Cryptography and Security

Cunsheng DING

HKUST, Hong Kong

Version 3

Lecture 06: Key Management for One-key Ciphers

Topics of this Lecture

1. Passive and active attacks.
2. The generation and distribution of secret keys.
3. A key distribution protocol with a key distribution center.
4. The Diffie-Hellman key exchange protocol.

Passive and Active Attacks

Passive and Active attacks

Passive attacks: Any attack on a security system under the assumption that the attacker can only intercept texts exchanged over a communication channel is called a **passive attack**.

Active attacks: Any attack on a security system under the assumption that the attacker can stop, intercept, delete, modify, and replay texts exchanged over a communication channel or insert his/her texts into the channel is called an **active attack**. In such a scenario, we say that the attacker has **full control** over the communication channel.

Remark: Active attacks are more powerful than passive attacks.

Remark: A system may be secure with respect to passive attacks, but insecure with respect to active attacks.

Secret Key Generation

An Aspect of Key Management

Secret Key Generation

Question: How do you generate a secret key for a one-key cipher?

Answer: It depends on the specific cipher.

Case I: The secret key k is a binary string $k_1 k_2 \cdots k_n$, where k_i are independent of each other.

Solution 1: If n is not large, say 128, one can flip a coin n times. This will generate a **random binary string** of length n.

Solution 2: Use a "pseudorandom number generator".

Case II: Key bits must satisfy certain relations.

In this case, no general approach exits. It differs from system to system.

Key Generation in a Cipher: Example

- The plaintext and ciphertext spaces: $\mathcal{M} = \mathcal{C} = \{0,1\}^*$.
- $\mathcal{K}$ consisting of all binary 128×128 invertible matrices.
- Encryption is block by block (block size 128 bits). For a secret key $K \in \mathcal{K}$ and a plaintext block m_i, the encryption is

$$E_K(m_i) = m_i K = c_i.$$

The decryption function is

$$D_K(c_i) = c_i K^{-1} = m_i.$$

Question: How do you generate a binary 128×128 invertible matrix K?

Remark: Flipping a coin 128×128 times does not work!

Secret Key Establishment

The Necessity of Establishing a Common Key

- To use an one-key cipher for secure communication, two parties must share the same key.
- The key must be protected from access by others.
- The key should be changed regularly (an adversary or enemy may learn the key in some way).

Key distribution: One party generates a key and delivers it to others confidentially.

Key agreement: The communication parties first exchange key materials and then compute a common key with their exchanged key materials.

Remark: Today, we will see a key distribution protocol and a key agreement protocol.

A Key Distribution Protocol with a Key Distribution Center

A Key Distribution Protocol

Parties involved: A key distribution center (KDC), a group of people to communicate with each other.

Requirements: Whenever A wants to communicate with B, the KDC should generate a one-time key (called **session key**) and distribute it to A and B. Both confidentiality and authenticity must be achieved.

Remark: The session key (one-time key) is established only for this session of communication between A and B.

A Key Distribution Protocol – Continued

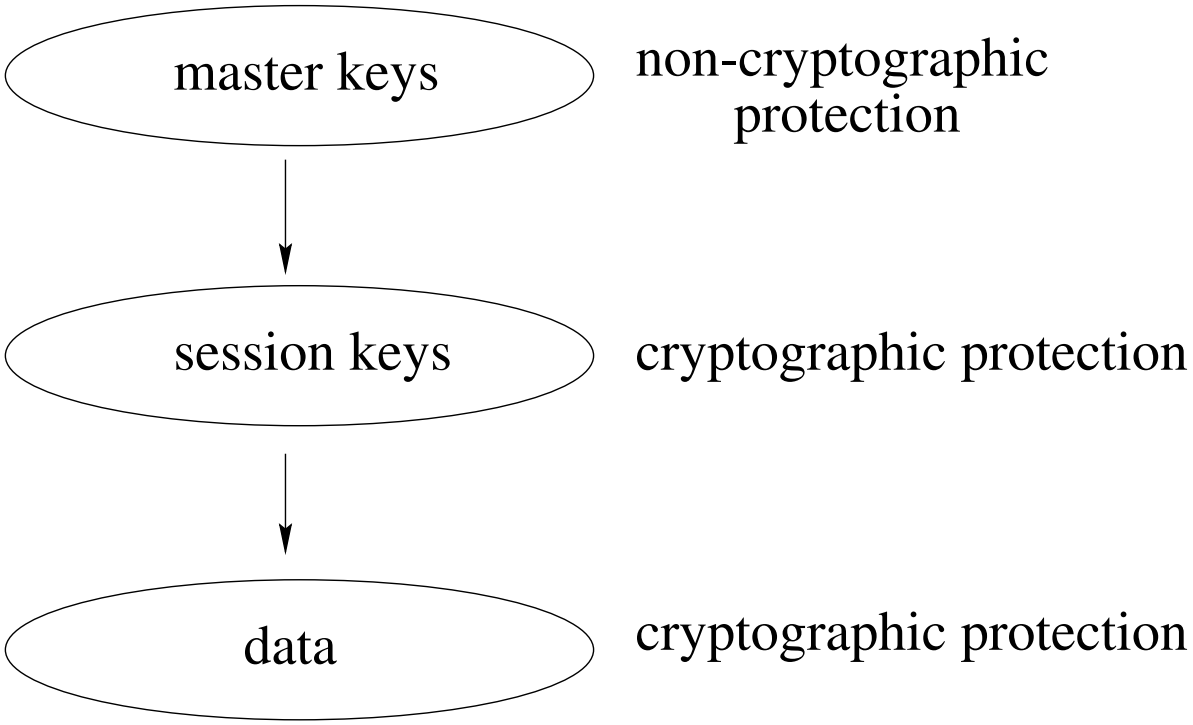
Building blocks needed:

