# Lab Manual-4 Report: Programming Symmetric & Asymmetric Crypto

**Introduction:** This lab report presents the implementation of a cryptographic program that demonstrates symmetric and asymmetric encryption techniques. The program implements AES encryption/decryption, RSA encryption/decryption, RSA digital signatures, SHA-256 hashing, and performance analysis of these cryptographic operations.

## Programming Language and Libraries Used

**Programming Language:** Python 3 in google colab

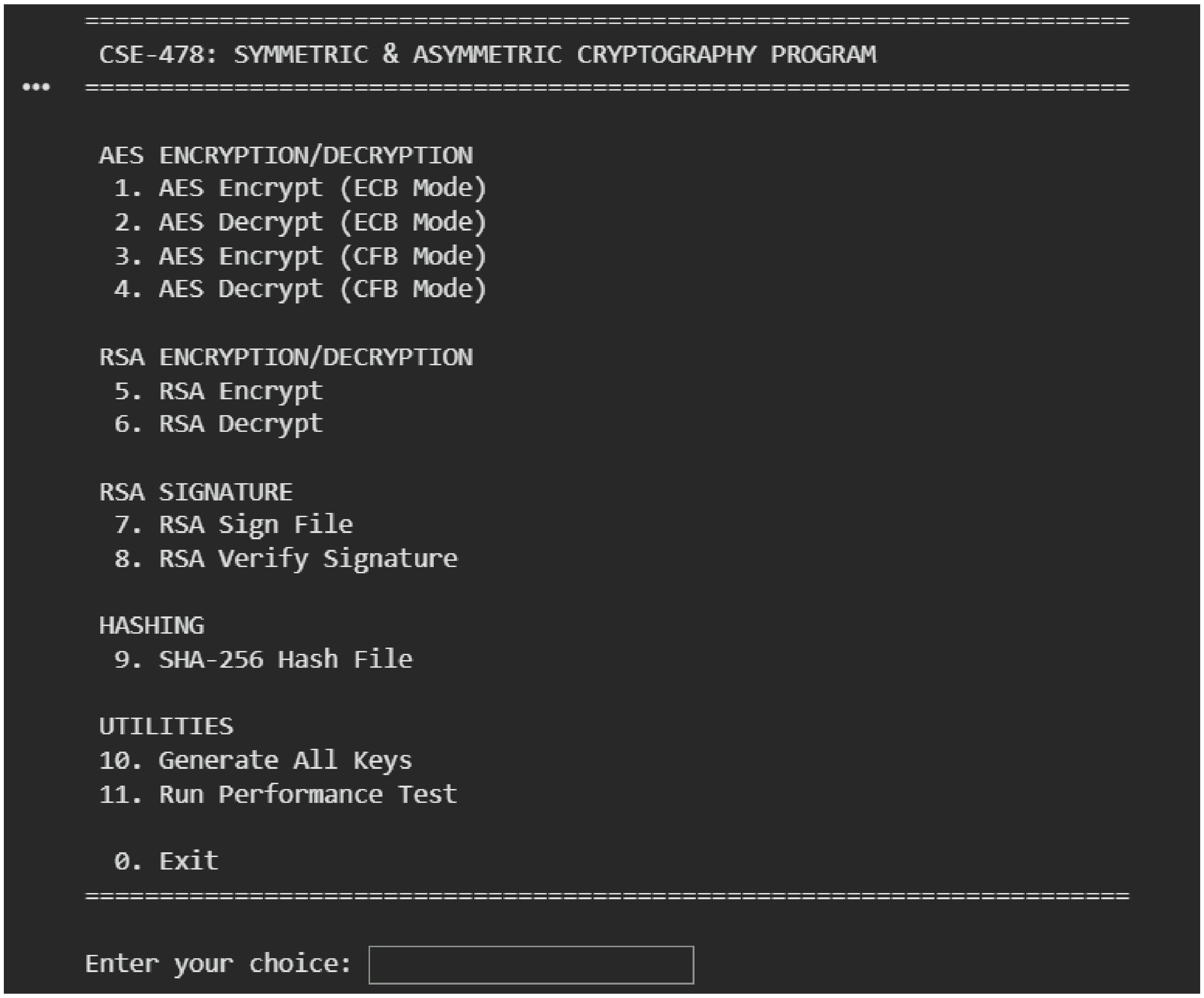
**Libraries Used:**

**PyCryptodome:** For cryptographic operations (AES, RSA, SHA-256)
**Matplotlib:** For plotting performance graphs
**OS:** For file and directory management
**Time:** For measuring execution time

