### Message Integrity:

#### Message confidentiality

Two or more hosts communicate securely, typically using encryption. The communication cannot be monitored (sniffed) by untrusted hosts. The communication between trusted parties is confidential.

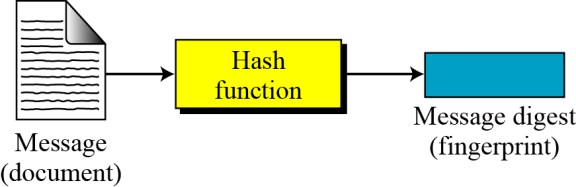
#### Message integrity

The message transported has not been tampered with or altered. A message has integrity when the payload sent is the same as the payload received.

Sending a message confidentially does not guarantee data integrity. Even when two nodes have authenticated each other, the integrity of a message could be compromised during the transmission of a message. We can have integrity of a message without confidentiality.

One way to **preserve** the **integrity** of a document is through the use of a **fingerprint**. If Alice needs to be sure that the contents of her document will not be changed, she can put her fingerprint at the bottom of the document.

The electronic equivalent of the document and fingerprint pair is the message and digest pair.



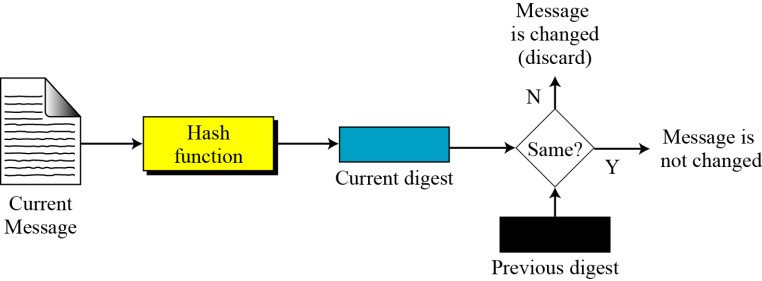
### Difference between Fingerprint and Digest:

The two pairs (document / fingerprint) and (message / message digest) are similar, with some differences.

* The document and fingerprint are physically linked together.
* The message and message digest can be unlinked separately, and, most importantly, the message digest needs to be safe from change.

**Checking Integrity Process:**

To check the integrity of a message, run the cryptographic hash function again and compare the new message digest with the previous one. If both are the same, it assures that the original message has not been changed.

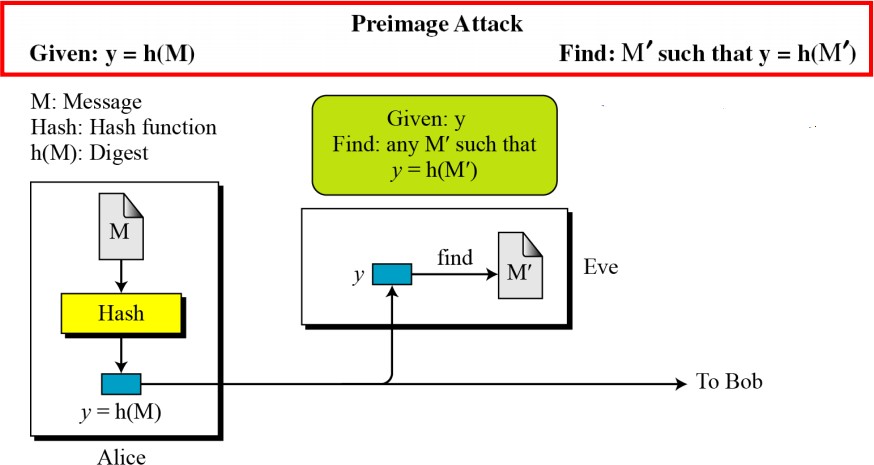


X- -

### Properties of Hash Functions

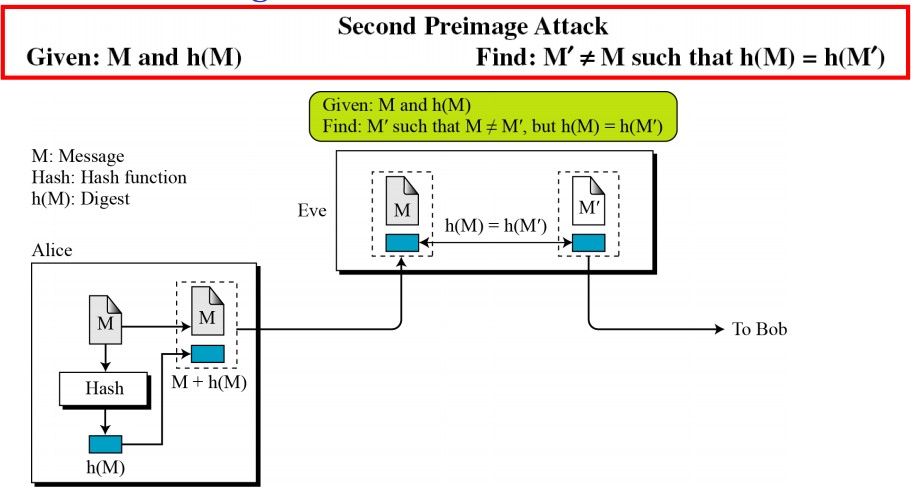
In order to be an effective cryptographic tool, the hash function is desired to possess following properties −

### Pre-Image Resistance

* + - * This property means that it should be computationally hard to reverse a hash function.
      * In other words, if a hash function h produced a hash value z, then it should be a difficult process to find any input value x that hashes to z.
      * This property protects against an attacker who only has a hash value and is trying to find the input.

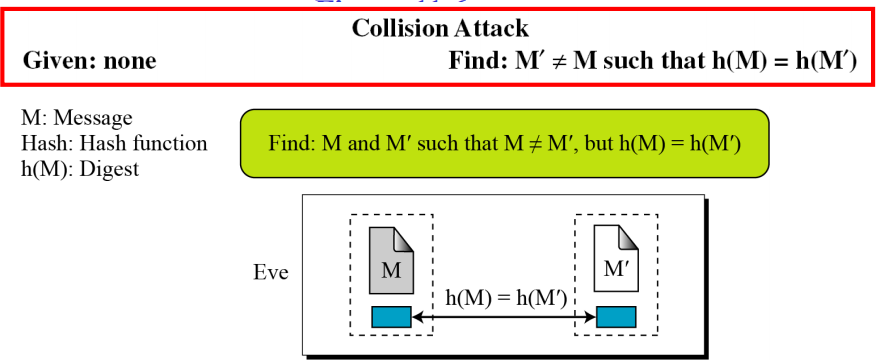
### Second Pre-Image Resistance

* + - * This property means given an input and its hash, it should be hard to find a different input with the same hash.
      * In other words, if a hash function h for an input x produces hash value h(x), then it should be difficult to find any other input value y such that h(y) = h(x).
      * This property of hash function protects against an attacker who has an input value and its hash, and wants to substitute different value as legitimate value in place of original input value.



### Collision Resistance

* + - * This property means it should be hard to find two different inputs of any length that result in the same hash. This property is also referred to as collision free hash function.
      * In other words, for a hash function h, it is hard to find any two different inputs x and y such that h(x) = h(y).
      * Since, hash function is compressing function with fixed hash length, it is impossible for a hash function not to have collisions. This property of collision free only confirms that these collisions should be hard to find.
      * This property makes it very difficult for an attacker to find two input values with the same hash.
      * Also, if a hash function is collision-resistant **then it is second pre-image resistant.**



**Example:** Ahmad may write a will to distribute his estate upon his death. The will does not need to be encrypted because anyone can examine the will after he died. However, the integrity of the will needs to be preserved so that the contents of the will are unchanged.

A cryptographic hash function creates a message digest out of a message that meets three criteria: preimage resistance, second preimage resistance, and collision resistance.

A **MDC (Modification Detection Code)** is a message digest that can prove the integrity of the message that not been changed.

* To prove the integrity of a message and the data origin authentication (message authentication), we need to change MDC to MAC with a secret between sender and receiver.

-X - -

### RANDOM ORACLE MODEL

The Random Oracle Model, which was introduced in 1993 by Bellare and Rogaway, is an ideal mathematical model for a hash function.

A function based on Random Oracle Model behaves as follows:

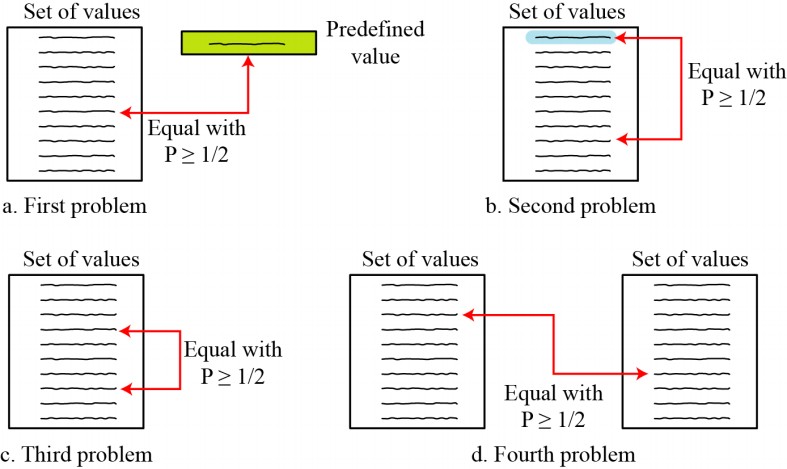
1. When a new message of any length is given, the oracle creates and gives a fixed-length message digest that is a random string of 0 s and 1 s. The oracle records the message and the message digest.
2. When a message is given for which a digest exists, the oracle simply gives the digest in the record.
3. The digest for a new message needs to be chosen independently from all previous digests. This implies that oracle cannot use a formula or an algorithm to calculate the digest.

**Random Oracle Model follows Pigeonhole Principle:** It states that

If n pigeonholes are occupied by n + 1 pigeons, then at least one pigeonhole is occupied by two pigeons.

The generalized version of the pigeonhole principle is that if n pigeonholes are occupied by kn + 1 pigeons, then at least one pigeonhole is occupied by k+1 pigeons.

### Birthday Problems related to the Random Oracle Model:



**Problem1:** What is the maximum number of instances such that at least one instance is equal to a predefined value?

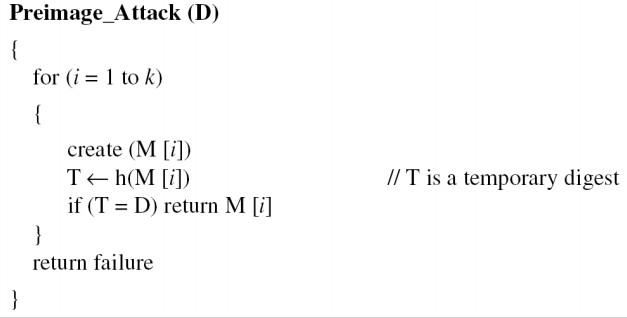
**Problem2:**What is the minimum number of instances such that at least one instance is equal to selected one?

**Problem3:** What is the minimum number of instances such that at least two instances are equal?

**Problem4:** What is the minimum number of instances such that at least one instance from first set is equal to one instance in the second set?

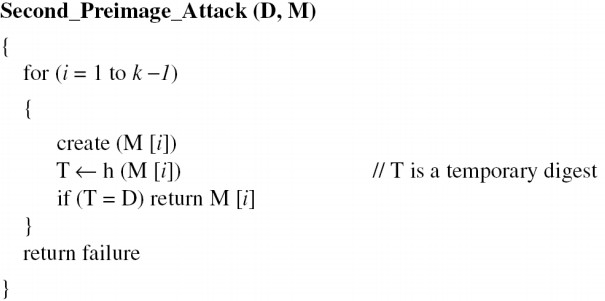
### Attacks on Random Oracle Model :

1. **Preimage Attack: Algorithm**



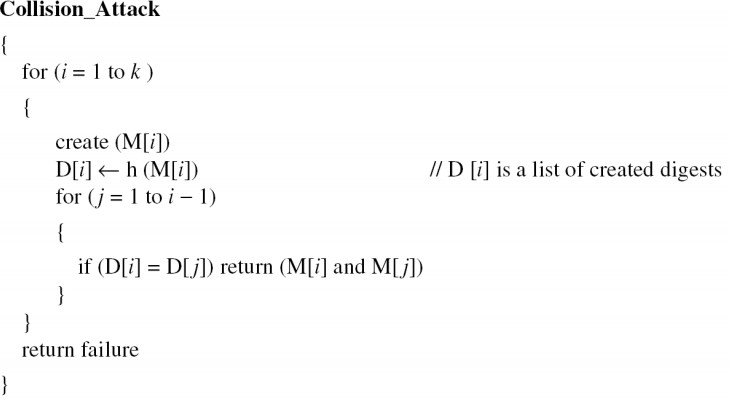
The difficulty of a preimage attack is proportional to 2n.

### Second-Preimage Attack: Algorithm



The difficulty of a second-preimage attack is proportional to 2n.

### Collision Attack: Algorithm



The difficulty of a collosion attack is proportional to 2n/2.

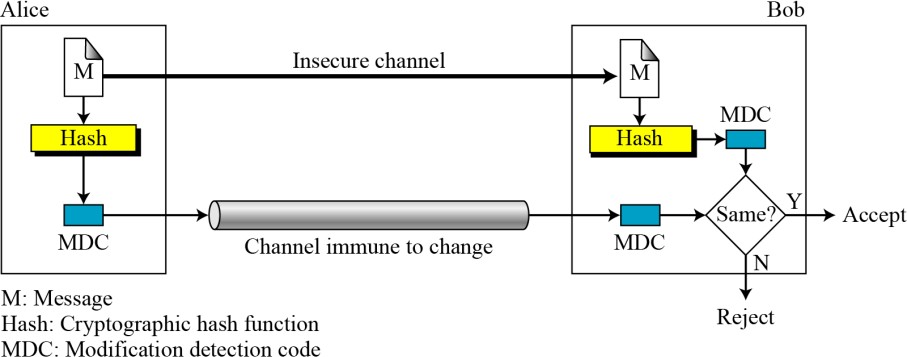
-X - -

### Message Authentication

A message digest does not authenticate the sender of the message. To provide message authentication, Alice needs to provide proof that it is Alice sending the message and not an impostor. The digest created by a cryptographic hash function is normally called a modification detection code (MDC). What we need for message authentication is a message authentication code (MAC).

### Modification Detection Code:

A modification detection code (MDC) is a message digest that can prove the integrity of the message: that message has not been changed. If Alice needs to send a message to Bob and be sure that the message will not change during transmission, Alice can create a message digest, MDC, and send both the message and the MDC to Bob. Bob can create a new MDC from the message and compare the received MDC and the new MDC. If they are the same, the message has not been changed.

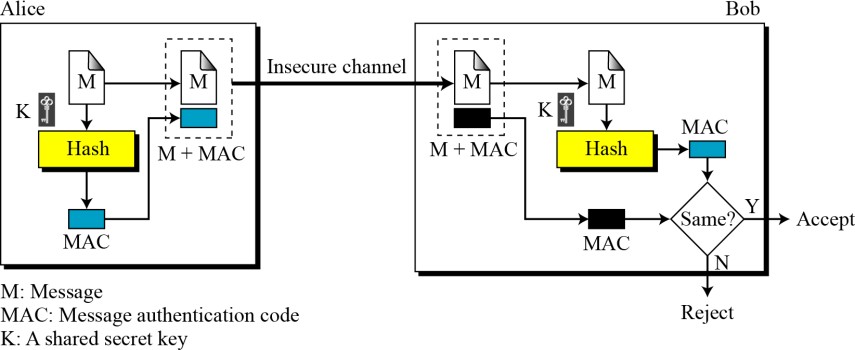


### Message Authentication Code (MAC)

MAC algorithm is a symmetric key cryptographic technique to provide message authentication. For establishing MAC process, the sender and receiver share a symmetric key K.

Essentially, a MAC is an encrypted checksum generated on the underlying message that is sent along with a message to ensure message authentication.

