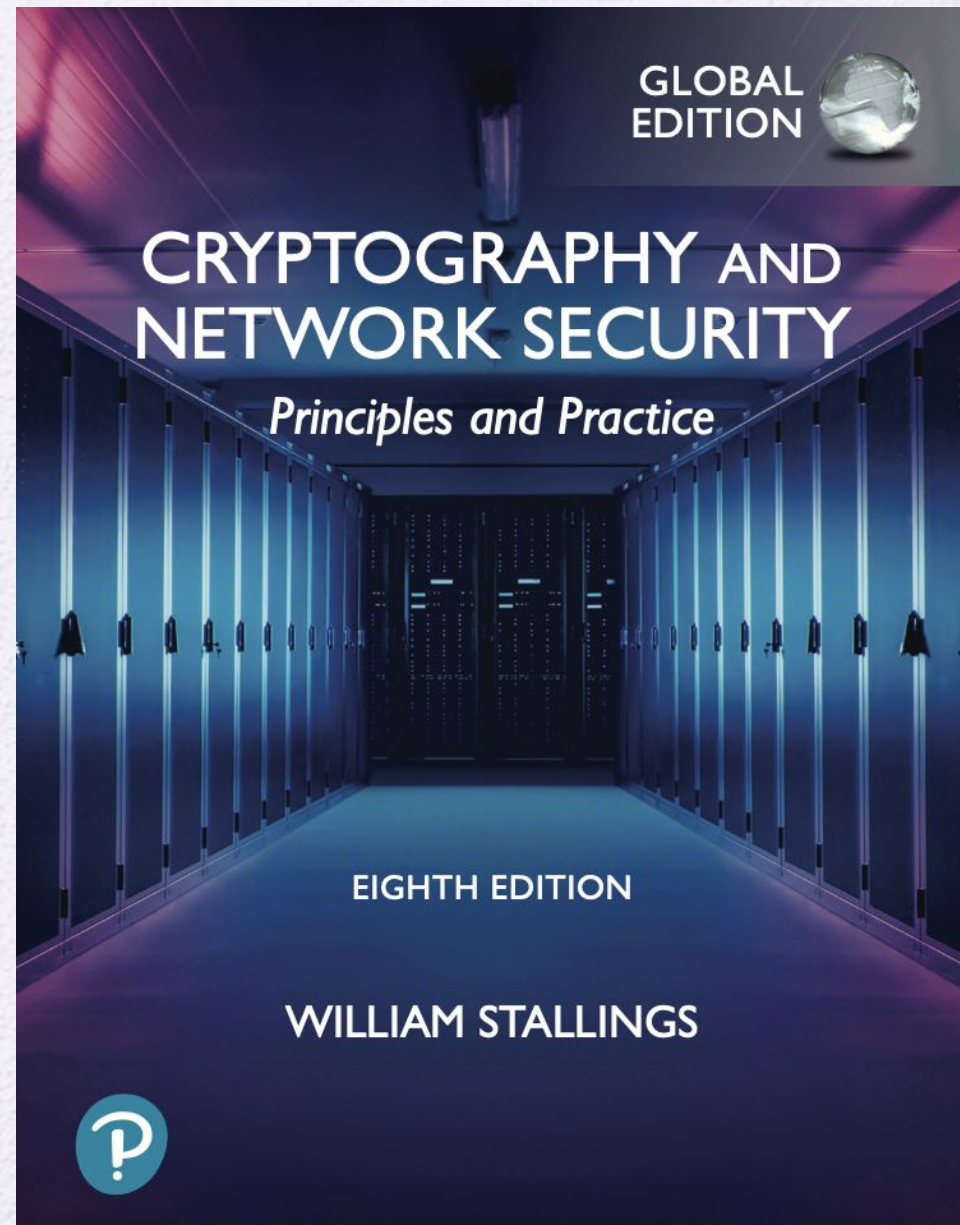


University of Nevada – Reno
Computer Science & Engineering
Department

CS454/654 Reliability and Security
of Computing Systems - Fall 2024

Lecture 14

Dr. Batyr Charyyev
bcharyyev.com



PART FOUR: CRYPTOGRAPHIC DATA INTEGRITY ALGORITHMS

CHAPTER 11

CRYPTOGRAPHIC HASH FUNCTIONS

11.1 Applications of Cryptographic Hash Functions

- Message Authentication
- Digital Signatures
- Other Applications

11.2 Two Simple Hash Functions

11.3 Requirements and Security

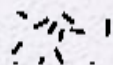
- Security Requirements for Cryptographic Hash Functions
- Brute-Force Attacks
- Cryptanalysis

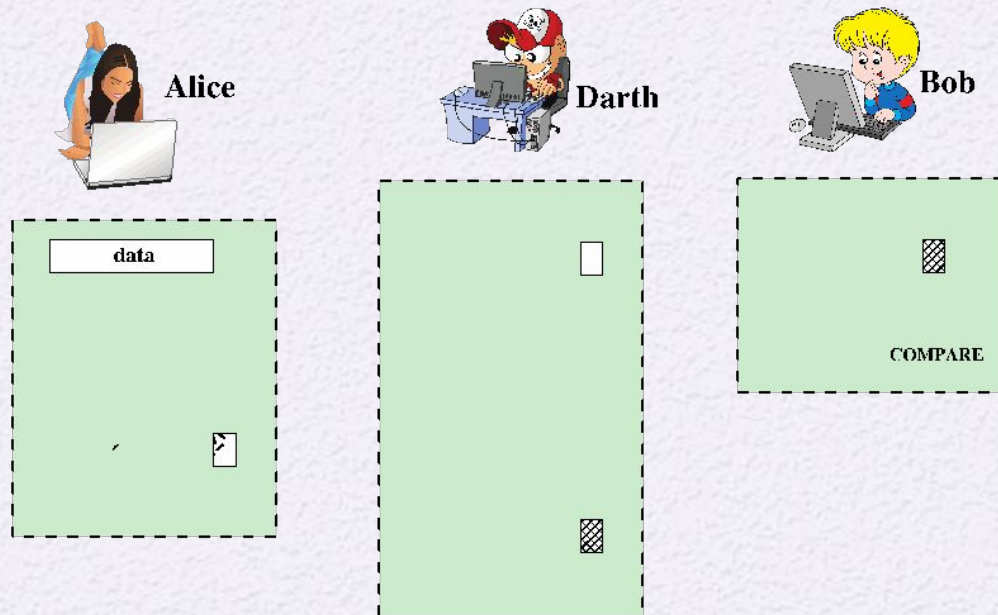
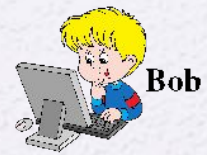
11.4 Secure Hash Algorithm (SHA)

- SHA-512 Logic
- SHA-512 Round Function
- Example

Hash Functions

- A hash function H accepts a variable-length block of data M as input and produces a fixed-size hash value
 - $h = H(M)$
 - Principal object is data integrity
- Cryptographic hash function
 - An algorithm for which it is computationally infeasible to find either:
 - (a) a data object that maps to a pre-specified hash result (the one-way property)
 - (b) two data objects that map to the same hash result (the collision-free property)





(b) Man-in-the-middle attack

Figure 11.2 Attack Against Hash Function

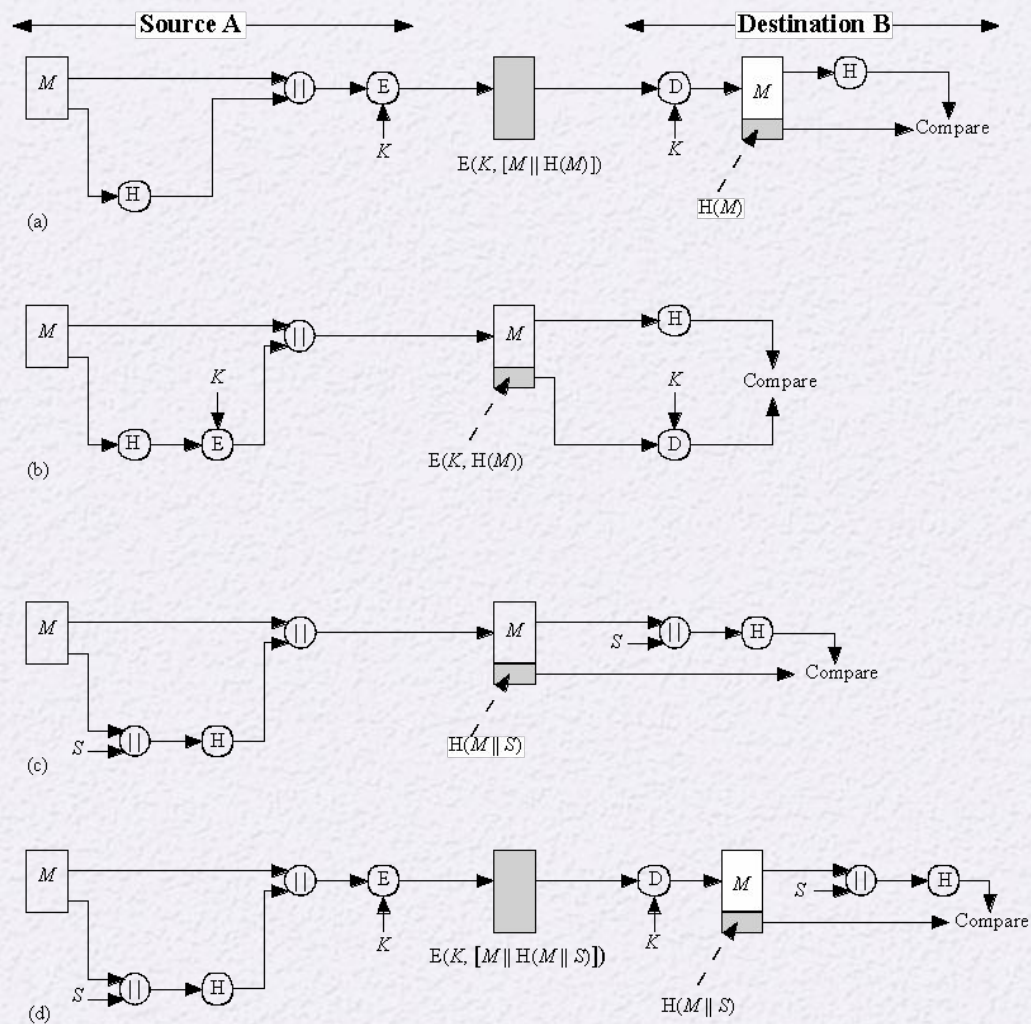


Figure 11.3 Simplified Examples of the Use of a Hash Function for Message Authentication

