Lecture 20: Public-key Encryption & Hybrid Encryption
Lecture 20: Public-key Encryption & Hybrid Encryption
Suppose there is a 2-round Key-Agreement protocol. This means that there exists a protocol where

- Bob sends the first message $m_B$
- Alice sends the second message $m_A$
- Now, parties can compute a secret key $key$ that is hidden from an eavesdropper (who got to see the first message by Bob and the second message by Alice)
- For example, the Diffie-Hellman key-exchange protocol. Bob sends $m_B = g^b$, Alice sends $m_A = g^a$, and both parties compute the key $key = g^{ab}$, but it remains hidden from any computationally bounded adversary who sees only $A = g^a$ and $B = g^b$.

Using this 2-round key-agreement protocol we can construct a public-key encryption scheme. For example, using the Diffie-Hellman key-exchange protocol, we shall construct the ElGamal public-key encryption scheme
Suppose we have a protocol $\Pi_{2-KA}$, which is a 2-round key-agreement protocol that looks like the following:

\begin{center}
\begin{tikzpicture}
  \node [draw] (alice) {Alice};
  \node [draw] (bob) at (3,0) {Bob};
  \draw[->] (alice) -- node[midway,above] {$m_B$} (bob);
  \draw[->] (bob) -- node[midway,above] {$m_A$} (alice);
  \node at (1.5,-1) {Compute key $k$};
  \node at (2.5,-1) {Compute key $k$};
\end{tikzpicture}
\end{center}

Note that $\Pi_{2-KA}$ can be any 2-round key-agreement protocol. One such example is the Diffie-Hellman key-agreement protocol. The next slide presents this protocol in this template.
For example, we consider $\Pi_{2-KA}$ to be the Diffie-Hellman key agreement protocol.

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m_B = g^b$</td>
<td></td>
</tr>
<tr>
<td>$m_A = g^a$</td>
<td></td>
</tr>
</tbody>
</table>

Compute key $k = m_B^a$  
Compute key $k = m_A^b$
Suppose we have a private-key encryption scheme \((\text{Gen}, \text{Enc}, \text{Dec})\). Without loss of generality, we can assume that \(\text{Gen}()\) outputs a uniformly random key \(sk\) from a set \(S\). Recall that a private-key encryption scheme looks as follows:

\[
\begin{align*}
\text{Gen} () & \quad \text{Gen} () \\
\downarrow & \quad \downarrow \\
sk & \quad sk \\
\text{Enc}_{sk}(m) & \quad \text{Dec}_{sk}(c) \\
\end{align*}
\]

\[
c = \text{Enc}_{sk}(m)
\]

\[
\tilde{m} = \text{Dec}_{sk}(c)
\]
Consider, for example, the one-time pad encryption scheme.

\[ \text{Gen}(\cdot) \]

\[ \text{sk} \rightarrow \text{Gen}(\cdot) \]

\[ \text{sk} \rightarrow \text{sk} \]

\[ c = m \circ \text{sk} \]

\[ \tilde{m} = c \circ \text{inv}(\text{sk}) \]
If the key of the first component is random over the set $S$ (from which the private-key of the second-component is chosen) then we can stick together these two protocols as follows:

**Alice**
- Compute key $k$
- Set $sk = k$
- $c = \text{Enc}_{sk}(m)$

**Bob**
- Compute key $k$
- Set $sk = k$
- $\tilde{m} = \text{Dec}_{sk}(c)$

Diagram:
```
<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$m_B$</td>
</tr>
<tr>
<td></td>
<td>$m_A$</td>
</tr>
<tr>
<td>Compute key $k$</td>
<td>Compute key $k$</td>
</tr>
<tr>
<td>Set $sk = k$</td>
<td>Set $sk = k$</td>
</tr>
<tr>
<td>$c = \text{Enc}_{sk}(m)$</td>
<td>$\tilde{m} = \text{Dec}_{sk}(c)$</td>
</tr>
</tbody>
</table>
We can merge the message $m_A$ and $c$ into one-single message. And we get the following scheme.

\[
\begin{align*}
\text{Alice} & \quad m_B \\
\text{Bob} & \\
\text{Compute} & \quad \text{key } k \\
\text{Set } sk &= k \\
\text{Bob} & \quad m_A, c = \text{Enc}_{sk}(m) \\
\text{Compute} & \quad \text{key } k \\
\text{Set } sk &= k \\
\tilde{m} &= \text{Dec}_{sk}(c)
\end{align*}
\]
Every time we want to encrypt a message $m$, we calculate a fresh key $k$. And we get the following scheme.

**Alice**
- Compute key $k$
- Set $sk = k$

**Bob**
- $m_A, c = Enc_{sk}(m)$
- Compute key $k$
- Set $sk = k$
- $\tilde{m} = Dec_{sk}(c)$
Finally, we interpret the message $m_B$ as the public-key for Bob. And the messages $(m_A, c)$ as the encryption of the message $m$. This gives us our public-key encryption scheme!

\[
\begin{array}{c}
\text{Alice} \\
\text{pk} = m_B \\
\text{Compute key } k \\
\text{Set } sk = k
\end{array} \quad \Rightarrow \quad \begin{array}{c}
\text{Bob} \\
\text{c} = (m_A, c'), \text{ where } c' = \text{Enc}_{sk}(m) \\
\text{Compute key } k \\
\text{Set } sk = k \\
\hat{m} = \text{Dec}_{sk}(c)
\end{array}
\]
Example 1

Suppose our first component is Diffie-Hellman key-agreement protocol and the second component is one-time pad. Then we get the following public-key encryption scheme.

Alice

pk = \( g^b \)

Bob

\[ c = (m_A = g^a, c' = m \cdot g^{ab}) \]

Compute
key \( k = g^{ab} \)
Set \( sk = k \)
\[ \tilde{m} = c' \cdot \text{inv}(g^{ab}) \]
This is the ElGamal public-key encryption scheme!
Let us summarize the ElGamal Public-key Encryption as an instantiation of 2-round Diffie-Hellman key-agreement protocol and the one-time pad private-key encryption scheme.

Recall that to describe a private-key encryption scheme we had to provide the algorithms \((\text{Gen}, \text{Enc}, \text{Dec})\). Similarly, to describe a public-key encryption scheme, we will have to provide the \((\text{Gen}, \text{Enc}, \text{Dec})\) algorithms.

Assume that the DDH Assumption holds for the group \((G, \circ)\) of size \(N\), and the group \(G\) has a generator \(g\).

For perspective, \(N\) is large and is in the order of \(2^n\), where \(n = 1024\). Our algorithms have to be polynomial in \(n\) and the adversary, to break the scheme, has to invest roughly \(2^{\text{constant} \cdot n}\) effort.
Generation Algorithm.

Recall that in the private-key encryption scheme the generation algorithm Gen() outputs the secret-key for the encryption scheme. In public-key encryption, the generation algorithm has to output the public-key pk for the scheme. Additionally, it has to output the “trapdoor” trap that assists the receiver to decrypt the cipher-text. If such a trapdoor does not exist, then Bob gets no additional advantage over an eavesdropper to decrypt the cipher-text.

\begin{itemize}
  \item \text{Gen}():
    \begin{enumerate}
      \item Sample \( b \leftarrow \{0, 1, 2, \ldots, N - 1\} \)
      \item Compute \( B = g^b \) (using repeated squaring technique)
      \item Return \((pk = B, \text{trap} = b)\)
    \end{enumerate}
\end{itemize}

Now, the receiver can broadcast the pk to everyone and keep trap secret with herself to assist in the decryption algorithm.
