

INFORMATION AND SYSTEM SECURITY

A Final Report of

SECURING FILE STORAGE ON CLOUD USING CRYPTOGRAPHY

BY

M.PREETHI – 16MIS1013

ABSTRACT

The aim is to securely store information into the cloud, by splitting data into several chunks and storing parts of it on cloud in a manner that preserves data confidentiality, integrity and ensures availability. The rapidly increased use of cloud computing in the many organization and IT industries provides new software with low cost. It is beneficial in terms of low cost and accessibility of data. It gives lot of benefits with low cost and of data accessibility through Internet. Ensuring the security of cloud computing is a major factor in the cloud computing environment, as users often store sensitive information with cloud storage providers, but these providers may be untrusted. So sharing data in secure manner while preserving data from an untrusted cloud is still a challenging issue. The approach ensures the security and privacy of client sensitive information by storing data across single cloud, using AES algorithm.

INTRODUCTION

Cryptography is the protecting technique of data from the unauthorized party by converting into the non-readable form. The main purpose of cryptography is maintaining the security of the data from third party. For data security and privacy protection issues, the fundamental challenge of separation of sensitive data and access control is fulfilled. Cryptography technique translates original data into unreadable form. There are following two types of algorithms such as:

- (i) symmetric key based algorithm, sometimes known as conventional key algorithm and
- (ii) asymmetric key based algorithm, also known as public-key algorithm. Symmetric algorithm can be further divided into two types. This technique uses keys for translate data into unreadable form. So only authorized person can access data from cloud server. Cipher text data is visible for all people.

DESCRIPTION

AES (ADVANCED ENCRYPTION STANDARD)

The more popular and widely adopted symmetric encryption algorithm likely to be encountered nowadays is the Advanced Encryption Standard (AES). It is found at least six time faster than triple DES.

A replacement for DES was needed as its key size was too small. With increasing computing power, it was considered vulnerable against exhaustive key search attack. Triple DES was designed to overcome this drawback but it was found slow. Aes is the block chain symmetric algorithm.

A block cipher is a method of encrypting text (to produce ciphertext) in which a cryptographic key and algorithm are applied to a block of data (for example, 64 contiguous bits) at once as a group rather than to one bit at a time. The main alternative method, used

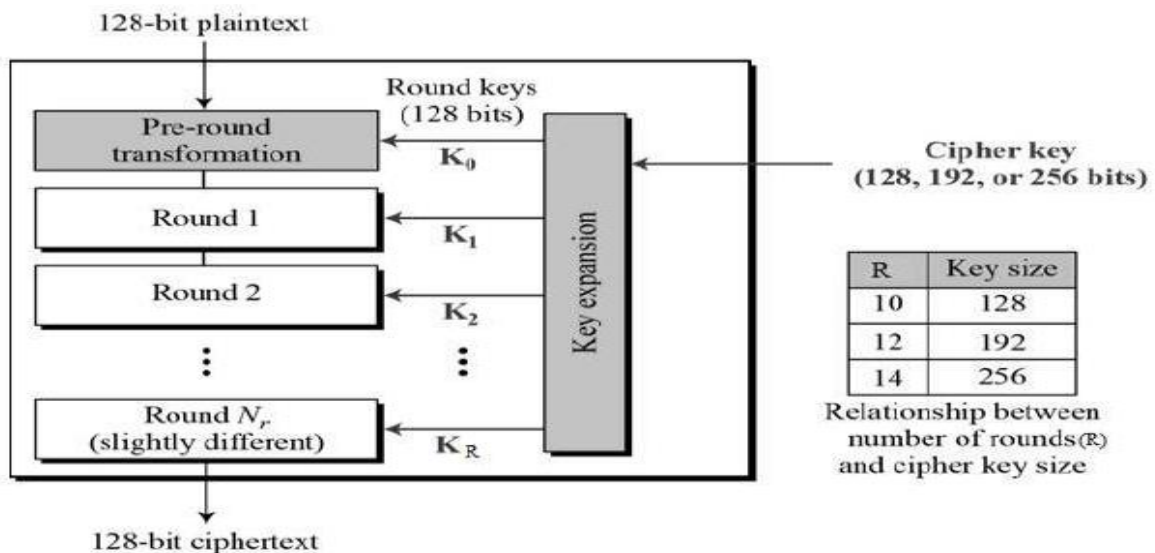
much less frequently, is called the stream cipher. So that identical blocks of text do not get encrypted the same way in a message (which might make it easier to decipher the ciphertext), it is common to apply the ciphertext from the previous encrypted block to the next block in a sequence. So that identical messages encrypted on the same day do not produce identical ciphertext, an initialization vector derived from a random number generator is combined with the text in the first block and the key. This ensures that all subsequent blocks result in ciphertext that doesn't match that of the first encrypting.

