Unit-2: Number Theory: The Euclidean Algorithm, Modular Arithmetic, Fermat's and Euler's Theorems, The Chinese Remainder Theorem, Discrete Logarithms, Finite Fields: Finite Fields of the Form GF(p), Finite Fields of the Form GF(2n) .Public Key Cryptography: Principles, Public Key Cryptography Algorithms, RSA Algorithm, Diffie Hellman Key Exchange, Elliptic Curve Cryptography.

---

1. Number Theory:

The Euclidean Algorithm:

One of the basic techniques of number theory is the Euclidean algorithm, which is a simple procedure for determining the **greatest common divisor** of two positive integers. First, we need a simple definition: Two integers are relatively prime if and only if their only common positive integer factor is 1.

Greatest Common Divisor (GCD)

A nonzero integer **b** is said to divide an integer **a** if

$$a = mb$$

for some integer **m**.
This is written as **b | a**.

The **greatest common divisor (gcd)** of two integers **a** and **b**, written as

$$\gcd(a, b),$$

is the **largest positive integer that divides both a and b**.

We define:

$$\gcd(0,0) = 0$$

Formal Definition of GCD

A positive integer **c** is the greatest common divisor of **a** and **b** if:

1. **c divides both a and b**.
2. **Every common divisor of a and b also divides c**.

Another way to define gcd is:

$$\gcd(a, b) = \max\{k \mid k \text{ divides both } a \text{ and } b\}$$

Example 1: Obtain GCD (12,33)

Q	A	B	R
2	33	12	9
1	12	9	3
3	9	3	0
X	3	0	X

A→ Dividend B→ Divisor R→Remainder Q→ Quotient

Therefore, GCD (12,33) = 3

Example 2: Obtain GCD (80808,31863)

Q	A	B	R
2	80808	31863	17082
1	31863	17082	14781
1	17082	14781	2301
6	14781	2301	975
2	2301	975	351
2	975	351	273
1	351	273	78
3	273	78	39
2	78	39	0
X	39	0	X

Therefore, GCD (80808,31863) = 39

Properties of GCD

• The gcd is always **positive**.
• Changing the sign of a number does not change the gcd:

$$\gcd(a,b) = \gcd(a,-b) = \gcd(-a,b) = \gcd(-a,-b)$$

Example:1)

$$\gcd(60,24) = \gcd(60,-24) = 12$$

Q	A	B	R
2	60	24	12
2	24	12	0
X	12	0	X

• Since every nonzero integer divides 0, we have:

$$\gcd(a,0) = |\,a\,|$$

Finding the Greatest Common Divisor:

1. Suppose we wish to determine the greatest common divisor d of the integers a and b; that is determine **d = gcd (a,b).** Because **gcd (|a|, |b|) = gcd (a,b),** there is no harm in assuming **a ≥ b> 0**
2. Dividing **a** by **b** and applying the division algorithm, we can state:

   a = q1b+r1 0 ≤ r1 <b ---- (1)
3. First consider the case in which **r1= 0**. Therefore, b divides a and clearly no larger number divides both **b** and **a,** because that number would be larger than **b.** So we have **d =gcd (a,b) = b**
4. The other possibility from **Equation (1)** is **r1≠0**. For this case, we can state that **d|r1.** This is due to the basic properties of divisibility: the relations **d|a** and **d|b** together imply that **d|(a−q1b),** which is the same as **d|r1**.
5. Before proceeding with the Euclidean algorithm, we need to answer the question: What is the **gcd (b,r1)?** We know that **d|b** and **d|r1**. Now take any arbitrary integer **c** that divides both **b** and **r1**. Therefore, **c|(q1b+r1)=a**. Because c divides both **a** and b, we must have **c≤d**, which is the greatest common divisor of a and b. Therefore, **d=gcd(b,r1)**

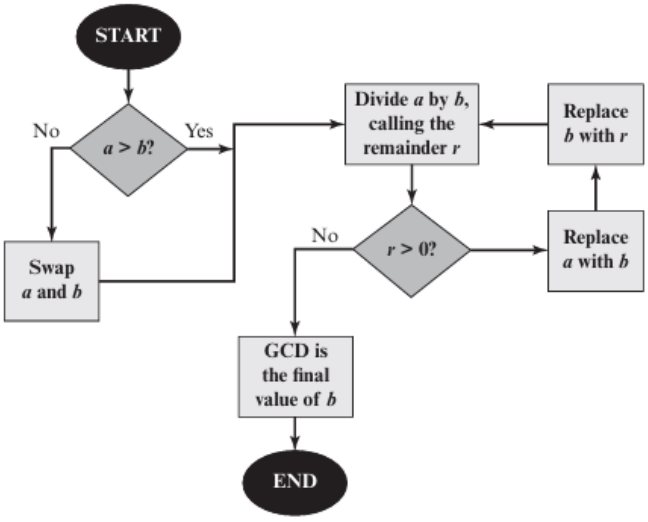


Figure: 1 Euclidean Algorithm

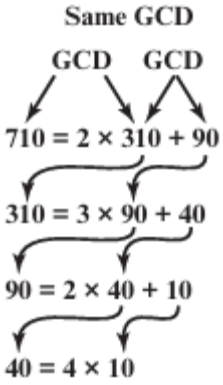


Figure: 2 Euclidean Algorithm Example GCD(710,310)

Let us now return to Equation (1) and assume that r1 ≠ 0. Because b >r1, we can divide b by r1 and apply the division algorithm to obtain:

$$\mathbf{b = q2r1 + r2 \qquad 0 \le r2 < r1}$$

If r2 = 0, then d = r1 and if r2 ≠ 0, then d = gcd(r1, r2).

$$\left.\begin{array}{ll} a = q_1 b + r_1 & 0 < r_1 < b \\ b = q_2 r_1 + r_2 & 0 < r_2 < r_1 \\ r_1 = q_3 r_2 + r_3 & 0 < r_3 < r_2 \\ \quad\vdots & \quad\vdots \\ r_{n-2} = q_n r_{n-1} + r_n & 0 < r_n < r_{n-1} \\ r_{n-1} = q_{n+1} r_n + 0 & \end{array}\right\}$$
$$d = \gcd(a, b) = r_n$$

$$\boxed{gcd(a,b) = r_n}$$

At each iteration, we have d = gcd (r_i, r_{i+1}) until finally d = gcd(r_n, 0) = r_n.

$$r_i = q_{i+2} \times r_{i+1} + r_{i+2}$$

Thus, we can find the greatest common divisor of two integers by repetitive application of the division algorithm. This scheme is known as the **Euclidean algorithm**.

