ASYMMETRIC KEY CRYPTOGRAPY

EXTC – BE – DATA COMPRESSION AND CRYPTOGRAPHY

Ms. Vandana Sawant

Assistant Professor

Dept. of Electronics & Telecommunication Engineering,

SIES Graduate School of Technology

QUESTIONS

- Explain RSA algorithm with an example
- Explain the RSA encryption and decryption algorithm. Specifically explain why the decrypted message is the same as the plain text
- Using modular arithmetic and theorem prove that decrypted text is same as plain text in the RSA algorithm
- explain RSA in detail and also discuss attacks on RSA
- What is significance of prime numbers in public key cryptography ? Explain RSA algorithm with suitable example

QUESTIONS

- Write shory note on Deffe Hellmen key exchange
- prove that the key exchanged between user A and B with Deffe Hellmen key exchange algorithm is the same
- Explain Deffe Hellmen key exchange algorithm with an example. Also explain attack on Deffe Hellmen key exchange
- Explain HASH and MAC functions with their role in cryptography
- Write short note on HASH and MAC functions
- What do you mean by secure HASH algorithm explain in detail what are the characteristics of secure HASH algorithm
- What is message digest ? Explain HMAC algorithm
- What is MDC and MAC ? Explain HMAC in detail
- Explain hashed MAC with suitable diagram

QUESTIONS

- What is digital signature ? How are they implemented
- Write short note on digital signatures
- Explain digital signature using RSA with example
- Explain anyone digital signature algorithm in detail

OBJECTIVES OF LECTURE

- Students should be able to
  - Know about asymmetric key cryptography.

PUBLIC KEY CRYPTOGRAPHY

- The public key cryptography is based on the asymmetric keys
- Following are some basic principles of public key based cryptosystems
- Public key cryptosystem required the use of two keys a public key and a private key
- Public keys are widely known
- Private keys is kept secrete with its owner
- The two keys are mathematically related and form a key pair
- One key in the pair cannot be used to derive the other key in the key pair

PUBLIC KEY CRYPTOGRAPHY

- Any key in the key pair can be used for encryption. The other key then must be used for decryption .
- Sender and the receiver both must have their own key pairs
- In the section you are going to learn about various asymmetric key base algorithm

RSA

- RSA named after its inventors Ron Rivest , Adi Shamir and Leonard Adleman, is an asymmetric key based algorithm
- It can be used for confidentiality , authentication and nonrepudiation
- RSA is based on finding prime factors for very large numbers
- Number of digits 309,RSA key length 1024 bit
- Let's understand how RSA derives public and private keys and how does encryption and decryption process work
- Choose two random large prime numbers p and q
- Multiply the numbers , n=p*q

RSA

- Choose a random integer to be encryption key e such that e and (p-1)(q-1) are relatively prime
- Decryption key is computed as d=e^{-1}mod([p-1]*[q-1])
- the public key=(n,e)
- The private key=(n,d)
- For encrypting message M with public key (n,e) you get cipher text C=M^emodn
- For decrypting cipher text with private key(n,d) you get plain text M=C^dmodn.

RSA

- Encrypt the plain text 63 using RSA algorithm which uses prime numbers p=7 and q=11. The public key e=13. Verify that the decrypted text is same as the plain text.
- Solution:
- n=p*q
- n=7*11=77
- r=(p-1)*(q-1)
- r=(7-1)*(11-1)=60
- d=e^{-1}mod r
- ed=1 mod 60
- 13d=1 mod 60

RSA

- Let's calculate modulo inverse using extended Euclidean algorithm(swapping 13 and 60) q2 can be written as qi+1 where i=1

Index I	Quotient q for I qi+1=ri-1/ri	Remainder r for I ri+1=ri-1-qiri	s for I si+1=si-1-qisi	t for I ti+1=ti-1-qiti
0		ro=60	so=1	to=0
1		r1=13	s1=0	t1=1
2	q2=ro/r1 = 60/13=4	r2=ro-r1*q2 = 60-13*4=8	s2=so-s1*q2 = 1-0*4=1	t2=to-t1*q2 = 0-1*4=-4
3	q3=r1/r2 13/8=1	r3=r1-r2*q3 = 13-8*1=5	s3=s1-s2*q3 =0-1*1=-1	t3=t1-t2*q3 =1-(-4)*1=5
4	q4=r2/r3 8/5=1	r4=r2-r3*q4 =8-5*1=3	s4=s2-s3*q4 =1-(-1)*1=2	t4=t2-t3*q4 =-4-5*1=-9
5	q5=r3/r4 =5/3=1	r5=r3-r4*q5 =5-3*1=2	s5=s3-s4*q5 =-1-2*1=-3	t5=t3-t4*q5 =5-(-9)*1=14
6	q6=r4/r5 =3/2=1	r6=r4-r5*q6 =3-2*1=1	s6=s4-s5*q6 =2-(-3)*1=5	t6=t4-t5*q6 =-9-14*1=-23
7	q7=r5/r6	r7=r6-r7*q7	Do not	Do not

RSA

- Let's calculate modulo inverse using extended Euclidean algorithm(swapping 13 and 60) q2 can be written as qi+1 where i=1

Index I	Quotient q for I qi+1=ri-1/ri	Remainder r for I ri+1=ri-1-qiri	s for I si+1=si-1-qisi	t for I ti+1=ti-1-qiti
0		60	1	0
1		13	0	1
2	60/13=4	60-13*4=8	1-0*4=1	0-1*4=-4
3	13/8=1	13-8*1=5	0-1*1=-1	1-(-4)*1=5
4	8/5=1	8-5*1=3	1-(-1)*1=2	-4-5*1=-9
5	5/3=1	5-3*1=2	-1-2*1=-3	5-(-9)*1=14
6	3/2=1	3-2*1=1	2-(-3)*1=5	-9-14*1=-23
7	2/1=2	2-1*2=0	Do not calulate	Do not calculate

RSA

- Let's calculate modulo inverse using extended Euclidean algorithm(swapping 13 and 60) q2 can be written as qi+1 where i=1