AES FEATURES

The selection process for this new symmetric key algorithm was fully open to public scrutiny and comment; this ensured a thorough, transparent analysis of the designs submitted. NIST specified the new advanced encryption standard algorithm must be a block cipher capable of handling 128 bit blocks, using keys sized at 128, 192, and 256 bits; other criteria for being chosen as the next advanced encryption standard algorithm included:

- **Security:** Competing algorithms were to be judged on their ability to resist attack, as compared to other submitted ciphers, though security strength was to be considered the most important factor in the competition.
- **Cost:** Intended to be released under a global, nonexclusive and royalty-free basis, the candidate algorithms were to be evaluated on computational and memory efficiency.
- **Implementation:** Algorithm and implementation characteristics to be evaluated included the flexibility of the algorithm; suitability of the algorithm to be implemented in hardware or software; and overall, relative simplicity of implementation.

AES STRUCTURE

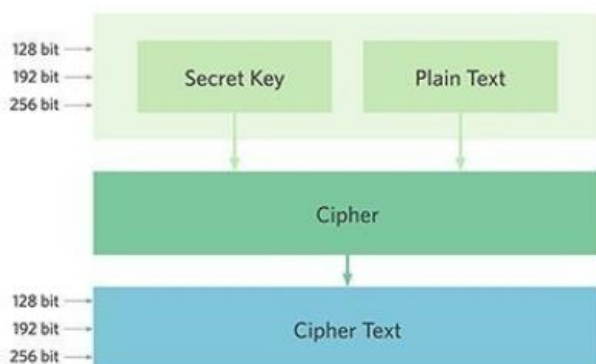


WORKING

ENCRYPTION

AES comprises three block ciphers: AES-128, AES-192 and AES-256. Each cipher encrypts and decrypts data in blocks of 128 bits using cryptographic keys of 128-, 192- and 256-bits, respectively. The Rijndael cipher was designed to accept additional block sizes and key lengths, but for AES, those functions were not adopted. Symmetric (also known as secret-key) ciphers use the same key for encrypting and decrypting, so the sender and the receiver must both know -- and use -- the same secret key. All key lengths are deemed sufficient to protect classified information up to the "Secret" level with "Top Secret" information requiring either 192- or 256-bit key lengths. There are 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys -- a round consists of several processing steps that include substitution, transposition and mixing of the input plaintext and transform it into the final output of ciphertext.

AES Design



The AES encryption algorithm defines a number of transformations that are to be performed on data stored in an array. The first step of the cipher is to put the data into an array; after which the cipher transformations are repeated over a number of encryption rounds. The number of rounds is determined by the key length, with 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. The first transformation in the AES encryption cipher is substitution of data using a substitution table; the second transformation shifts data rows, the third mixes columns. The last transformation is a simple exclusive or (XOR) operation performed on each column using a different part of the encryption key -- longer keys need more rounds to complete.

DECRYPTION

The process of decryption of an AES ciphertext is similar to the encryption process in the reverse order. Each round consists of the four processes conducted in the reverse order –

- Add round key
- Mix columns
- Shift rows
- Byte substitution

Since sub-processes in each round are in reverse manner, unlike for a Feistel Cipher, the encryption and decryption algorithm needs to be separately implemented, although they are very closely related.

ADVANTAGES

- The stored image file is completely secured, as the file is being encrypted not by just using one but three encryption algorithm which are AES, DES and RC6.
- The key is also safe as it embeds the key in image using LSB.
- The system is very secure and robust in nature.
- Data is kept secured on cloud server which avoids unauthorized access.

TOOLS

- Anaconda for python
- AWS for cloud service

PACKAGES

PYCRYPTO: To install this “PIP INSTALL PYCRYPTO” run it in the terminal. This is a collection of both secure hash functions (such as SHA256 and RIPEMD160), and various encryption algorithms (AES, DES, RSA, ElGamal, etc.). The package is structured to make adding new modules easy.

