UNIT 3

Discrete logarithms:

- In the RSA algorithm, we saw how the difficulty of factoring yields useful cryptosystems.
- There is another number theory problem, namely discrete logarithms, that has similar applications.
- Fix *a* prime *p*. Let α and β be non zero integers mod p and suppose

$$\beta \equiv \alpha^x \pmod{p}.$$

- The problem of finding x is called the discrete logarithm problem.
- If n is the smallest positive integer such that $\alpha^n \equiv 1 \pmod{p}$, we may assume $0 \le x \le n$, and then we denote

$$x = L_{\alpha,p}(\beta)$$

and call it the discrete log of *β* with respect to α.

Eg. let p = 11, *β = 9* and let α = 2. since $2^6 \equiv 9 \pmod{11}$, we have $L_{2,11}(9) = 6$. Of course, $2^6 \equiv 2^{16} \equiv 2^{26} \equiv 9 \pmod{11}$,so we could consider taking any one of 6, 16, 26 as the discrete logarithm. But we fix the value by taking the smallest nonnegative value, namely 6.note that we could have defined the discrete logarithm in this case to be the congruence class (6 mod).

Often, α is taken to be a primitive root mod p, which means that every *β* is a power of α(mod p). That is, if ***a*** is a primitive root of the prime number *p*, then the numbers

$a \bmod p, a^2 \bmod p, \ldots, a^{p-1} \bmod p$ are distinct and consist of the integers from 1 through p – 1 in some permutation.

- If α is not a primitive root, then the discrete logarithm will not be defined for certain values of *β*.

Given a prime p, it is fairly easy to find a primitive root in many cases.The discrete log behaves in many ways like the usual logarithm. In particular, if α is a primitive root mod p, then

$$L_{\alpha,p}(\beta_1\beta_2) \equiv L_{\alpha,p}(\beta_1) + L_{\alpha,p}(\beta_2) \pmod{p-1}$$

When p is small, it is easy to compute discrete logs by exhaustive search through all possible exponents. However, when p is large this is not feasible.

One way function:

- A function f (x) is called a **one-way function** if f(x) is easy to compute, but, given y, it is computationally in feasible to find x with f(x) = y.

- modular exponentiation is probably an example of such a function. It is easy compute α ^{x} (mod p), but solving α ^{x} ≡ β for x is probably hard.

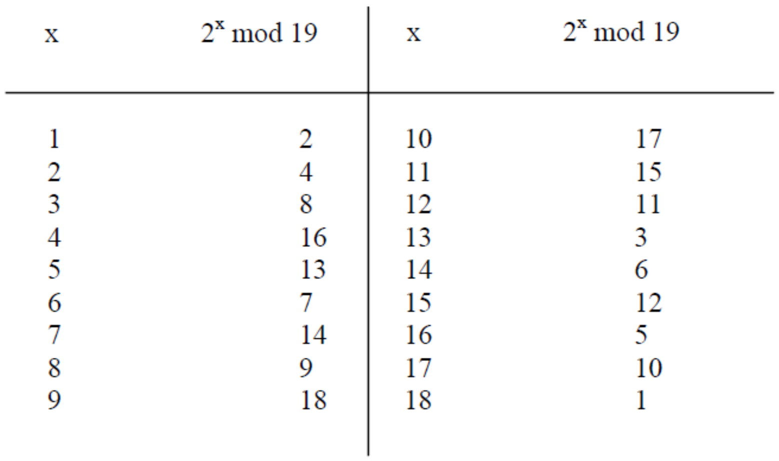
- Multiplication of large primes can also be regarded as a (probable) one-way function: it is easy to multiply primes but difficult to factor the result to recover the primes.

- One-way functions have many cryptographic uses.

Eg.let p = 19, d = 2 and β = 14. we are then looking for a choice of x such that

$2^x \equiv 14 \pmod{19}$. $\alpha^x \equiv \beta$ mod p.

We proceed by systematic trial and error, calculating values of 2^x modulo 19 until we either find an x that works or we exhaust all the possibilities and find that there is no solution. The following table shows a complete set of values, and we see that x = 7 is the only solution.



x	2^x mod 19	x	2^x mod 19
1	2	10	17
2	4	11	15
3	8	12	11
4	16	13	3
5	13	14	6
6	7	15	12
7	14	16	5
8	9	17	10
9	18	18	1

Key Management:

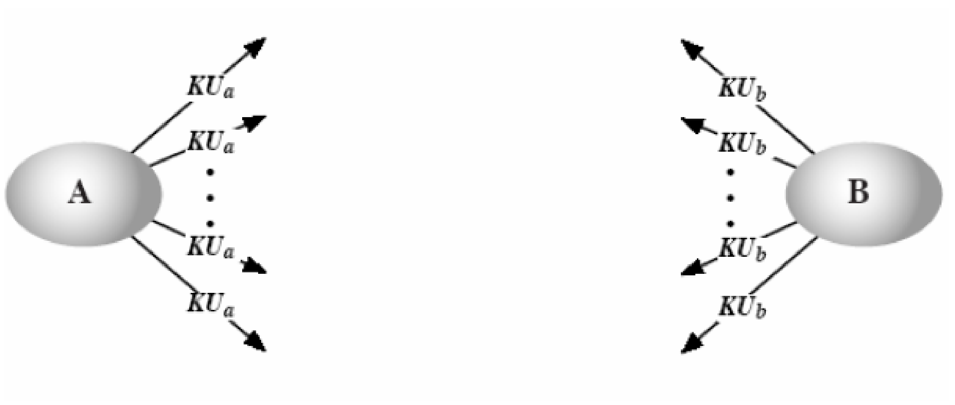
There are actually two distinct aspects to the use of public-key encryption in this regard:
- The distribution of public keys.
- The use of public-key encryption to distribute secret keys.

Several techniques have been proposed for the distribution of public keys. Virtually all of these proposals can be grouped into the following general schemes:
A. Public announcement
B. Publicly available directory
C. Public-key authority
D. Public-key certificates

Public Announcement:

- Users distribute public keys to recipients or broadcast to community at large
    - eg. append PGP keys to email messages or post to news groups or email list
- Major weakness is forgery

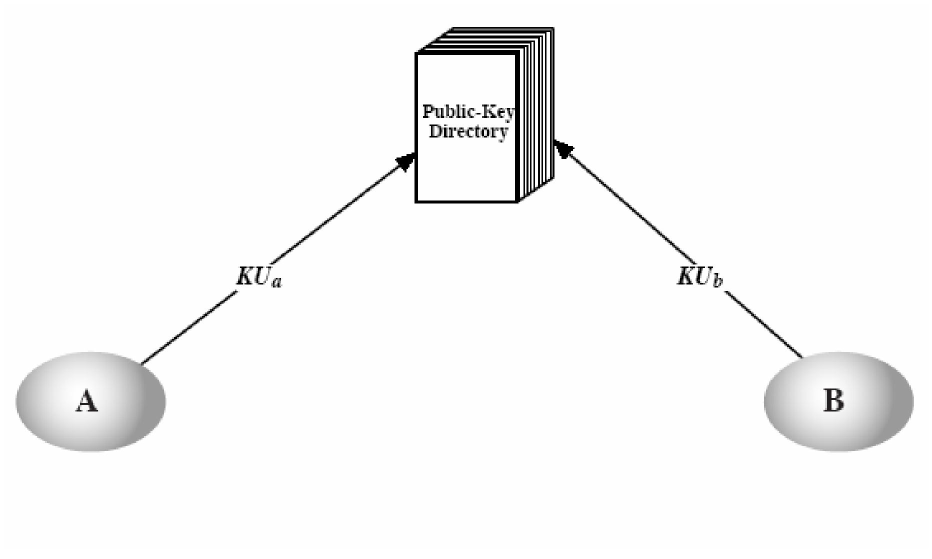
- anyone can create a key claiming to be someone else and broadcast it
- until forgery is discovered can masquerade as claimed use



<u>Publicly Available Directory:</u>

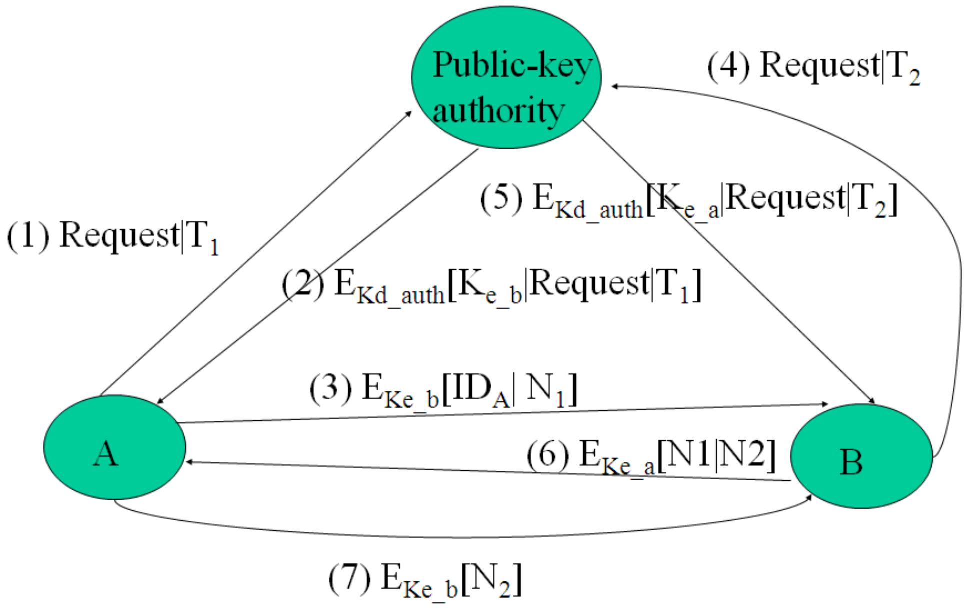
- Can obtain greater security by registering keys with a public directory
- Directory must be trusted with properties:
  - contains { name, public-key} entries
  - participants register securely with directory
  - participants can replace key at any time
  - directory is periodically published
  - directory can be accessed electronically

