Chapters 11

Hash Functions

Dr. Shin-Ming Cheng

NTUST
CONNECTIVITY LAB

CS4003701

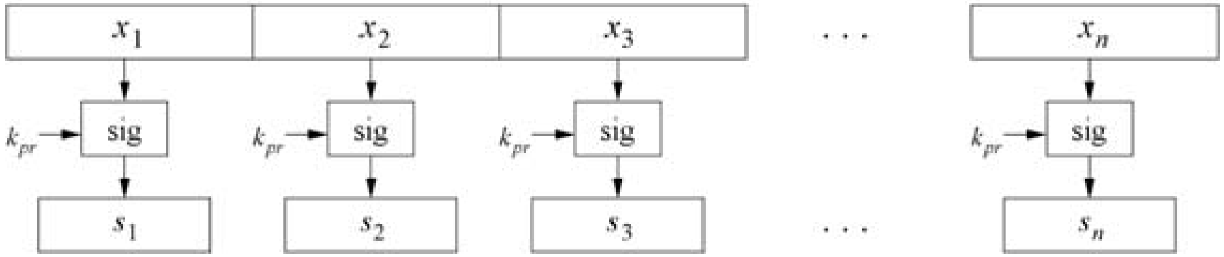
Introduction to Information Security

Hash Functions

- Why we need hash functions
- How does it work
- Security properties
- Algorithms
- The Secure Hash Algorithm MD4, SHA-1, SHA-512

Motivation

- Naive signing of long messages generates a signature of same length.



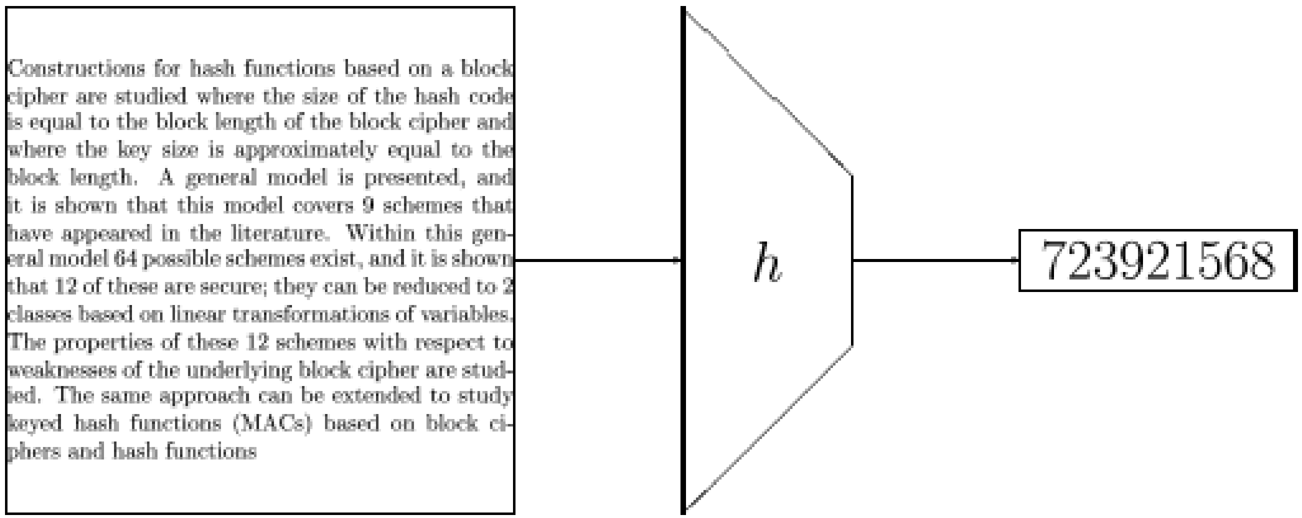
- Problems
  - Computational overhead
  - Message overhead
  - Security limitations
- Instead of signing the whole message, sign only a digest (=hash)
  - Also secure, but much faster

Motivation

- Encryption
  - Provides confidentiality
  - Does NOT necessarily provide integrity of data
    - Countermeasure: Use cryptographic function to get a check-value and send it with data
- Two mechanisms for data integrity
  - Hash functions
    - A special type of Manipulation Detection Code (MDC)
  - Message Authentication Codes (MAC)
    - A keyed version of a hash function

Hash Function

- Efficient function mapping binary strings of arbitrary length to binary strings of fixed length, called the hash-value or hash-code



Hash Functions

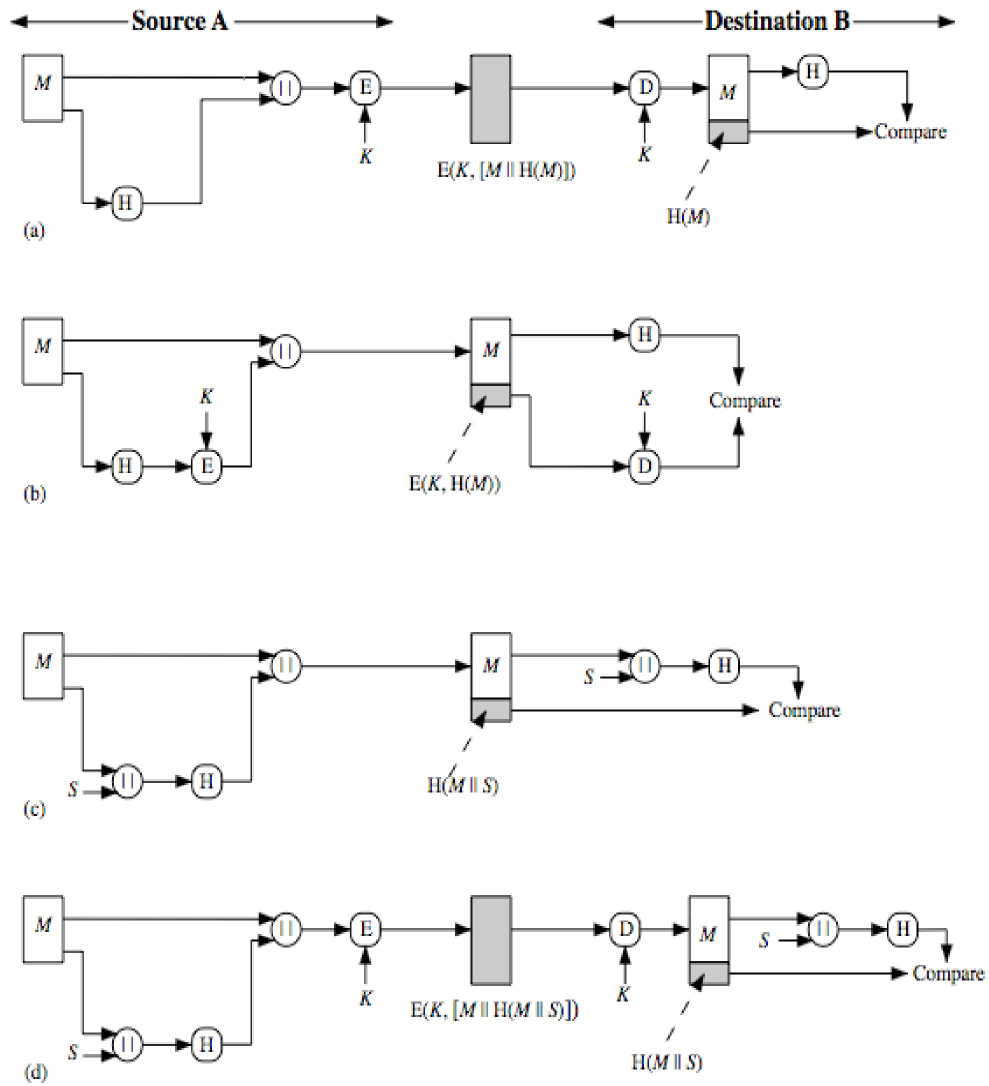
- Condenses arbitrary message to fixed size

