MANASCryptor

File Encryption Software

MAANASIKA	-	20BCI0188
SNEHA CHAKRABORTY	-	20BCI0208
NISHIT CHAUDHARY	-	20BCI0280
AMAN SINGHAL	-	20BCI0287
SATVIKA MAHAPATRA	-	20BCI0289

A report submitted for the J component of

CSE1011- CRYPTOGRAPHY FUNDAMENTALS

Supervisor: Mr. Boominathan P

School of Computer Science and Engineering Vellore Institute of Technology, Vellore

December 2021

Table of Contents

Abstract

The aim of the project is to implement multipurpose encryption software using python. Files are available in different types of formats; like .png,.jpeg,.pdf, etc. We will try to develop software that allows us to encrypt different types of files using single encryption software.

We will be using python libraries in order to encrypt the files. If required we will use the new state-of-the-art encryption techniques in order to give the software a better edge than the existing ones.

Project Specification:

Project Name: MANSASCryptor

Professor: Boominathan P

Purpose: to implement a multipurpose encryption software using python. Files are available in different types of formats; like .png,.jpeg,.pdf, etc. We will try to develop software that allows us to encrypt different types of files using single encryption software.

Description: MANASCryptor is a file encryption software that allows the user to encrypt multiple files and also send the encrypted file and its key to the receiver via different paths at different times in order to reduce man-in-the-middle attacks.

Objective :

1. Allow encryption of different types of files under 1 roof.
2. Dismiss the possibility of a Man in the Middle attack.
3. Allow the user to send the file and key to the destination using 2 different paths which makes the flow of information safe.

Functionality :

1. User uploads a file.

2. The application encrypts the file and presents the user with the key.

3. The applications ask the user for the file destination(via email) and the key destination(via phone number)

4. The software sends the information to the required destination.

Performance :

MANASCryptor can successfully encrypt images, CSV files, and pdf files within 10 seconds and send the information to the destination within 5 minutes. This makes the software fast and versatile.

Dependencies :

1. Python 3
2. email(python package)
3. PyPDF2(python package)
4. cryptography(python package)
5. smtplib(python package)

Interface: Graphical User Interface with colorful buttons and alert messages.

1. INTRODUCTION:

File encryption software allows us to keep the important information away from the prying eyes by encrypting the files and then decrypting them with a proper key. Without using file encryption software, your files are much more at risk, complying with regulations will ultimately be more difficult, and security can be compromised across the board.

File encryption software is the need of companies that want to keep their data safe. We will be encrypting the file and sending the file to the user using email. The key will be sent to the user on a different device after a small-time halt after the transfer of a file.

1.1. Idea:

The idea is to create software that encrypts different types of files and also provides a safe pathway for the transfer of information in order to eradicate Man-in-Middle attacks.

1.2. Scope:

Software that encrypts different types of files and also provides a secure way of file transfer is needed by secret societies for sharing confidential information. High-level administration and confidential data can be encrypted and sent using MANASCryptor.

1.3. Novelty:

1. There is no such software that provides encryption and decryption of files along with providing a channel for information transfer.
2. The channel provided is by using existing modes of the file transfer.

2. LITERATURE SURVEY:

Arun Thileeban[1] explains in his paper that photographs are now the most common type of data used by the average citizen. In the recent past, there have been numerous leaks of private images due to program bugs. Rather than protecting binary data in images with an application, this paper suggests using the XOR Cipher to encrypt binary data in images pixel by pixel such that it cannot be easily manipulated or cracked. The proposed model describes several ways to encrypt an image using the XOR Cipher, and the analysis shows that the images are correctly encrypted by using the proposed model.

The Novel Approach[2] to File Encryption application's main goal is to protect the files that users share. The idea is applied by developing a website that allows users to build accounts and share their personal information with other users who have registered on the site. Because of the rise in password cracking, even the user's password is encrypted and saved in a database. Various encryption-decryption algorithms, such as Advanced Encryption Standard (AES) and Data Encryption Standard (DES), are used to encrypt user-information (DES).

From[3] instant messaging apps to social media, multimedia data has become an integral part of our lives. WhatsApp and other instant messaging apps use the AES 256 bit key for encryption. The security measures are in place to keep the data safe from unauthorized

access and to maintain privacy. The focus of this paper is on image data as a source of encryption and decryption. Along with text, the AES algorithm is used in an appropriate way for image cryptography. Because of its highly stable encryption and decryption, the AES algorithm was chosen. The amount of time it takes to encrypt and decrypt data is also calculated.

The economy of today is increasingly dependent on information flow, according to this article[4]. The main technique for safe file transfer is to get the right information to the right person at the right time. It is important that this strategy is carried out in such a way that data storage and transmission are both efficient and stable. End-to-end visibility, protection, and compliance management are needed for file transfer. A stable and controlled file transfer approach can assist the user in meeting the challenge of sharing electronic data securely and efficiently. The aim of this project is to create an online forum for file management and sharing.

The[5] Portable Document Format, also known as PDF, is one of the most commonly used document formats in the world, and it promotes document encryption to ensure information confidentiality. In this article, we look at PDF encryption and demonstrate two new strategies for decrypting encrypted documents. To begin, we take advantage of the PDF function of partially encrypted documents to wrap the encrypted portion of the document inside attacker-controlled material, allowing us to exfiltrate the plaintext once the document is opened by a legitimate user. Second, we take advantage of a loophole in the PDF encryption specification to modify encrypted information at will.

This paper[6] states that the recovery of an encrypted Word document is essential not only in the case of decrypting a user's Word document without the forgotten password but also in the case of forensic justice evidence acquisition. We looked at the file structure, encryption theory, and decryption key derivation approach of a Word document in this paper, and then presented an effective method for decrypting it. Following a realistic test, we discovered that our system can quickly obtain the original plaintext document (within an average time of 1.5 minutes), almost meeting the real-time decryption of Word document requirements.

Nowadays[7], protecting our confidential information on a device or over the internet, such as in online banking, online shopping, stock trading, and bill payments, is extremely critical. Our information shared over the internet is not secure without encryption. Encryption Algorithms ensure that information transmitted over the internet is secure. In this project, we propose AEDS (Advanced Encryption and Decryption Standard), a new cryptographic algorithm that combines the properties of the DES and AES algorithms. After comparing all three algorithms, we discovered that AEDS is more secure and stable in terms of data protection.

One[8] of the most powerful methods for achieving data protection and privacy is encryption. The original content of data is hidden using encryption methods, and the original information can only be retrieved using a key known as the decryption method. Encryption's aim is to preserve or protect data from unauthorized access in terms of accessing or altering it. Encryption may be done with the aid of a replacement technique, a moving technique, or mathematical operations. In the last year, several symmetric key based algorithms have been created. An effective and reliable symmetric key-based algorithm for encrypting and decrypting text data is proposed in this paper.