**Installation:** Before run the main code, run the following code in  a cell in google colab:

```
!pip install pycryptodome matplotlib
```

Then run the main program file uploaded in the github (Lab_Manual_4.ipynb). After that you will see an output like the screenshot given in the **next page:**



```
==============================================================================
 CSE-478: SYMMETRIC & ASYMMETRIC CRYPTOGRAPHY PROGRAM
==============================================================================

 AES ENCRYPTION/DECRYPTION
  1. AES Encrypt (ECB Mode)
  2. AES Decrypt (ECB Mode)
  3. AES Encrypt (CFB Mode)
  4. AES Decrypt (CFB Mode)

 RSA ENCRYPTION/DECRYPTION
  5. RSA Encrypt
  6. RSA Decrypt

 RSA SIGNATURE
  7. RSA Sign File
  8. RSA Verify Signature

 HASHING
  9. SHA-256 Hash File

 UTILITIES
 10. Generate All Keys
 11. Run Performance Test

  0. Exit
==============================================================================

Enter your choice:
```

## How to Proceed:

1. If you choose 1, after selecting the encryption and decryption technique, you have to select key length (128, 256) for AES. Then you have to enter the plaintext to encrypt. Encrypted data is saved to encrypted/aes_[keysize]_ecb.bin. Besides, execution time will be displayed. Others have their own steps like this.

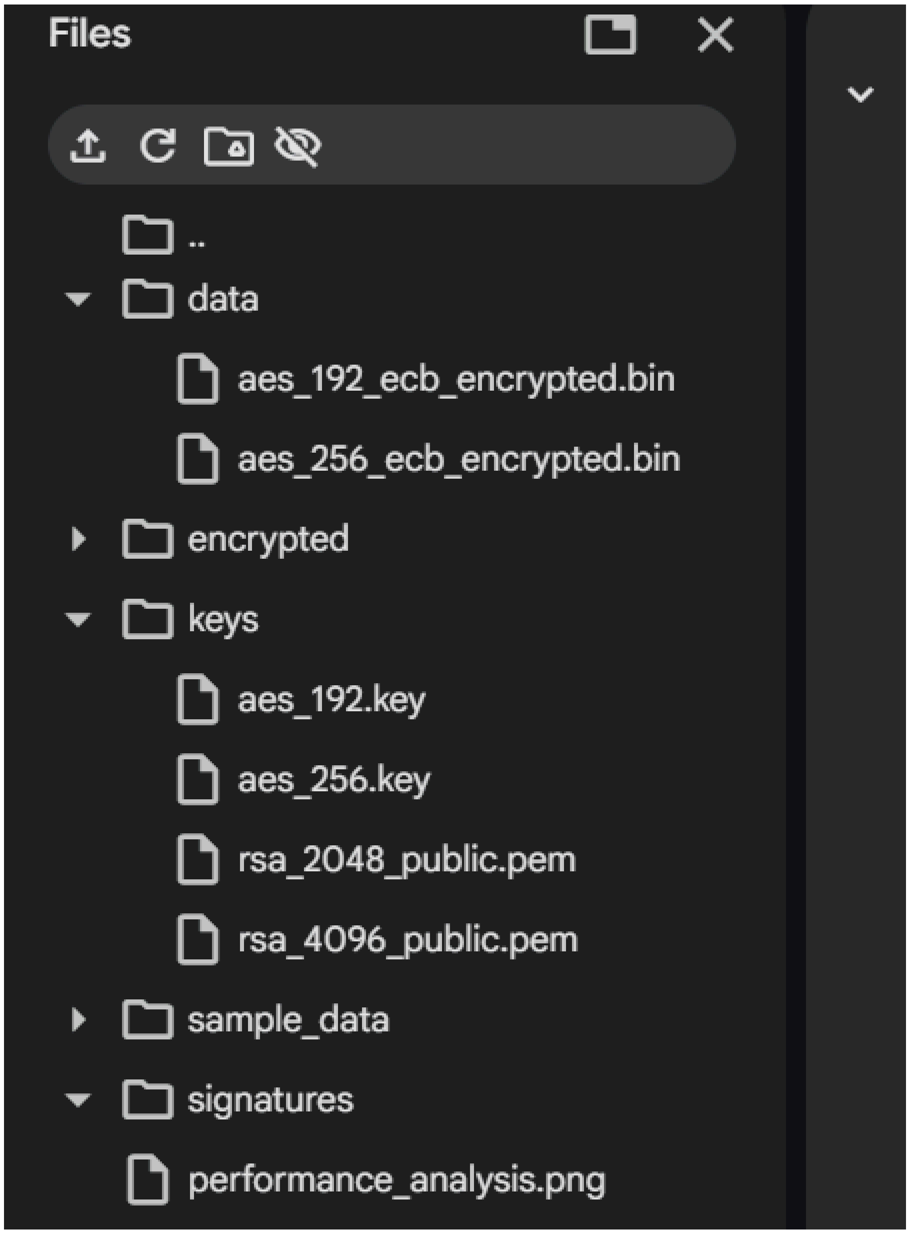
2. If you choose 5-6 (RSA Encryption/Decryption): Supported key sizes are 1024, 2048, 3072, 4096 bits. Then you have to enter the plaintext. Then the encrypted data will be saved in encrypted/rsa_[keysize].bin. Here also execution time will be displayed.