$$h = H(M)$$

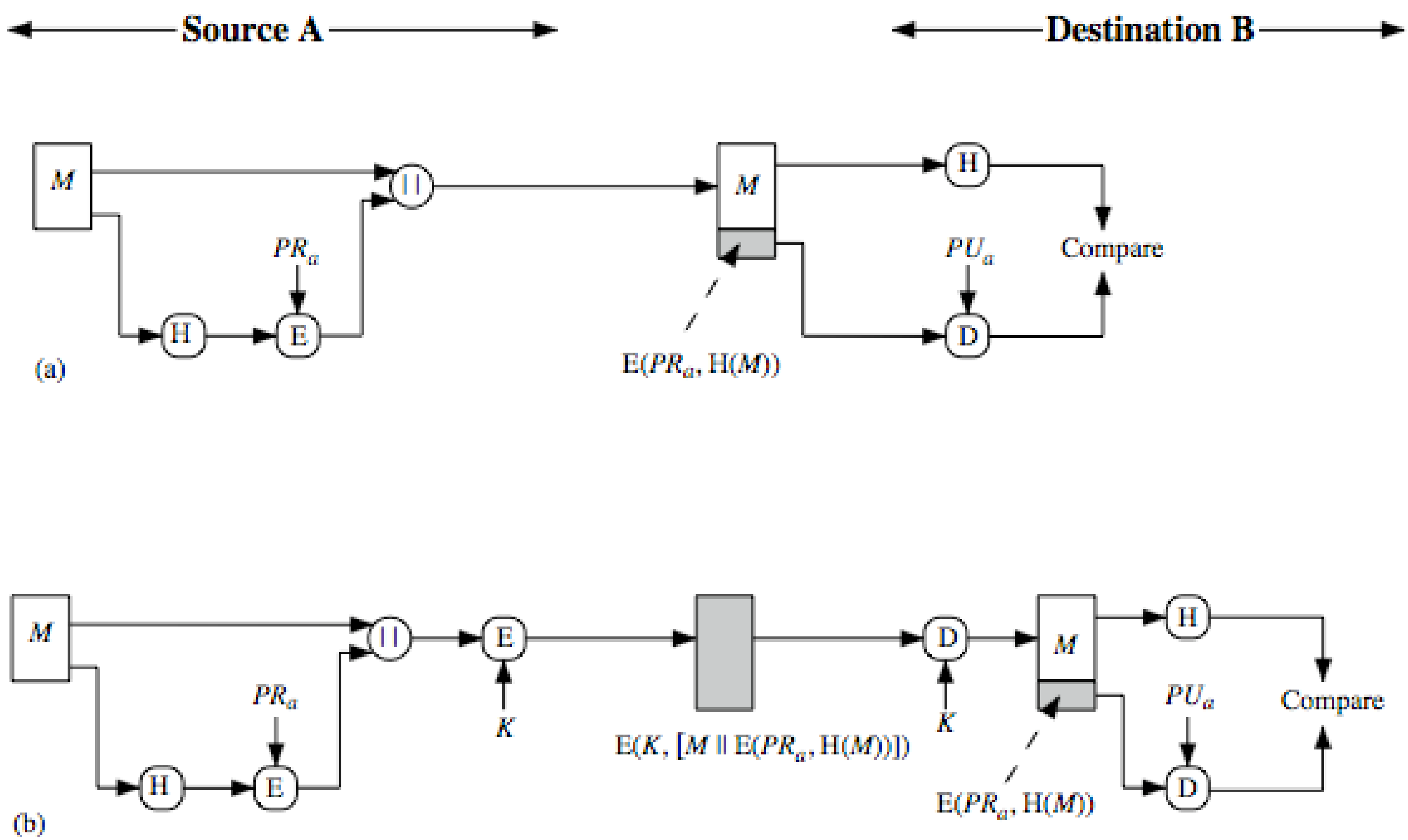
- Usually assume hash function is public
- Hash used to detect changes to message
- A cryptographic hash function
  - computationally infeasible to find data mapping to specific hash (one-way property)
  - computationally infeasible to find two data to same hash (collision-free property)

Hash Functions & Message Authentication

- the two parties share a common secret value S.
- A computes the hash value over the concatenation of M and S and appends the resulting hash value to M.
- Because B possesses S, it can recompute the hash value to verify.



