



**SOMAIYA**  
VIDYAVIHAR UNIVERSITY

K J Somaiya College of Engineering

# Asymmetric key Cryptography

Ms. Swati Mali

B-215

swatimali@gmail.com

Assistant Professor, Department of Computer Engineering  
K. J. Somaiya College of Engineering  
Somaiya Vidyavihar University

# Outline

- Symmetric vs Asymmetric key cryptography
- Stream ciphers vs block ciphers
- Public key cryptography
- RSA
- Elliptic curve cryptography
- Public Key infrastructure (PKI)

# Symmetric Key cryptography

- Uses same key in encryption and decryption process
- The encryption and decryption processes are inverse operations of each other
- Every user pair has to maintain the separate symmetric encryption key
- Both users have to maintain secrecy of the key
- Can be used to authenticate only each other, cannot be used to authenticate themselves to everybody else
- Classical symmetric key ciphers: Caesar Cipher, Playfair cipher, columnar cipher, railfence cipher etc
- Classical block ciphers :ECB, OFB, CFB, CBC
- Modern block ciphers : DES, AES

# Symmetric Key cryptography

- A group of 100 users needs how many symmetric keys in total?

# Symmetric Key cryptography

- A group of 100 users needs how many symmetric keys in total?

$$\begin{aligned}\text{No of keys needed} &= \frac{n(n-1)}{2} \\ &= 100 * \frac{100-1}{2} \\ &= 100 * \frac{99}{2}\end{aligned}$$

*No of shared secret keys = 4950*

# Asymmetric Key cryptography

- Uses different keys in encryption and decryption process
- The encryption and decryption processes might be inverse operations of each other
- Every user has to maintain a pair of keys: public key and private key
- Only the private keys are kept secret, public keys can be announced to world
- Can be used to authenticate themselves to everybody else
- Typically follows stream ciphers
- Examples: RSA, ECC

# Asymmetric Key cryptography

- A group of 100 users needs how many asymmetric keys in total?

# Symmetric Key cryptography

- A group of 100 users needs how many asymmetric keys in total?

$$\begin{aligned} \text{Total No of keys needed} &= 2 * N \\ &= 2 * 100 \\ &= 200 \end{aligned}$$

No of keys each user has to maintain: 99 public keys of other users, 1 public key of own, 1 private key of own  
 $= 99 + 1 + 1 = 101 \text{ keys}$



# Symmetric vs Asymmetric key differences

- **Speed:** Symmetric encryption is generally faster than asymmetric
- **Data Size:** Symmetric encryption is block cipher while Asymmetric is stream cipher
- **Key distribution:** In symmetric encryption, secure key distribution is crucial, as the same key is used for both encryption and decryption. Asymmetric encryption simplifies key distribution, as only the public key needs to be shared.
- **Key usage:** Symmetric encryption uses a single shared key for both encryption and decryption, while asymmetric encryption employs a pair of keys: a public key for encryption and a private key for decryption.
- **Use cases:** Symmetric encryption is ideal for bulk data encryption and secure communication within closed systems, whereas asymmetric encryption is often used for secure key exchanges, digital signatures, and authentication in open systems.

Ref: <https://preyproject.com/blog/types-of-encryption-symmetric-or-asymmetric-rsa-or-aes#:~:text=Asymmetric%20encryption%20uses%20a%20private%20key%20for%20decryption.>

# Symmetric vs Asymmetric key differences

- **Security:** Asymmetric encryption is considered more secure due to the use of two separate keys, making it harder for attackers to compromise the system. However, symmetric encryption can still provide strong security when implemented correctly with strong key management practices.
- **Scalability:** With a pair of keys, it is not difficult to communicate with multiple parties using Asymmetric key and that's how it is more scalable in large networks. With symmetric key, key management is a major issue
- **Resource Utilization:** Symmetric key encryption works on low usage of resources. Asymmetric encryption requires high consumption of resources.

Ref: <https://preyproject.com/blog/types-of-encryption-symmetric-or-asymmetric-rsa-or-aes#:~:text=Asymmetric%20and%20symmetric%20encryption,dion%20data%20private%20key%20for%20decryption.>

**Key Lengths:** 128 or 256-bit key size for symmetric key cryptography while RSA uses 2048-bit or higher key size.



**SOMAIYA**  
VIDYAVIHAR UNIVERSITY

K J Somaiya College of Engineering



# Public Key Cryptosystems

- Public-key/two-key/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 related private-key, known only to the recipient, used to decrypt messages, and sign(create) signatures
- Is asymmetric because—key used for encryption cannot be used for decryption and vice a versa
- Asymmetric algorithms rely on one key for encryption and a different but related key for decryption

# Some terms

- A message  $X$  in plaintext can be considered as  $X = [X_1, X_2, \dots, X_M]$
- The  $M$  elements of  $X$  are letters in some finite alphabet
- The message is sent by user  $A$  and is intended for destination user  $B$
- $B$  generates a related pair of keys a public key,  $PU_b$  and a private key,  $PR_b$
- $PR_b$  is known only to  $B$ , whereas  $PU_b$  is publicly available and therefore accessible by  $A$
- With the message  $X$  and the encryption key  $PU_b$  as input,  $A$  forms the ciphertext

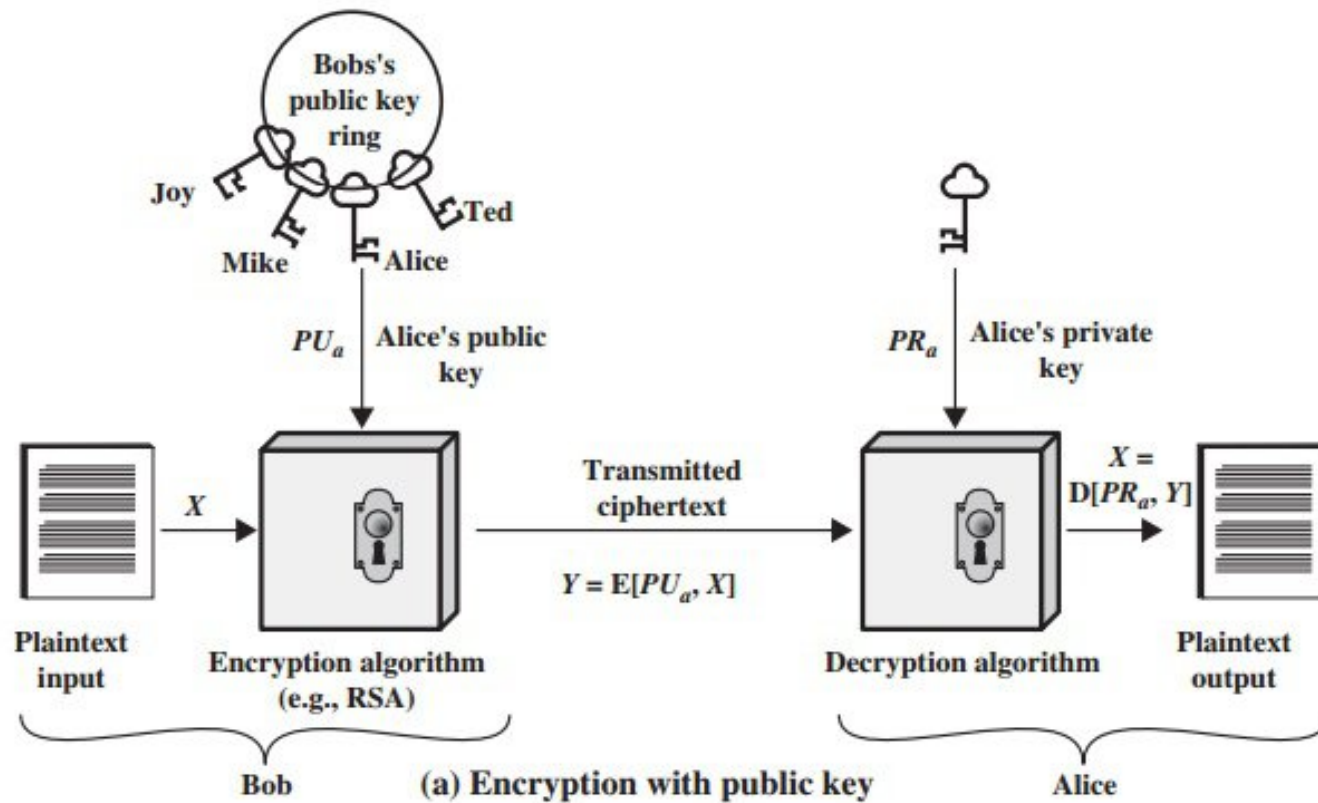
$$Y = [Y_1, Y_2, \dots, Y_N]$$

$$Y = E(PU_b, X)$$

- The intended receiver, in possession of the matching private key, is able to invert the transformation

