U-III PUBLIC KEY CRYPTOGRAPHY

Prime Numbers:

- ✓ Prime number is a positive integer greater than 1 whose only factors are 1 and number itself
- ✓ Prime number cannot be divided by any number other than 1 and itself

E.g. 2, 3, 5,7,11 ……. Are prime numbers

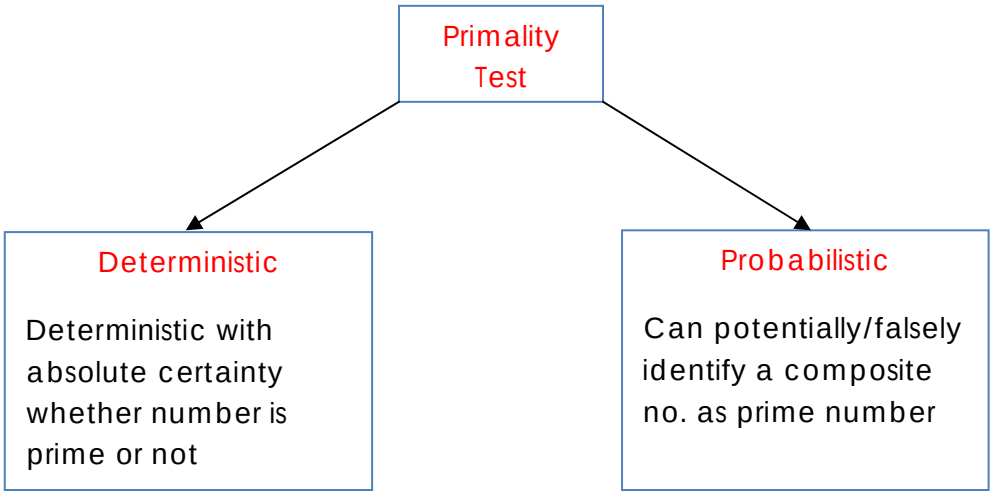
4, 6, 8, 10, 12 …..Are not prime

Relative prime numbers:

- ✓ Two numbers are relatively prime when they have no factors common other than 1

E.g. number 21 and 44 are relatively prime (because they have no factors in common)

Number 21 and 45 are not relatively prime as they have common factor 3



- ✓ As shown in above fig. Primality test is of two types i.e. Deterministic and Probabilistic
- ✓ Probabilistic test is fast as compared with deterministic one

Testing for Primality of number:

- ✓ Cryptographic algorithms select one or more very large prime number at random
- ✓ So it's important to check the Primality of selected random number

Fermatt's Little Theorem:

If P is prime and a is positive integer not divisible by P then $a^{p-1} \equiv 1 \bmod p$

(Symbol $\equiv$ stand for Identical)

Miller Rabin Test:

If either the first element in sequence is 1 or some other element is n-1 then n could be probably prime otherwise n is certainly not prime

Test (n)

1. $n-1 = 2^k q$ compute K and Q
2. select a random integer a
3. $1<a<n-1$
4. If $a^q \bmod n = 1$ then

   Return "No. is probably prime"
5. For j=0 to k-1 do
6. If $a^{2jq} \bmod n = n-1$ then

   Return "No. is probably prime"

   Return "Not Prime"

Composite Number:

- ✓ A positive integer which has positive divisor other than one and itself is called composite number
- ✓ By definition every integer greater than 1 is either a prime number or composite number
- ✓ No. 0 and 1 are considered to be neither prime nor composite

Properties:

- ✓ All even numbers greater than 2 are composite numbers
- ✓ The Smallest composite number is 4

Factoring large No.:

- ✓ Process of finding the factors of a number is called factoring
- ✓ Integer factorization is(breaking large no) or breaking down composite number into smaller nontrivial divisor which when multiplied together equals original number
- ✓ For very large no. there is no efficient integer factorization algorithm
- ✓ A RECENT EFFORT WHICH FACTORED 200 DIGIT NO.(RSA-200) TAKEN 18 MONTHS

Overview of Asymmetric key cryptography:

- ✓ Asymmetric key cryptography is also called as public key cryptography
- ✓ Two different keys which form key pair are used for encryption and decryption
- ✓ One of the two keys is called public key and other is called private key
- ✓ Public key is used for encryption and private key is used for decryption
- ✓ Private key will remain with user while public key is shared among the group of users
- ✓ In this scheme each person or node publishes its public key and by using this detail directory can be constructed where details of various nodes and keys are maintained

- ✓ If someone interested to communicate with concerned node he get details of public key through the directory

Key Details	A Should Know	B Should Know
A's Private Key	YES	NO
A's Public Key	NO	YES
B's Private Key	NO	YES
B's Public Key	YES	NO

Table: Directory of public and private key

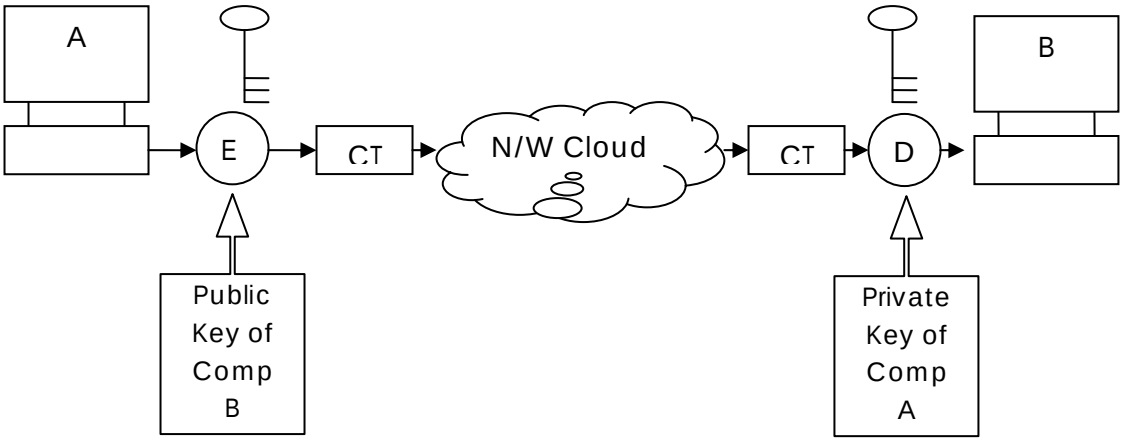


Fig: Asymmetric Key Cryptography

Working:

- ✓ Let Kua is public key and Kpa is private key of Compute A
- ✓ Similarly Kub & Kpb are public & private key of computer B
- ✓ Computer A wants to transfer message to computer B so A encrypts message by using B's public key and this is only possible when A knows B's public key
- ✓ A produces CT by using public key of computer B i.e. CT=Ekub(PT) and diverts it to B
- ✓ B decrypts message by using by using its own private key i.e.(Kpb)
- ✓ Similarly when B wants to send message to A it encrypts message by public key of computer A i.e.(Kua)

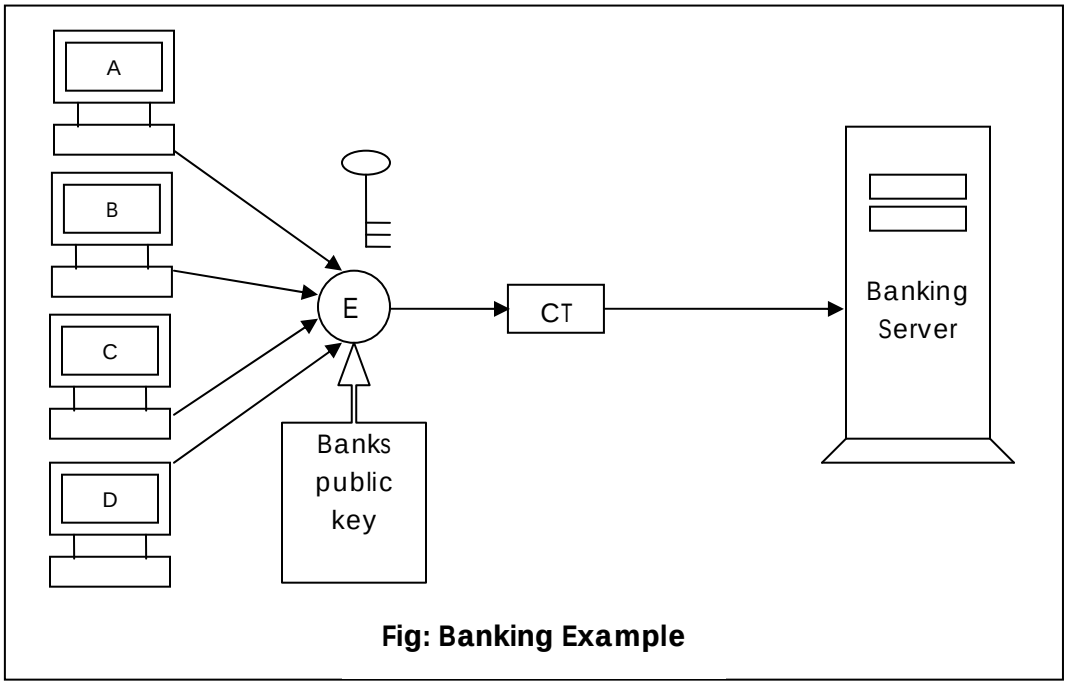


Fig: Banking Example

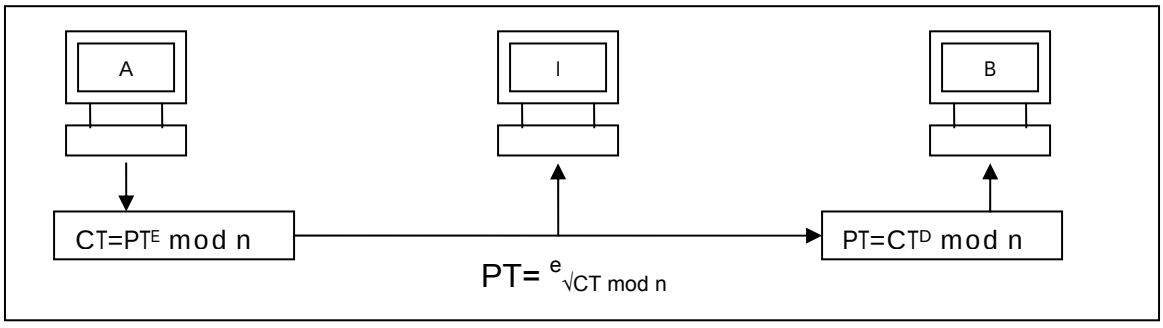
As shown in above fig bank publishes its public key to all its customer's .Customer can use banks public key for encrypting the message while they can decipher data received from bank by using their own private key