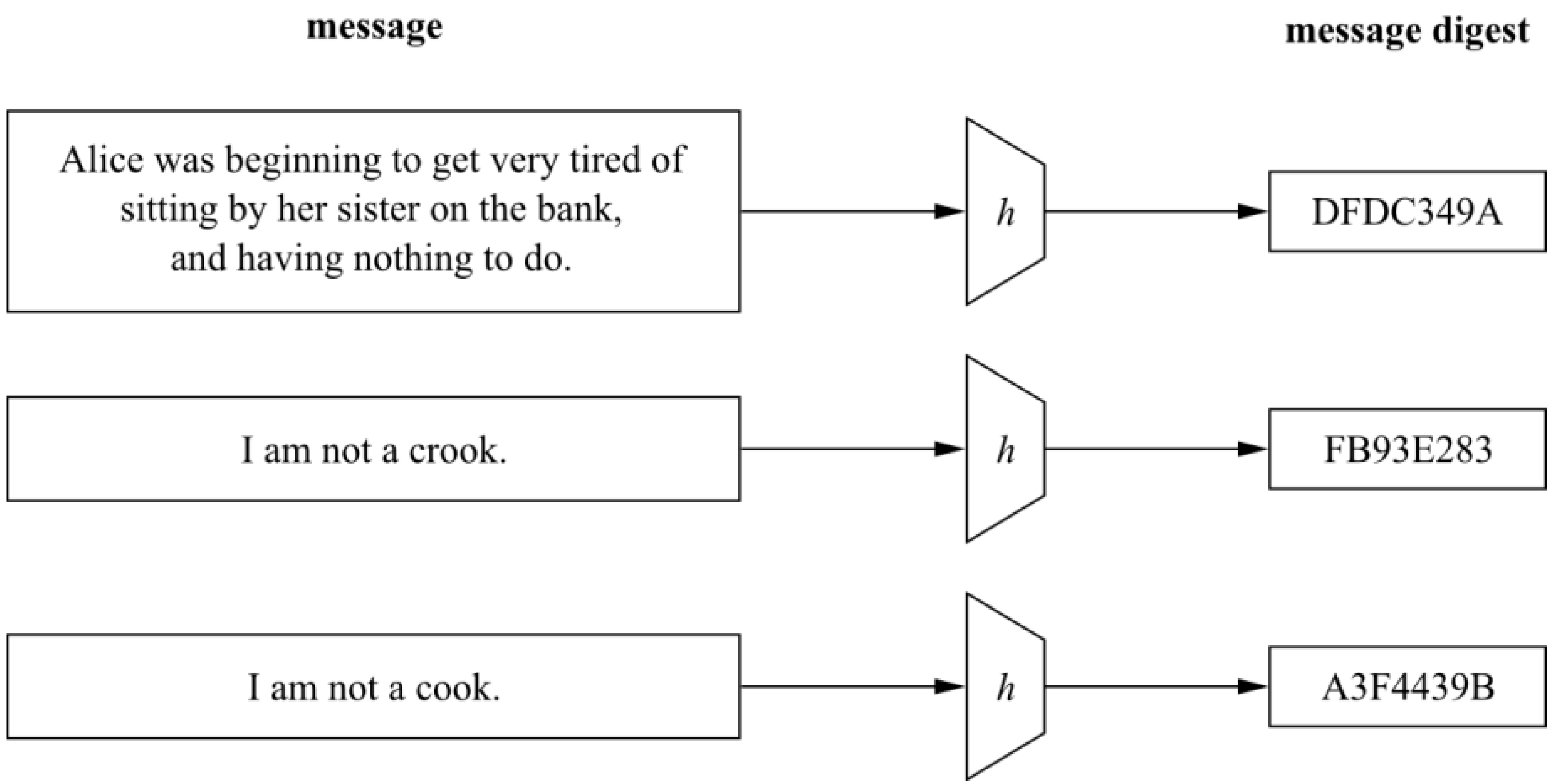
Hash Functions & Digital Signatures



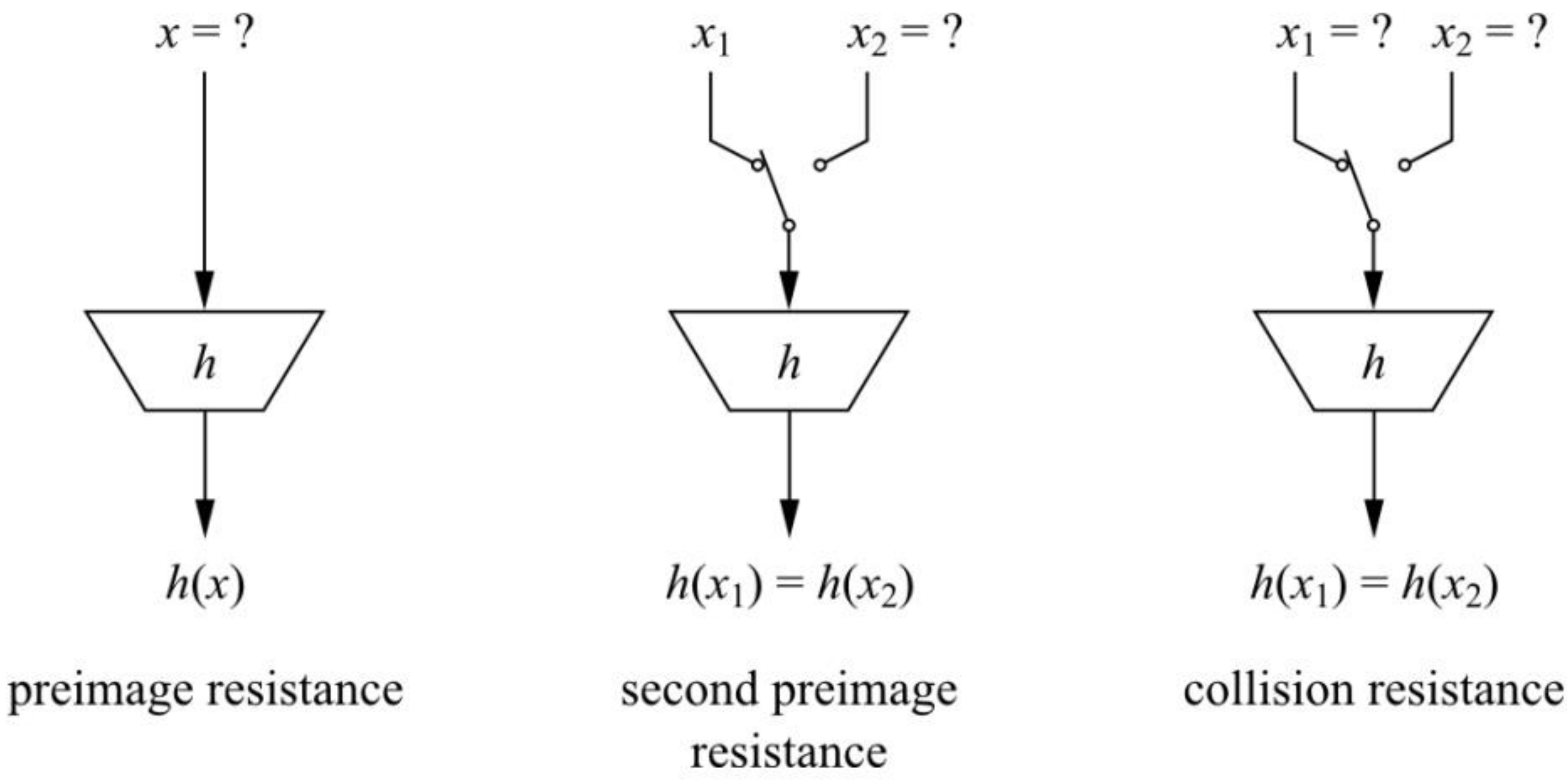
Other Hash Function Uses

- to create a one-way password file
  - store hash of password not actual password
- for intrusion detection and virus detection
  - keep & check hash of files on system
- pseudorandom function (PRF) or pseudorandom number generator (PRNG)

