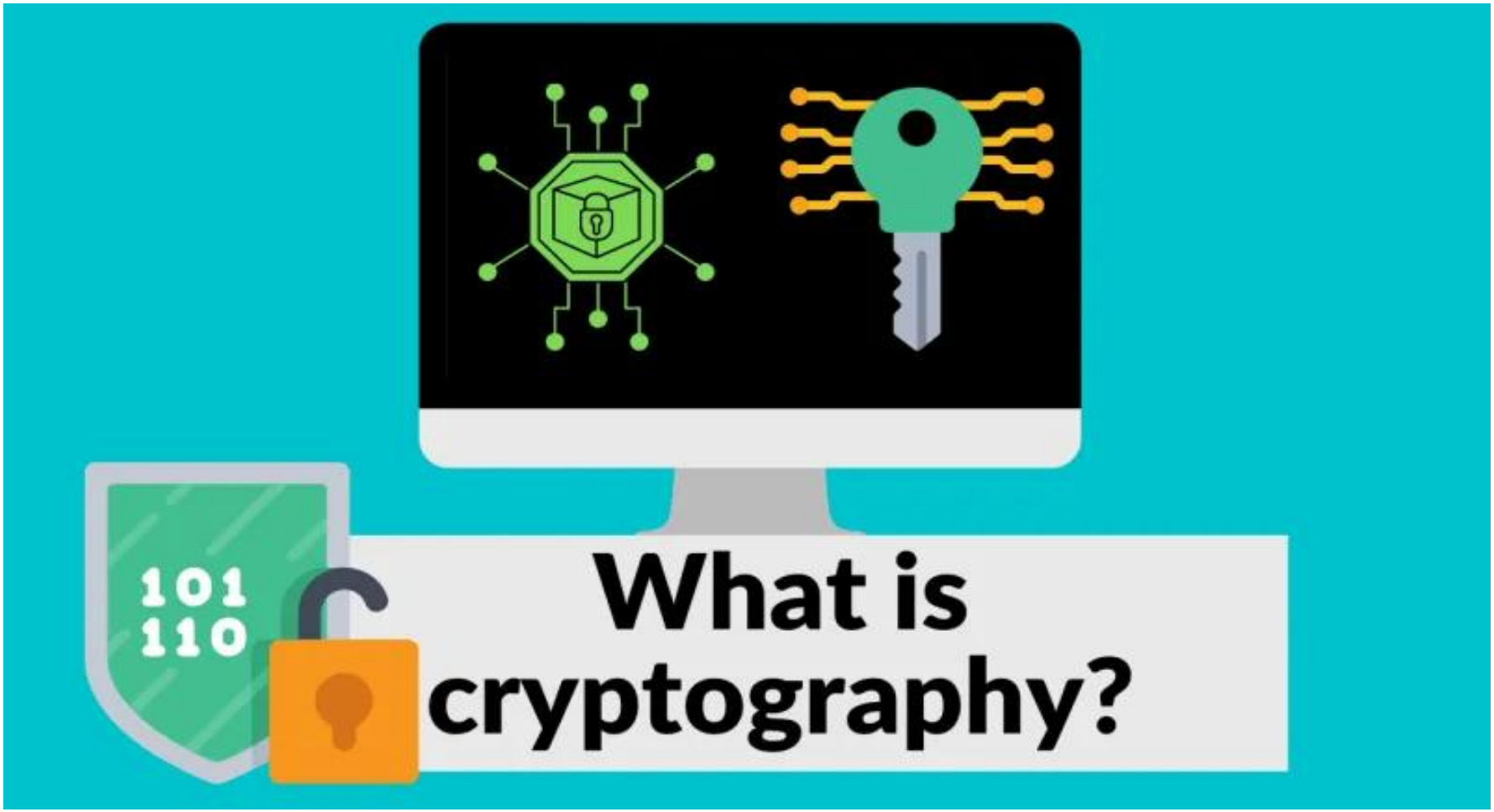


Rüya ÇELİK
xxxxx

Kadriye
xxxxxx

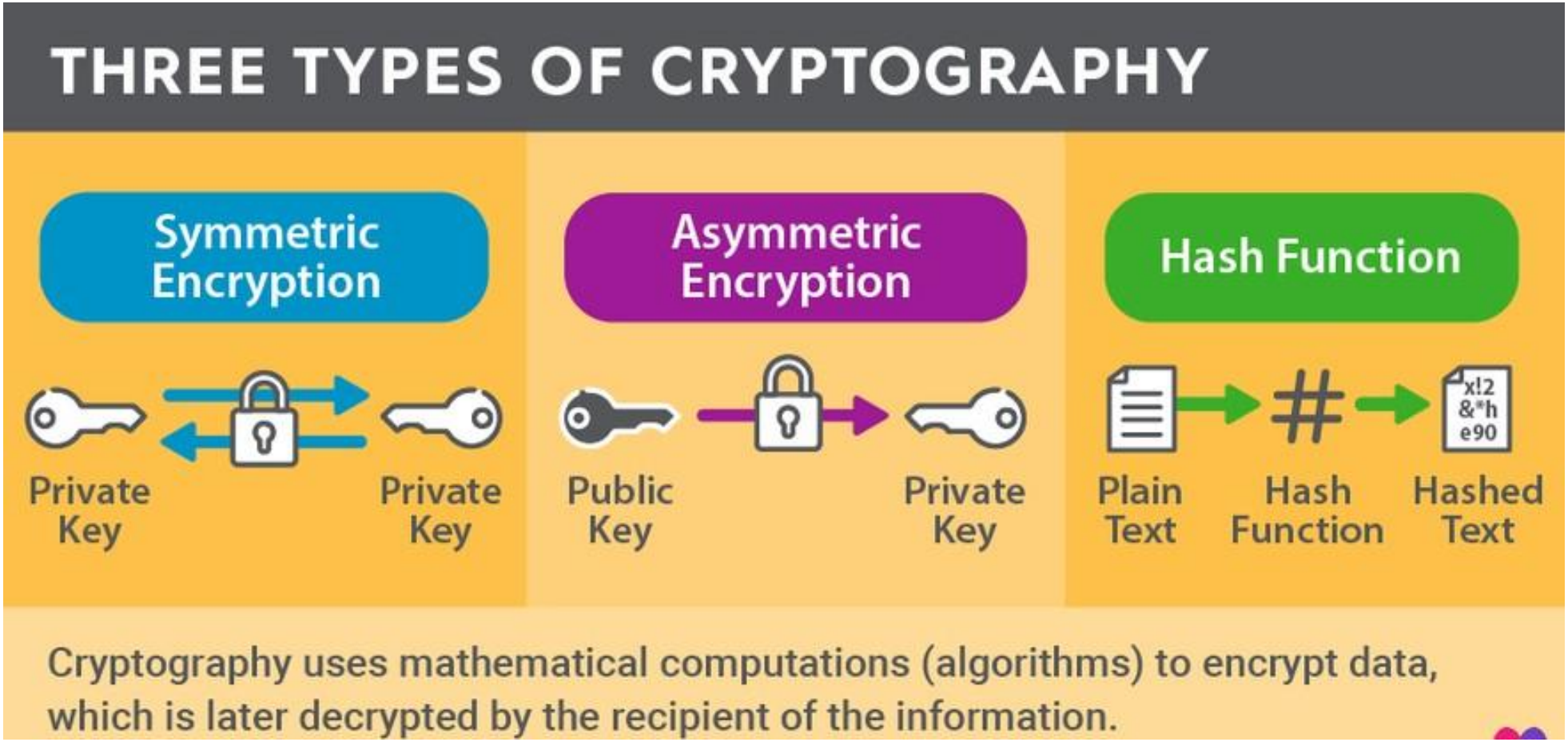
Department of Computer Engineering, Alanya University

30.12.2024

What is Cryptography?

Cryptography is the practice of securing communication and information through the use of mathematical techniques, algorithms, and protocols to protect data from unauthorized access, alteration, or disclosure.

