Cryptography

Cunsheng Ding, HKUST

COMP3631

Lecture 17: Hash Functions

Main Topics of This Lecture

1. Hash functions and general design requirements.
2. Several hash functions (MD5, SHA-1, SHA-2, SHA-3).

Part I: Hash Functions

A Recall of Hash Functions

Formal definition: A **hash function** h is a function from a **subset** of 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 some applications.

Remark: The security requirements of hash functions used in the Digital Signature Scheme were discussed in Lecture 16.

- The hash function should have the one-way property.
- The hash function should have the weak collision resistance property.

Weak collisions and **strong collisions** were defined in Lecture 16.

Requirements for Cryptographic Hash Functions

Remark: Hash functions for different security systems may be required to have different properties. Below are some important 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 protection of the password file in some operating systems and the digital signature scheme in Lecture 16.]

3. For any given message 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 in Lecture 16]

Remarks on Hash Functions

- The design of cryptographic hash functions is similar as the design of the encryption and decryption function of one-key block ciphers.
  - The hash function should have good confusion and diffusion.
- Hash functions are used in many real-world security systems.
- Hash functions used in other applications such as databases have different design requirements.

Part II: Several Hash Functions

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-world security systems.
- In 2004, strong collisions of MD5 were found. This may not be a threat for certain applications (e.g., the digital signature scheme in Lecture 16).

The SHA-1 and SHA-2 Hash Functions

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

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

- It was widely used in real-world 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 (Keccak)

1. The NIST announced SHA-3 in October 2012 and made it a federal information processing standard in 2015.
2. SHA-3 is not meant to replace SHA-2, as no significant attack on SHA-2 has been demonstrated.
3. The internal structure of SHA-3 is different from that of SHA-1 and SHA-2.
4. SHA-3 has different variants, including:

   SHA3-224, SHA3-256, SHA3-384, SHA-512 whose output length in bit is indicated by the digit.

Further information on SHA-3 can be found in

`https://en.wikipedia.org/wiki/SHA-3#Examples_of_SHA-3_variants`

Part III: An Outline of SHA-1

Attention: The input message m for SHA-1 is restricted to have at most $2^{64} - 1$ bits (about 17179869184 GB).

SHA-1 Description

Step 1: Append padding bits

- The message is padded so that its length in bits is $\equiv 448 \bmod 512$. Padding is always added, even if the message is already of the desired length. Thus the number of padding bits is in the range of 1 to 512.
- The padding rule: 1 followed by the number of necessary 0's.

Remark: 448 is chosen here because another 64 bits will be appended in Step 2. Note that $448 + 64 = 512$.

SHA-1 Description

Step 2: Append the binary representation of the length of the original message

- Since the bit length of the original message $< 2^{64}$, append the 64 bits of the binary representation of the bit length of the original message after Step 1.
- This block of 64 bits is treated as an unsigned 64-bit integer (most significant byte first).

Remark: After the padding of Step 2, the length of the new message is a multiple of 512 bits.



SHA-1 Description

Step 3: Initialize the MD (message digest) buffer

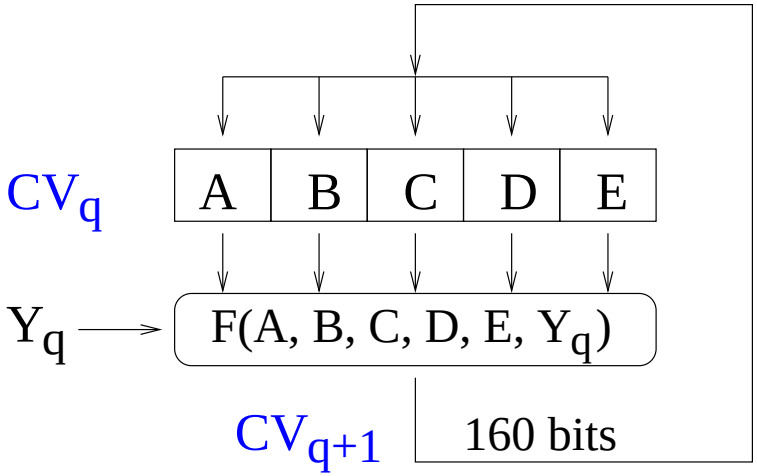
- A 160-bit buffer is used to hold intermediate and final results of the hash function. The buffer can be represented as five 32-bit registers A, B, C, D, E.
- These registers are initialized to the following 32-bit integers (hexadecimal values):

$$A = 67452301,\ B = efcdab89,\ C = 98badcfe,$$
$$D = 10325476,\ E = c3d2e1f0$$

SHA-1 Description

Step 4: Process padded message in 512-bit blocks

Let $Y_{L-1} \cdots Y_1 Y_0$ be the message after the padding, where Y_i is a 512-bit block. Then SHA-1 computes the hash value as follows.

