Random Number and Key Generation

Cunsheng Ding

HKUST, Hong Kong

Lecture 21: Cryptography

Contents

The Objectives of This Lecture

Necessity for secret keys and random numbers

Key management in real-world security systems

Design of key generation algorithms

Summary and concluding remarks

Appendix: Several real-world security systems

The Objectives of This Lecture

The Objectives of This Lecture

Key Management Issues Treated Before:

- *Key establishment* for one-key ciphers was studied in Lecture 14.
- *Public-key infrastructure* for public-key ciphers was covered in Lecture 19.

Objectives of This Lecture:

- We will study the random number generation and key generation for one-key ciphers.
- We will introduce other aspects of key management.

Necessity for secret keys and random numbers

Necessity for cryptographic keys and random numbers

- In many applications, we need the following security services:
  - data confidentiality, data integrity, mutual authentication, nonrepudiation
- Cryptographic keys, nonces and random numbers are needed, in order to provide these security services.

Classification of keys according to security services

- Encryption keys
  - for data confidentiality
- Authentication keys
  - for data origin authentication, data authentication (data integrity), sender authentication

Classification of keys according to lifetime

- Session keys (one-time keys)
- Long-term keys (e.g., passwords, master keys, root keys)

Classification of keys with respect to the protected data type

- Data protection keys
  - for protecting (encrypting or authenticating) communication data
- Master keys
  - for protecting session keys and private keys

Key management in real-world security systems

Key management

Key management encompasses techniques and procedures supporting the following:

- the initialization of systems users within a domain;
  - E.g., the generation of an initial password when someone applies for a HKUST account.
- the generation, distribution, and installation of **"keying materials"**;
  - E.g., a prime p and a primitive root α for the DH key exchange protocol are keying materials.
- the controlling of the use of keying materials;
- the update, revocation, and destruction of keying materials; and
- the storage, backup/recovery, and archiving of keying materials.

Comments on key management

- Key management is the **most complex** building block in real-world security systems.
- Most attacks are aimed at the key management level rather than cryptographic algorithms.
- In many real-word security systems, session keys are used. Different sessions of communications use different session keys.
- Management of public/private pairs is critical, as they may be used for session key distribution.
- Protecting the **root key** is the most difficult problem in certain security systems.

Key management

Attention:

- *Key establishment* for one-key ciphers was studied in Lecture 14.
- *Public-key infrastructure* for public-key ciphers was covered in Lecture 19.

In the remaining time, we will talk about random number generation and key generation.

Key generation for one-key ciphers

Question: How is a secret key for a one-key cipher generated?

Answer: It depends on the specific cipher.

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

Solution 1: If n is not large, say 128, flip a coin n times.

Solution 2: Use a "pseudorandom number generator" (in short, PRNG), which is a machine with finite memory.

Case II: Key bits must satisfy certain relations. See the cipher in the next slide.

In this case, there is no general approach. It differs from cipher to cipher.

Key generation for a one-key cipher: example

- The plaintext and ciphertext spaces: $\mathcal{M} = \mathcal{C} = \{0,1\}^*$.
- $\mathcal{K}$ consists 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 function 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!

What is a good cryptographic key?

Characteristics:

- The total number of elements in the key space should be large enough, to prevent a key exhaustive search.
  —— Similarly, your password should be long enough.
  - E.g., the key space is the set of all 128×128 invertible binary matrices.
  - E.g., the key space is $\mathbf{Z}_2^{128} = \{0,1\}^{128}$.
- The key should take all possible elements in the key space **more or less equally likely**, to make the key **unpredictable**.
  —— Similarly, your password should have this property.

Question: How does one generate such keys **effectively**?

Answer: This is a hard question. We will try to address this question in the next part.

Design of key generation algorithms

Key generation approaches

Flipping a coin a number of times

- Not practical for online applications.

Using a pseudorandom number generator (PRNG), which is any machine with finite memory

- Model: Input $\rightarrow$ PRNG algorithm $\rightarrow$ output
- The input to the PRNG algorithm may be a password, keyboard characters typed by a user, noises in our physical world.
- It is the most widely used approach to key generation and nonce generation.

Question: How does one design a **good** pseudorandom number generator?

How to design a PRNG?

The model for designing PRNGs:

$$\text{Input} \rightarrow \boxed{\text{PRNG algorithm}} \rightarrow \text{output}$$

Design issues:

- The design of the **input generating algorithm**.
  - The input generating algorithm is usually simple and efficient.
- The design of the PRNG algorithm.
  - The algorithm may be “stateful” or “stateless”.
- The output of the PRNG should be as unpredictable (random) as possible.

