Final Project Report

Encrypted File System Architecture

Team members:
Soham Faldu 19BCI0024
Vidhi Moteria 19BCE0525
Rashi Maheshwari 19BDS0006
Rohan Allen 18BCI0247
Phanider Ekdukalla 18BCI0253

Submitted to:
Dr. Madhu Vishwanatham

B.Tech.

in

Computer Science and Engineering

School of Computer Science & Engineering

®

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Index

Abstract

Increasing thefts of sensitive data owned by individuals and organizations call for an integrated solution to the problem of storage security. Most existing systems are designed for personal use and do not address the unique demands of enterprise environments. An enterprise-class encrypting file system must take a cohesive approach towards solving the issues associated with data security in organizations.

Developing a high security and good speed solution for the problem. Using some of the fastest algorithms like twofish for encryption.

Keywords: Twofish, file system, OTP, SMTP, Rail fence.

Introduction

1. Theoretical background

Twofish Algorithm:
Twofish is a symmetric key block cipher with a block size of 128 bits and key sizes up to 256 bits. It was one of the five finalists of the Advanced Encryption Standard contest, but it was not selected for standardization.

Advantages:

- No weak keys
- Efficiency
- Flexible design
- Fastest algorithm

File system architecture:

The File System Architecture Specifies that how the Files will be stored into the Computer system means how the Files will be Stored into the System. Without a file system, data placed in a storage medium would be one large body of data with no way to tell where one piece of data stops and the next begins.

2. Motivation

Increasing thefts of sensitive data owned by individuals and organizations call for an integrated solution to the problem of storage security. Most existing systems are designed for personal use and do not address the unique demands of enterprise environments. An enterprise-class encrypting file system must take a cohesive approach towards solving the issues associated with data security in organizations.

3. Aim of the proposed Work

Developing an encrypted file system that is both secure and fast. Looking over the issue of time complexity with cryptanalysis. The architecture would be for organizational use for large companies. We are going to use some of the fastest encryption algorithm for securing the architecture. Encrypting files of organization so that if the opponent party tries to hack and take the file, he/she won't be able to encrypt it.

4. Objective(s) of the proposed work

- Make a highly secured file system architecture.
- Securing the vulnerabilities present in the current systems.
- Making it with good speed so convenient for the enterprise systems.
- Protecting the file using OTP and SMTP.

Literature Survey

1. Survey of existing models

[1] Research and Implementation of an Encrypted File System Used to NAS

Information security has become an inexorably significant factor in routine work with the advancement of technology, and clients want to acquire high security. In this article a cryptographic record framework called NAS_CFS utilized for NAS is planned. NAS_CFS has a few qualities as beneath: adding encryption capacity to record framework layer permits clients to encode information straightforwardly, acquiring high security by key the executives dependent on meeting ID and client ID, connection and breaks systems, and NAS_CFS is an in-part record arrangement of high execution utilizing stackable system. Stackable Vnode interface gives a novel programming mode for document framework advancement, working on the record framework improvement measure. NAS_CFS is executed by particular of stackable Vnode interface, so having such points of interest as basic and adaptable working, low execution overhead, solid transportability, etc. Key administration dependent on meeting ID and client ID, connection and breaks component can improve the security of information. In a word, NAS_CFS can make NAS framework more secure, dependable and easy.

[2] Efficient methodology for implementation of Encrypted File System in User Space

The Encrypted File System (EFS) pushes encryption administrations into the document framework itself. EFS underpins secure capacity at the framework level through a standard UNIX document framework interface to encoded records. Client can connect a cryptographic key with the catalogs they wish to ensure. Records in these registries (just as their pathname parts) are straightforwardly scrambled and decoded with the determined key minus any additional client mediation; clear text is never put away on a circle or shipped off a far off record worker. EFS can utilize any accessible document framework for its fundamental stockpiling without adjustments, including distant record workers, for example, NFS. Framework the board capacities, for example, document reinforcement, work in a typical way also, without information on the key. Execution is an significant factor to clients since

encryption can be time devouring. This paper portrays the plan and execution of EFS in client space utilizing quicker cryptographic calculations on UNIX Operating framework.

[3] Secure Outsourcing and Sharing of Cloud Data Using a User-Side Encrypted File System

Author: OSAMA AHMED KHASHAN

Abstract: In this paper, the author has pointed out the threats to confidentiality of remote and untrustworthy cloud storage systems. Cloud computing is an emerging paradigm that deals with massive data which suffers from security, efficiency and usability issues. The author has introduced a hybrid encryption method by combining symmetric and asymmetric methodology. Existing traditional encryption systems impose a heavy burden of managing files and encryption operations on data owners. OutFS is designed to preserve the integrity of outsourced file data and file system data structure. The security analysis shows that OutFS is extremely secure and robust against many attacks such as brute-force, eavesdropping, man-in-middle, and offline-dictionary attacks.

[4] An Overview of Cryptanalysis Research for the Advanced Encryption Standard

Authors: Alan Kaminsky, Michael Kurdziel, Stanisław Radziszowski

Abstract: This paper gives us detailed information on the existing techniques like Linear cryptanalysis, Differential cryptanalysis, cube attacks, boomerang attacks, algebraic attacks, etc. Also it gives us information on some new hybrid algorithms for decrypting AES which hasn't been breached yet. This paper shows the various cryptanalytic algorithms used on the military/ Government threat model. It shows that the cryptanalysis against the most secured algorithm is making progress. It also gives us information about some of the many famous cryptanalytic algorithm used against our systems. It shows that at this pace AES won't have a life time and we will find some cryptanalytic algorithm against it. Hence, we need to come with a secure and faster encryption algorithm for classified purposes.

