Unit 3: Message Authentication and Hash Functions

By: Sudhan Kandel

In This Chapter

- *Message Authentication*
- *Hash Functions*
- *Message Digests: MD4 and MD5*
- *Secure Hash Algorithms:*
  - *SHA-1*
  - *SHA-2*
  - *Hash Based MAC (HMAC)*
  - *Digital Signature*

Message Authentication

- Message authentication is a critical aspect of secure, communication, ensuring that a message is both authentic and untampered with. Here are some key concepts and methods involved in message authentication:
- Key concepts

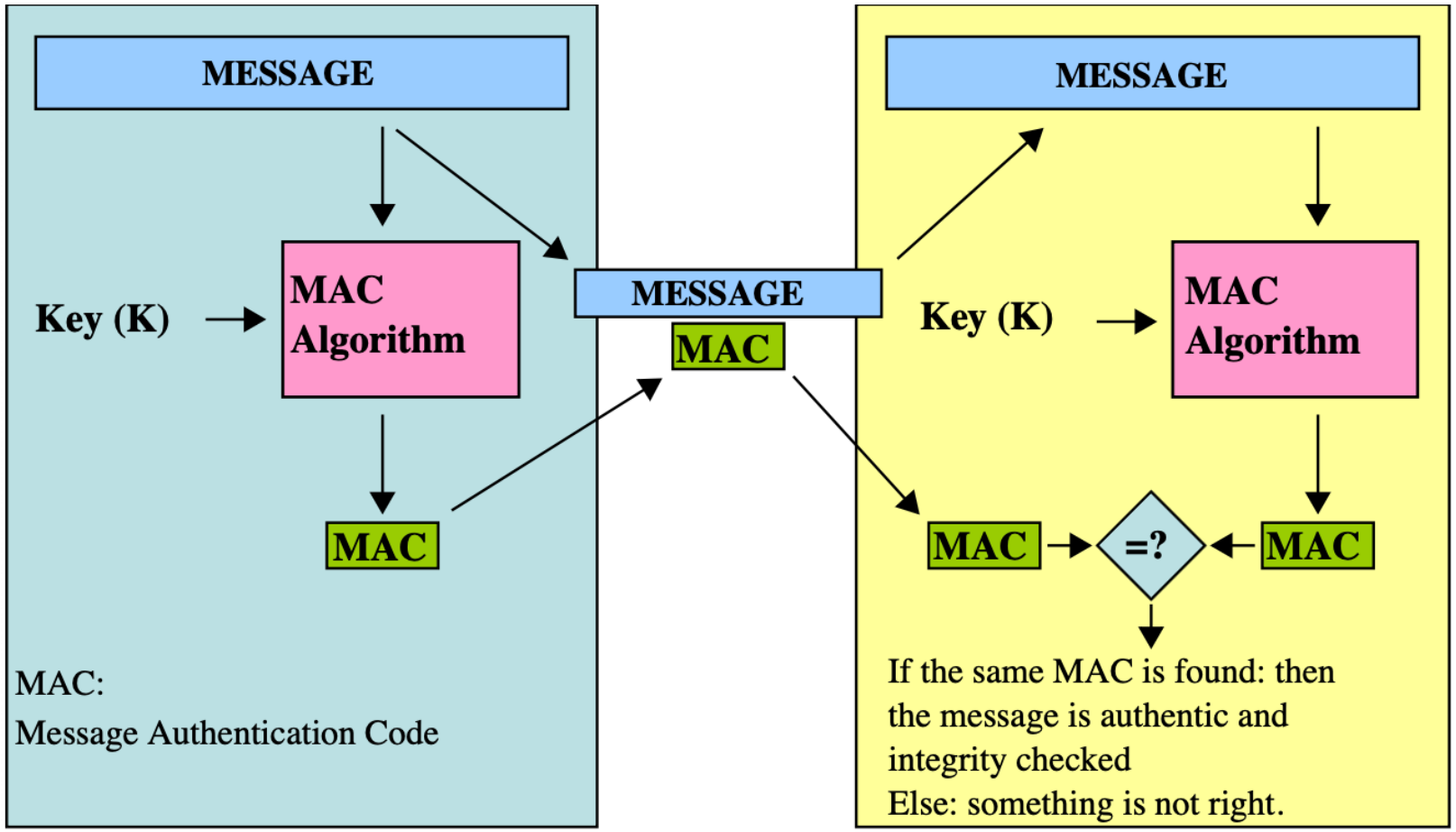
1. **Authentication:** Verifying the idneity of the sender.

2. **Integrity:** Ensuring that the message has not been altered during transmission.

Methods

1. Messgae Authentication Code (MAC)

- A short piece of information used to authenticate a message
- It ensures both the integrity and authenticity of the message.
- Uses a secret key shared between the sender and receiver.
- Common MAC algorithms include HMAC (Hash-based MAC), which is constructed using cryptographic hash function like SHA-256.

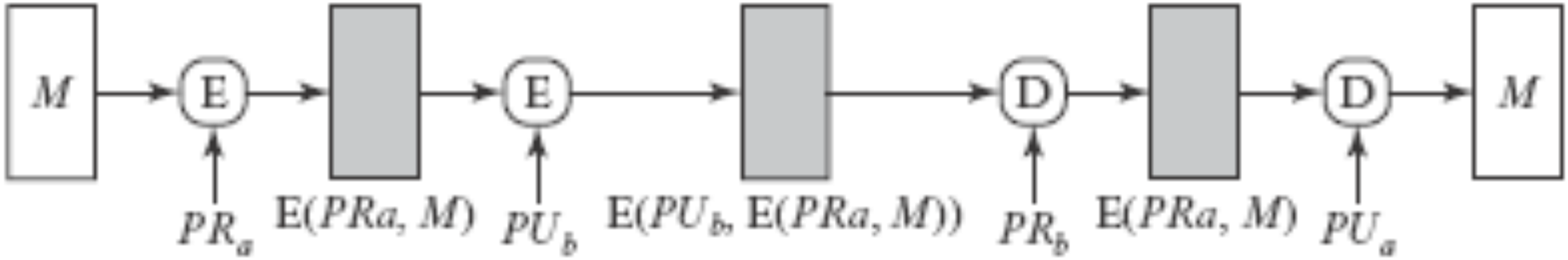

MESSAGE
Key (K)
MAC Algorithm
MAC
MESSAGE
MAC
MAC:
Message Authentication Code
MESSAGE
Key (K)
MAC Algorithm
MAC
=?
MAC
If the same MAC is found: then
the message is authentic and
integrity checked
Else: something is not right.