Principal input–output behavior of hash functions



Security Properties

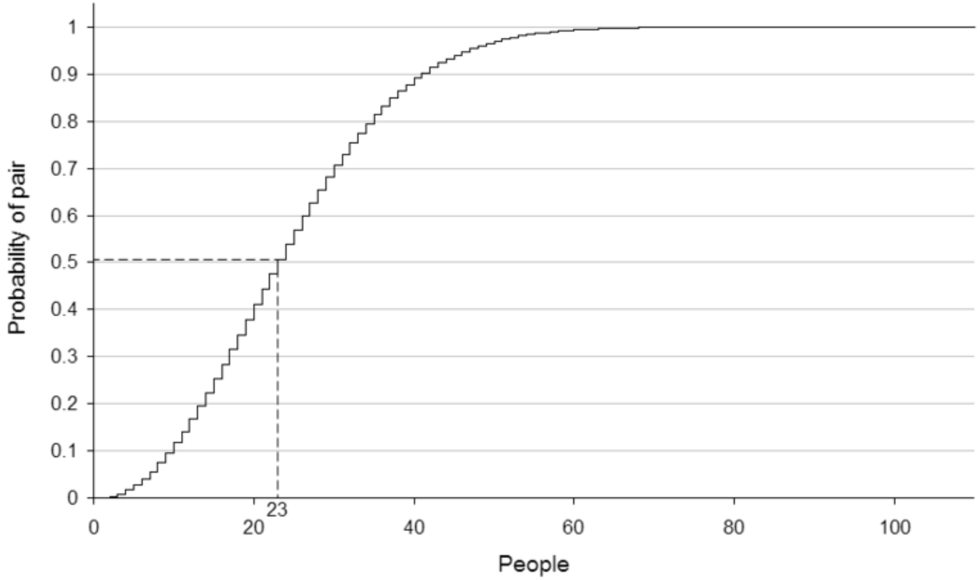


Security Properties

- Preimage resistance:
  - For a given output z, it is impossible to find any input x such that $h(x) = z$,
  - $h(x)$ is one-way
- Second preimage resistance:
  - Given x_1, and thus $h(x_1)$, it is computationally infeasible to find any x_2 such that $h(x_1) = h(x_2)$
- Collision resistance:
  - It is computationally infeasible to find any pairs $x_1 \neq x_2$ such that $h(x_1) = h(x_2)$

Security Properties

› Collision resistance causes most problems
  – How hard is it to find a collision with a probability of 0.5?
  – Birthday Problem: How many people are needed such that two of them have the same birthday with a probability of 0.5?
    › A counter-intuitive result



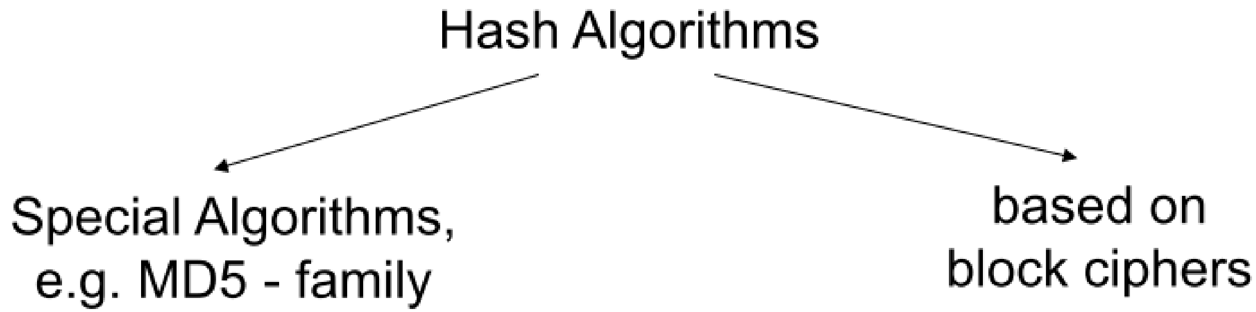
Security Properties
Birthday Paradox

- It is harder to construct collision free hash functions than one-way hash functions due to the birthday paradox
- If the function has output size of n bits, then we expect to find a collision after $2^{n/2}$ iterations
  - Whilst breaking the one-way property would require 2^n iterations on average
- Hence to achieve a security level of 80 bits we need 160 bits of output

Security Properties
Birthday Attack

- Given user prepared to sign a valid message x
- opponent generates
  - $2^{m/2}$ variations x' of x, all with essentially the same meaning, and saves them
  - $2^{m/2}$ variations y' of a desired fraudulent message y
- two sets of messages are compared to find pair with same hash
  - probability > 0.5 by birthday paradox
- have user sign the valid message, then substitute the forgery which will have a valid signature

Algorithms



- MD5 - family
- SHA-1: output – 160-bit; input – 512-bit chunks of message x; operations - bitwise AND, OR, XOR, complement und cyclic shifts
- RIPE-MD 160: output – 160-bit; input – 512-bit chunks of message x; operations – like in SHA-1, but two in parallel and combinations of them after each round

Hash from Block Ciphers

- A hash function can be constructed from a block cipher E
- There are a number of ways of doing this
  - All methods make use of a constant initial value IV
  - Some use a function g mapping n-bit inputs to keys
  - Pad the message into blocks $x_0, x_1, \dots, x_t$
  - Similar to CBC but without a key
  - The hash value is the final H_i in the iteration
    - $H_0 = IV$
    - $H_i = f(x_i, H_{i-1})$
- Resulting hash is too small (64-bit)
  - Vulnerable to attack

MD4 (Message-Digest algorithm 4)

- MD4 is a message digest algorithm (the fourth in a series) designed by Professor Ronald Rivest of MIT in 1990
- It implements a cryptographic hash function for use in message integrity checks
- The digest length is 128 bits
- The algorithm has influenced later designs, such as the MD5, SHA-1, RIPEMD, and SHA-2 algorithms

MD4 Family

- **MD4**: 3 rounds of 16 steps, output 128 bits
- **MD5**: 4 rounds of 16 steps, output 128 bits
- **SHA-1**: 4 rounds of 20 steps, output 160 bits
- **RIPEMD-160**: 5 rounds of 16 steps, output 160 bits
- **SHA-2**:
  - **SHA-224**: Truncated SHA-256, output 224 bits
  - **SHA-256**: 64 rounds of single steps, output 256 bits
  - **SHA-384**: Truncated SHA-512, output 384 bits
  - **SHA-512**: 80 rounds of single steps, output 512 bits

MD4

- There are three functions of three variables
  - $f\,(u, v, w)$
  - $g\,(u, v, w)$
  - $h\,(u, v, w)$