[5] Encryption and Decryption using Password Based Encryption, MD5, and DES

Authors: Hanna Willa Dhany, Fahmi Izhari, Hasanul Fahmi, Tulus, Sutarman

Abstract: This paper gives brief description on Password-Based Encryption, which is used in the application because usually the attacker repeatedly tries to guess undetected keywords and is beyond the original sender / recipient control, if the keyword is used to log in to the server, it can detect many possibilities that are not properly done and in the worst case is to shut down the server to prevent more effort, if a tapper takes encrypted files that we use. The process that will be discussed in this paper include two basic cryptography process that are Encryption and Decryption and uses the same key. The key is also called Secret Key, Shared Key or Symmetric Key. This paper also shows that Password Based Encryption with Message Digest (MD5) and Data Encryption Standard (DES) is a cryptographic method. Algorithms used in this is combination both hashing and standard encryption methods. MD5 was developed by Ronald Rivest where the MD5 takes messages of any length and generates a 128-bit message digest, and with the use of DES working in plaintext it is useful to return the same size ciphertext.

[6] SECURED FILE MANAGEMENT OVER INTERNET

Authors: Prof. Balasaheb B. Gite, Shailesh Navghare, Abhishek Gupta, Siddharth Jain

Abstract: This paper depicts the present scenario of file system, and the need of secured and reliable storage and transfer of information. Getting the right information to the right person at the right time is the key strategy for secured file transfer. File transfer must provide end-to-end visibility, security and compliance management. A secure and managed file transfer approach can help the user to meet the challenge of safely and reliably exchanging electronic information. The authors here would like to present a solution so that users can communicate and exchange the important files in a secured way. The abstract presents Secured File Management and Sharing System over the internet developed using the J2EE technologies.

[7] Design and implementation of encrypted and decrypted file system based on USBKey and hardware code

Authors: Kehe Wu, Yakun Zhang, Wenchao Cui, Ting Jiang

There are a lot of security risks in transferring files over a network. To ensure that important information is not stolen and destroyed, the most practical method is file encryption. This

paper describes design and implementation of an encrypted and decrypted file system based on USBKey and hardware key. The structure of the system includes two aspects: USBKey management and client application. The former includes the addition, deletion and modification of certificates, and the latter includes file choosing, encryption and decryption. This system uses a wizard interface. Insert the USBKey and input the PIN code correctly, and then you will enter the system. You can select a file and a certificate, click on the encryption button to complete the encryption process; or choose the file you want to decrypt, click on the decryption button to complete the decryption process. This system is less dependent on the kernel and is compatible with all versions of Windows. Because of the structural design and programming, it can provide support for a variety of encryption algorithms. This system uses hardware to encrypt, it has higher performance compared with other systems which use soft-algorithm. Encryption speed has been significantly improved. Only users who have legal USBKey and enter correct PIN code can log in it. In addition, the files are stored in the disk as cipher text; only legitimate users can decrypt and view them. The system has been proved is flexible and efficient. It ensures the security and privacy of important documents.

[8] TransCrypt: An Enterprise Encrypting File System over NFS

Authors: Abhay Khoje, Salih K A, Rajat Moona

Centralized storage devices are used in many organizations ans institutions. Securing confidential data against thefts which eventually impose risks of losing important personal and organizational data, is of utmost importance. Such attacks include the attacker getting access to files on shared storage, modifying the contents of those files etc. Several solutions have been devised to address this problem. One of them is TransCrypt. It is a secure, usable, transparent, efficient enterprise-class encrypting file system for Linux. However, it is not designed to work over network. The TransCrypt design is also vulnerable to many other attacks while accessing files over network from a workstation like masquerading attack, man-in-the-middle attack, replay attack. This paper gives an approach to solve the attacks when users access files over network from a workstation using NFS.

[9] A study of HSM based key protection in encryption file system

Authors: Wenqian Yu, Weigang Li, Junyuan Wang,

Brings potential security weakness, for example, the virus boot assault which can take the document encryption key and in the end the encoded record can be decoded by the taken key. A HSM based key protection solution is acquainted with improving the security of the encryption record framework, and a model is actualized dependent on Linux Ext4 document. Document encryption key is ensured in memory safely and in this way the security of the encryption record framework is improved. HSM doesn't have to oversee and store the distinctive FEKs during runtime. Just one root key is needed to be ensured inside HSM. Simple to incorporate with existing encryption document framework. Broke down the security impediment of existing encryption record frameworks for key protection. presented a HSM based key assurance answer for encryption document framework.

[10] Analysis and Implementation of NTFS File System Based on Computer Forensics

Authors: Zhang Kai, Cheng En, Gao Qinquan.

NTFS, which re-establishes and deals with the important data and information, is a typical document framework in Windows Operating System. Tapping and breaking down the valuable information of the NTFS record framework has become a significant method for current PC measurable. Through nitty gritty investigation and examination on the capacity standards of the NTFS document framework, the article situated strategy is advanced to plan NTFS record parsing framework. This framework parses the paired information put away in circle, accomplishing the aggregate examination of both the ordinary records and the erased documents. At that point, all the information recovered can be re-established into the type of a user friendly UI which can give a dependable information source to the computer forensics. Through examination and study on the Memory Principles of the NTFS record framework, this paper advances the article arranged plan to plan NTFS document parsing framework, parse the most derived binary data that was saved to the disk, accomplishing the total analysis of the normal records and the erased documents. At that point a user-friendly UI is utilized to show all this information of tree design to the clients. It gives a solid information source and an incredible asset for the PC crime scene investigation.

