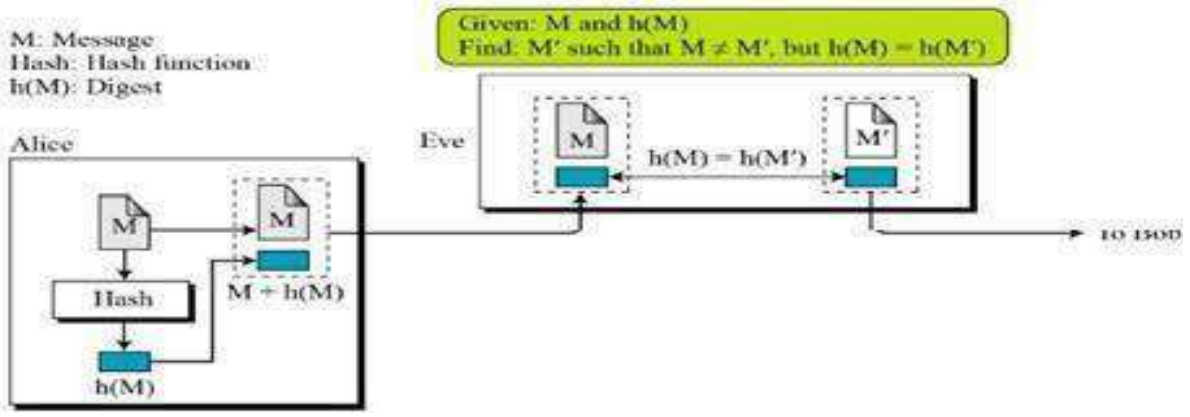
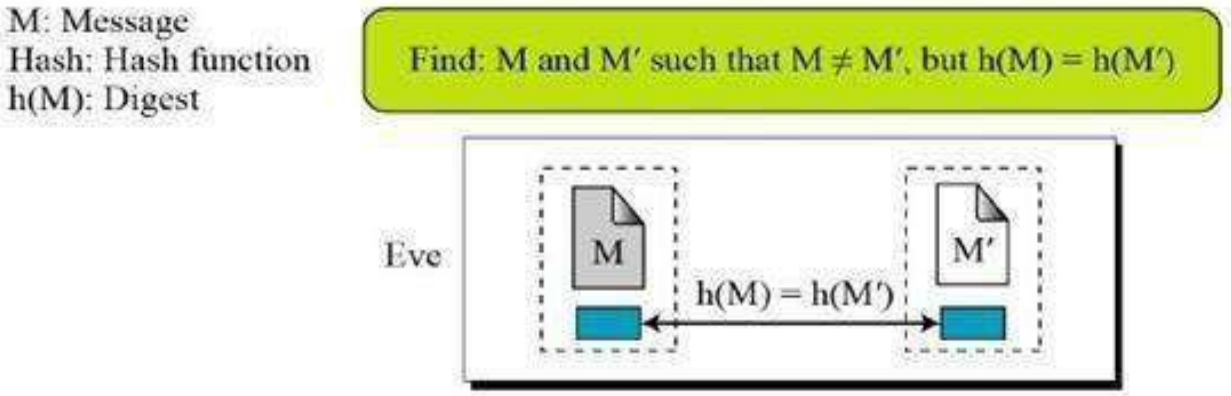
x and is asked to compute the value of x1 ≠ x, such that h(x) = h(x1).

If it difficult for the attacker to perform this computation we claim that the hash function is second pre-image resistant.



A function without preimage resistance is usually also not second preimage resistant: Given a message x_1, calculate $h := H(x_1)$ and then get a preimage x_2 from h, then we usually have $x_1 \neq x_2$ and $H(x_1) = H(x_2)$.

Collision Resistance: Collision of a hash function is the event when two values x and x1, such that x1 ≠ x hash to the same value, i.e., h(x) = h(x^1).



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 this model behaves as follows:

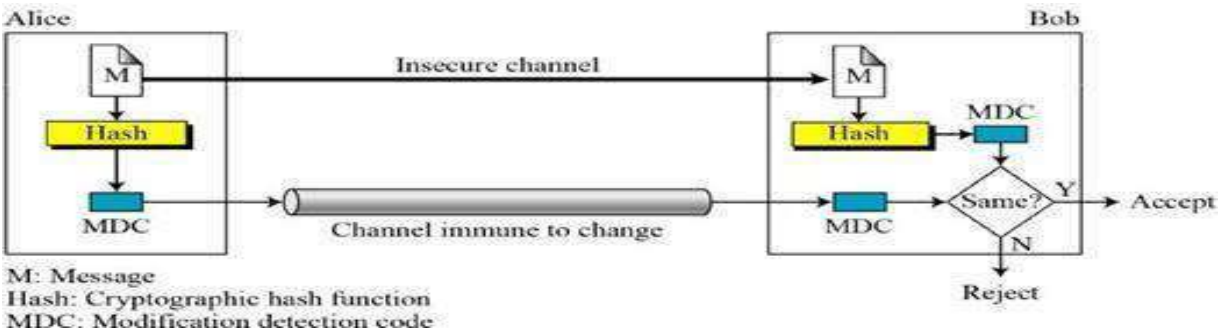
1. When a new message of any length is given, the oracle creates and gives a fixed-length message digest that is a random string of 0s and 1s. The oracle records the message and the message digest.
2. When a message is given for which a digest exists, the oracle simply gives the digest in the record.
3. The digest for a new message needs to be chosen independently from all previous digests.

2. Message Authentication:

- A message digest guarantees the integrity of a message. It guarantees that the message has not been changed.
- A message digest does not authenticate the sender of the message.
- When Alice sends a message to Bob, Bob needs to know if the message is coming from Alice.
- To provide message authentication, Alice needs to provide proof that it is Alice sending the message and not a fraud.
- The digest created by a cryptographic hash function is normally called a **Modification Detection Code (MDC)**. This code can detect any modifications in the message.
- What we need for message authentication is a **Message Authentication Code** (MAC).

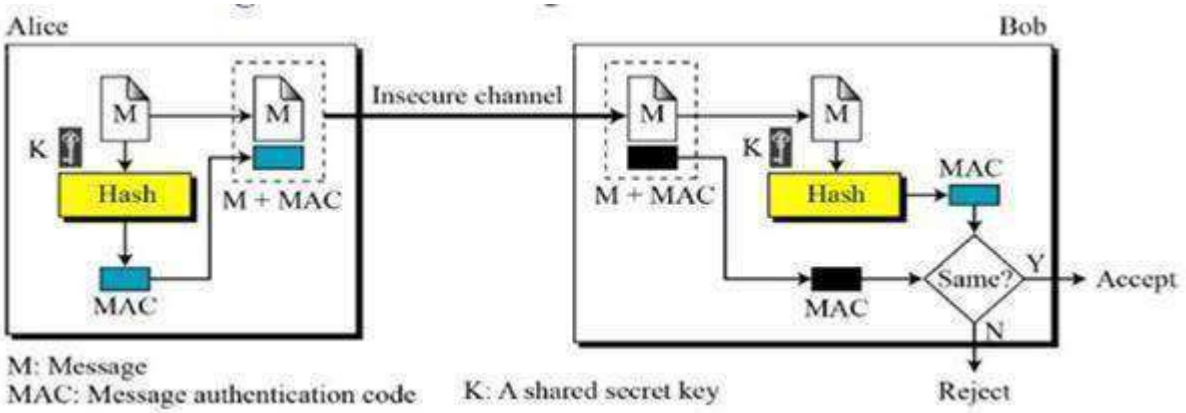
Modification Detection Code (MDC):

- 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 thenew MDC. If they are the same, the message has not been changed.



Message Authentication Code (MAC):

- To ensure the integrity of a message and the data origin authentication – we need to change a modification detection code (MDC) to a Message Authentication Code (MAC).
- The difference between MDC and MAC is that the second include a secrete key between Alice and Bob.



Alice uses a hash function to create a MAC from the concatenation of the key and the message, h(K | M). She sends the message and the MAC to Bob over the insecure channel. Bob separates the message from the MAC. He then makes a new MAC from the concatenation of the message and the secret key. Bob then compares the newly created MAC with the one received. If the two MACs match, the message is authentic and has not been modified by an adversary.

MAC Security

How can Eve forge a message without having the key?

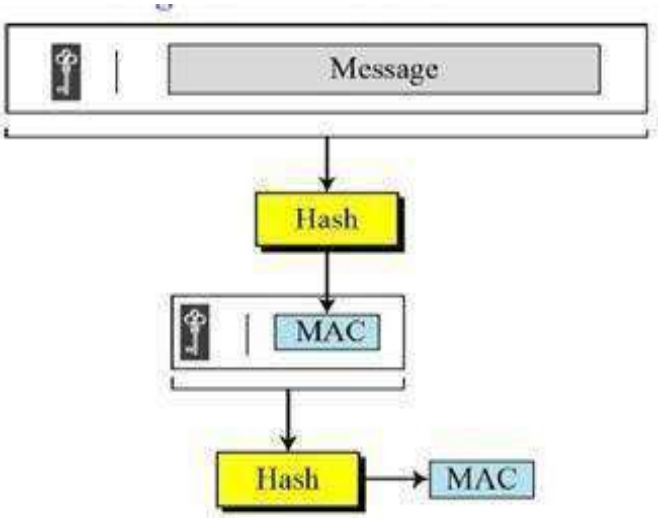
1. If size of the key allows exhaustive search, Eve may try all possible keys to digest the message.
2. Use preimageattack.
3. Given some pairs of messages and their MACs, Eve can manipulate them to come up with a new message and its digest

Note: The security of a MAC depends on the security of the underlying hash algorithm.

Nested MAC:

✓ To improve MAC security, nested MACs were designed in which hashing is performed twice.

- In 1st step, the key is concatenated with the message and is hashed to create an intermediate digest.
- In 2nd step, the key is concatenated with the intermediate digest to create the final digest.