With the introduction of internet technology[9], the number of unauthorized users gaining access to data has grown. As a result, the transmission of information via image is becoming more common. It also becomes a more efficient method of data transmission. This problem can be solved using a variety of algorithms. One efficient approach is to use the AES (Advanced Encryption Standard) algorithm, which is the most well-known and widely used cryptographic algorithm because it is six times faster than 3-DES and significantly faster than the RSA algorithm. We proposed an AES-based image encryption and decryption algorithm in which the encryption contains a random image and the decryption contains the original image in this paper.

The[10] number of systems and devices that use image data has grown in lockstep with technological advancements. Image processing devices embedded in systems like the Internet of Things, drones, and closed-circuit television can now capture and automatically exchange people's images with networks. As a result, the possibility of image leakage invading one's privacy has skyrocketed. Traditional image-security techniques, such as privacy masking and image encryption, have a number of drawbacks, like data padding waste, inability to decode, inability to recognize photos without decoding, and the disclosure of private details.

In this paper[11], an encrypted and decrypted file system based on USBKey and hardware code is developed and implemented to protect the privacy of sensitive data. To authenticate a user, this device uses a USBKey and a hardware code. To encrypt a file with symmetric encryption, we use the random key, and to encrypt a random key with asymmetric encryption, we use USBKey. At the same time, we calculate the hash of the file using the MD5 algorithm to ensure its integrity. Big files can be encrypted and decrypted in a very short time, according to the results of the experiments. The system is highly efficient and ensures document protection.

We[12] suggest a text-to-image technique to hide text inside an image in this paper. This method involves storing the ASCII value in the X, Y coordinate of the image. One of the three Red, Green, or Blue channels stores the Ascii value, while the other two channels

store the message's next coordinate. The machine selects random images with a size of 256*256 pixels to produce the random keys. The decryption process results in the creation of a decrypted text file containing the message. The system is secure, and it could be enhanced by combining it with other encryption standard algorithms.

Secure file transfer[13], based on well-designed file encryption and authorization systems, puts a lot of effort into keeping passwords and other credentials secure. Passwords that are transferred and stored in the plaintext are vulnerable to attackers, eavesdroppers, and spyware. To prevent such disclosure, efficient encryption/authentication schemes employ a variety of mechanisms to reduce the risk of unencrypted credentials being revealed while also ensuring that any authentication data that is transmitted and stored is of limited benefit to an attacker.

Microsoft Word[14] is the most well-known and widely used word processor in the world today, and its most recent versions depend on eXtended Markup Language (XML) and the Open Office XML format (OOXML). MS Word documents also contain confidential information, and they can be sent over the internet, by email, or even stored in the cloud. As a result, the information in this document is vulnerable to a variety of security threats. This paper proposes two algorithms, one for hiding user-selected content from the document into a zero-dimensional image, and the other for protecting data within MS-word documents using cryptography and steganography techniques.

Ibiam Akanu[15] In recent years, Federal Polytechnic has used information and communication technology (ICT) to acquire and update student registration and examination records in real-time via the internet. Due to the vulnerability of these confidential students' documents to different forms of attacks, a security measure to secure this information must be developed and implemented. The approach used in this paper is to investigate the institution's current protection method and construct a 128-key duration AES algorithm. VB.net, Ms. Access, and the.net system V4 were used as software tools.

With[16] the rise in popularity of information technology and its applications, the electronic document was adopted as a means of data transmission and storage. As a result, an increasing number of electronic documents are at risk of being illegally acquired and displayed. This paper proposes a novel file content security approach based on file content partitioning restructuring to solve this issue. This method enhances document security and strengthens file content protection. However, it significantly improves the speed of the encryption and decryption processes.

Databases[17] hold a lot of confidential and sensitive information from various entities all over the world in today's IT world. The most crucial aspect is database stability. Data in a

database is encrypted to keep it secure. All of the information will be converted to ciphertext. But there's no point in hacking it. To encrypt a database, a variety of methods and techniques are available. This paper presents a limited body of literature on various techniques drawn from a variety of sources.

In[18] the field of communication, security is still a major concern. It is critical to transmitting data with high security and confidentiality over the internet for safe data transmission; information security is the most important issue of data exchange in networks and the internet. It is important to conceal transferred information from intruders in order to keep it secure. Various techniques, such as steganography and cryptography, are used to ensure data confidentiality during transmission. In the presence of third parties, cryptography is the practice of encrypted communication.

The rise of cloud computing[19] began with the use of computerized clouds, especially in commercial, government, and healthcare organizations. Clouds store vital data, lowering management costs and ensuring easy access. Cryptographic approaches are used to ensure the data's security, as well as to secure access to user data and increase confidence in cloud technology. We propose a new scheme in our paper to enable an attribute-based encryption method (ABE) involving multiple parties such as data owners, data users, cloud servers, and authority.

Big data[20] can process a large amount of both unstructured and structured data. Today's technology can help businesses by extracting large amounts of data and identifying patterns in order to forecast future trends. It provides the necessary insight into business strategy in order to reap significant benefits. Hadoop is a dependable framework that was created to efficiently distribute processes and storage on large amounts of data. Hadoop, on the other hand, does not come with any encryption features by default. Hadoop's encryption zone has a security problem, similar to what key management does outside of HDFS. Data in HDFS that is sensitive or confidential can be vulnerable to security breaches.

3. System Architecture:

3.1 High-Level Design:

Fig1:It shows how a file is uploaded from your system and is encrypted using MANASCryptor and how we get Decrypted files back using key

This diagram shows that when we upload the desired file from the system to MANASCryptor we get an encrypted file and key via email or SMS and then we can use the key to decrypt the file.

