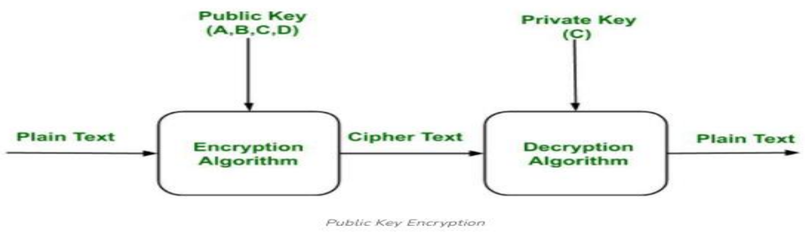
ASSYMETRIC KEY ENCRYPTION (PUBLIC KEY ENCRYPTION)

- Public key cryptography is a cryptographic technique that involves 'two distinct keys' termed as public key and private key,for encryption and decryption and also called as asymmetric key cryptography.
- The public key cryptography is totally based on the 'invertible mathematical' function which makes it different from the conventional symmetric key cryptography.
- Public key cryptography has become an essential means of providing confidentiality, especially through its need of key distribution, where users seeking private connection exchange encryption keys.
- It also features digital signatures which enable users to sign keys to check their identities.

Difference Between Symmetric Encryption and Public-Key Encryption

Basis	**Symmetric Encryption**	**Asymmetric Encryption (Public Key Encryption)**
Required for Work	• Same algorithm with the same key is used for encryption and decryption • The sender and receiver must share the algorithm and key.	• One algorithm is used for encryption and a related algorithm decryption with pair of keys, one for encryption and other for decryption • Receiver and sender must each have one of the matched pair of keys(not identical)
Required for Security	• Key must be kept secret • If the key is secret, it is very impossible to decipher message • Knowledge of the algorithm plus samples of ciphertext must be impractical to determine the key.	• One of the two keys must be kept secret • If one of the key is kept secret, it is very impossible to decipher message. • Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be impractical to determine the other key



Components of Public Key Encryption

- **Plain Text:** This is the message which is readable or understandable. This message is given to the Encryption algorithm as an input.
- **Cipher Text:** The cipher text is produced as an output of Encryption algorithm. We cannot simply understand this message.
- **Encryption Algorithm:** The encryption algorithm is used to convert plain text into cipher text.
- **Decryption Algorithm:** It accepts the cipher text as input and the matching key (Private Key or Public key) and produces the original plain text
- **Public and Private Key:** One key either Private key (Secret key) or Public Key (known to everyone) is used for encryption and other is used for decryption

PUBLIC KEY CRYPTOSYSTEMS – Principles, Applications, Requirements

- The keys generated in public key cryptography are too large including 512, 1024, 2048 and so on bits.
- The major issue in public key cryptosystems is that an attacker can masquerade as a legal user.
- It can substitutes the public key with a fake key in the public directory.
- Moreover, it can intercepts the connection or alters those keys.
- Public key cryptography plays an essential role in online payment services and ecommerce etc.
- These online services are ensure only when the authenticity of public key and signature of the user are ensure.

Weakness of the Public Key Encryption

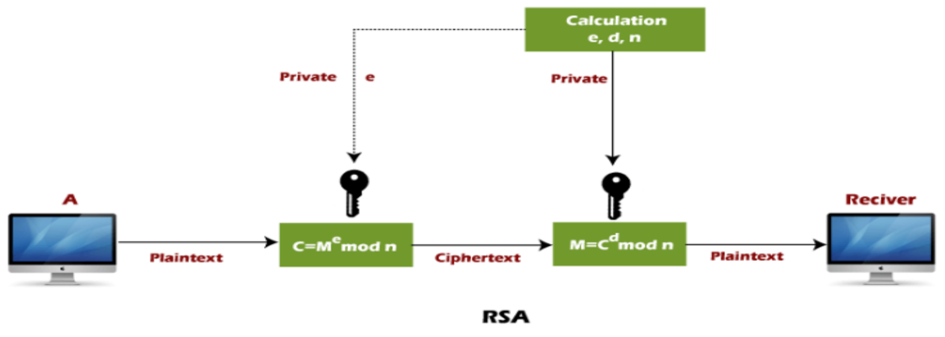
- Public key Encryption is vulnerable to Brute-force attack.
- This algorithm also fails when the user lost his private key, then the Public key Encryption becomes the most vulnerable algorithm.
- Public Key Encryption also is weak towards man in the middle attack. In this attack a third party can disrupt the public key communication and then modify the public keys.

