Unit 3: Public Key Cryptography

1) What are three broad categories of applications of public-key cryptosystems?

ANS:

2) What requirements must a public key cryptosystems fulfill to be a secure algorithm?

ANS:

1. It is computationally easy for a party B to generate a pair (public key Pub, private key PRb).

2. It is computationally easy for a sender A, knowing the public key and the message to be encrypted, M, to generate the corresponding ciphertext:
   C = E(PUb,M)

3. It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message:
M = D(PRb, C) = D[PRb, E(PUb,M)]

4. It is computationally infeasible for an adversary, knowing the public key, PUb, to determine the private key, PRb.

5. It is computationally infeasible for an adversary, knowing the public key, PUb, and a ciphertext, C, to recover the original message, M.

6. The two keys can be applied in either order:
M = D[PUb, E(PRb,M)] = D[PRb, E(PUb,M)]

3) Explain Rivest-Shamir-Adleman (RSA) algorithm.

ANS:

RSA Algorithm

- Plaintext is encrypted in blocks
- Block's binary value $2^k < n$
- Equivalently, block size $k < \log2(n)$
- Encryption (plaintext block M, ciphertext C):
  $C = M^e \bmod n$
- Decryption:
  $M = C^d \bmod n = (M^e)d \bmod n = M^{ed} \bmod n$

Assumptions

Sender and receiver know n

- Sender knows e
- Receiver only knows d
- Public key KU = {e, n}
- Private key KR = {d, n}

RSA Key Setup

- each user generates a public/private key pair by:
- selecting two large primes at random - p, q
- computing their system modulus N=p.q
  --note ø(N)=(p-1)(q-1)
- selecting at random the encryption key e
  -where 1<e<ø(N), gcd(e,ø(N))=1
- solve following equation to find decryption key d
  -e.d=1 mod ø(N) and 0≤d≤N
- publish their public encryption key: KU={e,N}
- keep secret private decryption key: KR={d,p,q}

RSA Use

- to encrypt a message M the sender:

-obtains public key of recipient KU={e,N}
-computes: $C=M^e \bmod N$, where $0 \leq M < N$

- to decrypt the ciphertext C the owner:
  -uses their private key KR={d,p,q}
  -computes: $M=C^d \bmod N$
- note that the message M must be smaller than the modulus N (block if needed)

Requirements and Conditions

- It is required to find e, d, n so that
  $M^{ed} \bmod n = M$
- Equivalently, $M^{ed} \equiv M \bmod n$ $(M < n)$
- From the corollary of Euler's theorem $m^{k\varphi(n)+1} = m^{k(p-1)(q-1)+1} \equiv m \pmod{n}$
- Conditions
  -p, q are prime numbers, $n = pq$
  -$0 < m < n$
- $ed = k\,\varphi(n) + 1$

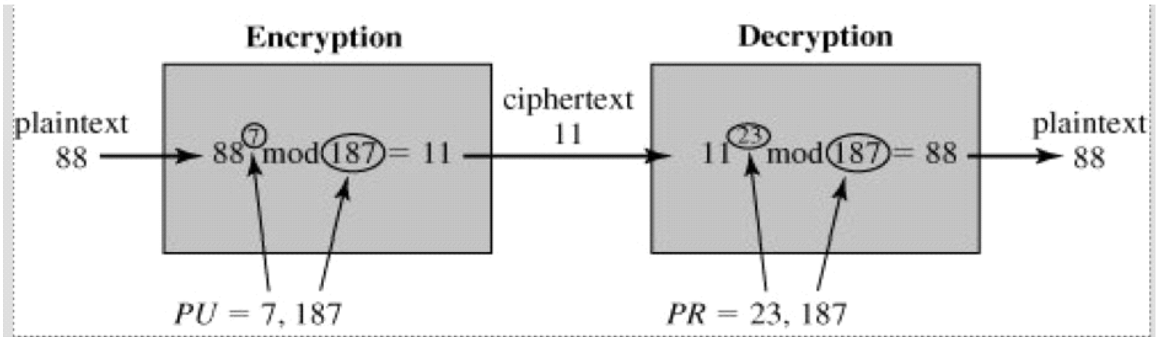
Mathematical Justification

- $ed = k\,\varphi(n) + 1$
- $ed \bmod \varphi(n) = k\,\varphi(n) \bmod \varphi(n) + 1 \bmod \varphi(n)$
- $ed \equiv 1 \bmod \varphi(n)$
- $d \equiv e^{-1} \bmod \varphi(n)$
- e, d are multiplicative inverses mod $\varphi(n)$
- Condition
  -d (and thus e) is relatively prime to $\varphi(n)$
  -Equivalently: $\gcd(\varphi(n), d) = 1$

Example

- Select $p = 17$, $q = 11$

- Calculate $n = pq = 17 \times 11 = 187$
- Calculate $\varphi(n) = (p - 1)(q - 1) = 160$
- Select $e < 160$, relatively prime to 160: $e = 7$
- Calculate $d < 160$, $de \equiv 1 \bmod 160$
  -repeat $d = (k\,\varphi(n) + 1) / e$
  -increment k until you get an integer value
  -$1 \times 160 + 1 = 161$; $d = 161/7 = 23$
- $KU = \{7, 187\}$, $KR = \{23, 187\}$
- Let $M = 88$



