Unit 4

Public key Cryptography

Presented by:
Dr. Mohammed Alkhanafseh

Terminology Related to Asymmetric Encryption

Asymmetric Keys
Two related keys, a public key and a private key that are used to perform complementary operations, such as encryption and decryption or signature generation and signature verification.

Public Key Certificate
A digital document issued and digitally signed by the private key of a Certification Authority that binds the name of a subscriber to a public key. The certificate indicates that the subscriber identified in the certificate has sole control and access to the corresponding private key.

Public Key (Asymmetric) Cryptographic Algorithm
A cryptographic algorithm that uses two related keys, a public key and a private key. The two keys have the property that deriving the private key from the public key is computationally infeasible.

Public Key Infrastructure (PKI)
A set of policies, processes, server platforms, software and workstations used for the purpose of administering certificates and public-private key pairs, including the ability to issue, maintain, and revoke public key certificates.

Principles of Public-Key Cryptosystems

- The concept of public-key cryptography evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption.

Key distribution

- **How to have secure communications in general without having to trust a KDC with your key**

Digital signatures

- **How to verify that a message comes intact from the claimed sender**

- Whitfield Diffie and Martin Hellman from Stanford University achieved a breakthrough in 1976 by coming up with a method that addressed both problems and was radically different from all previous approaches to cryptography.

A public-key encryption scheme has six ingredients

Plaintext	Encryption algorithm	Public key	Private key	Ciphertext	Decryption algorithm
The readable message or data that is fed into the algorithm as input	Performs various transforma-tions on the plaintext	Used for encryption or decryption	Used for encryption or decryption	The scrambled message produced as output	Accepts the ciphertext and the matching key and produces the original plaintext

Public key Cryptography

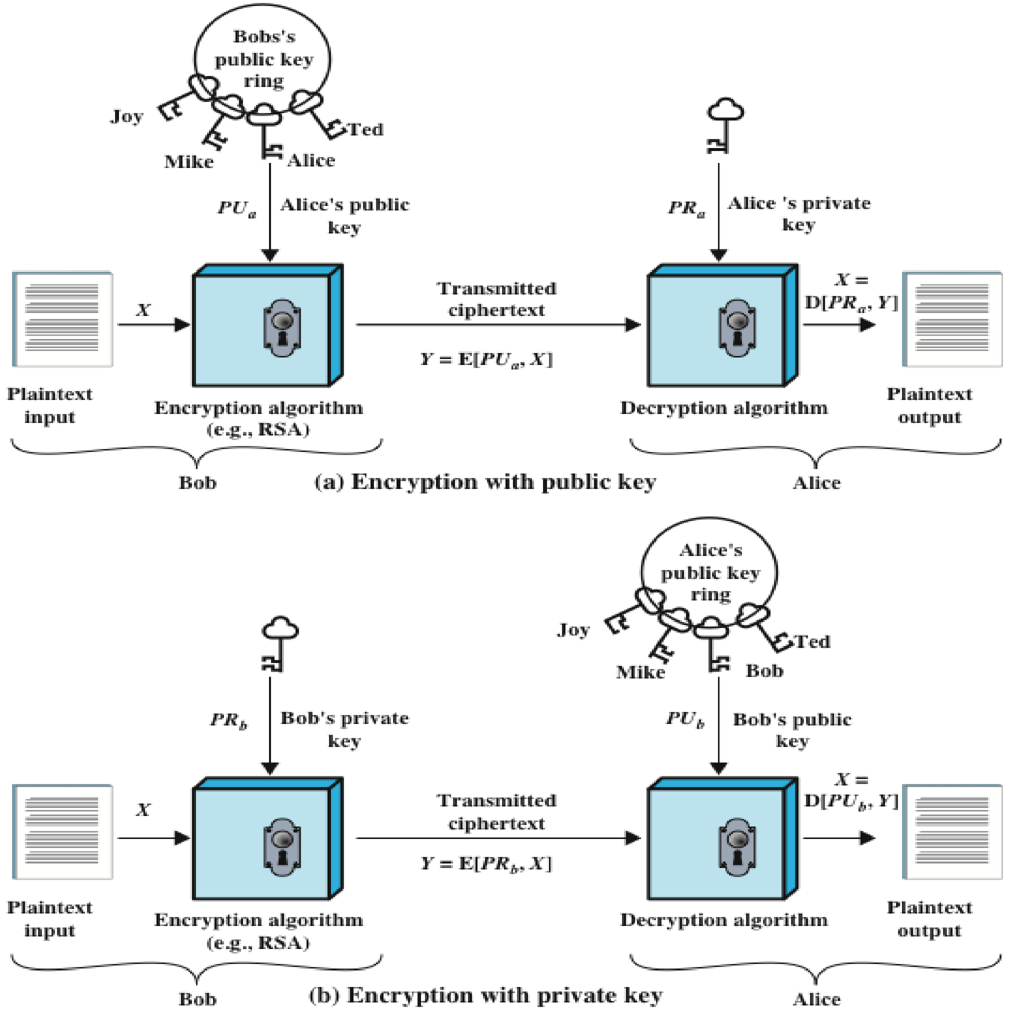


Figure 9.1 Public-Key Cryptography

Conventional and 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 the key is kept secret. 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 a related algorithm for 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 one of the keys is kept secret. 3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

Symmetric and Asymmetric Encryption

Comparison Factor	Symmetric Encryption	Asymmetric Encryption
Number of Cryptographic Keys	you need n(n−1)/2 keys.	you need 2n key pairs.
Complexity	Symmetric encryption is a simple technique compared to asymmetric encryption as only one key is employed to carry out both the operations.	Contribution from separate keys for encryption and decryption makes it a rather complex process
Algorithms Employed	RC4, AES, DES, 3DES, QUAD	RSA, Diffie-Hellman, ECC, El Gamal, DSA
Performance	Symmetric encryption is fast in execution.	Asymmetric Encryption is slow in execution due to the high computational burden.
Age	Symmetric encryption is an old technique of encryption.	Asymmetric encryption is a relatively new technique of encryption
Mathematical Representation	Mathematically, symmetric encryption is represented as P=D (K, E(P)), where: K is encryption and decryption key. P=Plain text D=Decryption E (P)= encryption of plain text	Mathematically asymmetric encryption is represented as P=D(Kd, E (Ke,P)), where: Ke=encryption key Kd=decryption key D=decryption E(ke, P)= encryption of plain text using private key.