PYAESCRYPT: To install this” PIP INSTALL PYAESCRYPT” run it in the terminal. PyAesCrypt is a Python 3 file-encryption module and script that uses AES256-CBC to encrypt/decrypt files and binary streams .pyAesCrypt is compatible with the AES Crypt file format

BOTO3: To install this “PIP INSTALL BOTO3”Boto3 is the Amazon Web Services (AWS) Software Development Kit (SDK) for Python, which allows Python developers to write software that makes use of services like Amazon S3 and Amazon EC2

CODE

```
# importing the required packages for cryptography (enc and dec)
from Crypto import Random
from Crypto.Cipher import AES
# importing the os package for specifying the file path and directory
import os
import os.path
from os import listdir
from os.path import isfile, join
import time
# importing boto3 for connecting this jupyter to AWS cloud
import boto3
from botocore.client import Config

# creating a class for enc

class Encryptor:
    def __init__(self, key):
        self.key = key

# creating a definition for padding, encryption and encrypting file

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

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)
#automatically uploading the enc file into the AWS cloud

ACCESS_KEY_ID = 'AKIAZ5NOIHR33NTI6MPS '
```

```

ACCESS_SECRET_KEY = 'lHu6Kye0NUKInJh8rqtyxbUUsQ4+gG/bw86zVBkQ'
BUCKET_NAME = 'preethireddy'

data = open(file_name+'.enc', 'rb')

s3 = boto3.resource(
    's3',
    aws_access_key_id=ACCESS_KEY_ID,
    aws_secret_access_key=ACCESS_SECRET_KEY,
    config=Config(signature_version='s3v4')
)
s3.Bucket(BUCKET_NAME).put_object(Key=file_name+'.enc', Body=data)
# creating def for dec and dec the file

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)

# automatically uploading the dec file in the AWS cloud
ACCESS_KEY_ID = 'AKIAZ5NOIHR33NTI6MPS '
ACCESS_SECRET_KEY = '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'
BUCKET_NAME = 'preethireddy'

data = open(file_name[:-4], 'rb')

s3 = boto3.resource(
    's3',
    aws_access_key_id=ACCESS_KEY_ID,
    aws_secret_access_key=ACCESS_SECRET_KEY,
    config=Config(signature_version='s3v4')
)
s3.Bucket(BUCKET_NAME).put_object(Key=file_name[:-4], Body=data)

#assigning the key

key = b'[EX\xc8\xd5\xbf{\xa2$\x05(\xd5\x18\xbf\xc0\x85)\x10nc\x94\x02)j\xdf\xcb\xc4\x94\x9d(\x9e'
enc = Encryptor(key)
clear = lambda: os.system('cls')

```

have to enter the password for the crypto process and that pwd will stored in the data.txt and it will be always in the enc format

```
if os.path.isfile('data.txt.enc'):
    while True:
        password = str(input("Enter password: "))
        enc.decrypt_file("data.txt.enc")
        p = ""
        with open("data.txt", "r") as f:
            p = f.readlines()
        if p[0] == password:
            enc.encrypt_file("data.txt")
            break
# user have to enter the choice for the operation
while True:
    clear()
    choice = int(input(
        "1. Press '1' to encrypt file.\n2. Press '2' to decrypt file.\n3. Press '3' to exit.\n"))
    clear()
    if choice == 1:
        enc.encrypt_file(str(input("Enter name of file to encrypt: ")))
    elif choice == 2:
        enc.decrypt_file(str(input("Enter name of file to decrypt: ")))
    elif choice == 3:
        exit()
    else:
        print("Please select a valid option!")

else:
    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")
    time.sleep(15)
```


SCREENSHOTS

```
In [*]: # importing the required packages for cryptograph(enc and dec)
from Crypto import Random
from Crypto.Cipher import AES
# importing the os package for specifying the file path and directory
import os
import os.path
from os import listdir
from os.path import isfile, join
import time
# importing boto3 for connecting this jupyter to AWS cloud
import boto3
from botocore.client import Config

# creating a class for enc

class Encryptor:
    def __init__(self, key):
        self.key = key

# creating a definition for padding, encryption and encrypting file

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

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

```
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)

