Lecture 13 Secure Hash Algorithm (SHA)

Basic Knowledge

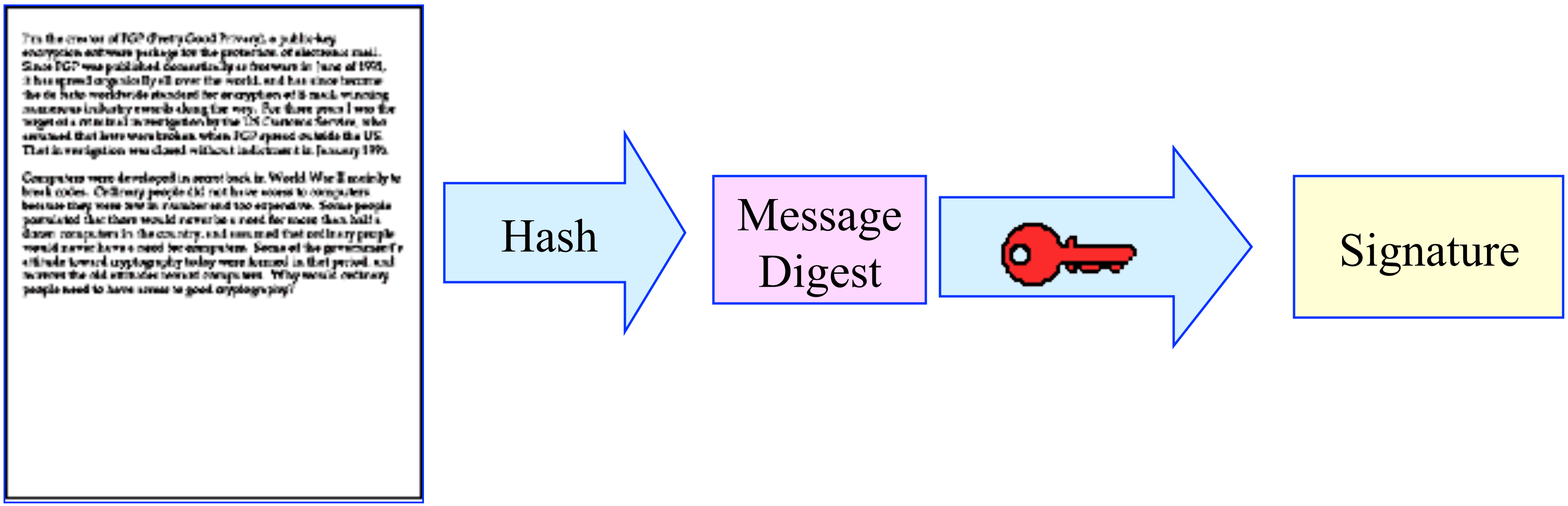
- Hash functions are important cryptographic primitive and are widely used in security protocols
- Compute digest of a message which is a short, fixed-length bit-string
  - Finger print of a message, i.e., unique representation of a message
- Does not have key

Properties

- Deterministic
- Fast
- Irreversible
- Utilize the ‘avalanche effect’

Application

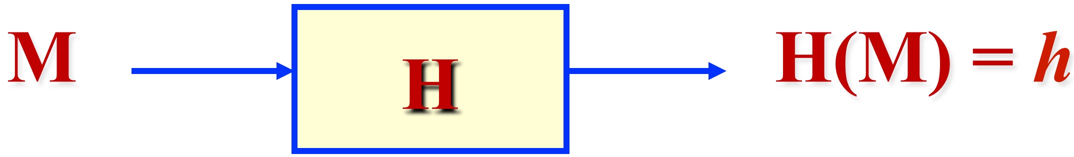
- Part of digital signature



Security Requirements of Hash Functions

- One-wayness
  - Given M, it is easy to compute h
  - Given any h, it is hard to find any M, such that H(M) = h
- Collision-resistant
  - Given M1, it is **difficult** to find M2, such that H(M1) = H(M2)

One-Way Hash Functions

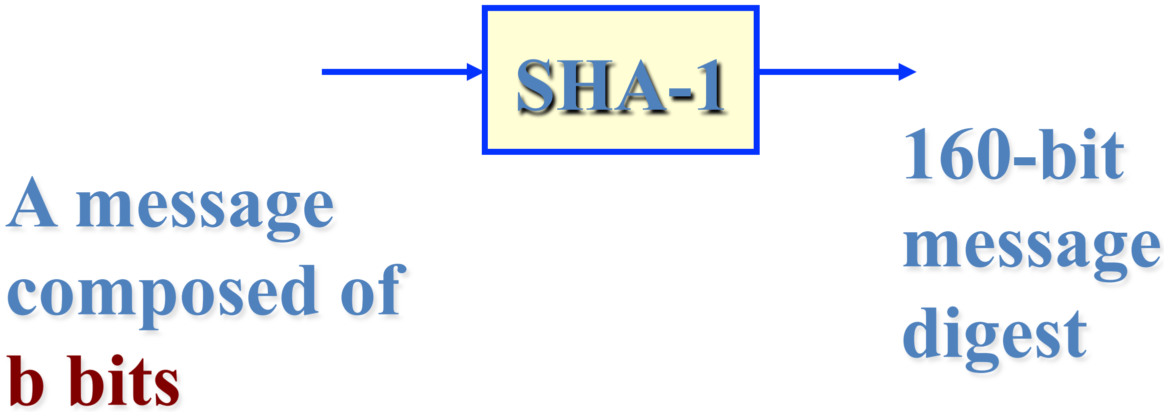


Example

- M = “Elvis”
- H(M) = (“E” + “L” + “V” + “I” + “S”) mod 26
- H(M) = (5 + 12 + 22 + 9 + 19) mod 26
- H(M) = 67 mod 26 =15

