Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110

(An Autonomous Institution, Affiliated to Anna University, Chennai)

OS Lab - Mini Project

Cryptographic File System

Team members:

Harish A	205002038	**Branch & Dept:** BTech- IT
Natarajan G	205002052	**Sec**: 'A'
Nithin BK	205002055	
Pavan GR	205002060	
Prajeeth B	205002062	

Problem Statement:

To implement Cryptographic File System.

Problem Description:

The idea is to encrypt and decrypt files inside a given directory using any encryption module. The encryption type used for this project is AES (Advanced Encryption Standard). Advanced Encryption Standard, also known as Rijndael, is an encoding algorithm that transforms plain text data into a version known as cipher text that's not possible for humans or machines to understand without an encryption key—a password. This tool can encrypt (Images/Videos/Audios/Text/) and other files as well. We can encrypt and decrypt any type of file using this tool. It is an Encryption and Decryption tool written in python which is used to encrypt any type of file based on AES Standards and the files that are encrypted using this script can also be decrypted.

Overview of the AES Algorithm:

Without going into the mathematical details of the algorithm, you need to understand the different parameters to have a better grasp of how libraries implement AES.

The AES algorithm can be divided into 3 main parts:

Generate a key
Generate a cipher
Encrypt/decrypt the data with the cipher

1. Generating the AES key

AES requires a secret passphrase known as a "key" to encrypt/decrypt data. Anybody with the key can decrypt your data, so you need it to be

strong and hidden from everyone—only the software program should be able to access it.

The key can be either 128, 192, 256, or 512 bit long. An AES cipher using a 512-bit key is abbreviated as AES 512, for example. The longer the key, the more secure it is, but the slower the encryption/decryption will be as well. 128 bits is equivalent to 24 characters in base64 encoding, 44 characters for 256 bits. Since storage space isn't usually a problem and the difference in speed between the versions is negligible, a good rule of thumb is to use 256-bit keys.

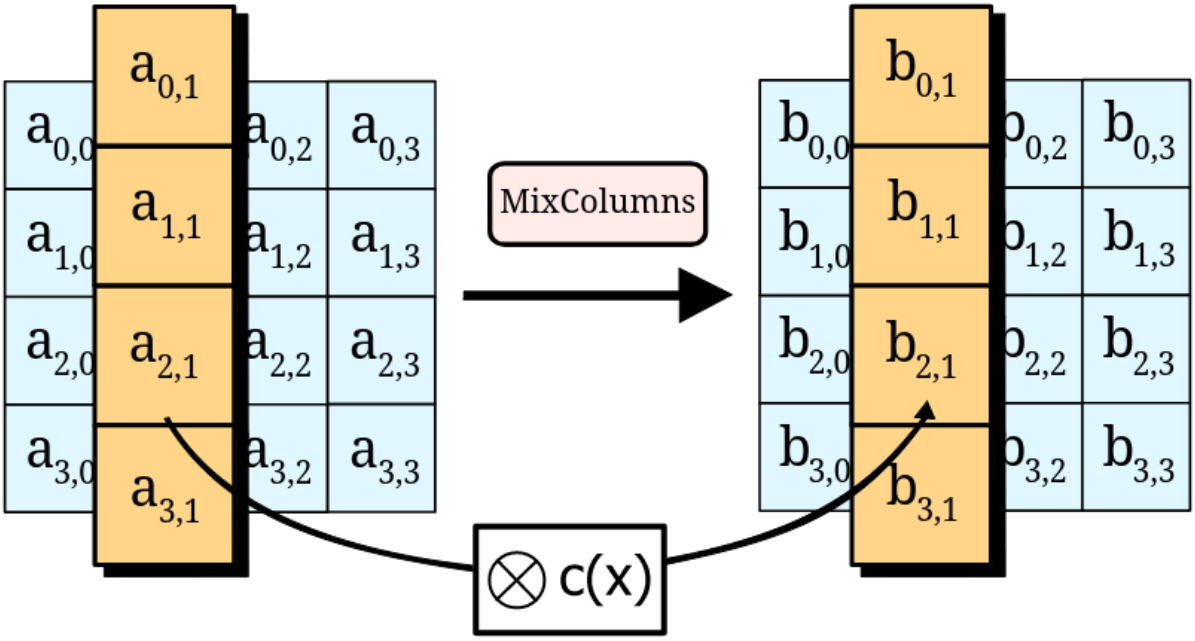
2. Generating the cipher

The cipher is the algorithm used to perform encryption/decryption. To handle data of arbitrary length and nature, several modes of operation for AES are possible. Each mode changes the general algorithm while offering pros and cons, as we will later read.

3. Encrypting/decrypting the data

AES operates on data blocks of 16 bytes represented as 4x4 two-dimensional arrays. The byte content of these matrices is transformed using a mathematical function defined by the cipher (also called block cipher), which gives us a cipher text—the encrypted result.