Message Authentication Code (MAC)

- Also known as a *keyed hash function*
- Typically used between two parties that share a secret key to authenticate information exchanged between those parties

Takes as input a secret key and a data block and produces a hash value (MAC) which is associated with the protected message

- If the integrity of the message needs to be checked, the MAC function can be applied to the message and the result compared with the associated MAC value
- An attacker who alters the message will be unable to alter the associated MAC value without knowledge of the secret key

Digital Signature

- Operation is similar to that of the MAC
- The hash value of a message is encrypted with a user's private key
- Anyone who knows the user's public key can verify the integrity of the message
- An attacker who wishes to alter the message would need to know the user's private key
- Implications of digital signatures go beyond just message authentication

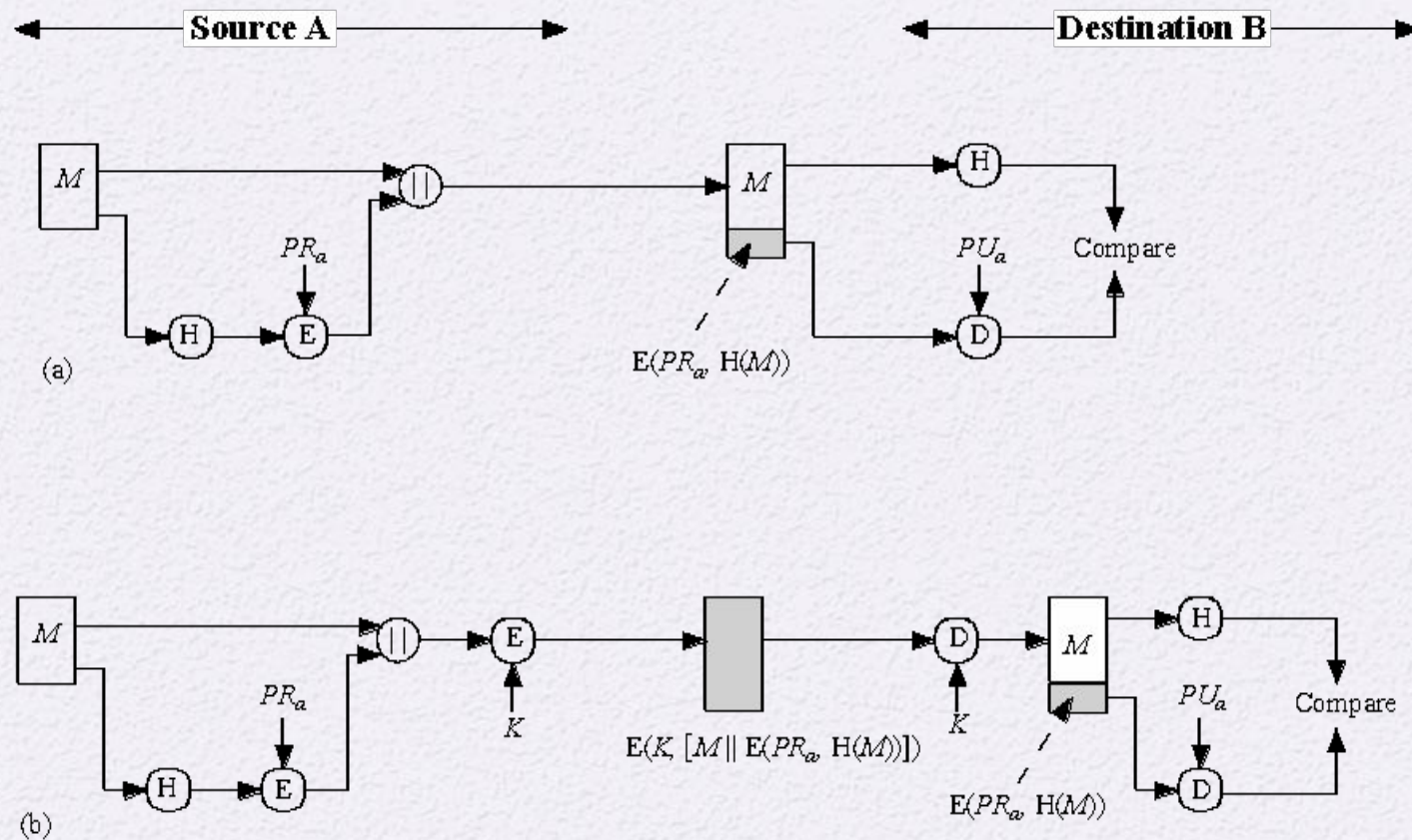


Figure 11.4 Simplified Examples of Digital Signatures

Other Hash Function Uses

Commonly used to create a one-way password file

When a user enters a password, the hash of that password is compared to the stored hash value for verification

This approach to password protection is used by most operating systems

Can be used for intrusion and virus detection

Store $H(F)$ for each file on a system and secure the hash values

One can later determine if a file has been modified by recomputing $H(F)$

An intruder would need to change F without changing $H(F)$

Can be used to construct a pseudorandom function (PRF) or a pseudorandom number generator (PRNG)

A common application for a hash-based PRF is for the generation of symmetric keys

Two Simple Hash Functions

- Consider two simple insecure hash functions that operate using the following general principles:
 - The input is viewed as a sequence of n -bit blocks
 - The input is processed one block at a time in an iterative fashion to produce an n -bit hash function
- Bit-by-bit exclusive-OR (XOR) of every block
 - $C_i = b_{i1} \text{ xor } b_{i2} \text{ xor } \dots \text{ xor } b_{im}$
 - Produces a simple parity for each bit position and is known as a longitudinal redundancy check
 - Reasonably effective for random data as a data integrity check
- Perform a one-bit circular shift on the hash value after each block is processed
 - Has the effect of randomizing the input more completely and overcoming any regularities that appear in the input

Two Simple Hash Functions

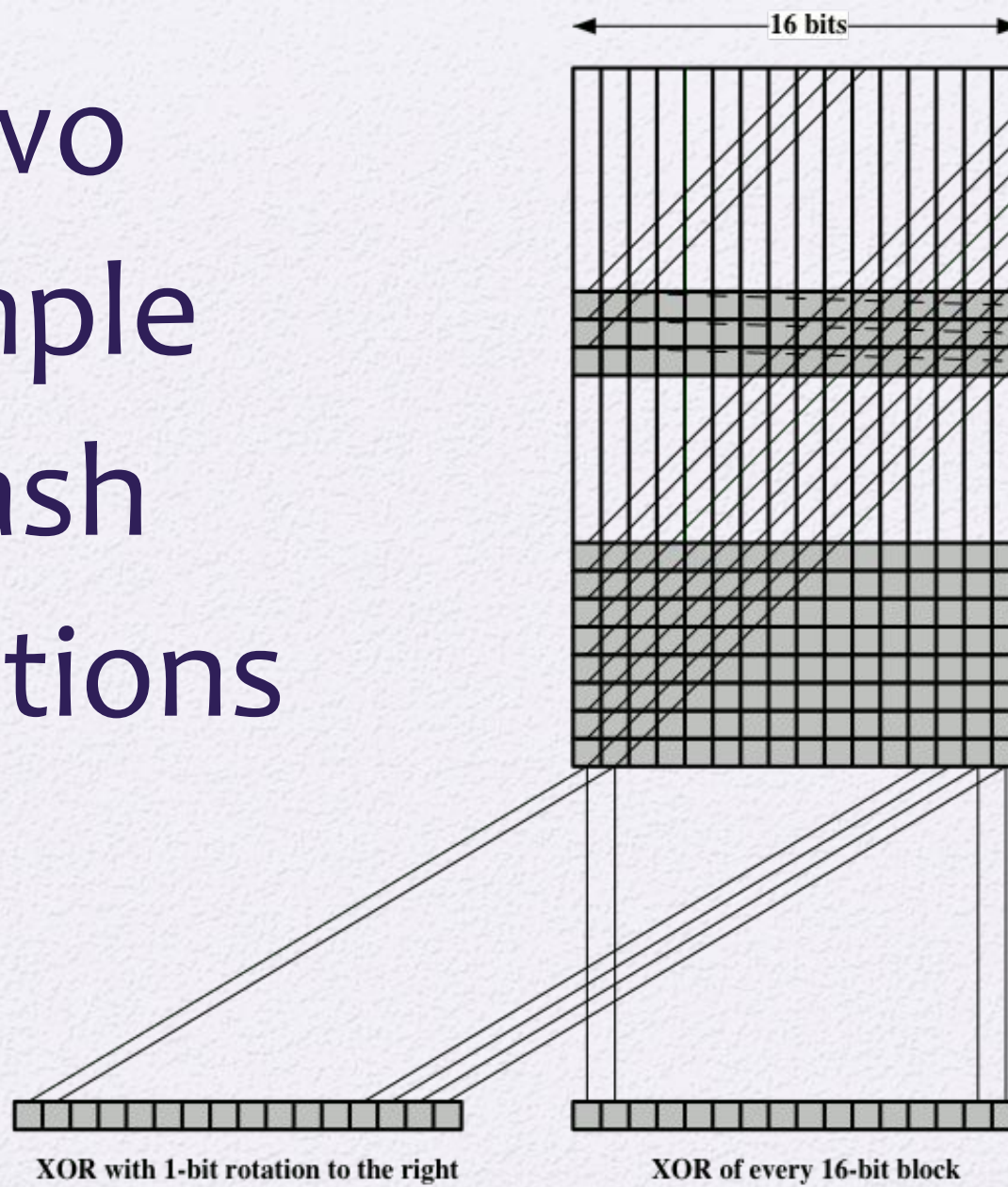


Figure 11.5 Two Simple Hash Functions

Requirements and Security

Preimage

- x is the preimage of h for a hash value $h = H(x)$
- Is a data block whose hash function, using the function H , is h
- Because H is a many-to-one mapping, for any given hash value h , there will in general be multiple preimages

