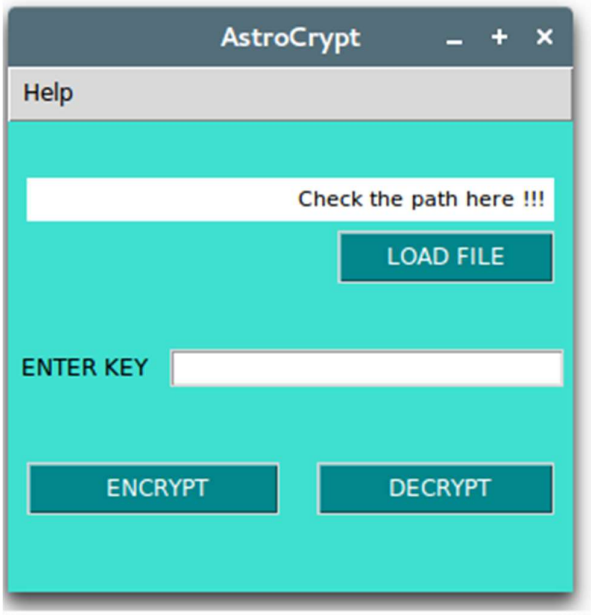
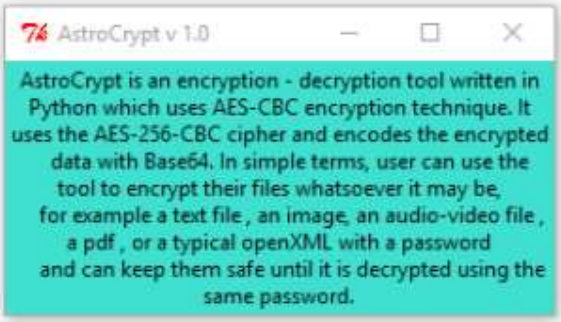
ASTROCRYPT

AstroCrypt is an encryption-decryption tool written in Python which uses AES-CBC encryption technique. It uses the AES-256-CBC cipher and encodes the encrypted data with Base64. In simple terms, user can use the tool to encrypt their files whatsoever it may be, for example a text file , an image, an audio-video file , a pdf , or a typical openXML with a password and can keep them safe until it is decrypted using the same password.

