**Cryptographic Library**

i. Introduction

In this project, I try to implement basic cryptographic algorithms with a focus on performance and security.

ii. Features

- AES encryption and decryption

- Base64 encoding for ciphertext

- Scenario included as well

iii. Installation

1. Clone the repository:

git clone https://github.com/asuai-cs/crypto-library.git

2. Navigate to the project directory:

cd crypto-library

3. Install the required dependencies:

pip install -r requirements.txt

iv.How to use it

1. Import the `CryptoLibrary` class:

from crypto\_library import CryptoLibrary

from Crypto.Random import get\_random\_bytes

key = get\_random\_bytes(16)

crypto = CryptoLibrary(key)

message = "Secret Message"

encrypted\_message = crypto.encrypt(message)

print(f"Encrypted: {encrypted\_message}")

decrypted\_message = crypto.decrypt(encrypted\_message)

print(f"Decrypted: {decrypted\_message}")

v. Dependencies

- pycryptodome

vi. The code itself written in Python

# crypto\_library.py

from Crypto.Cipher import AES

from Crypto.Random import get\_random\_bytes

import base64

class CryptoLibrary:

def \_\_init\_\_(self, key):

self.key = key

def encrypt(self, plaintext):

cipher = AES.new(self.key, AES.MODE\_EAX)

ciphertext, tag = cipher.encrypt\_and\_digest(plaintext.encode())

return base64.b64encode(cipher.nonce + tag + ciphertext).decode()

def decrypt(self, b64\_ciphertext):

ciphertext = base64.b64decode(b64\_ciphertext)

nonce, tag, ciphertext = ciphertext[:16], ciphertext[16:32], ciphertext[32:]

cipher = AES.new(self.key, AES.MODE\_EAX, nonce=nonce)

plaintext = cipher.decrypt\_and\_verify(ciphertext, tag)

return plaintext.decode()

# Scenario

if \_\_name\_\_ == "\_\_main\_\_":

key = get\_random\_bytes(16)

crypto = CryptoLibrary(key)

message = "Secret Message"

encrypted\_message = crypto.encrypt(message)

print(f"Encrypted: {encrypted\_message}")

decrypted\_message = crypto.decrypt(encrypted\_message)

print(f"Decrypted: {decrypted\_message}")