

Experiment 6

6.1 Aim:-Encrypt long messages using various modes of operation using AES or DES

6.2 Course Outcome:-

- Understand and implement symmetric key cryptography algorithms like AES and DES.
- Analyze various modes of operation (ECB, CBC, CFB, OFB, and CTR).
- Evaluate the security and effectiveness of encryption in different scenarios.

6.3 Lab Objective:-

- To demonstrate the encryption and decryption of long messages using AES or DES.
- To explore the characteristics and applications of various modes of operation.
- To understand the impact of mode selection on data security and performance.

6.4 Requirement:-

- Python programming language (or equivalent).
- Libraries such as `pycryptodome` for AES/DES implementation.
- A plaintext message for encryption and decryption.
- Keys and initialization vectors (IV) for the chosen cryptographic algorithm.

6.5 Theory:

AES and DES AES (Advanced Encryption Standard) and DES (Data Encryption Standard) are symmetric key cryptography algorithms. AES uses block sizes of 128 bits and supports key sizes of 128, 192, and 256 bits. DES operates on 64-bit blocks with a 56-bit key.

Modes of Operation

Modes of operation are techniques to apply block ciphers like AES/DES to encrypt data larger than a single block.

1. ECB (Electronic Codebook): Each block is encrypted independently.
 - Pros: Simple and fast.
 - Cons: Vulnerable to pattern leakage.
2. CBC (Cipher Block Chaining): Each block is XORed with the previous ciphertext block.
 - Pros: Provides better security than ECB.
 - Cons: Requires an initialization vector (IV).
3. CFB (Cipher Feedback): Converts the block cipher into a stream cipher.
 - Pros: Suitable for encrypting data streams.
 - Cons: Error propagation.
4. OFB (Output Feedback): Similar to CFB but without error propagation.
 - Pros: Avoids error propagation.
5. CTR (Counter Mode): Converts the block cipher into a stream cipher using a counter.
 - Pros: High performance, parallelizable.

