Public Key Cryptography

Unit II

Public-Key Cryptography

- Uses **two** keys – One public & another private key
- **Asymmetric** algorithms because one key is used for Encryption and another key is used for Decryption
- **Public-key/two-keys/asymmetric** cryptography involves the use of **two** keys:
  - A **public-key**, which may be known by anybody, and can be used to **encrypt messages**, and **verify signatures**
  - A **private-key**, known only to the recipient, used to **decrypt messages**, and **sign** (create) **signatures**

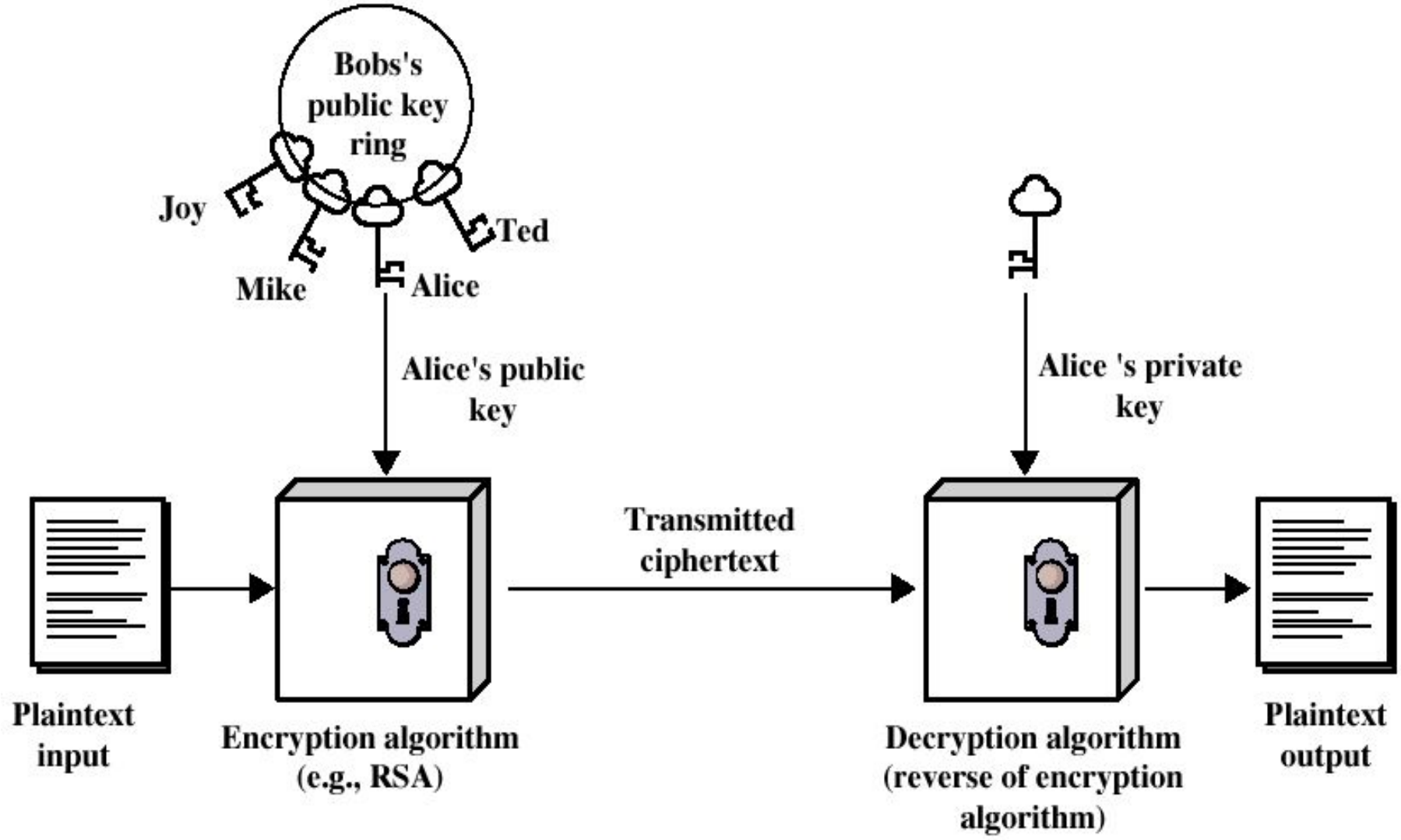
Ingredients of Public-Key Cryptography

- Plaintext
- Encryption Algorithm
- Public and Private Keys: If one is used for Encryption other is used for decryption
- Ciphertext
- Decryption Algorithm

Key Points of Public-Key Cryptography

- Asymmetric encryption is a form of cryptosystem in which encryption and decryption are performed using the different keys. one is a public key and one is a private key. It is also known as public-key encryption.
- Asymmetric encryption transforms plaintext into ciphertext using a one of two keys and an encryption algorithm. Using the paired key and a decryption algorithm, the plaintext is recovered from the ciphertext.
- Asymmetric encryption can be used for confidentiality, authentication, or both.
- The most widely used public-key cryptosystem is RSA. The difficulty of attacking RSA is based on the difficulty of finding the prime factors of a composite number.

Public-Key Cryptography



(a) Encryption

Steps for Public-Key Cryptography

- **Each user generates a pair of keys to be used for the encryption and decryption of messages.**
- Each user places one of the two keys in a public register or other accessible file. **This is the public key.**
- The companion key is kept private. Each user maintains a collection of public keys obtained from others.
- If Bob wishes to send a confidential message to Alice, **Bob encrypts the message using Alice's public key.**
- **When Alice receives the message, she decrypts it using her private key.** No other recipient can decrypt the message because only Alice knows Alice's private key.