Applications of the Public Key Encryption

- **Encryption/Decryption:** Confidentiality can be achieved using Public Key Encryption. In this the Plain text is encrypted using receiver public key. This will ensure that no one other than receiver private key can decrypt the cipher text.
- **Digital signature:** Digital signature is for senders authentication purpose. In this sender encrypt the plain text using his own private key. This step will make sure the authentication of the sender because receiver can decrypt the cipher text using senders public key only.
- **Key exchange:** This algorithm can use in both Key-management and securely transmission of data.

RSA CRYPTOSYSTEM

- RSA is the most common public-key algorithm, named after its inventors **Rivest, Shamir, and Adelman (RSA).**
- RSA algorithm is an **asymmetric cryptography algorithm**
- RSA encryption relies on few basic assets and quite a bit of math.
- These elements are required:
    - A public key (e)
    - A private key (d)
    - Two prime numbers (P and Q), multiplied (N)
- Security relies on the assumption that it's impossible to determine the value of d.



RSA ALGORITHM

RSA algorithm uses the following procedure to generate public and private keys:

- Select two large prime numbers, p and **q**.
- Multiply these numbers to find **n = p x q,** where **n** is called the modulus for encryption and decryption.
- Choose a number **e** less than **n**, such that n is relatively prime to **(p - 1) x (q -1).** It means that **e** and **(p - 1) x (q - 1)** have no common factor except 1. Choose "e" such that 1<e < φ (n), e is prime to φ (n),
  gcd (e,d(n)) =1
- If **n = p x q,** then the public key is <e, n>. A plaintext message **m** is encrypted using public key <e, n>. To find ciphertext from the plain text following formula is used to get ciphertext C.
  $C = m^e$ mod n
  Here**, m** must be less than **n**. A larger message (>n) is treated as a concatenation of messages, each of which is encrypted separately.
- To determine the private key, we use the following formula to calculate the d such that:
  D_e mod {(p - 1) x (q - 1)} = 1
  Or
  D_e mod φ (n) = 1
- The private key is <d, n>. A ciphertext message **c** is decrypted using private key <d, n>. To calculate plain text **m** from the ciphertext c following formula is used to get plain text m.
  $m = c^d$ mod n

Example of RSA encryption algorithm:

Q1) How we can encrypt plaintext 9 using the RSA public-key encryption algorithm. This example uses prime numbers 7 and 11 to generate the public and private keys.

Step 1: Select two large prime numbers, p, and **q**.

$p = 7$

$q = 11$

Step 2: Multiply these numbers to find **n = p x q,** where **n** is called the modulus for encryption and decryption.

First, we calculate

n = p x q

$n = 7 \times 11$

$n = 77$

Step 3: Choose a number **e** less that **n**, such that n is relatively prime to **(p - 1) x (q -1).**

It means that **e** and **(p - 1) x (q - 1)** have no common factor except 1.

Choose "e" such that $1<e < \varphi (n)$, e is prime to φ (n), gcd (e, d (n)) =1.

Second, we calculate

$\boldsymbol{\varphi (n) = (p - 1) \times (q-1)}$

$\varphi (n) = (7 - 1) \times (11 - 1)$

$\varphi (n) = 6 \times 10$

$\varphi (n) = 60$

Let us now choose relative prime e of 60 as 7.

Thus the **public key is <e, n> = (7, 77)**

Step 4: A plaintext message **m** is encrypted using public key <e, n>. To find ciphertext from the plain text following formula is used to get ciphertext C.

To find ciphertext from the plain text following formula is used to get ciphertext C.

$\mathbf{C = m^e \bmod n}$

$C = 9^7 \bmod 77$

$C = 37$

Step 5: The private key is <d, n>. To determine the private key, we use the following formula d such that:

$\mathbf{D_e \bmod \{(p - 1) \times (q - 1)\} = 1}$

7d mod 60 = 1, which gives d = 43

The **private key is <d, n> = (43, 77)**

Step 6: A ciphertext message **c** is decrypted using private key <d, n>. To calculate plain text **m** from the ciphertext c following formula is used to get plain text m.

