LABWORK 1: OPEN SSL

**NAME: ATAKPU EMILOMO JOYCE**

**LABWORK 1: OPEN SSL**



**What is OpenSSL?**

OpenSSL is an open-source software library that provides tools for implementing secure communication over computer networks. It is widely used for managing encryption, generating keys, creating digital signatures, and verifying certificates. OpenSSL supports several cryptographic algorithms like RSA, AES, and DSA, which are essential for securing data.

**Key Features of OpenSSL**

1. Data Encryption

OpenSSL can encrypt and decrypt data using both symmetric (same key for encryption and decryption) and asymmetric (public and private key pair) encryption algorithms.

2. Key Generation

OpenSSL can generate public and private keys for algorithms such as RSA and DSA. These keys are essential for secure communication.

3. Digital Signatures

OpenSSL allows you to create and verify digital signatures. Digital signatures provide data integrity and authentication.

4. SSL/TLS Support

OpenSSL can be used to test and configure SSL/TLS (Secure Sockets Layer/Transport Layer Security) connections. SSL/TLS is the protocol used for securing internet traffic.

5. Hashing

OpenSSL can compute hashes for data, helping ensure integrity by generating a fixed-size string (hash) for any input.

**Difference Between Symmetric and Asymmetric Encryption**

1. Key Usage

• Symmetric Encryption: Uses the same key to encrypt and decrypt data.

• Asymmetric Encryption: Uses two keys—a public key to encrypt and a private key to decrypt.

2. Speed

• Symmetric Encryption: Faster because it uses simpler methods.

• Asymmetric Encryption: Slower because it uses more complex methods.

3. Key Sharing

• Symmetric Encryption: Both the sender and receiver need to share the same key. If someone steals the key, they can read the data.

• Asymmetric Encryption: The public key can be shared openly, but the private key must be kept secret.

4. Use Cases

• Symmetric Encryption: Used for encrypting large amounts of data quickly (e.g., AES).

• Asymmetric Encryption: Used for secure communication, like sending private messages or digital signatures (e.g., RSA).

5. Security

• Symmetric Encryption: Security depends on keeping the key secret.

• Asymmetric Encryption: Security is based on keeping the private key secret. The public key can be shared safely.

Summary

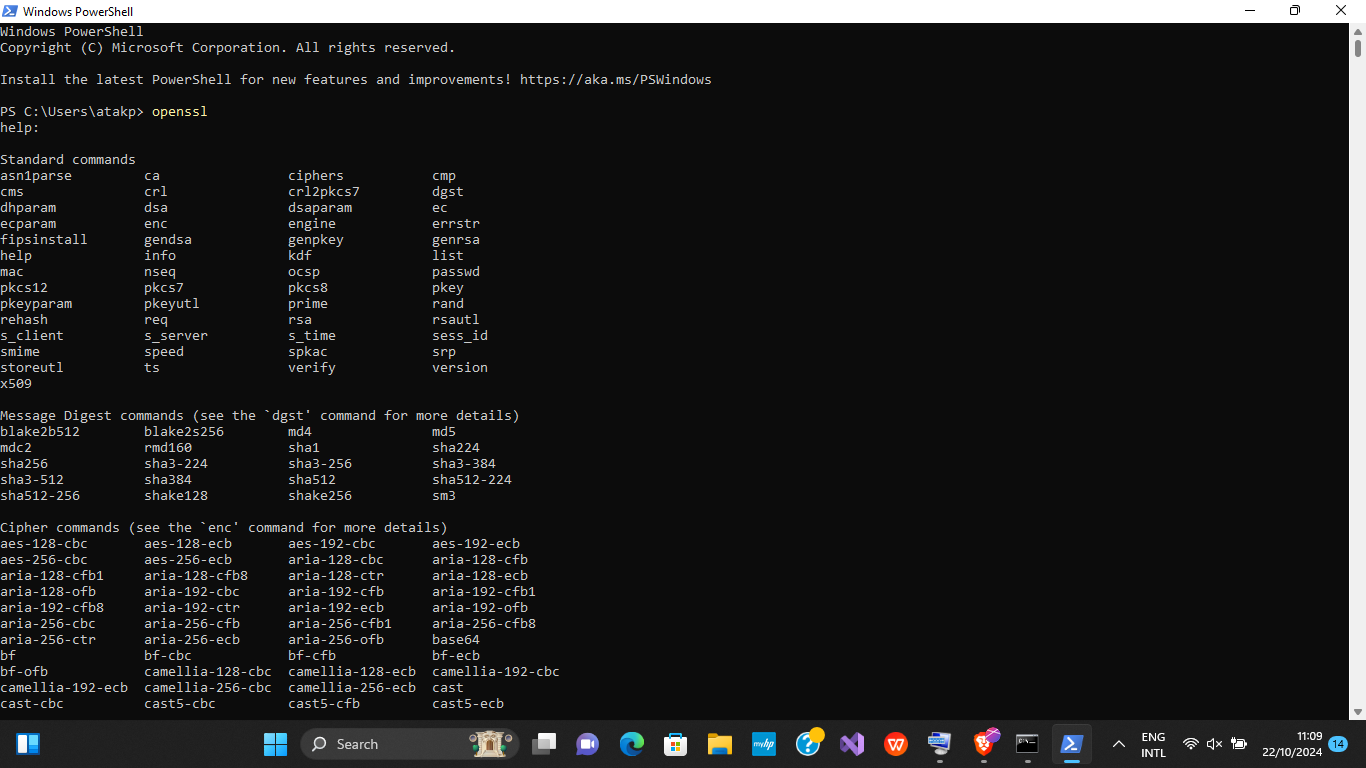
• Symmetric: Same key for both encryption and decryption. Fast but harder to manage securely.