Application of Public-Key Encryption.

- Public-key cryptosystems can be classified into three categories:

Encryption/decryption	• The sender encrypts a message with the recipient’s public key
Digital signature	• The sender “signs” a message with its private key
Key exchange	• Two sides cooperate to exchange a session key

- Some algorithms are suitable for all three applications, whereas others can be used only for one or two

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

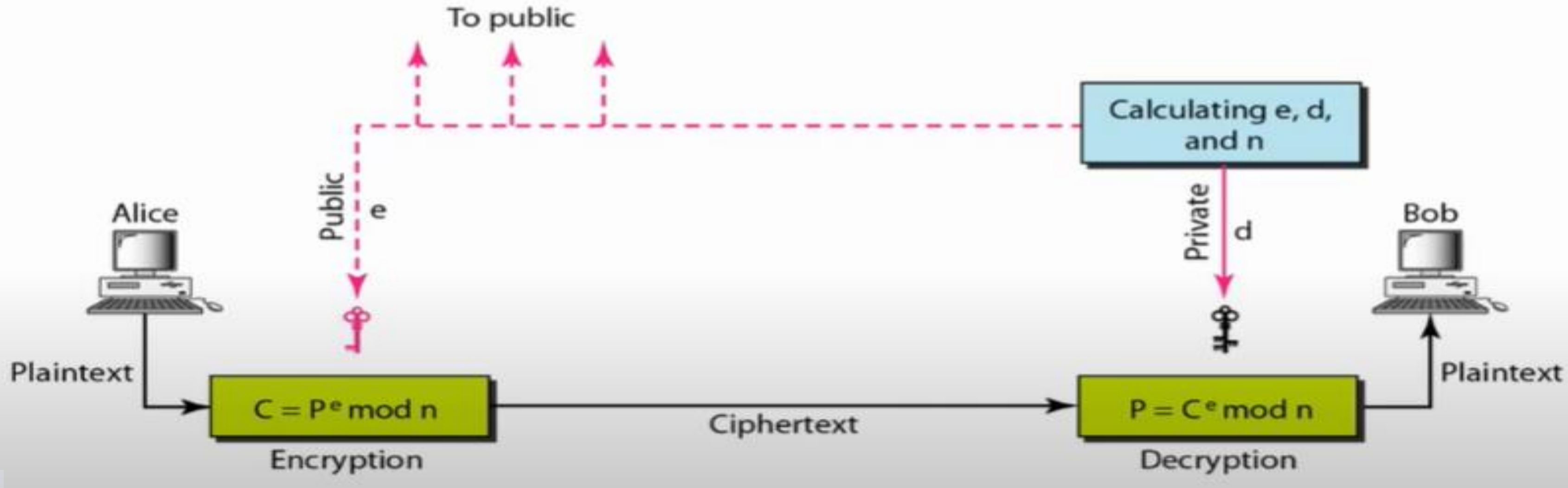
RSA

The most common public key algorithm is RSA, named for its inventors ***Rivest, Shamir, and Adelman.***

It uses two numbers:

e → public key
d → private

The two keys, ***e*** and ***d***, have a special relationship to each other.



RSA Applications

RSA is Useful for short messages but not for long messages, because it depends on high level of calculation and need more time and resources for computation.

It used in all digital signature and authentication process

RSA – selecting keys

- Bob uses the following steps to select the private and public keys:
- Bob chooses two very large prime numbers ***p*** and ***q***

By "large" we typically mean at least 512 bits

-Bob multiplies ***p*** and ***q*** to find ***n*** ➔ ***n=p x q***

-Bob calculates another number ***ϕ(n) = (p -1) X (q -1) coprime with n***

-Bob chooses a random number ***e, 1< e< ϕ*** . S.t

Choose a prime number "e", such that e is co-prime to Φ and N , i.e,

Φ OR N is not divisible by e.

gcd (ϕ,e)=1. You can use Euclidian algorithm to help you find correct e.

Gcd: greatest coming devisor

He then calculates ***d*** so that ***d X e mod ϕ = 1 or d.e=1mod ϕ***

1< d< ϕ . (using table method--Extended Euclidian Algorithm)

Bob announces ***e*** and ***n*** to the public; he keeps ***ϕ*** and ***d*** secret.

