CS558 Introduction to Computer Security

Chapter 11
Message Authentication and Hash Functions

Message Authentication and Digital Signature

- Message authentication
  - Verify that received messages come from the alleged source.
  - Verify the integrity of a message – data received are exactly as sent by.

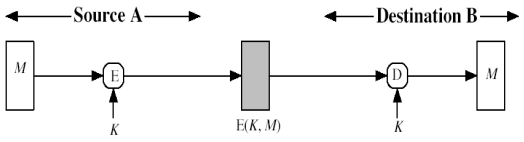
Attacks

- The following attacks can be identified
  - Message authentication
    - Masquerade: insertion of messages into the network from a fraudulent source, e.g. fraudulent acknowledgments of message receipt/nonreceipt by someone other than the message recipient
    - Content modification: changes to the content of a message, e.g. insertion, deletion
    - Sequence modification: modify a sequence of messages
    - Timing modification: delay or replay of messages

Authentication Functions

- Authentication functions: produce an authenticator, a value that can be used to authenticate the message.
  - Message encryption: the ciphertext of the entire message serves as its authenticator
  - Message authentication code (MAC): a function of the message and a secret key that produces a fixed-length value
  - Hash function: maps a message of any length into a fixed-length hash value

Message Encryption: Symmetric Encryption



(a) Symmetric encryption: confidentiality and authentication

- B knows A must have created it
  - Since only sender and receiver know key used
- Knows content cannot be altered

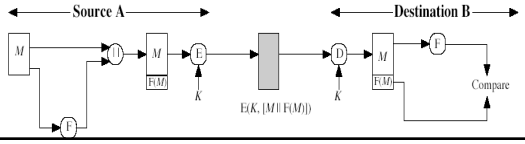
Message Encryption: Symmetric Encryption

- Problem: given a decryption function D and a secret key K, the receiver will accept any input X and produce output Y=D(K,X).
  - If X is the ciphertext of a legitimate message M, then Y is M, otherwise, meaningless sequence of bits.

7

Symmetric Encryption: Internal Control

- Solution: force the plaintext to have some structure
  - E.g. append an error-detecting code as a frame check sequence (FCS) to each message before encryption.
    - Sender: prepares a plaintext megs M, provides M as input to a function F that produces an FCS, appends FCS to M and encrypts entire block.
    - Receiver: decrypts the block, treats the result as a message with an appended FCS, and applies F to reproduce FCS. If the calculated FCS = incoming FCS, then the message is considered authentic.



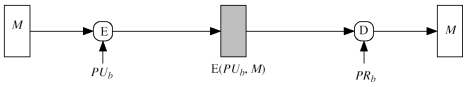
Summary: Message Encryption (Symmetric)

A → B: E(*K*, *M*)
- Provides confidentiality
  - Only A and B share *K*
- Provides a degree of authentication
  - Could come only from A
  - Has not been altered in transit
  - Requires some formatting/redundancy
- Does not provide signature
  - Receiver could forge message
  - Sender could deny message

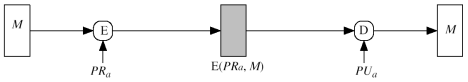
(a) Symmetric encryption

9

Public-key Encryption



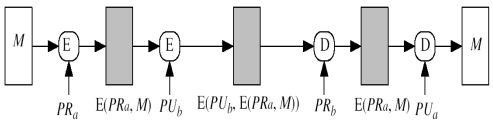
(b) Public-key encryption: confidentiality



(c) Public-key encryption: authentication and signature

10

Public-key Encryption



(d) Public-key encryption: confidentiality, authentication, and signature

- Sender signs message using their private-key
- Then encrypts with recipients public key
- Have both secrecy and authentication
- Again, plaintext needs to have some internal structure so that the receiver can distinguish between well-formed plaintext and random bits.

