Lab 4: Symmetric & Asymmetric Cryptography

The purpose of this lab is to implement **both symmetric and asymmetric cryptographic techniques**, including:

- AES (Advanced Encryption Standard) – Symmetric key algorithm
- RSA (Rivest–Shamir–Adleman) – Asymmetric key algorithm
- Digital Signature and Verification using RSA
- File hashing using SHA-256
- Performance benchmarking by varying key sizes (N) and plotting timing graphs

Execute the Program

1. Install Packages

```
pip install pycryptodome matplotlib
```

2. Generate Keys

```
python3 crypto.py genkeys
```

By this command

- AES-128 key file → `aes_key_128.bin`
- AES-256 key file → `aes_key_256.bin`
- RSA private/public key pair → `rsa_private.pem`, `rsa_public.pem`

3. AES Encryption & Decryption

Encryption:

```
python crypto.pyy aes encrypt --mode ECB --size 128 -i plain.txt
-o cipher.bin
```

Decryption:

```
python crypto.py aes decrypt --mode ECB --size 128 -i cipher.bin
-o decrypted.txt
```

You can also use **CFB** mode instead of ECB:

```
python crypto.py aes encrypt --mode CFB --size 256 -i plain.txt
-o cipher.bin
```

---

4. RSA Encryption & Decryption

Encryption:

```
python crypto.py rsa encrypt -i plain.txt -o rsa_cipher.bin
```

Decryption:

```
python crypto.py rsa decrypt -i rsa_cipher.bin -o
rsa_decrypted.txt
```

---

5. Digital Signature & Verification

Sign a file:

```
python crypto.py sign -i plain.txt -o signature.bin
```

Verify the signature:

```
python crypto.py.py verify -i plain.txt -s signature.bin
```

6. Hashing (SHA-256)

```
python crypto.py.py hash -i plain.txt
```

Will display the SHA-256 hash value of the file.

---

7. Benchmarking

Command:

```
python crypto.py.py bench
```

This will:

- Vary AES and RSA key sizes (`N`)
- Measure encryption/decryption times

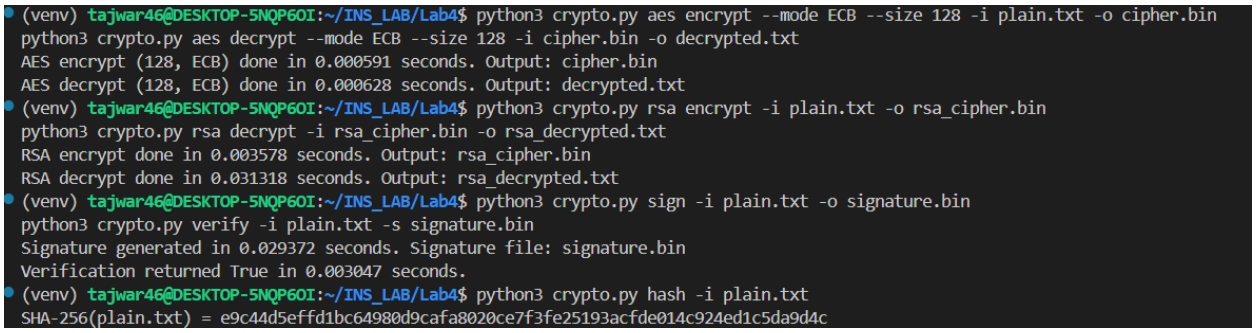
```
(venv) tajwar46@DESKTOP-5NQP6OI:~/INS_LAB/Lab4$ python3 crypto.py bench
Starting AES benchmark (deriving AES key from N bits of entropy)
AES N=16 bits: avg time 0.008398s over 3 runs
AES N=64 bits: avg time 0.006982s over 3 runs
AES N=128 bits: avg time 0.006259s over 3 runs
AES N=256 bits: avg time 0.042949s over 3 runs
Starting RSA benchmark (key generation + encrypt/decrypt of small message)
```

Functionalities

Functionality	Description
AES Encryption/Decryption	Supports 128-bit and 256-bit keys in ECB and CFB modes
RSA Encryption/Decryption	Uses PKCS1_OAEP for secure asymmetric encryption

RSA Signature & Verification	Implements SHA-256 hashing with PKCS#1 v1.5 signature
SHA-256 Hashing	Computes and prints hash of any file
Key Generation	AES keys (random bytes) and RSA key pair (2048 bits)
Benchmarking	Measures execution time for varying N and plots graphs

Output



References (Code Snippet Sources)

Source	Description
PyCryptodome Documentation	Reference for AES, RSA, OAEP, and pkcs1_15 usage
Stack Overflow – AES Padding Example	Helped in implementing AES padding/unpadding logic
Stack Overflow – RSA OAEP Chunking Example	Used for splitting data into chunks for RSA encryption
Python hashlib Docs	Used for SHA-256 file hashing implementation