- $88^7 \bmod 187 = [(88^4 \bmod 187) \times (88^2 \bmod 187) \times (88^1 \bmod 187)] \bmod 187$
- $88^1 \bmod 187 = 88$
- $88^2 \bmod 187 = 7744 \bmod 187 = 77$
- $88^4 \bmod 187 = 59{,}969{,}536 \bmod 187 = 132$
- $88^7 \bmod 187 = (88 \times 77 \times 132) \bmod 187 = 894{,}432 \bmod 187 = 11$
- $11^{23} \bmod 187 = [(11^1 \bmod 187) \times (11^2 \bmod 187) \times (11^4 \bmod 187) \times (11^8 \bmod 187) \times (11^8 \bmod 187)] \bmod 187$
- $11^1 \bmod 187 = 11$
- $11^2 \bmod 187 = 121$
- $11^4 \bmod 187 = 14{,}641 \bmod 187 = 55$
- $11^8 \bmod 187 = 214{,}358{,}881 \bmod 187 = 33$
- $11^{23} \bmod 187 = (11 \times 121 \times 55 \times 33 \times 33) \bmod 187 = 79{,}720{,}245 \bmod 187 = 88$

4) Explain security of Rivest-Shamir-Adleman (RSA) algorithm.
ANS:

The Security of RSA
Four possible approaches to attacking the RSA algorithm are:

1. Brute force: This involves trying all possible private keys.
2. Mathematical attacks: There are several approaches, all equivalent in effort to factoring the product of two primes.
3. Timing attacks: These depend on the running time of the decryption algorithm.
4. Chosen ciphertext attacks: This type of attack exploits properties of the RSA algorithm.

5) What are four possible approaches to attack the Rivest-Shamir-Adleman (RSA)
algorithm?
ANS:

The Security of RSA
Four possible approaches to attacking the RSA algorithm are:

1. Brute force: This involves trying all possible private keys.
2. Mathematical attacks: There are several approaches, all equivalent in effort to factoring the product of two primes.
3. Timing attacks: These depend on the running time of the decryption algorithm.
4. Chosen ciphertext attacks: This type of attack exploits properties of the RSA algorithm.

6) Explain Optimal Asymmetric Encryption Padding (OAEP).

ANS:

- RSA algorithm is vulnerable to a Chosen Ciphertext Attack (CCA).
- To overcome this simple attack, practical RSA-based cryptosystems randomly pad the plaintext prior to encryption
- RSA Security Inc recommends modifying the plaintext using a procedure known as Optimal Asymmetric Encryption Padding (OAEP).

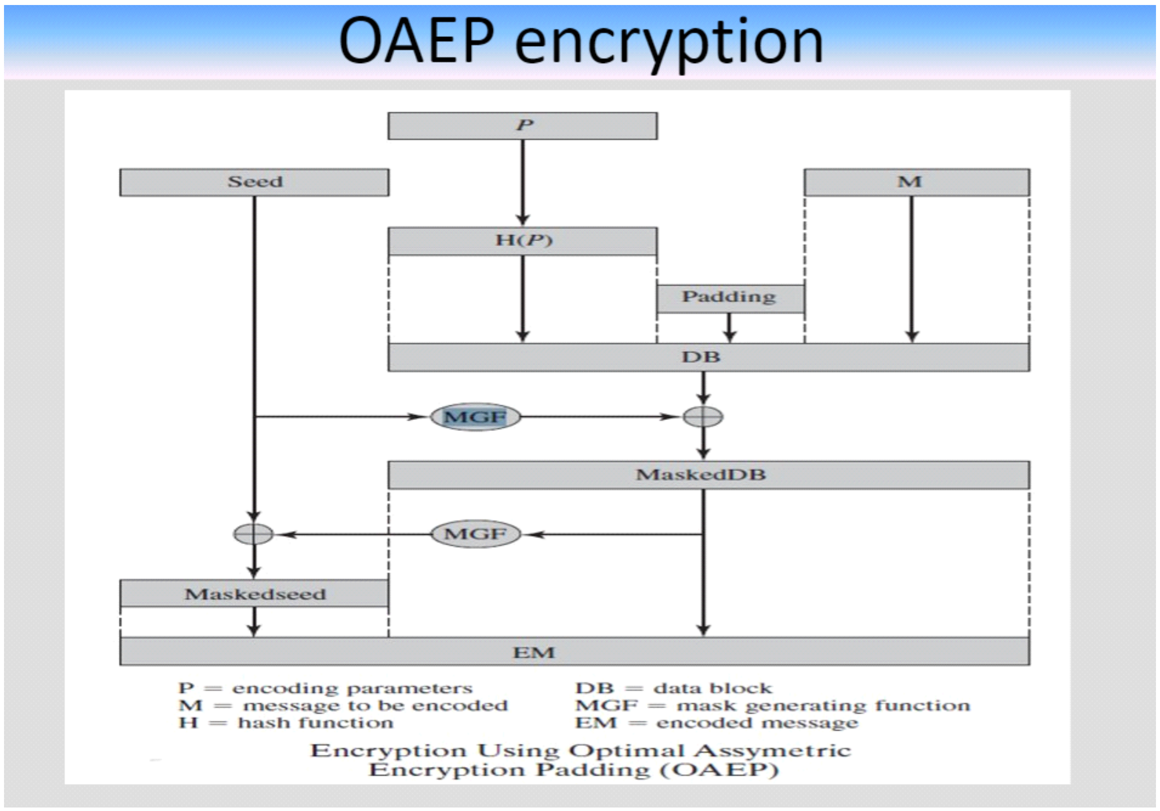


Figure depicts OAEP encryption. As a first step, the message M to be encrypted is padded.A set of optional parameters,P, is passed through a hash function,