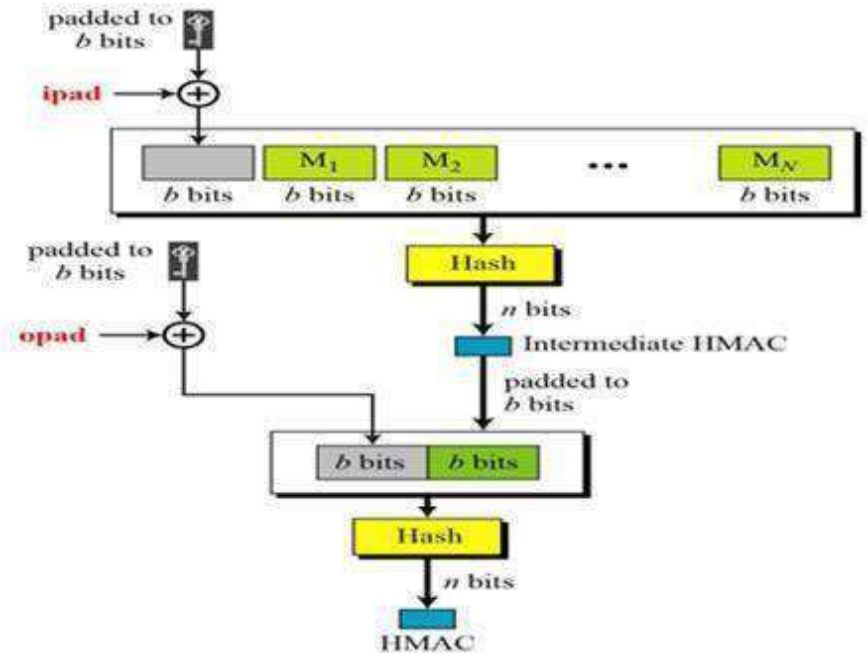
HMAC (Hashed MAC):

- HMAC algorithm stands for Hashed or Hash based Message Authentication Code
- it uses the Hashing concept twice, so great resistant to attacker
- HMAC consists of twin benefits of Hashing and MAC

✓ **The working of HMAC starts** with taking a message M containing blocks of length b bits.

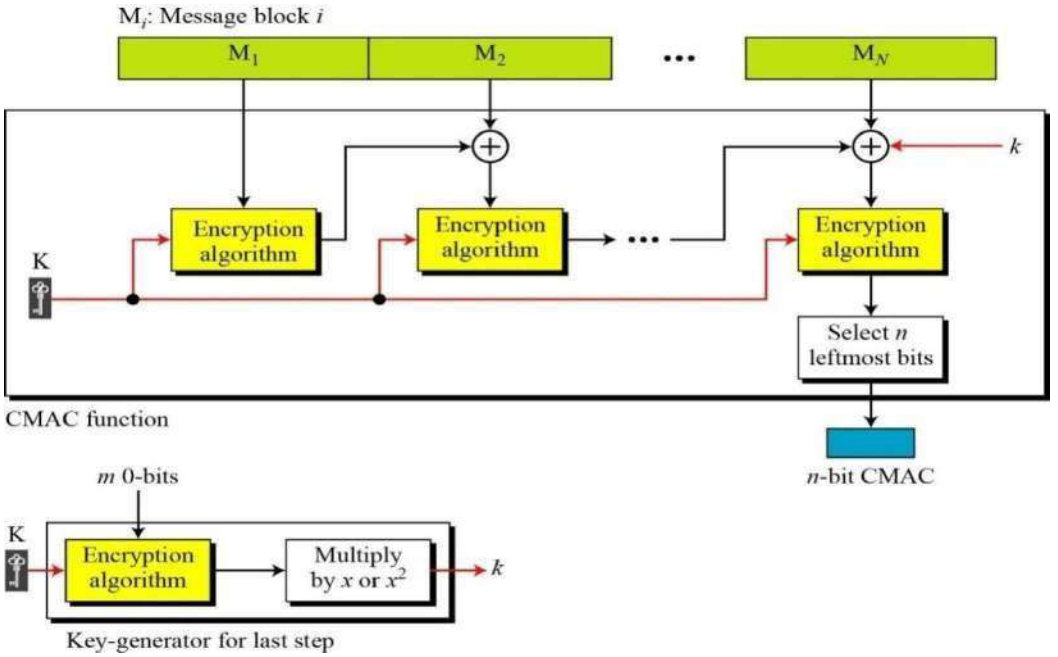
✓ An input signature is padded to the left of the message and the whole is given as input to a hash function which gives us a intermediate HMAC.

✓ Intermediate HMAC again is appended to an output signature and the whole is applied a hash function again, the result is our final HMAC of n bits



CMAC (Cipher based MAC)

- This is similar to CBC(Cipher Block Chaining),
- It takes N blocks of message but creates one block of MAC
- The message is divided into N blocks of m-bit size. If last block is not m-bit size,then
  padded with start 1 then 0000…, like 100000…
- The block is encrypted with key K then its output is XOR with the next block for 2nd
  encryption, so on.
- The last block is encrypted with some addtional k value for more scurity.



Cryptographic Hash Functions

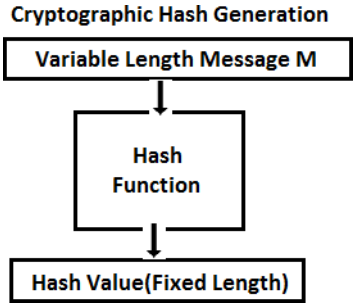
A ***cryptographic hash function*** takes a message of arbitrary length and creates a message digest of fixed length, also called hash.

A cryptographic hash function H accepts a variable-length block of data M as input and produces a fixed-size hash value.

There are two most promising cryptographic hash algorithms –

- SHA-512

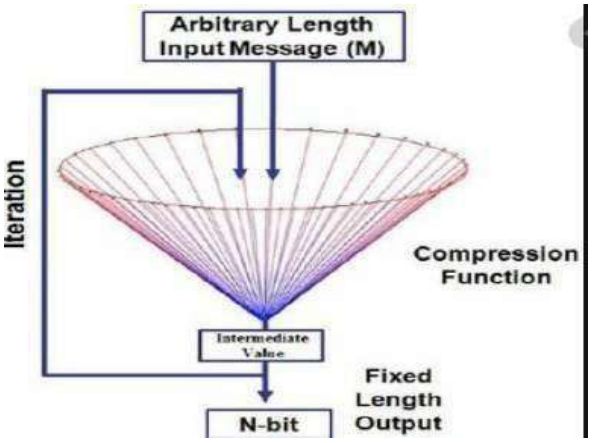
- Whirlpool



Iterated Hash Function

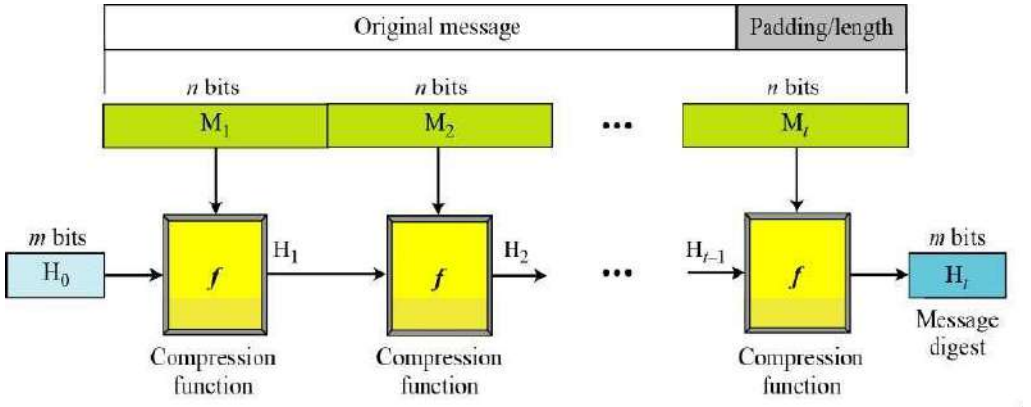
All cryptographic functions need to create a fixed size digest out of a variable-size message. Actually, the hash function is fixed size input function, but performs number of times.

This fixed-size hash function is referred to as a compression function, it compresses m-bit string input to n bit string.



Merkle-Damgard Scheme

- This is an iterated hash function that is collision resistant
- This is the basis for many cryptographic hash functions today.



- Message is divided into t-blocks of n-bit size. If necessary some bits are padded
- The blocks are $M_1,M_2,\ldots M_t$ and the digest created at each compression function are $H_1,H_2,\ldots H_t$
- Before starting the iteration, the digest H_0 is set to fixed Value called IV(initial value or initial vector)

The compression function operates on H_{i-1} and M_i to create a new H_i. $H_i=f(H_i\text{-}1,M_i)$ where f is a compression function

Hash Functions Invention

- Several Hash functions were designed by Ron Rivest.
- These are MD(Message Digest), MD2, MD4, and MD5
- MD5 takes blocks of size 512-bits and creates 128-bit digest.
- The 128-bit size digest is too small to resist collision attack.

Secure Hash Algorithm(SHA)

- SHA originally designed by NIST & NSA in 1993
- SHA was revised in 1995 as SHA-1
- adds 3 additional versions of SHA
- SHA-256, SHA-384, SHA-512structure & detail is similar to SHA-1

Table : 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

- SHA-512 is family of Secure Hash Algorithm
- SHA-512 creates a 512 bit message digest .
- The original message divided into multiple blocks of size 1024bits.
- The Processing of each block involves 80 rounds
- Each block of size(1024bits) can be assumed as 16 words of size 64bits
- The maximum size of message is less than 2^{128}. This means that if the length of a message equal to or greater than 2^{128}, it will not be processed by SHA-512

- SHA-512 based on Merkle-Damgard scheme.