2. Summary/Gaps identified in the Survey

After reading all the research paper we found out that the most vulnerable part in an enterprise filesystem is the head access. If the attacker gets access to the head part of the system, then he can easily find his way down to any person in the whole company. The second vulnerability we found was that the system proposed are really slow and so we intend to make them faster. So that large documents in the system can take less time to decrypt and the time is saved. We also saw that if we encrypt a file using one encryption then it is easier to break. So, we intend to add some traditional algorithms for making it more secure.

Overview of the Proposed System

1. Introduction and Related Concepts

Twofish Algorithm:
Twofish is a symmetric key block cipher with a block size of 128 bits and key sizes up to 256 bits. It was one of the five finalists of the Advanced Encryption Standard contest, but it was not selected for standardization.

Advantages:

- No weak keys
- Efficiency
- Flexible design
- Fastest algorithm

File system architecture:

The File System Architecture Specifies that how the Files will be stored into the Computer system means how the Files will be Stored into the System. Without a file system, data placed in a storage medium would be one large body of data with no way to tell where one piece of data stops and the next begins.

OS Module:

The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system-dependent functionality. The *os* and *os.path* modules include many functions to interact with the file system.

Pickle Module:

The Python pickle module is another way to serialize and deserialize objects in Python. It differs from the json module in that it serializes objects in a binary format, which means the result is not human readable.

Rail fence:

The rail fence cipher (also called a zigzag cipher) is a form of classical transposition cipher. It derives its name from the manner in which encryption is performed. In the rail fence cipher, the plaintext is written downwards diagonally on successive "rails" of an imaginary fence, then moving up when the bottom rail is reached, down again when the top rail is reached, and so on until the whole plaintext is written out. The ciphertext is then read off in rows.

SMTP:

SMTP stands for Simple Mail Transfer Protocol. SMTP is a set of communication guidelines that allow software to transmit an electronic mail over the internet is called Simple Mail Transfer Protocol. It is a program used for sending messages to other computer users based on e-mail addresses.

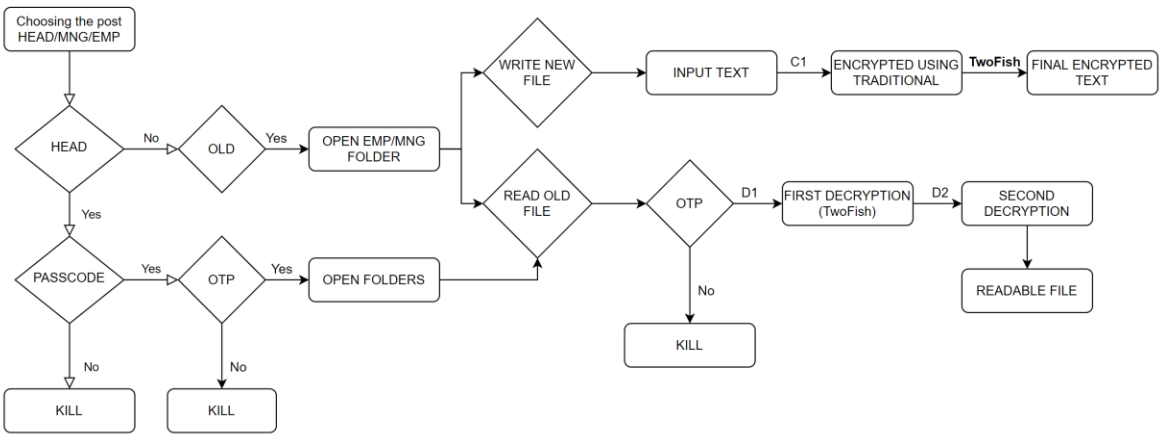
2. Framework, Architecture or Module for the Proposed System

We are going to use python for programming the code. Using the OS module, we are going to access the file system of the computer. Make folders and include all the functionalities that are available in the current model of file system architecture. Then using the pickle module, we are going to save the data so that whenever we relaunch the program the previous data loads with it. So, making a memory for our program. Using SMTP, we are going to send a mail in which there will be OTP for accessing the file which the user made. And finally, using random library we are going to generate the OTP and the random 128 bits key for the twofish algorithm. Then finally we will write the code for the twofish algorithm to encrypt the file made by the employee.

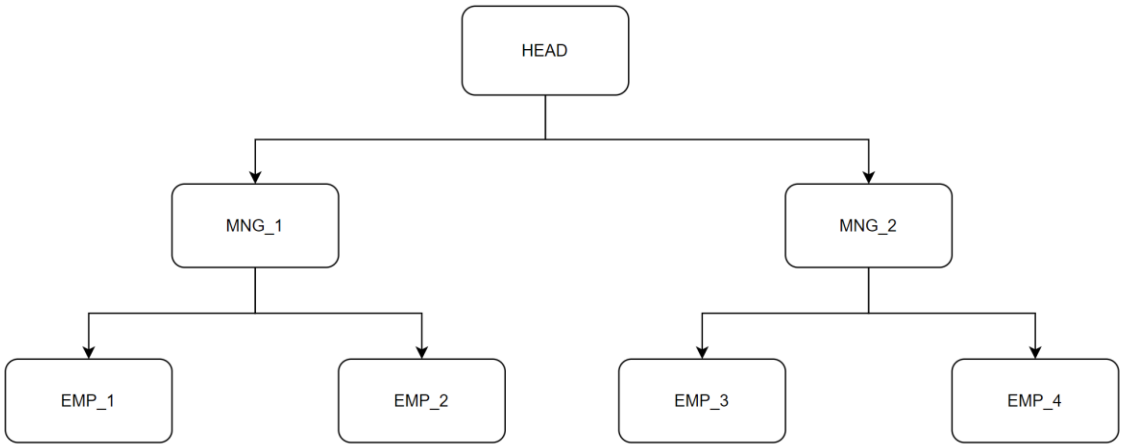
The file system is built I such a way that it will first ask for the OTP before accessing any file or deleting any file from the employee. The email address will be stored whenever a new employee comes and registered in the file system. Also, head will have to first enter a customized passcode then the OTP will be send to his officially registered email address.