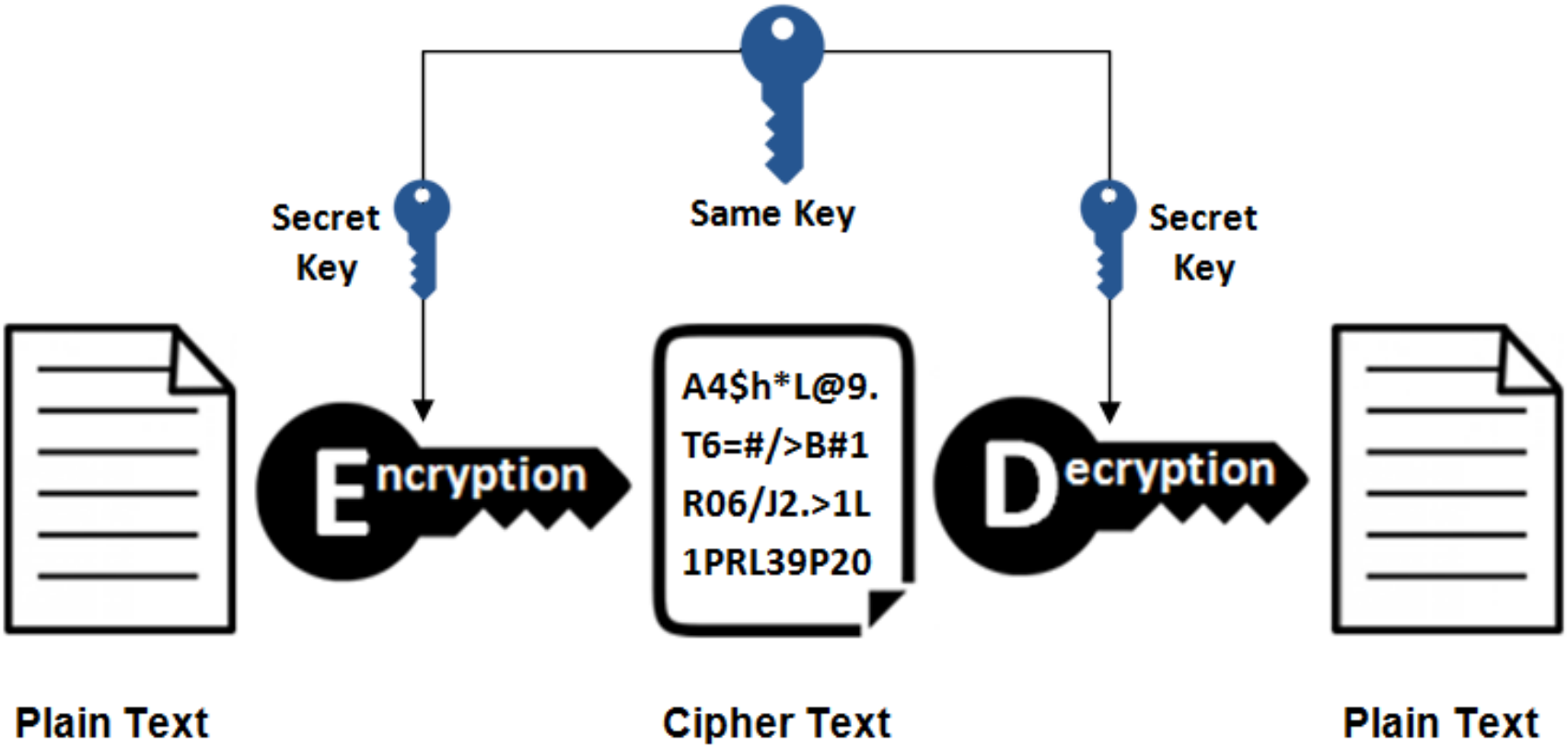
Techniques of Cryptography:



1-) Symmetric Encryption (Secret Key Cryptography)

- Both sender and receiver use the same key to encrypt and decrypt messages.
- Example: AES (Advanced Encryption Standard), DES (Data Encryption Standard).

Symmetric Encryption



java

Kodu kopyala

```java
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import java.util.Base64;

public class AESExample {

    // Encryption function
    public static String encrypt(String data, String key) throws Exception {
        SecretKeySpec secretKey = new SecretKeySpec(key.getBytes(), "AES");
        Cipher cipher = Cipher.getInstance("AES");
        cipher.init(Cipher.ENCRYPT_MODE, secretKey);
        byte[] encryptedData = cipher.doFinal(data.getBytes());
        return Base64.getEncoder().encodeToString(encryptedData);
    }

    // Decryption function
    public static String decrypt(String encryptedData, String key) throws Exception {
        SecretKeySpec secretKey = new SecretKeySpec(key.getBytes(), "AES");
        Cipher cipher = Cipher.getInstance("AES");
        cipher.init(Cipher.DECRYPT_MODE, secretKey);
        byte[] decodedData = Base64.getDecoder().decode(encryptedData);
        byte[] decryptedData = cipher.doFinal(decodedData);
        return new String(decryptedData);
    }

    public static void main(String[] args) {
        try {
            // Data to encrypt
            String originalData = "Hello, this is a secret message!";
            // AES requires a 16-byte (128-bit) key
            String key = "1234567890123456"; // 16-byte key

            // Encrypt the data
            String encryptedData = encrypt(originalData, key);
            System.out.println("Encrypted Data: " + encryptedData);

            // Decrypt the encrypted data
            String decryptedData = decrypt(encryptedData, key);
            System.out.println("Decrypted Data: " + decryptedData);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

When you run this program, you'll get the following output:

```
Encrypted Data: 9R6k5bXtZJTXHHoxpxLCpQ==
Decrypted Data: Hello, this is a secret message!
```

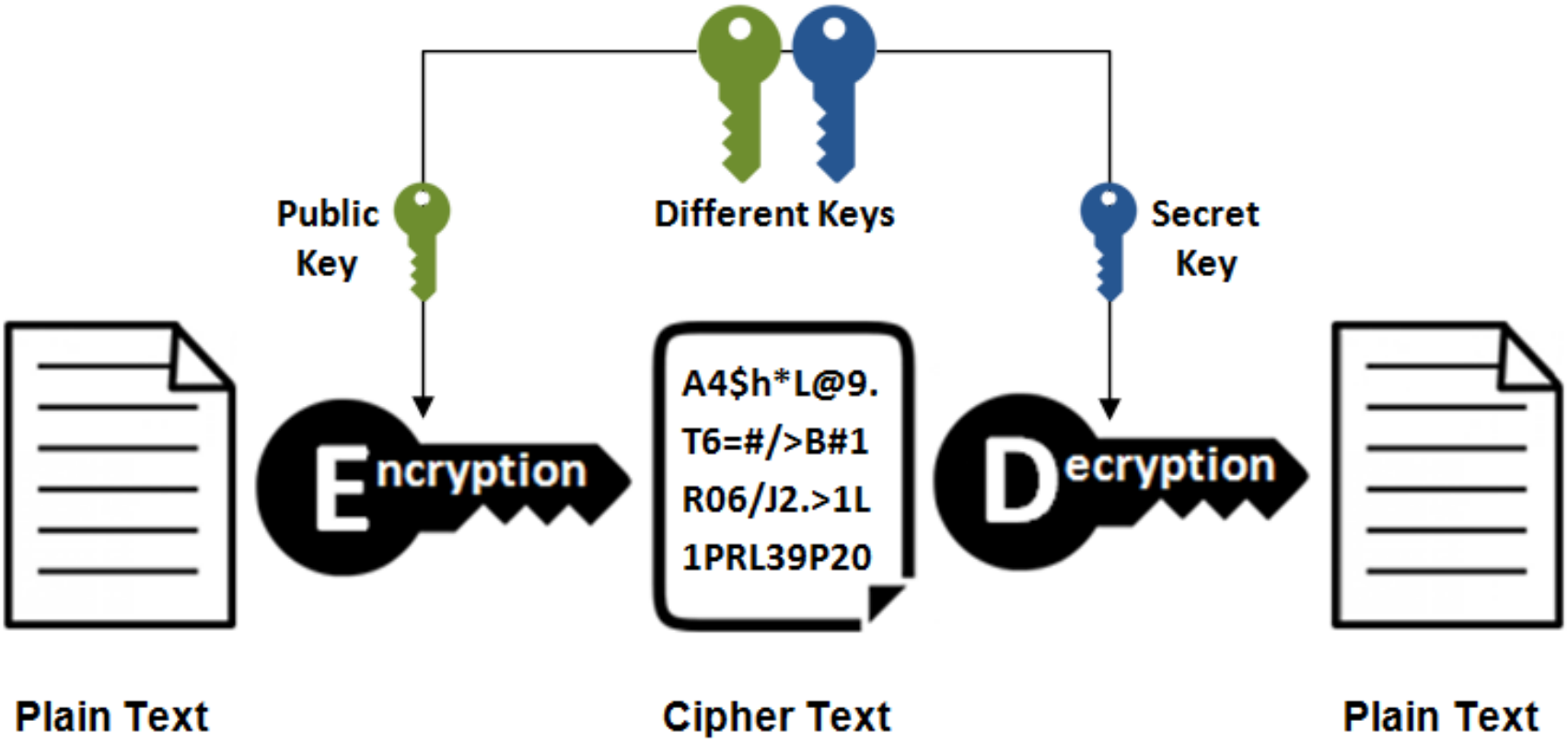
Steps in the Code:

1. **Original Data**: `"Hello, this is a secret message!"`
2. **AES Encryption**: The original data is encrypted using the AES algorithm.
3. **AES Decryption**: The encrypted data is decrypted with the correct key, and the original message is retrieved.

2-) Asymmetric Encryption (Public Key Cryptography)

- Uses a pair of keys: a public key for encryption and a private key for decryption.
- Example: RSA (Rivest-Shamir-Adleman), ECC (Elliptic Curve Cryptography).

Asymmetric Encryption



java Kodu kopyala

```java
import javax.crypto.Cipher;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PublicKey;
import java.security.PrivateKey;
import java.util.Base64;

public class AsymmetricEncryptionExample {

    public static void main(String[] args) throws Exception {
        // Generate RSA key pair
        KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance("RSA");
        keyPairGenerator.initialize(2048); // 2048-bit key length
        KeyPair keyPair = keyPairGenerator.generateKeyPair();