Still vulnerable to tampering or forgery



<u>Public-Key Authority:</u>

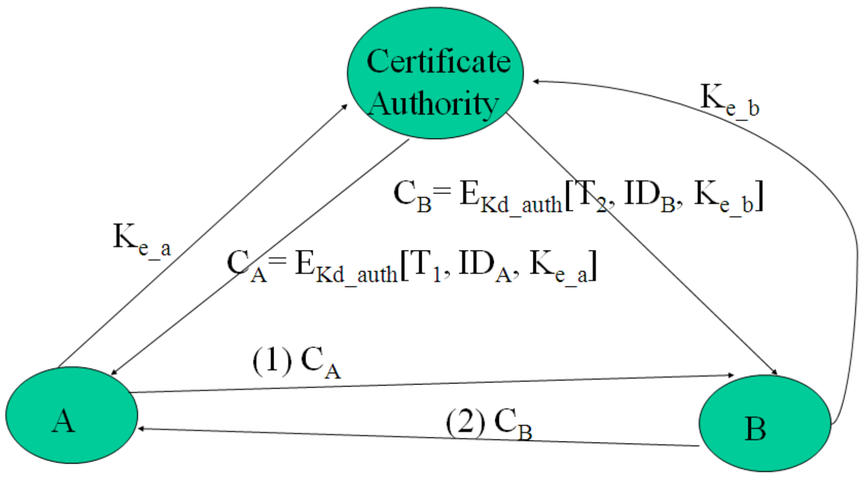
- Improve security by tightening control over distribution of keys from directory
- Has properties of directory
- And requires users to know public key for the directory
- Then users interact with directory to obtain any desired public key securely
  - does require real-time access to directory when keys are needed


Public-key authority
(4) Request|T_2
(5) $E_{Kd\_auth}[K_{e\_a}|Request|T_2]$
(1) Request|T_1
(2) $E_{Kd\_auth}[K_{e\_b}|Request|T_1]$
(3) $E_{Ke\_b}[ID_A| N_1]$
A
(6) $E_{Ke\_a}[N1|N2]$
B
(7) $E_{Ke\_b}[N_2]$

<u>Public-Key Certificates:</u>

- Certificates allow key exchange without real-time access to public-key authority
- A certificate binds **identity to public key**
  - usually with other info such as period of validity, rights of use etc
- With all contents **signed by a trusted Public-Key or Certificate Authority** (CA)

Can be verified by anyone who knows the certificate authorities public-key

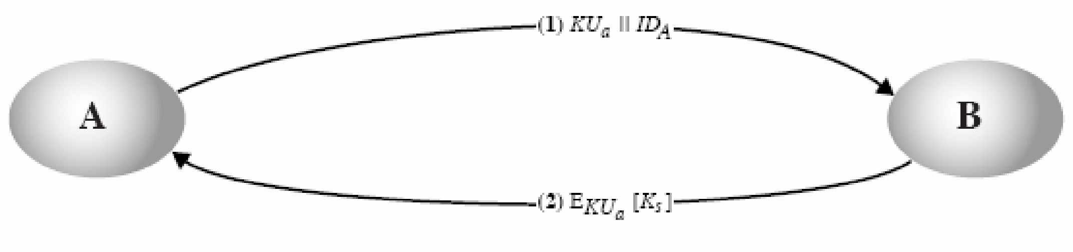

Certificate Authority
K_{e_b}
$C_B = E_{Kd\_auth}[T_2, ID_B, K_{e\_b}]$
K_{e_a}
$C_A = E_{Kd\_auth}[T_1, ID_A, K_{e\_a}]$
(1) C_A
A
(2) C_B
B

B does: $D_{Ke\_auth}(C_A) = D_{Ke\_auth}(E_{Kd\_auth}[T_1, ID_A, K_{e\_a}]) = (T_1, ID_A, K_{e\_a})$, hence gets the public key of A

•

Public-Key Distribution of Secret Keys

- Use previous methods to obtain public-key
- Can use for secrecy or authentication
- But public-key algorithms are slow
- So usually want to use private-key encryption to protect message contents
- Hence need a session key
- Have several alternatives for negotiating a suitable session



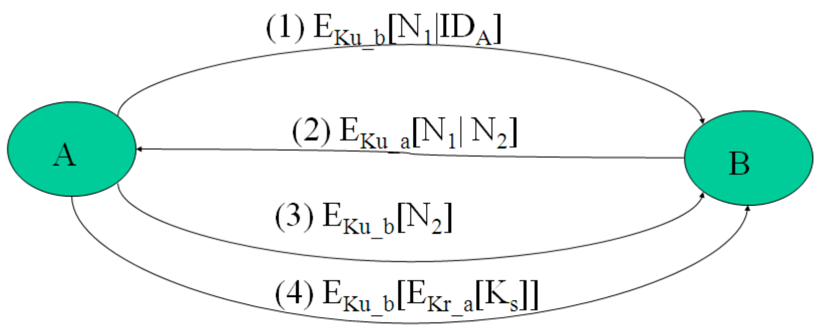
If A wishes to communicate with B, the following procedure is employed:

1. A generates a public/private key pair {KUA,KRA} and transmits a message to B consisting of KUA, 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 DKRA[EKUA[Ks]] to recover the secret key. Since only A can decrypt the message, only A and B will know the identity of Ks.

4. A discards KUA and KRA and B discards KUA.


(1) $E_{Ku\_b}[N_1|ID_A]$
A
(2) $E_{Ku\_a}[N_1| N_2]$
B
(3) $E_{Ku\_b}[N_2]$
(4) $E_{Ku\_b}[E_{Kr\_a}[K_s]]$

For this example it is assumed that A and B have exchanged public keys by one of the schemes described earlier in this section. Then the following steps occur:

1. A uses B's public key to encrypt a message to B containing an identifier of A (IDA) and a nonce (N1), which is used to uniquely identify this transaction.

2. B sends a message to A encrypted with KUA and containing A's nonce (N1) as well as a new nonce generated by B (N2). Since only B could have decrypted message (1), the presence of N1 in message (2) assures A that the correspondent is 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=EKUb[EKRA[Ks]] to B. Encryption of this message with B's public key ensures that only B can read it, encryption with A's private key ensures that only A could have sent it.

5. B computes DKUA[DKRb [M]] to recover the secret key.

Hybrid Scheme:

- Key Distribution Center(KDC) uses a public key scheme to distribute a secret master key with each user.
- KDC distribute secret session key encrypted with master key.
- Performance: Distribution of session keys by public key encryption could degrade overall system. With a hybrid scheme public key encryption is used occasionally to update the master key
- single KDC serves a widely distributed set of users
- Used in IBM mainframes

<u>Primitive root:</u>

- A primitive root of a prime number p as one whose powers generates, all the integers from 1 to 1-p
- i.e If **a** is a primitive root of prime number p then the numbers

a mod p,a^2 mod p,…. a^{p-1} mod p are distinct and Consist of numbers from 1 to p -1.

(i.e) $b \equiv a^i \bmod p$ where $0 \leq i \leq (p-1)$

Where ‘i’ is called as discrete logarithm or index

Eg.

The number 3 is a primitive root in (mod 7) because
$3^1=3\equiv3 \pmod 7$
$3^2=9\equiv2 \pmod 7$
$3^3=27\equiv6 \pmod 7$
$3^4=81\equiv4 \pmod 7$
$3^5=243\equiv3 \times 3^4 \pmod 7\equiv3 \times 4\equiv12 \pmod 7\equiv 5 \pmod 7$
$3^6=729\equiv6^2 \pmod 7 \equiv 36 \pmod 7 \equiv 1 \pmod 7$

Here we see that the **period** of 3^k modulo 7 is 6 .

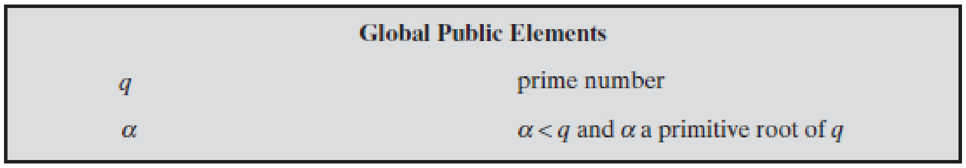
The remainders are 3,2,6,4,5,1.

Diffie-Hellman Key Exchange:

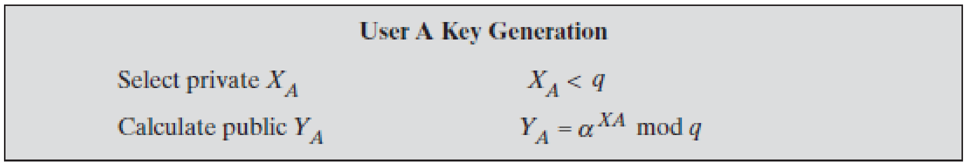
- This algorithm enables two user to exchange a key securely.
- That can be used for subsequent encryption of message.
- This algorithm uses two publicly known numbers
    - Prime number q .
    - Primitive root α (primitive root of q)
- User A and B selects two random number X_A and X_B
- Such that X_A and $X_B < q$
- Then A and B compute Y_A and Y_B such that

$$\mathbf{Y_A = \alpha^{X_A} \bmod q}$$

$$\mathbf{Y_B = \alpha^{X_B} \bmod q}$$



Global Public Elements	
q	prime number
α	$\alpha < q$ and α a primitive root of q



User A Key Generation	
Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = \alpha^{XA} \bmod q$

User B Key Generation

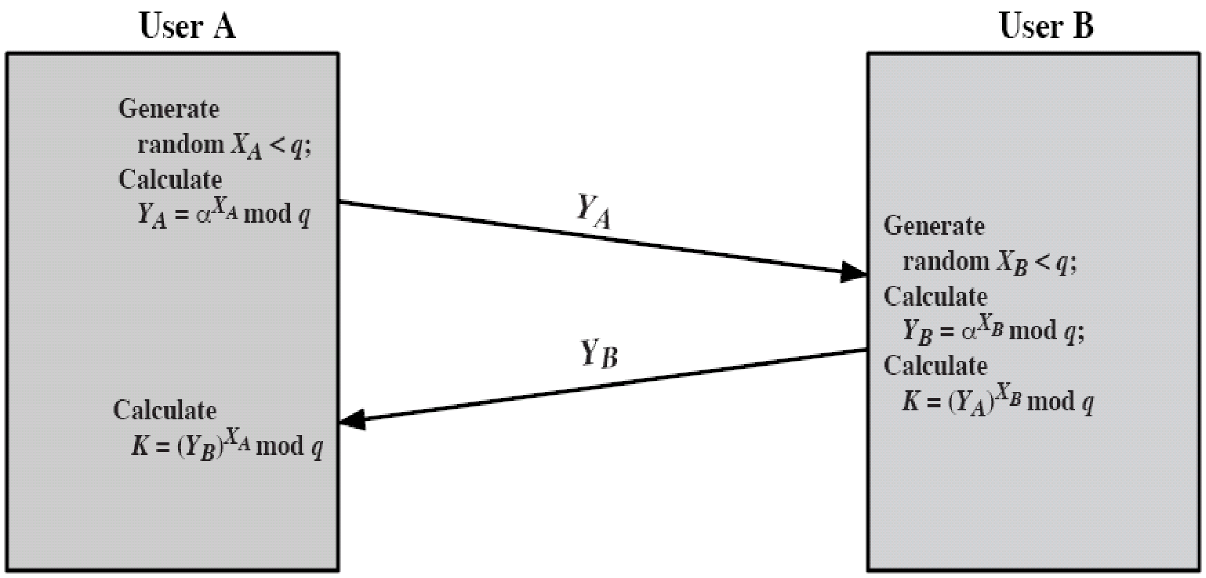
Select private X_B	$X_B < q$
Calculate public Y_B	$Y_B = \alpha^{XB} \bmod q$