Collision

- Occurs if we have $x \neq y$ and $H(x) = H(y)$
- Because we are using hash functions for data integrity, collisions are clearly undesirable



Table 11.1

Requirements for a Cryptographic Hash Function H

Requirement	Description
Variable input size	H can be applied to a block of data of any size.
Fixed output size	H produces a fixed-length output.
Efficiency	$H(x)$ is relatively easy to compute for any given x , making both hardware and software implementations practical.
Preimage resistant (one-way property)	For any given hash value h , it is computationally infeasible to find y such that $H(y) = h$.
Second preimage resistant (weak collision resistant)	For any given block x , it is computationally infeasible to find $y \neq x$ with $H(y) = H(x)$.
Collision resistant (strong collision resistant)	It is computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$.
Pseudorandomness	Output of H meets standard tests for pseudorandomness

(Table can be found on page 327 in textbook.)

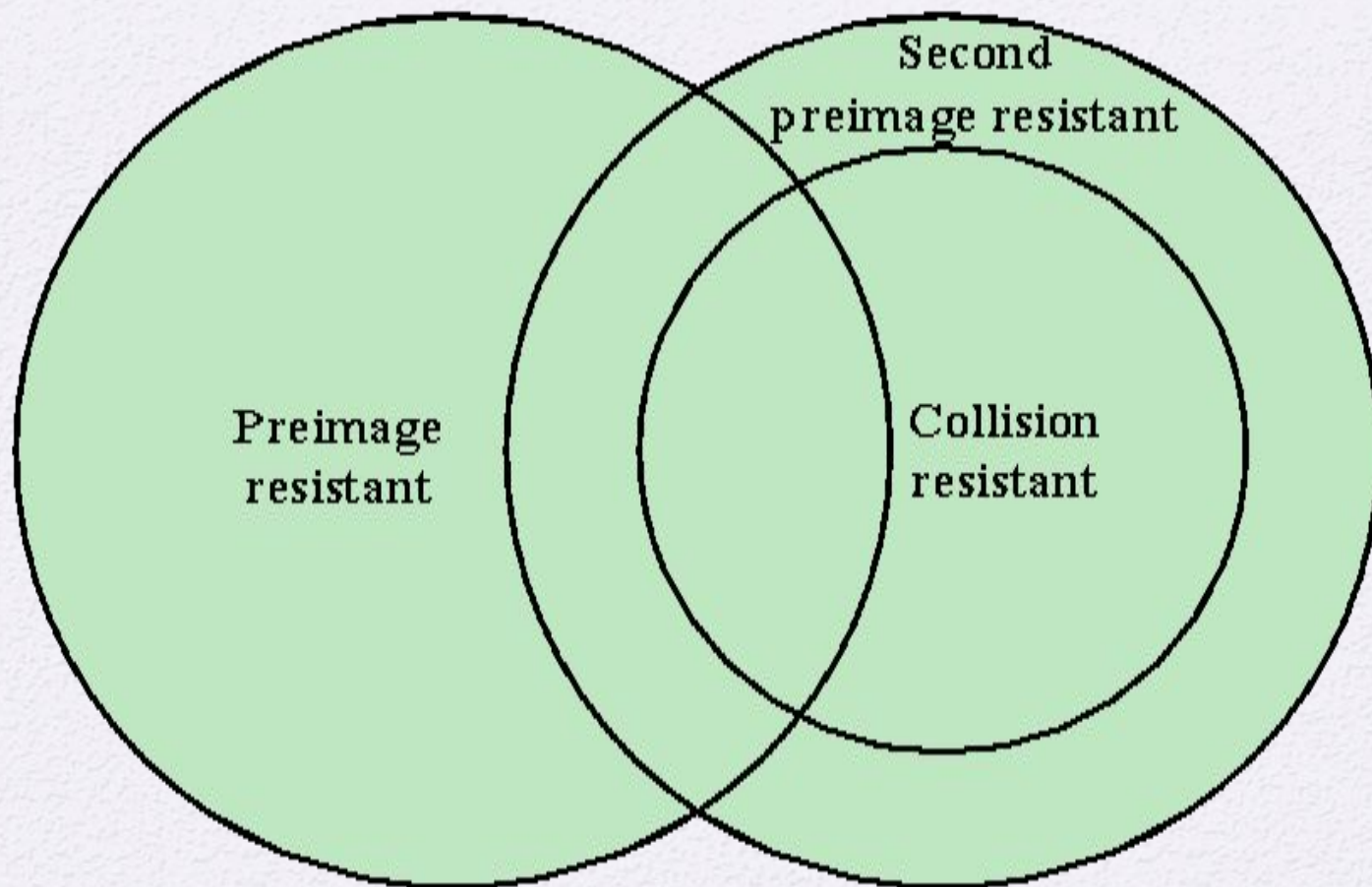


Figure 11.6 Relationship Among Hash Function Properties

Table 11.2

Hash Function Resistance Properties Required for Various Data Integrity Applications

	Preimage Resistant	Second Preimage Resistant	Collision Resistant
Hash + digital signature	yes	yes	yes*
Intrusion detection and virus detection		yes	
Hash + symmetric encryption			
One-way password file	yes		
MAC	yes	yes	yes*

* Resistance required if attacker is able to mount a chosen message attack

Attacks on Hash Functions

Brute-Force Attacks

- Does not depend on the specific algorithm, only depends on bit length
- In the case of a hash function, attack depends only on the bit length of the hash value
- Method is to pick values at random and try each one until a collision occurs

Cryptanalysis

- An attack based on weaknesses in a particular cryptographic algorithm
- Seek to exploit some property of the algorithm to perform some attack other than an exhaustive search



Collision Resistant Attacks

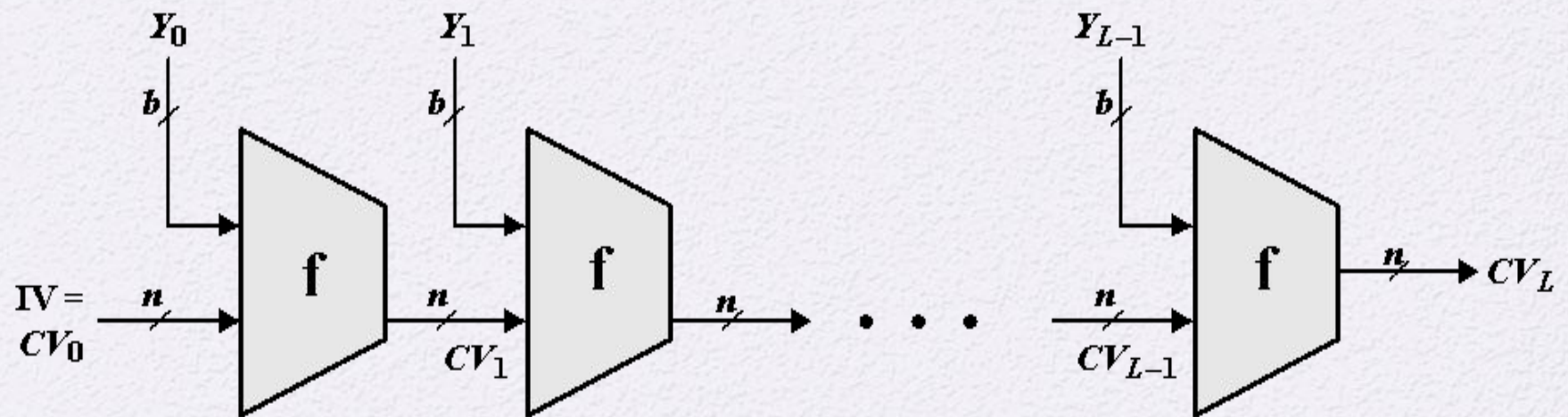
- For a collision resistant attack, an adversary wishes to find two messages or data blocks that yield the same hash function
 - The effort required is explained by a mathematical result referred to as the *birthday paradox*
- Yuval proposed the following strategy to exploit the birthday paradox in a collision resistant attack:
 - The source (A) is prepared to sign a legitimate message x by appending the appropriate m -bit hash code and encrypting that hash code with A's private key
 - Opponent generates $2^{m/2}$ variations x' of x , all with essentially the same meaning, and stores the messages and their hash values
 - Opponent prepares a fraudulent message y for which A's signature is desired
 - Opponent generates minor variations y' of y , all of which convey essentially the same meaning. For each y' , the opponent computes $H(y')$, checks for matches with any of the $H(x')$ values, and continues until a match is found. That is, the process continues until a y' is generated with a hash value equal to the hash value of one of the x' values
 - The opponent offers the valid variation to A for signature which can then be attached to the fraudulent variation for transmission to the intended recipient
 - Because the two variations have the same hash code, they will produce the same signature and the opponent is assured of success even though the encryption key is not known

A Letter in 2^{38} Variation

(Letter is located on page 330 in textbook)

