CSCI361
Computer Security

Key Management

Outline

- Symmetric keys.
  - Diffie-Hellman Key exchange.
  - Man-in-the-Middle attack.
- PKC keys.
  - Certificates.
    - X.509.
  - Key generation and storage.
  - Handshakes.
  - Certification paths.

Key management

- In a cryptosystem, symmetric or public-key, it is necessary for users to obtain the required keys. This creates another security problem.
- Secure communication depends on secure key exchange (distribution/establishment/generation).

- For **secret key cryptography** the main problems are *key management* and key *distribution*.
  - Keys need to be distributed via *secure channels*.
- In **public key (PK) systems**, we have the problem of *key authentication*. Which key (really) belongs to who?
  - Keys need to be distributed via *authentic channels*.

Symmetric key establishment

- In these systems the two users must share a common key. This sharing can be achieved in a number of ways:

1. Two users use a *supplementary secure channel*, such as a courier service.
   - **Disadvantages**: Costly, slow, questionable security.
2. Key exchange via a *trusted authority*: Each user can communicate securely with **T**, a trusted central authority.
   - **Disadvantages**: Requires a trusted node and creates a bottleneck. For every key between two users at least two communications involving **T** are necessary. **T** can be replaced by a network of authorities, but this increases the number of entry points for the intruder.
3. Key exchange using public channels: Something like the Diffie-Hellman key exchange protocol can be used.

Diffie-Hellman Key Exchange

- **A**lice and **B**ob agree on a common prime **p** and a common primitive element **g** of $\mathbf{Z_p}$.
- **Step One**: Secret keys $\mathbf{X_A}$ and $\mathbf{X_B}$.
  - **A** chooses a random number $\mathbf{X_A}$, $\mathbf{1 \leq X_A \leq p}$.
  - **B** chooses a random number $\mathbf{X_B}$, $\mathbf{1 \leq X_B \leq p}$.
- **Step Two**: Public keys $\mathbf{Y_A}$ and $\mathbf{Y_B}$.
  - **A** calculates $\mathbf{Y_A = g^{X_A} \bmod p}$.
  - **B** calculates $\mathbf{Y_B = g^{X_B} \bmod p}$.

$$A \rightarrow B : Y_A$$

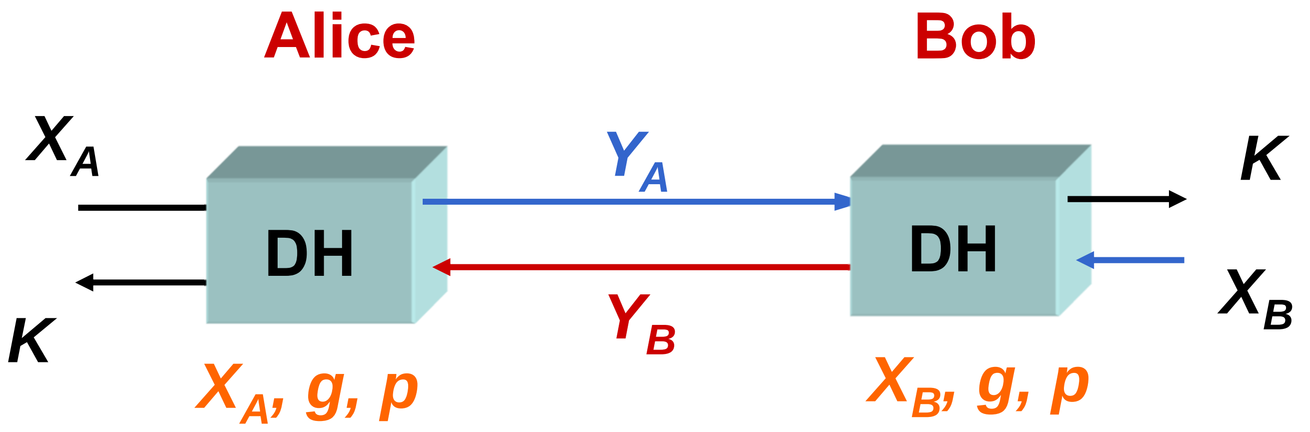
$$B \rightarrow A : Y_B$$

- **Step Three**:
  - **A** calculates $\mathbf{(Y_B)^{X_A}}$ **mod p.**
  - **B** calculates $\mathbf{(Y_A)^{X_B}}$ **mod p.**
  - These factors both equal $\mathbf{g^{X_A X_B}}$ **mod p**

