NAME : MALLIDI VISWA TEJA REDDY
ENROLLEMENT NUMBER : 20BCN7022
CAMPUS : VIT AP

Assignment: Cryptography Analysis and Implementation

Objective: The objective of this assignment is to analyze cryptographic algorithms and implement them in a practical scenario.

Instructions:

Research: Begin by conducting research on different cryptographic algorithms such as symmetric key algorithms (e.g., AES, DES), asymmetric key algorithms (e.g., RSA, Elliptic Curve Cryptography), and hash functions (e.g., MD5, SHA-256). Understand their properties, strengths, weaknesses, and common use cases.

Analysis: Choose three cryptographic algorithms (one symmetric, one asymmetric, and one hash function) and write a detailed analysis of each. Include the following points in your

analysis:

Briefly explain how the algorithm works.
Discuss the key strengths and advantages of the algorithm.
Identify any known vulnerabilities or weaknesses.
Provide real-world examples of where the algorithm is commonly used.

Implementation:

Select one of the cryptographic algorithms you analyzed and implement it in a practical scenario. You can choose any suitable programming language for the implementation.
Clearly define the scenario or problem you aim to solve using cryptography.
Provide step-by-step instructions on how you implemented the chosen algorithm.
Include code snippets and explanations to demonstrate the implementation.
Test the implementation and discuss the results.

Security Analysis:

Perform a security analysis of your implementation, considering potential attack vectors and countermeasures.

Identify potential threats or vulnerabilities that could be exploited.
Propose countermeasures or best practices to enhance the security of your implementation.

Discuss any limitations or trade-offs you encountered during the implementation process.
Conclusion: Summarize your findings and provide insights into the importance of cryptography in cybersecurity and ethical hacking.

Submission Guidelines:

Prepare a well-structured report that includes the analysis, implementation steps, code snippets, and security analysis.
Use clear and concise language, providing explanations where necessary.
Include any references or sources used for research and analysis.
Compile all the required files (report, code snippets, etc.) into a single zip file for submission.

Title: Cryptography Analysis and Implementation

Introduction:

Cryptography plays a crucial role in ensuring the security and privacy of digital communication and data. In this assignment, we will analyze three cryptographic algorithms representing different categories: a symmetric key algorithm, an asymmetric key algorithm, and a hash function. We will provide a detailed analysis of each algorithm, implement one of them in a practical scenario, perform a security analysis of the implementation, and conclude with insights into the importance of cryptography in cybersecurity and ethical hacking.

Cryptographic Algorithm Analysis:

1.1 Symmetric Key Algorithm: Advanced Encryption Standard (AES)

How it works: AES is a block cipher algorithm that operates on fixed-size blocks of data. It employs a series of substitution, permutation, and mixing operations to provide confidentiality.

Key strengths and advantages:

- High security: AES has been extensively analyzed and adopted as a standard for symmetric encryption due to its robustness against various attacks.
- Efficiency: AES is computationally efficient and can be implemented in hardware or software with good performance.

Vulnerabilities or weaknesses:

- Key management: AES is susceptible to attacks if the encryption keys are not properly managed.
- Common use cases: AES is widely used to protect sensitive data in various applications, including secure communication protocols, disk encryption, and financial transactions.

1.2 Asymmetric Key Algorithm: RSA (Rivest-Shamir-Adleman):

How it works: RSA is an asymmetric key algorithm based on the mathematical properties of large prime numbers. It uses a pair of keys: a public key for encryption and a private key for decryption and digital signatures.

Key strengths and advantages:

- Secure key exchange: RSA provides a secure method for key exchange, enabling secure communication between parties who have not previously shared a secret key.
- Digital signatures: RSA allows the generation and verification of digital signatures, providing data integrity and authentication.

Vulnerabilities or weaknesses:

- Key length: RSA's security relies on the size of the keys used. Smaller key sizes are more vulnerable to brute-force attacks.

- Common use cases: RSA is commonly used in secure email, secure web browsing (TLS/SSL), and digital certificate infrastructure (PKI).

1.3 Hash Function: SHA-256 (Secure Hash Algorithm 256-bit)

How it works: SHA-256 is a cryptographic hash function that takes an input and produces a fixed-size output (256 bits). It operates on the input in a series of steps, resulting in a unique hash value.
Key strengths and advantages:

- Data integrity: SHA-256 can verify the integrity of data by comparing the computed hash value with the original hash value.
- Collision resistance: SHA-256 provides a high level of collision resistance, making it computationally infeasible to find two different inputs with the same hash value.