$\mathbf{m = c^d \bmod n}$

$m = 37^{43} \bmod 77$

$m = 9$

Therefore, **Plain text = 9 and the ciphertext = 37**

Q2). In an RSA cryptosystem, a particular A uses two prime numbers, 13 and 17, to generate the public and private keys. If the public of A is 35. Then the private key of A is ……………?.

p = 13 , q = 17

n = p x q = 13 x 17 = 221

φ (n) = (p - 1) x (q-1)

φ (n) = (13 - 1) x (17 - 1) = 12 x 16 = 192

g.c.d (35, 192) = 1

To determine the private key, calculate the d such that:

Calculate **d = d$_e$ mod φ (n) = 1**

d = d x 35 mod 192 = 1

d = (1 + k.φ (n)) /e [let k =0, 1, 2, 3………………]

Put k = 0

d = (1 + 0 x 192)/35

d = 1/35

Put k = 1

d = (1 + 1 x 192)/35

d = 193/35

Put k = 2

d = (1 + 2 x 192)/35

d = 385/35

d = 11

The private key is < d, n > = (11, 221)

Hence, **private key i.e. d = 11**

Q3). By using RSA algorithm , calculate the private key and then encrypt the word " HI" , if p= 53, q= 59 and the public key ,e =3

P=53 q=59 e= 3

n= p*q = 53*59 = 3127

φ (n) = 52 * 58 = 3016

d e mod φ (n) = 1 // e= 3

d= 2011

Public Key (n ,e) = (3127 , 3) and Private Key , d = 2011

To encrypt the word " HI" ,Convert letters to numbers : H= 8 and I =9   // A=1, B=2 and so on.

Thus encrypted data , C = (M^e) mod n

So,H = (8^3) mod 3127

I = (9^3) mod 3127

Ie, encrypt " HI " as 1394

Now , we will decrypt 1394 as P = (C^d) mod n

Thus our data encrypted as 89

Ie, 8 =H and 9=I

Thus our data decrypted as " HI"

RSA SECURITY ATTACKS

Some of the possible attacks on RSA are:-

- **Brute force attack**
- **Timing attacks**
- **Computing phi(n)**

Brute force attack

Brute force attack in simple steps:

- Finding the two prime numbers, **p** and **q**, that were multiplied to get the modulus n is the first step in decrypting the private key.
- Factoring n is equivalent to calculating φ(n) for a given n.
- With the currently available algorithms, factoring the problem takes at least as long as figuring out d given e and n.
- Since factoring N allows us to brute-forcibly crack a private key, RSA's security depends on how challenging it is to factor huge numbers.

Since the algorithm can be attacked by brute force, the RSA designers have put some constraints on **p** and **q**.

Timing attacks

Timing attack in simple steps.

- A timing attack is similar to a thief figuring out a safe's combination by watching how long it takes someone to flip the dial from one number to the next.
- It has been found that the value of the key influences how long the RSA algorithm needs to complete its cryptographic operations.
- Therefore, a rough estimate of the private key can be generated based on the time needed to apply it to certain information.
- Depending on how close an attacker may approach the process of carrying out the crypto operation, the significance of this danger grows.
- If the attacker cannot watch the processing time closely, the attack is not practical.
- Rivest has proposed a solution that normalizes computation time so that different keys have comparable execution limits.

Computing φ(n)

- We know that computing **φ(n)** is no easier than factoring n.
- If we know **n** and **φ(n)**, and **n** is the product of two primes **p** and **q,** then **n** can be easily factored by solving the two equations:-

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

For the two "unknowns," **p** and **q**. The following steps make it simple to do this. When we put **q = n/p** into the second equation, we get a quadratic equation with the number **p** being unknown:-

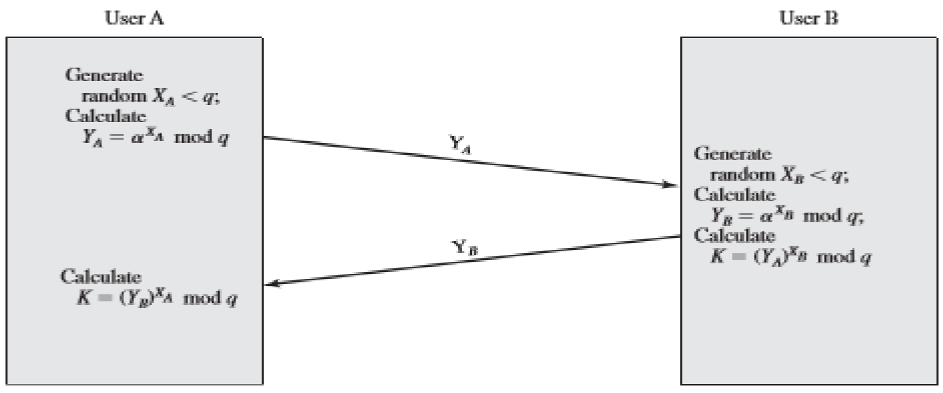
$$p2 - (n- \varphi(n)+1)p +n = 0.$$

The two roots of the above equation will be **p** and **q**, the factors of n. Therefore, a cryptanalyst can factor **n** and undermine the system if he discovers the value of **φ (n)**. In other words, computing **φ (n)** is not any simpler than factoring **n**.

