Cryptography and Security

Cunsheng DING

HKUST, Hong Kong

Version 3

Lecture 12: The Secure Hash Algorithm

Main Topics of This Lecture

1. Some information of SHA-1.
2. Explanation of the design ideas of SHA-1.
3. A brief introduction of SHA-2 and SHA-3.

Brief Information on SHA-1

- SHA was designed by the NIST, and published as a federal information processing standard in 1993. It is based on the MD4 algorithm, and its design closely models MD4.
- SHA-1 is a revised version of SHA, and has a 160-bit hash value. The maximum input size is 2^{64} bits, and the input is processed in 512-bit blocks.
- It is used in the Digital Signature Standard (DSS), and Pretty Good Privacy (PGP), and other security systems.

Design Considerations: Security & Performance

- It should be computationally infeasible to find a preimage given a hash value, and to find a collision.

  To this end, you need to have confusion and diffusion in a hash algorithm. To have confusion, nonlinear functions are need. To have diffusion, linear functions are needed.

- Most hash algorithms are software-oriented and are implemented in general-purpose computers. Such a hash algorithm must make full use of the processor size. So the operations should work on words of 32 bits for the implementation in a 32-bit processor machine.

Binary Representation of Numbers

$$\begin{aligned} i &= i_{t-1} i_{t-2} \cdots i_1 i_0 \\ &= i_{t-1} 2^{t-1} + i_{t-2} 2^{t-2} + \cdots + i_1 2 + i_0, \end{aligned}$$

where each $i_j \in \{0, 1\}$.

Example: $1011 = 2^3 + 2^1 + 2^0 = 11$.

Decimal Representation of Numbers

$$\begin{aligned} i &= i_{t-1} i_{t-2} \cdots i_1 i_0 \\ &= i_{t-1} 10^{t-1} + i_{t-2} 10^{t-2} + \cdots + i_1 10 + i_0, \end{aligned}$$

where each $i_j \in \{0, 1, ..., 9\}$.

Example: $7019 = 7 \cdot 10^3 + 1 \cdot 10^1 + 9 \cdot 10^0$.

Hexadecimal Representation of Numbers

Define

$$a = 10, b = 11, c = 12, d = 13, e = 14, f = 15.$$

Hexadecimal representation:

$$\begin{aligned} i &= i_{t-1}i_{t-2}\cdots i_1 i_0 \\ &= i_{t-1}16^{t-1} + i_{t-2}16^{t-2} + \cdots + i_1 16 + i_0, \end{aligned}$$

where each $i_j \in \{0, 1, ..., 9, a, b, c, d, e, f\}$.

Example:

$$\begin{aligned} 9ad1 &= 9 \times 16^3 + a \times 16^2 + d \times 16 + 1 \times 16^0 \\ &= 39633 \end{aligned}$$

SHA-1 Description

Step 1: Append padding bits

- The message is padded so that its length $\equiv 448 \bmod 512$. 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 length of the original message

- Since the length of the original message $< 2^{64}$, append the 64 bits of the binary representation of the 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.

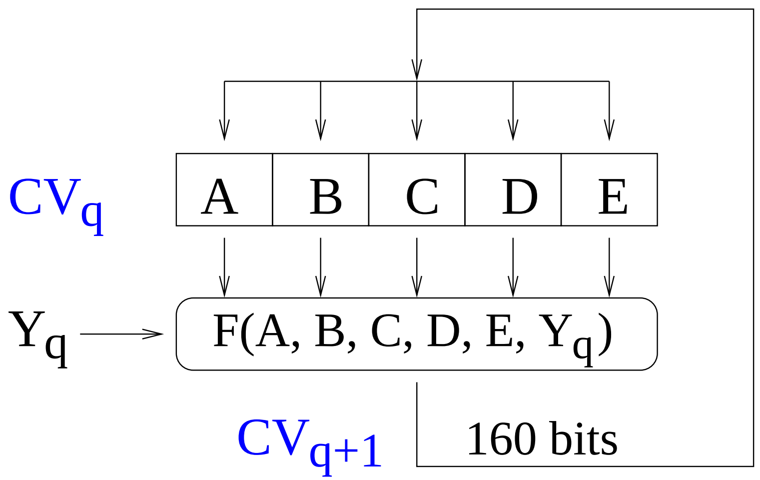
- These registers are initialized to the following 32-bit integers (hexadecimal values):

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

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.

