UNIT – III AUTHENTICATION

Digests – Requirements – MAC – Hash function – Security of Hash and MAC – Birthday Attack – MD5 – SHA – RIPEMD – Digital Signature Standard – Proof of DSS

Authentication Requirements

Disclosure

- Release of message contents to any person or process not possessing the appropriate cryptographic key

Traffic analysis

- Discovery of the pattern of traffic between parties.
- In a connection-oriented application, the frequency and duration of connections could be determined.
- the number and length of messages between parties could be determined on both environments

Masquerade

- Insertion of messages into the network from a fraudulent source.
- includes the creation of messages by an opponent that are purported to come from an authorized entity.
- Also included are fraudulent acknowledgments of message receipt or nonreceipt by someone else

Content modification

- Changes to the contents of a message, including insertion, deletion, transposition, and modification

Sequence modification

- Any modification to a sequence of messages between parties,including insertion, deletion, and reordering

Timing modification

- Delay or replay of messages.
- In a connection-oriented application, an entire session or sequence of messages could be a replay of some previous valid session, or individual messages in the sequence could be delayed or replayed.
- In a connectionless application, an individual message (e.g., datagram) could be delayed or replayed

Source repudiation

- Denial of transmission of message by source.

Destination repudiation

- Denial of receipt of message by destination

Authentication Functions

Message Authentication

- a mechanism or service used to verify the integrity of a message.
- assures that data received are exactly as sent (i.e., contain no modification, insertion, deletion, or replay).
- assures that purported identity of the sender is valid.
- When a hash function is used to provide message authentication, the hash function value is often referred to as a message digest.

Authentication function is of two levels of functionality

Lower Level

produces an authenticator: a value to be used to authenticate a message.

Higher-Level

enables a receiver to verify the authenticity of a message

Grouped Into Three Classes

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 that serves as the authenticator

Hash function

A function that maps a message of any length into a fixed-length hash value, which serves as the authenticator

Message Encryption

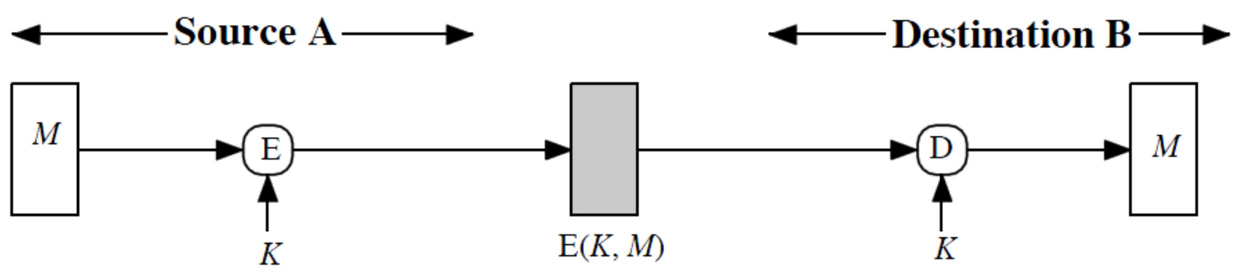
- Message encryption by itself can provide a measure of authentication.
- The analysis differs for symmetric and public-key encryption schemes

Topics

- Basic Uses of Message Encryption
- Symmetric Encryption
  - Internal Error Control
  - External Error Control
- Public-Key Encryption

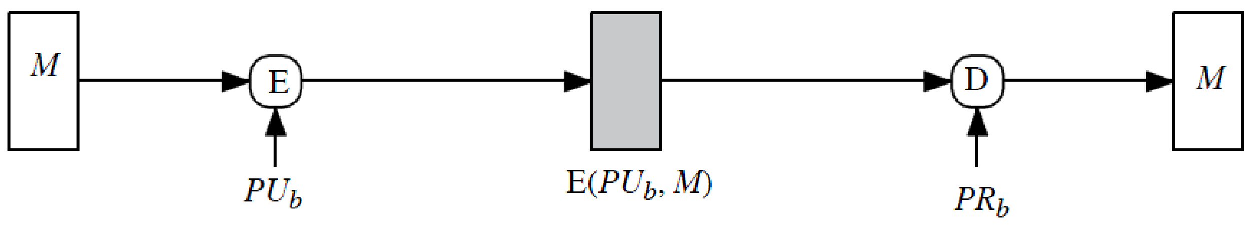
Basic Uses of Message Encryption

a) Symmetric encryption: confidentiality and authentication: A --→ B:E(K, M)



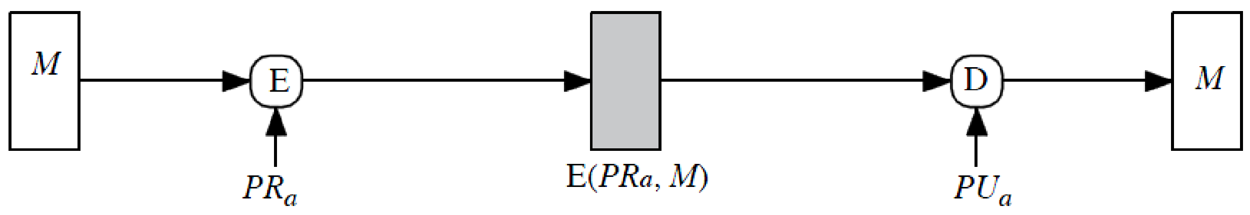
- Provides confidentiality
  - Only A and B share K
- Provides a degree of authentication
  - Could come only from A
  - Has not been altered in transit
  - Requires some formatting/redundancy
- Does not provide signature
  - Receiver could forge message
  - Sender could deny message

b) Public-key encryption: confidentiality: A → B:E(PUb, M)



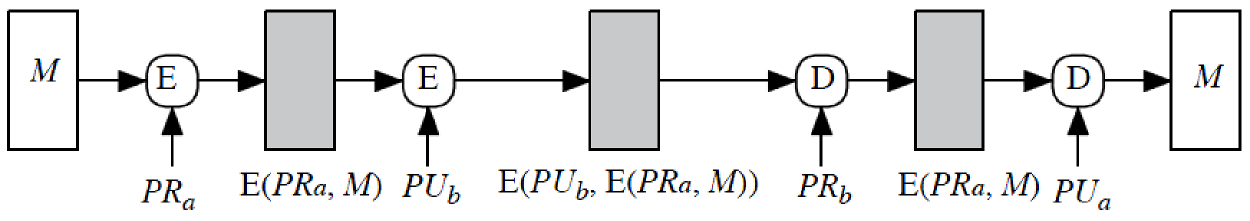
- Provides confidentiality
  - Only B has PRb to decrypt
- Provides no authentication
  - Any party could use PUb to encrypt message and claim to be A

c) Public-key encryption: authentication and signature: A → B:E(PRa, M)



- Provides authentication and signature
  - Only A has PRa to encrypt
  - Has not been altered in transit
  - Requires some formatting/redundancy
  - Any party can use PUa to verify signature

d) Public-key encryption: confidentiality, authentication, and signature: A →B:E(PUb, E(PRa, M))



- Provides confidentiality because of Pub
- Provides authentication and signature because of Pra

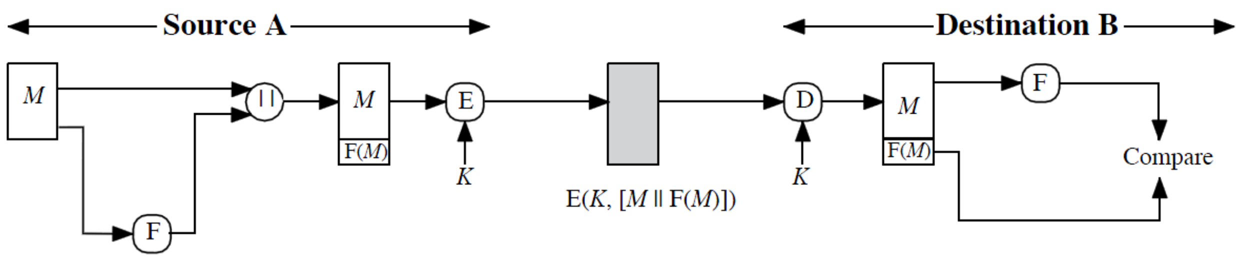
Symmetric Encryption

- A message M transmitted from source A to destination B is encrypted using a secret key K shared by both
- If no other party knows the key, then confidentiality is provided
- B is assured that the message was generated by A because A is the only other party that possesses K. Hence, authentication is provided.
- Hence, symmetric encryption provides authentication as well as confidentiality
- It may be difficult to determine automatically if incoming ciphertext decrypts to intelligible plaintext or not
  - an opponent could achieve a certain level of disruption