$$X = D(PR_b, Y)$$

# Encryption with Public key



# Encryption with Private key

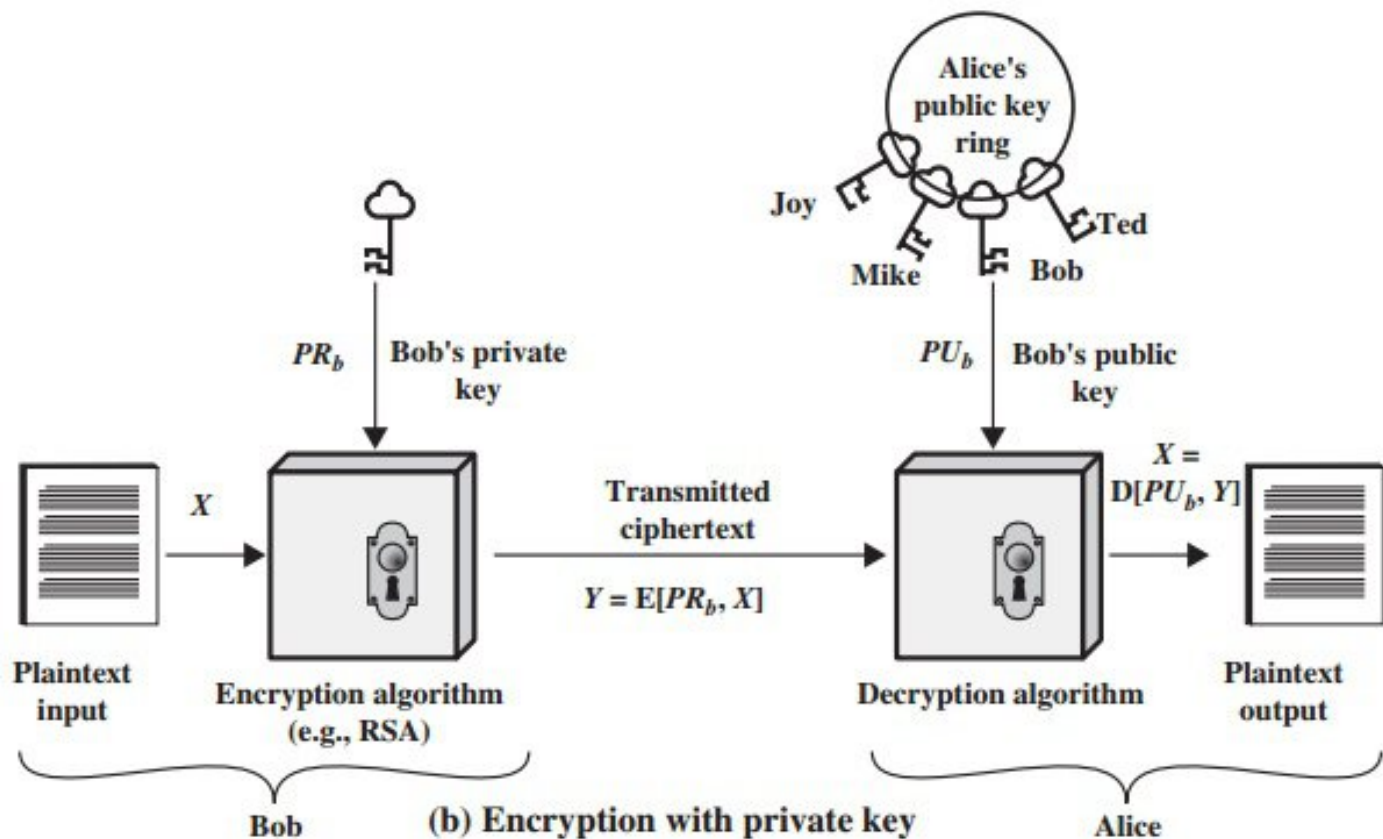


Figure 9.1 Public-Key Cryptography

# Confidentiality with Public key cryptography

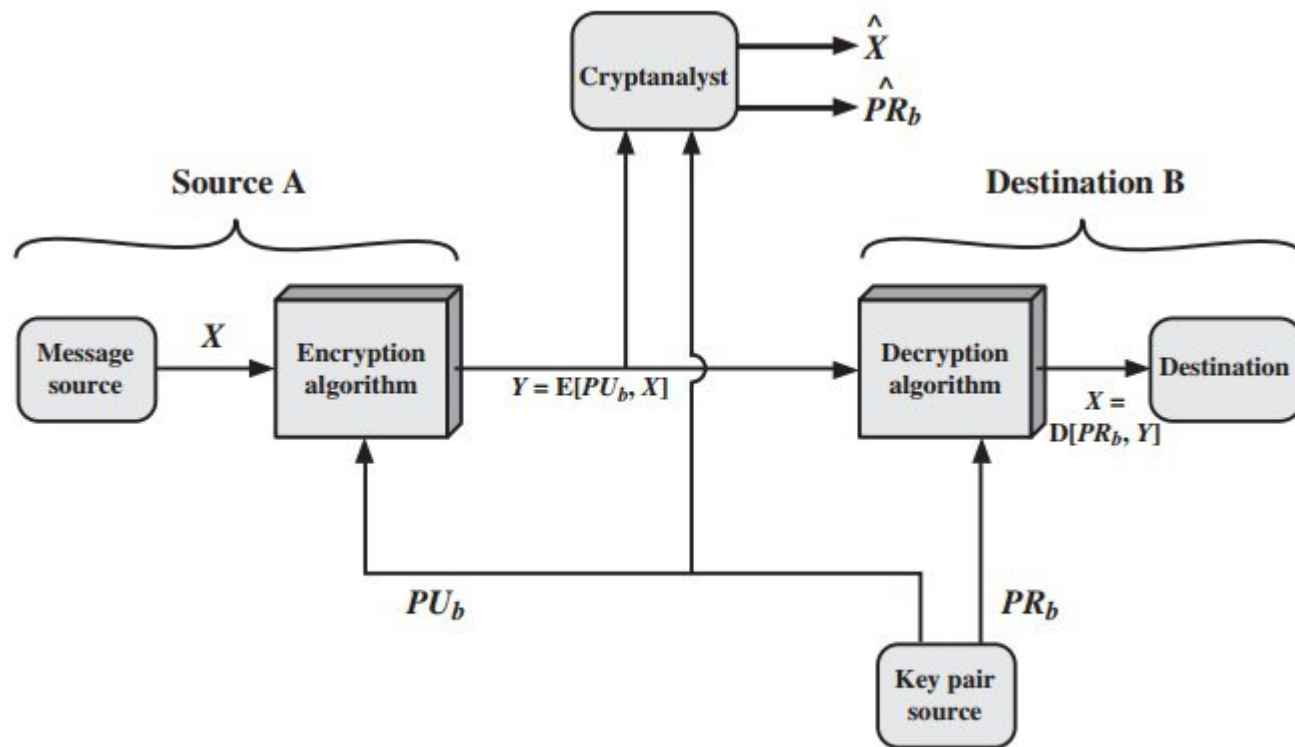


Figure 9.2 Public-Key Cryptosystem: Secrecy

# Authentication with Public key cryptography

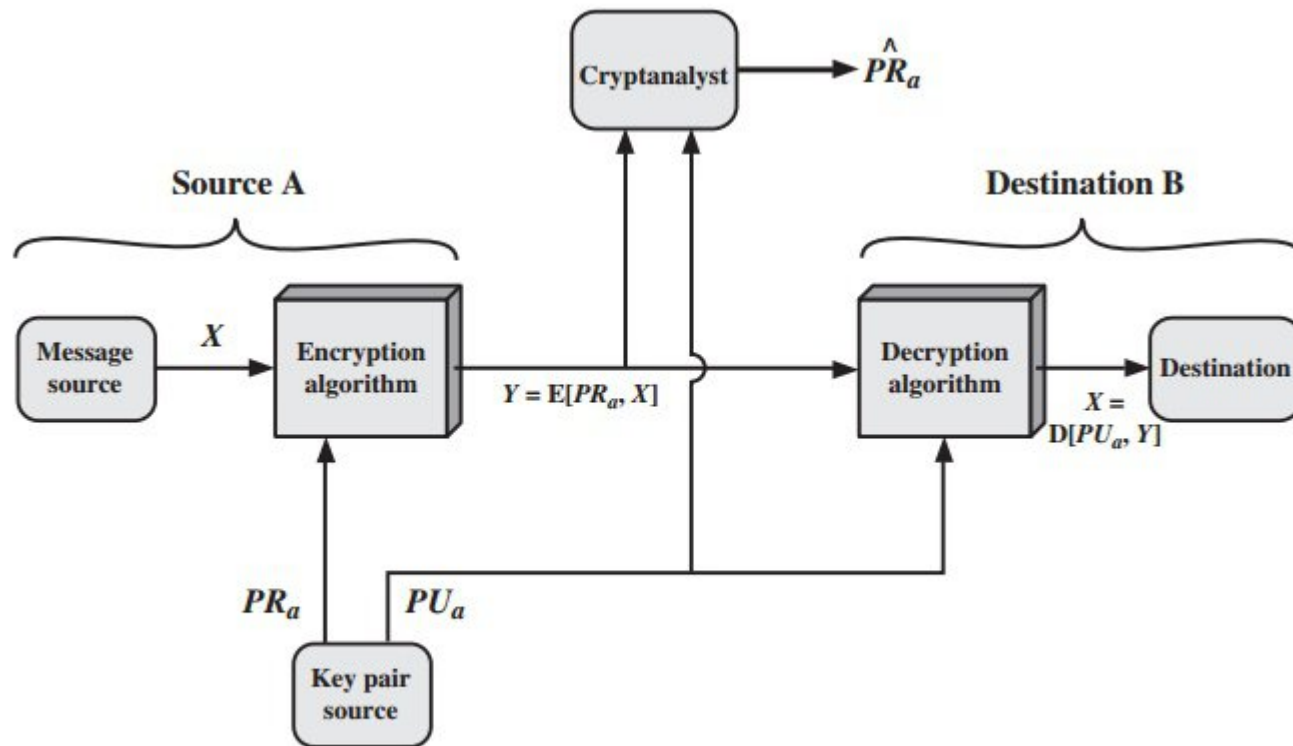


Figure 9.3 Public-Key Cryptosystem: Authentication



# Combining Authentication and confidentiality

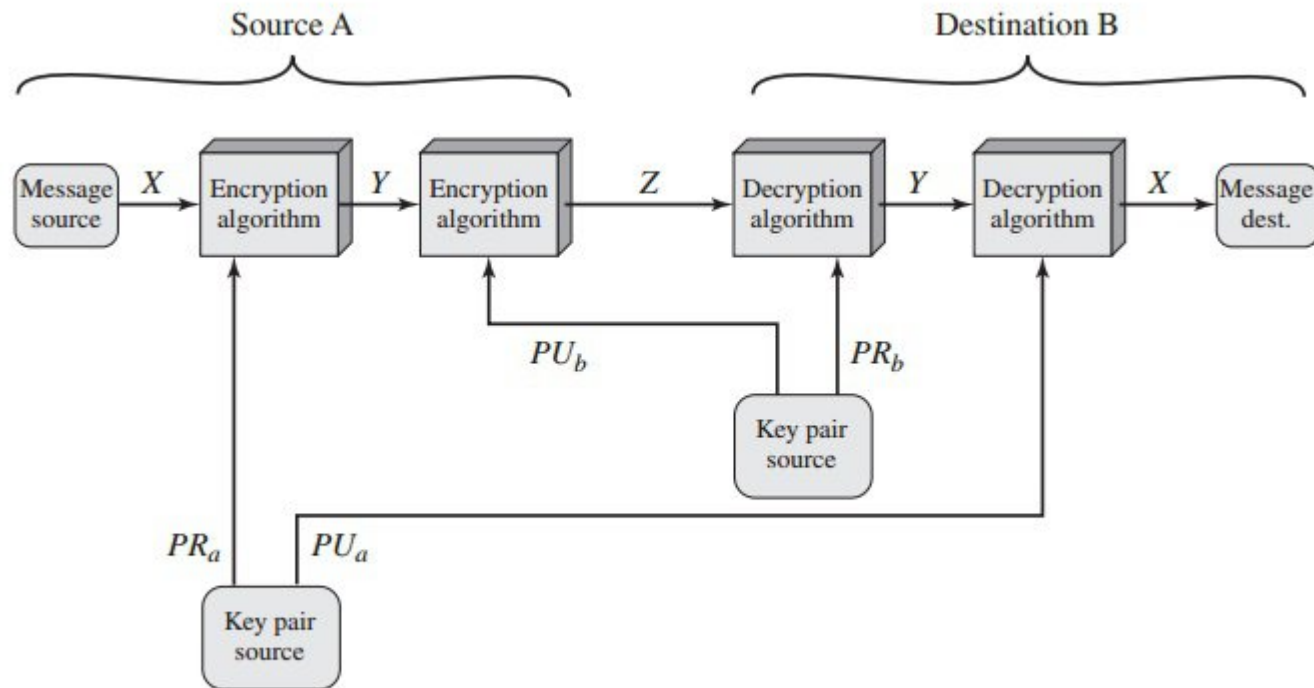


Figure 9.4 Public-Key Cryptosystem: Authentication and Secrecy

# Requirements of Public Key Cryptography

- It is computationally easy for a party B to generate a key pair (public key  $PU_b$  private key  $PR_b$ )
- It is computationally easy for a sender A, knowing the public key and the message to be encrypted,  $M$ , to generate the corresponding ciphertext
- $C = E(Pu_b, M)$
- It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message

$$M = D(PR_b, C) = D[PR_b, E(PU_b, M)]$$

# Requirements of Public Key Cryptography

- Algorithm must fulfill:
- It is computationally infeasible for an adversary, knowing the public key,  $PU_b$  to determine the private key  $PR_b$
- It is computationally infeasible for an adversary, knowing the public key,  $PU_b$  and a ciphertext  $C$ , to recover the original message  $M$ .

# Public key cryptography

- The two keys can be applied in either order
- $M = D[P_{u_b}, E(P_{r_b}, M)] = D[P_{r_b}, E(P_{u_b}, M)]$

# RSA

- "RSA" comes from the surnames of [Ron Rivest](#), [Adi Shamir](#) and [Leonard Adleman](#),
- RSAs publicly described the algorithm in 1977 at MIT, first published in 1978[RIVE78]
- Stream cipher system

# RSA

- The RSA algorithm involves four steps:
  - key generation,
  - key distribution,
  - encryption,
  - and decryption.
- RSA uses modular exponentiation for encryption/decryption
- To attack it, Eve needs to calculate  $e\sqrt{c} \bmod n$ .

# 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  $e$

$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$   
 $de \bmod \phi(n) = 1$

Calculate  $d$

Public key

$KU = \{e, n\}$

Private key

$KR = \{d, n\}$

## Encryption

Plaintext:

$M < n$

Ciphertext:

$C = M^e \bmod n$

## Decryption

Plaintext:

$C$

Ciphertext:

$M = C^d \bmod n$

# Example 1

$P=3, q=5$  , thus  $n = p*q = 15$

Euler's Totient function to compute no of numbers co-prime with  $n$ :

$$\Phi(n) = (p-1)*(q-1) = 2 * 4 = 8$$

Choose  $e$ , such that  $\text{Gcd}(e, \Phi(n)) = 1$  and ,  $1 < e < \Phi(n)$

Thus  $e$  could be any value from the set:  $=\{3,5,7\}$  as

$$\text{GCD}(e, \Phi(n)) \text{ is } 1 \text{ and } 1 < e < 8$$

Let  $e = 3$

Compute  $d$  as:  $d*e \bmod \Phi(n) = 1$

$d * 3 \bmod 8 = 1$  (Use extended Euclidean's method to compute multiplicative inverse)

So,  $d = 3$

Public key =  $(e, n) = (3, 15)$

Private key =  $(d, n) = (3, 15)$

Let  $M = 4$

$$C = M^e \bmod n = 4^3 \bmod 15 = 4$$

$$M = C^d \bmod n = 4^3 \bmod 15 = 4$$



# Example 2

- $P=11, q=3$ , thus  $n = p*q = 33$
- $\phi(n) = (p-1)*(q-1) = 10*2 = 20$
- Choose  $e$  such that  $\text{Gcd}(e, \Phi(n)) = 1$  and  $1 < e < \Phi(n)$
- So  $e = \{3, 7, 9, 11, 13, 17, 19\}$
- Let  $e = 3$
- Compute  $d$  such that:  $d * e \bmod \Phi(n) = 1$
- $d * 3 \bmod 20 = 1$ 
  - So,  $d = 7$
- Public key  $= (e, n) = (3, 33)$
- Private key  $= (d, n) = (7, 33)$
- Let  $M = 7$
- $C = M^e \bmod n = 7^3 \bmod 33 = 13$
- $M = C^d \bmod n = 13^7 \bmod 33 = 7$

# Applications of RSA

- RSA is a versatile encryption and authentication algorithm with a wide range of applications in securing :
  - Digital communication,
  - Data security , and
  - Digital transactions.
- Its strength lies in its ability to provide secure encryption and digital signatures while allowing for public key distribution, making it a cornerstone of modern cryptography.
- Sign: Encrypt with private key
- Verify: Decrypt with associated public key

# Applications of RSA

- **Secure Data Transmission:** RSA is commonly used for securing data transmission over insecure networks, such as the internet. It's employed in protocols like SSL/TLS for securing web traffic, SSH for secure remote access, and S/MIME for secure email communication.