The process of using MAC for authentication is depicted in the following illustration −



Let us now try to understand the entire process in detail −

* + - The sender uses some publicly known MAC algorithm, inputs the message and the secret key K and produces a MAC value.
    - Similar to hash, MAC function also compresses an arbitrary long input into a fixed length output. The major difference between hash and MAC is that MAC uses secret key during the compression.
    - The sender forwards the message along with the MAC. Here, we assume that the message is sent in the clear, as we are concerned of providing message origin authentication, not confidentiality. If confidentiality is required then the message needs encryption.
    - On receipt of the message and the MAC, the receiver feeds the received message and the shared secret key K into the MAC algorithm and re-computes the MAC value.
    - The receiver now checks equality of freshly computed MAC with the MAC received from the sender. If they match, then the receiver accepts the message and assures himself that the message has been sent by the intended sender.
    - If the computed MAC does not match the MAC sent by the sender, the receiver cannot determine whether it is the message that has been altered or it is the origin that has been falsified. As a bottom-line, a receiver safely assumes that the message is not the genuine.

#### Security of a MAC

Three possible cases of Eve forging a message:

1. Exhaustive search of key.
2. The preimage attack: Eve can find the key from the intercepted MAC (= h(K | M) ). (\* Thus Eve can successfully replace the message with a forged one. \*)
3. Given some pairs of messages and their MACs, Eve can manipulate them to come up with a new message and its MAC.

### Limitations of MAC

There are two major limitations of MAC, both due to its symmetric nature of operation −

### Establishment of Shared Secret.

* + - * It can provide message authentication among pre-decided legitimate users who have shared key.
      * This requires establishment of shared secret prior to use of MAC.

### Inability to Provide Non-Repudiation

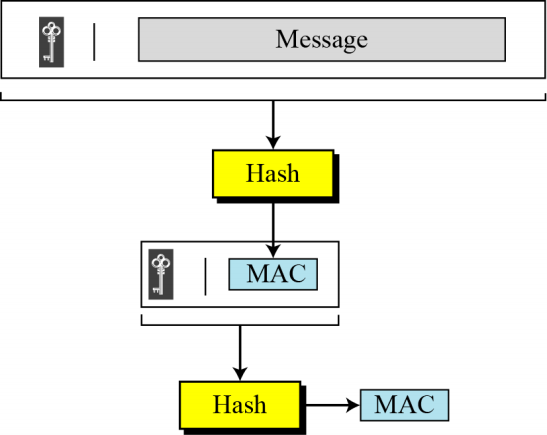
* + - * Non-repudiation is the assurance that a message originator cannot deny any previously sent messages and commitments or actions.
      * MAC technique does not provide a non-repudiation service. If the sender and receiver get involved in a dispute over message origination, MACs cannot provide a proof that a message was indeed sent by the sender.
      * Though no third party can compute the MAC, still sender could deny having sent the message and claim that the receiver forged it, as it is impossible to determine which of the two parties computed the MAC.

Both these limitations can be overcome by using the public key based digital signatures discussed in following section.

-X

### Nested MAC

To improve the security of a MAC, **nested MACs** were designed in which hashing is done in two steps.

1. The key is concatenated with the message and is hashed to create an intermediate digest.
2. The key is concatenated with the intermediate digest to create the final digest.

-X

### HMAC – Hashed MAC

The implementation Of HMAC is much more complex than the simplified nested MAC.

### HMAC Procedure:

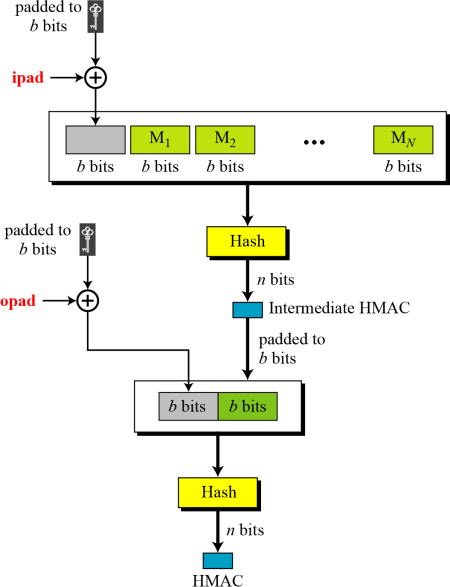
1. The message is divided into N blocks, each of b-bits.
2. The Secrete Key is left-padded with 0’s to create b-bits.
3. The result of step 2 is ex-ored with a constant called **ipad (input pad)** to create a b-bit block.

The value of ipad is the b/8 repetition of the sequence 00110110( 36 in Hexa-Decimal)

1. The resulting block is prepended to the N-block message. The result is N+1 blocks.
2. The result of step 4 is hashed to create an n-bit digest. We call the digest the intermediate HMAC.
3. The intermediate n-bit HMAC is left padded with 0’s to make a b-bit block.
4. Step 2 and 3 are repeated by a different constant opad (**output pad**).

The value of opad is the b/8 repetition of the sequence 01011100( 5C in Hexa-Decimal)

1. The result of step 7 is prepended to the block of step 6.
2. The result of step 8 is hashed with the same hashing algorithm to create the final n-bit HMAC.

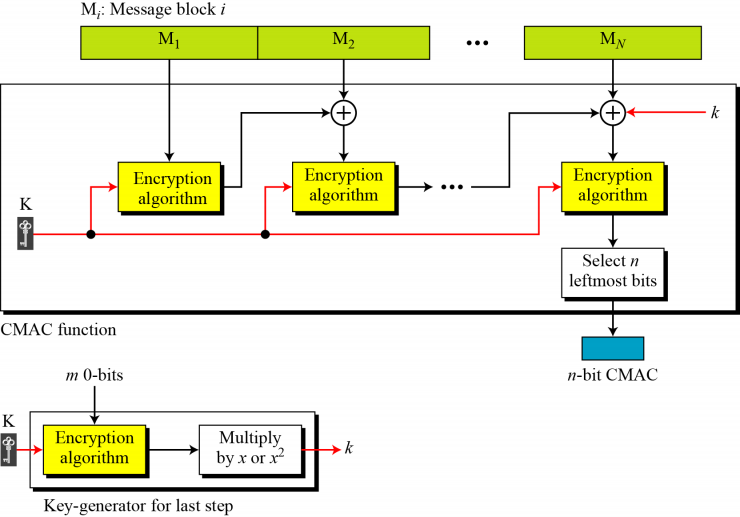


-X

* 1. **CMAC: (Cipher-based Message Authentication Code)**
     + It is a block cipher-based message authentication code **algorithm**. It may be used to provide assurance of the authenticity and, hence, the integrity of binary data.
     + CMAC codes are a tool for calculating message authentication codes using a block cipher coupled with a secret key. You can use an CMAC to verify both the integrity and authenticity of a message.
     + The method is similar to CBC mode.
     + However, the idea is to create one block of MAC from *N* blocks of plaintext, not to create

*N* blocks of ciphertext from *N* blocks of plaintext.

* + - The message is divided into *N* blocks, each *m* bits long.
    - The size of CMAC is *n* bits.
    - If the last block is not *m* bits, it is padded with a 1-bit followed by enough 0-bits to make it *.*
    - The *n* leftmost bit from the last block is the CMAC.
    - In addition to the symmetric key, K, CMAC also uses another key, *k*, which applied only at the last step.
    - The result from the Encryption algorithm is multiplied by *x* if no padding is applied and is multiplied by *x*2 if padding is applied.
    - The multiplications is in GF(2*m*) with irreducible polynomial of degree *m* selected by the particular protocol used.



-X -

#### Cryptographic hash function

A cryptographic hash function takes a message of arbitrary length and creates a message digest of fixed length. . The string is called the 'hash value', 'message digest', 'digital fingerprint', 'digest' or 'checksum'.

The ideal hash function has three main properties:

* + 1. It is extremely easy to calculate a hash for any given data.
    2. It is extremely computationally difficult to calculate an alphanumeric text that has a given hash.
    3. It is extremely unlikely that two slightly different messages will have the same hash.

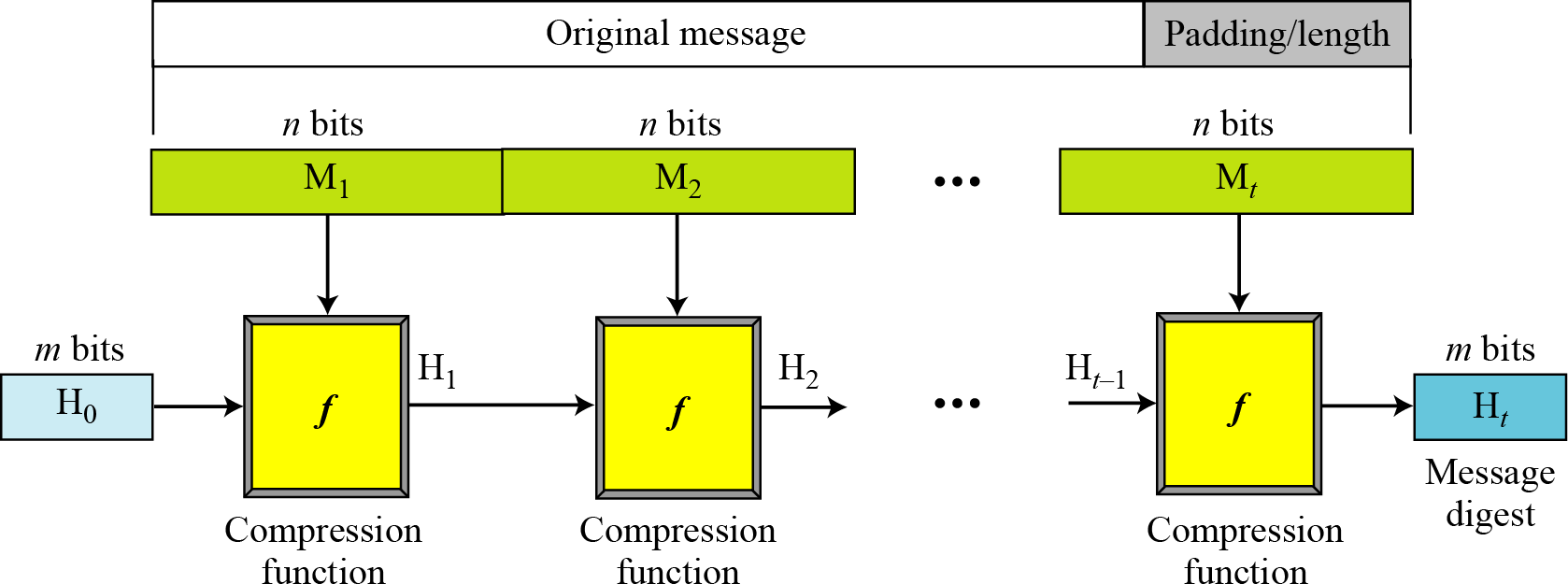
### ------------------------------------------------------------------X-----------------------------------------------------------

#### Iterated Hash Function:

* All cryptographic hash functions need to create a fixed-size digest out of a variable-size message.
* Creating such a function is best accomplished using iteration.
* Instead of using a hash function with variable-size input, a function with fixed-size input is created and is used a necessary number of times.
* The fixed-size input function is referred to as a compression function.It compresses an n-bit string to create an m-bit string, where normally n > m.

The scheme is referred to as an **iterated cryptographic hash function.**

***Merkle-Damgard Scheme:*** The Merkle-Damgard scheme is an iterated hash function that is collision resistant if the compression function is collision resistant



### Procedure:

* + 1. The message length and padding are appended to the message to divide into **t** blocks of size n bits, M1, M2, Mt
    2. The digest is created at t iterations H1, H2, Ht .
    3. Before starting the iteration, the digest H0 is set to a fixed value, normally called IV (Initial Value or Initial Vector)
    4. The compression function at each iteration operates on Hi-1 and Mi to create a new Hi . in other words, we have Hi =f(Hi-1 , Mi ), where f is the compression function.
    5. Ht is the cryptographic hash function of the original message, i.e h(M).

### ----------------------------------------------------------X----------------------------------------------------------

**2.2 Categories of Compression Functions**

Two different approaches in designing a hash function:

### Hash functions made from scratch.

* 1. **Message Digest (MD):** MD2, MD4 and MD5 are belongs to message digest family and are designed by Ron Rivest. The last version MD5 is strengthened version of MD4 that divides the message into blocks of 512 bits and creates a 128-bit digest.
  2. **Secure Hash Algorithm(SHA):** Also called Secure Hash Standards, developed by NIST. The standard is based on MD5 . New versios: SHA-224, SHA-256, and SHA-512.
  3. **Other algorithms :** RACE Integrity Primitive Evaluation Message Digest (RIPMED)

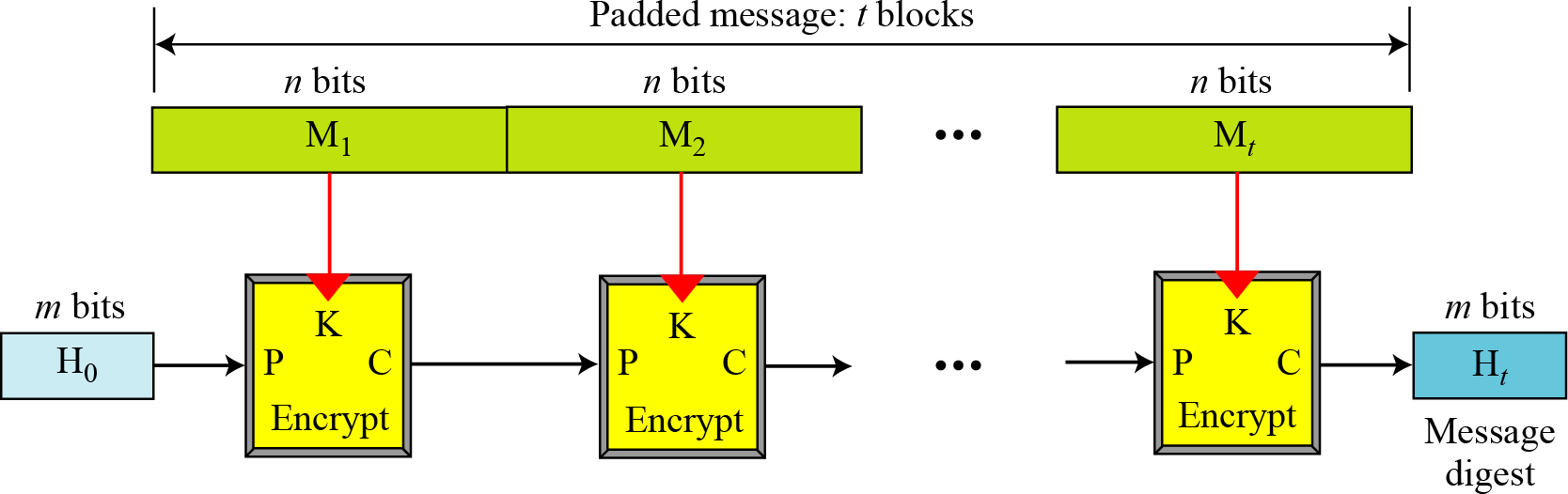
### Hash functions based on block cipher.

* A symmetric-key block cipher can be used as a compression function.
* Several secure symmetric-key block cipher, such as *DES* or *AES*, can be used to make a one- way function instead of creating a new compression function.
* The block cipher in this case only performs encryption.
* The most promising one is *Whirlpool*.

