PROJECT REPORT

Computer Networks Lab

<u>Implementation Of AES-128 over WLAN</u>

Submitted by: Samiksha Jain (17103175)
Ashish Kr Khetan (17103178)
Kartik Ahluwalia (17103189)
Disha Sawlani (17103195)

Project Overview

Introduction

The project is an implementation of the AES algorithm and the encryption of files of some supported formats i.e. Jpg, jpeg, png, mp3, mp4 etc. and its transfer over a shared WLAN network followed by decryption of the downloaded file over the host using HttpServer method for local file sharing. Providing both server to client, client to server with a single program utility.

Project Scope

The project aims at providing a utility for data sharing in multiple formats over shared connections in private or public networks. Also to prevent misuse of the encryption and decryption program over the sender as well as receiver end, the project is password protected as well. The project is mainly divided into two main modules, description of which follows:

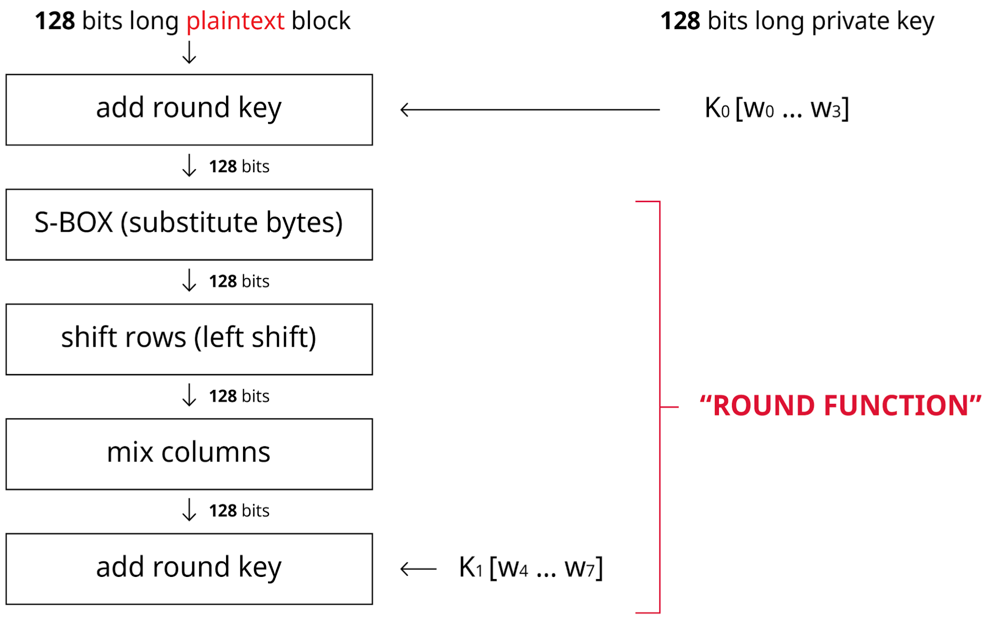
- **AES Algorithm**

  The National Institute of Standards and Technology (NIST) started development of AES in 1997 when it announced the need for a successor algorithm for the Data Encryption Standard (DES), which was starting to become vulnerable to brute-force attacks. The main objective of the AES Technique was to develop an encryption environment friendly for both the software as well as hardware interfaces. The AES algorithm is currently one of the most encryption techniques used worldwide for text encryption as well as file encryption.

- **FILE Share using WLAN**

  A web server processes incoming network requests over HTTP and several other related protocols. The primary function of a web server is to store, process and deliver web pages to clients. The communication between client and server takes place using the Hypertext Transfer Protocol (HTTP).

How does AES Work?



AES encryption needs a strong key. The stronger the key, the stronger your encryption. The algorithm implements the Block cipher technique, cryptosystem which encrypts data not by a bit but by block which is group of bits, applying algorithm per block. AES encryption in the project uses a 16-Bit Key for encryption of the imported files. Which is later sent over the local area network and decrypted accordingly.

How is AES Applied on audio files , Images and other similar files?

