Unit IV
Cryptographic Hash Functions and Digital Signatures

Presented by:
Er. Kobid Karkee
Himalaya College of Engineering

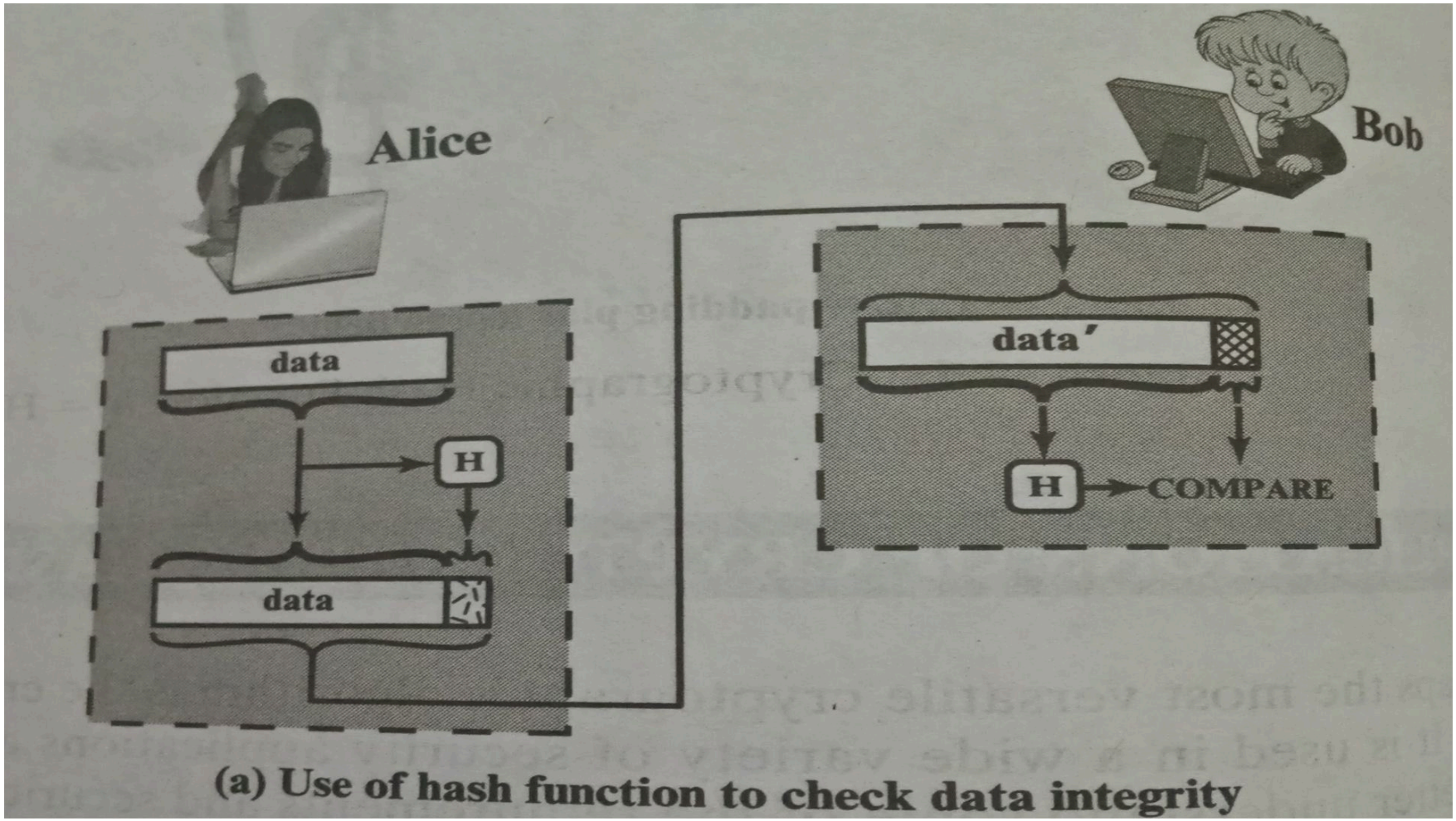
Message Authentication

- Message authentication is a mechanism or service used to verify the integrity of a message.
- Message authentication assures that data received are exactly as sent (i.e., there is no modification, insertion, deletion, or replay).
- In many cases, there is a requirement that the authentication mechanism 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.

Message Authentication

- The essence of the use of a hash function for message integrity is as follows.
  - The sender computes a hash value as a function of the bits in the message and transmits both the hash value and the message.
  - The receiver performs the same hash calculation on the message bits and compares this value with the incoming hash value.
  - If there is a mismatch, the receiver knows that the message (or possibly the hash value) has been altered.

Message Authentication

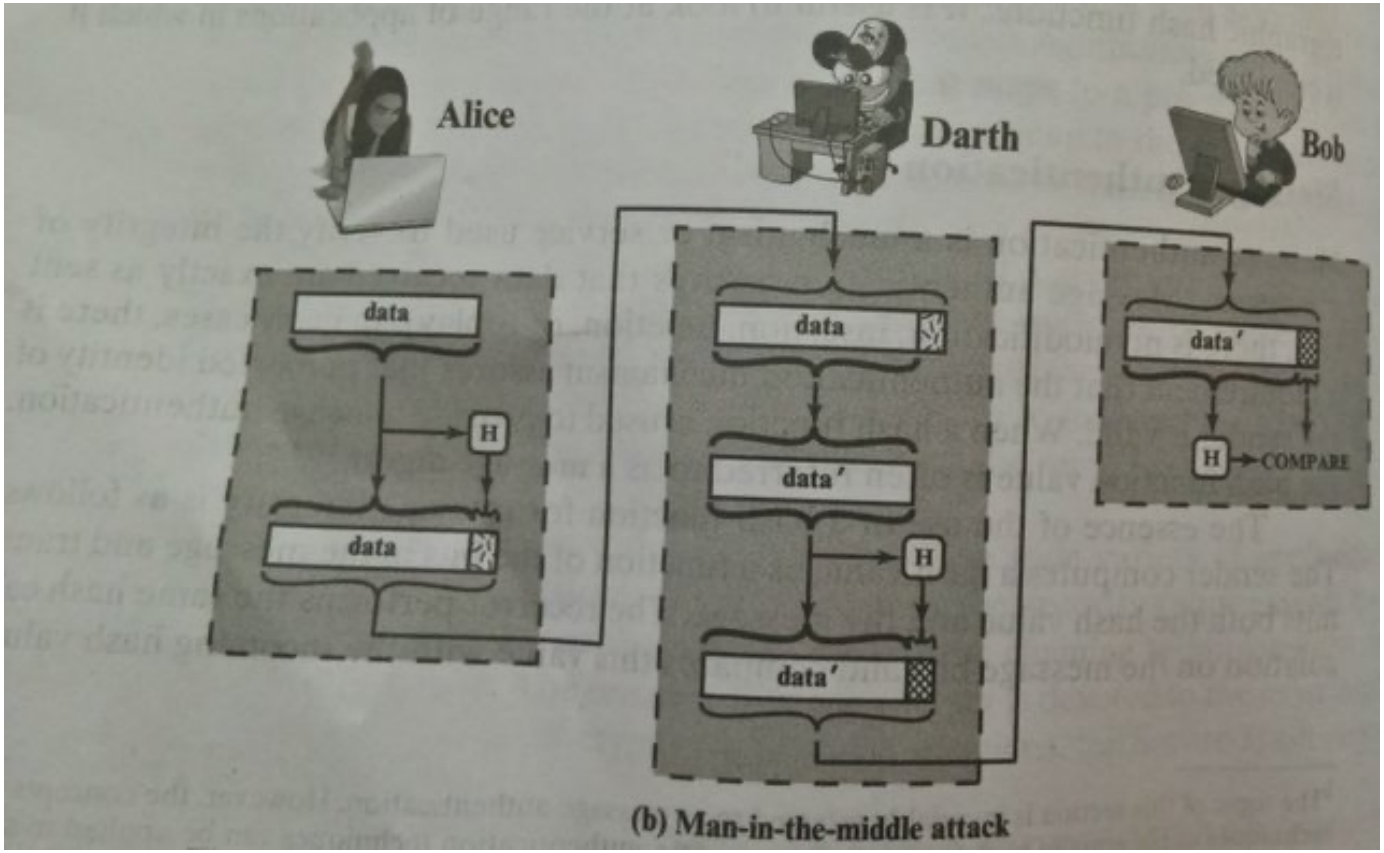


(a) Use of hash function to check data integrity

Message Authentication

- The hash value must be transmitted in a secure fashion.
- That is, the hash value must be protected so that if an adversary alters or replaces the message, it is not feasible for adversary to also alter the hash value to fool the receiver.
- This type of attack is shown in Figure (b).
- In this example, Alice transmits a data block and attaches a hash value.
- Darth intercepts the message, alters or replaces the data block, and calculates and attaches a new hash value.
- Bob receives the altered data with the new hash value and does not detect the change.
- To prevent this attack, the hash value generated by Alice must be protected.

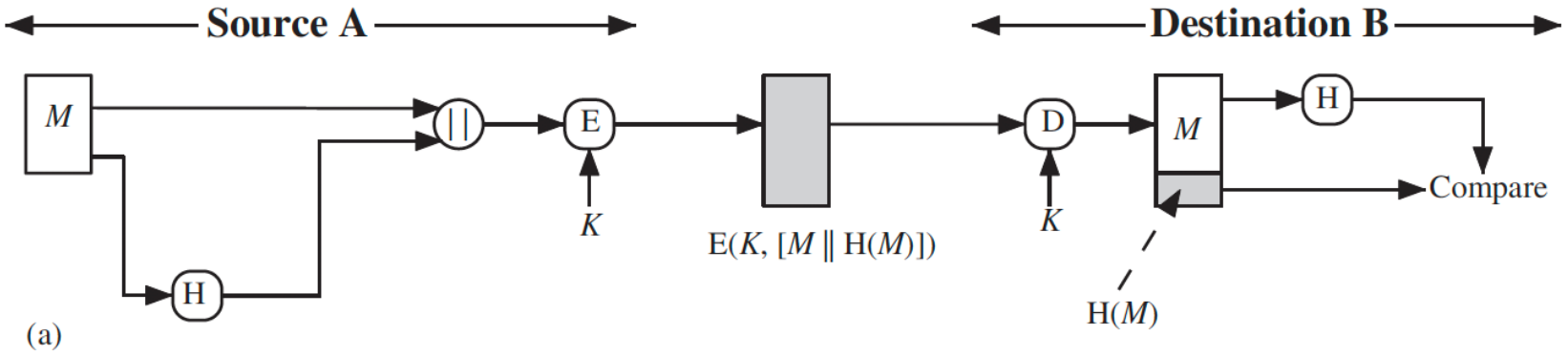
Message Authentication



Use of Hash Function for Message Authentication

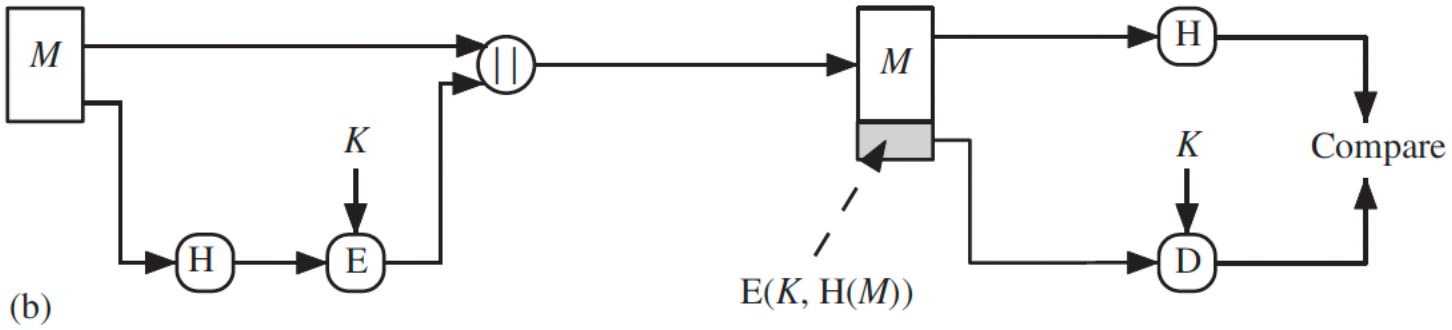
Example 1

- The message plus concatenated hash code is encrypted using symmetric encryption.
- Because only A and B share the secret key, the message must have come from A and has not been altered.
- The hash code provides the structure or redundancy required to achieve authentication.
- Because encryption is applied to the entire message plus hash code, confidentiality is also provided.



