ENCRYPTION AND DECRYPTION USING MONOALPHABETIC CIPHER

A COURSE PROJECT REPORT

By

P VINAY(RA2011030010138)
PAVAN SREERAM (RA2011030010127)

Under the guidance of

Dr. Mary Subaja Christo

(Associate Professor, Department of Networking and Communication)

In partial fulfilment for the Course

of

18CSC381T - CRYPTOGRAPHY

in Networking and Communication

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur, Chenpalpattu District

NOVEMBER 2022

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this mini project report "AES Encryption and Decryption using pycrypto library" is the bonafide work of **K pavansreeram-RA2011030010127 P Vinay (RA2011030010138)** who carried out the project work under my supervision.

SIGNATURE

Dr. Mary Subaja Christo
Associate Professor
Networking and Communication
SRM Institute of Science and Technology

ABSTRACT

In the digital era, being hacked is a common happening worldwide. With communications over the cloud, the privacy of data sent and received, is vulnerable. Cryptography is being a protector by safeguarding the data

communicated. In today's world where everything is possible to get hacked or being tempered while

communicating between sender and receiver, in such situation we do want anyone else to access our data or private messages. With digital currencies a.k.a. cryptocurrencies on the rise, it is of utmost priority to build a stronger anti-hack mechanism to protect them. The block-chain, that protects the digital currencies, is fundamentally based on cryptography. The process of converting ordinary and plain text into unintelligible text and vice versa is known as cryptography. In this method, data is stored and transmitted in a specific form in order to make it available for only particular people to read and process. Its importance lies in the fact that it protects data from hacking and alteration, while making it useful for user authentication. Phil Zimmermann defines cryptography as the science of using mathematics to encrypt and decrypt data Bruce Schneier says, Cryptography is the art and science of keeping messages secure.

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy,** for his encouragement

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.Gopal,** for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman,** for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professor Dr.Annapurani Panaiyappan, Professor and Head, Department of Networking and Communications** and **Course Coordinators** for their constant encouragement and support.

We are highly thankful to our my Course project Faculty **Dr. Mary Subaja Christo , Associate Professor , Networking and Communication,** for his/her assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our **Dr. Annapurani Panaiyappan, Course Coordinator, Professor and Head, Department of Networking and Communication** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

TABLE OF CONTENTS

1. INTRODUCTION

Advanced Encryption Standard(AES) is a symmetric encryption algorithm. AES is the industry standard as of now as it allows 128 bit, 192 bit and 256 bit encryption. Symmetric encryption is very fast as compared to asymmetric encryption and are used in systems such as database system. Following is an online tool to generate AES encrypted password and decrypt AES encrypted password. It provides two mode of encryption and decryption ECB and CBC mode. The AES algorithm has a 128-bit block size, regardless of whether you key length is 256, 192 or 128 bits. When a symmetric cipher mode requires an IV, the length of the IV must be equal to the block size of the cipher. Hence, you must always use an IV of 128 bits (16 bytes) with AES.

AES provides 128 bit, 192 bit and 256 bit of secret key size for encryption. If you are selecting 128 bits for encryption, then the secret key must be of 16 bits long and 24 and 32 bits for 192 and 256 bits of key size respectively. For example if the key size is 128 then a valid secret key must be of 16 characters i.e. 16*8=128 bits

