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 messages 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 messages exchanged over a communication channel or insert his/her messages into the channel is called an **active attack**. In such a scenario, we say that the attacker has **full control** over the communication channel.

Secret Key Generation

Secret Key Generation

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

Answer: It depends on the specific cryptosystem.

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 long, say 128, flipping a coin n times.

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 message 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 message 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 Generation

Key size: 128 bits or more are recommended (brute-force attack).

Equivalent keys: If $E_{k_1}(m) = E_{k_2}(m)$ for every message m, then k_1 and k_2 are called **equivalent keys**.

Weak keys: A key k is called a **weak key** if cryptanalysis with respect to k is easier, compared with most of the keys.

Remark: It is usually hard to find equivalent keys and weak keys.

Secret Key Establishment

Key Distribution: Necessity

- For conventional encryption, the 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.

Key Distribution: some General Approaches

- A selects a key, and physically delivers it to B.
- A third party can select the key and and physically deliver it to both A and B.
- If A and B have previously and recently used a key, one party can transmit the new key to the other, encrypted using the old key.
- If A and B each has an encrypted connection to a third party C, C can deliver a key on the encrypted links to A and B.

Key Distribution: more General Approaches

- Secret key distribution using a “public key cipher”.
  (It will be introduced later.)

- Other key distribution protocols.

Remark: As an example of protocols for key distribution, we introduce a key distribution protocol using 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 temporary key (called **session key**) and distribute it to A and B. Both confidentiality and authenticity must be achieved.

Remark: The session key (temporary key) is established only for this communications between A and B.

A Key Distribution Protocol – Continued

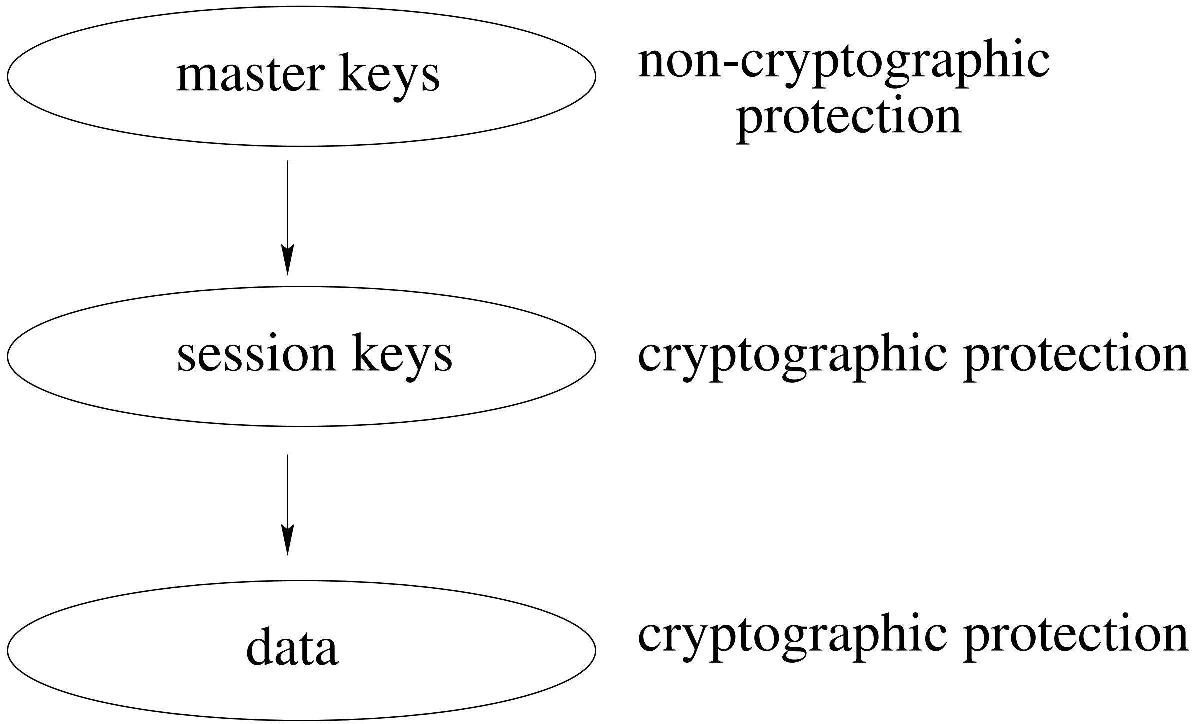
Building blocks needed:

- 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**.