• Asymmetric: Two keys (public and private). Slower but easier to manage and more secure for communication.

\*\*\*\*\*\*\*

Now, let’s dive into some common **OpenSSL commands** that we can use for various cryptographic tasks. These commands are essential for generating keys, encrypting data, and managing certificates.

Below when I run the command openssl in my command prompt, OpenSSL opens its interactive command-line interface. This interface allows me to run various cryptographic operations such as key generation, data encryption, and certificate management.



As we can see when I typed **openssl**, I was presented with a list of available commands like genrsa, req, x509, enc, and dgst. These commands allow you to perform various cryptographic operations. For example,

• genrsa: Generates RSA private keys.

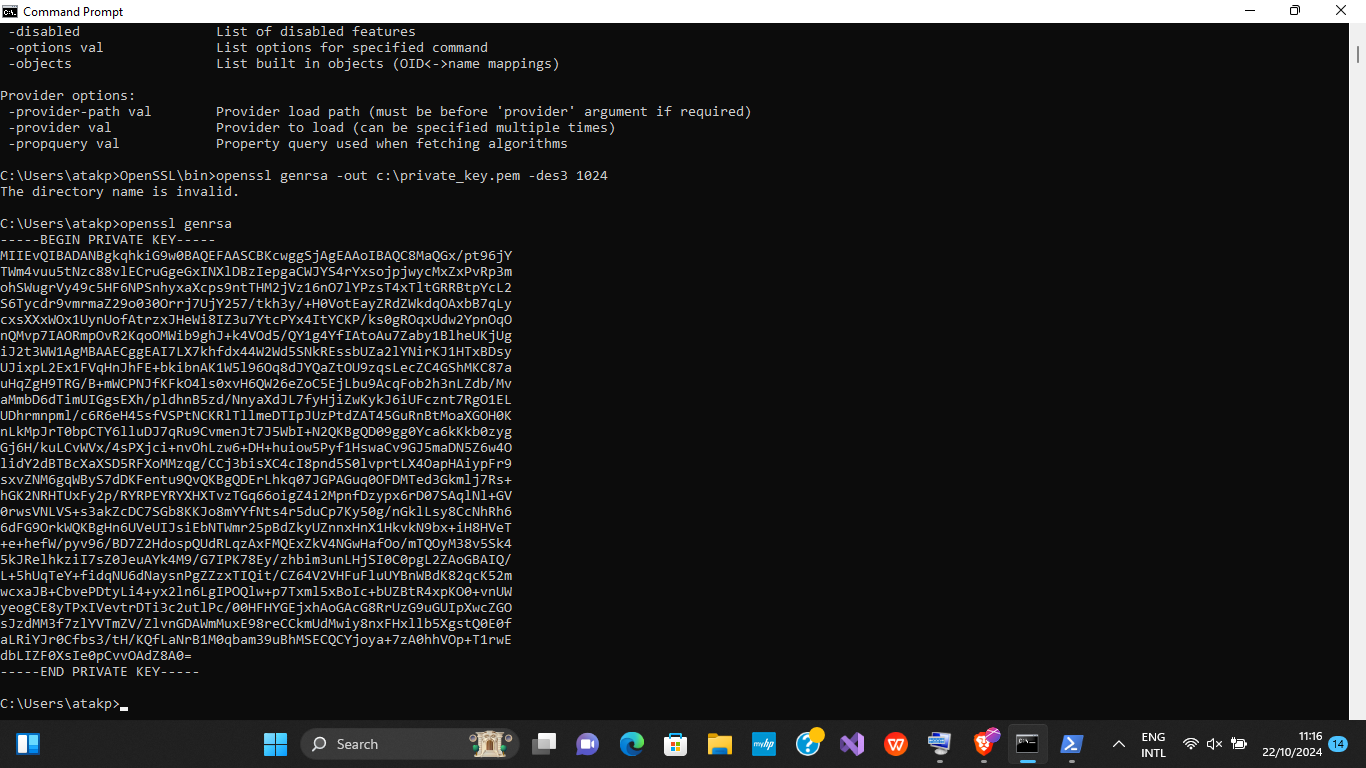
• req: Creates certificate signing requests (CSRs).