3. Proposed System Model

System Design



File system hierarchy



C1 encryption process

Proposed System Analysis and Design

- A 3-nary enterprise system which extensible to n-nary system.
- Used Twofish algorithm (fastest symmetric algorithm known).
- For high security, using OTP protected file so that either the employee who have written it or the superior of the employee can read it.
- Using some fast traditional cryptographic algorithm during the input to increase security without loss of speed.

Why TwoFish Algorithm?

Details:

- 128-bit symmetric cipher
- Variable key lengths of 128-192 & 256

- 16 rounds
- Feistel network

Advantages:

- No weak keys
- Efficiency
- Flexible design
- Fastest algorithm

Table 1: Comparison of Encryption time (in ms) of the Algorithms

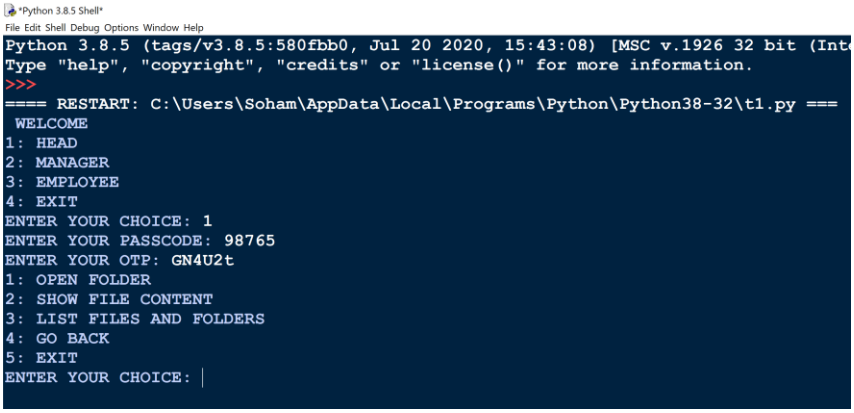
Algorithm	Encryption time (in ms)
AES	8.9
Twofish	3.1
Blowfish	4.2

Table 2: Comparison of Decryption time(in ms) of the Algorithms

Algorithm	Decryption time (in ms)
AES	7.4
Twofish	4.1
Blowfish	4.9

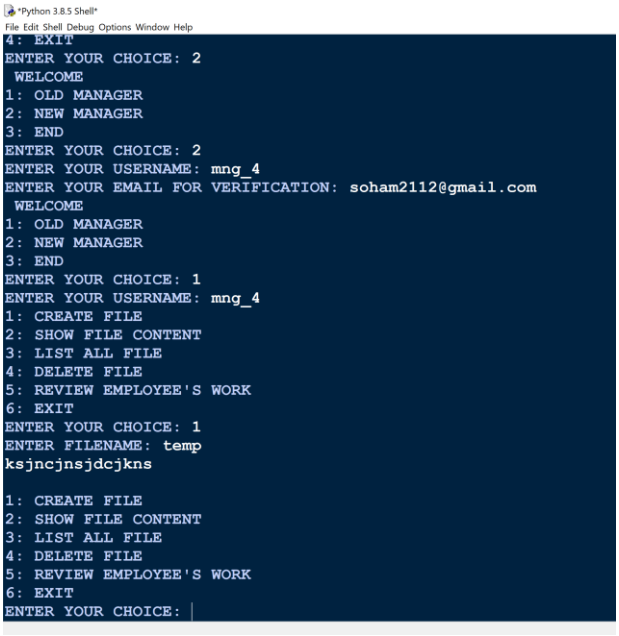
Program Output

1. Vulnerability of the head



Explanation: Asking for passcode and OTP sent to the email.

2. File created by a manager



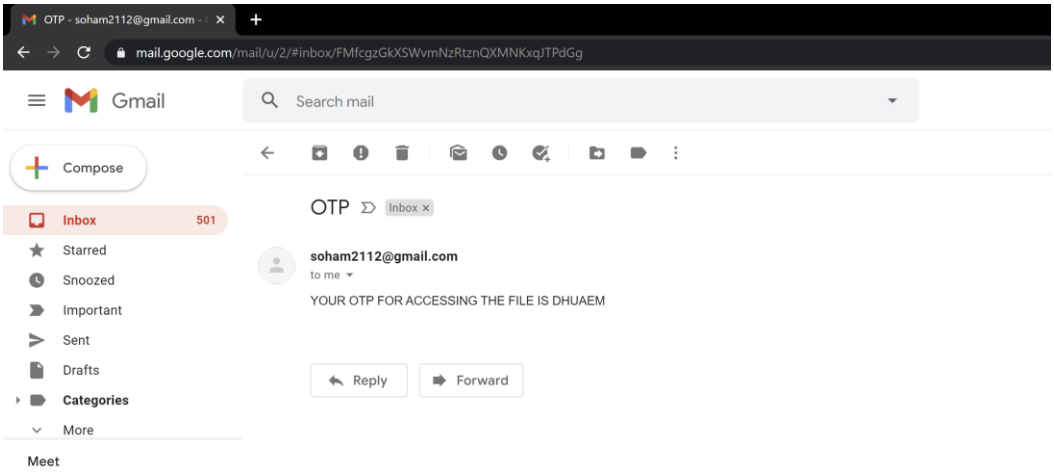
Explanation: New manager registered and a file is created. Also, manager's email address is registered.

