Cryptography and Security

Cunsheng DING, HKUST

March 3, 2024

Lecture 7: Introduction to Public-Key Cryptography

Objectives of this Lecture

1. Introduce the idea of public-key cryptography.
2. Present the history of public-key cryptography.
3. Outline three applications of public-key ciphers.

A Disadvantage of One-Key Block Ciphers

A Disadvantage of One-Key Block Ciphers

One-key block ciphers: $(\mathcal{M}, \mathcal{C}, \mathcal{K}, E_k, D_k)$, where the encryption and decryption keys are the same.

- The sender and receiver must share the same secret key. Key distribution is a must.
- If 10000 people want to communicate (two and two, in all possible ways), each must keep 9999 secret keys, and the system requires a total of

$$9999 \cdot 10000/2 = 4995000$$

secret keys. This makes key management difficult.

The Idea of Public-Key Cryptography

Two-key Ciphers

A six-tuple $(\mathcal{M}, \mathcal{C}, \mathcal{K}_e, \mathcal{K}_d, E_{k_e}, D_{k_d})$, where

- $\mathcal{M}$, $\mathcal{C}$, $\mathcal{K}_e, \mathcal{K}_d$ are respectively the plaintext space, ciphertext space, encryption key space, and decryption key space;
- $k_e \in \mathcal{K}_e$ and $k_d \in \mathcal{K}_d$ are corresponding encryption and decryption keys respectively;
- E_{k_e} and D_{k_d} are the encryption and decryption transformations, and

  $$D_{k_d}(E_{k_e}(m)) = m,$$

  for all $m \in \mathcal{M}$ (unique and correct decryption).

The Idea of Public-Key Cryptography

Suppose that our university has a two-key cipher $(\mathcal{M}, \mathcal{C}, \mathcal{K}_e, \mathcal{K}_d, E_{k_e}, D_{k_d})$, which is in the public domain.

I generate my encryption and decryption key pair (k_e, k_d), and then publicize k_e in the public domain, in order for anybody else to encrypt a message and send it to me. **Everyone in our university does the same.**

A two-key cipher used in this special way is called a **public-key cipher**.

Comment: The encryption key k_e is called the **public key**, and the decryption key k_d is called the **private key**, which must be kept confidential by its holder.

The Security of Public-Key Ciphers

A public-key cipher $(\mathcal{M}, \mathcal{C}, \mathcal{K}_e, \mathcal{K}_d, E_{k_e}, D_{k_d})$ is **computationally-secure** if and only if the following two conditions are satisfied:

C1: it is "computationally infeasible" to derive the decryption key k_d from the given encryption key k_e; and

C2: it is "computationally infeasible" to derive the plaintext m if the corresponding ciphertext c is known.

- In theory, a public key k_e should contain all information about the private key k_d. But it should be computationally infeasible to retrieve all the information about k_d.
- E_{k_e} is known to everyone. So, its inverse function D_{k_d} is also known in theory. But it should be computationally infeasible to derive the decryption function D_{k_d}.

A Public-key Cipher not Satisfying C1 & C2

Matrix: An $n \times m$ matrix $A = [a[i,j]]$ over $\{0,1\}$ is a 2-dimensional array

$$A = \begin{bmatrix} a[1,1] & a[1,2] & \cdots & a[1,m-1] & a[1,m] \\ a[2,1] & a[2,2] & \cdots & a[2,m-1] & a[2,m] \\ \vdots & \vdots & & \vdots & \vdots \\ a[n,1] & a[n,2] & \cdots & a[n,m-1] & a[n,m] \end{bmatrix},$$

which has n rows and m columns, and each $a[i,j] \in \{0,1\}$.

A Public-key Cipher not Satisfying C1 & C2

Given an $n \times m$ matrix A and an $m \times l$ matrix B, the multiplication $C = AB$ over $\mathbf{Z}_2$ is an $n \times l$ matrix given by

$$c[i,j] = \sum_{k=1}^{m} a[i,k]b[k,j]$$

for $1 \leq i \leq n$ and $1 \leq j \leq l$, where operations in the sum are mudulo-2 additions and mudulo-2 multiplications.