11

Summary: Message Encryption (Public-key)

A → B: E(PU_b, *M*)
- Provides confidentiality
  - Only B has PR_b to decrypt
- Provides no authentication
  - Any party could use PU_b to encrypt message and claim to be A

(b) Public-key (asymmetric) encryption: confidentiality

A → B: E(PR_a, *M*)
- Provides authentication and signature
  - Only A has PR_a to encrypt
  - Has not been altered in transit
  - Requires some formatting/redundancy
  - Any party can use PU_a to verify signature

(c) Public-key encryption: authentication and signature

A → B: E(PU_b, E(PR_a, *M*))
- Provides confidentiality because of PU_b
- Provides authentication and signature because of PR_a

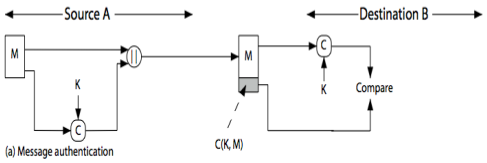
(d) Public-key encryption: confidentiality, authentication, and signature

Message Authentication Codes

- A MAC function is similar to encryption
  - Difference: MAC alg. needs not be reversible, as it must for decryption. In general, MAC function is a many-to-one function - many different messages may generate the same MAC value.

13

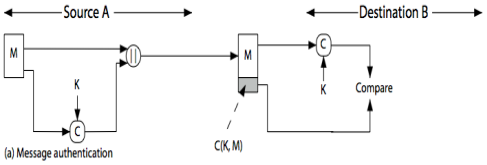
Message Authentication Code (MAC)



- Use a secret key to generate a small fixed-size block of data MAC = C(K,M)
  - M: input mesg. C: MAC function. K: shared secret key
- The message + MAC are transmitted.
- Receiver performs same computation on message and checks it matches the MAC

14

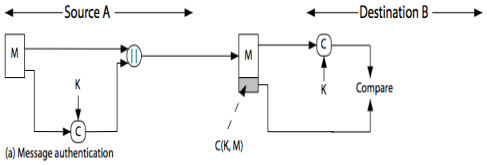
Message Authentication Code (MAC)



- Does this assure that message is unaltered and comes from sender?

15

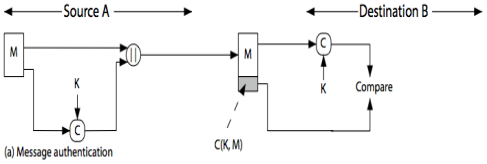
Message Authentication Code (MAC)



- Does this assure that message is unaltered and comes from sender?
  - Yes

16

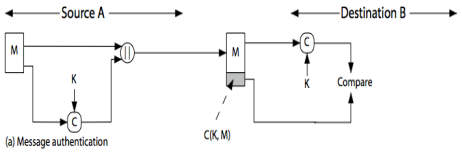
Message Authentication Code (MAC)



- Does this assure that message is unaltered and comes from sender?
  - Yes
  - If an attacker alters the message, then the receiver's calculation of the MAC will differ from the received MAC.
  - No one else knows the secret key, no one else could prepare a message with a proper MAC.

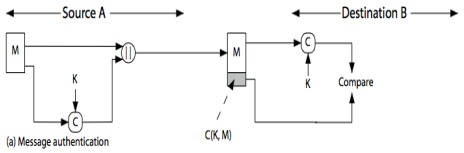
17

Message Authentication Codes



18

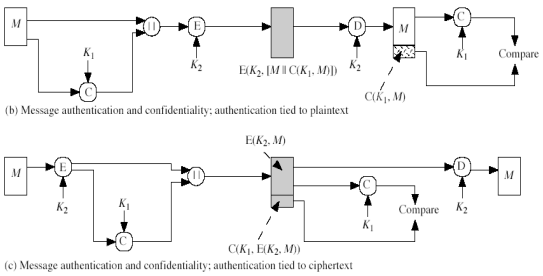
Message Authentication Codes



- Provides authentication but not confidentiality
  - because the message is transmitted in the clear.

19

Message Authentication Codes



- Provides both authentication and confidentiality
  - Performing encryption after or before the MAC algorithm.

20

Message Authentication Codes

- Symmetric encryption will provide authentication. Why use a MAC?

21

Message Authentication Codes

- Symmetric encryption will provide authentication. Why use a MAC?
  - Sometimes only authentication is needed.
  - One side has a heavy load and cannot afford the time to decrypt all incoming messages – messages are chosen at random for checking
- MAC does not provide a digital signature because both sender and receiver share the same key.