3. Reading a file

```
1: CREATE FILE
2: SHOW FILE CONTENT
3: LIST ALL FILE
4: DELETE FILE
5: REVIEW EMPLOYEE'S WORK
6: EXIT
ENTER YOUR CHOICE: 2
ENTER FILENAME: temp
ENTER YOUR OTP: DHUAEM
ksjncjnsjdcjkns
Time taken =  0.034912109375
1: CREATE FILE
2: SHOW FILE CONTENT
3: LIST ALL FILE
4: DELETE FILE
5: REVIEW EMPLOYEE'S WORK
6: EXIT
ENTER YOUR CHOICE:
```

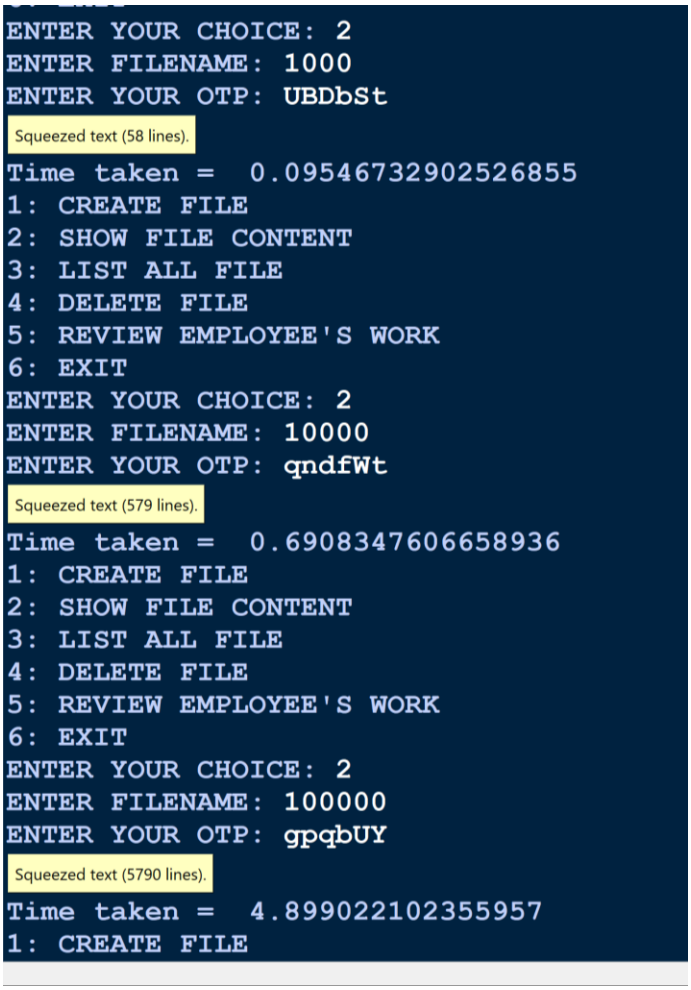
Explanation: Asks for file name and OTP. Gives the time taken for decryption.

4. OTP on email



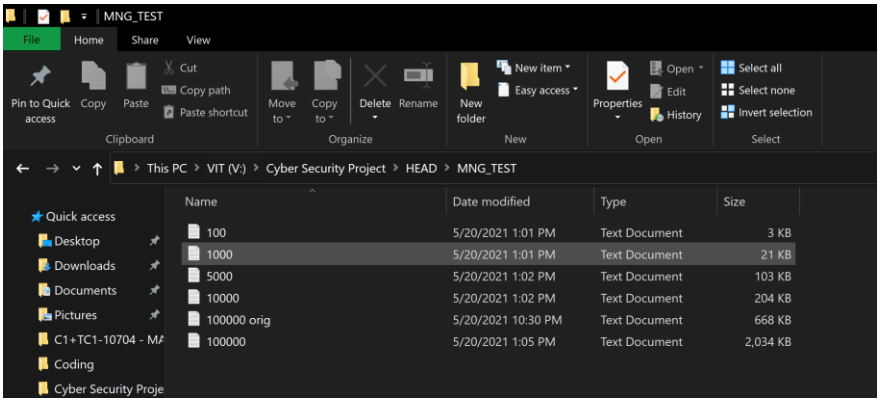
Explanation: OTP will be sent from a company email.

5. Decryption time of 1000, 10000 & 100000 words.



Explanation: 100000 words contain 632500 characters.

Drawback:



Explanation: 100000 original .txt file size = 668KB & 100000 encrypted file size = 2034KB

Therefore, the file size increases three times the original size.