Secure Hash Algorithm (SHA)

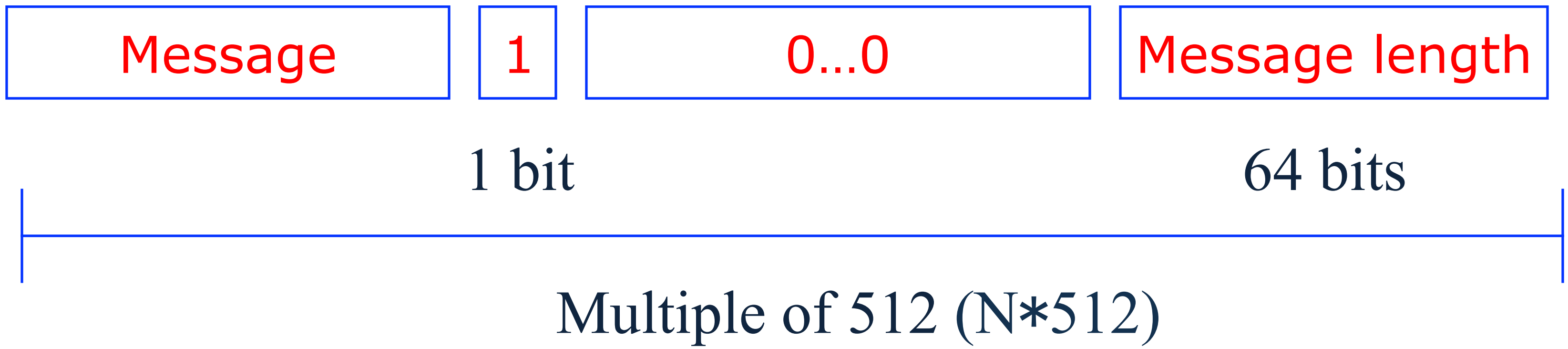
- Input: 0-2^{64} bits
  - 2^{30} bits~ 1G bits
- Output: 160 bits, contant



Preprocess-- Padding

- *Padding* → the total length of a padded message is multiple of 512

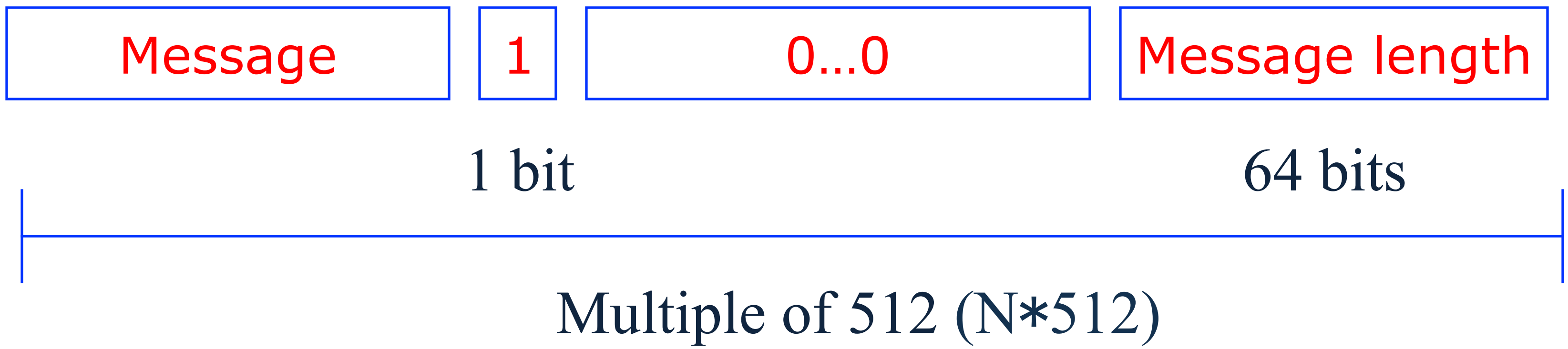
Padding (cont.)