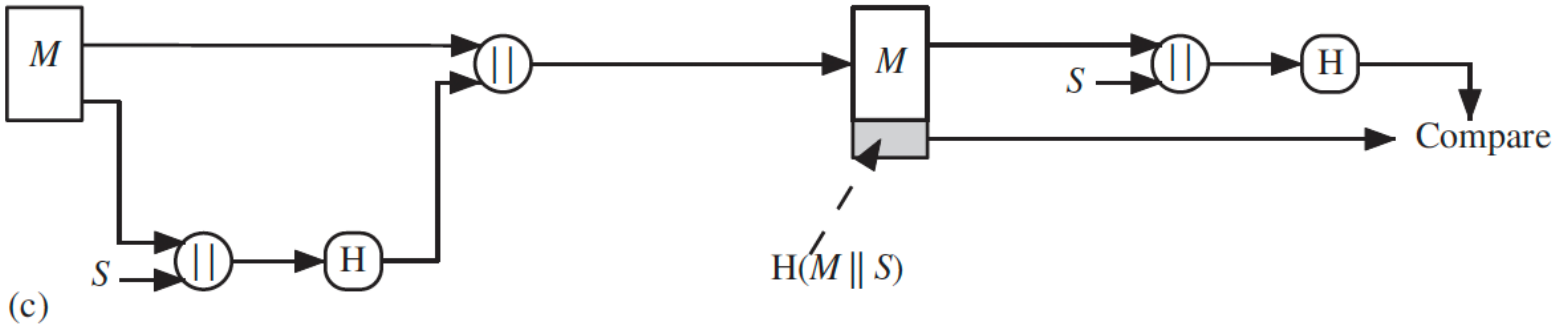
Example 2

- Only the hash code is encrypted, using symmetric encryption.
- This reduces the processing burden for those applications that do not require confidentiality.



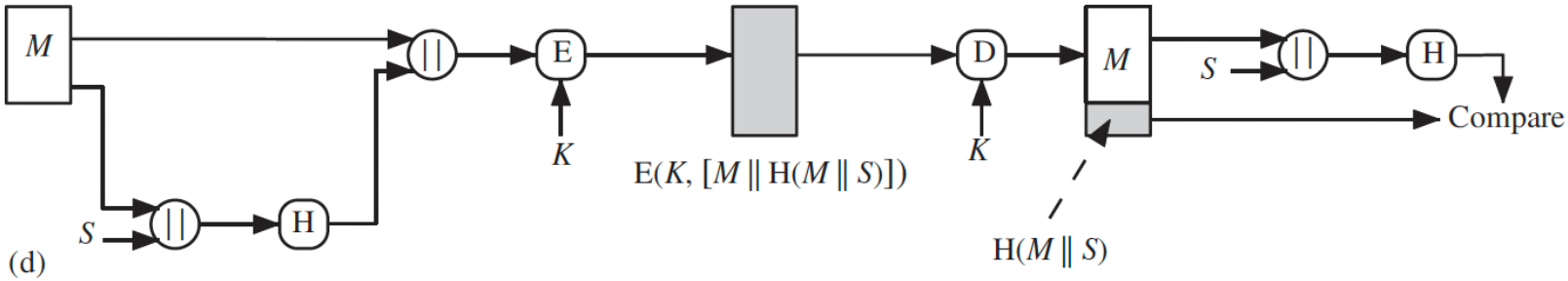
Example 3

- It is possible to use a hash function but no encryption for message authentication.
- The technique assumes that the two communicating parties share a common secret value S.
- A computes the hash value over the concatenation of M and S and appends the resulting hash value to M.
- Because B possesses S, it can recompute the hash value to verify.
- Because the secret value itself is not sent, an opponent cannot modify an intercepted message and cannot generate a false message.



Example 4

- Confidentiality can be added to the approach of example 3 by encrypting the entire message plus the hash code.



- When confidentiality is not required, method (2) has an advantage over methods (1) and (3), which encrypts the entire message, in that less computation is required.
- Nevertheless, there has been growing interest in techniques that avoid encryption.
- Several reasons for this interest are pointed out as:
  - Encryption software is relatively slow. Even though the amount of data to be encrypted per message is small, there may be a steady stream of messages into and out of a system.
  - Encryption hardware costs are not negligible. Low-cost chip implementations of DES are available, but the cost adds up if all nodes in a network must have this capability.
  - Encryption hardware is optimized toward large data sizes. For small blocks of data, a high proportion of the time is spent in initialization/invocation overhead.
  - Encryption algorithms may be covered by patents, and there is a cost associated with licensing their use.

Message Authentication Functions

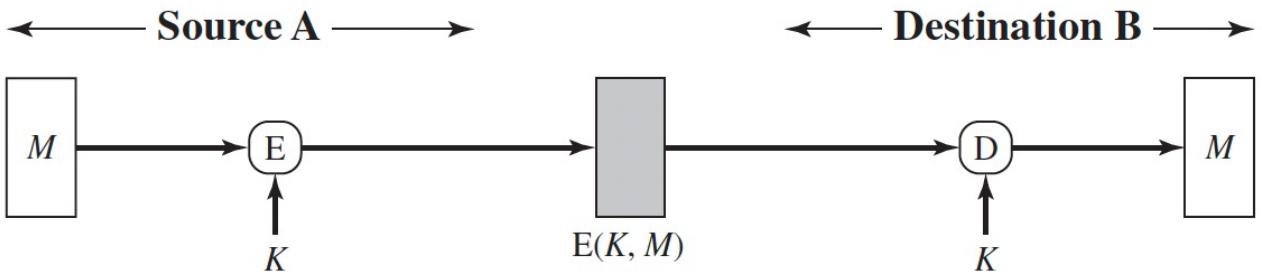
- Any message authentication mechanism has two levels of functionality.
- At the lower level, there must be some sort of function that produces an authenticator: a value to be used to authenticate a message.
- This lower-level function is then used as a primitive in a higher-level authentication protocol that enables a receiver to verify the authenticity of a message.

Message Authentication Functions

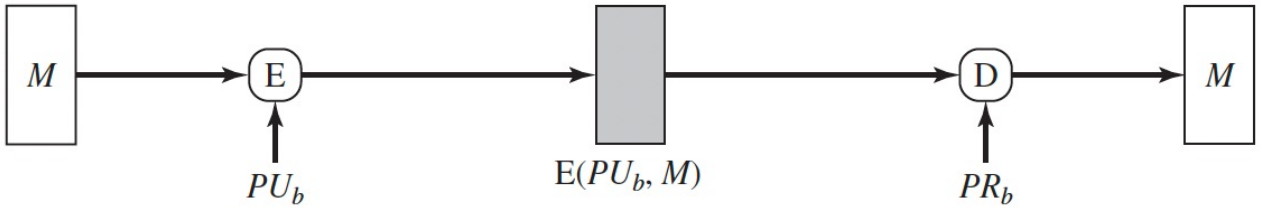
- There are three types of functions that may be used to produce an authenticator.
  - Hash function: A function that maps a message of any length into a fixed- length hash value, which serves as the authenticator.
  - Message encryption: The ciphertext of the entire message serves as its authenticator.
  - Message authentication code (MAC): A function of the message and a secret key that produces a fixed-length value that serves as the authenticator.

Message Encryption

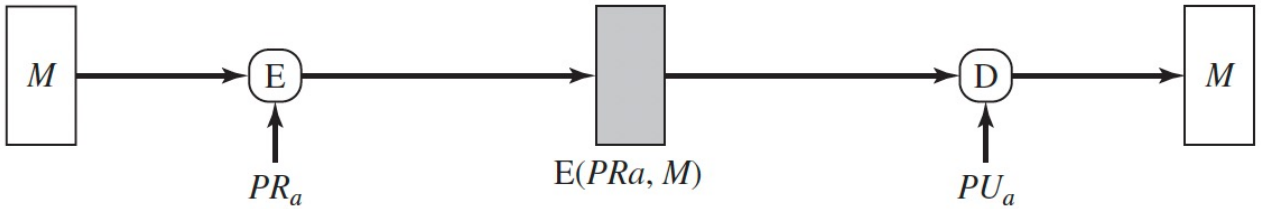
- Message encryption by itself can provide a measure of authentication.
- The analysis differs for symmetric and public-key encryption schemes.
- if symmetric encryption is used then:
  - receiver knows sender must have created it
  - since only sender and receiver know key used
  - know content cannot have been altered
  - if message has suitable structure, redundancy or a checksum to detect any changes



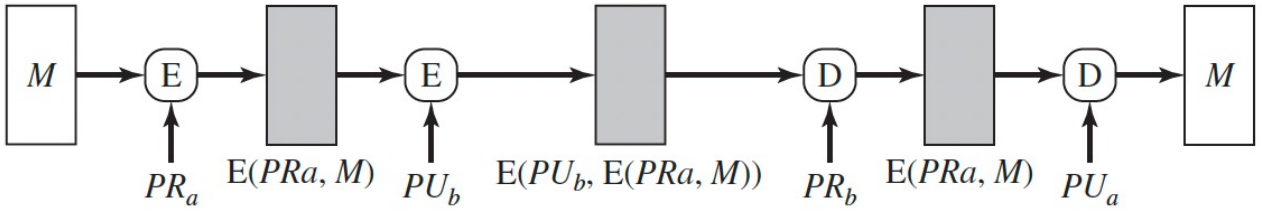
(a) Symmetric encryption: confidentiality and authentication



(b) Public-key encryption: confidentiality



(c) Public-key encryption: authentication and signature



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

Figure 12.1 Basic Uses of Message Encryption

Message Encryption

- if public-key encryption is used:
  - encryption provides no confidence of sender
  - since anyone potentially knows public-key
  - however if
    - sender **signs** message using their private-key
    - then encrypts with recipients public key
    - have both secrecy and authentication
  - again need to recognize corrupted messages
  - but at cost of two public-key uses on message

Message Authentication Code (MAC)

- An authentication technique that involves the use of a secret key to generate a small fixed-size block of data that is appended to the message is known as Message Authentication Code (MAC).
- This technique assumes that two communicating parties, say A and B, share a common secret key K.
- 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

Message Authentication Code (MAC)

- 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.

Message Authentication Code (MAC)

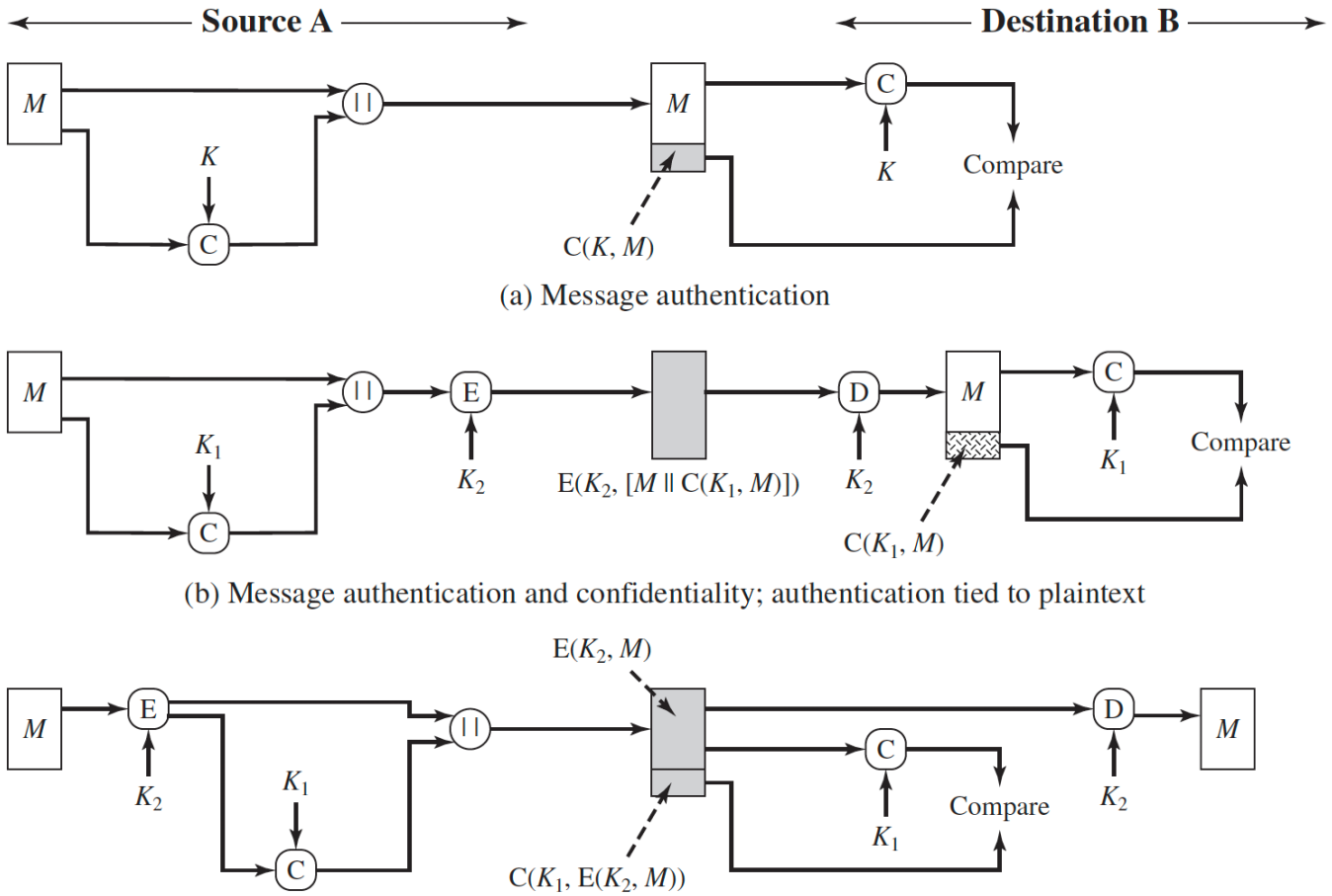


Figure 12.4 Basic Uses of Message Authentication code (MAC)