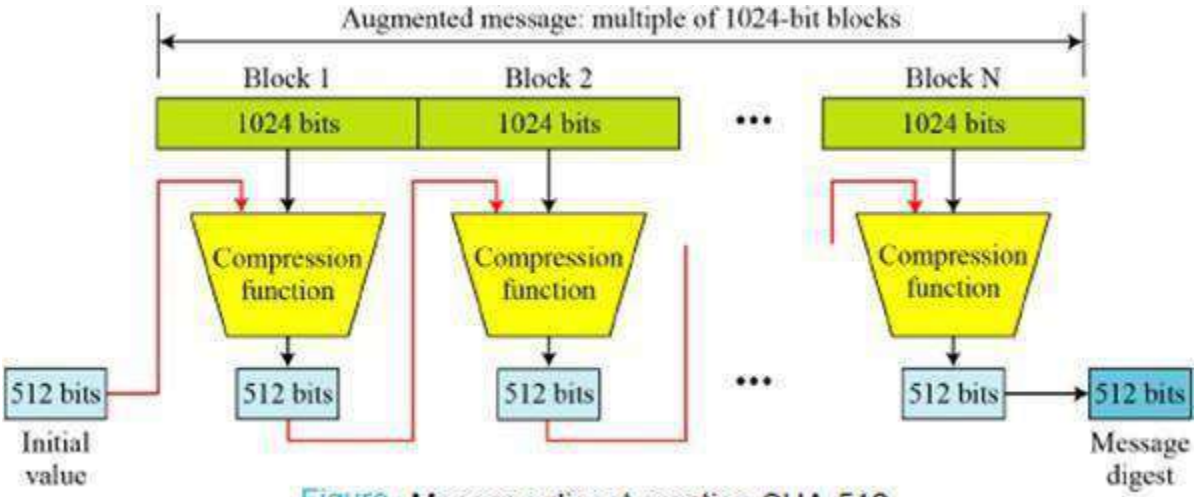
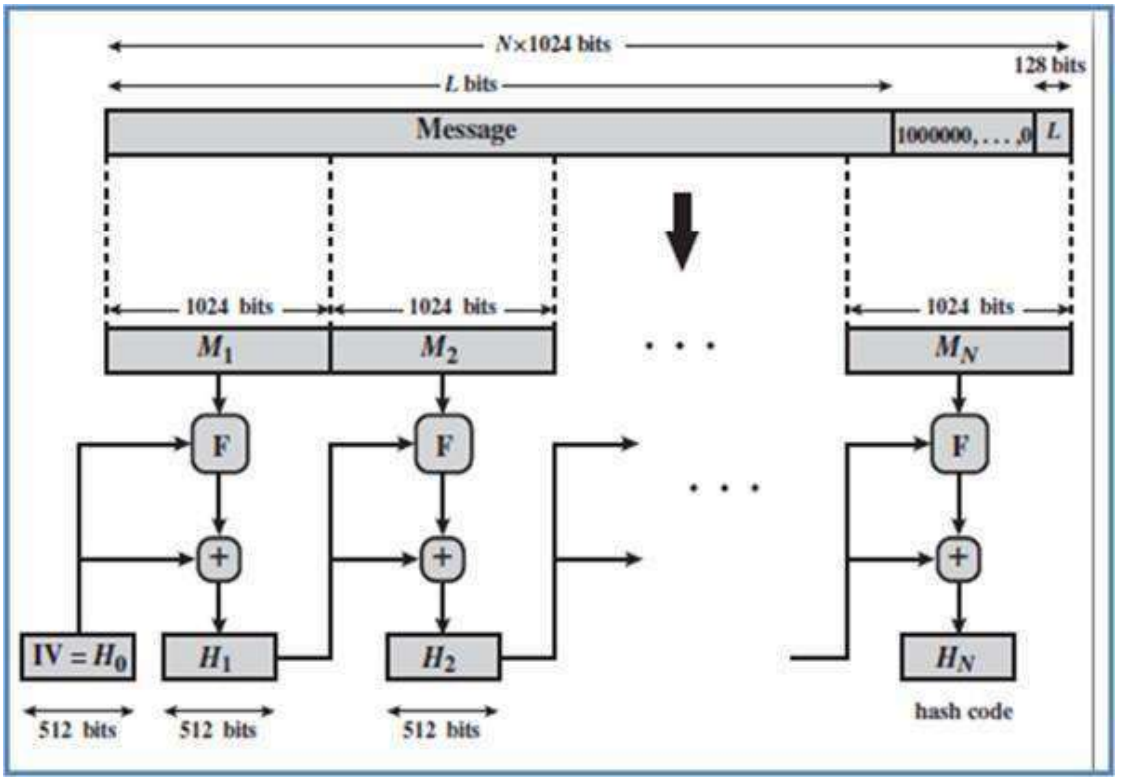


Figure Message digest creation SHA-512

The Following Figure shows internal logic of the SHA-512



STEPS:

1. **Append padding bits:**

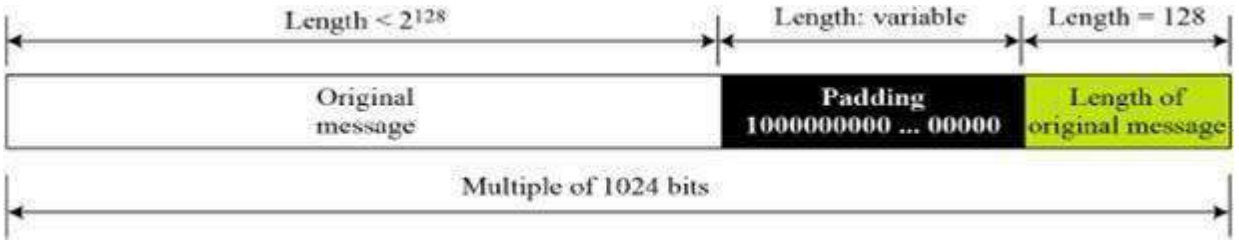
The message is padded with 1000000…. To make the message multiples of 1024.

2. Append length of the message:

A block of 128 bits is appended to the message. Contains the length of the original message.
Before addition of the length of message , we need to pad as specified in the first step.
The size of padding bits is calculated
as: (|M|+|P|+128)=0 mod 1024 □
|P|=-|M|-128 mod 1024
Example: What is the number of padding bits if the length of the original message is 2590
Solution: |P|=-2590-128 mod 1024
=-2718 mod 1024 = -670 mod 1024
=(1024-670) mod 1024 = 354
The padding consists of one 1 followed by 353 0's

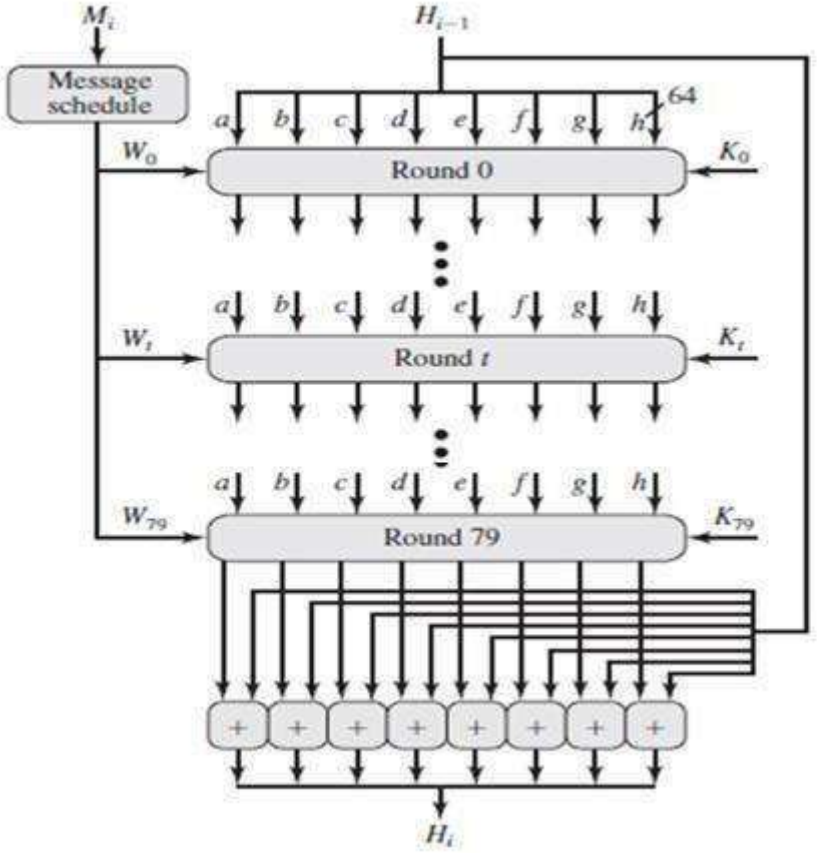
Length Field and Padding:

Before the message digest can be created, SHA-512 requires the addition of a 128-bit length field (0-(2^{128}-1)to the message that defines the length of the message in bits.



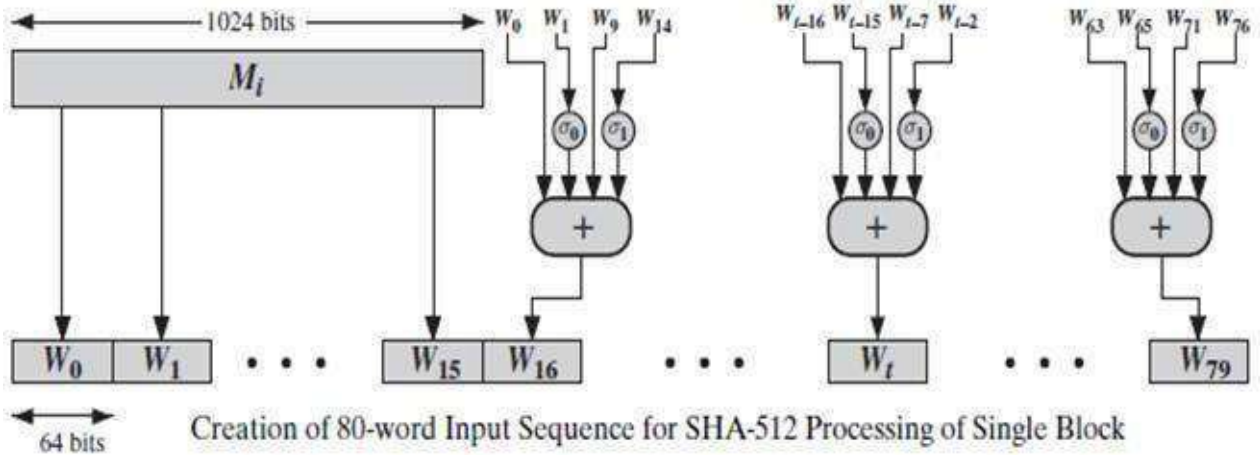
Compression Function

The heart of the algorithm is a module that consists of 80 rounds; this module is labeled as F in Block Diagram.
Each round t takes as input the 512-bit buffer value, abcdefgh, and updates the contents of the buffer.
Each round t makes use of a 64-bit value Wt, derived from the current 1024-bit block being processed (Mi).
Each round t also makes use of an additive constant Kt (64-bit)
The output of the 80th round is added to the input to the first round (Hi-1) to produce Hi.



Firgure:Processing of a single 1024-bit block

80-Word Input Sequence



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

$$W_t = \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16}$$

where

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

Constants

428a2f98d728ae22	7137449123ef65cd	b5c0fbcfec4d3b2f	e9b5dba58189dbbc
3956c25bf348b538	59f111f1b605d019	923f82a4af194f9b	ab1c5ed5da6d8118
d807aa98a3030242	12835b0145706fbe	243185be4ee4b28c	550c7dc3d5ffb4e2
72be5d74f27b896f	80deb1fe3b1696b1	9bdc06a725c71235	c19bf174cf692694
e49b69c19ef14ad2	efbe4786384f25e3	0fc19dc68b8cd5b5	240ca1cc77ac9c65
2de92c6f592b0275	4a7484aa6ea6e483	5cb0a9dcbd41fbd4	76f988da831153b5
983e5152ee66dfab	a831c66d2db43210	b00327c898fb213f	bf597fc7beef0ee4

…..