### Rabin Scheme

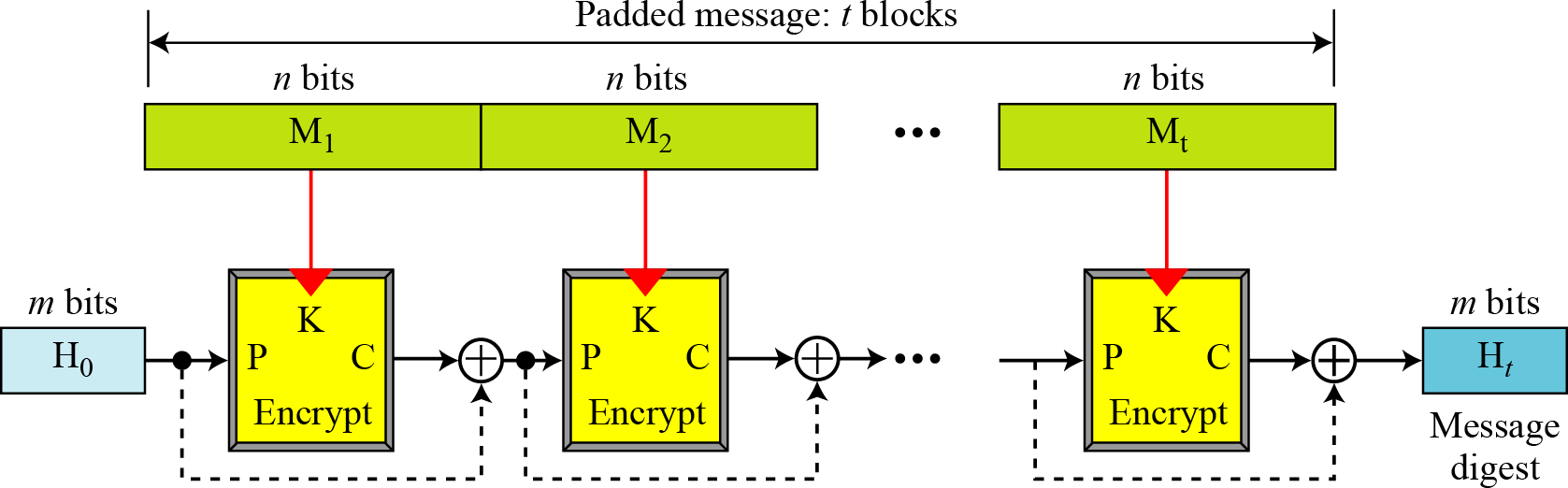
This scheme is based on the Merkle-Damgard scheme. The message block is used as the key. The size of digest is the size of data block cipher in the underlying cryptosystem.

This scheme is subject to a meet-in-the-middle attack, Because the adversary can use the decryption algorithm.



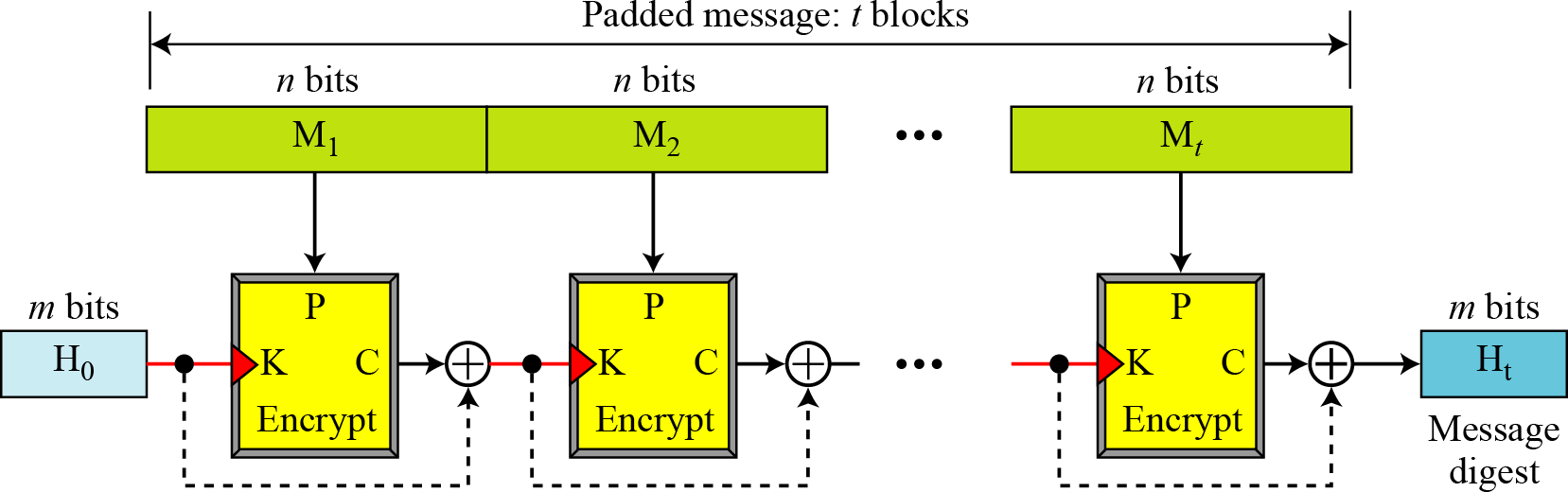
### Davies-Meyer Scheme

This scheme is basically the same as the Rabin scheme except the it uses forward feed to protect against meet-in-the-middle attack.

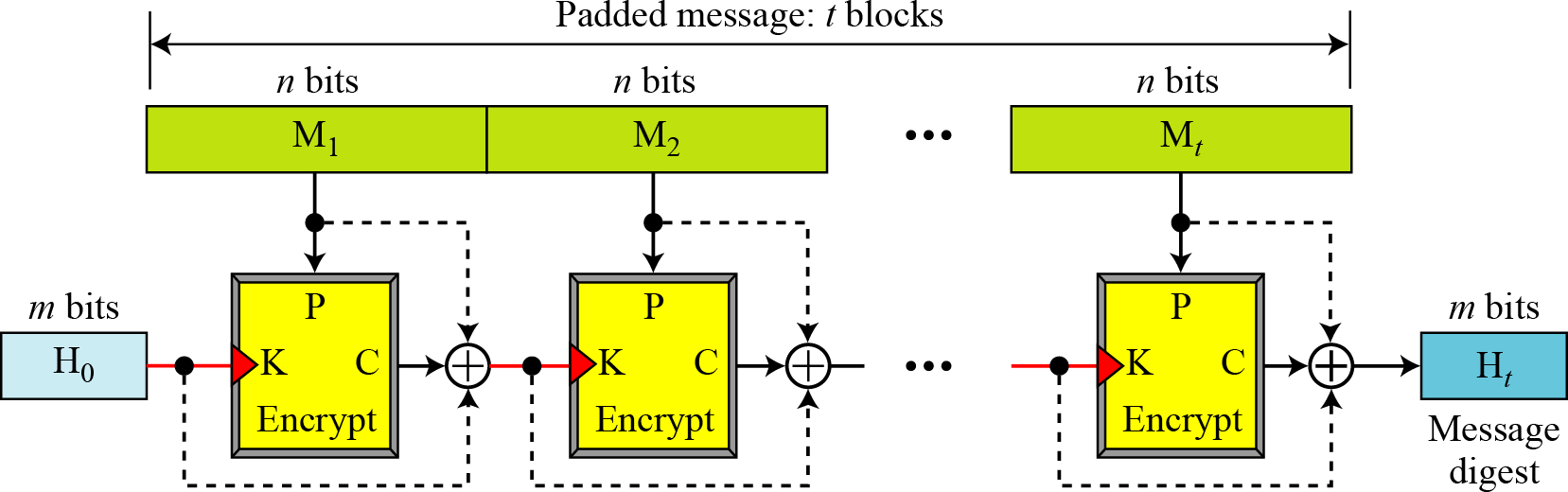


### Matyas-Meyer-Oseas Scheme

This scheme is a dual version of the Davis-Meyer scheme. The scheme can be used if the data block and the cipher key are the same size. *AES* is a good candidate for this purpose.



### Miyaguchi-Preneel Scheme

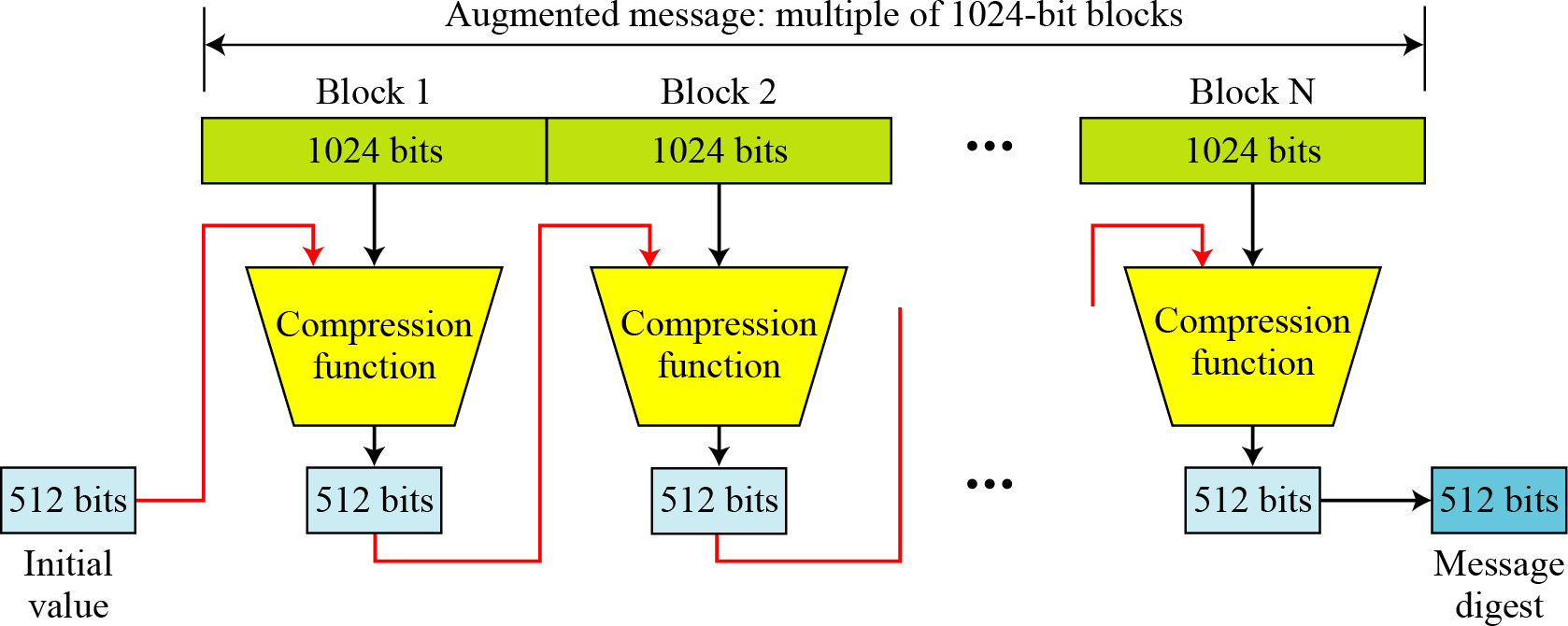
This scheme is an extended version of the Matias-Meyer-Oseas scheme. This is the scheme used by the Whirlpool hash function.

### ----------------------------------------------------------X----------------------------------------------------------

**2.3 SHA-512**

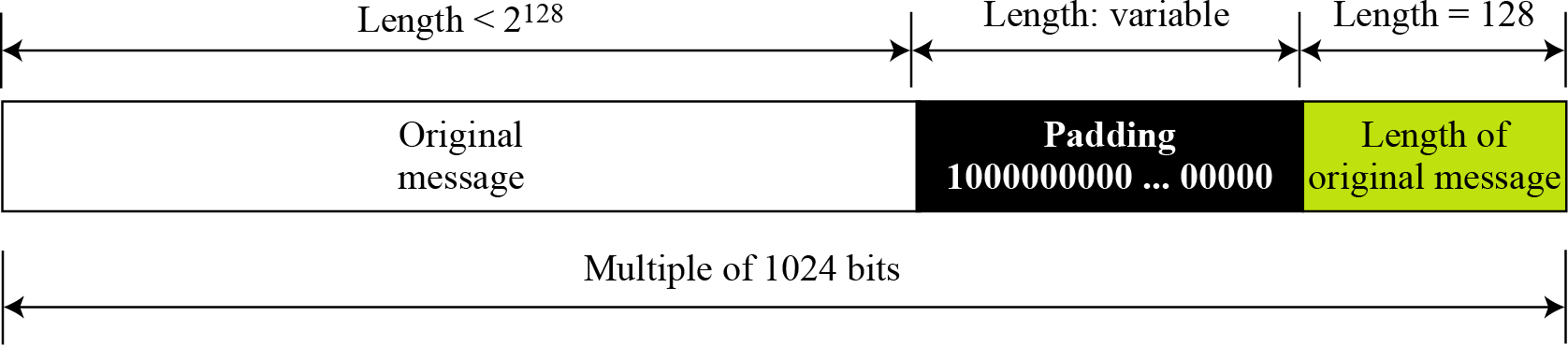
* + SHA-512 is the version of SHA with a 512-bit message digest. It is based on the Merkle- Damgard scheme. SHA-512 is the latest version of SHA family with a more complex structure than others and the longest message digest.
  + **Input:** a message with a maximum length of 2128 bit
  + **Output:** a 512-bit message digest.
  + The input is processed in 1024-bit blocks.

### Message Digest Creation in SHA-512:

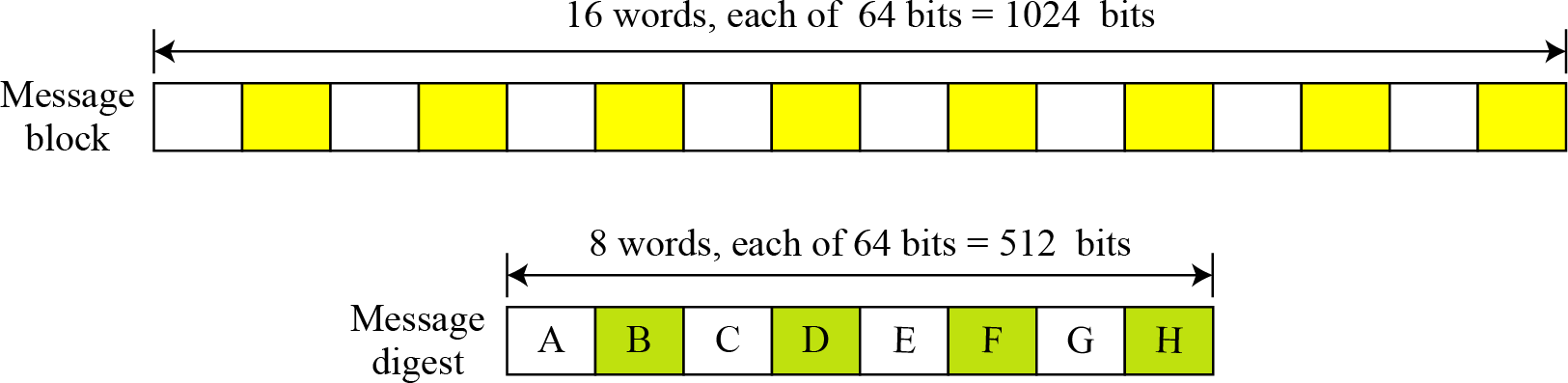


#### Message Preparation:

* + SHA-512 insists that the length of the original message be less than 2128 bits. Note: SHA-512 creates a 512-bit message digest out of a message less than 2128 ***Length Field and Padding:***
  + Before the message digest can be created, SHA-512 requires the addition of a 128-bit unsigned-integer length field to the message that defines the length of the message in bits.
  + The field can define a number between 0 and 2128-1.