Results and Discussion

1. **Cryptanalysis of the encrypted file**
   - TwoFish algorithm: 2^256 (256 bit algorithm)
   - Rail Fence algorithm: L = length of text, K = possible key, K<L
   - Reversing the string will confuse the attacker more. Because even after decrypting rail fence algorithm using brute force method the attacker won't get any meaningful answer in any possible key.
   - Rail fence is weak if the small key size is small. But in large company the text is large.
   - Final cryptanalysis: 2^256×(L−1)

2. **Cryptanalysis of the file system**
   - Our file system is secured using OTP which is shared through SMTP (Simple Mail Transfer Protocol).
   - SMTP uses PGP for its security which is not broken till date.
   - And PGP is secured using RSA-2048 which is also not broken till date.
   - Total Cryptanalysis: 2^2048
   - For OTP: 62^6 Possible combinations and only 1 try.
   - 62^6 = 1.999×10^13
3. **Results of decryption time:**

No of Words	Decryption Time (in seconds)
1000	0.0987

10000	0.689
100000	4.729

4. Conclusion

- We have formed a extensible file system which uses OTP to protect the file from false use from inside the system.
- And using twofish and other traditional methods we are encrypting the file to preventing the attacker from using the information from other the system where the file is stored.
- Accomplishing the problem, speed with security!

References

Weblinks:

1. https://www.geeksforgeeks.org/os-module-python-examples/#:~:text=The%20OS%20module%20in%20Python,interacting%20with%20the%20operating%20system.&text=This%20module%20provides%20a%20portable,interact%20with%20the%20file%20system.
2. https://realpython.com/python-pickle-module/#:~:text=The%20Python%20pickle%20module%20is,result%20is%20not%20human%20readable.
3. https://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol

Papers:

1. Huang Jianzhong, Xie Changsheng and Cai Bin, "Research and Implement of an Encrypted File System Used to NAS," Second IEEE International Security in Storage Workshop, Washington, DC, USA, 2003, pp. 73-73, doi: 10.1109/SISW.2003.10007.
2. Kumar, Shishir, et al. "Efficient methodology for implementation of Encrypted File System in User Space." arXiv preprint arXiv:0908.0551 (2009).
3. An Overview of Cryptanalysis Research for the Advanced Encryption Standard Conference: IEEE Military Communications Conference (MILCOM 2010)
4. Advances in Social Science, Education and Humanities Research (ASSEHR), volume 141 International Conference on Public Policy, Social Computing and Development 2017 (ICOPOSDev 2017)
5. International Journal of Advanced Research in Computer and Communication Engineering Vol. 3, Issue 1, January 2014
6. Kehe Wu, Yakun Zhang, Wenchao Cui, Ting Jiang, “Design and implementation of encrypted and decrypted file system based on USBKey and hardware code”, AIP Conference Proceedings 1839, 020215 (2017); https://doi.org/10.1063/1.4982580.
7. Abhay Khoje, Salih K A, Rajat Moona, “TransCrypt: an Enterprise Encrypting File System over NFS”, Proceedings of the World Congress on Engineering 2009 Vol I WCE 2009, July 1 - 3, 2009, London, U.K., ISBN: 978-988-17012-5-1.

8. Wenqian Yu, Weigang Li, Junyuan Wang, “A study of HSM based key protection in encryption file system”, 2016 IEEE Conference on Communications and Network Security (CNS), 17-19 Oct. 2016, Philadelphia, PA, DOI: 10.1109/CNS.2016.7860507.
9. Zhang Kai; Cheng En; Gao Qinquan, “Analysis and Implementation of NTFS File System Based on Computer Forensics” 2010 Second International Workshop on Education Technology and Computer Science, 6-7 March 2010, DOI: 10.1109/ETCS.2010.434