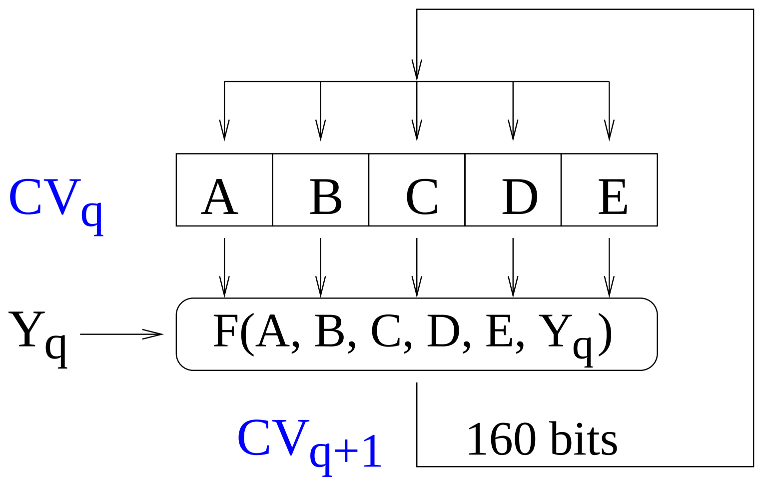


Where $F(A, B, C, D, E, Y_q)$ is a function to be specified later, CV_q is the content of the buffer at time unit q, and CV_{q+1} is the new value for replacing CV_q. The final content of the buffer is the hash code.

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.

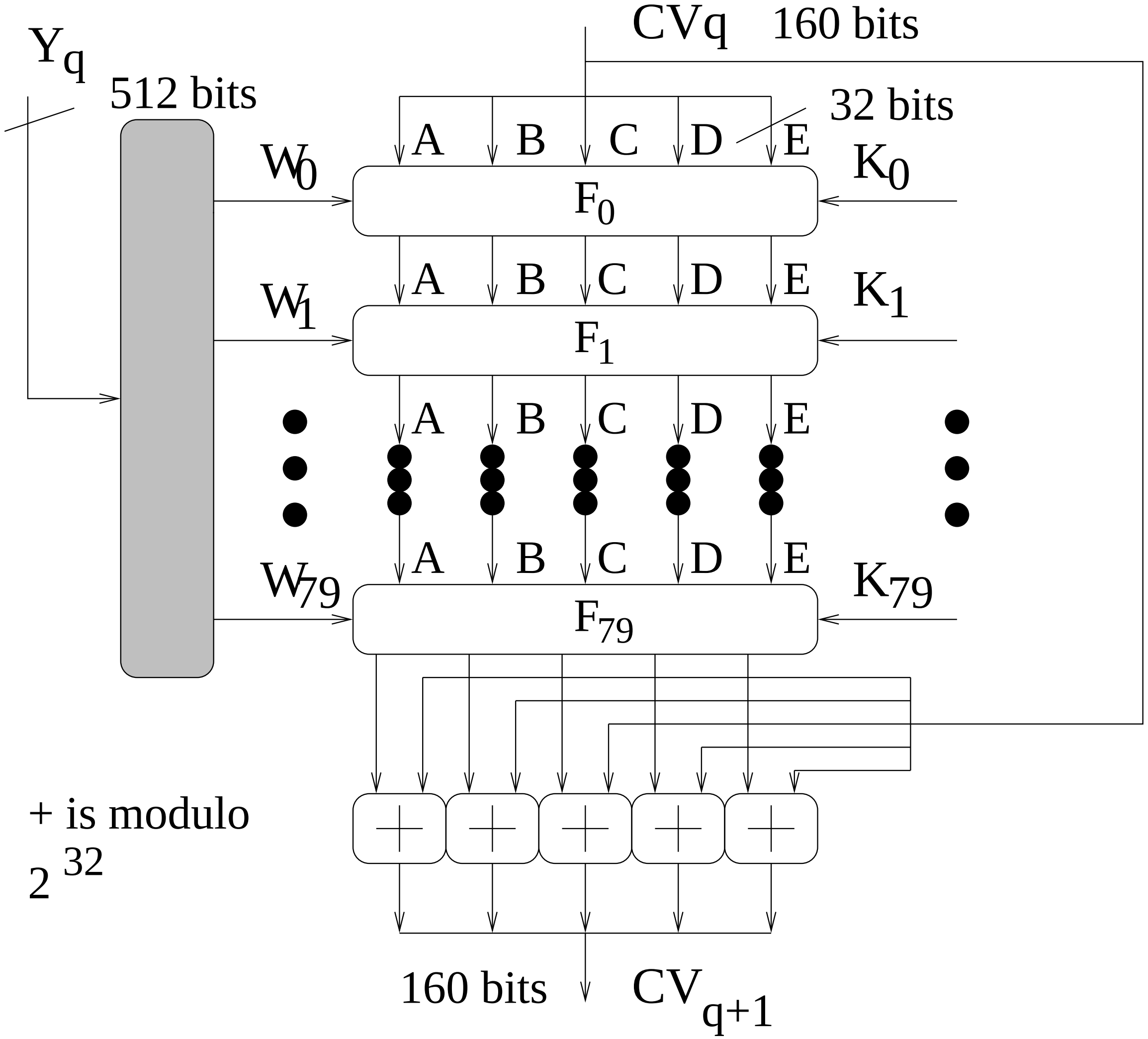
SHA-1 Description: Function $F(A, B, C, D, E, Y_q)$