A Public-key Cipher not Satisfying C1 & C2

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix},$$

then

$$C = AB = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

A Public-key Cipher not Satisfying C1 & C2

Definition: Let A be an $n \times n$ matrix over $\mathbf{Z}_2$. If there exists an $n \times n$ matrix $B \in \mathbf{Z}_2$ such that $AB = I_n$, i.e., the $n \times n$ identity matrix, then A is said **invertible**, and B is the **inverse matrix** of A.

Example: A is the inverse of itself:

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

A Public-key Cipher not Satisfying C1 & C2

Let $\mathcal{M} = \mathcal{C} = \{0,1\}^*$, all the finite binary strings, and let $\mathcal{K}$ be the set of all invertible 512×512 matrices k over $\mathbf{Z}_2 = \{0,1\}$ with $k \neq k^{-1}$. Each message is broken into blocks of length 512 bits. The encryption and decryption algorithms work on blocks.

Encryption and decryption: For a 512-bit plaintext block x and ciphertext block y,

$$E_k(x) = kx, \quad D_{k^{-1}}(y) = k^{-1}y,$$

where all the arithmetic operations involved in computing kx are modulo-2, and $(k_e, k_d) = (k, k^{-1})$

Comment: C1 and C2 are not satisfied. Why?

Design Requirements for Public-Key Ciphers

The C1 and C2 described before plus the following efficiency requirements:

1. It is “computationally easy” for a party B to generate a pair $\left(k_e^{(B)}, k_d^{(B)}\right)$.
2. It is “computationally easy” for a sender A, knowing the public key and the message to be encrypted, m, to generate the corresponding ciphertext $c = E_{k_e^{(B)}}(m)$.
3. It is “computationally easy” for the receiver B to recover the message $m = D_{k_d^{(B)}}(c)$.

Existence and Construction Problems

Question: Is there any public-key cipher meeting the five requirements described in the previous page?

Answer: Several public-key ciphers in the literature are believed to meet these requirements. But there is no proof.

How to construct a public-key cipher?

Use a problem that is believed to be hard to solve, e.g., the discrete logarithm problem.

Advantages and Disadvantages

- With a public-key cipher, a user does not need to share many keys with others. This is an advantage of public-key ciphers over private-key ciphers.
- The **disadvantage** of public-key ciphers is their performance in hardware and software, as no **efficient** and **secure** public-key cipher is known.

History of Public-Key Cryptography

History of Public-Key Cryptography (I)

- The idea of public-key cryptography was published by W. Diffie and M. Hellman, and independently by R. Merkle in 1976. It is regarded as a REVOLUTION in the history of cryptography!
- Admiral Bobby Inman, while director of the NSA, claimed that public-key cryptography had been discovered at NSA in the mid-1960s.
- The first (???) *documented* introduction of these concepts was given in 1970 by the Communications-Electronics Security Group, Britain's counterpart of NSA, in a classified report by James Ellis.

History of Public-Key Cryptography (II)

- The Knapsack public-key cipher was developed by Ralph Merkle and Martin Hellman in 1978, but was broken in 1982 by Shamir and Zipple.
- In the same year (1978), another public-key block cipher was invented by Ron **R**ivest, Adi **S**hamir, and Leonard **A**dleman. It is known as RSA. It is easy to understand and to implement, and is one of a few that are still regarded as secure. It is widely used in real-world security systems.
- Many other public-key ciphers have been proposed. Most of them have been broken.

Three Applications of Public-Key Ciphers

Application in Encrypting Data

Given a public-key cipher $(\mathcal{M}, \mathcal{C}, \mathcal{K}_e, \mathcal{K}_d, E_{k_e}, D_{k_d})$:

- Alice generates a key pair $\left(k_e^{(A)}, k_d^{(A)}\right)$, keeps the decryption key $k_d^{(A)}$ confidential, and publishes the encryption key $k_e^{(A)}$ and the encryption algorithm in a public directory.
- If Bob wants to send a message m to Alice, he finds Alice's encryption key $k_e^{(A)}$ and the encryption algorithm in the public directory, encrypts the message to get $c = E_{k_e^{(A)}}(m)$, and sends c to A.
- After receiving c, Alice uses her decryption key and computes

$$D_{k_d^{(A)}}(c) = D_{k_d^{(A)}}(E_{k_e^{(A)}}(m)) = m.$$

Remark: This is recommended for encrypting data of small size.