• x509: Handles certificate operations.

• enc: Performs encryption and decryption of data.

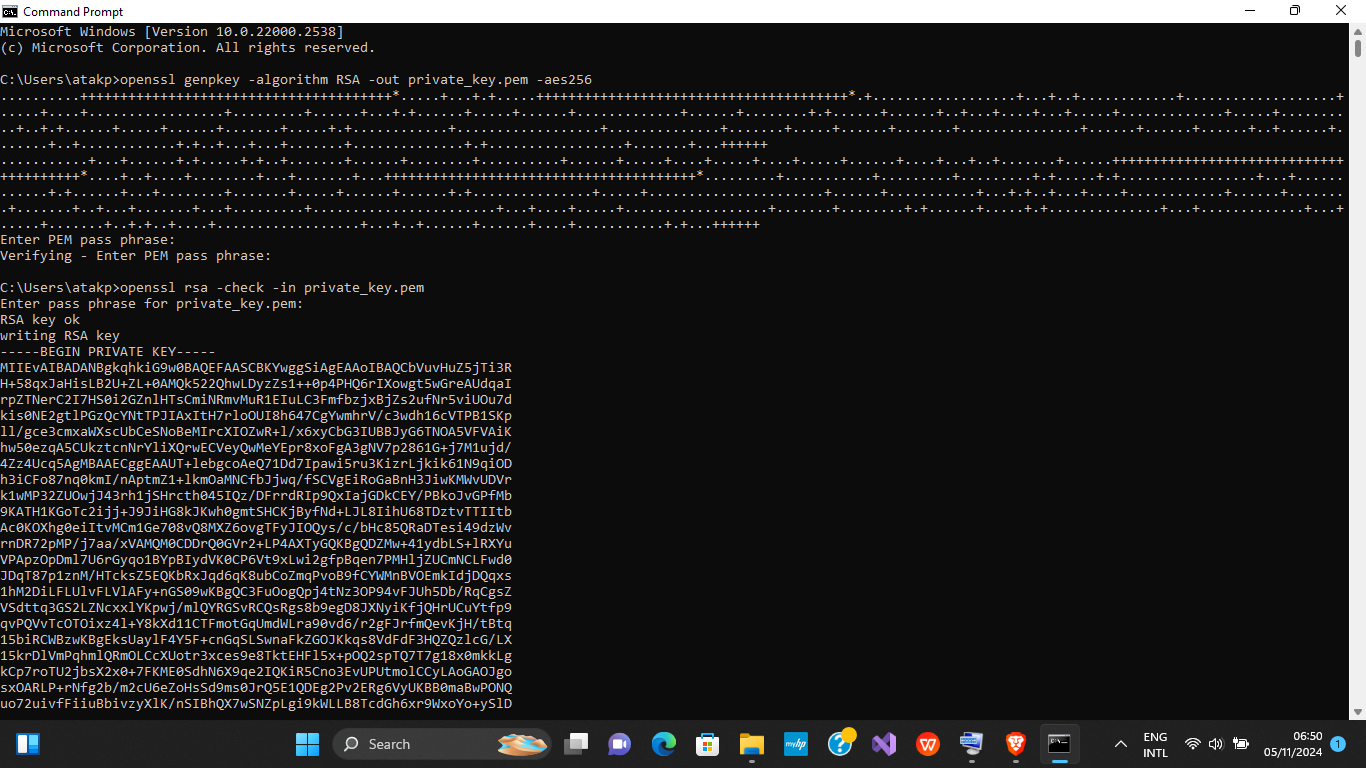
• dgst: Calculates message digests (hashes).