Without going into the technical details, the mathematical function guarantees the strength of the encryption, which is why AES is considered unbreakable as long as the properties of the function are respected—strong key, correctly implemented cipher, unique nonce, unique initialization vector, etc.



AES Mode Used:

There are several modes available in AES. We have implemented the encryption decryption using the EAX (Encrypt-then-authenticate-then-translate) mode. EAX mode is a mode of operation for cryptographic block ciphers. It is an Authenticated Encryption with Associated Data (AEAD) algorithm designed to simultaneously provide both authentication and privacy of the message (authenticated encryption) with a two-pass scheme, one pass for achieving privacy and one for authenticity for each block.

Problem Implementation:

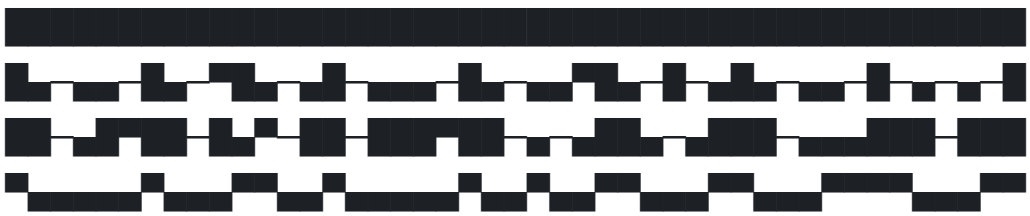
In Encryption Module, we will ask the user to enter the directory which needs to be encrypted. Once we get the directory name, using python's Walk method, we traverse through all the sub-directories and finds all the files that need to be encrypted. Upon getting the files, we read the file content and send through the AES module where it encrypts the data which will be overwritten. A JSON file will be generated which will contain the key, tag and nounce value of all files present in the directory.

In Decryption Module, we will feed the JSON file generated from Encryption module. From the JSON file, the program will fetch all the necessary details for directory to be decrypted. The encrypted data is fetched using the paths available in the JSON file. These data are sent through AES module to be decrypted. After this process, the file contents are overwritten with the decrypted data.

Code:

Encrypt

```
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes
import os
import json

key = get_random_bytes(16)


def enc(inp):
    data = inp
    cipher = AES.new(key, AES.MODE_EAX)
    ciphertext, tag = cipher.encrypt_and_digest(data)
    return (ciphertext, tag, cipher.nonce)
data_dict = {}

print(
    """
```



```
    """
```

```
)



print(
    "\n\n\nWelcome To Encrypter. Enter the folder name of the folder to be encrypted in the
below prompt... "
)
print("The folder must be present in root of the application")
folder_name = input("Enter Folder name:: ")

print("\n\nChecking folder existance... \n\n")

if not os.path.exists(folder_name):
    print(
        "\nFolder Doesn't Exist \n\nThe folder must be there in root of this application\n\n\n"
    )

else:
    print("\n\nGG! The Folder exists \n\n")

    for (root, dirs, files) in os.walk(folder_name, topdown=True):
        # print("root=", root)
        # print("dirs = ", dirs)
        # print("files = ", files)

        for file in files:
            path1 = root + "\\" + file
            # print("path = ", path1)

            f = open(path1, "rb")
            inp = f.read()
            f.close()
```

```
            fl = open("temp.bin", "wb")
            add = enc(inp)
            fl.write(add[0])
            fl.close()

            a = bytearray(key)
            key1 = ''.join(chr(x) for x in a)

            a = bytearray(add[1])
            tag1 = ''.join(chr(x) for x in a)

            a = bytearray(add[2])
            nonce1 = ''.join(chr(x) for x in a)

            #print("key1 = ",key1)
            #print("tag1 = ",tag1)
            #print(" nonce1 = ",nonce1)
            os.remove(path1)
            os.rename("temp.bin", path1)
            data_dict[path1] = {
                "key": key1,
                "tag": tag1,
                "nonce": nonce1,
            }

    #print(data_dict)
    print()
# a = bytearray(b'\x00\x00\x00\x00\x07\x80\x00\x03')
# key = ''.join(chr(x) for x in a)
# a = bytearray()
# for i in key:
# a.append(ord(i))
```

```
# byteString = bytes(a)
    with open("key.json", "w") as fp:
        json.dump(data_dict, fp)

    print(
        "Encryption SuccessFull and The key is stored in key.json. \n"
    )

    x=input("Enter any key to exit")
```

Decrypt

```
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes
import json
import os
def decrypt(tup):
    cipher = AES.new(tup[1], AES.MODE_EAX, tup[3])
    data = cipher.decrypt_and_verify(tup[0], tup[2])
    file = open("temp1.bin", "wb")
    file.write(data)
    file.close()

    os.remove(tup[4])
    os.rename("temp1.bin", tup[4])
    print(tup[4], "is succesfully decrypted")



print(
    """
```

```
    """
```