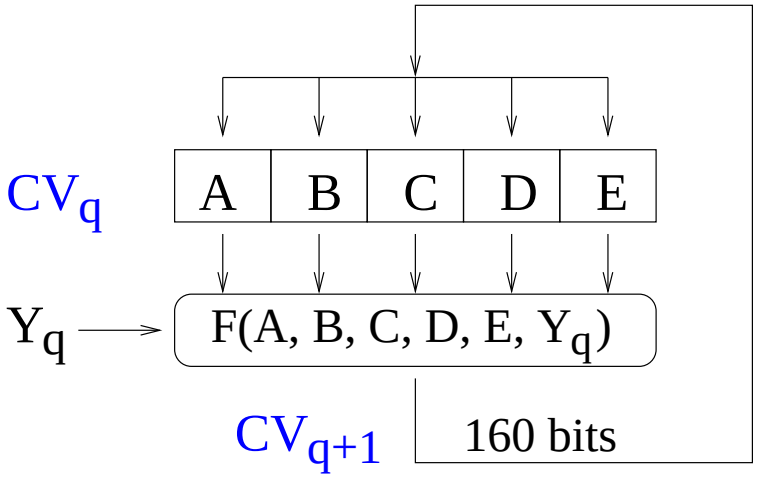


Where $F(A, B, C, D, E, Y_q)$ is a function to be specified later, CV_q is the content of the buffer at time unit q, and CV_{q+1} is the new value for replacing CV_q. The final content of the buffer is the hash code.

SHA-1 Description

Step 5: Output

After processing the last block of message, the content of the buffer is the hash code. That is, $MD = CV_L$.



Remark: The core part of SHA-1 is the function

$$F(A, B, C, D, E, Y_q).$$

SHA-1 Description: Function $F(A, B, C, D, E, Y_q)$

The constants K_i:

step number	hexadecimal	where from
$0 \le t \le 19$	$K_t = 5a827999$	$[2^{30}\sqrt{2}]$
$20 \le t \le 39$	$K_t = 6ed9eba1$	$[2^{30}\sqrt{3}]$
$40 \le t \le 59$	$K_t = 8f1bbcdc$	$[2^{30}\sqrt{5}]$
$60 \le t \le 79$	$K_t = ca62c1d6$	$[2^{30}\sqrt{10}]$

Remark: These constants are needed inside the function $F(A, B, C, D, E, Y_q)$, and are called **magic numbers**.

SHA-1 Description: Function $F(A, B, C, D, E, Y_q)$

The $W[0..79]$ computed from each block Y_q:

For each message block Y_q, 80 words W_i of 32 bits are computed:

- Define $W_0||W_1||\cdots||W_{15} = Y_q$, i.e., divide Y_q into 16 blocks of 32 bits.
- For $16 \leq t \leq 79$, W_t is computed by