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

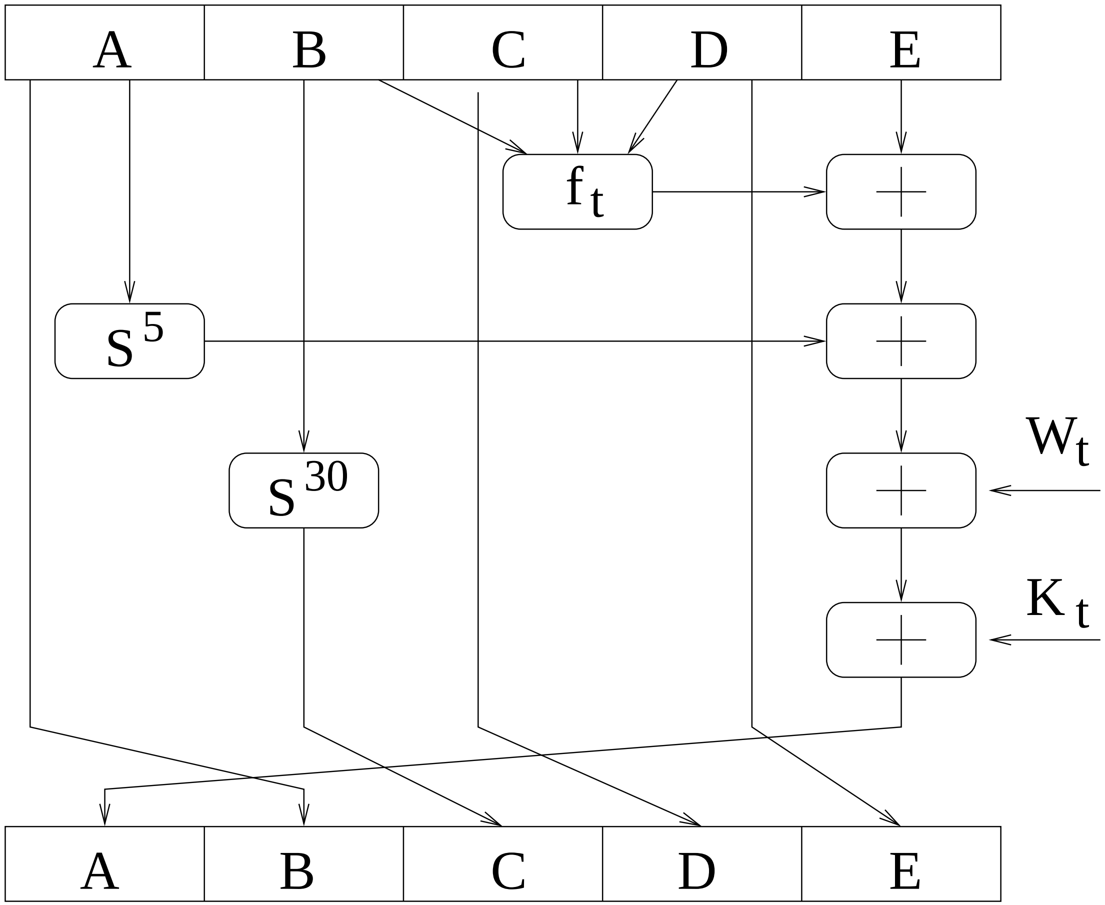
Comments and design ideas: It would be desirable to have all the 80 words $W_0, \cdots, W_{79}$ take as many possible 32-bit strings as possible. Hence linear recurrence approach is a good choice.