Example: Obtain gcd(80808,31863)

Sol:

$a = q_1 b + r_1$

80808 = 2×31863 + 17082

31863 = 1×17082 + 14781

17082 = 1×14781 +2301

14781 = 6×2301 +975

2301 = 2×975 +351

975 = 2×351 +273

351 = 1×273 +78

273 = 3×78 +39

78 = 2×39 + 0 --- stop here

Therefore, Gcd (80808, 31863) = 39

2. Modular Arithmetic:

1.The Modulus:

If **a** is an integer and **n** is a positive integer, we define a mod n to be the remainder when **a** is divided by **n**. The integer **n** is called the **modulus.**

- Thus, for any integer a, we can rewrite Equation as follows:

$$a = qn + r \qquad 0 \le r < n; q = \lfloor a/n \rfloor$$
$$a = \lfloor a/n \rfloor \times n + (a \bmod n)$$

Ex:

$$11 \bmod 7 = 4; \qquad -11 \bmod 7 = 3$$

2.Modulus Operation:

The expression **a mod n** means the **remainder when integer a is divided by n**.

Ex:

$$17 \bmod 5 = 2 \quad \text{because}$$

$$17 = 5 \times 3 + 2$$

Thus the remainder is **2**.

3. Congruence Relation

Two integers are **congruent modulo n** if they leave the same remainder when divided by n.

Two integers a and b are said to be congruent modulo n, if (a mod n) = (b mod n). This is written as $a \equiv b \pmod{n}$

Congruences have the following properties

1. $a \equiv b \pmod{n}$ if $n \mid (a - b)$.
2. $a \equiv b \pmod{n}$ implies $b \equiv a \pmod{n}$.
3. $a \equiv b \pmod{n}$ and $b \equiv c \pmod{n}$ imply $a \equiv c \pmod{n}$.

4. Properties of Modular Arithmetic

Modular arithmetic exhibits the following properties:

1. $[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$
2. $[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$
3. $[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$

Modular arithmetic follows similar rules to ordinary arithmetic.

Here are examples of the three properties:

$$11 \bmod 8 = 3; 15 \bmod 8 = 7$$
$$[(11 \bmod 8) + (15 \bmod 8)] \bmod 8 = 10 \bmod 8 = 2$$
$$(11 + 15) \bmod 8 = 26 \bmod 8 = 2$$
$$[(11 \bmod 8) - (15 \bmod 8)] \bmod 8 = -4 \bmod 8 = 4$$
$$(11 - 15) \bmod 8 = -4 \bmod 8 = 4$$
$$[(11 \bmod 8) \times (15 \bmod 8)] \bmod 8 = 21 \bmod 8 = 5$$
$$(11 \times 15) \bmod 8 = 165 \bmod 8 = 5$$

Exponentiation is performed by repeated multiplication, as in ordinary arithmetic.

To find $11^7 \bmod 13$, we can proceed as follows:

$11^2 = 121 \equiv 4 \pmod{13}$

$11^4 = (11^2)^2 \equiv 4^2 \equiv 3 \pmod{13}$

$11^7 = 11 \times 11^2 \times 11^4$

$11^7 \equiv 11 \times 4 \times 3 \equiv 132 \equiv 2 \pmod{13}$

Thus, the rules for ordinary arithmetic involving addition, subtraction, and multiplication carry over into modular arithmetic.

5.Properties of Modular Arithmetic for Integers in Zn:

Property	Expression
Commutative Laws	$(w + x) \bmod n = (x + w) \bmod n$ $(w \times x) \bmod n = (x \times w) \bmod n$
Associative Laws	$[(w + x) + y] \bmod n = [w + (x + y)] \bmod n$ $[(w \times x) \times y] \bmod n = [w \times (x \times y)] \bmod n$
Distributive Law	$[w \times (x + y)] \bmod n = [(w \times x) + (w \times y)] \bmod n$
Identities	$(0 + w) \bmod n = w \bmod n$ $(1 \times w) \bmod n = w \bmod n$
Additive Inverse $(-w)$	For each $w \in Z_n$, there exists a z such that $w + z \equiv 0 \bmod n$

6.Importance of Modular Arithmetic in Cryptography

Modular arithmetic is the **foundation of many cryptographic algorithms**, including:

- RSA
- Diffie–Hellman key exchange
- AES finite field operations
- Elliptic curve cryptography

It allows computations to remain within a **finite set of numbers**, which is essential for secure cryptographic systems

Extended Euclidean Algorithm:

Given two integers ***a*** and ***b*** **,** we often need to find other two integers, ***s*** and ***t***, such that

$s \times a + t \times b = gcd\ (a, b)$

Example : gcd (161, 28) = 7, $s = -1$ and $t = 6$.

$(-1) \times 161 + 6 \times 28 = 7$

The extended Euclidean algorithm can calculate the ***gcd (a, b)*** and at the same time calculate the value of ***s*** and ***t.***

Extended Euclidean algorithm

$r_1 \leftarrow a;$ $r_2 \leftarrow b;$
$s_1 \leftarrow 1;$ $s_2 \leftarrow 0;$ (Initialization)
$t_1 \leftarrow 0;$ $t_2 \leftarrow 1;$

while $(r_2 > 0)$
{
$q \leftarrow r_1 / r_2;$

$r \leftarrow r_1 - q \times r_2;$
$r_1 \leftarrow r_2;\ r_2 \leftarrow r;$ (Updating r's)

$s \leftarrow s_1 - q \times s_2;$
$s_1 \leftarrow s_2;\ s_2 \leftarrow s;$ (Updating s's)

$t \leftarrow t_1 - q \times t_2;$
$t_1 \leftarrow t_2;\ t_2 \leftarrow t;$ (Updating t's)

}
gcd $(a, b) \leftarrow r_1;\ s \leftarrow s_1;\ t \leftarrow t_1$

Ex:

Given a = 161 and b = 28, find gcd (a, b) and the values of s and t.

q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	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
	7	0		−1	4		6	−23	

i	r_i	q_i	s_i	t_i
0	161		1	0
1	28	5	0	1
2	21	1	1	-5
3	7	3	-1	6
4	0			

We get gcd (161, 28) = 7, $s = -1$ and $t = 6$.

$$s \times a + t \times b = \gcd(a, b)$$

3. Fermat's and Euler's Theorems

Two theorems that play important roles in public-key cryptography are Fermat's theorem and Euler's theorem.

Fermat's theorem:

Fermat's theorem states the following: If p is prime and a is a positive integer not divisible by p, then

$$a^{p-1} \equiv 1 \pmod{p}$$

This theorem is very important in **cryptography**, especially in algorithms such as **RSA** and modular arithmetic operations.

Proof:

Consider a Set of Integers. Take all **positive integers less than p**:

$$S = \{1, 2, 3, ..., p-1\}$$

This set contains **(p − 1) integers**.

Multiply each element of the set by **a** and take **mod p**.

$$X = \{a \bmod p, 2a \bmod p, 3a \bmod p, ..., (p-1)a \bmod p\}$$

This creates a **new set X**.

If an element were **0 mod p**, then:

$$ka \equiv 0 \pmod{p}$$

This would mean **p divides ka**.