Message Authentication Code (MAC)

- If we assume that only the receiver and the sender know the identity of the secret key, and if the received MAC matches the calculated MAC, then:
  - The receiver is assured that the message has not been altered. If an attacker alters the message but does not alter the MAC, then the receiver's calculation of the MAC will differ from the received MAC. Because the attacker is assumed not to know the secret key, the attacker cannot alter the MAC to correspond to the alterations in the message.
  - The receiver is assured that the message is from the alleged sender. Because no one else knows the secret key, no one else could prepare a message with a proper MAC.

Message Authentication Code (MAC)

- If the message includes a sequence number, then the receiver can be assured of the proper sequence because an attacker cannot successfully alter the sequence number.

- A MAC function is similar to encryption. One difference is that the MAC algorithm need not be reversible, as it must be for decryption.
- In general, the MAC function is a 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 $N >> 2^n$.
- Furthermore, with a *k*-bit key, there are 2^kpossible keys.

Message Authentication Code (MAC)

- For example, suppose that we are using 100-bit messages and a 10-bit MAC.
- Then, there are total of 2^{100} different messages but only 2^{10} different MACs.
- So, on average, each MAC value is generated by a total of $2^{100}/2^{10} = 2^{90}$ different messages.
- If a 5-bit key is used, then there are $2^5 = 32$ different mappings from the set of messages to the set of MAC values.
- It turns out that, because of the mathematical properties of the authentication function, it is less vulnerable to being broken than encryption.

Message Authentication Code (MAC)

- For example, suppose that we are using 100-bit messages and a 10-bit MAC.
- Then, there are total of 2^{100} different messages but only 2^{10} different MACs.
- So, on average, each MAC value is generated by a total of $2^{100}/2^{10} = 2^{90}$ different messages.
- If a 5-bit key is used, then there are $2^5 = 32$ different mappings from the set of messages to the set of MAC values.
- It turns out that, because of the mathematical properties of the authentication function, it is less vulnerable to being broken than encryption.

Cryptographic Hash Functions

- A cryptographic hash function (CHF) is a hash function that is suitable for use in cryptography.
- It is a mathematical algorithm that maps data of arbitrary size (often called the "message") to a bit string of a fixed size (the "hash value", "hash", or "message digest") and is a one-way function, that is, a function which is practically infeasible to invert.
- Ideally, the only way to find a message that produces a given hash is to attempt a brute-force search of possible inputs to see if they produce a match or use a table of matched hashes.
- Cryptographic hash functions are a basic tool of modern cryptography.

Cryptographic Hash Functions

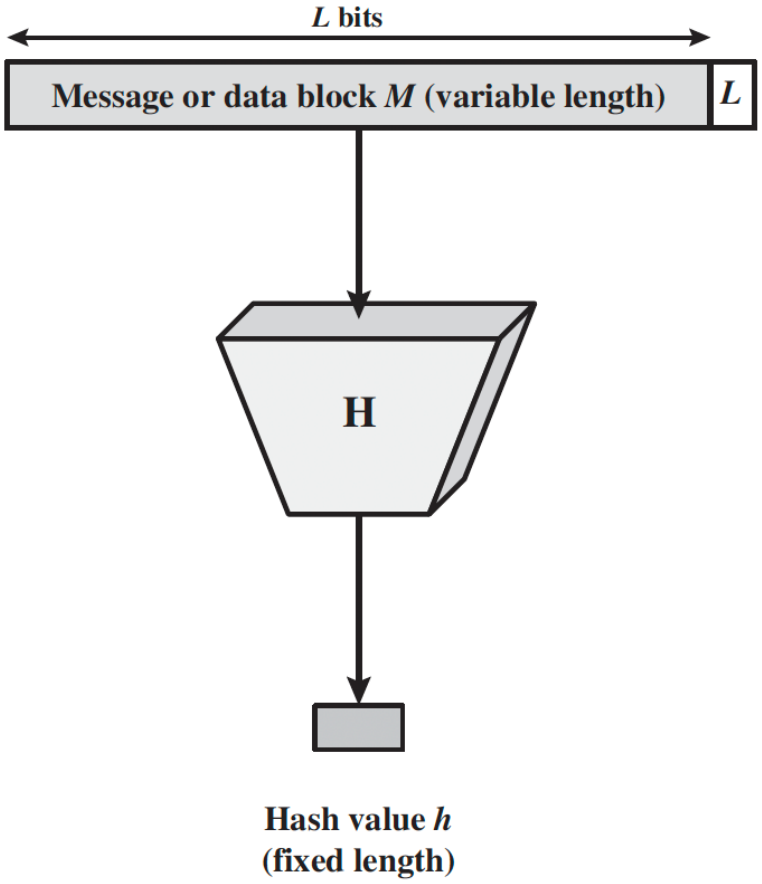


Figure 11.1 Black Diagram of Cryptographic Hash Function; $h = H(M)$

Properties of Hash Functions

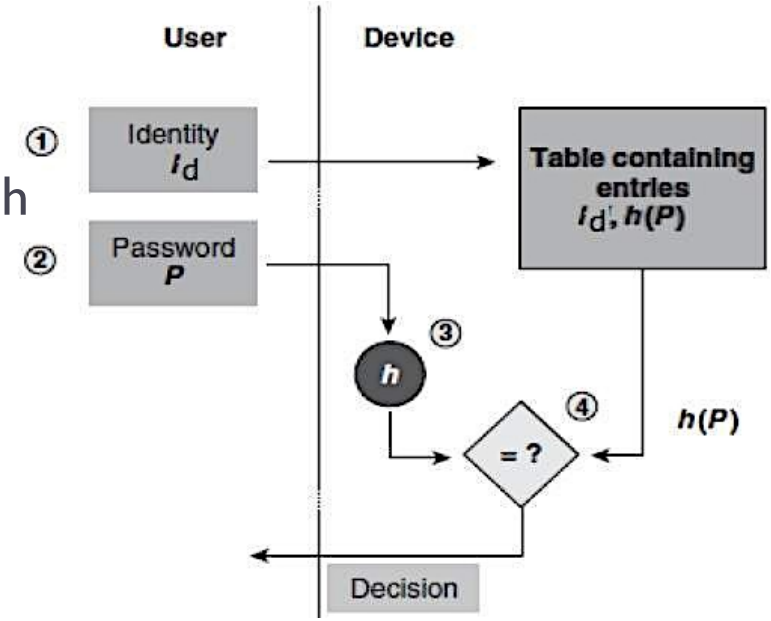
- The ideal cryptographic hash function has the following main properties:
  - It is deterministic, meaning that the same message always results in the same hash
  - It is quick to compute the hash value for any given message
  - It is infeasible to generate a message that yields a given hash value (Pre-Image Resistant)
  - It is infeasible to find two different messages with the same hash value (Collision Resistance)
  - A small change to a message should change the hash value so extensively that the new hash value appears uncorrelated with the old hash value (avalanche effect)

Applications of Hash Functions

- Given that the hash depends on the input to the hash function and will change with the input hash functions are used:
  - to ensure that messages have not been tampered with (message authentication, digital signatures, checksums)
  - Password storage

Password Storage

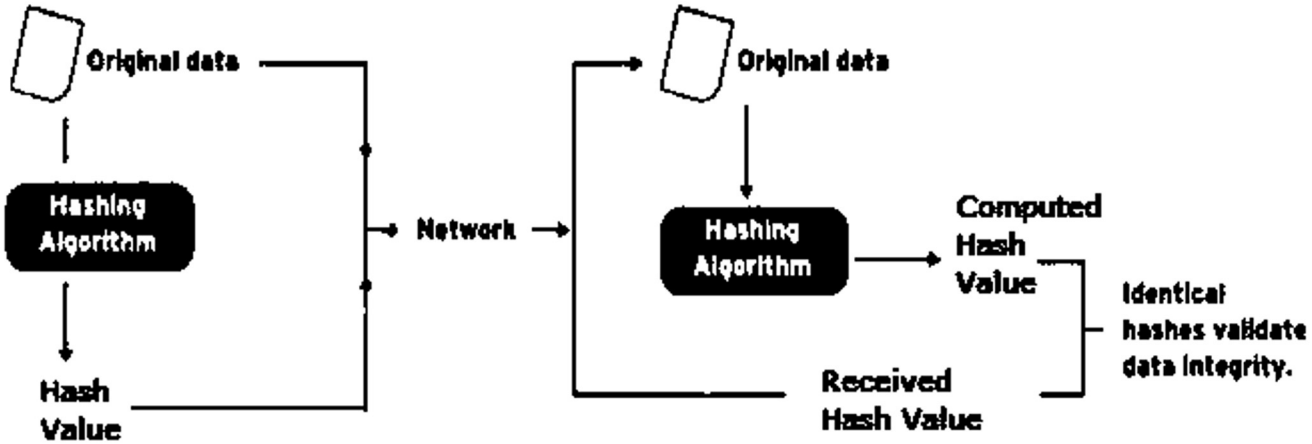
- Hash functions provide protection to password storage.
  - Instead of storing password in clear, mostly all logon processes store the hash values of passwords in the file.
  - The Password file consists of a table of pairs which are in the form (user id, h(P)).
- The process of logon is depicted in the following illustration:



- An intruder can only see the hashes of passwords, even if he accessed the password.
- S/He can neither logon using hash nor can he derive the password from hash value since hash function possesses the property of pre-image resistance.

Data Integrity Check

- Data integrity check is a most common application of the hash functions.
- It is used to generate the checksums on data files.
- This application provides assurance to the user about correctness of the data.
- The process is depicted in the following illustration:



Data Integrity Check

- The integrity check helps the user to detect any changes made to original file. It however, does not provide any assurance about originality.
- The attacker, instead of modifying file data, can change the entire file and compute all together new hash and send to the receiver.
- This integrity check application is useful only if the user is sure about the originality of file.

Digital Signature

- Digital Signature is a process that guarantees that the contents of a message have not been altered in transit.
- In other words, a digital signature is a mathematical technique used to validate the authenticity and integrity of a message, software or digital document.
- As the digital equivalent of a handwritten signature or stamped seal, a digital signature offers far more inherent security, and it is intended to solve the problem of tampering and impersonation in digital communications.

Digital Signature

- Digital signatures can provide the added assurances of evidence of origin, identity and status of an electronic document, transaction or message and can acknowledge informed consent by the signer.
- Digital signatures are based on public key cryptography. Using a public key algorithm, such as RSA, one can generate two keys that are mathematically linked: one private and one public.

Digital Signature

- Suppose that Bob wants to send a message to Alice.
- Although it is not important that the message be kept secret, he wants Alice to be certain that the message is indeed from him.
- For this purpose, Bob uses a secure hash function, such as SHA-512, to generate a hash value for the message.
- That hash value, together with Bob's private key serves as input to a digital signature generation algorithm, which produces a short block that functions as a digital signature.
- Bob sends the message with the signature attached.