A PRNG is **stateful** if a memory device **has to** be used for storing intermediate computational results when computing the output. Otherwise, it is **stateless**.

Stateful pseudomrandom number generators

A finite state machine model for stateful PRNGs:

State at time unit i: X_i stored in a memory device, initial state (seed) X_0.

Input at time unit i: W_i.

Output at time unit i: $\text{output}_i := \text{output}(\text{input } W_i, \text{state } X_{i-1})$ for $i \geq 1$.

State transition: $X_i := \text{state}(W_i, X_{i-1})$ for $i \geq 1$.

Question and comment:

- How does one design the state transition function and the output function?
- How does one generate the input W_i?
  — See the next two pages for two examples.
- Two session keys generated with such PRNG have dependency because two states have dependency.

An example of stateful PRNGs (1)

Let p be a prime:

Initial state (called seed): $X_0 \in \mathbf{Z}_p$, stored in a memory device.

Input at time unit i: $W_i := i$ for $i \geq 1$.

Output at time unit i: $\text{output}_i := (2W_i) \oplus_p X_{i-1} = 2i \oplus_p X_{i-1}$ for $i \geq 1$.

State transition: $X_i := W_i \oplus_p X_{i-1} = i \oplus_p X_{i-1}$ for $i \geq 1$.

Comment: the randomness of the output X_i depends on the randomness of the seed X_0. See the next slide for detail.

Dependency between two outputs:

$$\begin{aligned} \text{output}_{i+1} &= 2(i+1) \oplus_p X_i = 2(i+1) \oplus_p i \oplus X_{i-1} = \\ &= 2i \oplus_p X_{i-1} \oplus_p i \oplus_p 2 \\ &= \text{output}_i \oplus_p i \oplus_p 2 \text{ for } i \geq 0. \end{aligned}$$

An example of stateful PRNGs (2)

Let p be a prime:

Initial state (called seed): $X_0 \in \mathbf{Z}_p$, stored in a memory device.

Input at time unit i: $W_i := i$ for $i \geq 1$.

Output at time unit i: $\text{output}_i := (2W_i) \oplus_p X_{i-1} = 2i \oplus_p X_{i-1}$ for $i \geq 1$.

State transition: $X_i := W_i \oplus_p X_{i-1} = i \oplus_p X_{i-1}$ for $i \geq 1$.

Proposition: The output X_{i+1} takes on each element of $\mathbf{Z}_p$ equally likely, provided that X_0 takes on each element of $\mathbf{Z}_p$ equally likely.

Proof: Using the formula in the previous slide and induction, we have

$$\text{output}_{i+1} = X_0 \oplus_p i(i+1)/2 \oplus_p 2(i+1) \text{ for } i \geq 0.$$

Another example of stateful PRNGs

Let E_k denote the encryption function of the AES, where $k \in \{0,1\}^{128}$. We identify $i \in \{0,1,\ldots,2^{128}-1\}$ with its binary expression $\overline{i} \in \{0,1\}^{128}$.

Initial state (called seed): $X_0 \in \{0,1\}^{128}$, stored in a memory device.

Input at time unit i: $W_i := \overline{i \bmod 2^{128}}$ for $i \geq 1$.

Output at time unit i: $\text{output}_i := E_k(W_i \oplus X_{i-1})$ for $i \geq 1$.

State transition: $X_i := W_i \oplus X_{i-1}$ for $i \geq 1$.

Comment: the randomness of the output X_i depends on the randomness of the seed X_0.
Comment: the initial value X_0 should be kept confidential in certain applications.
Comment: k can be a chosen constant.

Stateless pseudomrandom number generators

A type of stateless PRNGs: $\mathrm{output}(x, y)$**:**
Only an output function $\mathrm{output}(x, y)$ with two variables x and y.

- No state memory.
- Two outputs generated are independent of each other.
- One of the two inputs x and y should be as unpredictable as possible. This guarantees that the output $\mathrm{output}(x, y)$ is unpredictable.

Questions:

- How does one design the output function $\mathrm{output}(x, y)$?
- How does one generate the inputs x and y?

— An example will be given in the next slide.

A stateless generator (not practical)

A function with two variables:

- $f(x, y) = x \oplus_p y$ for $(x, y) \in \mathbf{Z}_p \times \mathbf{Z}_p$.

The stateless generator:

Input: (x, y), a seed element $x \in \mathbf{Z}_p$, another element $y \in \mathbf{Z}_p$ typed by a user.

Output: $f(x, y) = x \oplus_p y$.