RSA Algorithm:

- ✓ Developed by **R**ivest, **S**hamir, **A**dleman
- ✓ Most popular asymmetric key cryptographic algorithm
- ✓ RSA algorithm is based on mathematical fact that it is easy to find and multiply to large prime numbers together but it is extremely difficult to factor their products
- ✓ The public and private key in RSA are based on very large prime number(made of 100 or more digit prime number)
- ✓ Algorithm is quiet simple however challenge was related with selection and generation of public and private key

Algorithm:

1. Select two large prime numbers let **P** and **Q**
2. Calculate **N= P * Q**
3. Select the public key (i.e. Encryption key) E such that it is not the factor of **(P – 1)** and **(Q – 1)**
4. Select private key (i.e. Decryption key) D such that the following equation becomes true **(D * E) mod (P – 1) * (Q – 1) = 1**
5. For encryption calculate Cipher text CT from plain text PT as follows **$CT=PT^E$ mod N**
6. Send **CT** i.e. cipher text to receiver
7. For decryption calculate PT from CT i.e. **$PT= CT^D$ Mod N**



As shown for user A & B there is polynomial complexity while for intruder I there is logarithmic complexity i.e. hard to break cipher by Intruder

Example:

1. Take P=7 and Q=17 as two prime numbers
2. N=P * Q =7*17 =119
3. (P-1)* (Q-1)=6*16=96 so factors are 2,2,2,2,2 and 3 so public key should not have factor of 2 and 3 let us choose public key value as 5
4. Select private key D such that (D *E) mod (P-1)*(Q-1)=1 so choose 77 as D because it satisfies the equation
5. i.e. (5*77) mod 96 => 385 mod 96=1 which satisfy our condition
6. E=5 and D=77

Diffie and Hellman Key exchange Algorithm:

- ✓ Whitefield Diffie and Martin Hellman devised amazing solution to the problem of key arrangement or key exchange in 1976
- ✓ Two personnel who wants to communicate securely can agree on a symmetric key using this technique
- ✓ This key can be used for encryption or decryption
- ✓ Diffie and Hellman algo can be used for key arrangement and for encryption and decryption

Description of Algorithm:

1. A and B agree on two large prime number n and g these two integers are not kept secret A and B use an insecure communication channel to agree on them
2. A (Alice) choose another large random number **X** and calculate A such that $\mathbf{A=g^x \bmod n}$
3. Alice (A) send the no A to Bob (B)
4. Meanwhile B independently selects another large random number **Y** and calculates B such that $\mathbf{B=g^y \bmod n}$
5. Bob sends B to A
6. Now Alice compute key $\mathbf{K1=B^x \bmod N}$
7. Now Bob compute key $\mathbf{K2=A^y \bmod N}$

It's surprising that K1 and K2 are equal this means K1=K2=K is the symmetric key

Note: Diffie and Hellman key exchange algorithm gets its security from the difficulty of calculating discrete logarithm in a finite field as compared with ease of computing exponentiation from the same field

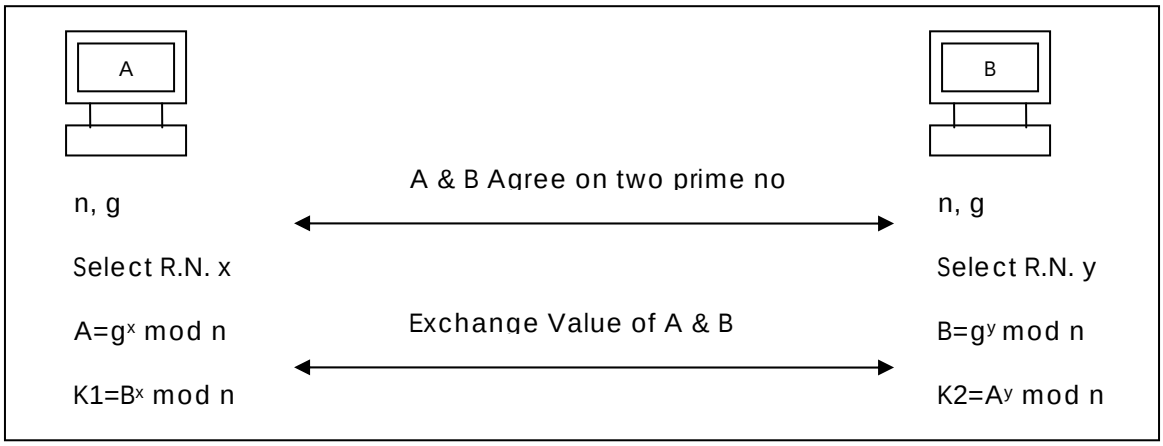


Fig: Shows the working process of Diffie & Hellman Key Exchange

Example:

1. Firstly Alice and Bob agree on two large prime no. n and g, These integers need not be kept secret they can use unsecured communication channel to agree on it let n=11 and g=7
2. Alice select another large random no x and calculates A such that

   $A=g^x \bmod n$

   Let X= 3 then we have $A=7^3 \bmod 11$

   A= 343 mod 11 i.e. A=2
3. Alice sends value of A i.e. A=2 to Bob
4. Bob independently selects another large random number y and calculates B such that

   $B=g^y \bmod n$

   Let y=6 then we have $b=7^6 \bmod 11$

   b=117649 mod 11 .i.e. b=4

5. Bob sends value of B i.e. B=4 to Alice
6. Now Alice computes the secret key k1 at his end

   $K1=b^x \bmod n$

   Where $K1= 4^3 \bmod 11=9$
7. Bob calculates key k2 at his end

   $K2=A^y \bmod n$

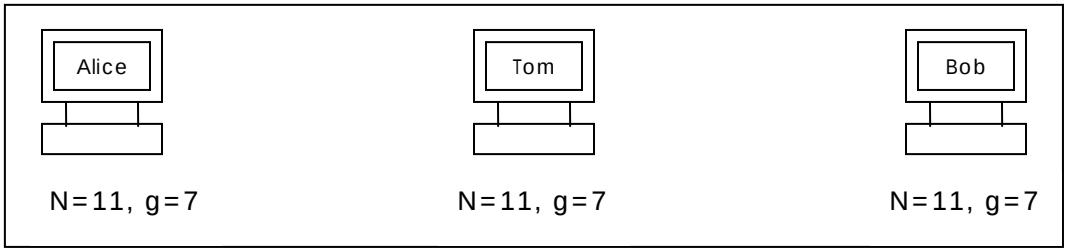
   Where $K2= 2^6 \bmod 11=9$

So as shown secret keys computed at both the ends are same

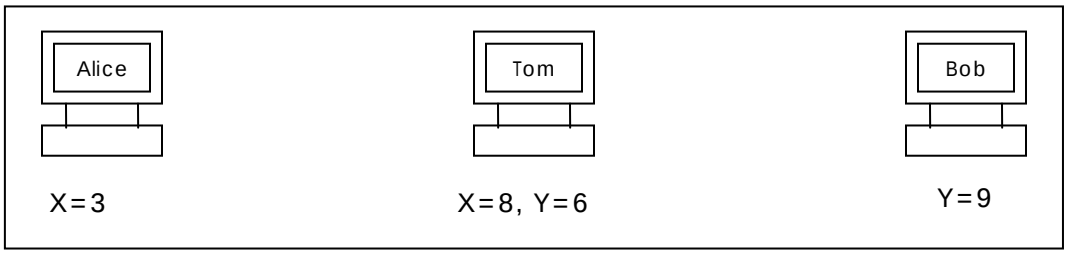
Problem with Diffie and Hellman key exchange Algorithm:

It can fall pray to the man in middle attack (or to be politically correct, woman in middle attack), also called bucket bridge attack

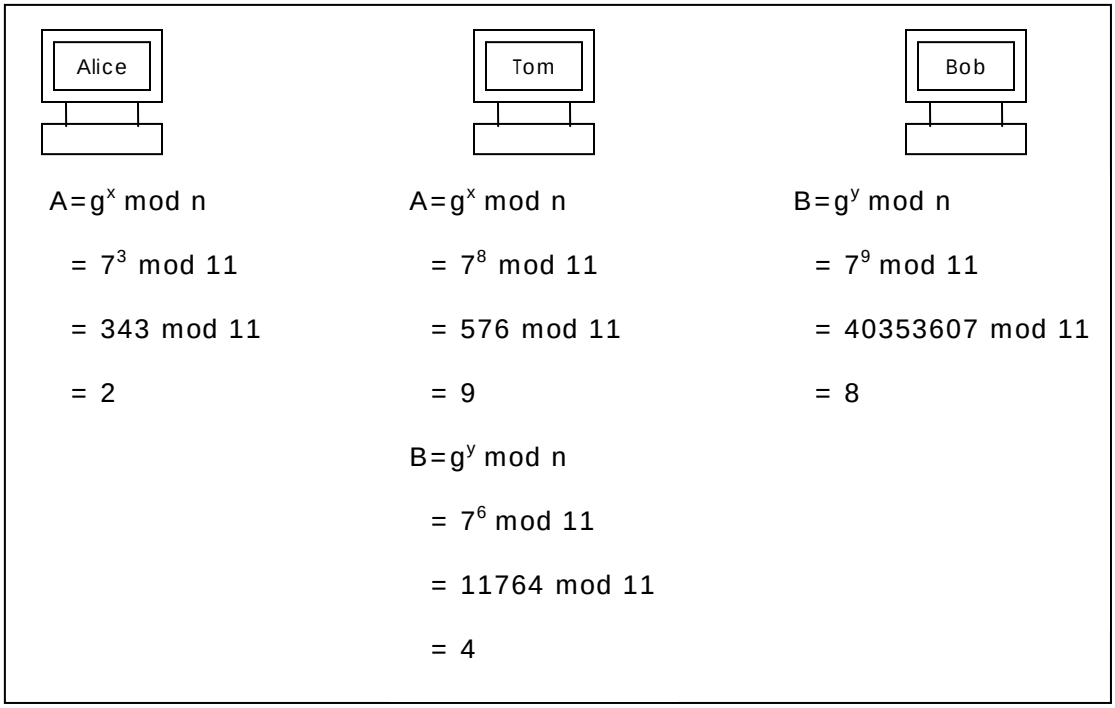
1. Alice and Bob wants to communicate so first Alice sends value of **g** and **n** to Bob as usual let n=11 and g=7
2. Alice does not realize that attacker Tom is listening the channel. Tom simply picks the value of n and g and forwards it to Bob