DIFFIE - HELLMAN KEY EXCHANGE

- The Diffie-Hellman key exchange (**also known as exponential key exchange**) is a method for securely exchanging cryptographic keys over an insecure channel.
- It is a fundamental building block of many secure communication protocols, including **SSL/TLS** and SSH.
- The Diffie-Hellman key exchange works by allowing two parties (Alice and Bob) to agree on a shared secret key over an insecure channel, without any other party being able to intercept the key or learn anything about it.
- The key exchange involves the following steps –
  - Alice and Bob agree on two large prime numbers, α and q, and a public key exchange algorithm.
  - Alice chooses a secret integer, X_A, and computes Y_A = $(\alpha)^{X_A}$mod q. She sends A to Bob.

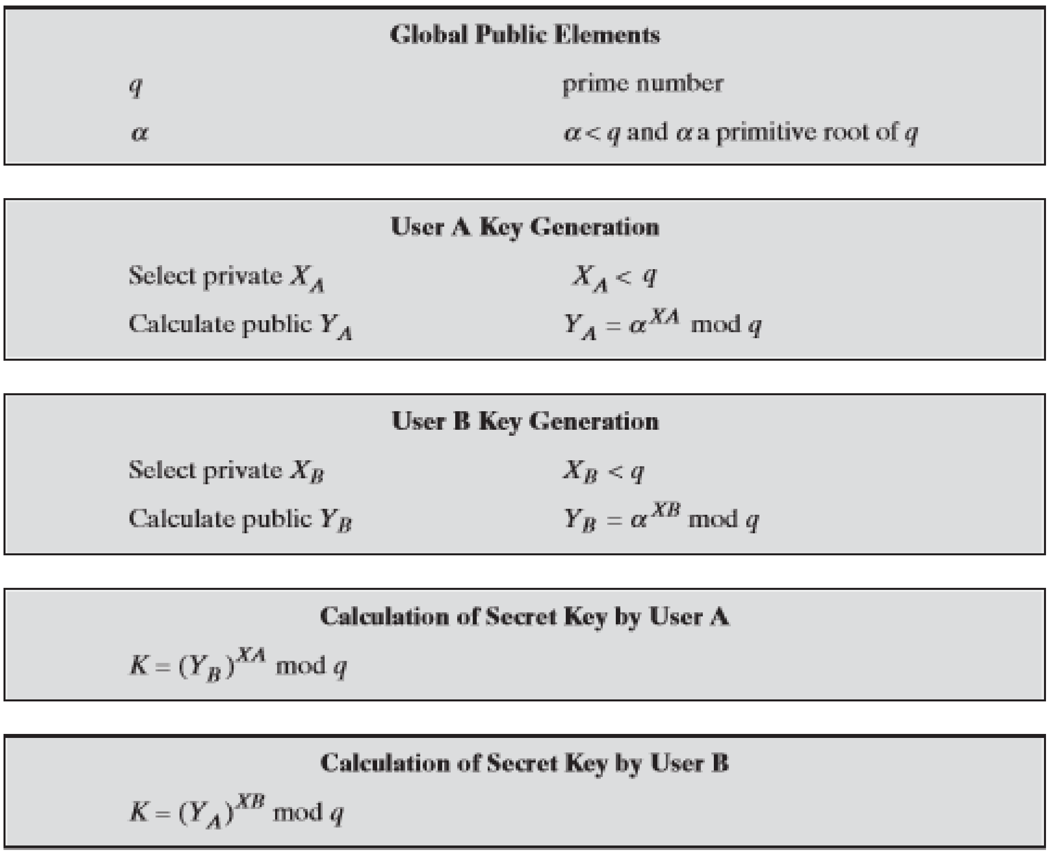
- Bob chooses a secret integer, X_B, and computes Y_B = $(\alpha)^{X_B}$ mod q. He sends B to Alice.
- Alice computes K = $(Y_B)^{X_A}$ mod q. Bob computes K = $(Y_A)^{X_B}$mod q.
- Alice and Bob now both have shared secret keys, which they can use to establish a secure communication channel.