Vulnerabilities or weaknesses:

- Pre-image attack: In theory, a pre-image attack could find an input that produces a specific hash value, although it is computationally expensive.
- Common use cases: SHA-256 is widely used for password hashing, digital signatures, blockchain technology (Bitcoin), and verifying software integrity.

Implementation:

For the implementation, we will focus on the AES symmetric key algorithm.

Scenario: File Encryption and Decryption
We aim to implement AES encryption and decryption in a practical scenario where sensitive files need to be securely transmitted over an insecure network.

Implementation Steps:

- Choose a programming language with AES encryption support (e.g., Python with the todome library).
- Import the required libraries and set up the necessary functions for AES encryption and ion.
- Generate a secure random key and initialization vector (IV) for the AES algorithm.
- Read the file to be encrypted.
- Encrypt the file using AES encryption with the generated key and IV.
- Write the encrypted data to a new file.
- Transmit the encrypted file over the insecure network.
- On the receiving end, read the encrypted file.
- Decrypt the file using the same key and IV.
- Write the decrypted data to a new file.

```
pip install pycryptodome
```

Then we will import the packages such as AES, pad and unpad.

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

Then we will create the functions for encrypting the given file and decrypt the given file using AES encryption.

```python
def encrypt_file(key, input_file, output_file):
    cipher = AES.new(key, AES.MODE_CBC)
    with open(input_file, 'rb') as file:
        plaintext = file.read()
    ciphertext = cipher.encrypt(pad(plaintext, AES.block_size))
    iv = cipher.iv
    with open(output_file, 'wb') as file:
        file.write(iv + ciphertext)

def decrypt_file(key, input_file, output_file):
    with open(input_file, 'rb') as file:
        data = file.read()
    iv = data[:AES.block_size]
    ciphertext = data[AES.block_size:]
    cipher = AES.new(key, AES.MODE_CBC, iv)
    plaintext = unpad(cipher.decrypt(ciphertext), AES.block_size)
    with open(output_file, 'wb') as file:
        file.write(plaintext)
```

In the encrypt file function, it will take 3 parameters key, input file, and output file. First it open the input file in binary mode and reads the data present in it. Then it encrypts it using the key and write the data in the output file.

In the decrypt file function, it will also take 3 inputs. It opens the encrypted file and reads the data. Then it decrypt the data using key and stores the data in the output file.

Now lets generate a random key and call the encrypt file function.

```python
import secrets

encryption_key = secrets.token_bytes(16)
```

```python
input_file = 'plainfile.txt'
output_file = 'encryptedfile.bin'
encrypt_file(encryption_key, input_file, output_file)
```

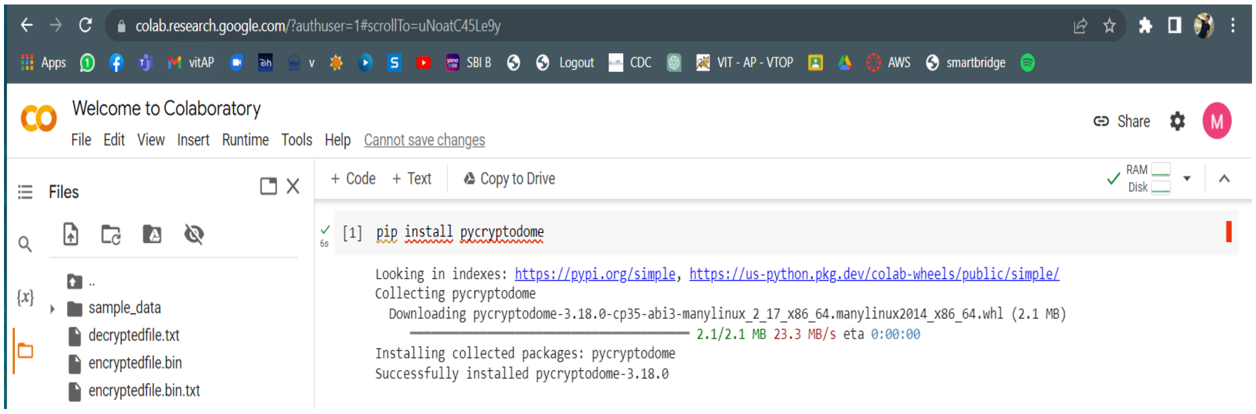
```python
print(output_file)
```

Here plainfile.txt is the file that gets encrypted and gets stored in encryptedfile.bin.