RSA – Encryption and Decryption

Encryption $$C = P^e \pmod{n}$$

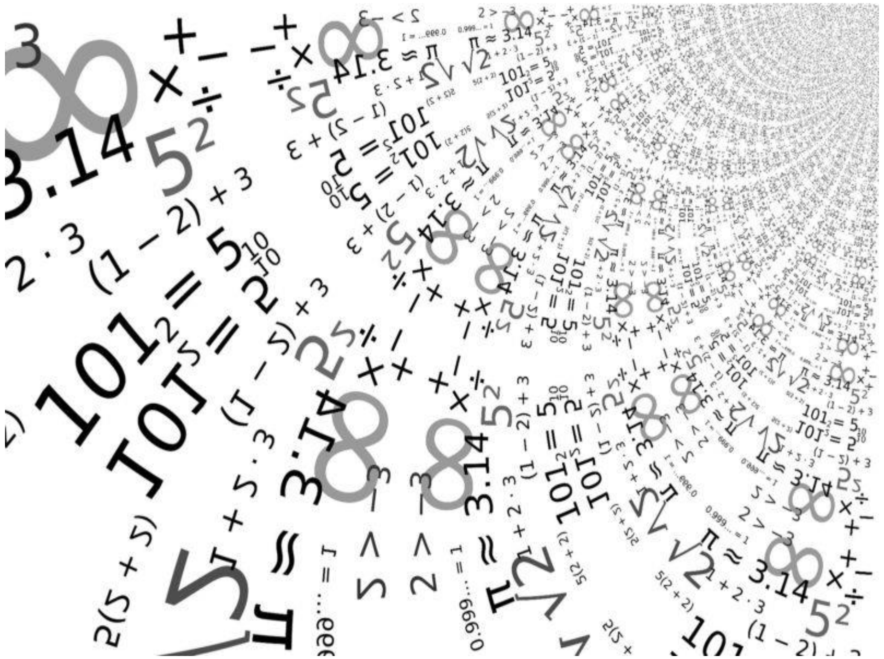
Decryption $$P = C^d \pmod{n}$$

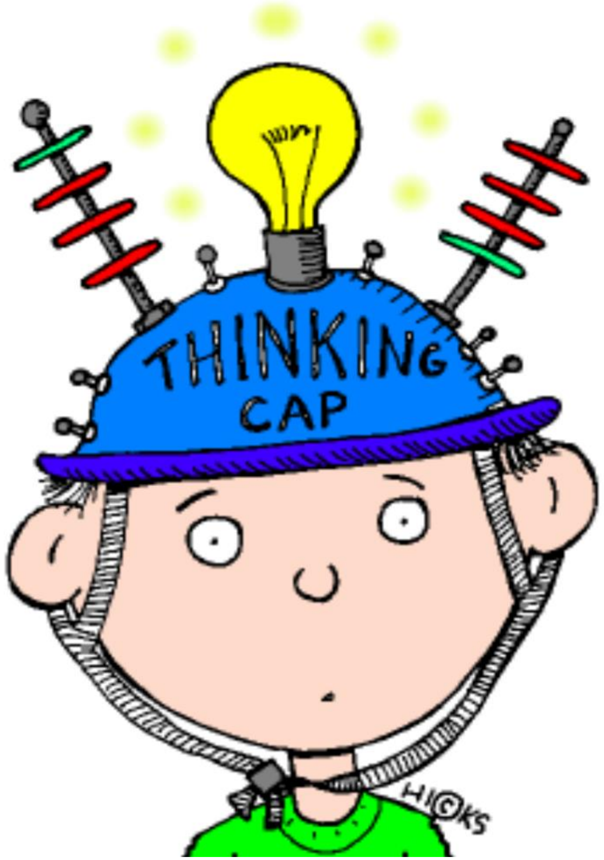
Restriction

P **<** ***n*** , if not, the plaintext needs to be divided into blocks to make ***P*** less than ***n***.

Now the way to solve this problem uses Modular exponentiation Method also called as Right to Left binary method.

Pre RSA





Examples of Modular Exponentiation

Calculate the value of: 3^{200} ***mod* 50**

$3^1 = 3 \quad 3 mod 50 = 3$

$3^2 = \quad 3^2\ mod\ 50 = 9 mod 50 = 9$

$3^4 = \quad 9^2\ mod\ 50 = 81 mod 50 = 31$

$3^8 = \quad 31^2 mod 50 = 961$ mod $50 = 11$

$3^{16} = \quad 11^2 mod 50 = 121$ mod $50 = 21$

$3^{32} = \quad 21^2 mod 50 = 441$ mod $50 = 41$

$3^{64} = \quad 41^2 mod 50 = 1681$ mod $50 = 31$

$3^{128} = \quad 31^2 mod 50 = 961$mod $50 = 11$

$\mathbf{3^{256} = (3^{128})^2}$ ***Stop* 256 > 200**

$\mathbf{3^{200} = 3^{128+64+8}}$

3^1	3^2	3^4	$\boxed{3^8}$	3^{16}	3^{32}	$\boxed{3^{64}}$	$\boxed{3^{128}}$	n
3	9	31	$\boxed{11}$	21	41	$\boxed{31}$	$\boxed{11}$	n mod 50

11*31*11 mod50

3751Mod 50=1

The answer 3^{200} *mod* 50=1

Examples of Modular Exponentiation

Calculate the value of: $\mathbf{23^{391}}$ ***mod* 55**

$23^1 = 23 \quad 23 mod 55 = 23$

$23^2 = 529 \quad 529 mod 55 = 34$

$23^4 = (34^2) mod\ 55 = 1156\ mod 55 = 1$

$23^8 = (23^4)^2 \quad 1^2 mod 55 = 1$

$23^{16} = (23^8)^2 \quad 1^2 mod 55 = 1$

$23^{32} = (23^{16})^2 \quad 1^2 mod 55 = 1$

$23^{64} = (23^{32})^2 \quad 1^2 mod 55 = 1$

$23^{128} = (23^{64})^2 \quad 1^2 mod 55 = 1$

$\mathbf{23^{256}} = (23^{128})^2 \quad 1^2 mod 55 = 1$

$\mathbf{23^{512}} = (3^{256})^2 \boldsymbol{Stop\ 512 > 391}$

23

$\mathbf{23^{391} = 3^{256+128+4+2+1}}$

23^1	23^2	23^4	23^8	23^{16}	23^{32}	23^{64}	23^{128}	23^{256}	**n**
23	34	1	1	1	1	1	1	1	**n mod 55**

(23*34*1*1*1) mod55

782Mod 55=12

The answer $\mathbf{23^{391}}$ ***mod*** **55=12**

Basics

- We'll need **Euler's Theorem:**

  If a is relatively prime to n then

  $\gcd(a,n)=1$

  We have

  $a^{\varphi(n)} = 1 \bmod n$

Basics

- We'll need **Euler's Theorem:**

  If x is relatively prime to n then

  $$x^{\varphi(n)} = 1 \bmod n$$

Case1 :