Solution to this problem

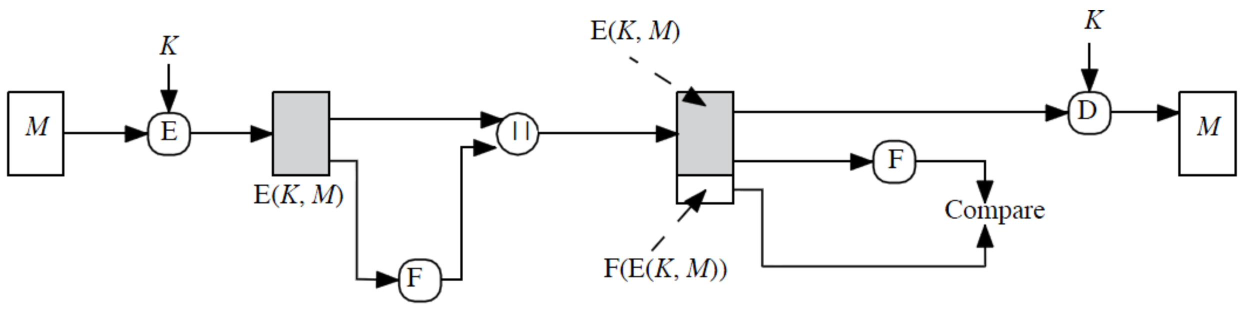
- force the plaintext to have some structure that is easily recognized but that cannot be replicated without recourse to the encryption function
- for example, append an error-detecting code, also known as a frame check sequence (FCS) or checksum, to each message before encryption
- the order in which the FCS and encryption functions are performed is critical
- Two classifications: Internal, External

Internal Error Control



- With internal error control, authentication is provided because an opponent would have difficulty generating ciphertext that, when decrypted, would have valid error control bits.
- If instead the FCS is the outer code, an opponent can construct messages with valid errorcontrol codes
- he or she can still hope to create confusion and disrupt operations

External Error Control



TCP Segment

- any sort of structuring added to the transmitted message serves to strengthen the authentication capability
- Such structure is provided by the use of a communications architecture consisting of layered protocols.
- As an example, consider the structure of messages transmitted using the TCP/IP protocol architecture
- each pair of hosts shared a unique secret key, so that all exchanges between a pair of hosts used the same key, regardless of application
- header includes not only a checksum (which covers the header) but also other useful information, such as the sequence number

Message Authentication Code

- use of a secret key to generate a small fixed-size block of data, known as a cryptographic checksum or MAC that is appended to the message.
- This technique assumes that two communicating parties, say A and B, share a common secret key K.

Theory of operation

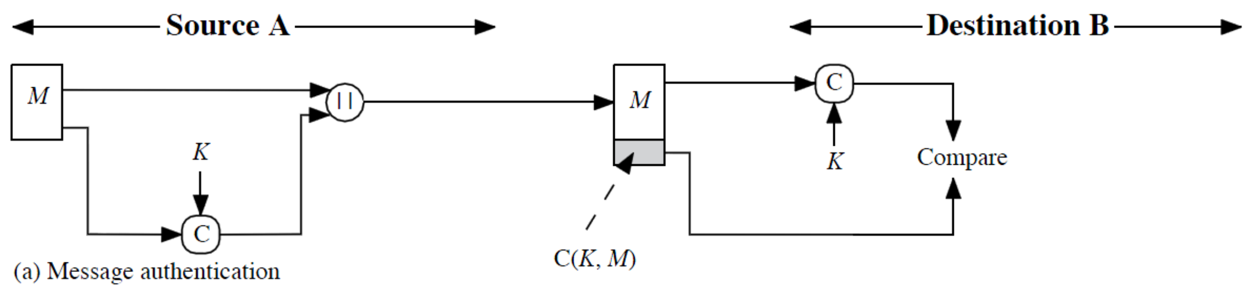
- When A has a message to send to B, it calculates the MAC as a function of the message and the key:
- MAC = C(K, M), where
  - M = input message
  - C = MAC function
  - K = shared secret key
  - MAC = message authentication code
- The message plus MAC are transmitted to the intended recipient.
- The recipient performs the same calculation on the received message, using the same secret key, to generate a new MAC.
- The received MAC is compared to the calculated MAC
- if the received MAC matches the calculated MAC, then
  - The receiver is assured that the message has not been altered
  - The receiver is assured that the message is from the alleged sender
  - If the message includes a sequence number (such as is used with HDLC, X.25, and TCP), then the receiver can be assured of the proper sequence

MAC function

- similar to encryption, difference is that the MAC algorithm need not be reversible
- many-to-one function
- The domain of the function consists of messages of some arbitrary length, whereas the range consists of all possible MACs and all possible keys
  - If an n-bit MAC is used, then there are 2^n possible MACs, whereas there are N possible messages with $N >> 2^n$
  - with a k-bit key, there are 2^k possible keys
- MAC does not provide a digital signature because both sender and receiver share the same key

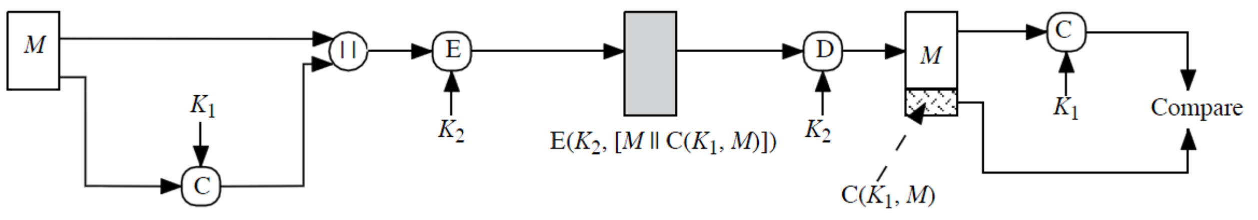
Basic Uses of Message Authentication Code (MAC)

(a) Message authentication: A→B: M||C(K, M)



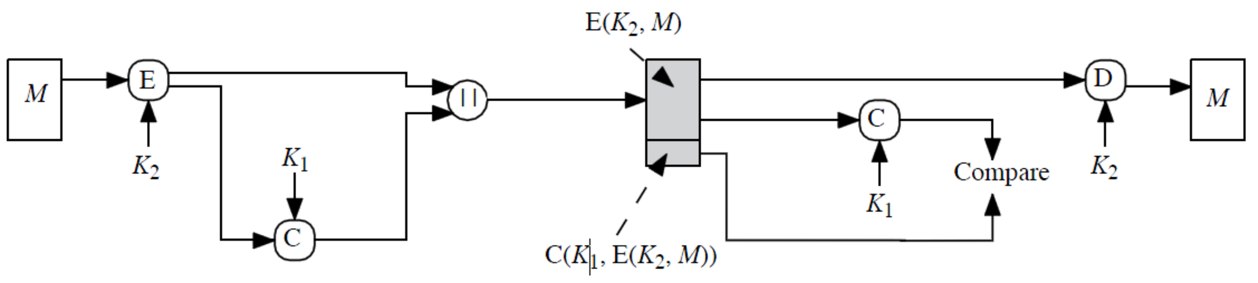
- Provides authentication: Only A and B share K

(b) Message authentication and confidentiality; authentication tied to plaintext



- A →B:E(K2, [M||C(K, M)])
- Provides authentication
  - Only A and B share K1
- Provides confidentiality
  - Only A and B share K2

(c) Message authentication and confidentiality; authentication tied to ciphertext



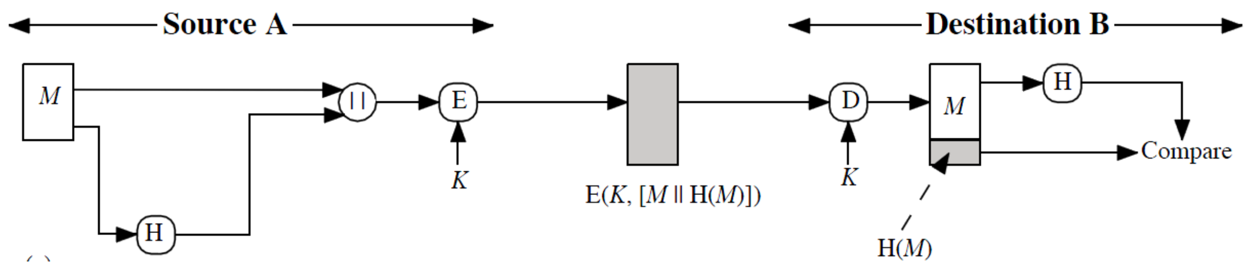
- A → B:E(K2, M)||C(K1, E(K2, M))
- Provides authentication
  - Using K1
- Provides confidentiality
  - Using K2

