CLASS NOTES
E-COMM & E-SECURITY
TEACHING PHASE -II

RSA

Used for 3 purpose:

i) Key generation

ii) Key exchange

iii) Encryption/ decryption

(i) RSA for key generation

1) Take 2 prime numbers p and q: p=7, q=11
2) Compute n = p x q = 11 x 7 = 77
3) Compute ø(n)=(p-1)x(q-1) = (7-1)x(11-1) = 60
4) Select e such that it is relatively prime to ø(n) and 1<e< ø; gcd(e, ø(n)) =1

   60 = 2 x 3 x 3 x 5 → it is not present in table of 2,3 or 5

   e = 13
5) Find **d** (d is private)

   d x e = 1 mod ø ....(1)

   d = e^-1(1 mod ø)

   Condition for d:

   If d> ø , then d = d mod ø

   If d is –ve , then d = d+ ø

➔ compute d using Extended Euclid's algorithm

The equation is ax + by = gcd(a,b)................(2)

Where a = ø , b= e

60x + 13y = gcd(60,13) = 1

row	a_i	b_i	d	k
1	1	0	60 (ø)	-
2	0	1	13(e public)	(d1/d2) 60/13 = 4
3	1	-4	8	1
4	-1	5	5	1
5	2	-9	3	1
6	-3	14	2	1
7	5	**-23 (private key)**	1	1

a3 = a1 – a2 x k2 = 1 – 0 x 4 = 1

b3 = b1 – b2 x k2 = 0 – 1 x 4 = -4

d3 = d1 – d2 x k2 = 60 – 13 x 4 = 8

k3 = d2 / d3 = 13 / 8 = 1

a4 = a2- a3*k3 =

b4= b2-b3*k3 =

d4 = d2- d3 *k3=

k4= d3/d4 =

Solve till d = 1 → corresponding value of b is private key “d”

D= -23

E = 13

N = 77

Ø = 60

Condition for d:

If d> ø , then d = d mod ø

If d is –ve , then d = d+ ø

d = -23 = -23+60 = 37

decryption key d= 37

Public key : {e,n} : { 13, 77} → published

Private key: {d,n} : {37, 77} → secret

ii) RSA Encryption/ Decryption

public key: 13(e), 77(n)

private key: 37(d) , 77(n)

plain text **M: 8**

cipher text C = M ^ e (mod n)

C = 8^13 mod 77 ➔

Decryption:

M = C^d (mod n) ➔ C ^ 37 mod 77 ➔

Public key , msg → private key , C

Q: Seema publishes her RSA public key {37, 77}

Janki wants to send a msg M=20 to seema using RSA algo.

What cipher text does Janki sends to Seema?

And how seema will retrieve the plain text from the received cipher text?

Janki : → refer to the directory where public keys are stored → {37,77}, {e, n}

1) C = M^e mod n → 20^37 mod 77

Find → value of d

2) = C^d mod n →

Q: Perform encryption and decryption using RSA algorithm for the following:

P=3, q= 11, e=7, M=5

Find d (private key) and C (cipher text)

Fast Modular Exponentiation Algorithm

- → **Algorithm to compute a^b mod n**

C← 0; d← 1

For I ← k down to 0

do c ← 2*c

d ← (d x d) mod n // 160 * 160 mod 561 = 355

if b_i = 1 // true

then c← c+1 //

d ← (d x a) mod n // (355 * 7) mod 561

return d

Ex: a=7, b=560, n = 561

Compute: 7^{560} mod 561 = 1

1) **Compute binary of b = 560**

→ 1000 110000

i	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
b_i	**1**	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**0**
C	**1**	**2**	**4**	**8**	**17**	**35**	**70**	**140**	**280**	**560**
d	**7**	**49**	**157**	**526**	**160**	**241**	**298**	**166**	**67**	**1**

Ex: Solve the following using Fast Modular Exponentiation Algorithm **5^{117} mod 19.**

A = 5 , b = 117 ,n=19

Ex: Solve the following using Fast Modular Exponentiation Algorithm **7^{234} mod 37.**

<u>Diffie – Hellman Key exchange algorithm</u>

Two global element

q → prime number

a → a < q , primitive root of q

a^0 mod q

a^1 mod q

a^2 mod q ……..