Calculation of Secret Key by User A

$$K = (Y_B)^{X_A} \bmod q$$

Calculation of Secret Key by User B

$$K = (Y_A)^{X_B} \bmod q$$


User A
User B
Generate random $X_A < q$;
Calculate $Y_A = \alpha^{X_A} \bmod q$
Y_A
Generate random $X_B < q$;
Calculate $Y_B = \alpha^{X_B} \bmod q$;
Calculate $K = (Y_A)^{X_B} \bmod q$
Y_B
Calculate $K = (Y_B)^{X_A} \bmod q$

$K = (Y_B)^{XA} \bmod q$

$= (\alpha^{XB} \bmod q)^{XA} \bmod q$

$= (\alpha^{XB})^{XA} \bmod q$

$= \alpha^{XBXA} \bmod q$

$= (\alpha^{XA})^{XB} \bmod q$

$= (\alpha^{XA} \bmod q)^{XB} \bmod q$

$= (Y_A)^{XB} \bmod q$ (by rules of modular arithmetic)

The result is that the two sides have exchanged a secret value. Furthermore, because XA and XB are private, an adversary only has the following ingredients to work with: q, α, YA, and YB. Thus, the adversary is forced to take a discrete logarithm to determine the key. For example, to determine the private key of user B, an adversary must compute

$$XB = \mathbf{dlog}_{a,\, q}\,(YB)$$

The adversary can then calculate the key *K* in the same manner as user B calculates it.

The security of the Diffie-Hellman key exchange lies in the fact that, while it is relatively easy to calculate exponentials modulo a prime, it is very difficult to calculate discrete logarithms. For large primes, the latter task is considered infeasible. Here is an example. Key exchange is based on the use of the prime number

EXAMPLE:

q = 353, primitive root of q, in this case, α = 3

A and B selects secret keys $X_A = 97$ and $X_B = 233$

Each computes public key;

A computes $Y_A = 3^{97} \bmod 353 = 40$,

B computes $Y_B = 3^{233} \bmod 353 = 248$

After exchanging public keys, each compute the common secret key:

A computes $K = (Y_B)^{XA} \bmod 353 = 248^{97} \bmod 353=160$

B computes $K = (Y_A)^{XB} \bmod 353 = 40^{233} \bmod 353=160$

ELGAMAL CRYPTOGRAPHIC SYSTEM:

As with Diffie-Hellman, the global elements of ElGamal are a prime number q and α, which is a primitive root of q.

User A generates a private/public key pair as follows:

1. Generate a random integer $\mathbf{X_A}$, such that $1<X_A<q-1$.

2. Compute $\mathbf{Y^A= \alpha^{XA} \bmod q}$.

3. A's private key is $\mathbf{X_A}$;A's pubic key is $\mathbf{\{q, \alpha, Y^A\}}$

Any user B that has access to A's public key can encrypt a message as follows:

1. Represent the message as an integer M in the range 0<=M<=(q-1). Longer messages are sent as a sequence of blocks, with each block being an integer less than q.

2. Choose a random integer **k** such that 1<=k<=(q-1) .

3. Compute a one-time key $\mathbf{K=(Y_A)^k \bmod q}$.

4. Encrypt M as the pair of integers **(C1,C2)** where

$\boldsymbol{C_1 = \alpha^{k} \bmod q;}$

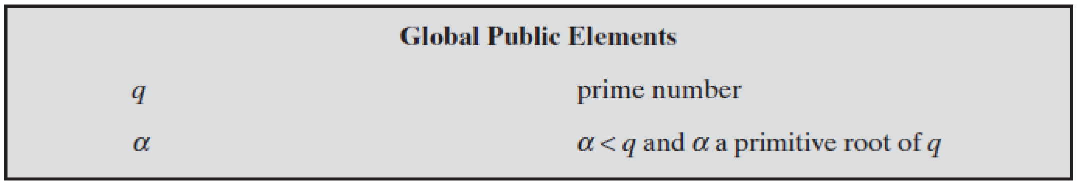
$\boldsymbol{C_2 = KM \bmod q}$

User A recovers the plaintext as follows:

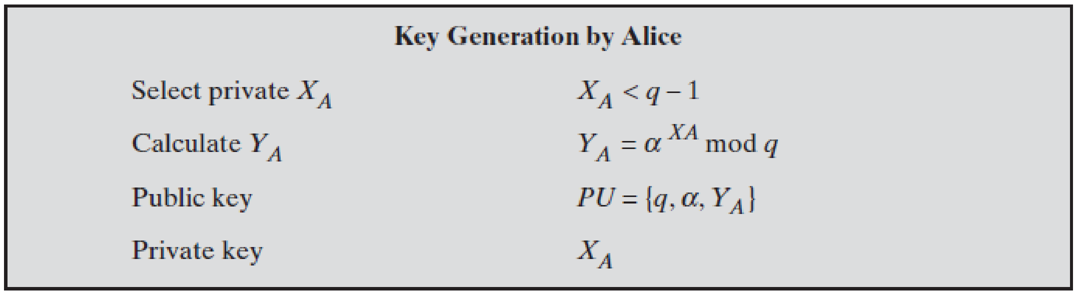
1. Recover the key by computing

   $\boldsymbol{K = (C_1)^{XA} \bmod q}$

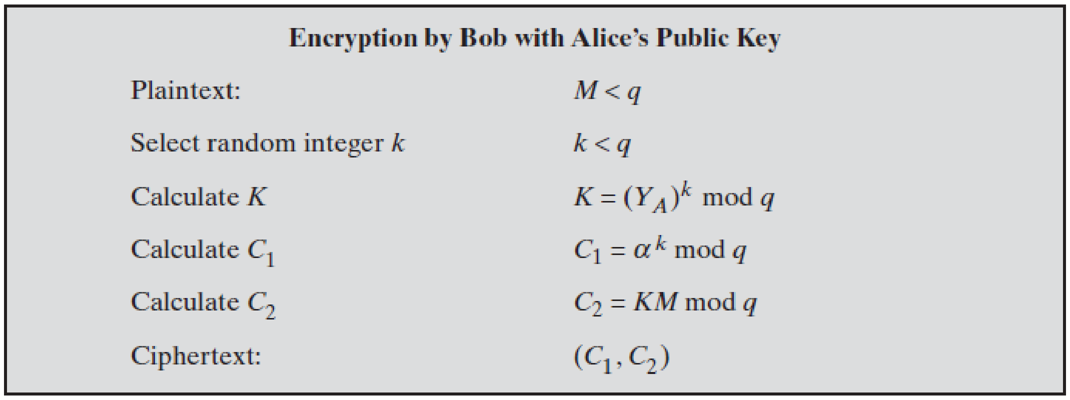
2. Compute $\mathbf{M = (C_2K^{-1}) \bmod q}$



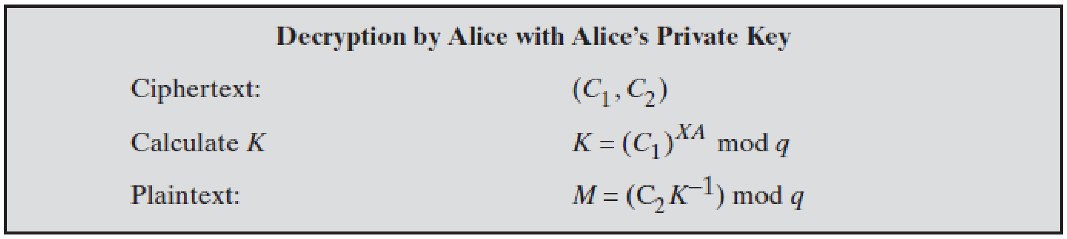
Global Public Elements	
q	prime number
α	$\alpha < q$ and α a primitive root of q



Key Generation by Alice	
Select private X_A	$X_A < q - 1$
Calculate Y_A	$Y_A = \alpha^{XA} \bmod q$
Public key	$PU = \{q, \alpha, Y_A\}$
Private key	X_A



Encryption by Bob with Alice's Public Key	
Plaintext:	$M < q$
Select random integer k	$k < q$
Calculate K	$K = (Y_A)^k \bmod q$
Calculate C_1	$C_1 = \alpha^k \bmod q$
Calculate C_2	$C_2 = KM \bmod q$
Ciphertext:	(C_1, C_2)



Decryption by Alice with Alice's Private Key	
Ciphertext:	(C_1, C_2)
Calculate K	$K = (C_1)^{XA} \bmod q$
Plaintext:	$M = (C_2 K^{-1}) \bmod q$

Let us demonstrate why the ElGamal scheme works. First, we show how K is recovered by the decryption process:

$$K = (Y_A)^k \bmod q$$
$$K = (\alpha^{X_A} \bmod q)^k \bmod q$$
$$K = \alpha^{kX_A} \bmod q$$
$$K = (C_1)^{X_A} \bmod q$$

Next, using , we recover the plaintext as

$$C_2 = KM \bmod q$$
$$(C_2K^{-1}) \bmod q = KMK^{-1} \bmod q = M \bmod q = M$$

For example, let us start with the prime field GF(19); that is, q =19. It has primitive roots {2, 3, 10, 13, 14, 15}.We choose α =10 Alice generates a key pair as follows:

1. Alice chooses XA = 5.
2. Then $Y^A = \alpha^{XA}$ mod q = 10^5mod 19 = 3
3. Alice's private key is 5; Alice's pubic key is {q, a, YA} = {19, 10, 3}.

Suppose Bob wants to send the message with the value $M = 17$. Then,

1. Bob chooses $k = 6$.
2. Then $K = (Y_A)^k \bmod q = 3^6 \bmod 19 = 729 \bmod 19 = 7$.
3. So
   $C_1 = \alpha^k \bmod q = \alpha^6 \bmod 19 = 11$
   $C_2 = KM \bmod q = 7 \times 17 \bmod 19 = 119 \bmod 19 = 5$
4. Bob sends the ciphertext (11, 5).

For decryption:

1. Alice calculates $K = (C_1)^{X_A} \bmod q = 11^5 \bmod 19 = 161051 \bmod 19 = 7$.
2. Then K^{-1} in GF(19) is $7^{-1} \bmod 19 = 11$.
3. Finally, $M = (C_2K^{-1}) \bmod q = 5 \times 11 \bmod 19 = 55 \bmod 19 = 17$.