But **p is prime and does not divide a**, therefore **p must divide k**. Since **k < p**, this is impossible.

Therefore: **No element of X is equal to 0.**

Assume two elements of X are equal:

$$ja \equiv ka \pmod{p}$$

Where

$$1 \leq j < k \leq p-1$$

Since **a is relatively prime to p**, we can cancel **a** from both sides.

Thus:

$$j \equiv k \pmod{p}$$

But **j and k are positive integers less than p**, so this equality is impossible.

Therefore:**All elements of X are distinct.**

Relationship Between Sets S and X; We know:

- X has **(p − 1) elements**
- None are **0**

- No two are **equal**

Therefore, X must contain exactly the same elements as S, but **possibly in a different order**.

Thus,

$$X = \{1, 2, 3, ..., p-1\}$$

Multiply all elements of set **S**:

$$1 \times 2 \times 3 \times ... \times (p-1)$$

Multiply all elements of set **X**:

$$(a)(2a)(3a)...((p-1)a)$$

$$= a^{p-1}(1 \times 2 \times 3 \times ... \times (p-1))$$

Take Modulo p;

$$1 \cdot 2 \cdot 3 \cdots (p-1) \equiv a^{p-1}(1 \cdot 2 \cdot 3 \cdots (p-1)) \pmod{p}$$

Since

$$1 \cdot 2 \cdot ... \cdot (p-1)$$

is relatively prime to **p**, it can be cancelled.

Thus,

$$\boxed{a^{p-1} \equiv 1 \pmod{p}}$$

This completes the **proof of Fermat's Little Theorem**.

Ex:1

Using Fermat's theorm, check whether 7 is prime or not? consider a=3.

Soln: **Given**

p=7, a=3

$$a^{p-1} \equiv 1 \pmod{p}$$

Substitute the values:

$$3^{7-1} \equiv 1 \pmod{7}$$

$$3^6 \equiv 1 \pmod{7}$$

$3^6 \bmod 7 = 34.32 \bmod 7$

$= 81.9 \bmod 7$

$=4.2 \bmod 7$

$=8 \bmod 7 = 1$

$3^6 \bmod 7 \equiv 1$

Therefore, 7 is a prime.

Ex: 2 : Using Fermat's Theorem solve a=7, p=19

Soln:

$$a^{p-1} \equiv 1 \pmod{p}$$

$$7^{19-1} \equiv 1 \pmod{19}$$

$$7^{18} \equiv 1 \pmod{19}$$

$$7^{18} = (7^3)^6$$

$$7^{18} \equiv 1^6 \equiv 1 \pmod{19}$$

$$7^{18} \equiv 1 \pmod{19}$$

$$7^{18} \bmod 19 = 1$$

The condition of Fermat's theorem is satisfied. Hence, **19 behaves as a prime number according to Fermat's Little Theorem**.

Euler's Totient Function:

The **Euler's totient function**, usually written as **φ(n)** (phi of n), is a function in **Number Theory** that counts how many positive integers **less than or equal to n** are **coprime with n** (their GCD with n is 1).

The function is named after the mathematician **Leonhard Euler**.

For a positive integer **n**,
φ(n) = number of integers k such that 1 ≤ k ≤ n and

gcd(k,n) = 1

It is fundamental in number theory and RSA cryptography, used to determine the count of coprime numbers efficiently.

Criteria of n	Formula for φ(n)
If **n is prime**	$\varphi(n) = n - 1$
If **n = p × q**, where **p and q are prime numbers**	$\varphi(n) = (p - 1)(q - 1)$
If **n = a × b**, where **a or b may be composite**	$\varphi(n) = n \times (1 - 1/p_1)(1 - 1/p_2) \ldots$
When $p_1, p_2, \ldots$ **are distinct prime factors of n**	$\varphi(n) = n \times \prod (1 - 1/p_i)$

Example 1: When n is Prime

Find **φ(13)**.

Since **13 is a prime number**, the formula is:

$$\phi(n) = n - 1$$

$$\phi(13) = 13 - 1 = 12$$

Answer: φ(13) = **12**

Example 2: When n = p × q (p and q are primes)

Find **φ(15).**

Prime factorization:

Formula:

$$\phi(n) = (p - 1)(q - 1)$$

$$\phi(15) = (3 - 1)(5 - 1)$$

$$= 2 \times 4 = 8$$

Answer: φ(15) = **8**

Example 3: General Formula (Distinct Prime Factors)

Find **φ(20).**

Prime factorization:

$$20 = 2^2 \times 5$$

P1 = 2, p2 = 5
Formula:

$$\phi(n) = n\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right)$$

$$\phi(20) = 20\left(1 - \frac{1}{2}\right)\left(1 - \frac{1}{5}\right)$$

$$= 20 \times \frac{1}{2} \times \frac{4}{5}$$

$$= 8$$

Answer: **φ(20) = 8**

Euler's Theorem:

Euler's theorem is a fundamental result in modular arithmetic that generalizes Fermat's little theorem.

Statement

If n is a positive integer and a is an integer such that **gcd(a, n) = 1** (i.e., a and n are coprime), then:

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

where $\phi(n)$is the Euler's Totient function **(**number of positive integers less than or equal to n that are coprime to n**).**

- The theorem says that raising a to the power $\phi(n)$**,** then dividing by n, leaves a remainder of **1**.
- This works only when a and n share no common factors other than **1**.

Proof:

Consider the set of integers R such that integers are less than n,

$$R = \{x_1, x_2, \ldots, x_{\phi(n)}\}$$

After considering these set of integers, perform multiplication operation with a **modulo** n operation with n for each and every term**.**

$$S = \{ax_1 \bmod n, ax_2 \bmod n, \ldots, ax_{\phi(n)} \bmod n\}$$

By observing $\boldsymbol{R}$ and $\boldsymbol{S}$, we identify a relation that is $\boldsymbol{S}$ is a permutation of $\boldsymbol{R}$.

$\boldsymbol{S}$ is relatively prime to $\boldsymbol{n}$.
$\boldsymbol{x_i}$ is also relatively prime to $\boldsymbol{n}$.

Therefore, $\boldsymbol{ax_i}$ is also relatively prime to $\boldsymbol{n}$.

Therefore,

$$\prod_{i=1}^{\phi(n)} ax_i \bmod n = \prod_{i=1}^{\phi(n)} x_i \bmod n$$

or,

$$a^{\phi(n)} \times \prod_{i=1}^{\phi(n)} x_i \equiv \prod_{i=1}^{\phi(n)} x_i \pmod{n}$$

which implies

$$\boxed{a^{\phi(n)} \equiv 1 \pmod{n}}$$

4. The Chinese Remainder Theorem: (CRT)

The Chinese Remainder Theorem provides a way to solve a system of simultaneous congruences with different moduli, given that the moduli are pairwise coprime (i.e., their greatest common divisor is 1).