The linear operation $S^1(x)$ is used as it will not destroy the uniform distribution of W_i, but change the exact linear recurrence formula.

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



SHA-1 Description: Function F_t



The function F_t, where f_t will be described next.

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

The functions f_t:

step number	function	function value
$0 \leq t \leq 19$	$f_t(B, C, D)$	$(B \wedge C) \vee (\overline{B} \wedge D)$
$20 \leq t \leq 39$	$f_t(B, C, D)$	$B \oplus C \oplus D$
$40 \leq t \leq 59$	$f_t(B, C, D)$	$(B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
$60 \leq t \leq 79$	$f_t(B, C, D)$	$B \oplus C \oplus D$

Remark: $\vee$ = OR, $\wedge$ = AND, $^{-}$ = NOT, $\oplus$ = EXOR.

Remark: $B \wedge C$ is the bitwise logical AND.

Understanding the Design Ideas

1. The 80 functions use only three distinct functions. What is the idea of using only three distinct functions?
2. In the single step operation, why four different kinds of operations $\wedge$, $\vee$, $\oplus$, and $+_{2^{32}}$ are used?
3. Why shift operations are used in the single step operation?
4. What is the purpose of having 80 steps of operations?

Security and other Issues

1. It is derived from MD4 (one of the hash algorithms designed by Ron Rivest). But no design criteria are publically known.

2. It is simple to describe and simple to implement and do not require large programs or substitution tables.

3. Strong collisions of SHA-1 and MD5 were found. But for any message x, it is not known how to find a y such that $H(x) = H(y)$. So it is not really a threat for real applications. But it is time to design new hash functions.

Other Variants of SHA

1. In 2002, NIST published three additional hash functions in the SHA family, each with longer digests, collectively known as SHA-2. The individual variants are named after their digest lengths (in bits): "SHA-256", "SHA-384", and "SHA-512".

2. In February 2004, NIST published an additional variant, "SHA-224", defined to match the key length of two-key Triple DES.

Other Variants of SHA

1. The operations in SHA-256 and SHA-512 work on 32-bit and 64-bit words, respectively.
2. SHA-256 and SHA-512 use different shift amounts and additive constants, but their structures are otherwise virtually identical, differing only in the number of rounds.
3. SHA-224 and SHA-384 are simply truncated versions of the first two, computed with different initial values.

Other Variants of SHA

Parameters of SHA variants in bits are listed below.

Algorithm	Output	Internal state	Block	Word	Passes
SHA-1	160	160	512	32	80
SHA-256/224	256/224	256	512	32	64
SHA-512/384	512/384	512	1024	64	80

SHA3 (Keccak)

1. NIST announced SHA-3 in October 2012 and made it an 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:

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

Further information on SHA3 can be found in

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