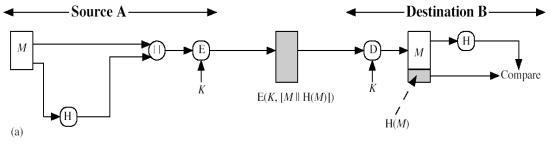
22

Hash Functions

- Is a variation on the message authentication code
- A hash function accepts a variable-size message M as input and produces a fixed-size output, referred to as a hash code H(M)
  - Hash code: message digest or hash value
- Unlike a MAC, a hash code does not use a key, but is a function only of the input message.
- Hash code is used to detect changes to message – a change to any bit(s) in the message results in a change to the hash code
- Can use in various ways with message
- Most often to create a digital signature

23

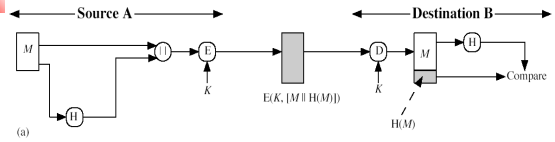
Using Hash Code (1)



- The message plus concatenated hash code is encrypted using symmetric encryption.
- The message must come from A because only A and B share the secret key.
- The hash code provides the structure required to achieve authentication.

24

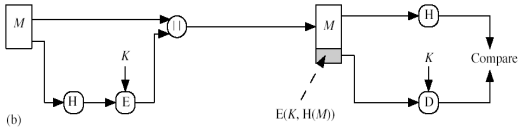
Using Hash Code (1)



- The message plus concatenated hash code is encrypted using symmetric encryption.
- The message must come from A because only A and B share the secret key.
- The hash code provides the structure required to achieve authentication.
- Confidential is also provided because encryption is applied to the entire message.

25

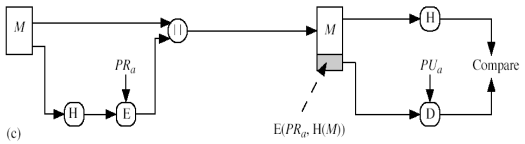
Using Hash Code (2)



- Only the hash code is encrypted using symmetric encryption
  - For applications that do not require confidentiality.

26

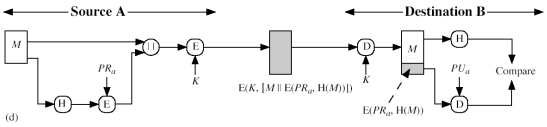
Using Hash Code (3)



- Only the hash code is encrypted using public-key encryption and using sender's private key

27

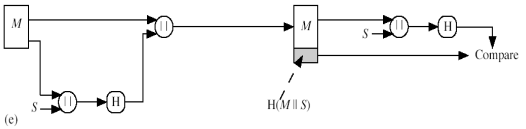
Using Hash Code (4)



- If confidentiality as well as a digital signature is desired, then the message plus the private-key-encrypted hash code can be encrypted using a symmetric secret key.

28

Using Hash Code (5)



- It is possible to use a hash function but no encryption for message authentication
- 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
- B has S, so can recompute the hash value to verify.
- Because S is not sent, an attacker cannot modify a message or generate a false message.

29

Requirements for Hash Functions

- The purpose of a hash function is to produce a fingerprint of a file, message, etc
- A hash function must have the following properties:
  - Can be applied to any sized message M
  - Produces fixed-length output hc
  - Is easy to compute hc=H(M) for any message M
  - Given hc is infeasible to find x s.t. H(x)=hc
    - Example in the previous slide: if not hold, an attacker can easily discover the secret value s
  - Given x is infeasible to find y s.t. H(y)=H(x)
  - It's infeasible to find any x,y s.t. H(y)=H(x)

30

Simple Hash Functions

- The input is viewed as a sequence of n-bit blocks.
- A simple hash function: XOR of message blocks

$$Ci = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{im}$$

  - Ci: ith bit of the hash code, $1 \le i \le n$
  - m: number of n-bit blocks in the input
  - B_{ij}: ith bit in jth block

31

Simple Hash Functions

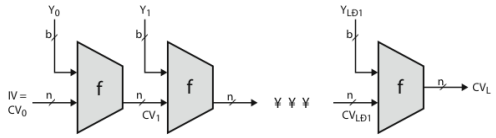
- Not secure
  - It is easy to produce a new message that yields the hash code.
  - Simply prepare the desired alternate message and then append an n-bit block that forces the new message plus block to yield the desired hash code
- Need a stronger cryptographic function (next chapter)

32

Chapter 12
Hash Algorithms

Hash Algorithm Structure

- The hash function takes an input message and partitions it into L fixed-sized blocks of b bits each.
- The final block also includes the value of the total length of the input to the hash function – makes the job of the attacker more difficult