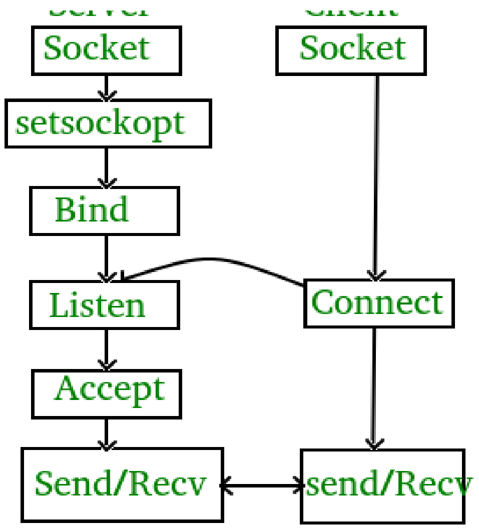
Every file is a composition of binary data regardless of what is assumed otherwise. Hence, the AES encryption works well on all most all file types.

File Sharing using WLAN

Through simple http host commands, over cmd.exe, we will use some simple and basic technique to share files and decrypt them using the httpServer message. The ipconfig command lists the IP Address of all possible and in-use ports over the PC used for data sharing over the network.

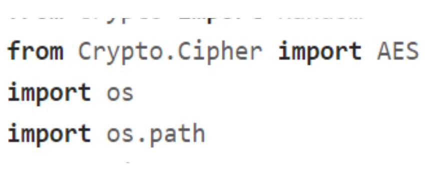
Socket Programming to stimulate multiple user transaction

State diagram for server and client model



IMPLEMENTATION

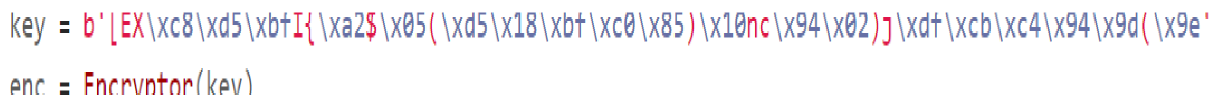
Imports:



```
from Crypto.Cipher import AES
import os
import os.path
```

From the Crypto Library in Python, the Program imports Random for generation of a random key used for encryption, alongside which AES is imported for the normal implementation of the algorithm.

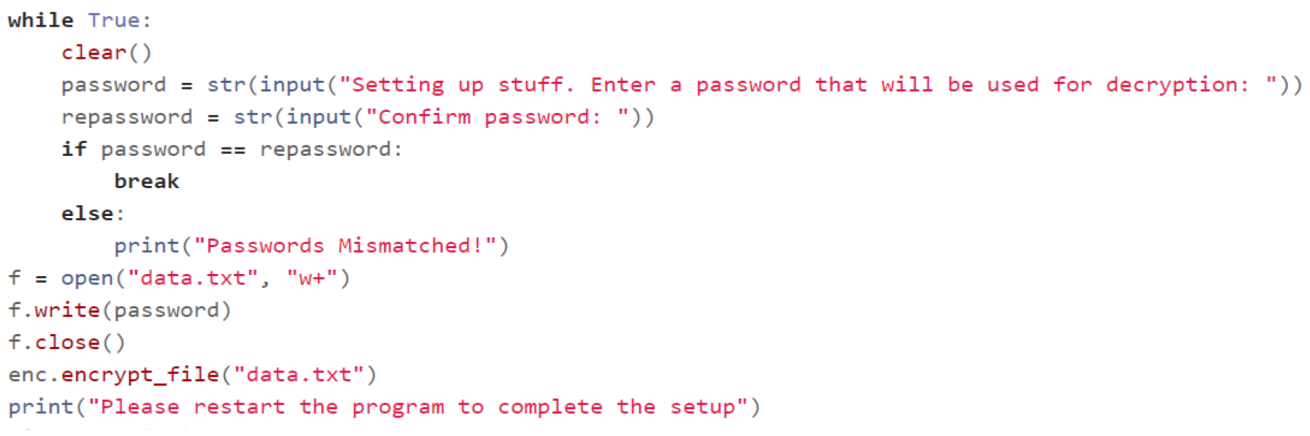
Encryption using a 128-Bit key:



```
key = b'[EX\xc8\xd5\xbfI{\xa2$\x05(\xd5\x18\xbf\xc0\x85)\x10nc\x94\x02)j\xdf\xcb\xc4\x94\x9d(\x9e'
enc = Encryptor(key)
```