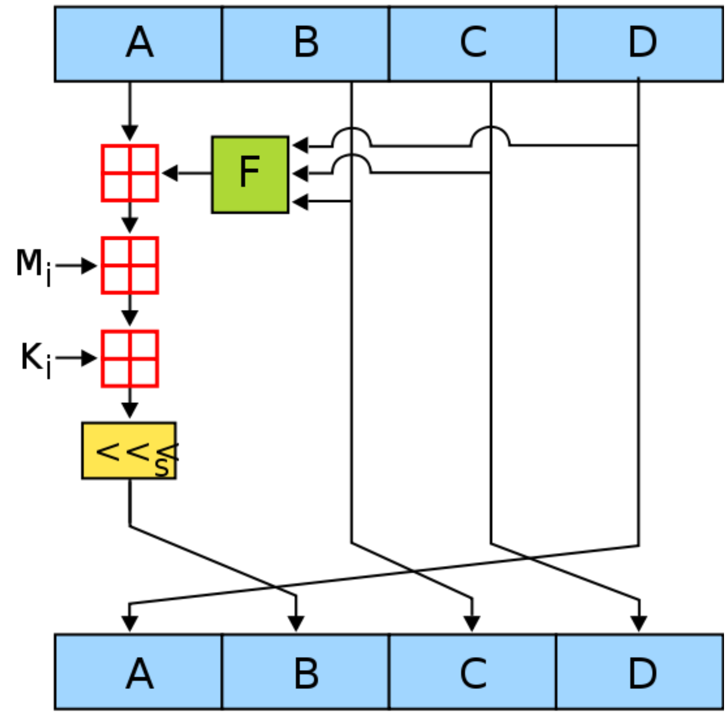
  which are just logical bitwise operations
- A current hash state (H_1, H_2, H_3, H_4) of 32-bit values initialized with a fixed IV
- There are various fixed constants (y_j, z_i, s_i)

MD4

- The data stream is loaded 16 words at a time into $X[j]$, $0 \leq j < 16$
- Execute the following steps
  - $(A, B, C, D) = (H_1, H_2, H_3, H_4)$
  - Execute Round 1
  - Execute Round 2
  - Execute Round 3
  - $(H_1, H_2, H_3, H_4) = (H_1 + A, H_2 + B, H_3 + C, H_4 + D)$
- After all data has been read in, the output is the concatenation of the final value of H_1, H_2, H_3, H_4

MD4

- Round 1
  - For $j = 0$ to 15 do
    - $t = A + f\,(B, C, D) + X\,[j] + y_j$
    - $(A, B, C, D) = (D, t \lll s_j, B, C)$
- Round 2
  - For $j = 16$ to 31 do
    - $t = A + g\,(B, C, D) + X\,[j] + y_j$
    - $(A, B, C, D) = (D, t \lll s_j, B, C)$
- Round 3
  - For $j = 32$ to 47 do
    - $t = A + h\,(B, C, D) + X\,[j] + y_j$
    - $(A, B, C, D) = (D, t \lll s_j, B, C)$



- The symbol $\lll$ denotes a bit-wise rotate to the left

Avalanche Effect

- A small change in the input produces a large unpredictable change in the output
  - A basic design principle when designing hash functions
- This design principle is typified in the MD4 family
- Several hash functions in MD4 family are widely used
  - They are all iterative in nature
  - MD5 has been popular but is not as popular now as SHA-1
  - SHA-1 (Secure Hash Algorithm) is a US and ISO standard

Avalanche Effect

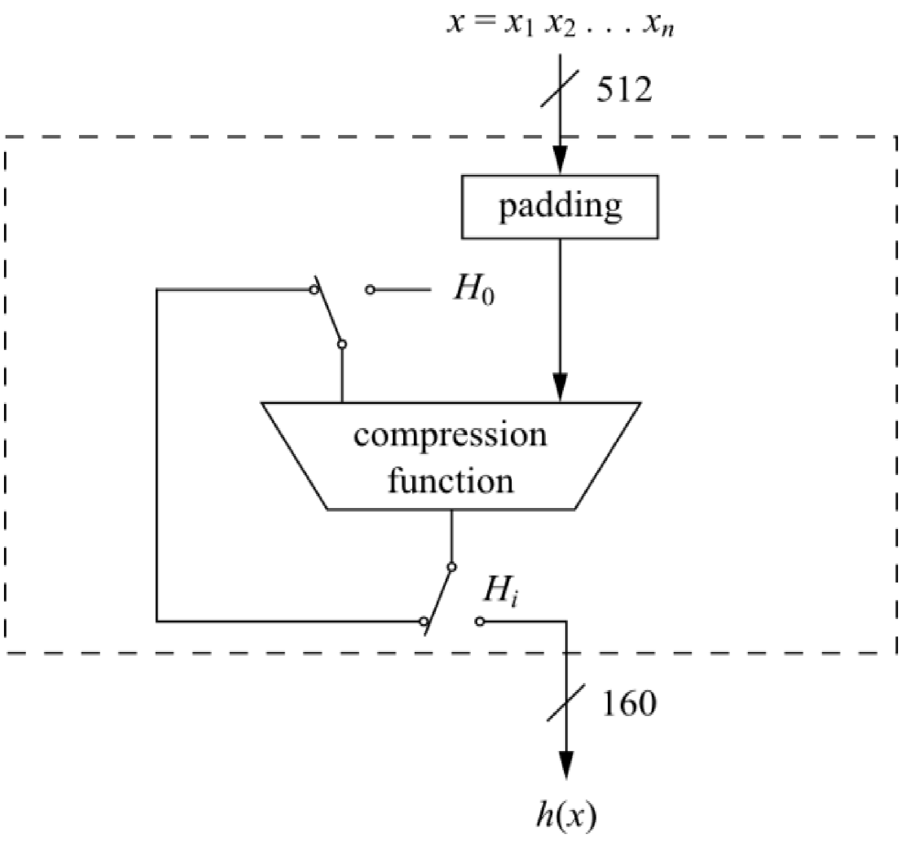
- Examples
  - MD5("The quick brown fox jumps over the lazy dog") = 9e107d9d372bb6826bd81d3542a419d6
  - MD5("The quick brown fox jumps over the lazy cog") = 1055d3e698d289f2af8663725127bd4b
  - SHA1("The quick brown fox jumps over the lazy dog") = 2fd4e1c67a2d28fced849ee1bb76e7391b93eb12
  - SHA1("The quick brown fox jumps over the lazy cog") = de9f2c7fd25e1b3afad3e85a0bd17d9b100db4b3

Secure Hash Algorithm

- SHA originally designed by NIST & NSA in 1993
- was revised in 1995 as SHA-1
- US standard for use with DSA signature scheme
  - standard is FIPS 180-1 1995, also Internet RFC3174
  - the algorithm is SHA, the standard is SHS
- based on design of MD4 with key differences
- produces 160-bit hash values
- SHA-2 is strongly recommended by NIST today
  - Four variants with output length 224, 256, 384, and 512 bits
    - http://csrc.nist.gov/CryptoToolkit/tkhash.html