User A
Generate random $X_A < q$;
Calculate $Y_A = \alpha^{X_A} \bmod q$
Calculate $K = (Y_B)^{X_A} \bmod q$
Y_A
Y_B
User B
Generate random $X_B < q$;
Calculate $Y_B = \alpha^{X_B} \bmod q$;
Calculate $K = (Y_A)^{X_B} \bmod q$

- The security of the Diffie-Hellman key exchange relies on the fact that it is computationally infeasible for an attacker to determine the shared secret keys from the public values. This allows Alice and Bob to exchange the key securely, even over an insecure channel.

KEY EXCHANGE ALGORITHM

- 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.
- The Diffie-Hellman algorithm depends for its effectiveness on the difficulty of computing discrete logarithms.



Global Public Elements	
q	prime number
α	$\alpha < q$ and α a primitive root of q

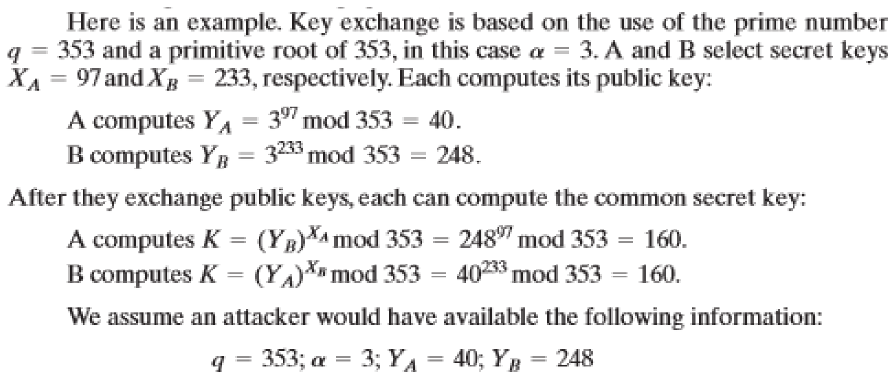
User A Key Generation	
Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = \alpha^{XA} \bmod q$

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

Example:



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

We assume an attacker would have available the following information:

$$q = 353;\ \alpha = 3;\ Y_A = 40;\ Y_B = 248$$

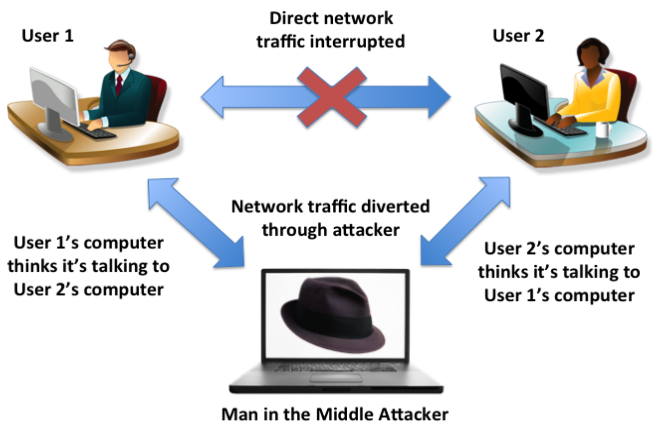
Where is Diffie-Hellman Key Exchange Used?

- **Secure communication protocols** –
  The Diffie-Hellman key exchange is used in many secure communication protocols, such as SSL/TLS and SSH, to establish a secure channel between two parties. It allows the parties to agree on a shared secret key that can be used to encrypt and decrypt messages exchanged over the channel.
- **Virtual private networks (VPNs)** –
  The Diffie-Hellman key exchange is often used in **VPNs** to establish a secure connection between a client and a server. It allows the client and server to agree on a shared secret key that can be used to encrypt and decrypt traffic exchanged over the VPN connection.
- **Secure file transfer protocols** –
  The Diffie-Hellman key exchange is used in many secure file transfer protocols, such as SFTP and FTPS, to establish a secure channel for transferring files between two parties. It allows the parties to agree on a shared secret key that can be used to encrypt and decrypt the transferred files.
- **Other applications** –
  The Diffie-Hellman key exchange is also used in many other applications where secure communication is required, such as secure email, secure web browsing, and secure voice over IP (VoIP). It is a flexible and widely supported technique for establishing secure communication channels.