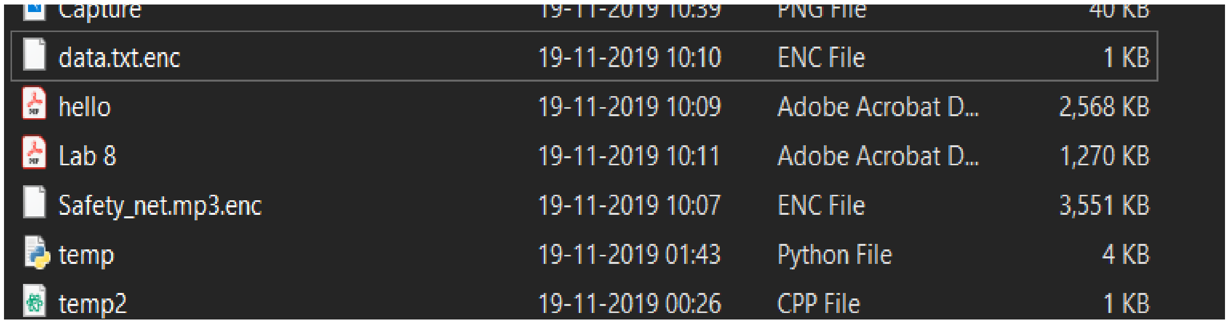
The Program uses a 128-Bit key encryption technique which is constant over the server host as well as the client host using the data file sharing over the Local Area Network created. The Key is kept is constant over client as well as server side to make it possible to decrypt files, which elsewise would not have been possible.

Password Creation on first Program Prompt:



```
while True:
    clear()
    password = str(input("Setting up stuff. Enter a password that will be used for decryption: "))
    repassword = str(input("Confirm password: "))
    if password == repassword:
        break
    else:
        print("Passwords Mismatched!")
f = open("data.txt", "w+")
f.write(password)
f.close()
enc.encrypt_file("data.txt")
print("Please restart the program to complete the setup")
```

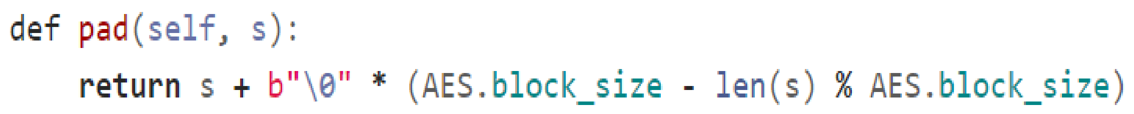
The program is password protected as well. On the very first call of the program in which the python file exists, the program prompts user to create a password to allow encryptions and decryption of the files in the directory, the next time program runs. The details of the passwords are stored in an encrypted file in the directory itself. (data.txt.enc)



Capture	19-11-2019 10:39	PNG File	40 KB
data.txt.enc	19-11-2019 10:10	ENC File	1 KB
hello	19-11-2019 10:09	Adobe Acrobat D...	2,568 KB
Lab 8	19-11-2019 10:11	Adobe Acrobat D...	1,270 KB
Safety_net.mp3.enc	19-11-2019 10:07	ENC File	3,551 KB
temp	19-11-2019 01:43	Python File	4 KB
temp2	19-11-2019 00:26	CPP File	1 KB

Which when decrypted using the same program, contains the password that would allow subsequent runs of the program in the directory.