3. Now lets assume Alice, bob as well as tom selects random numbers x and y independently (where Tom select both X & Y)



4. Now based on selected Random values Alice and Bob calculates A & B However Tom calculates both A & B



5. Now the real drama begins Alice sends A=2 to bob Tom interpret and instead of sending A=2 send A=9 to Bob

6. In return Bob sends B i.e. B=8 to Alice Tom interpret it again and sends his B i.e. B=4 to Alice

Now Secret key computed by Alice and bob are based on the value of A & B given by Tom i.e. Tom too knows key used by Alice and Bob Tom is able to Decipher CT used by Alice & Bob for communication

Elgamal Algorithm:

- Elgamal is symmetric key encryption algorithm for public key cryptography which is based on Diffie and Hellman key exchange algorithm
- Developed by Taher Elgamal in 1984
- Algorithm is used in free GNU privacy guard software
- Security of Elgamal is based on the difficulty of computing discrete logarithm in finite field

Working:

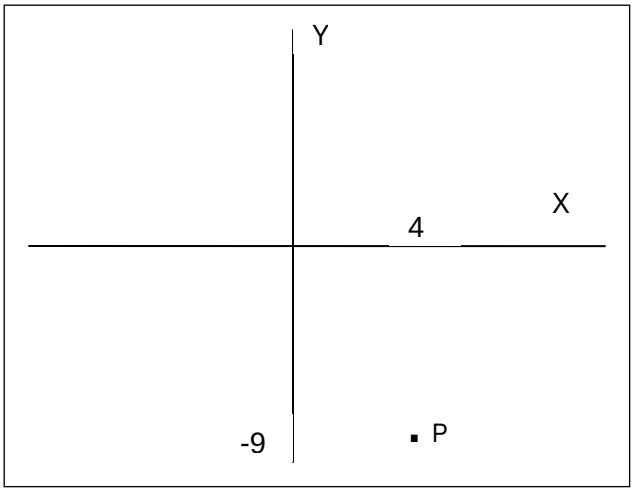
1. To generate key pair first select prime number **P** and two random number **g** and **x** so that both **g** and **x** are less than **P**
2. Find the value of $y=g^x \bmod p$
3. Public key becomes y, g, p both g and p can be shared among the group of users
4. The private key is X
5. For encrypting plaintext message M first select a random number k which is relatively prime to p-1
6. Now find a and b where $a=g^k \bmod p$ and $b=y^k M \bmod p$ here $M=(ax + kb) \bmod (p-1)$
7. Pair (a, b) becomes the cipher text
8. To decrypt (a, b) find out plaintext M calculate $M=b/a^x \bmod p$

Elliptical Curve cryptography (ECC):

- ✓ RSA is the most popular encryption algorithm used in public key cryptography
- ✓ Over the years key length of RSA is increasing which keeps considerable burden on RSA
- ✓ Another public key cryptography algorithm is gaining popularity is known as ECC
- ✓ Main difference between RSA and ECC is ECC offers same level of security by smaller key sizes
- ✓ ECC is highly mathematical in nature

Elliptical Curve:

- ✓ Elliptical curve is similar to normal curve drawn on X and Y axis
- ✓ It has some point and each point can be designated by and (X,y) coordinates just like any graph
- ✓ A point can be designated as P(4,-9) as shown in graph which means that it is 4 unit on x axis and 9 below on y axis



- ✓ Consider elliptical curve **E** with a point P no generate a random number **d**
- ✓ Let we have Q=d * P
- ✓ As per the mathematics of ECC E, P and Q are public values and the challenge is to find d
- ✓ This challenge is called as elliptical curve discrete logarithmic problem
- ✓ As long as curve is big enough it is almost impossible to find d
- ✓ Thus E, P, Q together form a public key
- ✓ D is correspondingly private key

Strengths of ECC:

- ✓ Small key size (RSA-1024 bit equivalent to ECC-160bit)
- ✓ Less computational overheads than RSA since it does not analyze prime numbers
- ✓ Requires less storage , less power, less memory and less bandwidth
- ✓ Can be used in wireless devices, handhelds and smart cards

Weaknesses of ECC:

- ✓ Not extensively researched as RSA
- ✓ New details are still being resolved
- ✓ Many ECC techniques are still new to be trusted
- ✓ Not widely used or supported

Key Management:

One of the major role of public key encryption algorithm is to address the problem of key distribution, there are two distinct aspects

1. Distribution of public key
2. Use of public key encryption to distribute secret key

Distribution of public key:

Several techniques are proposed for distribution of public key and all these proposals are grouped to form following methods

1. Public announcement
2. Public available directory
3. Public key authority
4. Public key certificates

Public key announcement:

Public key is public so if there is some broadly accepted algorithm such as RSA any participant can send his or her public key to any other participant or broadcast the key to the community at large

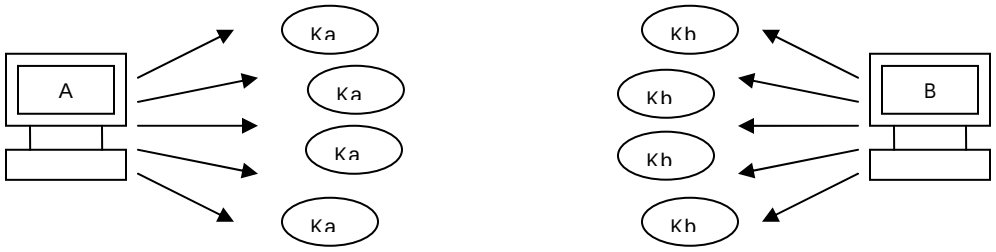


Fig. Uncontrolled Public key Distribution

Ex: because of growing popularity of PGP (pretty good privacy) that makes uses of RSA many PGP users have adopted practice to attach public key to the message that they send to public forum such as USENET newsgroup and internet mailing list

Above approach is convenient but it has major problem/weakness that anyone can forge such public announcement

Public available directory:

- ✓ Greater security can be maintained by using publically available dynamic directories
- ✓ Maintenance and distribution of publically available directory is done by some trusted entity or organization

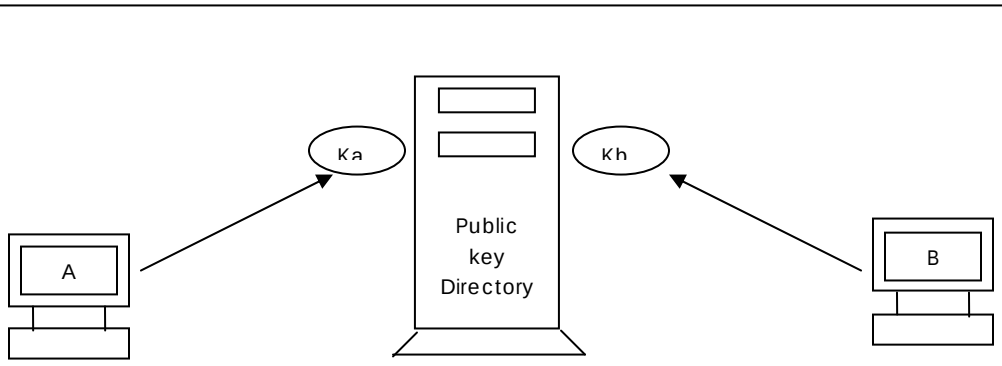


Fig. Public key publication

- ✓ Authority maintains a directory with name and public key for each participants
- ✓ Each participant register PK with directory authority
- ✓ Registration to be done in person or by some form of authenticated communication
- ✓ Participant may replace the existing key with new one at any time
- ✓ Periodically the authority publishes the entire directory or updates made in the directory
- ✓ Participants can access directory electronically

Public Key Authority:

Stronger security for public key distribution can be achieved by providing tighter control over distribution of public key

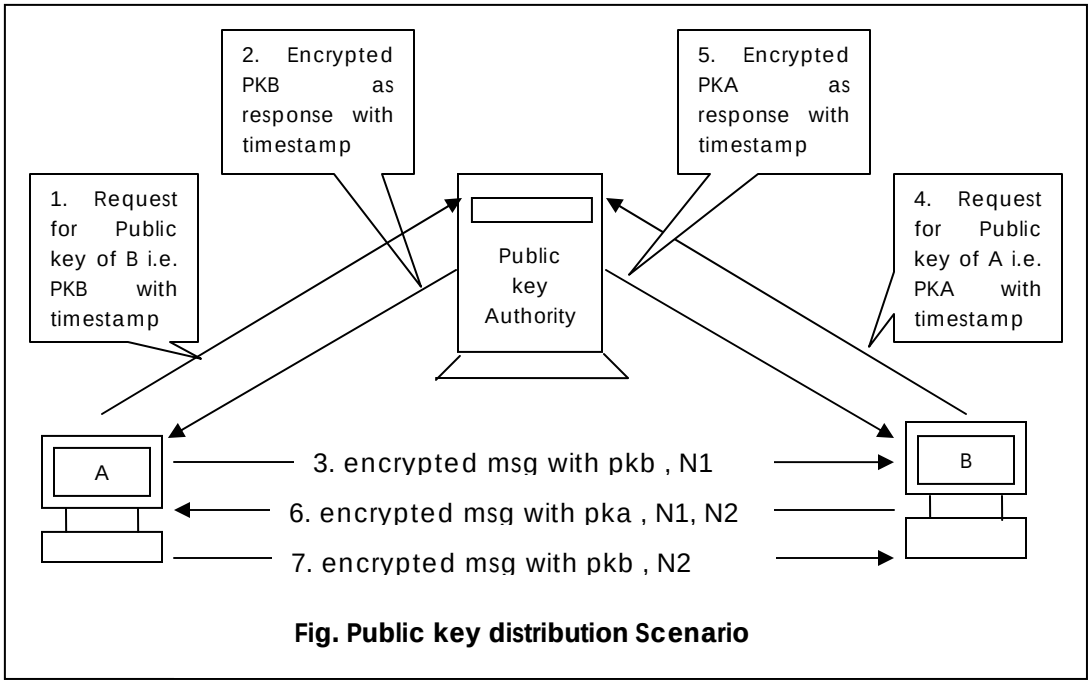


Fig. Public key distribution Scenario

1. A sends time stamped message to Public key authority containing request for the public key of computer B i.e. (PKB)

2. Public key authority responds with encrypted public key of computer B i.e. PKB with timestamp generated from A so that it recognize message is genuine
3. A uses public key of B and nonce (N1) which is used to identify transaction uniquely
4. B sends times tamed message to public key authority containing request for the public key of computer A i.e. (PKA)
5. Public key authority responds with encrypted public key of computer A i.e. PKA with timestamp generated from B so that it recognize message is genuine
6. B send encrypted message to A which contains A's nonce (N1) as well new nonce (N2) generated by B so that A could assure that the correspondent is B
7. A returns message with N2 encrypted by using B's public key to assure B that correspondent is A