        // Extract public and private keys
        PublicKey publicKey = keyPair.getPublic();
        PrivateKey privateKey = keyPair.getPrivate();

        // Message to encrypt
        String message = "This is an asymmetric encryption example.";

        // Encrypt the message
        String encryptedMessage = encryptMessage(message, publicKey);
        System.out.println("Encrypted Message: " + encryptedMessage);

        // Decrypt the message
        String decryptedMessage = decryptMessage(encryptedMessage, privateKey);
        System.out.println("Decrypted Message: " + decryptedMessage);
    }

    public static String encryptMessage(String message, PublicKey publicKey) throws Exception {
        Cipher cipher = Cipher.getInstance("RSA");
        cipher.init(Cipher.ENCRYPT_MODE, publicKey);
        byte[] encryptedBytes = cipher.doFinal(message.getBytes());
        return Base64.getEncoder().encodeToString(encryptedBytes);
    }

    public static String decryptMessage(String encryptedMessage, PrivateKey privateKey) throws Exception {
        Cipher cipher = Cipher.getInstance("RSA");
        cipher.init(Cipher.DECRYPT_MODE, privateKey);
        byte[] decodedBytes = Base64.getDecoder().decode(encryptedMessage);
        byte[] decryptedBytes = cipher.doFinal(decodedBytes);
        return new String(decryptedBytes);
    }
}
```

When you run this program, you'll get the following output:

```
Encrypted Message: aGVsbG8gdGhpcyBpcyBhIHNlY3JldCBtZXNzYWdlLg==
Decrypted Message: This is an asymmetric encryption example.
```

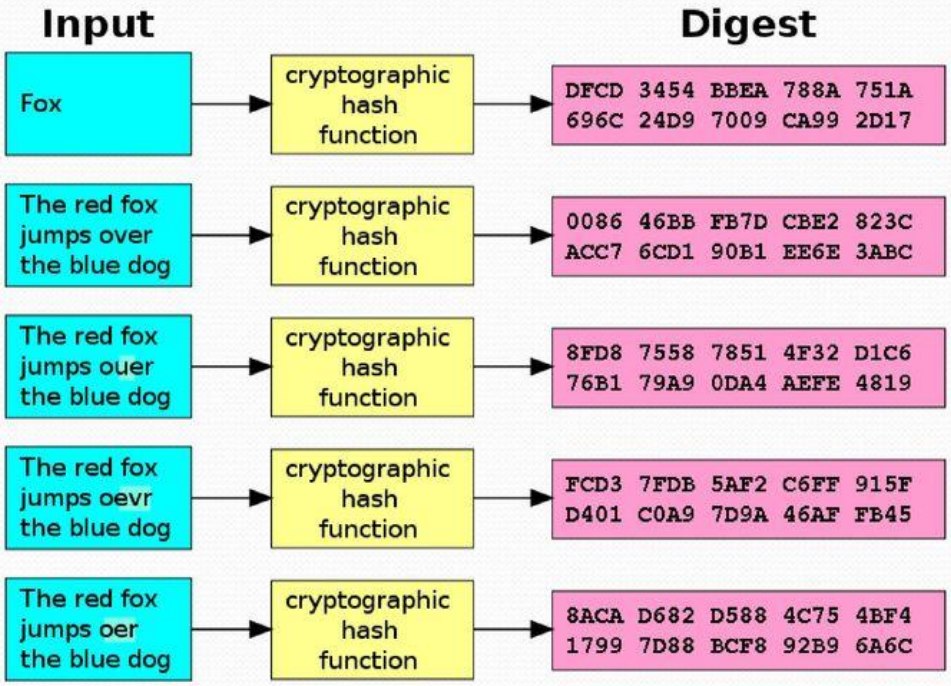
Summary of Steps:

1. **Generate Key Pair**: Use **KeyPairGenerator** to generate public and private keys.
2. **Encrypt Message**: Use the **public key** to encrypt the original message.
3. **Base64 Encode**: Convert the encrypted message to a readable Base64 string.
4. **Decrypt Message**: Use the **private key** to decrypt the Base64 encoded message.
5. **Retrieve Original Message**: Convert the decrypted byte array back to the original string.

3-) Hash Functions

- Converts input data into a fixed-size string (hash), which is typically irreversible.
- Used for data integrity and authentication.
- Example: SHA (Secure Hash Algorithm), MD5 (Message Digest Algorithm 5).

Secured Hash Function(SHF)



java Kodu kopyala

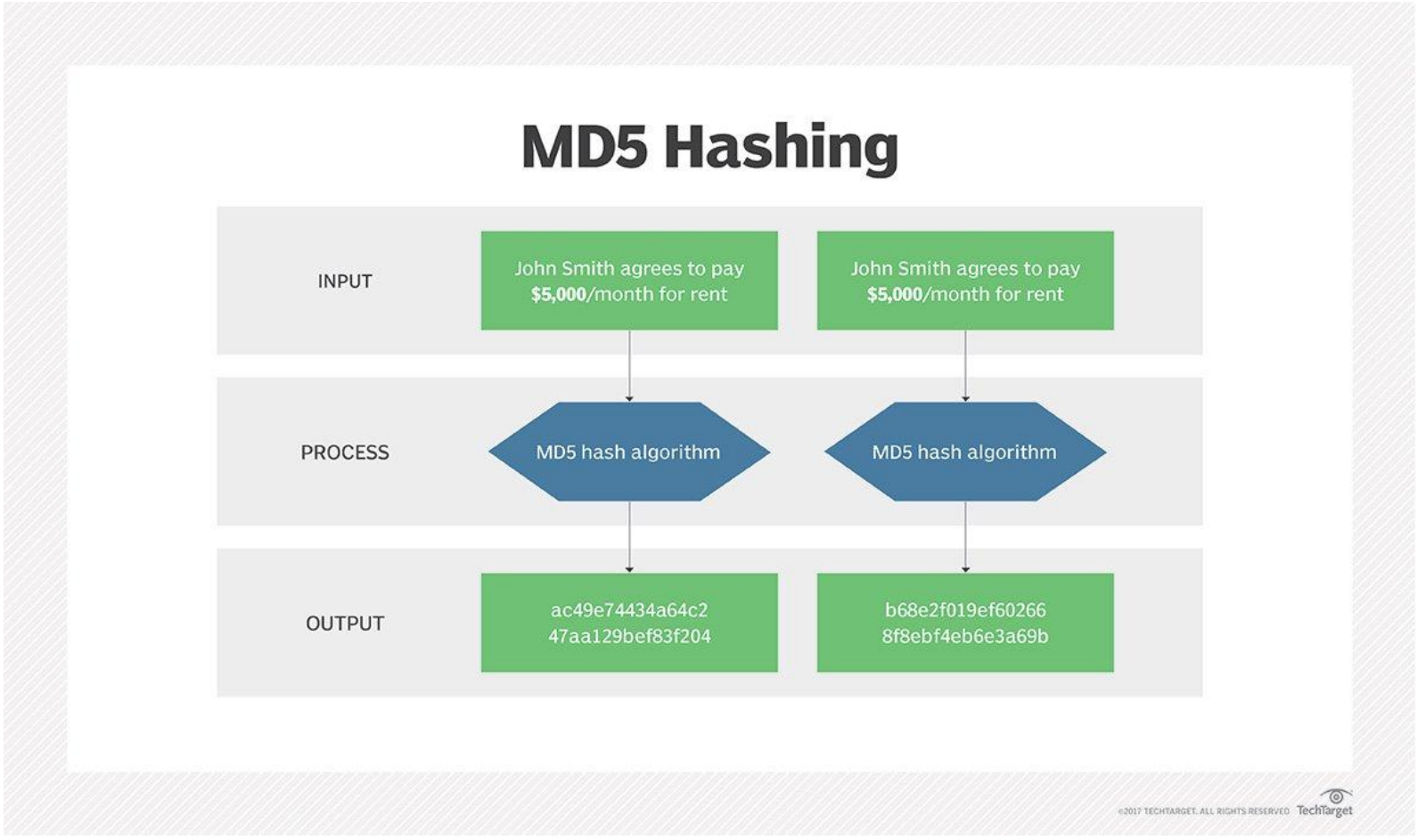
```java
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class SHA256Example {

    public static void main(String[] args) {
        // Input message to be hashed
        String message = "This is an example of SHA-256 hashing.";

        try {
            // Get SHA-256 MessageDigest instance
            MessageDigest digest = MessageDigest.getInstance("SHA-256");

            // Perform the hashing
            byte[] hashBytes = digest.digest(message.getBytes());

            // Convert the hash bytes to a hexadecimal format string
            StringBuilder hexString = new StringBuilder();
            for (byte b : hashBytes) {
                hexString.append(String.format("%02x", b));  // Convert byte to hex format
            }
            System.out.println("Original Message: " + message);
            System.out.println("SHA-256 Hash: " + hexString.toString());

