CNS Assignment
Mayur Jaiswal
D15B
26
AES Algorithm

Introduction:

The Advanced Encryption Standard (AES) is a symmetric encryption algorithm widely used for securing data. It encrypts data in fixed-size blocks (128 bits) and supports key sizes of 128, 192, and 256 bits. AES is known for its speed and security and is the standard encryption method for many applications, including data protection and secure communications.

Key Concepts:

- **Symmetric Encryption**: The same key is used for both encryption and decryption.
- **Block Cipher**: Data is processed in fixed-size blocks.
- **Modes of Operation**: AES can operate in various modes, such as CBC (Cipher Block Chaining) or GCM (Galois/Counter Mode), which define how multiple blocks are encrypted.

Steps to Perform AES Encryption and Decryption

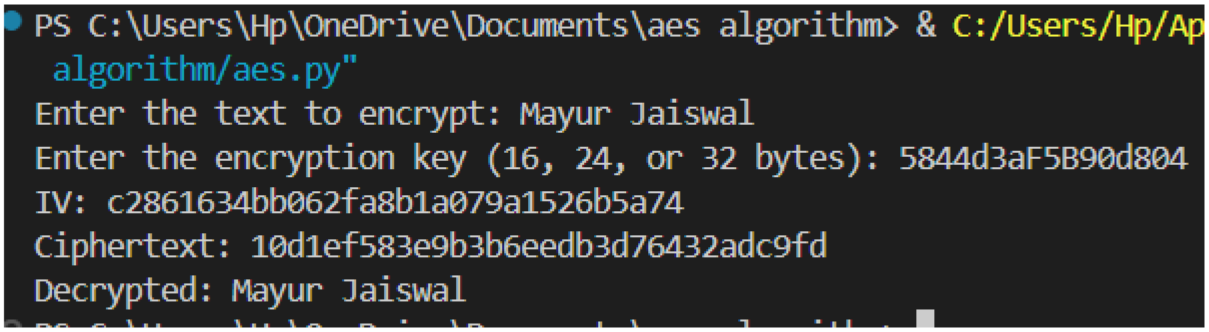
1. **Install Required Libraries**:
    - Before implementing AES, install necessary libraries like `pycryptodome`, which provides cryptographic services in Python.
2. **Generate a Key**:
    - Generate a secure key for AES. Ensure that the key length is appropriate (16 bytes for AES-128, 24 bytes for AES-192, or 32 bytes for AES-256).
3. **Create an Initialization Vector (IV)**:
    - For modes like CBC, generate a random IV. The IV should be unique for each encryption session but does not need to be kept secret.
4. **Define the Data to Encrypt**:
    - Prepare the plaintext data that you want to encrypt. Ensure it is properly formatted (e.g., padded if necessary).
5. **Encrypt the Data**:
    - Use the AES algorithm to encrypt the plaintext. This typically involves:
        - Creating an AES cipher object with the specified key and mode.
        - Encrypting the data and storing the resulting ciphertext.
6. **Decrypt the Data**:
    - To retrieve the original plaintext, use the same AES key and IV to decrypt the ciphertext.
7. **Handle Exceptions**:
    - Implement error handling to manage issues such as incorrect padding or invalid keys.

8. **Output Results**:
    - Print or store the encrypted data (ciphertext) and the decrypted data (original plaintext) for verification.

Code:

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad

def aes_encrypt(key, plaintext):
    key_bytes = key.encode()
    cipher = AES.new(key_bytes, AES.MODE_CBC)
    ct_bytes = cipher.encrypt(pad(plaintext.encode(), AES.block_size))
    return cipher.iv, ct_bytes

def aes_decrypt(key, iv, ciphertext):
    key_bytes = key.encode()
    cipher = AES.new(key_bytes, AES.MODE_CBC, iv)
    pt = unpad(cipher.decrypt(ciphertext), AES.block_size)
    return pt.decode()

if __name__ == "__main__":
    plaintext = input("Enter the text to encrypt: ")
    key = input("Enter the encryption key (16, 24, or 32 bytes): ")

    # Ensure the key length is valid
    if len(key) not in [16, 24, 32]:
        print("Invalid key length. Key must be 16, 24, or 32 bytes.")
        exit()

    # Encrypt
    iv, ciphertext = aes_encrypt(key, plaintext)
    print(f"IV: {iv.hex()}")
    print(f"Ciphertext: {ciphertext.hex()}")

    # Decrypt
    decrypted_text = aes_decrypt(key, iv, ciphertext)
    print(f"Decrypted: {decrypted_text}")
```

Screenshots:



```
PS C:\Users\Hp\OneDrive\Documents\aes algorithm> & C:/Users/Hp/Ap
 algorithm/aes.py"
Enter the text to encrypt: Mayur Jaiswal
Enter the encryption key (16, 24, or 32 bytes): 5844d3aF5B90d804
IV: c2861634bb062fa8b1a079a1526b5a74
Ciphertext: 10d1ef583e9b3b6eedb3d76432adc9fd
Decrypted: Mayur Jaiswal
```

Conclusion

AES is a robust encryption standard widely used to secure sensitive data. Its symmetric nature, combined with various modes of operation, makes it versatile for different applications. Proper implementation is critical, especially regarding key management and IV handling.