This list provides all the available functionalities I can use in OpenSSL, depending on the task I want to perform.



From the picture above, when I entered the command **openssl genrsa** in the command prompt, it generated a new RSA private key. The output started with the phrase “BEGIN PRIVATE KEY”, which indicates that the key being generated is in a specific format called PEM (Privacy-Enhanced Mail) format.

**PEM format** is a text-based format that represents the private key between the markers ”—–BEGIN PRIVATE KEY—–” and ”—–END PRIVATE KEY—–”. This format is used to store cryptographic data, such as private keys, certificates, and other security-related information. The private key is encoded in base64 and displayed within these markers.

This private key is used in asymmetric encryption, where it will be paired with a public key for encrypting and decrypting data or creating digital signatures. The BEGIN PRIVATE KEY and END PRIVATE KEY markers help identify and separate the private key from other data when it’s stored In files.

When I ran the command **openssl genpkey**, OpenSSL displayed a list of options with plus (+) and minus (-) signs. These are different flags or parameters that can be used to customize how the private key is generated.

• Plus signs (+) indicate options that can be added to the command to include extra features or properties, such as specifying the key algorithm or its length.

• Minus signs (-) show parameters that can be used to modify or configure the key generation process, such as encryption options or file output.

For example:

• -algorithm: Specifies the type of algorithm to use, like RSA, DSA, or EC (Elliptic Curve).

• -out: Determines where to save the generated key (such as a file name).

• -pkeyopt: Allows setting additional options, like key size or encryption method.

This allow for a more flexible way of generating private keys, letting you tailor the process according to your specific needs, such as choosing the algorithm or setting key strength.

\*\*\*

When I typed **openssl rsa** in the command prompt, it displayed the private key associated with an RSA key pair. This command is specifically used to manage RSA private keys, including viewing, modifying, or converting them into different formats.

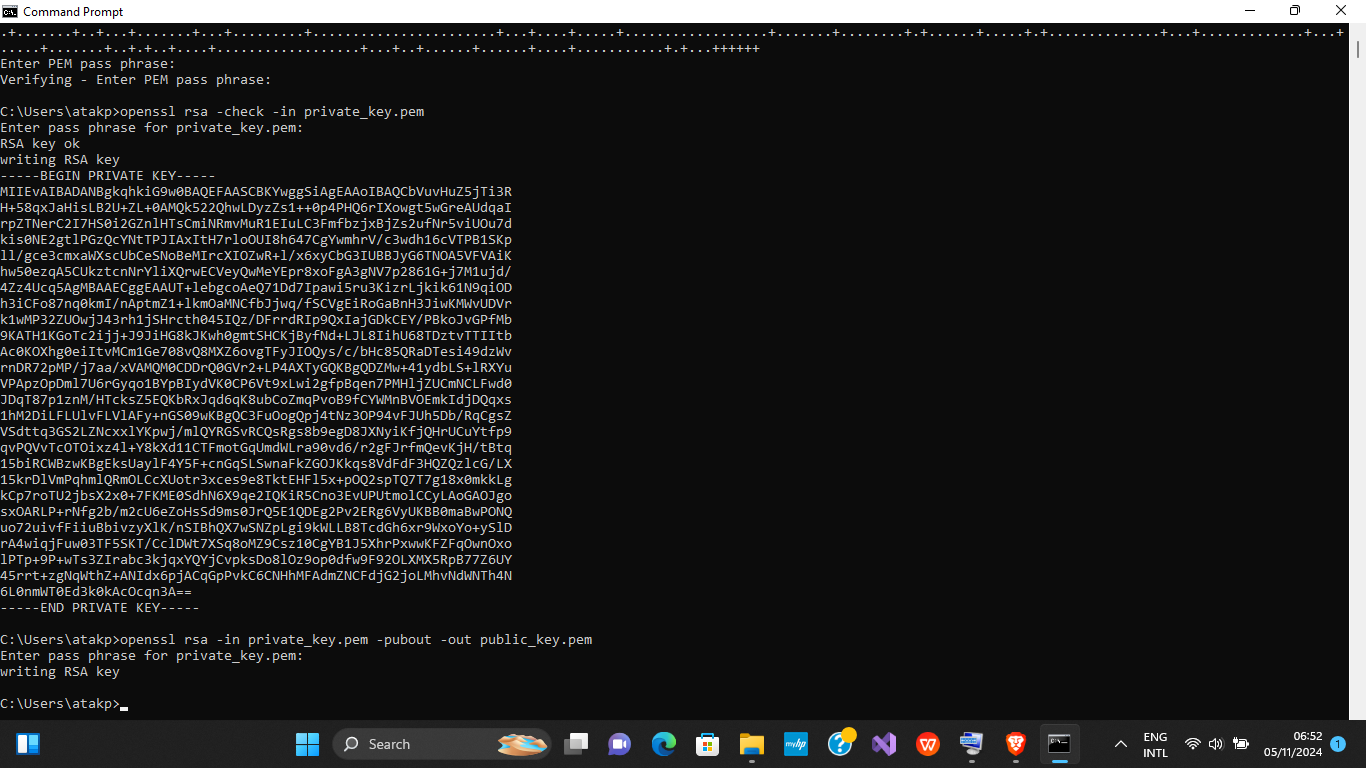
The openssl rsa command is used to process RSA private keys. When I ran this command, OpenSSL displayed the private key associated with the RSA key pair in PEM format, which is a base64-encoded text format. The key is surrounded by markers like ”—–BEGIN RSA PRIVATE KEY—–” and ”—–END RSA PRIVATE KEY—–”.