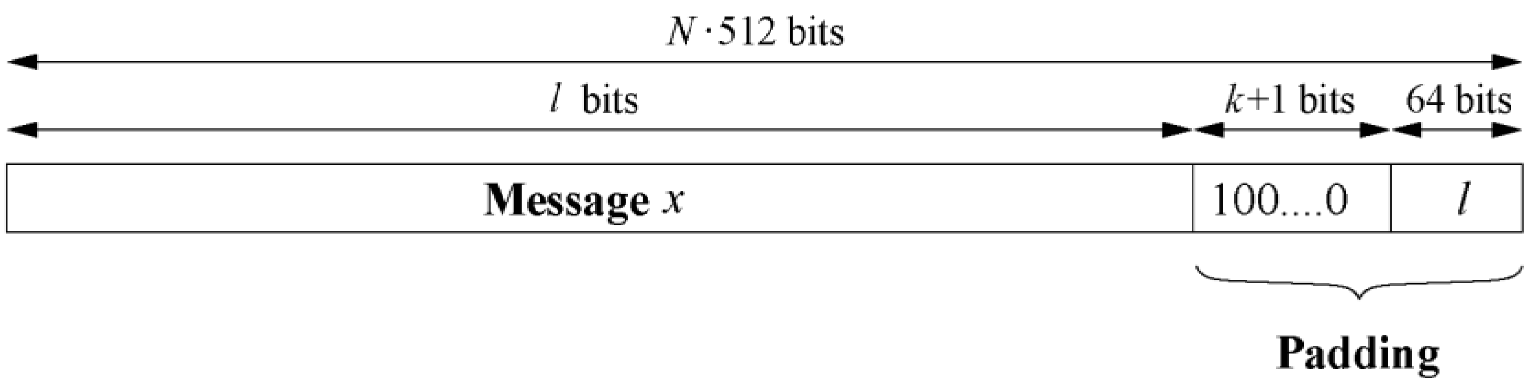
SHA-1 High Level Diagramm

- Compression Function consists of 80 rounds which are divided into four stages of 20 rounds each



SHA-1: Padding

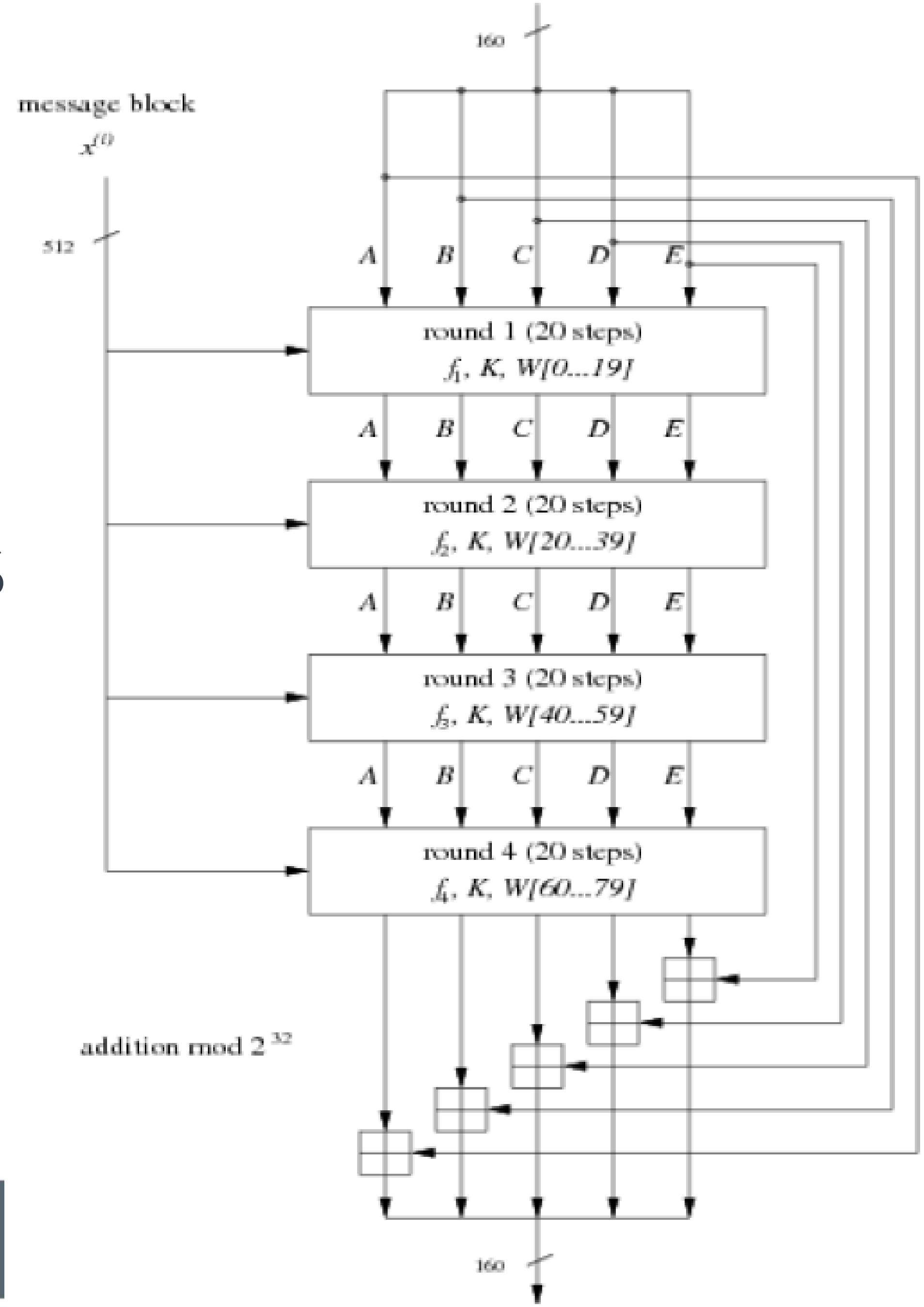
- Message x has to be padded to fit a size of a multiple of 512 bit
- $k \equiv 512 - 64 - 1 - l = 448 - (l + 1) \bmod 512$