Index I	Quotient q for I qi+1=ri-1/ri	Remainder r for I ri+1=ri-1-qiri	s for I si+1=si-1-qisi	t for I ti+1=ti-1-qiti
0		60	1	0
1		13	0	1
2	60/13=4	60-13*4=8	1-0*4=1	0-1*4=-4
3	13/8=1	13-8*1=5	0-1*1=-1	1-(-4)*1=5
4	8/5=1	8-5*1=3	1-(-1)*1=2	-4-5*1=-9
5	5/3=1	5-3*1=2	-1-2*1=-3	5-(-9)*1=14
6	3/2=1	3-2*1=1	2-(-3)*1=5	-9-14*1=-23
7	2/1=2	2-1*2=0	Do not calulate	Do not calculate

RSA

- Let's re swap the values
- X=-23
- Y=5
- Putting the values in the extended Euclidean algorithm you get
- ax+by=1
- 13(-23)+60(5)=1
- Since you have to find multiplicative inverse in mod 60 divide both the sides by mod 60
- 13(-23)mod60+300mod60=1mod60
- 13(-23)mod60+0=1mod60

RSA

- Recall our discussion on calculating mod for negative numbers, you need to keep adding mod until the number turns positive and then calculate mod on the positive number you got
- -23+60=37
- 37mod60=37
- Hence 13 (37)mod60
- Hence the value of decrypting key d is 37
- Now you have all the values needed for encryption and decryption
- As per RSA

RSA

- C=M^emodn
- C=63^{13}mod77
- C=28
- Hence encrypting the plain text message 63 would give encrypted text 28
- Let's verify that using decrypting key d=37,do we get the plain text back
- As per RSA
- M=C^dmodn
- M=28^{37}mod77
- M=63
- Hence encrypted text 28 when decrypted using the decryption key 37 , gives the plain text 63 back

RSA

- Compute the encrypted and decrypted text using RSA algorithm for plaintext 88 public key is (n, e) =(187, 7)
- Solution:
- n=187
- n=p*q
- Assuming p=11 and q=17
- r=(p-1)*(q-1)
- r=(11-1)*(17-1)=160
- d=e^{-1}mod r
- ed=1 mod 160
- 7d=1 mod160

RSA

- Let's calculate modulo inverse using extended Euclidean algorithm(swapping 7 and 160) q2 can be written as qi+1 where i=1

Index I	Quotient q for I qi+1=ri-1/ri	Remainder r for I ri+1=ri-1-qiri	s for I si+1=si-1-qisi	t for I ti+1=ti-1-qiti
0		ro=160	so=1	to=0
1		r1=7	s1=0	t1=1
2	q2=ro/r1 = 160/7=22	r2=ro-r1*q2 = 160-7*22=6	s2=so-s1*q2 = 1-0*22=1	t2=to-t1*q2 = 0-1*22=-22
3	q3=r1/r2 7/6=1	r3=r1-r2*q3 = 7-6*1=1	s3=s1-s2*q3 =0-1*1=-1	t3=t1-t2*q3 =1-(-22)*1=23
4	q4=r2/r3 6/1=6	r4=r2-r3*q4 =6-1*6=0	Do not calculate	Do not calculate

RSA

- Let's calculate modulo inverse using extended Euclidean algorithm(swapping 7 and 160) q2 can be written as qi+1 where i=1

Index I	Quotient q for I qi+1=ri-1/ri	Remainder r for I ri+1=ri-1-qiri	s for I si+1=si-1-qisi	t for I ti+1=ti-1-qiti
0		160	1	0
1		7	0	1
2	160/7=22	160-7*22=6	1-0*22=1	0-1*22=-22
3	7/6=1	7-6*1=1	0-1*1=-1	1-(-22)*1=23
4	6/1=6	6-1*6=0	Do not calculate	Do not calculate

RSA

- Let's re swap the values
- X=23
- Y=-1
- Putting the values in the extended Euclidean algorithm you get
- ax+by=1
- 7(23)+160(-1)=1
- Since you have to find multiplicative inverse in mod 160 divide both the sides by mod 160
- 7(23)mod160-160mod160=1mod160
- 7(23)mod160-0=1mod160
- 7(23) mod 160=1mod160

RSA

- The modulo inverse of
- 7mod160=23
- Hence the value of decrypting key d is 23
- Now you have all the values needed for encryption and decryption
- As per RSA

RSA

- C=M^e modn
- C=88^7 mod187
- C=11
- Hence encrypting the plain text message 88 would give encrypted text 11
- Let's verify that using decrypting key d=23,do we get the plain text back
- As per RSA
- M=C^d modn
- M=11^{23} mod187
- M=88
- Hence encrypted text 28 when decrypted using the decryption key 23 , gives the plain text 88 back

Attacks on RSA

- 1 brute force attack
- 2 common modulus
- 3 choosing smaller number
- 4 man in the middle attack

Diffie Hellman key exchange algorithm

- The Diffie Hellman algorithm provides a way of generating a shared secrete between the sender and receiver in such a way that the secrete need not be exchanged or transferred over the communication medium
- Alex chooses two prime numbers g and p and also secrete number a . he calculates value of A such that A=g^a mod p. he then sends g, p and A to Bobby. Note here that Alex does not share the secrete number a with Bobby
- Similarly Bobby chooses a secrete number band computes the value of B such that B= g^b mod p. she then send B to Alex.
- Alex computes the shared key at his end as shared key S= $B^a \bmod p$

Diffie Hellman key exchange algorithm

- Bobby computes the shared key at her end as shared key
  S= $A^b \ mod p$
- The values of the shared key S derive in above steps are equal due to mod operation.
- (g^a modp)^b modp= $g^{ab} \ mod p$
- (g^b modp)^a modp= $g^{ba} \ mod p$
- It doe snot matter which step you do earlier. both the keys created would be equal and can be used with any of the algorithm DES and AES to encrypt the information and communicate confidentially .

