PROJECT REPORT

on

VISUAL CRYPTOGRAPHY

(CSE V Semester Mini project PCS-604)

2020-2021

Submitted to:

Mr.Samir Rana

(CC-CSE-J-V-Sem)

Guided by:

Mr. Indrajeet Sir

(Resource Person)

Submitted by:

Mr. Virender Kumar

Roll. No.: 1918805

CSE-D-VI-Sem

Session: 2020-2021

DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

GRAPHIC ERA HILL UNVERSITY, DEHRADUN

CERTIFICATE

Certified that Mr. Virender Kumar (Roll No.- 1918805) has developed mini project on "Visual Cryptography" for the CS V Semester Mini Project Lab (PCS-504) in Graphic Era Hill University, Dehradun. The project carried out by Students is their own work as best of my knowledge.

Date:

(Mr. Samir Rana)	(Mr. Indrajeet Sir)
Class Co-ordinator	**Project Guide**
CSE-J-V-Sem	Resource Person
(CSE Department)	(CSE Department)
GEHU Dehradun	GEHU Dehradun

ACKNOWLEDGMENT

I would like to express our gratitude to The Almighty God, the most Beneficent and the most Merciful, for completion of project.

I would wish to thank our parents for their continuing support and encouragement. We also wish to thank them for providing us with the opportunity to reach this far in our studies.

I would like to thank particularly our project Co-ordinator Mr. Samir Rana and our Project Guide Mr. Indrajeet Sir for his patience, support and encouragement throughout the completion of this project and having faith in us.

At last but not the least We greatly indebted to all other persons who directly or indirectly helped us during this work.

Mr. Virender Kumar

Roll No.- 1918805

CSE-J-V-Sem

Session: 2020-2021

GEHU, Dehradun

TABLE OF CONTENTS

CHAPTER 1

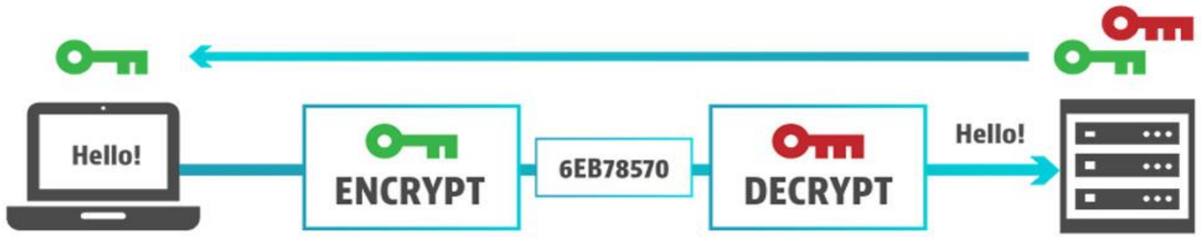
INTRODUCTION

1.1 ABOUT PROJECT

Implementation of Image Encryption and decryption using Advanced Encryption Standard (AES).

These In today's world data security is the major problem which is to be face. In order to secure data during communication, data storage and transmission we use Advance encryption standard(AES). AES is a symmetric block cipher intended to replace DES for commercial applications.it uses 128-bit block size and a key size of 128, 192, or 256 bits. The AES algorithms use to secure data from unauthorized user. The available AES algorithm is used for text data as well as for image data. In this paper an image is given as input to AES encryption algorithm which gives encrypted output. This encrypted output is given as input to AES decryption algorithm and original image is regained as output. The AES algorithm for image encryption and decryption which synthesizes and simulated with the help of python in pyCharm IDE

