Data Integrity Algorithms

Jiuqin ZHOU

Outline

1. Cryptographic Hash Functions
   i. Examples & Requirements
   ii. SHA-512: A variant of the SHA-2 family
2. Message Authentication Codes
   i. MAC Overview
   ii. Hash-based vs. Cipher-based MAC
3. Digital Signature
   i. Signature Scheme Overview
   ii. Elgamal Digital Signature Scheme

Cryptographic Hash Functions » Examples & Requirements

A Simple Hash Function

Security Requirements

General Principle

The input (message, file, etc.) is viewed as a sequence of n -bit blocks. The input is processed one block at a time in an iteration to produce an n-bit hash function.

A Simple Hash Function

Security Requirements

One of the simplest hash functions is the bit-by-bit exclusive-OR (XOR) of every block. This can be expressed as

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

where C_i is the i-th bit of the hash code, $1 \leq i \leq n$, m is the number of n-bit blocks in the input, b_{ij} is the i-th bit in the j-th block, and $\oplus$ is the XOR operation.

A Simple Hash Function

Security Requirements

- Consider the case when $n = 8$, then the hash function is the same as the byte-wise XOR operation over all input bytes.
- For a given hash value $00000001_{(2)}$, it is easy to find two preimage $00000000, 00000001_{(2)}$ and $00000001, 00000000_{(2)}$ that have the same bit length.
- Therefore, this hash function cannot be called a secure cryptographic hash function.

Cryptographic Hash Functions » Examples & Requirements

A Simple Hash Function

Security Requirements

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) with $x \neq y$, such that $H(x) = H(y)$.
Pseudorandomness	Output of H meets standard tests for pseudorandomness.

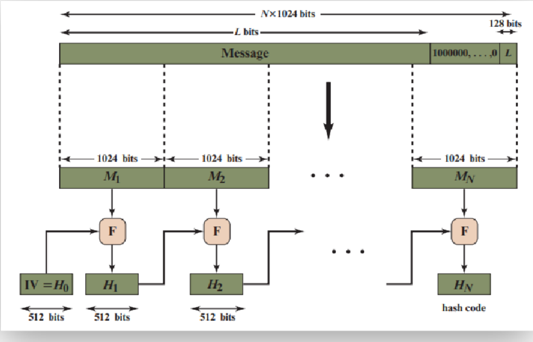
Cryptographic Hash Functions » SHA-512: A variant of the SHA-2 family

Overall Structure

Block Operation

Word Generation

Round Function



Input and Output

- The algorithm takes as input a message with a maximum length of less than 2^{128} bits and produces as output a 512-bit message digest. The input is processed in 1024-bit blocks.

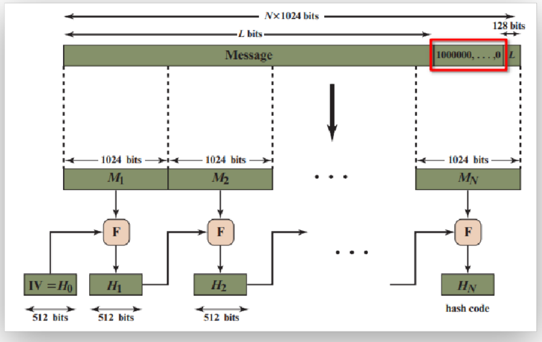
Cryptographic Hash Functions » SHA-512: A variant of the SHA-2 family

Overall Structure

Block Operation

Word Generation

Round Function



S1: Append Padding

- The message is padded so that its length is congruent to 896 modulo 1024.
- Padding is always added, even if the message is already of the desired length. Thus, the number of padding bits is in the range of 1 to 1024.
- The padding consists of a single 1 bit followed by the necessary number of 0 bits.

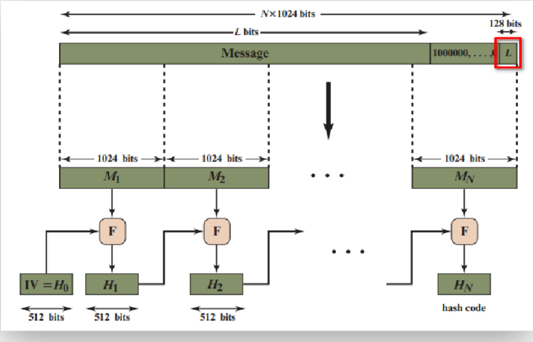
Cryptographic Hash Functions » SHA-512: A variant of the SHA-2 family

Overall Structure

Block Operation

Word Generation

Round Function



S2: Append Length

- A block of 128 bits is appended to the message.
- This block is treated as an unsigned 128-bit integer and contains the length of the original message in bits before the padding.