Diffie Hellman key exchange algorithm

- Alice chooses her private key x=3 and Bob chooses y=6. if both of them use the primitive route g=7 for prime p=23 what is the key exchange between Alice and Bob using Diffie Hellman key exchange
- Solution
- g=7
- p=23
- Alice private key is 3
- A=g^a mod p
- A=7^3 mod 23
- A=21

Diffie Hellman key exchange algorithm

- Alice now sends g, p and A to Bob
- Bob's private key is 6
- B= g^b mod p
- B= 7^6 mod 23
- B=4
- Bob sends B to Alice
- Now both the users computes the shared key S at their respective end
- Alice calculates it as
- S= B^a mod p
- S= 4^3 mod 23
- S=18

Diffie Hellman key exchange algorithm

- Bob calculates it as
- S= $A^b \ mod p$
- S= $21^6 \ mod 23$
- S=18
- Hence the shared key between Alice and bob is 18.

Diffie Hellman key exchange algorithm

- Consider a Diffie hellman scheme with a common prime q=11 and primitive root a=2
- (i) show that 2 is primitive root of 11
- (ii) if user A h has public key Y_A=9 what is 'A's private key X_A ?
- (III) if user B has public key Y_B=3 what is the share secret key K?
- Solution
- Lets assume that 2 is a primitive root of 11 then power of 2 mod 11 should produce all the integers from 1 to (11-1) which means it should produce (1,2,3,4,5,6,7,8,9,10).

Diffie Hellman key exchange algorithm

Integer	Mod operation	Resultant Integer
2^0	1mod 11	1
2^1	2mod 11	2
2^2	4mod 11	4
2^3	8mod 11	8
2^4	16mod 11	5
2^5	32mod 11	10
2^6	64mod 11	9
2^7	128mod 11	7
2^8	256mod 11	3
2^9	512mod 11	6

Diffie Hellman key exchange algorithm

- You see that 2 is indeed a primitive root of 11. It produces all the numbers from 1 to 10 as expected
- g=2
- p=11
- For user A
- Public key YA=9 means

A=g^x mod p

- 9=2^x mod 11
- From primitive root table above 2^6mod 11 gives 9
- Hence private key of user A is 6

Diffie Hellman key exchange algorithm

- For user B
- Public key Y_B=3 means

B=g^y mod p

- 3=2^y mod 11
- From primitive root table above 2^8mod 11 gives 3
- Hence private key of user B is 8
- Now both the the users computes the shred KEY S at their respective ends

Diffie Hellman key exchange algorithm

- User A calculate it as
- S=B^xmod p
- S=3^6mod 11
- S=3
- User B calculate it as
- S=A^Ymod p
- S=9^8mod 11
- S=3
- Hence the shared key between user A and B is 3

KEY MANAGEMENT AND DISTRIBUTION

- Asymmetric keys involve public keys and private keys . Public keys are known in general. private keys are kept secure and are in a constant possession of heir respective owners.
- The goal of asymmetric key distribution is to distribute public keys only. Private keys are not distributed .
- Ways to distribute asymmetric keys
- Broadcast
- Public directory
- Public key authority
- certificates

Message integrity and authentication requirements

- Message authentication is a process to ensure that the received message is exactly the same as it was sent
- The message has not been altered in any way
- No addition
- No modification
- No deletion

Message integrity and authentication requirements

- Message authentication function
- Hashing
- Message encryption
- Message authentication code(MAC)

Cryptographic hash function

- Hashing is process of taking any length of input information and finding a unique fixed length representation of that input information
- How does this work ?
- Working of hash algorithm