Fig1.1



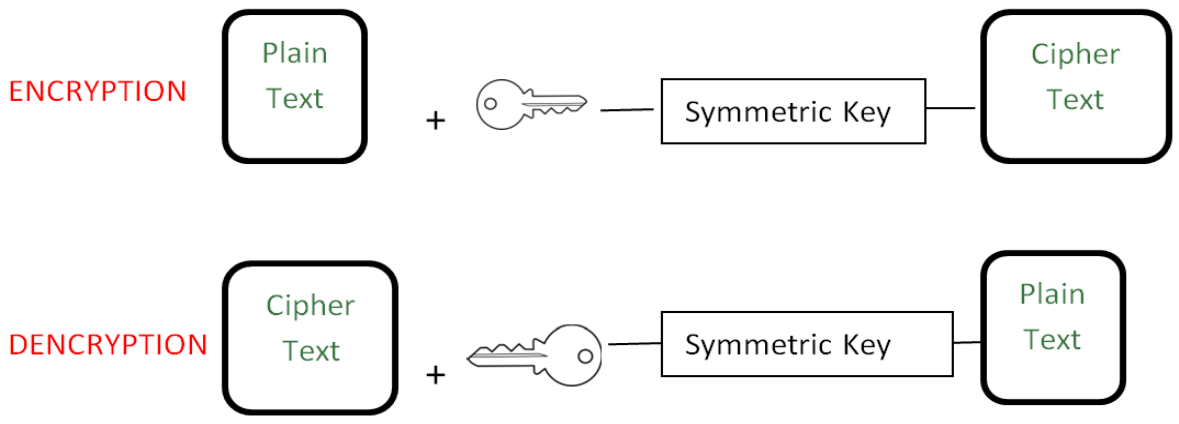
1.2 VISUAL CRYPTOGRAPHY

The Internet is the fastest growing communication medium and essential part of the infrastructure, nowadays. To cope with the growth of internet it has become a constant struggle to keep the secrecy of information and when profits are involved, protect the copyright of data. To provide secrecy and copyright of data, many of the steganographic techniques have been developed. But each of the technique has their respective pros and cons. Where one technique lacks in payload capacity, the other lacks in robustness. So, the main emphasis of cryptography is to overcome these shortcomings.

1.2.1 CRYPTOGRAPHY

The word cryptography is derived from two Greek words which mean **“secret writing”**. Cryptography is the process of scrambling the original text by rearranging and substituting the original text, arranging it in a seemingly unreadable format for others.

