$

We have looked at two proposals: encryption based (enc-based) and MAC-based.

Enc-based scheme:

- **alice**
  - $m \leftarrow \$ $c \leftarrow E_s(m)$
  - $\bar{m} = D_s(c)$
  - $\bar{m} \rightarrow accept$ iff $m = \bar{m}$

**Lemma 1** For an enc-based identification scheme:

L2-1
• If the encryption scheme is **One-way CPA secure (OW-CPA)**, we get passive security.
• If the encryption scheme is **OW-CCA1**, we get active security.

**Definition 2** OW-CPA security is defined by the following game:

<table>
<thead>
<tr>
<th>Adversary</th>
<th>Challenger</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A^{E_s(\cdot)} \rightarrow \text{state}$</td>
<td>$m \leftarrow $; c \leftarrow E_s(m)$</td>
</tr>
<tr>
<td>$\leftarrow c$</td>
<td>$\leftarrow c$</td>
</tr>
<tr>
<td>$A(c, \text{state}) \rightarrow \tilde{m}$</td>
<td>$\tilde{m} \rightarrow \tilde{m}$</td>
</tr>
</tbody>
</table>

We say that $E$ is OW-CPA if $\Pr\{\tilde{m} = m\} \leq negl(\lambda)$

**Definition 3** OW-CCA security is defined by the following game:

<table>
<thead>
<tr>
<th>Adversary</th>
<th>Challenger</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A^{E_s(\cdot), D_s(\cdot)} \rightarrow \text{state}$</td>
<td>$m \leftarrow $; c \leftarrow E_s(m)$</td>
</tr>
<tr>
<td>$\leftarrow c$</td>
<td>$\leftarrow c$</td>
</tr>
<tr>
<td>$A(c, \text{state}) \rightarrow \tilde{m}$</td>
<td>$\tilde{m} \rightarrow \tilde{m}$</td>
</tr>
</tbody>
</table>

We say that $E$ is OW-CCA if $\Pr\{\tilde{m} = m\} \leq negl(\lambda)$

**Examples:**

- CBC, CFB, etc are all OW-CPA.
- If $f_s$ is a strong PRP (pseudo random permutation), then $c = f_s(m)$ is OW-CCA1.
- $E_s(m) = (r, f_s(r) \oplus m)$, where $r$ is random and $f$ is a PRF, is OW-CCA.
MAC-based scheme:

\[\begin{align*}
\text{alice} & \quad \text{bob} \\
\sigma &= MAC_s(r) \\
\end{align*}\]

\[\begin{align*}
r & \quad m \leftarrow \$ \\
\sigma & \quad \text{accept iff } \sigma = MAC_s(r) \\
\end{align*}\]

**Lemma 4** If MAC is universally unforgeable under chosen message attack (UUF-CMA), then we get active security.

**Definition 5** The UUF-CMA game:

\[\begin{align*}
\text{adversary} & \quad \text{challenger} \\
\text{Stage 1:} & \quad A^{MAX(\cdot)},D_{\mathcal{S}}(\cdot) \rightarrow \text{state} \\
\text{Stage 2:} & \quad m \leftarrow \$ \\
A(m, \text{state}) & \rightarrow \sigma \\
\sigma & \quad \text{accept iff } \sigma = MAC_s(m) \\
\end{align*}\]

For a MAC to be UUF-CMA secure we require \(Pr\{\sigma = MAC_s(m)\} \leq \text{negl}\)

Examples of MAC-based schemes:

- PRF for an existentially unforgeable function (EUF-CMA.)
- PRP.

**Question:** what about leakage resilient symmetric-key ID (SK-ID) schemes

**Answer:** we don’t know. Known solutions use public-key cryptography: LR-PK-ID (Leakage Resilient Public Key ID) schemes.