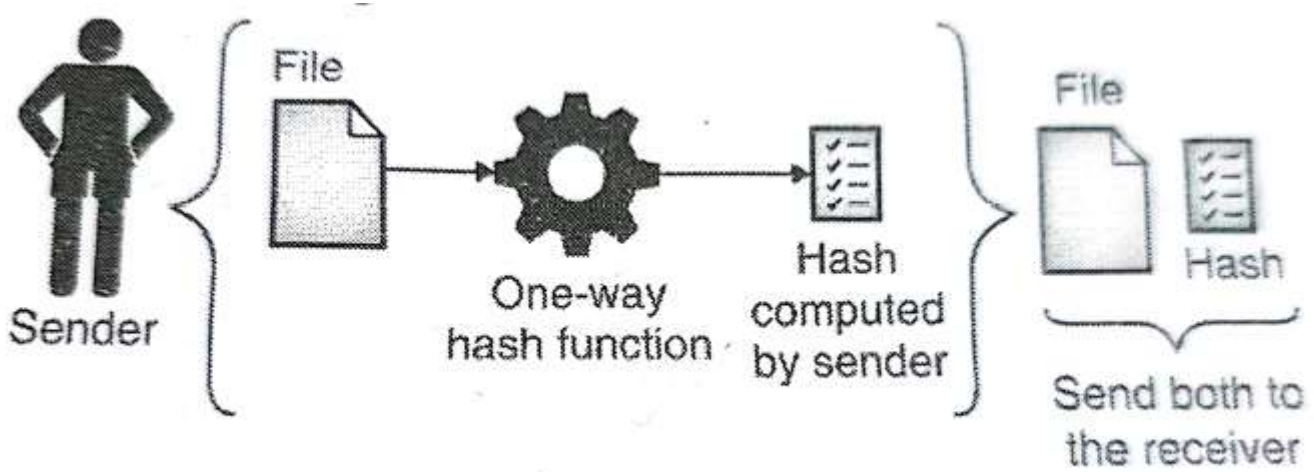


Fig. 5.11.2 : Working of hash algorithms

Cryptographic hash function

- Hashing used for proving data integrity

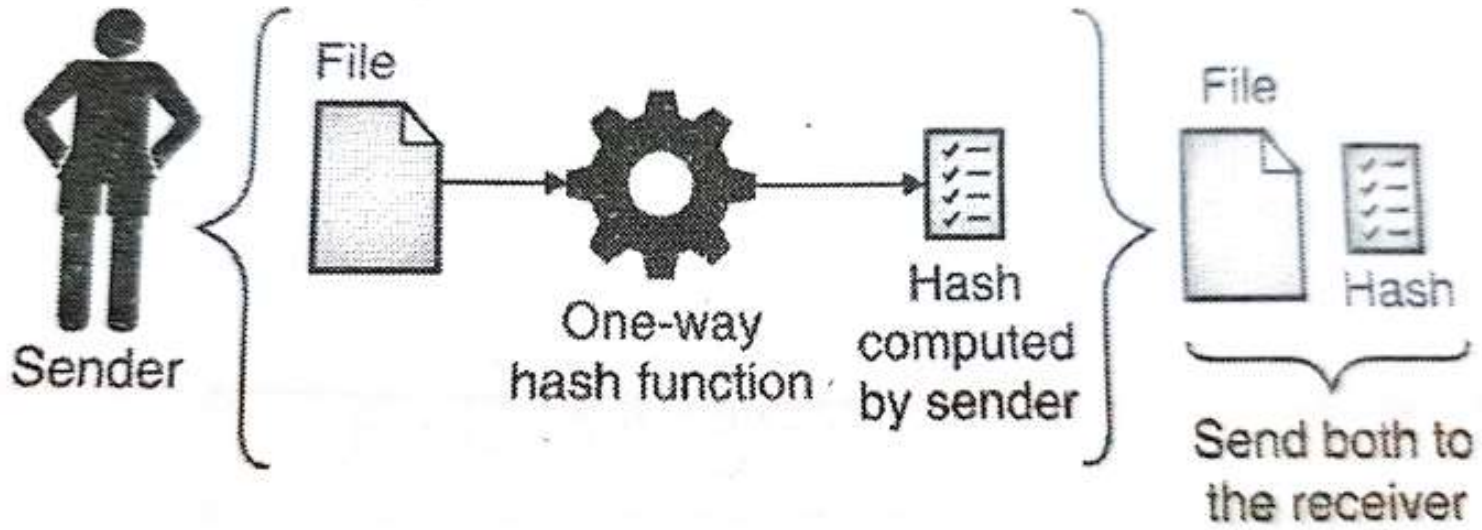


Fig. 5.11.2 : Working of hash algorithms

Cryptographic hash function

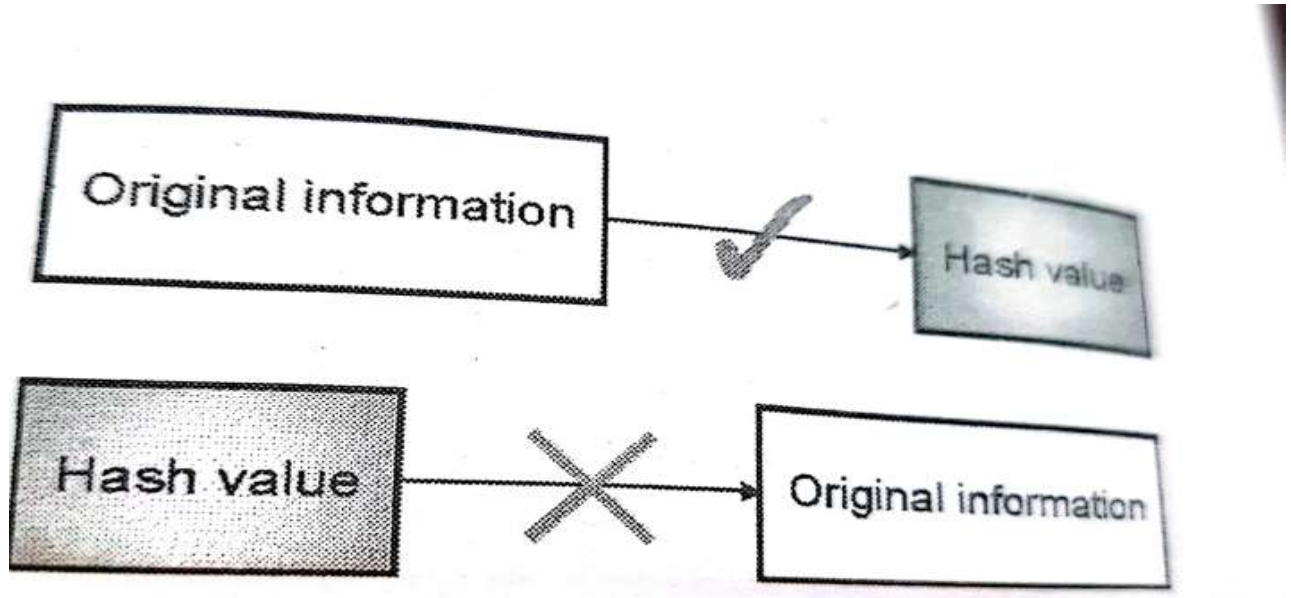
- If the information is to be encrypted following could be one of the sequences for integrity checking
- Create information
- Calculate the hash value
- Encrypt information
- Send encrypted information and hash value
- Decrypt information
- Calculates its hash value at the receiving end
- Match source and destination hash
- If matched process information
- If not matched reject information

Cryptographic hash function

- Characteristics of hash function
- One way only
- Any length input, fixed length output
- No secrecy involved
- Small input variation produces large output variation
- Collisions are possible