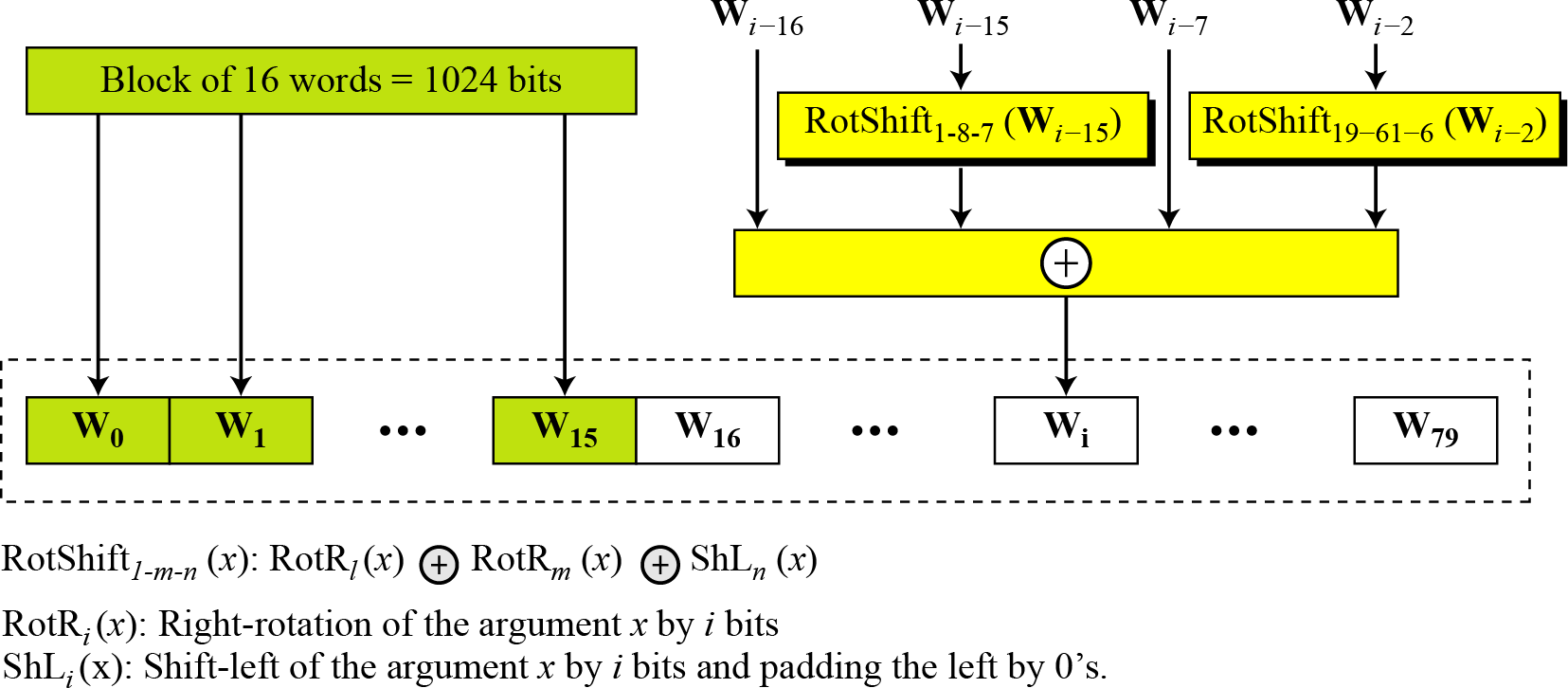


#### Words :

* + SHA-512 operates on words. Each word is defined as 64 bit. Each block of the message consists of sixteen 64-bit words.
  + The message digest consists of only eight words.

#### Word Expansion :

* Before processing, each message block (16 64-bit words) is expanded to 80 words.



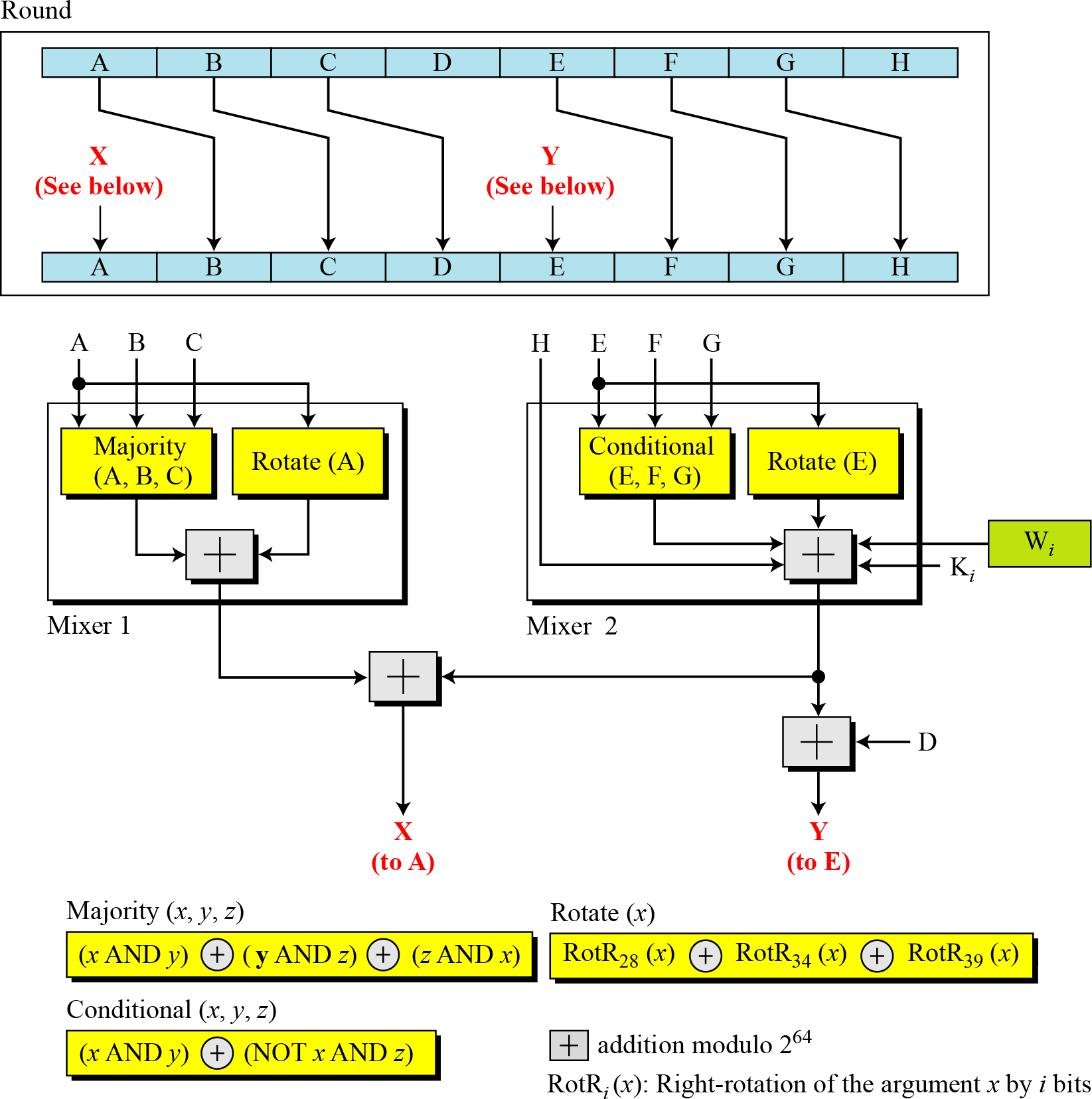
#### Message Digest Initialization : (ABCDEFGH)

* The constants are calculated from the first 8 primes (2, 3, 5, 7, 11, 17, and 19).
* Each value is the fraction part of *square root* of the corresponding prime number after converting to binary and keeping only the first 64 bits.

### Compression Function:

|  |  |
| --- | --- |
|  | * In ***each round,*** the contents of 8 previous buffers, one word (W*i*) from the expanded block, and one 64-bit constant (K*i*) are mixed together and then operated on to create a new set of 8 buffers. * Final adding between the output of 80th round and input of first round. |

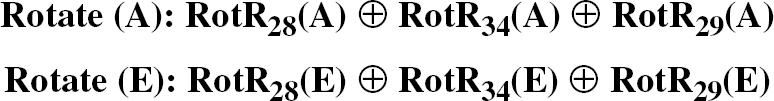
***Structure of each round in SHA-512:***



### Majority Function

This is a bitwise function. If two or three bits are 1’s, the resulting bit is 1; otherwise it is 0.

### Conditional Function

If E*j* then F*j*; else G*j*.

### Rotate Functions Constants:

There are 80 constants, K0 to K79, each of 64 bits. Similar to the initial values for the eight digest buffers, these values are calculated from the first 80 prime numbers (2, 3,…, 409).

Each value is the fraction part of the *cubic root* of the corresponding prime number.

### Analysis:

With a message digest of 512 bits, SHA-512 is expected to be resistant to all attacks, including collision attacks. It has been claimed that this version’s improved design makes it more efficient and more secure than the previous versions. However, more research and testing are needed to confirm this claim.

### ----------------------------------------------------------------X-------------------------------------------------------------

* 1. **WHIRLPOOL**
     + Whirlpool is designed by Vincent Rijmen and Paulo S. Barreto. It is endorsed by the New European Schemes for Signatures, Integrity, and Encryption (NESSIE).
     + Whirlpool is an iterated cryptographic hash function, based on the Miyaguchi-Preneel scheme, that uses a symmetric-key block cipher in place of the compression function.
     + The block cipher is a modified AES cipher that has been tailored for this purpose.

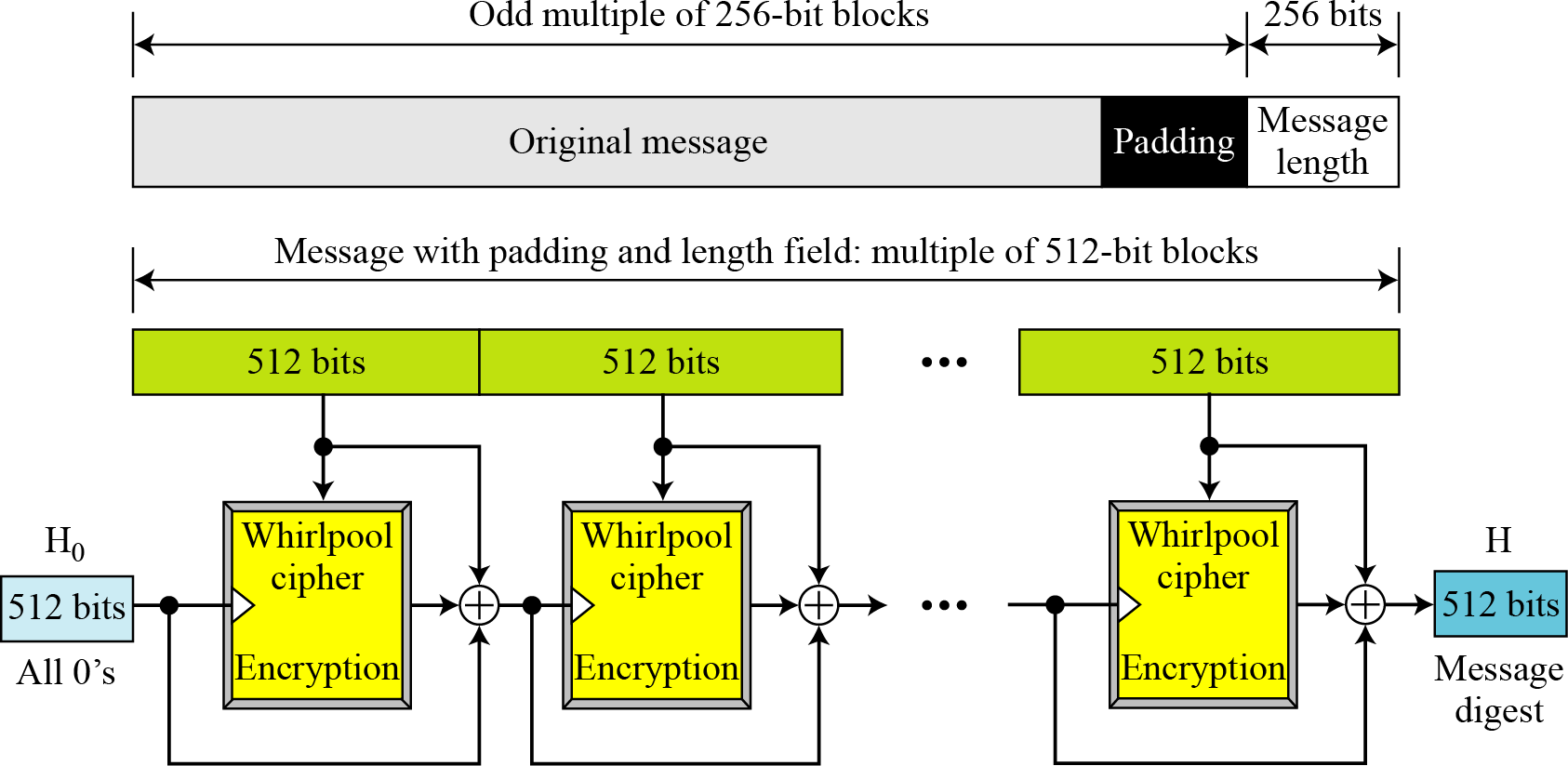
### Preparation

* + - Whirlpool requires that the length of the original message be less than 2 256 bits.
    - A message needs to be padded before being processed. The padding is a single 1 -bit followed by the necessary numbers of 0 -bits to make the length of the padding an odd multiple of 256 bits.
    - After padding, a block of 256 bits is added to define the length of the original message. This block is treated as an unsigned number. H 0 is initialized to all 0 ’s.
    - General idea of the Whirlpool cipher Whirpool cipher is a non-Feistel cipher like AES that was mainly designed as a block cipher to be used in a hash algorithm.

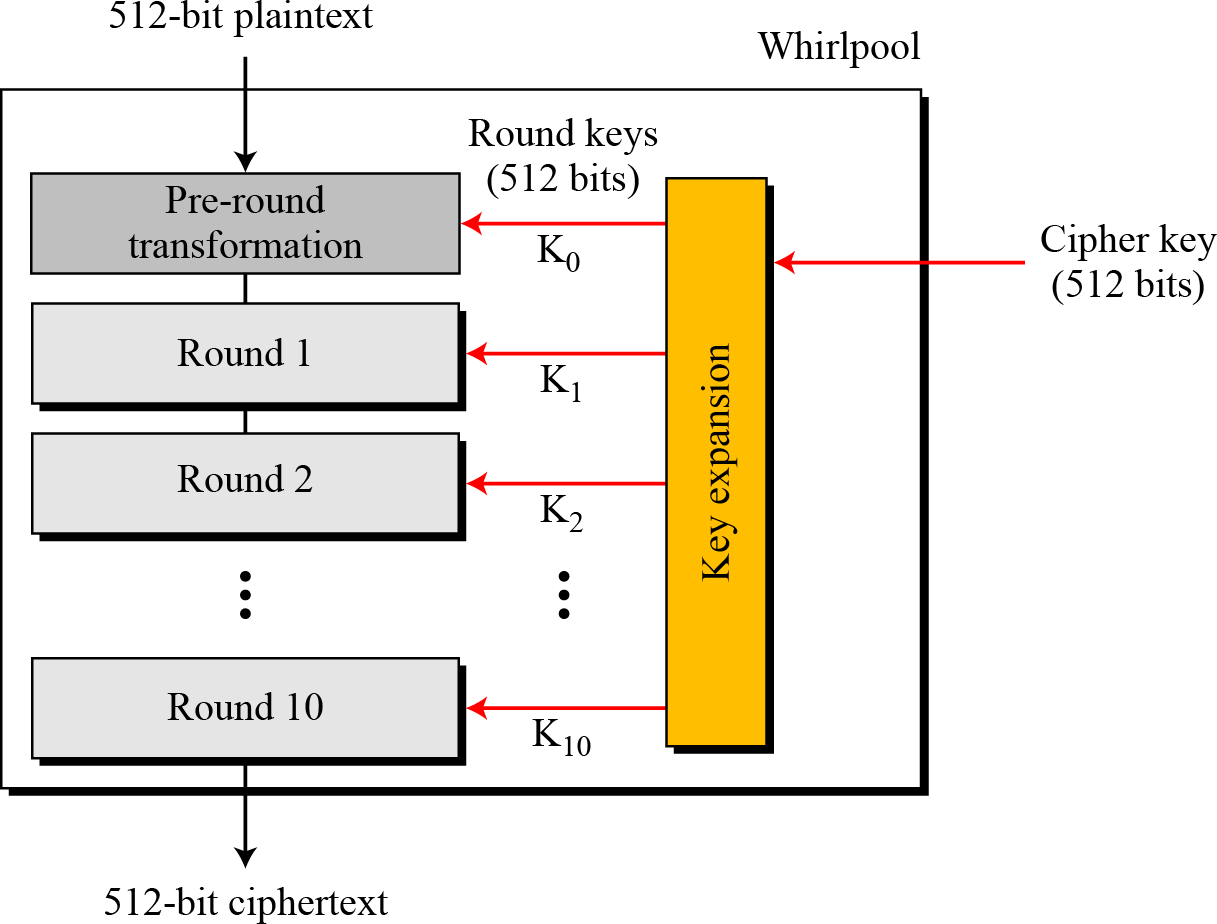
### Whirlpool hash function

* + - **Rounds**: Whirlpool is a round cipher uses 10 rounds. The block size and key size are 512 bits. The cipher uses 11 round keys, K0 to K10.
    - **States and Blocks:** Whirlpool uses states and blocks. The size of the state or block is 512 bits.