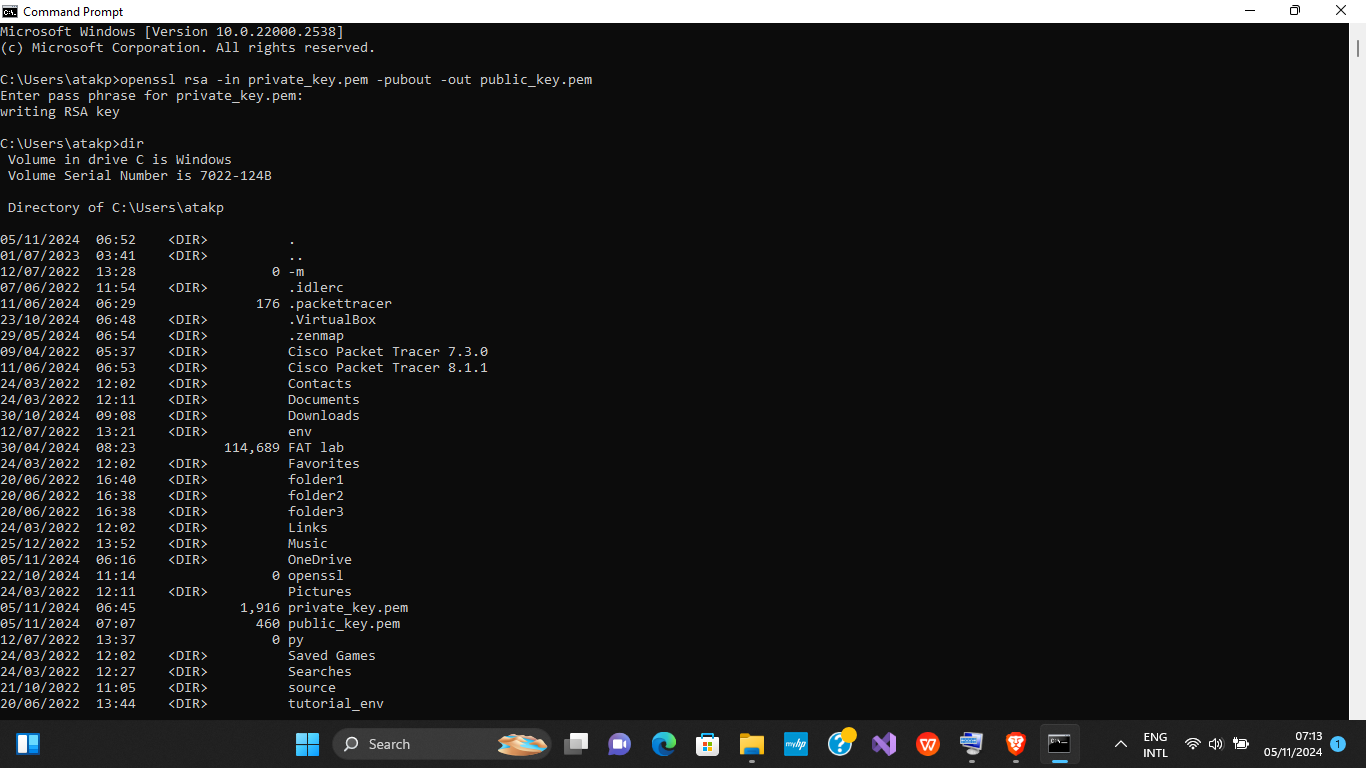
This command allows me to:

• View the private key.