Private-Key vs. Public-Key Encryption

Conventional Encryption	Public-Key Encryption
Needed to Work: 1. The same algorithm with the same key is used for encryption and decryption. 2. The sender and receiver must share the algorithm and the key. *Needed for Security:* 1. The key must be kept secret. 2. It must be impossible or at least impractical to decipher a message if no other information is available. 3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	*Needed to Work:* 1. One algorithm is used for encryption and decryption with a pair of keys, one for encryption and one for decryption. 2. The sender and receiver must each have one of the matched pair of keys (not the same one). *Needed for Security:* 1. One of the two keys must be kept secret. 2. It must be impossible or at least impractical to decipher a message if no other information is available. 3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

Applications of Public-Key Cryptography

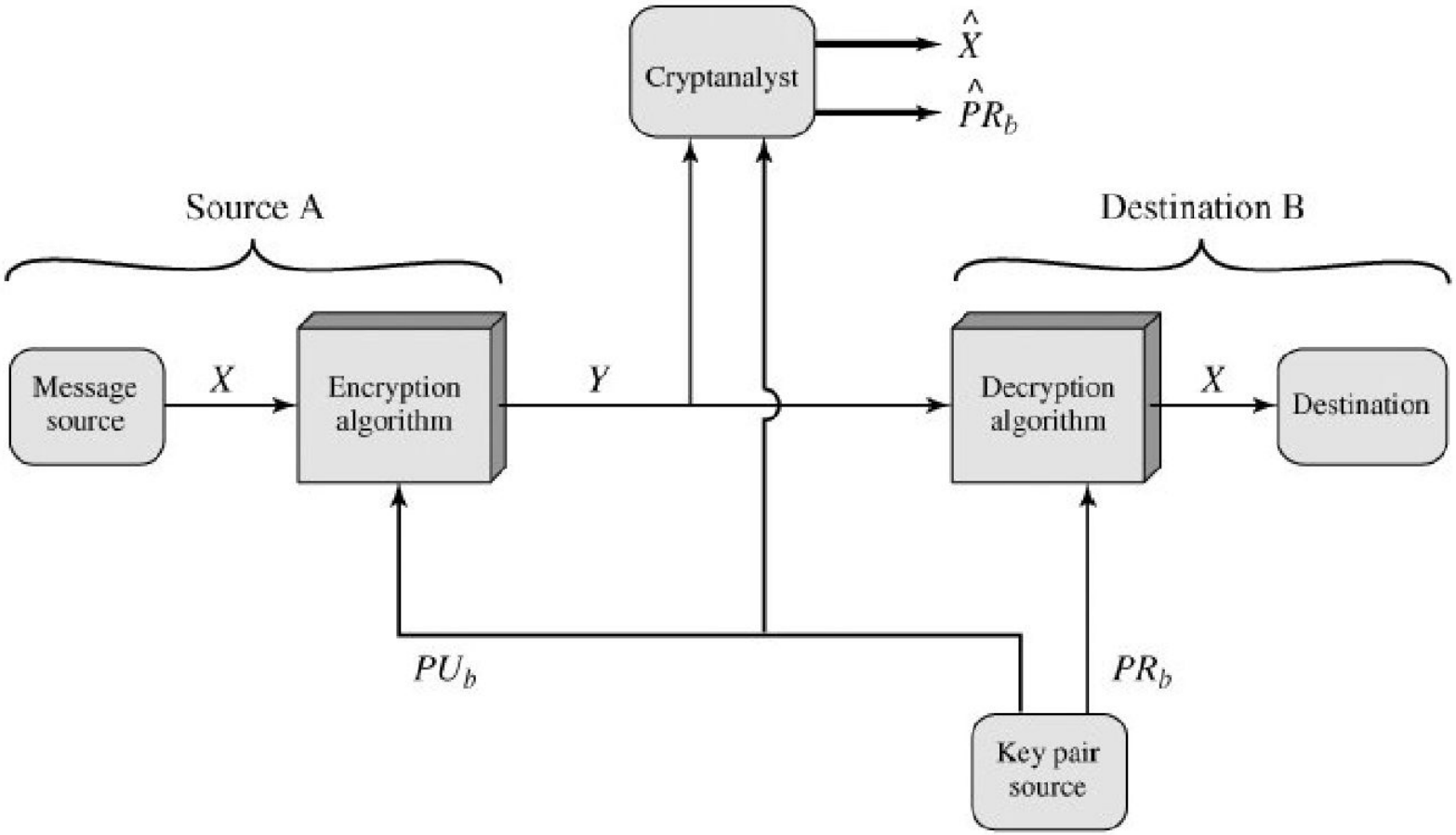
- **Encryption/decryption:** The sender encrypts a message with the recipient's public key.
- **Digital signature:** The sender "signs" a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message.
- **Key exchange:** Two sides cooperate to exchange a session key. Several different approaches are possible, involving the private key(s) of one or both parties.