Vulnerabilities of Diffie-Hellman Key Exchange

MAN-IN-THE-MIDDLE ATTACKS

- If an attacker is able to intercept and modify the messages exchanged between Alice and Bob during the key exchange, they may be able to impersonate Alice or Bob and establish a secure channel with the other party. This can be prevented by using certificate-based authentication and/or by verifying the authenticity of the messages using message authentication codes (MACs).


User 1
Direct network traffic interrupted
User 2
Network traffic diverted through attacker
User 1's computer thinks it's talking to User 2's computer
User 2's computer thinks it's talking to User 1's computer
Man in the Middle Attacker

- 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 $K2 = (Y_A)^{X_{D2}} \bmod q$.
4. Bob receives Y_{D1} and calculates $K1 = (Y_{D1})^{X_B} \bmod q$.
5. Bob transmits Y_B to Alice.
6. Darth intercepts Y_B and transmits Y_{D2} to Alice. Darth calculates $K1 = (Y_B)^{X_{D1}} \bmod q$.
7. 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 and Alice and Darth share secret key .
- 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.

- The key exchange protocol is vulnerable to such an attack because it does not authenticate the participants.
- This vulnerability can be overcome with the use of digital signatures and public-key certificates

ELLIPTIC CURVE ARITHMETIC / ELLIPTIC CURVE CRYPTOGRAPHY (ECC)

- Elliptical curve cryptography (ECC) is a public key encryption technique based on elliptic curve theory that can be used to create faster, smaller and more efficient cryptographic keys.

- ECC is an alternative to the Rivest-Shamir-Adleman (RSA) cryptographic algorithm and is most often used for digital signatures in cryptocurrencies, such as Bitcoin and Ethereum, as well as one-way encryption of emails, data and software.

- An elliptic curve is not an ellipse, or oval shape, but it is represented as a looping line intersecting two axes, which are lines on a graph used to indicate the position of a point.

- The curve is completely symmetric, or mirrored, along the x-axis of the graph.

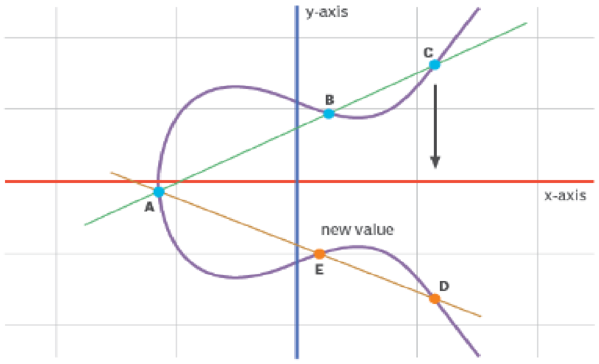
- Public key cryptography systems, like ECC, use a mathematical process to merge two distinct keys and then use the output to encrypt and decrypt data.

- One is a public key that is known to anyone, and the other is a private key that is only known by the sender and receiver of the data.

- ECC generates keys through the properties of an elliptic curve equation instead of the traditional method of generation as the product of large prime numbers. From a cryptographic perspective, the points along the graph can be formulated using the following equation:

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

- ECC is based on the properties of a set of values for which operations can be performed on any two members of the group to produce a third member, which is derived from points where the line intersects the axes