Hash Function

- a hash function accepts a variable-size message M as input and produces a fixedsize output, referred to as a hash code H(M).
- a hash code does not use a key but is a function only of the input message
- The hash code is also referred to as a message digest or hash value
- The hash code is a function of all the bits of the message and provides an error-detection capability:
- A change to any bit or bits in the message results in a change to the hash code

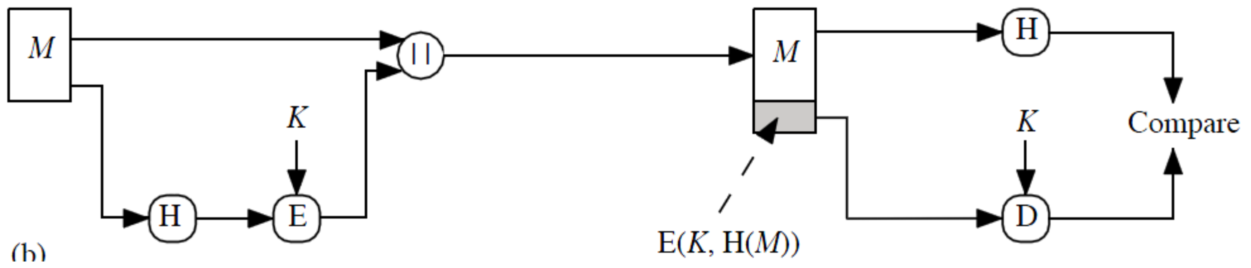
Basic Uses of Hash Function

a) Encrypt message plus hash code



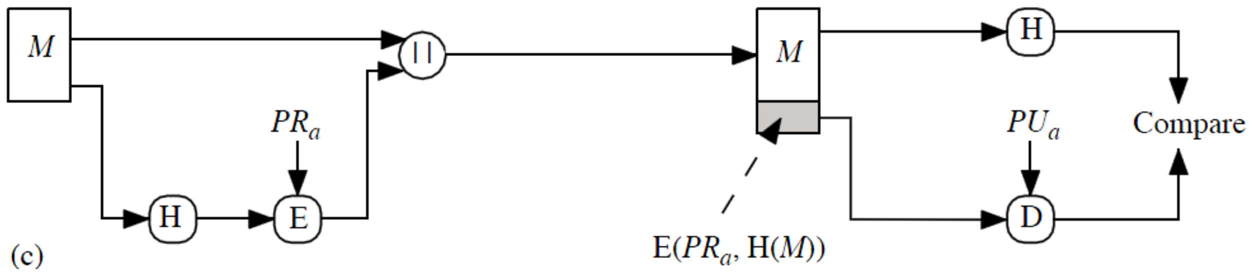
- A → B:E(K, [M||H(M)])
- Provides confidentiality
    - Only A and B share K
- Provides authentication
    - H(M) is cryptographically protected

(b) Encrypt hash code shared secret key



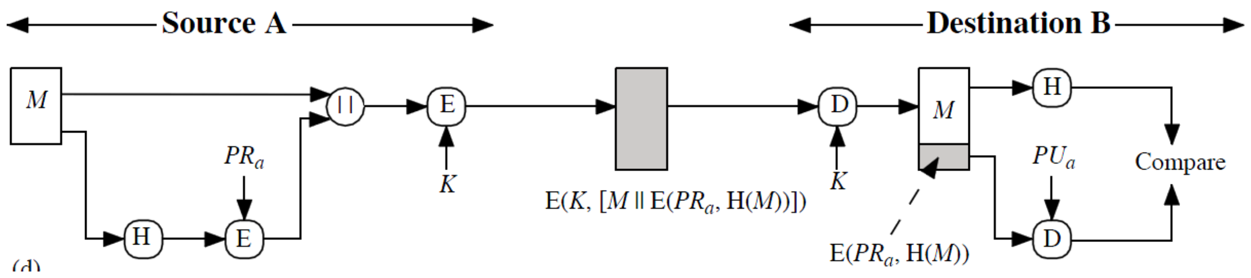
- A → B: M||E(K, H(M))
- Provides authentication
    - H(M) is cryptographically protected

(c) Encrypt hash code sender's private key



- A → B: M||E(PRa, H(M))
- Provides authentication and digital signature
    - H(M) is cryptographically protected
- Only A could create E(PRa, H(M))

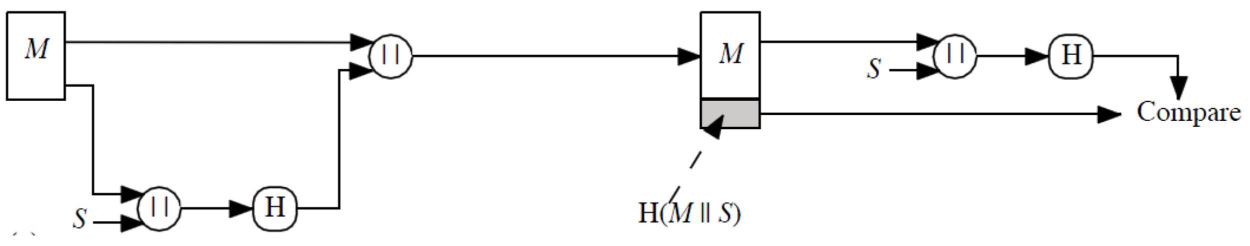
(d) Encrypt result of (c)shared secret key



- A → B: E(K, [M||E(PRa, H(M))])
- Provides authentication and digital signature

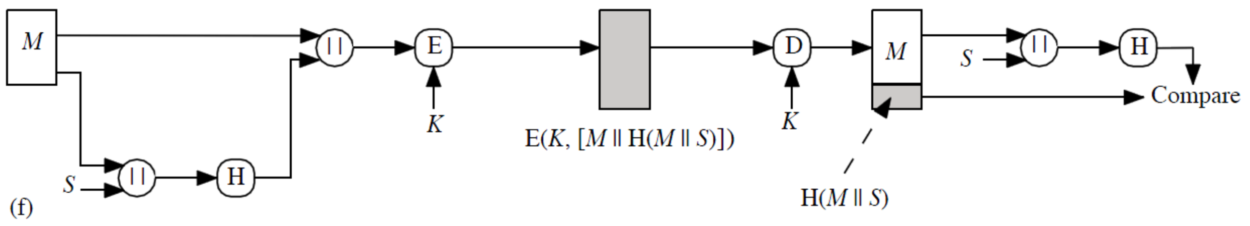
- Provides confidentiality
  - Only A and B share K

(e) Compute hash code of message plus secret value



- A → B: M||H(M||S)
- Provides authentication
  - Only A and B share S

(f) Encrypt result of (e)



- A → B: E(K, [M||H(M||S])
- Provides authentication
  - Only A and B share S
- Provides confidentiality
  - Only A and B share K

Message Authentication Codes

- General Description
- Requirements for MACs
- Message Authentication Code Based on DES
- Data Authentication Algorithm

General Description

- A MAC, is also known as a cryptographic checksum
- It is generated by a function C of the form MAC = C(K, M)
- M is a variable-length message,
- K is a secret key shared only by sender and receiver
- C(K, M) is the fixed-length authenticator
- The MAC is appended to the message at the source
- The receiver authenticates that message by recomputing the MAC

Requirements for MACs

- the security of the scheme generally depends on the bit length of the key
- an attack will require $2^{(k-1)}$ attempts for a k-bit key
- for a ciphertext-only attack, the opponent, given ciphertext C, would perform Pi = D(Ki, C) for all possible key values Ki until a Pi was produced that matched the form of acceptable plaintext.
- the MAC function is a manyto-one function