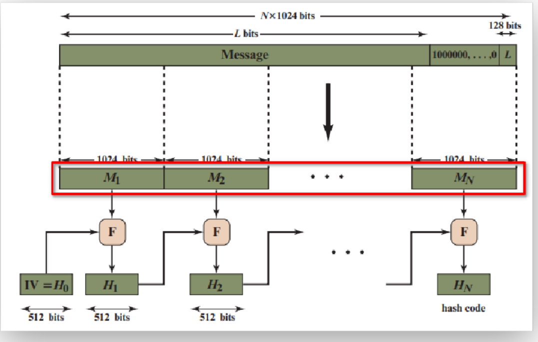
Cryptographic Hash Functions » SHA-512: A variant of the SHA-2 family

Overall Structure

Block Operation

Word Generation

Round Function



Outcome of First Two Steps

- The outcome of the first two steps yields a message that is an integer multiple of 1024 bits in length.
- In the picture, the expanded message is represented as the sequence of 1024-bit blocks $M_1, M_2, \ldots, M_N$, so that the total length of the expanded message is $N * 1024$ bits.

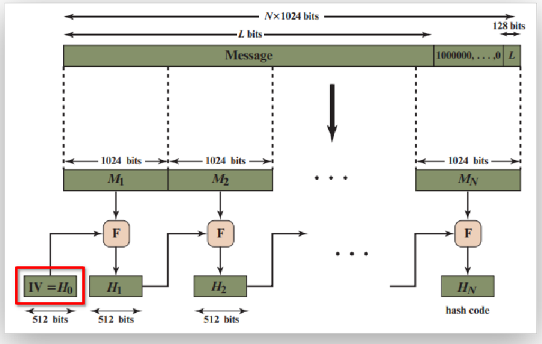
Cryptographic Hash Functions » SHA-512: A variant of the SHA-2 family

Overall Structure

Block Operation

Word Generation

Round Function



S3: 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).

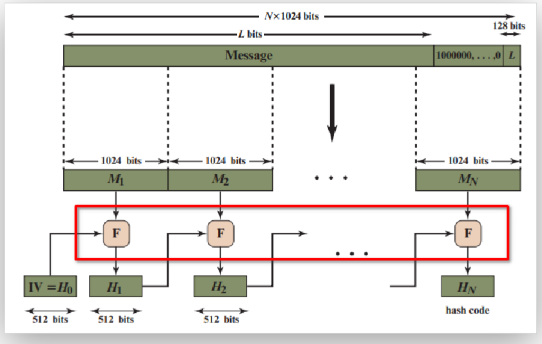
Cryptographic Hash Functions » SHA-512: A variant of the SHA-2 family

Overall Structure

Block Operation

Word Generation

Round Function

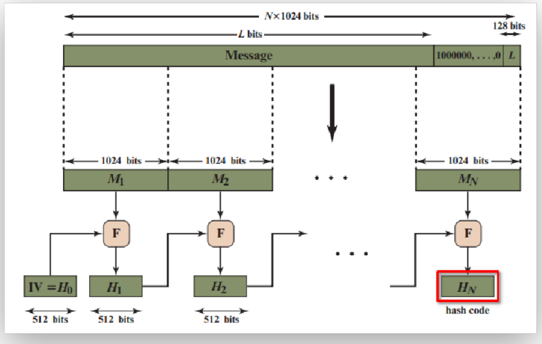


S4: Process Message in Blocks

- The heart of the algorithm is a module that consists of 80 rounds; this module is labeled F in the picture.

Cryptographic Hash Functions » SHA-512: A variant of the SHA-2 family

- **Overall Structure**
- Block Operation
- Word Generation
- Round Function



S5: Output Message Digest

- After all N 1024-bit blocks have been processed, the output from the N-th stage is the 512-bit message digest.

Cryptographic Hash Functions » SHA-512: A variant of the SHA-2 family

Overall Structure

Block Operation

Word Generation

Round Function

- We can summarize the behavior of SHA-512 as follows:
  - $H_0 = IV$.
  - $Hi = F(H_{i-1}, M_{i-1})$, for $0 \leq i \leq N$.
  - $MD = H_N$.