3.2 Detailed Design:

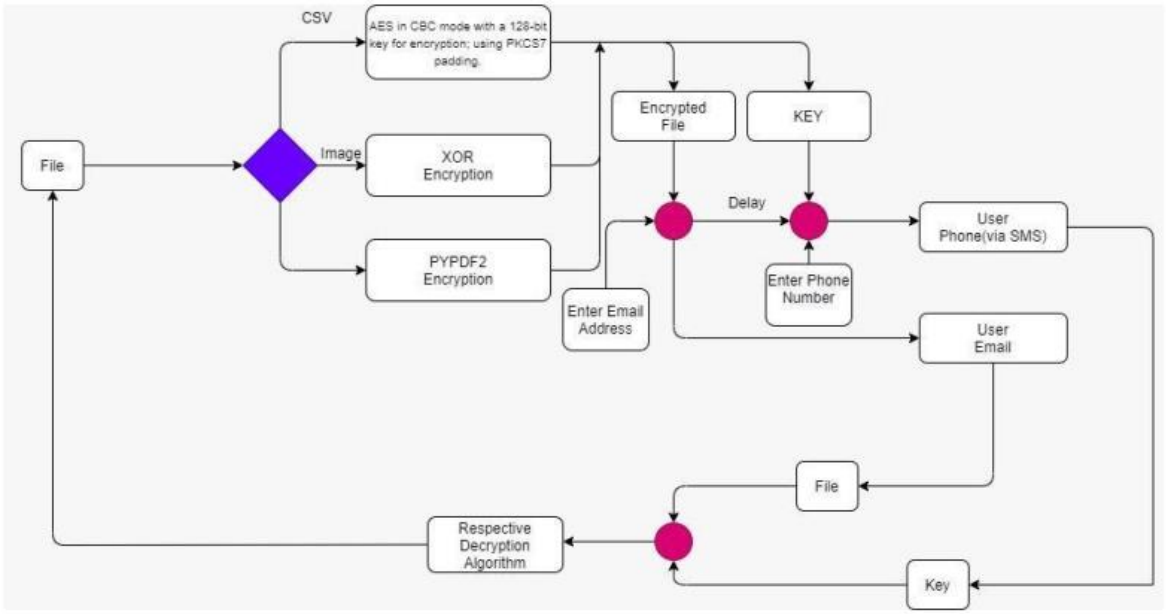


Fig 2: It shows a detailed diagram of different files being encrypted using MANASCryptor and with the generation of key how it is decrypted

This diagram shows that we can select files from the different options of files (CSV, image, pdf) then perform the required function on it then it will generate encrypted files and keys via mail or phone no. Then we can use the key generated to decrypt the file.

4. Algorithms Used

4.1 CSV File Encryption

The CSV files have been encrypted using fernet encryption. Fernet is a symmetric encryption implementation. The algorithm is actually Advanced Encryption Standard in CBC mode with a 128-bit key used for encryption and using PKCS7 padding.

Fernet takes in a user-provided message (an arbitrary sequence of bytes), a key (256 bits), and the current time in order to produce a token(encrypted message).

Key Format :
The fernet key is a base64url encoding of the given fields :
[Signing-Key|Encryption-Key] Both keys are 128 bits

Token Format:
[Version | Timestamp | IV | Ciphertext | HMAC]

1. Version → 8 bits
2. Timestamp → 64 bits
3. IV → 128 bits
4. Ciphertext → variable length, multiple of 128 bits
5. HMAC → 256 bits

Token Fields:

1. Version: Version of the format used by the token. 128(0x80)

2. Timestamp: Number of seconds elapsed between January 1, 1970, UTC and the time when the token was created

3. IV: It is unique for all tokens.

4. Ciphertext: It is of variable size but always a multiple of 128 bits.

5. HMAC : [Version | Timestamp | IV | Ciphertext]

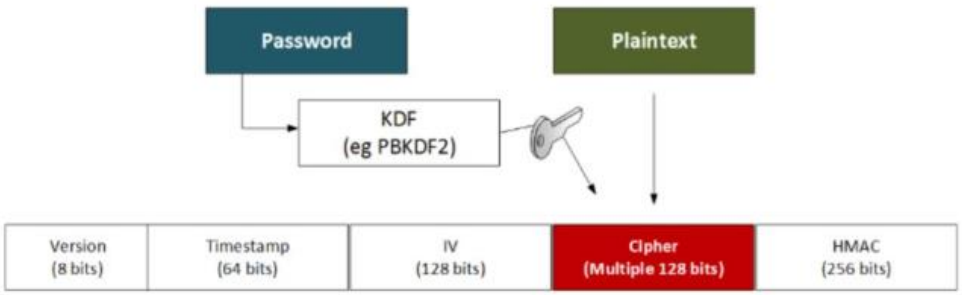


Fig 3.Diagram showing CSV file encryption

Encryption:

1. Record the current time for the timestamp field.

2. Choose a unique IV.

3. Construct the ciphertext:

    i. Use PKCS7 padding and pad the message to a multiple of 16 bytes

    ii. Encrypt the message using AES-128 in CBC mode.

4. Compute HMAC field

5. Concentrate all fields in the format above

6. Convert the entire token into base64 URL format.

Decryption:

1. Decode the token from base64url format
2. See if the 1st byte of the token is 0x80
3. If the span of life has been specified, see that the span has not been completed.
4. Recompute the HMAC from other fields and the user-supplied signing key.
5. Ensure that the recomputed HMAC matches the HMAC field in the token.
6. Decrypt the ciphertext using AES-128(CBC) with recorder IV and encryption key.
7. Unpad the encrypted plain text.

The cryptography module in python helps us to use this algorithm and encrypt files. This saves our time and thus we don't have to write code for file acquisition.

4.2 Image File Encryption

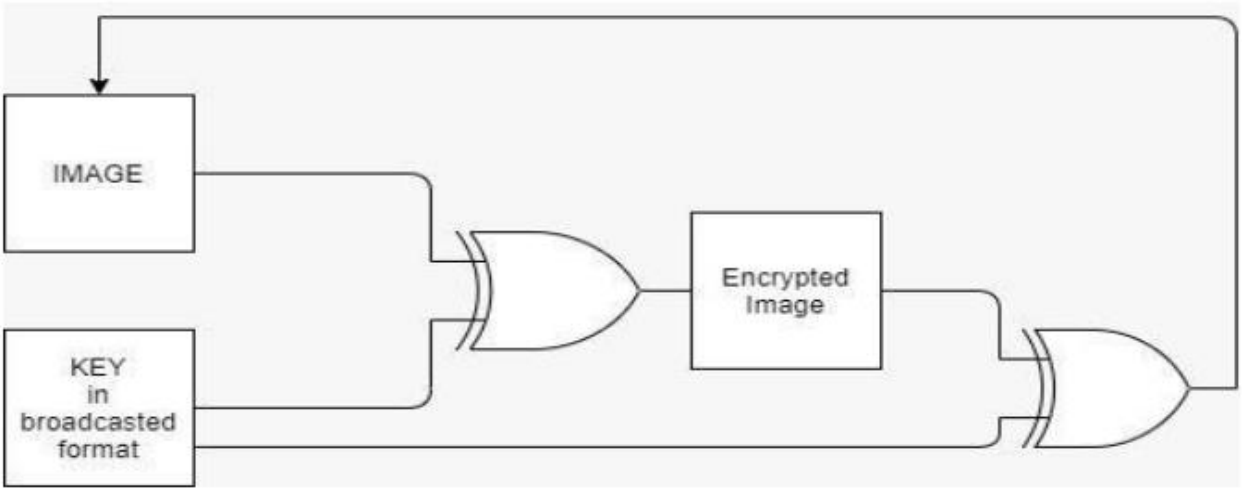


Fig 4: Diagram Showing Image file Encryption

1. Select a random number. (Key)

2. Traverse all the pixels of the image and compute the XOR of the Key and the Pixels and save the result in the image.

3. The Image gets encrypted, save the image.

Decryption

1. Get the Key from the User.

2. Traverse all the pixels of the image and compute the XOR of the Key and the Pixels and save the result in the image.

3. The Image is decrypted, save the image.

4.3 PDF Encryption

We use the PYPDF2 module in python in order to encrypt the file. The PYPDF2 password protects the file using the AES-128 encryption algorithm.(AES-256 is also used sometimes). AES is a symmetric-key block cipher. It was established in 2001 by NSIT(National Institute of Standards and Tech).

