Date: 23 October, 2023

Disclaimer: These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor.

Recap Last class we talked about:

Pseudo-random Function: functions that seem indistinguishable to a computationally bounded attacker.

Secret-key Encryption:

\[
\text{KeyGen}(1^n) \rightarrow \text{Key } sk \\
\text{Enc}(sk, m) \rightarrow \text{Ciphertext } c \\
\text{Dec}(sk, c) \rightarrow m
\]

This is also known as "Symmetric-key Encryption" because for both side the key SK is pre-shared (identical).

Two properties for secret-key encryption:

- Correctness:
  \[
  \forall m, \Pr_{sk \leftarrow \text{KeyGen}(1^n), c \leftarrow \text{Enc}(sk, m)}[\text{Dec}(sk, c) = m] = 1
  \]

- Security (Multi-message):
  \[
  \forall (m_0, 1, m_1, 1), \ldots (m_{0,q}, m_{1,q}) \text{ for any polynomial } q:
  \[
  \begin{align*}
  &sk \leftarrow \text{KeyGen}(1^n), \{\text{Enc}(sk, m_{0,1}), \ldots, \text{Enc}(sk, m_{0,q})\} \\
  &sk \leftarrow \text{KeyGen}(1^n), \{\text{Enc}(sk, m_{1,1}), \ldots, \text{Enc}(sk, m_{1,q})\}
  \end{align*}
  \]

  The Multi-message Secure Encryption is also known as "Left-Right Encryption" because the encryptions of the left and right messages should be computationally indistinguishable.

Good Exercise: Suppose we are playing the following game between Adversary and Challenger:

\[
\begin{align*}
\text{Adv} &\rightarrow \text{Challenger, } sk \leftarrow \text{KeyGen}(1^n) \\
\text{Adv} &\left( m_{0,1}, m_{1,1}, \ldots (m_{0,q}, m_{1,q}) \right) \rightarrow \text{Challenger} \\
\text{Adv} &\left\{ \text{Enc}(sk, m_{b,i}) \right\} \rightarrow \text{Challenger, } b \leftarrow \{0, 1\} \\
\text{Adv} &\left(b' \rightarrow \text{Challenger, and we want } \Pr[b' = b] \leq \frac{1}{2} + \text{negl} \right)
\end{align*}
\]

6-1
Note the fact that $Pr[b' = b] \leq \frac{1}{2} + \text{negl}$ is equivalent to the security property above.

**What do we have so far?** Based on what we did in the past few lectures, we have the following transformations:

![Transformation Tree So Far](image)

**Figure 6.1: Our Transformation Tree So Far**

### 6.1 Secret-key Encryption (Multi-message):

**Secret-key Encryption:** (Setup, Eval) be a PRF.

- $KeyGen(1^n)$: $k \leftarrow \text{Setup}(1^n)$, $sk = k$

- $Enc(sk, m)$: $m \in \{0, 1\}^n$, $r \leftarrow \{0, 1\}^n$, $c = (r, \text{Eval}(k, r) \oplus m)$

- $Dec(sk, c)$: $c = (c_1, c_2)$, where $c_1 = r, c_2 = \text{Eval}(k, r) \oplus m$, output $c_2 \oplus \text{Eval}(k, c_1)$

**Proof of Correctness:** We can tell this from the decryption method where it outputs: $c_2 \oplus \text{Eval}(k, c_1)$ then the Eval() term got cancelled out because it’s been XORed by itself and we can obtain the message m. Therefore, as long as we have the pre-shared key, we are able to retrieve the message m, the encryption method is correct.

**Proof of Security:** We will prove this using Hybrid Argument.
• **Left Hybrid (LH):**

\[ sk \leftarrow \text{KeyGen}(1^n) \]

\[ \text{Enc}(sk, m_{0,1}) \ldots \text{Enc}(sk, m_{0,q}) \]

Then:

\[ k \leftarrow \text{Setup}(1^n), \quad r_1 \leftarrow \{0,1\}^n, \quad r_2 \leftarrow \{0,1\}^n, \quad \ldots \]

\[ (r_1, \text{Eval}(k, r_1) \oplus m_{0,1}), \quad (r_2, \text{Eval}(k, r_2) \oplus m_{0,2}), \quad \ldots \]

Here the only primitive is the PRF. It guarantees that its output is computationally indistinguishable from the output of a RF.

