CISC 468: CRYPTOGRAPHY

LESSON 16: HASH FUNCTIONS

Furkan Alaca

READINGS

- Section 11.3: Overview of Hash Algorithms, Paar & Pelzl
- Section 11.4: The Secure Hash Algorithm SHA-1, Paar & Pelzl

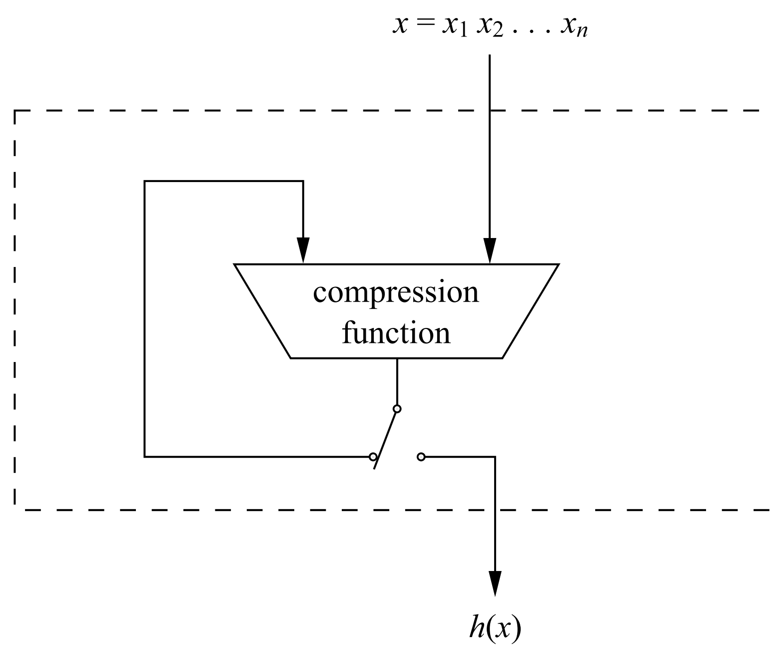
OVERVIEW OF HASH ALGORITHMS

There are two general types of hash functions:

1. *Dedicated hash functions* are specifically designed to serve as hash functions.
2. *Block cipher-based hash functions* are hash functions constructed from block ciphers.

DEDICATED HASH FUNCTIONS: MERKLE-DAMGARD CONSTRUCTION

- The *Merkle-Damgard construction* segments the input message into equal-sized blocks and feeds them sequentially into a *compression function* that outputs a fixed-length block
- The input to the compression function is the current input message block and the previous output block
  - The hash value of the message is the final output block



DEDICATED HASH FUNCTIONS: THE MD4 FAMILY

- MD4 was a popular hash function, based entirely on bitwise Boolean functions (AND, OR, XOR, and NOT)
- MD4 influenced the design of MD5, SHA-1, and SHA-2
- SHA-3 uses an all-new design (not part of the MD4 family)
- MD4 and MD5 compute a 128-bit output; in the absence of analytical attacks they possess a collision resistance of 2^{64}
  - But weaknesses in MD4 made it more susceptible to more powerful analytical attacks, which motivated the design of the stronger MD5

DEDICATED HASH FUNCTIONS: THE MD4 FAMILY (2)

- SHA-1 computes a 160-bit output, and consists of more rounds than MD5
- SHA-2 is a family of functions: SHA-224, SHA-256, SHA-384, and SHA-512, which compute, 224-, 256-, 384-, and 512-bit outputs, respectively
  - Truncated variants: SHA-512/224 and SHA-512/256
- MD5, SHA-1, and SHA-2 are all built using a Merkle-Damgard construction
- MD5 and SHA-1 are considered broken; collisions can be found in far fewer steps than their output lengths suggest