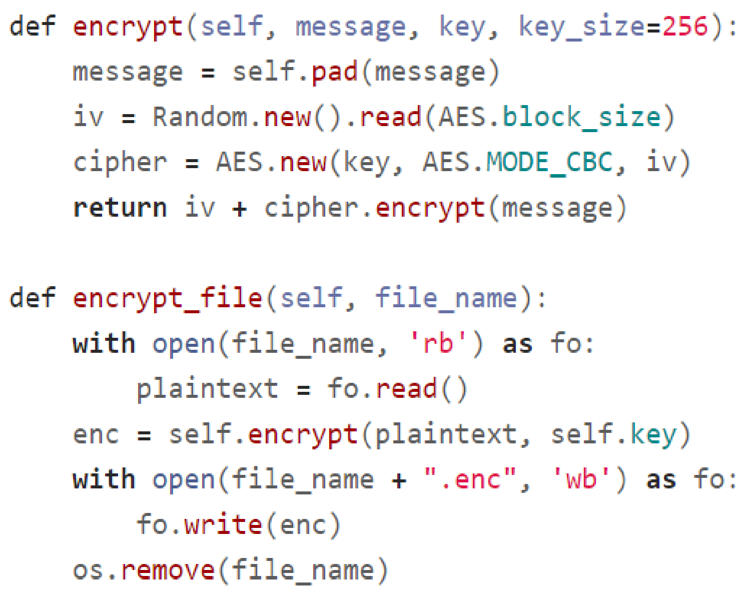
Data Padding:



```
def pad(self, s):
    return s + b"\0" * (AES.block_size - len(s) % AES.block_size)
```

Data padding is used in the implementation of the program as for the normal functioning of the AES encryption technique, data has to be divided into blocks of 4*4 matrices, which is essential for the implementation of the 128-Bit key algorithm used in our program.

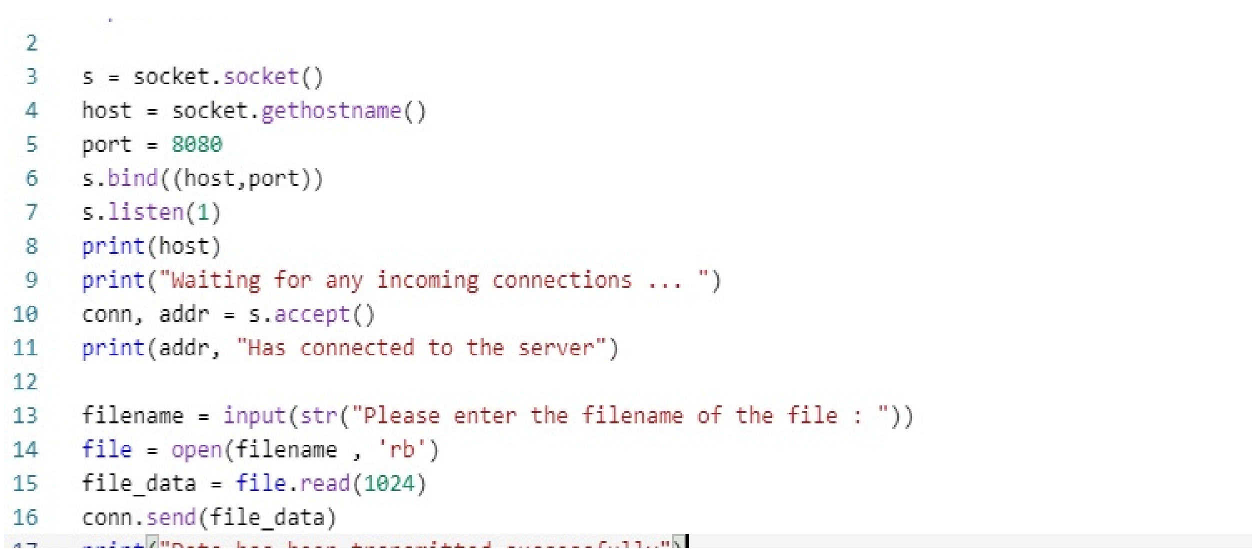
Encryption:



```
def encrypt(self, message, key, key_size=256):
    message = self.pad(message)
    iv = Random.new().read(AES.block_size)
    cipher = AES.new(key, AES.MODE_CBC, iv)
    return iv + cipher.encrypt(message)

def encrypt_file(self, file_name):
    with open(file_name, 'rb') as fo:
        plaintext = fo.read()
    enc = self.encrypt(plaintext, self.key)
    with open(file_name + ".enc", 'wb') as fo:
        fo.write(enc)
    os.remove(file_name)
```