$$W_t = S^1(W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3})$$

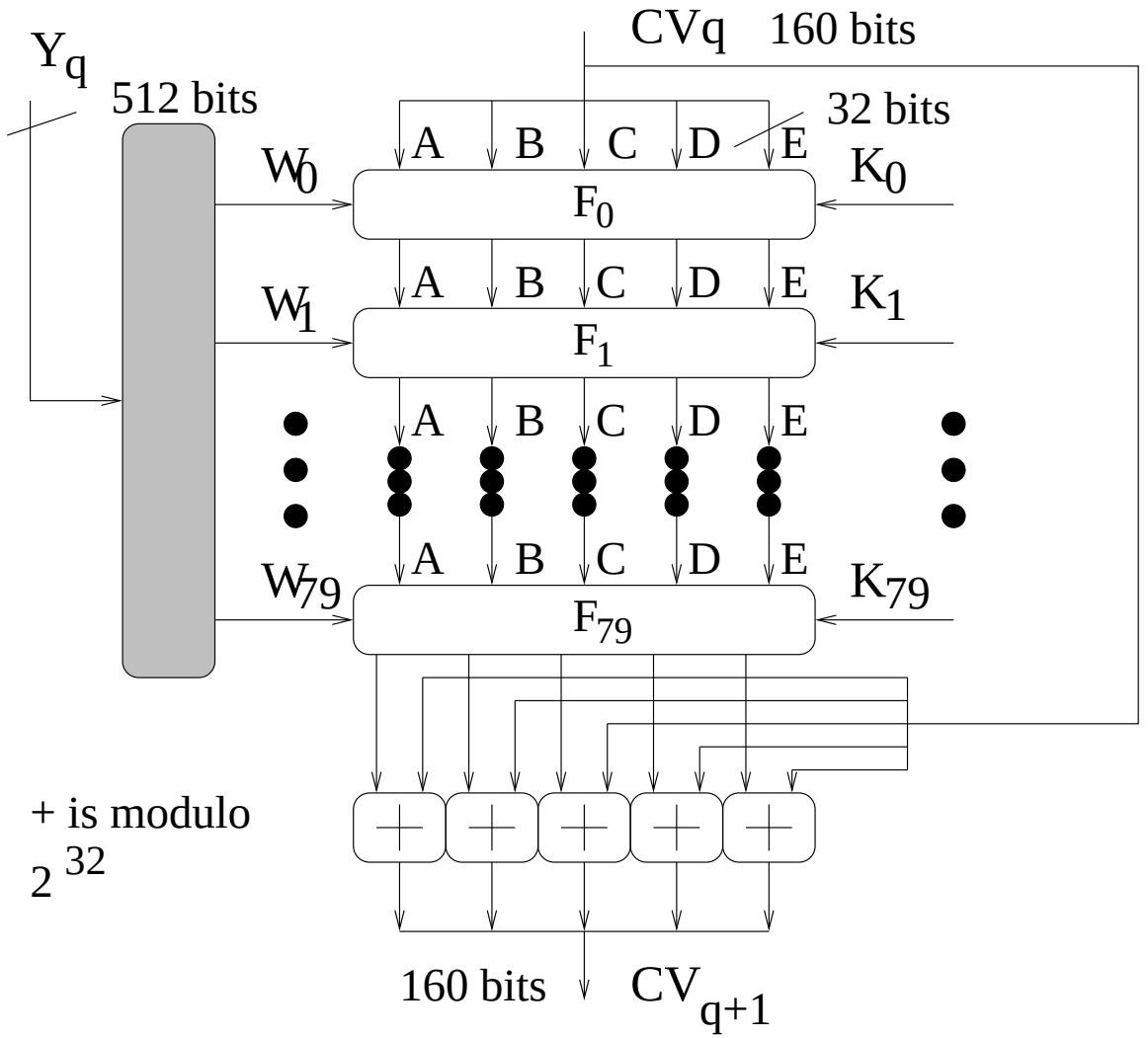
recursively, where $S^i(x)$ is the circular left shift of the 32-bit word x for i positions. For example,

$$S^1(x_1x_2 \ldots x_{32}) = x_2x_3 \ldots x_{32}x_1$$

and

$$S^2(x_1x_2 \ldots x_{32}) = x_3x_4 \ldots x_{32}x_1x_2.$$

SHA-1 Description: Function $F(A, B, C, D, E, Y_q)$



Remarks on SHA-1

- Details on the functions F_i can be found in https://www.cse.ust.hk/faculty/cding/COMP5631/slides.html
- Note that the length of an input message in bit to SHA-1 is limited at $2^{64} - 1$. Why?
- In practice, you do not have a message with bit length more than $2^{64} - 1$, as there is no space to store such a piece of data!

Online Demo of Some Hash Functions

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

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

Appendix: Hash Functions from a Block Cipher

Hash Functions Using a Block Cipher: 1st Constr.

Building block: Given a one-key block cipher

$$(\mathcal{M}, \mathcal{C}, \mathcal{K}, E_k, D_k),$$

where E_k maps a block of n bits into a block of n bits, and each key k has l bits.

Construction: see the next page.

Hash Functions using a Block Cipher: 1st Constr.

Construction: For a given message m, pad m with 0...01 on the left side so that the length of the modified message m' is a multiple of ℓ. Even if the length of m is a multiple of ℓ, padding is also done.

Then divide m' into blocks of length ℓ: $m' = m_t \cdots m_2 m_1$. The hash value H is computed as follows:

$$\begin{aligned} H_0 &= \text{fixed initial value of } n \text{ bits} \\ H_i &= E_{m_i}(H_{i-1}) \\ H &= H_t \end{aligned}$$

Question: If n and l are large enough, and the block cipher is well designed, does this hash function have the weak collision resistance property and the one-way property?

Hash Functions using a Block Cipher: 2nd Constr.

Building block: Given a one-key block cipher

$$(\mathcal{M}, \mathcal{C}, \mathcal{K}, E_k, D_k),$$

where E_k maps a block of n bits into a block of n bits, and each key k has l bits.

Construction: see the next page.

Hash Functions Using a Block Cipher: 2nd Constr.

Construction: For a given message m, pad m with 0...01 on the left side so that the length of the modified message m' is a multiple of ℓ. Even if the length of m is a multiple of ℓ, padding is also done.

Then divide m' into blocks of length ℓ: $m' = m_t \cdots m_2 m_1$. The hash value H is computed as follows:

$$\begin{aligned} H_0 &= \text{fixed initial value of } n \text{ bits} \\ H_i &= E_{m_i}(H_{i-1}) \oplus H_{i-1} \\ H &= H_t \end{aligned}$$

Question: If n and l are large enough, and the block cipher is well designed, does this hash function have the weak collision resistance property and the one-way property?