Note: Step No. 6 and 7 are desirable steps and acts as transaction & user identifiers

Drawbacks:

- ✓ Public key authority could be somewhat bottleneck in the system since user must have to appeal to authority for public key for every other user it wishes to contact
- ✓ It may be possible that directory of name and public keys maintained by the authority is vulnerable

Public key Certificates:

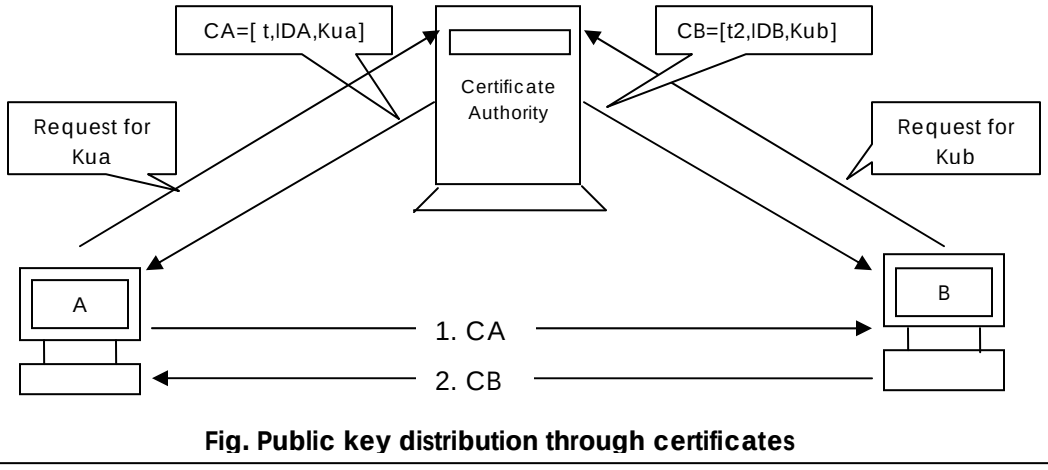


Fig. Public key distribution through certificates

- ✓ This approach is suggested by Kohnfelder[KOHN78] is to use certificate that can be used by participants to exchange the keys without contacting a public key authority
- ✓ Overcomes drawbacks of public key authority
- ✓ Participants convey their key information to others by transmitting certificates
- ✓ Digital certificate would actually be a computer file such as a.cer
- ✓ Certificates are similar to our passport but they are in electronic form
- ✓ Any participant can read the certificate to determine name and public key of certificate owner
- ✓ Any participant can verify that certificate is originated from certificate authority
- ✓ Only certificate authority can create and update certificates
- ✓ Any participant can verify the currency of certificate
- ✓ Recipient uses certificate authority's key to decrypt the certificate because it's only readable by he authority's public key. This verifies that certificate came from the certificate authority

Use of public key encryption to establish session:

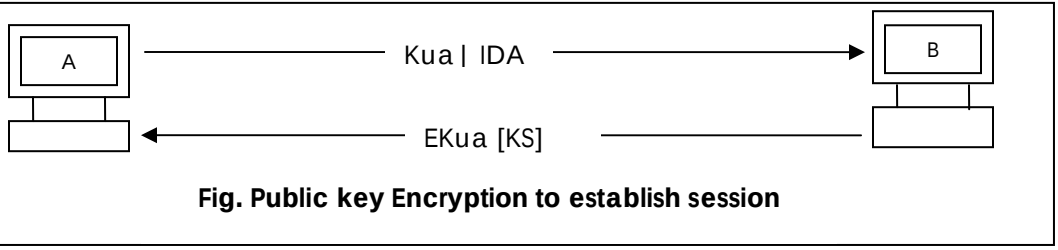


Fig. Public key Encryption to establish session

- ✓ **A** generates public and private key pair and transmit message with kua i.e. public key of **A**
- ✓ **B** generates secret key **KS** and transmit it to A, encrypted with A's public key
- ✓ **A** decrypts message to recover secret key **KS** as only **A** can decipher message by using his own private key
- ✓ **A** discards Kua and and they start communication with session key KS (Secret key)

Secret key distribution with confidentiality and authentication:

Following figure is based on the approach (NEED 78), provide protection against active and passive attacks

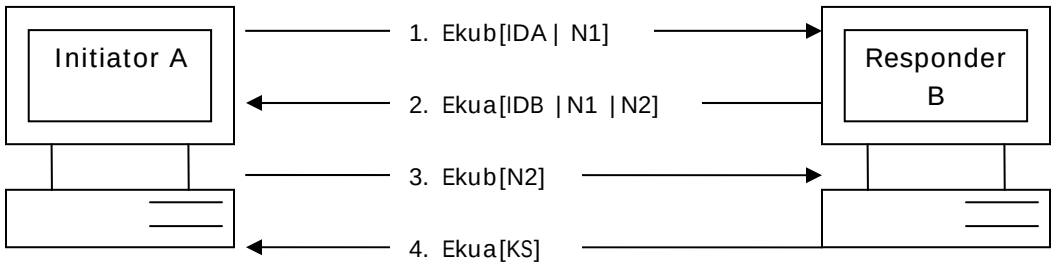


Fig. Secret key distribution with confidentiality

- ✓ Lets consider A & B has exchanged their public keys with each other
- ✓ A uses B's public key to encrypt message containing identifier of computer A (IDA)and nonce N1 used to identify transaction uniquely

- ✓ B encrypts message and sends N1 as well N2 nonce generated at B
- ✓ A returns N2 encrypted by using B's public key to assure B that correspondent is A
- ✓ B select secret key KS and send encrypted Ks to A

Public Key Cryptography Standards:

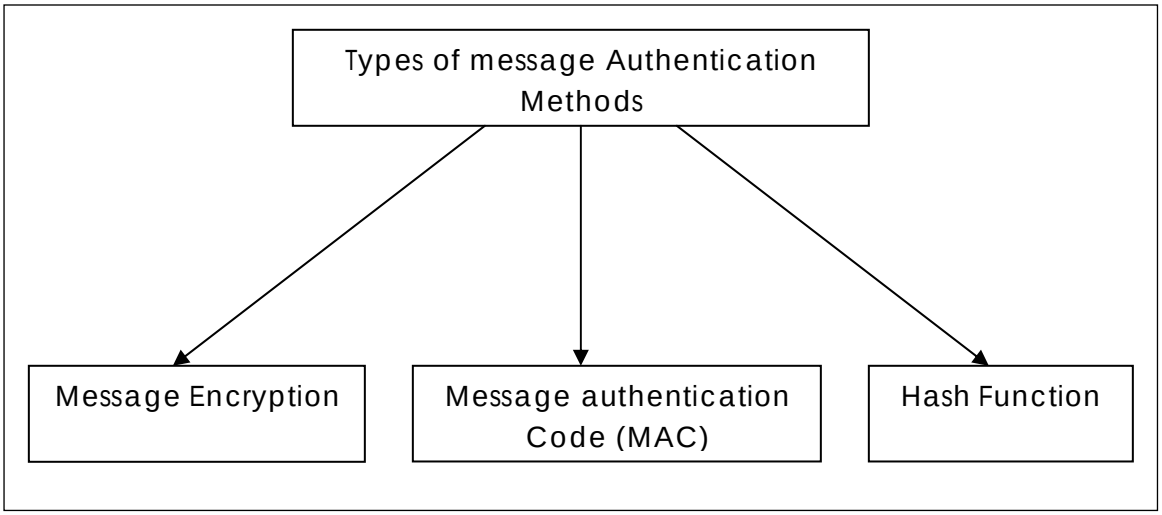
PKCS model was originally developed by RSA lab with the help of representatives from Government and Academia. Main purpose of PKCS is to standardize public key infrastructure (PKI)

Standard	Purpose	Description
PKCS#1	RSA Encryption Standard	Describe basic formatting rules for RSA public key function more specifically digital signature
PKCS#2	RSA Encryption Standard for message digest	This std. outlines message digest calculation, However this is now merged with PKCS#1
PKCS#3	Diffie and Hellman key arrangement	Define mechanism to implement Diffie and Hellman key arrangement protocol
PKCS#4	NA	Merged with PKCS#1
PKCS#5	Password based	Describe method for encrypting the octet with symmetric key , where symmetric key is derived from password
⋮	⋮	⋮
PKCS#13	ECC	Currently under development
PKCS#14	PRNG Standard	Generates pseudo random numbers
PKCS#15	Cryptographic token information syntax standard	

Message Authentication:

Goal:

- ✓ We have received some message now someone like to check whether message is altered in way or not
- ✓ Produce short sequence of bit(identifier) that depends on message at sender end
- ✓ Send message and short sequence of bit(identifier) toward destination user
- ✓ Receiver authenticate message by computing sequence of small bit pattern of received message and compares computed and received identifier.
- ✓ If computed and received bit patterns(identifier) matches then receiver concludes that there is no alteration in message during transaction



1. **Message Encryption:** Cipher text acts as authenticator
   - ✓ Main idea is receiver get assured that message came from A because CT can be decrypted by only his private key
   - ✓ Also none of the bit in message is altered because opponent doesn't know how to manipulate bits of CT to induce meaningful changes to PT
   - ✓ Conclusion: Encryption provides authentication as well as confidentiality

2. **Message Authentication code (MAC):** Public function of message and secret key produces a fixed length value to serve as authenticator

3. **Hash function:** Way of creating a small digital fingerprint from any kind of data

Message Digest:

A message digest is fingerprint or summary of message. It is similar to the concept of CRC (cyclic redundancy check) or LRC (Longitudinal redundancy check) that is used to verify the integrity of data

