# Lecture 29: 2-round Key Agreement and Public-key Encryption

# Overview

- 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, 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 the adversary.
- 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

# First Component: 2-round Key-Agreement Protocol I

- Suppose we have a protocol $\Pi_{2-\mathsf{KA}}$, which is a 2-round key-agreement protocol that looks like the following

| <u>Alice</u> | | <u>Bob</u> |
|---|---|---|
| | $\xleftarrow{\quad m_B \quad}$ | |
| | $\xrightarrow{\quad m_A \quad}$ | |
| Compute key k | | Compute key k |

- Note that $\Pi_{2-\mathsf{KA}}$ 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.

# First Component: 2-round Key-Agreement Protocol II

- For example, we consider $\Pi_{2-\mathrm{KA}}$ to be the Diffie-Hellman key agreement protocol

| Alice | | Bob |
|---|---|---|
| | $\xleftarrow{\quad m_B = g^b \quad}$ | |
| | $\xrightarrow{\quad m_A = g^a \quad}$ | |
| Compute<br>key $\mathrm{k} = m_B^a$ | | Compute<br>key $\mathrm{k} = m_A^b$ |

- Suppose we have a private-key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. Without loss of generality, we can assume that $\mathsf{Gen}()$ outputs a uniformly random key $\mathsf{sk}$ from a set $S$. Recall that a private-key encryption scheme looks as follows

$$\mathsf{Gen}\ ()$$

$$\mathsf{sk} \qquad\qquad\qquad\qquad \mathsf{sk}$$

$$\xrightarrow{c = \mathsf{Enc}_{\mathsf{sk}}(m)}$$

$$\widetilde{m} = \mathsf{Dec}_{\mathsf{sk}}(c)$$

# Second Component: Private-key Encryption II

- Consider, for example, the one-time pad encryption scheme

Gen ()

sk　　　　　　　　sk

$$\xrightarrow{c = m \circ \mathsf{sk}}$$

$$\widetilde{m} = c \circ \mathsf{inv}(\mathsf{sk})$$

# Combining to obtain a Public-key Encryption Scheme I

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 | | Bob |
|---|---|---|
| | $\xleftarrow{\quad m_B \quad}$ | |
| | $\xrightarrow{\quad m_A \quad}$ | |
| Compute key k | | Compute key k |
| Set $\mathsf{sk} = \mathsf{k}$ | | Set $\mathsf{sk} = \mathsf{k}$ |
| | $\xrightarrow{c = \mathrm{Enc}_{\mathsf{sk}}(m)}$ | |
| | | $\widetilde{m} = \mathrm{Dec}_{\mathsf{sk}}(c)$ |

# Combining to obtain a Public-key Encryption Scheme II

We can merge the message $m_A$ and $c$ into one-single message. And we get the following scheme.

| Alice | | Bob |
|---|---|---|
| | $\xleftarrow{\quad m_B \quad}$ | |
| Compute key k | | |
| Set sk $=$ k | $\xrightarrow{\quad m_A, c = \mathsf{Enc}_{\mathsf{sk}}(m) \quad}$ | |
| | | Compute key k |
| | | Set sk $=$ k |
| | | $\widetilde{m} = \mathsf{Dec}_{\mathsf{sk}}(c)$ |

# Combining to obtain a Public-key Encryption Scheme III

Every time we want to encrypt a message $m$, we calculate a fresh key k. And we get the following scheme.

| <u>Alice</u> | | <u>Bob</u> |
|---|---|---|
| | $\xleftarrow{\quad m_B \quad}$ | |
| Compute key k<br>Set sk $=$ k | | |
| | $\xrightarrow{\quad m_A,\, c = \mathsf{Enc}_{\mathsf{sk}}(m) \quad}$ | |
| | | Compute key k<br>Set sk $=$ k<br>$\widetilde{m} = \mathsf{Dec}_{\mathsf{sk}}(c)$ |

# Combining to obtain a Public-key Encryption Scheme IV

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!

| <u>Alice</u> | | <u>Bob</u> |
|---|---|---|
| | $\xleftarrow{\mathsf{pk} = m_B}$ | |
| Compute key k<br>Set $\mathsf{sk} = \mathsf{k}$ | | |
| | $c = (m_A, c'),$ $\longrightarrow$ where $c' = \mathrm{Enc}_{\mathsf{sk}}(m)$ | |
| | | Compute key k<br>Set $\mathsf{sk} = \mathsf{k}$<br>$\widetilde{m} = \mathrm{Dec}_{\mathsf{sk}}(c)$ |

# Example I

- 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 | | Bob |
|---|---|---|
| | $\xleftarrow{\mathsf{pk} = g^b}$ | |
| Compute<br>key $\mathsf{k} = g^{ab}$<br>Set $\mathsf{sk} = \mathsf{k}$ | | |
| | $\xrightarrow{c = (m_A = g^a, c' = m \cdot g^{ab})}$ | |
| | | Compute<br>key $\mathsf{k} = g^{ab}$<br>Set $\mathsf{sk} = \mathsf{k}$<br>$\widetilde{m} = c' \cdot \mathsf{inv}(g^{ab})$ |

# Example II

This is the ElGamal public-key encryption scheme!