- The KDC and all parties involved in this communication system use a one-key block cipher.
- The KDC and each party A share a secret key k_a, which is called a **master key**.
- The KDC is a trustful party.

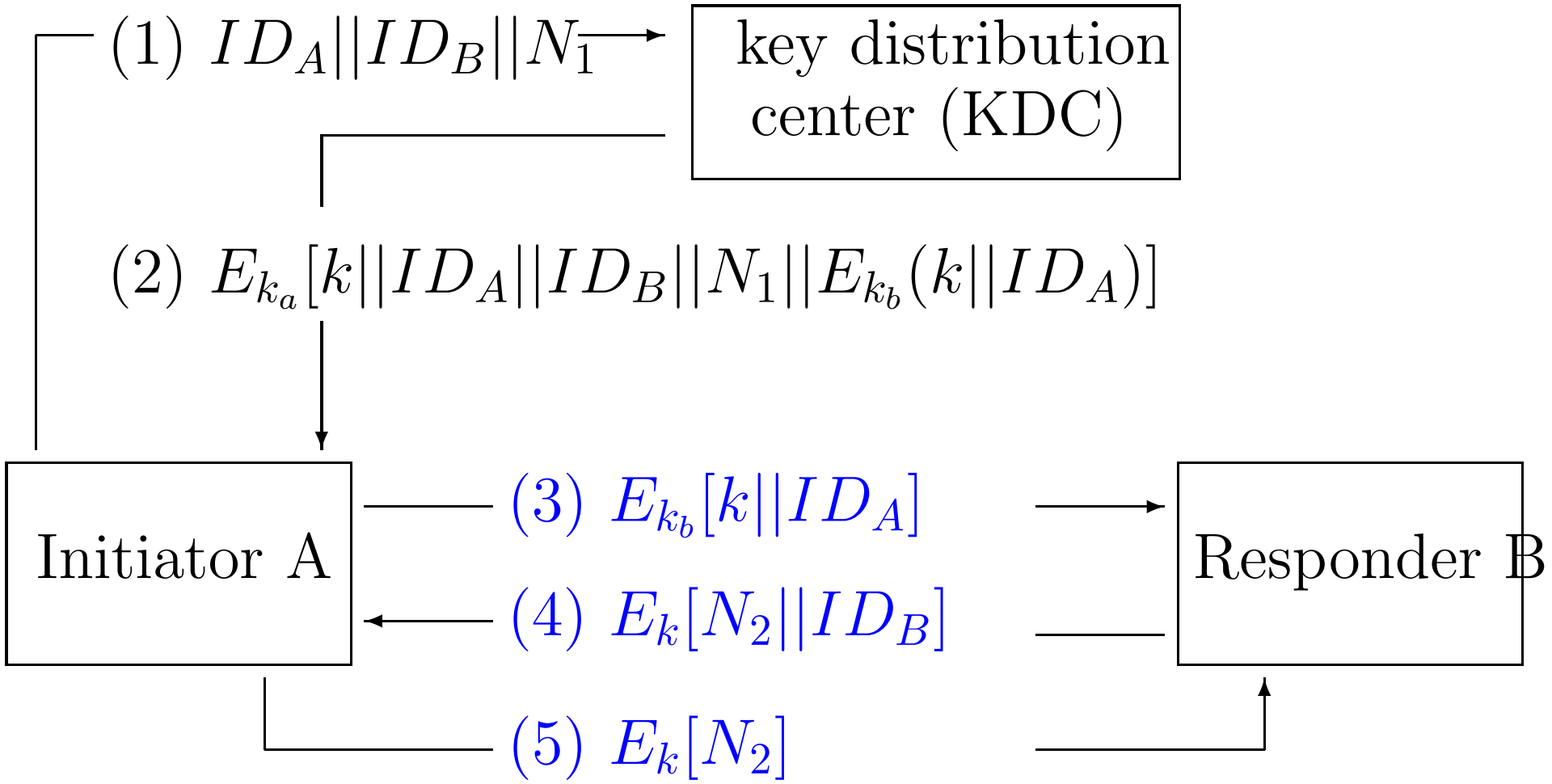
Remark: The master keys are used to protect the sessions keys when they are distributed.