- **Digital Signatures:** RSA is used to create digital signatures, which verify the authenticity and integrity of digital documents or messages. Digital signatures are vital for ensuring that the sender of a message is who they claim to be and that the message has not been tampered with.
- **Secure Key Exchange:** RSA can be used to securely exchange encryption keys between two parties. For example, it's used in the Diffie-Hellman key exchange protocol to establish secure communication channels between parties.

# Applications of RSA

- **Secure Email:** RSA encryption is used in email encryption schemes like PGP (Pretty Good Privacy) and S/MIME to secure the content of emails, ensuring that only the intended recipient can decrypt and read the message.
- **Secure File Transfer:** RSA can be used to secure file transfers, ensuring that files are encrypted during transmission and can only be decrypted by authorized recipients.
- **Digital Certificates:** RSA is a fundamental component of digital certificates, which are used to verify the identity of websites, organizations, or individuals on the internet. Certificate Authorities (CAs) use RSA to create and sign digital certificates.

# Applications of RSA

- **Secure Login and Authentication:** RSA is used in secure login mechanisms, such as two-factor authentication (2FA) and Secure Shell (SSH) logins. It helps ensure that only authorized users can access systems or accounts.
- **Secure E-commerce Transactions:** RSA plays a crucial role in securing online transactions, including credit card transactions and online banking. It protects sensitive financial information during transmission.
- **VPN Encryption:** Virtual Private Networks (VPNs) use RSA for encryption to establish secure connections over the internet. RSA encryption ensures that data transmitted between the user and the VPN server remains confidential.

# Applications of RSA

- **Data Integrity Checking:** RSA signatures can be used to verify the integrity of software updates and downloads. Users can be confident that the software has not been tampered with during the download process.
- **Secure Chat and Messaging Apps:** Many secure chat and messaging apps use RSA encryption to secure conversations and messages, ensuring that only the intended recipients can read the messages.
- **Secure IoT Communication:** RSA can be used to secure communication in the Internet of Things (IoT) by encrypting data transmitted between IoT devices and cloud servers, protecting sensitive information and device control commands.
- **Secure Cloud Storage:** RSA encryption can be employed to protect data stored in the cloud, ensuring that only authorized users can access and decrypt it.

# Advantages of RSA

- Security: highly secure when used with sufficiently long key lengths. Breaking RSA is a computationally intensive and becomes increasingly difficult as the key length grows.
- Public/Private Key Pair: Dual-key system in RSA eliminates the need for both parties to share a secret key beforehand
- Digital Signatures: used in secure communication and document verification.
- Key Exchange: To securely exchange secret keys for symmetric encryption algorithms such as SSL/TLS.
- Mathematical Soundness: RSA is based on the mathematical difficulty of factoring large semiprime numbers, which is believed to be a hard problem. Its security is rooted in well-established mathematical principles.
- Standardization: RSA has been widely standardized and is supported by many cryptographic libraries and software implementations, making it easy to integrate into various applications.

# Disadvantages of RSA

- **Key Length:** key lengths can impact performance and increase computational overhead. Longer key lengths also require more storage for keys and certificates.
- **Computational Intensity:** encryption and decryption operations are computationally intensive, especially with longer key lengths. This slows down processing, making it less suitable for resource-constrained devices.
- **Vulnerability to Quantum Computing:** The security of RSA relies on the difficulty of factoring large numbers, which could be broken by quantum computers. As quantum computing technology advances, RSA's security may be compromised.
- **Key Management:** RSA key management can be challenging, especially in large-scale systems. Safeguarding private keys and ensuring their integrity is critical. Loss or compromise of a private key can have severe security consequences.