If a message must be broken up into blocks and sent as a sequence of encrypted blocks, a unique value of k should be used for each block. If k is used for more than one block, knowledge of one block m1 of the message enables the user to compute other blocks as follows.

$$C_{1,1} = \alpha^k \bmod q;\ C_{2,1} = KM_1 \bmod q$$

$$C_{1,2} = \alpha^k \bmod q;\ C_{2,2} = KM_2 \bmod q$$

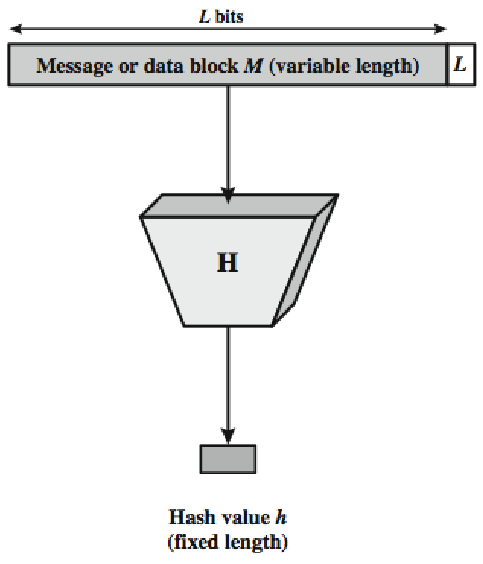
$$\frac{C_{2,1}}{C_{2,2}} = \frac{KM_1 \bmod q}{KM_2 \bmod q} = \frac{M_1 \bmod q}{M_2 \bmod q}$$

If M_1 is known, then M_2 is easily computed as

$$M_2 = (C_{2,1})^{-1} C_{2,2} M_1 \bmod q$$

Hash function

- Converts a variable size message M into fixed size hash code H(M) (*message digest*)
- a hash code does not use a key but is a function.

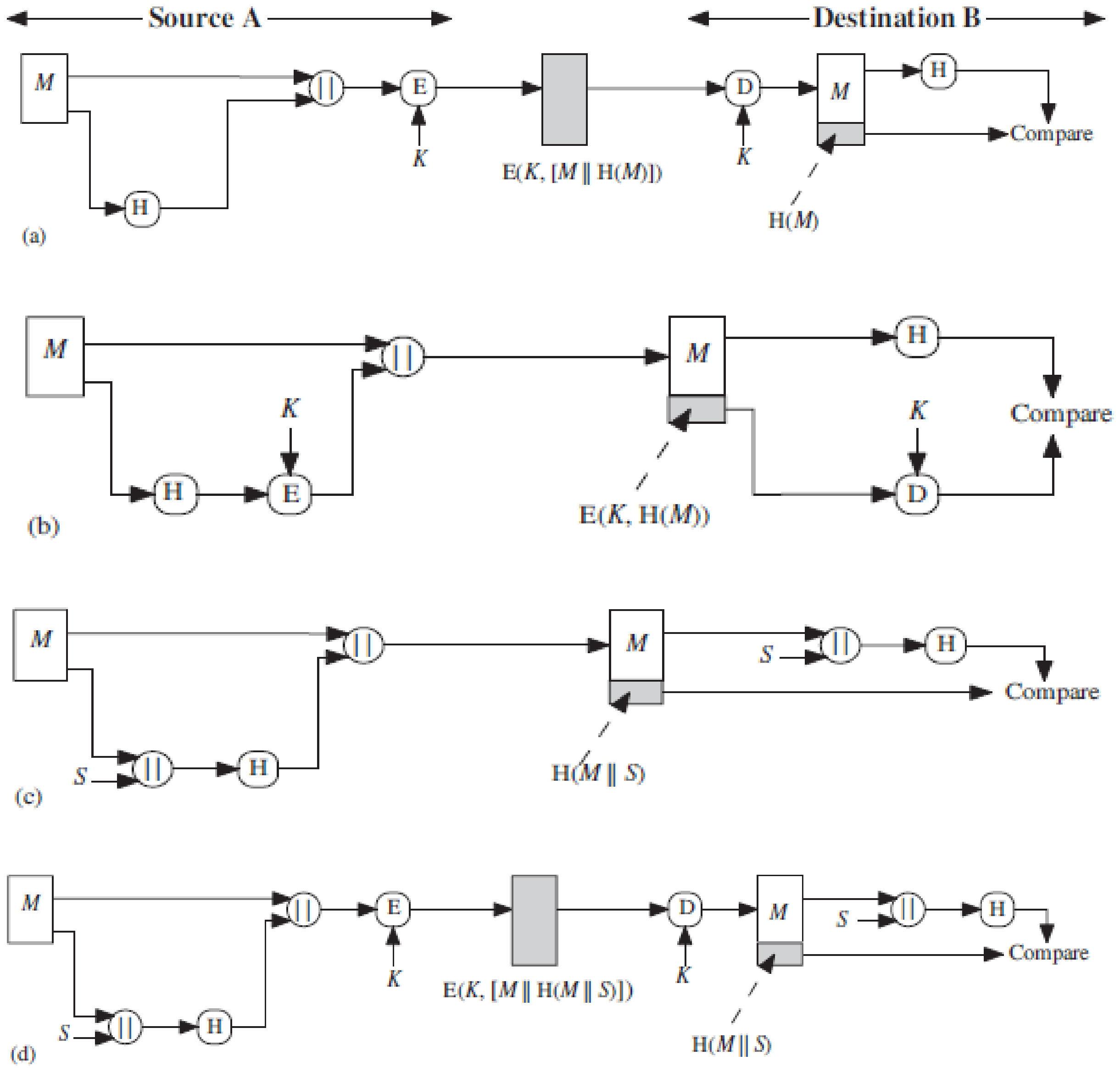


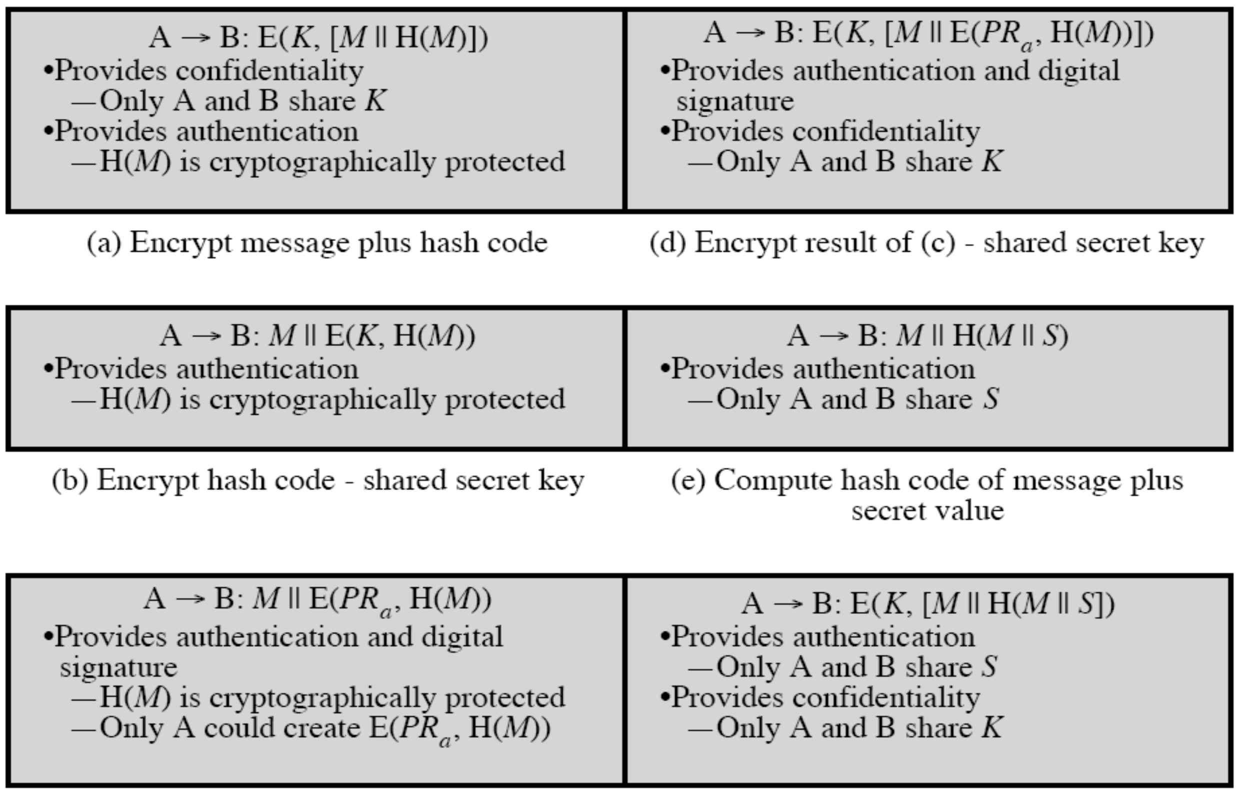
- h = H(M)
  - M is a variable-length message
  - h is a fixed-length hash value
  - H is a hash function
- The hash value is appended at the source
- The receiver authenticates the message by re-computing the hash value
- Because the hash function itself is not considered to be secret, some means is required to protect the hash value

<u>Applications of Cryptographic Hash Function:</u>

Message Authentication:

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





$A \rightarrow B: E(K, [M \| H(M)])$
- •Provides confidentiality
  - —Only A and B share K
- •Provides authentication
  - —$H(M)$ is cryptographically protected

(a) Encrypt message plus hash code

$A \rightarrow B: M \| E(K, H(M))$
- •Provides authentication
  - —$H(M)$ is cryptographically protected

(b) Encrypt hash code - shared secret key

$A \rightarrow B: M \| E(PR_a, H(M))$
- •Provides authentication and digital signature
  - —$H(M)$ is cryptographically protected
  - —Only A could create $E(PR_a, H(M))$

(c) Encrypt hash code - sender's private *key*

$A \rightarrow B: E(K, [M \| E(PR_a, H(M))])$
- •Provides authentication and digital signature
- •Provides confidentiality
  - —Only A and B share K

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

$A \rightarrow B: M \| H(M \| S)$
- •Provides authentication
  - —Only A and B share S

(e) Compute hash code of message plus secret value

$A \rightarrow B: E(K, [M \| H(M \| S])$
- •Provides authentication
  - —Only A and B share S
- •Provides confidentiality
  - —Only A and B share K