Contd..

2. Digital Signatures

- A cryptographic technique that provides non-repudiation, integrity, and authenticity.
- Involves two keys: a private key (known only to the sender) and a public key(known to everyone).
- The sender signs the message with their private key, and the recipient verifies the signature with the sender's public key.
- Common algorithms include RSA, DSA and ECDSA.

(c) Public-key encryption: authentication and signature



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

Contd..

3. Hash Functions:

- Used to produce a fixed-size hash value from an input message
- While hash functions themselves do not provide authentication, they are often used in conjunction with MACs and digital signatures.
- Examples include SHA-256 and SHA-3.

Contd..

4. Symmetric Encryption with MAC(Encrypt-then-MAC):

- Combines encryption and MA
- Combined to provide confidentiality, integrity and authenticity
- The message is first encrypted, then a MAC is computed over the ciphertext
- Both the ciphertext and the MAC are sent to the recipient.

1. Message Authentication Code (MAC)

- MAC is a short piece of information, typically a fixed-size value, generated using cryptographic techniques.
- Used to authenticate the integrity and authenticity of a message or data transmission.

Purpose:

- MACs are used to ensure that message has not been tampered with during transmission and to verify the identity of the sender.
- Provides a means of detecting any unauthorized changes or alterations to the message.

Contd..

MACs serve two main purposes:

- **Integrity:** they verify that the contents of the message have not been altered or modified
- **Authenticity:** they confirm that the message was indeed sent by the purported sender and has not been forged by an unauthorized party

Common MAC Algorithms:

- HMAC(Hash-based MAC) is a widely used MAC algorithm. It employes a cryptographic hash function(such as SHA-256) in combination with a secret key to generate the MAC
- Other MAC algorithms include CBC-MAC (Cipher Block Chaining MAC) and GMAC(Galois/Counter Mode MAC), which are used in specific contexts and protocols.

HMAC (Hash-based Message Authentication Code)

- HMAC is a widely used cryptographic algorithm which combines a cryptographic hash function with a secret key for generating message authentication codes(MAC)

Purpose:

- HMAC is used to provide message authentication and integrity verification in communication protocols and applications.
- It ensures that a message has not been tampered with during transmission and verifies the identity of the sender.

Key components:

- **Message:** the data or message that needs to be authenticated
- **Secret key:** a shared secret key known only to the sender and receiver
- **Cryptographic Hash Function:** HMAC uses a cryptographic hash function such as SHA-256, SHA-512, or MD5

Algorithm Steps

1. Key preparation:

- Before the HMAC computation begins, the secret key is optionally modified to meet the requirements of the hash function being used.
- This modification might involve padding or truncation of the key to ensure it matches the block size of the hash function.

2. Inner Padding:

- The secret key, after potential modification, is XORed(exclusive OR) with a specific constant value, often represented as 0x36 in hexadecimal format.
- This XOR operation creates the inner padding which is used to ensure that the key length matches the block size of the hash function

Contd..

3. Outer Padding:

- Similarly, another XOR operation is performed on the modified key, but this time with a different constant value, typically represented as 0x5C.
- This XOR operation creates the outer padding, which is distinct from the inner padding.

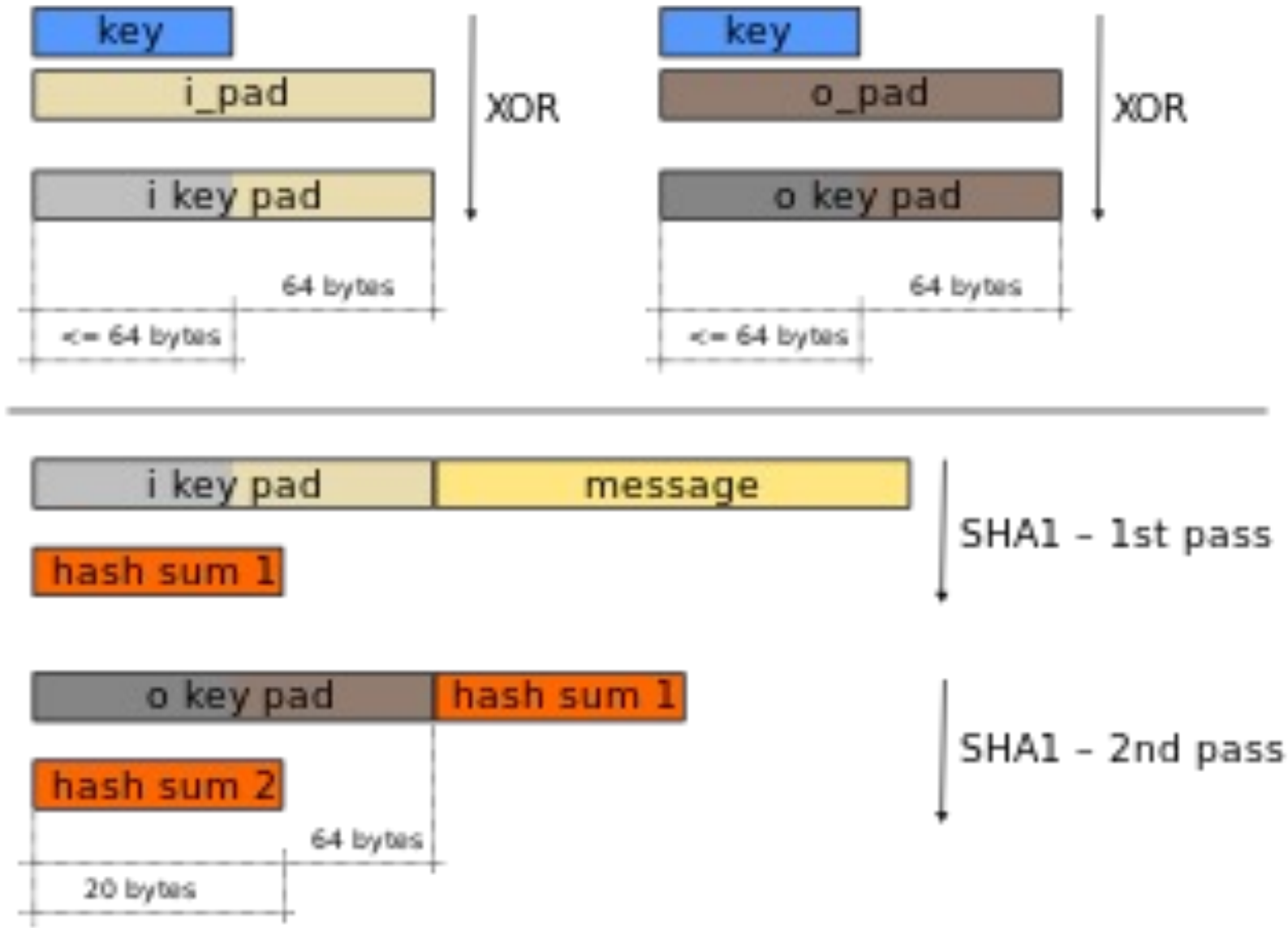
4. Hashing Process:

- The inner padded key is concatenated with the message that needs to be authenticated. This combined data is then hashed using the chosen cryptographic hash function (e.g. SHA-256, SHA-512).
- The resulting hash value from the first hash operation is then concatenated with the outer padded key.
- This concatenated data (hash of inner padding + outer padded key) is hashed again using the same hash function.

Contd..

5. Final MAC Generation:

- The final hash value obtained from the second hash operation serves as the HMAC. This hash value is typically truncated to the desired length if necessary to obtain the final MAC.
- The resulting MAC is a fixed-length hash value or authentication tag that is appended to the message for transmission or verification.


key
i_pad
XOR
i key pad
64 bytes
<= 64 bytes
key
o_pad
XOR
o key pad
64 bytes
<= 64 bytes
i key pad
message
SHA1 – 1st pass
hash sum 1
o key pad
hash sum 1
SHA1 – 2nd pass
hash sum 2
64 bytes
20 bytes

Advantages:

- **Security:** HMAC provides strong security against various cryptographic attacks, including collision attacks and birthday attacks.
- **Efficiency:** it offers efficient computation and verification of MACs, making it suitable for real-time applications and high-volume systems
- **Versatility:** HMAC can be implemented using functions, allowing flexibility and compatibility with various systems and protocols.

Applications:

- HMAC is used in various security protocols and applications, including SSL/TLS, IPsec, SSH, and VPNs, for ensuring the integrity and authenticity of transmitted data
- It is also employed in authentication mechanism for APIs, web services, and cryptographic software libraries.

2. Hash Functions

- Fundamental components in cryptography, used to map data of arbitrary size to fixed-size values.
- These functions are deterministic i.e. the same input will always produce the same output but even a small change in input will produce a significantly different output, a property known as the avalanche effect.

Key properties of Hash Functions

1. Deterministic: the same input will always result in the same hash value.

2. Fast Computation: Hash function are designed to be computationally efficient

3. Pre-image Resistance: It should be computationally infeasible to reverse a hash value to find the original input.

4. Small change sensitivity: A small change in input should produce a significantly different hash

5. Collision Resistance: It should be computationally infeasible to find two different inputs that produce the same hash value

6. Fixed Output Length: Regardless of the input size, the hash output is always of a fixed length

Common Hash Functions

1. MD5 (Message Digest Algorithm 5)

- Produces a 128-bit hash value

2. SHA-1 (Secure Hash Algorithm 1)

- Produces a 160-bit hash value

3. SHA-256 (part of the SHA-2 family)

- Produces a 256-bit hash value
- Widely used and considered secure

Applications of Cryptographic Hash Functions

- **Message Authentication:** Message authentication is a mechanism or service used to verify the integrity of a message. It assures that data received are exactly as sent.
- When a hash function is used to provide message authentication, the hash function value is often referred to as a message digest.
- In many cases, there is a requirement that the authentication mechanism assures that purported identity of the sender is valid.

Applications of Cryptographic Hash Functions

- **Digital Signatures:** The operation of digital signature is similar to that of MAC.
- In the case of digital signature, the hash value of a message is encrypted with a user's private key. Anyone who knows the user's public key can verify the integrity of the message that is associated with the digital signatures.
- An attacker who wishes to alter the message would need to know the user's private key

Applications of Cryptographic Hash Functions

- **Password:** Hash functions are commonly used to create a one-way password file.
- Hash value of a password is stored by an operating system rather than password itself.
- Thus, the actual password is not retrievable by a hacker who gains access to the password file.
- In simple terms, when a user enters a password, the hash of the password is compared to the stored hash value for verification.
- This approach to password protection is used by most operating system.

A. Message digest

- Fixed-size numerical representation(hash) of a message, generated through a hash function and serves as a unique identifier for data integrity and verification purposes.
- Message digests are crucial in cryptographic applications such as digital signatures, password hashing, and data integrity checks.

Key characteristics of Message Digests

1. Fixed Size: Regardless of the size of the input, the output(digest) is always of a fixed size. For instance, SHA-256 always produces a 256-bit digest

2. Uniqueness: Ideally, different inputs shoult produce diffesent digests, minimizing the likelihood of collisions(two different inputs producing the same digest).

3. Deterministic: The same input will always produce the same digest

4. Irreversibility: it should be computationally infeasible to retrive the original input from its digest.

5. Sensitivity to Input changes: A small change in the input should produce a significantly different digest.

Common Hash Functions for Generating Message Digests

1. MD5 (Message Digest Algorithm 5)

- Produces a 128-bit hash value

2. SHA-1 (Secure Hash Algorithm 1)

- Produces a 160-bit hash value

3. SHA-2 Family (SHA-256, SHA-512)

- Produces a 256-bit hash value
- Widely used and considered secure

4. SHA-3

- Uses a different underlying structure(keccak) and is part of the latest series of hash functions standardized by NIST

Practical users of Message Digests

1. **Data Integrity:** Verify that data has not been altered during transmission or storage. For example, downloading a file and checking its digest against a known value ensures the file is intact.
2. **Digital Signatures:** Sign a hash of a message rather than the message itself to save on computational resources and maintain security
3. **Password Storage:** Store hashes of passwords rather than plaintext passwords to enhance security. When a user logs in, the provided password is hashed and compared to the stored hash
4. **Checksum Verification:** Quickly verify the integrity of files by comparing their current hash to a previously computed hash

Message Digests

Message Digest Version 4 (MD4)

The MD4 function is a cryptographic algorithm that takes a message of arbitrary length as input and produces a 128 bit message digest or hash value or fingerprint as output.

Suppose a b-bit message as input and we need to find its message digest.It is assumed that the bits of the message are $m_0, m_1, \ldots\ldots, m_{b-1}$.