Initialize hash buffer

a = 6A09E667F3BCC908	e = 510E527FADE682D1
b = BB67AE8584CAA73B	f = 9B05688C2B3E6C1F
c = 3C6EF372FE94F82B	g = 1F83D9ABFB41BD6B
d = A54FF53A5F1D36F1	h = 5BE0CD19137E2179

Example of SHA-512

ASCII characters: "abc", which is equivalent to the following 24-bit binary string:

01100001 01100010 01100011 = 616263 in Hexadecimal

The original length is 24 bits, or a hexadecimal value of 18.
the 1024-bit message block, in hexadecimal, is

6162638000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000018

W0 = 6162638000000000	W5 = 0000000000000000
W1 = 0000000000000000	W6 = 0000000000000000
W2 = 0000000000000000	W7 = 0000000000000000
W3 = 0000000000000000	W8 = 0000000000000000
W4 = 0000000000000000	W9 = 0000000000000000
W10 = 0000000000000000	W13 = 0000000000000000
W11 = 0000000000000000	W14 = 0000000000000000
W12 = 0000000000000000	W15 = 0000000000000018

The resulting 512-bit message digest is

ddaf35a193617aba	cc417349ae204131	12e6fa4e89a97ea2	0a9eeee64b55d39a
2192992a274fc1a8	36ba3c23a3feebbd	454d4423643ce80e	2a9ac94fa54ca49f

DIGITAL SIGNATURE

- A digital signature is a technique used to validate the authenticity and integrity of a message.
- In the physical world, A person signs a document to show that it originated from him or was approved by him. The signature is proof to recipient that the document comes from the correct entity.
- Similarly, a digital signature is a technique that binds a person/entity to the digital data. This binding can be independently verified by receiver as well as any third party.
- Digital signature is a cryptographic value that is calculated from the data and a secret key known only by the signer.

COMPARISON of conventional signature & DIGITAL SIGNATURE

Inclusion

A conventional signature is included in the document; it is part of the document.
But when 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, he 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.

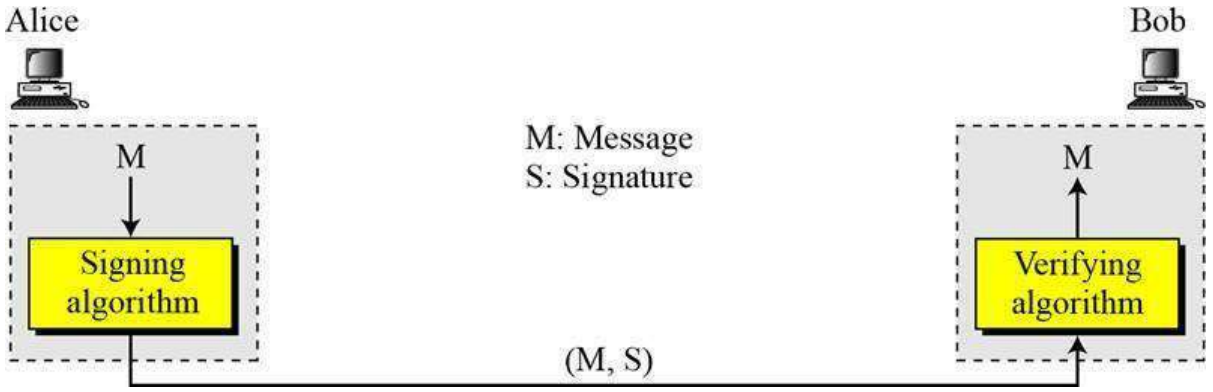
Duplicity

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

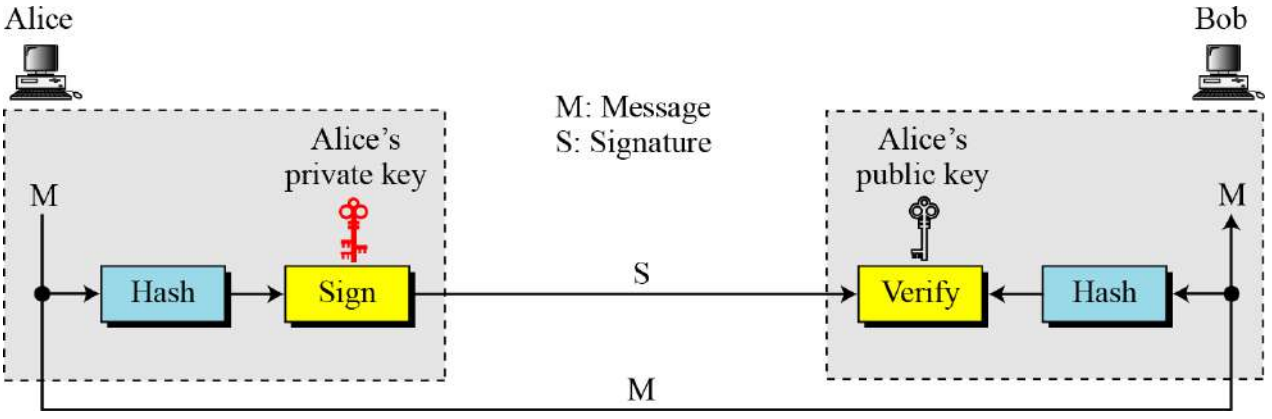
file. In digital signature, there is no such distinction unless there is a factor of time on the document.

PROCESS OF DIGITAL SIGNATURE

Figure shows the digital signature process. 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.



SIGNING THE DIGEST



The drawback of Asymmetric key cryptosystems that is "inefficient for long messages" .t In a digital signature system can be overcome by "signing the digest of the message".

SERVICES

The services in cryptography are:
Message confidentiality, authentication, Integrity and Non-repudiation.

- A digital signature system can provide Message authentication, Integrity and Non-repudiation, but still need encryption/decryption for message confidentiality.

Message Authentication

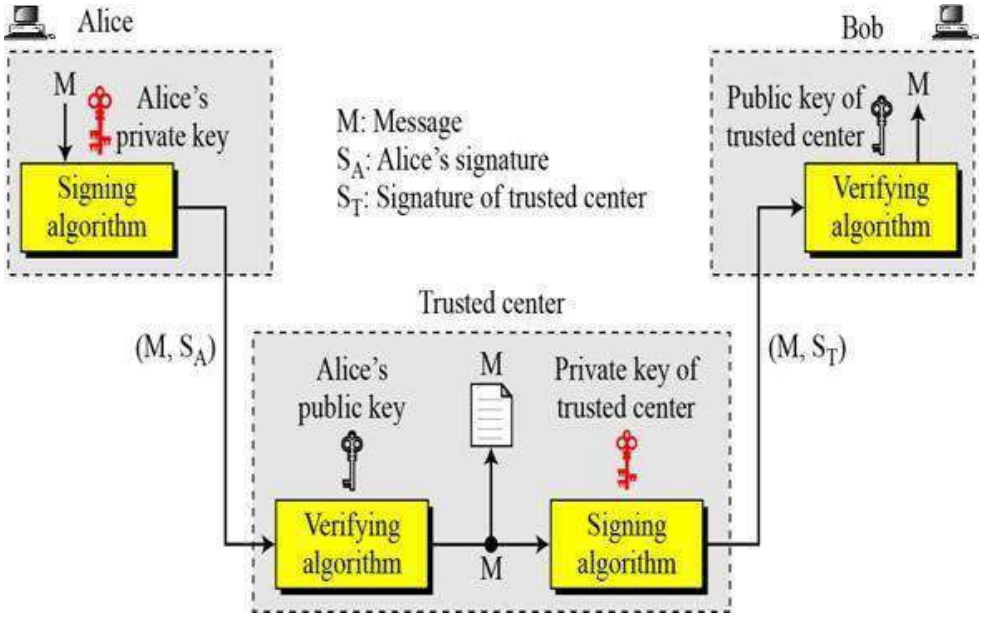
- A secure digital signature scheme, like a secure conventional signature can provide message authentication
- *Example, Bob can verify that the message is sent by Alice because Alice's public key is used in verification.*

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.

Nonrepudiation

Nonrepudiation can be provided using a trusted party.



Confidentiality

A digital signature does not provide privacy.
If there is a need for privacy, another layer of encryption/decryption must be applied.

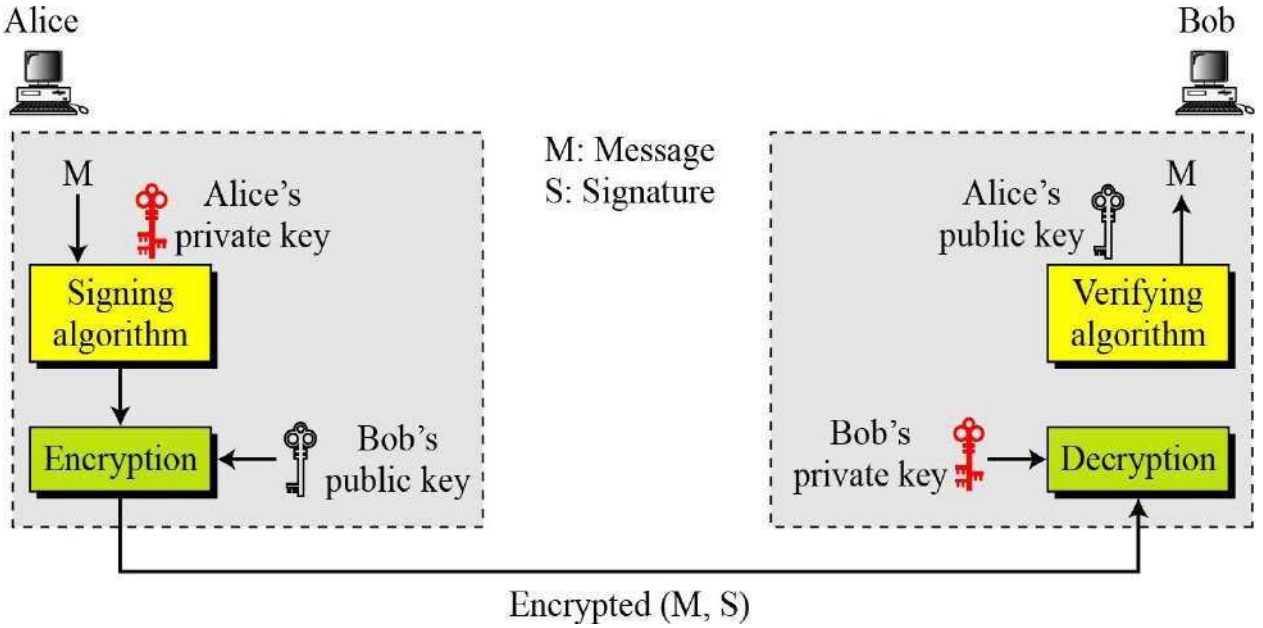


Figure *Adding confidentiality to a digital signature scheme*

ATTACKS ON DIGITAL SIGNATURE

Attack Types

1. *Key-Only Attack*

In key-only attack, the public key of A is available to every one and C makes use of this fact and try to recreate the signature of A and digitally sign the documents that A does not intend to do.