- Padding is done by appending to the input
  - A single bit, 1
  - Enough additional bits, all 0, to make **the final block** exactly 512 bits long
  - A 64-bit integer representing the length of the original message in bits

Example

- M = 01100010 11001010 1001 (20 bits)



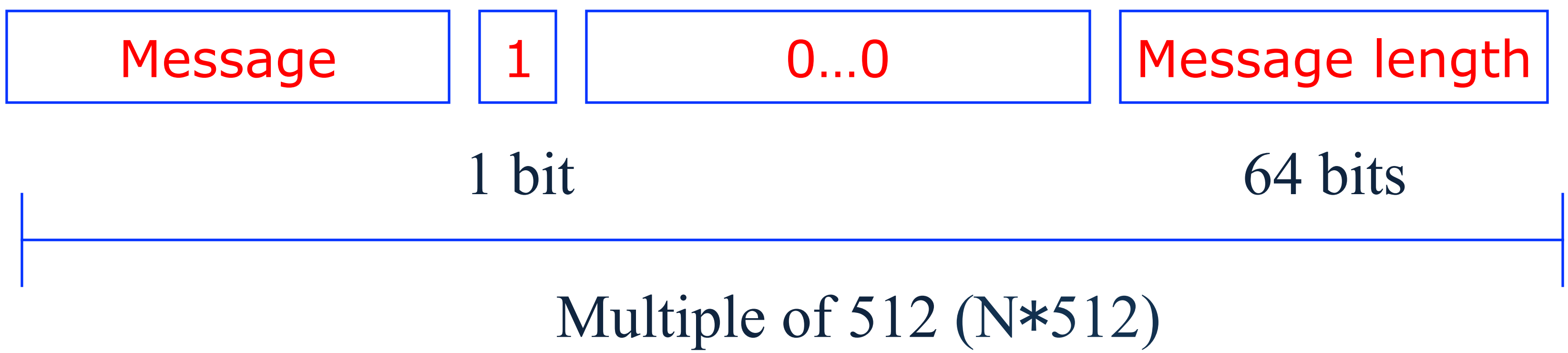
- How many 0’s?
- Representation of “Message length”?

Example

- M = 01100010 11001010 1001 (20 bits)
- Padding is done by appending to the input
  - A single bit, 1
  - 427 0s=512-1-64-20
  - A 64-bit integer representing 20
- Pad(M) = 01100010 11001010 10011000 … 00010100
- Length of Pad(M): 512 bits (N=1)

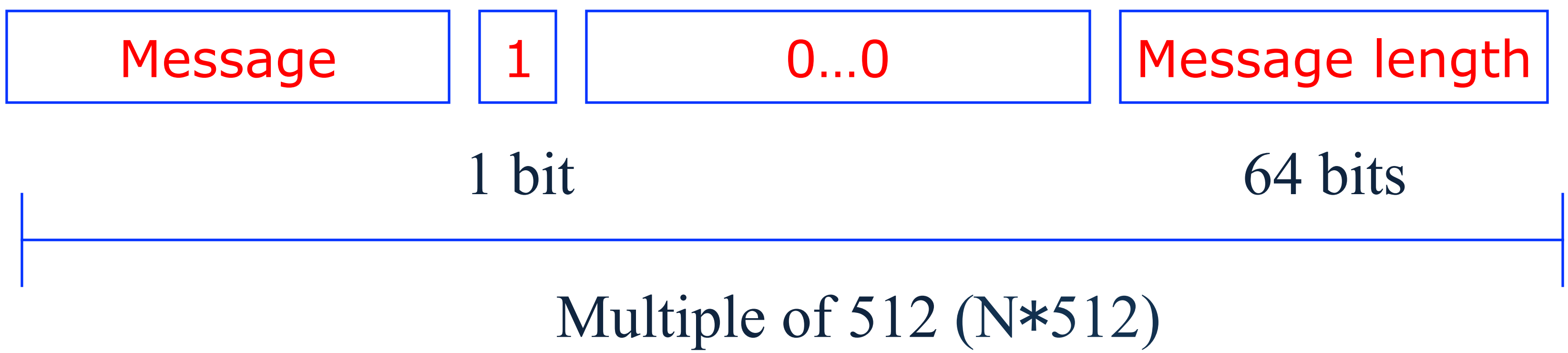
Example 2

- Length of M = 500 bits
- How many blocks? (N=?)



Example 2

- Length of M = 500 bits→N=2
- How many 0’s?
- “Message length”?