Digital Signature

- When Alice receives the message plus signature, she
  1. calculates a hash value for the message;
  2. provides the hash value and Bob's public key as inputs to a digital signature verification algorithm.
- If the algorithm returns the result that the signature is valid, Alice is assured that the message must have been signed by Bob. No one else has Bob's private key and therefore no one else could have created a signature that could be verified for this message with Bob's public key.
- In addition, it is impossible to alter the message without access to Bob's private key, so the message is authenticated both in terms of source and in terms of data integrity.

Digital Signature

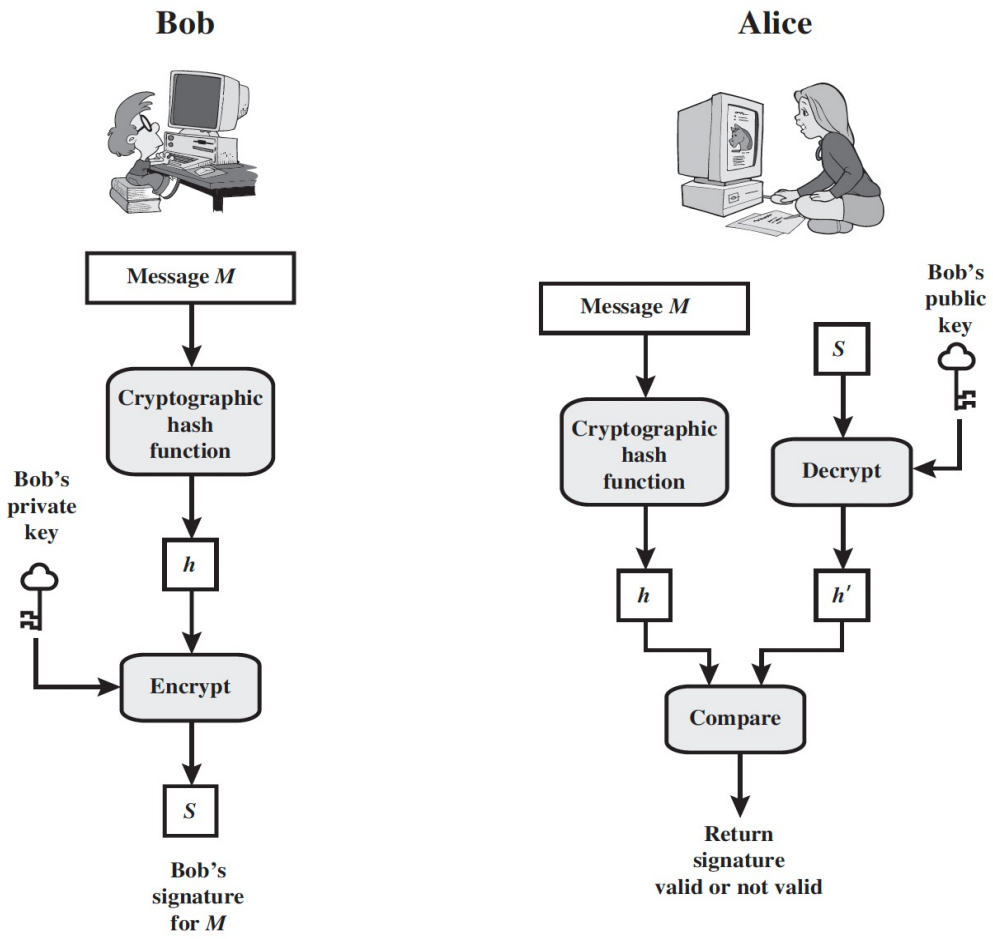


Fig: Simplified Depiction of Essential Elements of Digital Signature Process

Properties of Digital Signature

- Message authentication protects two parties who exchange messages from any third party. However, it does not protect the two parties against each other. Several forms of dispute between the two parties are possible.
- In situations where there is not complete trust between sender and receiver, something more than authentication is needed. The most attractive solution to this problem is the digital signature.
- The digital signature must have the following properties:
  - It must verify the author and the date and time of the signature.
  - It must authenticate the contents at the time of the signature.
  - It must be verifiable by third parties, to resolve disputes.
- Thus, the digital signature function includes the authentication function.

Direct Digital Signature

- The term direct digital signature refers to a digital signature scheme that involves only the communicating parties (source, destination).
- It is assumed that the destination knows the public key of the source.
- Confidentiality can be provided by encrypting the entire message plus signature with a shared secret key (symmetric encryption).
- Note that it is important to perform the signature function first and then an outer confidentiality function.

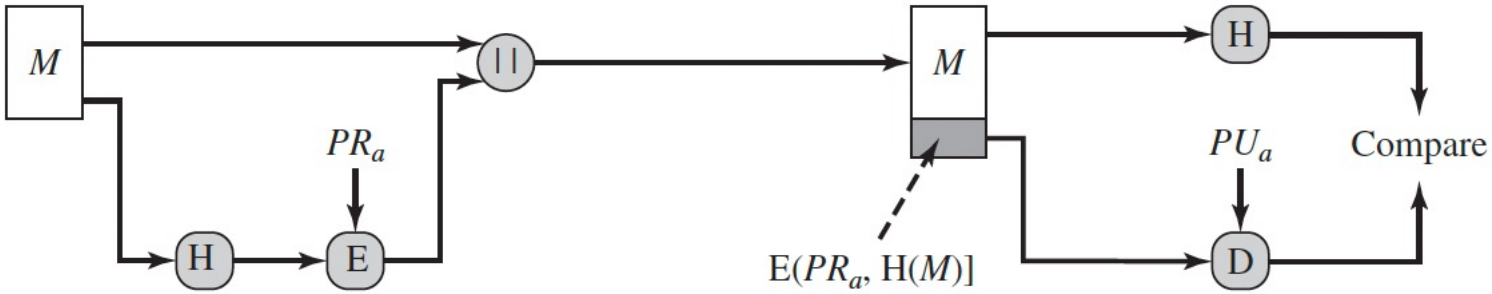
Arbitrated Digital Signature

- An arbitrated digital signature invites a third party into the process called a "trusted arbiter".
- The Arbitrated Digital Signature includes three parties in which one is sender, second is receiver and the third is arbiter who will become the medium for sending and receiving message between them.
- The role of the trusted arbiter is usually twofold: first this independent third party verifies the integrity of the signed message or data. Second, the trusted arbiter dates, or time-stamps, the document, verifying receipt and the passing on of the signed document to its intended final destination.
- The sender cannot deny sending of the message since it has to go through the arbiter before the recipient sees the message.
- The message is less prone to get corrupted because of timestamp being included by default.

Digital Signature: The 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.

Digital Signature: The RSA Approach



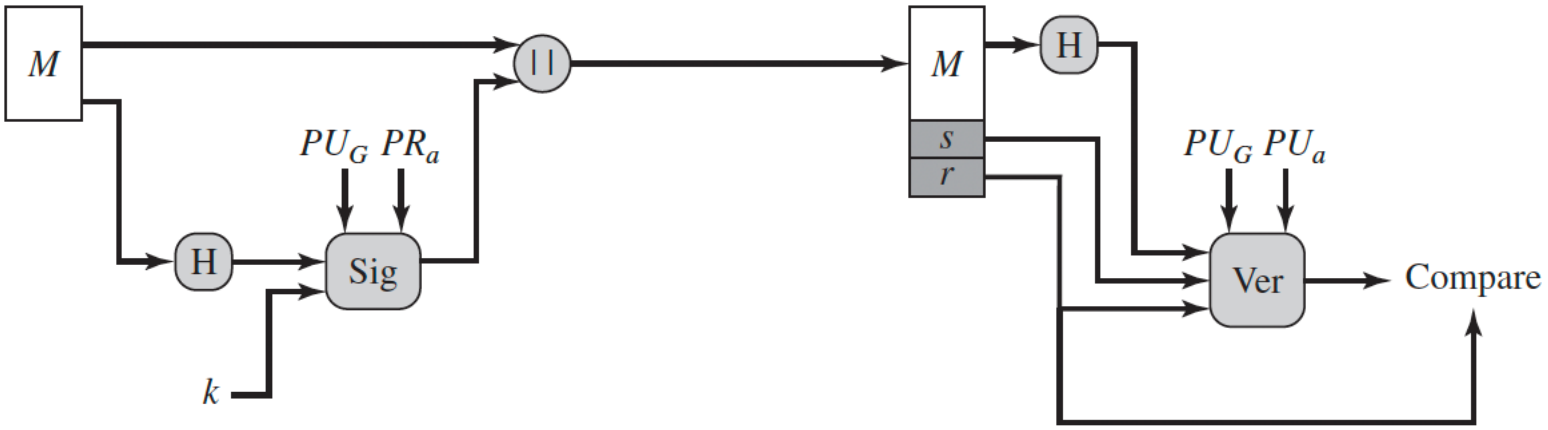
Digital Signature: The DSA Approach

- The DSA uses an algorithm that is designed to provide only the digital signature function. Unlike RSA, it cannot be used for encryption or key exchange, but it is a public-key technique.
- The DSA approach also makes use of a hash function. The hash code is provided as input to a signature function along with a random number k generated for this particular signature.
- The signature function also depends on the sender's private key (PR_a) and a set of parameters known to a group of communicating principals.
- We can consider this set to constitute a global public key (PU_G). The result is a signature consisting of two components, labeled s and r.

Digital Signature: The DSA Approach

- At the receiving end, the hash code of the incoming message is generated. The hash code and the signature are inputs to a verification function.
- The verification function also depends on the global public key as well as the sender's public key (PU_a), which is paired with the sender's private key.
- The output of the verification function is a value that is equal to the signature component r if the signature is valid.
- The signature function is such that only the sender, with knowledge of the private key, could have produced the valid signature.

Digital Signature: The DSA Approach



Digital Signature Algorithm (DSA)

- DSA is based on the difficulty of computing discrete logarithms and is based on schemes originally presented by Elgamal.
- Figure below summarizes the algorithm.

Digital Signature Algorithm (DSA)

Global Public-Key Components

p prime number where $2^{L-1} < p < 2^L$ for $512 \leq L \leq 1024$ and L a multiple of 64; i.e., bit length of between 512 and 1024 bits in increments of 64 bits

q prime divisor of $(p - 1)$, where $2^{159} < q < 2^{160}$; i.e., bit length of 160 bits

g $= h^{(p-1)/q} \bmod p$, where h is any integer with $1 < h < (p - 1)$ such that $h^{(p-1)/q} \bmod p > 1$

User's Private Key

x random or pseudorandom integer with $0 < x < q$

User's Public Key

$y = g^x \bmod p$

User's Per-Message Secret Number

k = random or pseudorandom integer with $0 < k < q$

Signing

$r = (g^k \bmod p) \bmod q$

$s = [k^{-1} (\mathrm{H(M)} + xr)] \bmod q$

Signature = (r, s)

Verifying

$w = (s')^{-1} \bmod q$

$u_1 = [\mathrm{H(M')}w] \bmod q$

$u_2 = (r')w \bmod q$

$v = [(g^{u1} y^{u2}) \bmod p] \bmod q$

TEST: $v = r'$

M = message to be signed

$\mathrm{H}(M)$ = hash of M using SHA-1

M', r', s' = received versions of M, r, s

Digital Signature Algorithm (DSA)

- There are three parameters that are public and can be common to a group of users.
- An N-bit prime number q is chosen.
- Next, a prime number p is selected with a length between 512 and 1024 bits such that q divides (p - 1).
- Finally, g is chosen to be of the form $h^{(p-1)/q}$ mod p, where h is an integer between 1 and (p - 1) with the restriction that g must be greater than 1.
- With these parameters in hand, each user selects a private key and generates a public key.