If n is a prime number for example

n=13

$\varphi(n) = 13-1 = 12$

No.s of relatively prime

(1,2,3,4,5,6,7,8,9,10,11,12)

n=7

$\varphi(n) = 7-1 = 6$

No.s of relatively prime

(1,2,3,4,5,6)

Basics

Case2 :

If n is a product of 2 primes P &Q

$$\varphi(n)= \varphi(p)*\varphi(Q)= (p-1)(Q-1)$$

$\varphi(21)= \varphi(7)*\varphi(3)= (7-1)(3-1)= 6*2=12$

No.s of relatively prime =12

(1,2,4,5,8,10,11,13,16,17,19,20)

Basics

Case3 :

n=p^e If P is a prime

φ(n)= $(p^e$-$p^{e-1})$

φ(8)= 2^3

= $(2^3$-$2^2)$

= (8-4)=4

No.s of relatively prime =4

(1,3,5,7)

Basics

- We'll need **Euler's Theorem:**

  If a is relatively prime to n then $a^{\varphi(n)} = 1 \bmod n$

- Facts:

  1) $ed = 1 \bmod (p-1)(q-1)$
  2) By definition of "mod", $ed = k(p-1)(q-1) + 1$
  3) $\varphi(N) = (p-1)(q-1)$

- Then $ed - 1 = k(p-1)(q-1) = k\varphi(N)$

- So, $\mathbf{C^d} = M^{ed} = M^{(ed-1)+1} = M \cdot M^{ed-1} = M \cdot M^{k\varphi(N)}$
  $= M \cdot (M^{\varphi(N)})^k \bmod N = M \cdot 1^k \bmod N = \mathbf{M \bmod N}$

Extended Euclidean algorithm to find Modular inverse

Euclidean algorithm, which used in RSA algorithm can be used to find the gcd(x,y)m where the GCD refers to the greatest common divisor between numbers, as example the GCD(18,12)

The divisor of 18 is : 1, 2, 3, 6,9

The divisor of 12 is: 1,2,3,4,6

Based on that the 6 is the GCD for both 12 and 18

It is easy to find the common divisor when the two numbers is small

Euclidean algorithm

Example in complicated GCD numbers

Find the GCD for both (2024, and 748).

First step refers to divide the 2024 by 748.

The result is 2 and some reminders.

Next step refers to 2*748 = 1496.

2024 – 1496 = 528.

Now find GCD(748,528)

748/528 = 1 and reminders

748-528 = 220

The new GCD is (528, 220)

528/220 = 2 and reminders

2*220 = 440

The 528-440 = 88

Example in complicated GCD numbers

New GCD is between (220,88)

220/88 = 2 and some fractions,

2*88= 176, based on that 220-176 = 44

New GCD is between (88 and 44)

88/44 = 2 and 0 fractions

In this step the Euclidean algorithm will be stop and the GCD for (2024 and 748) is 44

Extended Euclidean algorithm

One of methods that used to find Extended Euclidean algorithm refers to build a table and go through set of rows until reach to the 1 in r and the method here is to find both EEA(Extended Euclidean algorithm) and both S& t which required

Example find gcd(161, 28)

The value of R1 is the highest value which is 161 and the value of R2 is the lowest which is 28, and the value of q is result of dividing 161/28 which equal 5 and the value of **R** is the reminder which is 21

The value of s1 is 1 and value of s2 is 0 and the value t1 is 0 and the value of t2 is 1. The value of S and T is as follow

S= S1-(S2*q) and the value of T= T1-(T2*q), First S = 1-(0*5) =1 and T = t1-(t2*~~1~~) = 0-1*5 = -5

Q	R1	R2	R	S1	S2	S	T1	T2	T
5	161	28	21	1	0	1	0	1	-5
1	28	21	7	0	1	-1	1	-5	6
…	…	…	…	…	…	…	..	..	…
3	21	7	0	1	-1	4	-5	6	-23
X	7	0	X	-1	4	X	6	-23	x

The value of S gcd is 7 and value of t is 6 and value of s is -1

Extended Euclidean algorithm

How we can calculate d using Extended Euclidean algorithm

ax+by=gcd(a,b)……..(1)

d*e=1mod **φ…………(2)**

In our case we have a-> φ and b->e

φ x+**e**y=gcd(**φ**, **e**)……..(3)

From above equations we are getting the value of d

Here y will gives us value of d

Extended Euclidean algorithm example

P=7 and q=11

N=77 and **φ=60**

1<e< φ -> e=13

(d.e)=1mod60 -> (13d)=1 mod 60

60x+13y=gcd(60,13) (here y will give us value of d)

n	a	b	**d**	k
1	1	0	60	-
2	0	1	13	4
3	1-(0*4)=1	0-(1*4)=-4	(60-13*4)=8	13/8=1
4	-1	5	5	1
5	1-(-1*1)=2	-4-(5*1)=-9	8-5*1=3	1
6	-3	14	2	1
7	**5**	**-23**	**1**	**2**

Rules

1. $a_n = a_{n-2} - (a_{n-1} * K_{n-1})$
2. $b_n = b_{n-2} - (b_{n-1} * K_{n-1})$
3. $d_n = d_{n-2} - (d_{n-1} * K_{n-1})$ for n>2
4. $K_n = d_{n-1} / d_n$ for n>1

Extended Euclidean algorithm example

n	a	b	d	k
1	1	0	60	-
2	0	1	13	4
3	1-(0*4)=1	0-(1*4)=-4	(60-13*4)=8	13/8=1
4	-1	5	5	1
5	1-(-1*1)=2	-4-(5*1)=-9	8-5*1=3	1
6	-3	14	2	1
7	**5**	**-23**	**1**	**2**