Example 2

- Length of M = 500 bits

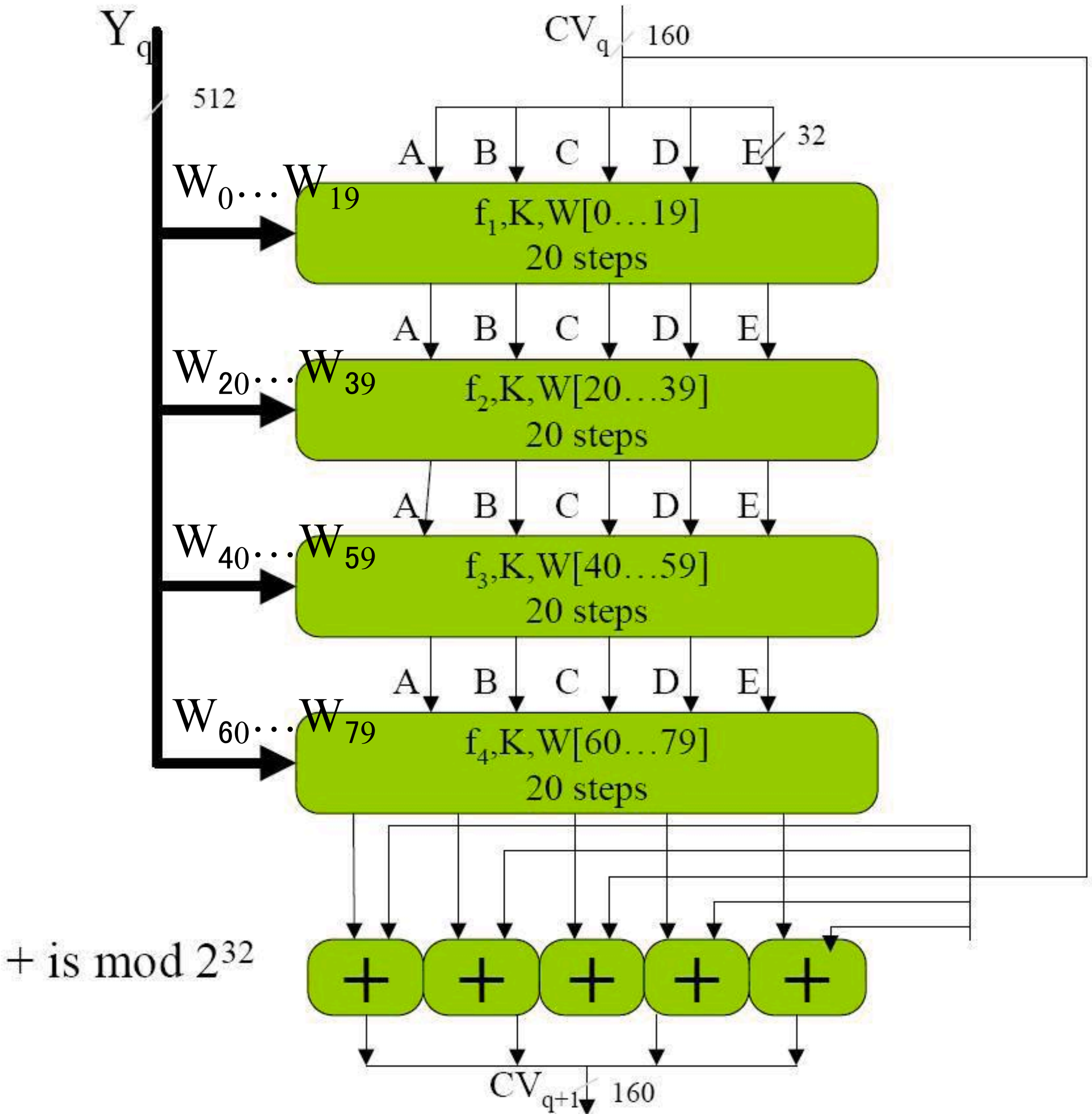
- Padding is done by appending to the input:
  - A single bit, 1
  - 459 0s=1024-500-1-64
  - A 64-bit integer representing 500

- Length of Pad(M) = 1024 bits

Step 1 -- Dividing Pad(M)

- Pad (M) = $B_1, B_2, B_3, \ldots, B_n$
- Each B_i denote a 512-bit block
- Each B_i is divided into 16 32-bit words
  - $W_0, W_1, \ldots, W_{15}$