# Disadvantages of RSA

- **Performance Overhead:** The computational overhead of RSA encryption and decryption can impact the performance of systems, particularly in high-throughput applications like web servers.
  - Solution : Use hybrid encryption schemes (using RSA for key exchange and symmetric cryptography for data encryption) are often employed.
- **Lack of Forward Secrecy:** RSA does not provide forward secrecy, which means that if an attacker obtains a private key, they can decrypt all past and future communications encrypted with that key. This is a limitation in situations where forward secrecy is desired.
- **Key Length vs. Security:** As computing power advances, key lengths required to maintain security may need to be increased. This can be a challenge because longer key lengths increase computational demands and may not be supported by older hardware and software.

# Summary: Attacks on RSA

- Brute Force Attack:
  - attacker attempts every possible private key until they find the one that successfully decrypts a message.
  - not practical against RSA when long key lengths
- Factorization Attack:
  - based on the difficulty of factoring the product of two large prime numbers.
  - Attack aims to find the prime factors of the modulus ( $n$ ) to compute the private key.
  - Advanced algorithms like Pollard's rho algorithm and the General Number Field Sieve (GNFS) have been developed to factor large numbers efficiently.
  - The security of RSA relies on the difficulty of this factorization problem.
- Timing Attacks:
  - Timing attacks involve measuring the time it takes for an RSA operation to execute.
  - By analyzing the timing variations, an attacker can gain information about the private key, particularly in scenarios where RSA operations take different amounts of time based on the input data.

# Summary: Attacks on RSA

- Chosen Plaintext Attack:
  - attacker can choose specific plaintexts and observe the corresponding ciphertexts produced by an RSA
  - With enough pairs of plaintext/ciphertext, an attacker may be able to deduce information about the private key.
- Ciphertext-Only Attack:
  - attacker has access only to encrypted messages without knowledge of the corresponding plaintexts.
  - This is a challenging scenario, but in some cases, it may be possible to gain partial information about the plaintext or the private key.

# Summary: Attacks on RSA

- Common Modulus Attack:
  - attacker has access to two different ciphertexts encrypted with the same RSA modulus but different public exponents.
  - By exploiting mathematical relationships between these ciphertexts, an attacker may recover the plaintext.
- Quantum Computing Attacks:
  - Quantum computers, if developed sufficiently, could potentially factor large numbers efficiently using algorithms e.g. Shor's algorithm.
  - This would render RSA encryption vulnerable, highlighting the need for post-quantum cryptographic solutions.

# Details- RSA attacks with examples

# Common Modulus Attack

❖ If multiple entities share the same modulus  $n=pq$  with different pairs of  $(e_i, d_i)$ , it is not secure. Do not share the same modulus!

❖ Cryptanalysis: If the same message  $M$  was encrypted to different users

$$\text{User } u_1 : C_1 = M^{e_1} \bmod n$$

$$\text{User } u_2 : C_2 = M^{e_2} \bmod n$$

If  $\gcd(e_1, e_2) = 1$ , there are  $a$  and  $b$  s.t.  $ae_1 + be_2 = 1 \bmod n$

Then,

$$(C_1)^a (C_2)^b \bmod n = (M^{e_1})^a (M^{e_2})^b \bmod n = M^{ae_1 + be_2} \bmod n = M \bmod n$$

# Short Message Attack

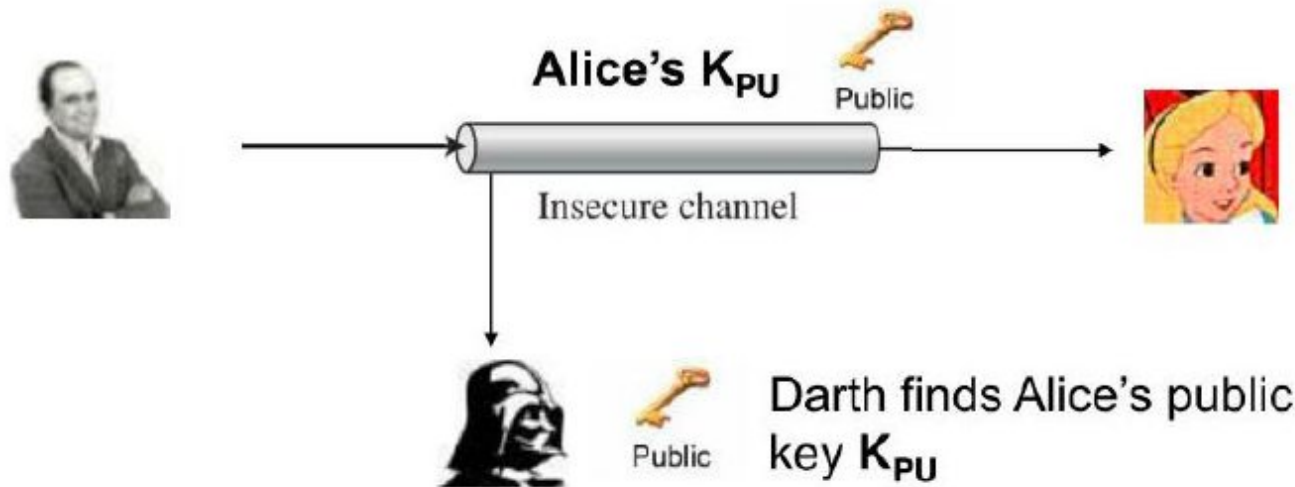
- Typical use of public key algorithm:  
Generating short messages
    - Symmetric keys (used then to send rest of message)
    - Social security numbers, etc.
  - Idea:
    - Adversary acquires public key  $E, n$
    - Uses them to encrypt all possible messages that may be sent (plausible if messages are short enough!) and stores in table
    - Intercepts encrypted message  $C$  and searches for match in the table
- Adversary can recover plaintext without decryption key!



# Short Message Attack (cont.)

- Example:

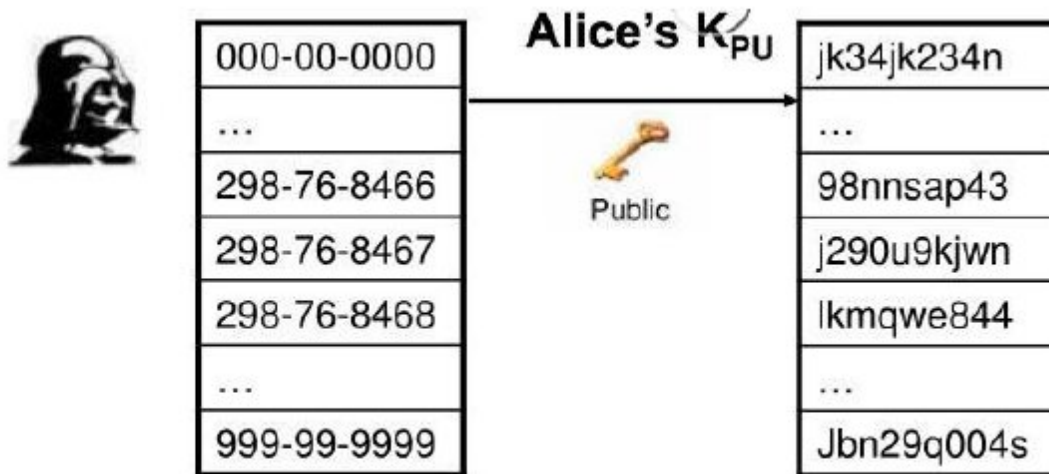
Darth knows that Bob will use Alice's public key to send her a Social Security Number (9 digits)





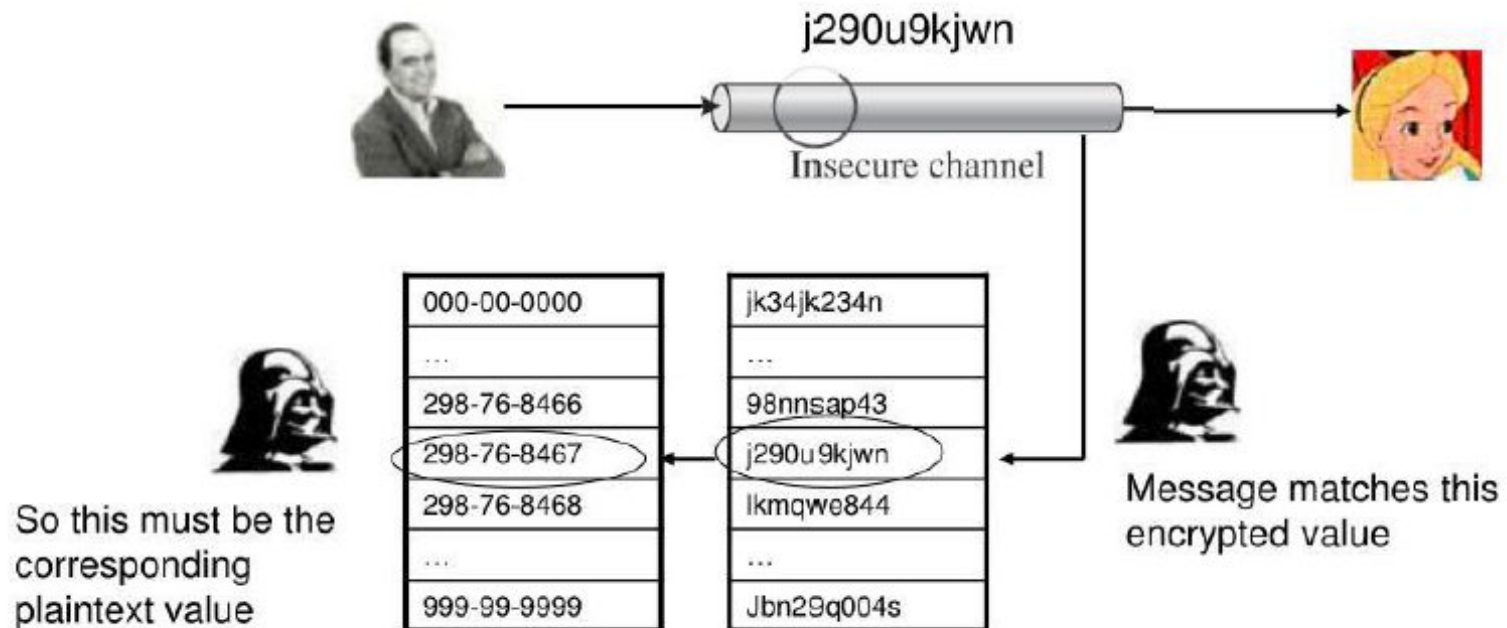
# Short Message Attack (cont.)