# automatically uploading the dec file in the AWS cloud
ACCESS_KEY_ID = 'AKIAZ5NOIHR33NTI6MPS '
ACCESS_SECRET_KEY = 'lh6kYe0NUKInJh8rtqyxbUUsQ4+gG/bw86zvBKQ'
BUCKET_NAME = 'preethireddy'

data = open(file_name[:-4], 'rb')

s3 = boto3.resource(
    's3',
    aws_access_key_id=ACCESS_KEY_ID,
    aws_secret_access_key=ACCESS_SECRET_KEY,
    config=Config(signature_version='s3v4')
)
s3.Bucket(BUCKET_NAME).put_object(Key=file_name[:-4], Body=data)


#assing the key
key = b'[\x08\xd5\xbfI\xa25\x05(\xd5\x18\xbf\xc0\x85)\x10nc\x94\x02]j\xdf\xcb\xc4\x94\x9d(\x9e'
enc = Encryptor(key)
clear = lambda: os.system('cls')

# have to enter the password for the crypto process and that pwd will stored in the data.txt and it will be always in the enc
```

```
choice = int(input(
    "\n1. Press '1' to encrypt file.\n2. Press '2' to decrypt file.\n3. Press '3' to exit.\n"))
clear()
if choice == 1:
    enc.encrypt_file(str(input("Enter name of file to encrypt: ")))
elif choice == 2:
    enc.decrypt_file(str(input("Enter name of file to decrypt: ")))
elif choice == 3:
    exit()
else:
    print("Please select a valid option!")









else:
    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")
    time.sleep(15)
```

STEP-1 when we run this code for the first time we have to enter the pwd and confirm the pwd again. Then the pwd will be written in the data.txt.enc i.e. this will file be always be in encrypted format only .it will be stored in the directory where our program is running. program will ask us to restart the kernel


jupyter

Quit

Logout

| | | | |
|--------------------------|---|-----------------------|---------|
| <input type="checkbox"/> |  Untitled4.ipynb | a month ago | 4.35 kB |
| <input type="checkbox"/> |  Untitled5.ipynb | 24 days ago | 38.5 kB |
| <input type="checkbox"/> |  Untitled6.ipynb | Running 4 minutes ago | 6.61 kB |
| <input type="checkbox"/> |  Untitled7.ipynb | 5 months ago | 141 kB |
| <input type="checkbox"/> |  vader.ipynb | 7 months ago | 6.85 kB |
| <input type="checkbox"/> |  2019-062.db | 4 months ago | 16.4 kB |
| <input type="checkbox"/> |  checkpoint | 4 months ago | 125 B |
| <input type="checkbox"/> |  data.txt.enc | 4 minutes ago | 32 B |

```
time.sleep(15)
```

Setting up stuff. Enter a password that will be used for decryption: chamo1629

Confirm password: chamo1629

Please restart the program to complete the setup

STEP-2 Again we have to run the code and it will ask us to enter the pwd again for the process. Then it will display the three options and user should select it. We have to the txt or img or audio file in same directory.

```
Enter password: chamo1629
1. Press '1' to encrypt file.
2. Press '2' to decrypt file.
3. Press '3' to exit.
```

STEP-3 first we are going encrypt the file so enter 1 for enc process. Enter the file to be encrypted. then it will overwrite the older file and save the enc file as”filename.enc”