• Convert the private key to a different format.

• Remove or add passphrases to the key.

The private key is crucial for decrypting data that was encrypted using the corresponding public key, as well as for signing messages or certificates in RSA-based encryption systems.



Looking above, when I typed the **dir** command, it displayed a list of the files and directories in the current working directory. This is a standard command in the Windows Command Prompt that shows all the files, folders, and subdirectories that are present in the location where you’re currently operating.

The dir command is used to list the contents of a directory in the command prompt. It provides details like:

• File names

• Subfolders

• File sizes

• Modification dates

This command helps me quickly check what files are in the current folder, which is useful when working with OpenSSL, as I can easily verify if a generated key or certificate file is saved in the correct location.

**Use Cases of OpenSSL**

• Web Servers: OpenSSL is used to configure SSL/TLS for web servers, ensuring encrypted communication between clients and servers.

• Secure Email: OpenSSL can generate keys for signing and encrypting emails.

• Secure File Storage: OpenSSL can be used to encrypt files for secure storage and sharing.

• VPNs: OpenSSL is often used in Virtual Private Networks (VPNs) for secure tunneling.

**Conclusion**

OpenSSL is a powerful and essential tool for anyone working with secure communications, encryption, and digital certificates. Its flexibility and wide range of cryptographic features make it a cornerstone of modern cybersecurity practices.

**Practice Questions**

**1. What is the essence of asymmetric data encryption?**

**2. What is the essence of symmetric data encryption?**

**3. What is the difference between symmetric encryption and asymmetric encryption?**

**4. How is asymmetric data encryption done?**

**5. How is the digital signature of a message generated?**

**6. What is the verification of the digital signature of a message?**

**7. What can the error indicate when verifying the digital signature of a message?**

**Answers**

1. Essence of Asymmetric Data Encryption

Asymmetric encryption uses a pair of cryptographic keys: a public key and a private key. The public key is used for encryption, and the private key is used for decryption. It is often used to securely exchange symmetric keys, authenticate users, or provide confidentiality in communications.

Key features:

• Public key: Can be shared openly.

• Private key: Must be kept secret.

• Provides secure communication even when public keys are exposed.

2. Essence of Symmetric Data Encryption

Symmetric encryption uses a single shared key for both encryption and decryption of data. It is faster and more efficient than asymmetric encryption but requires secure key sharing between parties before communication begins.

Key features:

• Both sender and receiver must share the same key.

• It is ideal for encrypting large amounts of data (e.g., files, databases).

3. Difference Between Symmetric and Asymmetric Encryption

Aspect Symmetric Encryption Asymmetric Encryption

Keys Single shared key Key pair: public and private keys

Speed Faster Slower

Security Requires secure key sharing No need to share private key

Use Cases Large data encryption Key exchange, digital signatures

Example Algorithms AES, DES, RC4 RSA, ECC

4. How Asymmetric Data Encryption Is Done

1. Key Pair Generation: A user generates a public-private key pair using an algorithm like RSA or ECC.

2. Encryption: The sender encrypts the message using the recipient’s public key.

3. Decryption: The recipient decrypts the message using their private key.

This ensures:

• Only the intended recipient (who has the private key) can decrypt the data.

• The public key alone cannot decrypt the message.

5. How the Digital Signature of a Message Is Generated

1. Hashing: The message is hashed using a cryptographic hash function (e.g., SHA-256) to create a fixed-size digest.

2. Signing: The hash is encrypted using the sender’s private key, creating the digital signature.

3. Appending: The signature is sent along with the message.

This process provides:

• Integrity (detects tampering).

• Authentication (proves the sender’s identity).

6. What Is the Verification of the Digital Signature of a Message?

Verification ensures the digital signature is valid:

1. The recipient:

• Hashes the received message.

• Decrypts the sender’s signature using the sender’s public key to get the original hash.

2. Compares both hashes:

• If they match, the signature is valid (message is authentic and untampered).

• If they don’t match, the signature is invalid.

7. What Can the Error Indicate When Verifying the Digital Signature of a Message?

Errors during signature verification can indicate:

• Message tampering: The message was altered in transit.

• Incorrect public key: The wrong public key was used for verification.

• Corrupted signature: The digital signature was corrupted or improperly generated.

• Key mismatch: The signature was not created with the corresponding private key.

**THANK YOU**