COLLISION ATTACKS ON MD5

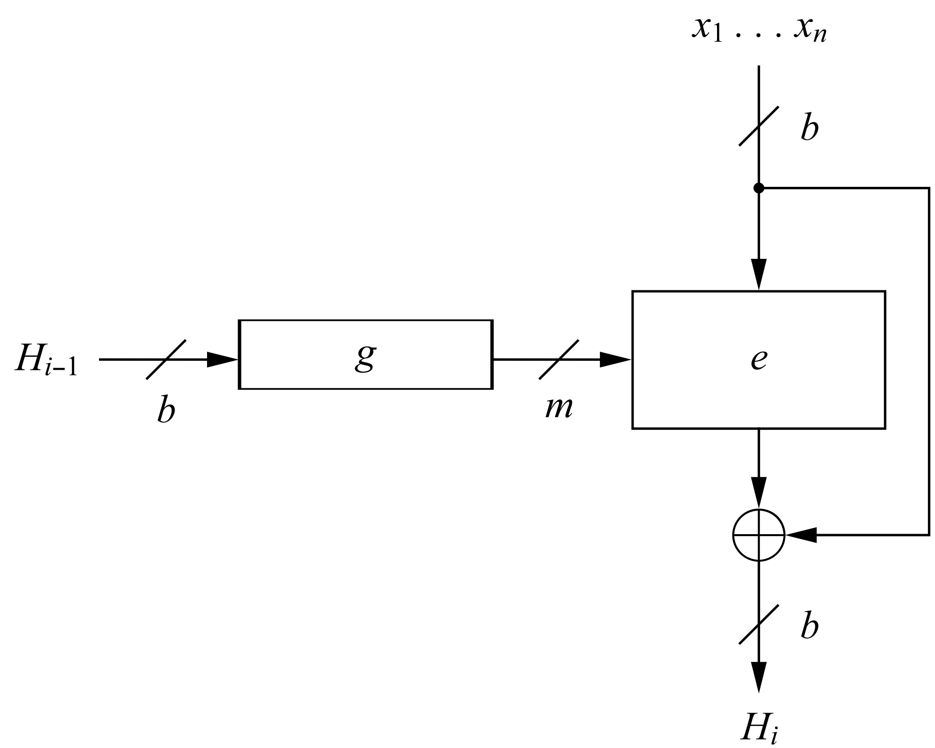
- MD5 collisions can easily be found on modern hardware
- This tutorial blog post from Oct. 2014 demonstrates how the two images below were modified to generate the same MD5 hash, in 10 hours of computation time on an Amazon Web Services GPU instance, at a cost of about 65 cents

COLLISION ATTACK ON SHA-1: *SHATTERED*

- Feb. 2017: Google construed two distinct PDF files with the same SHA-1 hash
- Required 9,223,372,036,854,775,808 SHA-1 computations
- Equal to 6,500 years of single-CPU and 110 years of single-GPU computation
- 100K times faster than a brute-force attack

HASH FUNCTIONS FROM BLOCK CIPHERS

- The Matyas-Meyer-Oseas construction inputs each message block m_i into a block cipher to generate an output H_i
- The previous output H_{i-1} is used as the key
  - If the key size differs from the block size, it is converted by the b-to-m-bit mapping function g
- The last output value H_n is the hash of the entire message