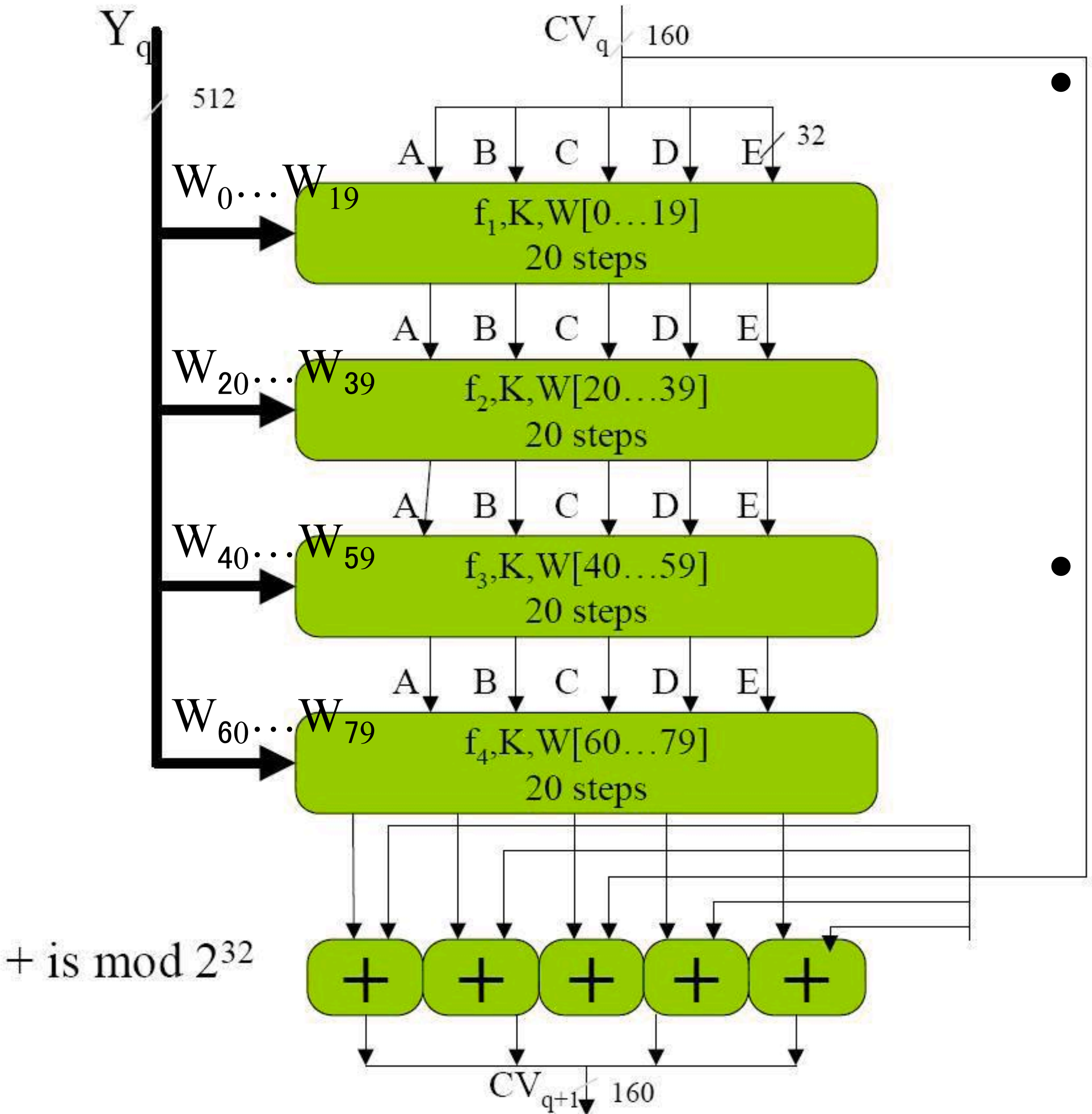
SHA-1 operation over one block



Step 2 – Compute W_{16} – W_{79}

- To Compute word W_j ($16 <= j <= 79$)
  - $W_j = (W_{j-3} \text{ XOR } W_{j-8} \text{ XOR } W_{j-14} \text{ XOR } W_{j-16}) <<< 1$
    - W_{j-3}, W_{j-8}, W_{j-14} , W_{j-16} are XORed
    - The result is circularly left shifted one bit