Encryption:

1. Generate a random password using alphabet, numbers, and special characters.

2. Read the PDF file as a string.

3. Encrypt the string using the AES-128 algorithm and the random password generated.

4. Present the random password to the user.

Decryption:

1. Get the Key(random password)

2. Read the encrypted file as a string.

3. Decrypt the string using the AES-128 algorithm and the key.

4. Present the file to the receiver.

5. Results and Discussion:

5.1. Implementation and Screenshot:

5.1.1.HomeScreen

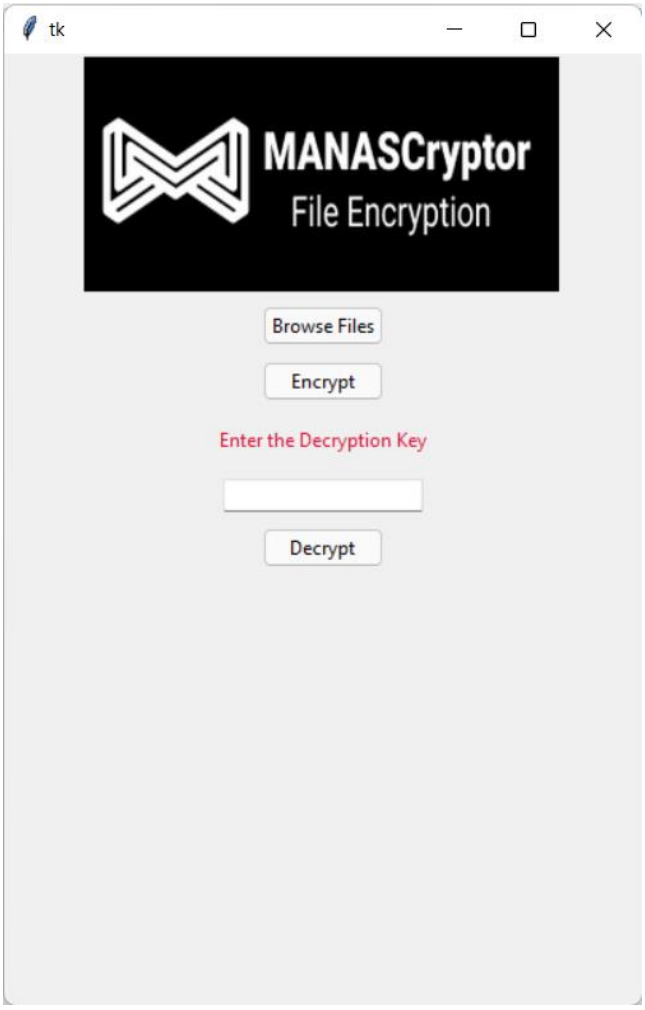
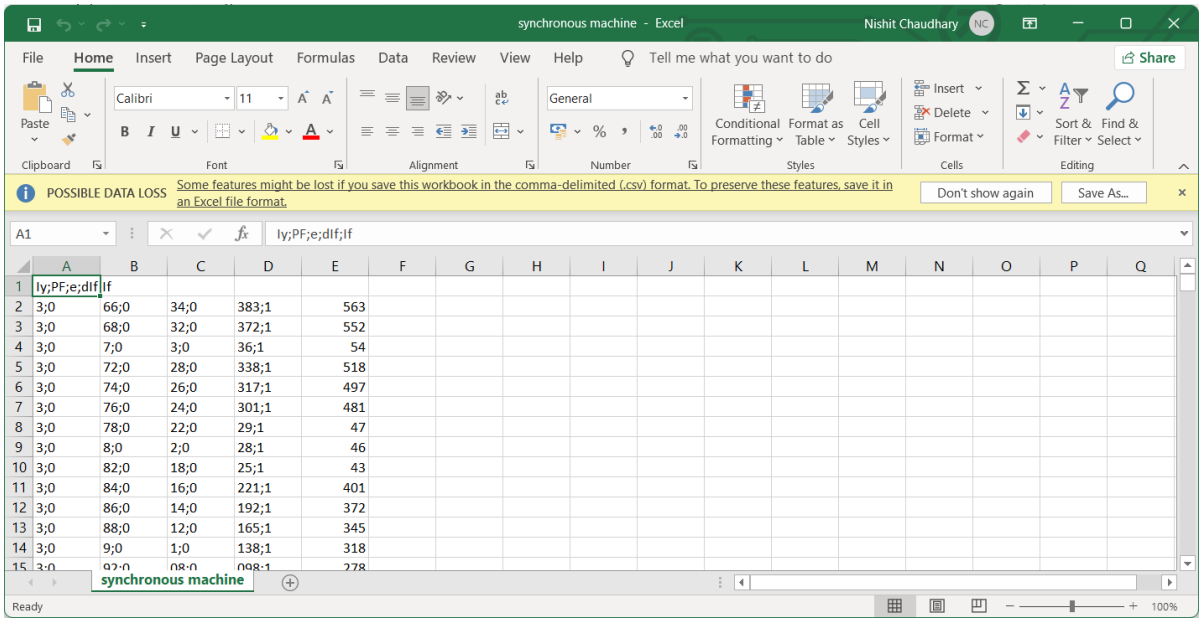


Fig 5:It shows the Homescreen of MANASCryptor

5.1.2. Normal File:



	A	B	C	D	E
1	Iy;PF;e;dIf;If				
2	3;0	66;0	34;0	383;1	563
3	3;0	68;0	32;0	372;1	552
4	3;0	7;0	3;0	36;1	54
5	3;0	72;0	28;0	338;1	518
6	3;0	74;0	26;0	317;1	497
7	3;0	76;0	24;0	301;1	481
8	3;0	78;0	22;0	29;1	47
9	3;0	8;0	2;0	28;1	46
10	3;0	82;0	18;0	25;1	43
11	3;0	84;0	16;0	221;1	401
12	3;0	86;0	14;0	192;1	372
13	3;0	88;0	12;0	165;1	345
14	3;0	9;0	1;0	138;1	318
15	3;0	92;0	08;0	098;1	278