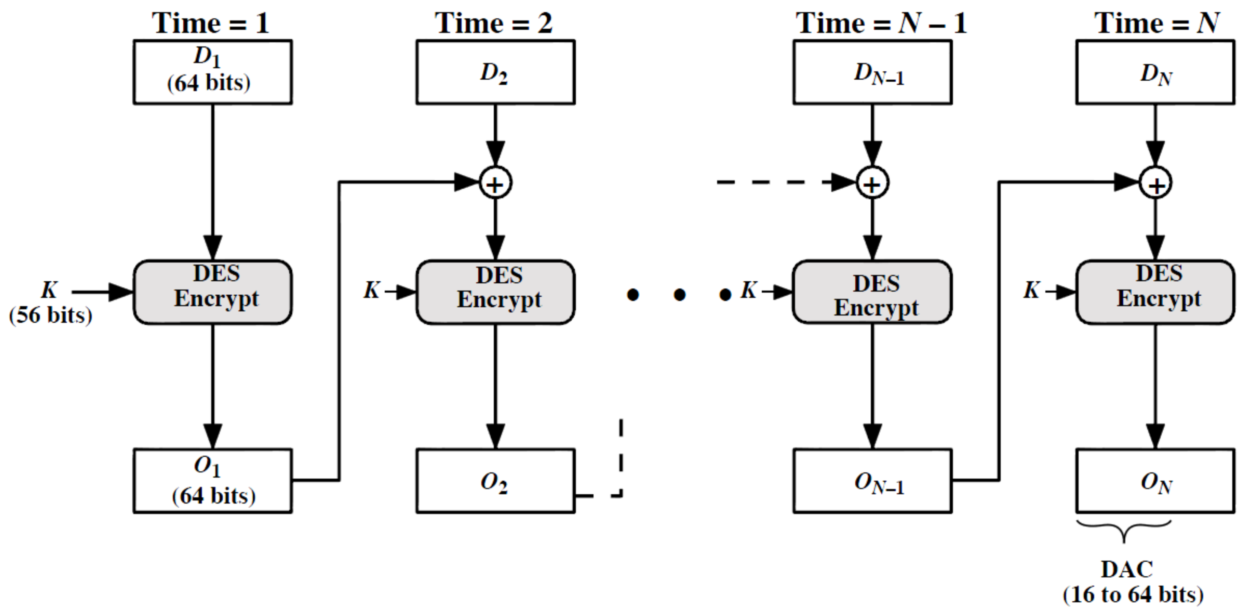
MAC function should satisfy the following requirements

- If an opponent observes M and C(K, M), it should be computationally infeasible for the opponent to construct a message M' such that C(K, M') = C(K, M).
- C(K, M) should be uniformly distributed in the sense that for randomly chosen messages, M and M', the probability that C(K, M) = C(K, M') is 2^n, where n is the number of bits in the MAC.
- Let M' be equal to some known transformation on M. That is, M' = f(M).
    - For example, f may involve inverting one or more specific bits. In that case, Pr[C(K, M) = C(K, M')] = 2^n

Message Authentication Code Based on DES

- The Data Authentication Algorithm, based on DES, has been one of the most widely used MACs
- The algorithm can be defined as using the cipher block chaining (CBC) mode of operation of DES with an initialization vector of zero
- Data is grouped into contiguous 64-bit blocks: D1, D2,..., DN.
- the final block is padded on the right with zeroes to form a full 64-bit block
- Using the DES encryption algorithm, E, secret key, K, a data authentication code (DAC) is calculated as
    - O1 = E(K, D1)
    - O2 = E(K, [D2 + O1])
    - …
    - O_N = E(K, [D_N + O_{N1}])

Data Authentication Algorithm (DAA)



The DAC consists of either the entire block O_N or the leftmost M bits of the block, with 16 ≤ M ≤ 64.

Hash Functions

- Introduction
- Requirements for a Hash Function
- Simple Hash Functions
- Two Simple Hash Functions
- Birthday Attacks
- Block Chaining Techniques

Introduction

- A hash value h is generated by a function H of the form h = H(M)
- M is a variable-length message and H(M) is the fixed-length hash value.

- The hash value is appended to the message at the source at a time when the message is assumed or known to be correct.
- The receiver authenticates that message by recomputing the hash value.
- Because the hash function itself is not considered to be secret, some means is required to protect the hash value

Requirements for a Hash Function

1. H can be applied to a block of data of any size
2. H produces a fixed-length output
3. H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical
4. For any given value h, it is computationally infeasible to find x such that H(x) = h. This is sometimes referred to in the literature as the **one-way property**.
5. For any given block x, it is computationally infeasible to find y x such that H(y) ≠ H(x). This is sometimes referred to as **weak collision resistance**.
6. It is computationally infeasible to find any pair (x, y) such that H(x) = H(y). This is sometimes referred to as **strong collision resistance**

Simple Hash Functions

- All hash functions operate using the following general principles.
- 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 iterative fashion to produce an n-bit hash function.

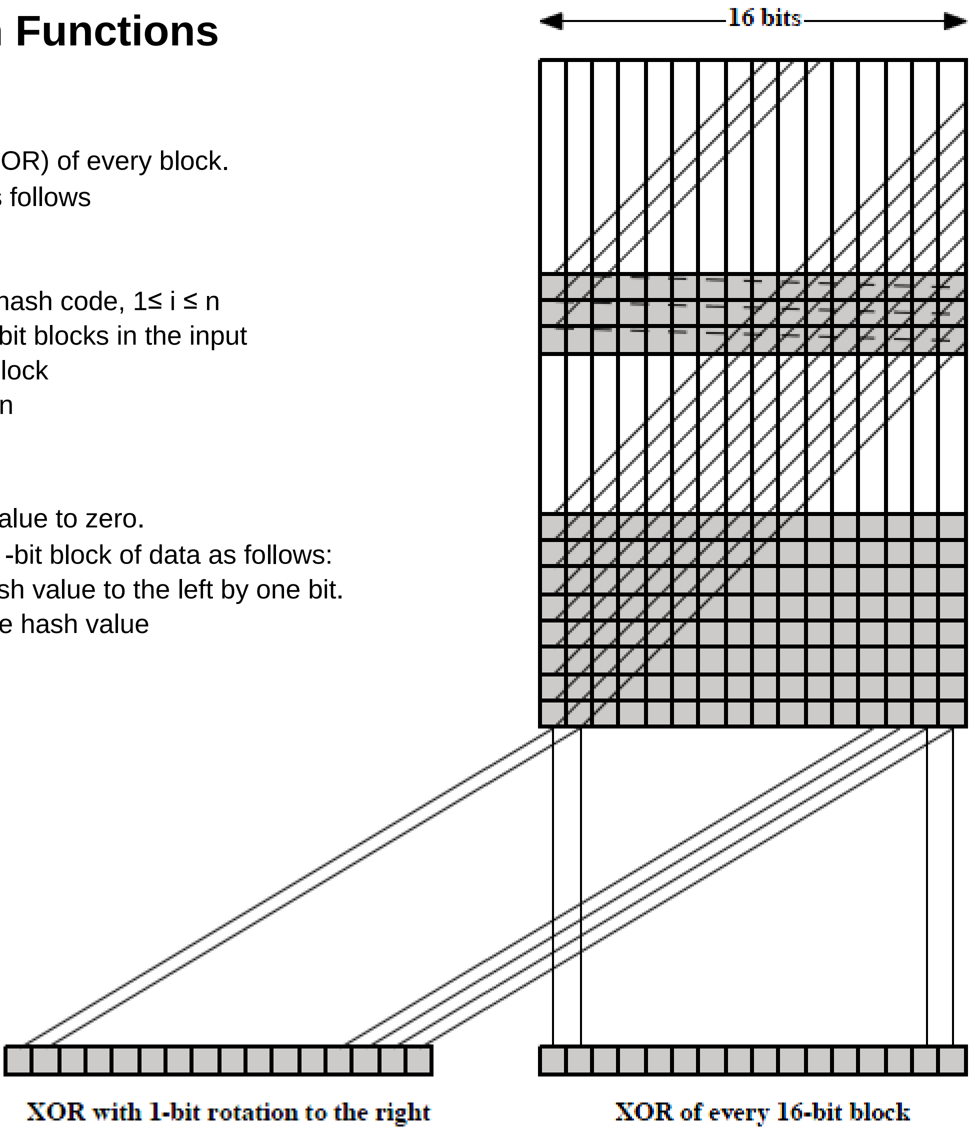
Two Simple Hash Functions

Method 1

- bit-by-bit exclusive-OR (XOR) of every block.
- This can be expressed as follows
  - $C_i = b_{i1} \wedge b_{i2} ..\wedge. b_{im}$
  - where
    - Ci = ith bit of the hash code, 1≤ i ≤ n
    - m = number of n-bit blocks in the input
    - bij = ith bit in jth block
    - ^ = XOR operation

Method 2

- Initially set the -bit hash value to zero.
- Process each successive -bit block of data as follows:
  - Rotate the current hash value to the left by one bit.
  - XOR the block into the hash value


16 bits
XOR with 1-bit rotation to the right
XOR of every 16-bit block

Birthday Attacks

Birthday paradox

- The birthday paradox is often presented in elementary probability courses to demonstrate that probability results are sometimes counterintuitive.
- What is the minimum value of k such that the probability is greater than 0.5 that at least two people in a group of k people have the same birthday?
- Ignore February 29 and assume that each birthday is equally likely