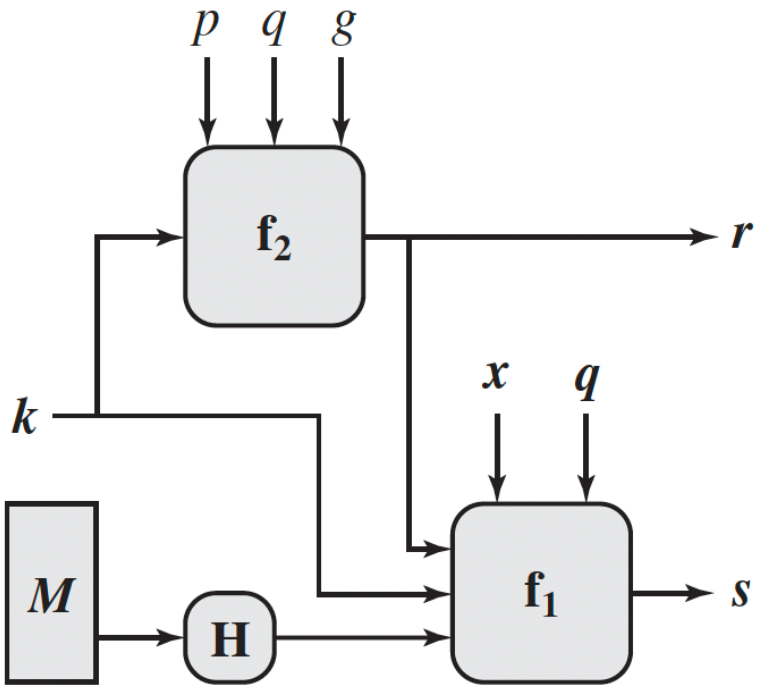
Digital Signature Algorithm (DSA)

- The private key x must be a number from 1 to (q - 1) and should be chosen randomly or pseudorandomly.
- The public key is calculated from the private key as $y = g^x \bmod p$.
- The calculation of y given x is relatively straightforward.
- However, given the public key y, it is believed to be computationally infeasible to determine x, which is the discrete logarithm of y to the base g, mod p.

Digital Signature Algorithm (DSA)

- The signature of a message M consists of the pair of numbers r and s, which are functions of the public key components (p, q, g), the user's private key (x), the hash code of the message H(M), and an additional integer k that should be generated randomly or pseudo-randomly and be unique for each signing.
- Let M, r′, and s′ be the received versions of M, r, and s, respectively.
- Verification is performed using the formulas shown in slide 46.
- The receiver generates a quantity v that is a function of the public key components, the sender's public key, the hash code of the incoming message, and the received versions of r and s.
- If this quantity matches the r component of the signature, then the signature is validated.
- Figure below depicts the functions of signing and verifying.

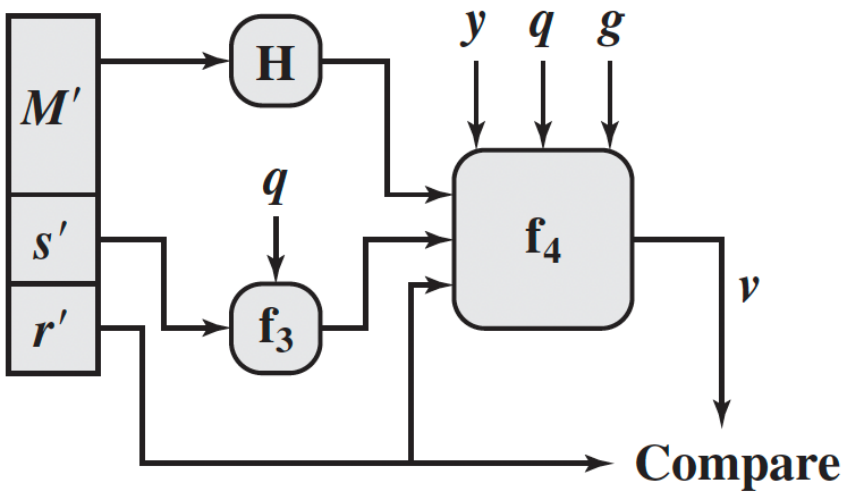
Digital Signature Algorithm (DSA)



$s = f_1(H(M), k, x, r, q) = (k^{-1} (H(M) + xr)) \bmod q$

$r = f_2(k, p, q, g) = (g^k \bmod p) \bmod q$

(a) Signing



$w = f_3(s', q) = (s')^{-1} \bmod q$

$v = f_4(y, q, g, H(M'), w, r')$

$= ((g^{(H(M')w) \bmod q} y^{r'w \bmod q}) \bmod p) \bmod q$

(b) Verifying

Message Digest: MD4

- MD4 is a message digest algorithm (the fourth in a series) designed by Professor Ronald Rivest of MIT in 1990.
- It implements a cryptographic hash function for use in message integrity checks.
- The digest length is 128 bits. The block size is 256 bits.
- The algorithm has influenced later designs, such as the MD5, SHA algorithms.
- MD4 is also used to compute NT-hash password digests on Microsoft Windows NT, XP and Vista.

Message Digest: MD4

- Let the symbol "+" denote addition of words (i.e., modulo-2^{32} addition).
- Let (X <<< s) denote the 32-bit value obtained by circularly shifting (rotating) X left by s bit positions.
- Let ¬X denote the bit-wise complement of X, and let X **V** Y denote the bit-wise OR of X and Y.
- Let $X \oplus Y$ denote the bit-wise XOR of X and Y, and let XY denote the bit-wise AND of X and Y.
- We begin by supposing that we have a b-bit message as input, and that we wish to find its message digest.
- Here b is an arbitrary nonnegative integer; b may be zero, it need not be a multiple of 8, and it may be arbitrarily large.
- We imagine the bits of the message written down as follows:

$$m_0, m_1, \ldots m_{b-1}.$$

MD4 Algorithm

Step 1. Append padding bits

- The message is padded (extended) so that its length (in bits) is congruent to 448, modulo 512.
- Padding is performed as follows: a single "1" bit is appended to the message, and then "0" bits are appended so that the length in bits of the padded message becomes congruent to 448, modulo 512.
- In all, at least one bit and at most 512 bits are appended.

MD4 Algorithm

Step 2. Append Length

- A 64-bit representation of b (the length of the message before the padding bits were added) is appended to the result of the previous step.
- This point the resulting message (after padding with bits and with b) has a length that is an exact multiple of 512 bits.

MD4 Algorithm

Step 3. Initialize MD Buffer

- A four-word buffer (A,B,C,D) is used to compute the message digest.
- Here each of A, B, C, D is a 32-bit register.
- These registers are initialized to the following values in hexadecimal, low-order bytes first):
  - word A: 01 23 45 67
  - word B: 89 ab cd ef
  - word C: fe dc ba 98
  - word D: 76 54 32 10

MD4 Algorithm

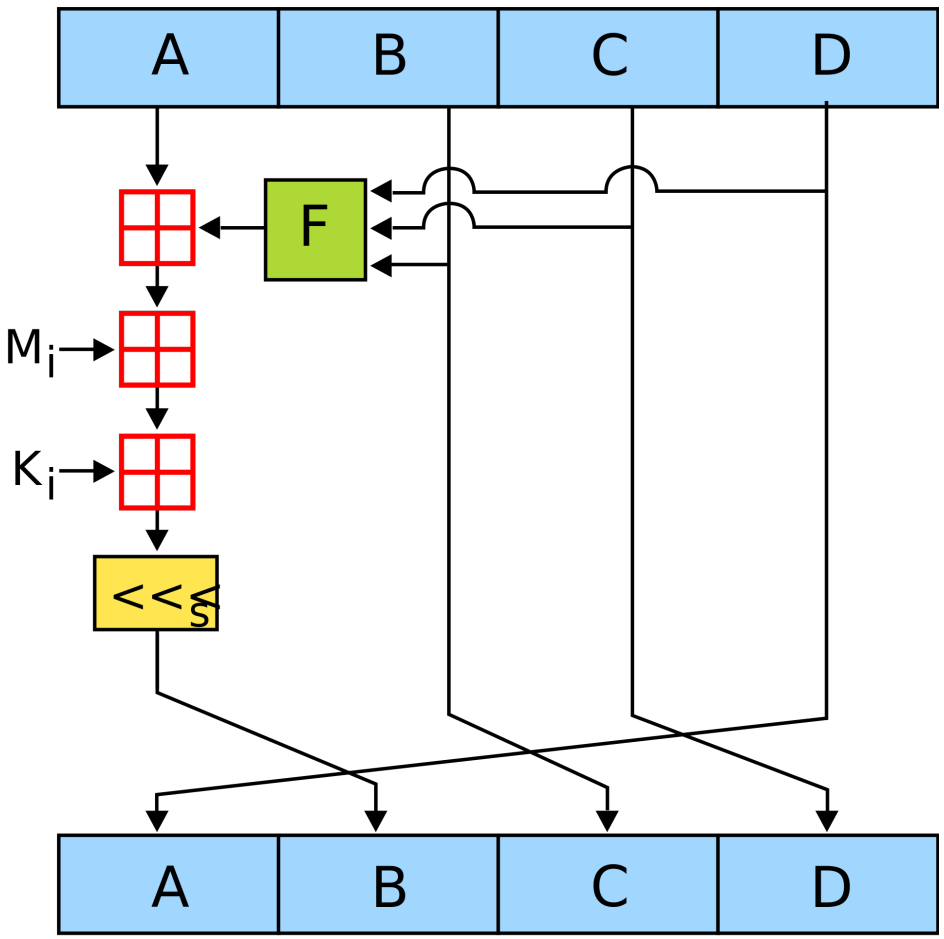
Step 4. Process Message in 16-Word Blocks

- We first define three auxiliary functions that each take as input three 32-bit words and produce as output one 32-bit word.
  - $F(X,Y,Z) = XY \vee (\neg X)\, Z$
  - $G(X,Y,Z) = XY \vee XZ \vee YZ$
  - $H(X,Y,Z) = X \oplus Y \oplus Z$

Step 5. Output

- The message digest produced as output is A, B, C, D. That is, we begin with the low-order byte of A, and end with the high-order byte of D.

MD4 Algorithm



- One MD4 operation. MD4 consists of 48 of these operations, grouped in three rounds of 16 operations.
- F is a nonlinear function; one function is used in each round.
- M_i denotes a 32-bit block of the message input, and K_i denotes a 32-bit constant, different for each round.
- $<<<_s$ denotes a left bit rotation by s places; s varies for each operation.
- $\boxplus$ denotes addition modulo 2^{32}.

MD4 Security

- The security of MD4 has been severely compromised. The first full collision attack against MD4 was published in 1995 and several newer attacks have been published since then.
- As of 2007, an attack can generate collisions in less than 2 MD4 hash operations.
- In 2008, the preimage resistance of MD4 was also broken by Gaëtan Leurent, with a 2^{102} attack.

MD5 Algorithm

- The MD5 message-digest algorithm is a widely used hash function producing a 128-bit hash value.
- Although MD5 was initially designed to be used as a cryptographic hash function, it has been found to suffer from extensive vulnerabilities.
- It can still be used as a checksum to verify data integrity, but only against unintentional corruption.
- The structure used in MD5 is Merkle–Damgård construction.

MD5 Algorithm

- The MD5 message digest hashing algorithm processes data in 512-bit blocks, broken down into 16 words composed of 32 bits each.
- The output from MD5 is a 128-bit message digest value.
- MD5 was designed by Ronald Rivest in 1991 to replace an earlier hash function MD4, and was specified in 1992 as RFC 1321.

Working of MD5 Algorithm

- We begin by supposing that we have a b-bit message as input, and that we wish to find its message digest.
- Here b is an arbitrary nonnegative integer; b may be zero, it need not be a multiple of eight, and it may be arbitrarily large.
- We imagine the bits of the message written down as follows:

  m_0, m_1, ... m_{b-1}

- The following five steps are performed to compute the message digest of the message:

MD5 Algorithm

Step 1. Append Padding Bits

- The message is "padded" (extended) so that its length (in bits) is congruent to 448, modulo 512. That is, the message is extended so that it is just 64 bits shy of being a multiple of 512 bits long..
- Padding is performed as follows: a single "1" bit is appended to the message, and then "0" bits are appended so that the length in bits of the padded message becomes congruent to 448, modulo 512. In all, at least one bit and at most 512 bits are appended.

MD5 Algorithm

Step 2. Append Length

- A 64-bit representation of b (the length of the message before the padding bits were added) is appended to the result of the previous step.
- At this point the resulting message (after padding with bits and with b) has a length that is an exact multiple of 512 bits.