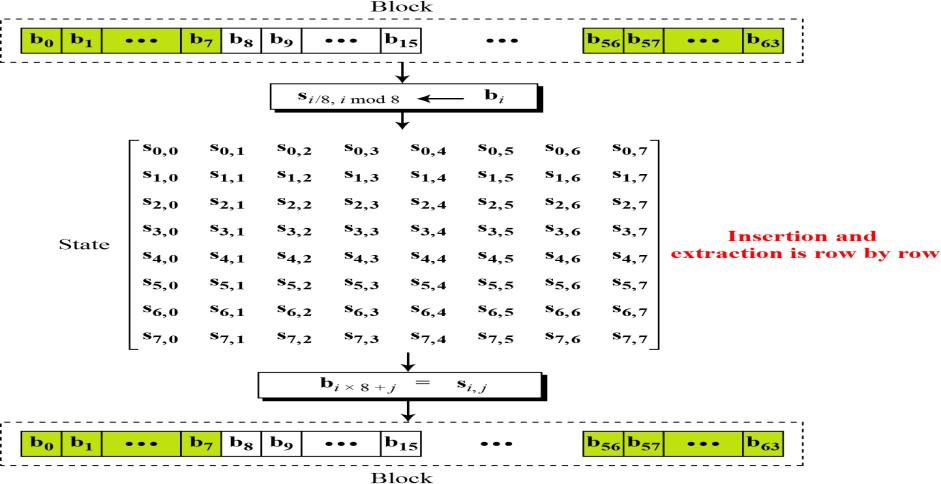
Block = row matrix of 64 bytes; State = square matrix of 8X8 bytes



### General idea of the Whirlpool Cipher:



#### Block and state in the Whirlpool cipher



***Structure of Each Round***

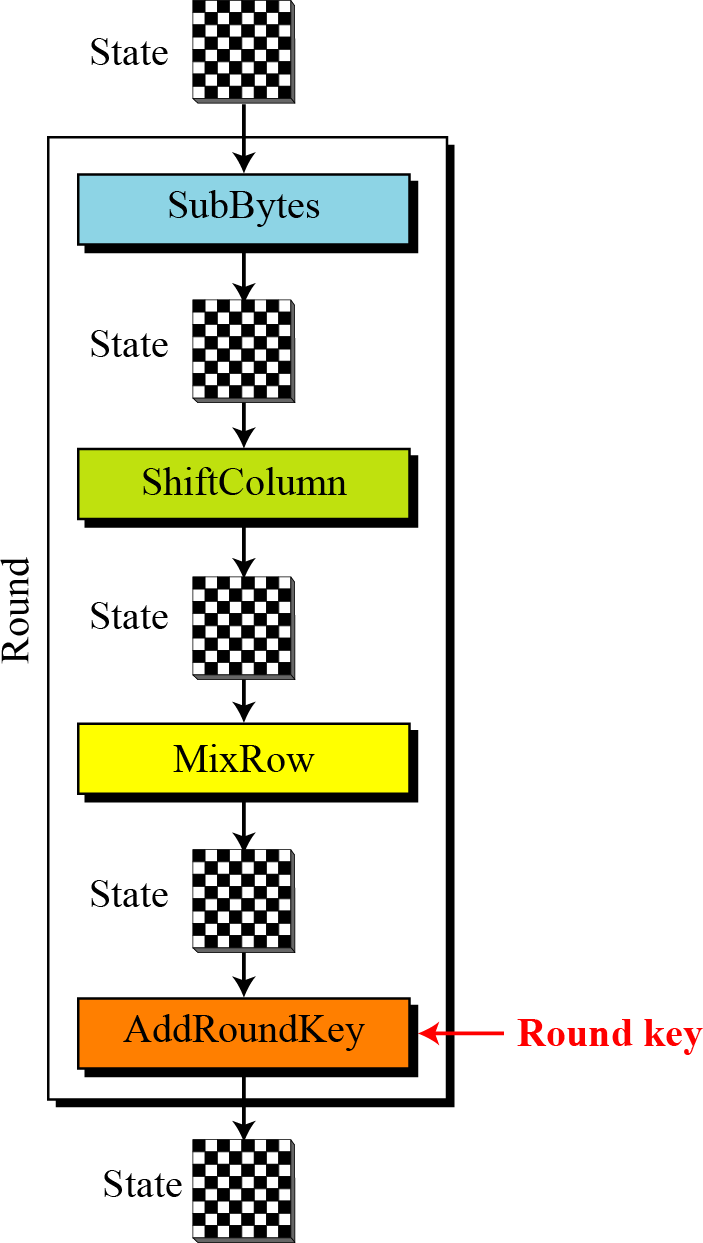
*Each round uses four transformations.*

### SubBytes

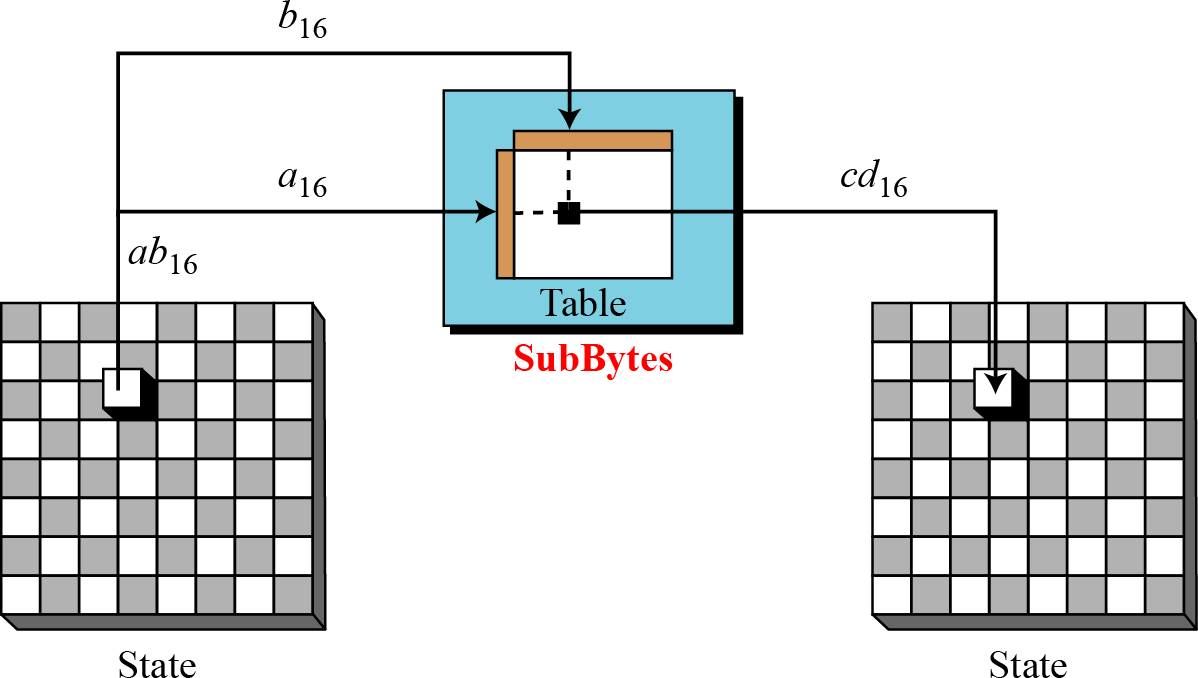
1. **ShiftColumns**

### MixRows

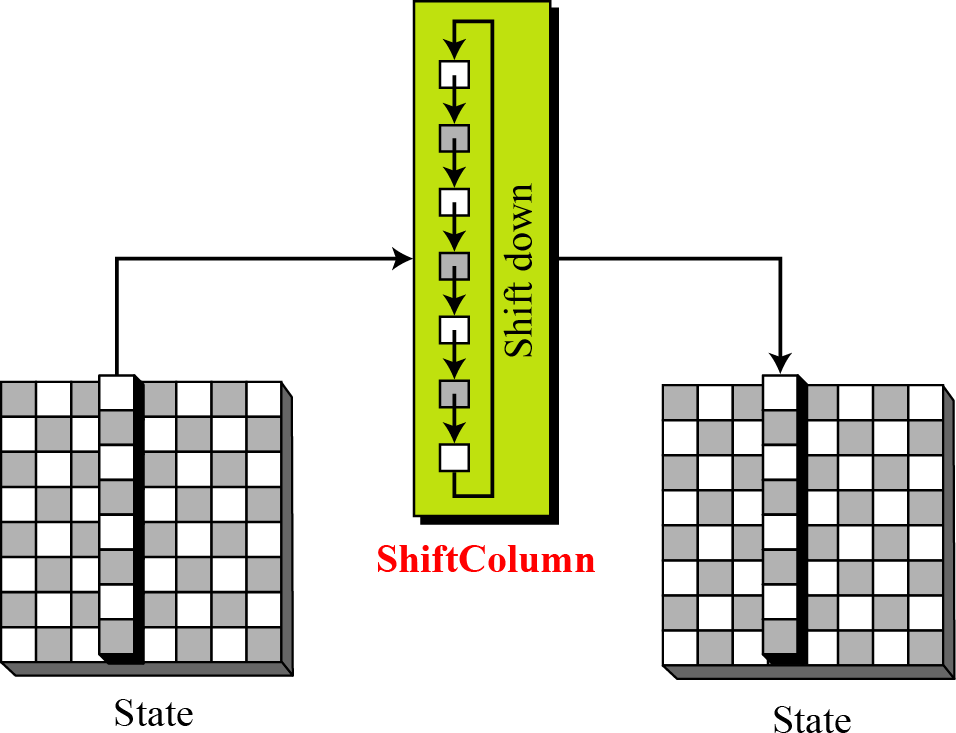
#### AddRoundKey



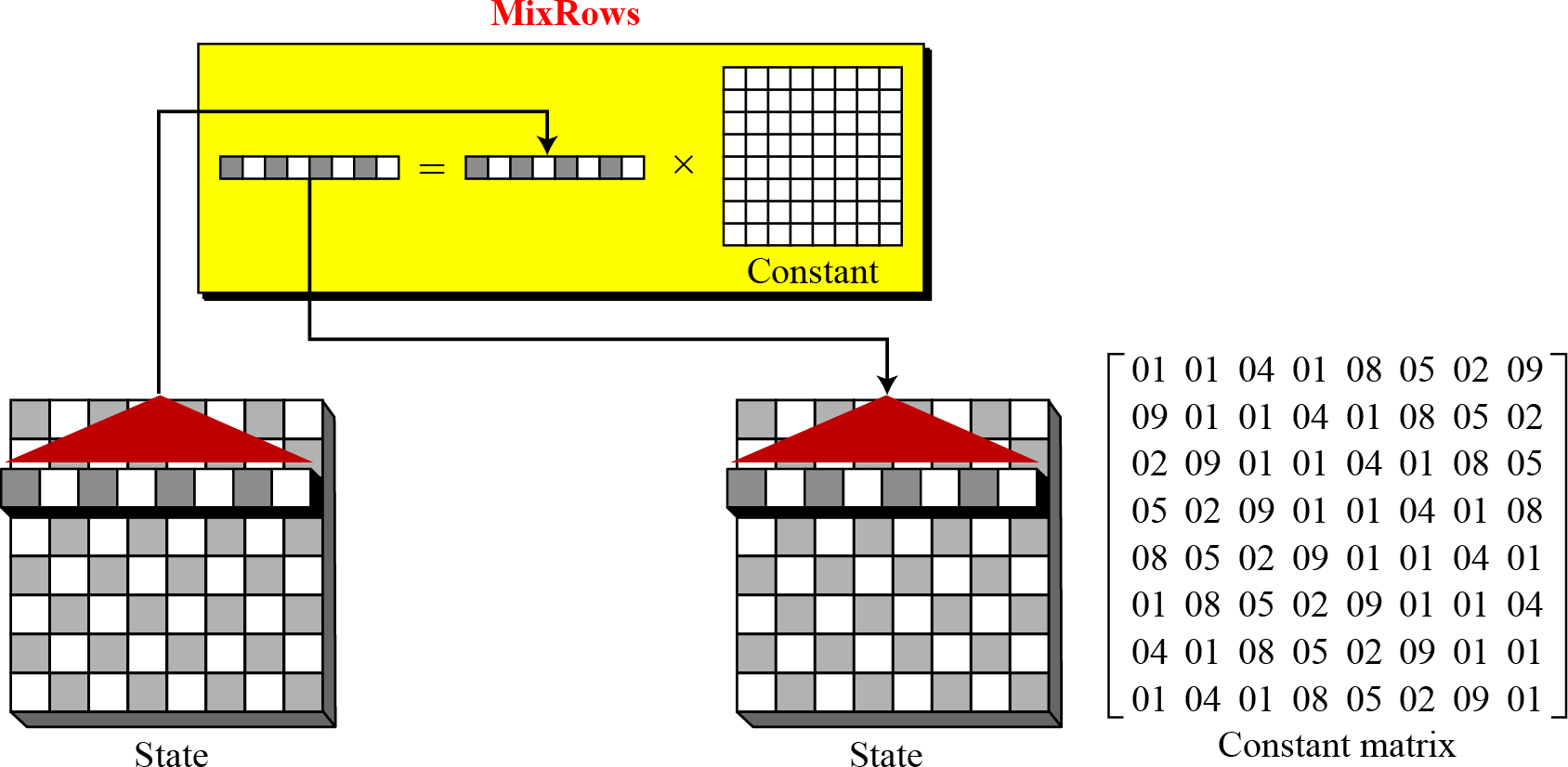
***SubBytes:*** *SubBytes provide a nonlinear transformation.*



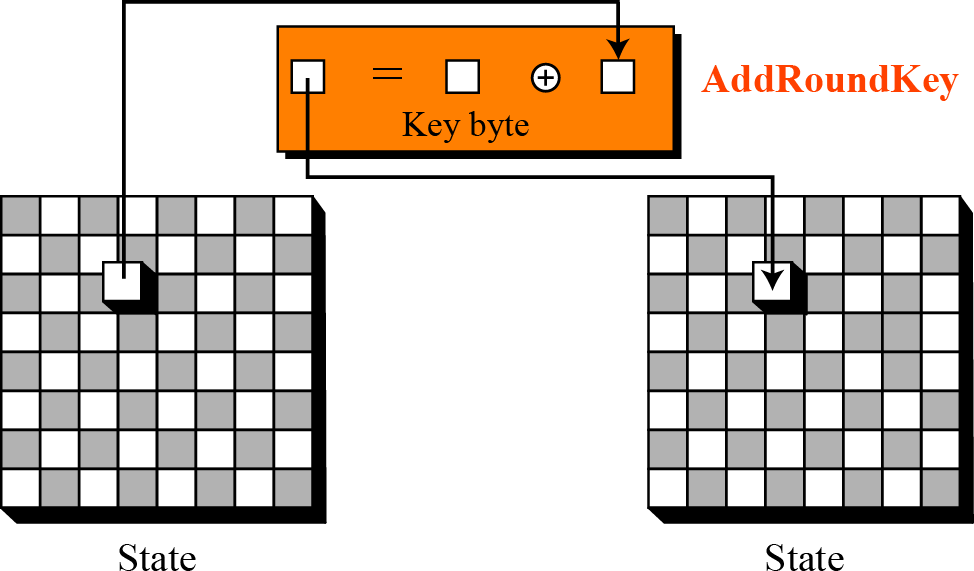
**ShiftColumns:** transformation in the Whirlpool cipher