- Darth uses Alice's public key  $K_{PU}$  to encrypt all possible Social Security Numbers (only a billion)



# Short Message Attack (cont.)

- Darth intercepts Bob's SSN encrypted with Alice's public key
- Searches for match in table of encrypted values



# RSA Cryptanalysis: Timing Attack (Theoretical)

- Timing attacks measure the time taken to perform cryptographic operations.
- Let's assume you're performing RSA encryption with an insecure implementation that leaks information through timing variations.
- You collect timing data for multiple encryptions of the same plaintext.
  - Encryption 1: 5 ms
  - Encryption 2: 4 ms
  - Encryption 3: 6 ms
- From the timing data, one might observe that the encryption operation took longer for Encryption 3.
- This could indicate that the plaintext had a particular property or structure.
- Over time, one might gather enough information to make educated guesses about the private key.
- In practice, mitigating timing attacks requires implementing secure cryptographic libraries and protecting against side-channel attacks.

# Timing Attack

- If adversary knows the following:
  - Ciphertext  **$C$** 
    - Can be intercepted
    - Can compute how long it takes to multiply ciphertext and compute mods
  - Total time decryption takes
    - Can be observed

They could compute number of 1's in private  **$D$**

- Given enough known plaintexts, can reliably guess  **$D$**  completely

# Timing Attack (cont.)

- Fast exponentiation algorithm used for decryption to compute  $C^D \bmod n$ :

result = 1

for (i = 0 to number of bits in  $D - 1$ ) {

    if ( $i^{\text{th}}$  bit of  $D = 1$ )

        result = (result \* C) mod n

        C =  $C^2 \bmod n$

}

- Speed of decryption depends on number of 1's in  $D$ 
  - Each **1** requires additional multiplication operation
  - Each **0** skips that step

# Timing Attack (cont.)

## Solutions:

- “Pad” algorithm so all decryptions take same time

```
for (i = 0 to number of bits in  $D - 1$ ) {  
  if ( $i^{\text{th}}$  bit of  $D = 1$ ) result = (result *  $C$ ) mod  $n$   
  else garbageVariable = (result *  $C$ ) mod  $n$   
   $C = C^2$  mod  $n$   
}
```



# Cycling attack on RSA

## Cycling Attack

Since  $c = m^e \bmod n$ , encryption maps message  $m$  to one of the elements of the message space  $Z_n = \{0, 1, \dots, n-1\}$ . If the encryption is applied repeatedly on  $c$ , eventually a stage<sup>7</sup> will arrive when  $c$  will get mapped to  $m$ . The adversary uses this fact to his advantage. He intercepts a ciphertext  $c$ , he carries out repeat encryptions of  $c$  till he gets back the intercepted ciphertext  $c$ . He goes back by one step because the message encrypted last must be the original plaintext  $m$ . It has been shown that computational complexity of this attack is equivalent to the complexity of factoring  $n$ .

**Example 2** Bob sends ciphertext 37 to Alice after using her RSA public key  $\{7, 77\}$ . The adversary intercepts the ciphertext and launches cycling attack using the public key of Alice. Write the computation carried out by the adversary to decrypt the ciphertext.

**Solution** The adversary encrypts 37 repeatedly as given below:

$$c_1 = 37^7 \bmod 77 = 16$$

$$c_2 = 16^7 \bmod 77 = 58$$

$$c_3 = 58^7 \bmod 77 = 9$$

$$c_4 = 9^7 \bmod 77 = 37$$

In the fourth step he gets  $c_4 = 37$ . Therefore, he concludes Bob's message is 9.

## Unconcealed Messages

An unconcealed message is one that encrypts to itself, i.e.,  $c = m^e \bmod n = m$ . For example, messages 0, 1,  $n-1$  always remain unconcealed. The number of unconcealed messages in  $Z_n$  is given by

$$[1 + \gcd(e-1, p-1)] \times [1 + \gcd(e-1, q-1)]$$

# RSA Cryptanalysis: Factoring Small Modulus (Insecure Key Lengths):

Problem: Suppose you have an RSA public key :  $\{3, 187\}$ . You want to find the private key ( $d$ ) and factor the modulus.

- First, you need to find the prime factors of 187.
    - Attempt to factor 187:
    - $11 \times 17 = 187$
  - Calculate  $\phi(n) = (11-1)(17-1) = 160$
  - Compute the private exponent ( $d$ ) as the modular multiplicative inverse of 3 modulo 160. In this case,  $d = 107$ .
  - With this small modulus, RSA is insecure, and it's possible to factor the modulus and compute the private key efficiently.
- Real-world RSA implementations use much larger moduli (e.g., 2048 bits or more) to resist factorization attacks.



# RSA Cryptanalysis: Factoring Small Modulus (Insecure Key Lengths)

- **Key = {3,33}**
- **Factorize N:**  
N=33 can be expressed as  $33=3 \times 11$   
So,  $p=3$  and  $q=11$
- **Calculate  $\phi(N) = 2 * 10 = 20$**
- **Public key e:** Assume  $e=3$  'e' must satisfy  $1 < e < \phi(N)$  and  $\gcd(e, \phi(N))=1$
- Here,  $\gcd(3, 20)=1$  so e is valid.
- **Private key d :** d is the modular multiplicative inverse of e modulo  $\phi(N)$

Solve  $e * d \equiv 1 \pmod{\phi(N)}$

$3 * d \equiv 1 \pmod{20}$

Using the extended Euclidean algorithm:  $d=7$

- **Break RSA**  
Using the factorization of N, the private key d can now be computed. With d, you can decrypt any ciphertext encrypted with the public key (N,e)

This is a toy example; real RSA uses large N (hundreds of digits) to make factorization infeasible.

# RSA Cryptanalysis: Chosen Plaintext Attack (Theoretical)

- An attacker can choose plaintexts to be encrypted and observe the resulting ciphertexts.
- This scenario is **highly simplified** for illustration purposes:
- Attacker chooses plaintexts  $P1 = 10$  and  $P2 = 20$ .
- Encrypted ciphertexts are :  $C1 = 37$  and  $C2 = 77$ .
- If the attacker knows value of public key  $e = 3$  (a common choice), they can calculate potential private keys:
- Private key  $d1$ :  $C1^{d1} \equiv P1 \pmod{n} \Rightarrow 37^{d1} \equiv 10 \pmod{n}$
- Private key  $d2$ :  $C2^{d2} \equiv P2 \pmod{n} \Rightarrow 77^{d2} \equiv 20 \pmod{n}$
- In reality, RSA encryption uses padding schemes that complicate this type of attack.
- Cryptanalysis of real-world RSA encryption typically involves far more complex and resource-intensive techniques.

# How to make RSA strong?

- Use strong key lengths
- Employ secure padding schemes
- Regularly update cryptographic libraries and algorithms
- Follow best practices in key management and protection
- Actively explore post-quantum cryptographic alternatives.
- Use padding in message text

# Elliptic Curve Cryptography- ECC

# What is ECC

- branch of public key cryptography
- based on the mathematical properties of elliptic curves
- **Elliptic Curves** are mathematical curves defined by an equation of the form:
$$y^2 = x^3 + ax + b$$
- The curve's set of mathematical are used for cryptographic purposes.

# Why ECC?

- Strong Security with Shorter Key Lengths:
  - strong security even with shorter key lengths compared to traditional RSA encryption.
  - For example, a 256-bit ECC key is as secure as a 3072-bit RSA key.
  - faster encryption and lower computational overhead.
- Efficiency and Performance
  - ECC is computationally efficient and
  - requires fewer computational resources= processing power and memory
  - well-suited for cryptographic hardware implementations, where space and power constraints are critical factors
  - Lower Energy Consumption making it suitable for battery-powered devices and energy-efficient applications.
  - important for resource-constrained devices, such as mobile devices, IoT devices, and embedded systems
  - Compactness in Cryptographic Hardware
- Faster Key Generation and Key Exchange:

# Why ECC?

- Lower Bandwidth and Storage Requirements:
  - ECC's shorter key lengths result in smaller digital certificates and cryptographic messages,
  - reducing the bandwidth and storage requirements
  - beneficial in web security (HTTPS) and email encryption.
- Resilience Against Quantum Attacks:
- Wide Industry Adoption in various industries and standards, e.g. TLS/SSL , digital signatures, and secure communications in IoT devices.
- Forward Secrecy: i.e. even if an attacker compromises a private key at some point in the future, they cannot decrypt past communications, This is important for long-term security.
- Security Provenance:
  - ECC's security is based on the mathematical complexity of elliptic curve problems.
  - It has been extensively analyzed and scrutinized by the cryptographic community.