Step 1: Append padded bits

The message is padded so that its length is congruent to 448, modulo 512. A single '1' bit is append to the message and then '0' bits are appended so that the length in bits equals 448 modulo 512.

Message	100000	

(Message length + padded bits) % 512 = 448

Step 2: Append length

A 64-bit representation of b is appended to the result of the previous step. The resulting message has a length that is an exact multiple of 512 bits.

Message	100000	64 bits

(Message length + padded bits + 64 bits) % 512 = 0

Step 3: Initialize MD buffer

A 4-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 are initialized to the following values in hexadecimal, low-order byte first:

A: 01 23 45 67
B: 89 ab cd ef
C: fe dc ba 98
D: 76 54 32 10

Step 4: Process message in 16-word (512 bit) blocks

It contains ***three passes (rounds)*** with ***16 steps or operations*** each. We first define three auxiliary functions that each take as input three 32-bit words and produce as output one 32-bit word.

Pass 1: $F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$ [step 0 to 15]
Pass 2: $G(X,Y,Z) = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$ [step 16 to 31]
Pass 3: $H(X,Y,Z) = X \oplus Y \oplus Z$ [step 32 to 47]

One operation is illustrated in the figure below.

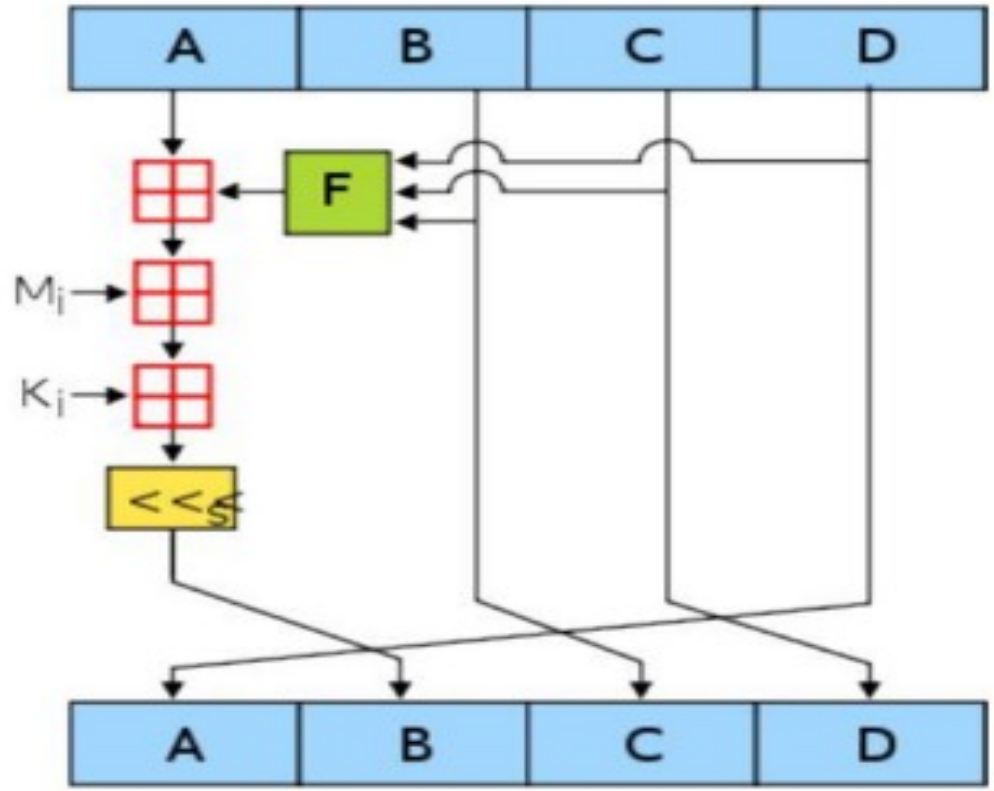


Fig: One MD4 operation

The figure shows how the auxiliary function F is applied to the four buffers (A, B, C and D), using message word M_i and constant K_i. The item "<<<s" denotes a binary left shift by s bits.

Step 5: output

After all rounds have been performed, the buffers A, B, C, D contains the MD4 output starting with lower bit A and ending with higher bit D.

Message Digest Version 5 (MD5)

The MD5-message digest algorithm is a widely used cryptographic hash function producing a 128-bit (16-byte) hash value, typically expressed in text format as a 32 digit hexadecimal number.

MD5 were invented by Ron Rivest as an improved version of MD4.

Operations:

Step 1 – step 3

Same as of MD4

Step 4: Process message in 16-word (512 bit) blocks

It contains ***four passes (rounds)*** with ***16 steps or operations*** each. We first define four auxiliary functions that each take as input three 32-bit words and produce as output one 32-bit word.

Pass 1: $F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$ [step 0 to 15]
Pass 2: $G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$ [step 16 to 31]
Pass 3: $H(X,Y,Z) = X \oplus Y \oplus Z$ [step 32 to 47]
Pass 4: $H(X,Y,Z) = Y \oplus (X \wedge \neg Z)$ [step 48 to 64]

One operation is illustrated in the figure below.

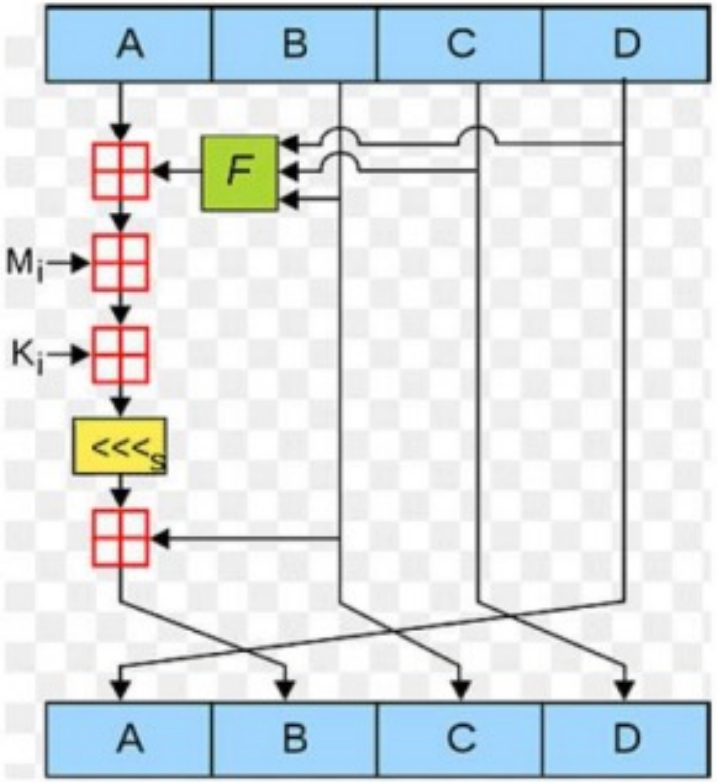


Fig: One MD5 operation

The figure shows how the auxiliary function F is applied to the four buffers (A, B, C and D), using message word M_i and constant K_i. The item "<<<s" denotes a binary left shift by s bits.