As { the } Dean of Blakewell College, I have { had the pleasure of knowing } Cherise
 Rosetti for the { last } four years. She { has been } { a tremendous } { asset to }
 { past } { was } { an outstanding } { role model in }
 { our } school. I { would like to take this opportunity to } recommend Cherise for your
 { the } { wholeheartedly }
 { school's } graduate program. I { am } { confident } { that } { she } will
 { -- } { feel } { certain } { -- } { Cherise }
 { continue to } succeed in her studies. { She } is a dedicated student and
 { -- } { Cherise }
 { thus far her grades } { have been } { exemplary } . In class, { she }
 { her grades thus far } { are } { excellent } { Cherise }
 { has proven to be } a take-charge { person } { who is } able to successfully develop
 { has been } { individual } { -- } plans and implement them.
 { She } has also assisted { us } in our admissions office. { She } has
 { Cherise } { -- }
 { successfully } demonstrated leadership ability by counseling new and prospective students.
 { -- }
 { Her } advice has been { a great } help to these students, many of whom have
 { Cherise's } { of considerable }
 { taken time to share } their comments with me regarding her pleasant and { encouraging }
 { shared } { reassuring }
 attitude. { For these reasons } I { highly recommend } Cherise
 { It is for these reasons that } { offer high recommendations for }
 { without reservation } . Her { ambition } and { abilities } will { truly } be an
 { unreservedly } { drive } { potential } { surely }
 { asset to } your { establishment } .
 { plus for } { school }

Figure 11.7 A Letter in 2^{38} Variations



IV = Initial value
 CV_i = chaining variable
 Y_i = i th input block
 f = compression algorithm

L = number of input blocks
 n = length of hash code
 b = length of input block

Figure 11.8 General Structure of Secure Hash Code

Secure Hash Algorithm (SHA)

- SHA was originally designed by the National Institute of Standards and Technology (NIST) and published as a federal information processing standard (FIPS 180) in 1993
- Was revised in 1995 as SHA-1
- Based on the hash function MD4 and its design closely models MD4
- Produces 160-bit hash values
- In 2002 NIST produced a revised version of the standard that defined three new versions of SHA with hash value lengths of 256, 384, and 512
 - Collectively known as SHA-2

Table 11.3

Comparison of SHA Parameters

Algorithm	Message Size	Block Size	Word Size	Message Digest Size
SHA-1	$< 2^{64}$	512	32	160
SHA-224	$< 2^{64}$	512	32	224
SHA-256	$< 2^{64}$	512	32	256
SHA-384	$< 2^{128}$	1024	64	384
SHA-512	$< 2^{128}$	1024	64	512
SHA-512/224	$< 2^{128}$	1024	64	224
SHA-512/256	$< 2^{128}$	1024	64	256

Note: All sizes are measured in bits.