Applications of Public-Key Cryptography

Table 9.2. Applications for Public-Key Cryptosystems

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No

Public Key Cryptography



Requirements for Public Key Cryptography

- It is computationally easy for a party B to generate a pair (public key KU_b, private key KR_b).
- It is computationally easy for a sender A, knowing the public key and the message to be encrypted, M, to generate the corresponding ciphertext:

$$C = \mathrm{E}_{KUb}(M)$$

- It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message:

$$M = \mathrm{D}_{KRb}(C) = \mathrm{D}KRb\{\mathrm{E}_{KUb}(M)\}$$

- It is computationally infeasible for an adversary, knowing the public key, KU_b, to determine the private key, KR_b.
- It is computationally infeasible for an adversary, knowing the public key, KU_b, and a ciphertext, C, to recover the original message, M.

Knapsack cryptography

- It was developed by Ralph Markle and Martin Hellman
- It is based on the subset sum problem (a special case of the knapsack problem).
- The problem is as follows: given a set of numbers A and a number b, find a subset of A which sums to b. In general, this problem is known to be NP-complete.
- It was one of the first general public key algorithm
- It was based on the basic Knapsack problem
- Eg
- If in a bag the weights are to be filled such that the weight of bag is exact 20
- The choice is on the basis of the weights available eg ..{2,3,5,7,8,10,12}
- So the weights chosen can be {3,7,10} or {2,8,10} or ……..

Knapsack Cryptography

- It uses two keys
- Public key- this follows the Hard Knapsack approach
- Private Key- It follows the Easy Knapsack approach
-

Easy Knapsack Approach

An easy knapsack problem is one in which set

$A = \{a_1, a_2, \ldots a_n\}$ is a super-increasing sequence

A super-increasing sequence is one in which the next term of the sequence is greater than the sum of all preceding terms:

$$a_2 > a_1,\ a_3 > a_1 + a_2, \ldots,\ a_n > a_1 + a_2 + \ldots + a_{n-1}$$

Example: A= $\{1, 2, 4, 8, \ldots 2^{n-1}\}$ is super-increasing sequence

Knapsack Cryptosystem

Imagine you had a set of weights 1, 6, 8, 15 and 24. To pack a knapsack weighing 30, you could use weights 1, 6, 8 and 15. It would not be possible to pack a knapsack that weighs 17 but this might not matter.

You might represent the weight 30 by the binary code 11110 (one 1, one 6, one 8, one 15 and no 24).

Plain text	1 0 0 1 1	1 1 0 1 0	0 1 0 1 1	0 0 0 0 0
Knapsack	1 6 8 15 24	1 6 8 15 24	1 6 8 15 24	1 6 8 15 24
Cipher text	1 + 15 + 24 = 40	1 + 6 + 15 = 22	6 + 15 + 24 = 45	0 = 0

Example

Step 1 : Key Generation at Receiver End

To produce a normal knapsack sequence, take a superincreasing sequence; e.g. **b={1, 2, 4, 10, 20, 40}**. **Multiply all the values by a number, r, modulo n.** The modulus should be a number greater than the sum of all the numbers in the sequence, for example, 110. The multiplier should have no factors in common with the modulus. So let's choose 31. The normal knapsack sequence would be:
×

1 * 31 mod(110) = 31 1

2* 31 mod(110) = 62

4 * 31 mod(110) = 14

10 * 31 mod(110) = 90

20 * 31 mod(110) = 70

40 * 31 mod(110) = 30

So the public key is: **a={31, 62, 14, 90, 70, 30}** and the private key is **b={1, 2, 4, 10, 20.40}**. Public key will share with sender who want to send data to receiver.

Example

Step 2 : Encryption at Sender End

Let's sender want to send a message that is in binary code:

x=1001001111100101110

The knapsack contains six weights so we need to split the message into groups of six:

100100

111100

101110

This corresponds to three sets of weights with totals as follows

The public key used for encryption is **{31, 62, 14, 90, 70, 30}**

100100 = 31 + 90 = 121

111100 = 31+62+14+90 = 197

101110 = 31+14+90+70 =205

So the coded cipher message is **s={121, 197, 205}**, will transfer using some channel.

Example

Step 3 : Decryption at Receiver End

Receiver receive cipher message **s={121, 197, 205}**, Now the receiver has to decode the message..

The person decoding must know the two numbers 110 and 31 (the modulus and the multiplier). Let's call the modulus "n" and the number you multiply by "r".

We need r^{-1} , which is a multiplicative inverse of n mod m, i.e.