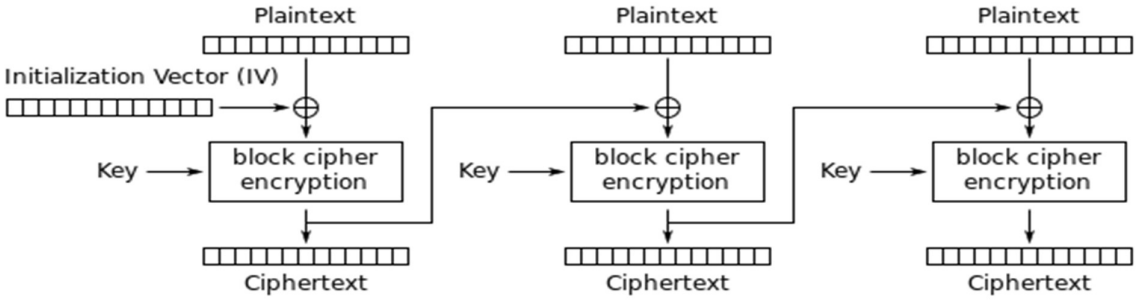




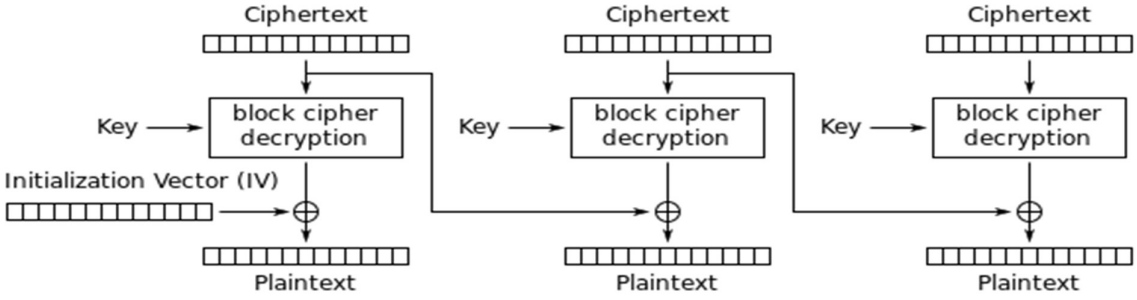
INTRODUCTION

AES (Advanced Encryption Standard) is a symmetric encryption algorithm. The algorithm was developed by two Belgian cryptographer Joan Daemen and Vincent Rijmen.AES was designed to be efficient in both hardware and software, and supports a block length of 128 bits and key lengths of 128, 192, and 256 bits. AES is very fast and secure, and it is the de facto standard for symmetric encryption.

CBC (Cipher block chaining) is a mode of operation for a block cipher (one in which a sequence of bits are encrypted as a single unit or block with a cipher key applied to the entire block). Cipher block chaining uses what is known as an initialization vector (IV) of a certain length. One of its key characteristics is that it uses a chaining mechanism that causes the decryption of a block of ciphertext to depend on all the preceding ciphertext blocks. As a result, the entire validity of all preceding blocks is contained in the immediately previous ciphertext block. A single bit error in a ciphertext block affects the decryption of all subsequent blocks. Rearrangement of the order of the ciphertext blocks causes decryption to become corrupted. Basically, in cipher block chaining, each plaintext block is XORed (see XOR) with the immediately previous ciphertext block, and then encrypted.Identical ciphertext blocks can only result if the same plaintext block is encrypted using both the same key and the initialization vector, and if the ciphertext block order is not changed.



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

SOFTWARE REQUIREMENTS

A standalone version is available for the tool, still if you want to use the tool by using the source code you require to have the below packages in your system.

1. Python 2.7+

2. PyCrypto

3. Tkinter

For links and installation guideline follow the next topic.

INSTALLATION

A. Using the direct executable to run:

A standalone executable file is being made available if user wants to run the tool without installing other softwares as required.

B. Using the tool in a windows operating system by running the source code:

1. Download and install Python 2.7

    1.a. Download link : https://www.python.org/downloads/

    1.b. Add the Python27 folder in your Environment Variable list.

2. Download and install Tkinter libraries

    2.a. Download link : http://www.activestate.com/activetcl/downloads

    2.b. Follow the instructions to install tkinter. Active tcl includes tcl, tk an other libraries required

3. Download and install PyCrypto

    3.a. Download link : https://pypi.python.org/pypi/pycrypto

    3.b. Copy it to the Python27 folder , unzip the file, and run the following commands in the cmd after changing the current directory position to the PyCrypto folder.

3.c. The modules are packaged using the Distutils, so you can simply run “python setup.py build” to build the package, and “python setup.py install” to install it.

4. To run AstroCrypt

4.a. Open cmd

4.b. Change the current directory to the directory containing the AstroCrypt.py file.

4.c. Run command : python AstroCrypt.py

C. Using the tool in a Linux operating system by running the source code:

1. Install Tkinter

1.a. In Ubuntu \ Debian \ Kali : sudo apt-get install python python-tk

1.b. In Fedora \ RedHat \ CentOS : sudo yum install tkinter

2. Download and install PyCrypto

2.a. Run the following command :