Two possibilities of d

1. d> **φ then**
   d=dmod φ
2. d is negative then
   d=d+ φ

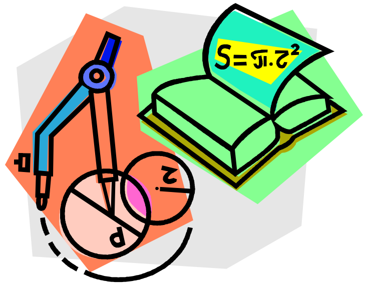
60x+13y=gcd(60,13)
60(5)+13(-23)=gcd(60,13)
300+-299=1
Value of y is d
d =-23 which is negative
d=d+ **φ**
d= -23+60=37
d*e=1mod φ
37*13=1(mod60)
(37*13)-1 =should be some multiple of 60
481-1=480 which is a multiple of 60

Rivest-Shamir-Adleman (RSA) Algorithm

- Developed in 1977 at MIT by Ron Rivest, Adi Shamir & Len Adleman
- Most widely used general-purpose approach to public-key encryption
- Is a cipher in which the plaintext and ciphertext are integers between 0 and $n - 1$ for some n
  - A typical size for n is 1024 bits, or 309 decimal digits

Algorithm Requirements

- For this algorithm to be satisfactory for public-key encryption, the following requirements must be met:
    1. It is possible to find values of *e, d, n* such that M^{ed} mod $n = M$ for all $M < n$
    2. It is relatively easy to calculate M^e mod n and C^d mod n for all values of $M < n$
    3. It is infeasible to determine *d* given *e* and *n*



RSA (Rivest , Shamir, and Adelman)Algorithm

- RSA makes use of an expression with exponentials
- Plaintext is encrypted in blocks with each block having a binary value less than some number *n*
- Encryption and decryption are of the following form, for some plaintext block *M* and ciphertext block C

  $$C = M^e \bmod n$$

  $$M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$$

- Both sender and receiver must know the value of *n*
- The sender knows the value of *e,* and only the receiver knows the value of *d*
- This is a public-key encryption algorithm with a public key of ***PU={e,n}*** and a private key of ***PR={d,n}***

RSA Algorithm Steps

1. Key Generation

1.1 choose two prime number (P,Q)
1.2 compute n=P*Q
1.3 compute Euler φ=(P-1)(Q-1)
1.4 choose e : 1<e< φ and comprime with φ Key (n,e)
1.5 calculate value of d using Extended Euclidean algorithms

2. Message Encryption

$C = M^e \bmod n$

3. Message Decryption

$M = C^d \, mod \, n =$

$d = e^{-1} \, mod \, \varphi$

- **PK(e,n)**
- **PR(d,n)**

n	a	b	**d**	k
1	1	0	160	-
2	0	1	7	22
3	1-(0*4)=1	0-(1*22)=-22	(160-7*22)=6	7/6=1
4	0-1*1 = -1	1-(-22*1= 23	1	6

Example 1 for RSA Algorithm

1. Select two prime numbers, p = 17 and q = 11.

2. Calculate $n = pq$ = 17 * 11 = 187.

3. Calculate $\varphi(n) = (p - 1)(q - 1)$ = 16 * 10 = 160.

4. Select e such that e is relatively prime to $\varphi(n)$ = 160 and less than $\varphi(n)$; we choose e = 7.

5. Determine d such that $d*e$ =1 (mod 160) and d < 160. The correct value is

***d* = 23**, because 23 * 7 = 161 = (1 * 160) + 1;

***d* can be calculated using the extended Euclid's algorithm**

Rules

1. $a_n = a_{n-2} - (a_{n-1} * K_{n-1})$
2. $b_n = b_{n-2} - (b_{n-1} * K_{n-1})$
3. $d_n = d_{n-2} - (d_{n-1} * K_{n-1})$
4. $K_n = d_{n-1} / d_n$ for n>1

23*7mod160 = 1 --→ d = 23, e = 7

C = P^e mod n = 11*17 = 187 NO

M – C^dmod n

Example 1 for RSA Algorithm

The resulting keys are public key $PU = \{7, 187\}$ and private key $PR = \{23, 187\}$. The example shows the use of these keys for a plaintext input of $\boldsymbol{M} = \mathbf{88}$. For encryption, we need to calculate $C = 88^7 \bmod 187$. Exploiting the properties of modular arithmetic, we can do this as follows.

For Encryption

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

$88^1 \bmod 187 = 88$

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

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

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

For Decryption

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

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

$11^1 \bmod 187 = 11$

$11^2 \bmod 187 = 121$

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

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

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

Example 1 for RSA Algorithm

Encrypt the Plain text **No** using RSA algorithm

Let p = 13, Q = 23

N = P* Q = 13*23 = 299

$\varphi(n) = (p - 1)(q - 1) = 12*22 = 264$

For Encryption

88^7 mo $\varphi(n) = (p - 1)(q - 1)$ d 187 = [(88^4 mod 187) * (88^2 mod 187) * (88^1 mod 187)] mod 187

88^1 mod 187 = 88

88^2 mod 187 = 7744 mod 187 = 77

88^4 mod 187 = 59,969,536 mod 187 = 132

88^7 mod 187 = (88 * 77 * 132) mod 187 = 894,432 mod 187 = 11

For Decryption

we calculate $M = 11^{23}$ mod 187:

11^{23} mod 187 = [(11^1 mod 187) * (11^2 mod 187) * (11^4 mod 187)* (11^8 mod 187) * (11^8 mod 187)] mod 187