$r(r^{-1})$= 1 mod n

Which can be found using Euclidean algo

The value is 71

Example

The coded cipher message **s={121, 197, 205}:**

121 × 71 mod(110) = 11 = 1+10=>100100

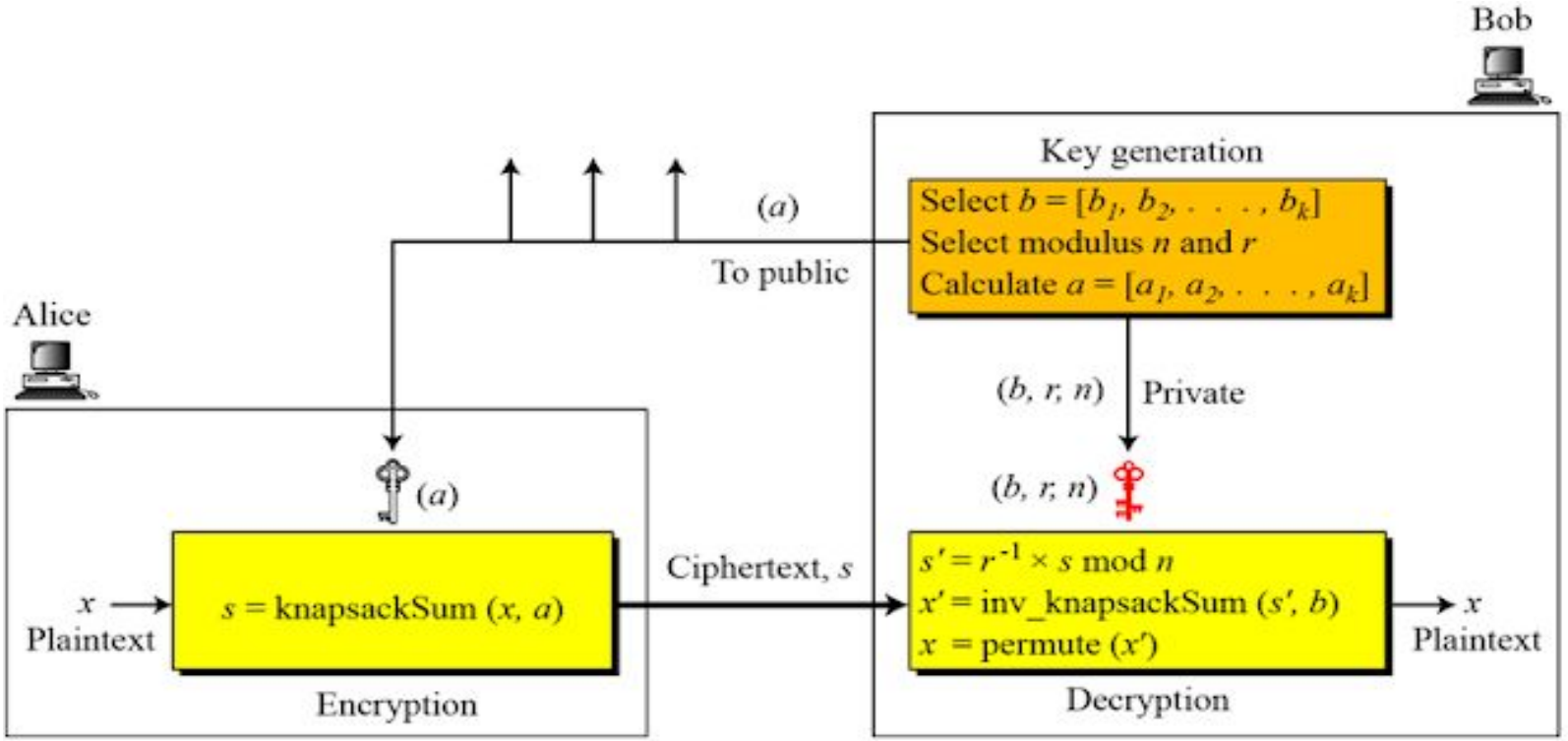
197 × 71 mod(110) = 17 = 1+2+4+10=>111100

205 × 71 mod(110) = 35 = 1+4+10+20=>101110

The decoded message is:

x=100100111100101110.

Knapsack Cryptosystem



Secret Communication with Knapsack Cryptosystem

RSA

- Based on the idea that factorization of integers into their prime factors is hard.
- Proposed by Rivest, Shamir, and Adleman in 1977 and a paper was published in The Communications of ACM in 1978
- Uses a key of size generally 1024 bits 0r 309 digits
- Uses prime number for key generation
- Factorization is the strength
- RSA is a block cipher **in which plain text and cipher text are integers** between 0 and n-1 for some n.

RSA

- It is possible to find values of e, d, n such that

  $M^{ed} \bmod n = M$ for all $M < n$.

- It is relatively easy to calculate mod M^e mod n and C^d for all values of $M < n$.
- It is infeasible to determine d given e and n.

RSA

- 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
  - $e.d=1 \bmod ø(N)$ and $0 \leq d \leq N$
- publish their public encryption key: KU={e,N}
- keep secret private decryption key: KR={d,p,q}

RSA

- 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)

RSA Algorithm

Key Generation

Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

Encryption

Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

Decryption

Ciphertext:	C
Plaintext:	$M = C^d \bmod n$

Example of RSA Algorithm

- Select P and Q two Prime Numbers
- Calculate n=P x Q
- Calculate φ(n) = ?