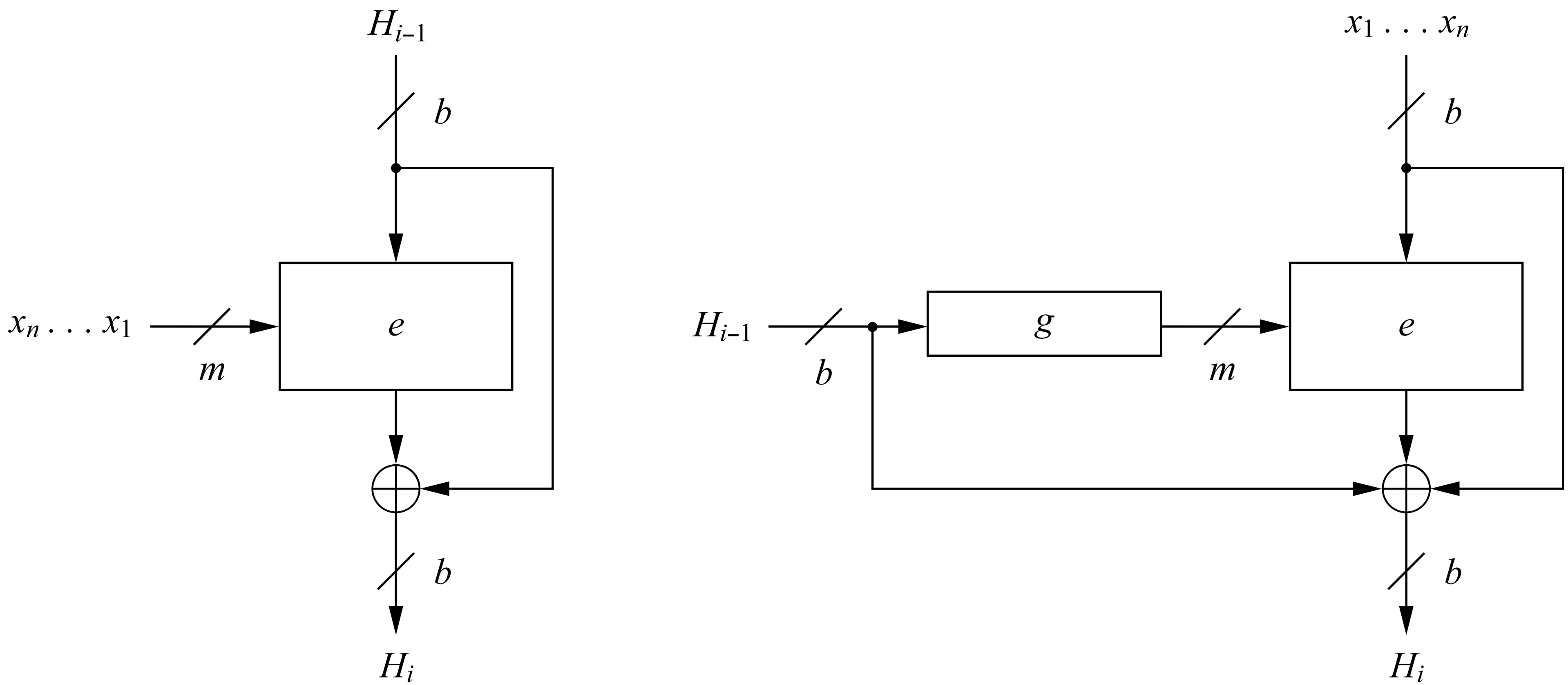


HASH FUNCTIONS FROM BLOCK CIPHERS (2)

- The Matyas-Meyer-Oseas construction with AES produces a 128-bit hash value provides 64-bit collision resistance
  - Insufficient if collision resistance is required
- Other constructions exist

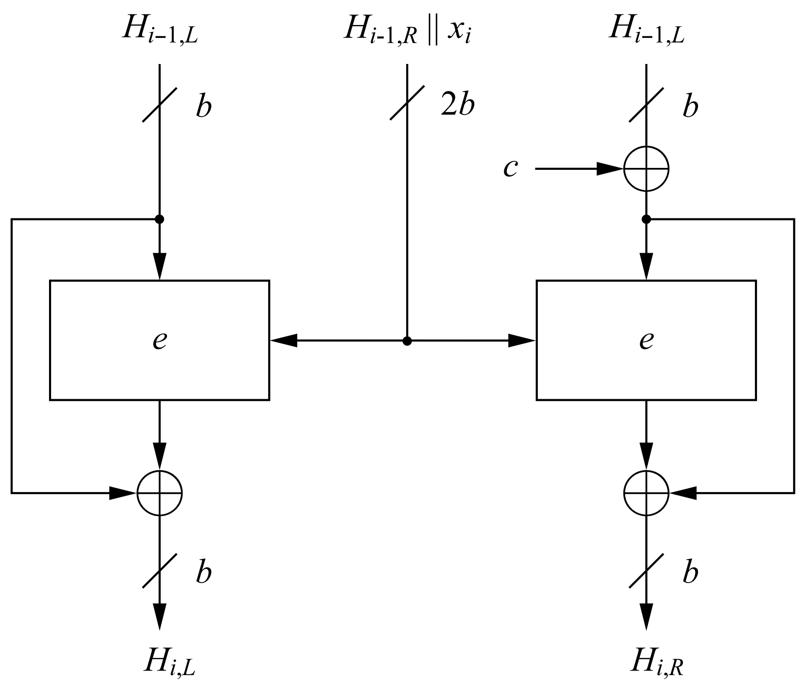
DAVIES-MEYER AND MIYAGUCHI-PRENEEL CONSRUCTIONS

Davies-Meyer (left) and Miyaguchi-Preneel (right):



