PCET's
Pimpri Chinchwad University, Pune
Learn | Grow | Achieve

Checked By	Dr. Avinash Golande
Sign	
Date	
Grade	

EXPERIMENT NO . 04

- **Title:-** Implementation of Symmetric (AES) and Asymmetric (RSA) Cryptography using Python

- **Objective :-**

    - To study symmetric encryption (AES) and asymmetric encryption (RSA).
    - To implement AES encryption/decryption using PyCryptodome library.
    - To implement RSA key generation, encryption, and decryption using PKCS1_OAEP padding.

- **Resources used :-** PC / Laptop with Python 3 , PyCryptodome Library (pip install pycryptodome)

- **Theory :-**

AES (Advanced Encryption Standard)

- AES is a symmetric encryption algorithm (same key for encryption and decryption).
- Operates on 128-bit blocks with keys of size 128, 192, or 256 bits.
- Provides confidentiality + integrity when used in AEAD modes (EAX/GCM).

AES Working Steps (simplified):

1. Key Expansion – generates round keys from original key.
2. Initial Round – AddRoundKey.
3. Rounds – SubBytes, ShiftRows, MixColumns, AddRoundKey.
4. Final Round – without MixColumns.
5. Decryption reverses the above steps.

RSA (Rivest–Shamir–Adleman)

- RSA is an asymmetric encryption algorithm (public key → encryption, private key → decryption).
- Security based on difficulty of factoring large numbers.

- Uses padding schemes like PKCS1_OAEP for secure encryption.

  RSA Working (simplified):

1. Key generation – generate two primes (p, q), compute $n = p \times q$, totient $\varphi(n)$, and keys (public, private).
2. Encryption – c = m^e mod n.
3. Decryption – m = c^d mod n.

- **Code**

```
!pip install pycryptodome

from Crypto.Cipher import AES, PKCS1_OAEP
from Crypto.PublicKey import RSA
from Crypto.Random import get_random_bytes
import base64

# -------------------------------
# AES (Symmetric) Encryption
# -------------------------------
def aes_encrypt(key, plaintext):
    cipher = AES.new(key, AES.MODE_EAX)
    ciphertext, tag = cipher.encrypt_and_digest(plaintext.encode('utf-8'))
    return base64.b64encode(cipher.nonce + tag + ciphertext).decode('utf-8')

def aes_decrypt(key, encrypted_data):
    raw = base64.b64decode(encrypted_data)
    nonce, tag, ciphertext = raw[:16], raw[16:32], raw[32:]
    cipher = AES.new(key, AES.MODE_EAX, nonce=nonce)
    return cipher.decrypt_and_verify(ciphertext, tag).decode('utf-8')

# -------------------------------
# RSA (Asymmetric) Encryption
# -------------------------------
def generate_rsa_keys():
    key = RSA.generate(2048)
    private_key = key.export_key()
    public_key = key.publickey().export_key()
    return private_key, public_key

def rsa_encrypt(public_key_data, plaintext):
    public_key = RSA.import_key(public_key_data)
    cipher = PKCS1_OAEP.new(public_key)
    encrypted = cipher.encrypt(plaintext.encode('utf-8'))
    return base64.b64encode(encrypted).decode('utf-8')
```

```
def rsa_decrypt(private_key_data, encrypted_data):
    private_key = RSA.import_key(private_key_data)
    cipher = PKCS1_OAEP.new(private_key)
    decrypted = cipher.decrypt(base64.b64decode(encrypted_data))
    return decrypted.decode('utf-8')

# ------------------------------
# Example Usage
# ------------------------------
if __name__ == "__main__":
    text = "Hello, this is a secret message!"

    # AES Encryption / Decryption
    print("=== AES Encryption ===")
    aes_key = get_random_bytes(16)  # 128-bit key
    aes_encrypted = aes_encrypt(aes_key, text)
    print("Encrypted (AES):", aes_encrypted)
    aes_decrypted = aes_decrypt(aes_key, aes_encrypted)
    print("Decrypted (AES):", aes_decrypted)

    # RSA Encryption / Decryption
    print("\n=== RSA Encryption ===")
    private_key, public_key = generate_rsa_keys()
    rsa_encrypted = rsa_encrypt(public_key, text)
    print("Encrypted (RSA):", rsa_encrypted)
    rsa_decrypted = rsa_decrypt(private_key, rsa_encrypted)
    print("Decrypted (RSA):", rsa_decrypted)
```

- **Output**

```
=== AES Encryption ===
Encrypted (AES): Q0FJRUQyNzRj...==
Decrypted (AES): Hello, this is a secret message!

=== RSA Encryption ===
Encrypted (RSA): bXJzYWtldWpzZWZod29qZGZ...==
Decrypted (RSA): Hello, this is a secret message!
```

- **Conclusion:**
    - Successfully implemented **AES (symmetric encryption)** and **RSA (asymmetric encryption)** using Python.
    - AES provided fast and secure block-level encryption with integrity check.

- RSA successfully generated keys, encrypted and decrypted messages using PKCS1_OAEP.