Birthday Attack

- The source, A, is prepared to "sign" a message by appending the appropriate m-bit hash code and encrypting that hash code with A's private key
- The opponent generates $2^{m/2}$ variations on the message, all of which convey essentially the same meaning
  - The opponent prepares an equal number of messages, all of which are variations on the fraudulent message to be substituted for the real one.
- The two sets of messages are compared to find a pair of messages that produces the same hash code.
  - The probability of success, by the birthday paradox, is greater than 0.5.
  - If no match is found, additional valid and fraudulent messages are generated until a match is made
- The opponent offers the valid variation to A for signature.
  - This signature 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; the opponent is assured of success even though the encryption key is not known

Block Chaining Techniques (Skip)

Meet in the middle attack possible

Security of Hash Functions and Macs

- Brute-Force Attacks
  - Hash Functions
  - Message Authentication Codes
- Cryptanalysis
  - Hash Functions
  - Message Authentication Codes

Brute-Force Attacks

- The nature of brute-force attacks differs somewhat for hash functions and MACs

Hash Functions

The strength of a hash function against brute-force attacks depends solely on the length of the hash code produced by the algorithm

there are three desirable properties

Property	Description	Effort Needed
One-way	For any given code h, it is computationally infeasible to find x such that H(x) = h	2^n
Weak collision resistance	For any given block x, it is computationally infeasible to find y ≠ x with H(y) = H(x).	2^n
Strong collision resistance	It is computationally infeasible to find any pair (x, y) such that H (x) = H(y).	$2^{n/2}$

Message Authentication Codes

- A brute-force attack on a MAC is a difficult undertaking because it requires known message-MAC pairs
- Security of the MAC algorithm depends on the relative size of the key and the MAC.
- we need to state the desired security property of a MAC algorithm

Computation resistance

Given one or more text-MAC pairs [xi, C(K, xi)], it is computationally infeasible to compute any text-MAC pair [x, C(K, x)] for any new input x ≠ xi

There are two lines of attack possible:

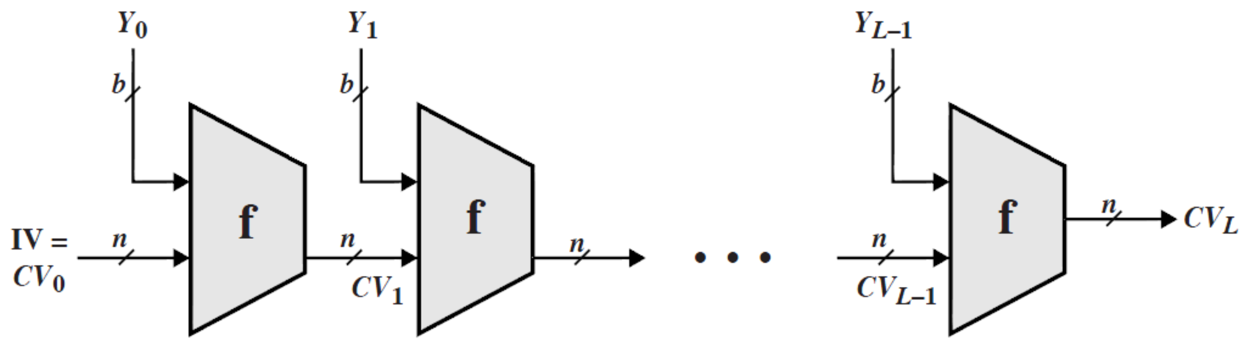
- Attack the key space
- attack the MAC value

Cryptanalysis

an ideal hash or MAC algorithm will require a cryptanalytic effort greater than or equal to the brute-force effort

Hash Functions

General Structure of Secure Hash Code



- The hash function takes an input message and partitions it into L fixed-sized blocks of b bits each.
- If necessary, the final block is padded to b bits.
- The final block also includes the value of the total length of the input to the hash function.
- The inclusion of the length makes the job of the opponent more difficult.
- Either the opponent must find two messages of equal length that hash to the same value or two messages of differing lengths that, together with their length values, hash to the same value.
- The hash algorithm involves repeated use of a compression function, f,

The hash function can be summarized as

$$CV_0 = IV = \text{initial } n\text{-bit value}$$
$$CV_i = f(CV_{i-1}, Y_{i-1}) \quad 1 \le i \le L$$
$$H(M) = CV_L$$

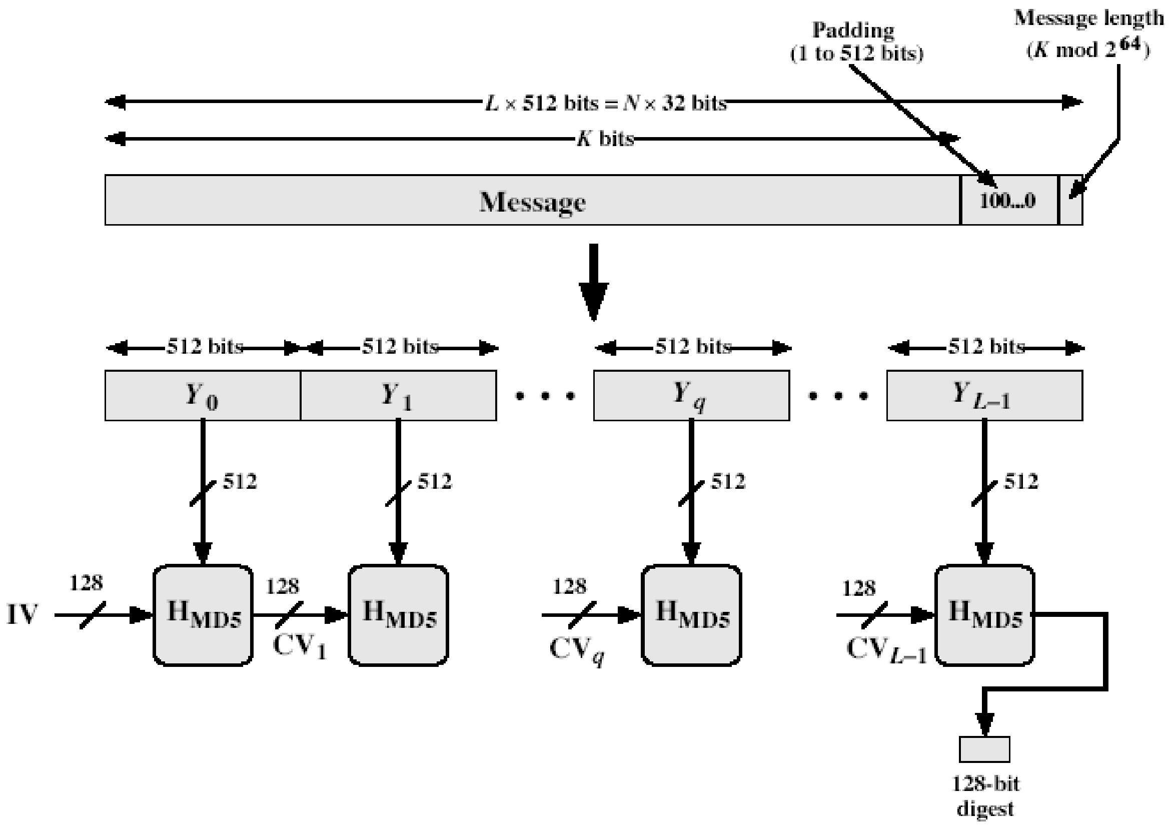
Message Authentication Codes

- There is much more variety in the structure of MACs than in hash functions
- Hence it is difficult to generalize about the cryptanalysis of MACs.
- far less work has been done on developing such attacks

MD5

MD5 Overview

- pad message so its length is 448 mod 512
- append a 64-bit length value to message
- initialise 4-word (128-bit) MD buffer (A,B,C,D)
- process message in 16-word (512-bit) blocks:
  - using 4 rounds of 16 bit operations on message block & buffer
  - add output to buffer input to form new buffer value
- output hash value is the final buffer value



MD5 Compression Function

- each round has 16 steps of the form:
- a = b+((a+g(b,c,d)+X[k]+T[i])<<<s)
- a,b,c,d refer to the 4 words of the buffer
  - this updates 1 word only of the buffer
  - after 16 steps each word is updated 4 times
- where g(b,c,d) is a different nonlinear function in each round (F,G,H,I)
- T[i] is a constant value derived from sin

Strength of MD5

- MD5 hash is dependent on all message bits
- Rivest claims security is good as can be
- known attacks are:
  - Berson 92 attacked any 1 round using differential cryptanalysis (but can't extend)
  - Boer & Bosselaers 93 found a pseudo collision (again unable to extend)
  - Dobbertin 96 created collisions on MD compression function (but initial constants prevent exploit)
- conclusion is that MD5 looks vulnerable soon