Step 5: output

After all rounds have been performed, the buffers A, B, C, D contains the MD5 output starting with lower bit A and ending with higher bit D.

Secure Hash Standard (SHS)

- The secure Hash Standard (SHS) is a set of cryptographically secure hash algorithms specified by the National Institute of Standards and Technology.
- This standard specifies a number of Secure Hash Algorithms (SHA), SHA-1, SHA-2,SHA-3, SHA-224, SHA-256 etc.
- All of the algorithms are iterative, one way hash functions that process a message to produce a message digest.
- These algorithms enable the determination of a message's integrity; any change to the message will, with a very high probability, results in a different message digest.
- This properties is useful in the generation and verification of digital signatures and message authentication codes.

Secure Hash Standard (SHS)

- Each algorithm can be described in two stages: **Preprocessing** and **Hash Computation**.
- Preprocessing involves padding a message, parsing the padded message into m-bit blocks, and setting initialization values to be used in the hash computation.
- The hash computation generates a message schedule from the padded message and uses that schedule along with functions, constants and word operations to iteratively generates a series of hash values.
- The final hash value generated by the hash computation is used to determine the message digest.

Parameters for various version of SHA

Algorithm	Message Digest Size	Message Size	Block Size	Word Size	No of Step
SHA-1	160	$<2^{64}$	512	32	80
SHA-224	224	$<2^{64}$	512	32	64
SHA-256	256	$<2^{64}$	512	32	64
SHA-384	384	$<2^{128}$	1024	64	80
SHA-512	512	$<2^{128}$	1024	64	80

Note: *All sizes are measured in bits.*

SHA-1 (Secure Hash Algorithm)

- **SHA-1** (Secure Hash algorithm) was proposed by NIST as message digest function.
- **SHA-1** takes message of length at most 2^{64} bits and produces a 160 bit output.
- It is similar to the **MD5** message digest function but it is little slower to execute and presumably more secure.
- MD4 made three passes over each block of data, MD5 made four; SHA-1 makes five.
- It also produces a 160 bit digest as opposed to the 128 of the MDs.

Operations:

1. ***Message Padding:***

   Padding is performed as follows: a single '1' bit is append to the message, and then enough zero bits are append so that the length in bits of the padded message becomes congruent to 448, modulo 512.

2. ***SHA-1 Message digest computation:***

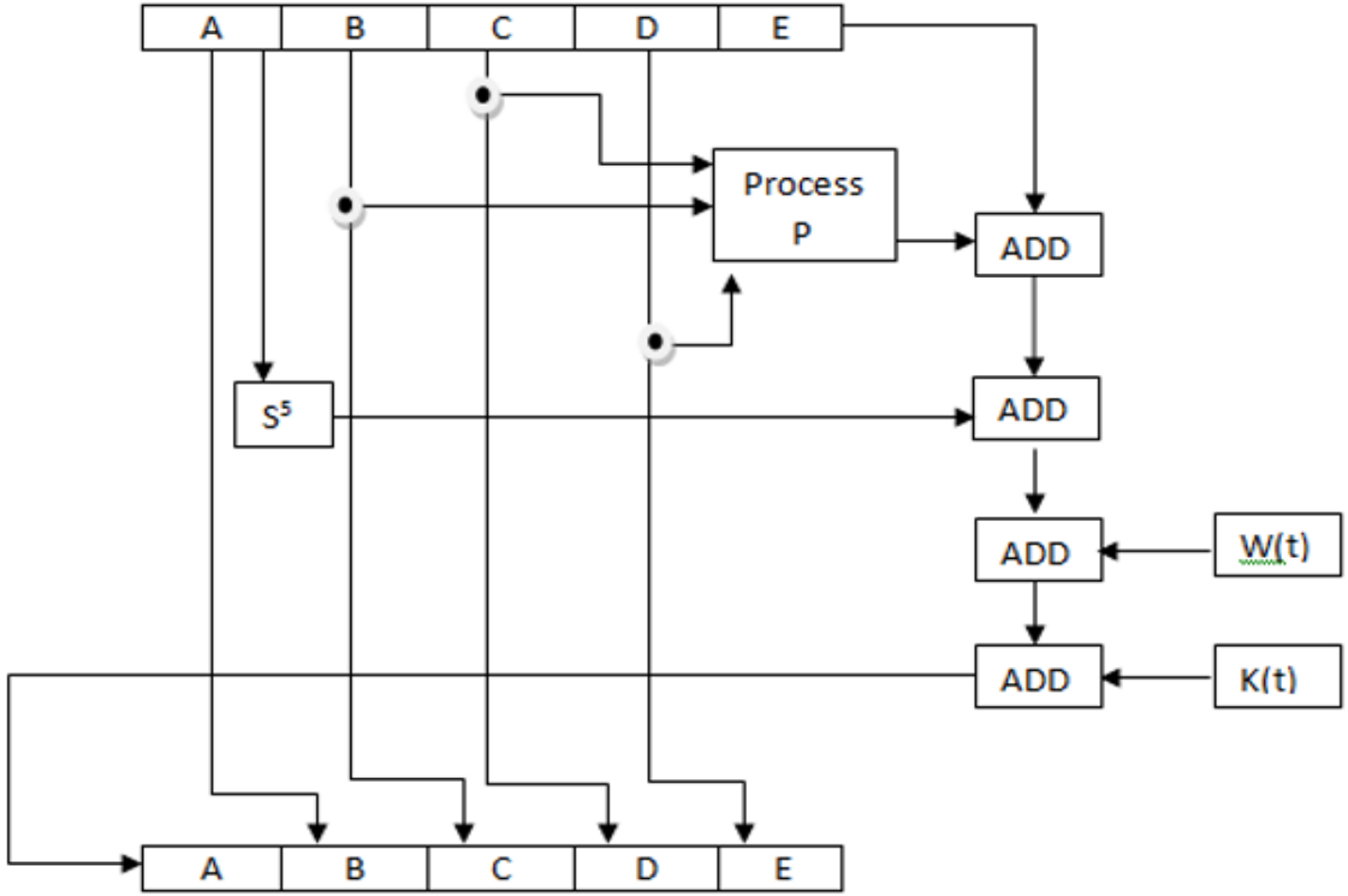
   SHA-1 consists of 80 iterations. Each time it process the 512 bits message and at last it produces the output of 160 bits hash value.