A Key Distribution Protocol Using a Public-Key Cipher

Application in Key Distribution

Session key: Two parties want to communicate using a one-key cipher for encryption. They need a session key for each session of communication.

Session key distribution with a public-key cipher

- Alice generates a session key k and then sends $E_{k_e^{(B)}}(k)$ to Bob.
- Bob uses his private key $k_e^{(B)}$ to decrypt $E_{k_d^{(B)}}(k)$ and recovers k.

Remark: The $E_{k_e^{(B)}}(k)$ is called a **digital envelope** and this protocol is called the **digital envelop protocol**, which is widely used in real-world security systems!

Remark: In this protocol, we assume that Alice and Bob exchanged their public keys beforehand.

A Digital Signature Scheme Using a Public-Key Cipher

The Digital Signature Scheme: Signing Process

Suppose that we have a hash function h and public-key cipher $(\mathcal{M}, \mathcal{C}, \mathcal{K}_e, \mathcal{K}_d, E_{k_e}, D_{k_d})$, where the range of h is a subset of the domain of D_{k_d}. Both are put in the public domain.

Alice can use her private key $k_d^{(A)}$ to **sign** messages.

- To sign a message m, Alice computes $h(m)$, which is called the **message digest**.
- She then uses her private key to sign on the message digest, obtaining $D_{k_d^{(A)}}(h(m))$, i.e., *her digital signature* on m. Then she sends the data $m||D_{k_d^{(A)}}(h(m))$ to the receiver Bob.

Properties of the Digital Signature

The text from Alice to Bob: $m||D_{k_d^{(A)}}(h(m))$

Property 1: The digital signature $D_{k_d^{(A)}}(h(m))$ has a fixed length for all messages m.

Property 2: The message m and the digital signature $D_{k_d^{(A)}}(h(m))$ have the following relationship:

$$h(m) = E_{k_e^{(A)}}\left(D_{k_d^{(A)}}(h(m))\right).$$

Thus, if the received text c by Bob was indeed created by Alice, and is partitioned into $c = c_1||c_2$, where c_2 has the same length as the digital signature, then

$$h(c_1) = E_{k_e^{(A)}}(c_2).$$

This relation is the basis of the digital signature verification process.

The Digital Signature Scheme: Signature Verification

- Bob partitions the received text c into two parts $c_1||c_2$, where c_2 has the same length as the digital signature.
- Then he uses Alice's public key $k_e^{(A)}$ to compute $E_{k_e^{(A)}}(c_2)$.
- Then he computes $h(c_1)$ (the hash function is public).
- Finally, he compares $h(c_1)$ with $E_{k_e^{(A)}}(c_2)$.

  If $h(c_1) = E_{k_e^{(A)}}(c_2)$, he accepts $c_1||c_2$ as a valid message with signature from Alice.

  - In this case, $c_1||c_2$ may be a modified or forged one. But the probability of this event should be very small if the public-key cipher and h are well designed.

  If $h(c_1) \neq E_{k_e^{(A)}}(c_2)$, he is sure that c_2 is not the digital signature on c_1 created by Alice.

Security Requirements of the Two Building Blocks

Question: How should we design the public-key cipher and the hash function so that the success probability of forging the signer's digital signature is very small?

Answer: The answer depends on feasible specific forgery attacks on the digital signature scheme.

Remark: We will answer the question above later.

Remark: This digital signature scheme is used in certain real-world security systems.

Applications of Public-Key Cryptography

Three types of applications:

Encryption, digital signature, key distribution.

Remark: In the digital signature scheme, the hash function h should be designed such that the range of h is a subset of the domain of D_{k_d}.