Statement:

Suppose you have the system:

$$\begin{cases} x \equiv c_1 \pmod{n_1} \\ x \equiv c_2 \pmod{n_2} \\ \quad\vdots \\ x \equiv c_k \pmod{n_k} \end{cases}$$

where $\boldsymbol{n_1, n_2, \ldots, n_k}$ are pairwise coprime (for every $\boldsymbol{i \neq j}$, $\boldsymbol{\gcd(n_i, n_j) = 1}$).

Then there exists a unique solution modulo $N = n_1 \times n_2 \times \cdots \times n_k$, i.e., there is exactly one $x$such that:

$$\mathbf{0 \leq X < N}$$

and $\boldsymbol{X}$ satisfies all the congruences simultaneously

The unique solution modulo *N*is:

$$\boxed{X \equiv \sum_{i=1}^{k} c_i \cdot N_i \cdot d_i \pmod{N}}$$

Where:

- c_i are your remainders
- $N_i = N/n_i$
- d_i is the modular inverse of $N_i \bmod n_i$

$$N_i = \frac{N}{n_i} \quad \text{for } i = 1, 2, \ldots, k$$

$$\boxed{X = (c_1 N_1 d_1 + c_2 N_2 d_2 + \cdots + c_k N_k d_k) \bmod N}$$

modular inverse of $N_i \bmod n_i$, call it d_i

$$N_i \cdot d_i \equiv 1 \pmod{n_i}$$

$$\boxed{d_i \equiv N_i^{-1} \pmod{n_i}}$$

Where

$$\boxed{N = n_1 \times n_2 \times \cdots \times n_k}$$

$$\begin{aligned} N_1 &= \frac{N}{n_1} \\ N_2 &= \frac{N}{n_2} \\ &\vdots \\ N_k &= \frac{N}{n_k} \end{aligned} \qquad \left| \qquad \begin{aligned} d_1 &\equiv N_1^{-1} \pmod{n_1} \\ d_2 &\equiv N_2^{-1} \pmod{n_2} \\ &\vdots \\ d_k &\equiv N_k^{-1} \pmod{n_k} \end{aligned} \right.$$

Example 1: solve the following using Chinese Remainder Theorem (CRT)

X= 2 mad 3
X= 3 mod 5
X= 2 mod 7
Soln:

Step1:

Co-primes n_1, n_2, n_3 are 3,5,7

Step 2:

Unique solution Modulo $N = n_1 \times n_2 \times n_3$

$= 3 \times 5 \times 7$

$N = 105$

Step 3: Remainder residues

$$c_1 = 2$$
$$c_2 = 3$$
$$c_3 = 2$$

Step 4:

$$N_1 = \frac{N}{n_1}$$

=105/3

N_1 =35

$$d_1 \equiv N_1^{-1} \pmod{n_1}$$

$d_1=35^{-1}$ mod 3 = 2

$$N_2 = \frac{N}{n_2}$$

=105/5

N_2=21

$$d_2 \equiv N_2^{-1} \pmod{n_2}$$

$d_2 = 21^{-1}$ mod 5 =1

lly, N_3 = 105/7

N_3 = 15

lly, $d_3=15^{-1}$ mod 7 = 1

Step 6:

$$X = (c_1 N_1 d_1 + c_2 N_2 d_2 + \cdots + c_k N_k d_k) \bmod N$$

= (2×35×2)+(3×21×1)+(2×15×1))

X = 233

Step 7:

X mod N = 233 mod 105

X= 23

5.Discrete Logarithms:

- In modular arithmetic, exponentiation is easy:

$$y = a^x \mod p$$

T

- he reverse problem (finding xxx) is difficult:

$$x = \log_a y \mod p$$

This is called the **Discrete Logarithm Problem (DLP).**
Definition: For a prime number p and base a:

$$a^x \equiv b \pmod p$$

Then:

$$x = \log_a b \pmod p$$

- a → base (generator)
- b → result
- x → discrete logarithm

Example:1 Find x:

$$2^x \equiv 7 \pmod{11}$$

Check powers:

- $2^1 = 2$
- $2^2 = 4$
- $2^3 = 8$
- $2^4 = 16 \equiv 5$
- $2^5 = 10$
- $2^6 = 20 \equiv 9$
- $2^7 = 18 \equiv 7$

$$x = 7$$

6.Public Key Cryptography: Principles

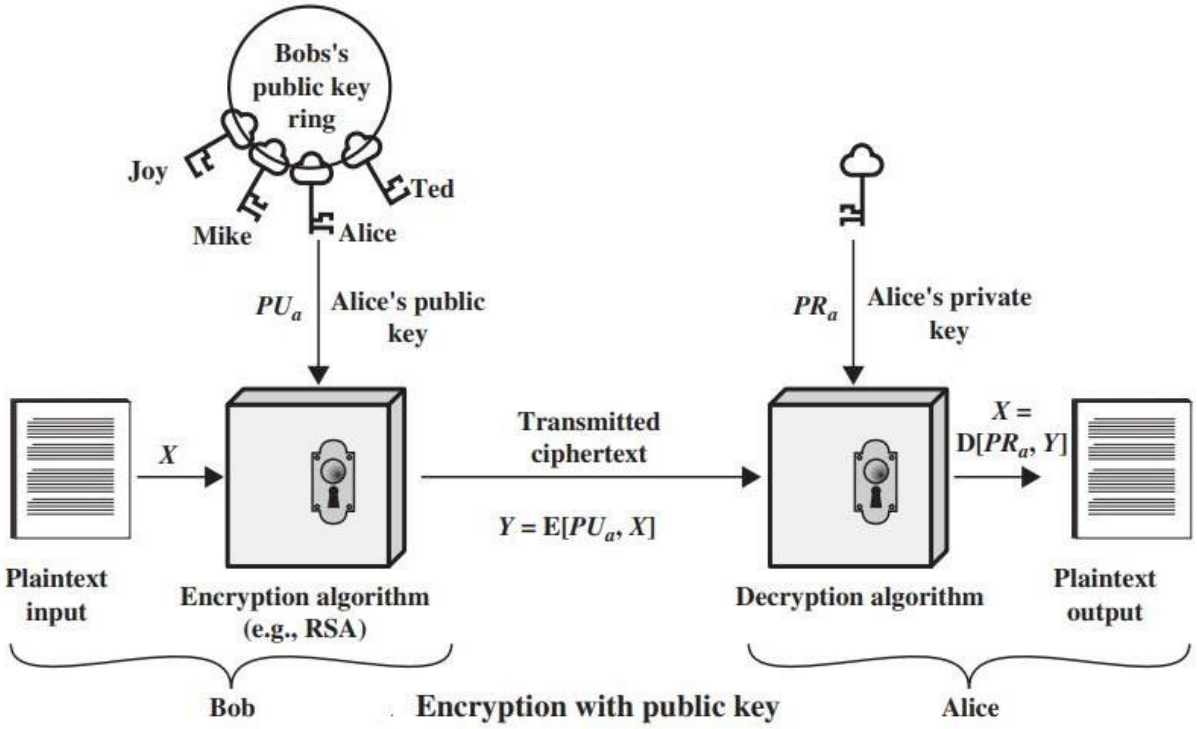
- Asymmetric encryption is a form of cryptosystem in which encryption and decryption are performed using the different keys—one a public key and one a private key. It is also known as public-key encryption.
- Asymmetric encryption transforms plaintext into ciphertext using a one of two keys and an encryption 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 Cryptosystems