Fig 6:It is the file that is being encrypted

5.1.3. Feeding Information:

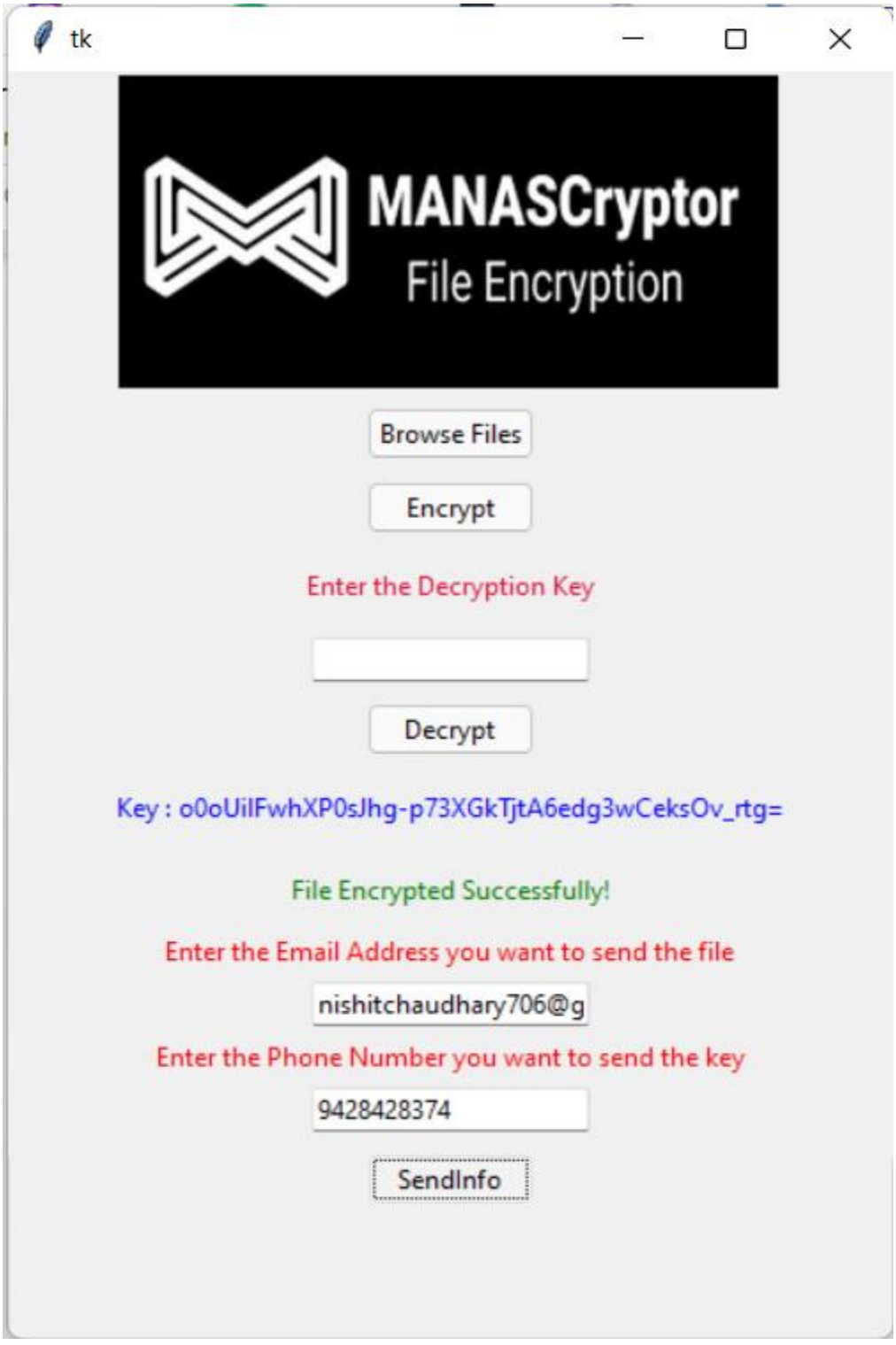


Fig 7:It shows how we feed information like email id and phone no. to get keys

5.1.4. Encrypted Mail:

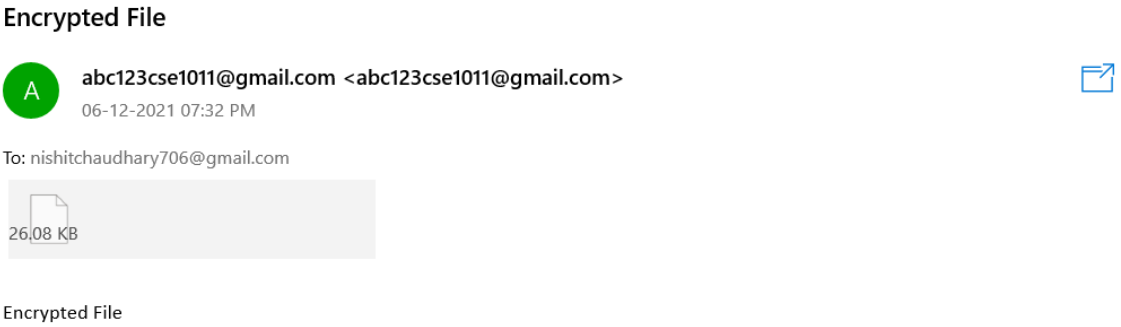


Fig 8:It shows encrypted mail

5.1.5. Encrypted file:

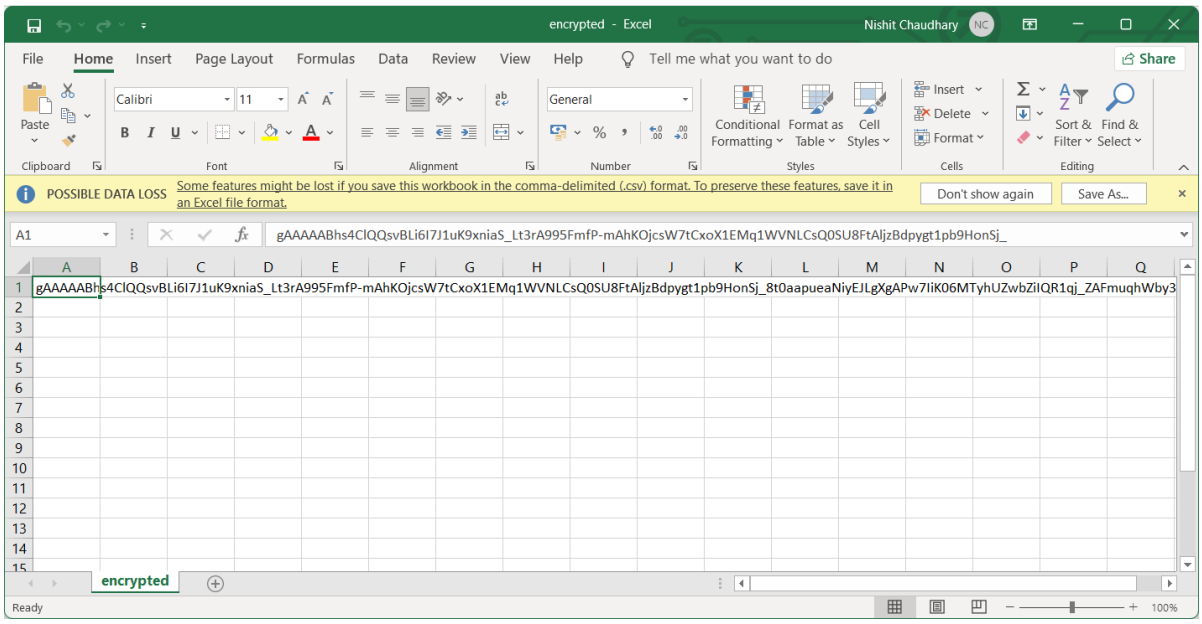


Fig 9:This figure shows an Encrypted file

5.1.6. Decryption key:

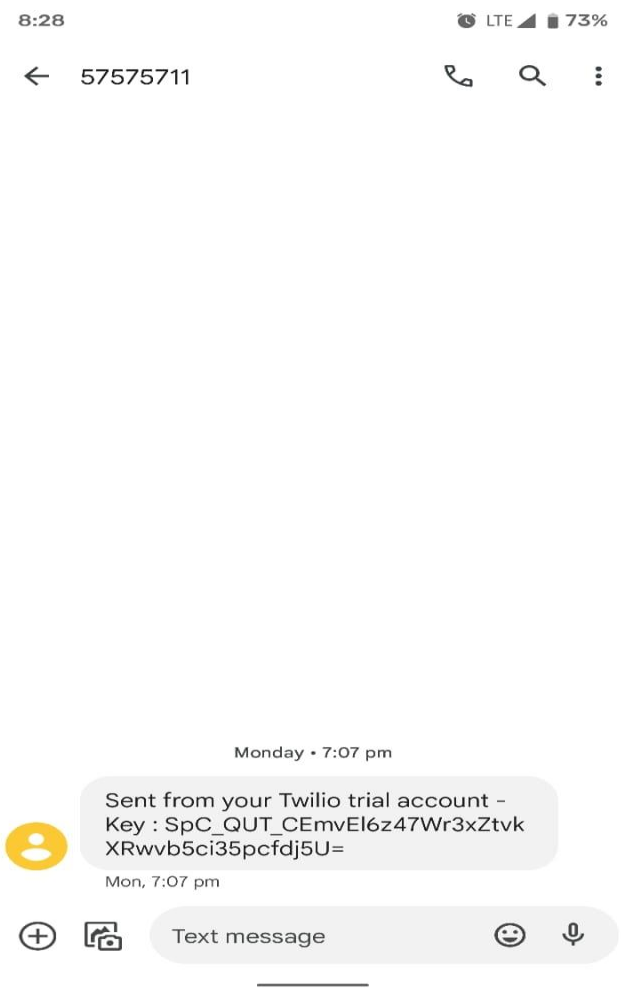
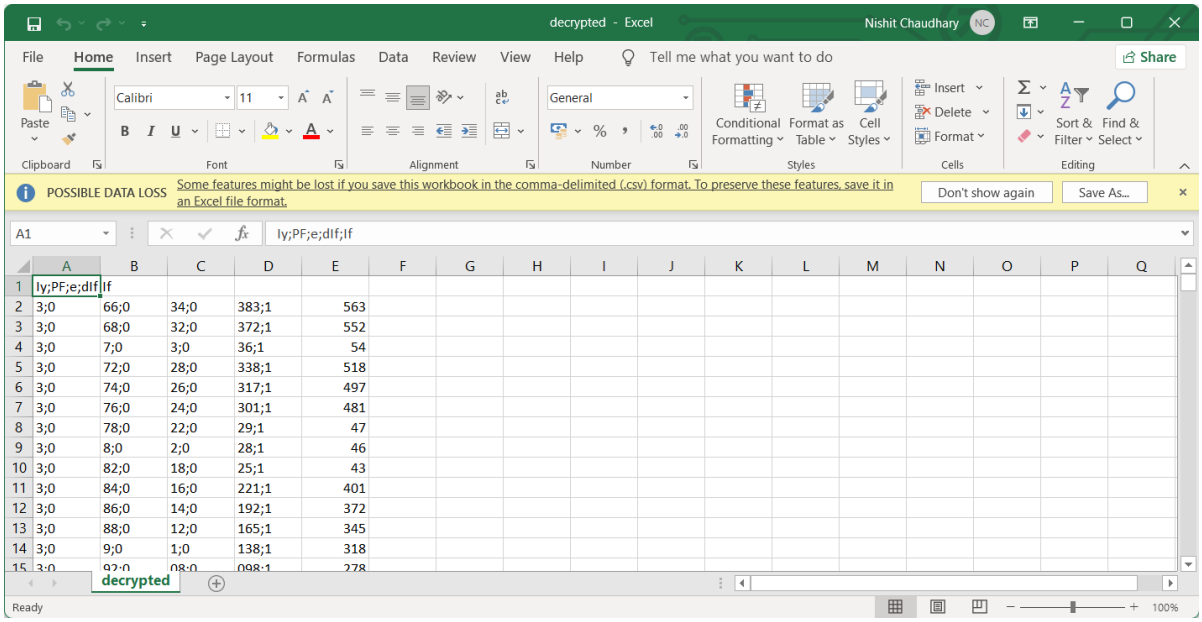


Fig 10:Figure shows decryption key

5.1.7. Decrypted File:



	A	B	C	D	E
1	Iy;PF;e;dIf;If				
2	3;0	66;0	34;0	383;1	563
3	3;0	68;0	32;0	372;1	552
4	3;0	7;0	3;0	36;1	54
5	3;0	72;0	28;0	338;1	518
6	3;0	74;0	26;0	317;1	497
7	3;0	76;0	24;0	301;1	481
8	3;0	78;0	22;0	29;1	47
9	3;0	8;0	2;0	28;1	46
10	3;0	82;0	18;0	25;1	43
11	3;0	84;0	16;0	221;1	401
12	3;0	86;0	14;0	192;1	372
13	3;0	88;0	12;0	165;1	345
14	3;0	9;0	1;0	138;1	318
15	3;0	92;0	08;0	098;1	278

Fig 11:It shows the Decrypted file(Original file)