- The security of this system depends on the difficulty of computing discrete logarithms.
  - In this context obtaining X_A from Y_A.

Diffie-Hellman Key Exchange

- The Protocol



1: $A \rightarrow B$: Y_A

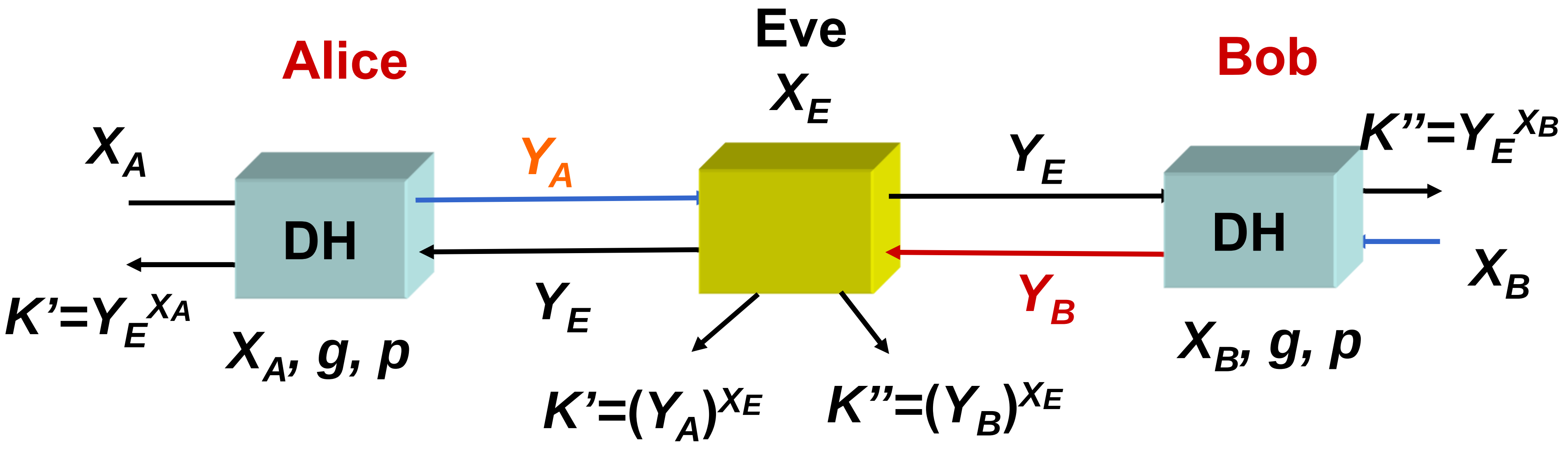
2: $B \rightarrow A$: Y_B

- **Example**: **p = 13**, **g = 2**.
  - **A** chooses $\mathbf{X_A} = 4$.
  - **B** chooses $\mathbf{X_B} = 5$.
  - $\mathbf{Y_A} = 2^4 \bmod 13 = 3$.
  - $\mathbf{Y_B} = 2^5 \bmod 13 = 6$.
  - $(\mathbf{Y_B})^{\mathbf{X_A}}$ **mod p** $= 6^4 \bmod 13 = 9$.
  - $(\mathbf{Y_A})^{\mathbf{X_B}}$ **mod p** $= 3^5 \bmod 13 = 9$.

Alice and Bob have established a common key 9.