$$ed \bmod \phi(n) = 1$$

7,119 which is public key is e,n

E = 7

N = 119 = p*q = 7*17

phi(n) = 6*16 = 96

d =

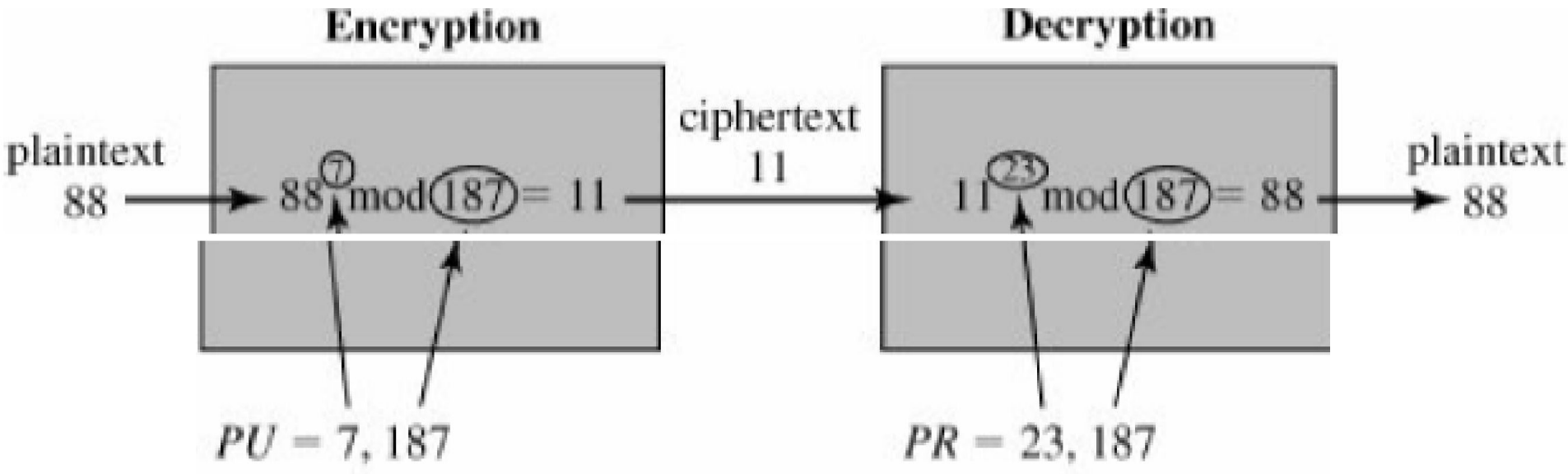
Example of RSA Algorithm

- Choose p = 3 and q = 11
- Compute n = p * q = 3 * 11 = 33
- Compute φ(n) = (p - 1) * (q - 1) = 2 * 10 = 20
- Choose e such that 1 < e < φ(n) and e and n are coprime. Let e = 7
- Compute a value for d such that (d * e) % φ(n) = 1. One solution is d = 3 [(3 * 7) % 20 = 1]
- Public key is (e, n) => (7, 33)
- Private key is (d, n) => (3, 33)
- The encryption of $m = 2$ is $c = 2^7 \% 33 = 29$
- The decryption of $c = 29$ is $m = 29^3 \% 33 = 2$

Example of RSA Algorithm

- Select primes: p=17 & q=11
- Compute $n = pq$ =17×11=187
- Compute ø(n)=(p–1)(q-1)=16×10=160
- Select e *such that e is relatively prime to* ø(n)=160 and less than ø(n); we choose e=7.
- Determine d such that $de \equiv 1 \pmod{160}$ and $d<160$ Value is d=23 since 23×7=161=10×160+1, d can be calculated **using Extended Euclid's algorithm.**
- Generate the keys: $ed \bmod \phi(n) = 1$
  - public key: KU={7,187}
  - private key: KR={23,187}

RSA Example


Encryption
Decryption
plaintext
88
$88^{7} \bmod 187 = 11$
ciphertext
11
$11^{23} \bmod 187 = 88$
plaintext
88
PU = 7, 187
PR = 23, 187

$88^7 \bmod 187 = [(88^4 \bmod 187) \times (88^2 \bmod 187) \times (88^1 \bmod 187)] \bmod 187$

$88^1 \bmod 187 = 88$

$88^2 \bmod 187 = 7744 \bmod 187 = 77$

$88^4 \bmod 187 = 59{,}969{,}536 \bmod 187 = 132$

$88^7 \bmod 187 = (88 \times 77 \times 132) \bmod 187 = 894{,}432 \bmod 187 = 11$

For decryption, we calculate $M = 11^{23} \bmod 187$:

$11^{23} \bmod 187 = [(11^1 \bmod 187) \times (11^2 \bmod 187) \times (11^4 \bmod 187) \times (11^8 \bmod 187) \times (11^8 \bmod 187)] \bmod 187$