## 2 Public-key ID schemes

\[\begin{align*}
\text{alice} & \quad \text{generates } (Pk, Sk) \text{ and gives } Pk \text{ to bob and eve.} \\
\end{align*}\]
Passive and active definitions are the same as for SK-ID schemes. Note: during the impersonation stage eve does not need bob (she knows all that he knows.) For enc/MAC-based schemes we only need to replace SK-enc by PK-enc and MAC by Sign as is shown below:

Enc-based scheme (PKC):

\[
\begin{align*}
alice & \quad \quad bob \\
\multicolumn{2}{c}{m \leftarrow \$} \\
\multicolumn{2}{c}{c \leftarrow E_{P_k}(m)} \\
\multicolumn{2}{c}{\tilde{m} = D_{S_k}(c)} \\
\multicolumn{2}{c}{\tilde{m}} \\
\multicolumn{2}{c}{\text{accept iff } m = \tilde{m}}
\end{align*}
\]

MAC-based scheme (PKC):

\[
\begin{align*}
alice & \quad \quad bob \\
\multicolumn{2}{c}{r \leftarrow \$} \\
\multicolumn{2}{c}{\sigma = \text{Sign}_{S_k}(r)} \\
\multicolumn{2}{c}{\sigma} \\
\multicolumn{2}{c}{\text{accept iff } \text{Verify}_{P_k}(\sigma, r)}
\end{align*}
\]

With these small changes all the previous results about security still hold.

**Question:** what about Trap-door permutations (TDP) \( c = f(m) \) as in the following scheme?

\[
\begin{align*}
alice(tk) & \quad \quad bob(Pk) \\
\multicolumn{2}{c}{m \leftarrow \$} \\
\multicolumn{2}{c}{c = f_{P_k}(m)} \\
\multicolumn{2}{c}{\tilde{m} = f^{-1}_{tk}(c)} \\
\multicolumn{2}{c}{\tilde{m}} \\
\multicolumn{2}{c}{\text{accept iff } f_{P_k}(\tilde{m}) = c}
\end{align*}
\]

Is this secure?

**Answer:** passively, yes. Actively? This is left as a question for home work (hint: \( X^e \mod n \)).
3 \( \Sigma \)-Protocols

We will start with a specific protocol and generalize it to a useful tool:

**Schnorr’s Protocol:** \( Pk = g^x \mod p, Sk = x \) where \( g \) is a generator of a subgroup \( G \) of \( \mathbb{Z}_p^* \). *alice* wants to prove that she knows \( x \) (note that there are no trapdoors here.)

<table>
<thead>
<tr>
<th>*alice((Sk))</th>
<th>*bob((Pk))</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Sk = x )</td>
<td>( Pk = y = g^x \mod p )</td>
</tr>
<tr>
<td>( r \leftarrow \mathbb{Z}_q )</td>
<td>( a = g^r \mod p )</td>
</tr>
<tr>
<td>( a )</td>
<td>( c \leftarrow \mathbb{Z}_q )</td>
</tr>
<tr>
<td>( c )</td>
<td>( z = r + cx )</td>
</tr>
<tr>
<td>( z )</td>
<td>accept iff ( g^z = a \cdot y^c )</td>
</tr>
</tbody>
</table>

Why is this test true:
\[
g^z = g^{r+cx} = (g^r)(g^x)^c = ay^c
\]

Properties:

- **First property:**

**Definition 6** HVZK (Honest Verifier Zero Knowledge.)
There exists an efficient simulator (Sim) which can generate \((a, c, z)\) distributed the same as the corresponding triplets in the real bob - alice interaction.

Example: Schnorr’s algorithm has HVZK. This is what *Sim* does:

\[\text{Sim}(y) := \]

\[
z \leftarrow G
\]

\[
c \leftarrow \mathbb{Z}_q
\]

\[
a = g^z/y^c
\]

\( z \) is generated at random so we get the same distribution as the real \( z \) which is also a random element of \( G \). What remains is to solve for \( a = g^z/y^c \).

**Claim 7** The simulated \((a, c, z)\) distribution has the same distribution as the variables in the real interaction.
Intuitively, this says that in the passive attack eve is not learning anything new from her observation.