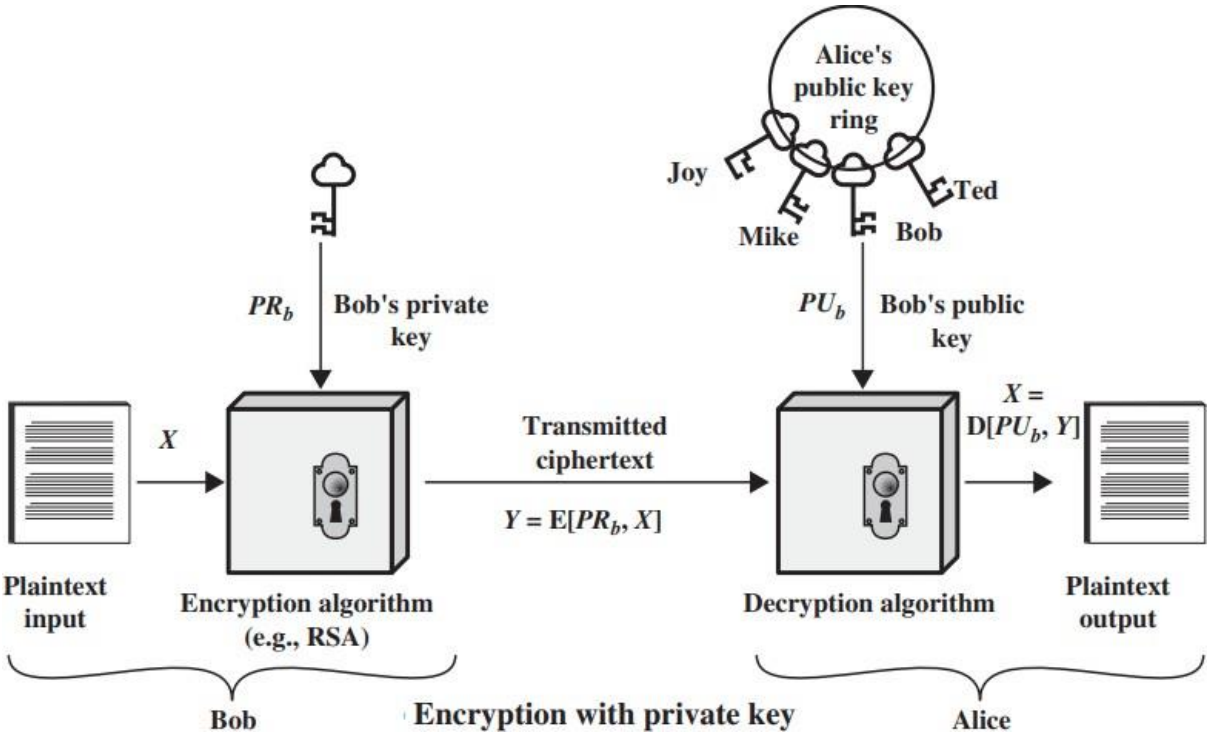
Asymmetric algorithms rely on one key for encryption and a different but related key for decryption. A public-key encryption scheme has six ingredients as shown in below:

- **Plaintext:** This is the readable message or data that is fed into the algorithm as input.
- **Encryption algorithm:** The encryption algorithm performs various transformations on the plaintext.
- **Public and private keys:** This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the algorithm depend on the public or private key that is provided as input.
- **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertext.
- **Decryption algorithm:** This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

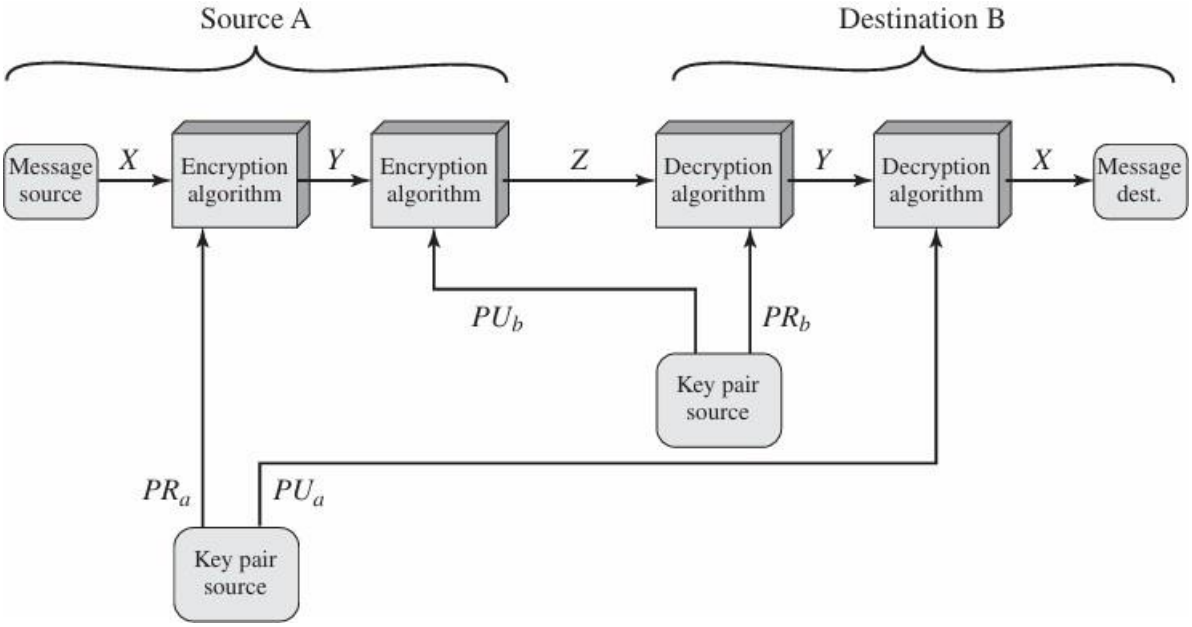
The essential elements of a public-key encryption scheme, as follows in which some source can perform encryption with receiver's public key and decryption with private key by the receiver that provides confidentiality as shown in bellow:



Source can perform encryption with private key and decryption with source's public key by the receiver that provides authentication as shown in bellow:



To provide both the authentication function and confidentiality by a double use of the public-key scheme as follows:



The essential steps are the following:

- 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.
- 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.