$11^1 \bmod 187 = 11$

$11^2 \bmod 187 = 121$

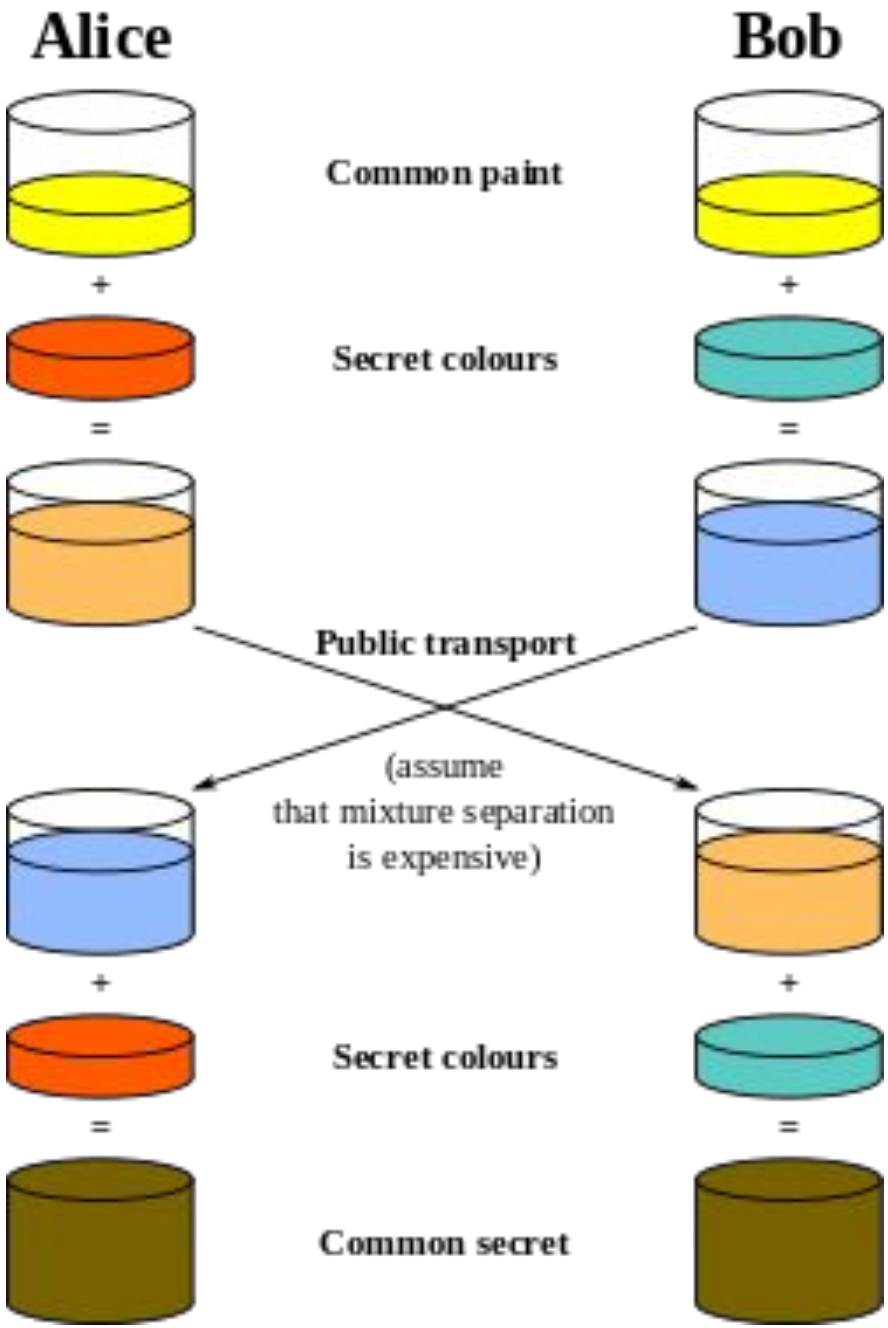
$11^4 \bmod 187 = 14{,}641 \bmod 187 = 55$

$11^8 \bmod 187 = 214{,}358{,}881 \bmod 187 = 33$

$11^{23} \bmod 187 = (11 \times 121 \times 55 \times 33 \times 33) \bmod 187 = 79{,}720{,}245 \bmod 187 = 88$

Diffie-Hellman Key Exchange

- Suppose two people, Alice and Bob [traditional names], want to use insecure email to agree on a secret "shared key" that they can use to do further encryption for a long message. **How is that possible?**
- The so-called Diffie-Hellman method provides a way.
- The purpose of the algorithm to enable two users to exchange the key securely that then can be used for subsequent encryption of message.
- It is a practical method for public agreement of a secret key
- It is used in a number of commercial products
- EXCHANGE THE KEYS WITHOUT ACTUALLY EXCHANGING THEM


Alice
Bob
Common paint
+
Secret colours
=
Public transport
(assume
that mixture separation
is expensive)
+
Secret colours
=
Common secret

D H exchange procedure

- Diffie-Hellman key agreement protocol
  - Exponential key agreement
  - Allows two users to exchange a secret key
  - Requires no prior secrets
  - Real-time over an untrusted network
- Requires two large numbers, one prime (P), and (G), a primitive root of P
- P and G are both publicly available numbers
  - P is at least 512 bits
- Users pick private values a and b
- Compute public values
  - $x = g^a \bmod p$
  - $y = g^b \bmod p$
- Public values x and y are exchanged
- Compute shared, private key
  - $k_a = y^a \bmod p$
  - $k_b = x^b \bmod p$
- Algebraically it can be shown that $k_a = k_b$
  - Users now have a symmetric secret key to encrypt

D-H Key Exchange Algorithm

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

User B Key Generation

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

Calculation of Secret Key by User A

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

Calculation of Secret Key by User B

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

Diffie-Hellman Key Exchange

Alice

Bob and Alice know and have the following :
$p = 23$ (*a prime number*) $g = 11$ (*a generator*)