```
Enter password: chamo1629
1. Press '1' to encrypt file.
2. Press '2' to decrypt file.
3. Press '3' to exit.
1
Enter name of file to encrypt: dog.txt
```

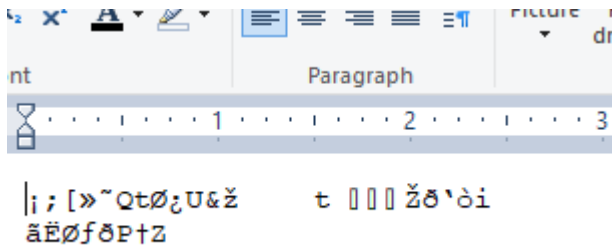
| | | |
|--------------------------|--|---------------|
| <input type="checkbox"/> |  2019-062.db | 4 months ago |
| <input type="checkbox"/> |  checkpoint | 4 months ago |
| <input type="checkbox"/> |  data.txt.enc | 9 minutes ago |
| <input type="checkbox"/> |  dog.jpeg | 6 minutes ago |
| <input type="checkbox"/> |  dog.txt.enc | 6 minutes ago |

STEP-4 after the enc process both the enc file will be automatically stored into the cloud.by logging into the aws cloud ->select->services->s3->bucketname->preethireddy->there all the enc files are stored.

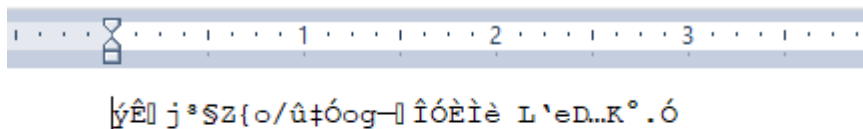
| <div> <div>aws</div> <div>Services</div> <div>Resource Groups</div> <div></div> </div> <div> <div>chandanamohan</div> <div>Global</div> <div>Support</div> </div> | | | | | | | | | | | | | | | | | | | |
|--|----------------------------------|---------|---------------|-------------------------------|---------------|------|---------------|---------------------------------------|----------------------------------|--------|----------|---------------------------------------|----------------------------------|---------|----------|--------------------------------------|----------------------------------|--------|----------|
| <div> <div>Q</div> <div>Type a prefix and press Enter to search. Press ESC to clear.</div> </div> | | | | | | | | | | | | | | | | | | | |
| <div> <div>Upload</div> <div>Create folder</div> <div>Download</div> <div>Actions</div> </div> <div>Asia Pacific (Mumbai)</div> | | | | | | | | | | | | | | | | | | | |
| Viewing 1 to 3 | | | | | | | | | | | | | | | | | | | |
| <table> <tr> <th><input type="checkbox"/> Name</th><th>Last modified</th><th>Size</th><th>Storage class</th></tr> <tr> <td><input type="checkbox"/> data.txt.enc</td><td>Oct 19, 2019 7:10:14 PM GMT+0530</td><td>32.0 B</td><td>Standard</td></tr> <tr> <td><input type="checkbox"/> dog.jpeg.enc</td><td>Oct 19, 2019 7:24:15 PM GMT+0530</td><td>19.5 KB</td><td>Standard</td></tr> <tr> <td><input type="checkbox"/> dog.txt.enc</td><td>Oct 19, 2019 7:13:26 PM GMT+0530</td><td>32.0 B</td><td>Standard</td></tr> </table> | | | | <input type="checkbox"/> Name | Last modified | Size | Storage class | <input type="checkbox"/> data.txt.enc | Oct 19, 2019 7:10:14 PM GMT+0530 | 32.0 B | Standard | <input type="checkbox"/> dog.jpeg.enc | Oct 19, 2019 7:24:15 PM GMT+0530 | 19.5 KB | Standard | <input type="checkbox"/> dog.txt.enc | Oct 19, 2019 7:13:26 PM GMT+0530 | 32.0 B | Standard |
| <input type="checkbox"/> Name | Last modified | Size | Storage class | | | | | | | | | | | | | | | | |
| <input type="checkbox"/> data.txt.enc | Oct 19, 2019 7:10:14 PM GMT+0530 | 32.0 B | Standard | | | | | | | | | | | | | | | | |
| <input type="checkbox"/> dog.jpeg.enc | Oct 19, 2019 7:24:15 PM GMT+0530 | 19.5 KB | Standard | | | | | | | | | | | | | | | | |
| <input type="checkbox"/> dog.txt.enc | Oct 19, 2019 7:13:26 PM GMT+0530 | 32.0 B | Standard | | | | | | | | | | | | | | | | |
| Viewing 1 to 3 | | | | | | | | | | | | | | | | | | | |

STEP-5 The encrypted files can be downloaded but the contents cannot be read or even understand by the humans

Enc file of pwd (data.txt.enc)



Enc file of txt (dog.txt.enc)



Enc file of img (dog.jpeg.enc)