H.8 The output is then padded with zeros to get the desired length in the overall data block (DB). Next, a random seed is generated and passed

through another hash func_tion, called the mask generating function (MGF).The resulting hash value is bit-by-bit
XORed with DB to produce a maskedDB. The maskedDB is in turn passed through the MGF to form a hash that is XORed with the seed to produce the masked seed.The concatenation of the maskedseed and the maskedDB forms the encoded message EM.
Note that the EM includes the padded message, masked by the seed, and the seed, masked by the maskedDB.The EM is then encrypted using RSA.

7) Explain Diffie-Hellaman key exchange algorithm.

ANS:

Diffie-Hellaman key exchange algorithm

- first public-key type scheme proposed
- by Diffie & Hellman in 1976 along with the exposition of public key concepts

  -note: now know that James Ellis (UK CESG) secretly proposed the concept in 1970
- is a practical method for public exchange of a secret key
- used in a number of commercial products
- a public-key distribution scheme

  -cannot be used to exchange an arbitrary message

  -rather it can establish a common key

  -known only to the two participants
- value of key depends on the participants (and their private and public key information)
- based on exponentiation in a finite field (modulo a prime or a polynomial) - easy
- security relies on the difficulty of computing discrete logarithms

(similar to factoring) – hard

Diffie-Hellman Setup

- Primitive Root: Primitive root of a prime number p as one whose powers generates all the integers from 1 to p-1.

If a is primitive root of the prime number p, then the numbers

$a \bmod p, a^2 \bmod p, \ldots\ldots\ldots, ap^{-1} \bmod p$

are distinct and consists of integers from 1 through p-1.

For any integer b and primitive root a of prime number p we can find unique exponent i such that

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

The exponent i is referred to as discrete logarithm, or index of b for base a, mod p

- all users agree on global parameters:
  - -large prime integer or polynomial q
  - -α a primitive root mod q
- each user (eg. A) generates their key
  - -chooses a secret key (number): $x_A < q$
  - -compute their public key: $y_A = \alpha^{xA} \bmod q$
- each user makes public that key y_A

Diffie-Hellman Key Exchange

- shared session key for users A & B is K_{AB}:

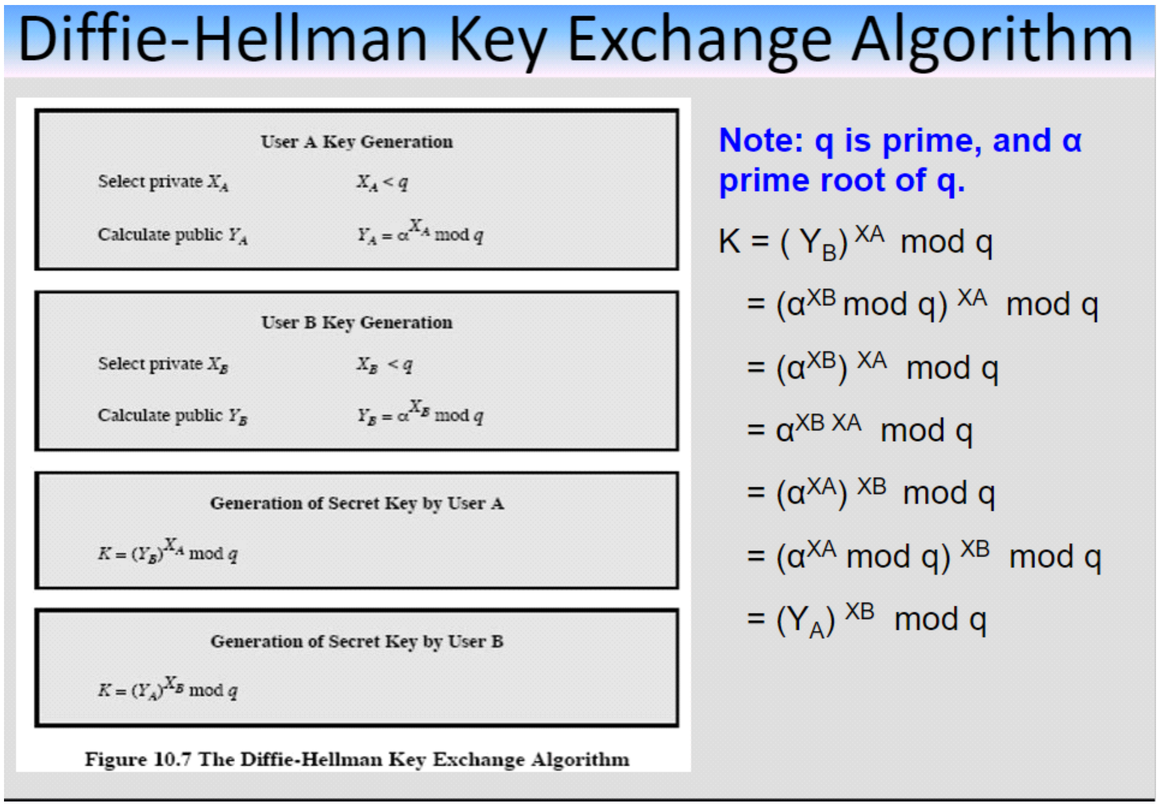
$K_{AB} = \alpha^{xA.xB} \bmod q$

$= y_A^{xB} \bmod q$ (which B can compute)