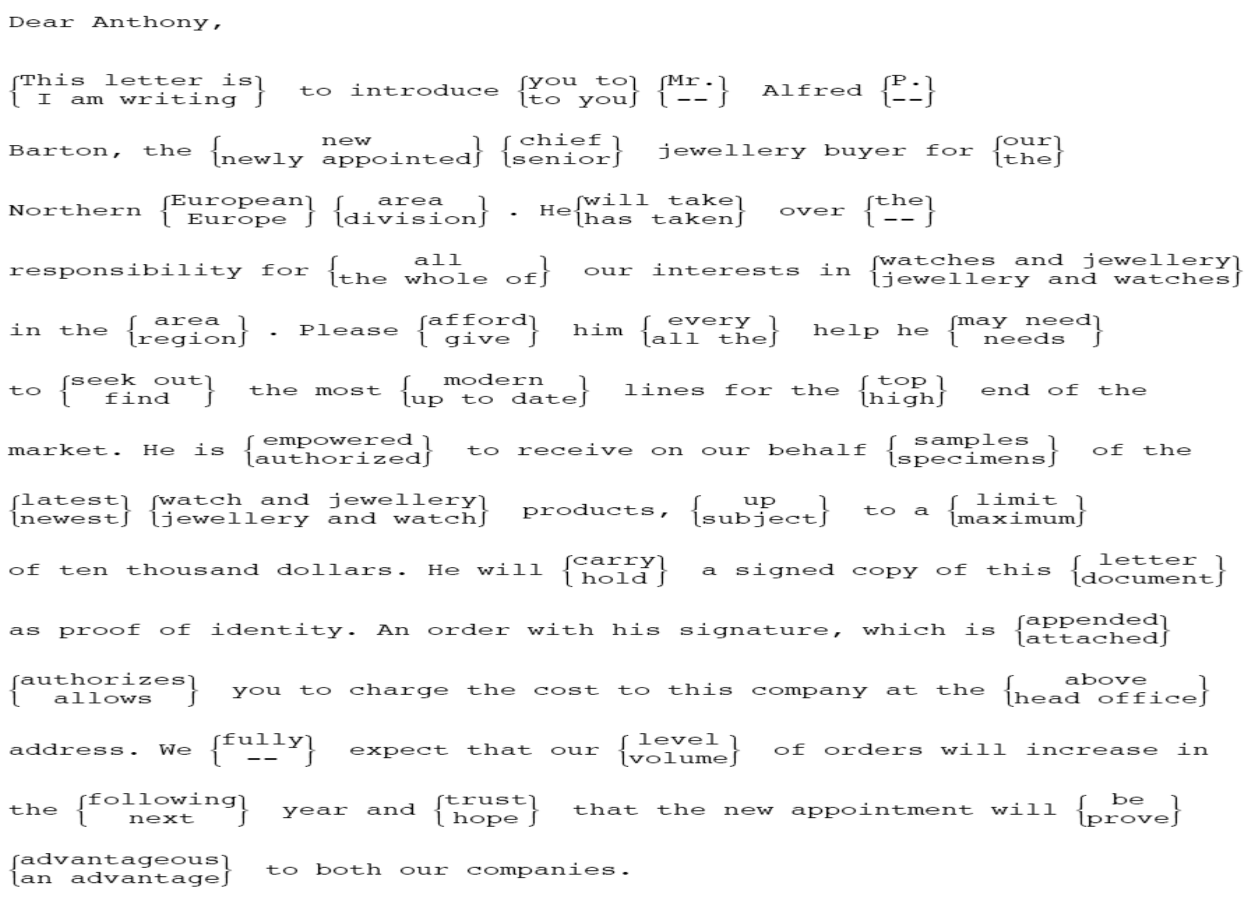
(f) Encrypt result of (e)

<u>Requirements of Hash Functions:</u>

- H can be applied to block of data of any size.
- H produces a fixed length output.
- H(x) is relatively easy to compute for any given x.
- For any given value h, it is computationally infeasible to find x, such that H(x) =h, also called as one way property.
- For any given block x, it is computationally infeasible to find y ≠ x, such that H(y)= H(x), also called as **weak collision resistance**.
- It is infeasible to find (x,y) such that H(x)= H(y) also called as **strong collision resistance.**

<u>Birthday Attack:</u>

- Type of attack in which an opponent would have to try about $2^{(\text{hashcode size -1})}$ messages to find one that matches the hash code of the intercepted messges.
- If the adversary can generate $2^{m/2}$ variants.
- The set of messages are compared to find one message with a common hash value
- The valid message is offered for signature
- The fraudulent message with the same hash value is inserted in its place



Dear Anthony,

$\begin{Bmatrix}\text{This letter is}\\\text{I am writing}\end{Bmatrix}$ to introduce $\begin{Bmatrix}\text{you to}\\\text{to you}\end{Bmatrix}$ $\begin{Bmatrix}\text{Mr.}\\\text{--}\end{Bmatrix}$ Alfred $\begin{Bmatrix}\text{P.}\\\text{--}\end{Bmatrix}$

Barton, the $\begin{Bmatrix}\text{new}\\\text{newly appointed}\end{Bmatrix}$ $\begin{Bmatrix}\text{chief}\\\text{senior}\end{Bmatrix}$ jewellery buyer for $\begin{Bmatrix}\text{our}\\\text{the}\end{Bmatrix}$

Northern $\begin{Bmatrix}\text{European}\\\text{Europe}\end{Bmatrix}$ $\begin{Bmatrix}\text{area}\\\text{division}\end{Bmatrix}$. He $\begin{Bmatrix}\text{will take}\\\text{has taken}\end{Bmatrix}$ over $\begin{Bmatrix}\text{the}\\\text{--}\end{Bmatrix}$

responsibility for $\begin{Bmatrix}\text{all}\\\text{the whole of}\end{Bmatrix}$ our interests in $\begin{Bmatrix}\text{watches and jewellery}\\\text{jewellery and watches}\end{Bmatrix}$

in the $\begin{Bmatrix}\text{area}\\\text{region}\end{Bmatrix}$. Please $\begin{Bmatrix}\text{afford}\\\text{give}\end{Bmatrix}$ him $\begin{Bmatrix}\text{every}\\\text{all the}\end{Bmatrix}$ help he $\begin{Bmatrix}\text{may need}\\\text{needs}\end{Bmatrix}$

to $\begin{Bmatrix}\text{seek out}\\\text{find}\end{Bmatrix}$ the most $\begin{Bmatrix}\text{modern}\\\text{up to date}\end{Bmatrix}$ lines for the $\begin{Bmatrix}\text{top}\\\text{high}\end{Bmatrix}$ end of the

market. He is $\begin{Bmatrix}\text{empowered}\\\text{authorized}\end{Bmatrix}$ to receive on our behalf $\begin{Bmatrix}\text{samples}\\\text{specimens}\end{Bmatrix}$ of the

$\begin{Bmatrix}\text{latest}\\\text{newest}\end{Bmatrix}$ $\begin{Bmatrix}\text{watch and jewellery}\\\text{jewellery and watch}\end{Bmatrix}$ products, $\begin{Bmatrix}\text{up}\\\text{subject}\end{Bmatrix}$ to a $\begin{Bmatrix}\text{limit}\\\text{maximum}\end{Bmatrix}$

of ten thousand dollars. He will $\begin{Bmatrix}\text{carry}\\\text{hold}\end{Bmatrix}$ a signed copy of this $\begin{Bmatrix}\text{letter}\\\text{document}\end{Bmatrix}$

as proof of identity. An order with his signature, which is $\begin{Bmatrix}\text{appended}\\\text{attached}\end{Bmatrix}$

$\begin{Bmatrix}\text{authorizes}\\\text{allows}\end{Bmatrix}$ you to charge the cost to this company at the $\begin{Bmatrix}\text{above}\\\text{head office}\end{Bmatrix}$

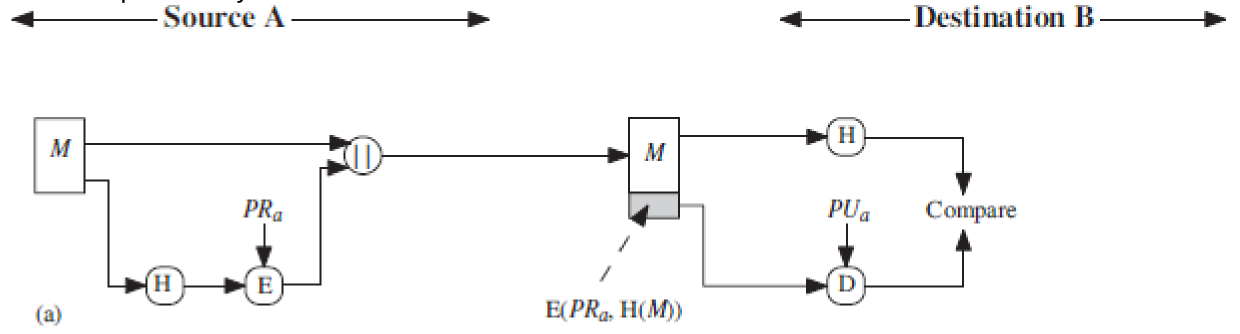
address. We $\begin{Bmatrix}\text{fully}\\\text{--}\end{Bmatrix}$ expect that our $\begin{Bmatrix}\text{level}\\\text{volume}\end{Bmatrix}$ of orders will increase in

the $\begin{Bmatrix}\text{following}\\\text{next}\end{Bmatrix}$ year and $\begin{Bmatrix}\text{trust}\\\text{hope}\end{Bmatrix}$ that the new appointment will $\begin{Bmatrix}\text{be}\\\text{prove}\end{Bmatrix}$

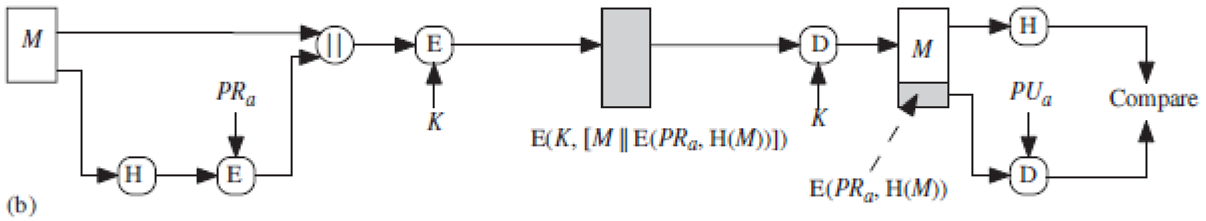
$\begin{Bmatrix}\text{advantageous}\\\text{an advantage}\end{Bmatrix}$ to both our companies.

Digital Signatures:

In the case of the digital signature, the hash value of a message is encrypted with a user's private key. Anyone who knows the user's public key can verify the integrity of the message that is associated with the digital signature. In this case, an attacker who wishes to alter the message would need to know the user's private key.



The hash code is encrypted, using public-key encryption with the sender's private key. This provides authentication. It also provides a digital signature, because only the sender could have produced the encrypted hash code. In fact, this is the essence of the digital signature technique.



If confidentiality as well as a digital signature is desired, then the message plus the private-key-encrypted hash code can be encrypted using a symmetric secret key. This is a common technique.