2. *Known-Message Attack*

In the known message attack, C has few previous messages and signatures of A. Now C tries to forge the signature of A on to the documents that A does not intend to sign by using the brute force method by analyzing the previous data to recreate the signature of A

3. *Chosen-Message Attack*

In this method C has the knowledge about A's public key and obtains A's signature on the messages and replaces the original message with the message C wants A to sign with having A's signature on them unchanged.

Forgery Types

1. ***Existential Forgery***

Adversary can create a pair (message, signature), such that the signature of the message is valid. Adversary has no control on the messages whose signature is forged

2. *Selective Forgery*

Adversary is able to create valid signatures on a message chosen by someone else, with a significant probability.

Adversary controls the messages whose signature is forged

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

RSA DIGITAL SIGNATURE SCHEMES

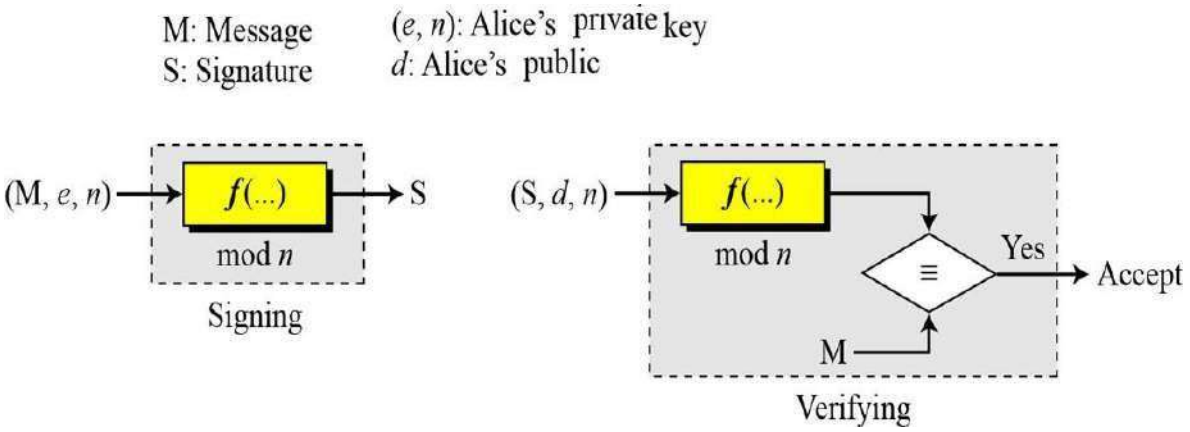


Figure : General idea behind the RSA digital signature scheme

The sender uses his own private key tosign the documemnet, the receivr uses the senders public key to verify it

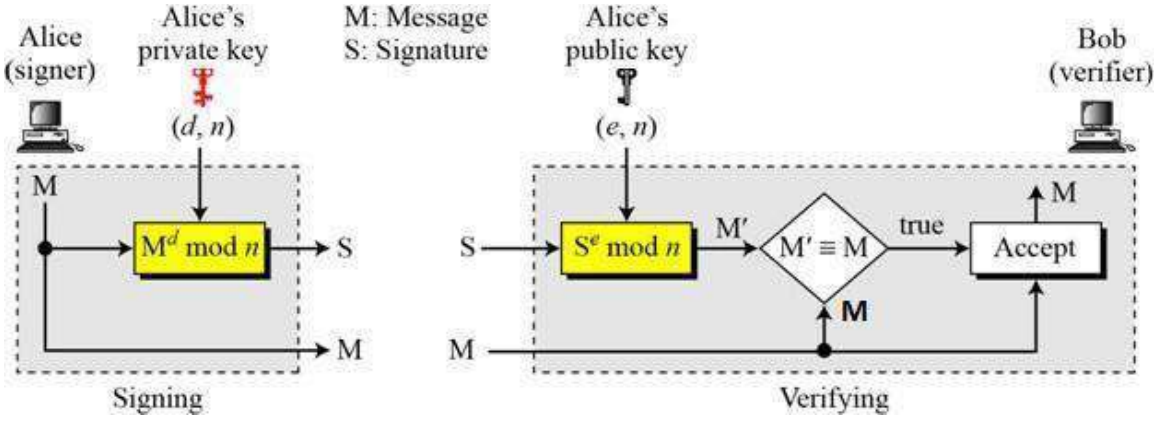
RSA DIGITAL SIGNATURE SCHEMES – Key Generation

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

1. Sender chooses two prime numbers p and q
2. Calculate n=pxq
3. Calculate f(n) = (p-1) x (q-1)
4. Chooses the public exponent e and calculates d (private exponent) such that e x d = 1 mod f(n)

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

DIGITAL SIGNATURE SCHEMES – Signing and verifying



Signing: Alice create a signature out of the message using her private exponent,
$S=M^d \bmod n$ and sends the signature to Bob

Verifying: Bob receives M and S. Bob applies A lice public exponent to the signature to create a copy of the message $M^1 = S^e \bmod n$. Bob compares M and M^1. If both are congruent, accepts the message.

$M^1 \equiv M \pmod n \rightarrow S^e \equiv M \pmod n \rightarrow M^{dxe} \equiv M \pmod n$

RSA DIGITAL SIGNATURE SCHEMES – EXAMPLE

As a trivial example, suppose that Alice chooses $p = 823$ and $q = 953$, and calculates $n = 784319$. The value of f(n) is 782544. Now she chooses $e = 313$ and calculates $d = 160009$. At this point key generation is complete. Now imagine that Alice wants to send a message with the value of M = 19070 to Bob. She uses her private exponent, 160009, to sign the message:

$$M: 19070 \quad \rightarrow \quad S = (19070^{160009}) \bmod 784319 = 210625 \bmod 784319$$

Alice sends the message and the signature to Bob. Bob receives the message and the signature. He calculates

$$M' = 210625^{313} \bmod 784319 = 19070 \bmod 784319 \quad \rightarrow \quad M \equiv M' \bmod n$$

Bob accepts the message because he has verified Alice's signature

ElGamal Digital Signatures

- signature variant of ElGamal, related to D-H
  - so uses exponentiation in a finite Galois field
  - security based difficulty of computing discrete logarithms, as in D-H
- use private key for encryption (signing)

• uses public key for decryption (verification)
• each user (eg. A) generates their key

– chooses a secret key: $1 < x_A < q-1$
– compute their **public key**: $y_A = a^{x_A} \bmod q$

- Alice signs a message M to Bob by computing
  - the hash $m = H(M)$, 0 <= m <= (q-1)
  - chose random integer K with 1 <= K <= (q-1) and gcd(K,q-1)=1
  - compute temporary key: $S_1 = a^k \bmod q$
  - compute K^{-1} the inverse of K mod (q-1)
  - compute the value: $S_2 = K^{-1}(m-x_A S_1) \bmod (q-1)$
  - signature is:(S_1,S_2)
- any user B can verify the signature by computing
  - $V_1 = a^m \bmod q$
  - $V_2 = y_A^{S1} S_1^{S2} \bmod q$
  - signature is valid if $V_1 = V_2$

ElGamal Signature Example

- use field GF(19) q=19 and a=10
- Alice computes her key:
  - A chooses x_A=16 & computes $y_A=10^{16} \bmod 19 = 4$
- Alice signs message with hash m=14 as (3,4):
  - choosing random K=5 which has gcd(18,5)=1
  - computing $S_1 = 10^5 \bmod 19 = 3$
  - finding $K^{-1} \bmod (q-1) = 5^{-1} \bmod 18 = 11$
  - computing $S_2 = 11(14-16.3) \bmod 18 = 4$
- any user B can verify the signature by computing
  - $V_1 = 10^{14} \bmod 19 = 16$
  - $V_2 = 4^3.3^4 = 5184 = 16 \bmod 19$

since 16 = 16signature is valid

Schnorr Digital Signatures

- also uses exponentiation in a finite (Galois)
  - security based on discrete logarithms, as in D-H
- minimizes message dependent computation
  - multiplying a 2*n-bit* integer with an *n-bit* integer
- main work can be done in idle time
- have using a prime modulus *p*
  - *p–1* has a prime factor *q* of

appropriate size typically *p* 1024-bit and *q* 160-bit numbers

Schnorr Key Setup

- choose suitable primes *p , q*
- choose *a* such that $a^q = 1 \bmod p$

- (a,p,q) are global parameters for all
- each user (eg. A) generates a key
  - chooses a secret key (number): $0 < s_A < q$
  - compute their **public key**: $v_A = a^{-s_A} \bmod q$

- user signs message by
  - choosing random r with $0<r<q$ and computing $x = a^r \bmod p$
  - concatenate message with x and hash result to computing: $e = H(M \| x)$
  - computing: $y = (r + se) \bmod q$
  - signature is pair (e, y)
- any other user can verify the signature as follows:
  - computing: $x' = a^y v^e \bmod p$
  - verifying that: $e = H(M \| x')$

Digital Signature Standard (DSS)

- US Govt approved signature scheme
- designed by NIST & NSA in early 90's
- published as FIPS-186 in 1991
- revised in 1993, 1996 & then 2000
- uses the SHA hash algorithm
- DSS is the standard, DSA is the algorithm
- FIPS 186-2 (2000) includes alternative RSA & elliptic curve signature variants
- DSA is digital signature only unlike RSA is a public-key technique

Digital Signature Algorithm (DSA)

- creates a 320 bit signature
- with 512-1024 bit security
- smaller and faster than RSA
- a digital signature scheme only
- security depends on difficulty of computing discrete logarithms
- variant of ElGamal & Schnorr schemes

DSA Key Generation

- have shared global public key values (p,q,g):
  - choose 160-bit prime number q
  - choose a large prime p with $2^{L-1} < p < 2^L$
    - where L= 512 to 1024 bits and is a multiple of 64
    - such that q is a 160 bit prime divisor of (p-1)
  - choose $g = h^{(p-1)/q}$
    - where $1<h<p-1$ and $h^{(p-1)/q} \bmod p > 1$
- users choose private & compute public key:
  - choose random private key: $x<q$
  - compute public key: $y = g^x \bmod p$

DSA Signature Creation

- to **sign** a message M the sender:
  - generates a random signature key k, $k<q$
  - nb. k must be random, be destroyed after use, and never be reused
- then computes

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

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

- sends signature (r,s) with message M
- having received M & signature (r,s)