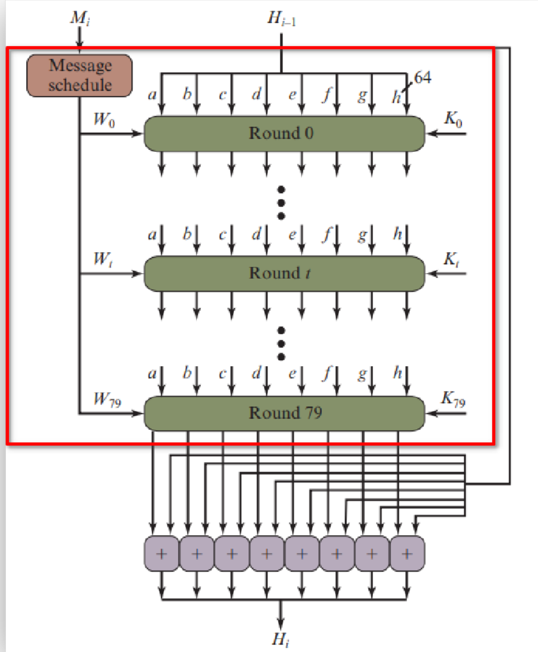
Cryptographic Hash Functions » SHA-512: A variant of the SHA-2 family

Overall Structure

Block Operation

Word Generation

Round Function



80 Rounds

- Each round takes as input the 512-bit buffer value, (a, b, c, d, e, f, g, h), and updates the contents of the buffer. At input to a round, the buffer has the value of the intermediate hash value, H_{i-1}.
- Each round t makes use of a 64-bit value W_t , derived from the current 1024-bit block being processed M_i.
- Each round also uses an additive constant K_t , where $0 \leq t \leq 79$ indicates one of the 80 rounds.

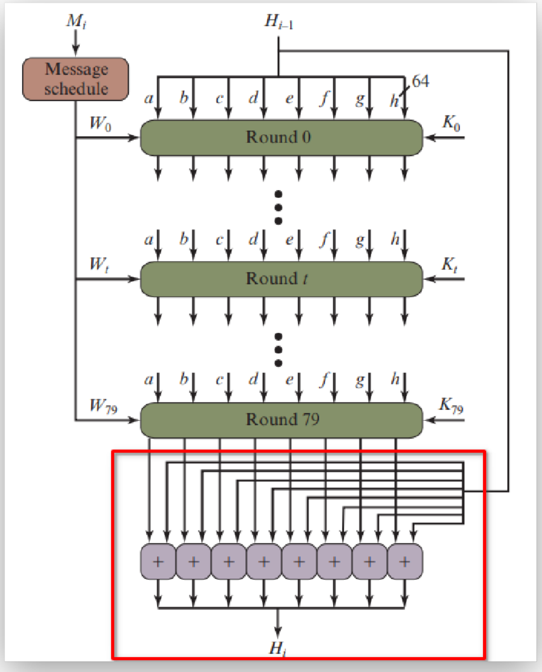
Cryptographic Hash Functions » SHA-512: A variant of the SHA-2 family

Overall Structure

Block Operation

Word Generation

Round Function



Addition in 2^{64}

- The output of the eightieth round is added to the input to the first round (H_{i-1}) to produce H_i .
- The addition is done independently for each of the eight words in the buffer with each of the corresponding words in H_{i-1}, using addition modulo 2^{64} .

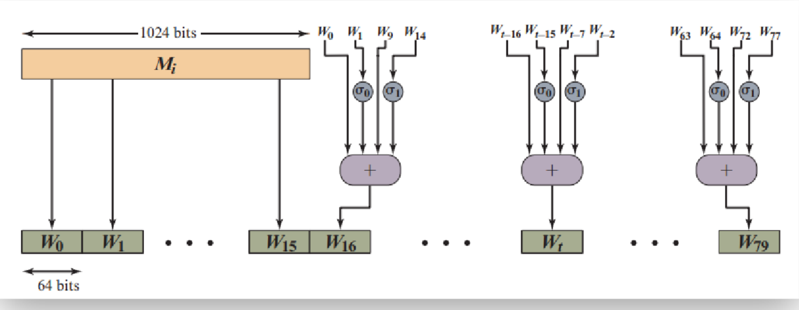
Cryptographic Hash Functions » SHA-512: A variant of the SHA-2 family

Overall Structure

Block Operation

Word Generation

Round Function

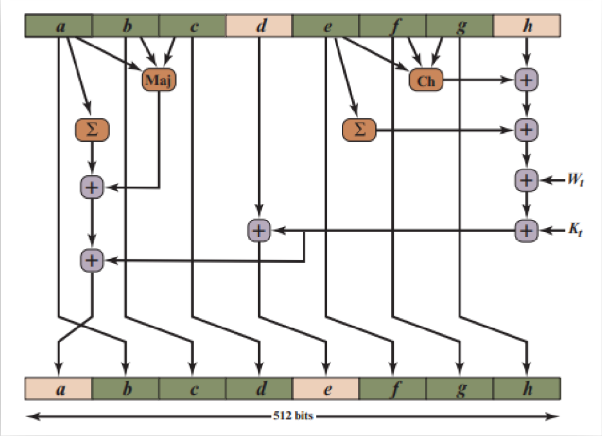


First 16 Bytes

- The first 16 values of W_t are taken directly from the 16 words of the current block.