Other Applications:
To create one way password file.
Intrusion detection
Virus detection.
To construct Pseudo Random Function (PRF) or a pseudorandom number generator (PRNG)

Hash Algorithms:
MD5 (Message Digest):

- designed by Ronald Rivest, a Public key encryption algorithm.
- latest in a series of MD2, MD4
- produces a 128-bit hash value
- until recently was the most widely used hash algorithm.

Overview:

Takes as input a message of arbitrary length and produces as output a 128 bit message digest.

1. **Append padding bits:**
   the message is padded so that its length is 448 mod 512 .
   The padding consists of one 1 bit followed by necessary number of zeros.
2) **Append length:**
   the length of the original message is appended to the result of step 1.

If the original length is greater than 2^{64} then only the low order 64 bits of the length are used.
The expanded message is split into 512 bit blocks y_o, y_1.....

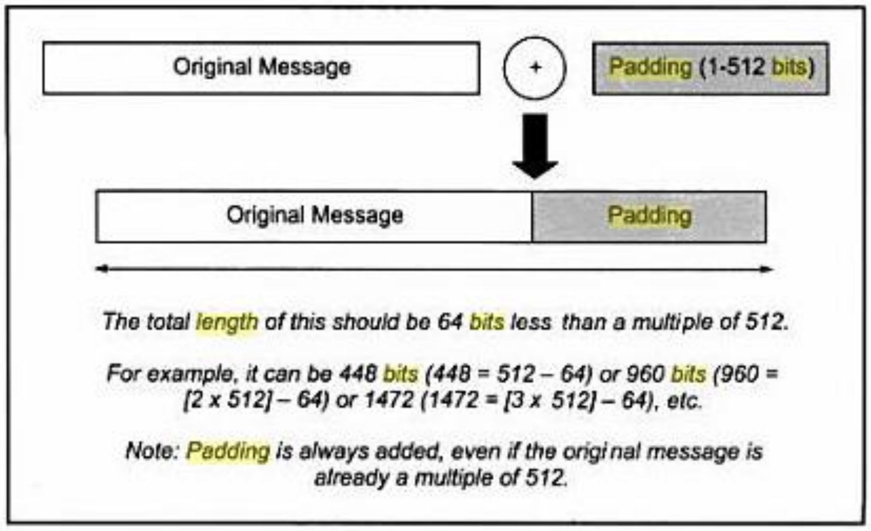

Original Message
+
Padding (1-512 bits)
Original Message
Padding
The total length of this should be 64 bits less than a multiple of 512.
For example, it can be 448 bits (448 = 512 – 64) or 960 bits (960 = [2 x 512] – 64) or 1472 (1472 = [3 x 512] – 64), etc.
Note: Padding is always added, even if the original message is already a multiple of 512.

Fig. 4.23 Padding process

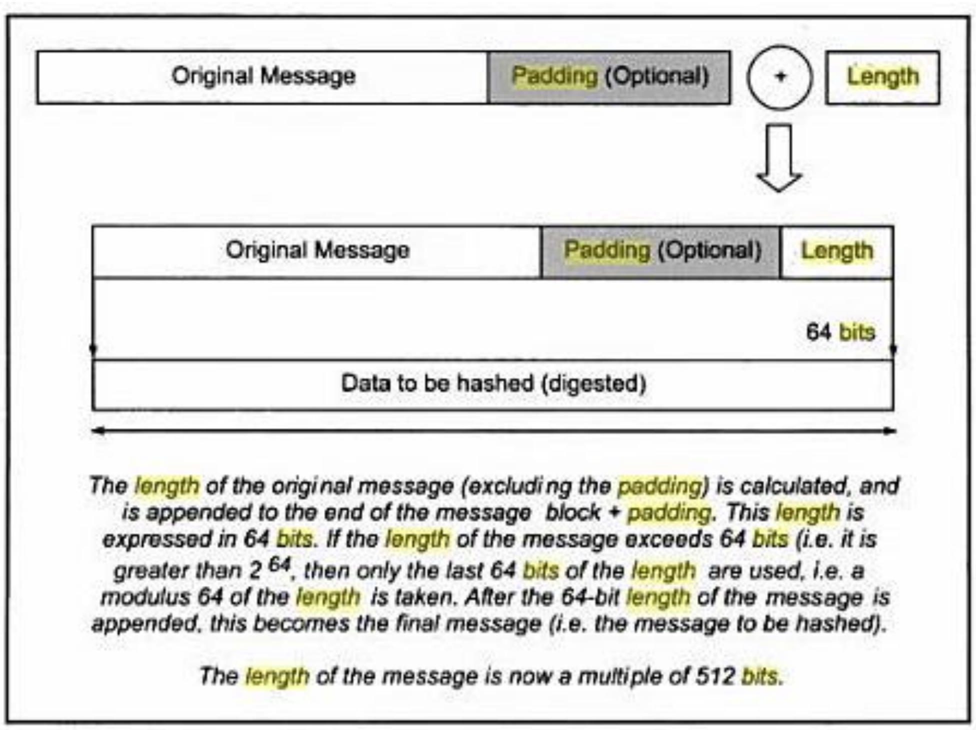

Original Message
Padding (Optional)
+
Length
Original Message
Padding (Optional)
Length
64 bits
Data to be hashed (digested)
The length of the original message (excluding the padding) is calculated, and is appended to the end of the message block + padding. This length is expressed in 64 bits. If the length of the message exceeds 64 bits (i.e. it is greater than 2 64, then only the last 64 bits of the length are used, i.e. a modulus 64 of the length is taken. After the 64-bit length of the message is appended, this becomes the final message (i.e. the message to be hashed).
The length of the message is now a multiple of 512 bits.

Fig. 4.24 *Append length*

3. **Intialize MD buffer:**
   - A 128 bit buffer is used to store the intermediate and finla results of the hash function.
   - The buffer is represented as four 32 bit registers a,b,c,d.
   - The values are stored in little endian format.

(eg) A= 67452301
Word A 01 23 45 67

4. **process message in 16-word (512-bit) blocks**:
   - The heart of the algorithm is a compression function that consists of four rounds of processing called HMD5.

   Four rounds have similar structure, each round uses a different primitive logical function called F,G,H,I.

5) **Output:**

   after all L 512 bit blocks have been processed , the output from the Lth stage is the 128 bit message digest.

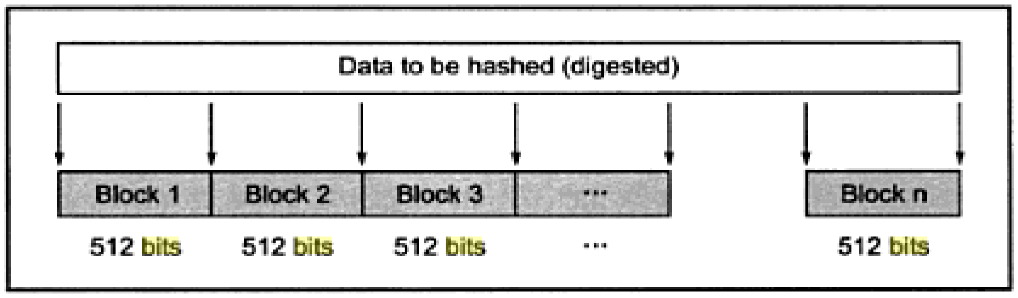
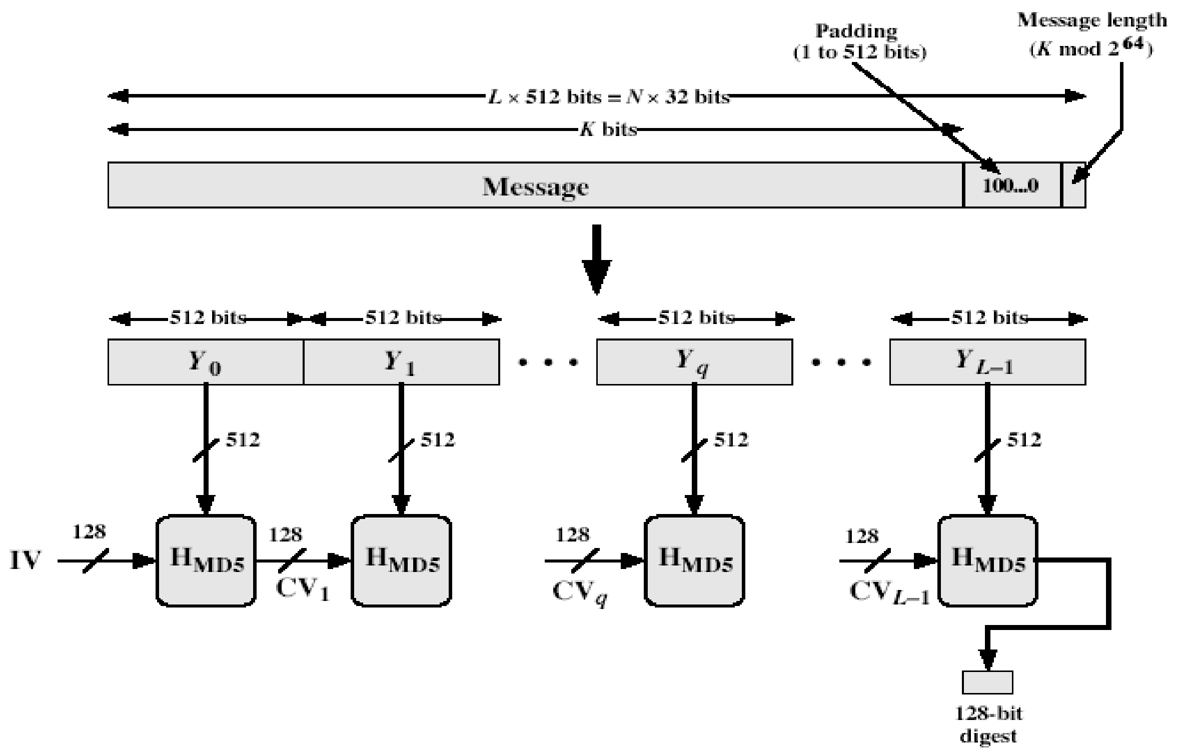