- to **verify** a signature, recipient

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

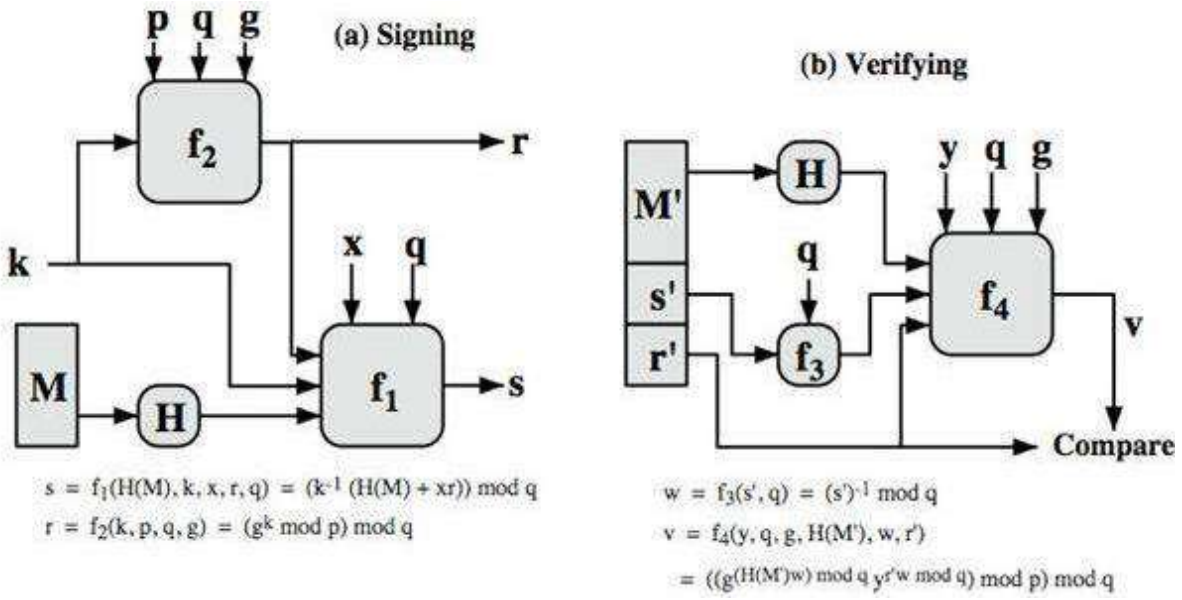
$u1 = [H(M)w] \bmod q$

$u2 = (rw) \bmod q$

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

- if v=r then signature is verified

DSS Overview



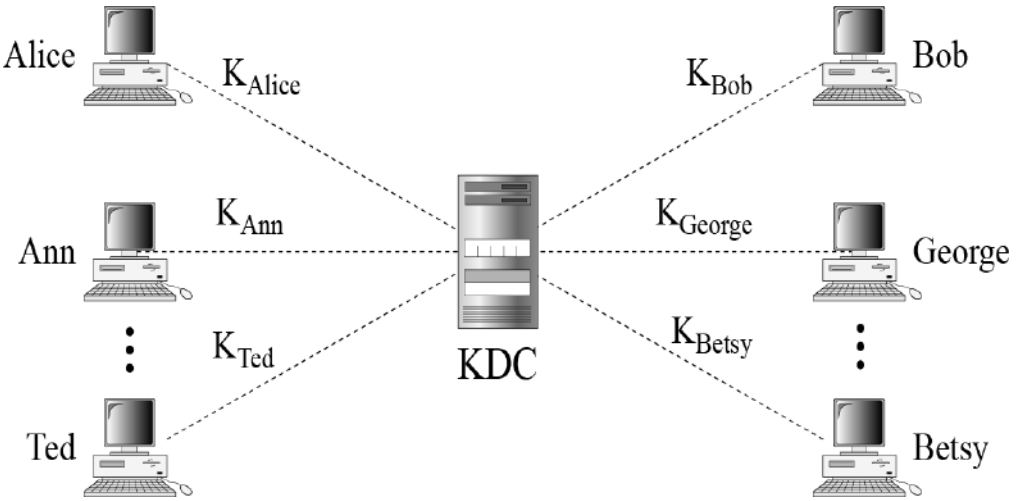
KEY MANAGEMENT

SYMMETRIC-KEY DISTRIBUTION

- Symmetric-key cryptography is more efficient than asymmetric-key cryptography for enciphering large messages.
- Symmetric-key cryptography, however, needs a shared secret key between two parties.
- Example: If Alice needs to exchange confidential messages with N people, she need N different keys and if N people need to exchange with each other, they need N(N-1) keys. If 1 million people need to communicate with each other , they need more than trillions of keys.
- This proble normally referred as N^2 problem, because the number of required keys for N entitesis N^2
- We also has a problem of the distribution of keys through the internet which is unsecure.

Key-Distribution Center: KDC

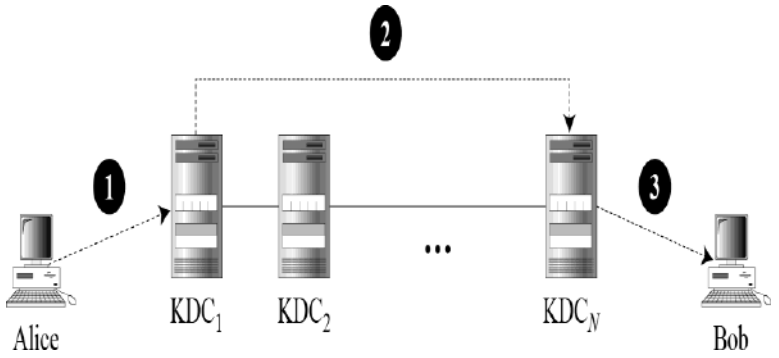
A practical solution for the above problem is the use of a trusted thord party, referred as Key-Distribution Center(KDC)



1. Alice sends a request to the KDC stating that she needs a session secrete key between her and Bob
2. KDC inform Bob about Alice request

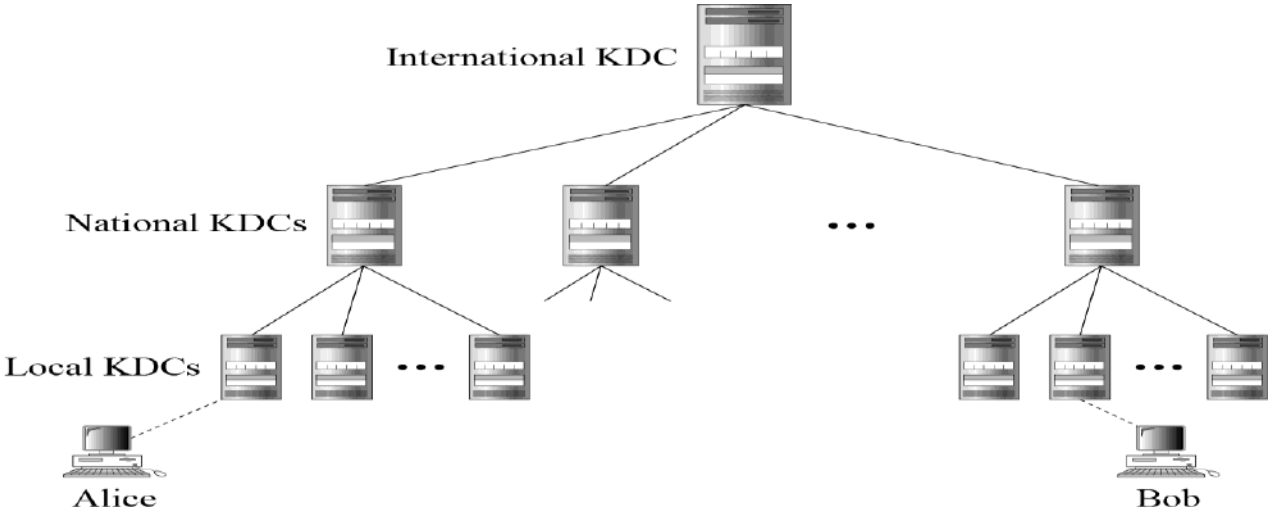
If Bob agrees, a session key is created between the two.

Flat Multiple KDCs



When the number of people using a KDC increases, the system becomes unmanageable.
To solve the problem, we use multiple KDCs. We devide the world into domains

Hierarchical Multiple KDCs



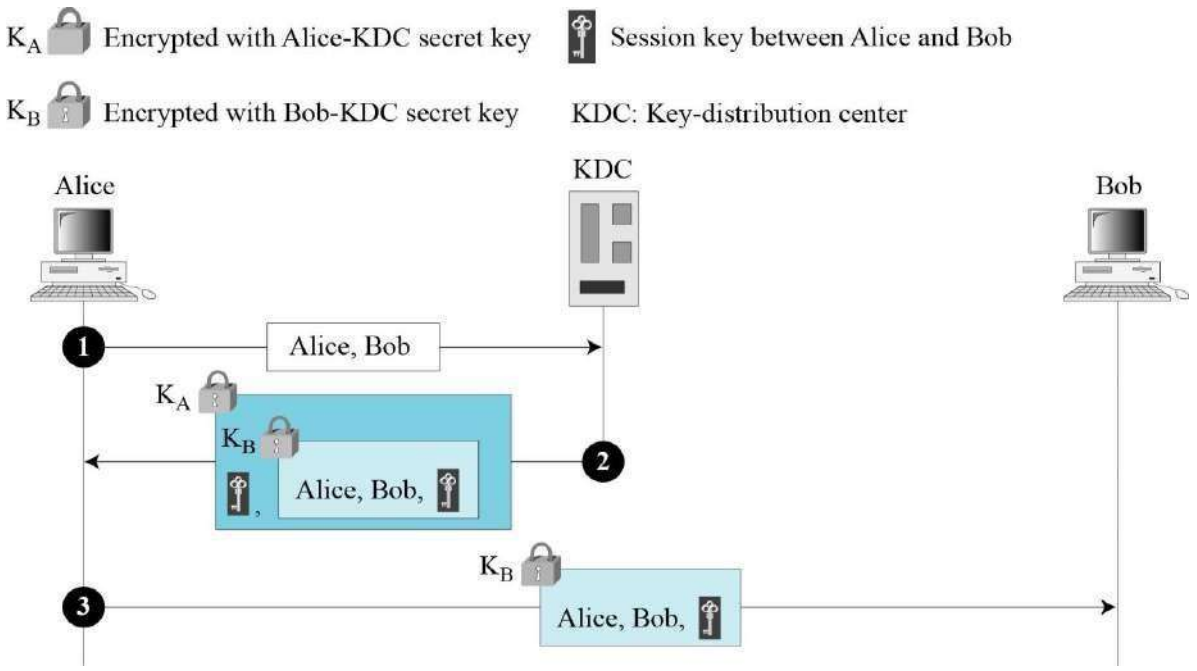
In this, KDCs are arranged in hierarchical model, the international KDC are at root, then national next and local KDCs at lower level.