**MixRows:** transformation in the Whirlpool cipher

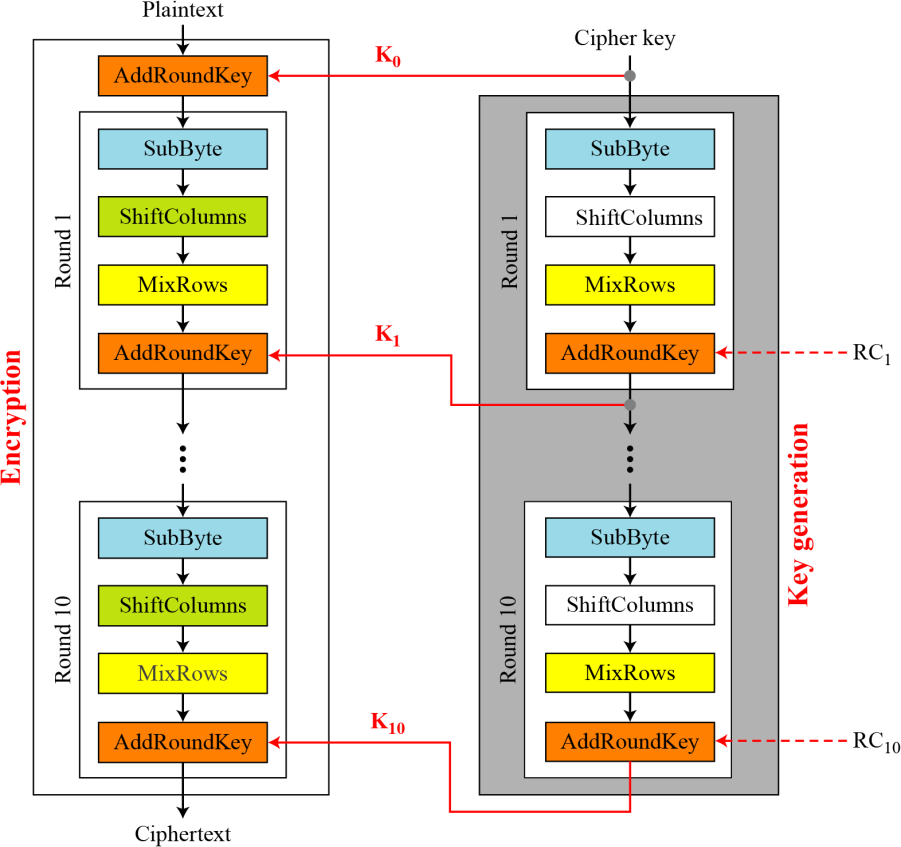


***AddRoundKey*** transformation in the Whirlpool cipher



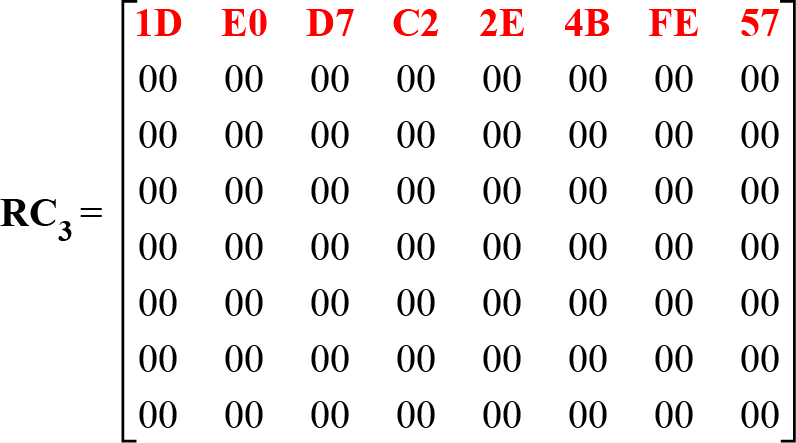
#### Key expansion in the Whirlpool cipher:

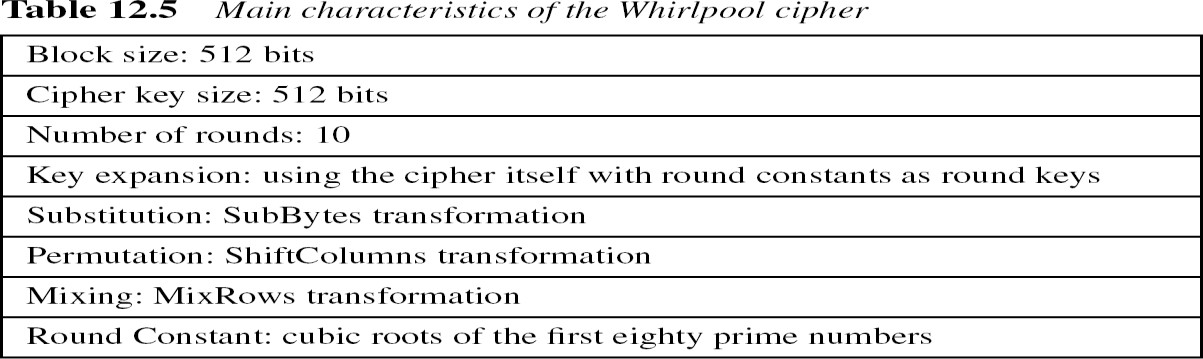
* + - AES encryption algorithm (without pre-round) is used as KEY-EXPANSION to create the round keys
    - 11 round keys are used : K0 to K11
    - Plaintext for algorithm is Cipher key = K0
    - The output of each round in the encryption algorithm is the round key for that round.
    - Remaining keys generated using key-expansion algorithm.



### Round constants:

* + - Each round constant RCr , is an 8X8 matrix in which first row has all non-zero values , and remaining has all 0’s.
    - Ex:





### Digital Signature

A digital signature is a mathematical scheme for verifying the authenticity of digital messages or documents. A valid digital signature, where the prerequisites are satisfied, gives a recipient very strong reason to believe that the message was created by a known sender (authentication), and that the message was not altered in transit (integrity).

A digital signature scheme typically consists of 3 algorithms:

* + - A **key generation** algorithm that selects a private key uniformly at random from a set of possible private keys. The algorithm outputs the private key and a corresponding public key.
    - A **signing algorithm** that, given a message and a private key, produces a signature.
    - A **signature verifying algorithm** that, given the message, public key and signature, either accepts or rejects the message's claim to authenticity.

Two main properties are required. **First**, the authenticity of a signature generated from a fixed message and fixed private key can be verified by using the corresponding public key. **Secondly**, it should be computationally infeasible to generate a valid signature for a party without knowing that party's private key.

A digital signature is an authentication mechanism that enables the creator of the message to attach a code that acts as a signature.

X

### 3.2.The differences between conventional signatures and digital signatures: Inclusion:

A conventional signature is included in the document; it is part of the document. We sign a document digitally, we send the signature as a separate document.

**Verification Method:**

For a conventional signature, when the recipient receives a document, she compares the signature on the document with the signature on file.

For a digital signature, the recipient receives the message and the signature. The recipient needs to apply a verification technique to the combination of the message and the signature to verify the authenticity.

**Relationship:**

For a conventional signature, there is normally a one-to-many relationship between a signature and documents. For a digital signature, there is a one-to-one relationship between a signature and a message.

**Duplicity:**

In conventional signature, a copy of the signed document can be distinguished from the original one on file.

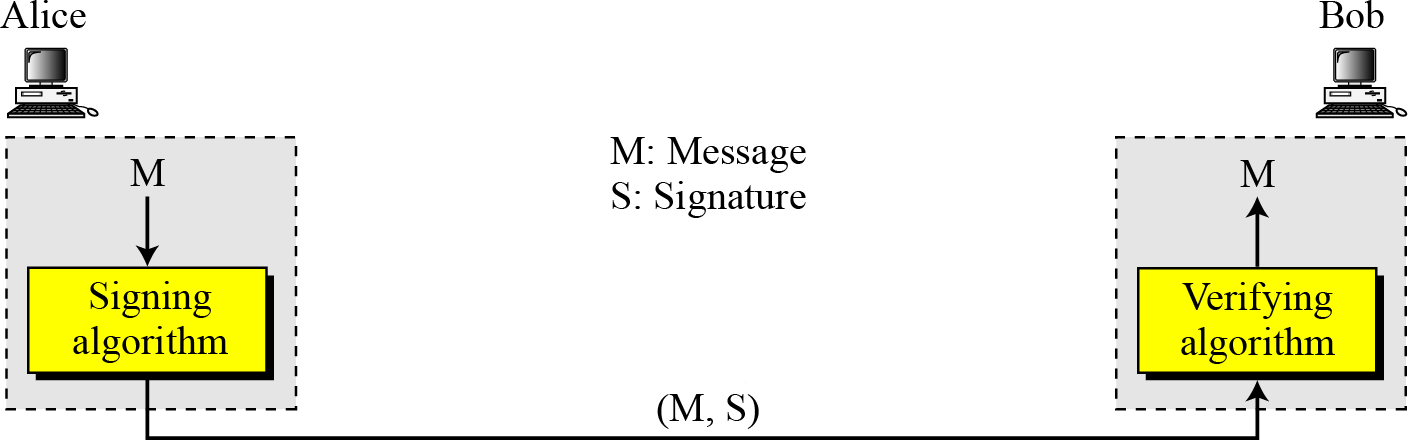
In digital signature, there is no such distinction unless there is a factor of time on the document.

X

### PROCESS OF CREATING DIGITAL SIGNATURE

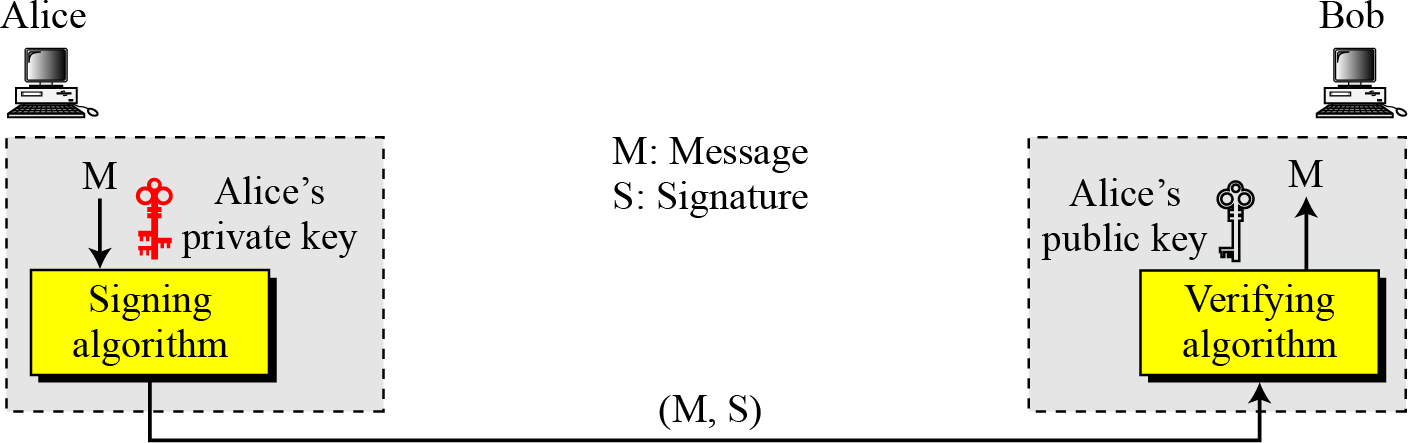
The sender uses a **signing algorithm** to sign the message. The message and the signature are sent to the receiver. The receiver receives the message and the signature and applies the **verifying algorithm** to the combination. If the result is true, the message is accepted; otherwise, it is rejected.

### Digital signature process



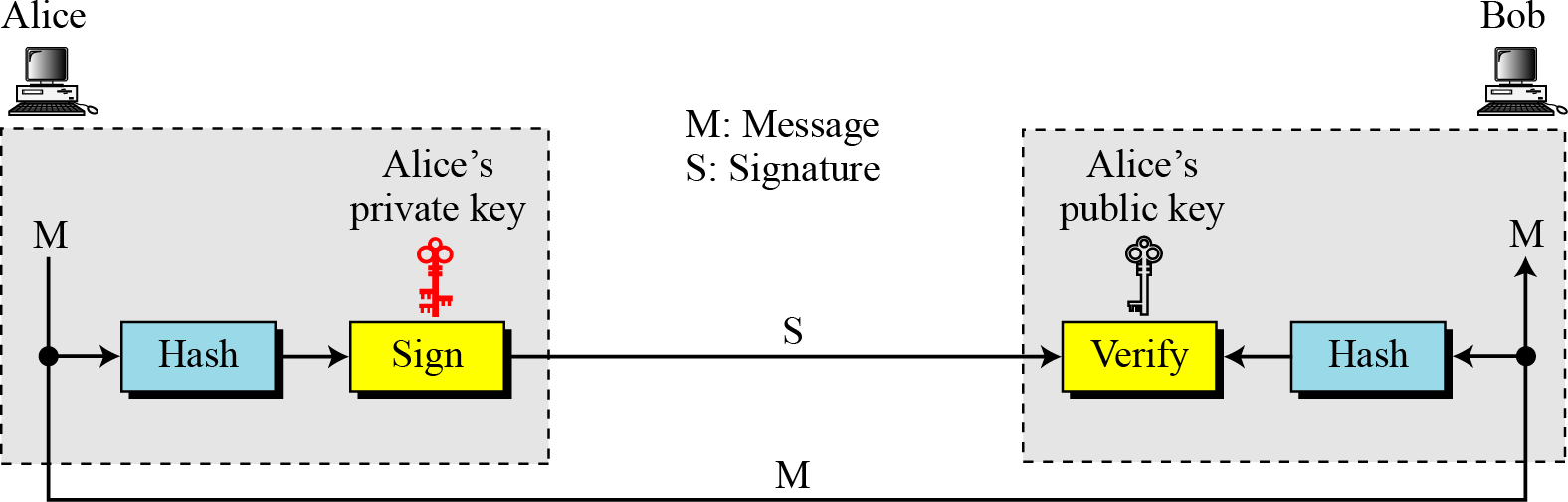
***Need for Keys:***

A conventional signature is like a private “key”, belonging to the signer of the document.



A digital signature needs a public-key system. The signer signs with her private key; the verifier verifies with the signer’s public key. A cryptosystem uses the private and public keys of the receiver: a digital signature uses the private and public keys of the sender.

***Signing the Digest:***



X -

### DIGITAL SIGNATURE SERVICES

Security services include message confidentiality, message authentication, message integrity, and non-repudiation. A digital signature can directly provide the last three; for message confidentiality we still need encryption/decryption.

1. *Message Authentication:*

*A secure digital signature scheme, like a secure conventional signature can provide message authentication.* A digital signature provides message authentication.

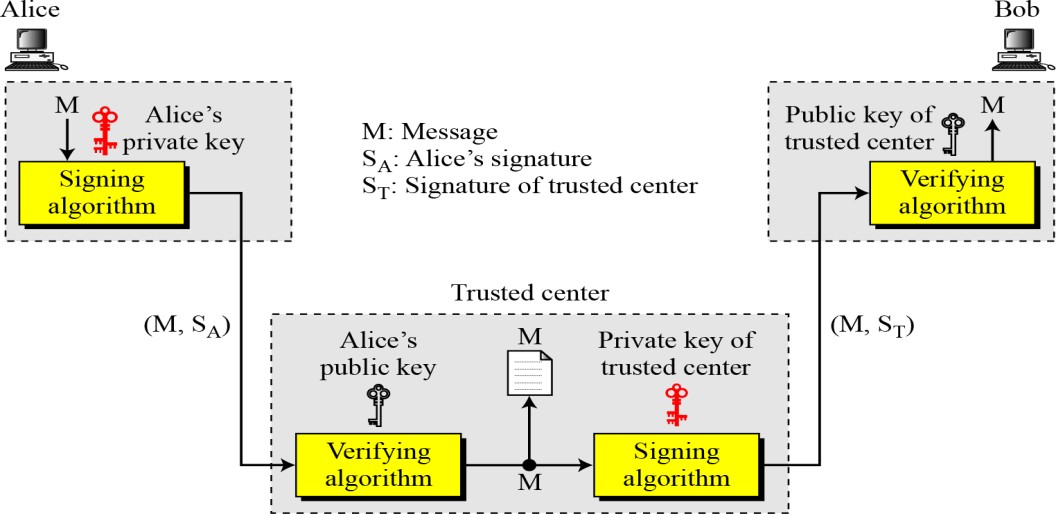
1. *Message Integrity:*

*The integrity of the message is preserved even if we sign the whole message because we cannot get the same signature if the message is changed.*

A digital signature provides message integrity.