Cryptographic Hash Functions » SHA-512: A variant of the SHA-2 family

- The remaining values are defined as:
  - $W_t = \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16}$
  - where
    - $\sigma_0^{512}(x) = ROTR^1(x) \oplus ROTR^8(x) \oplus SHR^7(x)$
    - $\sigma_1^{512}(x) = ROTR^{19}(x) \oplus ROTR^{61}(x) \oplus SHR^6(x)$
    - $ROTR^n(x) =$ circular right shift (rotation) of the 64-bit argument x by n bits
    - $SHR^n(x) =$ right shift of the 64-bit argument x by n bits with padding by zeros on the left.
    - All add operations are performed under module 2^{64}.



Logic of Round Function

- $T_1 = h + Ch(e, f, g) + (\Sigma_1^{512} e) + W_t + K_t$.
- $T_2 = (\Sigma_0^{512} a) + Maj(a, b, c)$.
- $h = g$, $g = f$, $f = e$, $e = d + T_1$, $d = c$, $c = b$, $b = a$, $a = T_1 + T_2$

Cryptographic Hash Functions » SHA-512: A variant of the SHA-2 family

- $Ch(e, f, g) = (e \text{ AND } f) \oplus (\text{NOT } e \text{ AND } g)$.
- $Maj(a, b, c) = (a \text{ AND } b) \oplus (a \text{ AND } c) \oplus (b \text{ AND } c)$.
- $(\Sigma_0^{512} a) = ROTR^{28}(a) \oplus ROTR^{34}(a) \oplus ROTR^{39}(a)$.
- $(\Sigma_1^{512}) = ROTR^{14}(e) \oplus ROTR^{18}(e) \oplus ROTR^{41}(e)$.
- $ROTR^n(x) =$ circular right shift (rotation) of the 64-bit argument x by n bits.
- $W_t =$ a 64-bit word derived from the current 1024-bit input block.
- $K_t =$ a 64-bit additive constant.
- t is the step number, for $0 \leq t \leq 79$.
- The addition is moded by 2^{64}.

Message Authentication Codes » MAC Overview

Message authentication

MAC General Structure

- Message authentication is a procedure to verify that **received messages have not been altered**.
- There are three classes of functions that may serve as authenticators: **Hash Function**, **Message Encryption**, and **Message Authentication Code (MAC)**.

Message Authentication Codes » MAC Overview

Message authentication

MAC General Structure

- **Hash function** is a function that maps a message of any length into a fixed-length hash value, which serves as the authenticator, because of its property of second preimage resistant.
- **Message Encryption** uses the ciphertext of the entire message serves as its authenticator, which takes extra space equal to the plaintext.
- **Message authentication code (MAC)** is A function of the message and a secret key that produces a fixed-length value that serves as the authenticator.

Message Authentication Codes » MAC Overview

Message authentication

MAC General Structure

Message authenticators	Space Consumption	Extra Secrets
Hash Function	**Low**	No
Encryption	High	**Yes**
MAC	**Low**	**Yes**

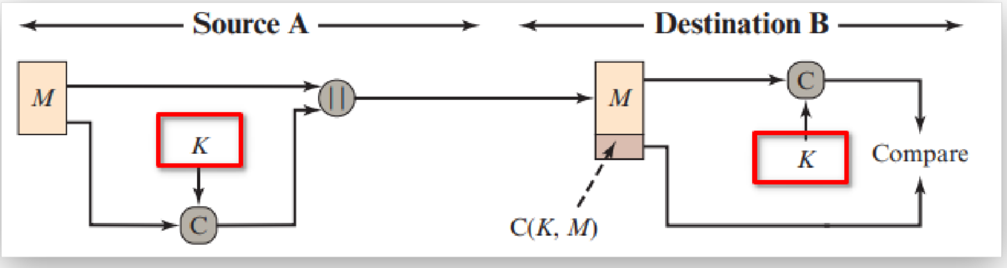
Comparison between Message Authenticators

- From the Table, we can see that Message Authentication Code are most competitive in the field of message authentication.