5.2. Source Code:

```
from tkinter import *
from tkinter.ttk import *
from tkinter.filedialog import askopenfile,asksaveasfile
from PIL import ImageTk,Image
from cryptography.fernet import Fernet
from PyPDF2 import PdfFileWriter, PdfFileReader
import time
import random
import email
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.base import MIMEBase
from email import encoders
from os import name
import smtplib
from twilio.rest import Client

fromaddr = "abc123cse1011@gmail.com"
password = "crypto_project"

account_sid = "AC6a27af0eca5e5ffd12ce0d6c05393ddb"
auth_token = "a3c5fd76763ba6daa008040c9eaa4768"

# -Send Msg-#

def send_sms(phone_no,key):
    client = Client(account_sid, auth_token)
    message = client.messages.create(body="Key : "+key, from_=+18336205884, to=
int (phone_no))

# -Send Msg-#


# Send Email #

def send_email(filename,attatchment,toaddr,subject):
    msg = MIMEMultipart()
    msg['From'] = fromaddr
    msg['To'] = toaddr
    msg['Subject'] = subject

    body = subject
    msg.attach(MIMEText(body,'plain'))

    attatchment = open(attatchment,'rb')
```

```
    P = MIMEBase('application','octet-stream')
    P.set_payload((attatchment).read())
    encoders.encode_base64(P)

    P.add_header('Content-Disposition','attatchment : filename = %s ' % filename
    msg.attach(P)

    s = smtplib.SMTP('smtp.gmail.com',587)
    s.starttls()
    s.login(fromaddr,password)
    text = msg.as_string()
    s.sendmail(fromaddr,toaddr,text)
    s.quit()

# Send Email #
#Encryption
f = None

# CSV #

def file_encryption_csv(filename):
    key = Fernet.generate_key()
    fernet = Fernet(key)
    with open(filename, 'rb') as file:
        original = file.read()
    encrypted = fernet.encrypt(original)
    p = filename.split("/")
    p[-1] = "encrypted.csv"
    p = "/".join(p)
    with open(p, 'wb') as encrypted_file:
        encrypted_file.write(encrypted)
    key = key.decode('utf-8')
    Label(root, text='Key : '+str(key), foreground='blue').pack(side = TOP, pady
=10)
    return key,p

def file_decryption_csv(key,p):
    key = key.encode('utf-8')
    fernet = Fernet(key)
    with open(p, 'rb') as enc_file:
        encrypted = enc_file.read()

    decrypted = fernet.decrypt(encrypted)
    p = p.split("/")
    p[-1] = "decrypted.csv"
```

```
    p = "/".join(p)
    with open(p, 'wb') as dec_file:
        dec_file.write(decrypted)


# CSV #

# png #

def file_encryption_png(filename):
    key = random.randint(0,255)
    fin = open(filename,'rb')
    image = fin.read()
    fin.close()
    image = bytearray(image)
    for index, values in enumerate(image):
        image[index] = values ^ key
    fin = open(filename, 'wb')
    fin.write(image)
    fin.close()
    return key,filename

def file_decryption_png(key,p):
    key = int(key)
    fin = open(p,'rb')
    image = fin.read()
    fin.close()
    image = bytearray(image)
    for index, values in enumerate(image):
        image[index] = values ^ key
    fin = open(p, 'wb')
    fin.write(image)
    fin.close()

# png #

# password-generator#

def generate_password():
    alphabets = ['a','b','c','d','e','f','g','h','i','j' 'k','l','m','n','o','p'
                 ,'q','r','s','t','u','v','w','x','y','z']
    chars = ['.','/',',',',','!','@','#','$','%','^','&','*',]
    nums = ['0','1','2','3','4','5','6','7','8','9']
    random.shuffle(alphabets)
    random.shuffle(chars)
    random.shuffle(nums)
```

```
    l = alphabets[:8]+chars[:3]+nums[:3]
    l = list(l)
    random.shuffle(l)
    l = "".join(l)
    return l


# password-generator #

# pdf #

def file_encryption_pdf(filename):
    out = PdfFileWriter()
    file = PdfFileReader(filename)
    num = file.numPages

    for idx in range(num):
        page = file.getPage(idx)
        out.addPage(page)

    password = generate_password()
    out.encrypt(password)

    p = filename.split("/")
    p[-1] = "encrypted.pdf"
    p = "/".join(p)

    with open(p,'wb') as f:
        out.write(f)

    return password,p

def file_decryption_pdf(key,p):
    key = str(key)
    out = PdfFileWriter()
    file = PdfFileReader(p)
    password = key

    if file.isEncrypted:
        file.decrypt(password)
        for idx in range(file.numPages):
            page = file.getPage(idx)
            out.addPage(page)

    p = p.split("/")
    p[-1] = "decrypted.pdf"
```

```
    p = "/".join(p)

    with open(p, "wb") as f:
        out.write(f)

# pdf #

root = Tk()
root.geometry("400x600")

canvas = Canvas(root, width = 300, height = 150)
canvas.pack()

img1 = Image.open("logo1.png")
image1 = img1.resize((300,150), Image.ANTIALIAS)
img = ImageTk.PhotoImage(image1)
canvas.create_image(0,0, anchor=NW, image=img)

def open_file():
    file = askopenfile(mode ='r', filetypes =[('CSV Files', '*.csv'),
('Image Files', '*.png'),('PDF Files', '*.pdf')])
    if file is not None:
        global f
        f = str(file.name)
        pass

def Encrypt():
    if f is None:
        Label(root, text='Not uploaded a file yet!!', foreground='red').pack
(side = TOP, pady=10)
    else:
        if(f[-3:]=="csv"):
            t,q = file_encryption_csv(f)
        elif(f[-3:]=="png"):
            t,q = file_encryption_png(f)
        elif(f[-3:]=='pdf'):
            t,q = file_encryption_pdf(f)
        else:
            print(f[-3:])
    bar = Progressbar(root,orient=HORIZONTAL,length=200,mode='determinate')
    bar.pack(side = TOP, pady=20)
    for i in range(5):
        root.update_idletasks()
        bar['value'] += 20
        time.sleep(1)
    bar.destroy()
```

```
    Label(root, text='File Encrypted Successfully!', foreground='green').pack
(side = TOP, pady=10)

    Label(root, text='Enter the Email Address you want to send the file',
foreground='red').pack(side = TOP)
    emaddr = Entry(root)
    emaddr.pack(side=TOP,pady=5)

    Label(root, text='Enter the Phone Number you want to send the key',
foreground='red').pack(side = TOP)
    phnno = Entry(root)
    phnno.pack(side=TOP,pady=5)


    def send_info():
        send_email("File",q,str(emaddr.get()),"Encrypted File")
        time.sleep(500)
        send_sms("+91"+str(phnno.get()),t)

    btn = Button(root, text ='SendInfo', command = lambda:send_info())
    btn.pack(side = TOP, pady = 5)
    print(t)
    print(q)

def Decrypt(key):
    key = key.get()
    if(f[-3:]=='csv'):
        try:
            file_decryption_csv(key,f)
            bar = Progressbar(root,orient=HORIZONTAL,length=200,mode=
'determinate')
            bar.pack(side = TOP, pady=20)
            for i in range(5):
                root.update_idletasks()
                bar['value'] += 20
                time.sleep(1)
            bar.destroy()
            Label(root, text='File Decrypted Successfully!', foreground=
'red').pack( side = TOP, pady=10)

        except:
            Label(root, text='Error', foreground='blue').pack(side = TOP,
pady=10)
    elif(f[-3:]=='png'):
        try:
            file_decryption_png(key,f)
```

```
            bar = Progressbar(root,orient=HORIZONTAL,length=200,mode=
'determinate')
            bar.pack(side = TOP, pady=20)
            for i in range(5):
```

```
            bar.pack(side = TOP, pady=20)
            for i in range(5):
                root.update_idletasks()
                bar['value'] += 20
                time.sleep(1)
            bar.destroy()
            Label(root, text='File Decrypted Successfully!', foreground=
'red').pack( side = TOP, pady=10)

        except:
            Label(root, text='Error', foreground='blue').pack(side = TOP,
pady=10)
    elif(f[-3:]=='pdf'):
        try:
            file_decryption_pdf(key,f)
            bar = Progressbar(root,orient=HORIZONTAL,length=200,mode=
'determinate')
            bar.pack(side = TOP, pady=20)
            for i in range(5):
                root.update_idletasks()
                bar['value'] += 20
                time.sleep(1)
            bar.destroy()
            Label(root, text='File Decrypted Successfully!', foreground=
'red').pack( side = TOP, pady=10)

        except:
            Label(root, text='Error', foreground='blue').pack(side = TOP,
pady=10)
    else:
        print(f[-3:])

btn = Button(root, text ='Browse Files', command = lambda:open_file())
btn.pack(side = TOP, pady = 5)

upld = Button(root, text='Encrypt', command=Encrypt)
upld.pack(side = TOP, pady=5)

Label(root, text='Enter the Decryption Key', foreground='crimson').pack(side =
TOP, pady=10)
inputtxt = Entry(root)
inputtxt.pack(side=TOP,pady=5)

dec = Button(root,text='Decrypt',command = lambda:Decrypt(inputtxt))
dec.pack(side = TOP,pady = 5)

root.mainloop()
```