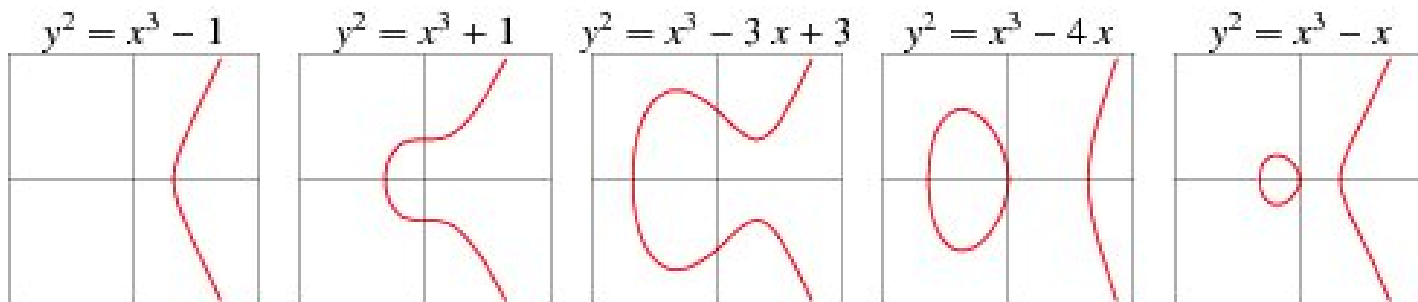
# Some Terms

- ECC Equations
- Points on curve
- Public and private keys



# Elliptic curves

- Elliptic curves are geometric objects
- They don't resemble ellipses in any way,
- and they are not defined by ellipses either.
- Yet, the word “elliptic” is really close to the word “ellipse”.
- In an ellipse, two lines starting from the center of the ellipse to two different points on the circumference form an arc



# Elliptic functions and Elliptic Curve

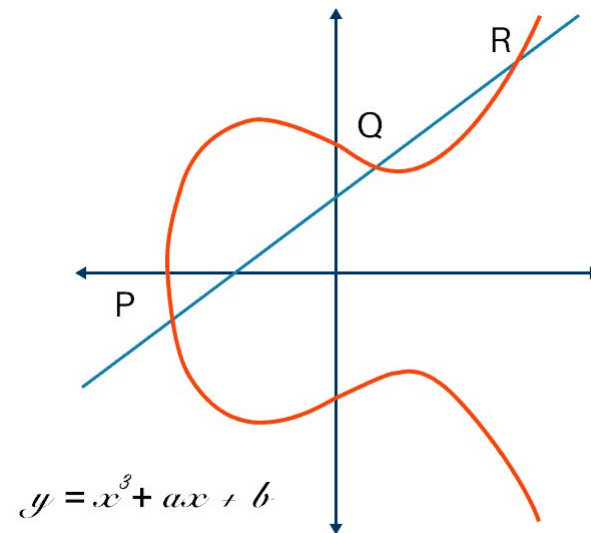
- Elliptic functions have a limited set of possible solutions
- These solutions (points) when plotted, form a curve - elliptic curves.
- **Definition:** elliptic curves are the set of points that are obtained as a result of solving elliptic functions over a predefined space.

# Elliptic functions and Elliptic Curve

- Max number of points on curve are bounded by:  
 $p+1-2\sqrt{p} \leq N \leq (p+1)+2\sqrt{p}$
- Horizontal symmetry: Any point on the curve can be reflected over the x axis and remain on the same curve.
- A more interesting property is that any non-vertical line will intersect the curve in at most three places.

# Why Elliptic curves are symmetric?

- $Y^2 = x^3 + ax + b$
- $\Rightarrow y = \sqrt{x^3 + ax + b}$
- $y$  has +ve and -ve roots/values on graph (+-x)
- when plotted, gives a symmetric curve along  $y=0$



# ECC types

- ECC is defined as
  - EC over  $\mathbb{Z}_p$ , prime curve
  - EC over  $\text{GF}(2^m)$

# EC over $\mathbb{Z}_p$

- Prime curve
- Equation:  $y^2 \bmod p = (x^3 + ax + b) \bmod p$
- variables and coefficients are restricted to finite field
- values range from (0 to  $p-1$ ), values beyond  $p$  are taken as (value  $\bmod p$ )
- curve is focused in only one quadrant from (0,0) through( $p-1, p-1$ ) with positive integers

# Key computations involved in ECC

## 1. Scalar Multiplication:

- multiply a point on the elliptic curve by a scalar (an integer).
- E.g. if  $2 * p$  where  $p = p(x, y)$
- The result is another point on the curve.
- Scalar multiplication is used in key generation, key exchange (ECDH), and digital signatures (ECDSA).

## 2. Point Addition and Subtraction:

- allows combining and canceling points on the curve.
- Diff formulae for  $p - q$  and  $p + q$  based on if  $p = q$  and  $p \neq q$
- Point addition is used in scalar multiplication and key exchange operations to compute new points on the curve.

# Key computations involved in ECC

## 3. Modular Arithmetic:

- ECC operations are modular arithmetic in a finite field. All calculations are performed modulo a prime number  $p$ , which defines the field.
- Modular addition, subtraction, multiplication, and inversion are used extensively in ECC.

## 4. Curve Parameter Computation:

- ECC requires definition of an elliptic curve over a finite field with specific parameters  $a$ ,  $b$ , and the prime modulus  $p$ .
- These parameters are used to define the curve equation  $y^2 = x^3 + ax + b$ .
- The parameters are chosen carefully to ensure security and efficiency.



# Key computations involved in ECC

## 5. Key Pair Generation:

- a random private key is chosen.
- public key = scalar multiplication of the curve's base point by the private key.
- The private key is typically a randomly generated integer, and the public key is a point on the curve.

## 6. Elliptic Curve Key Exchange (ECDH):

- both parties compute shared secrets by performing scalar multiplications using their private keys and the other party's public key.
- The shared secret is derived from the result of scalar multiplication.

## 7. Digital Signatures (ECDSA) (Not in syllabus)

# Input pre-processing

- Conversion of input data into ECC points → out of scope of current honours course study

# ECC arithmetic

1. Given  $Z_p$ , compute all points on the curve
2. Compute  $-p$ , given a point  $p$  on EC
3. Addition over  $Z_p$
4. addition over  $GF(2^m)$

# ECC examples

- Given Elliptic curve :  $E_{11}(1,6)$ , find matching points on curve in ECC over  $Z_p$

Solution:

Given ECC data:  $E_{11}(1,6)$

i.e.  $p=11$ ,  $a=1$ ,  $b=6$ ,

$Z_p$  ECC is defined as:  $y^2 \bmod p = (x^3+ax+b) \bmod p$

i.e.  $y^2 \bmod 11 = x^3+x+6 \bmod 11$

By ECC definition, range of points is  $= (0,p-1) \rightarrow (0,10)$

i.e. Values of  $x$  &  $y$  can be from 0 to 10

Problem: find values of  $x',y'$  such that  $LHS=RHS$

- Computed points also lie on the elliptic curve

# Matching points

x,y	$x^3+x+6 \bmod 11$	$y^2 \bmod 11$		
0	6	0		(2,4), (2,7)
1	8	1		(3,5), (3,6)
2	5	4		(5,2), (5,9)
3	3	9		(7,2), (7,9)
4	8	5		(8,3), (8,8)
5	4	3		(10,2), (10,9)
6	8	3		
7	4	5		
8	9	9		
9	7	4		
10	4	1		

# General process to find all points on the curve

- Define the curve's equation: Specify the coefficients 'a' and 'b' and the finite field modulus 'p' to obtain the curve's equation.
- Iterate through all 'x' values within the field (0 to p-1):
  - For each 'x' value, calculate the right-hand side of the curve equation, i.e.,  $x^3 + ax + b \pmod{p}$ .
- Check if the result is a quadratic residue modulo 'p':
  - A quadratic residue is a value ' $y^2 \pmod{p}$ ' that has a square root modulo 'p.' You can use modular exponentiation to check if the result is a quadratic residue.
- If ' $y^2 \pmod{p}$ ' is a quadratic residue, calculate the square root(s):
  - You'll have two possible values for 'y' for each 'x,' corresponding to the positive and negative square roots.
- Create pairs of (x, y) points:
  - For each 'x' value with valid 'y' values, you have two points on the curve: (x, y) and (x, -y) if 'y' is not zero.
- Store the valid points:
  - Store the pairs (x, y) and (x, -y) as the valid points on the curve.
- the number of points on the curve is finite and depends on the specific curve parameters.

# Problems

- Given Elliptic curve :  $y^2 = x^3 + 4x + 4 \pmod{7}$ , find matching points on curve in ECC over  $Z_p$
- Given Elliptic curve :  $y^2 = x^3 + 4x + 4 \pmod{5}$ , find matching points on curve in ECC over  $Z_p$

# Compute $-p$ , given a point $p$ on EC

- To compute the negation ( $-p$ ) of a point ' $p$ ' on an elliptic curve (EC), you need to find the point with the same x-coordinate but the opposite y-coordinate. Here are the steps to compute  $-p$ :
  - Given a point ' $p$ ' on the elliptic curve, let's denote ' $p$ ' as  $(x, y)$ .
  - Compute the negation ( $-p$ ) by changing the sign of the y-coordinate:
  - If ' $p$ ' =  $(x, y)$ , then  $(-p) = (x, -y)$ .



# Compute $-P$ , given a point $P$ on EC

- Assume ECC equation :  $y^2 = x^3 + 4x + 20$  defined over finite field  $p=29$ .
- Given Point  $P$  on the Curve:  $P = (x, y) = (4, 8)$
- To find  $-P$ , we simply negate the 'y' coordinate of  $P$  while keeping the 'x' coordinate unchanged:
- Negation of  $P$  ( $-P$ ):  $= (x, -y) = (4, -8)$
- So, the negation of point  $P$  on the elliptic curve is:  $-P = (4, -8)$

Taking additive inverse

- $= (4, 21)$

- For  $E_{23}(1,1)$  and  $P=(13,7)$ , find  $-P$

Solution:

$$Y^2 = x^3 + ax + b \pmod{p} \text{ i.e. } = x^3 + x + 1 \pmod{23}$$

$$P=(13,7)$$

$$\begin{aligned} -P &= (x, -y) \\ &= (13, -7) \end{aligned}$$

taking additive inverse of -7,

$$-P = (13, 16)$$

# Addition over $\mathbb{Z}_p$

- It's a fundamental operation used in various cryptographic protocols, including key exchange and digital signatures.
- The addition of points in ECC follows specific rules that are defined by the curve's equation and the finite field modulus 'p.'