Components of Elliptic Curve Cryptography

1. **ECC keys:**

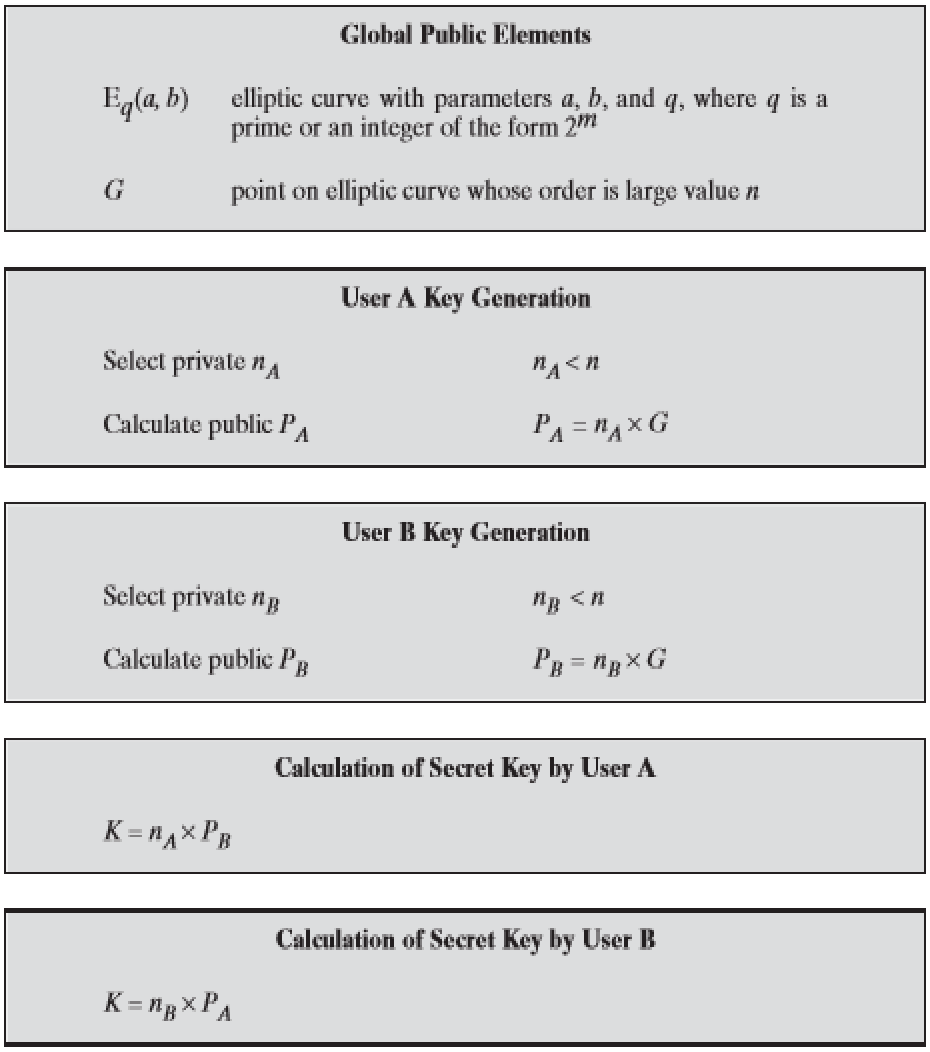
- **Private key:** ECC cryptography's private key creation is as simple as safely producing a random integer in a specific range, making it highly quick. Any integer in the field represents a valid ECC private key.

- **Public keys:** Public keys within ECC are EC points, which are pairs of integer coordinates x, and y that lie on a curve. Because of its unique features, EC points can be compressed to a single coordinate + 1 bit (odd or even). As a result, the compressed public key corresponds to a 256-bit ECC.

2. **Generator Point:**

- ECC cryptosystems establish a special pre-defined EC point called generator point G (base point) for elliptic curves over finite fields, which can generate any other position in its subgroup over the elliptic curve by multiplying G from some integer in the range [0…r].
- The number r is referred to as the "ordering" of the cyclic subgroup.
- Elliptic curve subgroups typically contain numerous generator points, but cryptologists carefully select one of them to generate the entire group (or subgroup), and is excellent for performance optimizations in calculations. This is the "G" generator.

Algorithm :



Global Public Elements

$E_q(a, b)$	elliptic curve with parameters a, b, and q, where q is a prime or an integer of the form 2^m
G	point on elliptic curve whose order is large value n

User A Key Generation

Select private n_A	$n_A < n$
Calculate public P_A	$P_A = n_A \times G$

User B Key Generation

Select private n_B	$n_B < n$
Calculate public P_B	$P_B = n_B \times G$

Calculation of Secret Key by User A

$K = n_A \times P_B$

Calculation of Secret Key by User B

$K = n_B \times P_A$

ECC Encryption/ Decryption

- Several alternatives, will consider simplest
- Must first encode any message M as a point on the elliptic curve P_m
- Select suitable curve and point G as in D-H
- Each user chooses private key n_A < n
- Computes public key ,P_A = n_A G
- To encrypt P_m : C_m = { k G, P_m +k P_b } , k random
- To decrypt C_m : compute , P_m + k P_b - n_B (k G)