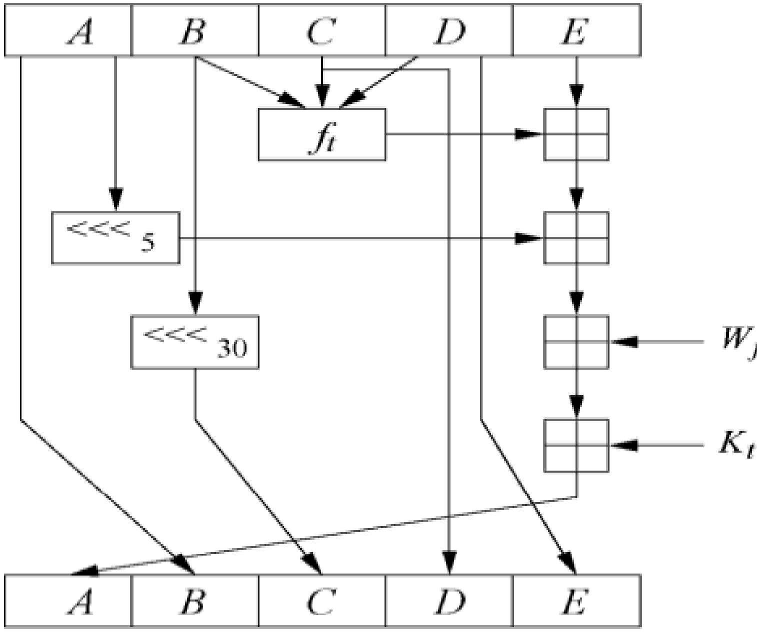
SHA-1: Hash Computation

- Each message block x_i is processed in four stages with 20 rounds each
- SHA-1 uses:
  - A message schedule which computes a 32-bit word $W_0, W_1, \ldots, W_{79}$ for each of the 80 rounds
  - Five working registers of size of 32 bits A, B, C, D, E
  - A hash value H_i consisting of five 32-bit words $H_i^{(0)}, H_i^{(1)}, H_i^{(2)}, H_i^{(3)}, H_i^{(4)}$
  - In the beginning, the hash value holds the initial value H_0, which is replaced by a new hash value after the processing of each single message block
  - The final hash value H_n is equal to the output $h(x)$ of SHA-1

SHA-1: All four stages



SHA-1: Internals of a Round



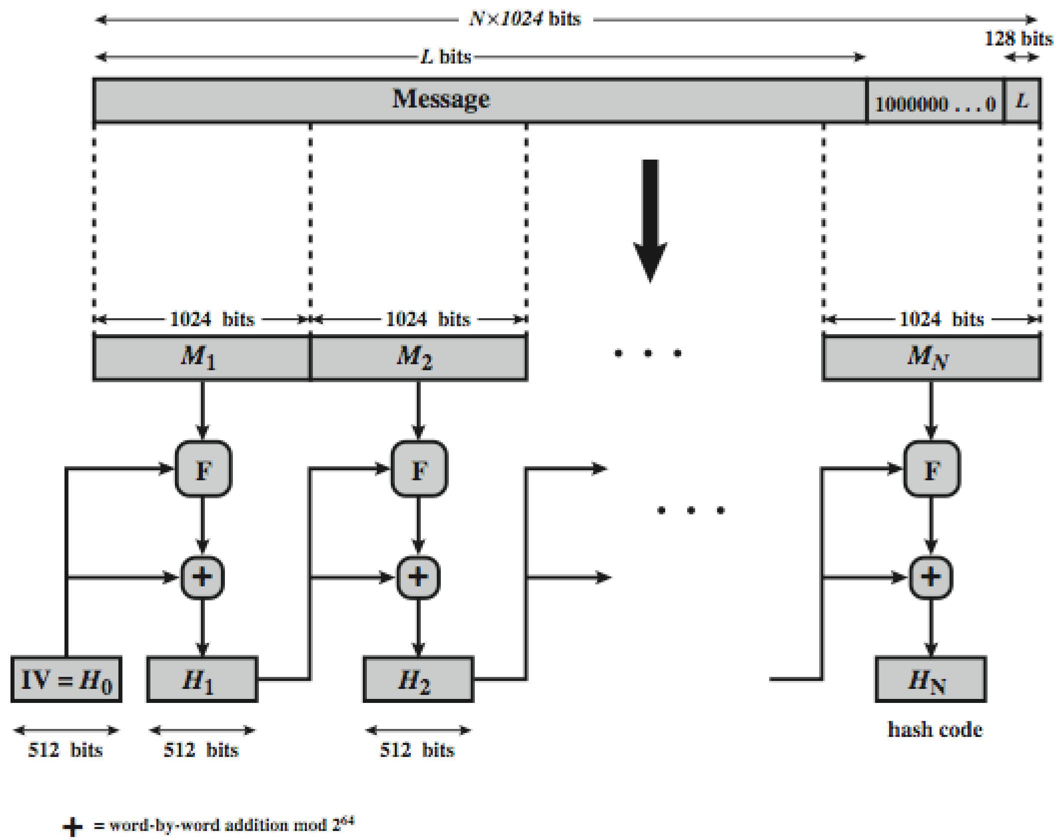
Stage t	Round j	Constant K_t	Function f_t
1	00...19	K=5A827999	$f(B,C,D) = (B \wedge C) \vee (\neg B \wedge D)$
2	20...39	K=6ED9DBA1	$f(B,C,D) = B \oplus C \oplus D$
3	40...59	K=8F1BBCDC	$f(B,C,D) = (B \oplus C) \vee (B \oplus D) \ \vee (C \oplus D)$
4	60...70	K=CA62C1D6	$f(B,C,D) = B \oplus C \oplus D$

Collisions on SHA-1

- Collisions on SHA-1
  - 王小雲 - 山東大學 數學系
  - Rump session of CRYPTO 2004
  - Security: $2^{80} \rightarrow 2^{63}$ computations
  - “Finding Collisions in the Full SHA-1”, CRYPTO 2005, Lecture Notes in Computer Science vol. 3621, pp. 17-36
- Complexity of 2^{52}?
  - McDonald, Hawkes, and Pieprzyk presented a hash collision attack with claimed complexity 2^{52} at the Rump session of Eurocrypt 2009
  - However, the accompanying paper has been withdrawn due to the authors' discovery that their estimate was incorrect