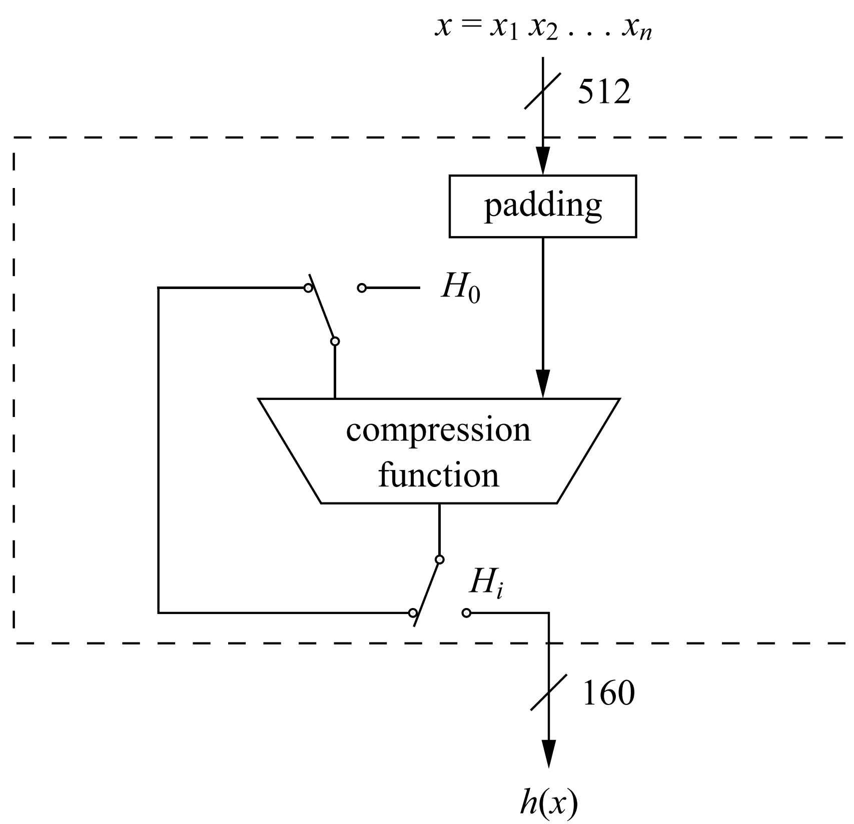
THE HIROSE CONSTRUCTION

- The Hirose generates a $2n$ output for block ciphers where the key size k is greater than the block size n
  - Generates a 256-bit hash with AES-192 or AES-256, providing 128-bit collision resistance



SHA-1

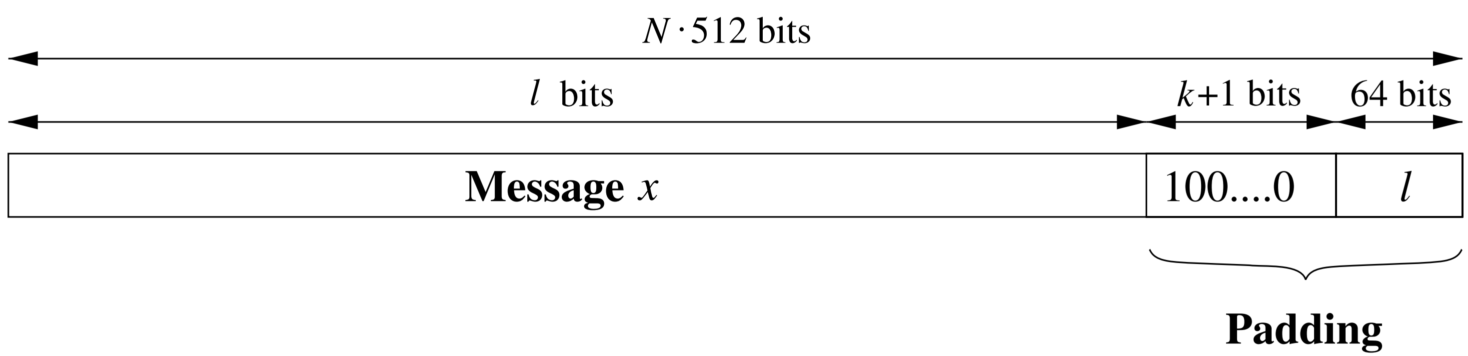
- Despite the attacks, it is still instructive to look at SHA-1, as the stronger SHA-2 family uses a very similar structure
- SHA-1 pads the input message to a multiple of 512 bits, and the compression function processes it in 512-bit chunks