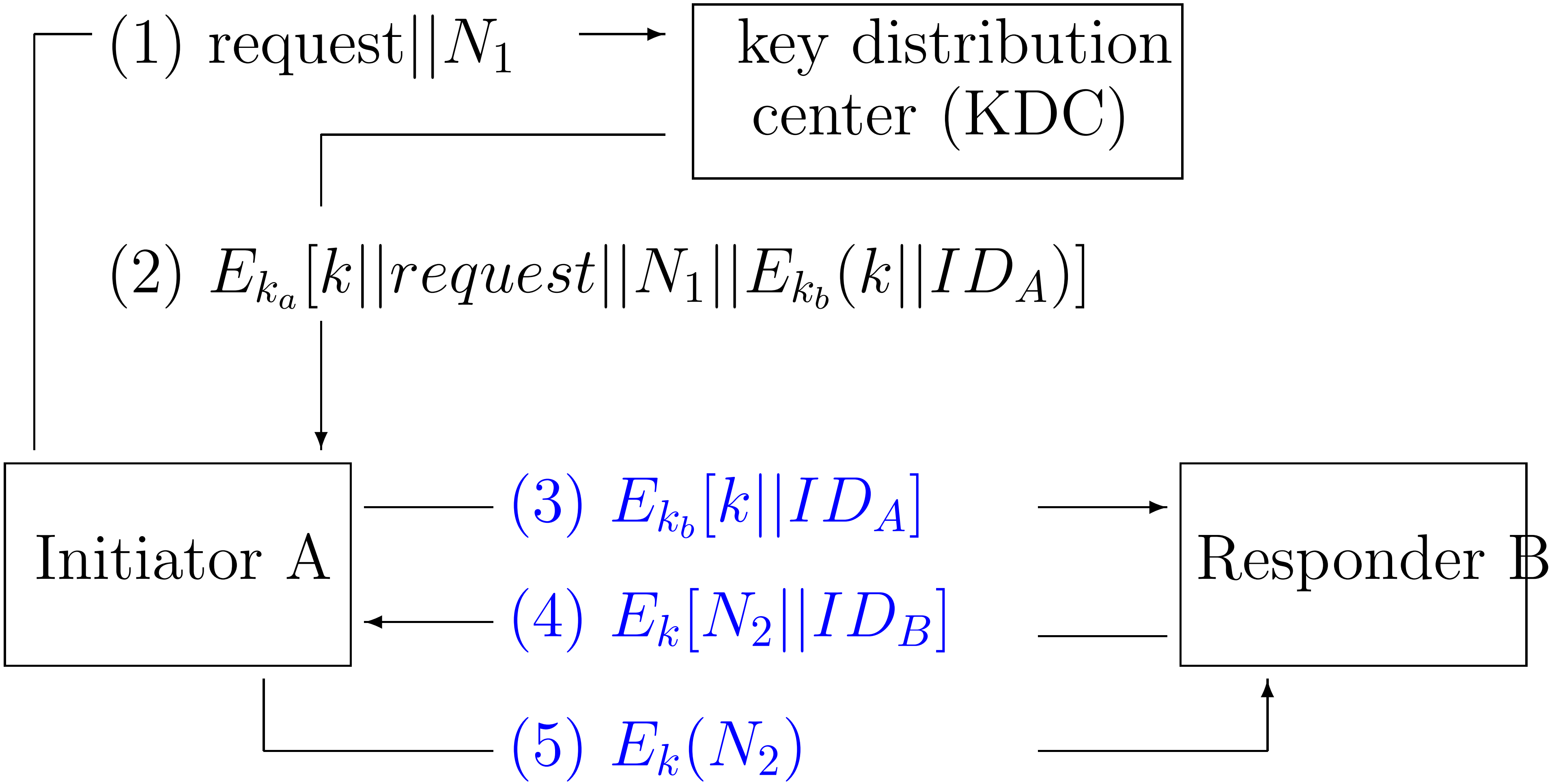
Remark: The master keys are used to protect the sessions keys when they are distributed.

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, used as identifier for that transaction.
- k_a, k_b master keys, k secret key.
- ID_A, the identifier of A.

Explaining the Key Distribution Protocol

- The first message may be a forged or modified one by anyone other than A. But after receiving the message in Step (2), A will detect such a forgery or modification.
- When A receives (2), A can authenticate the sender of the message in Step (2) due to the nonce.

  The message (2) also includes two items intended for B: the one-time session key k, and an identifier of A, i.e., the ID_A.
- The messaged in Step (3) may be a replayed earlier message. Hence, mutual authentication must be done.
- Steps (4), (5) and (3) are for mutual authentication.

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 $0 \le i \le p-2$.

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 in general very hard.
Brute force solution: compute $b = \alpha^i \bmod p$ for all i,
$0 \le i \le p-2$ and check if $b = a$.

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 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 any book on number theory).

To find primitive roots

Rule of tumb: For most primes p have a small primitive root. For example, for the primes less than 100000, approximately 37.5% have 2 as a primitive root, and approximately 87.4% have a primitive root of value 7 or less.

For primes of reasonable size, many programming languages for mathematics have commands for finding primitive roots.

Diffie-Hellman Key Exchange Protocol

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