Cryptographic hash function

- Characteristics of hash function
- One way only



(O-566) Fig. 5.11.5 : One-way only

Cryptographic hash function

- Characteristics of hash function
- Any length input, fixed length output

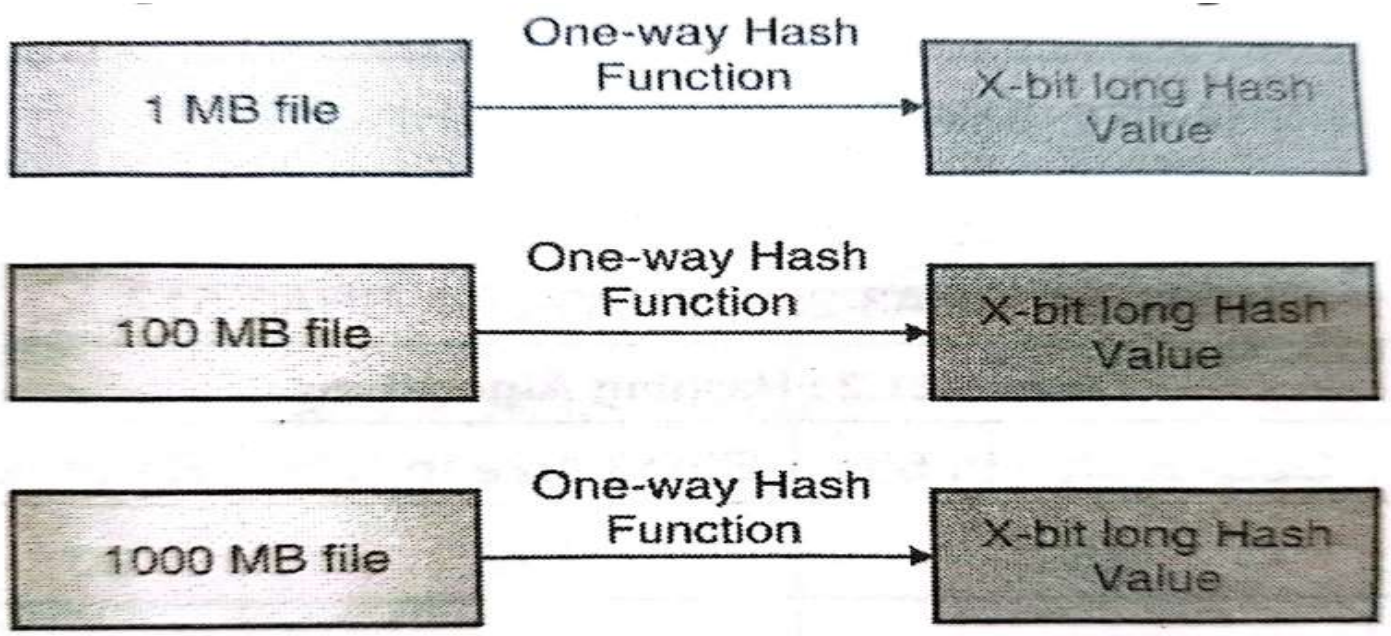


Fig. 5.11.6 : Any length input, fixed length output

Cryptographic hash function

- Characteristics of hash function
- No secrecy involved
- Small input variation produces large output variation
- Collisions are possible

Family of hashing algorithm

- National institute of standard and technology(NIST) has specified the family of hashing algorithm that may or may not be fit for current use in the industry

SHA-1

- Secure hash algorithm 1 is a cryptographic hash function that produces 160 bit long hash value from a given input
- Collisions have been found in the algorithm and hence it is avoided in the industry wherever possible
- The algorithm has two stages
- Pre - processing :
- Involves padding a message, parsing the padded message into equal sized block and then setting initial values to be used for hash computation .
- Hash computation :
- The hash computation generate a message schedule from the padded message and uses the schedule along with various function, constatant and mathematical operation to generate a series of intermediate hash values per round.the final hash value is computed after the last round and is 160 bit long

SHA-3

- Secure hash algorithm 3 is newest addition to the hash function family.it is comprised of various hash function that computes secure hash value .
- SHA-3 is structurally quiet different from earlier version SHA-1 & SHA-2.SHA-3 is a subset of the border cryptography primitive family KECCAK.
- KECCAK is the family of all sponge functions with a KECCAK –p permutation as the underline function and multi-rate padding as padding rule.
- The idea behind the sponge function is to absorb the information and squeeze out the hash value.
- KECCAK-p permutation are specified with two parameter:
- The fixed length of the string that are permuted
- The number of rounds

SHA-3

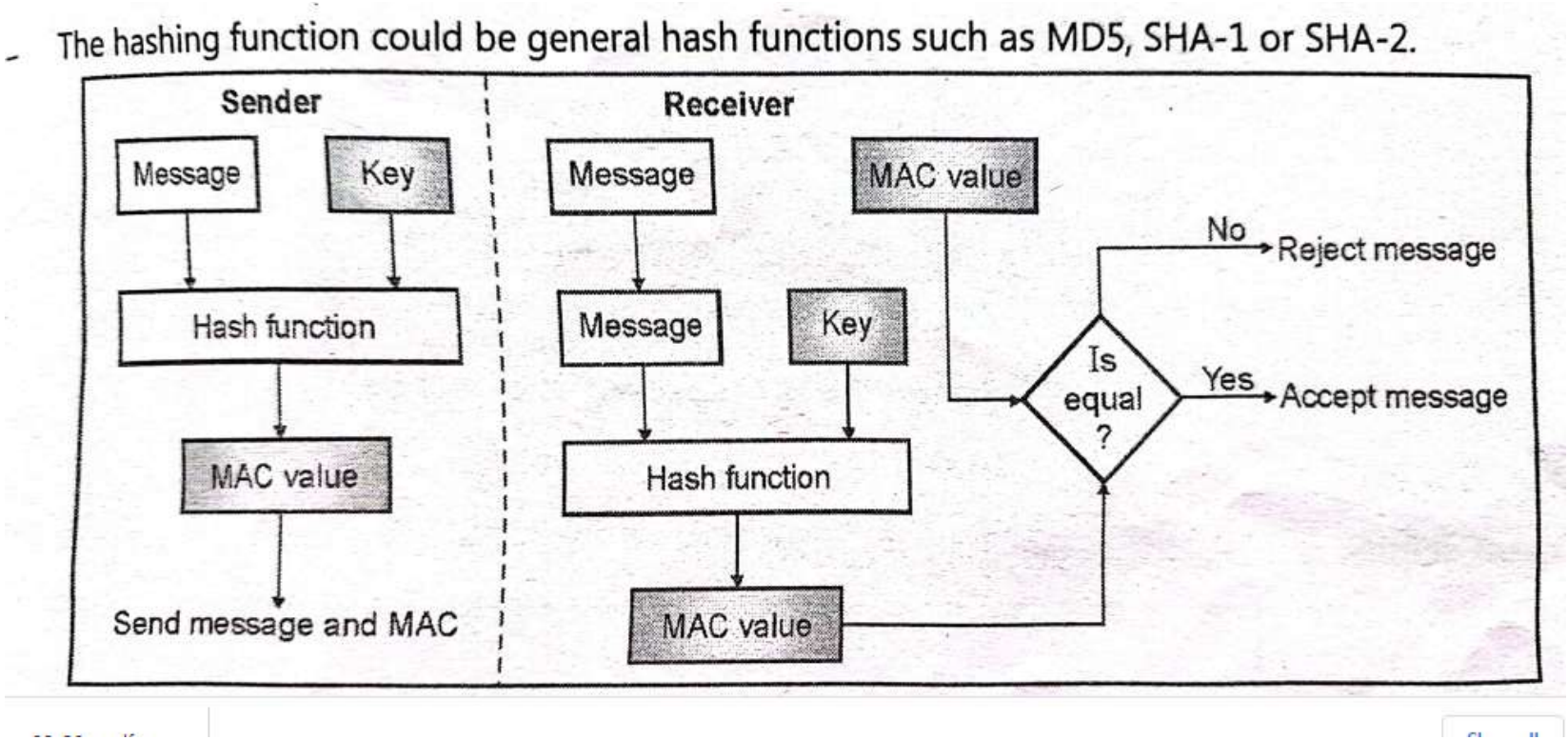
- The sponge construction is a framework for specifying function on binary data. It has 3 component :
- function f that works on fixed length strings
- Parameter r that determine rate of operation
- A padding rule denoted by pad
- The sponge function can be denoted as SPONGE (f, pad, r)
- A sponge function takes 2 inputs:
- a bit string that is denoted by N
- The bit length that is denoted by d
- In the sponge function, absorb phase involves:
- XORing of message block into a subset of state
- Using the permutation function for transferring the whole state