SHA-1 operation over one block

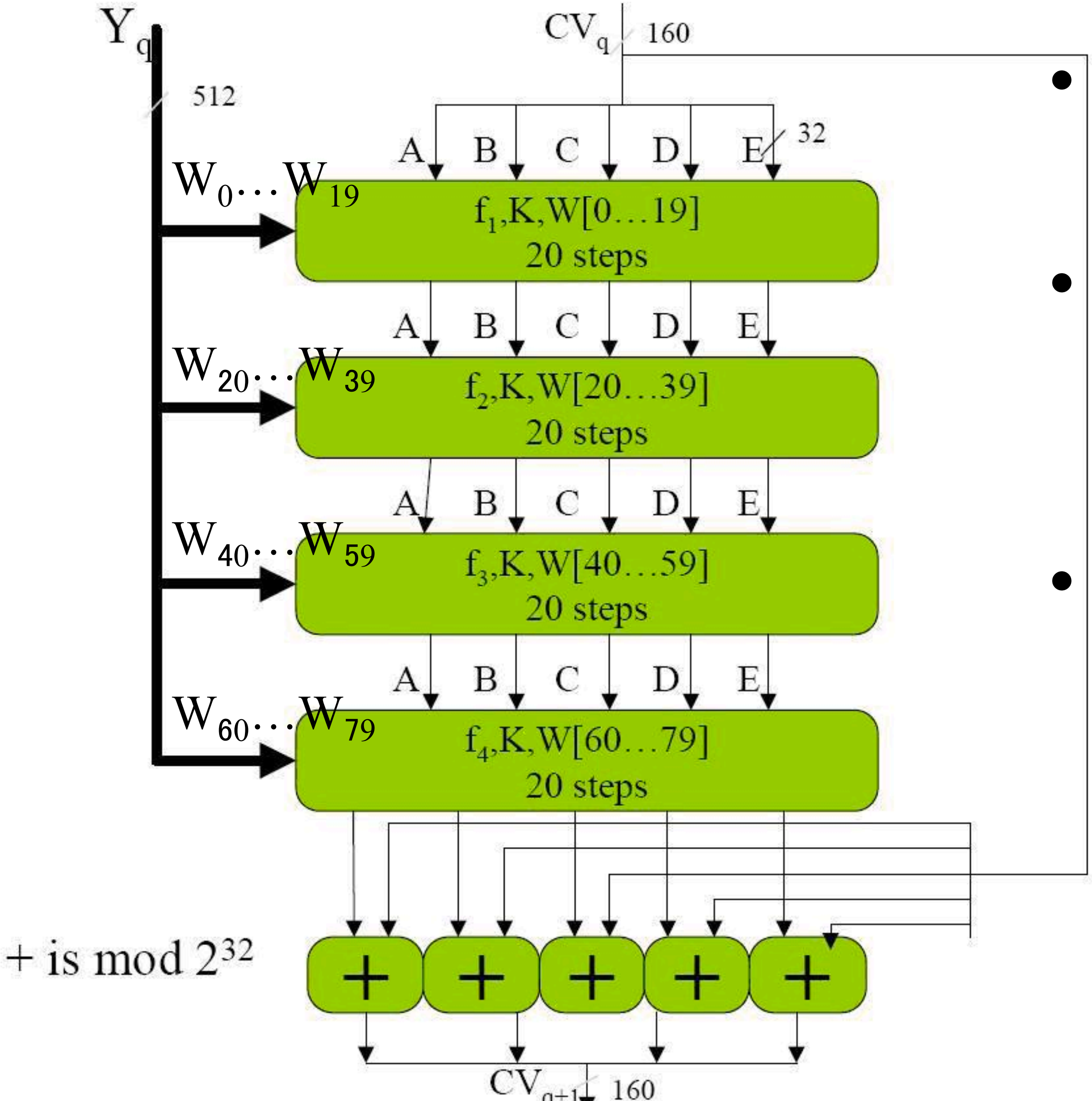


- The output of last block operation (CV_q) is the input of this block operation (q+1)
- How to obtain CV_0(A--E)

Step 3 Initialization

- **A** = CV_0 (0)= 67452301
- **B** = CV_0 (1) = EFCDAB89
- **C** = CV_0 (2) = 98BADCFE
- **D** = CV_0 (3) = 10325476
- **E** = CV_0 (4) = C3D2E1F0

SHA-1 operation over one block

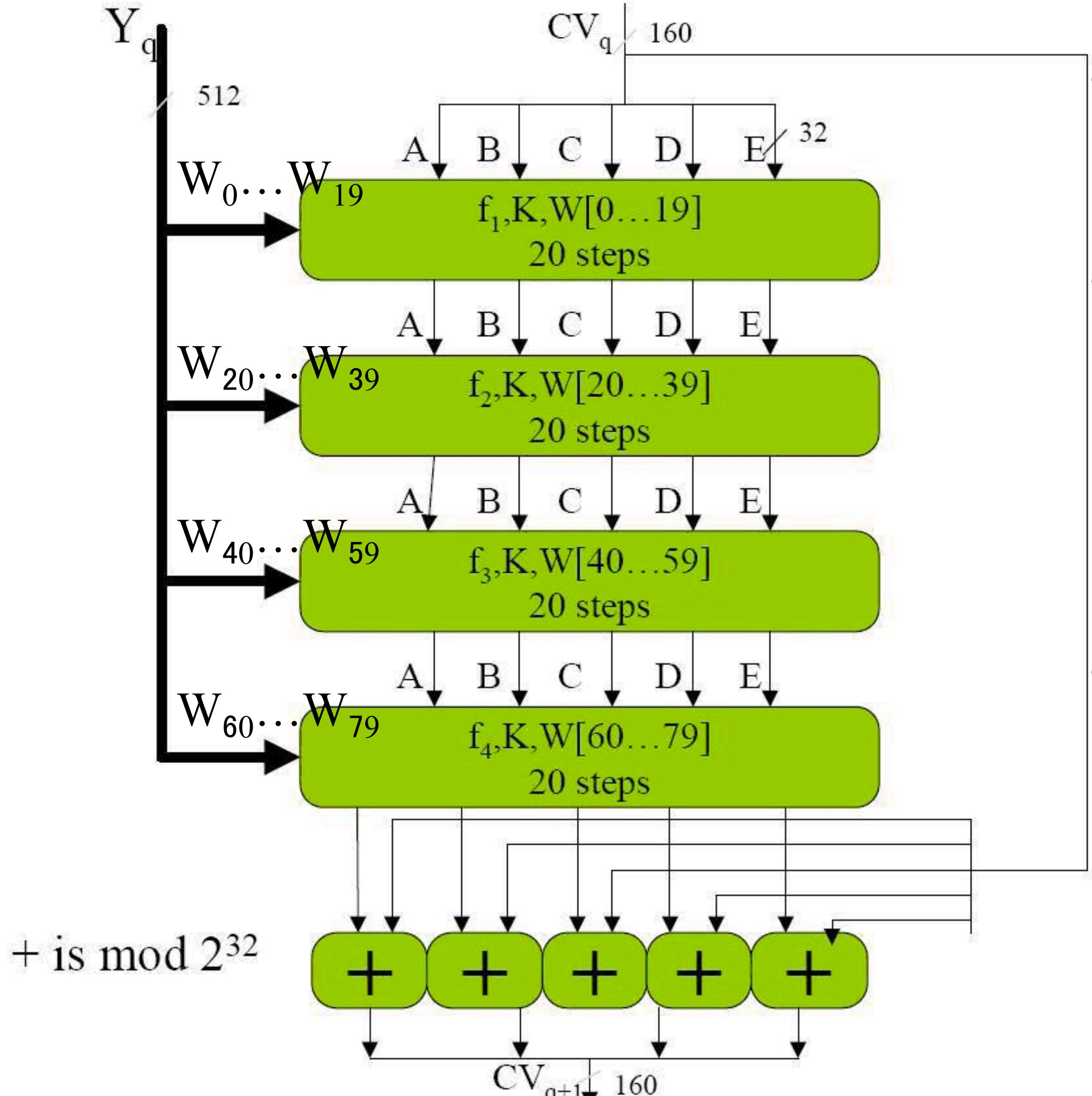


- 4 stage, each with 20 steps
- In each stage t, there is a stage-dependent K_t
- K_t's are constant values