Appendix [CODE]

```
import os
import pickle
import random
import string
import binascii
import base64
import time
from twofish import Twofish

IMP = dict()
PATH = dict()
EX = dict()
KEY = dict()
LIST = dict()
def multinput():
    lines = []
    while True:
        line = input()
        if line:
            lines.append(line)
        else:
            break
```

```
    text = '\n'.join(lines)
    return text
def listToString(l):
    s = ''
    for char in l:
        s+=str(char)
    return s
def listToStringD(l):
    s = ''
    for char in l:
        s+=str(char.decode('utf8'))
    return s

def equilizer(n, data, FileName):
    rem = len(data)//n
    list=[]
    k=0
    j=n
    for i in range(rem+1):
        if i == rem:
            x = len(data[k:])
            y = n - x
            EX[FileName]=y
            sub = rand_pass(y)
            temp = data[k:]+sub
            list.append(temp)
            break
        temp = data[k:j]
        k+=n
        j+=n
        list.append(temp)
```

```
    clist = twofishEn(list,FileName)
    return clist

def dequilizer(FileName):
    plist=[]
    clist = LIST[FileName]
    key = KEY[FileName]
    t = Twofish(key)
    for byte in clist:
        plist.append((t.decrypt(byte)))
    z = EX[FileName]
    n = len(clist[0])
    y = n - z
    tot = len(clist)
    temp = plist[tot-1]
    temp = temp[:y]
    plist[tot-1] = temp
    return plist

def rand(size):
    ra = base64.b16encode(random.getrandbits(size).to_bytes(16,
byteorder='little'))
    return ra
def rand_pass(size):
    generate_pass = ''.join([random.choice( string.ascii_uppercase +
string.ascii_lowercase + string.digits)for n in range(size)])
    return generate_pass

def twofishEn(elist,FileName):
    key = rand(16)
    key = binascii.unhexlify(key)
```

```
    KEY[FileName]=key
    t = Twofish(key)
    clist = []
    for bits in elist:
        clist.append((t.encrypt(bits.encode('utf8'))))
    return clist


def MisplaceData(data):
    if len(data)%2!=0:
        data+='`'
    rev = data[::-1]
    even = rev[0::2]
    odd = rev[1::2]
    return even + odd

def AntiMisplaceData(data):
    even = list(data[:len(data)//2])
    odd = list(data[len(data)//2:])
    final = [even,odd]
    string=[]
    for j in range(len(final[0])):
        for i in range(2):
            string.append(final[i][j])
    for char in string:
        if char == '`':
            string.remove('`')
    str1 = ''
    return str1.join(string[::-1])

def Mail(receiver_email,OTP):
```

```
    import smtplib, ssl
    from email.message import EmailMessage
    port = 465
    smtp_server = "smtp.gmail.com"
    sender_email = "soham2112@gmail.com"
    password = 'sohamfaldu2001'
    msg = EmailMessage()
    msg.set_content('YOUR OTP FOR ACCESSING THE FILE IS
{}'.format(OTP))
    msg['Subject'] = 'OTP'
    msg['From'] = 'soham2112@gmail.com'
    msg['To'] = receiver_email
    context = ssl.create_default_context()
    with smtplib.SMTP_SSL(smtp_server, port, context=context) as
server:
        server.login(sender_email, password)
        server.send_message(msg)

def rand_pass(size):
    generate_pass = ''.join([random.choice( string.ascii_uppercase +
string.ascii_lowercase + string.digits)for n in range(size)])
    return generate_pass

######################################### DRIVER CODE
#####################################################################

parent_dir = 'V:\Cyber Security Project'
IMP = pickle.load(open(parent_dir+'/save1.p','rb'))
PATH = pickle.load(open(parent_dir+'/save2.p','rb'))
EX = pickle.load(open(parent_dir+'/save3.p','rb'))
KEY = pickle.load(open(parent_dir+'/save4.p','rb'))
```

```
LIST = pickle.load(open(parent_dir+'/save5.p','rb'))
os.chdir(parent_dir)
choice = 0
while choice!=4:
    print(' WELCOME ')
    print('1: HEAD')
    print('2: MANAGER')
    print('3: EMPLOYEE')
    print('4: EXIT')
    choice = int(input('ENTER YOUR CHOICE: '))
    if choice == 1:
        email = 'sohamfaldu@gmail.com'
        code = int(input('ENTER YOUR PASSCODE: '))
        if code!= 98765:
            break
        otp = rand_pass(6)
        receiver = email
        Mail(receiver,otp)
        temp_otp = input('ENTER YOUR OTP: ')
        if temp_otp != otp:
            break
        path = os.path.join(os.getcwd(),'HEAD')
        temp = 0
        while temp!=5:
            print('1: OPEN FOLDER')
            print('2: SHOW FILE CONTENT')
            print('3: LIST FILES AND FOLDERS')
            print('4: GO BACK')
            print('5: EXIT')
            temp = int(input('ENTER YOUR CHOICE: '))
            if temp == 1:
```

```
                name = input('ENTER FOLDER NAME: ').upper()
                if name[0:3] == 'MNG':
                    path = os.path.join(parent_dir,'HEAD',name)
                    os.chdir(path)
                    print(os.listdir(path))
                elif name[0:3] == 'EMP':
                    mng = input('ENTER MANAGER\'S NAME: ').upper()
                    path = os.path.join(parent_dir,'HEAD',mng,name)
                    os.chdir(path)
                    print(os.listdir(path))

            elif temp == 2:
                FileName = input('ENTER FILE NAME: ')
                otp = rand_pass(6)
                receiver = email
                Mail(receiver,otp)
                temp_otp = input('ENTER YOUR OTP: ')
                if temp_otp == otp:
                    path = PATH[name]
                    with open(os.path.join(path, FileName)+'.txt',
'r') as fp:
                        start = time.time()
                        x = fp.read()
                        y = dequilizer(FileName)
                        z = listToStringD(y)
                        print(AntiMisplaceData(z))
                        end = time.time()
                        print('Time taken = ',end-start)
                    fp.close()
                else:
                    break
```

```
            elif temp == 3:
                print(os.listdir(path))
            elif temp==4:
                if os.getcwd() == 'V:\Cyber Security Project\HEAD':
                    break
                os.chdir('../')
                path = os.getcwd()

    elif choice == 2:
        temp = 0
        while temp!=3:
            print(' WELCOME ')
            print('1: OLD MANAGER')
            print('2: NEW MANAGER')
            print('3: END')
            temp = int(input('ENTER YOUR CHOICE: '))
            if temp == 2:
                username = input('ENTER YOUR USERNAME: ').upper()
                directory = username
                email = input('ENTER YOUR EMAIL FOR VERIFICATION:
').lower()
                IMP[username]= email
                path = os.path.join(parent_dir, 'HEAD',directory)
                os.mkdir(path)
                PATH[username] = path
                pickle.dump(IMP,open(parent_dir+'/save1.p','wb'))
                pickle.dump(PATH,open(parent_dir+'/save2.p','wb'))
            elif temp == 1:
                username = input('ENTER YOUR USERNAME: ').upper()
                temp1=0
                while temp1!=6:
```

```
                    print('1: CREATE FILE')
                    print('2: SHOW FILE CONTENT')
                    print('3: LIST ALL FILE')
                    print('4: DELETE FILE')
                    print('5: REVIEW EMPLOYEE\'S WORK')
                    print('6: EXIT')
