# Computer Security

**Cunsheng Ding, HKUST**

**COMP4631**

# Lecture 13: Hash and Keyed Hash Functions

## Main Topics of this Lecture

1. Hash functions and general design requirements.
2. Keyed hash functions and general design requirements.
3. The HMAC

# Part I: Hash Functions

# Hash Functions

**Formal definition:** A **hash function** $h$ is a mapping from the set of all finite strings of characters from an alphabet $\mathcal{A}_1$ to a string of characters from an alphabet $\mathcal{A}_2$ with fixed length.

For any $x$, $h(x)$ is called the **hash value**, or **message digest**.

**Remark:** A hash function $h$ is publicly known for many applications.

# The Hash Function in the Digital Signature Scheme

**The digital signature scheme:** the sender Bob sends $m||D_{k_d^{(B)}}(f(m))$ to the receiver Alice.

**The public-key cipher:** it should be secure, otherwise one may be able to derive the private key from the public key of Bob, and then forge Bob's digital signature.

**Question:** How to design the hash function $f$, so that it is difficult for someone to forge Bob's digital signature?

# Attacks on the Digital Signature Scheme

**An attack:** After receiving $m||D_{k_d^{(B)}}(f(m))$, Alice finds a different message $m'$ such that $f(m) = f(m')$. Then Alice forges Bob's digital signature on $m'$ as $m'||D_{k_d^{(B)}}(f(m))$.

**Collision:** Any pair of distinct messages $m$ and $m's$ such that $f(m) = f(m')$ is called a collision. Hence, it should be computationally infeasible to find a collision of the hash function $f$.

**One-way property:** Given $m$, it is easy to compute $f(m)$. But given a value $v$, it is hard to find an $m$ such that $f(m) = v$.

**Requirement:** $f$ should be one-way. Why?

# Requirements for Hash Functions

**Remark:** Hash functions for different applications may be required to have different properties. Below are some common requirements.

1. $h(x)$ is easy to compute for any given $x$, making both hardware and software implementation practical.

2. For any given value $v$, it is computationally infeasible to find $x$ such that $h(x) = v$. This is the **one-way property**. [E.g., the diginal signature application, where only the message is encrypted.]

3. For any given block $x$, it is computationally infeasible to find $y$ such that $h(x) = h(y)$. This is the **weak collision resistance property**. [E.g., the diginal signature application]

# Requirements for Hash Functions (Continued)

**Requirements implied by the ones listed in the previous page:**

- The size of the hash value $h(x)$ should be large enough (256 bits recommended), in order to thwart the brute-force attack.
- $h(x)$ should take on all the finite strings of fixed length as equally likely as possible. That is, the hash vaues are as uniformally distributed as possible.

# The MD5 Hash Function

- It was designed in 1991 by Ron Rivest at MIT.
- The size of the hash values of MD5 is 128 bits. For example,

```
MD5("The quick brown fox jumps over the lazy dog")
  = 9e107d9d372bb6826bd81d3542a419d6
```

- It was widely used in real security systems.
- In 2004, collisons of MD5 were found. This may not be a threat for real applications.

# The SHA-1, SHA-2 and SHA-3 Hash Functions

- SHA-1 was designed in 1995 by the NSA.
- The size of the hash values of SHA-1 is 160 bits. For example,

```
SHA1("The quick brown fox jumps over the lazy dog")
 = 2fd4e1c6 7a2d28fc ed849ee1 bb76e739 1b93eb12
```

- It is widely used in real security systems.
- In 2006, collisons of SHA-1 were found. This may not be a threat for real applications.
- So new versions of SHA were developed: SHA-256, SHA-224, SHA-512, SHA-384. They are called SHA-2.
- SHA-3 (called Keccak) was announced in Oct. 2012 by NIST.

# Part II: Keyed Hash Functions

# Keyed Hash Functions

**Formal definition:** A **keyed hash function** $h_k$ is a mapping from the set of all finite strings of characters from an alphabet $A$ to a string of characters from an alphabet $B$ with fixed length, where $k$ is a secret parameter from a space $\mathcal{K}$.

For any $x$, $h_k(x)$ is called the **hash value** or **message authentication code (MAC)**.

**Applications:** Authentication.

# Design Requirements for Keyed Hash Functions

**Authentication protocol using a keyed hash function:** Suppose that Alice and Bob share a secret key for a keyed hash function. The protocol works as follows:

$$\text{Alice} \implies m||h_k(m) \implies \text{Bob}$$

Suppose that the enemy has total control of the communication channel. Then we have the following security requirement.

**Requirement:** Given a number of pairs $(m, h_k(m))$, it should be computationally hard to find out the secret key $k$.

**Attention:** This protocol is used in many real-world security systems for sender authentication and data integrity checking purpose.

# The First Example of Keyed Hash Functions

**Example:** Let $h$ be a hash function with hash value of 256 bits. Define $h_k(m) := h(m) \oplus k$, where $k$ is a secret key of 256 bits. Then $h_k$ is a keyed hash function.

**Security:** Does $h_k$ meet the requirement in the previous slide?

## The Second Example of Keyed Hash Functions

**Example:** Let $E_k$ be the encryption transformation of one-key cipher and $h$ be a hash function. Then $h_k := E_k \circ h$ is a keyed hash function, where $\circ$ denotes the function composition.

**Security:** It should have good properties if both the one-key cipher and the hash function are well designed.

**Remark:** Left to students.

# Part III: the HMAC

# HMAC: A Specific Construction

- $h$ a hash function, with $n$-bit hash value.
- $b$ is a chosen positive integer and $8|b$.
- $K$ is the secret key with size at most $b$ bits.
- $\overline{K}$ is $K$ padded with 0's on the left so that the result is $b$ bits in length.
- ipad = 00110110 repeated $b/8$ times.
- opad = 01011100 repeated $b/8$ times.

$$\mathrm{HMAC}_K(m) = h\{(\overline{K} \oplus \mathrm{opad})||h[(\overline{K} \oplus \mathrm{ipad})||m]\}.$$

## Some Questions about the Design of HMAC

- What is the purpose of using the two constant binary strings?
- Why are the ipad and opad designed in that way?
- Why $h$ is used twice?

# Security of the HMAC

**Conclusion:** It depends in some way on the cryptographic strength of the underlying hash function. For details, see:

M. Bellare, R. Canetti and H. Krawzyk, *Keying hash functions for message authentication,* Advances in Cryptology – Crypto' 96, LNCS 1109, Springer-Verlag, 1996.