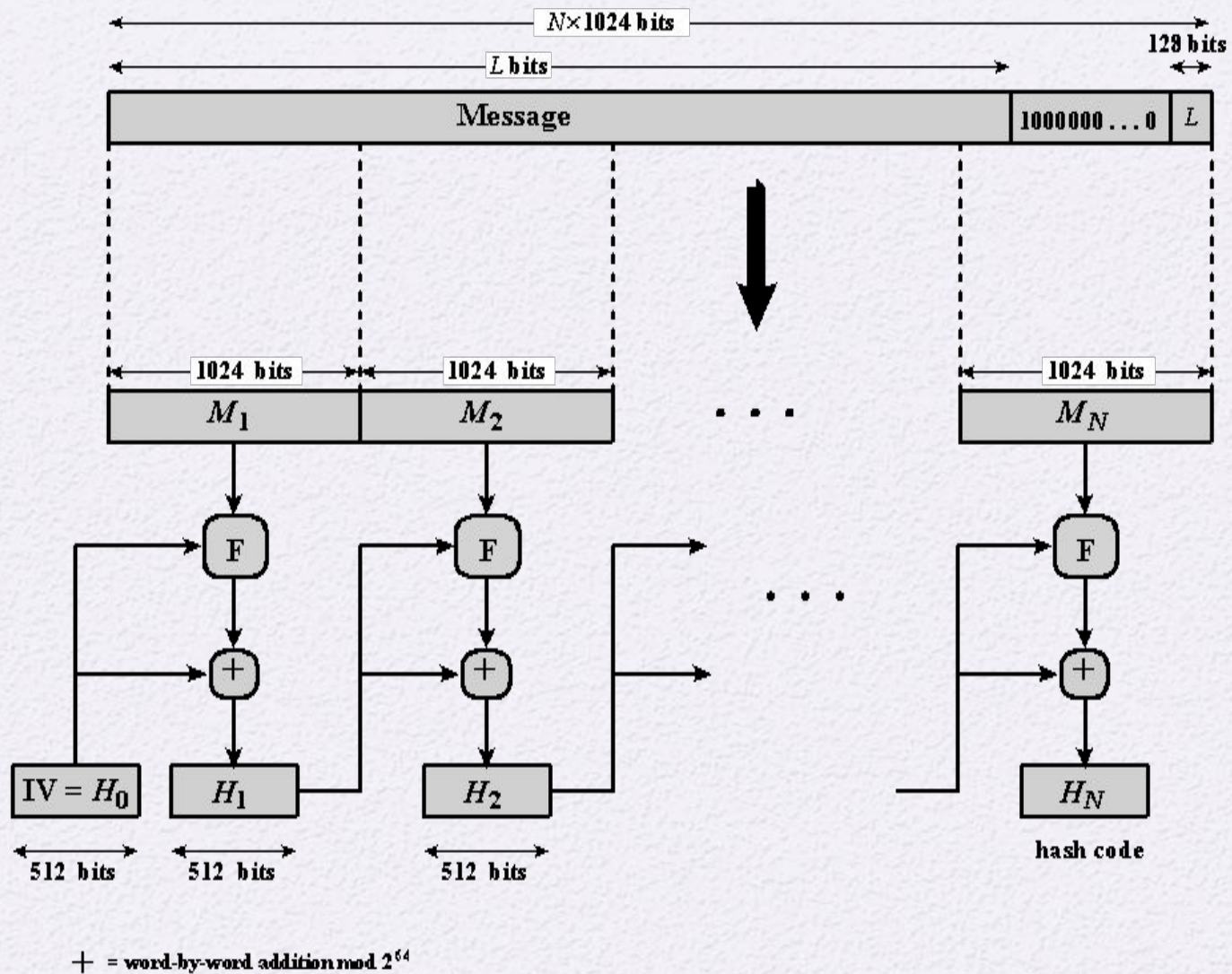


Figure 11.9 Message Digest Generation Using SHA-512

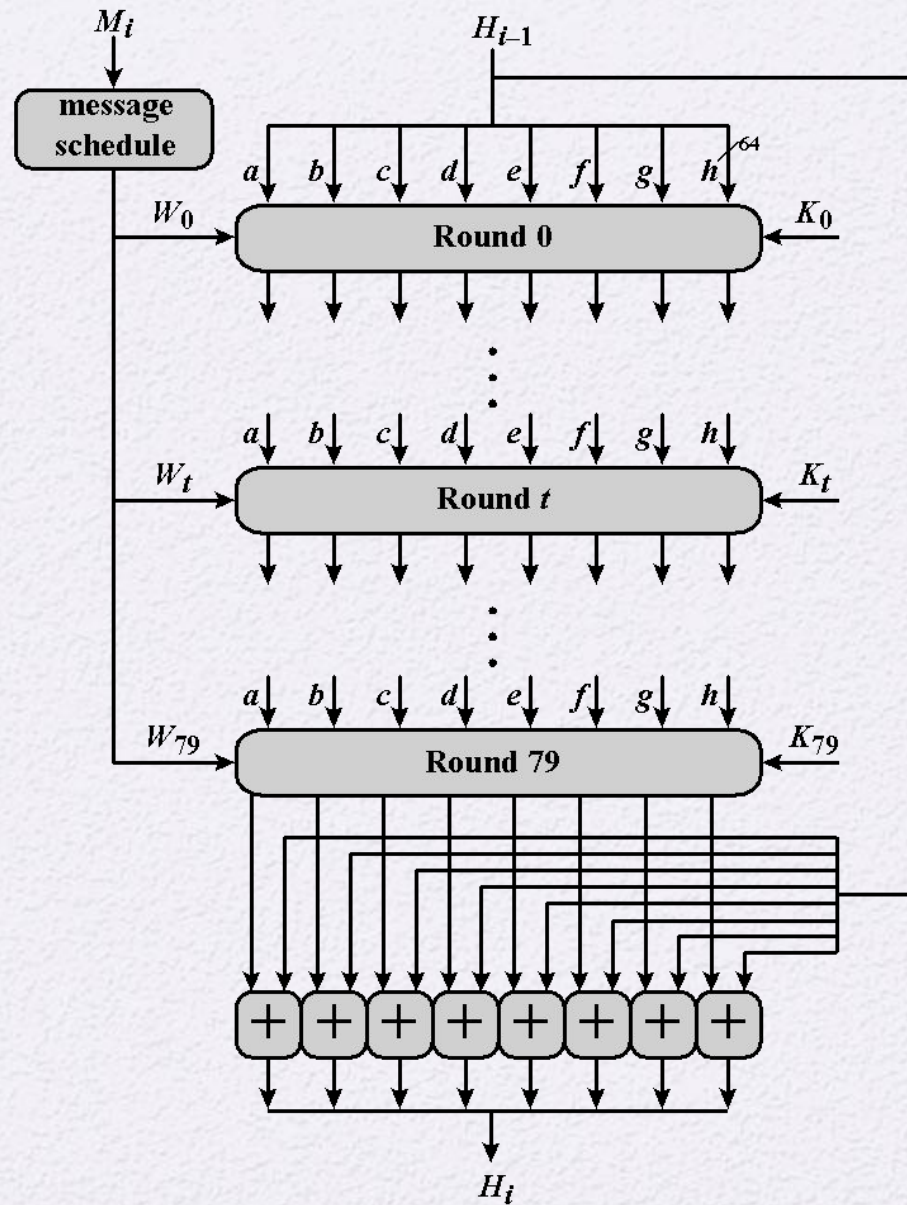


Figure 11.10 SHA-512 Processing of a Single 1024-Bit Block

Table 11.4

SHA-512

Constants

428a2f98d728ae22 e9b5dba58189dbbc	7137449123ef65cd	b5c0fbcfec4d3b2f
3956c25bf348b538 ab1c5ed5da6d8118	59f111f1b605d019	923f82a4af194f9b
d807aa98a3030242 550c7dc3d5ffb4e2	12835b0145706fbe	243185be4ee4b28c
72be5d74f27b896f c19bf174cf692694	80deb1fe3b1696b1	9bdc06a725c71235
e49b69c19ef14ad2 240ca1cc77ac9c65	efbe4786384f25e3	0fc19dc68b8cd5b5
2de92c6f592b0275 76f988da831153b5	4a7484aa6ea6e483	5cb0a9dcbd41fbd4
983e5152ee66dfab bf597fc7beef0ee4	a831c66d2db43210	b00327c898fb213f
c6e00bf33da88fc2 142929670a0e6e70	d5a79147930aa725	06ca6351e003826f
27b70a8546d22ffc 53380d139d95b3df	2e1b21385c26c926	4d2c6dfc5ac42aed
650a73548baf63de 92722c851482353b	766a0abb3c77b2a8	81c2c92e47edae6
a2bfe8a14cf10364 c76c51a30654be30	a81a664bbc423001	c24b8b70d0f89791
d192e819d6ef5218 106aa07032bbd1b8	d69906245565a910	f40e35855771202a
19a4c116b8d2d0c8 34b0bcb5e19b48a8	1e376c085141ab53	2748774cdf8eeb99
391c0cb3c5c95a63 682e6ff3d6b2b8a3	4ed8aa4ae3418acb	5b9cca4f7763e373
748f82ee5defb2fc 8cc702081a6439ec	78a5636f43172f60	84c87814a1f0ab72
90beffffa23631e28 c67178f2e372532b	a4506cebd82bde9	bef9a3f7b2c67915
ca273eceeaa26619c f57d4f7fee6ed178	d186b8c721c0c207	eada7dd6cde0eb1e
06f067aa72176fba 1b710b35131c471b	0a637dc5a2c898a6	113f9804bef90dae
28db77f523047d84 431d67c49c100d4c	32caab7b40c72493	3c9ebe0a15c9bebc
4cc5d4becb3e42b6 6c44198c4a475817	597f299cfc657e2a	5fcb6fab3ad6faec

(Table can
be found
on page
336 in
textbook)

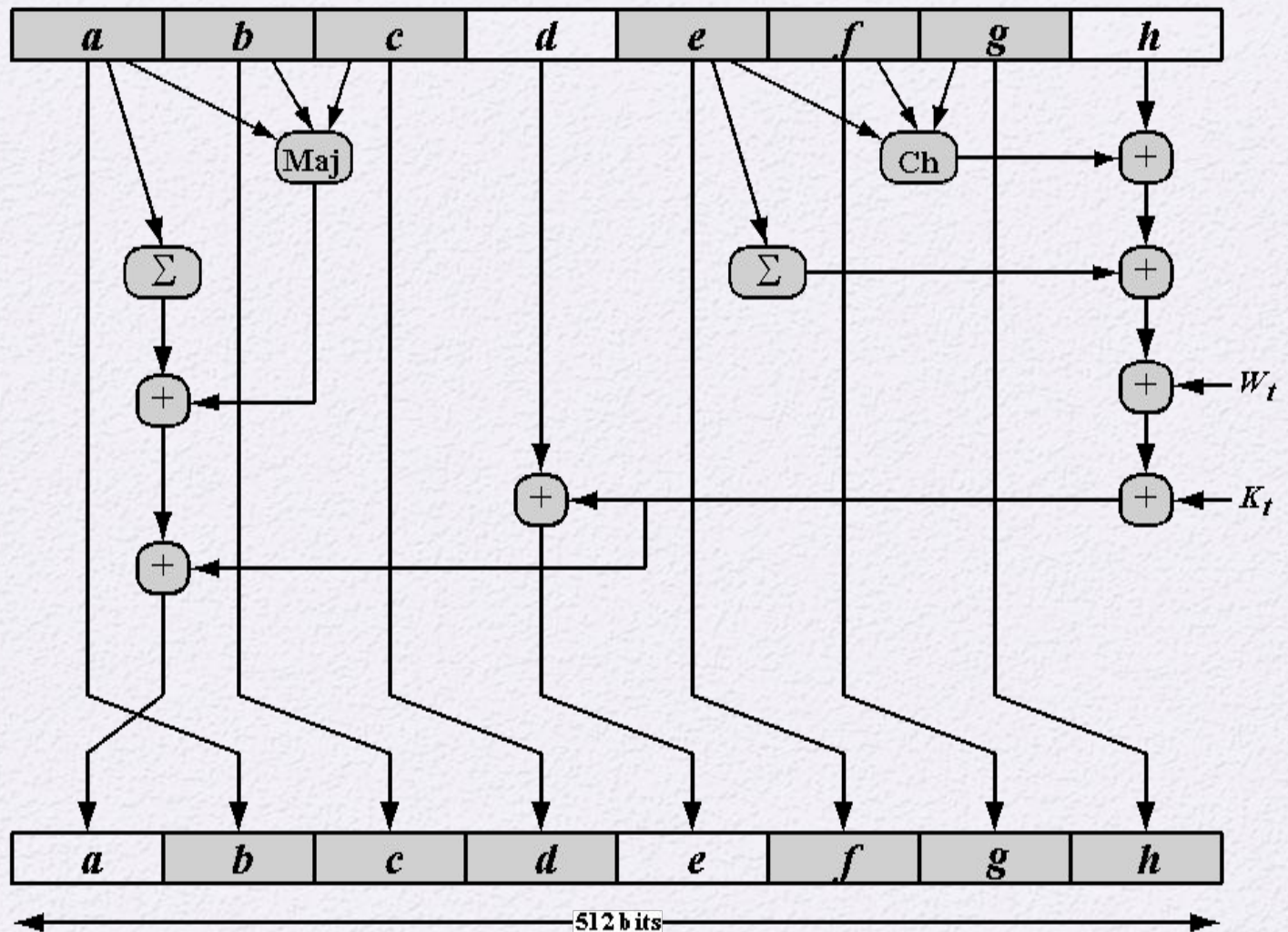


Figure 11.11 Elementary SHA-512 Operation (single round)

$$T_1 = h + \text{Ch}(e, f, g) + (\sum_1^{512} e) + W_t + K_t$$

$$T_2 = (\sum_0^{512} a) + \text{Maj}(a, b, c)$$

$$h = g$$

$$g = f$$

$$f = e$$

$$e = d + T_1$$

$$d = c$$

$$c = b$$

$$b = a$$

$$a = T_1 + T_2$$

where

t = step number; $0 \leq t \leq 79$

$\text{Ch}(e, f, g) = (e \text{ AND } f) \oplus (\text{NOT } e \text{ AND } g)$

the conditional function: If e then f else g

$\text{Maj}(a, b, c) = (a \text{ AND } b) \oplus (a \text{ AND } c) \oplus (b \text{ AND } c)$

the function is true only if the majority (two or three) of the arguments are true

$(\sum_0^{512} a) = \text{ROTR}^{28}(a) \oplus \text{ROTR}^{34}(a) \oplus \text{ROTR}^{39}(a)$

$(\sum_1^{512} e) = \text{ROTR}^{14}(e) \oplus \text{ROTR}^{18}(e) \oplus \text{ROTR}^{41}(e)$

$\text{ROTR}^n(x)$ = circular right shift (rotation) of the 64-bit argument x by n bits