                    temp1 = int(input('ENTER YOUR CHOICE: '))
                    if temp1 == 1:
                        path = PATH[username]
                        FileName = input('ENTER FILENAME: ')
                        with open(os.path.join(path, FileName)+'.txt',
'w') as fp:
                            x = multinput()
                            y = MisplaceData(x)
                            z = equilizer(16,y,FileName)
                            LIST[FileName] = z
                            w = listToString(z)
                            fp.write(w)
                        fp.close()

pickle.dump(EX,open(parent_dir+'/save3.p','wb'))

pickle.dump(KEY,open(parent_dir+'/save4.p','wb'))

pickle.dump(LIST,open(parent_dir+'/save5.p','wb'))
                    elif temp1==2:
                        FileName = input('ENTER FILENAME: ')
                        otp = rand_pass(6)
                        receiver = IMP[username]
                        Mail(receiver,otp)
                        temp_otp = input('ENTER YOUR OTP: ')
```

```
                        if temp_otp == otp:
                            path = PATH[username]
                            start1 = time.time()
                            with open(os.path.join(path,
FileName)+'.txt', 'r') as fp:
                                x = fp.read()
                                y = dequilizer(FileName)
                                z = listToStringD(y)
                                print(AntiMisplaceData(z))
                                end1 = time.time()
                                print('Time taken = ',end1-start1)
                            fp.close()
                        else:
                            break
                    elif temp1==3:
                        path = PATH[username]
                        print(os.listdir(path))
                    elif temp1==4:
                        FileName=input('ENTER FILENAME: ')
                        path = PATH[username]
                        os.remove(os.path.join(path, FileName)+'.txt')
                    elif temp1==5:
                        t=0
                        path =
os.path.join(parent_dir,'HEAD',username)
                        print(os.listdir(path))
                        while t!=4:
                            print('1: LIST CONTENTS OF AN EMPLOYEE\'S
FOLDER')
                            print('2: OPEN FILE OF AN EMPLOYEE')
                            print('3: GO BACK')
```

```
                            print('4: EXIT')
                            t = int(input('ENTER YOUR CHOICE: '))
                            if t == 1:
                                emp = input('ENTER EMPLOYEE\'S
USERNAME: ').upper()
                                path =
os.path.join(parent_dir,'HEAD',username,emp)
                                os.chdir(path)
                                print(os.listdir(path))
                            elif t == 2:
                                emp = input('ENTER EMPLOYEE\'S
USERNAME: ').upper()
                                path =
os.path.join(parent_dir,'HEAD',username,emp)
                                fn = input('ENTER FILENAME: ')
                                otp = rand_pass(6)
                                receiver = IMP[username]
                                Mail(receiver,otp)
                                temp_otp = input('ENTER YOUR OTP: ')
                                if temp_otp == otp:
                                    with open(os.path.join(path,
fn)+'.txt', 'r') as fp:
                                        x = fp.read()
                                        y = dequilizer(fn)
                                        z = listToStringD(y)
                                        print(AntiMisplaceData(z))
                                    fp.close()
                            elif t==3:
                                if os.getcwd()== path:
                                    break
                                os.chdir('../')
```

```
                                path = os.getcwd()

    elif choice == 3:
        temp=0
        while temp!=3 :
            print(' WELCOME ')
            print('1: OLD EMPLOYEE')
            print('2: NEW EMPLOYEE')
            print('3: END')
            temp = int(input("YOUR CHOICE: "))
            if temp == 2:
                username = input('ENTER YOUR USERNAME = ').upper()
                directory = username
                email = input('ENTER YOUR EMAIL FOR VERIFICATION =
').lower()
                IMP[username]= email
                temp_mng = input('ENTER YOUR MANAGER\'S USERNAME:
').upper()
                path = os.path.join(parent_dir,'HEAD',temp_mng,
directory)
                os.mkdir(path)
                PATH[username] = path
                pickle.dump(IMP,open(parent_dir+'/save1.p','wb'))
                pickle.dump(PATH,open(parent_dir+'/save2.p','wb'))
            elif temp == 1:
                username = input('ENTER YOUR USERNAME = ').upper()
                temp1=0
                while temp1!=5:
                    print('1: CREATE FILE')
                    print('2: SHOW FILE CONTENT')
                    print('3: LIST ALL FILE')
```

```
                    print('4: DELETE FILE')
                    print('5: EXIT')
                    temp1 = int(input('ENTER YOUR CHOICE: '))
                    if temp1 == 1:
                        path = PATH[username]
                        FileName = input('ENTER FILENAME = ')
                        with open(os.path.join(path, FileName)+'.txt',
'w') as fp:
                            x = multinput()
                            y = MisplaceData(x)
                            z = equilizer(16,y,FileName)
                            LIST[FileName] = z
                            w = listToString(z)
                            fp.write(w)
                        fp.close()

pickle.dump(EX,open(parent_dir+'/save3.p','wb'))

pickle.dump(KEY,open(parent_dir+'/save4.p','wb'))

pickle.dump(LIST,open(parent_dir+'/save5.p','wb'))
                    elif temp1==2:
                        FileName = input('ENTER FILENAME = ')
                        otp = rand_pass(6)
                        receiver = IMP[username]
                        Mail(receiver,otp)
                        temp_otp = input('ENTER YOUR OTP: ')
                        if temp_otp == otp:
                            path = PATH[username]
                            start2 = time.time()
```

```
                            with open(os.path.join(path,
FileName)+'.txt', 'r') as fp:
                                x = fp.read()
                                y = dequilizer(FileName)
                                z = listToStringD(y)
                                print(AntiMisplaceData(z))
                                end2 = time.time()
                                print('Time Taken = ',end2-start2)
                            fp.close()
                        else:
                            break
                    elif temp1==3:
                        path = PATH[username]
                        print(os.listdir(path))
                    elif temp1==4:
                        FileName=input('ENTER FILENAME = ')
                        path = PATH[username]
                        os.remove(os.path.join(path, FileName)+'.txt')
```