        } catch (NoSuchAlgorithmException e) {
            // Handle the exception when the algorithm is not available
            System.out.println("Hashing algorithm not found.");
            e.printStackTrace();
        }
    }
}
```

When you run this program, you'll get the following output:

```
Original Message: This is an example of SHA-256 hashing.
SHA-256 Hash: 7509e5bda0c762d2d7d6e9b6d7e8be75f7c3ad8ccfefb9c702b7671d4c9b0e71
```

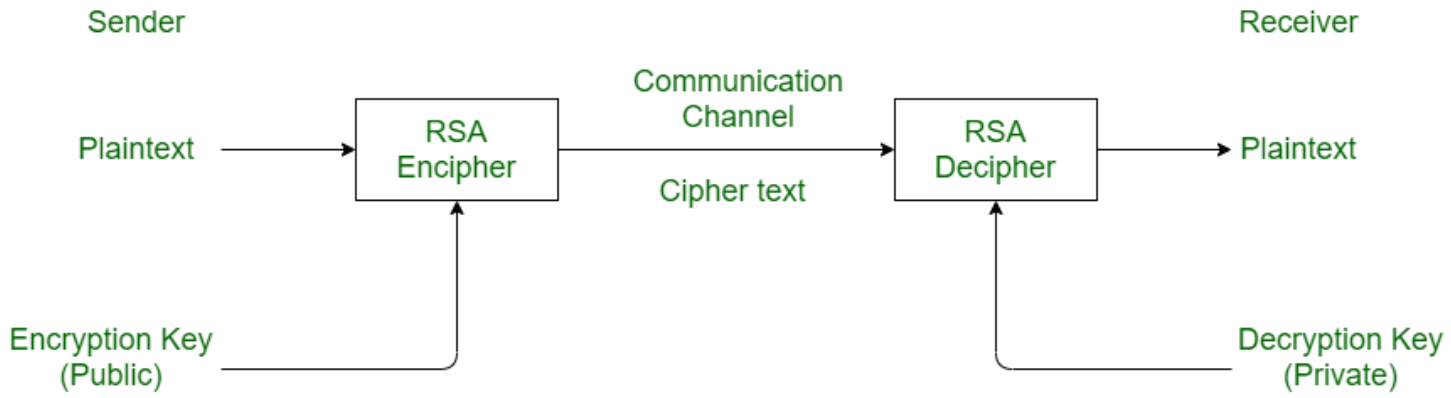
Summary:

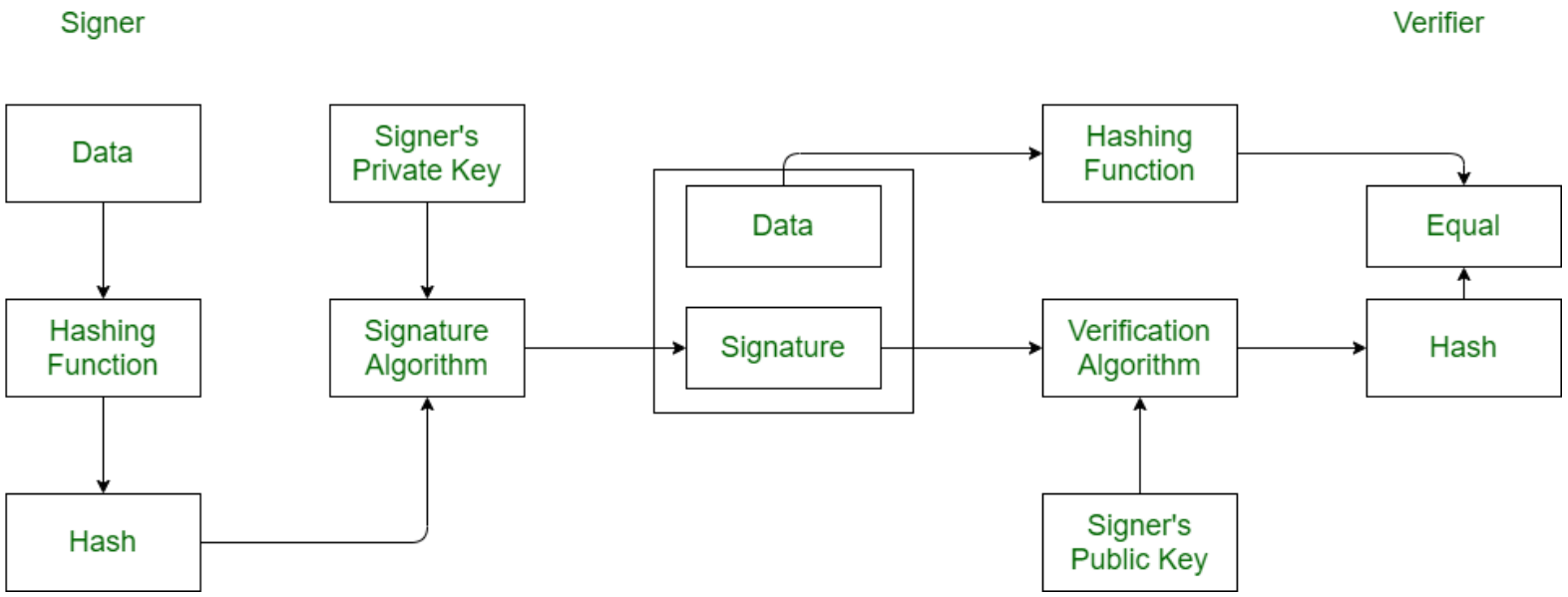
- **Step 1**: Define the message.
- **Step 2**: Get the SHA-256 hashing instance.
- **Step 3**: Compute the hash of the message.
- **Step 4**: Convert the hash to hexadecimal format.
- **Step 5**: Output the original message and its hash.


MD5 Hashing
INPUT
John Smith agrees to pay $5,000/month for rent
John Smith agrees to pay $5,000/month for rent
PROCESS
MD5 hash algorithm
MD5 hash algorithm
OUTPUT
ac49e74434a64c2 47aa129bef83f204
b68e2f019ef60266 8f8ebf4eb6e3a69b
©2017 TECHTARGET. ALL RIGHTS RESERVED
TechTarget

Digital Signatures

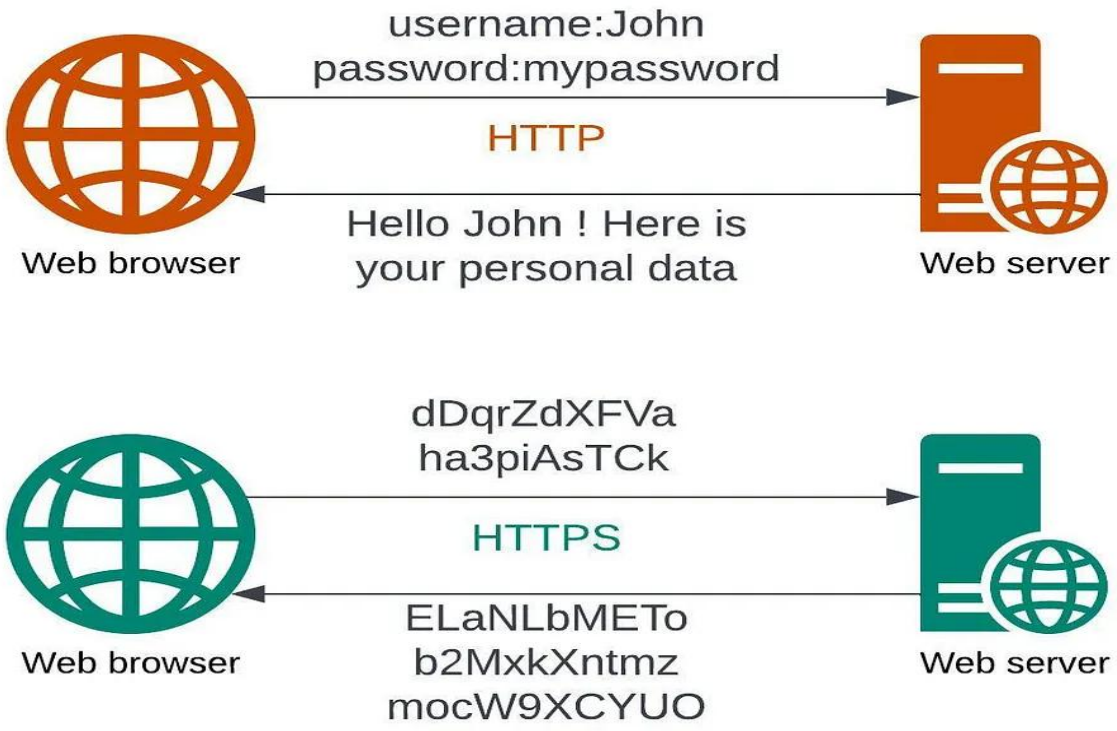
- A method for verifying the authenticity and integrity of a message or document using asymmetric encryption.
- Example: RSA (Rivest-Shamir-Adleman), DSA (Digital Signature Algorithm).