Difference between 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 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.

The use of public-key cryptosystems can classify 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. 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.

Requirements for Public-Key Cryptography:

1. It is computationally easy for a party B to generate a pair (public key PU_b, private key PR_b).
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: $C = E(PU_b, M)$
3. It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message: $M = D(PR_b, C) = D[PR_b, E(PU_b, M)]$
4. It is computationally infeasible for an adversary, knowing the public key, PU_b, to determine the private key, PR_b.
5. It is computationally infeasible for an adversary, knowing the public key,

PU_b, and a ciphertext, C, to recover the original message, M.

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

7.RSA Algorithm:

RSA was invented by three scholars **Ron Rivest, Adi Shamir, and Len Adleman** and hence, it is termed as **RSA** cryptosystem. There are two aspects of the RSA cryptosystem,

- firstly generation of key pair and
- Secondly encryption-decryption algorithms.

RSA scheme is a block 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. That is, n is less than 21024.

Description of the 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.

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

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. Thus, this is a public- key encryption algorithm with a public key of PU = {e, n} and a private key of PR = {d, n}. The RSA algorithm summarizes as:

Key Generation

Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calcuate $\phi(n) = (p-1)(q-1)$	
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:	$M = C^d \bmod n$

For example, the keys were generated as follows:

1. Select two prime numbers, $p = 17$ and $q = 11$.
2. Calculate $n = pq = 17 \times 11 = 187$.
3. Calculate $\phi(n) = (p-1)(q-1) = 16 \times 10 = 160$.
4. Select e such that e is relatively prime to $\phi(n) = 160$ and less than $\phi(n)$; choose $e = 7$.
5. Determine d such that $de \equiv 1 \pmod{160}$ and $d < 160$. The correct value is $d = 23$, because $23 \times 7 = 161 = (1 \times 160) + 1$

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 M = 88.

For encryption, we need to calculate

$C = 88^7 \bmod 187$

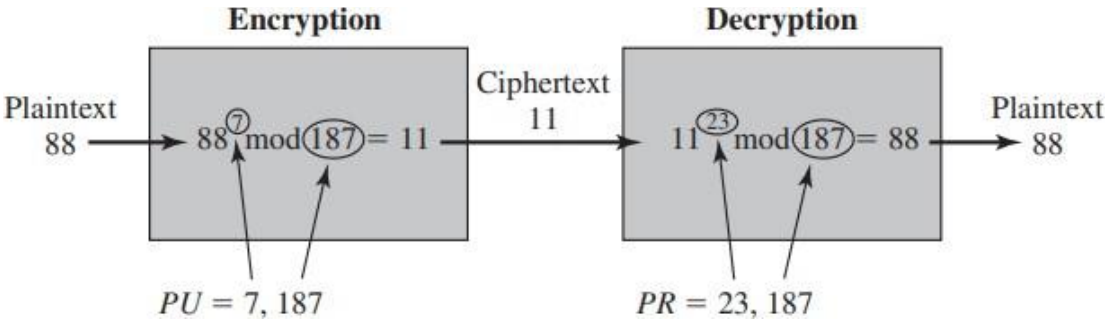
$= 11$

For decryption, we calculate M =

$11^{23} \bmod 187$

$=88$

As shown in below:



The Security of RSA:

Four possible approaches to attacking the RSA algorithm are

- **Brute force:** This 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.
- **Chosen ciphertext attacks:** This type of attack exploits properties of the RSA algorithm.

8.Diffie- Hellman Key Exchange:

A simple public-key algorithm is Diffie-Hellman key exchange. This protocol enables two users to establish a secret key using a public-key scheme based on discrete logarithms. The protocol is secure only if the authenticity of the two participants can be established.

The first published public-key algorithm by Diffie and Hellman that defined public-key cryptography and is generally referred to as Diffie-Hellman key exchange. The purpose of the algorithm is to enable two users to securely exchange a key that can then be used for subsequent encryption of messages. The algorithm itself is limited to the exchange of secret values.

Algorithm:

The Diffie-Hellman key exchange algorithm summarizes as

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

User B Key Generation	
Select private X_B	$X_B < q$
Calculate public Y_B	$Y_B = \alpha^{XB} \bmod q$

Calculation of Secret Key by User A
$K = (Y_B)^{XA} \bmod q$

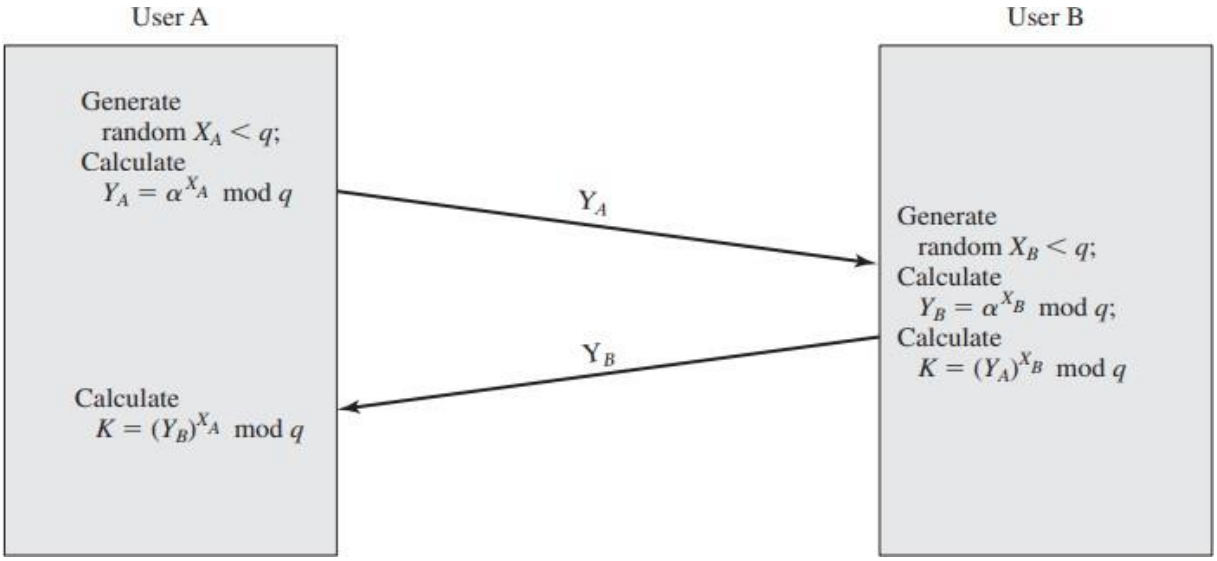
Calculation of Secret Key by User B
$K = (Y_A)^{XB} \bmod q$

For this scheme, there are two publicly known numbers: a prime number q and an integer α that is a primitive root of q . Suppose the users A and B wish to exchange a key. User A selects a random integer XA < q and computes YA = α^{XA} mod q.