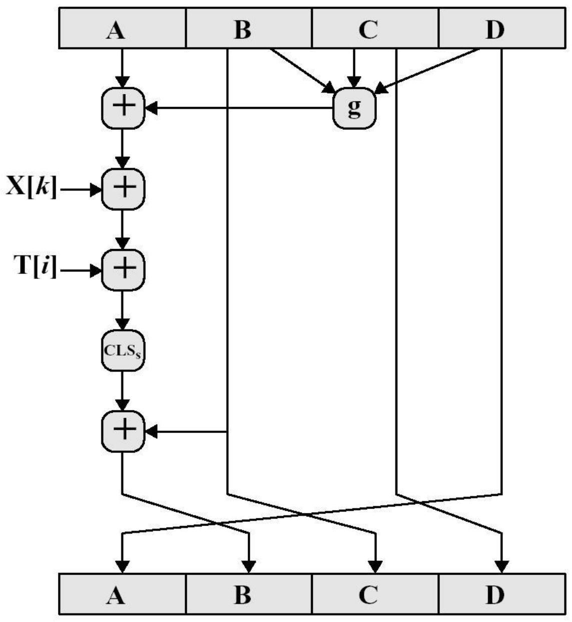


Figure 9.3 Elementary MD5 Operation (single step)

Processing 512 Bit Block

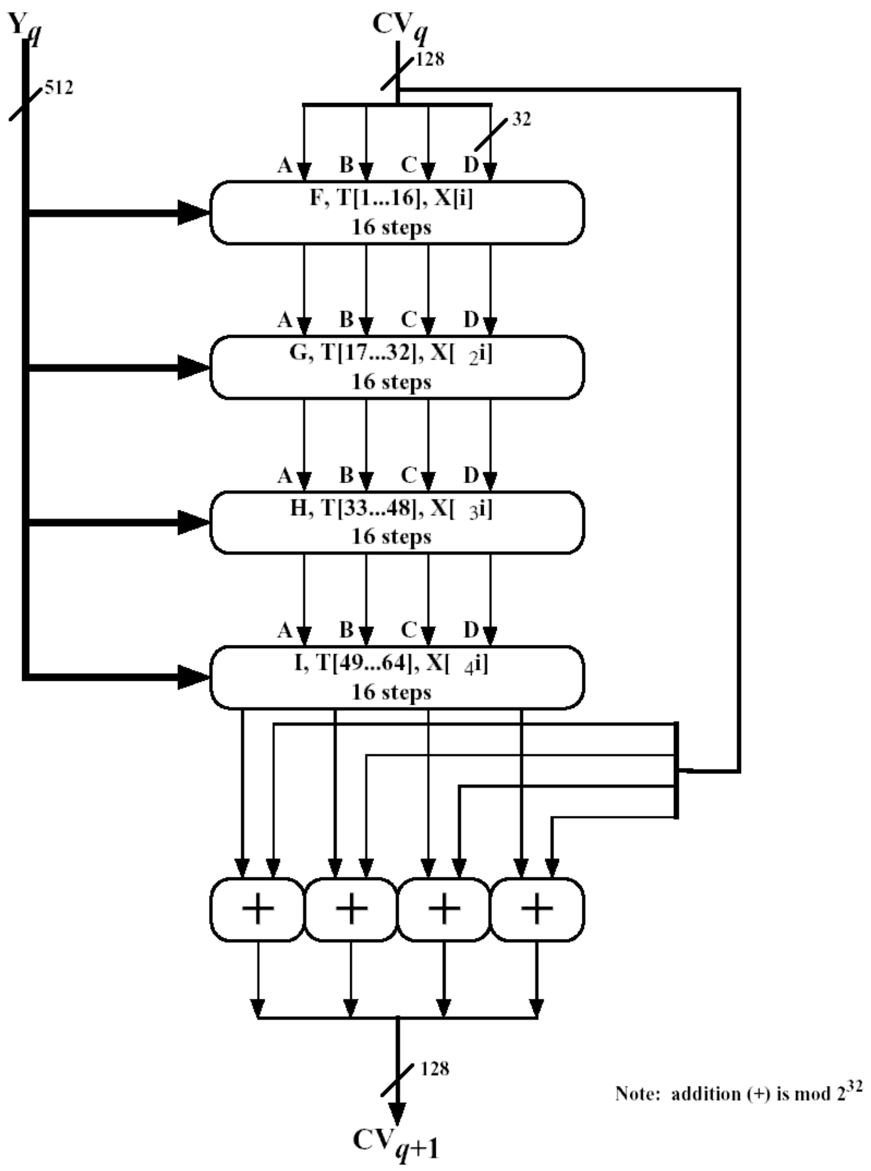

Y_q
512
CV_q
128
32
A B C D
F, T[1...16], X[i]
16 steps
G, T[17...32], X[ρ_2i]
16 steps
H, T[33...48], X[ρ_3i]
16 steps
I, T[49...64], X[ρ_4i]
16 steps
+
128
CV_{q+1}
Note: addition (+) is mod 2^{32}

Figure 9.2 MD5 Processing of a Single 512-bit Block (MD5 Compression Function)

Secure Hash Algorithm (SHA)

Comparison of SHA Parameters

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Message Digest Size	160	224	256	384	512
Message Size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block Size	512	512	512	1024	1024
Word Size	32	32	32	64	64
Number of Steps	80	64	64	80	80

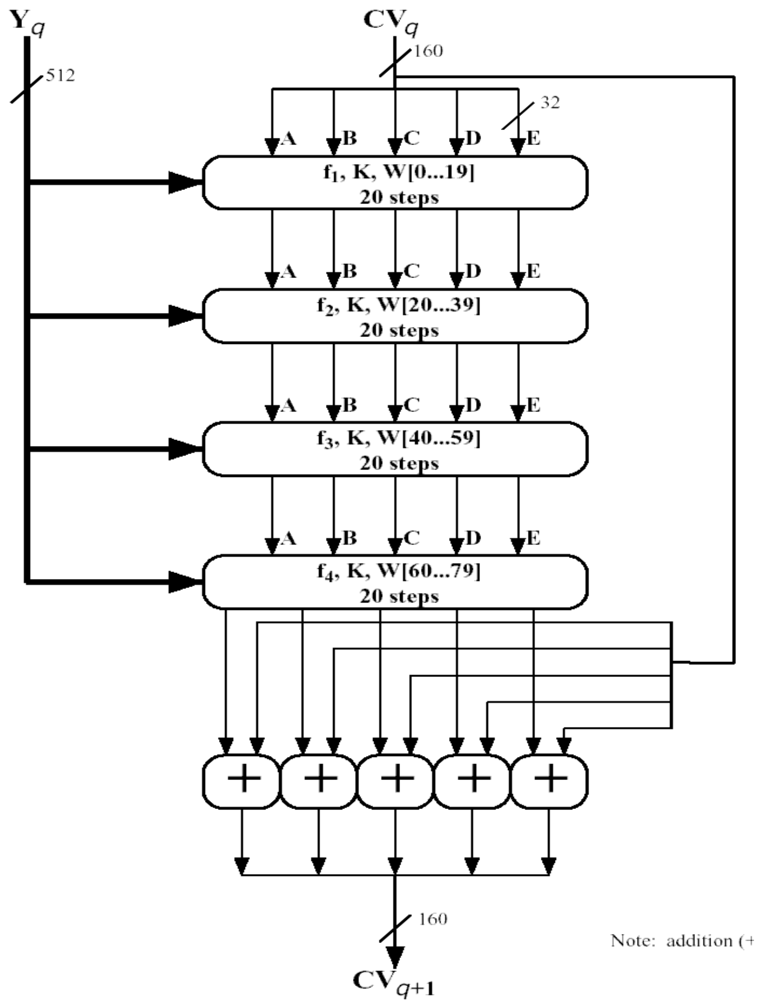
Note: All sizes are measured in bits.