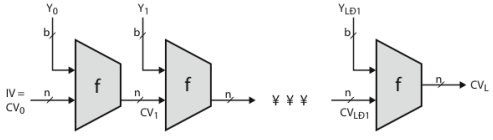


IV = Initial value
CV_i = chaining variable
Y_i = ith input block
f = compression algorithm

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

Hash Algorithm Structure

- The algorithm repeatedly uses a compression function f that takes two inputs
  - An n-bit output from previous step – chaining variable
  - A b-bit block
- And produces an n-bit output. b > n ➔ compression

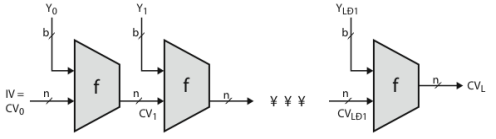


IV = Initial value
CV_i = chaining variable
Y_i = ith input block
f = compression algorithm

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

Hash Algorithm Structure

- $CV_0 = IV$ = initial n-bit value
- $CV_i = f(CV_{i-1}, Y_{i-1}), 1 <= i <= L$
- $H(M) = CV_L$, M consists of $Y_0, Y_1, \ldots, Y_{L-1}$



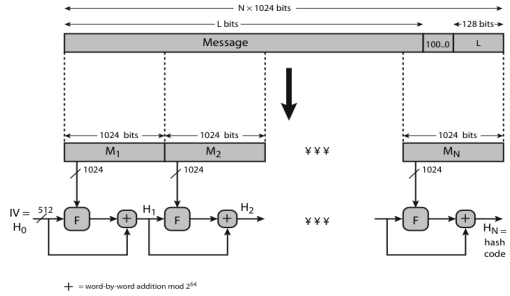
IV = Initial value
CV_i = chaining variable
Y_i = ith input block
f = compression algorithm

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

Secure Hash Algorithm (SHA)

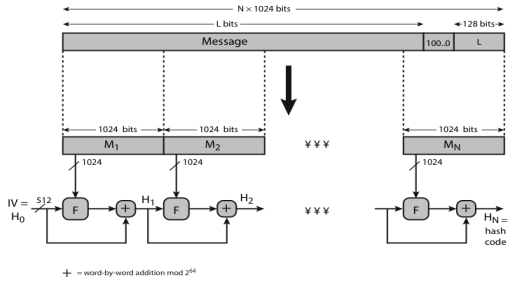
- Originally designed by National Institute of Standards and Technology (NIST) in 1993
- SHA-1 (160 bits), SHA-256 (256 bits), SHA-512 (512 bits) ……
- We will focus on SHA-512: takes as input a message with a maximum length of less than 2^{128} bits and produces as output a 512-bit message digest.

SHA-512 Overview



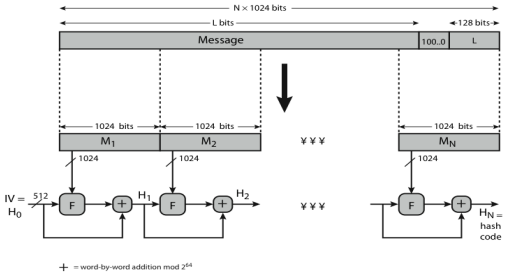
- Follows the general structure
- Processing message in 1024-bit blocks

SHA-512 Overview



- Step 1: Append padding bits
  - The message is padded so that length mod 1024 = 896. The padding consists of a single 1-bit followed by the necessary number of 0-bits.

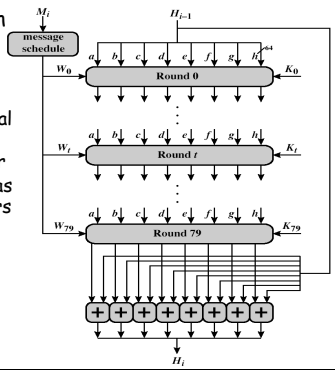
SHA-512 Overview



- Step 2: Append length
  - A block of 128 bits is appended to the message, that contains the length of the original message (does not include the padding)
  - Length of the expanded message: N*1024.