Sender
Receiver
Communication
Channel
Plaintext
RSA
Encipher
RSA
Decipher
Plaintext
Cipher text
Encryption Key
(Public)
Decryption Key
(Private)

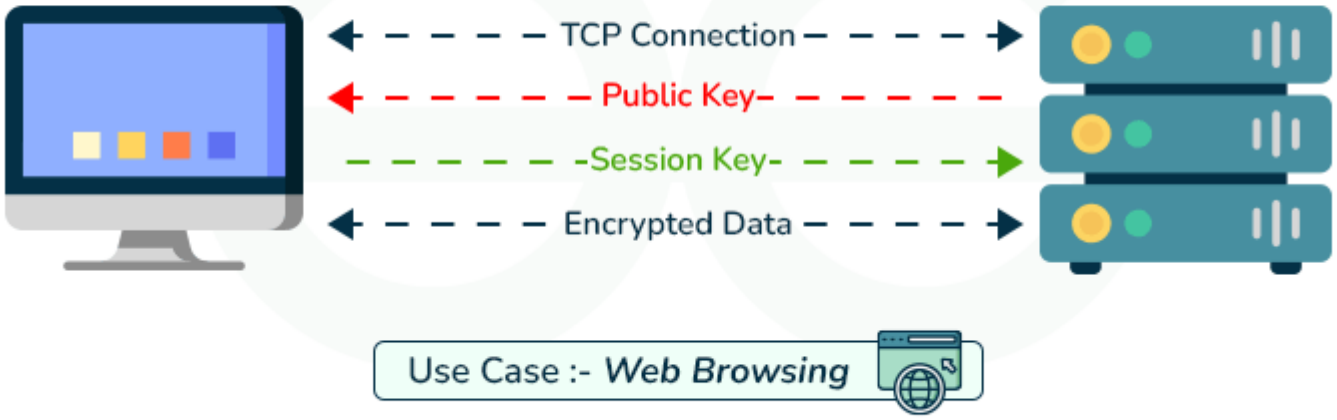

Signer
Verifier
Data
Signer's Private Key
Hashing Function
Signature Algorithm
Hash
Data
Signature
Hashing Function
Equal
Verification Algorithm
Hash
Signer's Public Key

Cryptographic Protocols

- Frameworks for secure communication, such as key exchange and authentication methods.
- Example: SSL/TLS (Secure Sockets Layer/Transport Layer Security), HTTPS (Hypertext Transfer Protocol Secure).

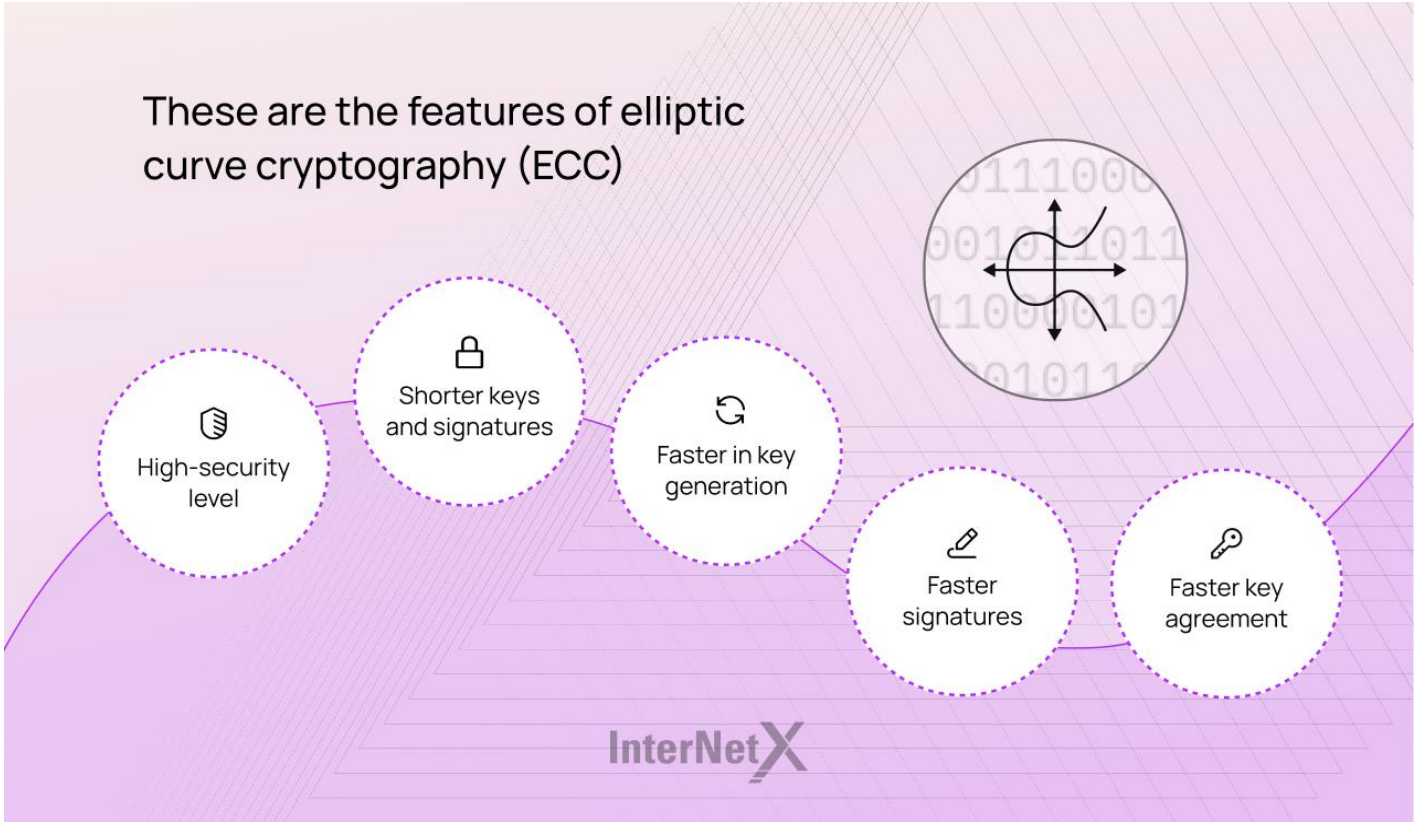

username:John
password:mypassword
HTTP
Hello John ! Here is
your personal data
Web browser
Web server
dDqrZdXFVa
ha3piAsTCk
HTTPS
ELaNLbMETo
b2MxkXntmz
mocW9XCYUO
Web browser
Web server

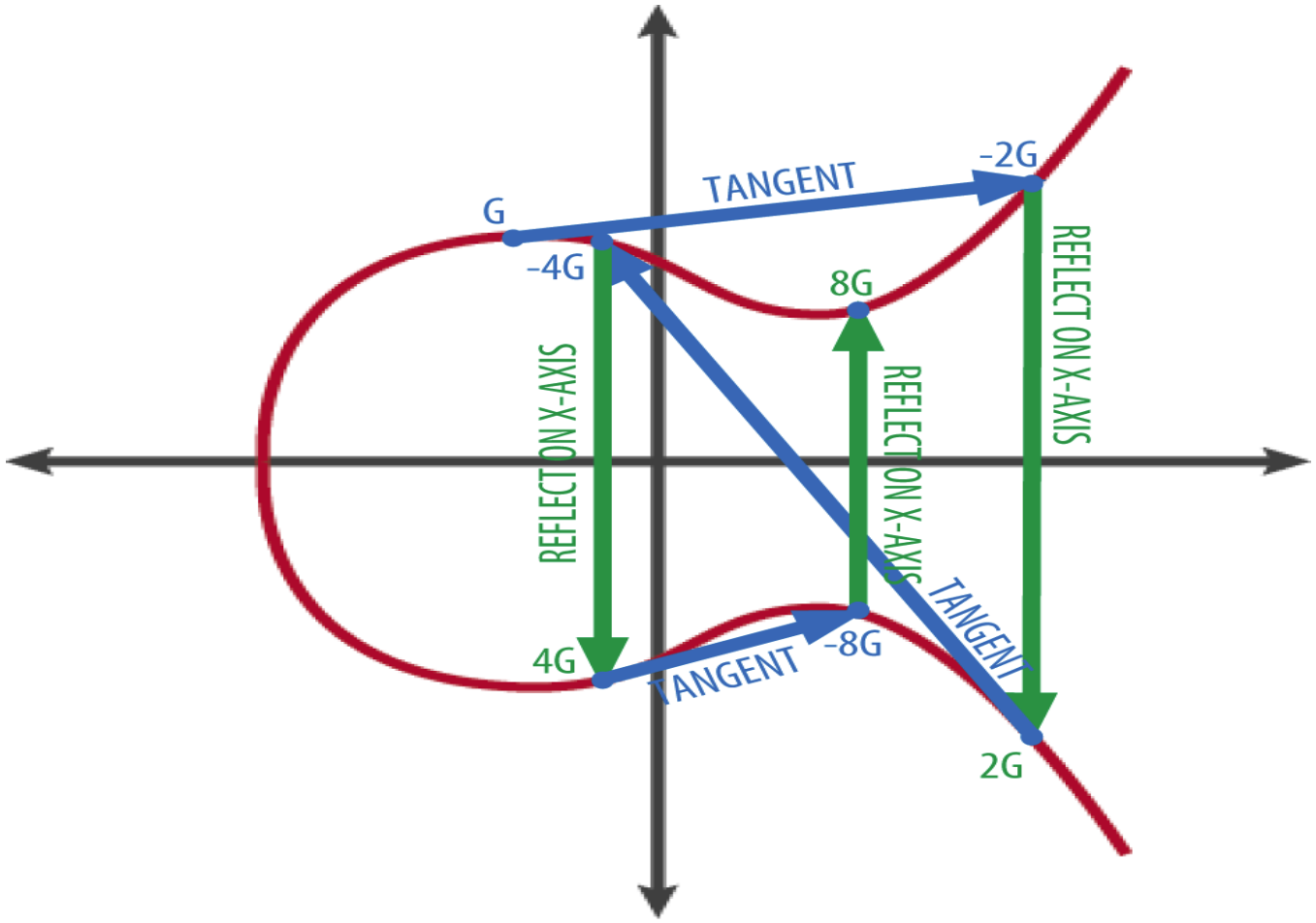
HTTPS


TCP Connection
Public Key
-Session Key-
Encrypted Data
Use Case :- Web Browsing

Elliptic Curve Cryptography (ECC)

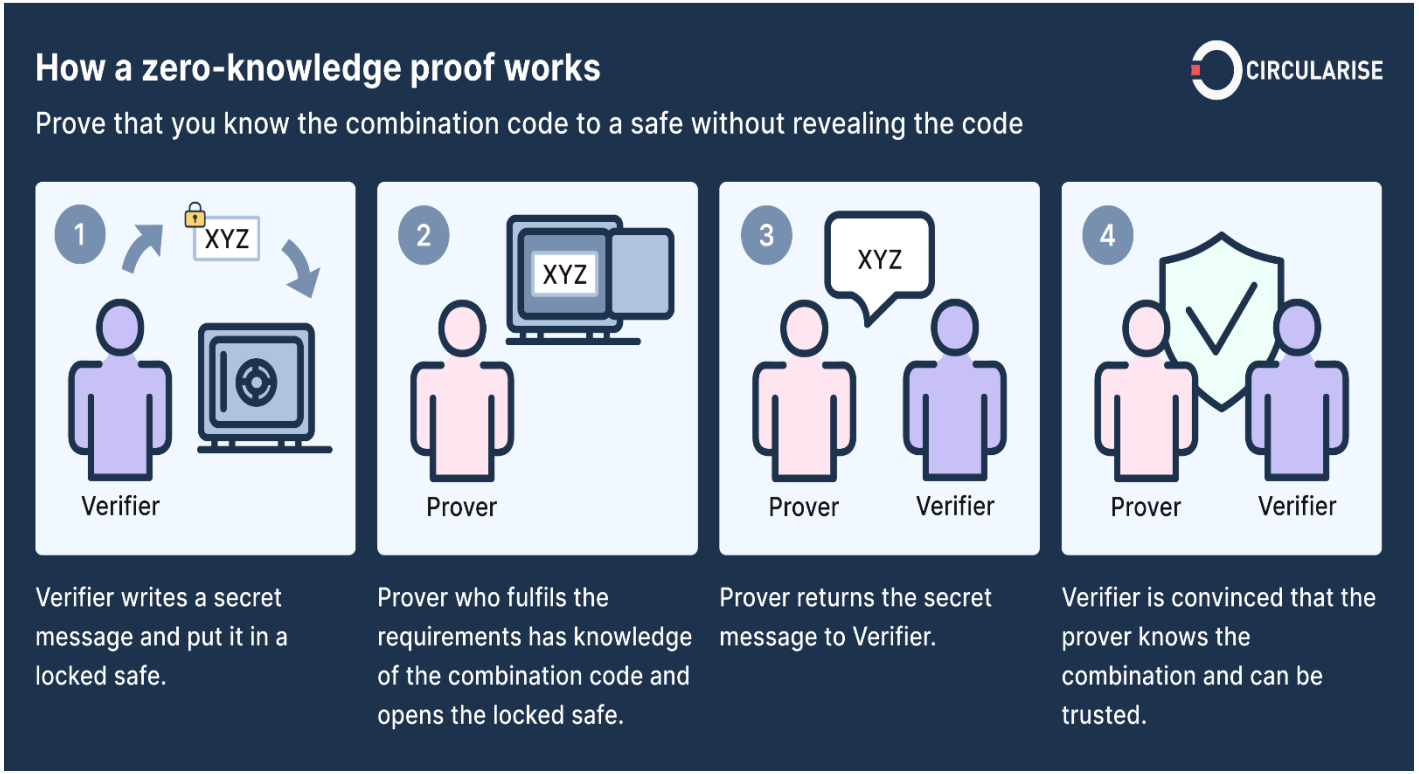
- A form of asymmetric encryption based on the mathematics of elliptic curves.
- Offers higher security with smaller key sizes compared to RSA.


These are the features of elliptic curve cryptography (ECC)
High-security level
Shorter keys and signatures
Faster in key generation