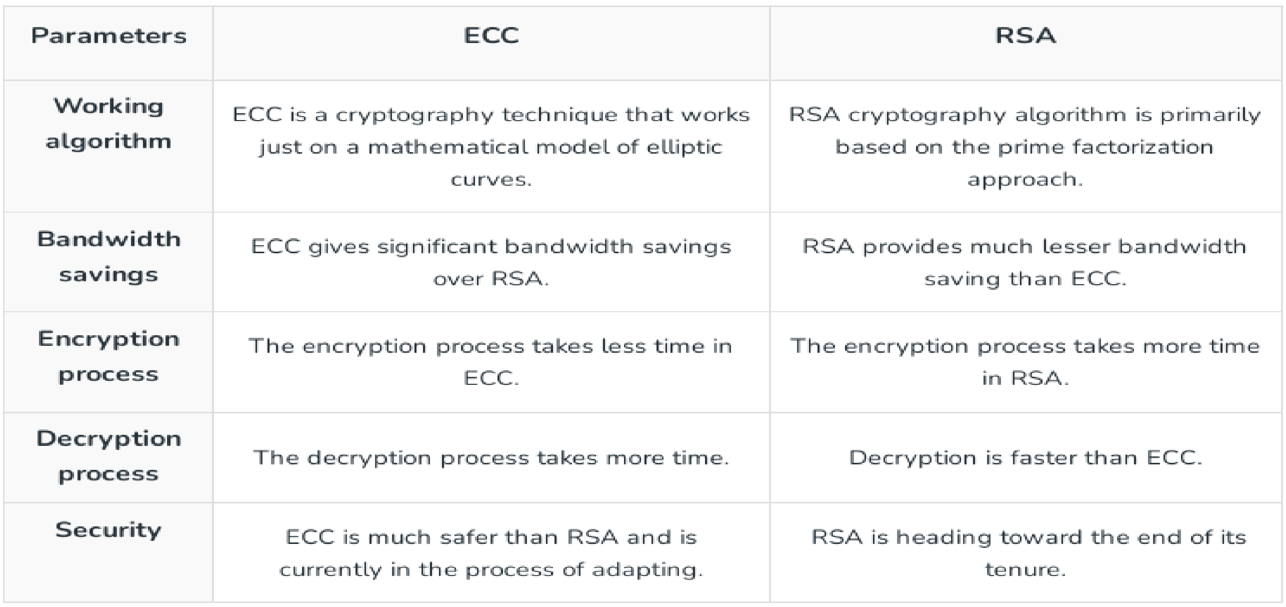
  = P_m + k (n_B G) - n_B (k G)

  = P_m

ECC Security

- Relies on elliptic curve logarithmic problem
- Compared to factoring, can use much smaller key sizes than with RSA
- For equivalent key lengths computations are roughly equivalent

ECC vs RSA



Parameters	ECC	RSA
Working algorithm	ECC is a cryptography technique that works just on a mathematical model of elliptic curves.	RSA cryptography algorithm is primarily based on the prime factorization approach.
Bandwidth savings	ECC gives significant bandwidth savings over RSA.	RSA provides much lesser bandwidth saving than ECC.
Encryption process	The encryption process takes less time in ECC.	The encryption process takes more time in RSA.
Decryption process	The decryption process takes more time.	Decryption is faster than ECC.
Security	ECC is much safer than RSA and is currently in the process of adapting.	RSA is heading toward the end of its tenure.

Benefits of Elliptic Curve Cryptography

- **Fast key generation:**
  ECC cryptography's key creation is as simple as securely producing a random integer in a specific range, making it highly quick. Any integer in the range represents a valid ECC secret key. The public keys in the ECC are EC points, which are pairs of integer coordinates x, and y that lie on a curve.

- **Smaller key size:**
  Cipher text, signatures, and Elliptic-curve cryptography (ECC) is a public-key encryption technique based on the algebraic structure of elliptic curves with finite fields. Compared to non-EC encryption (based on ordinary Galois fields), ECC allows for fewer keys to guarantee equal security**.**

- **Low latency:**
  Signatures can be computed in two stages, allowing latency much lower. By computing signatures in two stages, ECC achieves lower latency than the inverse throughout.

- **Less computation power:**
  Since the ECC key is shorter the computation power is also less computational power, ECC offers high security with faster, shorter keys compared to RSA and take more energy to factor than it does to calculate an elliptic curve objective function.

- **High security:**
  With ECC, you may obtain the same level of security with smaller keys.

Limitations of Elliptic Curve Cryptography

- **Large encryption size:**
  ECC increases the size of the encrypted message significantly more than RSA encryption. The default key length for ECC private keys is 256 bits, but many different ECC key sizes are conceivable depending on the curve.

- **A more complex:**
  The ECC algorithm is more complete and more difficult to implement than RSA.

- **Complex security:**
  Complicated and tricky to implement securely, mainly the standard curves. If the key size used is large enough, ECC is regarded to be highly secure
  .
- **Binary curves:**
  Processing of binary curves is costly.