6. Conclusion

Hence with the end of this project, we hereby conclude that we have successfully implemented MANASCryptor along with the implementation of the important and required features that we proposed in our project initiatives such as encryption of CSV files, image, and pdf files. The main objective of our project was to encrypt the specified file and send a private key to the desired person so that he can easily decrypt it without other people being informed.

File encryption software is in great demand by companies that want to keep their data safe. So we encrypted the file and sent it to the user using email. The key will be sent to the user on a different device after a small-time halt after the transfer of a file.

Bibliography:

[1] Encryption and Decryption of an Image Data – a Parallel Approach By Raghu M E., K C Ravishankar,2018

[2]Secure and transparent file encryption system by Jui Diwale, Samiksha Thakur, Urvashi Kodwani, vol5, issue 1,2019

[3] SECURED FILE MANAGEMENT OVER INTERNET BY Prof. Balasaheb B. Gite, Shailesh Navghare, Abhishek Gupta, Siddharth Jain, vol 2, issue 3, pg 37,2015

[4] Practical Decryption exFiltration: Breaking PDF Encryption By Jens Müller, Fabian Ising, Vladislav Mladenov, pg 1-15,2019

[5] An Efficient Recovery Method of Encrypted Word Document by Li-jun Zhang, Fei Yu, pg 40-48,2017

[6] A new Cryptographic Algorithm AEDS for data security by Ali Mohammed Ali Argabi, Md ImranAlam,2019

[7] .A Secure and Fast Approach for Encryption and Decryption of Message Communication by EktaAgrawal, Parashu Ram Pal vol 7,2019

[8] An Image Encryption & Decryption And Comparison With Text - AES Algorithm by Dr. N. SubaRani, Dr. A. Noble Mary Juliet, K. Renuka Devi, vol 8, issue 9, pg 668,2019

[9] Partial image encryption using format-preserving encryption by Wonyoung Jang, ,Sun-YoungLee, volume 16, issue 3,2020

[10] Design and implementation of encrypted and decrypted file system based on USBKey and hardware code by KEHE WU, YAKUN ZHANG, vol 1839,2017

[11] SECURED TEXT TO IMAGE ENCRYPTION USING ASCII VALUE ENCODING BY ASHWINI A, RADHIKA V, YAMINI V, vol 7 issue 4,2020[12] Secure Data Encryption Through a Combination of AES, RSA and HMAC by E. S. I. Harba, Volume: 7 Issue: 4 , Pages: 1781-1785, August 2017

[13] A Novel Method to Protect Content of Microsoft Word Document Using Cryptography by Mohamed Ahmed Mohamed, Obay G. Altrafi, Mohamed O. Ismail, Mawada O. Elobied, vol 7 ,issue 4,2015

[14] STUDENTS DATA ENCRYPTION SYSTEM USING ADVANCED ENCRYPTION STANDARD ALGORITHM By Eguzo Chinwendu, Ezeorah Ezekiel, Chimezie Eguzo, vol 29,2018

[15] The Research on File Encryption Method Based on File Content Partitioning Restructuring By By Hui Xiao, Hongbin Wang, and Meitong Lin, vol 18, issue 5,2016

[16] Various Schemes for Database Encryption by P.R.Hariharan & Dr. K.P. Thooyamani, Volume-07, Issue-04 , Page no. 70-76, Feb-2019

[17] An Empirical Study of Security Issues In Encryption Techniques by Gahan A V, Geetha D Devanagavi, Volume 14, Number 5 (2019)

[18] Dual Authentication-Based Encryption with a Delegation System to Protect Medical Data by Aymen Mudheher Badr Yi Zhang Hafiz Gulfam Ahmad Umar,2019

[19] Data protection on Hadoop distributed file system by using encryption algorithms by Meisuchi Naisuty , Achmad Nizar Hidayanto, Nabila Clydea Harahap, Ahmad Rosyiq, Agus Suhanto, and George Michael Samuel Hartono, Volume 1444, 2019

[20] Secure Data Encryption Through a Combination of AES, RSA and HMAC by E. S. I. Harba, Volume: 7 Issue: 4 , Pages: 1781-1785, August 2017