Step 3 Initialization

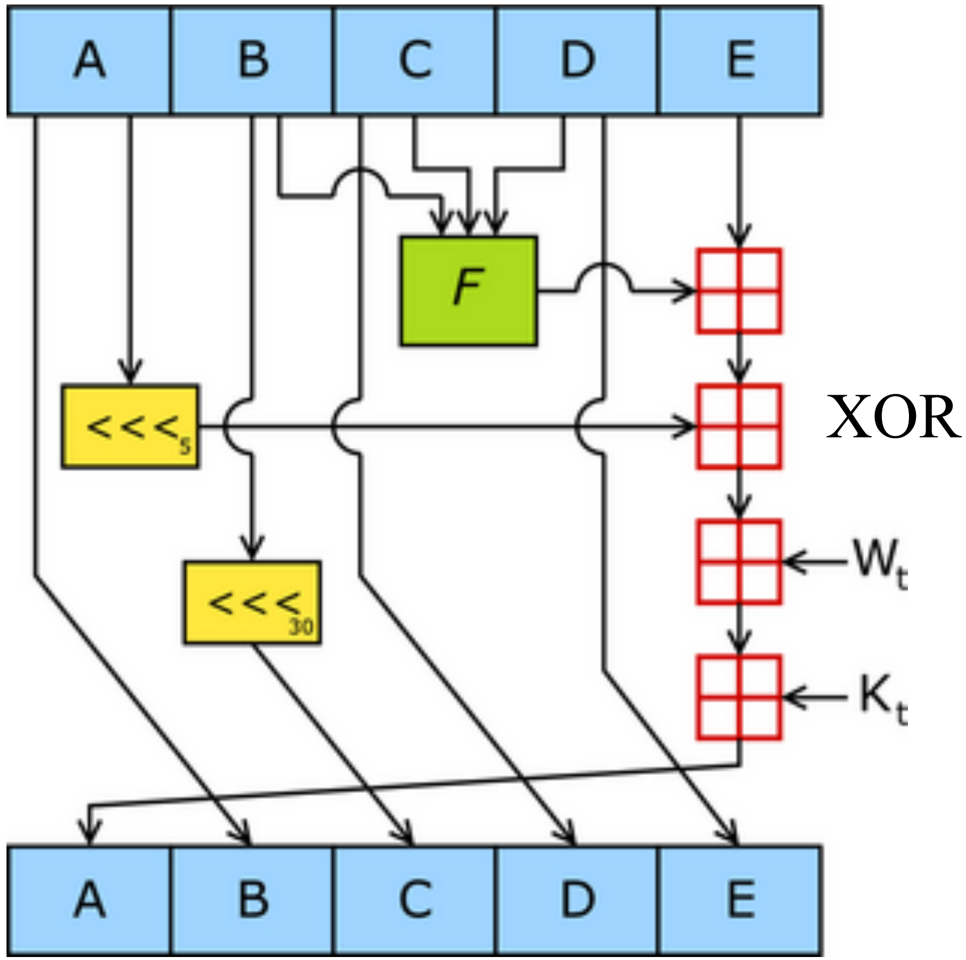
- K_0= 5A827999
- K_1= 6ED9EBA1
- K_2= 8F1BBCDC
- K_3= CA62C1D6

SHA-1 operation over one block



- Input for step i: W_i, (ABCDE)
- f_t: some internal function, different for each stage
- A-E: output from last step

Details of One Step (step 4 loop)



Details of One Step (step 4 loop)

For $j = 0 \ldots 79$

$TEMP = CircLeShift\_5(A) + f_j(B,C,D) + E + W_j + K_j$

$E = D$; $D = C$;

$C = CircLeShift\_30(B)$;