Faster signatures
Faster key agreement
InterNetX


G
TANGENT
-2G
-4G
8G
REFLECT ON X-AXIS
REFLECT ON X-AXIS
REFLECT ON X-AXIS
TANGENT
4G
TANGENT
-8G
2G

Zero-Knowledge Proofs

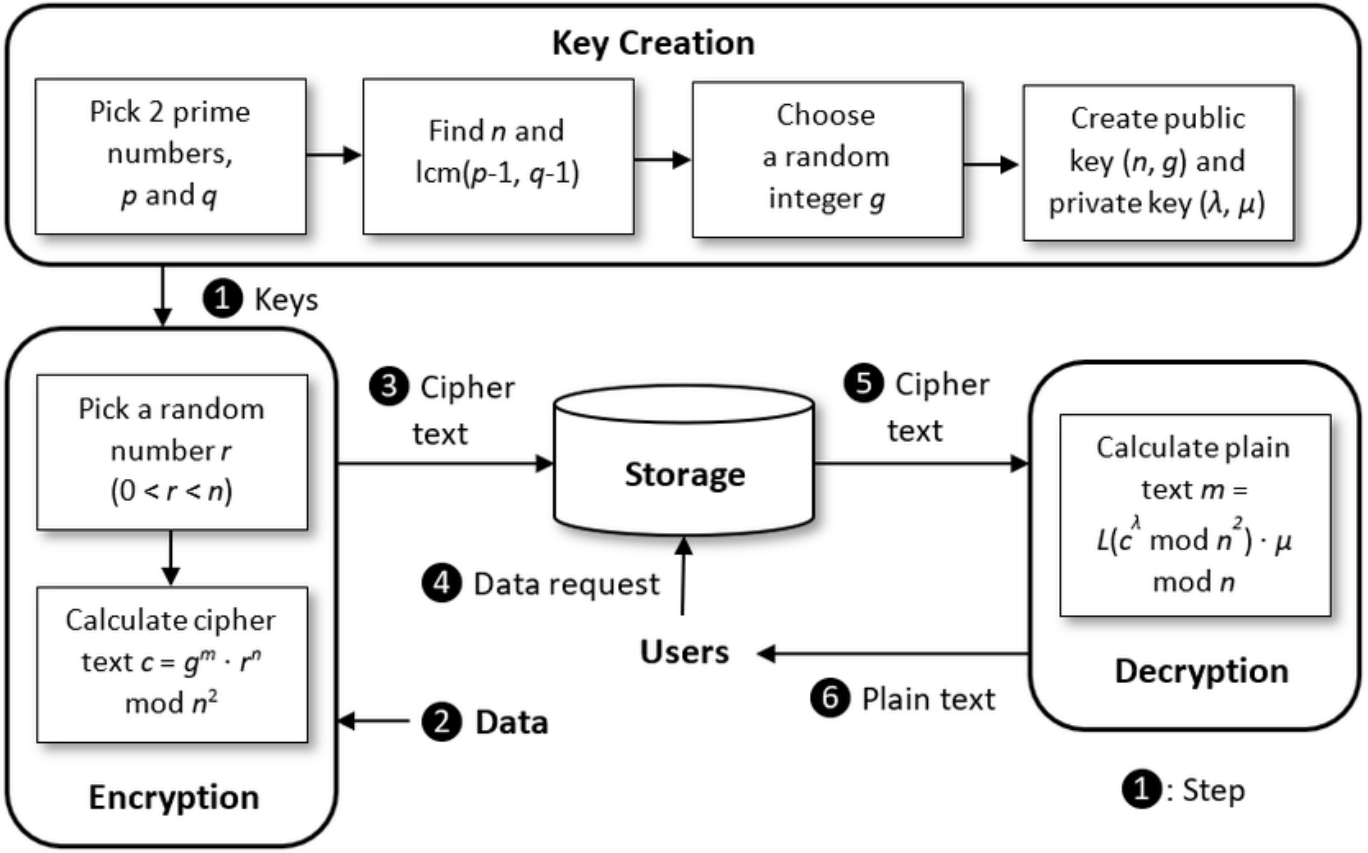
- Allows one party to prove to another that they know a secret without revealing the secret itself.
- Used in privacy-focused systems like Zcash.


How a zero-knowledge proof works
CIRCULARISE
Prove that you know the combination code to a safe without revealing the code
1
XYZ
Verifier
2
XYZ
Prover
3
XYZ
Prover
Verifier
4
Prover
Verifier
Verifier writes a secret message and put it in a locked safe.
Prover who fulfils the requirements has knowledge of the combination code and opens the locked safe.
Prover returns the secret message to Verifier.
Verifier is convinced that the prover knows the combination and can be trusted.


B
Bob does not know what path Alice took inside the cave, but randomly chooses a path to ask Alice to take back to the entrance. This process repeats multiple times.
B
A
Alice selected a path randomly within the cave to get to the passcode locked door

Homomorphic Encryption

- Enables computation on encrypted data without needing to decrypt it first, preserving privacy during processing.
- Example: Paillier encryption.


Key Creation