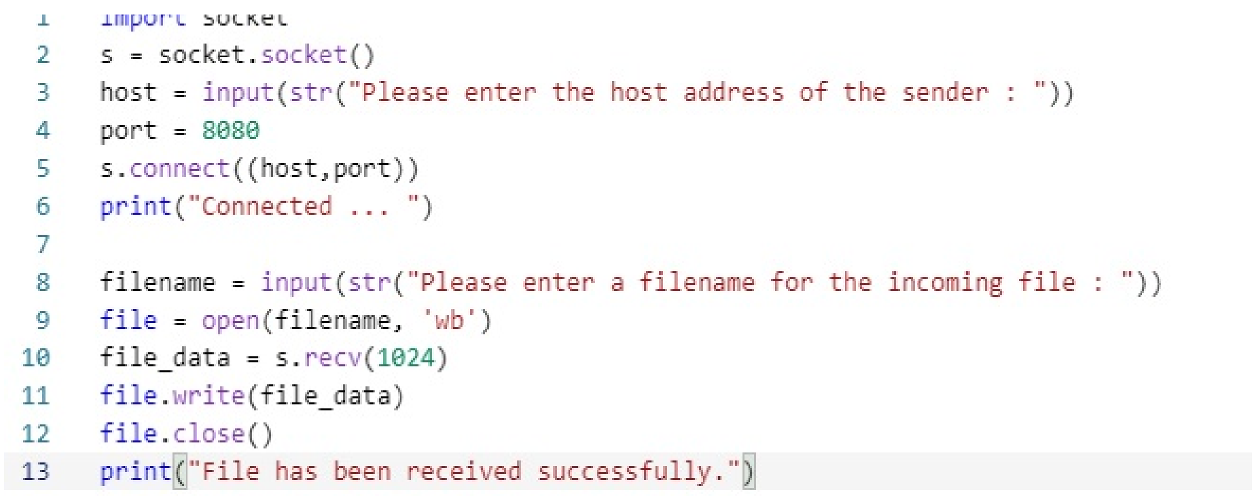
The snapshot represents the implementation of the AES-128 encryption technique used in our program. The encryption function uses a random matrix generation for the round matrix computations in subsequent follow-ups of the AES algorithm. The program encrypts the file irrespective of the file type, replaces the files with the encrypted file in the same directory.

Setting Up Server:



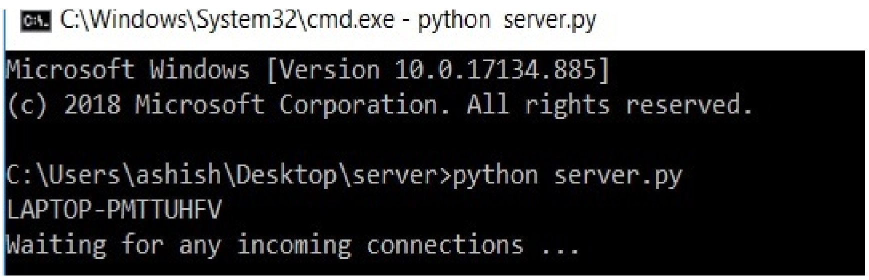
```

s = socket.socket()
host = socket.gethostname()
port = 8080
s.bind((host,port))
s.listen(1)
print(host)
print("Waiting for any incoming connections ... ")
conn, addr = s.accept()
print(addr, "Has connected to the server")

filename = input(str("Please enter the filename of the file : "))
file = open(filename , 'rb')
file_data = file.read(1024)
conn.send(file_data)
```

Setting Up Client:

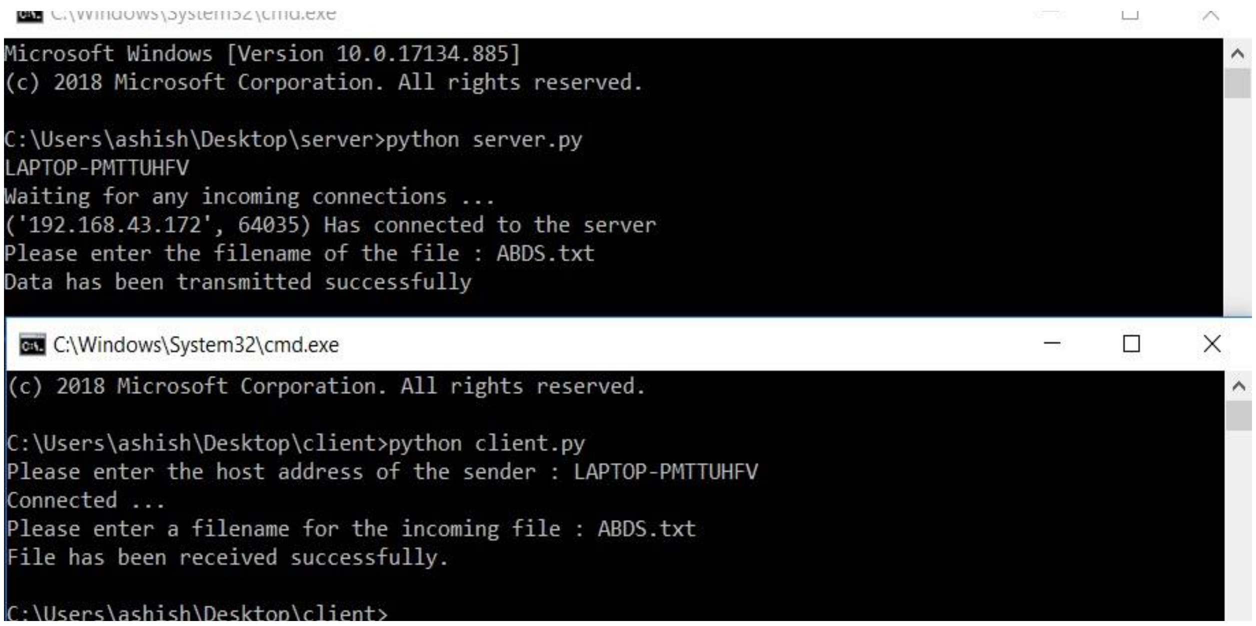


```
import socket
s = socket.socket()
host = input(str("Please enter the host address of the sender : "))
port = 8080
s.connect((host,port))
print("Connected ... ")

filename = input(str("Please enter a filename for the incoming file : "))
file = open(filename, 'wb')
file_data = s.recv(1024)
file.write(file_data)
file.close()
print("File has been received successfully.")
```

Stimulating Traversal:



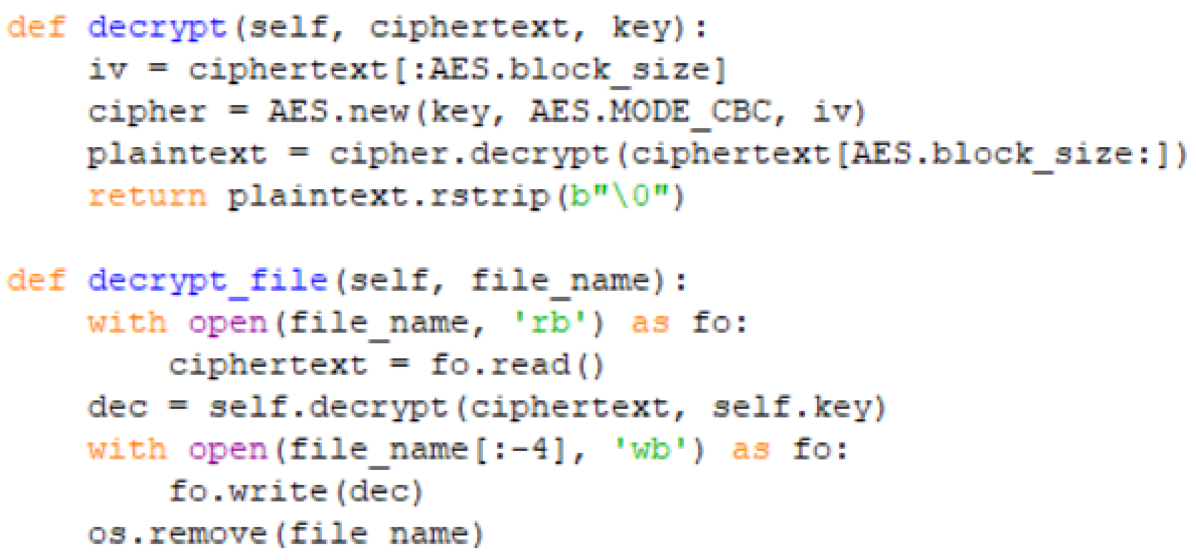
```
Microsoft Windows [Version 10.0.17134.885]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\ashish\Desktop\server>python server.py
LAPTOP-PMTTUHFV
Waiting for any incoming connections ...
```



```
Microsoft Windows [Version 10.0.17134.885]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\ashish\Desktop\server>python server.py
LAPTOP-PMTTUHFV
Waiting for any incoming connections ...
('192.168.43.172', 64035) Has connected to the server
Please enter the filename of the file : ABDS.txt
Data has been transmitted successfully
```

```
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\ashish\Desktop\client>python client.py
Please enter the host address of the sender : LAPTOP-PMTTUHFV
Connected ...
Please enter a filename for the incoming file : ABDS.txt
File has been received successfully.

C:\Users\ashish\Desktop\client>
```