```

eEOl pl=cpAlV, efel # 'l jôo 'm(fct-l Q) ūāē+
Ygā@YA' 'l jEc! ūā+ō' 'ku ,>, l j+J+cVQs l j±Z AcEē
» m @ ,HAl t H,,āEU.l IJkEō, ' Ā-Nē'+pō+l k l mQō<ē' l ō-ōPQn sō
\WUmK-
I+*f l a ā's-BNe l ži zi
l i-Kēō'T BāyEG' tuYl vl l 7 {vi GEFŮ'+4= ...l f'z'p'7 l ōi i o'>ā l -P
R Qey l B l K l Cl _mEōōQ l i ž...l TāPb
l E' (ā l iYtWōā», tn\wSs< e+*ā'ēēōō-ōE! nSōMl Dē āND
Wl 8 (E±āE±/E"HKPB"pēi±ā l i* . āā"āōōp l 8ā l i Yēw QīEōā ;Wl 8 5
±ēōō (āy; s) ōl EIS l Z' D+IEB l ūM7+IōE l ūi>" l iō
l cpm' 9 l ,.ē j . iōōāāōōōYēēāp' Sm l ōl sēō l ŌL l u>"< l jōtE' +/ l 7* I
āōā , nēKveē' ž; ( l "Yā l l "ž ā l 4-l q Ū l "98; l d l l G/. HŌōā×l N l āE uā j l ē
Ydā l 2'...c' cē >-s; l ēd Ā < tōT l ēēYā l >... l =sō
ō
OKý , -qōF
IUBāX=cl -l < \ ēYē KO ūb l +ō ± l ā' ±+l-
Ūōōē-ā; āēōōē
, WqŮ' s
ā āhō-Kōā Mā l D l ex
...ē;...cō>Hā
l xqB' pē l ē l ūēIŪōōON. p l -+*žō l M-Ge' l dōr
d l ' yō
} * ; ŪēēēōōZ, l ūēōō l ū+BēV {ēkēTCL l q l ū4āK>t l q l ūōīfB
pō l l " āōōō×āUā jō " ) ūM ūs 4<Y C) ( u l *ēā smēā RE+ēēYi - l *
i+s"
l , l
t ē2 ; ' d j Iēp' ē l ū ( J , + iōōēāy; Yōōý
m ē l l =ōāōŮ' *2h) āē, cō l i; ēŪTā l ' E ( ēāī±4ō l _mōē×iō l - l zōōp B+ū"
pō āxēē-ō; ā-ā āāōl sōō l nāō l ā "ōl WōōōōLūCō l l p l l ) Yēōō-P l <
y l l Tēā l l n± k l \-IāRōē; TēpōXā l l YēD' c-I*+āēēōē=ēyā l l -Dōō'nēēē'
vē, l ūōēēāēōē; nā-l Hō l l pū
ēāŪē' ±-ō l d7 (yŪf l 9 l ūō ē, ā
āW
qŪōō l \. AC ēPōēō
ēYē' 7 ē' ēā f' "D l ū< *ā l ē l , ū, * ā l *ēōā ā ōō -ēēā>ēc' q l
ēōēy' c...ēēiū'

```

STEP-6 then user should select the option 2 to dec the enc file

Enter name of file to decrypt: dog.txt.enc

1. Press '1' to encrypt file.
2. Press '2' to decrypt file.
3. Press '3' to exit.

2

Enter name of file to decrypt: dog.jpeg.enc

1. Press '1' to encrypt file.
2. Press '2' to decrypt file.
3. Press '3' to exit.


2

Enter name of file to decrypt: data.txt.enc

In our directory all the enc files will be dec to is original format.

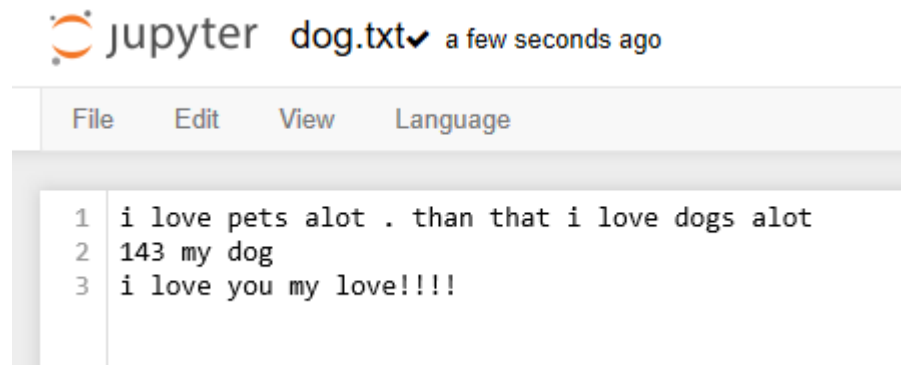
| | |
|---|---------------|
| <input type="checkbox"/>  checkpoint | 4 months ago |
| <input type="checkbox"/>  data.txt | 2 minutes ago |
| <input type="checkbox"/>  dog.jpeg | 2 minutes ago |
| <input type="checkbox"/>  dog.txt | 2 minutes ago |

1) data.txt (pwd)


jupyter
data.txt ✓ 3 minutes ago

| File | Edit | View | Language |
|------|-----------|------|----------|
| 1 | chamo1629 | | |

2)dog.txt



3)dog.jpeg



STEP-7 The dec files will be automatically stored in the cloud

| aws | | | | Services ▾ | Resource Groups ▾ | ★ | chandanamohan ▾ | Global ▾ |
|--|--|--|--|----------------------------------|-------------------|-----------------|-----------------|----------|
| Q Type a prefix and press Enter to search. Press ESC to clear. | | | | | | | | |
| Upload Create folder Download Actions ▾ | | | | Asia Pacific (Mumbai) ; | | | | |
| <input type="checkbox"/> Name ▾ | | | | Last modified ▾ | Size ▾ | Storage class ▾ | | |
| <input type="checkbox"/> data.txt | | | | Oct 19, 2019 7:37:36 PM GMT+0530 | 9.0 B | Standard | | |
| <input type="checkbox"/> data.txt.enc | | | | Oct 19, 2019 7:10:14 PM GMT+0530 | 32.0 B | Standard | | |
| <input type="checkbox"/> dog.jpeg | | | | Oct 19, 2019 7:37:08 PM GMT+0530 | 19.5 KB | Standard | | |