Pick 2 prime numbers, p and q
Find n and lcm(p-1, q-1)
Choose a random integer g
Create public key (n, g) and private key (λ, μ)
1 Keys
Pick a random number r (0 < r < n)
Calculate cipher text c = g^m · r^n mod n^2
Encryption
3 Cipher text
Storage
5 Cipher text
4 Data request
2 Data
Users
6 Plain text
Calculate plain text m = L(c^λ mod n^2) · μ mod n
Decryption
1 : Step

Encryption:

plaintext $m < n$

select a random $r < n$

cipherext $C = g^m r^n \bmod n^2$

Decryption:

$L(\mu) = \frac{\mu - 1}{n}$

ciphertext $C < n^2$

plaintext $m = \frac{L(C^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n$

where p, q are two big primes with equal length,

$n = pq$, $\lambda = \text{lcm}(p-1, q-1)$, $g = 1 + kn$ $(k \in Z_n^*)$

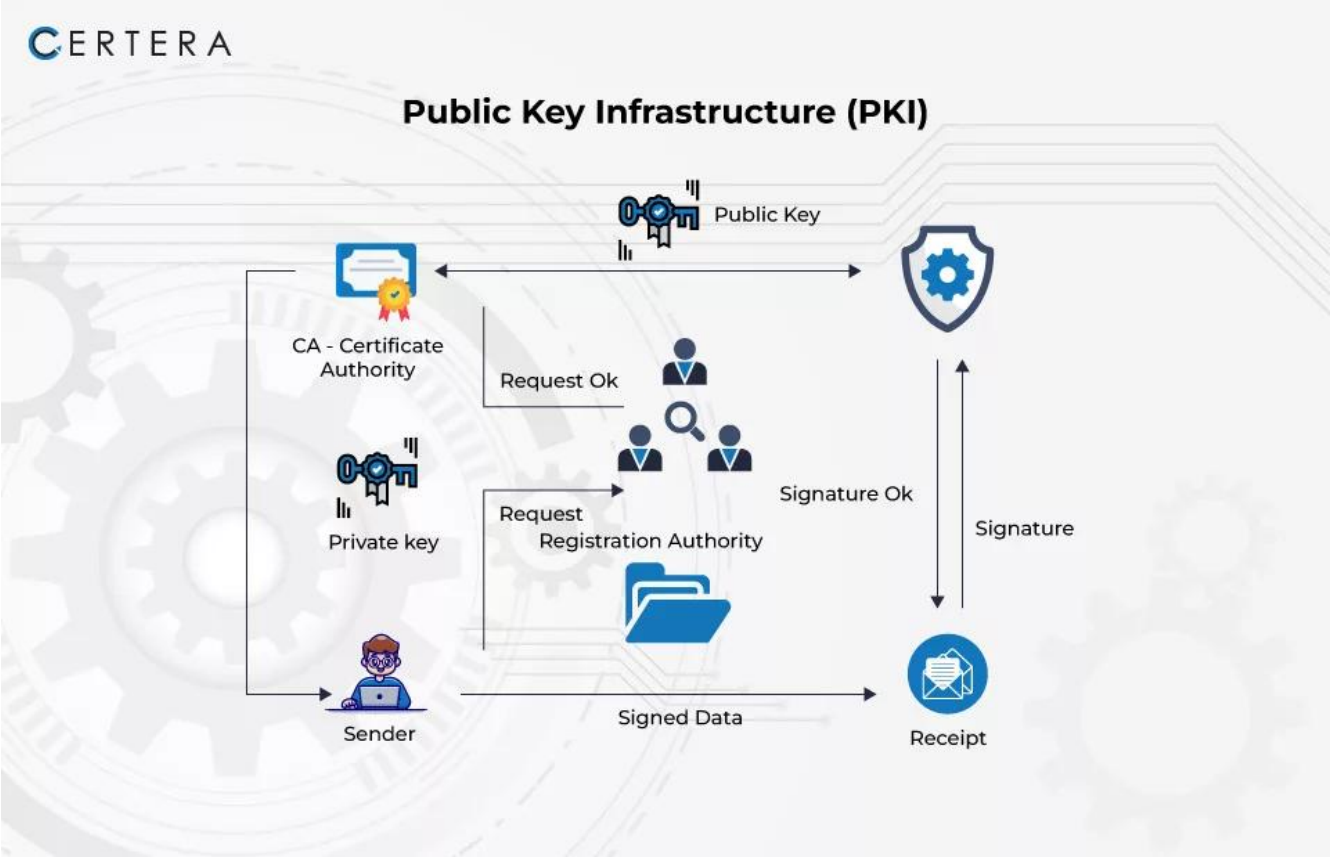
Public Key Infrastructure (PKI)

- A system for managing digital keys and certificates, ensuring secure communication and user authentication.
- Involves Certificate Authorities (CA) and registration authorities.

Encryption/Decryption

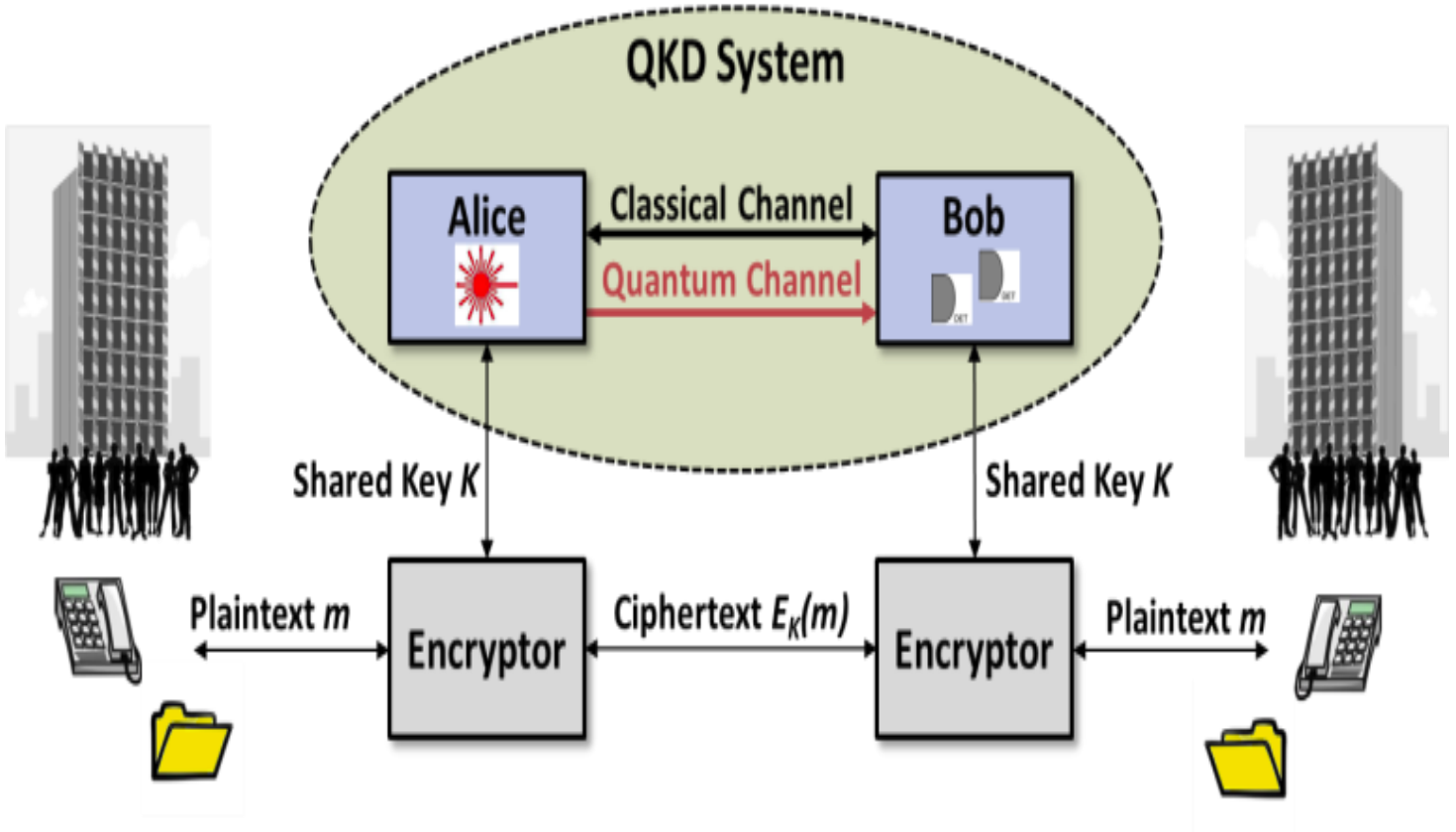
1. John uses Mary's public key to encrypt the email and sends it to Mary.

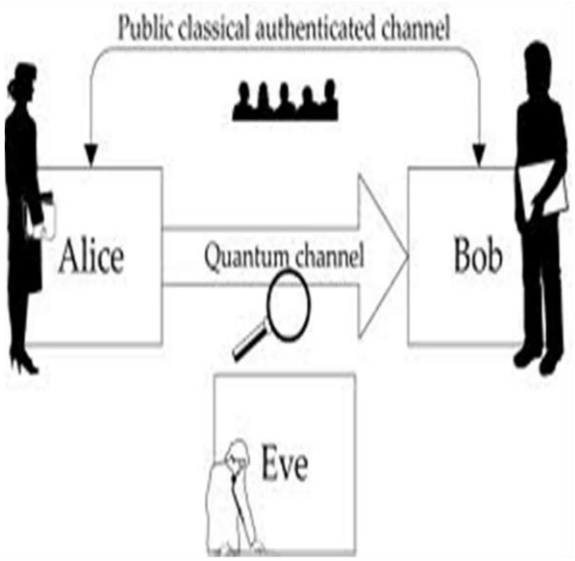
2. Upon receiving the email, Mary decrypts the email with her own private key.

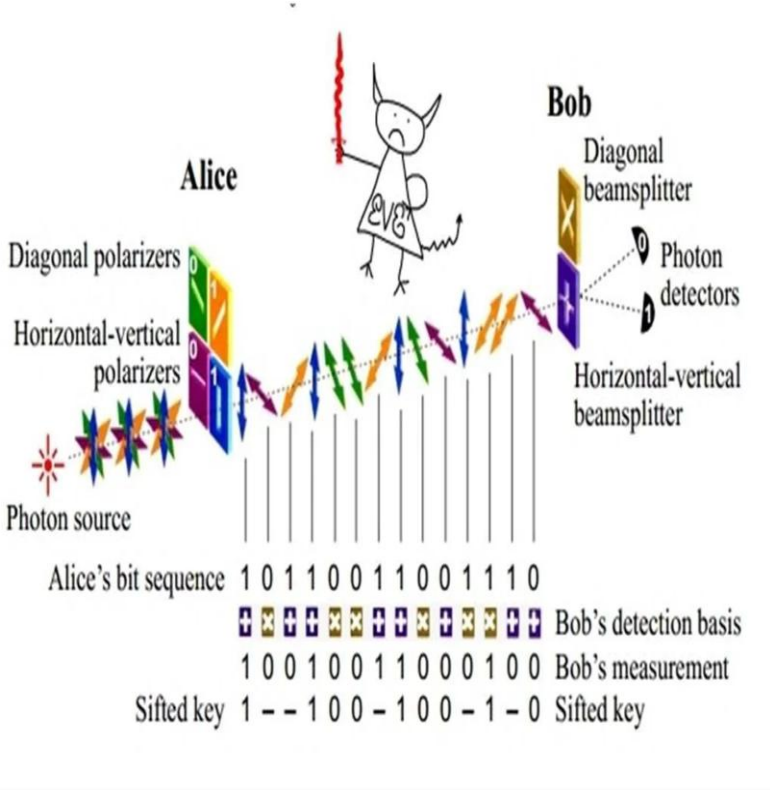

CERTERA