SHA-3

- In the sponge function, squeeze phase involves:
- Reading output blocks from the same subsets of state
- Replacing with state transformation function

HMAC

- In HMAC , a hashing function is used as a MAC function to calculate the MAC value

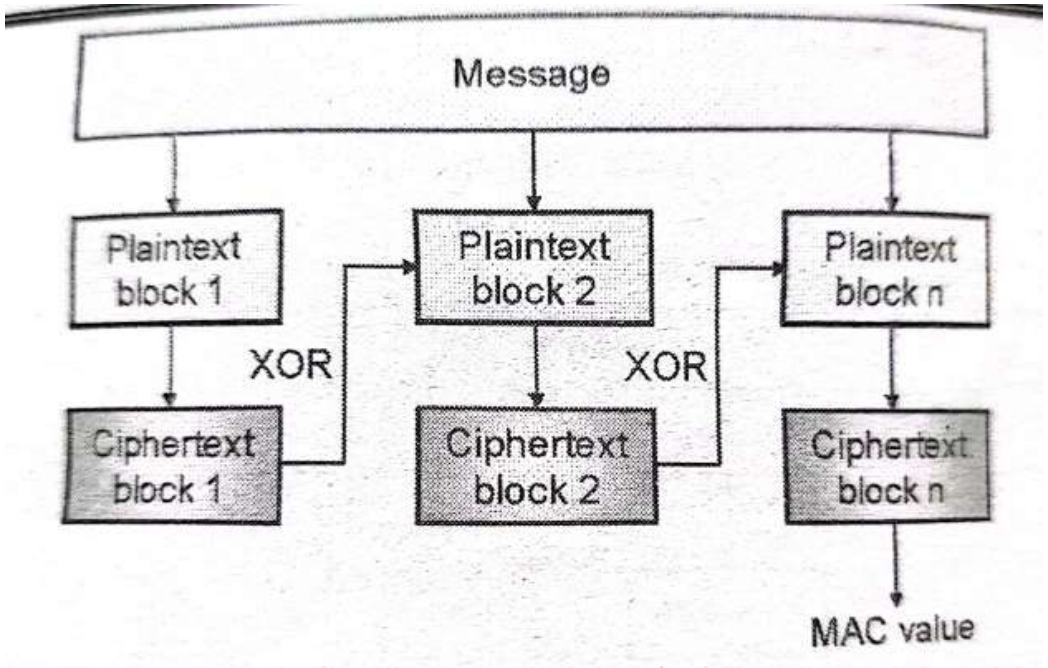


HMAC

- In HMAC , a hashing function is used as a MAC function to calculate the MAC value
- The hashing function could be general hash function such as MDS,SHA-1 or SHA-2
- On the sender side the message and the pre shared key I passed together into a hash function to compute a MAC
- The computed MAC along with the original message is send to the receiver the key is not send
- The key just provide a random value.
- On the receiver side the message is again passed through the same hashing function using the same pre shared key.
- A MAC value is computed at the receiver side and is matched with the MAC value that came from the sender
- If the 2 MAC value matched the message is accepted else the message is rejected