pip install pycrypto

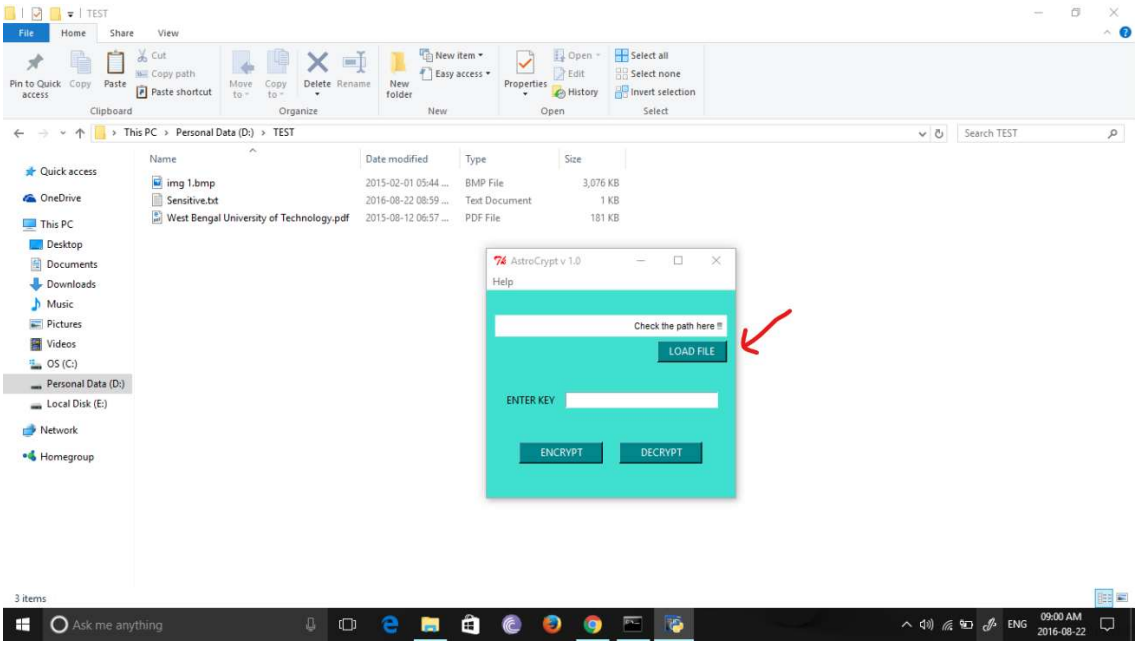
3. To run AstroCrypt

3.a. Open terminal

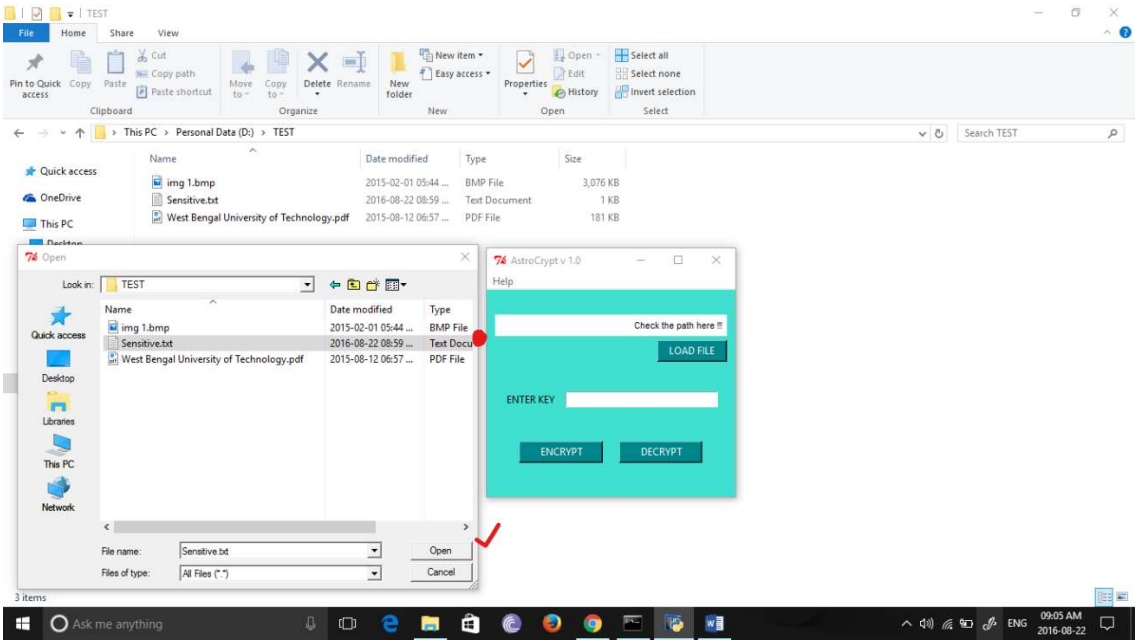
3.b. Change the current directory to the directory containing the AstroCrypt.py file.