A	B
1) Key generation Select a private key Xa Xa<q 2) Calculate its public key Ya→ a^Xa mod q Xa-private,Ya-pulic	1) Select private key Xb ; Xb < q 2) Caculate its public info Yb→a^Xb mod q Xb-private, Yb-public

Exchange public information: A →B; Ya	B →A; Yb
3) Yb	3) Ya
Calculate Session key independently; K = (Yb)^Xa mod q	Calculate Session key independently; K = (Ya)^Xb mod q
q=353 , a(alpha) = 3 Xa=97 Ya=? 3^97 mod 353 yb K=? K = (Yb)^Xa mod q	q=353 , a(alpha) = 3 Xb=233 Yb=? 3^233 mod 353 ya K=? K = (Ya)^Xb mod q

1) User A & B uses the D-H key exchange algorithm with common prime q=71, primitive root alpha = 7

i) If A's private key Xa=5, find public key Ya?

ii) If B's private key Xb=12, find Yb ?

iii) What is shared secret key K ?

2) Q=11, a=2

i) Ya=9; Xa=?

ii) Yb=3; Xb=?

iii) K=?

Message Autentication and Hash

1) M→ M + mac(M) -------------------------→ M → Fk→mac(M)
   | | |
   Fk → mac(M)   mac(M) -----→ Same (Security)

2) Msg auth & Conf : auth to plain text
   M→ M + mac(M) → Encryption(k2) -------------→ Decrypt(k2) → M+ Mac(M) -→ M—(fk1)--- Mac(M) == received mac

3) Msg auth & Conf: auth tied to CT
   1) M → E(k2)---------------------CT ------------→
               | +
            Fk1 -------------------mac

Msg : late

H(Msg) → x (hash value)

Msg2 : lete

H(Msg2) → x1(hash value)

Weak collision: Msg != Msg2 , hash value : x != x1

Any of the hash func does not follow this property, then attack can be possible.

How?

A: sender

h= x, M= late   C(attacker) (lite + Ek(x)

M + H(M) → late + Ek(x) -------------------------→   (lite + Ek(x)

Sender A   Rec B

Strong collision: any pair of msg (x,y)

M1: hi how are you ;

M2: how you are hi

Msg Digest Algo

MD5 algorithm

- Takes variable length input, & produces 128 bit output
- Variable length input → blocks → size 512 bit blocks

Step1: Append padding bits

- Msg is padded so that its length is congruent to 448 mod 512
- B1 B2 Blst(auth) → md5 last 128 bit – auth, sha1 last 160 auth
- 101010 10(padding) 0000 0110
- 64 bits are reserved for length of data

- Data + padding +length of msg = L * 512

- Padding bit sare always added.

Padding bits range → 1 to 512 bits

100000000000000.....

Step 2: append legth

64 bit representation of original msg

If original msg is > 2^64 → low order 64 bit

Step 3: initialize MD buffer

- Buffer 128 bit
- Used to hold intermediate and final result of hash func.
- 4 buffers each of 32 bits registers

A, B, C, D

A= 67452301

B= EFCDAB89

C=98BADCFE

D=10325476

LITTLE ENDIAN STORAGE ORDER IS USED BY THE MD5 BUFFER

Step 4: Process msg in 512 bits / 16 words block

- Heart of the algo is compression function , that consist of 4 rounds of processing
- This module is referred as MD5
- Rounds have similar structure but they are diff in logical func
- Lets refer the func as F,G,H,I
- T is table consist of 64 elements , table is constructed from sine function.
- The ith value of T is integer part of 2^32 x abs(sin(i))

  i is given in radians

Table provides radomized set of 32 bits pattern

Step5: all the blocks are processed → 128 bit msg digest/hash value/hashcodes

Summary:

CVq+1 = SUM32(CVq(Yq, I(Yq,H(Yq,G(Yq,F(Yq, CVq)))))

Blocks : 0 – L-1

MD = CV(L-1) : 128 bit

DATA || MD --------------------------------------→

Detail of Compression function:

HMAC

3) **Si = (k+) XOR (ipad)**
4) **ipad = 0011 0110 → b/8 times where b is block length**
5) **main Key K → 0's appeneded to the left of key K → derived key k+**
6) **key legth == block length (b)**
7) **opad = 0101 1100 → b/8 times**
8) **So = opad XOR k+**
9) **HMACk(M) → hash value/hash code/ msg digest**

ARBITRATED DIGITAL SIGNATURE

1) **Conventional encryption , arbiter sees the msg**

Step 1: X → A : M || Exa [IDx || H(M)]

Step 2: A → Y: Eay [IDx || M || Exa [IDx || H(M) || T]]

2) **Conventional encryption , arbiter cannot see the msg**

Step 1: X → A: Exy[M] || IDx || Exa [IDx || H (Exy(M))]

Step 2: A → Y: Eay [Exy[M] || IDx || Exa [IDx || H (Exy(M)) || T]]

3) **Public-encryption , arbiter does not see the msg**

Step 1: X → A: IDx || Eprx [IDx || Kpuy [Kprx[M]]]

Step 2: A→Y: Epra [IDx || Kpuy [Kprx[M]] || T]