**Definition 8** Special HVZK

There exists Sim* s.t. \( \forall c, Sim^*(c) \rightarrow (a, z) \) s.t. \( (a, z) \) is distributed just like the \( (a, z) \) pair from the real interaction conditioning on bob using the challenge \( c \).

- **Second property:**

**Definition 9** Proof of Knowledge (PoK): (note: alice is a Prover and bob is a Verifier.) If some \( P^* \) passes honest test from \( V \) with \( \Pr \geq \epsilon \), then \( \exists Ext \) s.t. \( Ext^{P*}(P_k) \rightarrow x \) with \( \Pr \geq \text{poly}(\epsilon) \). (\( P = \text{Prover}, V = \text{Verifier}, Ext = \text{Extractor.} \))

We will take a detour and define **Special Soundness**: \( \forall \) two accepting conversations \( (a, c, z) \) and \( (a, c', z') \) where \( c \neq c' \), one can extract \( x \). Formally: \( \exists Ext^*(a, c, z, c', z', P_k) \rightarrow Sk \)

Using Schnorr’s protocol as an example, we know:

\[
g^z = ay^c, \quad g^{z'} = ay^{c'}
\]

divide:

\[
g^{z-z'} = y^{c-c'}
\]

now,

\[
-q < c - c' < q, \quad c - c' \neq 0 \Rightarrow \\
\exists (c - c')^{-1} \mod q \Rightarrow \\
g^{(z-z')(c-c')^{-1}} \mod q = y \Rightarrow \\
x = (z - z') (c - c')^{-1} \mod q
\]

note: HVZK plus PoK \( \Rightarrow \) active security (assuming DL is a hard problem.)

**Claim 10** Special Soundness \( \Rightarrow \) PoK

**Proof:** Assume \( \exists P^* \) s.t. \( \Pr\{P^* \leftarrow V\} \geq \epsilon \). Construct \( Ext(y) \) computing \( x \) using \( P^* \):

1. Choose coins \( s \) for \( P^* \)
2. Choose \( c, c' \leftarrow \mathbb{Z}_q^* \)
3. Run \( P^* (s) \) twice with challenges \( c \) and \( c' \)
4. If both runs succeed, use \((a, c, z)\) and \((a, c', z')\) and run \(Ext^*\) (from special soundness) to get \(x\).

What about the case that \(P^*\) does not succeed twice. Are the runs independent? Let 
\[
\hat{\epsilon} = \hat{\epsilon}_{g,y} = \Pr\{P^* \text{ succeeds on } g,y\}.
\]
We have \(\epsilon = E[\hat{\epsilon}_{g,y}]\). Now, fix \(g\) and \(y\). Let \(\hat{\epsilon} = \Pr\{P^*(s) \text{ succeeds }\}\). Consider the following table where 1’s stand for successes and 0’s for failures:

<table>
<thead>
<tr>
<th>(s_1)</th>
<th>(\ldots)</th>
<th>(c_j)</th>
<th>(\ldots)</th>
<th>(c_q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(s_1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\ldots)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(s_i)</td>
<td>0</td>
<td>(\ldots)</td>
<td>1</td>
<td>(\ldots)</td>
</tr>
<tr>
<td>(\ldots)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(s_n)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We know \(\hat{\epsilon} = \text{fraction of 1’s in the table}\). We choose a row \(S_i\) and two columns and hope we chose two 1’s.

We have:

\[
\bar{\epsilon}_s = \text{fraction of 1’s in row } S_i, \hat{\epsilon} = E_s[\hat{\epsilon}_s]
\]

\[
\Pr\{Ext \text{ succeeds on } g,y\} = \sum_s \frac{1}{N} \bar{\epsilon}_s \left( \bar{\epsilon}_s - \frac{1}{q} \right) \overset{\text{Jensen}}{\geq} E_s[\bar{\epsilon}_s^2] - \frac{1}{q} E_s[\bar{\epsilon}] \geq (E_s[\bar{\epsilon}]^2) - \frac{\hat{\epsilon}}{q} \overset{\text{Jensen}}{\geq} \bar{\epsilon}^2 - \frac{\hat{\epsilon}}{q} \Rightarrow \Pr\{Ext \text{ succeeds }\} \geq E_{g,y} \left[ \bar{\epsilon}^2 - \frac{\hat{\epsilon}}{q} \right] \overset{\text{Jensen}}{\geq} \epsilon^2 - \frac{\epsilon}{q}
\]