# Addition over $\mathbb{Z}_p$

1. Identify the Curve Equation:
2. Select Points P and Q: These points should lie on the curve defined by the equation.
3. Compute the Slope (m) between P and Q:

If  $P \neq Q$ , the slope 'm' between P and Q :

$$\lambda = \frac{y_q - y_p}{x_q - x_p} \bmod p$$

If  $P = Q$ , the slope 'm' between P and Q :

$$\lambda = \frac{3x_p^2 + a}{2 * y_p} \bmod p$$

.....

# Addition over $\mathbb{Z}_p$

4. Calculate the Sum R:

$$\text{x-coordinate : } x_r = \lambda^2 - x_p - x_q \mod p$$

$$\text{y-coordinate } y_r = \lambda * (x_p - x_r) - y_p \mod p$$

- Ensure that the x and y coordinates of R are reduced modulo 'p' to stay within the finite field.
- Ensure you calculate modular inverses appropriately if needed.

The point  $R(x_R, y_R)$  is the sum of points P and Q on the elliptic curve.

### Negating a point:

➤ If  $Q = (x_q, y_q)$

➤ Then  $-Q = -(x_q, y_q) = (x_q, -y_q)$

**Subtraction:**  $P - Q$  can be  $P + (-Q)$ .

➤  $P - Q = (x_p, y_p) - (x_q, y_q) = (x_p, y_p) + (x_q, -y_q \text{ mod } p)$ . Now perform addition.

### Multiplication:

➤ Only Scalar multiplication is possible. Multiplication between two points are not possible. Repeated addition is performed.

➤  $2P = P + P$ ,  $3P = P + P + P$  and so on. Note for slope ( $\lambda$ ) calculation use the formula  $P=Q$ .

Given  $P = (3, 10)$ ,  $Q = (9, 7)$ , over  $E_{23}(1, 1)$ , Compute  $P+Q$ .

Solution:  $p = 23$ ,  $a = 1$ ,  $b = 1$ ,  $y^2 = x^3 + x + 1 \pmod{23}$

- $P \neq Q$
- Thus,
- $\lambda = \frac{y_q - y_p}{x_q - x_p} \pmod{p}$
- $= \frac{7 - 10}{9 - 3} \pmod{23}$
- $= \frac{-3}{6} \pmod{23} = \left(\frac{-1}{2}\right) \pmod{23} = -2^{-1} \pmod{23}$  (multiplicative inverse)
- $= -12 \pmod{23} = 11$  (additive inverse)

- x-coordinate :  $x_r = \lambda^2 - x_p - x_q \mod p$
- y-coordinate  $y_r = (\lambda * (x_p - x_r) - y_p) \mod p$

$$x_r = 11^2 - 3 - 9 = 121 - 12 \mod 23 = 109 \mod 23 = 17$$

$$y_r = (11 * (3 - 17) - 10) \mod 23 = -154 - 10 \mod 23$$

$$= -164 \mod 23 = 20 \mod 23 = 20$$

Thus,  $P(3,10) + Q(9,7) = R(17,23)$



- Given  $P = (3,10)$ ,  $Q = (9,7)$ , over  $E_{23}(1,1)$ , Compute  $2P$ .
- Given  $P = (3,10)$ ,  $Q = (9,7)$ , over  $E_{23}(1,1)$ , Compute  $3P$ . (Hint:  $3P = P + 2P$ )
- Given  $P = (3,10)$ ,  $Q = (9,7)$ , over  $E_{23}(1,1)$  Compute  $P+Q$

$$\therefore R = P + Q = (17, 20)$$

$$\begin{matrix} \uparrow & \uparrow \\ x_R & y_R \end{matrix}$$

ii. Multiplication in Elliptic Curve arithmetic.

$2P =$  ~~add~~ multiplication, i.e. consider repeated addition.

$$2P = P + P.$$

$$4P = P + P + P + P.$$

now,  $P = (3, 10)$ , and  $P = Q$ .

$$\lambda = \left( \frac{3x_P^2 + a}{2y_P} \right) \text{ mod } p.$$

$$= \left( \frac{3(3)^2 + 1}{2 \times 10} \right) \text{ mod } 23.$$

$$= \frac{28}{20} \text{ mod } 23 = \frac{28 \text{ mod } 23}{20}$$

$$= \frac{5}{20} \text{ mod } 23 = \frac{1}{4} \text{ mod } 23$$

$$= 4^{-1} \text{ mod } 23 = 6 \text{ mod } 23.$$

$$\therefore \lambda = 6.$$

$$x_R = (\lambda^2 - x_P - x_Q) \text{ mod } p.$$

$$= (6^2 - 3 - 3) \text{ mod } 23,$$

$$= (36 - 6) \text{ mod } 23.$$

$$= 30 \text{ mod } 23$$

$$x_R = 7.$$

$$y_R = (\lambda(x_P - x_R) - y_P) \text{ mod } p$$

$$= (6(3 - 7) - 10) \text{ mod } 23$$

$$= (6(-4) - 10) \text{ mod } 23$$

$$= (-24 - 10) \text{ mod } 23$$

$$= -34 \text{ mod } 23.$$

Hint

$$y_R = 12 \quad 23 \times 2 = 46$$

$$2P = (7, 12)$$

$$46 - 34 = 12$$

ex. 2. Given  $y^2 = x^3 + 2x + 3 \pmod{17}$ .

Point  $P = (5, 11)$

1. Find  $2P$ .

Slope of line  $\lambda$ .

$$\lambda = \frac{3 \times (5)^2 + 2}{2 \times 11} \pmod{17}.$$

$$= \frac{75 + 2}{22} \pmod{17} = \frac{9}{5} \frac{77}{22} \pmod{17}$$

$$77 \pmod{17} = 9$$

$$22 \pmod{17} = 5$$

$$= \frac{9}{5} \pmod{17} = 9 \times 5^{-1} \pmod{17}.$$

Now find  $5^{-1} \pmod{17}$

$$5 \times \underline{\quad} \pmod{17} = 1.$$

$$17 \times 2 = 34$$

$$5 \times \underline{7} \pmod{17} = 1$$

$$\begin{aligned} \text{so, } 9 \times 7 \pmod{17} &= 63 \% 17 \\ &= \underline{\underline{12}} \end{aligned}$$

Find coordinates  $x_3, y_3$ .

$$x_3 = (12)^2 - 5 - 5 \pmod{17}.$$

$\uparrow$  ( $x_1 \neq x_2$  are do

$$= 144 - 10 \pmod{17}.$$

$$x_3 = 134 \pmod{17} = 15.$$

$$y_3 = 12(5 - 15) - 11 \pmod{17}.$$

$$= 12(-10) - 11 \pmod{17}$$

$$= -120 - 11 \pmod{17}.$$

$$= -131 \pmod{17}.$$

$$\begin{aligned} \text{Find } -131 \pmod{17} &= -(131 \% 17) \\ &= -12. \end{aligned}$$

$$= -12 + 17 = 5$$

$$\therefore y_3 = 5.$$

$$2P = (15, 5).$$



ii. Find  $3P$ .

$$3P = P \oplus 2P.$$

$$= (x_1, y_1) \oplus (x_2, y_2)$$

$$= (5, 11) \oplus (15, 5)$$

and find  $(x_3, y_3)$

slope of line  $\rightarrow$  when  $P \neq Q$ .

$$\lambda = \frac{5-11}{15-5} \mod 17 = \frac{-6}{10} \mod 17$$

$$= 11 \times 10^{-1} \mod 17 =$$

$$10^{-1} \mod 17 = 10 \times 12 \mod 17 = 1$$

$$17 \times 7 = 119$$

$$\rightarrow 11 \times 12 \mod 17$$

$$= 132 \mod 17 = \underline{\underline{13}}$$

$$\text{Now, } x_3 = (13)^2 - 5 - 15 \mod 17.$$

$$= 169 - 20 \mod 17$$

$$= 149 \mod 17$$

$$x_3 = 20$$

$$y_3 = 13(5-13) - 11 \mod 17.$$

$$= 13(-8) - 11 \mod 17$$

$$= -115 \mod 17 = 4.$$

$$3P = (13, 4).$$

iii. Find  $-P$ .

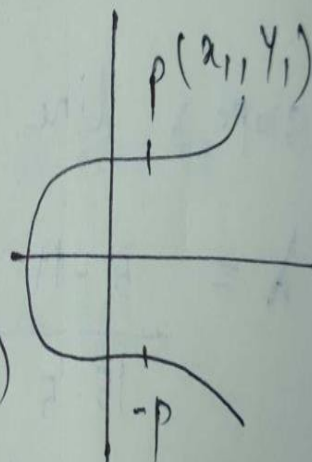
$$(x_1, y_1) = (5, 11)$$

$$(x_1, -y_1) = (5, -11+17)$$

$$-P = (5, 6)$$

add  
mod 17

$$(x_1, -y_1)$$



# Elliptic Curve Cryptography

- The **private keys** in the ECC are integers (in the range of the curve's field size, typically **256-bit** integers).
  - Example of 256-bit ECC private key (hex encoded, 32 bytes, 64 hex digits) is:  
0x51897b64e85c3f714bba707e867914295a1377a7463a9d  
ae8ea6a8b914246319.
- The **public keys** are **points on elliptic curve**  $\{x, y\}$ 
  - Example of ECC public key  
0x02f54ba86dc1ccb5bed0224d23f01ed87e4a443c47fc690  
d7797a13d41d2340e1a.