Public Key Infrastructure (PKI)
Public Key
CA - Certificate Authority
Request Ok
Private key
Request
Registration Authority
Signature Ok
Signature
Sender
Signed Data
Receipt

Quantum Cryptography

- Uses quantum mechanics principles to secure communication, ensuring data cannot be intercepted without detection.
- Example: Quantum Key Distribution (QKD).


QKD System
Alice
Classical Channel
Quantum Channel
Bob
Shared Key K
Shared Key K
Plaintext m
Encryptor
Ciphertext $E_K(m)$
Encryptor
Plaintext m


Public classical authenticated channel
Alice
Quantum channel
Bob
Eve

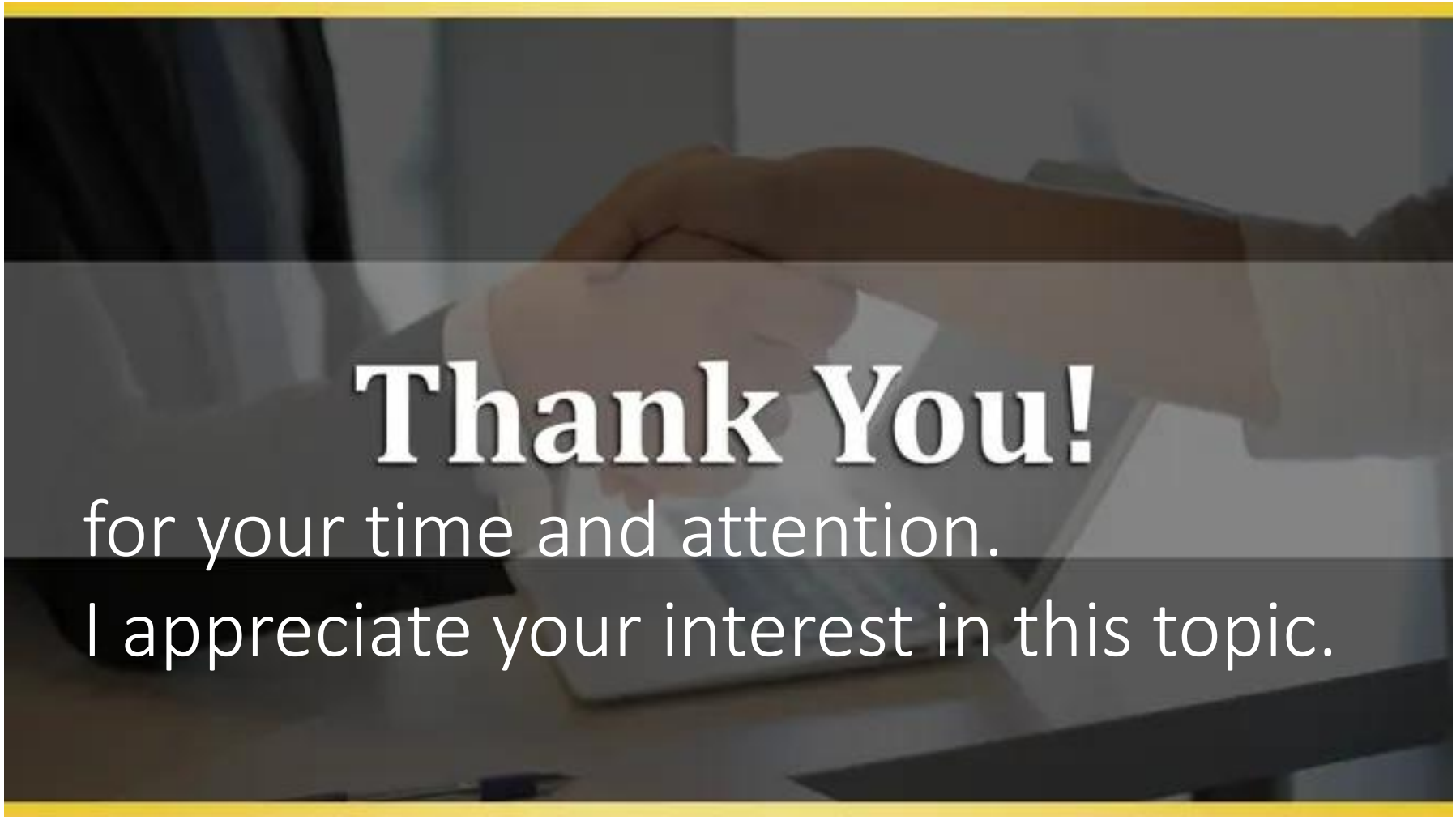

Alice
Bob
Diagonal polarizers
Horizontal-vertical polarizers
Photon source
Diagonal beamsplitter
Photon detectors
Horizontal-vertical beamsplitter
Alice's bit sequence 1 0 1 1 0 0 1 1 0 0 1 1 1 0
Bob's detection basis
1 0 0 1 0 0 1 1 0 0 0 1 0 0 Bob's measurement
Sifted key 1 – – 1 0 0 – 1 0 0 – 1 – 0 Sifted key

Conclusion

These techniques are fundamental to securing digital communication, protecting privacy, and ensuring data integrity in a variety of applications, such as online banking, email encryption, and secure communications.

Questions:

I would be happy to answer any questions you may have.


Thank You!
for your time and attention.
I appreciate your interest in this topic.