MD5 Algorithm

Step 3. Initialize MD Buffer

- A four-word buffer (A, B, C, D) is used to compute the message digest.
- Here each of A, B, C, D is a 32-bit register.
- These registers are initialized to the following values in hexadecimal, low-order bytes first):
  - word A: 01 23 45 67
  - word B: 89 ab cd ef
  - word C: fe dc ba 98
  - word D: 76 54 32 10

MD5 Algorithm

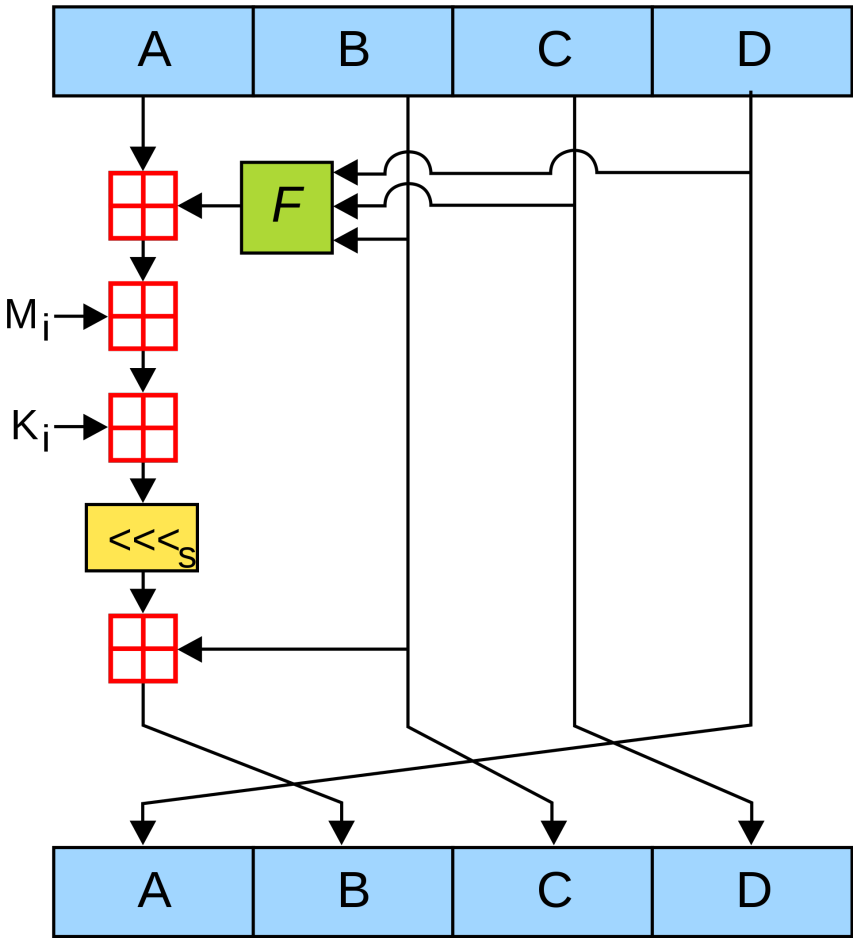
Step 4. Process Message in 16-Word Blocks

- We first define four auxiliary functions that each take as input three 32-bit words and produce as output one 32-bit word.
  - $F(X,Y,Z) = XY \vee (\neg X) Z$
  - $G(X,Y,Z) = XZ \vee Y (\neg Z)$
  - $H(X,Y,Z) = X \oplus Y \oplus Z$
  - $I(X,Y,Z) = Y \oplus (X \vee (\neg Z))$

Step 5. Output

- The message digest produced as output is A, B, C, D. That is, we begin with the low-order byte of A, and end with the high-order byte of D.

MD5 Algorithm



- One MD5 operation. MD5 consists of 64 of these operations, grouped in four rounds of 16 operations.
- *F* is a nonlinear function; one function is used in each round.
- M_i denotes a 32-bit block of the message input, and K_i denotes a 32-bit constant, different for each operation.
- $<<<_s$ denotes a left bit rotation by s places; s varies for each operation.
- $\boxplus$ denotes addition modulo 2^{32}.

MD5 Security

- Unfortunately, MD5 has been cryptographically broken and considered insecure.
- For this reason, it should not be used for anything.
- Instead, developers should switch to the Secure Hash Algorithm or a Symmetric Cryptographic Algorithm.
- With current GPUs and hash cracking tools, using MD5 is barely better than using nothing at all.

Secure Hash Algorithm (SHA)

- The Secure Hash Algorithms are a family of cryptographic hash functions published by the National Institute of Standards and Technology (NIST) as a U.S. Federal Information Processing Standard (FIPS), including:
- SHA-0: A change applied to the original version of the 160-bit hash function published in 1993 under the name "SHA". It was withdrawn shortly after publication due to an undisclosed "significant flaw" and replaced by the slightly revised version SHA-1.

Secure Hash Algorithm (SHA)

- SHA-1: A 160-bit hash function which resembles the earlier MD5 algorithm. This was designed by the National Security Agency (NSA) to be part of the Digital Signature Algorithm. Cryptographic weaknesses were discovered in SHA-1, and the standard was no longer approved for most cryptographic uses after 2010.
- SHA-2: A family of two similar hash functions, with different block sizes, known as SHA-256 and SHA-512. There are also truncated versions of each standard, known as SHA-224, SHA-384, SHA-512/224 and SHA-512/256. These were also designed by the NSA.

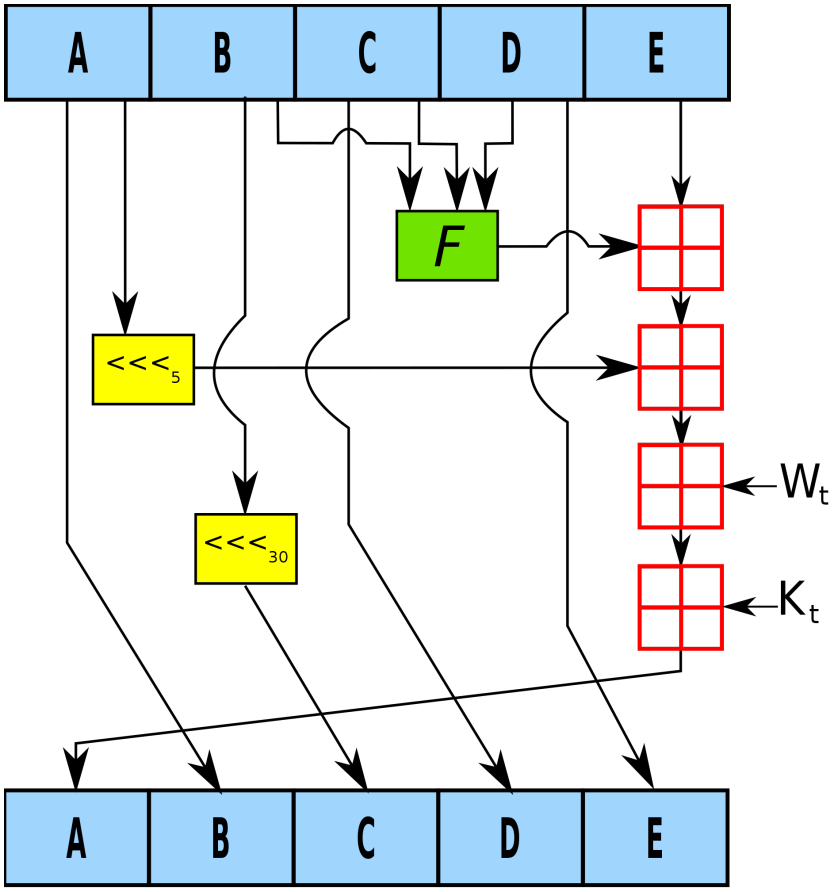
Secure Hash Algorithm (SHA)

- SHA-3: A hash function formerly called Keccak, chosen in 2012 after a public competition among non- NSA designers. It supports the same hash lengths as SHA-2, and its internal structure differs significantly from the rest of the SHA family.

SHA-1

- In cryptography, SHA-1 (Secure Hash Algorithm 1) is a cryptographic hash function which takes an input and produces a 160-bit (20-byte) hash value known as a message digest – typically rendered as a hexadecimal number, 40 digits long.
- Since 2005 SHA-1 has not been considered secure against well-funded opponents.
- NIST formally deprecated use of SHA-1 in 2011 and disallowed its use for digital signatures in 2013.
- It is recommended to remove SHA-1 from products as soon as possible and use instead SHA-256 or SHA-3.
- Replacing SHA-1 is urgent where it's used for signatures.
- All major web browser vendors ceased acceptance of SHA-1 SSL certificates in 2017.

SHA-1



- One iteration within the SHA-1 compression function: A, B, C, D and E are 32-bit words of the state;
- F is a nonlinear function that varies;
- $<<<_n$ denotes a left bit rotation by n places; n varies for each operation;
- W_t is the expanded message word of round t;
- K_t is the round constant of round t;
- $\boxplus$ denotes addition modulo 2^{32}.

SHA-1

- SHA-1 uses 80 rounds of cryptographic operations to encrypt and secure a data object.
- Some of the protocols that use SHA-1 include:
  - Transport Layer Security (TLS)
  - Secure Sockets Layer (SSL)
  - Pretty Good Privacy (PGP)
  - Secure Shell (SSH)
  - Secure/Multipurpose Internet Mail Extensions (S/MIME)
  - Internet Protocol Security (IPSec)

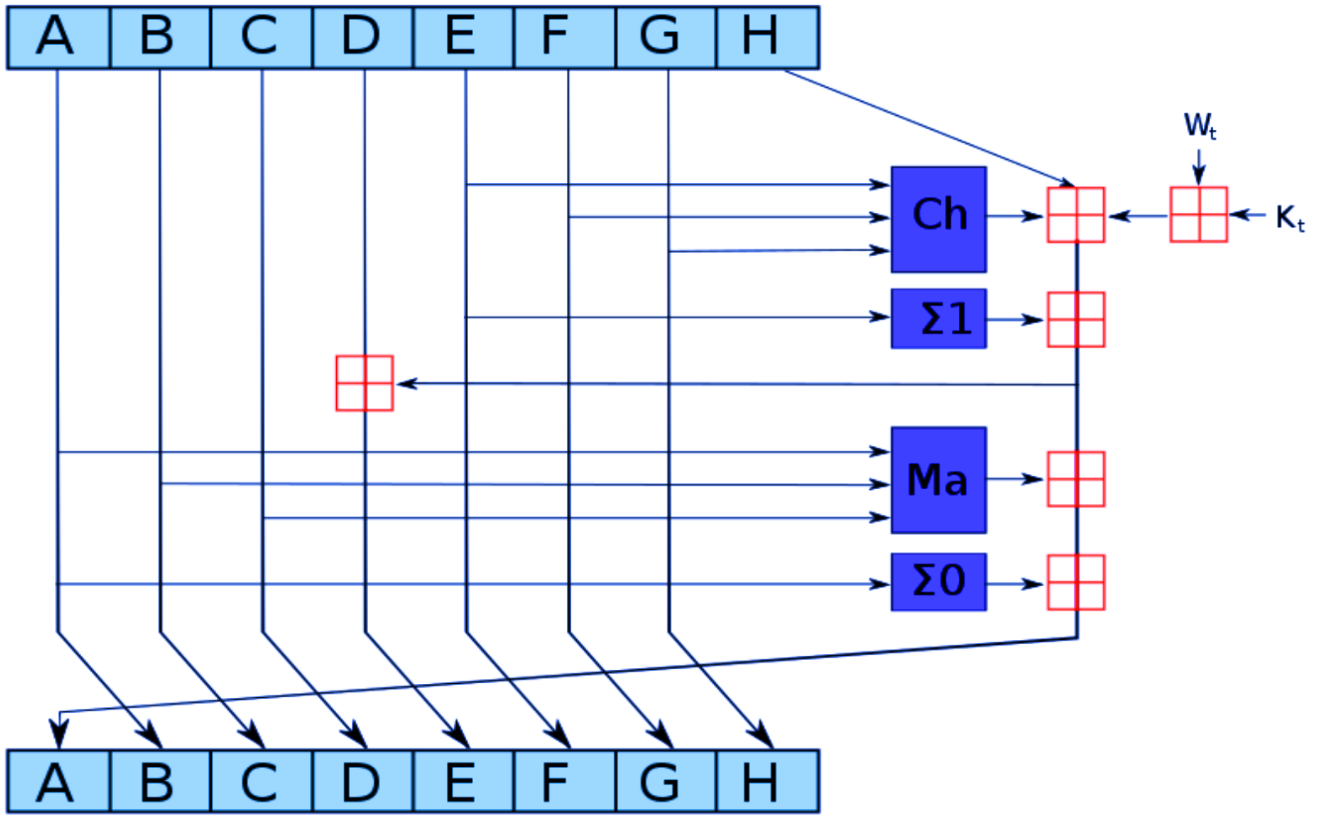
SHA-2

- SHA-2 is a family of hashing algorithms to replace the SHA-1 algorithm. SHA-2 features a higher level of security than its predecessor.
- It was designed through The National Institute of Standards and Technology (NIST) and the National Security Agency (NSA).
- One of the major benefits of using SHA-2 is that it addresses some weaknesses in the SHA-1 hashing algorithm.
- One of the drawbacks with SHA-2 is that there are some older applications and operating systems that do not support it. Compatibility problems are the main reason why SHA-2 algorithms have not been adopted more rapidly.
- The SHA-2 family consists of six hash functions with digests (hash values) that are 224, 256, 384 or 512 bits: SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256.