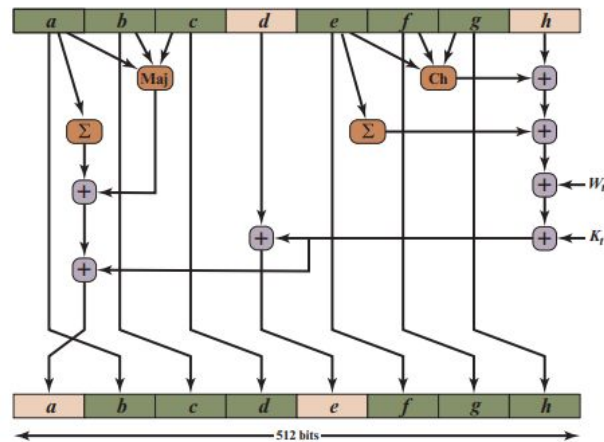
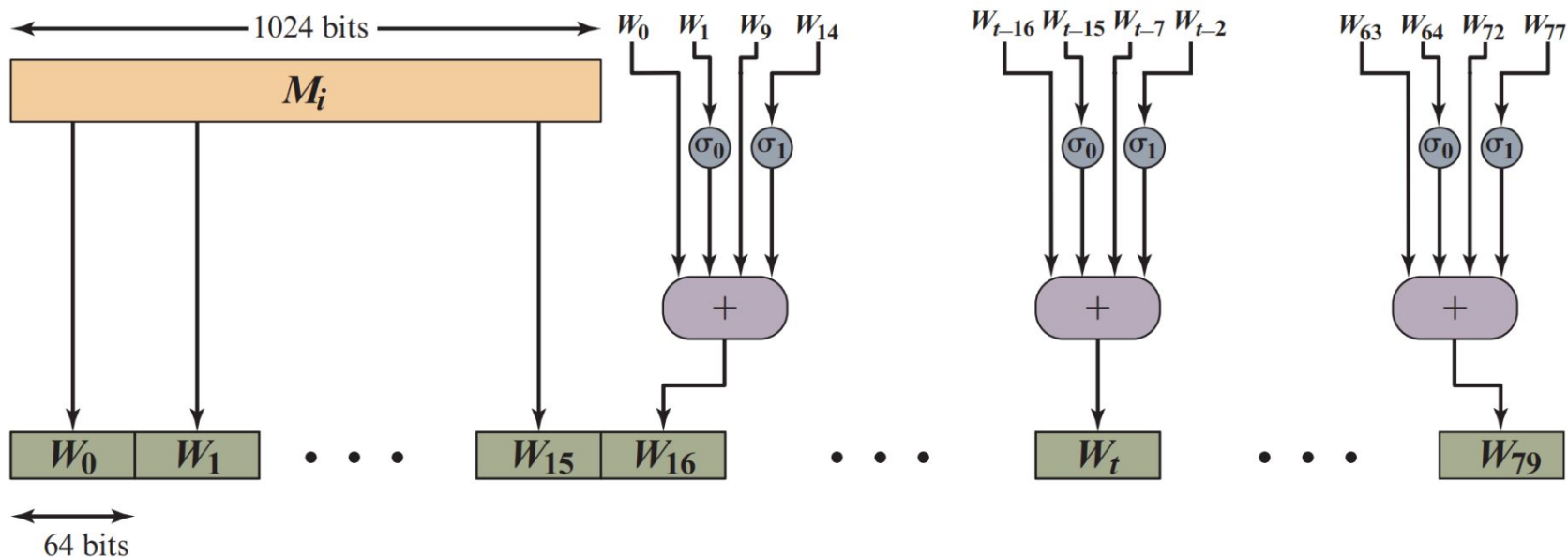


Figure 11.11 Elementary SHA-512 Operation (single round)



$$W_t = \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16}$$

where

$$\sigma_0^{512}(x) = \text{ROTR}^1(x) \oplus \text{ROTR}^8(x) \oplus \text{SHR}^7(x)$$

$$\sigma_1^{512}(x) = \text{ROTR}^{19}(x) \oplus \text{ROTR}^{61}(x) \oplus \text{SHR}^6(x)$$

$\text{ROTR}^n(x)$ = circular right shift (rotation) of the 64-bit argument x by n bits

$\text{SHR}^n(x)$ = right shift of the 64-bit argument x by n bits with padding by zeros on the left

$+$ = addition modulo 2^{64}

The padded message consists blocks M_1, M_2, \dots, M_N . Each message block M_i consists of 16 64-bit words $M_{i,0}, M_{i,1}, \dots, M_{i,15}$. All addition is performed modulo 2^{64} .

$H_{0,0} = 6A09E667F3BCC908$	$H_{0,4} = 510E527FADE682D1$
$H_{0,1} = BB67AE8584CAA73B$	$H_{0,5} = 9B05688C2B3E6C1F$
$H_{0,2} = 3C6EF372FE94F82B$	$H_{0,6} = 1F83D9ABFB41BD6B$
$H_{0,3} = A54FF53A5F1D36F1$	$H_{0,7} = 5BE0CDI9137E2179$

for $i = 1$ to N

1. Prepare the message schedule W :

for $t = 0$ to 15

$W_t = M_{i,t}$

for $t = 16$ to 79

$W_t = \left(\sum_{j=0}^{512} (W_{t-2}) + W_{t-7} + \sum_{j=0}^{512} (W_{t-15}) + W_{t-16} \right)$

2. Initialize the working variables

$a = H_{i-1,0}$ $e = H_{i-1,4}$

$b = H_{i-1,1}$ $f = H_{i-1,5}$

$c = H_{i-1,2}$ $g = H_{i-1,6}$

$d = H_{i-1,3}$ $h = H_{i-1,7}$

3. Perform the main hash computation

for $t = 0$ to 79

$T_1 = h + \text{Ch}(e, f, g) + \left(\sum_{j=0}^{512} e \right) + W_t + K_t$

$T_2 = \left(\sum_{j=0}^{512} a \right) + \text{Maj}(a, b, c)$

$h = g$

$g = f$

$f = e$

$e = d + T_1$

$d = c$

$c = b$

$b = a$

$a = T_1 + T_2$

4. Compute the intermediate hash value

$H_{i,0} = a + H_{i-1,0}$ $H_{i,4} = e + H_{i-1,4}$

$H_{i,1} = b + H_{i-1,1}$ $H_{i,5} = f + H_{i-1,5}$

$H_{i,2} = c + H_{i-1,2}$ $H_{i,6} = g + H_{i-1,6}$

$H_{i,3} = d + H_{i-1,3}$ $H_{i,7} = h + H_{i-1,7}$

return $(H_{N,0} \parallel H_{N,1} \parallel H_{N,2} \parallel H_{N,3} \parallel H_{N,4} \parallel H_{N,5} \parallel H_{N,6} \parallel H_{N,7})$

Figure 11.13 SHA-512 Logic

(Figure can be found on page 339 in textbook)

SHA-3

SHA-1 has not yet been "broken"

- No one has demonstrated a technique for producing collisions in a practical amount of time
- Considered to be insecure and has been phased out for SHA-2

NIST announced in 2007 a competition for the SHA-3 next generation NIST hash function

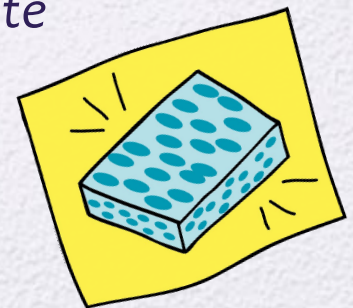
- Winning design was announced by NIST in October 2012
- SHA-3 is a cryptographic hash function that is intended to complement SHA-2 as the approved standard for a wide range of applications

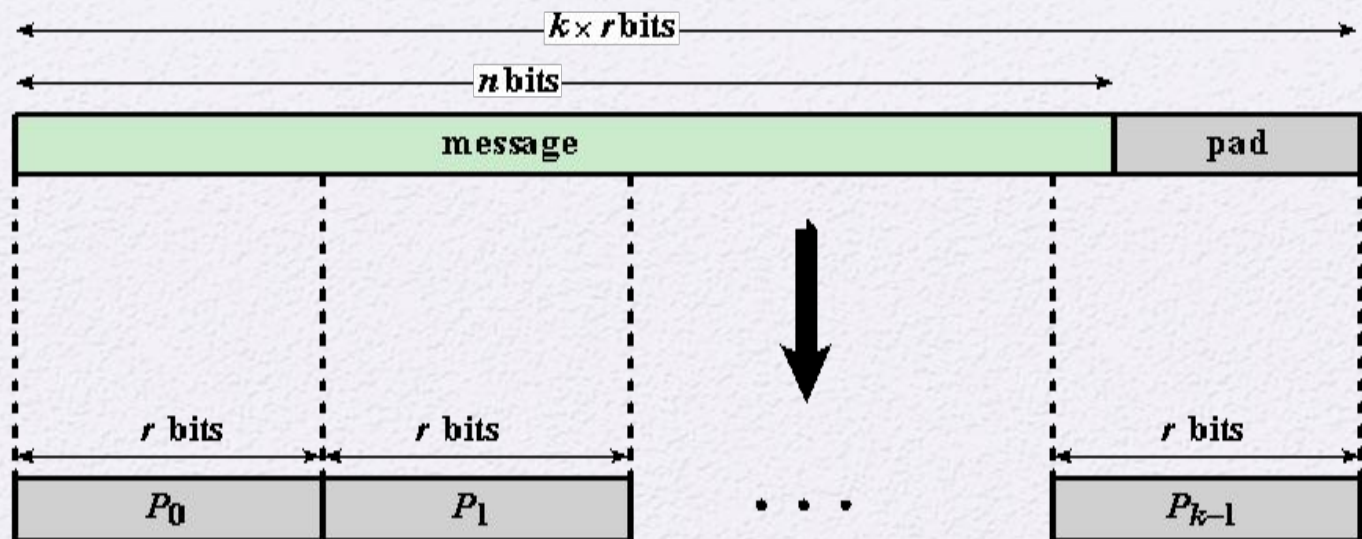
SHA-2 shares the same structure and mathematical operations as its predecessors so this is a cause for concern

- Because it will take years to find a suitable replacement for SHA-2 should it become vulnerable, NIST decided to begin the process of developing a new hash standard

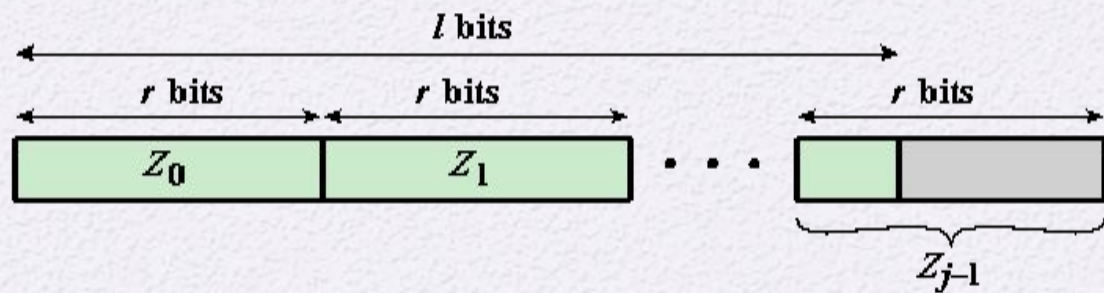
The Sponge Construction

- Underlying structure of SHA-3 is a scheme referred to by its designers as a *sponge construction*
- Takes an input message and partitions it into fixed-size blocks
- Each block is processed in turn with the output of each iteration fed into the next iteration, finally producing an output block
- The sponge function is defined by three parameters:
 - f = the internal function used to process each input block
 - r = the size in bits of the input blocks, called the *bitrate*
 - pad = the padding algorithm