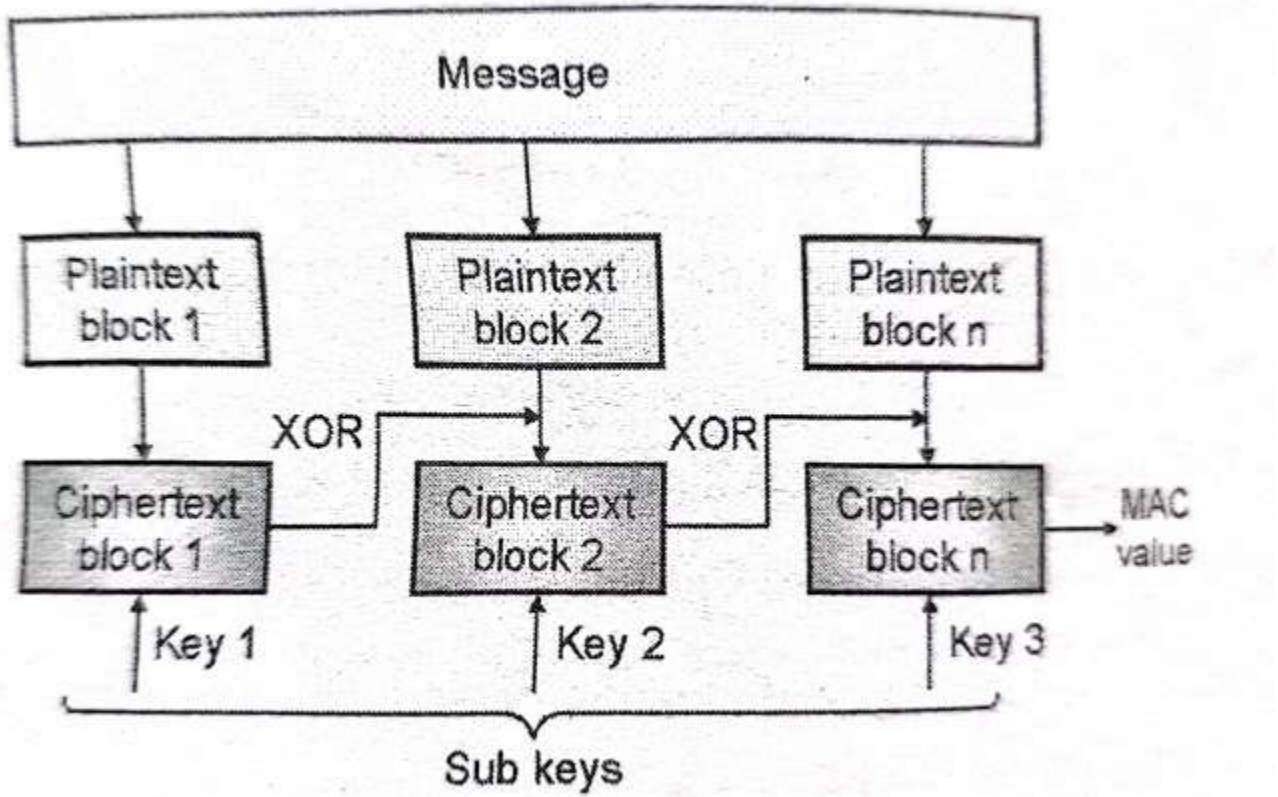
HMAC

- CBC-MAC
- In CBC-MAC asymmetric block cipher encryption function in the CBC mode is used as the MAC function to calculate MAC value



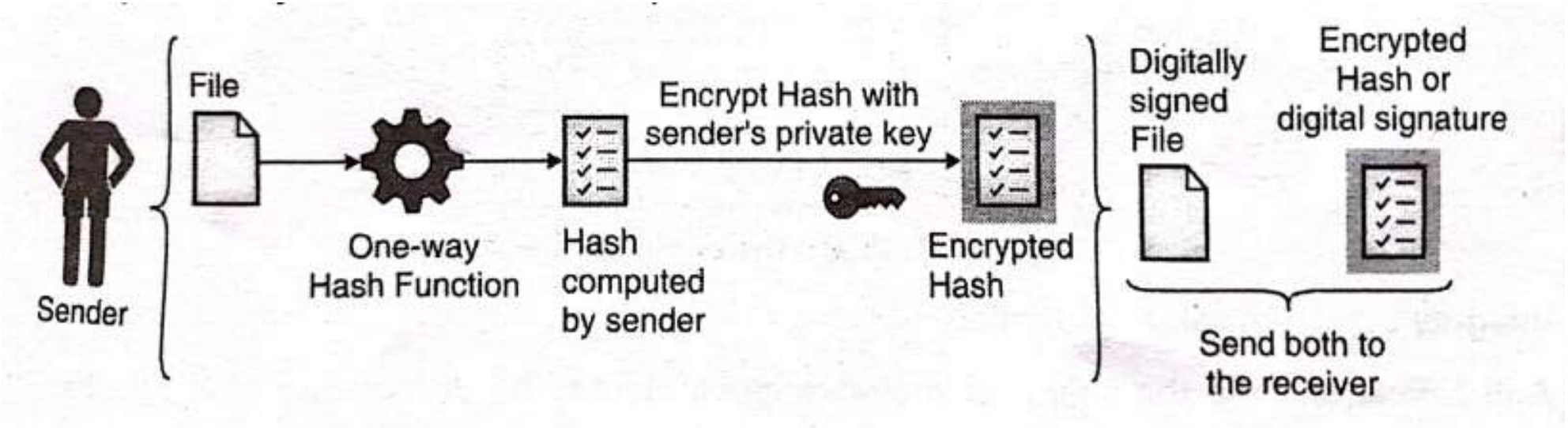
HMAC

- CMAC
- Cipher based MAC is very similar to CBC-MAC
- In CMAC, a symmetric block cipher encryption function is used as the MAC function to calculate the MAC values