$= y_B^{xA} \bmod q$ (which A can compute)

- K_{AB} is used as session key in private-key encryption scheme between Alice and Bob

- if Alice and Bob subsequently communicate, they will have the same key as before, unless they choose new public-keys
- attacker needs an x, must solve discrete log



Diffie-Hellman Key Exchange Algorithm

User A Key Generation

Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = \alpha^{X_A} \bmod q$

User B Key Generation

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

Generation of Secret Key by User A

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

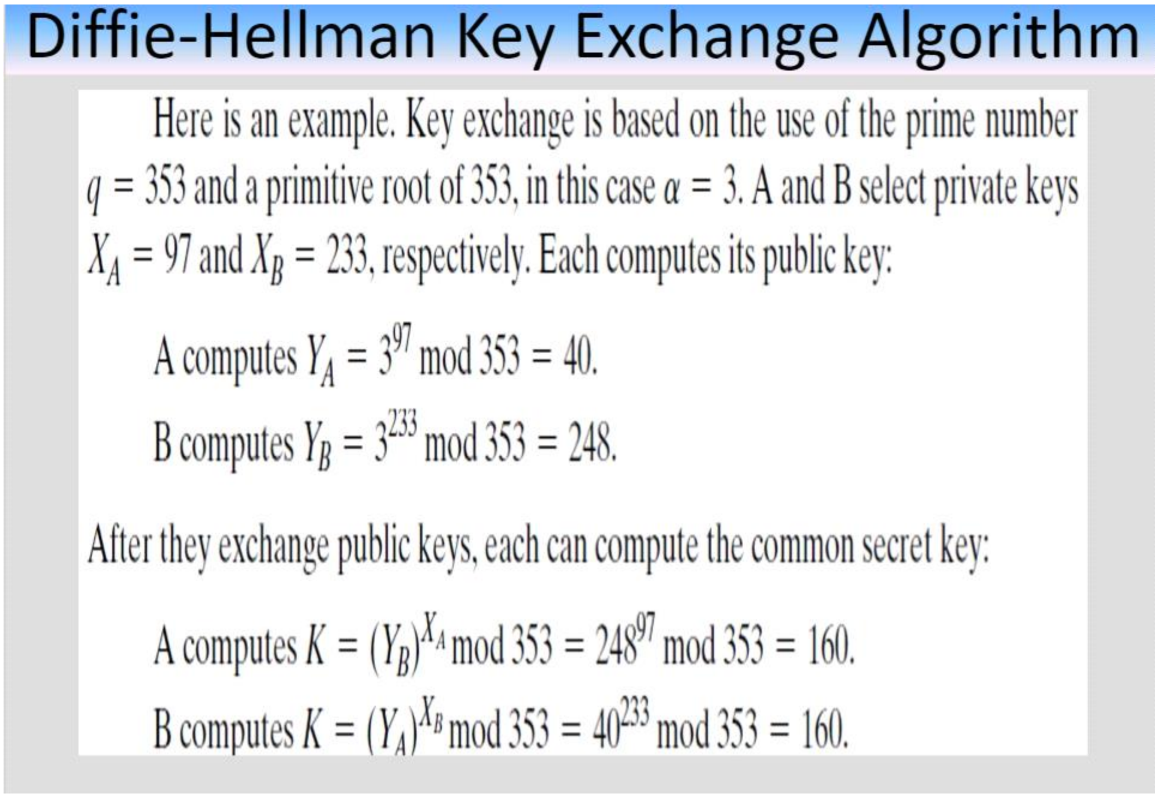
Generation of Secret Key by User B

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

Figure 10.7 The Diffie-Hellman Key Exchange Algorithm

Note: q is prime, and α prime root of q.

$$K = (Y_B)^{XA} \bmod q$$
$$= (\alpha^{XB} \bmod q)^{XA} \bmod q$$
$$= (\alpha^{XB})^{XA} \bmod q$$
$$= \alpha^{XB\,XA} \bmod q$$
$$= (\alpha^{XA})^{XB} \bmod q$$
$$= (\alpha^{XA} \bmod q)^{XB} \bmod q$$
$$= (Y_A)^{XB} \bmod q$$



Diffie-Hellman Key Exchange Algorithm

Here is an example. Key exchange is based on the use of the prime number $q = 353$ and a primitive root of 353, in this case $\alpha = 3$. A and B select private keys $X_A = 97$ and $X_B = 233$, respectively. Each computes its public key:

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

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

After they exchange public keys, each can compute the common secret key:

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

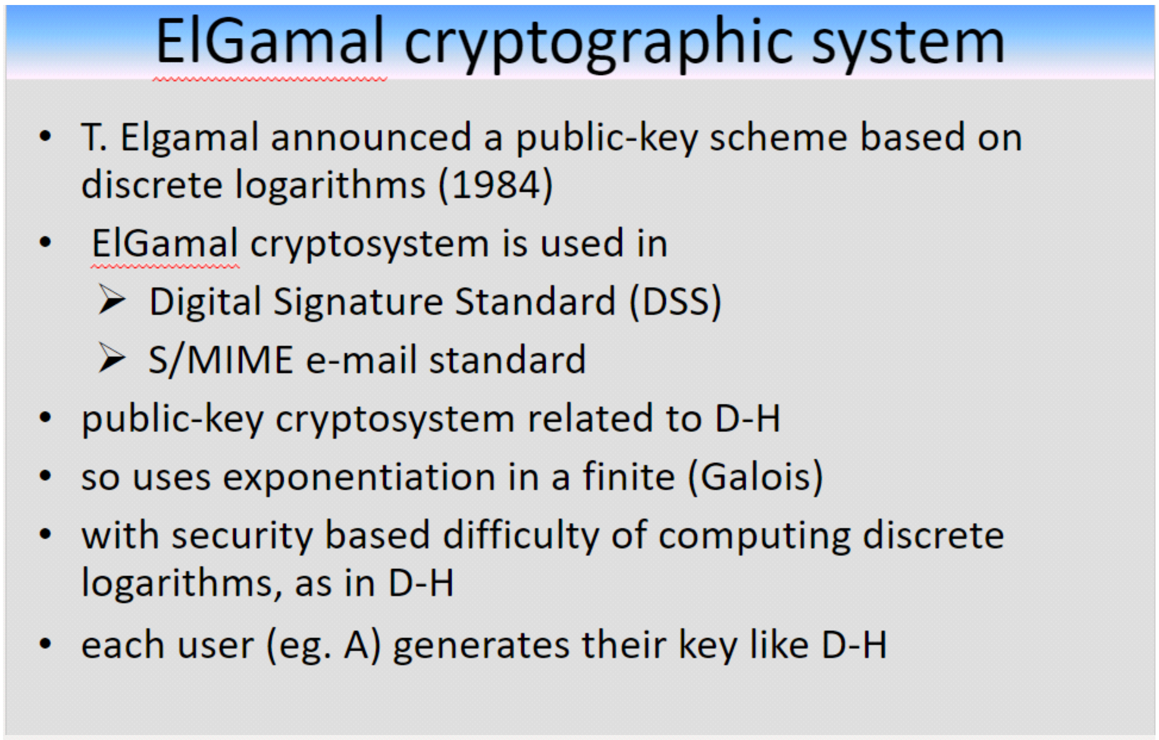
B computes $K = (Y_A)^{X_B} \bmod 353 = 40^{233} \bmod 353 = 160$.

8) Explain Man-in-the-Middle Attack against Diffie-Hellaman Key Exchange Protocol.

ANS:

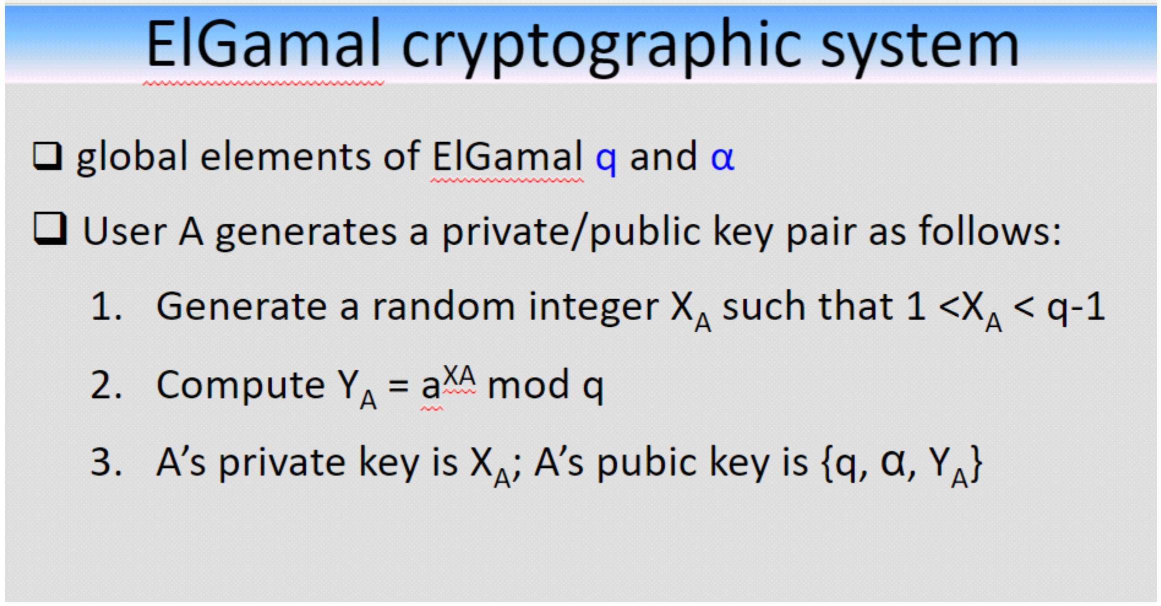
9) Explain Elgamal Cryptographic System.