• **H1:**

\[ r_1 \leftarrow \{0,1\}^n, \ldots, r_q \leftarrow \{0,1\}^n \]

\[ y_1, \ldots, y_q \text{ sampled conditioned on } y_i = y_j \text{ if } r_i = r_j \]

\[ (r_1, y_1 \oplus m_{0,1}), \ldots, (r_q, y_q \oplus m_{0,q}) \]

Suppose LH and H1 are distinguishable, that is:

\[ \exists D, \text{ s.t. } |\Pr[D(LH) = 1] - \Pr[D(H_1) = 1]| = \mu(n), \text{ which is non-negligible} \]

We can then construct D’ that breaks PRF:

1. D’ randomly samples \( r_1, \ldots, r_q \in \{0,1\}^n \)
2. D’ queries the oracle on \( O(r_1), \ldots, O(r_q) \), denoted as \( s_1, \ldots, s_q \)
3. D’ outputs \( D((r_1, s_1 \oplus m_{0,1}), \ldots, (r_q, s_q \oplus m_{0,q})) \)

Note the probability that D’ distinguishes between the two outputs of \( O(.) \) is the same as the probability that D distinguish between LH and H1. (When D’ uses Eval(), it is the same case as LH, and when it uses \( f(.) \), \( f \in F_n \), it is the same case as H1.)

• **H2:** Suppose \( \exists i, j \text{ s.t. } r_i = r_j \), we abort.

Note that fix some \( i, j \): \( \Pr[r_i = r_j] = \frac{1}{2^n} \), then \( \Pr[\exists i, j \text{ s.t. } r_i = r_j] \leq \frac{q^2}{2^n} \), where \( q \) is a poly(), which indicates that H1 and H2 are computationally indistinguishable.

• **H3:**

\[ r_1 \leftarrow \{0,1\}^n, \ldots, r_q \leftarrow \{0,1\}^n \]

\[ y_1, \ldots, y_q \text{ sampled conditioned on } y_i = y_j \text{ if } r_i = r_j \]

\[ (r_1, y_1 \oplus m_{1,1}), \ldots, (r_q, y_q \oplus m_{1,q}) \]

Note that H3 is identically distributed to H2 since each \( y_1, \ldots, y_q \) are sampled uniformly and independently.

• **H4:** Revert the change made in H2. Via a similar argument, we can show that H3 and H4 are indistinguishable.
- \( H_5 \): Switch to \( \text{Eval}(k,r) \), then we can tell that: \( H_5 \approx_c \text{Right Hybrid} \)

From above, for each step, the consecutive pair of Hybrids are computationally indistinguishable, so at the end we can get Left Hybrid \( \approx_c \) Right Hybrid, which is then a contradiction to our assumption, the proof is done.

### 6.2 Digital Signature

This can be used to check the integrity of the data.

**Motivation/Real-life Example:**

![Real-life Example for Digital Signature](image)

**Functions:**

\[
\text{KeyGen}(1^n) \rightarrow (sk, vk) \\
\text{Sign}(sk, m) \rightarrow \sigma \\
\text{Verify}(vk, (m, \sigma)) \rightarrow \text{accept/reject}
\]

We require the signature scheme to satisfy two properties: namely, correctness and security.

**Correctness:** This requires that \( \text{Verify}(vk, (m, \sigma)) \) will all signatures \( \sigma \) that are properly generated using \( sk \). And the probability that it will accept a correct signature is 1.

**Security:** To prove this, consider the following game:

The challenger generates a pair of keys: \( sk \) and \( vk \), it gives the adversary \( vk \), but keeps the \( sk \) secret. The adversary can now make signing queries, where it sends a message \( m \) to the challenger, and the challenger returns the signature of the message. After \( q \) number of queries, the adversary tries to produce a new valid signature on a new message. The adversary wins the game if it can produce a valid signature on a new
message without access to the sk.

\[
\text{Adv} \xleftarrow{vk} \text{Challenger}, (sk, vk) \leftarrow \text{KeyGen}(1^n)
\]

\[
\text{Adv} \xrightarrow{m_1} \text{Challenger}
\]

\[
\text{Adv} \xleftarrow{\sigma_1} \text{Challenger}, \sigma_1 \leftarrow \text{Sign}(sk, m_1)
\]

\[
\text{Adv} \xrightarrow{m_n} \text{Challenger}
\]

\[
\text{Adv} \xleftarrow{\sigma_q} \text{Challenger}, \sigma_q \leftarrow \text{Sign}(sk, m_q)
\]

\[
\text{Adv} \xrightarrow{(m^*, \sigma^*), m^* \notin \{m_1, \ldots, m_q\}} \text{Challenger}, \text{if Verify}(vk, (m^*, \sigma^*)) = \text{accept}, \text{Adv wins.}
\]