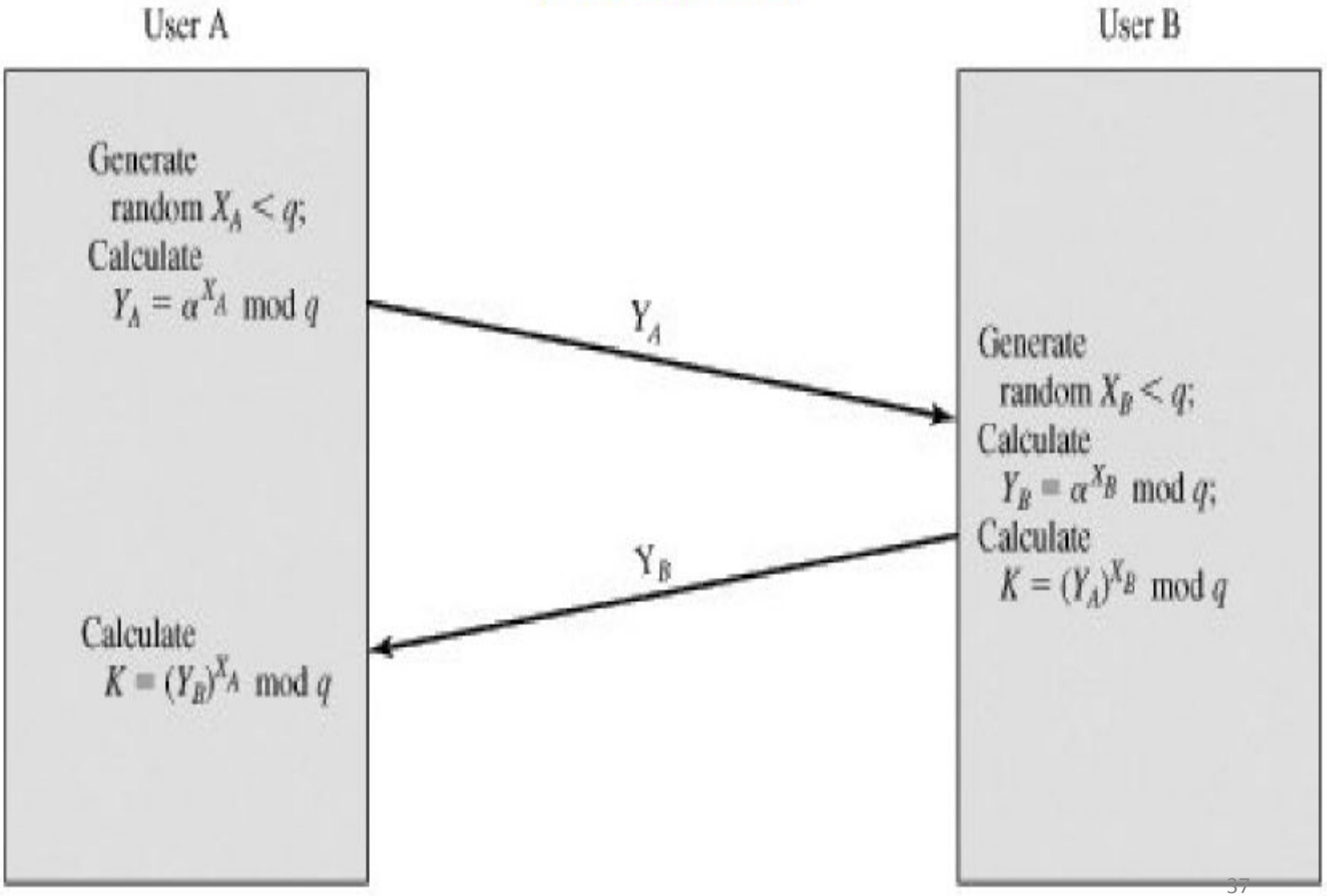
Bob

Alice	Bob
Alice chooses a secret random number a = 6	Bob chooses a secret random number b = 5
Alice computes : $A = g^a \bmod p$ $A = 11^6 \bmod 23 = 9$	Bob computes : $B = g^b \bmod p$ $B = 11^5 \bmod 23 = 5$
Alice receives B = 5 from Bob	Bob receives A = 9 from Alice
Secret Key = $K = B^a \bmod p$	Secret Key = $K = A^b \bmod p$
$K = 5^6 \bmod 23 = 8$	$K = 9^5 \bmod 23 = 8$

The common secret key is : 8

N.B. We could also have written : $K = g^{ab} \bmod$

Diffie-Hellman Key Exchange



Example

- Alice and Bob get public numbers
  - P = 23, G = 9
- Alice and Bob compute public values
  - $X = 9^4 \bmod 23 = 6561 \bmod 23 = 6$
  - $Y = 9^3 \bmod 23 = 729 \bmod 23 = 16$
- Alice and Bob exchange public numbers

Example

- Alice and Bob compute symmetric keys
  - $k_a = y^a \bmod p = 16^4 \bmod 23 = 9$
  - $k_b = x^b \bmod p = 6^3 \bmod 23 = 9$
- Alice and Bob now can talk securely!

Applications

- Diffie-Hellman is currently used in many protocols, namely:
  - Secure Sockets Layer (SSL)/Transport Layer Security (TLS)
  - Secure Shell (SSH)
  - Internet Protocol Security (IPSec)
  - Public Key Infrastructure (PKI)

Diffie-Hellman Example

- Alice & Bob who wish to swap keys: key exchange is based on the use of prime numbers, **prime q=353** and a **primitive root of 353, in this case α=3**
- They select the random secret keys:
  - A chooses x_A=97, B chooses x_B=233 respectively
- Each computes its public key:
  - $y_A = \alpha^{xA} \bmod q = 3^{97} \bmod 353 = 40$ (Alice)
  - $y_B = \alpha^{xB} \bmod q = 3^{233} \bmod 353 = 248$ (Bob)
- After they exchange public keys, each can compute common secrete keys as:

  $K_{AB} = y_B^{xA} \bmod 353 = 248^{97} \bmod 353 = 160$ (Alice)

  $K_{AB} = y_A^{xB} \bmod 353 = 40^{233} \bmod 353 = 160$ (Bob)
  - We assume that the attacker would have available the following information.
  - Q, α, Y_A and Y_B