(a) Input



(b) Output

Figure 11.14 Sponge Function Input and Output

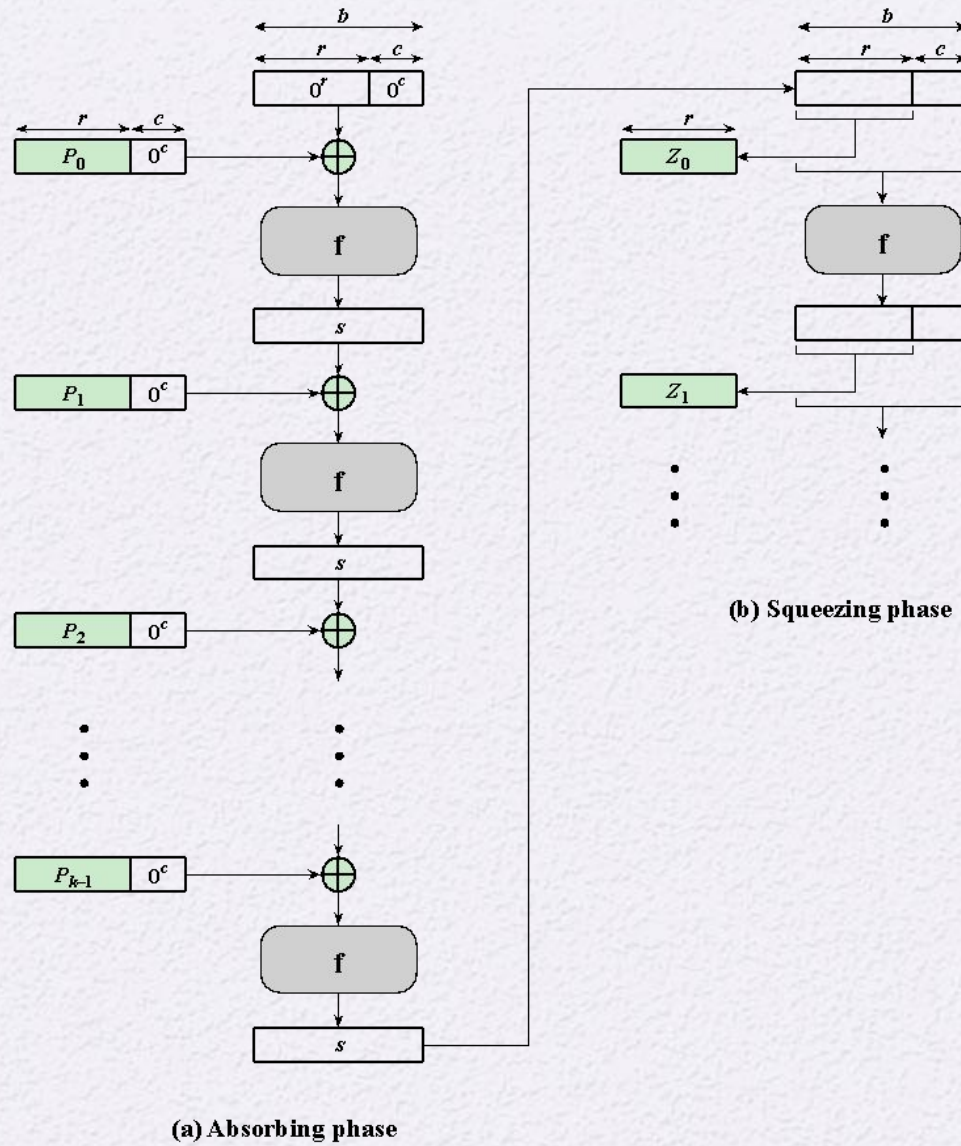


Figure 11.15 Sponge Construction

Table 11.5

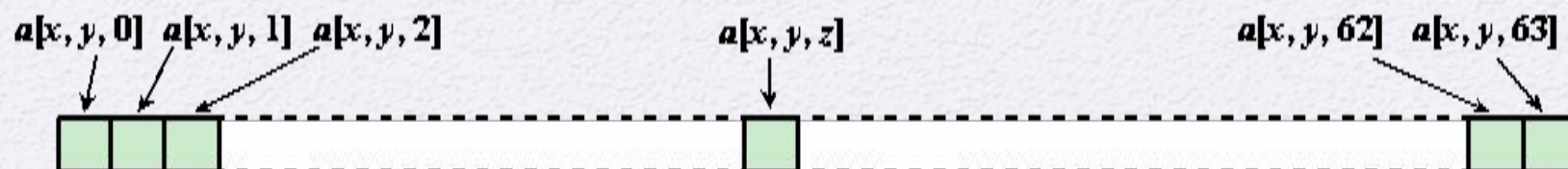
SHA-3 Parameters

Message Digest Size	224	256	384	512
Message Size	no maximum	no maximum	no maximum	no maximum
Block Size (bitrate r)	1152	1088	832	576
Word Size	64	64	64	64
Number of Rounds	24	24	24	24
Capacity c	448	512	768	1024
Collision resistance	2^{112}	2^{128}	2^{192}	2^{256}
Second preimage resistance	2^{224}	2^{256}	2^{384}	2^{512}

Note: All sizes and security levels are measured in bits.

	$x = 0$	$x = 1$	$x = 2$	$x = 3$	$x = 4$
$y = 4$	$L[0, 4]$	$L[1, 4]$	$L[2, 4]$	$L[3, 4]$	$L[4, 4]$
$y = 3$	$L[0, 3]$	$L[1, 3]$	$L[2, 3]$	$L[3, 3]$	$L[4, 3]$
$y = 2$	$L[0, 2]$	$L[1, 2]$	$L[2, 2]$	$L[3, 2]$	$L[4, 2]$
$y = 1$	$L[0, 1]$	$L[1, 1]$	$L[2, 1]$	$L[3, 1]$	$L[4, 1]$
$y = 0$	$L[0, 0]$	$L[1, 0]$	$L[2, 0]$	$L[3, 0]$	$L[4, 0]$

(a) State variable as 5 x 5 matrix A of 64-bit words



(b) Bit labeling of 64-bit words

Figure 11.16 SHA-3 State Matrix

SHA-3 Iteration Function f

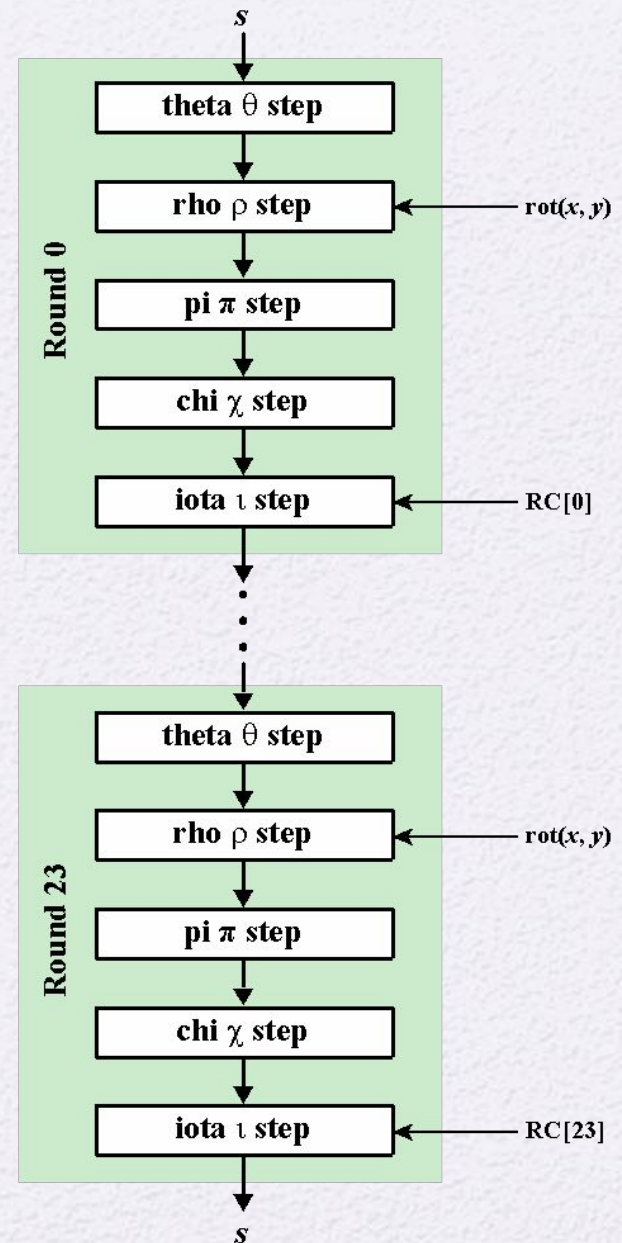
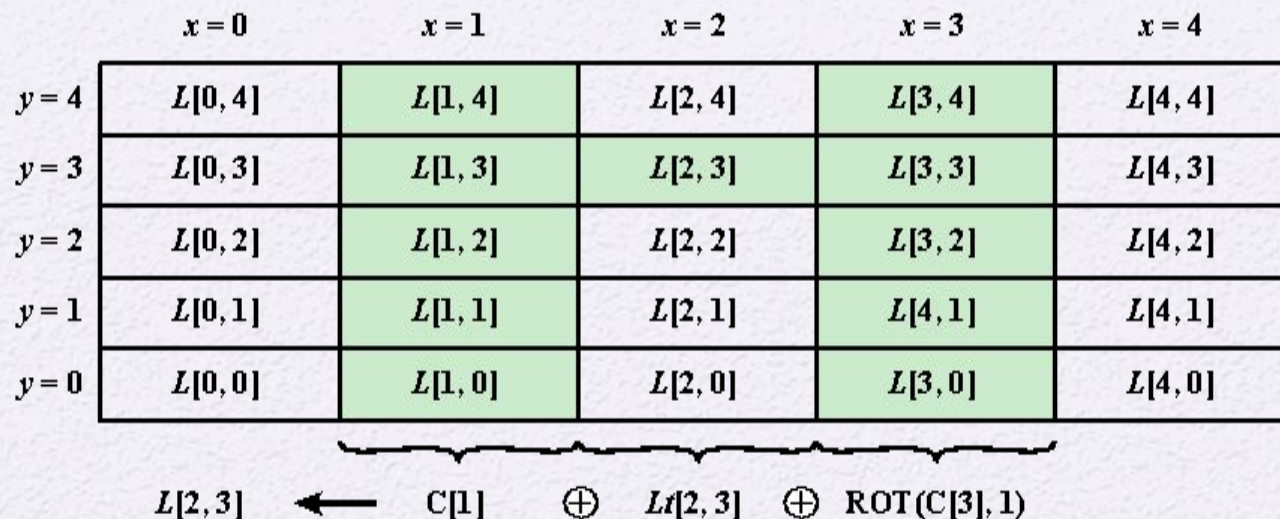