Consequently, in ECC we have:

- **Elliptic curve (EC)** over finite field  $\mathbb{F}_p$
- **G = generator point** (fixed constant, a base point on the EC)
- **private key** =integer
- **public key** =point

# Deffie-Hellman Key exchange using ECC- ECDH

ECC key exchange – similar to DH Key exchange

Global public elements

- $E_q(a,b)$  - Elliptic curve parameters –  $a, b$  and  $q$  – prime no. or integer of the form  $2m$ .
- $G$  – point on the elliptic curve

# ECDH

## User A key generation

- Select private key  $n_A$ , such that  $n_A < n$
- Calculate public key  $P_A$ ,  $P_A = n_A * G$

## • User B key generation

- Select private key  $n_B$ ,  $n_B < n$
- Calculate public key  $P_B$ ,  $P_B = n_B * G$

- Calculation of secret key by User A,  $K = n_A * P_B$
- Calculation of secret key by User B,  $K = n_B * P_A$



# ECDH

we compute some  $q = k * p$  where  $q, p \in E_p(a, b)$

- Given  $k$  &  $p$  its easy to compute  $q$
- but, knowing  $p$  &  $q$ , it's difficult to compute  $k \Rightarrow$  discrete logarithmic problem for EC
  - could be little easy with availability of a trapdoor
- in reality,  $k$  is very large making brute force infeasible to achieve

# Example ECDH

Assume Elliptic Curve Parameters:

- Elliptic Curve Equation:  $y^2 = x^3 + 2x + 2$  (over a prime field)
- Prime Modulus (p): 17
- Base Point (G): (15, 13)
- Let Alice's Private Key (a): 7
- Let Bob's Private Key (b): 5

# Example ECDH... contd

## Alice's Key Generation:

1. Alice selects private key 'a' (a random integer):  $a = 7$ .
2. Alice computes her public key 'QA' by scalar multiplication:  $QA = a * G$ .

$$QA = 7 * (15, 13) = (7, 7)$$

## Bob's Key Generation:

1. Bob selects private key 'b' (a random integer):  $b = 5$ .
2. Bob computes his public key 'QB' by scalar multiplication:  $QB = b * G$ .

$$QB = 5 * (15, 13) = (3, 13)$$

# Example ECDH... contd

## Key Exchange:

1. Alice sends her public key 'QA' to Bob.
2. Bob sends his public key 'QB' to Alice.

## Shared Secret Calculation:

1. Alice computes the shared secret 'S' using Bob's public key 'QB' and her private key 'a':

$$S = a * QB = 7 * (3, 13) = (10, 0)$$

2. Bob computes the shared secret 'S' using Alice's public key 'QA' and his private key 'b':

$$S = b * QA = 5 * (7, 7) = (10, 0)$$

- Now, both Alice and Bob have computed the same shared secret 'S' as (10, 0).

## Example 2

- Elliptic Curve Equation:  $y^2 = x^3 + 7$
- Prime Modulus (Field Size):  $p = 23$
- Base Point (a known point on the curve):  $G = (3, 10)$
- Private Key for Party A:  $d_A = 6$
- Private Key for Party B:  $d_B = 15$

## Example 2.. contd

- Key Exchange Steps:

1. Party A Computes Public Key:

$$Q_A = d_A * G = 6 * (3, 10) = (18, 6)$$

2. Party B Computes Public Key:

$$Q_B = d_B * G = 15 * (3, 10) = (7, 12)$$

3. Exchange Public Keys:

- Party A sends  $Q_A = (18, 6)$  to Party B.
- Party B sends  $Q_B = (7, 12)$  to Party A.

4. Shared Secret Calculation:

- $S_A = d_B * Q_A = 15 * (18, 6) = (22, 14)$
- $S_B = d_A * Q_B = 6 * (7, 12) = (22, 14)$

# Discrete logarithmic problem in ECC

- Parameters:
  - $n$  is a point on the elliptic curve.
  - $k$  is an integer (the private key or scalar).
  - $p$  is a fixed point on the elliptic curve (typically called the base point).
- Objective:
  - find the integer  $k$  given the base point  $p$  and the result  $n$  of multiplying  $p$  by  $k$ .
  - In mathematical terms, it's solving for  $k$  in the equation:  $n = k * p$ .

# Discrete logarithmic problem in ECC

- **Difficulty:**
  - The security of ECC relies on the computational difficulty of solving this equation for  $k$ .
  - The size of the elliptic curve group and the difficulty of solving the discrete logarithm problem depend on the size of the underlying finite field.
- **Security:**
  - The security of ECC is based on the belief that there is no efficient algorithm to compute  $k$  from  $n$  and  $p$ , except through exhaustive search
  - which is computationally infeasible for large enough key sizes.
- For example, if you have a public key  $n$  and a base point  $p$ , an attacker should not be able to efficiently determine the private key  $k$  used to generate that public key.
- ECC's security is built on this presumed computational hardness of the discrete logarithm problem.s



## ECC encryption

Let the message be M

1. encode message M into a point on elliptic curve:  $P_m$

- For encryption: choose a random positive integer K.
- The cipher point will be:

$$C_m = \{KG, P_m + KPB\}$$

- This point will be sent to receiver.

For Decryption:

- multiply x-coordinate with receiver's secret key :  $KG * nB$
- Then subtract from coordinate of cipher point;
- $P_m + KPB - (K * G * nB)$
- We know that,  $PB = nB * G$

So substitute in above equation and we get,

- $P_m + K * PB - K * PB$

$$= P_m$$

So, receiver gets the same point.

# Limitations and challenges of ECC

- Complexity of Implementation:
  - implementation can be more complex.
  - complexity can lead to implementation errors, which may introduce vulnerabilities if not handled properly.
- Performance Variability:
  - performance can vary based on the choice of elliptic curve parameters, hardware, and software implementations.
  - Some curves may be more efficient than others,
- Standardization and Interoperability:
  - ECC standards and implementations have not been as widely adopted as RSA and other older cryptographic algorithms.
  - Achieving interoperability between different ECC implementations and platforms can be challenging.

# Limitations and challenges of ECC

- Patent and Licensing Issues:
  - Some ECC algorithms and curve choices were initially patented, which raised concerns about licensing and intellectual property rights.
  - However, many of these patents have since expired or been made available for open use.
- Random Number Generation:
  - ECC relies on the generation of high-quality random numbers for key pair generation and other operations. Inadequate random number generation can weaken the security of ECC.

# Limitations and challenges of ECC

- Quantum Threats:
  - ECC is not entirely immune to Quantum Threats
  - Quantum computers with sufficient qubits and processing power could potentially threaten ECC-based encryption.
- Key Management:
  - Managing ECC keys can be challenging, especially in large-scale deployments.
  - Key distribution, storage, and revocation can present difficulties similar to those in other public key infrastructures.
- Side-Channel Attacks:
  - ECC implementations can be vulnerable to side-channel attacks, such as timing and power analysis attacks, if not carefully protected against these threats.

# Limitations and challenges of ECC

- Misuse of Weak Curves:
  - The security relies heavily on the choice of elliptic curves.
  - Using weak or non-standard curves can introduce vulnerabilities.
  - It is essential to use well-established and standardized curves with known security properties.
- Lack of Post-Quantum Security:
  - While ECC is more resilient to quantum attacks than some other cryptographic techniques, it is still a topic of ongoing research and debate.
  - The long-term security of ECC in a post-quantum world remains uncertain.

# Disadvantages of ECC

- Complicated and tricky to implement securely, mainly the standard curves.
- Standards aren't state-of-the-art, particularly ECDSA, which is a hack compared to Schnorr signatures.
- Newer algorithms could theoretically have unknown weaknesses. Binary curves are slightly scary.
- Signing with a broken or compromised random number generator compromises the key.
- It still has some patent problems, especially for binary curves. It might be costly...
- Public key operations (e.g., signature verification, as opposed to signature generation) are slow with ECC.

# Applications of ECC

- Secure Communication Protocols:
  - TLS/SSL: ECC is used in the Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols to secure data transmission over the internet, including secure web browsing (HTTPS).
- Digital Signatures:
  - ECDSA (Elliptic Curve Digital Signature Algorithm): ECC is employed for digital signatures, ensuring message authenticity and integrity in various applications, including email, software updates, and document verification.



# Applications of ECC

- Key Exchange:
  - ECDH (Elliptic Curve Diffie-Hellman): ECC is utilized in ECDH for secure key exchange between parties over insecure communication channels, facilitating secure symmetric key establishment for encryption.
- Cryptographic Libraries:
  - ECC is a fundamental component of many cryptographic libraries and frameworks, making it accessible to developers for building secure applications.
- IoT Security:
  - ECC's efficiency and small key sizes make it suitable for securing communication in resource-constrained Internet of Things (IoT) devices, ensuring data privacy and integrity in IoT applications.

# Applications of ECC

- Mobile Devices:
  - ECC is used in mobile devices, such as smartphones, to secure data transmission, app authentication, and device encryption. Its efficiency is especially beneficial in mobile environments.
- Wireless Communication:
  - ECC is employed in wireless communication standards like Bluetooth and Wi-Fi (WPA3) to secure data exchange between devices and networks.
- Digital Rights Management (DRM):
  - ECC can be used in DRM systems to protect copyrighted content and ensure that only authorized users can access and decrypt protected content.

# Applications of ECC

- Secure Shell (SSH):
  - ECC can be used in SSH for secure remote login and data transfer, providing a secure alternative to traditional password-based authentication.
- Smart Cards and Secure Tokens:
  - ECC is used in smart cards and hardware security tokens for secure authentication, access control, and digital signatures, providing strong security in a compact form factor.
- Blockchain and Cryptocurrencies:
  - ECC plays a critical role in many blockchain platforms and cryptocurrencies, such as Bitcoin, for generating public and private keys, signing transactions, and securing the blockchain ledger.

# Applications of ECC

- Secure Messaging and Chat Applications:
  - ECC is used in secure messaging and chat applications to ensure end-to-end encryption and message authenticity, protecting user privacy.
- Government and Military Applications:
  - ECC is used in government and military systems for secure communications, classified information protection, and digital signatures in various security-critical applications.
- Cloud Computing:
  - ECC can be used to secure data transmission and authentication in cloud computing environments, ensuring data privacy and integrity in the cloud.
- Payment Systems and Financial Services:
  - ECC is employed in payment systems, digital wallets, and financial services for securing online transactions and protecting sensitive financial data.

# Design a system for secure communication