2. DESIGN AND IMPLEMENTATION
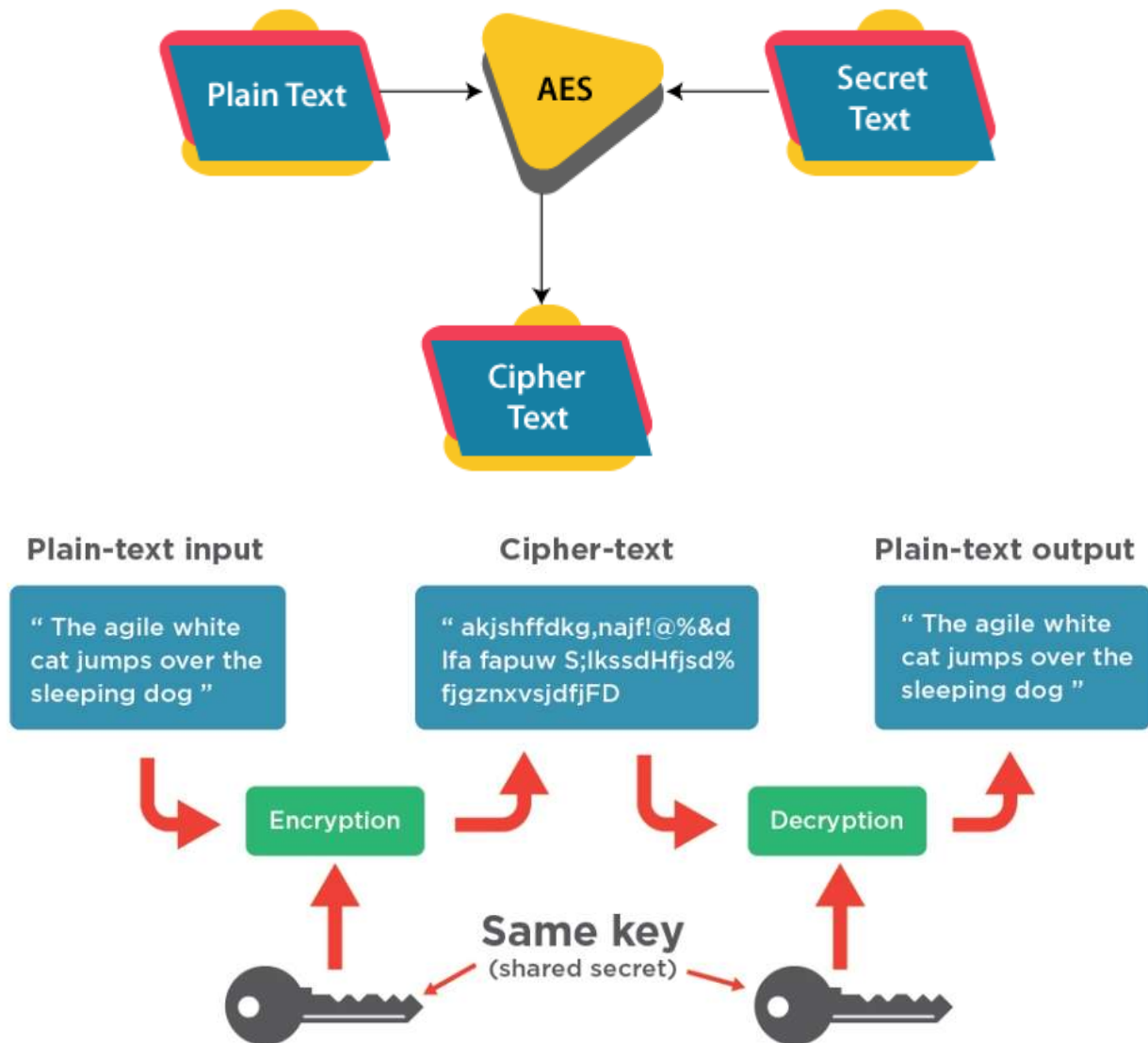
- Import important libraries
- Define variables and dictionaries
- Create important functions
- Import important libraries

```
pip install pycrypto
```

- Take a input password from user for the encryption.

```python
password = input("Enter encryption password:")
```

- define a function "encrypt" which takes the input entered by the user
- perform encryption by mapping password to the message.
- perform decryption by mapping cipher text to the password.

```python
def encrypt(raw, password):
    private_key = hashlib.sha256(password.encode("utf-8")).digest()
    raw = pad(raw)
    iv = Random.new().read(AES.block_size)
    cipher = AES.new(private_key, AES.MODE_CBC, iv)
    return base64.b64encode(iv + cipher.encrypt(raw))
```

```python
def decrypt(enc, password):
    private_key = hashlib.sha256(password.encode("utf-8")).digest()
    enc = base64.b64decode(enc)
    iv = enc[:16]
    cipher = AES.new(private_key, AES.MODE_CBC, iv)
    return unpad(cipher.decrypt(enc[16:]))
```

