**Symmetric Encryption (Theory + Demo)**

**Theory:**

Symmetric encryption is a method of cryptography where the **same key** is used for both **encrypting** and **decrypting** data. This means that both the sender and the receiver must have access to the shared secret key. It is generally fast and efficient but requires secure key distribution.

**Common Algorithms:**

* **AES (Advanced Encryption Standard)**: Widely used for securing data, it supports key sizes of 128, 192, and 256 bits.
* **DES (Data Encryption Standard)**: An older standard that uses a 56-bit key; now considered insecure due to its short key length.
* **3DES (Triple DES)**: An enhancement of DES that applies the algorithm three times to each data block, increasing security.
* **Blowfish**: A fast block cipher that uses variable-length keys (up to 448 bits) and is suitable for applications requiring high speed.
* **RC4**: A stream cipher known for its simplicity and speed, though it has vulnerabilities and is less commonly used today.
* **Twofish**: A successor to Blowfish, it supports key sizes up to 256 bits and is known for its security and performance.

**Demo**

from cryptography.fernet import Fernet

def generate\_key():

    key = Fernet.generate\_key()

    with open("key.key", "wb") as key\_file:

        key\_file.write(key)

    print("Key is generated and saved to key.key")

# Function to load the key from the file

def load\_key():

    return open("key.key", "rb").read()

# Function to encrypt a message using the loaded key

def encrypt\_message(message):

    key = load\_key()

    f = Fernet(key)

    encrypted\_message = f.encrypt(message.encode())

    return encrypted\_message

# Function to Decrypt a message using the loaded key

def decrypt\_message(encrypted\_message):

    key = load\_key()

    f = Fernet(key)

    return f.decrypt(encrypted\_message).decode()

# Generate a key (run once)

generate\_key()

# Encrypt a message

encrypted\_message = encrypt\_message("Hello, World!")

print(f"Encrypted Message: {encrypted\_message}")

# Decrypt a message

decrypted = decrypt\_message(encrypted\_message)

print(f"Decrypted: {decrypted}")

Key is generated and saved to key.key

Encrypted Message: b'gAAAAABonhl4wqo-A93ZpDh60BJaiFeJbF4TfdGPsWERouvf9yTN36VzPZEJyjbqYkaAxo1QDydmwqf5GXNuaJk6ZoB\_rCoLzQ=='

Decrypted: Hello, World!