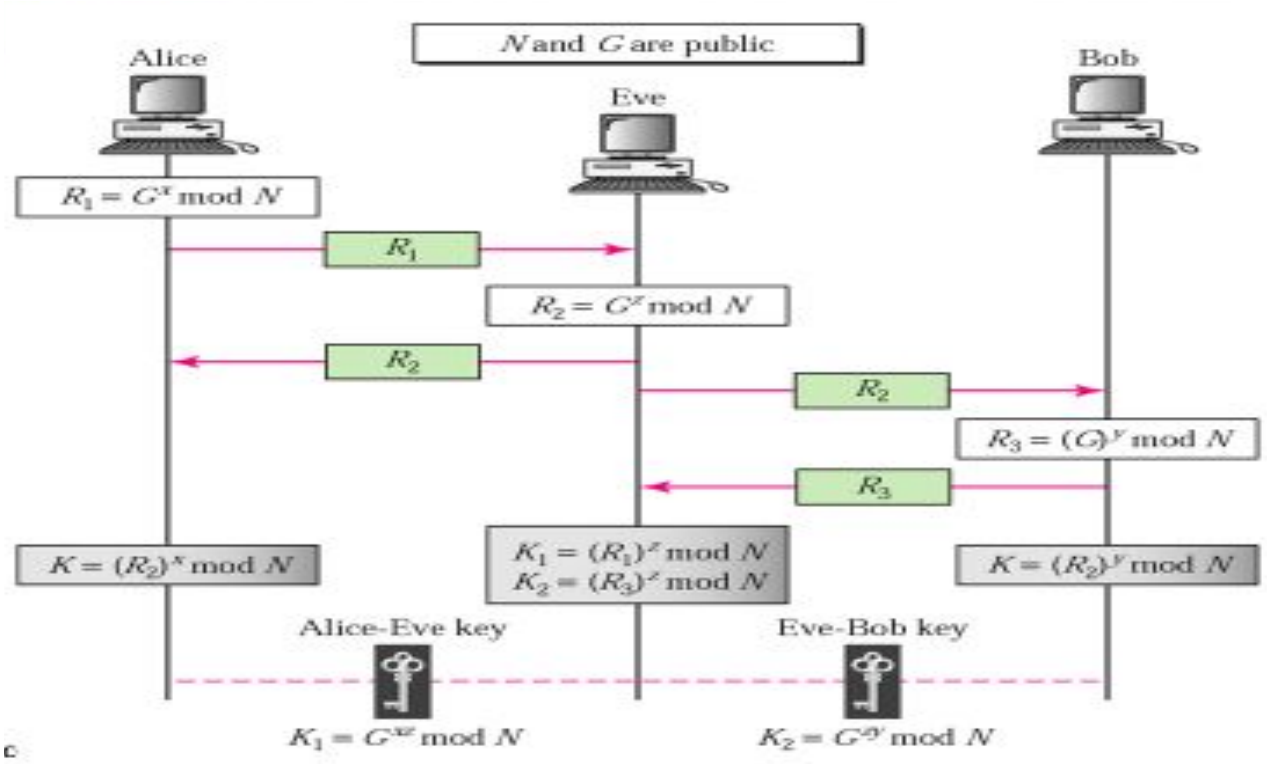
For calculations for modulus exponentiation refer to

https://www.khanacademy.org/computing/computer-science/cryptography/modarithmetic/a/fast-modular-exponentiation

How to Calculate Primitive Root?

- **Let p be a prime. Then b is a *primitive root* for p** if the powers of b i e 1,b, b^2, b^3, …include all of the residue classes mod p
- **Examples:** If p=7, then 3 is a primitive root for p because the powers of 3 are 1, 3, 2, 6, 4, 5---that is, **every number mod 7 occurs except 0.** But 2 isn't a primitive root because the powers of 2 are 1, 2, 4, 1, 2, 4, 1, 2, 4...missing several values.
- **Example:** If p=13, then 2 is a primitive root because the powers of 2 are 2, 4, 8, 3, 6, 12, 11, 9, 5, 10, 7---which is all of the classes mod 13 except 0. There are other primitive roots for 13.

Man in the middle attack



Man in the middle Attack

- Darth prepares for the attack by generating two random private keys X^D1 and X^D2 and then computing the corresponding **public keys Y^D1 and Y^D2.**
- Alice transmits Y^A to Bob.
- Darth intercepts Y^A and transmits Y^D1 to Bob. Darth also calculates

  $K2 = (Y^A)\ X^D2 \bmod q$
- Bob receives Y_D1 and calculates $K1 = (Y^D1)\ X^B \bmod q$.

  Bob transmits X^A to Alice.
- Darth intercepts X^A and transmits Y^D2 to Alice. Darth calculates
- $K1 = (Y^B)\ X^D1 \bmod q$

  Alice receives Y^D2 and calculates $K2 = (Y^D2)\ X^A \bmod q$.
- At this point, Bob and Alice think that they share a secret key, but instead **Bob and Darth share secret key *K1*** *and* ***Alice and Darth share*** **secret key *K2.*** *All future communication between Bob and Alice is compromised in the following way:*

Man in the middle Attack

- Alice sends an encrypted message *M : E(K2, M)*.
- **Darth intercepts the encrypted message and decrypts it, to recover *M*** Darth sends Bob E(K1, M) or E(K1, M'), where M' is any message.
- **In the first case,** Darth simply wants to eavesdrop on the communication without altering it.
- **In the second case,** Darth wants to modify the message going to Bob.

NOTE: The key exchange protocol is vulnerable to such an attack because it does not authenticate the participants.