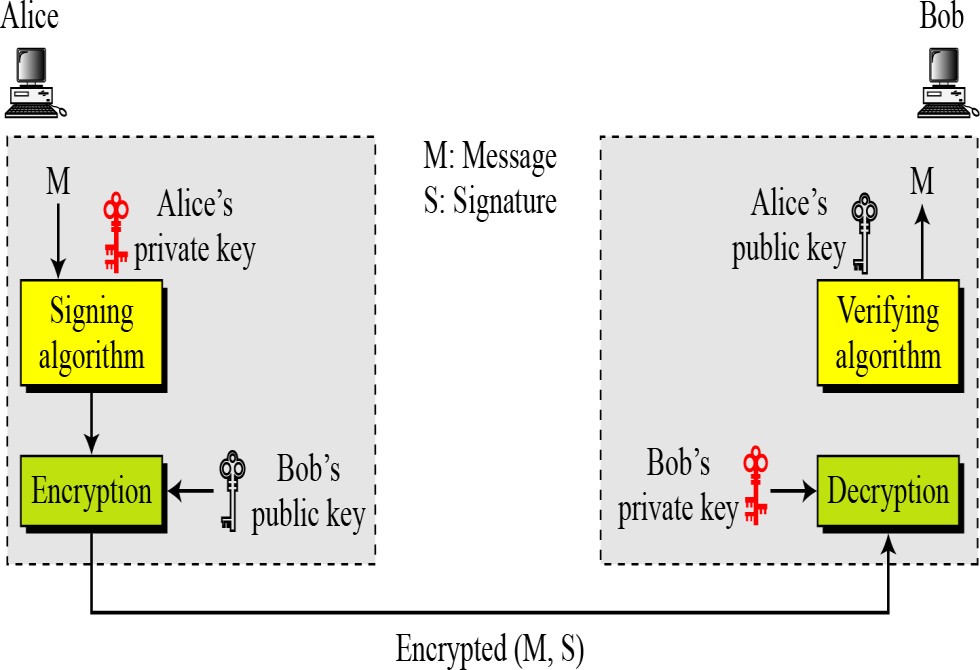
1. *Non-repudiation:*

*Using a trusted center for nonrepudiation*



Nonrepudiation can be provided using a trusted party.

1. *Confidentiality:*



A digital signature does not provide privacy.

If there is a need for privacy, another layer of encryption/decryption must be applied.

X

### ATTACKS ON DIGITAL SIGNATURE

Attack Types

1. *Key-Only Attack*

*Eev has access only public information released by Alice. To forge a message, Eve needs to create Alice's signature to convince Bob that the message is coming from Alice. This is similar to ciphertext-only attack.*

1. *Known-Message Attack*

*Eve has acedd to one or more message-signature pairs. In other words, she has access to some documents previously signed by Alice. Eve tries to create another message and forge Alice's*

*signature on it. This is similar to known-plaintext attack.*

1. *Chosen-Message Attack*

*Eve somehow makes Alice sign one or more messages for her. Eve nw has a chosen- message/signature pair. Eve later creates another message with the content she wants, and forges Alice's signature on it. This is similar to the chosen-palintext attack.*

#### Forgery Types

If the attack is successful, the result is forge. We can have two types of forgery:

#### Existential Forgery

Eve may able to create a valid message-signature pair, but not one that she can really use. In other words, a document has been forges, but the content is randomly calculated.

1. Selective Forgery

Eve may be able to forge Alice's signature on a message with the content selectively chosen be Eve. Although this is beneficial to Eve, and may be very detrimental to Alice, the probability of such forgery is low but not negligible.

X

### DIGITAL SIGNATURE SCHEMES

*Several digital signature schemes have evolved during the last few decades. Some of them have been implemented.*

* + 1. RSA Digital Signature Scheme
    2. ElGamal Digital Signature Scheme
    3. Schnorr Digital Signature Scheme
    4. Digital Signature Standard (DSS)
    5. Elliptic Curve Digital Signature Scheme
    6. RSA Digital Signature Scheme
       - RSA can also be used for signing and verifying a message. In this case it is called as **RSA digital signature scheme.**
       - The Digital signature scheme changes the roles of the private and public keys:

1. The Private and public keys of only the sender are used not the receiver
2. The Sender uses her own private key to sign the document and the receiver uses the sender’s public key to verify the document.
   * + - The ***signing*** and ***verifying*** sets use the same function, but with different parameters.
       - The verifier compares the message and the output of the function for **congruence**. If the result is true the message is accepted.

### Key generation in RSA

Key generation in RSA digital signature scheme is exactly the same as key generation in the RSA cryptosystem:

1. Select two large prime numbers p and q; where p≠q.
2. Compute n = p.q
3. Calculate ***ø(n)=(p–1)(q-1)***
4. Select integer **e, where gcd(*ø(n),* e)=1; 1<e< *ø(n)***
5. Calculate d : ***e.d=1 mod ø(n) or d*** ***e-1 (mod ø(n) )***
6. Public key, PU: {e,n}
7. Private key, PR: {d,n}

In the RSA digital signature scheme, d is private; e and n are public.

### Working of RSA digital signature scheme:

**Signing:**

Sender **Alice** wants to send a message **M** to the receiver **Bob** along with the digital signature **S**

calculated over the message **M**

1. Alice creates the message M.
2. Alice creates a signature using the private key d. i.e **S** = **Md** mod ***n***
3. Alice sends the message and signature to Bob.

### Verifying:

The recipient, Bob, receives M and S and performs the following:

1. Bob applies Alice’s public key to the signature to create a copy of the message M’ = Se mod n.
2. Bob compares the value of M’ with the value of M. If the two values are congruent, Bob accepts the message.

Proof:

M’ = Se mod n ---> Se  **M** mod ***n --->* Md****e** mod ***n***

### The last congruent holds d x e = 1 mod *ø(n)*

***----------------------------------X--------------------------------***

### Attacks on RSA Signature:

There are some attacks that Eve can apply to the RSA digital signature scheme to forge Alice’s signature.

* Key-only Attack: Eve has access only to Alice’s public key. Eve intercepts the pair(M,S) and tries to create another message M’ such that **M ‘** Se mod ***n***
* Known Message Attack
  + Eve uses multiplicative property of RSA.
  + Assume that Eve intercepts message pairs (***M*1**, ***S*1**) and (***M*2**, ***S*2**)
  + If ***M´*** = (***M*1**  ***M*2 ) mod n then**

Corresponding signature: ***S´*** = ***S*1**  ***S*2**

* + Idea: ***S´*** = ***S*1**  ***S*2** = (***M*1d**  ***M*2d**) mod ***n***

= (***M*1**  ***M*2**)**d** mod ***n***

= ***M´* d** mod ***n***

* + Eve now has fake message ***M´*** and matching signature ***S´*** without having to know

Alice’s private key!

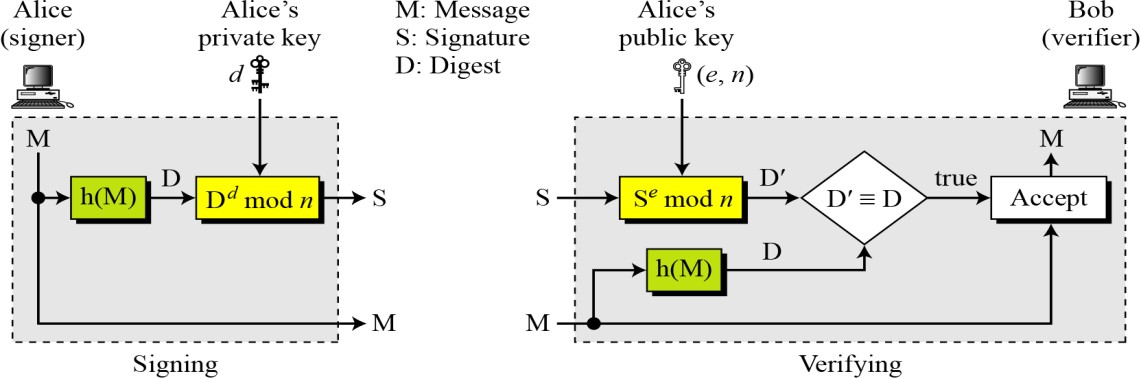
* + **Goal**: Create fake message ***M´*** and legitimate corresponding signature from those previous messages
* Chosen Message Attack
  + Adversary has ability to make sender sign messages that adversary chooses (“We like kittens”)
  + Goal: Choose those messages to make it possible to create fake message ***M´*** and legitimate corresponding signature

***----------------------------------X--------------------------------***

### RSA Signature on Message Digest

RSA can also be used to create a signature on Message Digest. To create a digest, it requires Hash algorithm.

Key generation is same as the key generation process in RSA cryptosystem.



**Signing**: Alice wants to send message to Bob and performs the following steps after creating keys:

1. Alice creates a message M.
2. Uses hash algorithm to create Message digest, D.
3. Creates a signature S, using private key, d. i.e **S** = D**d** mod ***n***
4. Alice sends the message and signed digest, (M,S), to Bob.

### Verifying:

The recipient, Bob, receives M and S and performs the following:

1. Bob applies Alice’s public key to the signature to create a copy of the message D’ = Se mod n.
2. Then, applies hash function on the message received, M, to create a digest, D.
3. Bob compares the value of D’ with the value of D. If the two values are congruent, Bob accepts the message, M.

Note:

When the digest is signed instead of the message itself, the susceptibility of the RSA digital signature scheme depends on the strength of the hash algorithm.

X

### ElGamal signature scheme

The ElGamal signature scheme is a digital signature scheme based on the algebraic properties of modular exponentiation, together with the discrete logarithm problem. The algorithm uses a key pair consisting of a public key and a private key. The private key is used to

generate a digital signature for a message, and such a signature can be verified by using the signer's corresponding public key.

The digital signature provides message authentication (the receiver can verify the origin of the message), integrity (the receiver can verify that the message has not been modified since it was signed) and non-repudiation (the sender cannot falsely claim that they have not signed the message).

The ElGamal signature scheme was described by Tahir Elgamal in 1985.

The scheme involves 3 operations:

1. Key generation (which creates the key pair),
2. Signing and
3. Signature verification.

### Key Generation :

The key generation procedure here is exactly the same as the one used in the cryptosystem.

1. Select a large prime number, ‘p’ and primitive root of a prime number ‘e1’
2. Generate a random integer 'd', such that 1 < d < p-1 .
3. Calculate e2 = e1d mod p.
4. Public key: (e1, e2, p), Private Key: ‘d’

In ElGamal digital signature scheme, (e1, e2, p) is Alice’s public key; d is her private key.

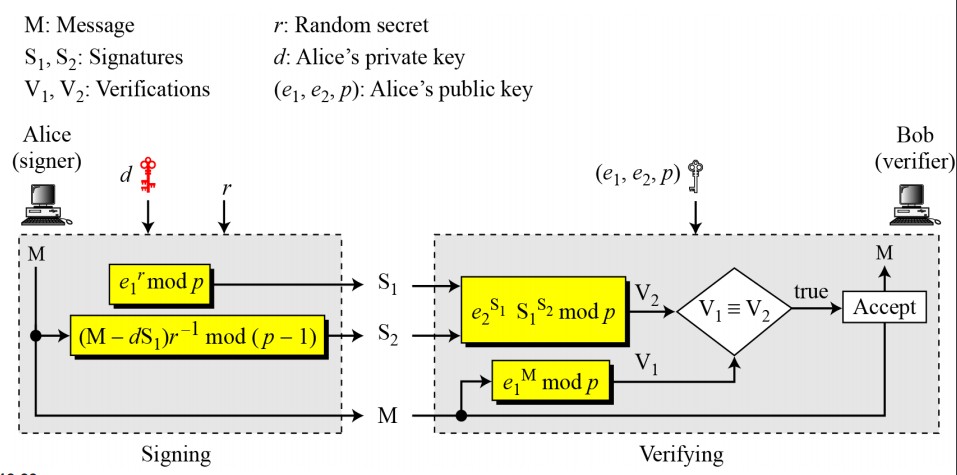
**Signing**: Alice can sign the digest of a message to any entity, including Bob:

1. Alice Choose a secret random number ‘r’. Note that although public and private keys can be used repeatedly, Alice needs a new ‘r’ each time she sign a new message
2. Alice calculates the first signature S1 = e1r mod p.
3. Alice calculates the second signature S2 = (M-Dxs1)r-1 mod (p-1), where r-1 is the multiplicative inverse of **r modulo p.**
4. Alice sends M, S1, and S2 to Bob.

**Verifying:** An entity such as Bob, receives M, S1 and S2, which can be verified as follows:

1. Bob checks to see if 0 < S1 < p
2. Bob checks to see if 0 < S2 < p-1
3. Bob calculates V1 = e1M mod p
4. Bob calculates V2 = e2s1 X S1S2mod p
5. If V1 is congruent to V2, the message is accepted. Otherwise, it is rejected. We can prove that verification criterion using e2 = e1d and S1 = e1r

V1



### -------------------------------------X----------------------------------------

**Forgery in ElGamal digital signature scheme:**

### This scheme is vulnerable to existential forgery, but it is hard to do a selective forgery.

1. **Key-only Forgery:** Two kind of forgery are possible: Eve has a predefined message M.

To forge Alice’s signature on it, Eve must find two valid signatures S1 and S2 for this message.This is a selective forgery.

* 1. Eve can choose S1 and calculates S2.

She needs to have *e*2S1S1S2 *≡ e*1M (mod *p*),

i.e., S1S2 *≡ e*1M *e*2-S1 (mod *p*), or S2 *≡* logS1 (*e*1M *e*2-S1 ) (mod *p*).

This means computing the discrete logarithm.

* 1. Eve can choose S2 and calculates S1. This is much harder than case a.

1. **Known-Message Forgery:** If Eve has intercepted a message M and its two signatures S1 and S2, she can find another message M’, with the same pair of signatures S1 and S2.

X

### Schnorr Digital Signature Scheme:

**Key Generation:** Before signing the message, Alice needs to generate keys and announce the public ones to the public:

1. Alice selects a prime p, |p|= 1024 bits
2. Alice selects another prime q.
3. Alice chooses e1 to be the qth root of 1 modulo p.
4. Alice chooses an integer, d, as her private key.
5. Alice calculates e2 = e1d mod p.
6. Alice’s public key is (e1, e2, p, q); her private key is (d).

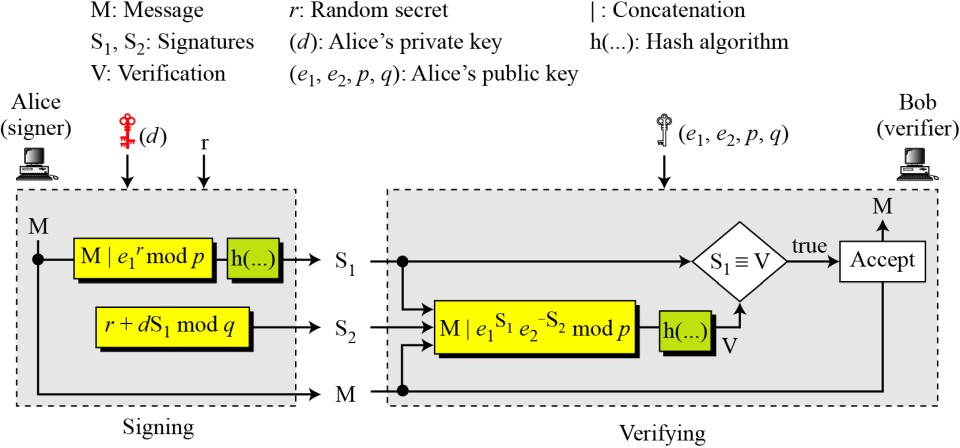
In the Schnorr digital signature scheme, Alice’s public key is (e1, e2, p, q); her private key (d).

### Signing:

1. Alice chooses a random number r.
2. Alice calculates S1 = h(M | e1r mod p).
3. Alice calculates S2= r + d × S1 mod q.
4. Alice sends M, S1, and S2.

### Verifying Message:

1. Bob calculates V = h (M | e1S2 e2−S1 mod p).
2. If S1 is congruent to V modulo p, the message is accepted; otherwise, it is rejected.



Forgery on Schnorr Signature Scheme:

X

### Digital Signature Standard:

The Digital Signature Standard (DSS) was adopted by the National Institute of Standards and Technology (NIST) in 1994. DSS uses a Digital signature algorithm (DSA) based on ElGamal Scheme.

In this signing process, two functions create signatures, in verifying process, the output of one function is compared to the first signature for verification.

**Key Generation:** Before signing a message to any entity, Alice needs to generate keys and announce the public ones to the public.