Message Authentication Codes » MAC Overview

Message authentication

MAC General Structure



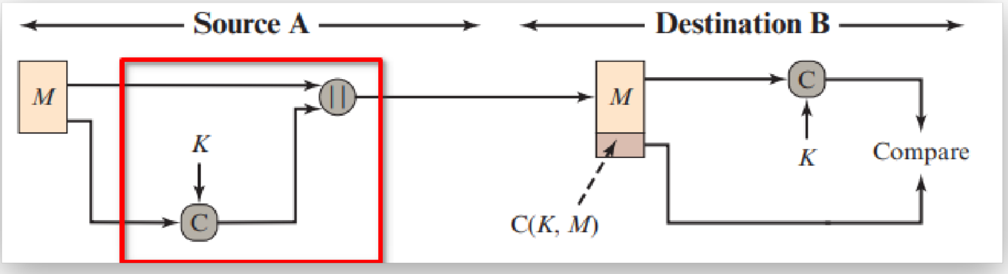
MAC on Symmetric Key

- Two communicating parties, A and B, share a common secret key K.

Message Authentication Codes » MAC Overview

Message authentication

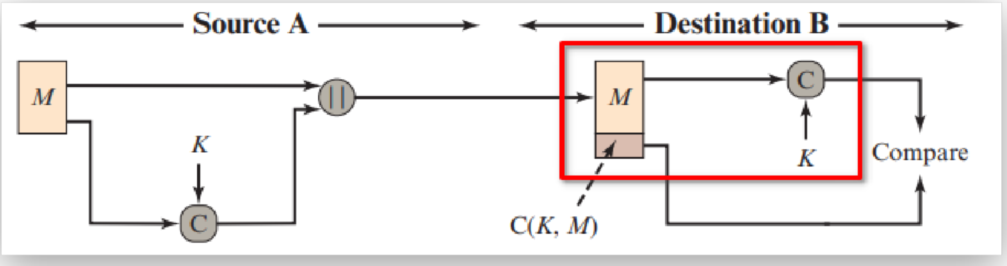
MAC General Structure



S1: Calculated Mac by A

- When A has a message to send to B, it calculates the MAC as a function of the message and the key: $\mathrm{MAC} = C(K, M)$ where M is the input message, C is the MAC function, K is the shared secret key, and MAC is the message authentication code.

Message Authentication Codes » MAC Overview



S2: Calculated MAC by B

Message authentication

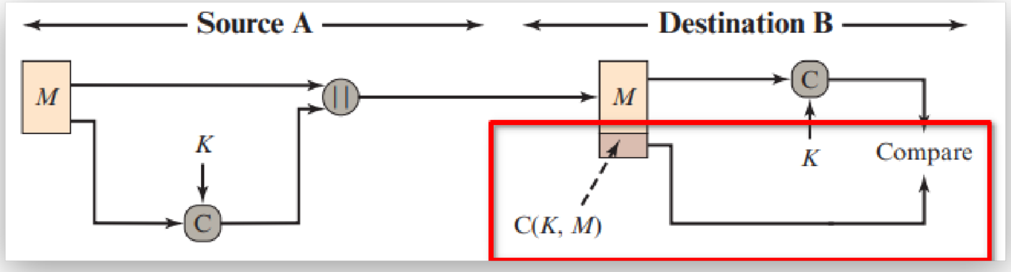
MAC General Structure

- The message plus MAC are transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the same secret key, to generate a new MAC.

Message Authentication Codes » MAC Overview

Message authentication

MAC General Structure



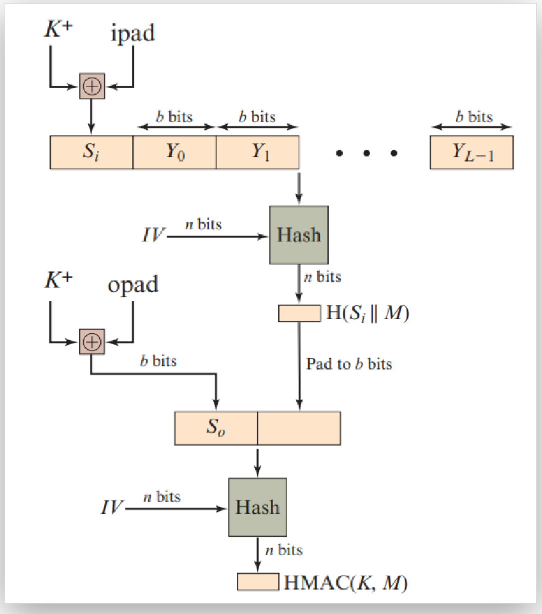
S3: MAC Comparison

- The received MAC is compared to the calculated MAC. If the received MAC matches the calculated MAC The receiver is assured that the message has not been altered.