3. RESULTS AND DISCUSSION

Secret password :


```
cryptography.ipynb  ☆
File  Edit  View  Insert  Runtime  Tools  Help

+ Code   + Text

import base64
import hashlib
from Crypto.Cipher import AES
from Crypto import Random

BLOCK_SIZE = 16
pad = lambda s: s + (BLOCK_SIZE - len(s) % BLOCK_SIZE) * chr(BLOCK_SIZE - len(s) % BLOCK_SIZE)
unpad = lambda s: s[:-ord(s[len(s) - 1:])]
password = input("Enter encryption password:")

def encrypt(raw, password):
    private_key = hashlib.sha256(password.encode("utf-8")).digest()
    raw = pad(raw)
    iv = Random.new().read(AES.block_size)
    cipher = AES.new(private_key, AES.MODE_CBC, iv)
    return base64.b64encode(iv + cipher.encrypt(raw))

def decrypt(enc, password):
    private_key = hashlib.sha256(password.encode("utf-8")).digest()
    enc = base64.b64decode(enc)
    iv = enc[:16]
    cipher = AES.new(private_key, AES.MODE_CBC, iv)
    return unpad(cipher.decrypt(enc[16:]))

# First let us encrypt secret message
encrypted = encrypt("This is a secret message", password)
print(encrypted)

# Let us decrypt using our original password
decrypted = decrypt(encrypted, password)
print(bytes.decode(decrypted))

...  Enter encryption password:
```

Fig 3.1 password given for encryption

Encryption:

```python
import hashlib
from Crypto.Cipher import AES
from Crypto import Random

BLOCK_SIZE = 16
pad = lambda s: s + (BLOCK_SIZE - len(s) % BLOCK_SIZE) * chr(BLOCK_SIZE - len(s) % BLOCK_SIZE)
unpad = lambda s: s[:-ord(s[len(s) - 1:])]

password = input("Enter encryption password: ")


def encrypt(raw, password):
    private_key = hashlib.sha256(password.encode("utf-8")).digest()
    raw = pad(raw)
    iv = Random.new().read(AES.block_size)
    cipher = AES.new(private_key, AES.MODE_CBC, iv)
    return base64.b64encode(iv + cipher.encrypt(raw))


def decrypt(enc, password):
    private_key = hashlib.sha256(password.encode("utf-8")).digest()
    enc = base64.b64decode(enc)
    iv = enc[:16]
    cipher = AES.new(private_key, AES.MODE_CBC, iv)
    return unpad(cipher.decrypt(enc[16:]))

a=input("enter the message to encrypt:")
# First let us encrypt secret message
encrypted = encrypt(a, password)
print(encrypted)


# Let us decrypt using our original password
decrypted = decrypt(encrypted, password)
print("The decrypted message is:",decrypted)
```

```
Enter encryption password: @yiwow##1
enter the message to encrypt:[            ]
```

Fig no 3.1: Output of the code which takes users input

Run the code , and it displays "Enter the message" which takes user input .

```python
from Crypto.Cipher import AES
from Crypto import Random

BLOCK_SIZE = 16
pad = lambda s: s + (BLOCK_SIZE - len(s) % BLOCK_SIZE) * chr(BLOCK_SIZE - len(s) % BLOCK_SIZE)
unpad = lambda s: s[:-ord(s[len(s) - 1:])]

password = input("Enter encryption password: ")


def encrypt(raw, password):
    private_key = hashlib.sha256(password.encode("utf-8")).digest()
    raw = pad(raw)
    iv = Random.new().read(AES.block_size)
    cipher = AES.new(private_key, AES.MODE_CBC, iv)
    return base64.b64encode(iv + cipher.encrypt(raw))


def decrypt(enc, password):
    private_key = hashlib.sha256(password.encode("utf-8")).digest()
    enc = base64.b64decode(enc)
    iv = enc[:16]
    cipher = AES.new(private_key, AES.MODE_CBC, iv)
    return unpad(cipher.decrypt(enc[16:]))

a=input("enter the message to encrypt:")
# First let us encrypt secret message
encrypted = encrypt(a, password)
print("The cipher text is:",encrypted)


# Let us decrypt using our original password
decrypted = decrypt(encrypted, password)
print("The decrypted message is:",decrypted)
```

```
Enter encryption password: @ylwow##1
enter the message to encrypt:these is cryptography mini project
The cipher text is: b'PVLCZi+2B9iU/JOcF8DpoOpmGymik0iWxB4CKGXF0CeX0sd0lssZJ50YNoW18xZteVk4IxSM3Zsq9Vq/zSUocQ=='
```