Diffie-Hellman Key Exchange

- **Man-in-the-Middle Attack**



1: $A \rightarrow E$: Y_A

2: $E \rightarrow B$: Y_E

3: $B \rightarrow E$: Y_B

4: $E \rightarrow A$: Y_E

There are various ways of fixing this problem (although we aren't going to look at them here).

Key Exchange in PKC

- To use a public key algorithm, **A**lice must have **B**ob's public key.
- This can be obtained
  - Directly from **B**ob.
  - From a public directory.

- While the public component need not be kept secret, **authenticity** and **integrity** are real problems:
  - **Authentication**: If **A**lice thinks that $\mathbf{Z_O}$ is **B**ob's key, whereas it is actually **O**scar's key, then **A**lice might encrypt using $\mathbf{Z_O}$ and (unknowingly) allow **O**scar to decrypt the message.
  - **Integrity**: Public keys are generally over 1,000 bits. A single bit error transmission of the public component makes it useless.

- Almost all key exchange systems have some kind of trusted authority. However, the implication of compromise of the trusted authority is not as severe as in the symmetric key system, mainly because the trusted authority does not necessarily know the secret component and so cannot read the communications.

- *Validity* is an important consideration. Keys are usually valid for certain period of time, their *lifetime*. Key updates may be:
  - Sent to users in real time.
  - Obtained by the users during periodic checks with the trusted authority.

Certificates

- A useful technique that provides a partial solution to the authenticity and integrity problems, is the use of **certificates**. This technique assumes:
  - A central (trusted) authority **T**.
  - A secure communication channel between each user and **T**: for example each user knows $\mathbf{Z_T}$, the public key of **T**.

- The system can be sketched as follows:
  - **A**lice securely sends $\mathbf{Z_A}$ to **T**.
  - **A**lice receives a certificate, $\mathbf{C_A}$, signed by **T** that binds $\mathbf{Z_A}$ to Alice.
  - This certificate is verifiable by everyone who has the public key of **T**.
- A certificate has the following form:

$$M = [Z_A, \textit{Alice's ID, validity period}].$$

$$C_A = D_{Z_T}(M)$$

- Look at a web browser, say **Mozilla**, under security preferences to see some certificates.

Encryption with a certificate

- When **A**lice wants to send an encrypted message to **B**ob:
  1. She obtains **B**ob's certificate.
  2. Verifies the signature of **T** on the certificate.
  3. Extracts $\mathbf{Z_B}$ and uses it to encrypt the message.

- A certificate may be invalidated before its expiry date if the key is known to be compromised. The central authority must periodically issue a list of invalidated certificates.

Key distribution in a certificate based PKC

- $\mathbf{C_A}$ and $\mathbf{C_B}$ are certificates issued by **T**. **A**lice may obtain **B**ob's certificate by:
  - Asking **B**ob directly.
  - Looking in a public directory.
- In both cases, there is a potential integrity problem, in particular the certificate obtained may have recently been invalidated.
- One solution is to obtain **B**ob's certificate through **T**.
  - **Advantage**: The key $\mathbf{Z_B}$ is authentic.
  - **Disadvantage**: **T** is a bottleneck, concentration of trust, etc.
- Alternatively one can implement a security policy which advises persons to check for up-to-date invalidations, perhaps a list on the trusted authorities website say, before using the key.

Who should generate keys?

- If **T** generates keys pairs:
  - **Advantages**: Keys may be chosen more carefully. Neither of the participants is advantaged.
  - **Disadvantages**: **T** can eavesdrop! Moreover it again produces a bottleneck in the system.
- If users generate their own keys the latter problem disappears.

- Users also need some provision for accessing previous keys so that the files that are encrypted with them remain accessible.

PKC for symmetric key distribution

- A major application of PKC is the distribution of symmetric keys. PKC can be used to establish the common information (secret key, initializing vectors, etc.) required by a symmetric cryptosystem.
- PKC can be used to send messages securely and authentically. The information shared for the symmetric cryptosystem is just a particular message.
- Remember that symmetric key cryptography is significantly faster then PKC for equivalent security.