   Let the 160 bits hash value be $ABCDE$.
   Initially,

   $A = 67\ 45\ 23\ 01$
   $B = ef\ cd\ ab\ 89$
   $C = 98\ ba\ dc\ fe$
   $D = 10\ 32\ 54\ 76$
   $E = c3\ d2\ e1\ f0$

   In each iteration, these values are updated as

   $B = old\ A$
   $C = old\ B \lrcorner 30$
   $D = old\ C$
   $E = old\ D$
   $A = E + (A \lrcorner 5) + W_t + K_t + f(t, B, C, D)$



W(t) is the expanded message word of round t;
K(t) is the round constant of round t;

Where,

$W_t = W_{n-3} \oplus W_{n-8} \oplus W_{n-14} \oplus W_{n-16}$ is the t^{th} 32 bits word block and K_t is the constant depending upon the value of t given by following relations

$K_t = 5a827999 \quad if\ 0 \leq t \leq 19$

$K_t = 6ed9eba1 \quad if\ 20 \leq t \leq 39$

$K_t = 8f1bbcdc \quad if\ 40 \leq t \leq 59$

$K_t = ca62c1d6 \quad if\ 60 \leq t \leq 79$

Again, $f(t, B, C, D)$ is a function that varies according to the following relations:

$f(t, B, C, D) = (B \wedge C) \vee (\neg B \wedge D) \quad if\ 0 \leq t \leq 19$

$f(t, B, C, D) = B \oplus C \oplus D \quad if\ 20 \leq t \leq 39$

$f(t, B, C, D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D) \quad if\ 40 \leq t \leq 59$

$f(t, B, C, D) = B \oplus C \oplus D \quad if\ 60 \leq t \leq 79$

Secure Hash Algorithm-2 (SHA-2)

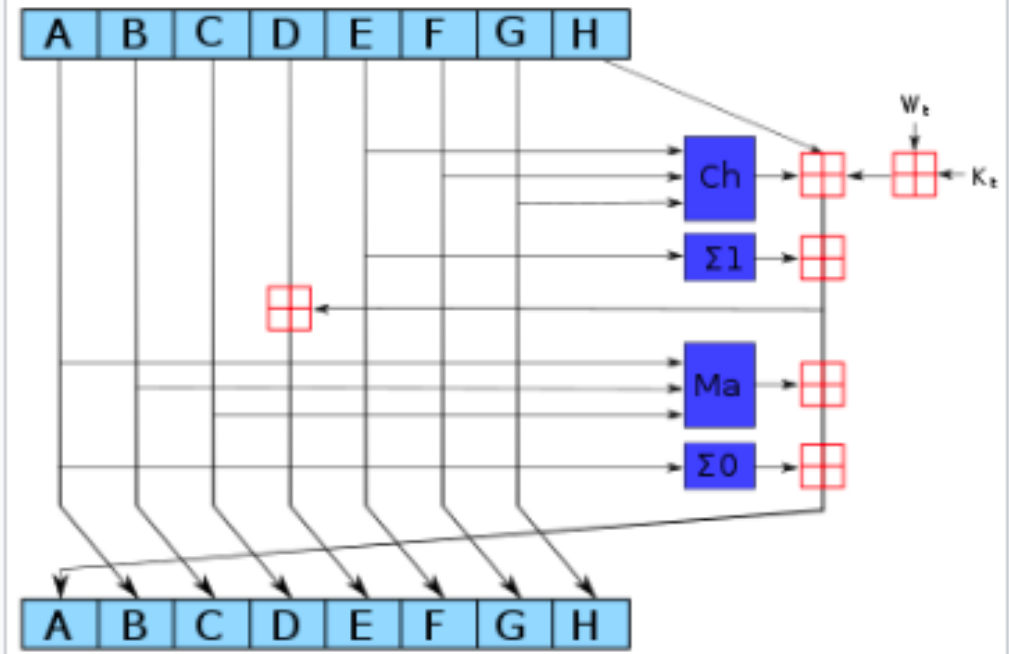
SHA-2 is a family of two similar hash functions known as SHA-256 and SHA-512, with different block sizes. Both algorithm belongs to SHA-2. They differ in the word size. SHA-256 uses 32-bit words where SHA-512 uses 64-bit words. There are also truncated versions of each standard, known as SHA-224, SHA-384, SHA-512, SHA-512/224, SHA-512/256.

Parameters for various version of SHA

Algorithm	Message Digest Size	Message Size	Block Size	Word Size	No of Step
SHA-1	160	$<2^{64}$	512	32	80
SHA-224	224	$<2^{64}$	512	32	64
SHA-256	256	$<2^{64}$	512	32	64
SHA-384	384	$<2^{128}$	1024	64	80
SHA-512	512	$<2^{128}$	1024	64	80

Note: *All sizes are measured in bits.*

- 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-256 and SHA-512 are novel hash functions computed with 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).

