**NATIONAL INSTITUTE OF TECHNOLOGY PUDUCHERRY**



(An Institution of National Importance under MoE, Govt. of India)

**KARAIKAL – 609 609**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**Roll Number:** CS22B1053 **Name:** SHRIHARI THAIKKOOTTATHIL

**Semester:** 6th semester **Class:** CSE

**Subject Code:** CS1702  **Subject Name:** NETWORK SECURITY

**Assignment - 1 Date:** 29-04-2025

**ADVANCED ENCRYPTION STANDARD**

**INTRODUCTION**  
AES (Advanced Encryption Standard) is a widely used symmetric key encryption algorithm that replaced the older DES (Data Encryption Standard). AES is a block cipher that operates on fixed-size blocks of data, specifically 128-bit blocks. It uses a symmetric key for both encryption and decryption, and supports key lengths of 128, 192, and 256 bits. In this implementation, AES-128 is used, which means the encryption key is 128 bits long. AES is known for its security and efficiency, and it is commonly used in various applications, including secure data transmission, file encryption, and password storage.

**AES Encryption Process**The AES encryption process begins by taking the 128-bit plaintext and expanding the encryption key into several round keys. The data undergoes a series of transformations through 10 rounds (for AES-128), where each round consists of four main steps: **SubBytes** (substituting bytes using an S-Box), **ShiftRows** (shifting rows of the state), **MixColumns** (mixing the columns for diffusion), and **AddRoundKey** (XORing the data with the round key). After completing the rounds, the result is the ciphertext, which is a securely encrypted version of the plaintext.

**AES Decryption Process**The AES decryption process closely mirrors encryption but with the round keys used in reverse order and inverse operations applied. The ciphertext is processed in reverse through the rounds, starting with an AddRoundKey operation using the last round key. Then, the decryption applies Inverse SubBytes, Inverse ShiftRows, and Inverse MixColumns (except in the final round), before adding the round keys in reverse order. After completing the rounds, the original plaintext is recovered.

**Implementation**For implementing AES encryption and decryption, **PyCryptodome** library Is used. The program allows for text encryption using a password-based key and decrypts the encrypted data to retrieve the original plaintext.

**AES ENCRYPTION FUNCTION**

def aes\_encrypt(plaintext: str, key: bytes) -> bytes:

cipher = AES.new(key, AES.MODE\_ECB)

padded\_text = pad(plaintext.encode(), AES.block\_size)

return cipher.encrypt(padded\_text)

The encryption function takes two inputs: the plaintext to be encrypted and a 128-bit key derived from a password. It first creates an AES cipher object in ECB mode using the provided key. The plaintext is then converted to bytes and padded to ensure its length is a multiple of 16 bytes, which is required by AES. Finally, the padded plaintext is encrypted using the cipher’s encrypt method, producing the ciphertext.

**AES DECRYPTION FUNCTION**

def aes\_decrypt(ciphertext: bytes, key: bytes) -> str:

cipher = AES.new(key, AES.MODE\_ECB)

decrypted\_data = cipher.decrypt(ciphertext)

return unpad(decrypted\_data, AES.block\_size).decode()

The decryption function takes the ciphertext (encrypted data) and the same key used during encryption as input. It creates a new AES cipher object in ECB mode using the provided key. The ciphertext is then decrypted using the cipher’s decrypt method. Since the original plaintext was padded during encryption, the decrypted data is unpadded using Crypto.Util.Padding.unpad to restore it to its original form. Finally, the unpadded byte data is decoded back into a string and returned as the plaintext.

**USER INPUT AND PROGRAM FLOW**

if \_\_name\_\_ == "\_\_main\_\_":

password = input("Enter password: ").strip()

key = hashlib.sha256(password.encode()).digest()[:16]

plaintext = input("Enter text to encrypt: ").strip()

ciphertext = aes\_encrypt(plaintext, key)

print(f"Encrypted (hex): {ciphertext.hex()}")

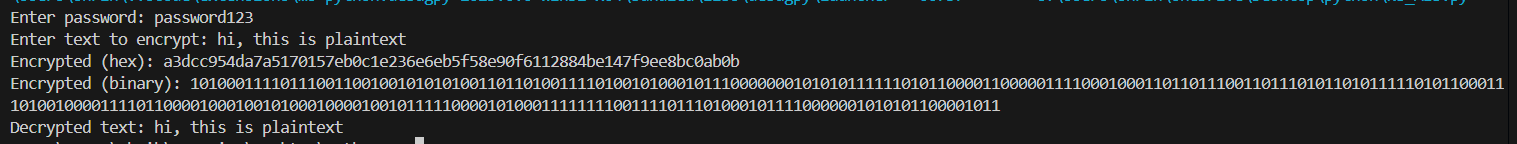
print(f"Encrypted (binary): {to\_binary(ciphertext)}")

decrypted\_text = aes\_decrypt(ciphertext, key)

print(f"Decrypted text: {decrypted\_text}"

The code prompts the user for a password, which is hashed with SHA-256 and truncated to 16 bytes to create an AES-128 key. It then takes a plaintext input from the user and encrypts it using the aes\_encrypt function. The resulting ciphertext is displayed in both hexadecimal and binary formats. To confirm the encryption process, the ciphertext is decrypted using the same key with the aes\_decrypt function, and the original plaintext is recovered and displayed.

**OUTPUT**

****

**CONCLUSION**This code demonstrates the use of AES encryption and decryption in Python with the PyCryptodome library. It effectively handles text input, encrypts it using a key derived from a password, and displays the ciphertext in both hexadecimal and binary formats. The implementation provides a clear and functional example of symmetric key encryption using AES.