3. If you choose 7 (RSA Sign File), enter the file name to sign, enter the key size, The program computes the SHA-256 hash for the file. The signature is generated

using the private key. The signature is saved to: `signatures/signature_[keysize].sig`

4. If you select 8, you have to enter the file name to verify. Enter the RAS key size. The program reads the signature from the file. Verification result will be displayed.

And the folder structure will be like this screenshot:



```
Files
..
data
    aes_192_ecb_encrypted.bin
    aes_256_ecb_encrypted.bin
encrypted
keys
    aes_192.key
    aes_256.key
    rsa_2048_public.pem
    rsa_4096_public.pem
sample_data
signatures
performance_analysis.png
```

5. If you select 9, compute file hash, you have to enter the filename, view SHA-256 hash displayed in hexadecimal.

6. The performance test measures execution time as a function of key size for both AES testing and RSA testing.

## Link of the websites code snippet taken from:

Overall library structure and API reference: https://pycryptodome.readthedocs.io/
AES implementation, modes (ECB, CFB), padding:
https://pycryptodome.readthedocs.io/en/latest/src/cipher/aes.html
RSA key generation, encryption, decryption:
https://pycryptodome.readthedocs.io/en/latest/src/public_key/rsa.html
RSA signature generation and verification:
https://pycryptodome.readthedocs.io/en/latest/src/signature/pkcs1_v1_5.html
SHA-256 hash implementation:
https://pycryptodome.readthedocs.io/en/latest/src/hash/sha256.html
Creating performance graphs and visualizations:
https://matplotlib.org/stable/contents.html

**Specific code sections adapted from**:

1. AES Encryption/Decryption (ECB and CFB modes):
   Sections: Key generation, cipher initialization, padding
   Link: https://pycryptodome.readthedocs.io/en/latest/src/cipher/aes.html
2. RSA Key Generation:
   Sections: RSA.generate(), key export/import
   Link: https://pycryptodome.readthedocs.io/en/latest/src/public_key/rsa.html
3. RSA Encryption with OAEP:
   Sections: encrypt/decrypt methods
   Link: https://pycryptodome.readthedocs.io/en/latest/src/cipher/oaep.html