ANS:



ElGamal cryptographic system

- T. Elgamal announced a public-key scheme based on discrete logarithms (1984)
- ElGamal cryptosystem is used in
  - ➢ Digital Signature Standard (DSS)
  - ➢ S/MIME e-mail standard
- public-key cryptosystem related to D-H
- so uses exponentiation in a finite (Galois)
- with security based difficulty of computing discrete logarithms, as in D-H
- each user (eg. A) generates their key like D-H

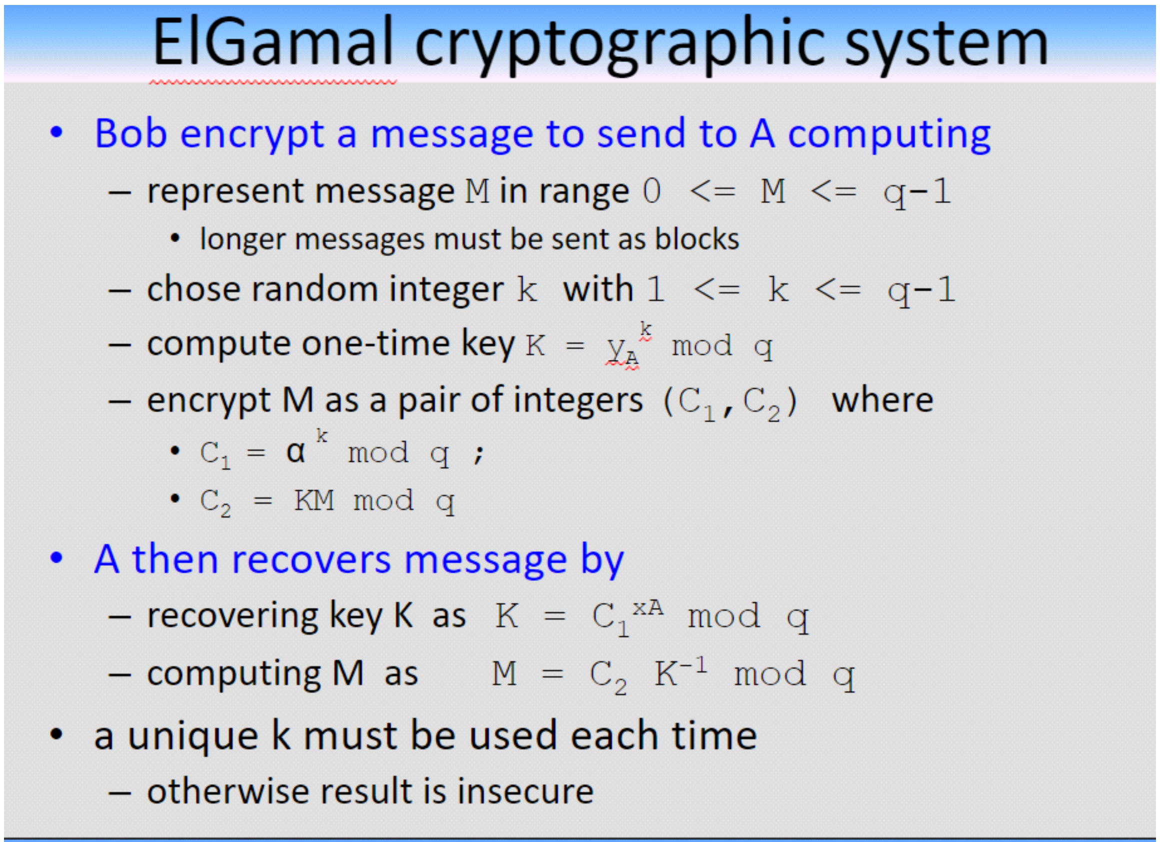


ElGamal cryptographic system

❑ global elements of ElGamal q and α

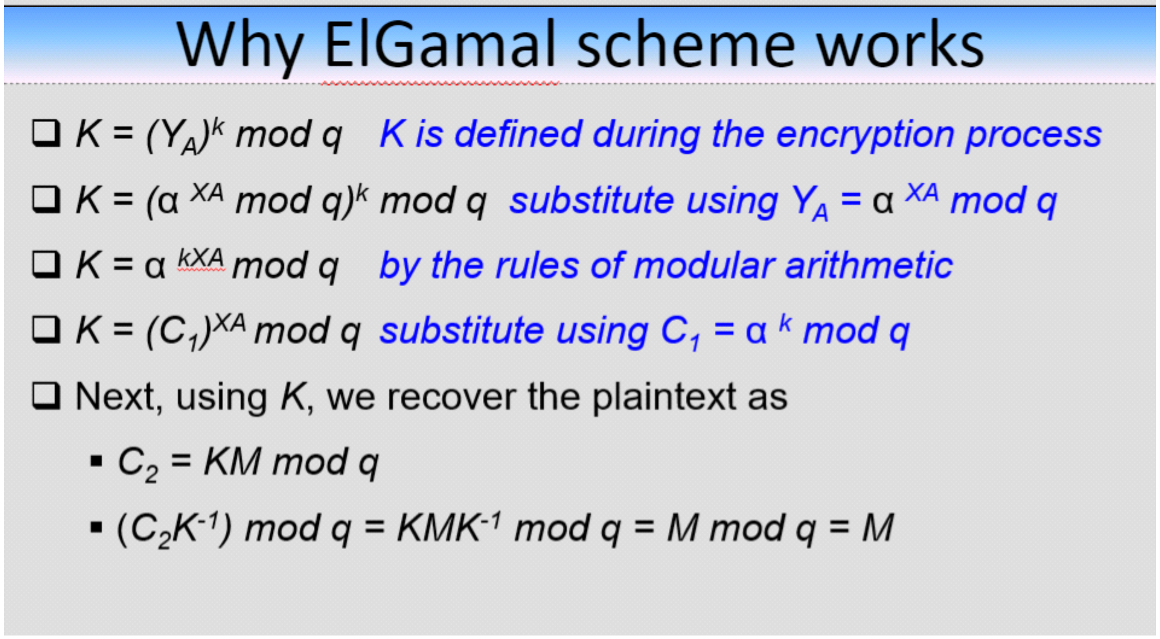
❑ User A generates a private/public key pair as follows:

1. Generate a random integer X_A such that $1 < X_A < q-1$
2. Compute $Y_A = a^{XA} \bmod q$
3. A's private key is X_A; A's pubic key is $\{q, \alpha, Y_A\}$



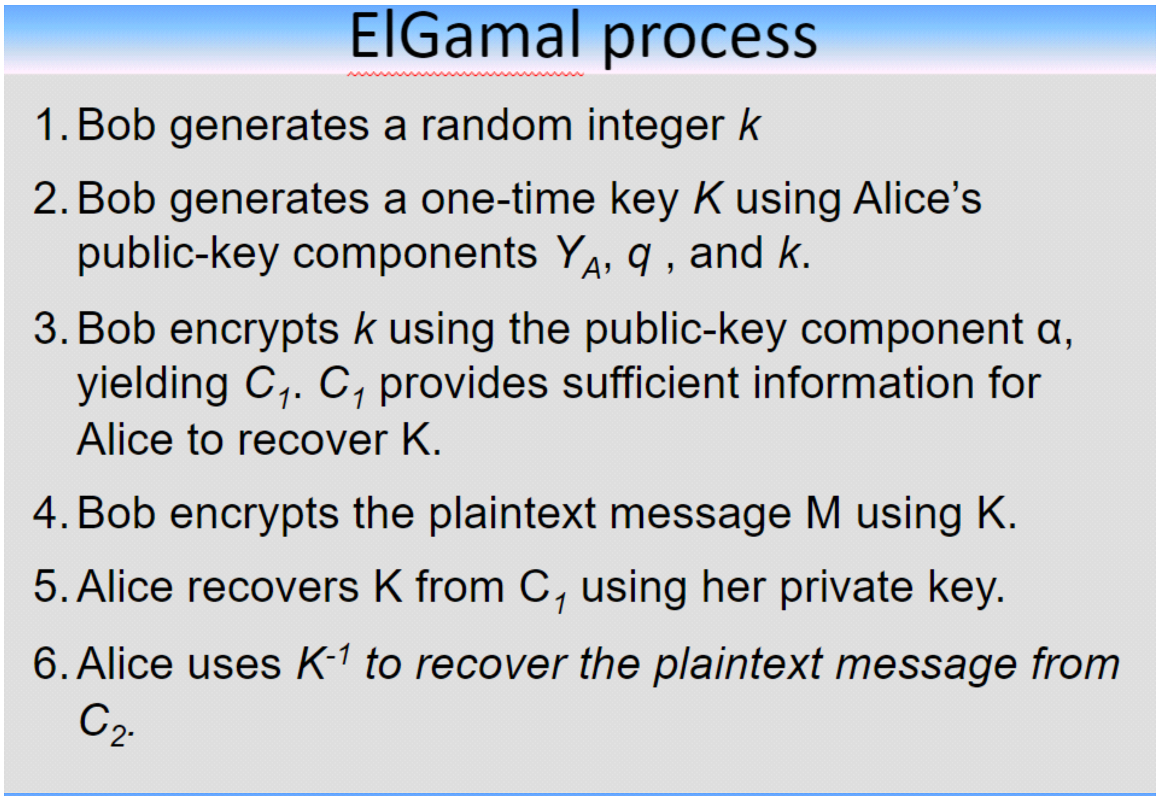
ElGamal cryptographic system

- Bob encrypt a message to send to A computing
  - represent message M in range $0 <= M <= q-1$
    - longer messages must be sent as blocks
  - chose random integer k with $1 <= k <= q-1$
  - compute one-time key $K = y_A^k \bmod q$
  - encrypt M as a pair of integers (C_1, C_2) where
    - $C_1 = \alpha^k \bmod q$;
    - $C_2 = KM \bmod q$
- A then recovers message by
  - recovering key K as $K = C_1^{xA} \bmod q$
  - computing M as $M = C_2 K^{-1} \bmod q$
- a unique k must be used each time
  - otherwise result is insecure