Fig no 3.2: Encrypted text of the given word

Decryption:

```python
from Crypto.Cipher import AES
from Crypto import Random

BLOCK_SIZE = 16
pad = lambda s: s + (BLOCK_SIZE - len(s) % BLOCK_SIZE) * chr(BLOCK_SIZE - len(s) % BLOCK_SIZE)
unpad = lambda s: s[:-ord(s[len(s) - 1:])]

password = input("Enter encryption password: ")


def encrypt(raw, password):
    private_key = hashlib.sha256(password.encode("utf-8")).digest()
    raw = pad(raw)
    iv = Random.new().read(AES.block_size)
    cipher = AES.new(private_key, AES.MODE_CBC, iv)
    return base64.b64encode(iv + cipher.encrypt(raw))


def decrypt(enc, password):
    private_key = hashlib.sha256(password.encode("utf-8")).digest()
    enc = base64.b64decode(enc)
    iv = enc[:16]
    cipher = AES.new(private_key, AES.MODE_CBC, iv)
    return unpad(cipher.decrypt(enc[16:]))

a=input("enter the message to encrypt:")
# First let us encrypt secret message
encrypted = encrypt(a, password)
print("The cipher text is:",encrypted)


# Let us decrypt using our original password
decrypted = decrypt(encrypted, password)
print("The decrypted message is:",decrypted)
```

```
Enter encryption password: @yiwow##1
enter the message to encrypt:these is cryptography mini project
The cipher text is: b'PVLCZi+2B9iU/JOcF8DpoOpmGymik0iWxB4CKGXF0CeX0sd0lssZJ50YNoW18xZteVk4IxSM3Zsq9Vq/zSUocQ=='
The decrypted message is: b'these is cryptography mini project'
```

Fig no 3.3 : Decryption of the given text

4. CONCLUSION AND FUTURE ENHANCEMENT

So , in this way we can encrypt and decrypt the message by using password with the help of AES.

AES offers 2 different modes of encryption - ECB and CBC modes.

ECB(Electronic Code Book) is the simplest encryption mode and does not require IV for encryption. The input plain text will be divided into blocks and each block will be encrypted with the key provided and hence identical plain text blocks are encrypted into identical cipher text blocks.

CBC(Cipher Block Chaining) mode is highly recommended, and it is an advanced form of block cipher encryption. It requires IV to make each message unique meaning the identical plain text blocks are encrypted into dissimilar cipher text blocks. Hence, it provides more robust encryption as compared to ECB mode, but it is a bit slower as compared to ECB mode. If no IV is entered then default will be used here for CBC mode and that defaults to a zero based byte[16].

These types helps to increase the security of the message and makes difficult for the unknown users to encrypt or decrypt the messages.

REFERENCES

1. https://www.tutorialspoint.com/what-is-monoalphabetic-cipher-in-information-security

2. https://russell.ballestrini.net/monoalphabetic-cipher-and-inverse-written-in-python/

3. https://crypto.interactive-maths.com/monoalphabetic-substitution-ciphers.html

4. https://www.techtarget.com/searchsecurity/definition/cryptography

APPENDIX

CODE:

```python
import base64
import hashlib
from Crypto.Cipher import AES
from Crypto import Random

BLOCK_SIZE = 16
def pad(s): return s + (BLOCK_SIZE - len(s) %
                        BLOCK_SIZE) * chr(BLOCK_SIZE - len(s) % BLOCK_SIZE)


def unpad(s): return s[:-ord(s[len(s) - 1:])]


password = input("Enter encryption password: ")


def encrypt(raw, password):
    private_key = hashlib.sha256(password.encode("utf-8")).digest()
    raw = pad(raw)
    iv = Random.new().read(AES.block_size)
    cipher = AES.new(private_key, AES.MODE_CBC, iv)
    return base64.b64encode(iv + cipher.encrypt(raw))


def decrypt(enc, password):
    private_key = hashlib.sha256(password.encode("utf-8")).digest()
    enc = base64.b64decode(enc)
    iv = enc[:16]
    cipher = AES.new(private_key, AES.MODE_CBC, iv)
    return unpad(cipher.decrypt(enc[16:]))


a = input("enter the message to encrypt:")
# First let us encrypt secret message
encrypted = encrypt(a, password)
print("The cipher text is:", encrypted)


# Let us decrypt using our original password
decrypted = decrypt(encrypted, password)
print("The decrypted message is:", decrypted)
```

OUTPUT:

Enter encryption password: @yiwow##1
enter the message to encrypt:these is cryptography mini project
The cipher text is: b'PVLCZi+2B9iU/JOcF8DpoOpmGymik0iWxB4CKGXF0CeX0sd0lssZJ50YNoW18xZteVk4IxSM3Zsq9Vq/zSUocQ=='
The decrypted message is: b'these is cryptography mini project'