11^1 mod 187 = 11

11^2 mod 187 = 121

11^4 mod 187 = 14,641 mod 187 = 55

11^8 mod 187 = 214,358,881 mod 187 = 33

11^{23} mod 187 = (11 * 121 * 55 * 33 * 33) mod 187
= 79,720,245 mod 187 = 88

Example 1 for RSA Algorithm

Encrypt the Plain text **No** using RSA algorithm

Let p = 13, Q = 23

N = P* Q = 13*23 = 299

$\varphi(n) = (p - 1)(q - 1)$ = 12*22 = 264

Generate d and e

e *is generated randmoly which*

The value of d is

For Decryption

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

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

$11^1 \bmod 187 = 11$

$11^2 \bmod 187 = 121$

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

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

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

Example 1 for RSA Algorithm

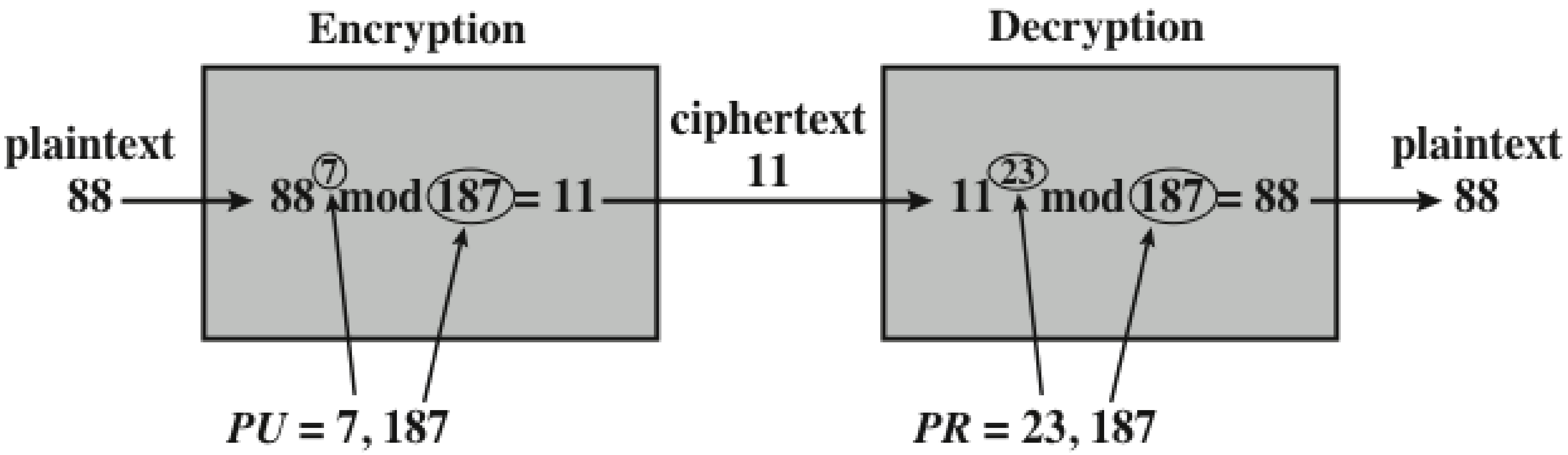


Figure 9.6 Example of RSA Algorithm

The Security of RSA

Five possible approaches to attacking RSA are:

Brute force
- Involves trying all possible private keys

Mathematical attacks
- There are several approaches, all equivalent in effort to factoring the product of two primes

Timing attacks
- These depend on the running time of the decryption algorithm

Hardware fault-based attack
- This involves inducing hardware faults in the processor that is generating digital signatures

Chosen ciphertext attacks
- This type of attack exploits properties of the RSA algorithm

Can RSA be brute forced

- Brute force attack is a generic attack that can be performed on RSA cryptosystem to find the private key. Brute force attack is a time-consuming attack due to the large sample space of possible keys that must search.

- **Why RSA hard to brute forced?** Brute force attack would not work as there are too many possible keys to work through. Also, this consumes a lot of time. Dictionary attack will not work in RSA algorithm as the keys are numeric and does not include any characters in it.

Can RSA be Mathematically Attacked

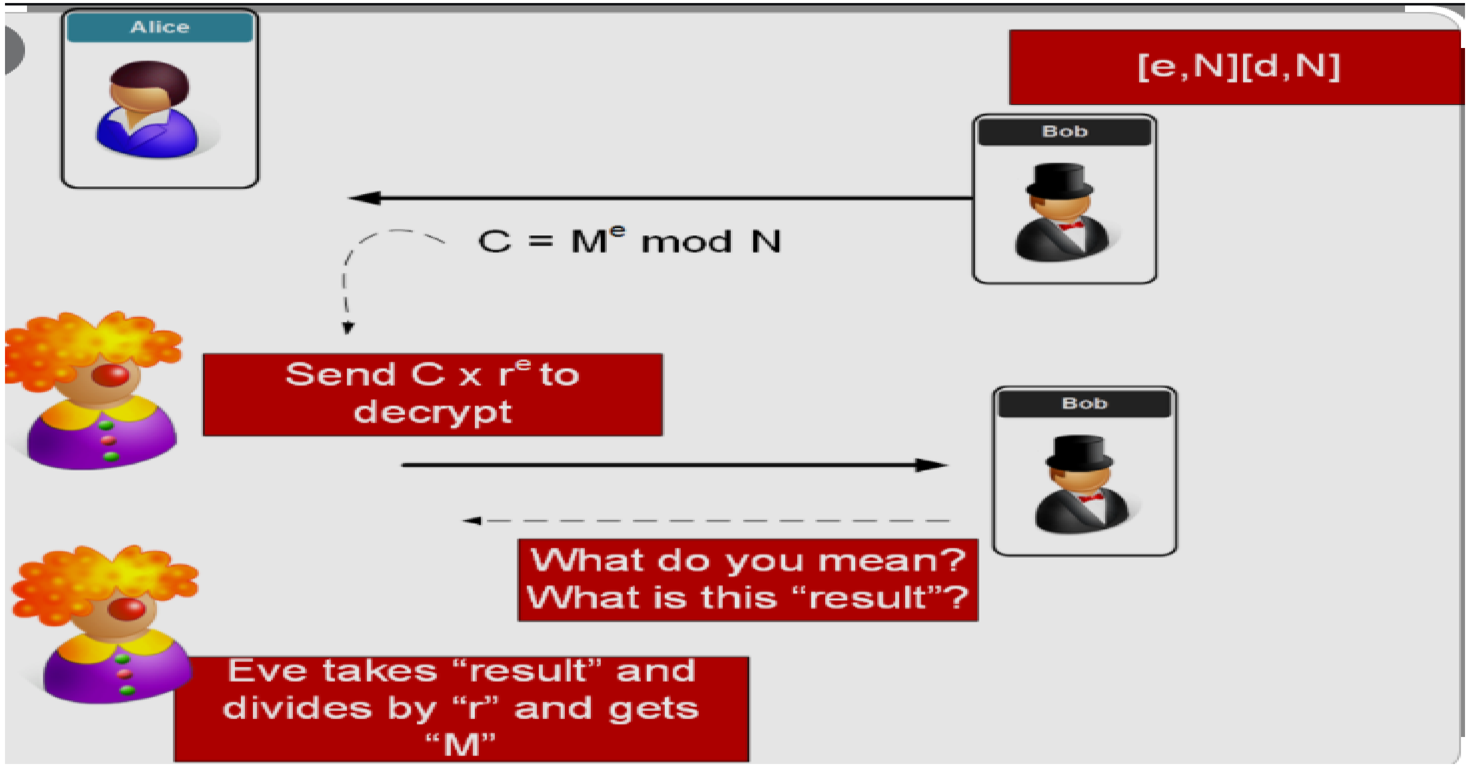
- What is Mathematical attack, A mathematical attack involves **the use of computation based on the mathematical properties of the encryption algorithm to attempt to decrypt data**.
- The Algorithm. The implementation of RSA makes heavy use of modular arithmetic, Euler's theorem, and Euler's totient function. Notice that each step of the algorithm only involves multiplication, so it is easy for a computer to perform: First, the receiver chooses two large prime numbers p p p and q q q.