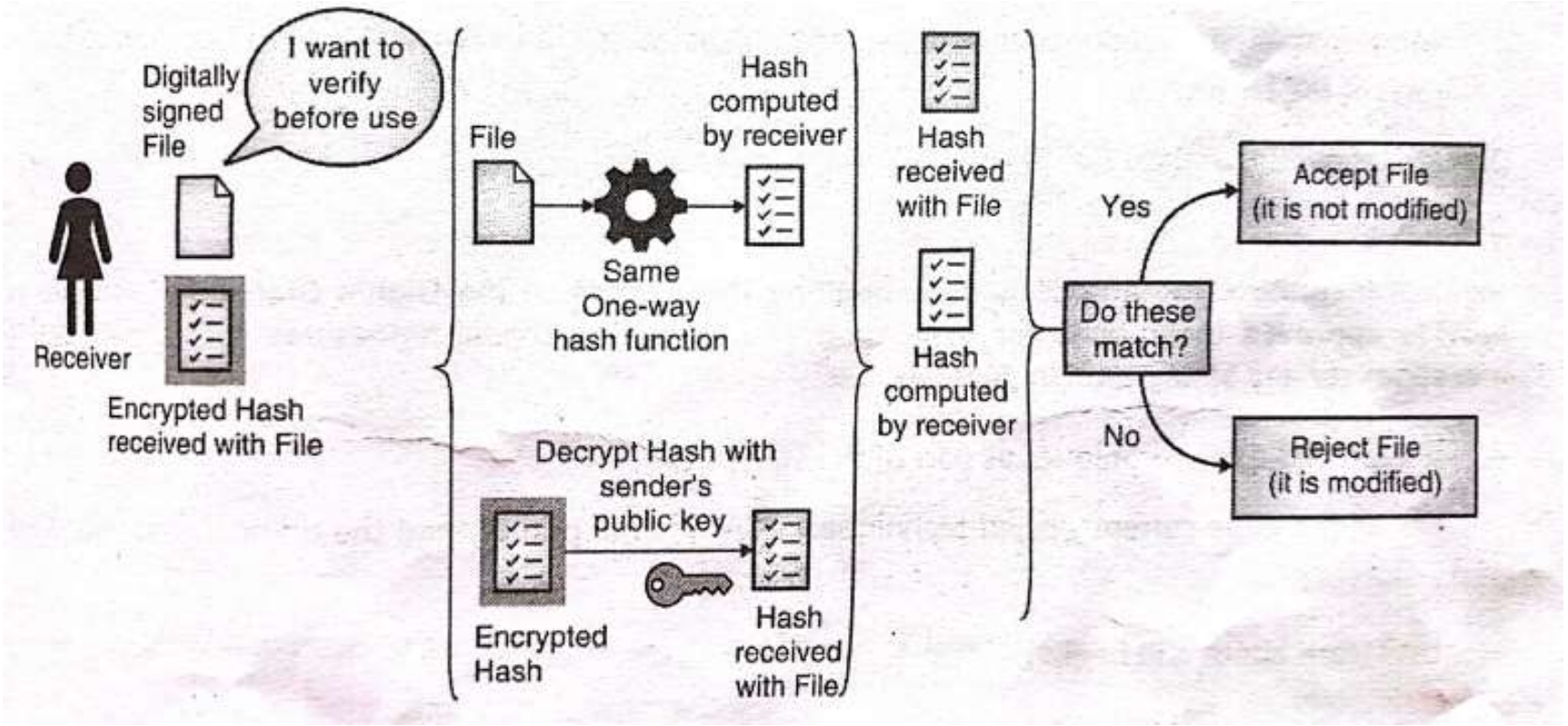
Digital signature

- A digital signature is a hash value that has been encrypted with the sender's private key
- Computation of digital signature at the sender's side



Digital signature

- Verification of digital signature at receiver side

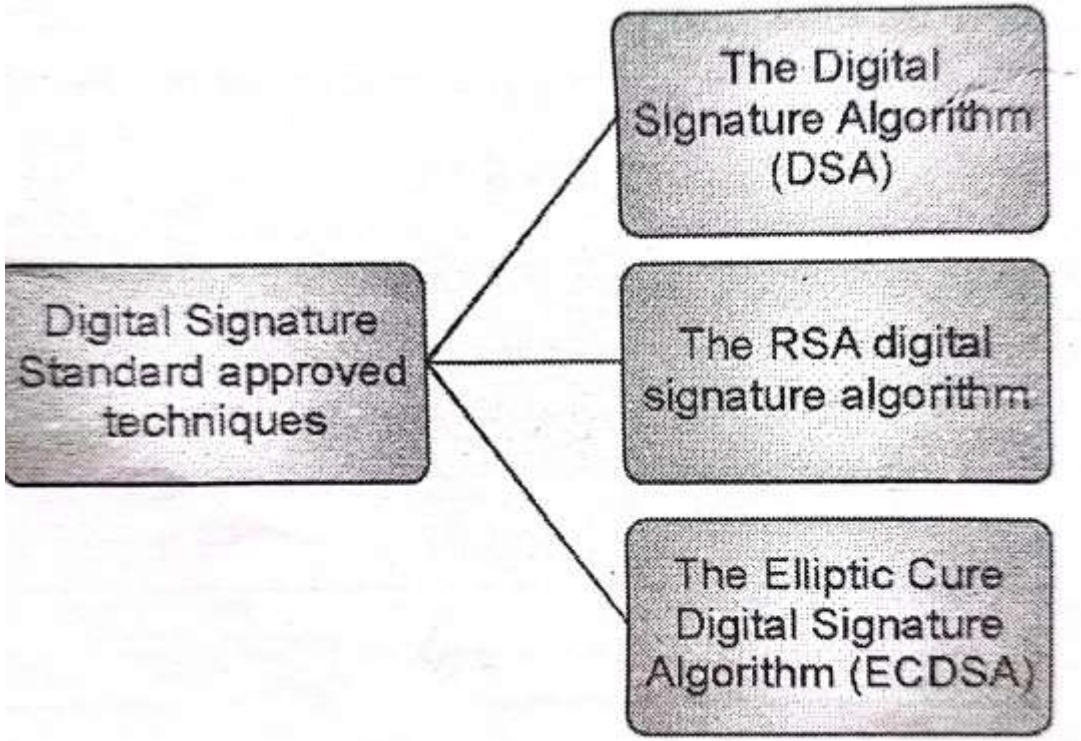


Digital signature

- Application and use of digital signature:
- Sending and receiving secure emails
- Signing documents
- Sending and receiving important files
- Properties of digital signature:
- Integrity
- Authentication
- Non repudiation

Digital signature

- Digital signature standards(DSS)
- National institute of standard and technology defines the digital signature standard to provide approved technics for generating and validating digital signature for authenticating messages



Digital signature algorithm(DSA)

- Digital signature algorithm is based on the difficulties of computing the discrete algorithm
- It is based on EIGamal and Schnorr digital signature schemes
- Key generation:
- p is a prime number
- q is a prime divisor of (p-1)
- g=$h^{(p-1)q}$mod p where h is any integer with1 <h<(p-1)
- x is user's private key
- y is user's public key

Digital signature algorithm(DSA)

- Message signing :
- M is message to be signed
- H (M) =SHA1(M)
- r=(g^kmod p)mod q
- S=[K^{-1}(H(M)+xr)]mod q
- Signature =(r,s)

Digital signature algorithm(DSA)

Signature verification:

Assume M',r',s' are as received at the receiver's end

w=$(s)^{-1}$mod q

u1=[H(M')w]mod q

u2=(r')w mod q

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

v should match r'

Comparison between hash,MAC and digital signature

Table 5.13.1

Comparison Attribute	Hash	MAC	Digital Signature
Integrity	Yes	Yes	Yes
Authentication	No	Yes	Yes
Non-repudiation	No	No	Yes
Keys Used	None	Symmetric Keys	Asymmetric Keys

Thank You!

(vandanas@sies.edu.in)