Encryption Algorithm.

Recall that in the private-key encryption scheme the encryption algorithm takes two inputs (the secret-key and the message) $Enc_{sk}(m)$ and outputs the cipher-text. In the public-key encryption, it will take the public-key and the message as input and output the cipher-text.

$Enc_{pk}(m)$:

1. Sample $a \leftarrow \{0, 1, 2, \ldots, N - 1\}$
2. Compute $A = g^a$ (using repeated squaring technique)
3. Compute mask $= pk^a$ (using repeated squaring technique)
4. Return the cipher-text $c = (A, m \circ \text{mask})$

In the ElGamal encryption scheme $pk = B$. Note that each time the encryption algorithm is invoked, it will create a new random mask. If the same mask is generated in two different invocations of the encryption algorithm, then it must be the case that the same $A$ was generate in those two invocations. That
implies that the same $a$ was generate in those two invocations, which has probability $\sqrt{2^{-n}} = 2^{-n/2}$ by the birthday bound)
Decryption Algorithm.

- Recall that in the private-key encryption scheme the decryption algorithm takes two inputs (the secret-key and the cipher-text) \( \text{Dec}_{sk}(c) \). In the public-key encryption, it will take the cipher-text and the trapdoor generated during the generation procedure as input.

\[
\text{Dec}_{\text{trap}}(\tilde{A}, \tilde{c}): \\
1. \text{Compute } \tilde{\text{mask}}(\tilde{A})^{\text{trap}} \\
2. \text{Return } \tilde{c} \circ \text{inv}(\tilde{\text{mask}})
\]

Recall that \( \text{trap} = b \). If \( \tilde{A} = g^a \), then \( \tilde{\text{mask}} = g^{ab} \).
We will combine any public-key encryption scheme with any private-key encryption scheme to create a new public-key encryption (called, the hybrid-encryption scheme).

We emphasize that any public-key encryption scheme can be used. It need not be the ElGamal Scheme. You can choose any encryption scheme that you prefer.

The benefit of hybrid-encryption is that it allows us to combine two encryption scheme in a modular fashion.

Suppose the public-key encryption scheme is provided by the triplet of algorithms

$$(\text{Gen}^{(\text{pub})}, \text{Enc}^{(\text{pub})}, \text{Dec}^{(\text{pub})})$$
Suppose the private-key encryption scheme is provided by the triplet of algorithms

$$(\text{Gen}^{\text{priv}}, \text{Enc}^{\text{priv}}, \text{Dec}^{\text{priv}})$$

Now, we need to describe the hybrid-encryption scheme algorithms

$$(\text{Gen}^{\text{hyb}}, \text{Enc}^{\text{hyb}}, \text{Dec}^{\text{hyb}})$$
Let us first draw a block-diagram for intuition purpose

![Block-diagram for Hybrid Encryption III]

- The secret-key $sk$ will be encrypted by the public-key encryption.
- The secret-key $sk$ will be used to encrypt the actual message $m$ using the private-key encryption.
Generation Algorithm for Hybrid-Encryption.

$\text{Gen}^{(\text{hyb})}()$:

1. Return $(pk, \text{trap}) = \text{Gen}^{(\text{pub})}()$

The receiver broadcasts $pk$ and keeps $\text{trap}$ safe with herself.
Encryption Algorithm for Hybrid-Encryption.

\[ \text{Enc}_{pk}(m): \]

1. Generate \( sk = \text{Gen}^{\text{priv}}() \)
2. Encrypt the secret-key \( c_1 = \text{Enc}_{pk}^{\text{pub}}(sk) \)
3. Encrypt the message \( c_2 = \text{Enc}_{sk}^{\text{priv}}(m) \)
4. Return the cipher-test \( (c_1, c_2) \)
Decryption Algorithm for Hybrid-Encryption.

\[ \text{Dec}_{\text{trap}}(\tilde{c}_1, \tilde{c}_2): \]

1. Decrypt the secret-key \( \tilde{k} = \text{Dec}^{(\text{pub})}_{\text{trap}}(\tilde{c}_1) \)
2. Return the decrypted message \( \tilde{m} = \text{Dec}^{(\text{priv})}_{\tilde{k}}(\tilde{c}_2) \)