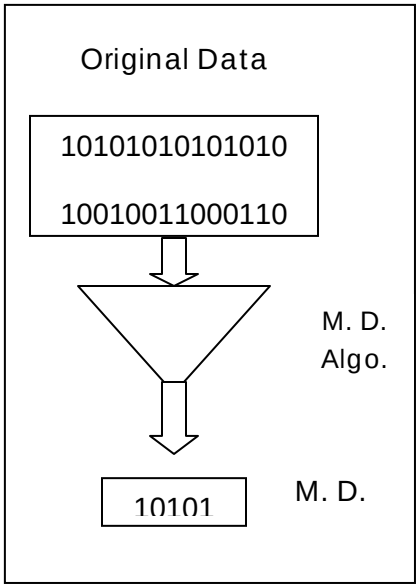
In CRC or LRC we compute parity bit and along with data we divert parity towards receiver. Now receiver can separates parity and data bit and computes parity of received data. If computed parity matches with the received parity then it indicate that there is no error otherwise there is the error present in the data. So here we can say CRC and LRC are the fingerprints of original message

Concept of message digests:

- ✓ Lets assume we want to calculate message digest o a number 7391753
- ✓ We multiply each digit in the number with the next digit (excluding if it is 0) and discarding the first digit of multiplication operation if result is two digit number
- ✓ The process is shown below for original number 739174 fig(a) shows simplest MD example and fig(b) shows MD Concept

Operation	Result
Multiply 7*3	21
Discard first digit	1
Multiply 1*9	9
Multiply 9*1	9
Multiply 9*7	63
Discard 6	3
Multiply 3*4	12
Discard 1	2

MD=2



- ✓ Above fig shows the simplest example of MD usually message digest consist of 128 or more bits
- ✓ This means that chance of any two message digest being the same is anything between 0 and at least 2^{128}
- ✓ MD length should be long with a purpose to minimize the scope for two messages digest being the same

Requirement of message digest:

- ✓ Given message digest algorithm should be very easy to find its corresponding message digest
- ✓ Given a message digest it should be very difficult to find the original message for which the digest was created
- ✓ For given any two messages if we calculate message digest ,then the two message digest must be different
- ✓ If any two messages produces the same message digest it violates the principles and it is called as a collision
- ✓ Usually M.D. algo uses 128 or 160 bits this means that the chances of any two message digest being the same are 2128 and 2 160 respectively. Clearly it seems to be possible in theory but not in practical

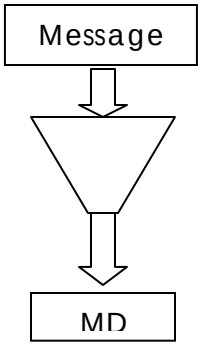


Fig. For same data M.D. may be always same

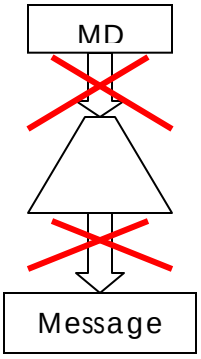


Fig. M.D. Should not work in apposite direction

Fig: Message digests should not reveal anything about original message as well should be collision free

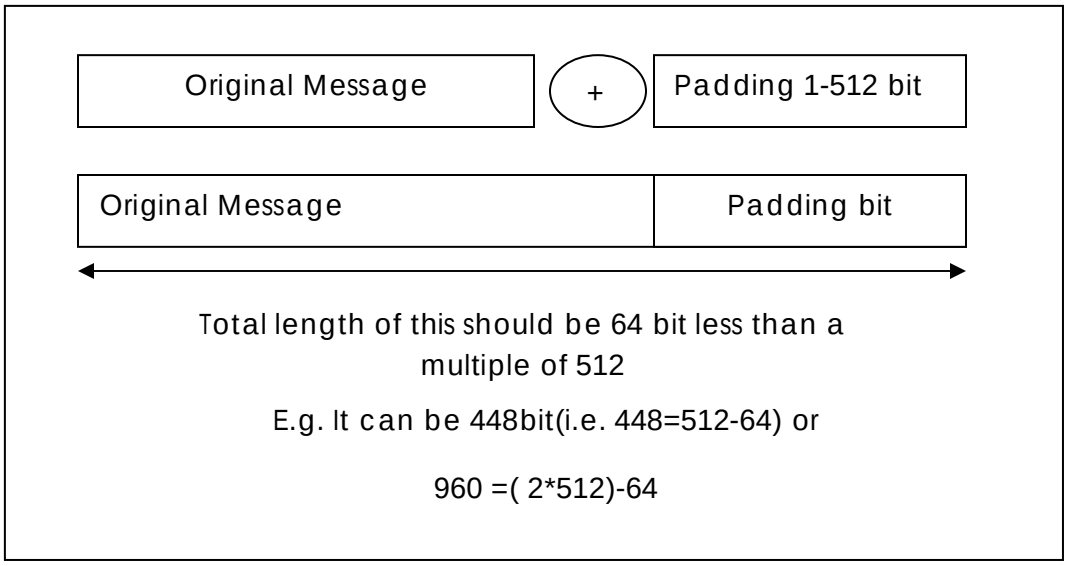
MD5:

- ✓ MD5 is message digest algorithm developed by Ron Rivest
- ✓ MD5 has roots in series of message digest algorithms which were predecessor to MD5 and all of them are developed by Rivest
- ✓ MD5 is quiet fast and produces 128 bit message digest
- ✓ Over years researchers have developed potential weakness in MD5. However so for MD5 has been able to successfully defend itself against collision

Working of MD5

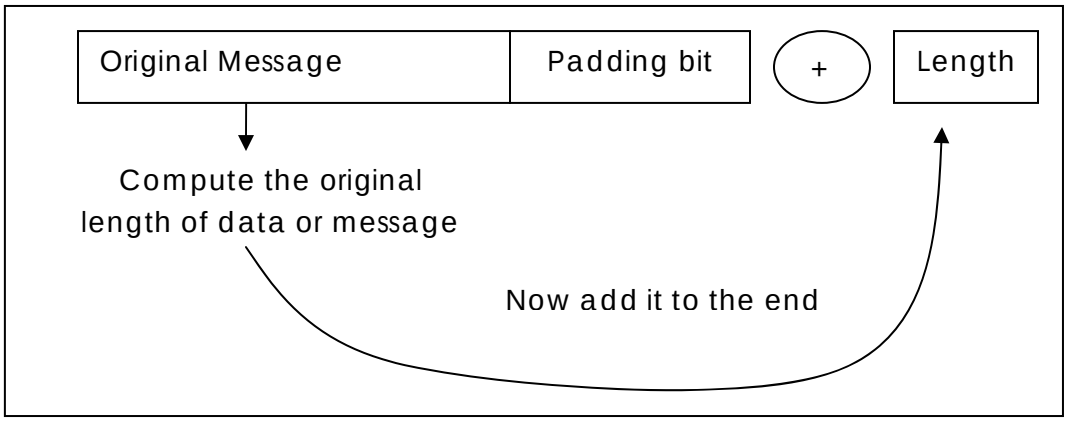
Step-1 Padding:-

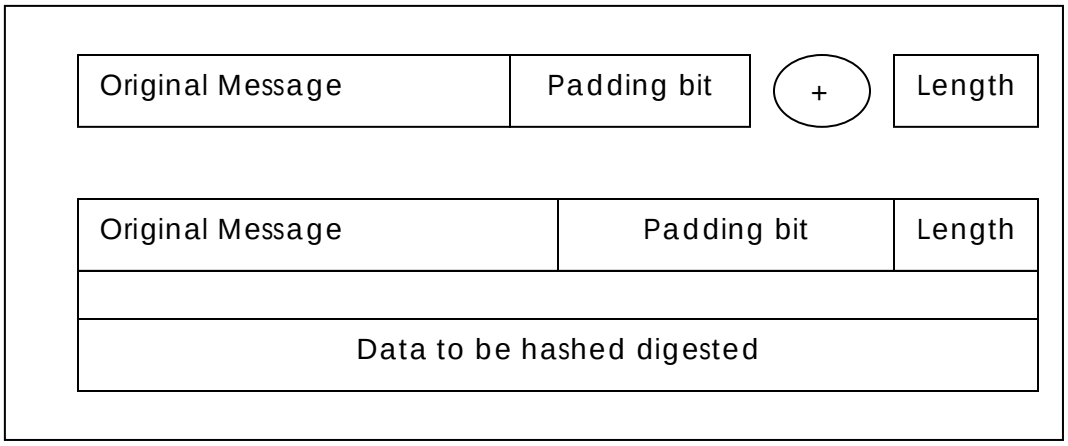
- ✓ Ist step in MD5 is to add padding bits to the original message aim of this step is to make the length of original message equal to a value which is 64 bit less than an exact multiple of 512
- ✓ The padding consist of single 1-bit followed by as many 0- bits as required
- ✓ Note: Padding is always added even if the original message is already a multiple of 512
- ✓ Following Fig. Shows the padding process



Step-2 Append length:

- ✓ After padding bits are added next step is to calculate the original length of message and add it to the end of the message





- ✓ Length of message is calculated excluding padding bits e.g. original message is of 1000 bit and we added 472 bit to make the length of message 64 bit less than 1536(a multiple of 512)
- ✓ Length is expressed in terms of 64 bit
- ✓ If length of message exceeds 64 bit(i.e. it is greater than 264) then only 64 bit of length is used by performing modulo operation
- ✓ After that 64 bit of message appended this becomes the final message (i.e. message to be hashed) which is exact multiple of 512

Step-3 Divide the Input in 512 bit block:

- ✓ Now we divide the input message in to block each of 512 bit lengthy as shown below

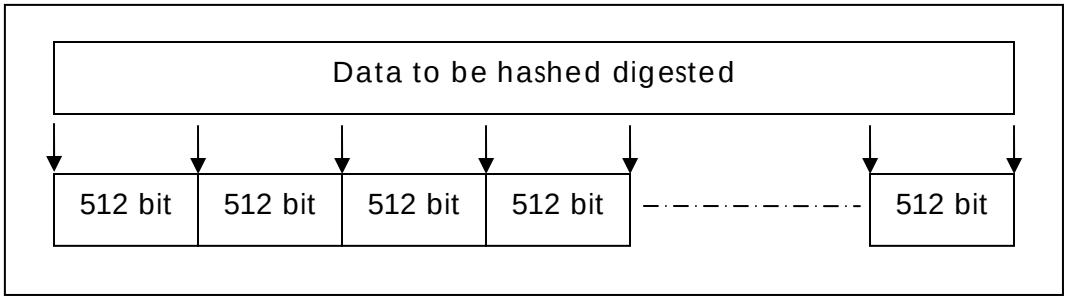


Fig: Data is divided into 512-bit blocks

Step-4 Initialize chaining variables:

- ✓ In this step 4 variables called as chaining variable are initialized they are called A,B,C,D each of which is 32 bit

Step-5 Process the blocks:

- ✓ After all the initialization real algorithm begins we will see the step by step working
- ✓ **5.1:** Copy all four chaining variables in corresponding variables a,b,c,d

  Thus a=A, b=B, c=C and d=D

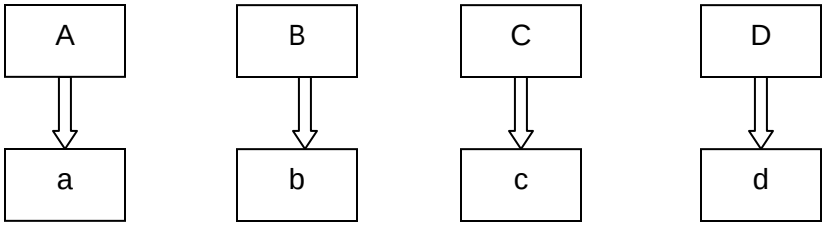


Fig: Copying Chaining variables to temporary variables

- ✓ Actually algo consider combination of a,b,c and d as a 128 bit register (abcd)
- ✓ Register abcd is useful in the actual algorithm for holding the intermediate as well final result

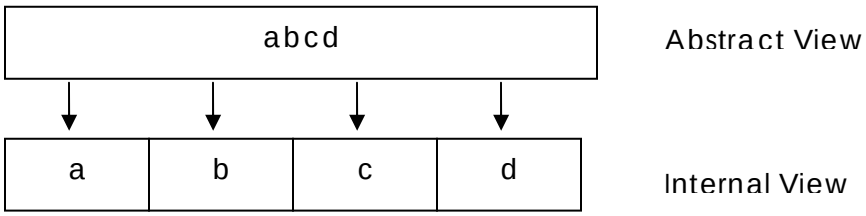


Fig Abstract and Internal view of variables

- ✓ **5.2:** Divide current 512 bit block to 16 subblocks thus each subblock contains 32 bits each as shown

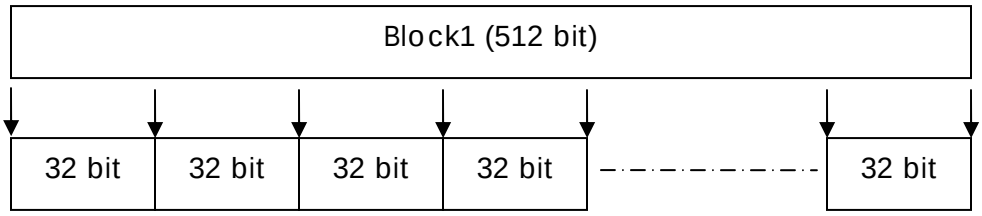


Fig: 16 Sub blocks within block

✓ **5.3:** Now we have **four** rounds in each round process all 16 sub blocks belonging to a block input to each round are

1. All 16 sub blocks
2. Variables a,b,c and d
3. some constant designated as t and shown in following fig

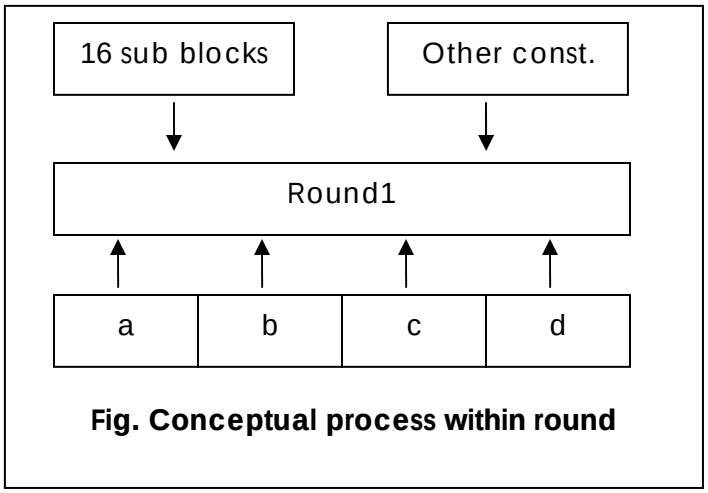


Fig. Conceptual process within round

✓ Let's summarize the iteration of all the four rounds. In each case output of intermediate as well as final iteration is copied into register abcd. Note we have 16 such iteration in each round

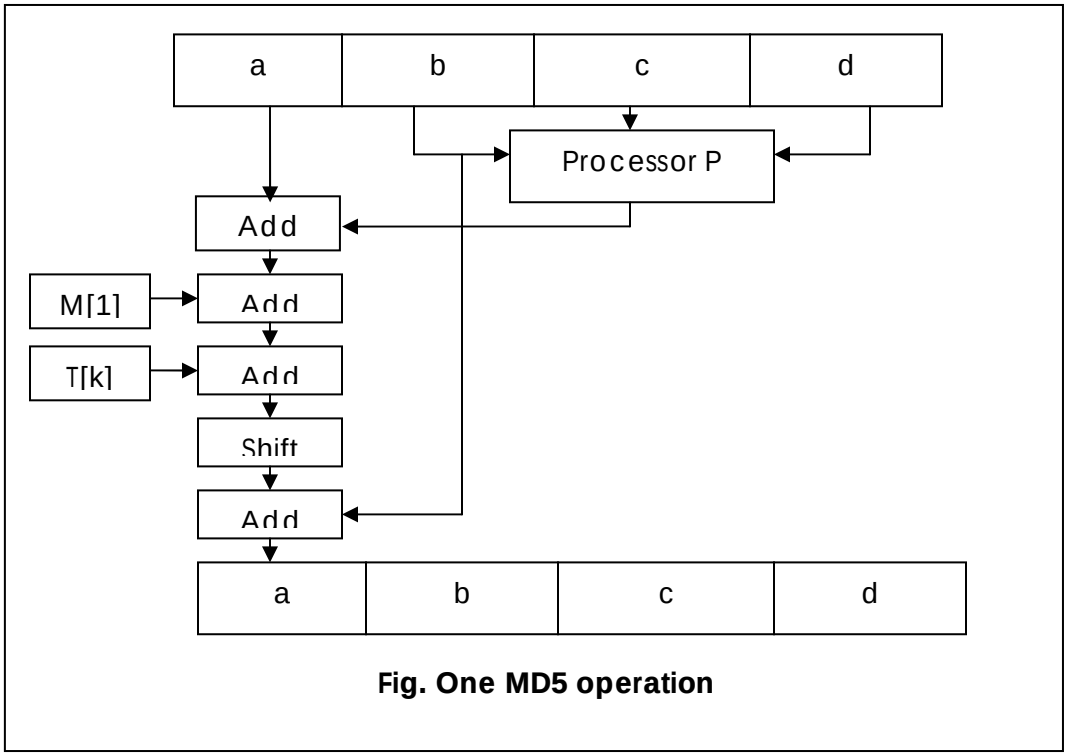


Fig. One MD5 operation

We can mathematically express MD5 as

- ✓ a=b+((a+process p(b,c,d) + M[i] +T[k])<<<S)
- ✓ where a,b,c,d are chaining variables
- ✓ Process P= A nonlinear operation described subsequently
- ✓ M[i]=M[2 * 16 + i] which is i^{th} 32 bitword in q^{th} 512 bit block of message
- ✓ T[k] = A constant as discussed
- ✓ <<< Circular left shift by s bits

Understanding Process P

Process p is different in four rounds in simple term process P is nothing but some basic Boolean operation on b, c, and d as shown

Round	Process P
1	(b AND C) OR ((NOT b) AND (d))
2	(b AND d) OR (C AND (NOT d))
3	B XOR c XOR d
4	C XOR (v OR (NOT d))

SHA (Secure hash algorithm):

- ✓ NIST (National Institute of standard and Technology) along with NSA(National standard agency) developed secure hash based algorithm SHA
- ✓ SHA was published as Federal information processing standard (FIPS pub 180)
- ✓ It was revise to FIPS Pub 180-1 in 1995 and name was changed to SHA-1
- ✓ SHA-1 is revised version of MD4
- ✓ SHA works with any message that is less than 264 bit in length
- ✓ The output of SHA is MD-160 bit length (i.e. 32 bit more than MD5)

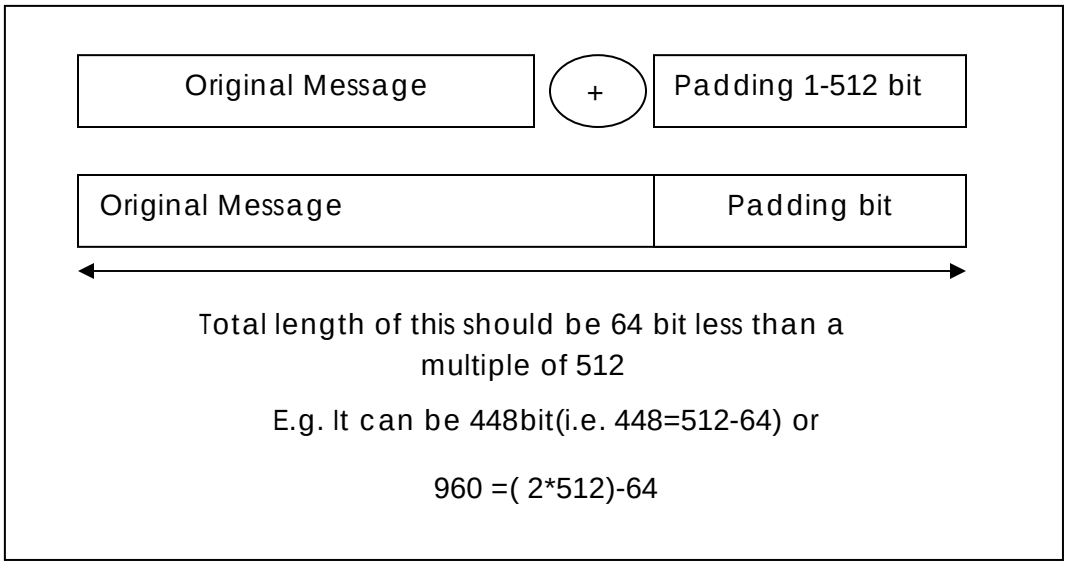
Word secure was decided based on two features , SHA is designed to be computationally infeasible to