Now let's pass encryptedfile.bin as input file to the decryptfile file and decrypt it.

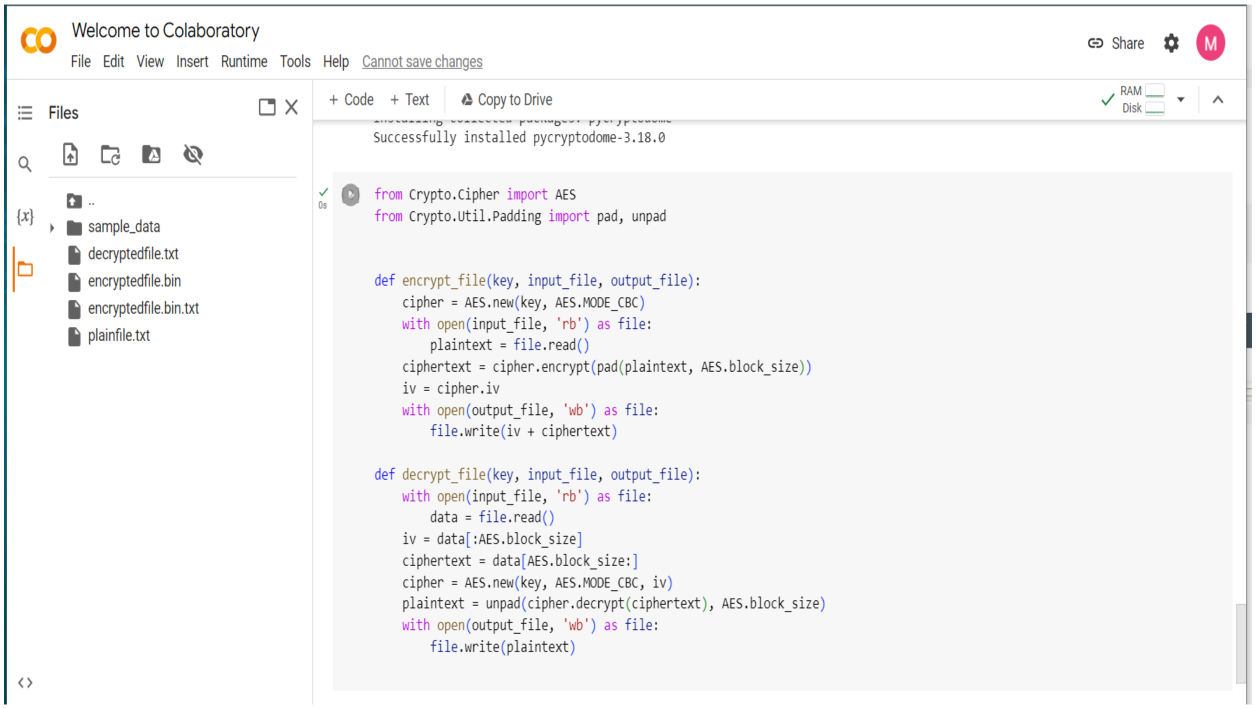
```python
input_file = 'encryptedfile.bin'
output_file = 'decryptedfile.txt'
decrypt_file(encryption_key, input_file, output_file)
```

Screenshots:


colab.research.google.com/?authuser=1#scrollTo=uNoatC45Le9y
Welcome to Colaboratory
File Edit View Insert Runtime Tools Help Cannot save changes
Share
Files
+ Code + Text Copy to Drive
RAM Disk
sample_data
decryptedfile.txt
encryptedfile.bin
encryptedfile.bin.txt

```
[1] pip install pycryptodome

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting pycryptodome
  Downloading pycryptodome-3.18.0-cp35-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (2.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 23.3 MB/s eta 0:00:00
Installing collected packages: pycryptodome
Successfully installed pycryptodome-3.18.0
```