SHA-2

- SHA-256 and SHA-512 are novel hash functions computed with eight 32-bit and 64-bit words, respectively.
- They use different shift amounts and additive constants, but their structures are otherwise virtually identical, differing only in the number of rounds (64 and 80 respectively).
- Applications of SHA-2:
  - The SHA-2 hash function is implemented in some widely used security applications and protocols, including TLS and SSL, PGP, SSH, S/MIME, and IPsec.
  - Used for verifying the transactions. E.g. In cryptocurrency like Bitcoin

SHA-2



One iteration in a SHA-2 family compression function. The blue components perform the following operations:

$$\mathrm{Ch}(E,F,G) = (E \wedge F) \oplus (\neg E \wedge G)$$
$$\mathrm{Ma}(A,B,C) = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C)$$
$$\Sigma_0(A) = (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22)$$
$$\Sigma_1(E) = (E \ggg 6) \oplus (E \ggg 11) \oplus (E \ggg 25)$$

The bitwise rotation uses different constants for SHA-512. The given numbers are for SHA-256.

The red ⊞ is addition modulo 2^{32} for SHA-256, or 2^{64} for SHA-512.

SHA

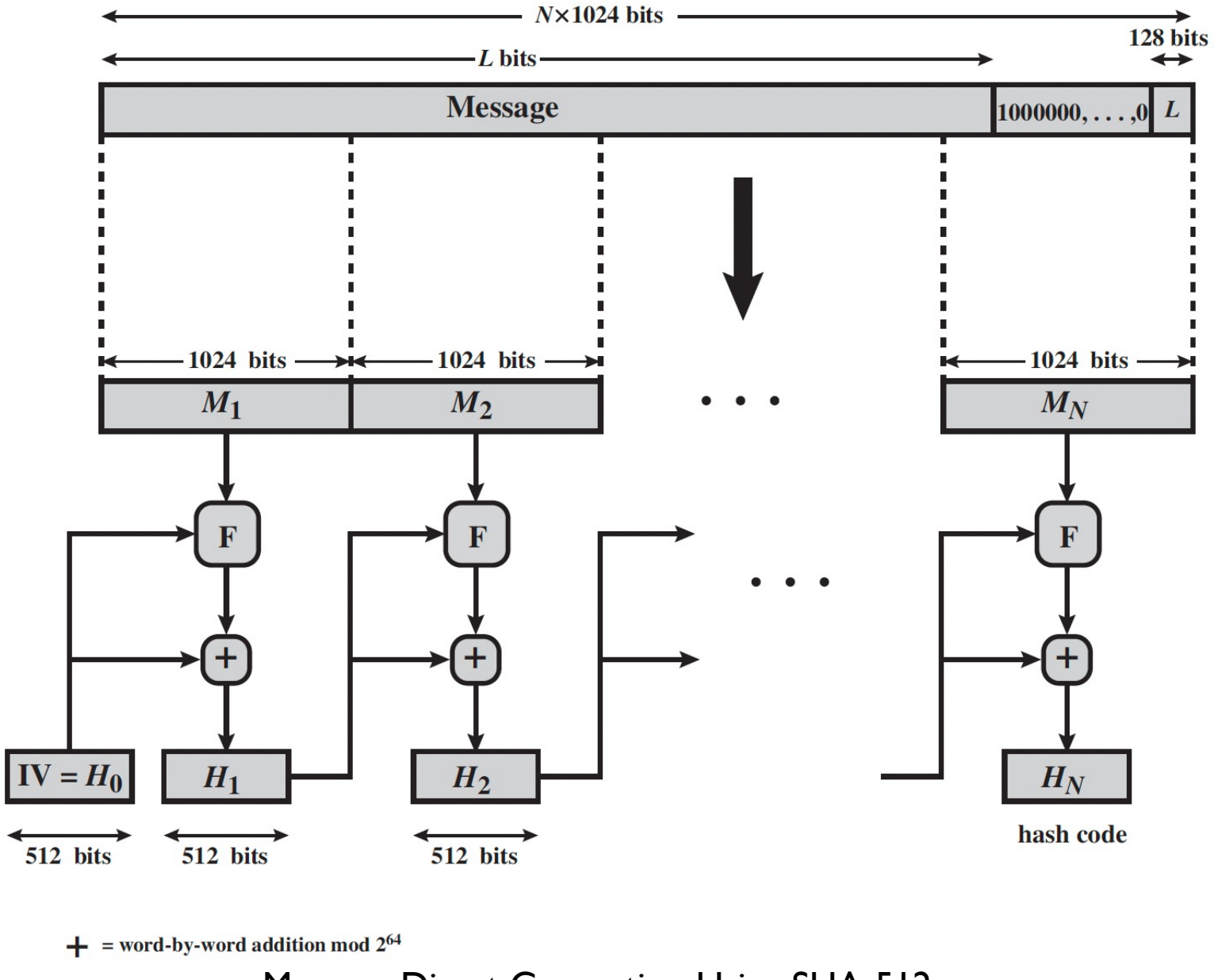
Table 11.3 Comparison of SHA Parameters

	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.

SHA-512

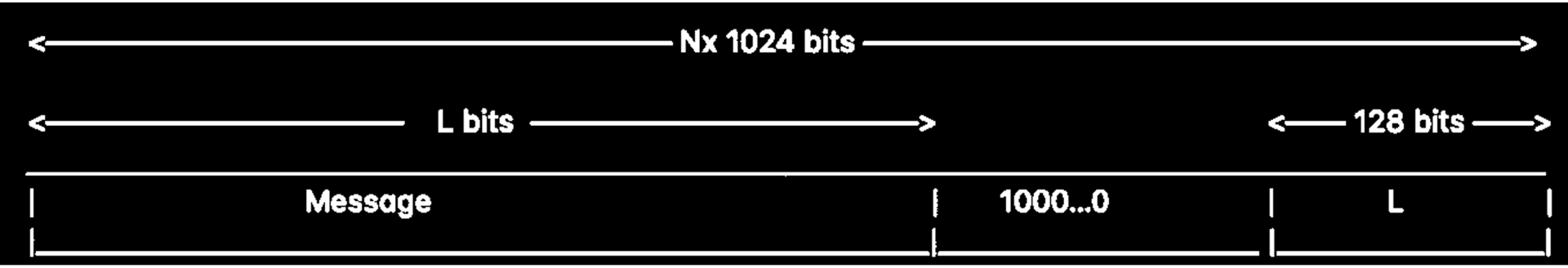
- The algorithm takes as input a message with a maximum length of less than bits 2^{128} and produces as output a 512-bit message digest.
- The input is processed in 1024-bit blocks.
- SHA-512 algorithm works in the following steps:
  I. Append padding bits
  II. Append length
  III. Initial hash buffer
  IV. Process message in 1024-bit (128-word) blocks
  V. Output



Message Digest Generation Using SHA-512

Step 1. Append Padding Bits

- The message is padded so that its length is congruent to 896 modulo 1024 (which is exactly 128 bits less than an exact multiple of 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.



Step 2: Append length

- A block of 128 bits is appended to the message.
- This block is treated as an unsigned 128-bit integer (most significant byte first) and contains the length of the original message (before the padding).
- The outcome of the first two steps yields a message that is an integer multiple of 1024 bits in length.
- The expanded message is represented as the sequence of 1024-bit blocks $M_1, M_2, \ldots\ldots, M_N$, so that the total length of the expanded message is N *X* 1024 bits.

Step 3: Initialise 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).
- These 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

Step 3: Initialise Hash Buffer

- These values are stored in big-endian format, which is the most significant byte of a word in the low-address (leftmost) byte position.
- These words were obtained by taking the first sixty-four bits of the fractional parts of the square roots of the first eight prime numbers.
- There are other default values that need to be initialized as well. These are the value for the 'K_t' variable which will be used.

```
k[0..79] := [ 0x428a2f98d728ae22, 0x7137449123ef65cd, 0xb5c0fbcfec4d3b2f, 0xe9b5dba58189dbbc, 0x3956c25bf348b538,
              0x59f111f1b605d019, 0x923f82a4af194f9b, 0xab1c5ed5da6d8118, 0xd807aa98a3030242, 0x12835b0145706fbe,
              0x243185be4ee4b28c, 0x550c7dc3d5ffb4e2, 0x72be5d74f27b896f, 0x80deb1fe3b1696b1, 0x9bdc06a725c71235,
              0xc19bf174cf692694, 0xe49b69c19ef14ad2, 0xefbe4786384f25e3, 0x0fc19dc68b8cd5b5, 0x240ca1cc77ac9c65,
              0x2de92c6f592b0275, 0x4a7484aa6ea6e483, 0x5cb0a9dcbd41fbd4, 0x76f988da831153b5, 0x983e5152ee66dfab,
              0xa831c66d2db43210, 0xb00327c898fb213f, 0xbf597fc7beef0ee4, 0xc6e00bf33da88fc2, 0xd5a79147930aa725,
              0x06ca6351e003826f, 0x142929670a0e6e70, 0x27b70a8546d22ffc, 0x2e1b21385c26c926, 0x4d2c6dfc5ac42aed,
              0x53380d139d95b3df, 0x650a73548baf63de, 0x766a0abb3c77b2a8, 0x81c2c92e47edaee6, 0x92722c851482353b,
              0xa2bfe8a14cf10364, 0xa81a664bbc423001, 0xc24b8b70d0f89791, 0xc76c51a30654be30, 0xd192e819d6ef5218,
              0xd69906245565a910, 0xf40e35855771202a, 0x106aa07032bbd1b8, 0x19a4c116b8d2d0c8, 0x1e376c085141ab53,
              0x2748774cdf8eeb99, 0x34b0bcb5e19b48a8, 0x391c0cb3c5c95a63, 0x4ed8aa4ae3418acb, 0x5b9cca4f7763e373,
              0x682e6ff3d6b2b8a3, 0x748f82ee5defb2fc, 0x78a5636f43172f60, 0x84c87814a1f0ab72, 0x8cc702081a6439ec,
              0x90befffa23631e28, 0xa4506cebde82bde9, 0xbef9a3f7b2c67915, 0xc67178f2e372532b, 0xca273eceea26619c,
              0xd186b8c721c0c207, 0xeada7dd6cde0eb1e, 0xf57d4f7fee6ed178, 0x06f067aa72176fba, 0x0a637dc5a2c898a6,
              0x113f9804bef90dae, 0x1b710b35131c471b, 0x28db77f523047d84, 0x32caab7b40c72493, 0x3c9ebe0a15c9bebc,
              0x431d67c49c100d4c, 0x4cc5d4becb3e42b6, 0x597f299cfc657e2a, 0x5fcb6fab3ad6faec, 0x6c44198c4a475817]
```