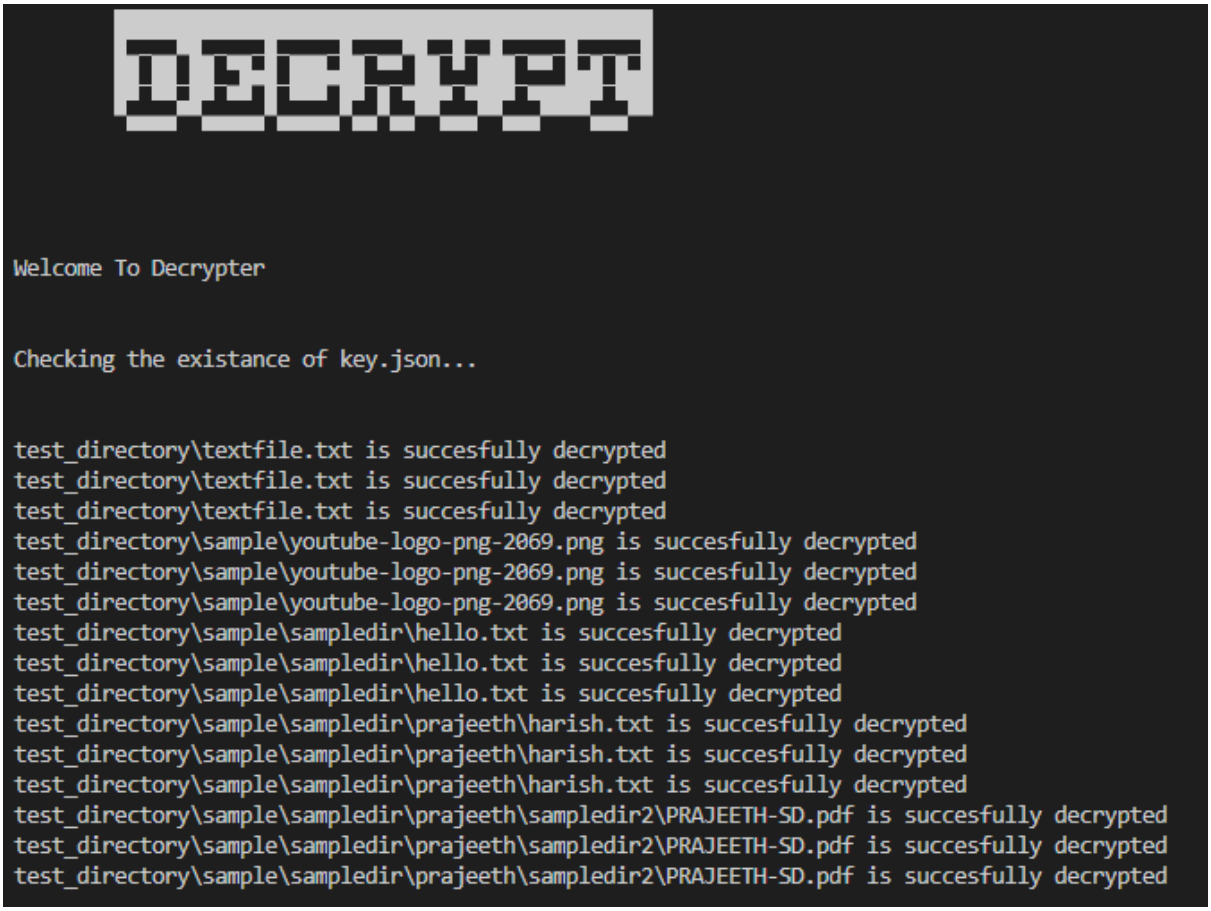
```
)


print("\n\n\nWelcome To Decrypter")

print("\n\nChecking the existance of key.json... \n\n")

try:
    with open("key.json", "r") as json_file:
        key_dict = json.load(json_file)



    #print("keys found!\n\n")
    for i in key_dict:
        file = open(i, "rb")
        data = file.read()
        file.close()
        for vals in key_dict[i]:

# a = bytearray()
# for i in key:
# a.append(ord(i))

# byteString = bytes(a)
            l=[]
            #print(vals, key_dict[i][vals])
            for j in key_dict[i].values():
                a = bytearray()
                for k in j:
                    a.append(ord(k))
                byteString = bytes(a)
                l.append(byteString)
```

```
            #print(" l= ",l)
            # = list(key_dict[i].values())
            tup = (data, l[0], l[1], l[2], i)
            decrypt(tup)

    print("\n\nDecryption Successful!!")

except FileNotFoundError:
    print(
        "FATAL ERROR: key.json not found. The key.json must be in same level as the application\n\n"
    )

x=input("Enter any key to exit= ")
```

Output:

Encyption

Decryption



Result:

Hence, a program to implement cryptographic files system have been implemented successfully.

References:

1. https://www.scs.stanford.edu/nyu/02fa/lab/cfs.pdf
2. https://csrc.nist.gov/csrc/media/projects/cryptographic-standards-and-guidelines/documents/aes-development/rijndael-ammended.pdf
3. https://onboardbase.com/blog/aes-encryption-decryption