Fig. 4.25 *Data is divided into 512-bit blocks*



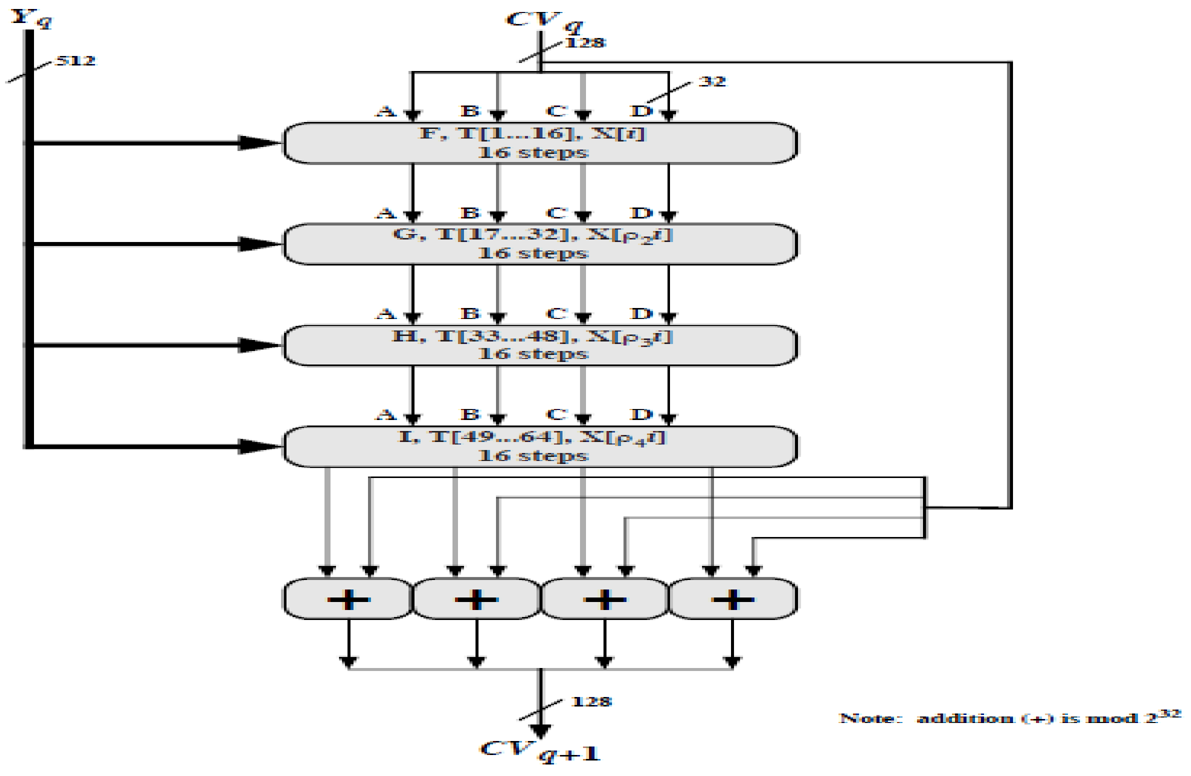
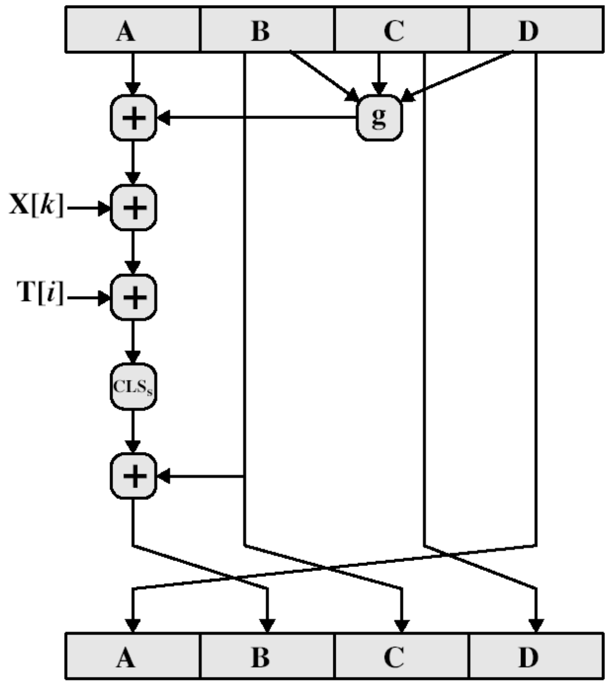

Y_q
512
CV_q
128
32
A B C D
F, T[1...16], X[i]
16 steps
G, T[17...32], X[$\rho_2 i$]
16 steps
H, T[33...48], X[$\rho_3 i$]
16 steps
I, T[49...64], X[$\rho_4 i$]
16 steps
+
128
CV_{q+1}
Note: addition (+) is mod 2^{32}

Figure 12.2 MD5 Processing of a Single 512-bit Block

Compression Function:

- each round has 16 steps of the form:
  a = b+((a+g(b,c,d)+X[k]+T[i])<<<s)
- a, b, c, d = refer to the 4 words of the buffer.
- g= one of the primitive functions F,G H , I.
- <<< s = left circular shift.
- T[i] = the ith 32 bit word in matrix T.
- + = addition modulo 2 ^{32}



Secure Hash Algorithm(SHA-1):

- SHA was designed by NIST1993, revised 1995 as SHA-1
- produces 160-bit hash values
- now the generally preferred hash algorithm
- based on design of MD4 with key differences

Overview:

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

(A=67452301, B=efcdab89, C=98badcfe,
D=10325476, E=c3d2e1f0)

1. process message in 16-word (512-bit) chunks:
   - use 4 rounds of operations on message block & buffer
   - add output to input to form new buffer value
2. output hash value is the final buffer value

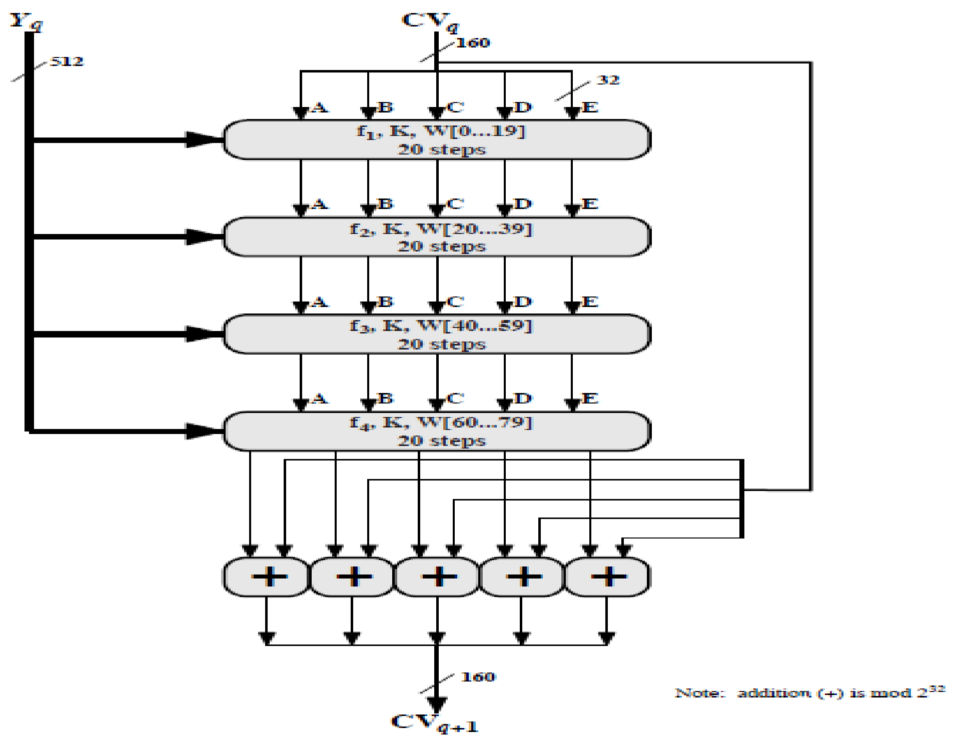
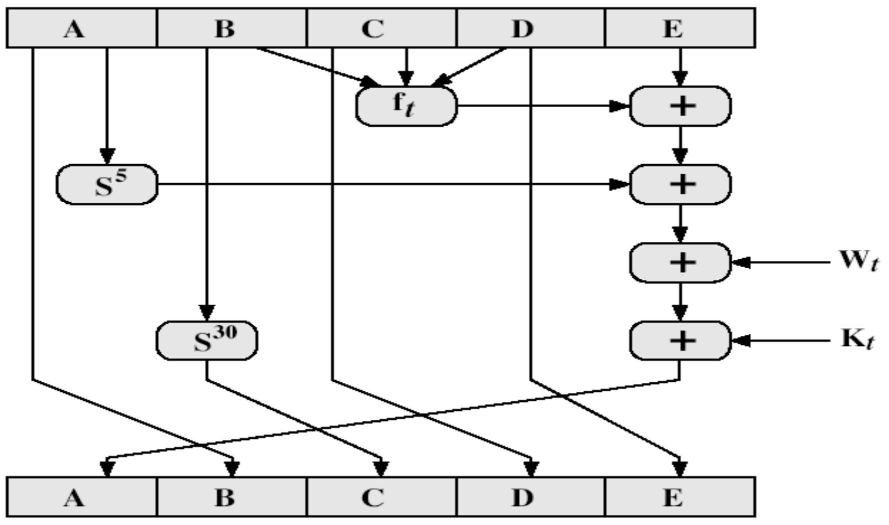


Figure 12.5 SHA-1 Processing of a Single 512-bit Block (SHA-1 Compression Function)

Compression Function:

- each round has 20 steps which replaces the 5 buffer words thus:

  $(A,B,C,D,E) <- (E+f_t(B,C,D)+(A<<5)+W_t+K_t),A,(B<<30),C,D)$
- a,b,c,d refer to the 4 words of the buffer
- t is the step number
- $f_t(B,C,D)$ is nonlinear function for round
- W_t is derived from the message block
- K_t is a constant value derived from sin



SHA-1 verses MD5:

	MD5	SHA
speed	Faster	Slower
endianess	Little Endian	Big Endian
Message digest	128	160
Software implementation	simple	
Attack to try and find the original message, given MD	Request 2^{128} operations to break	Request 2^{160} operations to break
register	4-32 bit register	5- 32 bit register
rounds	4 rounds of 16 steps	4 rounds of 20steps

DIGITAL SIGNATURES:

Message authentication protects two parties who exchange messages from any third party. However, it does not protect the two parties against each other. Several forms of dispute between the two are possible. Digital signature also protects the identity of each user.

