Cryptography and Security

Cunsheng DING

HKUST, Hong Kong

Version 3

Lecture 11: 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, which is a method for converting a hash function and a key into a keyed hash function.

Part I: Hash Functions

Hash Functions

Formal definition: A **hash function** h is a function from the set of all finite strings of characters from an alphabet $\mathcal{A}_1$ to the set of all strings 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 h in the Digital Signature Scheme

The digital signature scheme in Lecture 7: It has two building blocks, a hash function h and a public-key cipher, where h is in the public domain.

- Alice sends $m||D_{k_d^{(A)}}(h(m))$ to Bob, where $k_d^{(A)}$ is Alice's private key.
- After receiving a message from Alice, Bob will do signature verification.

Bob can try to forge Alice's signature as follows:

1. Find a different message m' such that $h(m) = h(m')$ (a weak collision).
2. If this is successful, Bob claims that Alice sent him $m'||D_{k_d^{(A)}}(h(m))$.

Requirement: For any given message x, it is computationally infeasible to find y such that $h(x) = h(y)$ (**weak collision resistance property**).

Security Requirements for Hash Functions

Hash functions for different security systems may be required to have different properties. The following are some common security requirements.

1. $h(x)$ is easy to compute for any given x, making both hardware and software implementation practical.
2. For any given x, it is computationally infeasible to find y such that $h(x) = h(y)$. This is the **weak collision resistance property**. [E.g., the digital signature scheme]
3. 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 digital signature scheme, Unix password file.]

Exercise: Justify why the h used in the digital signature scheme should have the one-way property.

Security 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 values are as uniformally distributed as possible.

Remark: The two properties above do not imply the weak collision resistance property or the one-way property.

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, "strong collisions" of MD5 were found. This may not be a threat for certain applications.
  - A **strong collision** is a pair (m, m') of messages with $h(m) = h(m')$, where m and m' are of free choice.

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 was widely used in real security systems.
- In 2006, strong collisions of SHA-1 were found. This may not be a threat for certain 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 adopted as a standard in 2015 by NIST.

Online Demo of Some Hash Functions

In the following URL, you find the demo of many hash functions:

`https://8gwifi.org/MessageDigest.jsp`

Part II: Keyed Hash Functions

Keyed Hash Functions

Formal definition: A **keyed hash function** h_k is a function from the set of all finite strings of characters from an alphabet A to the set of all strings 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: data authentication (data integrity) and data origin authentication in real-world security systems.

Attention: In many applications the algorithm $h_k(x)$ (i.e., the internal structure) is known but the key k is confidential to attackers.

Design Requirements for Keyed Hash Functions in a Protocol

Authentication protocol using a keyed hash function: Suppose that Alice and Bob share a secret key for a keyed hash function.

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

Suppose that the enemy has total control of the communication channel.

Security services: This protocol is used in many real-world security systems for data integrity and data origin authentication. Why? How?

Security requirements of h_k used in this protocol:

1. Given some pairs $(m_i, h_k(m_i))$, it should be computationally infeasible to find out the secret key k. Why?
2. Given a pair $(m, h_k(m))$, it should be computationally infeasible to find out a different m' such that $h_k(m) = h_k(m')$ without knowing k. Why?

The First Example of Keyed Hash Functions

Example: Let h be a hash function with hash values 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: Assume that the hash function h used in the system is known to everyone. Does h_k meet the requirements in the previous slide?

Remark: This is a simple method for converting a hash function h and a secret key k into a keyed hash function h_k.
But it is a very bad construction, as the key bits and the hash value bits have a very simple relation.

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: The keyed hash function h_k is believed to meet the two requirements in Slide No. 12 if its two building blocks have the following properties:

- The one-key cipher is computationally secure.
- The hash function h has the weak collision resistant property.

Remark: Any collision of h is a collision is h_k. This justifies the second requirement above.

Part III: the HMAC

A method for converting a hash function and
a key into a keyed hash function

HMAC: A Specific Construction

- h a hash function, with n-bit hash value.
- b is a chosen positive integer and $8|b$.
  - The upper bound on the key size in bits.
- 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.

Online Demo of the HMAC

In the following URL, you find the demo of the HMAC with some hash functions:

`https://8gwifi.org/hmacgen.jsp`