SHA-512 Overview

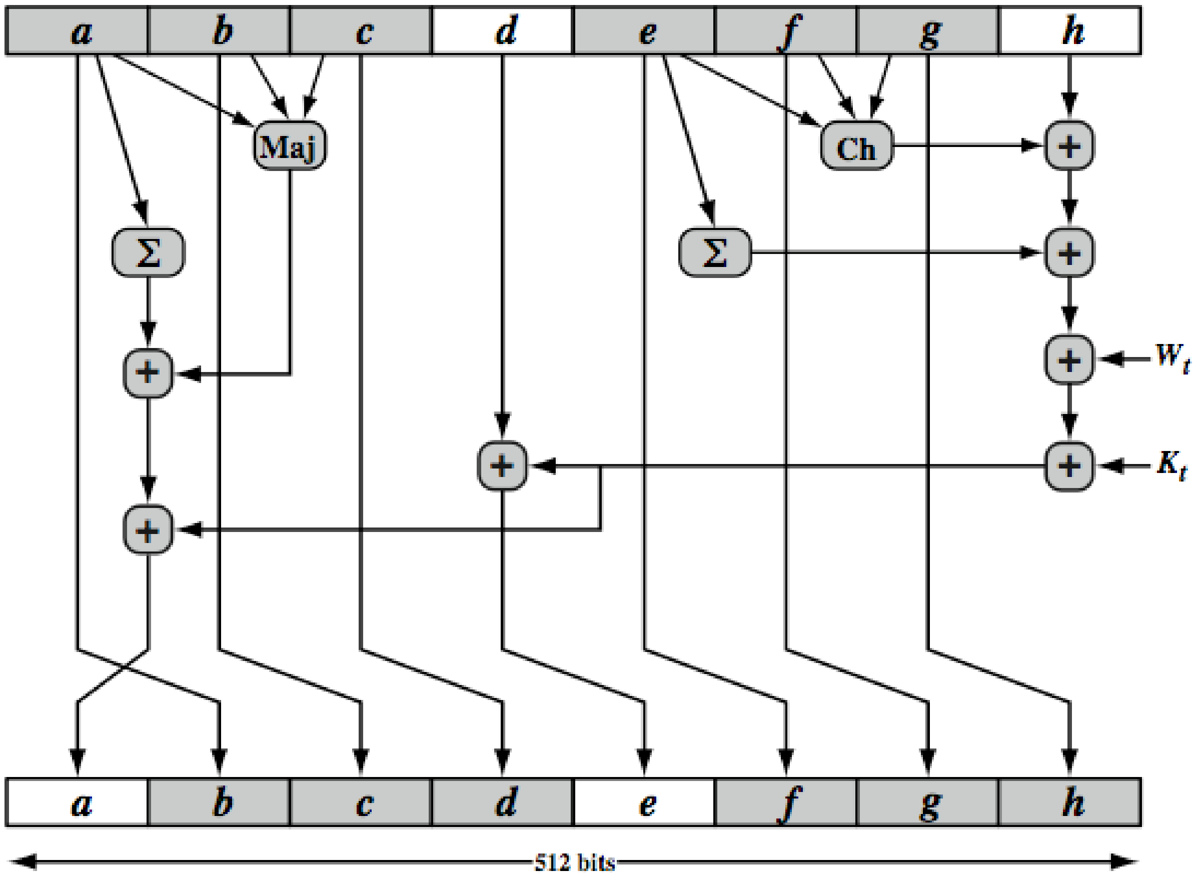
- Append padding bits
- Process the message in 1024-bit blocks
  - Each round takes as input the 512-bit buffer value H_i and updates the contents of that buffer
- Output the final state value



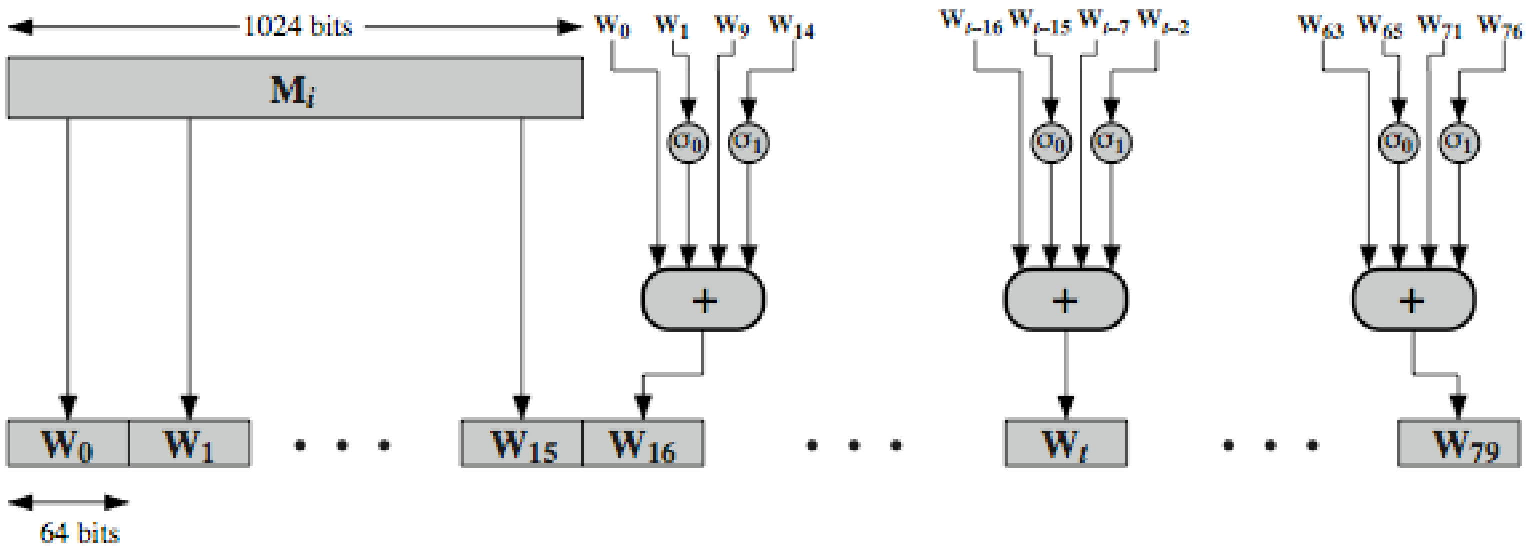
SHA-512 Compression Function

- heart of the algorithm
- processing message in 1024-bit blocks
- consists of 80 rounds
  - updating a 512-bit buffer
  - using a 64-bit value W_t derived from the current message block
  - and a round constant based on cube root of first 80 prime numbers

SHA-512 Round Function



SHA-512 Round Function



SHA-3

- The **NIST hash function competition** is an open competition held by the US NIST for a new **SHA-3** function to replace the older SHA-1 and SHA-2
  - Submissions were due Oct. 2008, with a list of candidates accepted for the first round published Dec. 2008
  - The list of 14 candidates accepted to Round 2 was published on July 24, 2009
  - Five SHA-3 finalists - BLAKE, Grøstl, JH, Keccak, and Skein advanced to the third (and final) round on December 9, 2010
  - October 2, 2012, Keccak was selected as the winner

Lessons Learned

- Hash functions are keyless. The two most important applications of hash functions are their use in digital signatures and in message authentication codes such as HMAC
- The three security requirements for hash functions are one-wayness, second preimage resistance and collision resistance
- Hash functions should have at least 160-bit output length in order to withstand collision attacks; 256 bit or more is desirable for long-term security
- MD5, which was widely used, is insecure. Serious security weaknesses have been found in SHA-1, and the hash functionshould be phased out. The SHA-2 algorithms all appear to be secure
- The ongoing SHA-3 competition will result in new standardized hash functions in a few years