SHA-1: PADDING

- Assume an l-bit message x
- A single 1 bit is appended, followed by k zero bits and a 64-bit representation of l
  - So, the input cannot be larger than 2^{64} bits
- The number of zero bits is obtained by

$$k \equiv 512 - 64 - 1 - l = 448 - (l + 1) \bmod 512.$$



SHA-1: INTERNAL STATE

- Each 512-bit block x_i is divided into 16 words of 32 bits each,

$$x_i^{(0)}, x_i^{(1)}, \ldots, x_i^{(15)}$$

- Each message block x_i is processed by the compression function, which generates a 160-bit hash value H_i that is added to a 160-bit *internal state*
- The internal state is carried over when the next block x_{i+1} is being computed
- The internal state is stored as five words in the 32-bit working registers A, B, C, D, E

SHA-1: INITIAL VALUE

- Before processing the first block x_1, an initial value H_0 is loaded into the internal state as follows:

$$
\begin{aligned}
A &= H_0^{(0)} = 67452301, \\
B &= H_0^{(1)} = EFCDAB89, \\
C &= H_0^{(2)} = 98BADCFE, \\
D &= H_0^{(3)} = 10325476, \\
E &= H_0^{(4)} = C3D2E1F0.
\end{aligned}
$$

SHA-1: HASH COMPUTATION (MESSAGE SCHEDULE)

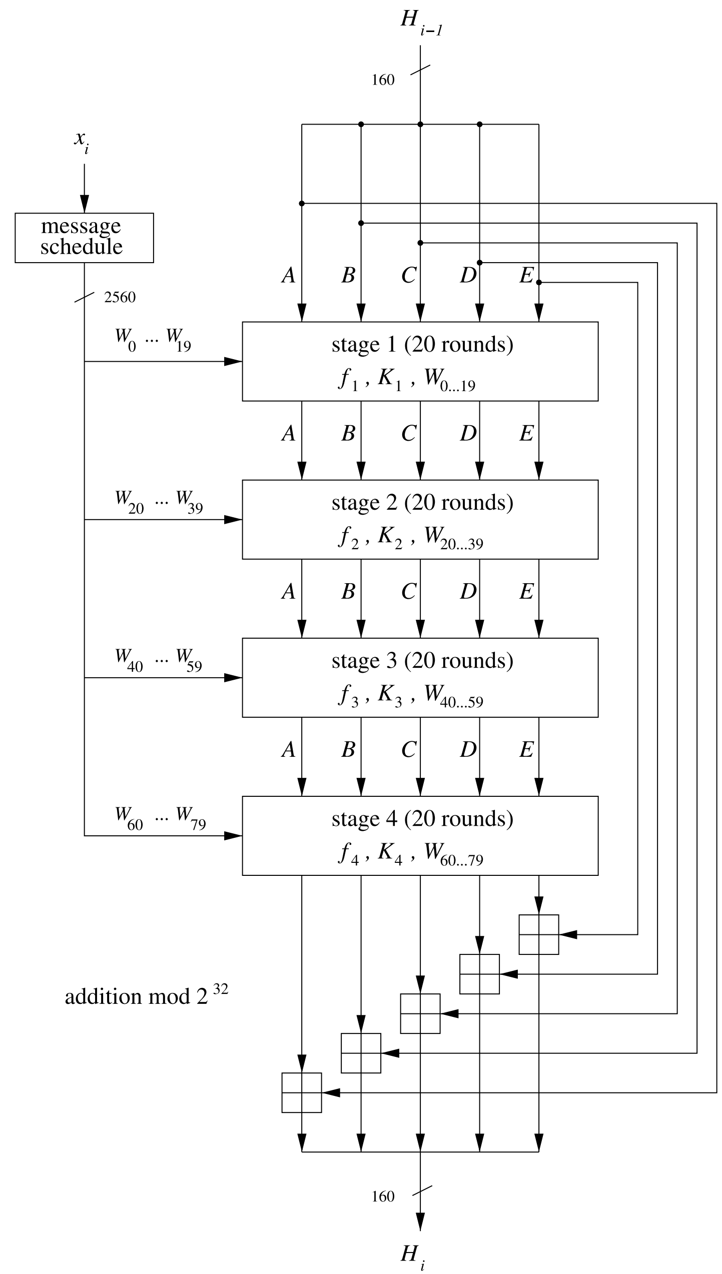
- Each block x_i is processed four stages with 20 rounds each
- For each block, a *message schedule* computes a 32-bit word W_j, where $0 \leq j \leq 79$, for each of the 80 rounds as

$$W_j = \begin{cases} x_i^{(j)} & 0 \leq j \leq 15 \\ (W_{j-16} \oplus W_{j-14} \oplus W_{j-8} \oplus W_{j-3})_{\lll 1} & 16 \leq j \leq 79. \end{cases}$$