Overview

- most widely used hash function
- produces 160-bit hash values
- based on MD5
- The input is a message with length is < 2^64 bits
- The output is a message digest of length 160 bit
- The processing is done 512 bit blocks

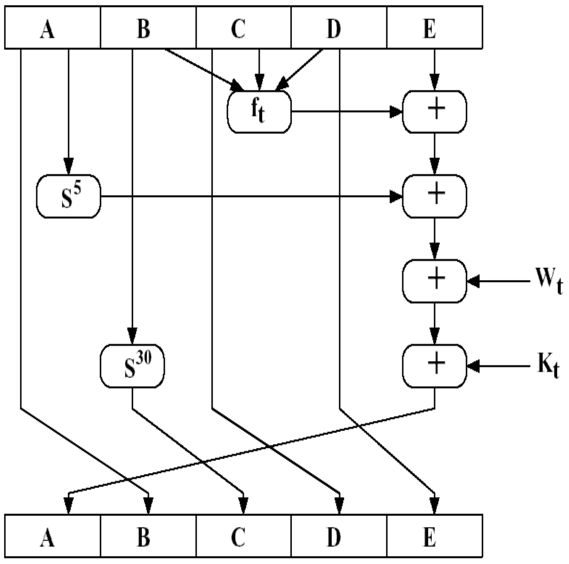
Algorithm

Processing of a single 512 bit block



Y_q
512
CV_q
160
32
A B C D E
f_1, K, W[0...19] 20 steps
f_2, K, W[20...39] 20 steps
f_3, K, W[40...59] 20 steps
f_4, K, W[60...79] 20 steps
160
CV_{q+1}
Note: addition (+

Single Round Function



Steps

Step 1: Append Padding Bits

Message is “padded” with a 1 and as many 0’s as necessary to bring the message length to 64 bits fewer than an even multiple of 512

Step 2: Append Length

64 bits are appended to the end of the padded message. These bits hold the binary format of 64 bits indicating the length of the original message

Step 3: Initialize MD buffer

- 160 bit buffer is used to hold the intermediate and final results
- The buffer is represented as 5 – 32 bit registes A, B, C, D, E
- Initialized as A = 0x67452301, B = 0xEFCDAB89, C = 0x98BADCFE, D = 0x10325476, E = 0xC3D2E1F0

Step 4: Process the message in 512 bit word blocks

- The number of rounds = 4
- Each round has 20 steps
- Four different logical functions f1, f2, f3 and f4
- Each round makes use of additive constant Kt where 0 < t < 79
- 4 different constants are used 0x5A827999, 0x6ED9EBA1, 0x8F1BBCDC, 0xCA62C1D6
- The output of the 80th round is added to the input to the 1st round to produce the output

Step 5: Output

- After all 512 bit block is processed, the output from the Lth stage is 160 bit message digest
- CV0 = IV
- Yq+1 = sum32 (CVq, ABCDEq)
- MD = CVL
- IV: Initial value of the ABCDE buffer
- ABCDEq = Output of the last round of processing of the qth block
- L = The no of blocks (after padding and appending length)
- Sum32 = Addition modulo 2^32 done on each word separately
- MD = Final Message Digest

SHA-1 Compression Function

Describe the picture

Comparison of SHA-1 and MD5

Security against Brute Force Attack

- SHA, operations are 2 ^ 160
- MD5, operations are 2 ^ 128
- It is difficult to have same message digest for 2 different messages
- brute force attack is harder for SHA-1(160 vs 128 bits for MD5)

Security against crypt analysis

- MD5 is vulnerable, SHA1 is not vulnerable

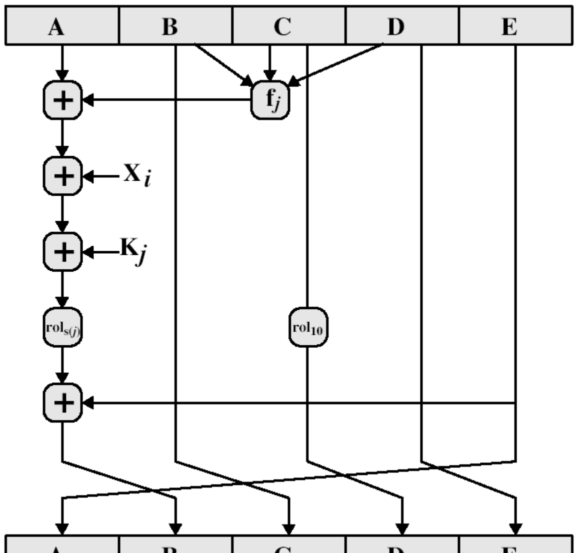
Speed

- MD5 executes faster when compared to SHA1 due to less bits needed, 64 steps versus 80 steps
- Both depend on addition modulo of 2^32 and could run well on 32 bit processors

Simplicity and Compactness

- Both are simple to describe and implement
- MD5 uses little endian, SHA 1 uses big endian

RACE Integrity Primitives Evaluation Message Digest (RIPEMD)

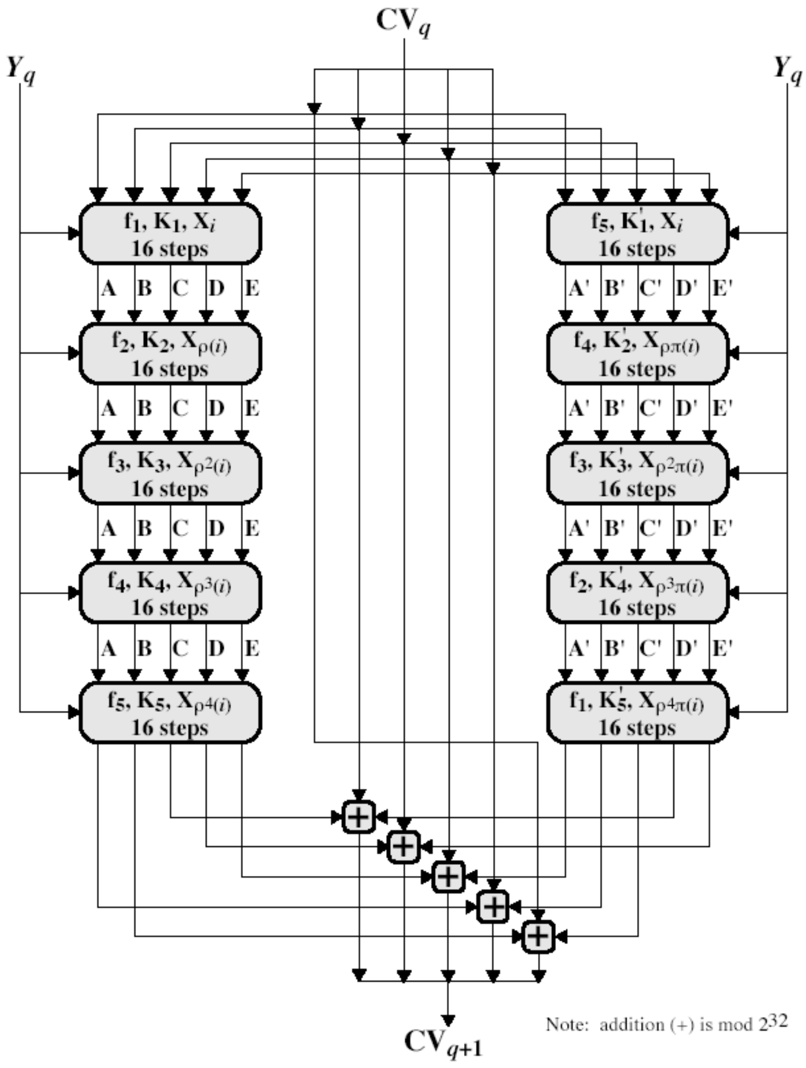


- RIPEMD-160 was developed in Europe as part of RIPE project in 96
- by researchers involved in attacks on MD4/5
- initial proposal strengthen following analysis to become RIPEMD-160
- somewhat similar to MD5/SHA
- uses 2 parallel lines of 5 rounds of 16 steps
- creates a 160-bit hash value
- slower, but probably more secure, than SHA

RIPEMD-160 Overview

- pad message so its length is 448 mod 512
- append a 64-bit length value to message
- initialise 5-word (160-bit) buffer (A,B,C,D,E) to

- (67452301,efcdab89,98badcfe,
- 10325476,c3d2e1f0)
- process message in 16-word (512-bit) chunks:
  - use 10 rounds of 16 bit operations on message block & buffer – in 2 parallel lines of 5
  - add output to input to form new buffer value
- output hash value is the final buffer value



CV$_q$
Y$_q$
Y$_q$
f$_1$, K$_1$, X$_i$ 16 steps
f$_2$, K$_2$, X$_{\rho(i)}$ 16 steps
f$_3$, K$_3$, X$_{\rho^2(i)}$ 16 steps
f$_4$, K$_4$, X$_{\rho^3(i)}$ 16 steps
f$_5$, K$_5$, X$_{\rho^4(i)}$ 16 steps
f$_5$, K'$_1$, X$_i$ 16 steps
f$_4$, K'$_2$, X$_{\rho\pi(i)}$ 16 steps
f$_3$, K'$_3$, X$_{\rho^2\pi(i)}$ 16 steps
f$_2$, K'$_4$, X$_{\rho^3\pi(i)}$ 16 steps
f$_1$, K'$_5$, X$_{\rho^4\pi(i)}$ 16 steps
A B C D E
A' B' C' D' E'
CV$_{q+1}$
Note: addition (+) is mod 2^{32}