3.c. Run command : python AstroCrypt.py

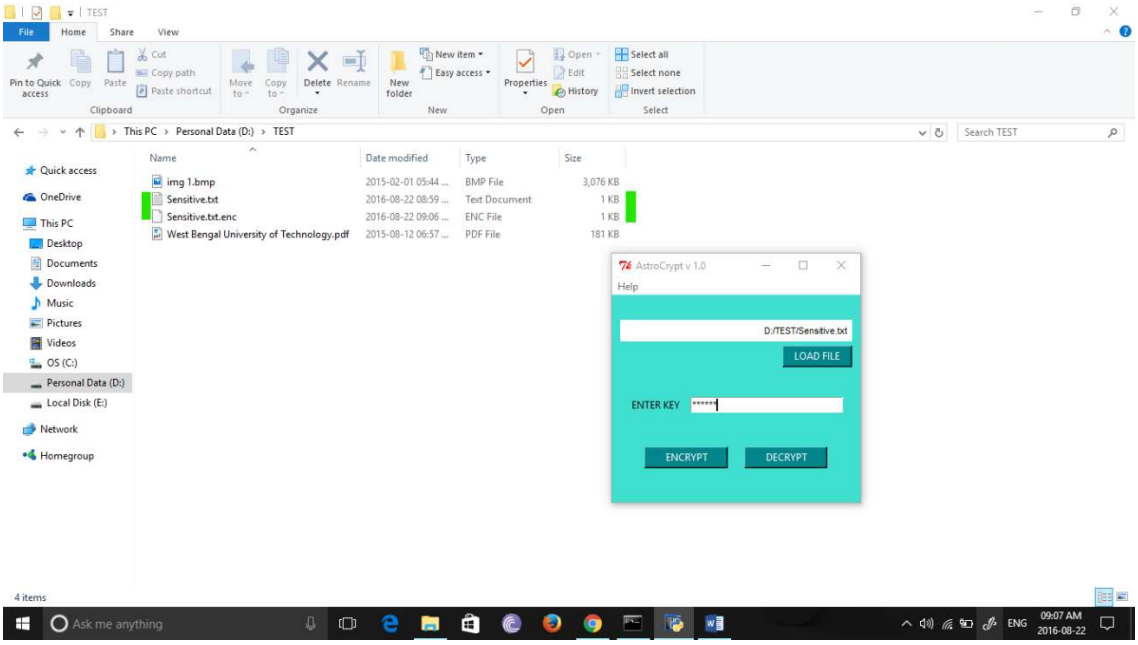
HOW TO USE:



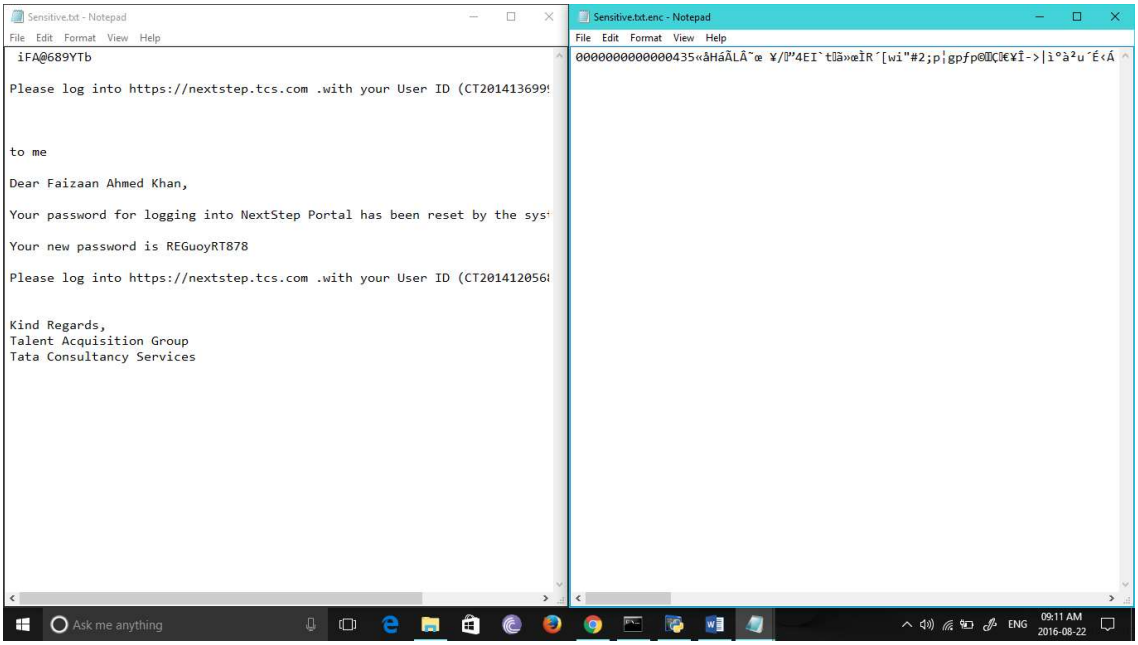
1. Load the file to encrypt



2. Choose the file to Encrypt and enter your password (You need to remember it to decrypt)
3. Click on encrypt.



4. “Filename.enc” is created , now you can delete the original file.



5. The original and the encrypted file.
6. When you need to Decrypt , load the file in the same way , enter the same password you have set while encrypting, you will find the decrypted file in the folder where your “.enc” file is kept.