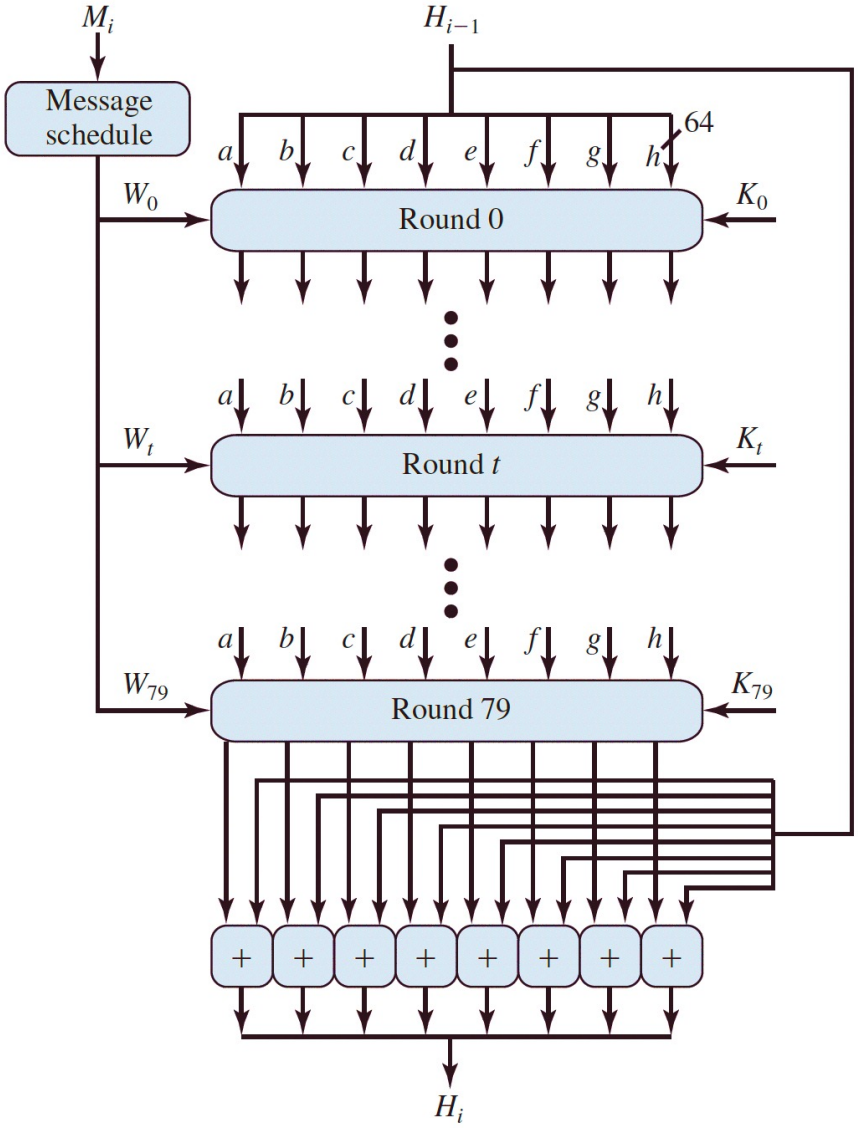
Step 4: Process message in 1024-bit (128-word) blocks

- The SHA-512 Compression Function is the heart of the algorithm.
- It processes the message in 1024-bit (128-word) blocks, using a module that consists of 80 rounds, labeled F in the figure shown in slide 69.
- Each round takes as input the 512-bit buffer value, and updates the contents of the buffer.
- Each round t makes use of a 64-bit value W_t derived using a message schedule from the current 1024-bit block being processed.

Step 4: Process message in 1024-bit (128-word) blocks

- Each round also makes use of an additive constant K_t, based on the fractional parts of the cube roots of the first eighty prime numbers.
- The output of the 80^{th} round is added to the input to the first round (H_{i-1}) to produce the final hash value (H_i) for this message block, which forms the input to the next iteration of this compression function.
- 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}.

Step 4: Process message in 1024-bit (128-word) blocks



SHA-512 Processing of a Single 1024-Bit Block

Step 5: Output

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

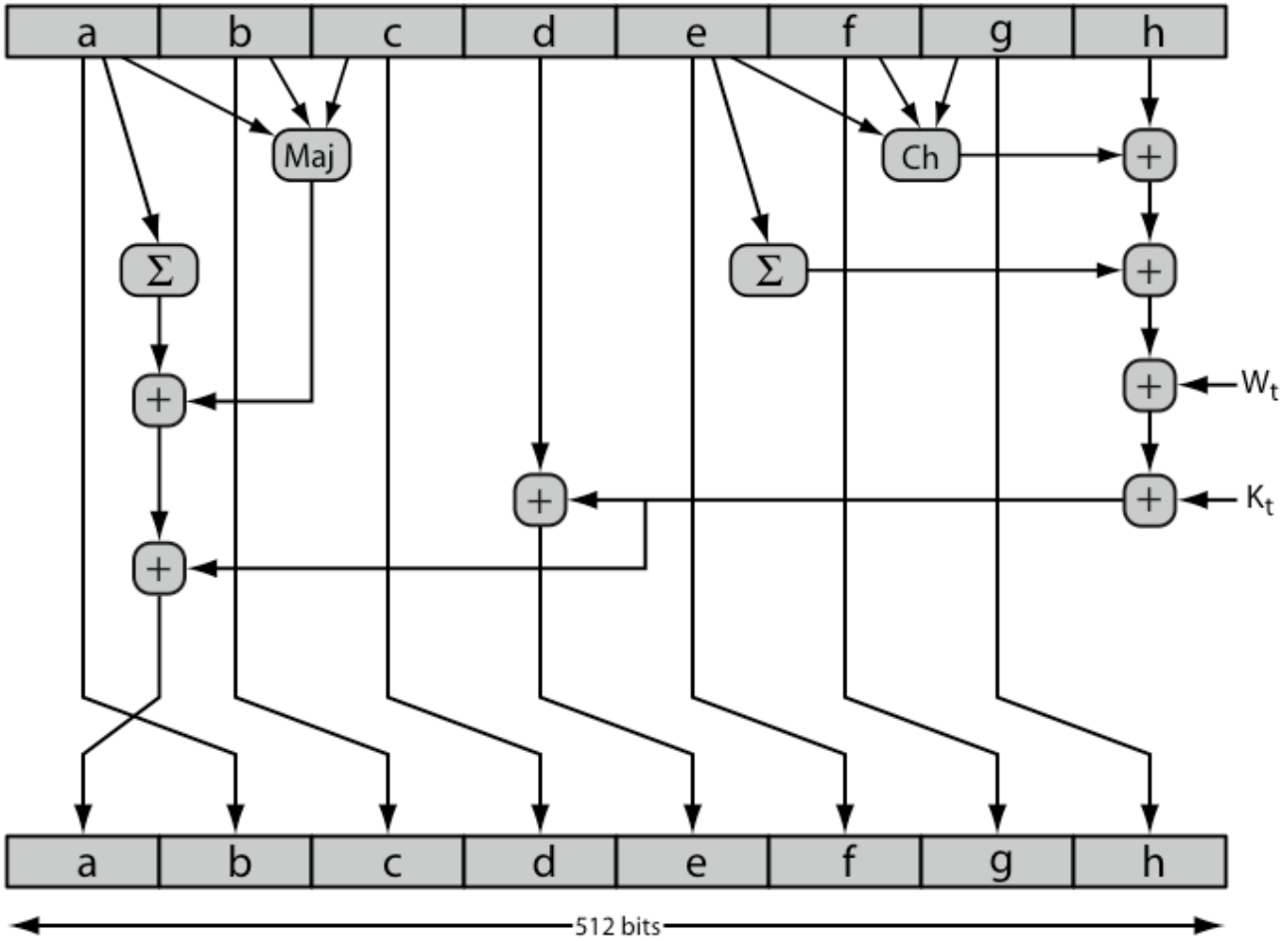
We can summarize the behavior of SHA-512 as follows:

$$H_0 = \text{IV}$$
$$H_i = \text{SUM}_{64}(H_{i-1}, \text{abcdefgh}_i)$$
$$MD = H_N$$

where

IV	=	initial value of the abcdefgh buffer, defined in step 3
abcdefgh_i	=	the output of the last round of processing of the *i*th message block
N	=	the number of blocks in the message (including padding and length fields)
SUM_{64}	=	addition modulo 2^{64} performed separately on each word of the pair of inputs
MD	=	final message digest value

SHA-512 Round Function

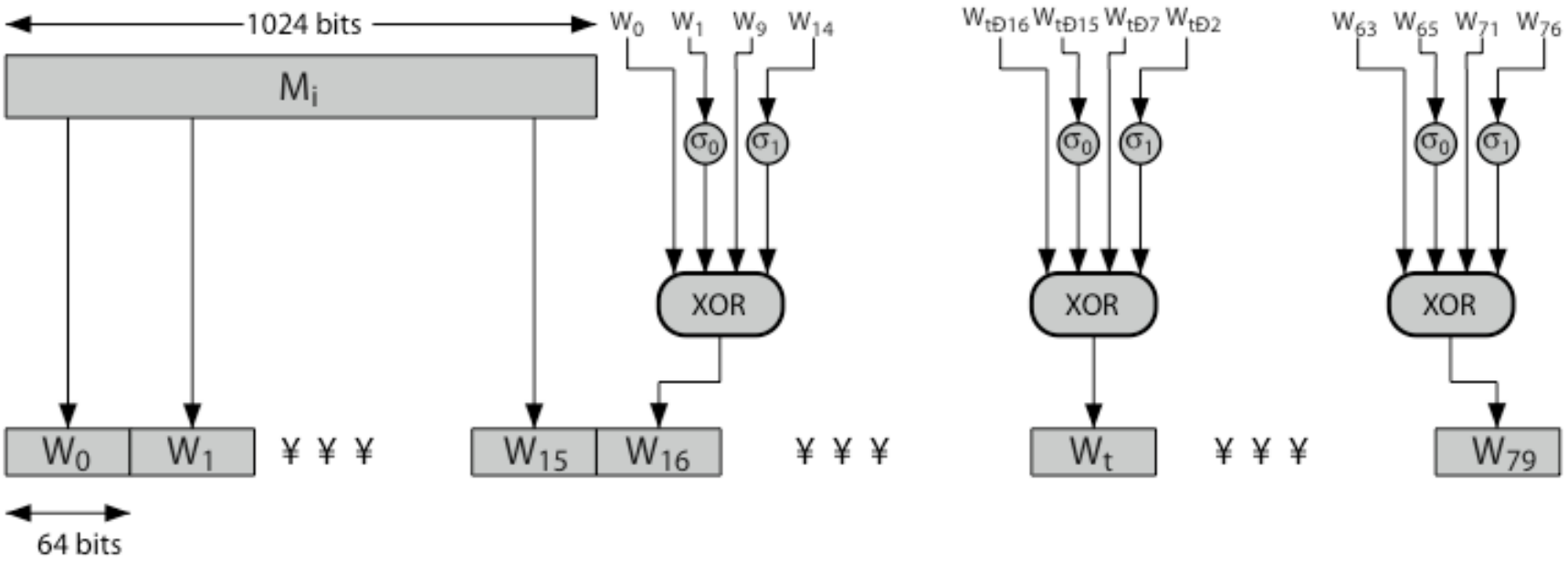


SHA-512 Round Function

- The structure of each of the 80 rounds is shown in the previous slide.
- Each 64-bit word shuffled along one place, and in some cases manipulated using a series of simple logical functions (ANDs, NOTs, ORs, XORs, ROTates), in order to provide the avalanche & completeness properties of the hash function.
- The elements are:

$$\begin{aligned}
\text{Maj}(a, b, c) &= (a \text{ AND } b) \oplus (a \text{ AND } c) \oplus (b \text{ AND } c) \\
&\quad \textit{the function is true only of the majority (two or three) of the arguments are true} \\
\left(\sum_0^{512} a\right) &= \text{ROTR}^{28}(a) \oplus \text{ROTR}^{34}(a) \oplus \text{ROTR}^{39}(a) \\
\left(\sum_1^{512} e\right) &= \text{ROTR}^{14}(e) \oplus \text{ROTR}^{18}(e) \oplus \text{ROTR}^{41}(e) \\
\text{ROTR}^n(x) &= \text{circular right shift (rotation) of the 64-bit argument } x \text{ by } n \text{ bits} \\
W_t &= \text{a 64-bit word derived from the current 512-bit input block} \\
K_t &= \text{a 64-bit additive constant} \\
+ &= \text{addition modulo } 2^{64}
\end{aligned}$$

SHA-512 Round Function



Creation of 80-word Input Sequence for SHA-512 Processing of Single Block

SHA-512 Round Function

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

$$\sigma_0^{512}(x) = \text{ROTR}^1(x) \oplus \text{ROTR}^8(x) \oplus \text{SHR}^7(x)$$
$$\sigma_1^{512}(x) = \text{ROTR}^{19}(x) \oplus \text{ROTR}^{61}(x) \oplus \text{SHR}^6(x)$$
$$\text{ROTR}^n(x) = \text{circular right shift (rotation) of the 64-bit argument } x \text{ by } n \text{ bits}$$
$$\text{SHR}^n(x) = \text{left shift of the 64-bit argument } x \text{ by } n \text{ bits with padding by zeros on the right}$$
$$+ = \text{addition modulo } 2^{64}$$

Thank you!!