The problem with the above method is that to succeed with \(\Pr \geq 1/2\), we need to run \(Ext^{P^*} \approx \frac{1}{\epsilon^2}\) times. We can improve this result to get \(\Pr \geq 1/2\) in \(\approx \frac{1}{\epsilon}\) runs of \(P^*\).

**Proof:** (Improvement to previous result) Construct \(Ext'(y)\) computing \(x\) using \(P^*\). Fix \(g, y\) (\(\hat{\epsilon}\) as before) and do:

1. Choose \(s\) and \(c\) until \(P^*(s)\) on challenge \(c\) succeeds.
2. For the fixed \(s\), sample two successful \((a, c, z)\) and \((a, c', z')\) transcripts.

3. If \(c \neq c'\), use the extractor (from special soundness) to solve for \(x\).

We get:

\[
\begin{align*}
E[\text{time of } Ext'] & \overset{\text{Linearity of Expectation}}{=} E[\text{time of stage 1}] + E[\text{time of stage 2}] = \\
&= \frac{1}{\epsilon} + \sum_s 2 \cdot \frac{1}{\epsilon_s} \Pr\{s \text{ was selected}\} = \\
&= \frac{1}{\epsilon} + \sum_s 2 \cdot \frac{1}{\epsilon_s} \sum_{s'} \epsilon_{s'} = \\
&= \frac{1}{\epsilon} + 2 \cdot \left( \frac{N}{\sum_{s'} \epsilon_{s'}} \right) = \\
&= \frac{1}{\epsilon} + 2 \cdot \left( \frac{1}{N} \right) = \\
&= \frac{1}{\epsilon} + 2 \cdot \frac{1}{E[\epsilon_s]} = \\
&= \frac{1}{\epsilon} + \frac{2}{\epsilon} = \frac{3}{\epsilon} = O\left(\frac{1}{\epsilon}\right)
\end{align*}
\]

We still need to consider the probability of failure at step 3. We will show that \(\Pr\{c = c'\} \leq 1/2\)

\[
\sum_s \frac{1}{\epsilon_s q} \Pr\{\text{select } s\} = \\
\sum_s \frac{1}{\epsilon_s q} \sum_{s'} \epsilon_{s'} = \\
\frac{1}{q} \frac{N}{\sum_{s'} \epsilon_{s'}} = \\
\frac{1}{q \epsilon} \text{ if } \epsilon \geq 2/q \leq 1/2
\]

The results above are derived for fixed \(g, y\). We need to show that this holds for any given \(g, y\). We will use random self reducibility to derive a running time of \(O(1/\epsilon)\) for any \(g, y\).
We define the following algorithm $A'$:
Given $g, y$ as input, set $g' = g^\alpha$ and $y' = y(g')^{\beta}$ for $\alpha, \beta \rightarrow \mathbb{Z}_q$. Now $(g', y')$ is random and also:
\[
y = g^x \Rightarrow y' = y(g')^{\beta} = g^x(g')^{\beta} = (g')^{x - \beta} = (g')^{x'} \Rightarrow x = (x' + \beta)\alpha
\]
So, in step (1), we can set random $g', y', s$ until we get a success and in step (2) use the fixed $g', y'$ from step (1).
Let $T = E[A']$ (expected running time), then
\[
E_{g',y'} \left[ \hat{\epsilon} \cdot O \left( \frac{1}{\hat{\epsilon}} \right) + (1 - \hat{\epsilon}) \cdot T \right] = O(1) + T \cdot (1 - \hat{\epsilon}) \Rightarrow \\
\epsilon \cdot T = O(1) \Rightarrow \\
T = O \left( \frac{1}{\epsilon} \right)
\]
\[\Box\]