A Key Distribution Protocol – Continued

Pictorial description of use of the key hierarchy:



A Key Distribution Protocol



Parameters in the Key Distribution Protocol

- N_i is a nonce (a string of characters), used as identifier for that transaction and is a challenge to the other party.
- k_a, k_b master keys, k secret key.
- ID_A, the identifier of A.
- ID_B, the identifier of B.

Explaining the Key Distribution Protocol (1)

- The first message may be a forged or modified one. But after receiving Message (2), A will detect such a forgery or modification.
- After receiving Message (2), A can authenticate the creator and sender of Message (2) (sender by $ID_A||ID_B$ and antireplay by N_1).
- Steps (3), (4) and (5) are for mutual authentication. Why?
  - The creator of Message (3) must be the KDC as people other than the KDC and B do not know k_b and cannot embed ID_A inside the ciphertext.
  - If Message (3) is a replayed message, A will discover this replay after processing Message (4) (see next page for justification).

  Hence, Message (3) is the KDC's current data, but its sender may be A or somebody else. Therefore, Message (5) is necessary.

Explaining the Key Distribution Protocol (2)

If Message (3) is a replayed message, A will discover this replay after processing Message (4).

- If Message (3) is a replayed message, then Messages in (3) and (4) should be $E_{k_b}(k'||ID_A)$ and $E_{k'}(N_2||ID_B)$, where $k' \neq k$ and k' is an earlier key. Then after computing $D_k(E_{k'}(N_2||ID_B))$, A cannot find ID_B in the decrypted text $D_k(E_{k'}(N_2||ID_B))$.

Steps (3), (4) and (5) are for mutual authentication. Why?

- The challenge to B is k, and the challenge to A is N_2.
- Mutual authentication is done via a challenge-response sub-protocol.

A Key Agreement Protocol

The Diffie-Hellman Key Exchange Protocol

Discrete logarithms

Primitive roots: Let p be a prime. An integer α is called a **primitive root** of p if each nonzero element $a \in \mathbf{Z}_p$ can be uniquely expressed as

$$a = \alpha^i \bmod p$$

for some integer i, where $0 \leq i \leq p-2$.

Discrete logarithm: The exponent i is referred to as the **discrete logarithm**, or **index**, of a for the base α, and is denoted $\log_\alpha a$ or $\mathrm{ind}_\alpha(a)$.

Discrete logarithm problem:

Given p, α, and a, find $\log_\alpha a$.

This is believed to be a very hard problem for large p.

Primitive roots

Example: 2 is a primitive root of the prime 11. Also we have $\log_2(6) = 9$.

i	0	1	2	3	4	5	6	7	8	9
$2^i \bmod 11$	1	2	4	8	5	10	9	7	3	6

Theorem: Every prime p has at least one primitive root.

The exact number of primitive roots in the range $\{1, 2, \ldots, p-1\}$ is given by $\phi(p-1)$ where

$$\phi(n) = n \prod_{q|n} \left(1 - \frac{1}{q}\right);$$

here $q|n$ means that the prime q divides n.

This is Euler's phi-function (more on this in later lectures).

(For a proof, see a book on number theory. You are not required to do so).

Diffie-Hellman Key Exchange Protocol

User A		User B
Generate random $X_A < p$		Generate random $X_B < p$
calculate $Y_A = \alpha^{X_A} \bmod p$	$Y_A \longrightarrow$	Calculate $Y_B = \alpha^{X_B} \bmod p$
	$\longleftarrow Y_B$	
Calculate $k = (Y_B)^{X_A} \bmod p$		Calculate $k = (Y_A)^{X_B} \bmod p$

Diffie-Hellman Key Exchange Protocol

- It is for two users to exchange a key securely that can then be used for subsequent encryption of message.
- $k = \alpha^{X_A X_B} \bmod p$. Also p and α are publicly known. But X_A and X_B must be kept secret.
- The security with respect to passive attacks is based on the belief that solving the discrete logarithm problem is hard in general.

  It is vulnerable to an active attack if an adversary has full control over the communication channel.

Exercise: Present an effective active attack on the DH key exchange protocol.