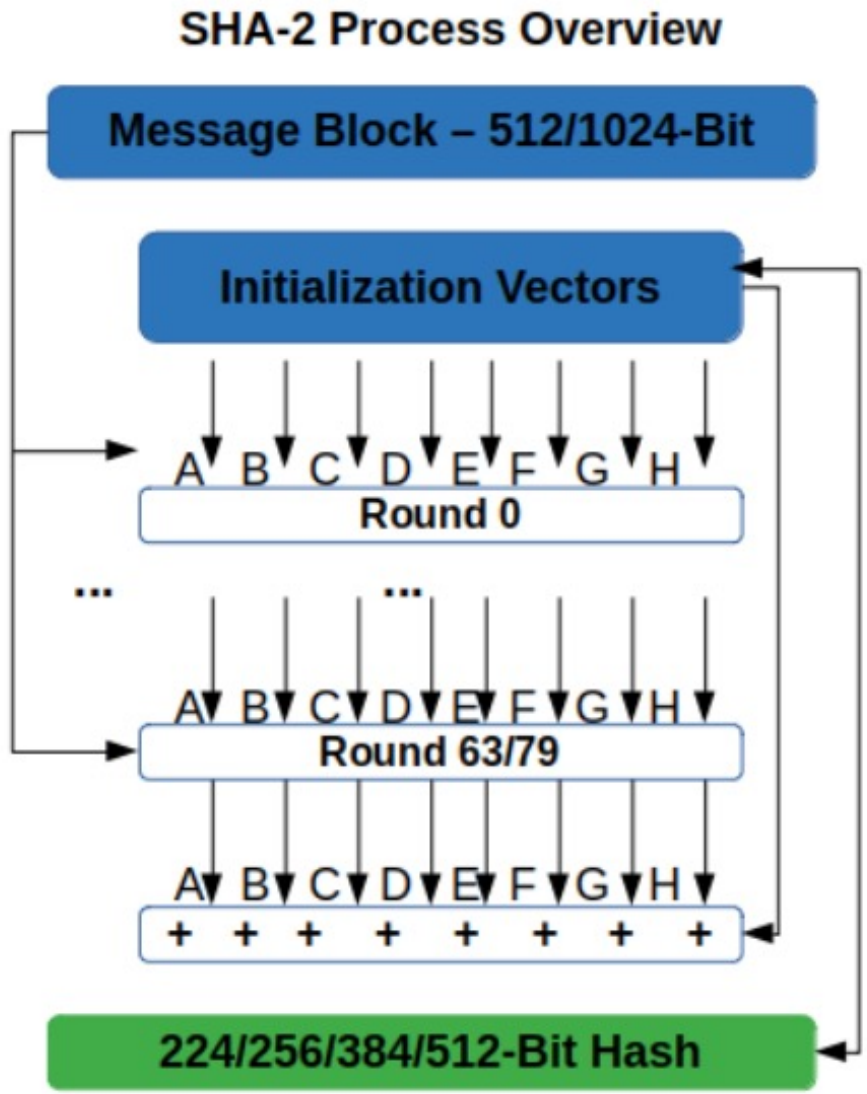


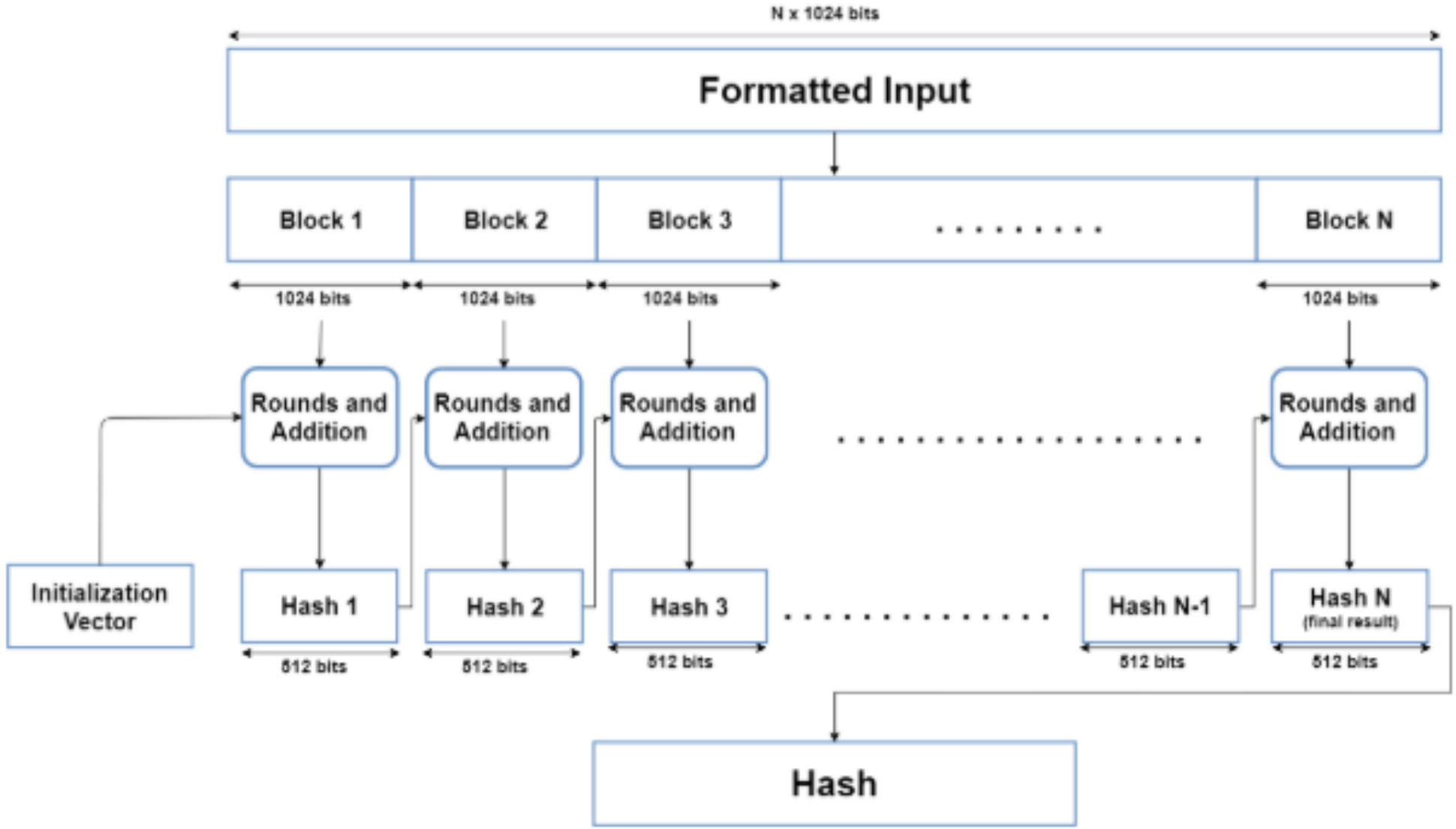
One iteration in a SHA-2 family compression function. The blue components perform the following operations:

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

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-2 Process Overview
Message Block – 512/1024-Bit
Initialization Vectors
A B C D E F G H
Round 0
...
...
A B C D E F G H
Round 63/79
A B C D E F G H
+ + + + + + + +
224/256/384/512-Bit Hash


N x 1024 bits
Formatted Input
Block 1
Block 2
Block 3
. . . . . .
Block N
1024 bits
1024 bits
1024 bits
1024 bits
Rounds and Addition
Rounds and Addition
Rounds and Addition
. . . . . .
Rounds and Addition
Initialization Vector
Hash 1
Hash 2
Hash 3
. . . . . .
Hash N-1
Hash N
(final result)
512 bits
512 bits
512 bits
512 bits
512 bits
Hash

SHA-512:

- SHA-512 works in four stages as follows:
  - Input formatting
  - Hash buffer initialization
  - Message Processing
  - Output

Digital Signature

Digital signature is an electronic signature that can be used to authenticate the identity of the sender of a message and to ensure that the original content of the message or document that has been sent is unchanged.