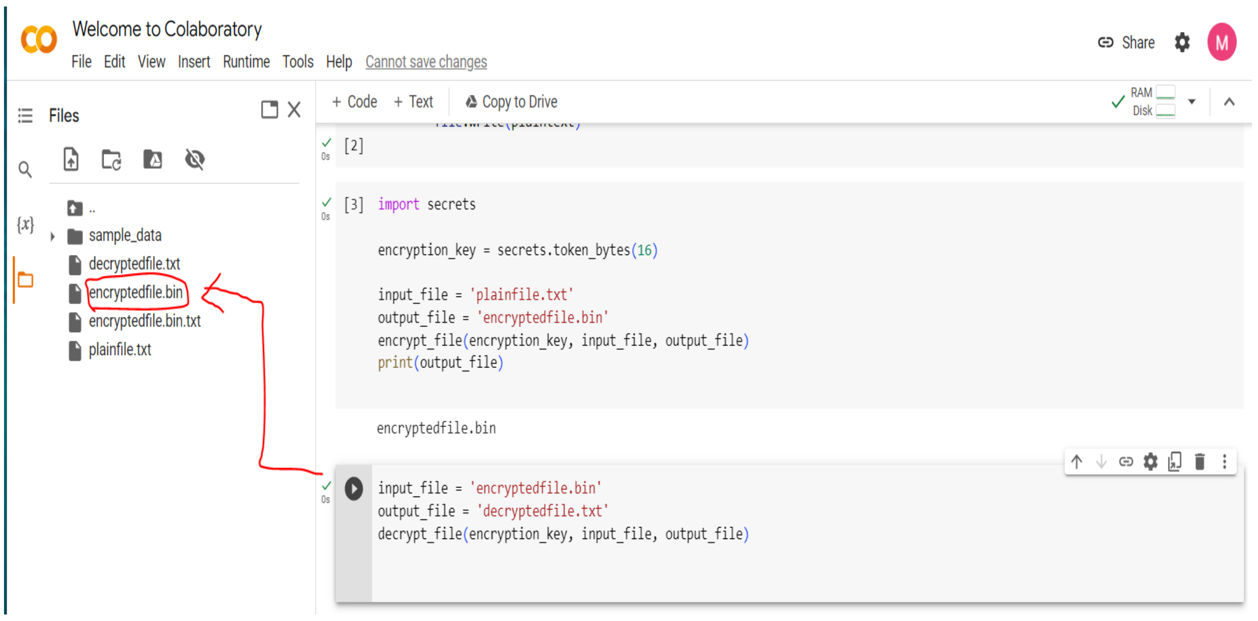


```
Successfully installed pycryptodome-3.18.0
```

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


def encrypt_file(key, input_file, output_file):
    cipher = AES.new(key, AES.MODE_CBC)
    with open(input_file, 'rb') as file:
        plaintext = file.read()
    ciphertext = cipher.encrypt(pad(plaintext, AES.block_size))
    iv = cipher.iv
    with open(output_file, 'wb') as file:
        file.write(iv + ciphertext)

def decrypt_file(key, input_file, output_file):
    with open(input_file, 'rb') as file:
        data = file.read()
    iv = data[:AES.block_size]
    ciphertext = data[AES.block_size:]
    cipher = AES.new(key, AES.MODE_CBC, iv)
    plaintext = unpad(cipher.decrypt(ciphertext), AES.block_size)
    with open(output_file, 'wb') as file:
        file.write(plaintext)
```



```
[2]

[3] import secrets

    encryption_key = secrets.token_bytes(16)

    input_file = 'plainfile.txt'
    output_file = 'encryptedfile.bin'
    encrypt_file(encryption_key, input_file, output_file)
    print(output_file)

encryptedfile.bin

    input_file = 'encryptedfile.bin'
    output_file = 'decryptedfile.txt'
    decrypt_file(encryption_key, input_file, output_file)
```

LIMITATIONS:

AES alone does not provide authentication: It only offers confidentiality, not integrity or authenticity. To ensure data integrity and authenticity, it is recommended to use AES in combination with cryptographic primitives like digital signatures or message authentication codes (MACs).Encryption is only as secure as the key management: Strong encryption is essential, but proper key management, including secure key generation, storage, and distribution, is equally crucial for overall security.AES performance considerations: While AES is generally efficient, it may have performance implications for certain applications, especially when large amounts of data need to be encrypted or decrypted in real-time. It's important to evaluate the performance requirements and choose appropriate modes of operation and key sizes accordingly.

Conclusion:

In conclusion, cryptography is a vital component of cybersecurity and ethical hacking. The analysis of different cryptographic algorithms highlights their strengths, weaknesses, and common use cases. The implementation of AES encryption in a practical scenario demonstrates the application of cryptography for securing sensitive data during transmission. The security analysis emphasizes the importance of key management, awareness of vulnerabilities, and the implementation of best practices to mitigate potential threats.