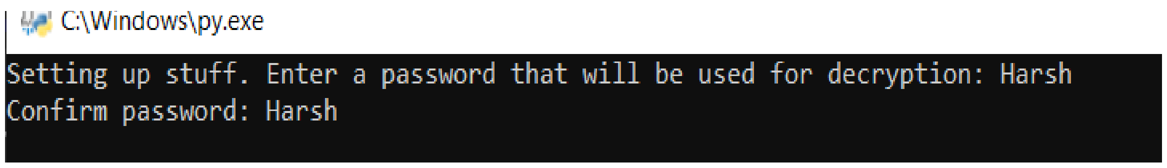
Decryption:



```
def decrypt(self, ciphertext, key):
    iv = ciphertext[:AES.block_size]
    cipher = AES.new(key, AES.MODE_CBC, iv)
    plaintext = cipher.decrypt(ciphertext[AES.block_size:])
    return plaintext.rstrip(b"\0")

def decrypt_file(self, file_name):
    with open(file_name, 'rb') as fo:
        ciphertext = fo.read()
    dec = self.decrypt(ciphertext, self.key)
    with open(file_name[:-4], 'wb') as fo:
        fo.write(dec)
    os.remove(file_name)
```

The snapshot represents the implementation of the AES-128 decryption technique used in our program. The same program is now run on the client side and using the deciphering function, files received over the WLAN can be decrypted and used as normal files

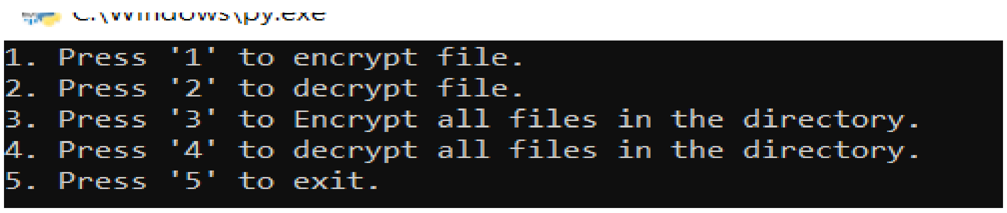
Working App Screenshots:



C:\Windows\py.exe

```
Setting up stuff. Enter a password that will be used for decryption: Harsh
Confirm password: Harsh
```

Password setup on first program prompt.



```
C:\Windows\py.exe
1. Press '1' to encrypt file.
2. Press '2' to decrypt file.
3. Press '3' to Encrypt all files in the directory.
4. Press '4' to decrypt all files in the directory.
5. Press '5' to exit.
```

Encryption and Decryption Menu on subsequent program runs.

Repository Link

https://github.com/JunkyPlatipuss/aes128cn/upload

References:

https://medium.com/@mstahir/how-aes-algorithm-works-701ef5cebc7c

https://medium.com/@14wnrkim/what-is-aes-step-by-step-fcb2ba41bb20

https://pycryptodome.readthedocs.io/en/latest/src/cipher/aes.html