6.6 Procedure:-

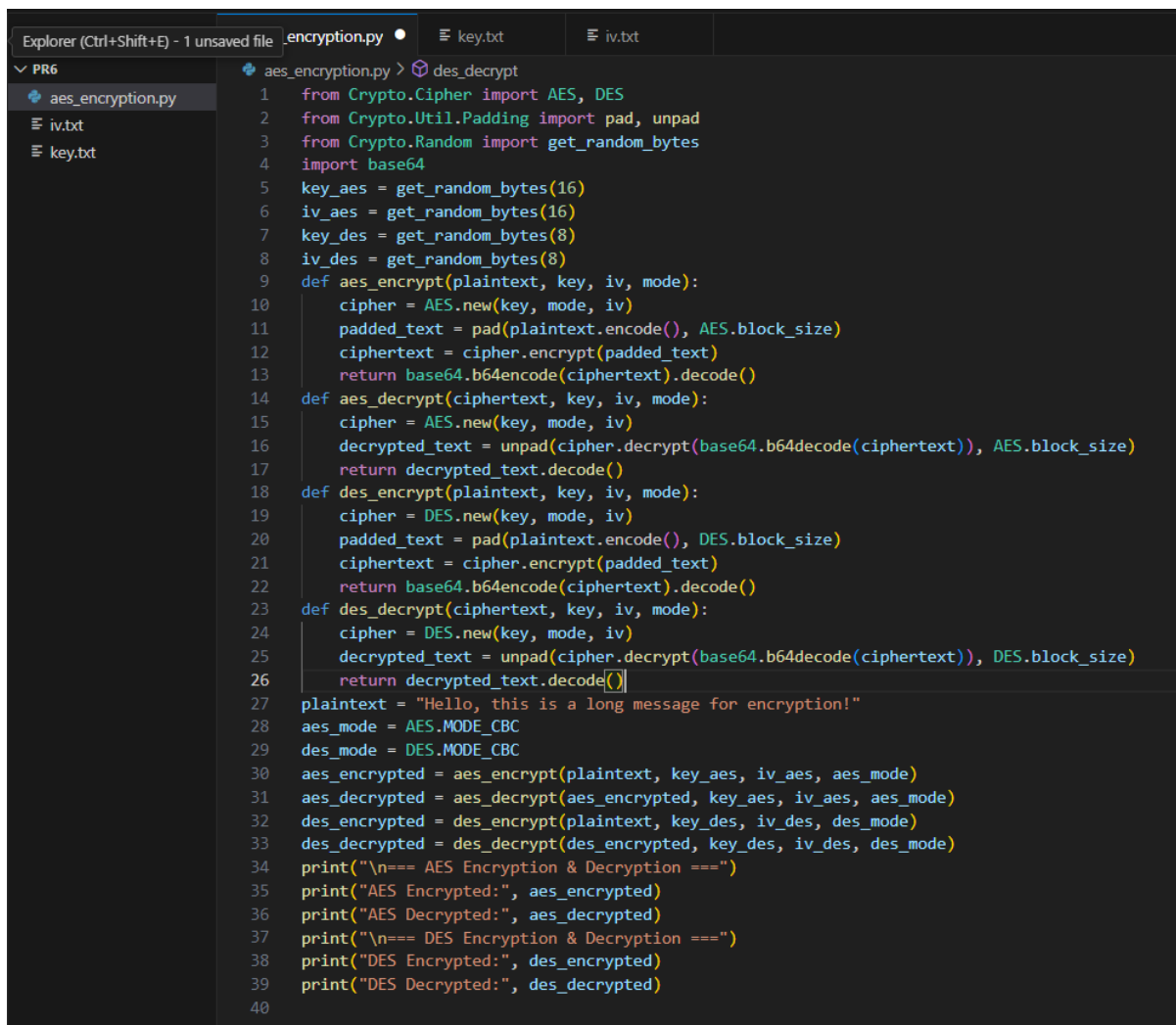
Encryption Process

1. Initialize Key and IV: Generate or provide a key and initialization vector.
2. Select Algorithm: Choose AES or DES for encryption.
3. Choose Mode of Operation: Select one of the modes (ECB, CBC, CFB, OFB, CTR).
4. Encrypt Message: Divide the plaintext into blocks and apply the chosen encryption method.

Decryption Process

1. Initialize Key and IV: Use the same key and IV as in encryption.
2. Decrypt Message: Apply the reverse of the encryption process to recover the plaintext.
3. Verify Integrity: Check the decrypted message against the original plaintext.

6.7 Result :-



```
Explorer (Ctrl+Shift+E) - 1 unsaved file .encryption.py • key.txt iv.txt
PR6
aes_encryption.py
iv.txt
key.txt
aes_encryption.py > des_decrypt
1 from Crypto.Cipher import AES, DES
2 from Crypto.Util.Padding import pad, unpad
3 from Crypto.Random import get_random_bytes
4 import base64
5 key_aes = get_random_bytes(16)
6 iv_aes = get_random_bytes(16)
7 key_des = get_random_bytes(8)
8 iv_des = get_random_bytes(8)
9 def aes_encrypt(plaintext, key, iv, mode):
10     cipher = AES.new(key, mode, iv)
11     padded_text = pad(plaintext.encode(), AES.block_size)
12     ciphertext = cipher.encrypt(padded_text)
13     return base64.b64encode(ciphertext).decode()
14 def aes_decrypt(ciphertext, key, iv, mode):
15     cipher = AES.new(key, mode, iv)
16     decrypted_text = unpad(cipher.decrypt(base64.b64decode(ciphertext)), AES.block_size)
17     return decrypted_text.decode()
18 def des_encrypt(plaintext, key, iv, mode):
19     cipher = DES.new(key, mode, iv)
20     padded_text = pad(plaintext.encode(), DES.block_size)
21     ciphertext = cipher.encrypt(padded_text)
22     return base64.b64encode(ciphertext).decode()
23 def des_decrypt(ciphertext, key, iv, mode):
24     cipher = DES.new(key, mode, iv)
25     decrypted_text = unpad(cipher.decrypt(base64.b64decode(ciphertext)), DES.block_size)
26     return decrypted_text.decode()
27 plaintext = "Hello, this is a long message for encryption!"
28 aes_mode = AES.MODE_CBC
29 des_mode = DES.MODE_CBC
30 aes_encrypted = aes_encrypt(plaintext, key_aes, iv_aes, aes_mode)
31 aes_decrypted = aes_decrypt(aes_encrypted, key_aes, iv_aes, aes_mode)
32 des_encrypted = des_encrypt(plaintext, key_des, iv_des, des_mode)
33 des_decrypted = des_decrypt(des_encrypted, key_des, iv_des, des_mode)
34 print("\n=== AES Encryption & Decryption ===")
35 print("AES Encrypted:", aes_encrypted)
36 print("AES Decrypted:", aes_decrypted)
37 print("\n=== DES Encryption & Decryption ===")
38 print("DES Encrypted:", des_encrypted)
39 print("DES Decrypted:", des_decrypted)
40
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

● PS C:\Users\darsh\Desktop\6th sem Practical\CNS\pr6> python aes_encryption.py

=== AES Encryption & Decryption ===
AES Encrypted: KIK8AGEiFK/Ks0aF6IqjbUBjU7Y0JDd/FN4L3PTqw5pt12en3GnSkWIzR9+3lc/t
AES Decrypted: Hello, this is a long message for encryption!

=== DES Encryption & Decryption ===
DES Encrypted: P9RHIUMCIGbg169CYH69K2dqEqjHn+foGEKa3t0N0z0H2BvQvtxQ6kjE/yKTYZGj
DES Decrypted: Hello, this is a long message for encryption!
❖ PS C:\Users\darsh\Desktop\6th sem Practical\CNS\pr6> █
```

6.8 Conclusion:-

In this practical, we successfully implemented AES and DES encryption & decryption techniques. Both encryption algorithms correctly transformed plaintext into ciphertext and then successfully retrieved the original message after decryption. This demonstrates a strong understanding of cryptographic principles and secure data handling.