1. to obtain the original message from MD
2. Find two messages producing same MD

Working of SHA

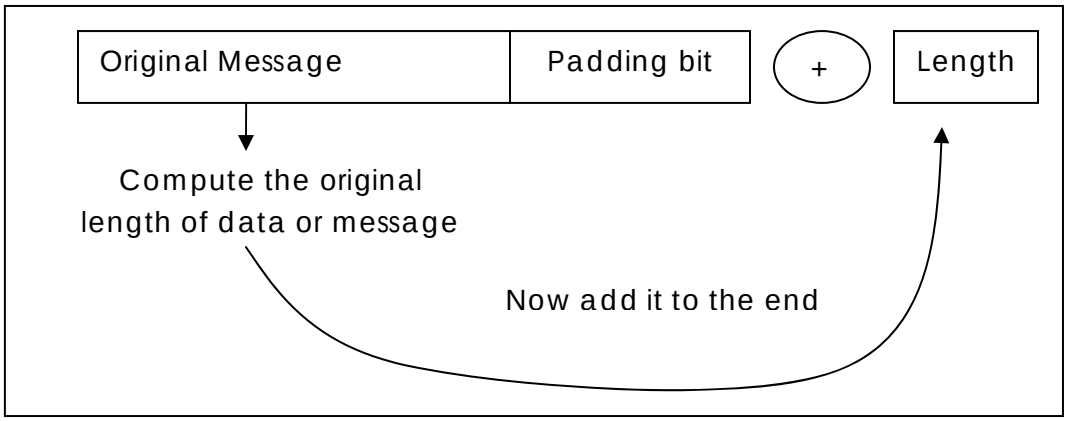
Step-1 Padding:-

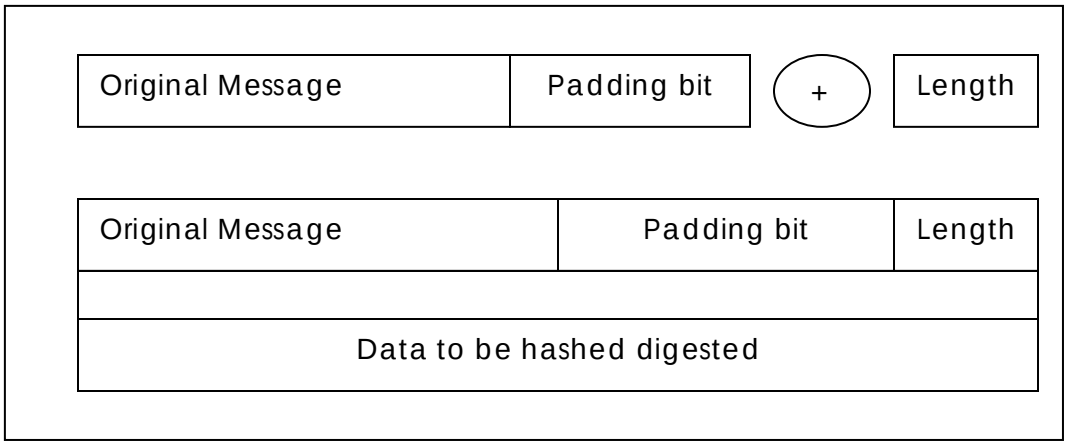
- ✓ Ist step in MD5 is to add padding bits to the original message aim of this step is to make the length of original message equal to a value which is 64 bit less than an exact multiple of 512
- ✓ The padding consist of single 1-bit followed by as many 0- bits as required
- ✓ Note: Padding is always added even if the original message is already a multiple of 512
- ✓ Following Fig. Shows the padding process



Step-2 Append length:

- ✓ After padding bits are added next step is to calculate the original length of message and add it to the end of the message





- ✓ Length of message is calculated excluding padding bits e.g. original message is of 1000 bit and we added 472 bit to make the length of message 64 bit less than 1536(a multiple of 512)
- ✓ Length is expressed in terms of 64 bit
- ✓ If length of message exceeds 64 bit(i.e. it is greater than 264) then only 64 bit of length is used by performing modulo operation
- ✓ After that 64 bit of message appended this becomes the final message (i.e. message to be hashed) which is exact multiple of 512

Step-3 Divide the Input in 512 bit block:

- ✓ Now we divide the input message in to block each of 512 bit lengthy as shown below

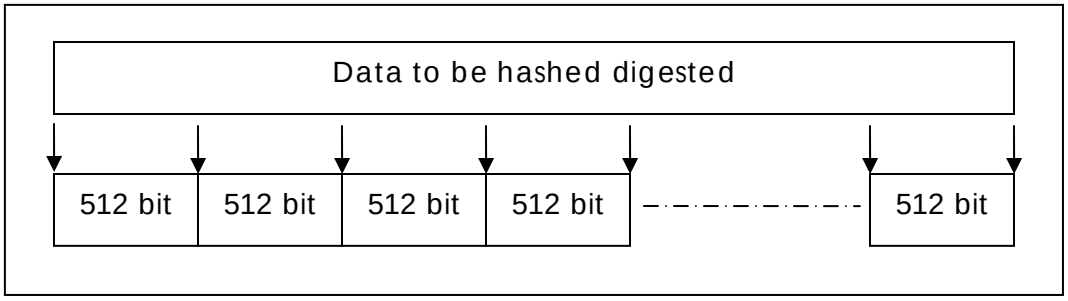


Fig: Data is divided into 512-bit blocks

Step-4 Initialize chaining variables:

- ✓ In this step 4 variables called as chaining variable are initialized they are called A,B,C,D & E each of which is 32 bit

Step-5 Process the blocks:

- ✓ After all the initialization real algorithm begins we will see the step by step working
- ✓ **5.1:** Copy all four chaining variables in corresponding variables a,b,c,d

  Thus a=A, b=B, c=C , d=D and e=E

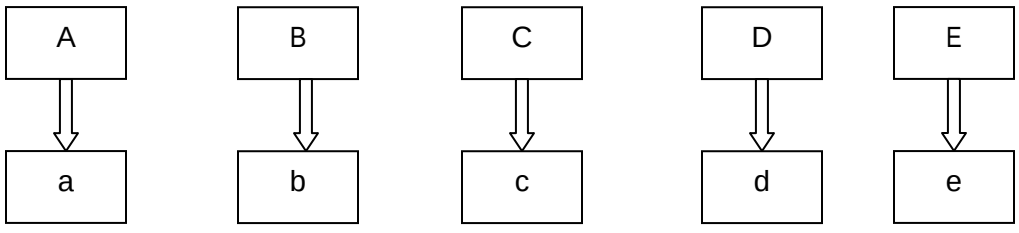


Fig: Copying Chaining variables to temporary variables

- ✓ Actually algo consider combination of a,b,c ,d and e as a 160 bit register (abcd)
- ✓ Register abcd is useful in the actual algorithm for holding the intermediate as well final result

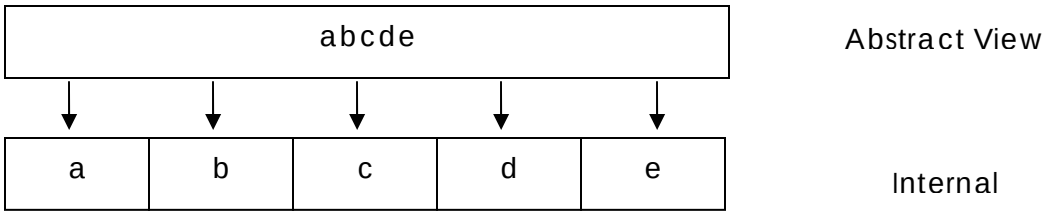


Fig Abstract and Internal view of variables

- ✓ **5.2:** Divide current 512 bit block to 16 subblocks thus each subblock contains 32 bits each as shown

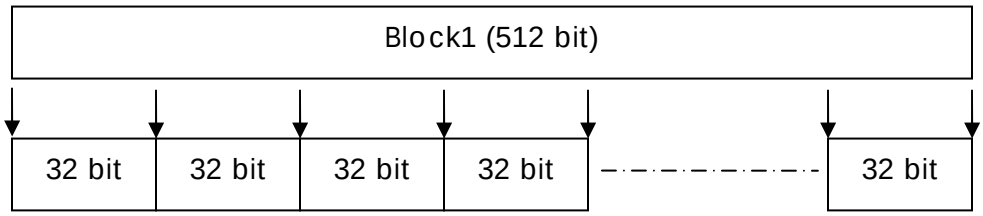


Fig: 16 Sub blocks within block

- ✓ **5.3:** Now we have **four** rounds in each round process all 16 sub blocks belonging to a block input to each round are