Session Keys

A KDC creates a secret key for each member. This secret key can be used only between the member and the KDC, not between two members.
A session symmetric key between two parties is used only once.

Simple protocol Using a KDC

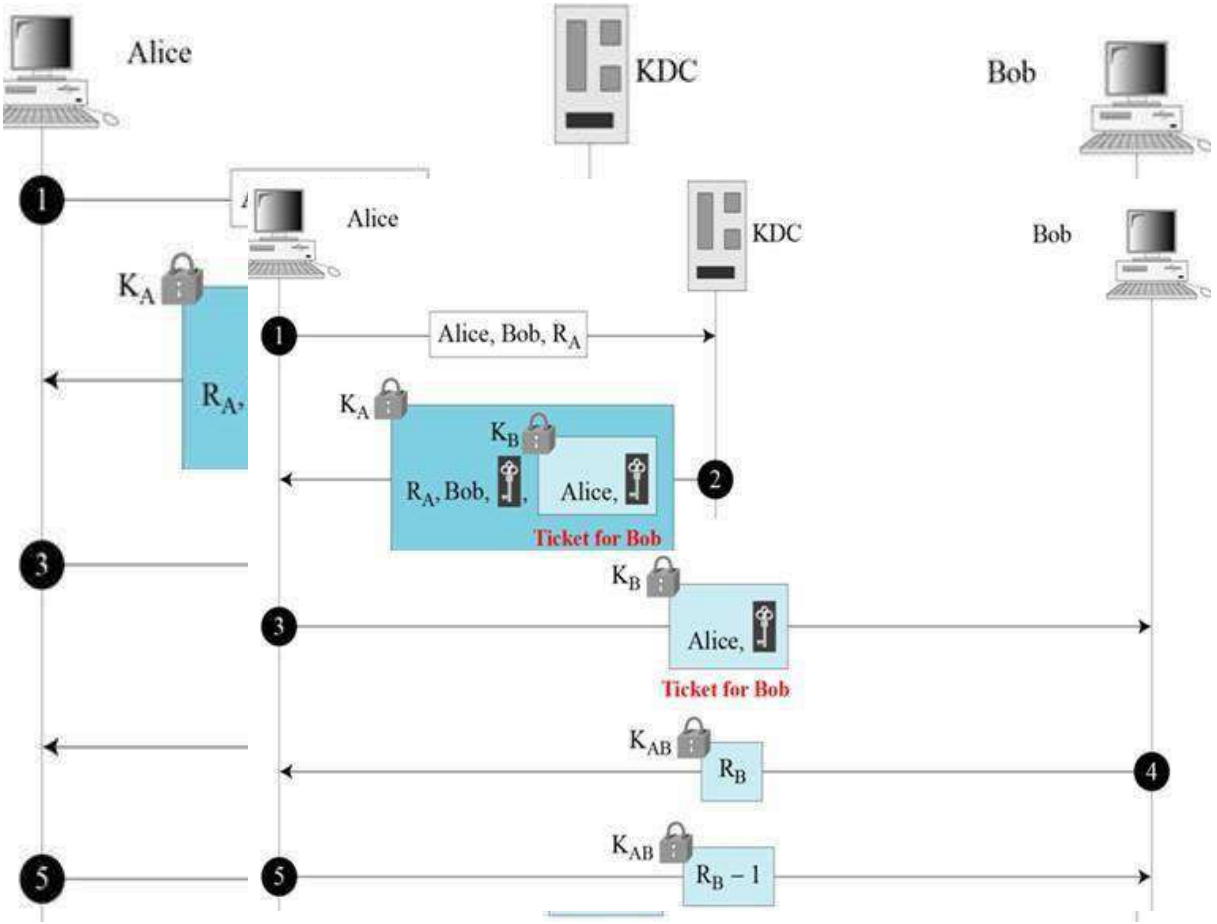
Figure shows first approach using KDC



1. Alice sends request to KDC
2. KDC creates ticket to Bob which is encrypted using Bob's key K_B. The ticket contains the session key (K_{AB}).
3. Alice extracts the Bob's ticket
4. Alice sends ticket to Bob. Bob opens the ticket and knows that Alice want to send message to him by using K_{AB}.

Drawback: Eve can use the replay attack at step 3.

Needham-Schroeder Protocol



1. Alice sends message to KDC that include her nonce, R_A
2. KDC sends encrypted ticket for Bob to Alice which contains session key
3. Alice sends Bobs ticket to him
4. Bob sends his challenge (R_B) to Alice which contains session key
5. Alice responds to Bobs challenge

KERBEROS

Kerberos is an authentication protocol, and at the same time a KDC, that has become very popular. Several systems, including Windows 2000, use Kerberos.
Originally designed at MIT, it has gone through several versions.

KERBEROS Servers

Three servers are involved in the Kerberos protocol.

Authentication Server (AS)

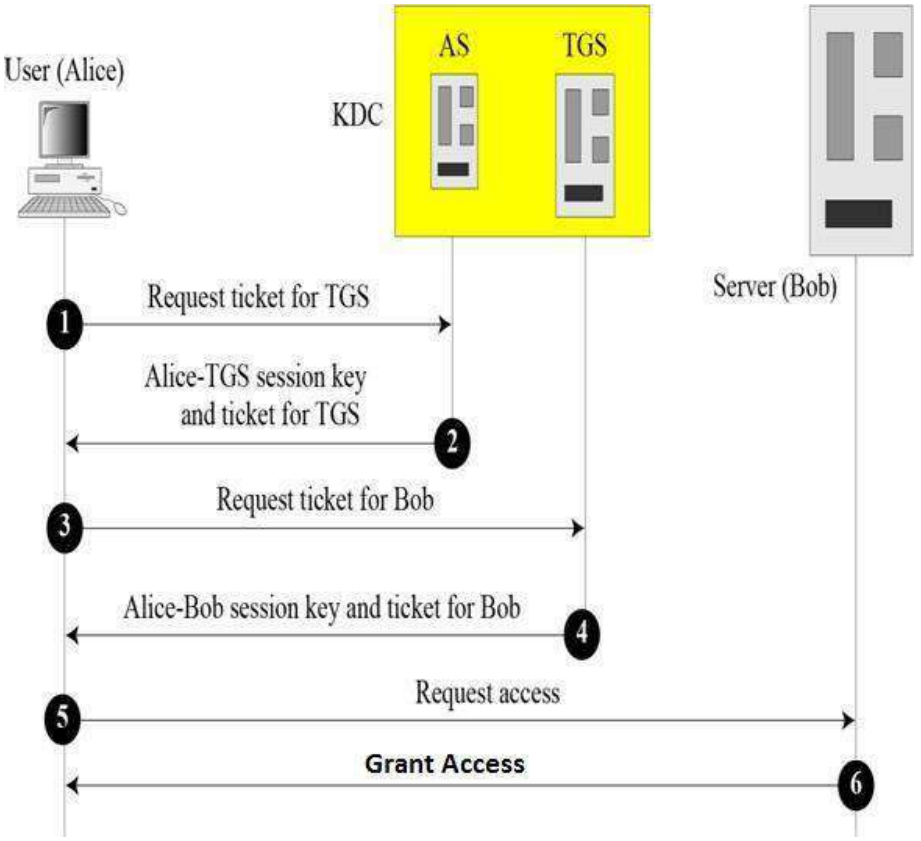
- ✓ The authentication server (AS) is the KDC in the Kerberos protocol.
- ✓ Each user registers with AS and is granted a user identity and a password.
- ✓ AS verifies the user, issues a session key to be used b/t Alice and TGS.
- ✓ and sends a ticket for TGS.

Ticket-Granting Server (TGS)

- ✓ The ticket-granting server (TGS) issues a ticket for the real server (Bob).
- ✓ Also provides the session key b/t Alice and Bob.
- ✓ Kerberos has a separated user verification from issuing of tickets.
- ✓ Alice can contact the TGS multiple times to obtained tickets for different real servers.

Real Server

- ✓ The real server (Bob) provides services for the user (Alice).
- ✓ Kerberos is designed for client-server programs.
- ✓ Kerberos is not used for person – to – person authentication

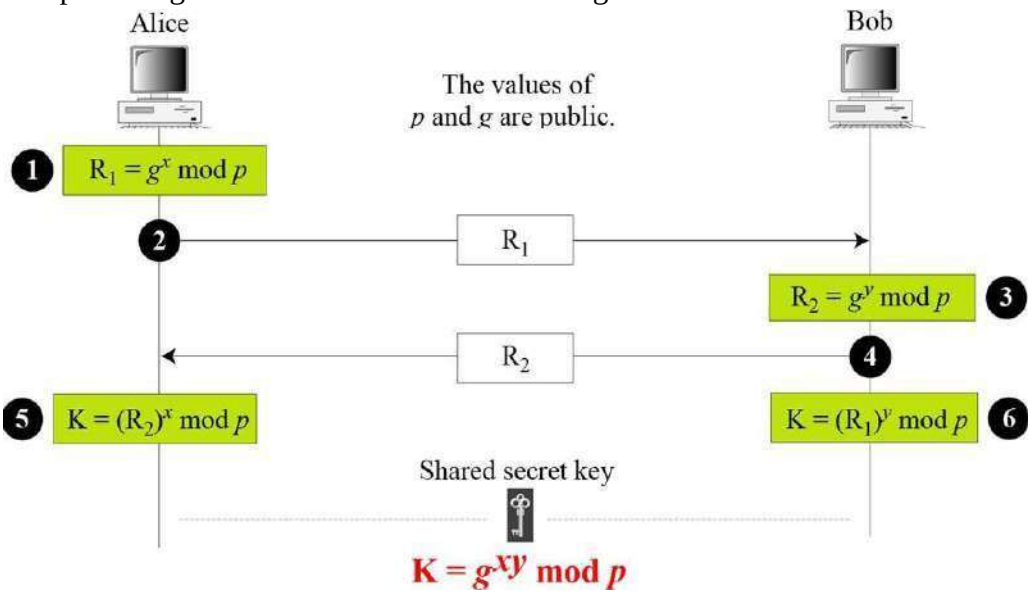


SYMMETRIC-KEY AGREEMENT

Alice and Bob can create a session key between themselves without using a KDC. This method of session-key creation is referred to as the symmetric-key agreement. Example: Diffie-Hellman Key Agreement

Diffie-Hellman Key Agreement

In this two parties are creating symmetric key without the need of a KDC.
Before establishing, the two parties need to choose two numbers **p and g**.
The p is a large number on the order of 300 digits.