To show security, we need to prove that for any PPT adversary A, we have \(\Pr[\text{Adv wins}] \leq \text{negl}(n)\)

**One-time Signature** *(q = 1)*: We will start with a weaker version where we only require security to hold as long as \(q = 1\). We call such a signature scheme to be one-time secure signature.

- Let \(\{f_n : \{0, 1\}^n \rightarrow \{0, 1\}^n\}\) be a one-way function.

- **KeyGen(1^n)**: sample a \(2 \times n\) matrix where each entry is \(x_{i,b} \leftarrow \{0, 1\}^n\)

\[
\begin{bmatrix}
  x_{1,0} & \ldots & x_{n,0} \\
  x_{1,1} & \ldots & x_{n,1}
\end{bmatrix} = sk,
\begin{bmatrix}
  f_n(x_{1,0}) & \ldots & f_n(x_{n,0}) \\
  f_n(x_{1,1}) & \ldots & f_n(x_{n,1})
\end{bmatrix} = vk \tag{6.1}
\]

- **Sign(sk, m \in \{0, 1\}^n)**: \(m = (m_1, \ldots, m_n), \sigma = (x_{1,m_1}, \ldots, x_{n,m_n})\)

- **Verify(vk, (m, \sigma))**: \(f_n(\sigma_i) = vk_{i,m_i}\) for all \((\sigma_1, \ldots, \sigma_n)\)

**Proof of One-time Security**: Consider the following game, note that we are only able to query once instead of \(q\) times as above.

\[
\text{Adv} \xleftarrow{vk} \text{Challenger}
\]

\[
\text{Adv} \xrightarrow{m} \text{Challenger}
\]

\[
\text{Adv} \xleftarrow{\sigma} \text{Challenger}
\]

\[
\text{Adv} \xrightarrow{(m^*, \sigma^*)} \text{Challenger}, m^* \neq m
\]

Suppose we have B that knows \(f_n(x)\): B will play the challenger role and try to invert \(f_n(x)\), and we will use this to break one-wayness of \(f_n(x)\)
$i^* \in \{1, \ldots n\}, b^* \in \{0, 1\}, v_{k_i, b^*} = f(x), B \xleftarrow{vk} \text{Adv}$

$B \xleftarrow{m} \text{Adv}, \text{if } m_{i^*} = b^* \text{ abort 1}$

$B \xrightarrow{\sigma} \text{Adv}$

$B \xleftarrow{(m^*, \sigma^*)} \text{Adv}, \text{if } m_{i^*}^* \neq b^* \text{ abort 2}$

\text{if not } \sigma^*_i \text{ is a pre-image of } f(x)$

**Normal game:** By contrast, if a normal game is played: $\text{Adv} \xrightarrow{(m^*, \sigma^*)} \text{Challenger}, \text{now suppose } \Pr[\text{Adv wins}] = \mu(n), \text{which is non-negligible.}$

$H_1:$

$\text{Adv} \xleftarrow{vk} \text{Challenger, } i^* \in \{1, \ldots n\}, b^* \in \{0, 1\}$

$\text{Adv} \xrightarrow{m} \text{Challenger, if } m_{i^*} = b^* \text{ abort 1}$

$\text{Adv} \xleftarrow{\sigma} \text{Challenger, note that } \sigma \text{ doesn’t give any information of } i^*$

$\text{Adv} \xrightarrow{(m^*, \sigma^*)} \text{Challenger, if } m_{i^*}^* \neq b^* \text{ abort 2}$

The probability $\Pr[\text{Adv wins in } H_1] = \frac{1}{2} \times \frac{1}{n} \times \mu(n) = \frac{\mu(n)}{2n}$, because $i^* \in \{1, \ldots n\}, b^* \in \{0, 1\}$. We can use the adversary in $H_1$ to invert the one-way function by embedding the one-way function challenge at position $(i^*, b^*)$. This is a contradiction.