Remark: The unpredictability of $f(x, y)$ depends on that of y and x.

Remark: The generator is **not autonomous** as a human user must be involved.

Comparison of stateful and stateless generators

Advantages & disadvantages between stateful & stateless generators:

- The outputs of a stateless generator are independent, but their randomness depends on the randomness of the inputs. It is likely that the generator is not **autonomous**, and hence may not be practical for online applications.
  - For example, the stateless generator on Slide No. 25.
- An advantage of a stateful generator is that it could be **autonomous** and thus very practical. But a disadvantage is that two outputs may have dependency.
  - For example, the stateful generator on Slide No. 21.

Summary and concluding remarks

Summary

- Key management is the most complex part in real-world security systems.
- Key generation is crucial to security.
- Designing a **proper** key generation algorithm is necessary in security systems. Both performance and security are important.
- More attention should be given to key generation and key distribution.

Appendix: Several real-world security systems

Several important real-world security systems

The objectives of this appendix are the following:

- Introduce briefly the most important real-world security systems.
- Introduce the types of key and random numbers that are used in these systems.
- Realize the importance of key generation in real-world security systems.
- Realize the importance of cryptography in cybersecurity and computer security.

Details of these real-world security systems can be found in the PG course (COMP5631) website:
https://www.cse.ust.hk/faculty/cding/COMP5631/

Two systems for email communication security

PGP (Pretty Good Privacy)

- Widely used in the 90's, not as popular as before.
- Supported in the following email systems:
  Pegasus, Simeon, Eudora, Outlook, Herald, Pine, ELM
- Supported also in Gmail currently.

SMIME (Secure/Multipurpose Internet Mail Extensions)

- Originally developed by RSA Data Security Inc.
- Supported in web mail systems and thus getting more popular.

Types of keys and random numbers needed in PGP and SMIME

- Session key for data confidentiality.
- Public-private key pair for data integrity, sender nonrepudiation, sender authentication, and session key distribution.
- A random number for the DSS if DSS is used for the creation of the digital signature.
- A random number for the ElGamal cipher if it is used for session key distribution.
- Random numbers are needed for session key generation in PGP.

Networking security

IP Security

- Networking layer security protocol.
- Sender authentication, data integrity, data confidentiality, anti-replay, limited traffic flow confidentiality.
- It can be used to establish VPNs.

Keys and random numbers needed in IP Security:

- Secret keys for sender authentication, data integrity; data confidentiality, limited traffic flow confidentiality.
- Public-private key pairs for session key distribution and mutual authentication.
- Random numbers for mutual authentication and computing **“keying materials”.**

World Wide Web security (1)

SSL and TLS

- Transport layer security protocols.
- Client-server authentication, data confidentiality, data integrity.
- Mainly used for establishing a secure tunnel through the Internet.
- They can be used to build VPNs.

Keys and random numbers

- Session keys (confidentiality and data integrity, also authentication), public-private key pairs for session key distribution and authentication.
- Random numbers for authentication and keying materials.

World Wide Web security (2)

Hypertext Transfer Protocol Secure (HTTPS)

- a protocol for secure communication over a computer network, with especially wide deployment on the Internet.
- Technically, it is not a protocol in and of itself; rather, it is the result of simply layering the Hypertext Transfer Protocol (HTTP) on top of the SSL/TLS protocol, thus adding the security capabilities of SSL/TLS to standard HTTP communications.

Keys and random numbers

- The same as those in SSL/TLS.

Secure connection (1)

The Secure Shell (SSH)

- Developed in Finland in 1995.
- Establishing a secure tunnel between a client and server.
- Client-server authentication, data confidentiality, data integrity.

Keys and random numbers

- Session keys (encryption and authentication).
- Public-private keys and random numbers.

Secure connection (2)

Virtual Private Network (VPN)

- Virtual Private Network is a type of private network that uses public telecommunication, such as the Internet, instead of leased lines to communicate.
- Became popular as more employees worked in remote locations.
- Sender authentication, data confidentiality and integrity.

Keys and random numbers used in VPNs

- Encryption and authentication keys.
- Public-private keys, and random numbers.

Secure connection (3)

VPN tunnelling technologies

- IPSec (end-to-end, transport mode)
  - IKE Internet Key Exchange for key management
  - ESP (Encapsulating Security Payload) and AH subprotocols for providing security services
- PPTP (gateway-to-gateway)
- L2TP (gateway-to-gateway)
- SSL (application-to-application)

Keys and random numbers are used in these tunnelling protocols:

- Encryption and authentication keys, and random numbers.