  1. All 16 sub blocks
  2. Variables a,b,c, d and e
  3. some constant designated as t and shown in following fig

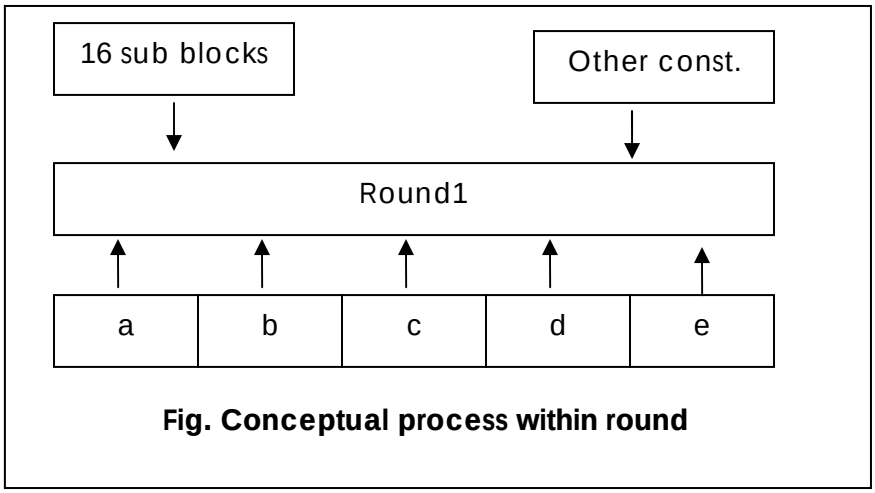
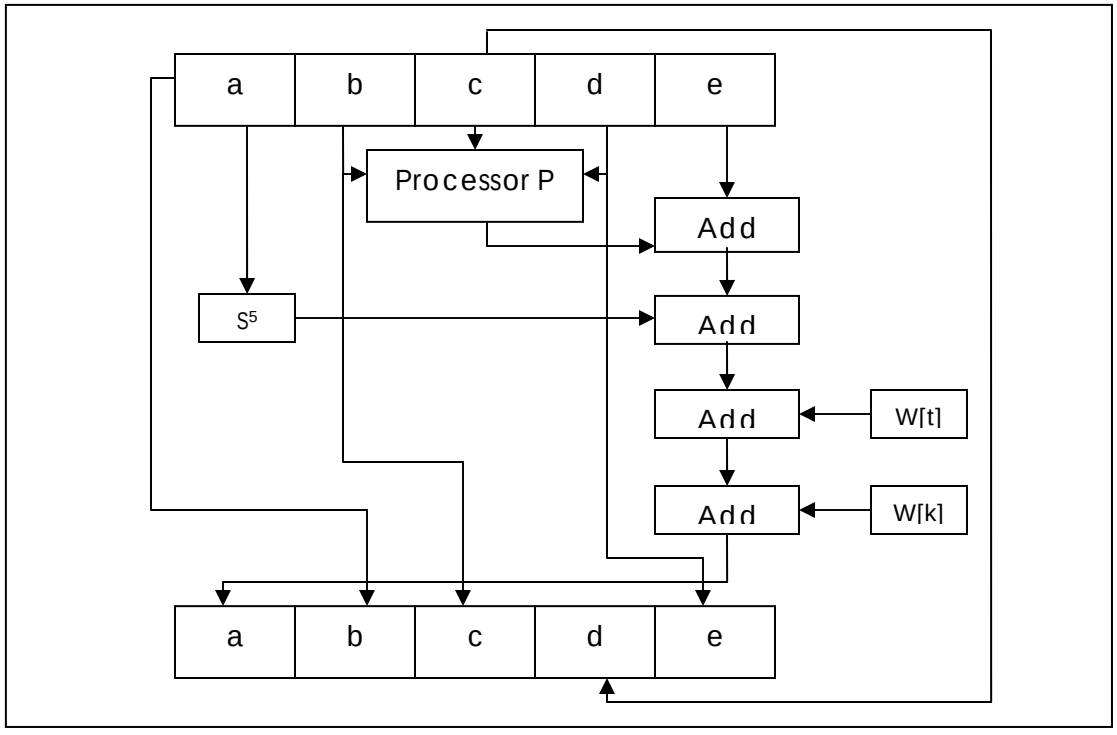


Fig. Conceptual process within round

- ✓ Let's summarize the iteration of all the four rounds. In each case output of intermediate as well as final iteration is copied into register abcd. Note we have 16 such iteration in each round



Mathematically iteration consist of following operations:

abcde=(e+process P + S^5(a)+W[t]+k[t] a.S^{30} (b) c, d)

where abcde=register made up of abcde var

Process P=logical operation

S^t – Circular left shift of 32 bit subblock by t bits

W[t] = A-32 bit value derived from current 32 bit subblock

K[t]= One of the 5 additive constants as defined earlier

Understanding Process P

Process p is different in four rounds in simple term process P is nothing but some basic Boolean operation on b, c, d and e as shown

Round	Process P
1	(b AND C) OR ((NOT b) AND (d))
2	(b XOR c) XOR d
3	(b AND c) OR (b and d) OR (c AND d)
4	b XOR c XOR d

Comparison of MD5 with SHA1:

Algorithm	MD5	SHA1
MD length	128 bit	160 bit
Breaking possibility	Requires 2128 operations to break	Requires 2160 possibilities to break i.e. more Secure
Attack	Attacks reported	No such claim
Implementation	Software implementation	Software Implementation

Message Authentication Code:

Similar to message digest (MD) however difference is in case MD there is no cryptographic encryption while in MAC there is need of cryptographic algorithm

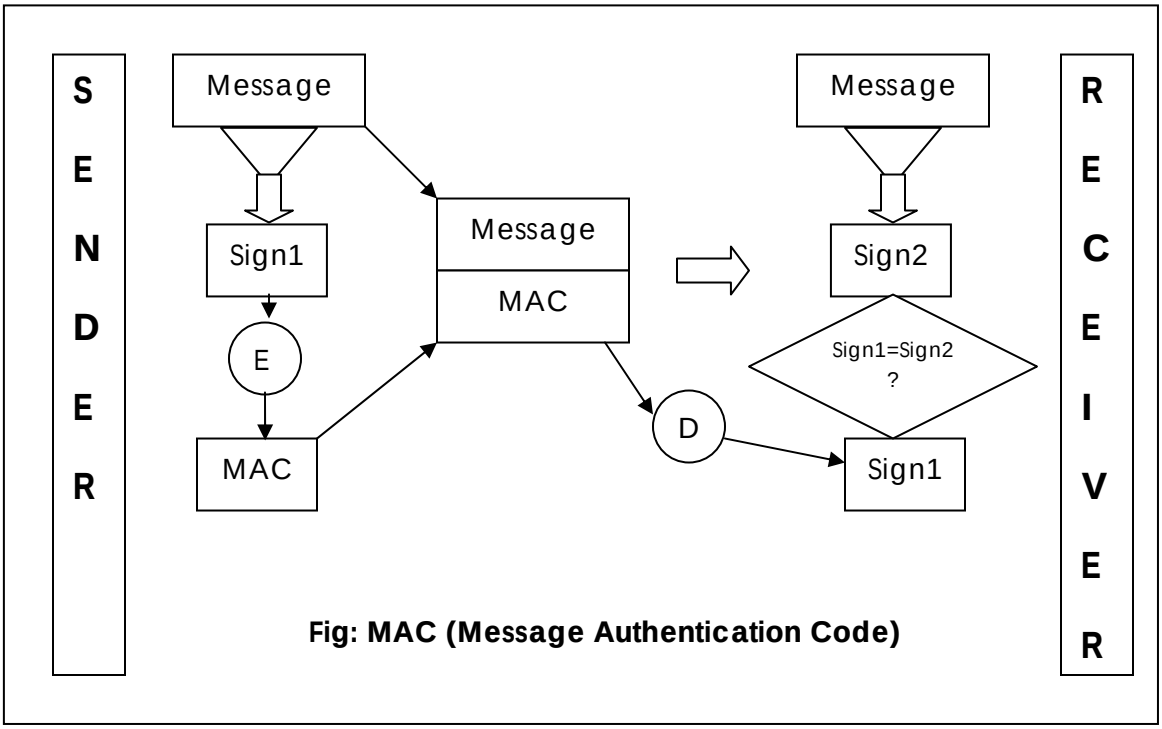


Fig: MAC (Message Authentication Code)

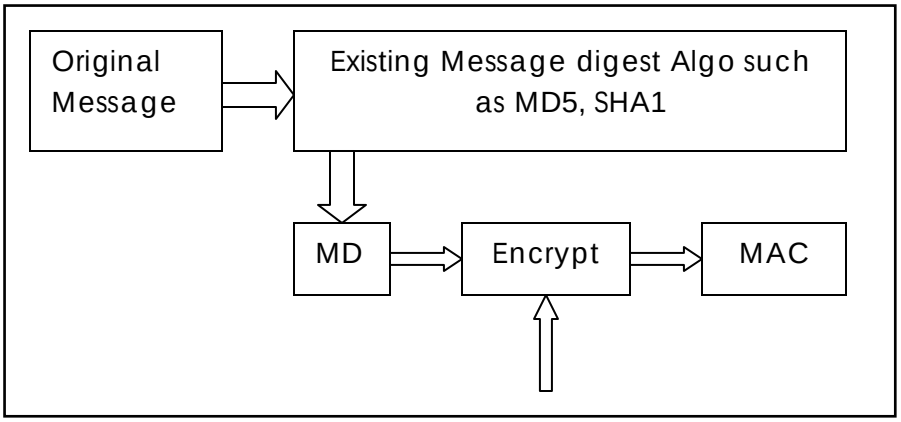
A and B share dynamic secret key not known to any one

- ✓ A sends original message and MAC to B
- ✓ B creates own signature and deciphers received MAC
- ✓ B compares two message equivalent signatures if there is the difference it concludes that there is error in the message

HMAC (Hash based message authentication code):

HMAC is selected as mandatory security implementation for Internet protocol and also used in SSL

Fundamental idea behind HMAC is to reuse the existing message digest algo such as MD-5 or SHA-1



Drawbacks of MAC & HMAC:

1. Its having problem regarding symmetric key exchange
2. HMAC can not be used if no of receivers are greater than one
3. No means to know that message was prepared by a particular user

RIPE-MD:

- ✓ Developed by European community project RIPE[1305]
- ✓ Algorithm is variant of MD4
- ✓ Designed to resist known crypt analytical attack and produces a 128 bit hash value
- ✓ The rotation and order of the message word are modified

Questions:

1. Write a note on primality checking algorithms
2. Explain Working of RSA
3. Discuss the working of asymmetric key cryptography
4. If A want to send message to B through asymmetric key cryptography , what would be the typical steps involved
5. Explain the working of Diffie & Hellman key exchange algorithm
6. What is man in middle or woman in middle attack in connection with Diffie and Hellman Algo explain with example
7. Explain the working of RSA with suitable example
8. Describe the advantages and disadvantages of symmetric and asymmetric key cryptography
9. What is key wrapping? How is it useful
10. Write a note of Elgamal and ECC with strengths and weaknesses
11. Write a note on key management techniques in PKCS
12. Explain the working of public announcement & public available directory with requirements and drawbacks
13. Explain the working of public certificates for exchanging keys
14. Explain the working of session key exchange through PKCS system with authentication and confidentiality
15. Write a note on public key cryptography standards
16. What are the key requirements of message digests?
17. What is the problem with exchanging of public keys?
18. What is collision related to message digest
19. Explain the step by step working of MD5
20. Explain Step by step working of SHA
21. Compare SHA with MD5
22. Why SHA is more secure than MD5?
23. What is the difference between message digest and MAC?
24. What is difference between MAC and HMAC? Explain by drawing suitable block diagram
25. Explain RIPEMD with its working details with bits/bytes of Hash value produced