Can RSA be Mathematically Attacked

- What is Mathematical attack, A mathematical attack involves **the use of computation based on the mathematical properties of the encryption algorithm to attempt to decrypt data**.
- The Algorithm. The implementation of RSA makes heavy use of modular arithmetic, Euler's theorem, and Euler's totient function. Notice that each step of the algorithm only involves multiplication, so it is easy for a computer to perform: First, the receiver chooses two large prime numbers p p p and q q q.

Chosen Cipher Attack in RSA

- The process of sending message between



The Diffie-Hellman Algorithm

- Discovered by Whitfield Diffie and Martin Hellman
    - "New Directions in Cryptography"

- 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

Step 1 –Publicly shared information

Alice & Bob publicly agree to a large **prime number** called the modulus, or ***p***.
Alice & Bob publicly agree to a number called the **generator**, or ***g***, which has a primitive root relationship with *p*.
In our example we'll assume

p = 17

g = 3

Eve is aware of the values of *p* or *g*.

Step 2 – Select a secret key

- Alice selects a secret key, which we will call ***a***.
- Bob selects a secret key, which we will call ***b***.
- For our example assume:
  - $a = 24$
  - $b = 54$
- Eve is **unaware** of the values of a or b.

Step 3 – Combine secret key with public information

- Bob combines his secret key of *b* with the public information to compute *B*.
- Public key = (**generator**$^{\text{secret key}}$)mod **prime number**
  - $B = g^b \bmod p$
  - $B = 3^{54} \bmod 17$
  - $B = 15$
- Alice combines his secret key of *a* with the public information to compute A.
- Public key = (**generator**$^{\text{secret key}}$)mod **prime number**
  - $A = g^a \bmod p$
  - $A = 3^{24} \bmod 17$
  - $A = 16$

Step 4 – Share combined values

- Alice shares her combined value, *A*, with Bob. Bob shares his combined value, *B*, with Alice.
- Sent to Bob
  - $A = 16$
- Sent to Alice
  - $B = 15$
- Eve is privy to this exchange and knows the values of *A* and *B*

Step 5 – Compute Shared Key

- Alice computes the shared key.
  - $s = (\text{public key B})^{a} \bmod p$
  - $s = 15^{24} \bmod 17$
  - $s = 1$
- Bob computes the shared key.
  - $s = (\text{public key A})^{b} \bmod p$
  - $s = 16^{54} \bmod 17$
  - $s = 1$

Alice & Bob have a shared encryption key, unknown to Eve

- Alice & Bob have created a shared secret key, s, unknown to Eve
- In our example s=1
- The shared secret key can now be used to encrypt & decrypt messages by both parties.

Alice & Bob have a shared encryption key, unknown to Eve

- Alice & Bob have created a shared secret key, s, unknown to Eve
- In our example s=1
- The shared secret key can now be used to encrypt & decrypt messages by both parties.

DIFFIE-HELLMAN KEY EXCHANGE
THE MATH: DISCRETE LOGARITHM PROBLEM

Let p be a large prime number

Let g be an integer < p

For every number n from 1 ... $(p - 1)$, inclusive, g must have a power k such that:

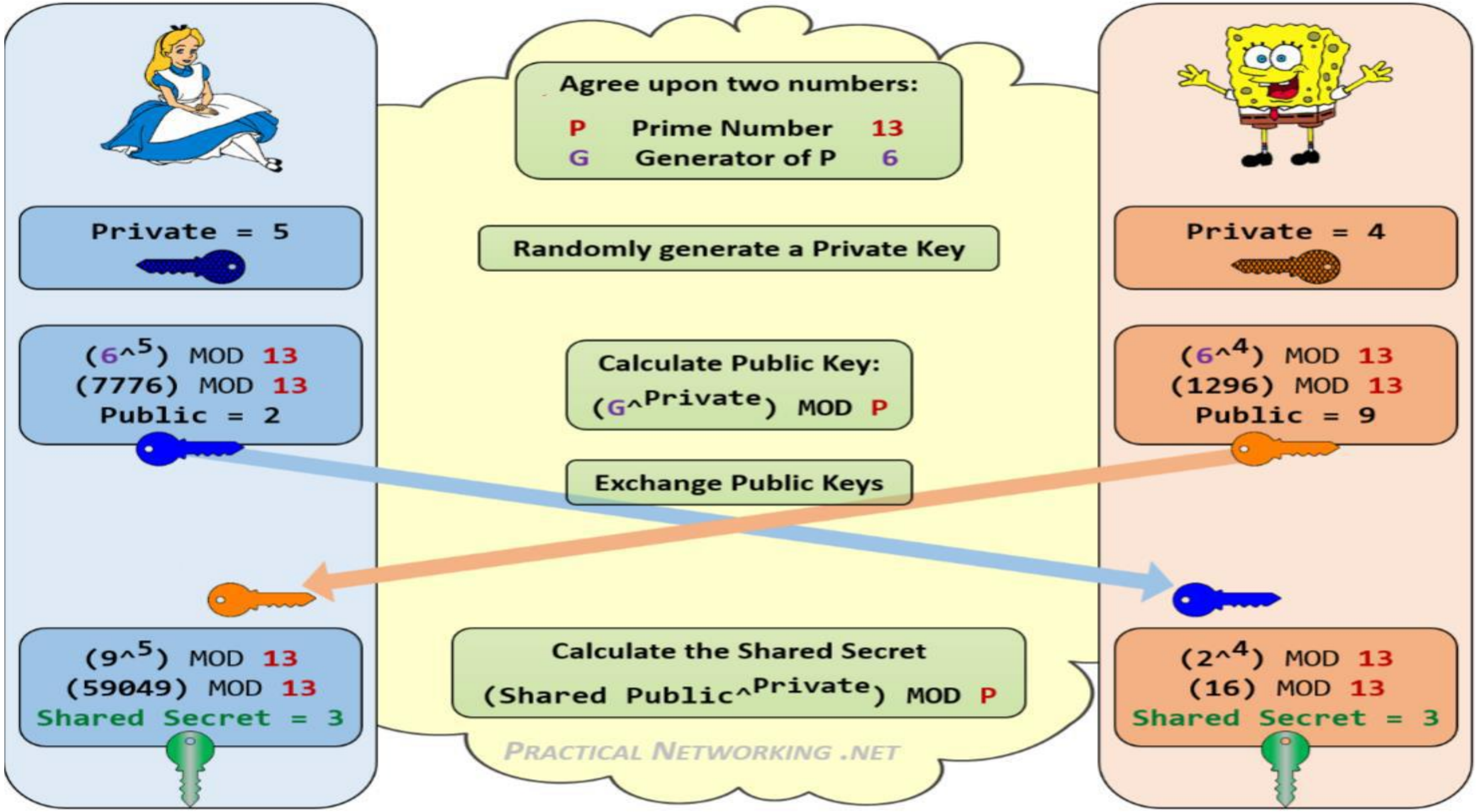
$$n = g^k \bmod p$$

$$6 = 3^{15}\ mod\ 17$$

$$6 = 3^{?}\ mod\ 17$$

- Solving the k^{th} root mod p is considered (but not proven) **hard to do in polynomial time**

Example #2



Agree upon two numbers:
P Prime Number 13
G Generator of P 6
Private = 5
Randomly generate a Private Key
Private = 4
(6^5) MOD 13
(7776) MOD 13
Public = 2
Calculate Public Key:
(G^Private) MOD P
(6^4) MOD 13
(1296) MOD 13
Public = 9
Exchange Public Keys
(9^5) MOD 13
(59049) MOD 13
Shared Secret = 3
Calculate the Shared Secret
(Shared Public^Private) MOD P
(2^4) MOD 13
(16) MOD 13
Shared Secret = 3
PRACTICAL NETWORKING .NET

Diffie Hellman Algorithm Weak-points

The parameter which is known to attacker is p , q, A, B as these parameters are exchanged on the public channel. So, to know the shared-secret key of the sender and the receiver, attacker would have to calculate the value of a and b which is known to only sender and receiver. So, it's tough for the attacker to get the secret key but not impossible. Plain text, Man-in- Middle attack, logjam attack and many more attacks which have found Diffie Hellman algorithm which is possible to be attacked.

Diffie Hellman Algorithm Man in the Middle

Alice

Generate random P=17 and random q=29
And random A=52
Calculate PublicKey= (29)^52Mod17=13

Calculate Secret Key= (PkM)^A modp
= (5)^52 mod 17
= 13

P=17,q=29, PK=13

PK=5

Man In Middle

Generate A'=46,B'=41
Calculate Pk based on received P,q
Pk=q^A'mod(p)
= 29^46 mod 17=1

Based on receive PK
Calculate secretKeyOne= (PK)^A'mod(p)
=(10)^46mod17=

Generate Pk for Alice
= (q)^B'mod(p)
=(29)^41 mod(17)=5
SecretKey= 8
(PkAlice)^B'mod(p)
=
(13)^41mod(17)= 13

P=17,q=29, PK=15

PK= 10

Bob

Generate B =63
Calculate PublicKey
Pk= q^B mod p
= (29)^63mod17=10
Calculate secretKey
= (Pk)^B mod(P)
=
(15)^63mod17= 8

El Gamal Cryptography

- public-key cryptosystem related to D-H
- uses exponentiation in a finite field
- with security based difficulty of computing discrete logarithms, as in D-H
- each user (eg. A) generates their key
  - chooses a secret key (number): $1 < x_A < q-1$
  - compute their **public key**: $y_A = a^{x_A} \bmod q$

Reference

1. Lecture slides prepared for “Cryptography and Network Security”, 7/e, by William Stallings. Chapter 1, “Computer and Network Security Concepts”.