Message Authentication Codes » Hash-based vs. Cipher-based MAC

HMAC

CMAC



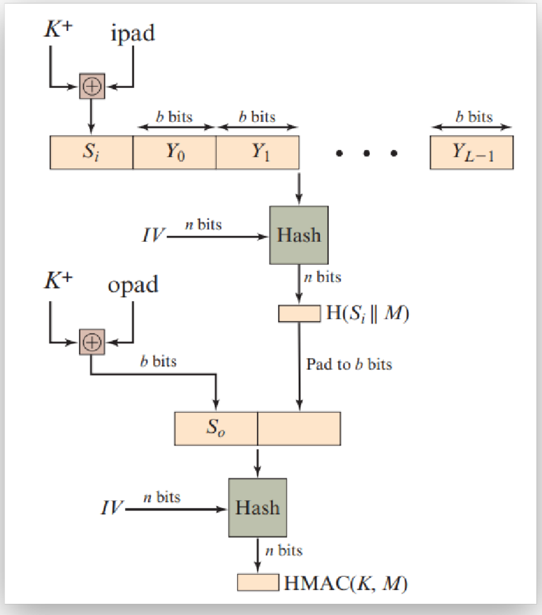
HMAC Structure

- H is embedded hash function, for example, MD5, or SHA-1. IV is initial value input to hash function.
- M is message input to HMAC (including the padding specified in the embedded hash function).
- Y_i is i-th block of M, $0 \leq i \leq (L-1)$.
- L is number of blocks in M.
- b is number of bits in a block. n is length of hash code produced by embedded hash function.

Message Authentication Codes » Hash-based vs. Cipher-based MAC

HMAC

CMAC



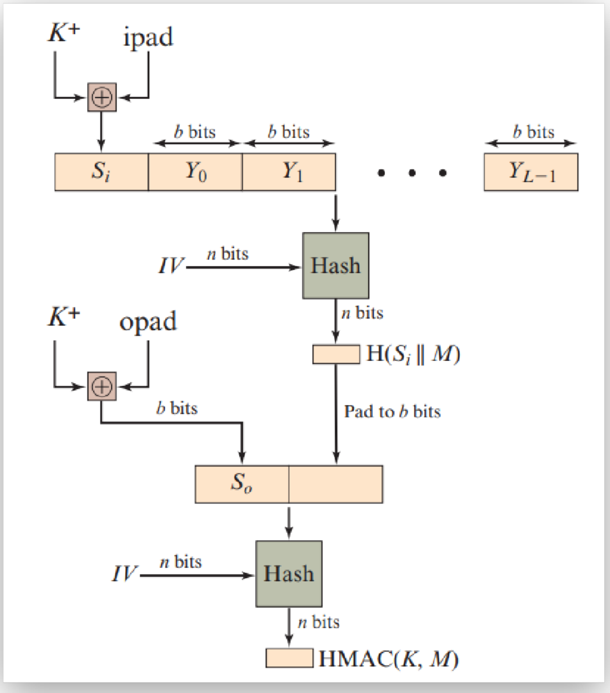
HMAC Structure

- K is secret key whose recommended length is $\geq n$. if key length is greater than b, the key is input to the hash function to produce an n-bit key.
- K^+ is K padded with zeros on the right so that the result is b bits in length.
- ipad is $00110110_{(2)}$ (36 in hexadecimal) repeated $b/8$ times.
- opad is $01011100_{(2)}$ (5C in hexadecimal) repeated $b/8$ times.

Message Authentication Codes » Hash-based vs. Cipher-based MAC

HMAC

CMAC



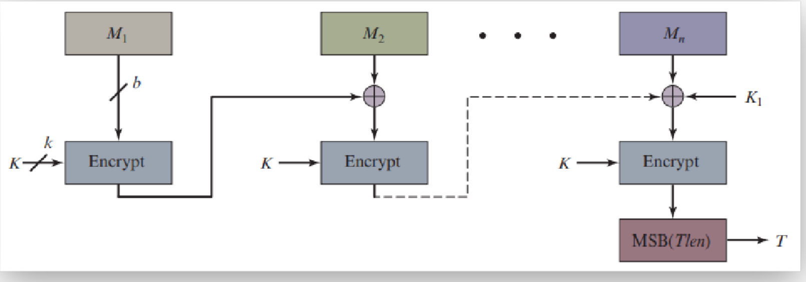
Algorithm Representation

$$HMAC(K, M) = H[(K^+ \oplus \text{opad})||H[(K^+ \oplus \text{ipad})||M]]$$

Message Authentication Codes » Hash-based vs. Cipher-based MAC

HMAC

CMAC



CMAC with K_1

- When the message is an integer multiple n of the cipher block length b, The algorithm makes use of a k-bit encryption key K and a b-bit constant, K_1.