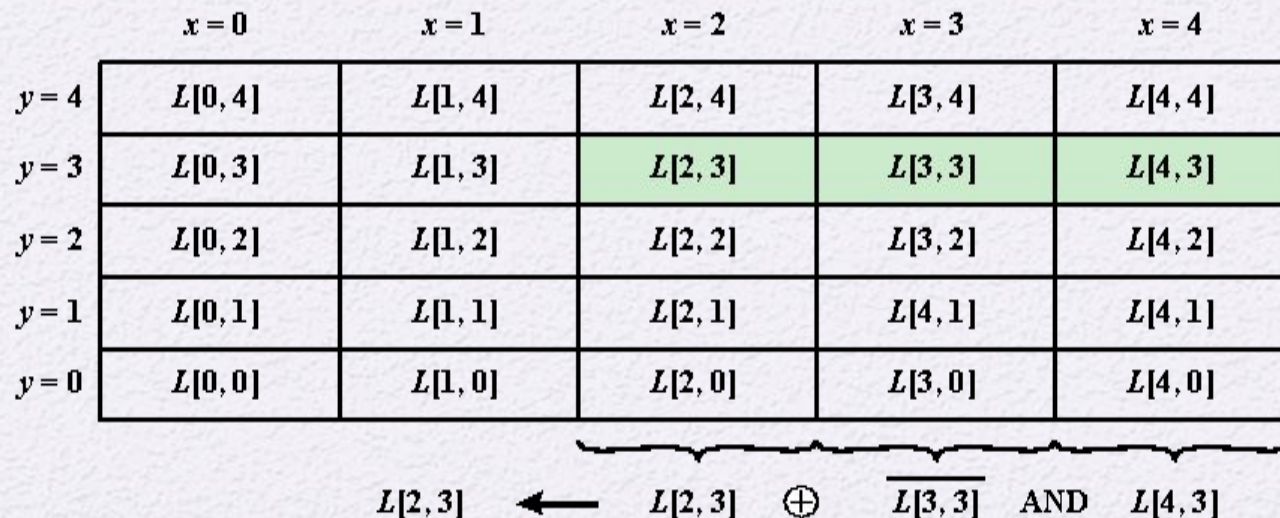
Figure 11.17 SHA-3 Iteration Function f

Table 11.6 -- Step Functions in SHA-3

Function	Type	Description
θ	Substitution	New value of each bit in each word depends its current value and on one bit in each word of preceding column and one bit of each word in succeeding column.
	Permutation	The bits of each word are permuted using a circular bit shift. $W[0, 0]$ is not affected.
	Permutation	Words are permuted in the 5×5 matrix. $W[0, 0]$ is not affected.
	Substitution	New value of each bit in each word depends on its current value and on one bit in next word in the same row and one bit in the second next word in the same row.
	Substitution	$W[0, 0]$ is updated by XOR with a round constant.



(a) θ step function



(b) χ step function

Table 11.7 Rotation Values Used in SHA-3

(a) Calculation of values and positions

T	$g(t)$	$g(t) \bmod 64$	x, y
0	1	1	1, 0
1	3	3	0, 2
2	6	6	2, 1
3	10	10	1, 2
4	15	15	2, 3
5	21	21	3, 3
6	28	28	3, 0
7	36	36	0, 1
8	45	45	1, 3
9	55	55	3, 1
10	66	2	1, 4
11	78	14	4, 4

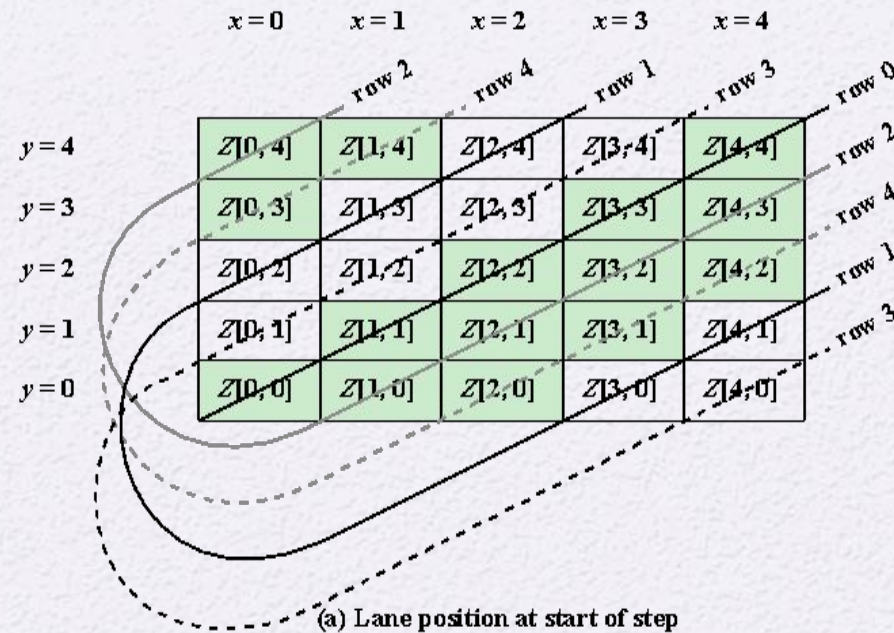
t	$g(t)$	$g(t) \bmod 64$	x, y
12	91	27	4, 0
13	105	41	0, 3
14	120	56	3, 4
15	136	8	4, 3
16	153	25	3, 2
17	171	43	2, 2
18	190	62	2, 0
19	210	18	0, 4
20	231	39	4, 2
21	253	61	2, 4
22	276	20	4, 1
23	300	44	1, 1

Note: $g(t) = (t + 1)(t + 2)/2$

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}^t \begin{pmatrix} 1 \\ 0 \end{pmatrix} \bmod 5$$

(b) Rotation values by word position in matrix

	$x = 0$	$x = 1$	$x = 2$	$x = 3$	$x = 4$
$y = 4$	18	2	61	56	14
$y = 3$	41	45	15	21	8
$y = 2$	3	10	43	25	39
$y = 1$	36	44	6	55	20
$y = 0$	0	1	62	28	27



	$x=0$	$x=1$	$x=2$	$x=3$	$x=4$
$y=4$	$Z[2, 0]$	$Z[3, 1]$	$Z[4, 2]$	$Z[0, 3]$	$Z[1, 4]$
$y=3$	$Z[4, 0]$	$Z[0, 1]$	$Z[1, 2]$	$Z[2, 3]$	$Z[3, 4]$
$y=2$	$Z[1, 0]$	$Z[2, 1]$	$Z[3, 2]$	$Z[4, 3]$	$Z[0, 4]$
$y=1$	$Z[3, 0]$	$Z[4, 1]$	$Z[0, 2]$	$Z[1, 3]$	$Z[2, 4]$
$y=0$	$Z[0, 0]$	$Z[1, 1]$	$Z[2, 2]$	$Z[3, 3]$	$Z[4, 4]$

(b) Lane position after permutation

Figure 11.19 Pi Step Function

Table 11.8

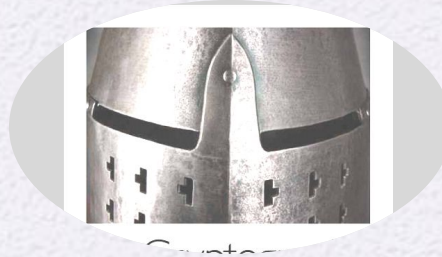
Round Constants in SHA-3

Round	Constant (hexadecimal)	Number of 1 bits
0	000000000000000001	1
1	00000000000000008082	3
2	8000000000000000808A	5
3	80000000080008000	3
4	00000000000000808B	5
5	00000000800000001	2
6	80000000080008081	5
7	80000000000008009	4
8	00000000000000008A	3
9	000000000000000088	2
10	0000000080008009	4
11	0000000080000000A	3

Round	Constant (hexadecimal)	Number of 1 bits
12	000000008000808B	6
13	80000000000000008B	5
14	800000000000008089	5
15	80000000000008003	4
16	80000000000008002	3
17	800000000000000080	2
18	0000000000000800A	3
19	8000000080000000A	4
20	8000000080008081	5
21	80000000000008080	3
22	00000000800000001	2
23	8000000080008008	4

Summary

- Summarize the applications of cryptographic hash functions
- Explain why a hash function used for message authentication needs to be secured
- Understand the operation of SHA-512



- Understand the differences among preimage resistant, second preimage resistant, and collision resistant properties
- Present an overview of the basic structure of cryptographic hash functions
- Describe how cipherblock chaining can be used to construct a hash function