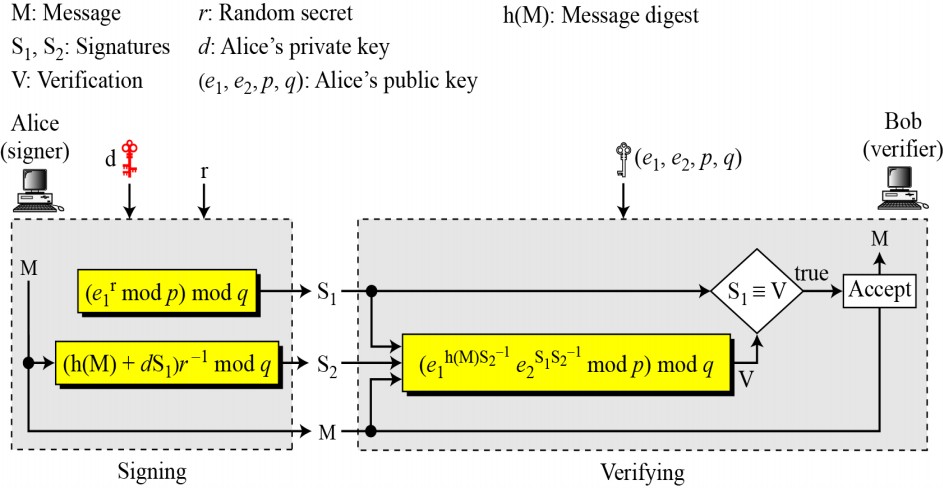
1. Alice chooses primes p and q, between 512 and 1024 bits in length. The number of bits in p must be multiples of 64.
2. Alice chooses a 160-bit prime q in such a way that q divides (p-1)
3. Alice uses two multiplication subgroups <Zp\*, × > and <Zq\*, ×>.
4. Alice creates e1 to be the qth root of 1 modulo p (e1p = 1 mod p)
5. Alice chooses d and calculates e2 = e1d.
6. Alice’s public key is (e1, e2, p, q); her private key is (d).

### Signing:

1. Alice chooses a random number r. 1<r<q
2. Alice calculates S1= (e1r mod p) mod q.
3. Alice create the digest h(M).
4. Calculates S2= (h(M) + d × S1) r-1 mod q.
5. Alice sends M, S1, and S2 to Bob.

### Verifying Message:

1. Bob checks to see if 0 < S1 < q
2. Bob checks to see if 0 < S2 < q
3. Bob calculates a message digest of M using the same hash algorithm used by Alice.
4. Bob calculates V = [(e1h(M)S2-1 e2S2S1-1) mod p] mod q.
5. If S1 is congruent to V modulo p, the message is accepted;



X

### Elliptic Curve Digital Signature Scheme:

The Elliptic Curve Digital Signature Algorithm (ECDSA) offers a variant of the Digital Signature Algorithm (DSA) which uses elliptic curve cryptography.

### Key generation follows these steps:

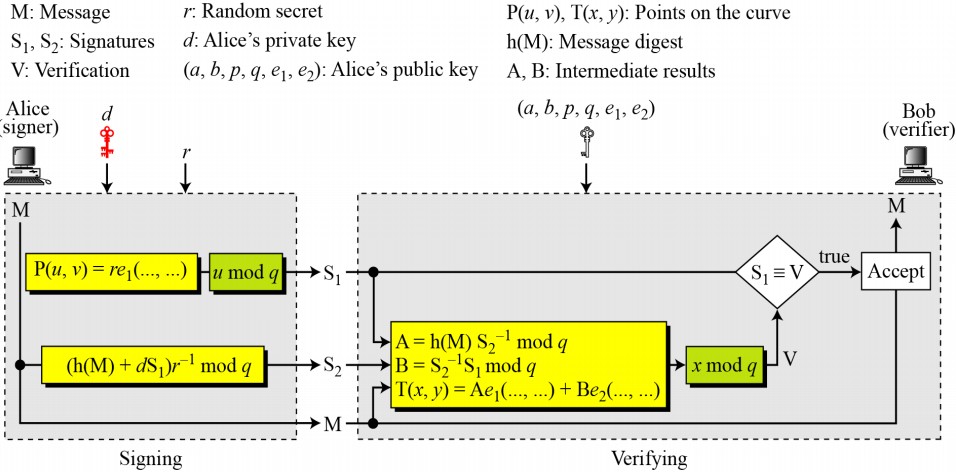
1. Alice chooses an elliptic curve Ep(a, b) with p a prime number.
2. Alice chooses another prime q
3. Alice chooses the private key d.
4. Alice chooses e1(…, …), a point on the curve.
5. Alice calculates e2(…, …) = d × e1(…, …), another point on the curve.
6. Alice’s public key is (a, b, p, q, e1, e2); her private key is d.

### Signing:

* 1. Alice chooses a random number r. a. 1<r<q-1
  2. Alice calculates a third point on the curve, P(u,v)=r × e1(…,…).
  3. Alice uses first coordinates of P(u,v) to calculate first signature S1. S1= u mod q.
  4. Alice create the digest h(M).
  5. Calculates S2 = (h(M) + d × S1) r-1 mod q.
  6. Alice sends M, S1, and S2to Bob.

### Verifying Message:

1. Bob create A=h(M) S2-1 mod q.
2. Bob computes B= S2-1S1 mod q .
3. Bob reconstructs T(x,y) = A X e1(…,…) ＋ B Xe2(…,…)
4. If S1 is congruent to x modulo q, themessage is accepted; otherwise, it is rejected.



# Key Management and distribution

* Key Management and distribution includes key generation, distribution and maintenance.
* There are two types of keys to be distribute
  1. Secrete key – symmetric encryptions require both parties to share a common secret key
  2. Public key – used in asymmetric encryption
* Public-key encryption helps to address **key distribution problems**

1. **Secret key distribution**

**Secret key is distributed using two techniques**

### Symmetric encryption

It consists of three methods…….

* + 1. Centralized key distribution
    2. Automatic key distribution
    3. Decentralized key distribution

### Asymmetric encryption

* + 1. Simple secret key distribution
    2. secret key distribution with confidentiality and authentication

1. ***Secret Key distribution using Symmetric encryption***

### There are number of ways to distribute the secret key to both the parties A and B

* 1. A can select key and physically deliver to B
  2. Third party can select & deliver key to A & B
  3. If A & B have communicated previously can use previous key to encrypt a new key
  4. If A & B have secure communications with a third party C, C can relay key between A & B

Option 1 & 2 delivers a key manually. These methods are not suitable for large systems.

In option 3, if an attacker succeeds in gaining access to one key, then all subsequent keys will be revealed.

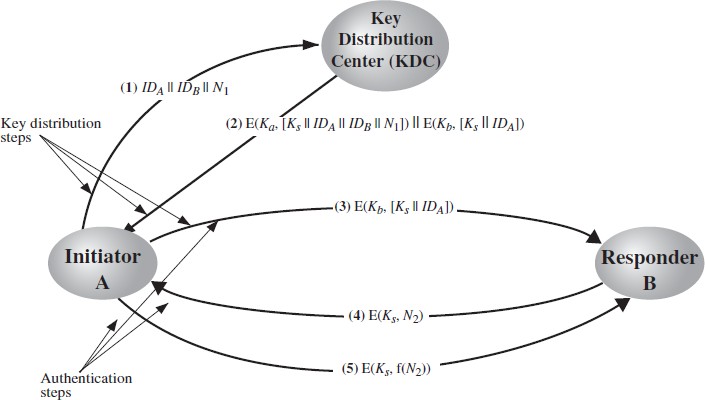
Option 4 has been widely adopted. In this scheme, a key distribution center is responsible for distributing keys to pairs of users. Each user must share a unique key with the key distribution center for purposes of key distribution.

## Centralized key distribution

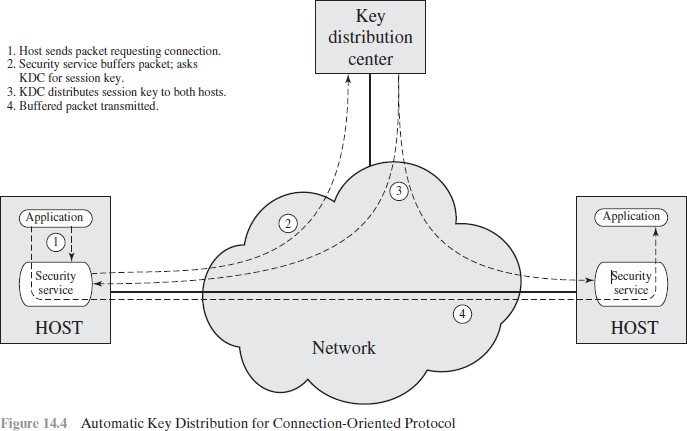
* + In this, each user shares a unique master key with the Key Distribution Center (KDC).
  + **Let A** shares a master key, Ka , with the KDC; B shares the master key, Kb with the KDC.

### If user A wishes to establish a logical connection with B, performs the following steps:

1. A issues a request to the KDC for a session key to protect a logical connection to B. The message includes the identity of A and B and a unique identifier, N1 ,
2. The KDC responds with a message encrypted using Ka. The message contains 2 items for A, session key Ks, and original request. And also it contains 2 items for B , Ks and IDa encrypted with Kb.
3. Thus, A reads the message, and sends information to B originated by KDC.
4. B sends N2 to A by encrypting it with Ks.
5. Using Ks, A responds with f(N2). ‘f’ is some function performed on N2.



### Automatic key distribution

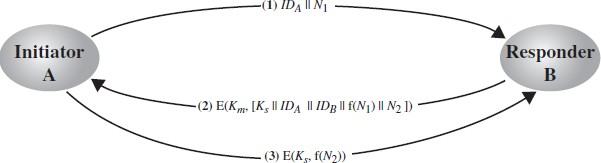


1. **Decentralized key distribution**

In this there is no key distribution center. Whenever a user wants to communicate with other user, sends request for the session key to that user.

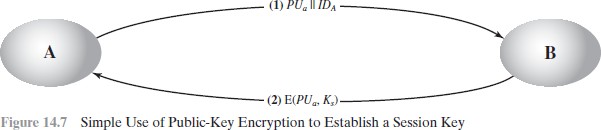
### A session key is distributed with the following sequence of steps:

1. A issues a request to B for a session key and includes a nonce,N1 .
2. B responds with a message that is encrypted using the shared master key. The response includes the session key selected by B, an identifier of B, the value f(N1) , and another nonce,N2.
3. Using the new session key,A returns f(N2) to B.



## Secret key is distributed using Asymmetric encryption

* 1. **Simple Secret Key Distribution**



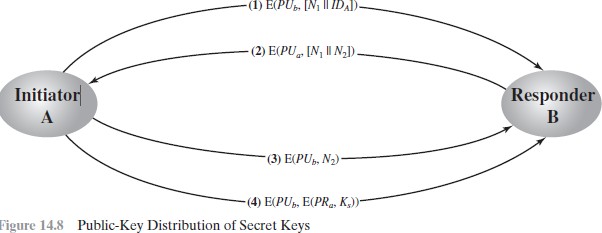
If A wishes to communicate with B, the following procedure is employed:

* + 1. A generates a public/private key pair **{*PUa, PRa}*** and transmits a message to B consisting of **PUa** and an identifier of **A**, **IDA**.
    2. B generates a secret key, ***Ks***, and transmits it to **A,** encrypted with **A's** public key.
    3. A computes **D(*PRa*, E(*PUa*, *Ks*))** to recover the secret key. Because only A can decrypt the message, only A and B will know the identity of Ks.
    4. **A** discards ***PUa*** and ***PRa*** and **B** discards ***PUa***.
* A and B can now securely communicate using conventional encryption and the session key Ks. At the completion of the exchange, both A and B discard Ks.
* No keys exist before the start of the communication and after the completion of communication.
* **Adv:** Secure from eavesdropping.
* **Problem:** man-in-the-middle attack

### Steps involved in man-in-the-middle attack:

1. A generates a public/private key pair {PUa, PRa} and transmits a message intended for B consisting of PUa and an identifier of A, IDA.
2. E intercepts the message, creates its own public/private key pair {PUe, PRe} and transmits PUe || IDA to B.
3. B generates a secret key, Ks, and transmits E(PUe, Ks).
4. E intercepts the message and learns Ks by computing D(PRe, E(PUe, Ks)).
5. E transmits E(PUa, Ks) to A.

## Secret Key Distribution with Confidentiality and Authentication



### Procedure to distribute public key:

* 1. A uses B’s public key to encrypt a message to B containing an identifier IDA and a nonce N1, which is used to identify this transaction uniquely
  2. B sends a message to A encrypted PUa with and containing A’s nonce (N1) and new nonce (N2) generated by B .
  3. A returns N2, encrypted using B’s public key, to assure B that its correspondent is A.
  4. A selects a secret key Ks and sends M=E(PUb, E(PRa, Ks))to B.
  5. B computes D(PUa, E(PRb, Ks))to recover the secret key.

This scheme ensures that confidentiality and authentication in the exchange of secret key.

# Distribution of Public Keys

* + Techniques for the distribution of public keys are grouped into:

1. public announcement
2. publicly available directory
3. public-key authority
4. public-key certificates

### Public Announcement

* + Users distribute public keys to recipients or broadcast to all the users in the netwrok
    - E.g., Append PGP (pretty good privacy) keys to **email messages** or post to **news groups** or **email list**

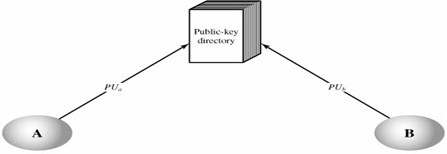


* + Major weakness is forgery
    - Anyone can create a key claiming to be someone else and broadcast it
    - Until forgery is discovered an attacker can masquerade as claimed user

### Publicly Available Directory

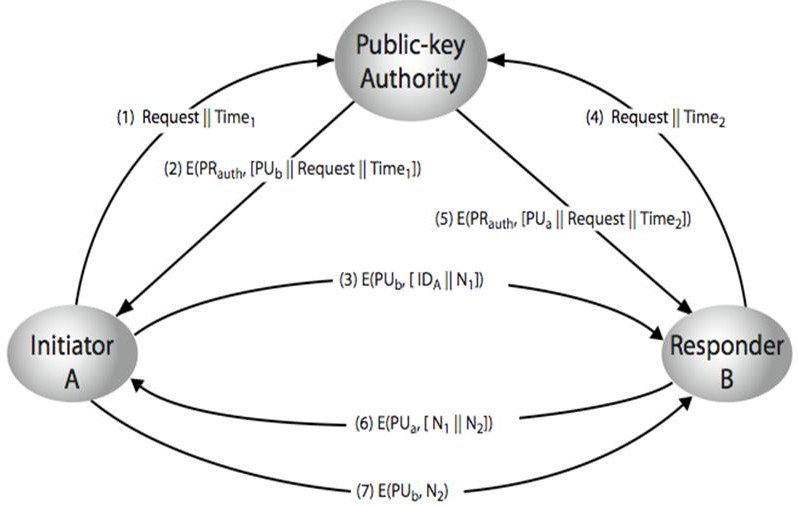
* + A public directory of public keys is used. When user A requires a public key of user B, he gets that key from the directory. Some organization or trusted one is responsible for maintenance and distribution of the public directory.

### Directory is having following properties:

1. The authority maintains a directory with a {name, public key} entry for each participant.
2. Each participant registers a public key with the directory authority. Registration would have to be in person or by some form of secure authenticated communication.
3. A participant may replace the existing key with a new one at any time.
4. Directory is periodically published
5. Participants could also access the directory electronically.
   * Still vulnerable to tampering or forgery

### Public-Key Authority

**In** this public key authority maintains a directory of public keys. Required uses can request and get the public key from the public key authority. The user must know the public key of the authority PUauth .



### Drawbacks:

* 1. There is only one authority, so performance may be decreased.
  2. If authority is hacked, the entire database in the hand of hacker.

### Public key certificates

* There exists a trusted Certification Authority (CA), which issues public key certificates. These certificates are used for exchanging the keys without interaction of public-key authority.
* The certificate authority can create and update certificates
* Any participant can get the certificates by supplying public key.
* A certificate binds ***identity*** of the user to ***public key***

– Contains other info such as period of validity, issuer’s info, etc

* All contents signed by a trusted Certification Authority (CA)
* Any participant can read a certificate to determine the name and public key of the certificate’s owner.
* Certificate can be verified by anyone who knows the CA public-key.

### Example:

Certificate of A

CA = E(PRauth, [T||IDA || PUa]) Where

PRauth – private key of Certificate Authority T- Timestamp

Any participants can read and verify the certificate of A as……. D(PUauth, CA) = D(PUauth, E(PRauth, [T || IDA||PUa])) = (T||IDA || PUa)