Message Authentication Codes » Hash-based vs. Cipher-based MAC

HMAC

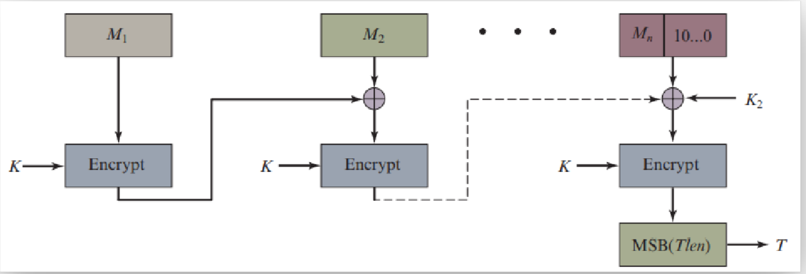
CMAC

- We have the following:
  - $C_1 = E(K, M_1)$,
  - $C_i = E(K, C_{i-1} \oplus M_i)$ for $2 \leq n - 1$
  - $C_n = E(K, C_{n-1} \oplus M_{n-1} \oplus K_1$
  - $T = MSB_{T_{len}}(C_n)$
  - where:
    - T is the message authentication code
    - T_{len} is bit length of T
    - $MSB_s(X)$ is the s leftmost bits of the bit string X.

Message Authentication Codes » Hash-based vs. Cipher-based MAC

HMAC

CMAC



CMAC with K_2

- If the message is not an integer multiple of the cipher block length, then the final block is padded to the right (least significant bits) with a 1 and as many 0s as necessary so that the final block is also of length b.
- The CMAC operation then proceeds as before, except that a different b-bit key K_2 is used instead of K_1.

Message Authentication Codes » Hash-based vs. Cipher-based MAC

HMAC

CMAC

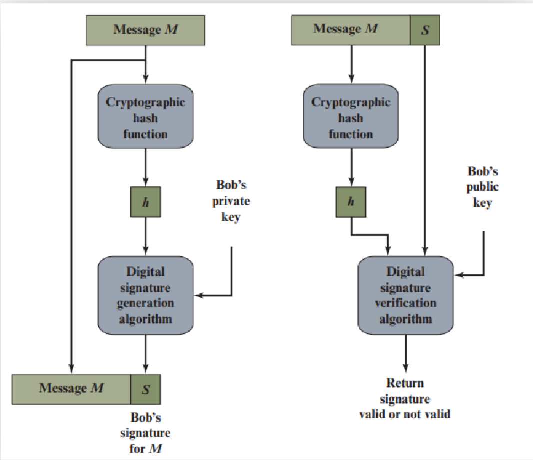
Key Generation

- The two b-bit keys are derived from the k-bit encryption key as follows. $L = E(K, \underbrace{0...0}_{b})$, $K_1 = L \cdot x$, and $K_2 = L \cdot x^2$.
- Note that multiplication is done in the finite field $GF(2^b)$.

Digital Signature » Signature Scheme Overview

Simplified Model

Security Requirements



Purpose of Digital Signature

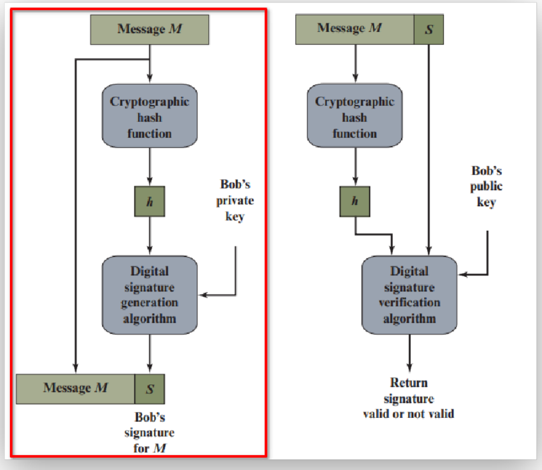
- Suppose that B wants to send a message to A. Although it is not important that the message be kept secret, he wants A to be certain that the message is indeed from him.

Digital Signature » Signature Scheme Overview

Simplified Model

Security

Requirements



S1: Calculated Signature by B

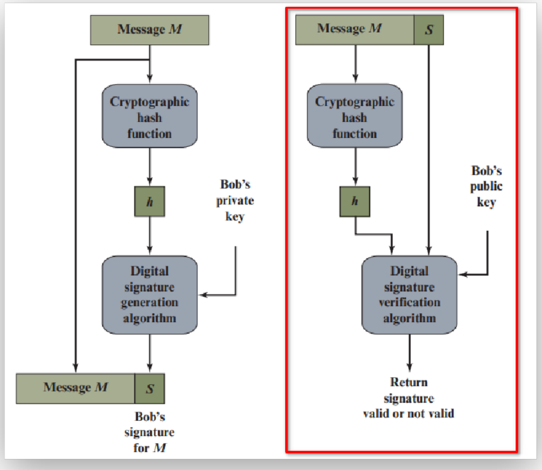
- For this purpose, B uses a secure hash function, such as SHA-512, to generate a hash value for the message. That hash value, together with B's private key serves as input to a digital signature generation algorithm, which produces a short block that functions as a digital signature.
- B sends the message with the signature attached.