Similarly, user B independently selects a random integer XB < q and computes YB = α^{XB} mod q and computes. Each side keeps the value private and makes the value available publicly to the other side. User A computes the key as K=$(YB)^{XA}$ mod q and user B computes the key as K=$(YA)^{XB}$ mod q . These two calculations produce identical results.

Key Exchange Protocols

A simple protocol that makes use of the Diffie-Hellman calculation as follows:



Suppose that user A wishes to set up a connection with user B and use a secret key to encrypt messages on that connection. User A can generate a one-time private key XA, calculate YA, and send that to user B. User B responds by generating a private value XB , calculating YB, and send to user A. Both users can now calculate the key.

EX: Take Alice and Bob as users

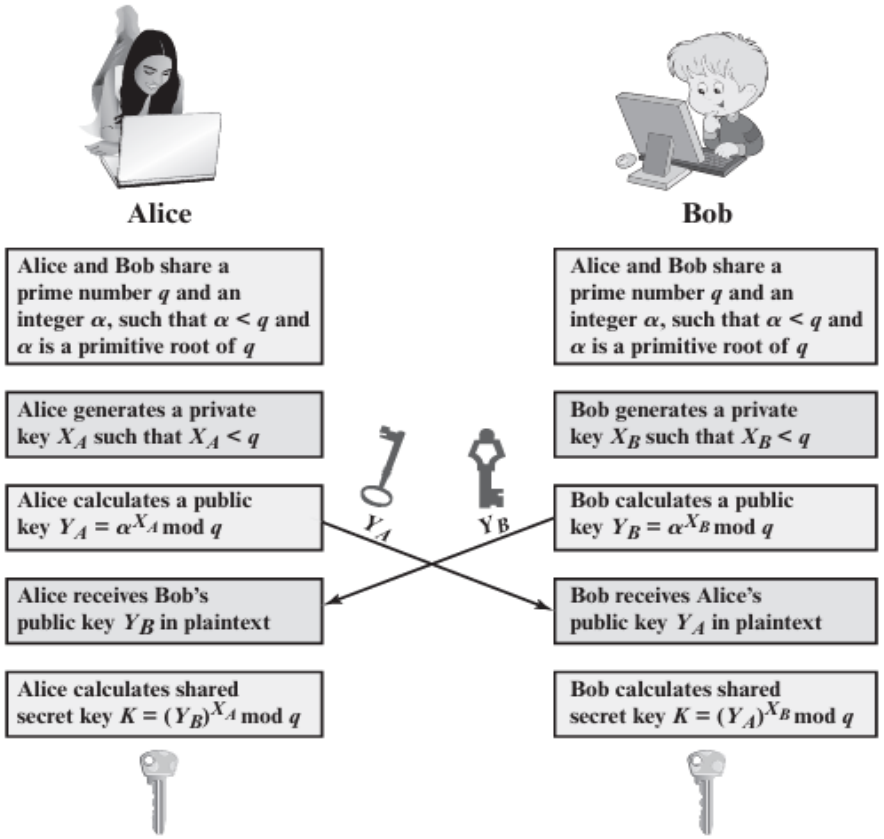


Figure 3: Diffie-Hellman Key exchange

Why both keys are same?

$$
\begin{aligned}
K &= (Y_B)^{X_A} \bmod q \\
&= (\alpha^{X_B} \bmod q)^{X_A} \bmod q \\
&= (\alpha^{X_B})^{X_A} \bmod q && \text{by the rules of modular arithmetic} \\
&= \alpha^{X_B X_A} \bmod q \\
&= (\alpha^{X_A})^{X_B} \bmod q \\
&= (\alpha^{X_A} \bmod q)^{X_B} \bmod q \\
&= (Y_A)^{X_B} \bmod q
\end{aligned}
$$

Example Problem: Two users, Alice and Bob, agree to use Diffie–Hellman key exchange.

- Prime number $q = 23$
- Primitive root $\alpha = 5$
- Alice chooses private key $X_A = 6$
- Bob chooses private key $X_B = 15$

Find the **shared secret key**.

Step 1: Compute Public Keys

Alice computes:

$$Y_A = \alpha^{X_A} \mod q = 5^6 \mod 23$$

$$5^6 = 15625 \Rightarrow 15625 \mod 23 = 8$$

So,
Alice sends

$$\boxed{Y_A = 8}$$

Bob computes:

$$Y_B = \alpha^{X_B} \mod q = 5^{15} \mod 23$$

$$5^{15} \mod 23 = 19$$

So,
Bob sends

$$\boxed{Y_B = 19}$$

Step 2: Compute Shared Secret Key

Alice computes:

$$K = (Y_B)^{X_A} \mod q = 19^6 \mod 23$$

$$19^6 \mod 23 = 2$$

Bob computes:

$$K = (Y_A)^{X_B} \mod q = 8^{15} \mod 23$$

$$8^{15} \mod 23 = 2$$

$$\boxed{Shared\ Secret\ Key = 2}$$

Man-in-the-Middle Attack:

The protocol depicted in above Figure is insecure against a man-in-the-middle attack. Suppose Alice and Bob wish to exchange keys, and Darth is the adversary. The attack proceeds as follows

1. 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}.
2. Alice transmits Y_A to Bob.
3. Darth intercepts Y_A and transmits Y_{D1} to Bob. Darth also calculates $K2 = (Y_A)^{X_{D2}} \mod q$.
4. Bob receives Y_{D1} and calculates $K1 = (Y_{D1})^{X_B} \mod q$.
5. Bob transmits Y_B to Alice.
6. Darth intercepts Y_B and transmits Y_{D2} to Alice. Darth calculates $K1 = (Y_B)^{X_{D1}} \mod q$.
7. Alice receives Y_{D2} and calculates $K2 = (Y_{D2})^{X_A} \mod q$.

At this point, Bob and Alice think that they share a secret key, but instead Bob and Darth share secret key K_1 and Alice and Darth share secret key K_2. All future communication between Bob and Alice is compromised in the following way:

1. Alice sends an encrypted message $M : E(K2, M)$.
2. Darth intercepts the encrypted message and decrypts it to recover M.
3. 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.