Steps:

1. Alice chooses a large random integer number x and calculates $R1=g^x \bmod p$
2. Bob chooses another large number y and calculates $R2=g^y \bmod p$
3. Alice sends R1 to Bob and Bob sends R2 to Alice
4. Alice calculates key $K=(R2)^x \bmod p$
5. Bob calculates key $K=(R1)^y \bmod p$ Where K is the symmetric key for the session

The symmetric key in the Diffie-Hellman method is $K=g^{xy} \bmod p$

Diffie-Hellman Key Agreement- EXAMPLE

Let us give a trivial example to make the procedure clear. Our example uses small numbers, but note that in a real situation, the numbers are very large. Assume that $g = 7$ and $p = 23$. The steps are as follows:

1. Alice chooses $x = 3$ and calculates $R_1 = 7^3 \bmod 23 = 21$.
2. Bob chooses $y = 6$ and calculates $R_2 = 7^6 \bmod 23 = 4$.
3. Alice sends the number 21 to Bob.
4. Bob sends the number 4 to Alice.
5. Alice calculates the symmetric key $K = 43 \bmod 23 = 18$.

6. Bob calculates the symmetric key K = 216 mod 23 = 18.
7. The value of K is the same for both Alice and Bob;

g^{xy} mod p = 7^{18} mod 35 = 18.

PUBLIC-KEY DISTRIBUTION

In asymmetric-key cryptography, people do not need to know a symmetric shared key; everyone shields a private key and advertises a public key.
In public key key cryptography, everyone have access to every one's public key: public keys are available to the public.
So, public keys need to be distributed.

1. Public Announcement
2. Trusted Center
3. Controlled Trusted Center
4. Certification Authority

5. X.509
6. Public-Key Infrastructures (PKI)

Public Announcement

The normal method is to announce public keys publicly, but is not secure

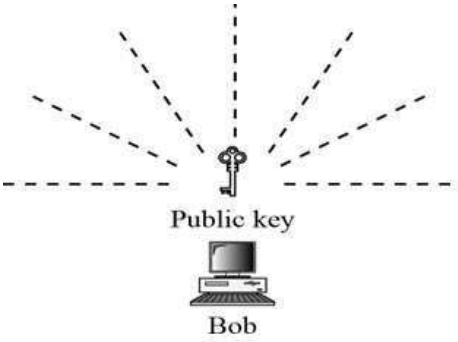
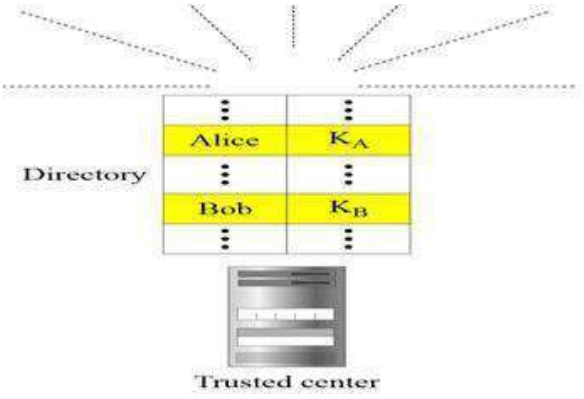


Figure *Announcing a public key*

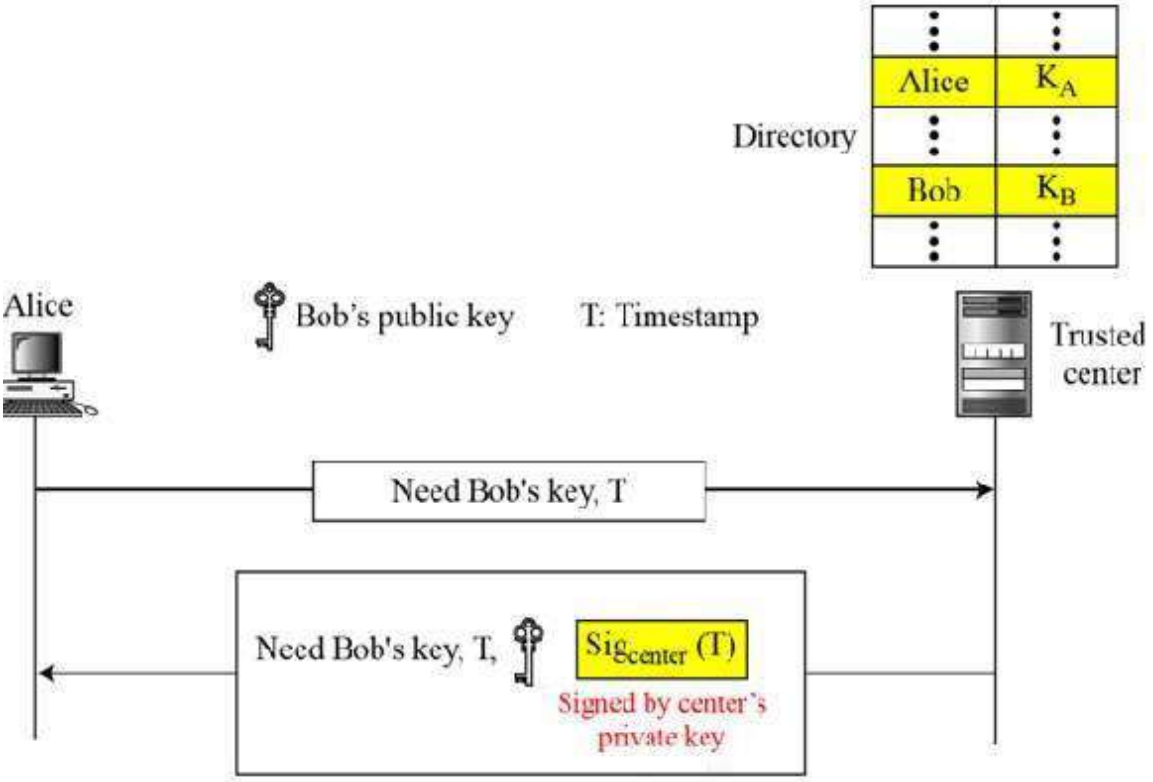
Trusted Center

A more secure approach is to have a trusted center retain a directory of public keys



Controlled Trusted Center

A higher level security can be achieved when there are added controls on



UNIT –V

Network Security-I

Security at application layer: PGP and S/MIME, Security at the Transport Layer: SSL and TLS

Security at Application layer and Transport Layer

Various business services are now offered online though client-server applications.
The most popular forms are web application and e-mail.
In both applications, the client communicates to the designated server and obtains services.

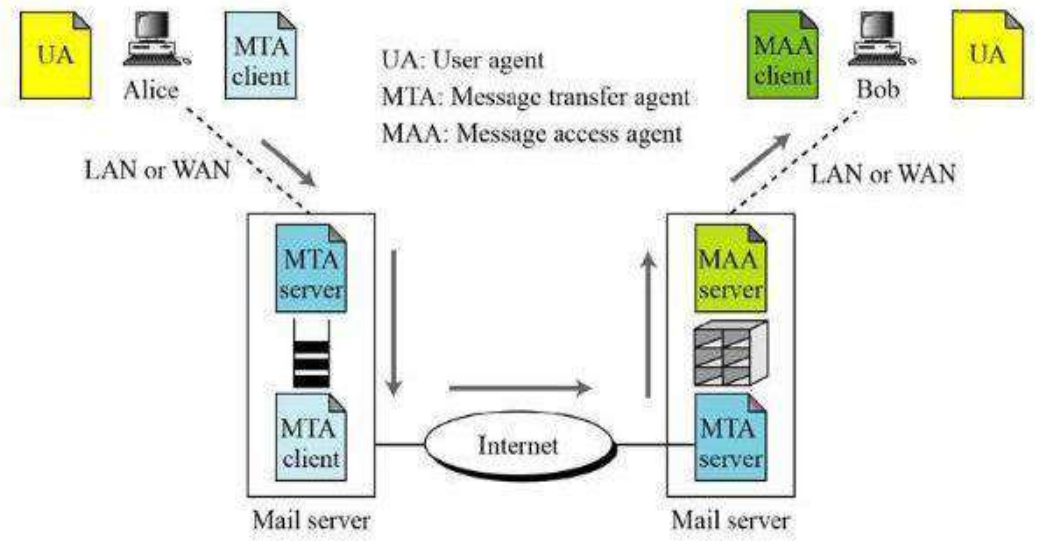
While using a service from any server application, the client and server exchange a lot of information on the underlying intranet or Internet. We are aware of fact that these information transactions are vulnerable to various attacks.
Network security entails securing data against attacks while it is in transit on a network.

E-mail Security

Nowadays, e-mail has become very widely used network application. Email is one of the most widely used and regarded network services. Currently message contents are not secure, may be inspected either in transit or by suitably privileged users on destination system.

E-mail Architecture:

Figure *E-mail architecture*



1. UA-User Agent is useful to prepare the messages
2. MTA-Message Transfer Agent is useful to send messages to mail server. This is the Push program
3. MAA-Message Access Agent is useful to receive messages from mail server. This is Pull program

PGP(Pretty Good Privacy)

- Provides a confidentiality and authentication service that can be used for electronic mail and file storage applications
- Developed by Phil Zimmermann
- Selected the best available cryptographic algorithms as building blocks
- Integrated these algorithms into a general-purpose application that is independent of operating system and processor and that is based on a small set of easy-to-use commands
- Made the package and its documentation, including the source code, freely available via the Internet, bulletin boards, and commercial networks
- Entered into an agreement with a company to provide a fully compatible, low –cost commercial version of PGP

PGP Growth

It is available free worldwide in versions that run on a variety of platforms
• The commercial version satisfies users who want a product that comes with vendor support
• It is based on algorithms that have survived extensive public review and are considered extremely secure
• It has a wide range of applicability
• It was not developed by, nor is it controlled by, any governmental or standards organization
• Is now on an Internet standards track, however it still has an aura of an antiestablishment endeavor.

PGP Notation:

Ks = session key used in symmetric encryption scheme
PRa = private key of user A, used in public-key encryption scheme
PUa = public key of user A, used in public-key encryption scheme
EP = public-key encryption
DP = public-key decryption
EC = symmetric encryption
DC = symmetric decryption
H = hash function
|| = concatenation
Z = compression using ZIP algorithm
R64 = conversion to radix 64 ASCII format1

PGP Operation – Authentication:

1. sender creates a message
2. SHA-1 used to generate 160-bit hash code of message
3. hash code is encrypted with RSA using the sender's private key, and result is attached to message
4. receiver uses RSA or DSS with sender's public key to decrypt and recover hash code
5. receiver generates new hash code for message and compares with decrypted hash code, if match, message is accepted as authentic