- The notation $X_{\lll n}$ indicates a circular left shift of the word X by n bit positions

SHA-1: HASH COMPUTATION (STAGES)

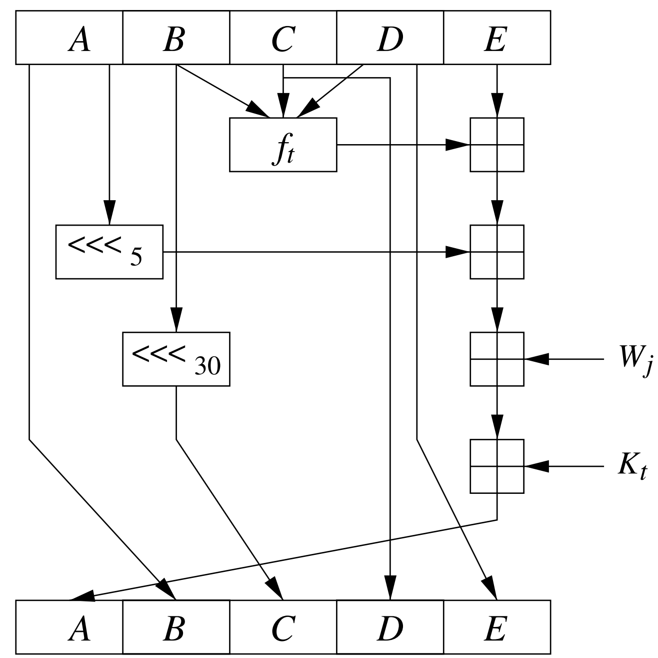
- Each stage updates the internal state A, B, C, D, E and feeds it to the next stage
- H_i is computed after stage 4 by adding the prior hash H_{i-1} to the internal state
  - H_i depends on x_i and H_{i-1}
  - Additions are in $\bmod 2^{32}$
- H_n is produced after processing the last message block x_n, and is the SHA-1 hash of the entire message x



SHA-1: HASH COMPUTATION (ROUNDS)

- Each stage $t \in \{1, 2, 3, 4\}$ contains 20 rounds
- All rounds share the same structure below, but each stage has a different function f_t and constant K_t
- Each round j updates the internal state as follows:

$$A = E + f_t(B, C, D) + A_{\lll 5} + W_j + K_t$$
$$B = A$$
$$C = B_{\lll 30}$$
$$D = C$$
$$E = D$$



SHA-1: HASH COMPUTATION (FUNCTIONS AND CONSTANTS)

The internal functions f_t and constants K_t used in each round are tabulated below. Note that the functions use only Boolean operations.

Stage t	Round j	Constant K_t	Function f_t
1	$0\ldots19$	$K_1 = \texttt{5A827999}$	$f_1(B,C,D) = (B \wedge C) \vee (\bar{B} \wedge D)$
2	$20\ldots39$	$K_2 = \texttt{6ED9EBA1}$	$f_2(B,C,D) = B \oplus C \oplus D$
3	$40\ldots59$	$K_3 = \texttt{8F1BBCDC}$	$f_3(B,C,D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
4	$60\ldots79$	$K_4 = \texttt{CA62C1D6}$	$f_4(B,C,D) = B \oplus C \oplus D$

LENGTH-EXTENSION ATTACKS

- Hash functions that use the Merkle-Damgard construction are vulnerable to *length-extension attacks*
- Since the hash is calculated iteratively (block by block), an attacker can use a hash $h(x)$ to compute $h(x||y)$ without knowing x
- We will learn the implications of this attack attack when we study message authentication codes

LENGTH-EXTENSION ATTACKS: SHA-2 VS SHA-3:

- SHA-2 is currently considered to be secure, but since it shares the same Merkle-Damgard construction of MD5 and SHA-1 it is thought that it may also be broken in the future
- SHA-3 uses a different construction, called a *sponge construction*, and is secure against length-extension attacks
  - Truncated variants of SHA-2, e.g., SHA-512/256, also help mitigate length-extension attacks, by forcing the attacker to determine the truncated portion of the hash before proceeding with the attack