SHA-512 Processing of a Single 1024-Bit Block

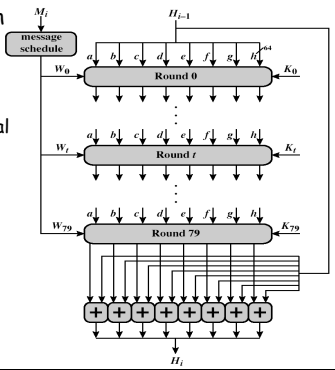
- Step 3: Initialize hash buffer
  - A 512-bit buffer is used to hold intermediate and final results of the hash function. The buffer can be represented as eight 64-bit registers (a,b,c,d,e,f,g,h)



SHA-512 Processing of a Single 1024-Bit Block

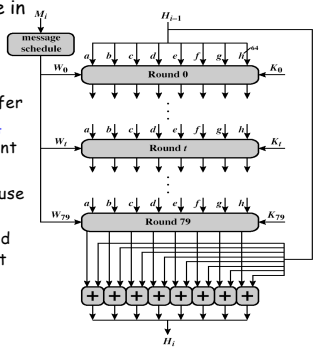
- Step 3: Initialize hash buffer
  - The 8 registers are initialized to the following 64-bit integers (hexadecimal values):

a = 6A09E667F3BCC908
b = BB67AE8584CAA73B
c = 3C6EF372FE94F82B
d = A54FF53A5F1D36F1
e = 510E527FADE682D1
f = 9B05688C2B3E6C1F
g = 1F83D9ABFB41BD6B
h = 5BE0CD19137E2179



SHA-512 Processing of a Single 1024-Bit Block

- Step 4: Process message in 1024-bit (128-word) blocks
  - Consists of 80 rounds
  - Updating a 512-bit buffer
  - Using a 64-bit value W_t derived from the current message block M_i
  - Each round also makes use of a constant K_t
  - The output of the Round 79 is added to the input to the first round to produce H_i.



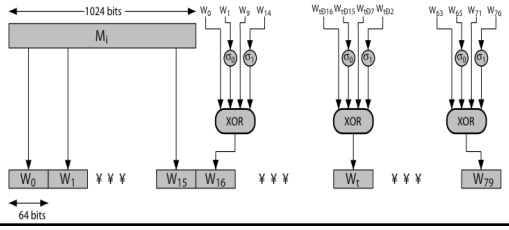
Creation of 80-Word Input Sequence

$\sigma_0(x) = ROTR^1(x) \oplus ROTR^8(x) \oplus SHR^7(x)$

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

$ROTR^n(x)$ = circular right shift of the 64-bit argument x by n bits

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



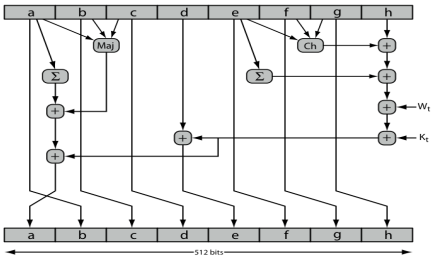
SHA-512 Round Function

$T1 = h + Ch(e,f,g) + \Sigma e + Wi + Ki$, $T2 = \Sigma a + Maj(a,b,c)$

$a = T1+T2$, $b = a$, $c = b$, $d = c$, $e = d + T1$, $f = e$, $g = f$, $h = g$

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

$ch(e,f,g)$ – if e then f else g



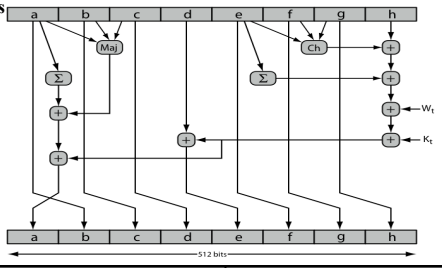
SHA-512 Round Function

$T1 = h + Ch(e,f,g) + \Sigma e + Wi + Ki$, $T2 = \Sigma a + Maj(a,b,c)$

$a = T1+T2$, $b = a$, $c = b$, $d = c$, $e = d + T1$, $f = e$, $g = f$, $h = g$

$\Sigma e = ROTR^{14}(e) \oplus ROTR^{18}(e) \oplus ROTR^{41}(e)$

$ROTR^n(x)$ = circular right shift of the 64-bit argument x by n bits



Midterm Exam

- Date: March 15th (Thursday)
- Time: 4:25pm - 5:50pm
- Place: LH-009
- Close book, close notes

47

Final Exam

- Date: May 11th (Friday)
- Time: 10:25am - 12:25
- Place: FA-258
- Close book, close notes

48