RIPEMD-160 Compression Function

Refer Picture

RIPEMD-160 Design Criteria

- use 2 parallel lines of 5 rounds for increased complexity
- for simplicity the 2 lines are very similar
- step operation very close to MD5
- permutation varies parts of message used
- circular shifts designed for best results

RIPEMD-160 verses MD5 & SHA-1

- brute force attack harder (160 like SHA-1 vs 128 bits for MD5)
- not vulnerable to known attacks, like SHA-1 though stronger (compared to MD4/5)
- slower than MD5 (more steps)
- all designed as simple and compact
- SHA-1 optimised for big endian CPU's vs RIPEMD-160 & MD5 optimised for little endian CPU's

Digital Signature Standard

- Introduction
- Digital Signature
  - Requirements
  - Categories
- Digital Signature Standard
- Approaches to Digital Signatures
  - DSS Approach
  - RSA Approach
- The Digital Signature Algorithm
- DSS Signing and Verifying

Introduction

- A digital signature is an authentication mechanism that enables the creator of a message to attach a code that acts as a signature. The signature is formed by taking the hash of the message and encrypting the message with the creator's private key.
- The signature guarantees the source and integrity of the message.
- Mutual authentication protocols enable communicating parties to satisfy themselves mutually about each other's identity and to exchange session keys.
- In one-way authentication, the recipient wants some assurance that a message is from the alleged sender.
- The digital signature standard (DSS) is an NIST standard that uses the secure hash algorithm (SHA).

Digital Signature

Requirements

- The signature must be a bit pattern that depends on the message being signed.
- The signature must use some information unique to the sender, to prevent both forgery and denial
- It must be relatively easy to produce the digital signature.
- It must be relatively easy to recognize and verify the digital signature.
- It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message.
- It must be practical to retain a copy of the digital signature in storage.

Two categories

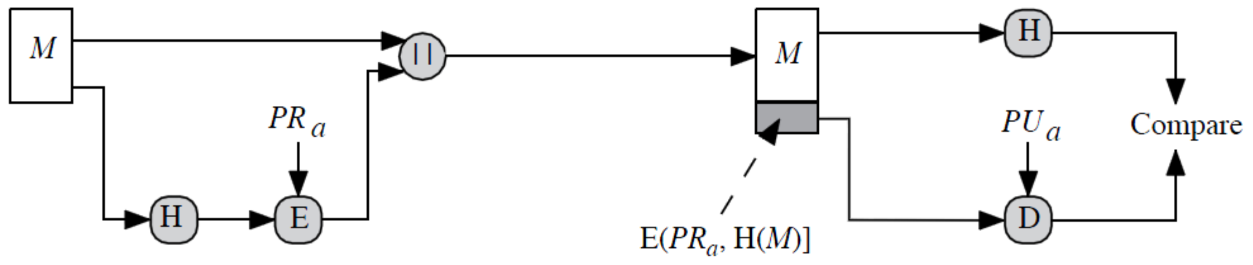
- Direct Digital Signature
- Arbitrated Digital Signature

Digital Signature Standard

- The National Institute of Standards and Technology (NIST) has published Federal Information Processing Standard FIPS 186, known as the Digital Signature Standard (DSS).
- The DSS makes use of the Secure Hash Algorithm (SHA)
- DSS presents a new digital signature technique, the Digital Signature Algorithm (DSA)
- latest version also incorporates digital signature algorithms based on RSA and on elliptic curve cryptography

Two Approaches to Digital Signatures

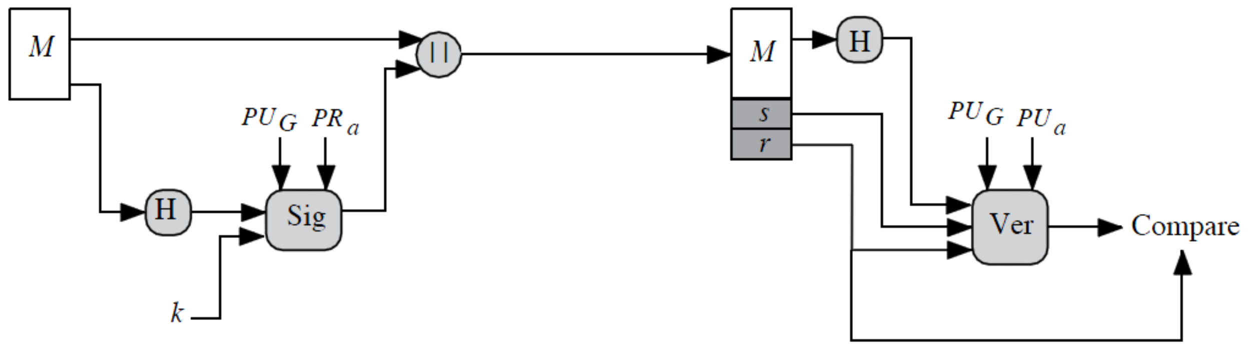
RSA Approach



- In the RSA approach, the message to be signed is input to a hash function that produces a secure hash code of fixed length.
- This hash code is then encrypted using the sender's private key to form the signature.
- Both the message and the signature are then transmitted.
- The recipient takes the message and produces a hash code.
- The recipient also decrypts the signature using the sender's public key.
- If the calculated hash code matches the decrypted signature, the signature is accepted as valid.
- Because only the sender knows the private key, only the sender could have produced a valid signature.

DSS Approach

- The DSS uses an algorithm that is designed to provide only the digital signature function
- cannot be used for encryption or key exchange
- it is a public-key technique

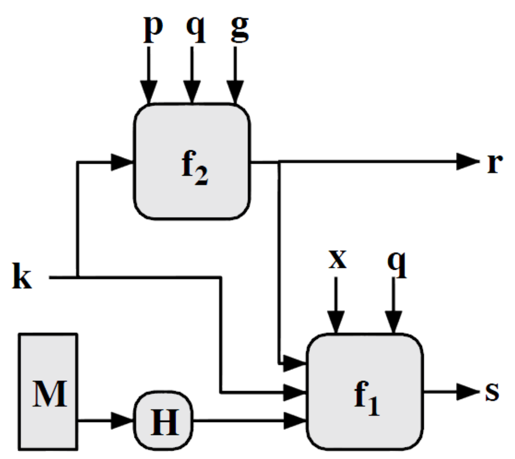


The Digital Signature Algorithm

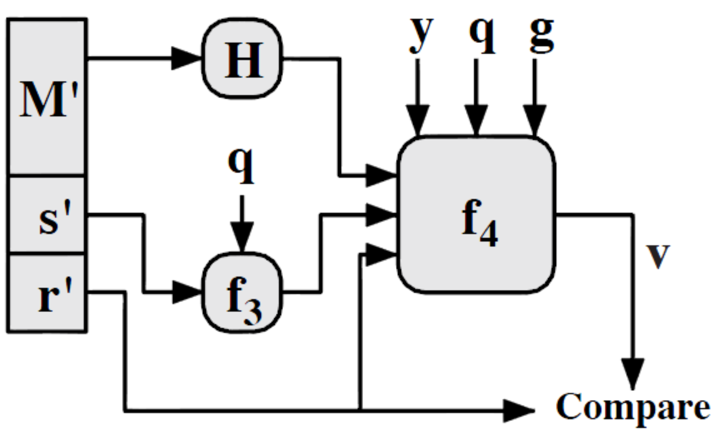
- Global Public-Key Components
- User's Private Key: x
  - random or pseudorandom integer with 0 < x < q
- User's Public Key: y
  - g^x mod p
- User's Per-Message Secret Number: k
  - random or pseudorandom integer with 0 < k < q
- Signing
  - r = (g^k mod p) mod q
  - s = [k^-1 (H(M) + xr)] mod q
  - Signature = (r, s)
- Verifying

DSS Signing and Verifying

Signing



Verifying



Proof of DSS (Skip)

TBD

Disclaimer

Intended for educational purposes only. Not intended for any sort of commercial use

Purely created to help students with limited preparation time

Text and picture used were taken from the reference items

Reference

Cryptography and Network Security, Fourth Edition – William Stallings

Credits

Thanks to my family members who supported me, while I spent considerable amount of time to prepare these notes.
Feedback is always welcome at GHCRajan@gmail.com