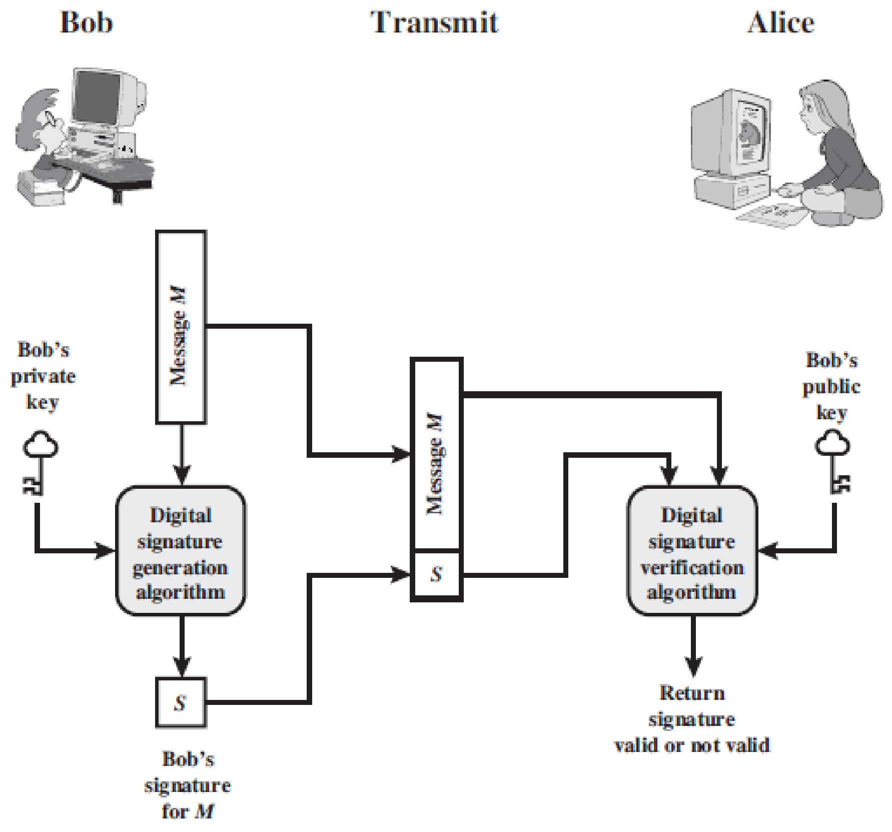

Bob
Transmit
Alice
Message M
Bob's private key
Digital signature generation algorithm
S
Bob's signature for M
Message M
S
Bob's public key
Digital signature verification algorithm
Return signature valid or not valid

Figure 13.1 Generic Model of Digital Signature Process

Authentication

$$A \xrightarrow{auth} B$$

protects against{C}

Signature

$$A \xrightarrow{sign} B$$

protects against{A,C}

A Digital signature is an authentication mechanism that enables the creator of a message to attach a code that acts as a signature. It ensures that the original content of the message that has been sent is unchanged.

Properties of digital signatures:

- verify author, date & time of signature
- authenticate message contents
- be verified by third parties to resolve disputes

Requirements of digital signature:

- The signature must be a bit pattern.
- It must use some information unique to the sender to prevent forgery and denial.
- It must be relatively easy to produce, recognize, and verify the digital signature.
- It must be practical to retain a copy of the digital signature in storage.

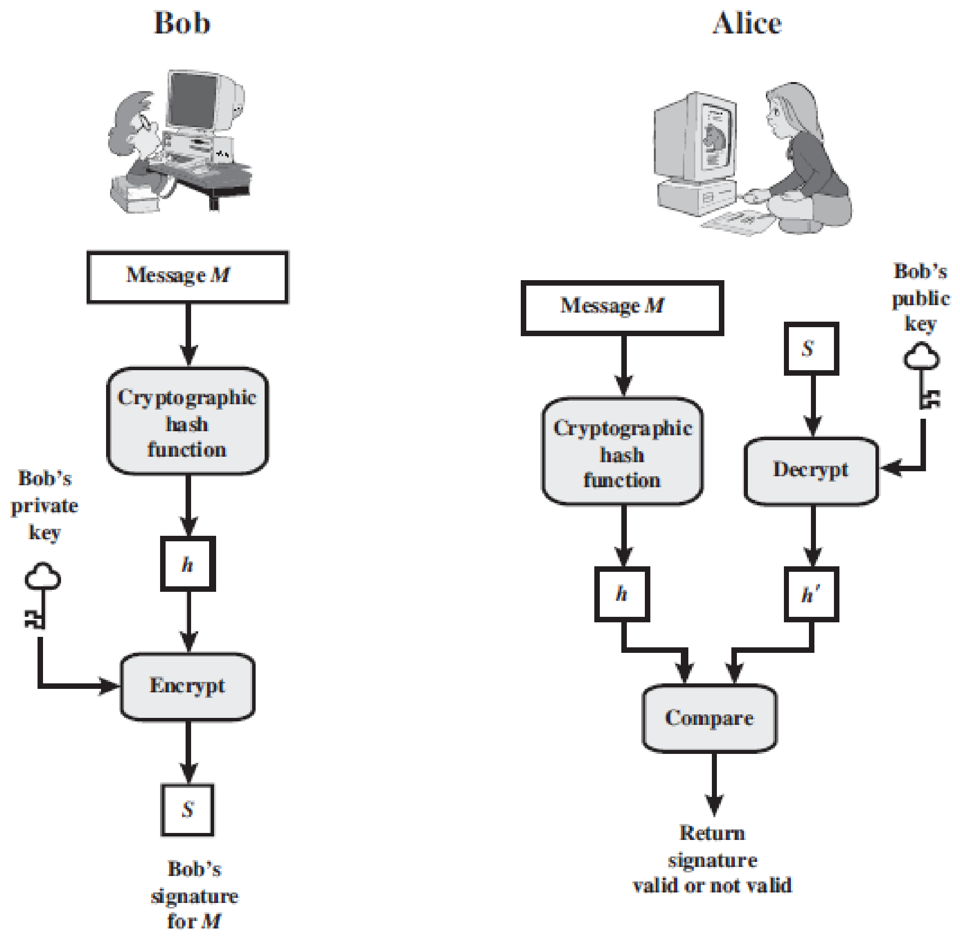


Figure 13.2 Simplified Depiction of Essential Elements of Digital Signature Process

SIGNATURE TYPES:

- Direct

  $X \rightarrow Y$

  weakness: security of private key
- Arbitrated

  $X \rightarrow \overset{\text{+ date}}{A} \rightarrow Y$

DIRECT DIGITAL SIGNATURES:

- involve only the sender & receiver
- assumed receiver has sender's public-key
- Digital signature may be performed by encrypting the entire message with the sender's private key or by encrypting the hash code of the message with the sender's private key.
- Confidentiality can de provided by further encrypting the entire message plus signature with receivers public-key or shared key.
- It is important to perform signature function first then encrypt message & signature
- security depends on sender's private-key

Drawback:

- If the sender later wishes to deny sending a particular message, the sender can claim that the private key was lost or stolen and that someone else forge his or her signature.
- To overcome the above problem it is required that every message to include a timestamp (date and time) and to require prompt reporting of compromised keys to central authority.

ARBITRATED DIGITAL SIGNATURES:

- The problem with direct digital signatures can be addressed by using an arbiter.
- Every signed message from a sender X to receiver Y first goes to an arbiter A, who subjects the message and its signature to a number of test to verify its origin and content.
- The message is then dated and sent to Y with an indication that it has been verified to the satisfaction of the arbiter.

Three different arbitrated digital signature techniques are:

1. Conventional encryption, arbiter sees message.
2. Conventional encryption, arbiter does not see message
3. Public key encryption, arbiter does not see message

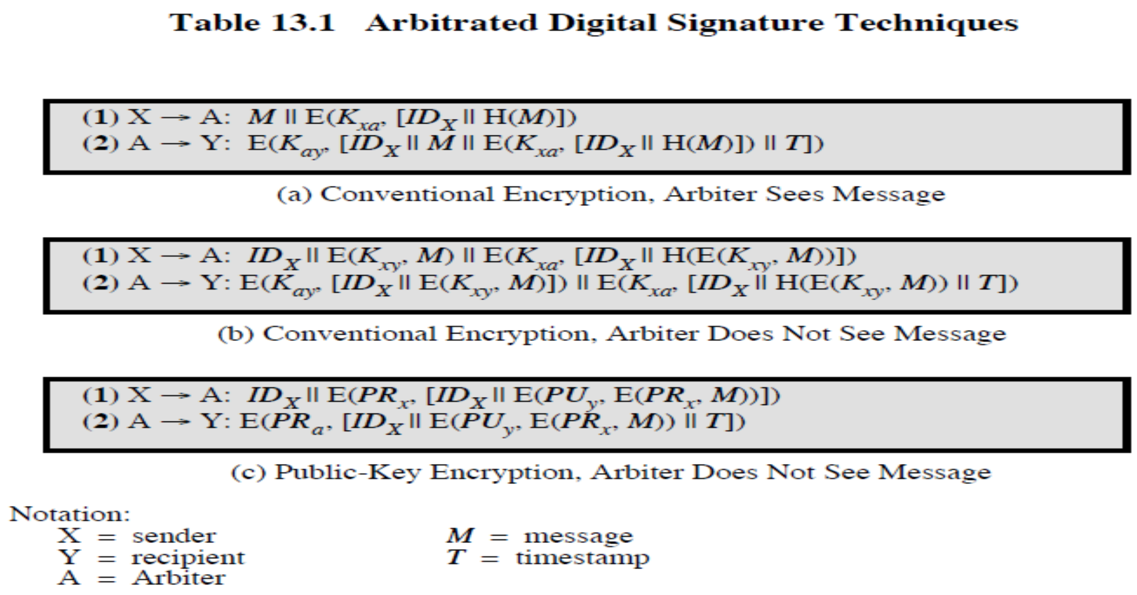


Table 13.1 Arbitrated Digital Signature Techniques

(1) $X \rightarrow A$: $M \| E(K_{xa}, [ID_X \| H(M)])$
(2) $A \rightarrow Y$: $E(K_{ay}, [ID_X \| M \| E(K_{xa}, [ID_X \| H(M)]) \| T])$

(a) Conventional Encryption, Arbiter Sees Message

(1) $X \rightarrow A$: $ID_X \| E(K_{xy}, M) \| E(K_{xa}, [ID_X \| H(E(K_{xy}, M))])$
(2) $A \rightarrow Y$: $E(K_{ay}, [ID_X \| E(K_{xy}, M)]) \| E(K_{xa}, [ID_X \| H(E(K_{xy}, M)) \| T])$

(b) Conventional Encryption, Arbiter Does Not See Message

(1) $X \rightarrow A$: $ID_X \| E(PR_x, [ID_X \| E(PU_y, E(PR_x, M))])$
(2) $A \rightarrow Y$: $E(PR_a, [ID_X \| E(PU_y, E(PR_x, M)) \| T])$

(c) Public-Key Encryption, Arbiter Does Not See Message

Notation:

X = sender	M = message
Y = recipient	T = timestamp
A = Arbiter	

Conventional encryption, arbiter sees message.

- Symmetric encryption is used.
- Both sender and receiver should trust an arbiter.

Drawback:

A can able to read the message from X to Y like an eavesdropper.

Conventional encryption, arbiter does not see message.

Drawback:

- Unable to read the message but the arbiter could form an alliance with the sender to deny a signed message or with the receiver to forge the sender's signature.

Public key encryption, arbiter does not see message:

Adv:

- Encrypting the message twice with public key algorithm increase the confidentiality.