Digital Signature » Signature Scheme Overview

Simplified Model

Security

Requirements



S2: Calculated Signature by A

- When A receives the message plus signature, she
  - calculates a hash value for the message;
  - provides the hash value and B's public key as inputs to a digital signature verification algorithm.
- If the algorithm returns the result that the signature is valid, A is assured that the message must have been signed by B.

Digital Signature » Signature Scheme Overview

Simplified Model

Security Requirements

Outcome of Digital Signature

- No one else has B's private key and therefore no one else could have created a signature that could be verified for this message with B's public key.
- In addition, it is impossible to alter the message without access to B's private key.
- So the message is authenticated both in terms of **source** and in terms of **data integrity**.

Digital Signature » Signature Scheme Overview

Simplified Model

Requirements

From Digital Signature's Application

1. The signature must be a bit pattern that depends on the message being signed.
2. The signature must use some information only known to the sender to prevent both forgery and denial.
3. It must be practical to retain a copy of the digital signature in storage.

Digital Signature » Signature Scheme Overview

Simplified Model

Requirements

From Digital Signature's Security

4. It must be relatively easy to produce the digital signature.
5. It must be relatively easy to recognize and verify the digital signature.
6. It must be computationally infeasible to forge a digital signature,
   1. either by constructing a new message for an existing digital signature
   2. or by constructing a fraudulent digital signature for a given message.

Digital Signature » Elgamal Digital Signature Scheme

Signature Generation & Verification

Proof of Scheme Correctness

- The global elements of ElGamal digital signature are a prime number q and a, which is a primitive root of q.
- User A generates a private/public key pair as follows:
  - Generate a random integer X_A, such that $1 < X_A < q - 1$.
  - Compute $Y_A = a^{X_A} \mod q$.
  - A's private key is X_A; A's public key is $\{q, a, Y_A\}$.
- To sign a message M, user A first computes the hash $m = H(M)$, such that m is an integer in the range $0 \leq m \leq q - 1$.

Digital Signature » Elgamal Digital Signature Scheme

Signature Generation & Verification

Proof of Scheme Correctness

- A then forms a digital signature as follows:
  - Choose a random integer K such that $1 \leq K \leq q-1$ and $\gcd(K, q-1) = 1$. That is, K is relatively prime to $q-1$.
  - Compute $S_1 = a^K \mod q$. Note that this is the same as the computation of C1 for ElGamal encryption.
  - Compute $K^{-1} \mod (q-1)$. That is, compute the inverse of K modulo q - 1.
  - Compute $S_2 = K^{-1}(m - X_A^{S_1}) \mod (q-1)$.
  - The signature consists of the pair $(S1, S2)$.

Digital Signature » Elgamal Digital Signature Scheme

Signature Generation & Verification

Proof of Scheme

Correctness

- Any user B can verify the signature as follows:
  - Compute $V_1 = a^m \mod q$.
  - Compute $V_2 = (Y_A)^{S_1}(S_1)^{S_2} \mod q$.

Digital Signature » Elgamal Digital Signature Scheme

Signature Generation & Verification

Proof of Scheme Correctness

- For a prime number q, if a is a primitive root of q, then $a^1, a^2, \ldots, a^{q-1} \mod q$ are distinct. Therefore, we have the following properties:
  - For any integer m, $a^m \equiv 1 \mod q$ if and only if $m \equiv 0 \mod (q-1)$.
  - For any integers, i, j, $a^i \equiv a^j \mod q$ if and only if $i \equiv j \mod (q-1)$.

Digital Signature » Elgamal Digital Signature Scheme

Signature Generation & Verification

Proof of Scheme Correctness

$a^m \mod q = (Y_A)^{S_1}(S_1)^{S_2} \mod q$	assume $V_1 = V_2$
$a^m \mod q = a^{X_A S_1} a^{K S_2} \mod q$	substituting for Y_A and S_1
$a^{m - X_A S_1} \mod q = a^{K S_2} \mod q$	rearranging terms
$m - X_A S_1 \equiv K S_2$	property of primitive roots
$m - X_A S_1 \equiv K K^{-1}(m - X_A^{S_1})$	substituting for S_2