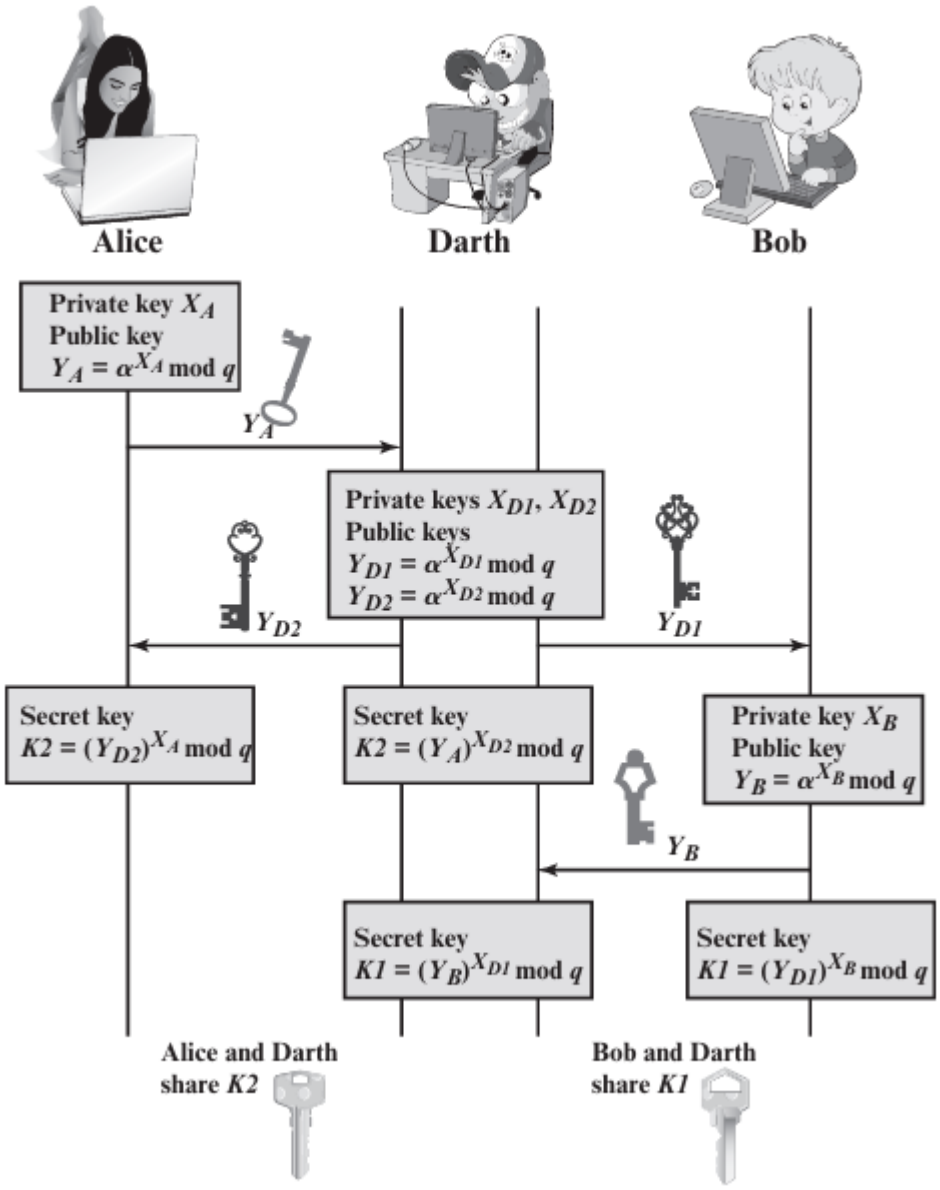


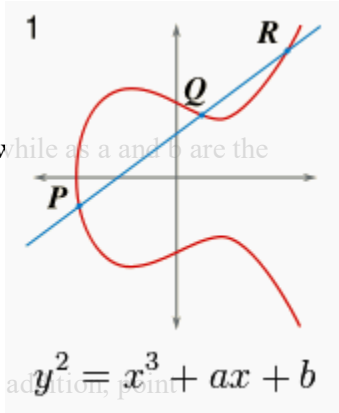
Figure 4: Man-in-the-Middle Attack

9.Elliptic Curve Cryptographic algorithm (ECC):

- Elliptic Curve Cryptography (ECC) was discovered in 1985 by Victor Miller (IBM) and Neil Koblitz (University of Washington) as an alternative mechanism for implementing public-key cryptography.
- Elliptical curve cryptography is a public key encryption technique which is based on the theory of elliptical curves.

- This encryption technique uses the properties of elliptic curve in order to generate keys instead of using the traditional methodology of generation of keys using the product of two very large prime numbers.
- ECC is a public key cryptosystem which is used to generate the public key and the private key in order to encrypt and decrypt the data.
- It is based on the mathematical complexity of solving the elliptic curve discrete logarithm problem which deals with the problem of calculating the number of steps or hops it takes to move from one point to another point on the elliptic curve.
- Elliptic curves are the binary curves and are symmetrical over x- axis. These are defined by the function:

$$y^2 = x^3 + ax + b$$



- Where x and y are the standard variables that define the function while a and b are the constant coefficients that define the curve.
- As the values of a and b change, elliptical curve also alters.
- For elliptical curves, the discriminate is non zero.
- The operations used on elliptical curves in cryptography are point addition, point multiplication and point doubling.
- The important characteristic of elliptic curve is the finite field concept. This means that there is a way to limit the values on the curve.

Few terms that will be used,

E -> Elliptic Curve
P -> Point on the curve
n -> Maximum limit (This should be a prime number)

Key Generation:

A key exchange between users A and B can be accomplished as follows:

1. A selects an integer n_A less than n. This is A's private key. A then generates a public key $P_A = n_A \times G$; the public key is a point in $E_q(a, b)$.
2. B similarly selects a private key n_B and computes a public key P_B.
3. A generates the secret key $k = n_A \times P_B$. B generates the secret key $k = n_B \times P_A$.

The two calculations in step 3 produce the same result because

$$n_A \times P_B = n_A \times (n_B \times G) = n_B \times (n_A \times G) = n_B \times P_A$$

$$\boxed{n_A \times P_B = n_B \times P_A}$$

Elliptic Curve Encryption/Decryption:

Encryption:

- Let the message be M
- First encrypt the message M into a point on elliptic curvr.
- Let this point be Pm.
- For encryption, choose a random positive integer K.

The Cipher point will be

$$C_m = \{kG, P_m + kP_B\}$$

Where G → Point on curve

This point will be sent to the receiver.

Decryption:

For decryption, multiply x- coordinate with receiver's secret key

$$kG \times n_B$$

Then Subtract from ($kG \times n_B$)Y- coordinate of cipher point

$$P_m + kP_B - n_B(kG)$$

w.k.t

$$P_B = (n_B \times G)$$

Therefore,

$$\boxed{P_m + kP_B - n_B(kG) = P_m + k(n_BG) - n_B(kG) = P_m}$$

Key sizes with equivalent security levels:

Symmetric Key Algorithms	Diffie–Hellman, Digital Signature Algorithm	RSA (size of *n* in bits)	ECC (modulus size in bits)
80	$L = 1024$ $N = 160$	1024	160–223
112	$L = 2048$ $N = 224$	2048	224–255
128	$L = 3072$ $N = 256$	3072	256–383
192	$L = 7680$ $N = 384$	7680	384–511
256	$L = 15{,}360$ $N = 512$	15,360	512+

Note: L = size of public key, N = size of private key.

****