Digital signature schemes normally gives two algorithms, one for signing which involves the user's secret or private key and one for verifying signatures which involves the user's public key.

Digital signature processes

- **Key Generation**
- **Signing**
- **Verification**

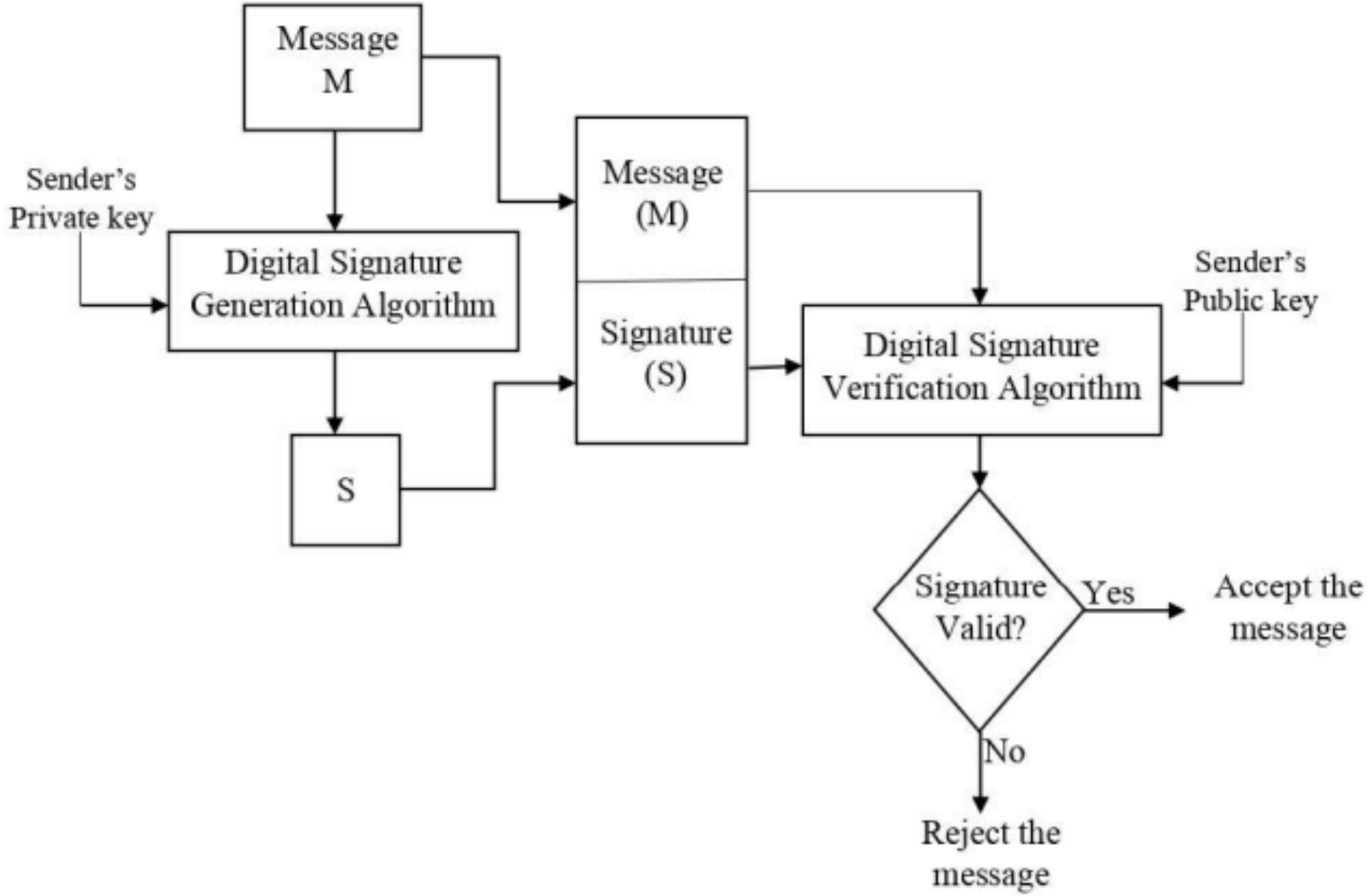


Fig: Generic model of Digital Signature

Advantages of Digital Signatures

- **Authentication**
- **Integrity**
- **Non-repudiation**
- **Efficiency**

Disadvantages of Digital Signatures

- Key Management
- Infrastructure Dependence
- Complexity
- Legal Recognition

Types of Digital Signatures

- **RSA Digital Signatures**
  - **Algorithm:** Based on the RSA algorithm, which involves the use of modular arithmetic and the difficulty of factoring large prime numbers.
  - **Security:** Widely used and considered secure, RSA digital signatures provide strong security when implemented correctly.
  - **Usage:** Commonly employed in various applications such as SSL/ TLS certificates, digital documents and software distribution.

DSA

- Digital Signature Algorithm
- **Standarization:** DSA is a federal Information processing standard(FIPS) for digital signatures, standardized by NIST
- **Usage**: It is often used in conjunction with the digital signature standard for secure digital signatures and key exchange in cryptographic protocols like Diffe-hellman Key exchange.

EDCSA

- Elliptic Curve Digital Signature Algorithm
- **Algorithm:** Based on Elliptic Curve Cryptography, EDSA offers similar security level to RSA but with smaller key size, making it more efficient for resource constrained environments.
- **Efficiency:** Due to its similar key sizes, EDSA is particularly suitable for applications where computational resources and bandwidth are limited, such as mobile devices and IOT devices.

PSS

- Probabilistic Signature Scheme
- **Enhance Security:** PSS is a probabilistic digital signature scheme that provides enhance security features compared to traditional deterministic schemes.
- **Usage:** PSS is often used in conjuction with RSA for applications requiring higher level of security assurance.

Thank You