$B = A$; $A = TEMP$

Done

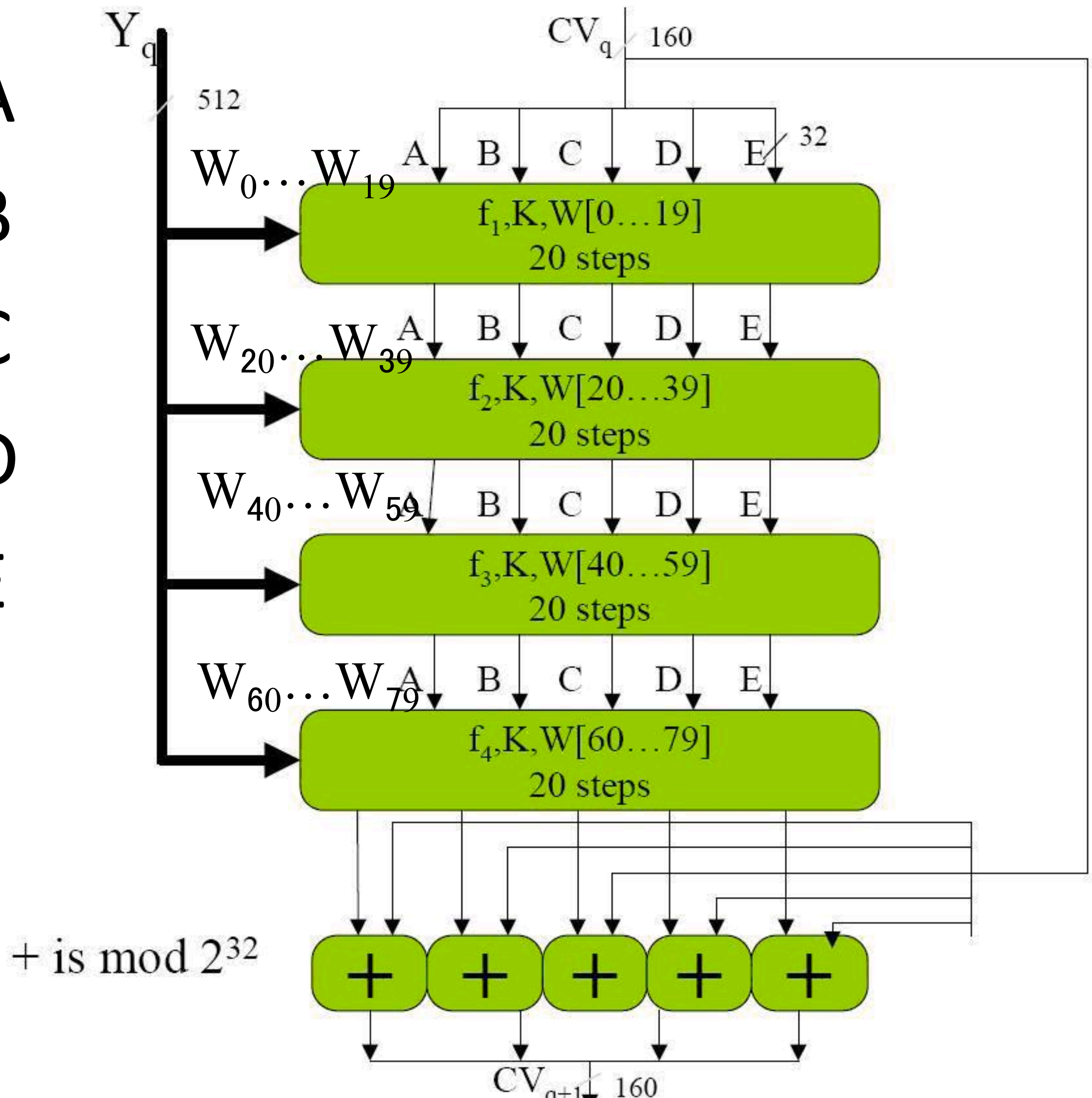
+ → addition (ignore overflow)

Four functions

- For j = 0 … 19
  - $f_j(B,C,D)$ = (B AND C) OR (B AND D) OR (C AND D)
- For j = 20 … 39
  - $f_j(B,C,D)$ = (B XOR C XOR D)
- For j = 40 … 59
  - $f_j(B,C,D)$ = (B AND C) OR ((NOT B) AND D)
- For j = 60 … 79
  - $f_j(B,C,D)$ = (B XOR C XOR D)

Step 5 – Final

- CV_{q+1} (0) = CV_q (0) + A
- CV_{q+1} (1) = CV_q (1) + B
- CV_{q+1} (2) = CV_q (2) + C
- CV_{q+1} (3) = CV_q (3) + D
- CV_{q+1} (4) = CV_q (4) + E



Done

- Once these steps have been performed on each 512-bit block (B_1, B_2, ..., B_n) of the padded message,
  - the 160-bit message digest is given by

$$CV_n\,(0)\ CV_n\,(1)\ Cn_1\,(2)\ CV_n\,(3)\ CV_n\,(4)$$

- Why SHA-1 is constructed in this way– to achieve
  - Onewayness
  - Collision resistance
- Derived by tons of experiments

SHA

	Output size (bits)	Internal state size (bits)	Block size (bits)	Max message size (bits)	Word size (bits)	Rounds	Operations	Collisions found
SHA-0	160	160	512	$2^{64} - 1$	32	80	+, and, or, xor, rot	Yes
SHA-1	160	160	512	$2^{64} - 1$	32	80	+, and, or, xor, rot	None (2^{51} attack)
SHA-2	256/224	256	512	$2^{64} - 1$	32	64	+, and, or, xor, shr, rot	None
	512/384	512	1024	$2^{128} - 1$	64	80	+, and, or, xor, shr, rot	None