CODE

```
import os, random
from Crypto.Cipher import AES
from Crypto.Hash import SHA256
import Tkinter
from Tkinter import *
import tkFileDialog
import tkMessageBox
filename = None
password = None


def encrypt(key , filename):
    chunksize = 64 * 1024
    outputFile = filename+".enc"
    filesize = str(os.path.getsize(filename)).zfill(16)
    IV = ''

    for i in range(16):
        IV += chr(random.randint(0, 0xFF))

    encryptor = AES.new(key, AES.MODE_CBC, IV)

    with open(filename, 'rb') as infile:
        with open(outputFile, 'wb') as outfile:
```

```
            outfile.write(filesize)
            outfile.write(IV)

            while True:
                chunk = infile.read(chunksize)

                if len(chunk) == 0:
                    break
                elif len(chunk) % 16 != 0:
                    chunk += ' ' * (16 - (len(chunk) % 16))

                outfile.write(encryptor.encrypt(chunk))


def decrypt(key, filename):
    chunksize = 64 * 1024
    outputFile = filename[:-4]

    with open(filename, 'rb') as infile:
        filesize = long(infile.read(16))
        IV = infile.read(16)

        decryptor = AES.new(key, AES.MODE_CBC, IV)

        with open(outputFile, 'wb') as outfile:
            while True:
```

```
            chunk = infile.read(chunksize)

            if len(chunk) == 0:
                break

            outfile.write(decryptor.decrypt(chunk))
        outfile.truncate(filesize)


def getKey(password):
    hasher = SHA256.new(password)
    return hasher.digest()

def load_file():
    global password, filename
    text_file = tkFileDialog.askopenfile()
    if text_file.name != None:
        filename = text_file.name
        var.set(filename)


def encrypt_the_file():
    global filename, password
    if filename != None:
        password = passInput.get()
        encrypt(getKey(password), filename)
```

```
        # sys.stdout.write('Password is ' + password)
        # encrypt_file(filename, password)
    else:
        tkMessageBox.showerror(title="Error", message="There was no file loaded to encrypt")


def decrypt_the_file():
    global filename, key
    if filename != None:
        password = passInput.get()
        decrypt(getKey(password), filename)
        # sys.stdout.write('gonna decrypt the file' + '\n')
        # decrypt_file(fname, password)
    else:
        tkMessageBox.showerror(title="Error", message="There was no file loaded to decrypt")


def donothing():
    filewin = Toplevel(root)
    mystring = StringVar()
    mystring= '''AstroCrypt is an encryption - decryption tool written in Python which uses AES-CBC encryption technique. It uses the AES-256-CBC cipher and encodes the encrypted
     data with Base64. In simple terms, user can use the tool to encrypt their files whatsoever it may be,
      for example a text file , an image, an audio-video file , a pdf , or a typical openXML with a password
```

```
    and can keep them safe until it is decrypted using the same password.'''


  labelU = Label(filewin, text=mystring)
  labelU.configure(wraplength=300, bg='turquoise',
fg='Black',activebackground='coral')
  labelU.pack()


root = Tkinter.Tk()
root.title("AstroCrypt v 1.0 ")
root.geometry("300x250")
root.configure(background='turquoise')


menubar = Menu(root)
helpmenu = Menu(menubar, tearoff=0, activebackground='coral',
bg='turquoise')


helpmenu.add_command(label="About AstroCrypt", command=donothing)
helpmenu.add_command(label="Quit the tool", command=root.quit)
menubar.add_cascade(label="Help", menu=helpmenu)


root.config(menu=menubar)



# GUI STUFF over here
# l = Label(root, text="AstroCrypt")
# l.configure(background='beige')
# l.pack(side=TOP, padx=0, pady=20, fill=X)
```

```
frame = Frame(root)
frame.pack(padx=0, pady=30)
frame.configure(background='turquoise')

var = StringVar()
Filelabel = Label(frame, textvariable=var, relief=FLAT)
Filelabel.configure(bg='white', fg='Black', anchor=E, font=("default", 8), padx=4,
pady=4, activebackground='coral', width=50)
var.set(" Check the path here !!!")
Filelabel.pack(side=TOP, padx=10, pady=0)

loadButton = Tkinter.Button(frame, text="   LOAD FILE   ", command=load_file)
loadButton.pack(side=RIGHT, padx=10, pady=5, fill=X)
loadButton.configure(bg='turquoise4', fg='white', activebackground='coral')

frame1 = Frame(root)
frame1.pack(padx=0, pady=0)
frame1.configure(background='turquoise')
frame2 = Frame(root)
frame2.pack(padx=0, pady=40)
frame2.configure(background='turquoise')

l1 = Label(frame1, text="ENTER KEY")
l1.configure(background='turquoise')
passInput = Entry(frame1, show="*", width=30)
password = passInput.get()
```

```
encryptButton = Tkinter.Button(frame2, text="       ENCRYPT      ",
command=encrypt_the_file)

encryptButton.configure(bg='turquoise4', fg='white', activebackground='coral')

decryptButton = Tkinter.Button(frame2, text="       DECRYPT      ",
command=decrypt_the_file)

decryptButton.configure(bg='turquoise4', fg='white', activebackground='coral')


l1.pack(side=LEFT, padx=5, pady=0, fill=X)

passInput.pack(side=RIGHT, padx=5, pady=0, fill=X)

encryptButton.pack(side=LEFT, padx=10, pady=0, fill=X)

decryptButton.pack(side=RIGHT, padx=10, pady=0, fill=X)

root.mainloop()
```