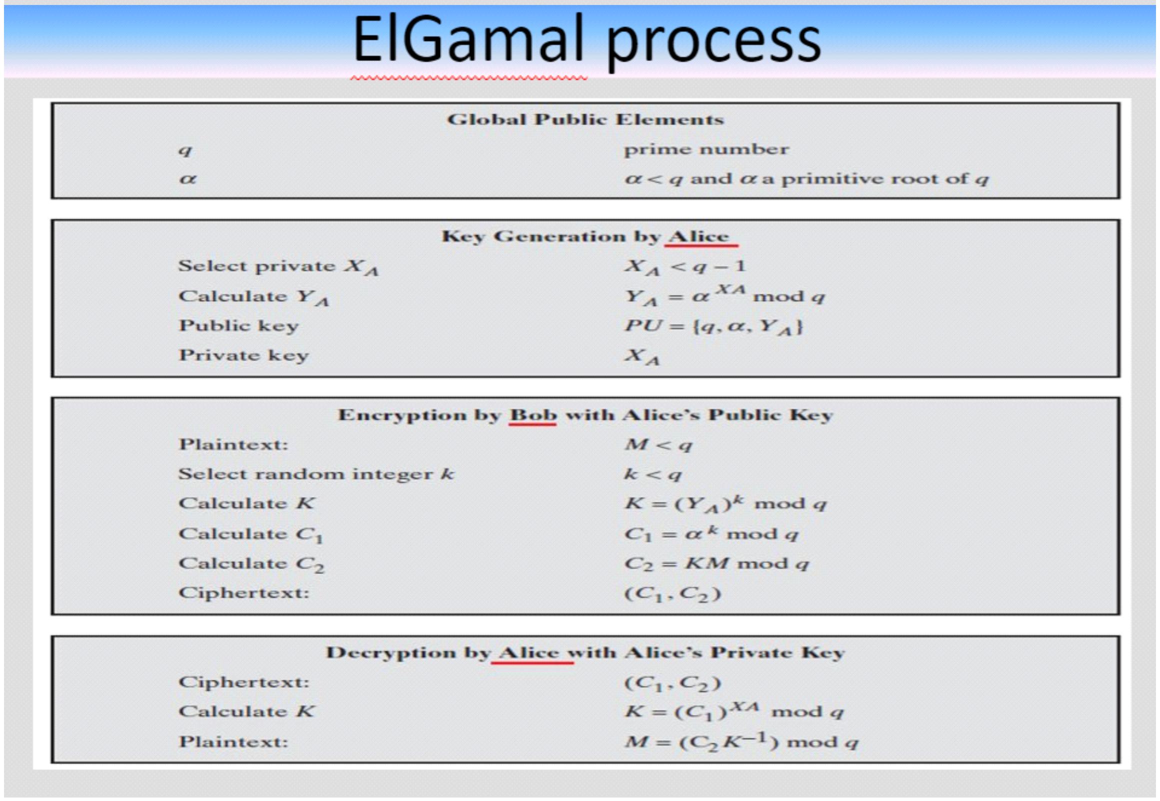
Why ElGamal scheme works

- ❑ $K = (Y_A)^k \bmod q$ *K is defined during the encryption process*
- ❑ $K = (\alpha^{XA} \bmod q)^k \bmod q$ *substitute using $Y_A = \alpha^{XA} \bmod q$*
- ❑ $K = \alpha^{kXA} \bmod q$ *by the rules of modular arithmetic*
- ❑ $K = (C_1)^{XA} \bmod q$ *substitute using $C_1 = \alpha^k \bmod q$*
- ❑ Next, using *K*, we recover the plaintext as
  - $C_2 = KM \bmod q$
  - $(C_2K^{-1}) \bmod q = KMK^{-1} \bmod q = M \bmod q = M$



ElGamal process

1. Bob generates a random integer *k*
2. Bob generates a one-time key *K* using Alice's public-key components Y_A, q , and k.
3. Bob encrypts *k* using the public-key component α, yielding C_1. C_1 provides sufficient information for Alice to recover K.
4. Bob encrypts the plaintext message M using K.
5. Alice recovers K from C_1 using her private key.
6. Alice uses K^{-1} *to recover the plaintext message from* C_2.



ElGamal process

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:	$M = (C_2 K^{-1}) \bmod q$

ElGamal cryptographic system Example

- use field GF(19) q=19 and α =10
- Alice computes her key:
  - A chooses $x_A=5$ & computes $y_A=10^5 \bmod 19 = 3$
- Bob send message **m=17** as **(11,5)** by
  - chosing random k=6
  - computing $K = y_A^k \bmod q = 3^6 \bmod 19 = \underline{7}$
  - computing $C_1 = \alpha^k \bmod q = 10^6 \bmod 19 =$ **11**;

    $C_2 = KM \bmod q = 7.17 \bmod 19 =$ **5**
- Alice recovers original message by computing:
  - recover $K = C_1^{xA} \bmod q = 11^5 \bmod 19 = \underline{7}$
  - compute inverse $K^{-1} = 7^{-1} = 11$
  - recover $M = C_2\ K^{-1} \bmod q = 5.11 \bmod 19 =$ **17**

ElGamal cryptographic system

❑ 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 of the message enables the user to compute other blocks as follows. Let

$$C_{1,1} = \alpha^k \bmod q;\ C_{2,1} = KM1 \bmod q$$
$$C_{1,2} = \alpha^k \bmod q;\ C_{2,2} = KM2 \bmod q$$

❑ Then,

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

❑ If is M1 known, then is easily computed as

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

Security of ElGamal is based on difficulty of computing discrete logarithms