| <input type="checkbox"/> dog.jpeg.enc | | | | Oct 19, 2019 7:24:15 PM GMT+0530 | 19.5 KB | Standard | | |
| <input type="checkbox"/> dog.txt | | | | Oct 19, 2019 7:36:52 PM GMT+0530 | 0 B | Standard | | |
| <input type="checkbox"/> dog.txt.enc | | | | Oct 19, 2019 7:13:26 PM GMT+0530 | 32.0 B | Standard | | |

VIDEO

The output of the file and the procedure for connecting the python to AWS cloud services are recorded.

REASON

The reason we chose the AES algorithm

- When confidentiality of the data comes the only algorithm is AES
- There are many algorithms for enc and dec but AES works more efficiently and effectively.
- It is an algorithm to overcome the DES
- It can handle the brute force attack also (i.e.)even the secured file can also be easily hacked but in AES it cannot
- Fifteen competing symmetric key algorithm designs were subjected to preliminary analysis by the world cryptographic community, including the National Security Agency (NSA). In August 1999, NIST selected five algorithms for more extensive analysis. These were: **MARS, RC6, Serpent, Twofish**
- In June 2003, the U.S. government announced that AES could be used to protect classified information, and it soon became the default encryption algorithm for protecting classified information as well as the first publicly accessible and open cipher approved by the NSA for top-secret information. The NSA chose AES as one of the cryptographic algorithms to be used by its Information Assurance Directorate to protect national security systems.
- Its successful use by the U.S. government led to widespread use in the private sector, leading AES to become the most popular algorithm used in symmetric key cryptography. The transparent selection process helped create a high level of confidence in AES among security and cryptography experts.
- AES is more secure than its predecessors -- DES and 3DES -- as the algorithm is stronger and uses longer key lengths.
- It also enables faster encryption than DES and 3DES, making it ideal for software applications, firmware and hardware that require either low latency or high throughput, such as firewalls and routers. It is used in many protocols such as Secure Sockets Layer (SSL)/Transport Layer Security (TLS) and can be found in most modern applications and devices that need encryption functionality.

The reason for choosing AWS

The only cloud service which is directly connecting the python with cloud and it is much user friendly.

The reason for connecting our python with cloud is the enc and dec file will be stored only in the machine which we work. If the machine gets problem and no backup then sender or receiver cannot fetch the file again. Even sender and receiver can use this file anywhere and anytime across the world without using the same machine by logging into the AWS cloud

CONCLUSION

The main goal is to securely store and access data in cloud that is not controlled by the owner of the data. Exploit the technique AES encryption to protect data files in the cloud. Two part of the cloud server improved the performance during storage and accessing of data. The ECC Encryption algorithm used for encryption is another advantage to improve the performance during encryption and decryption process.