FIG 1.2.1

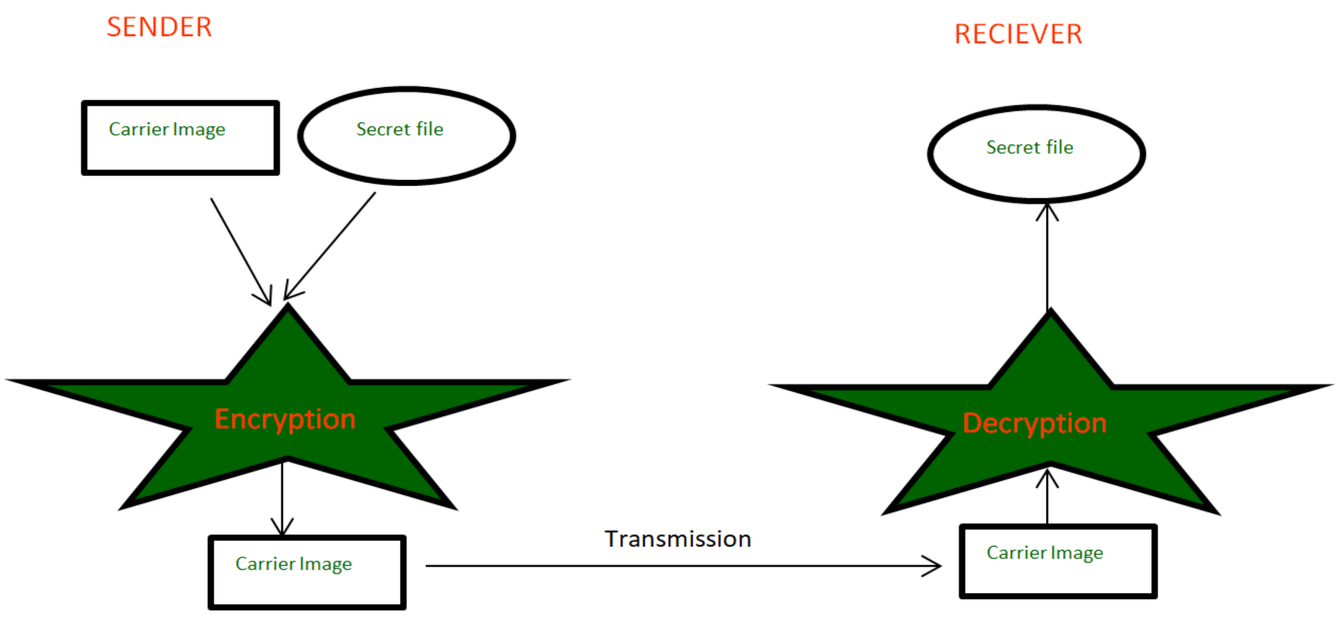


Cryptography is an effective way to protect the information that is transmitting through the network communication path.

1.2.2 VISUAL CRYPTOGRAPHY

Visual cryptography is a cryptographic technique which allows visual information (pictures, text, etc.) to be encrypted in such a way that decryption can be done just by sight reading. Visual cryptography, degree associated rising cryptography technology, uses the characteristics of human vision to rewrite encrypted photos. Visual cryptography provides secured digital transmission that is used just for merely the once.

FIG 1.2.2



Numerous guidance like military maps and business identifications are transmitted over the internet. Whereas pattern secret photos, security problems ought to be compelled to be taken into thought as a result of hackers may utilize weak link over the communication network to steal info that they need. To touch upon the protection problems with secret photos, varied image secret sharing schemes are developed. anyone will use it for coding with none science information and any computations.

1.3 IMAGE ENCRPTION /DECRYPTION METHOD

Algorithm: Image Encryption.
Input: Hided Image.
Output: Encrypted Image.

- **Step 1:** An input image will be selected. It must be an RGB image.
- **Step 2:**Red, Green and blue Channels are separated from an input Image.
- **Step 3:** Each Channel is then further encrypted into 8 shares. This encryption will depend on key used.
- **Step 4:** From Step 3, we get 24 shares, it means each channel has 8 shares each. These 8 shares of an each channel then further compress to 3 shares. Thus we get an o/p of 9 shares at step 4.
- **Step 5:** Compress 3 Shares from step 4 to one final encrypted image.

Algorithm: Image Decryption.
Input: Final Encrypted Image.
Output: Decrypted Image.

- **Step 1:** Select an Encrypted Image. It must be RGB Image.
- **Step 2:** Separate Red, Green and Blue Channels from an Encrypted image.
- **Step 3:** Create 3 Shares from each channel. So at step 3, 9 Encrypted images will be the output.
- **Step 4:** Create 8 Channels from Each channel.
- **Step 5:** From 8 shares each of step 4, Create 3 Shares (i.e red, green and Blue each).
- **Step 6:** Compress Step 5 Images to Plain Image (Decrypted Image).

1.3.1 APPLICATIONS :

- Secret Communication
- Copyright Protection
- Document Authentication
- Secret data storing

1.4 AES(ADVANCE ENCRYPTION SYSTEM)

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.

Every encryption and decryption process has two aspects: the algorithm and the key use for the encryption and decryption. However, it is the key used for encryption and decryption that makes the process of cryptography secure. There are two types of cryptographic mechanisms: symmetric key cryptography in which the same key is use for encryption and decryption. In case of asymmetric key cryptography two different keys are used for encryption and decryption. Symmetric key algorithm is much faster and easier to implement and required less processing power as compare to asymmetric key algorithm.

The advance encryption standard (AES) specifies a federal information processing standards publication (FIPS) approved cryptographic algorithm that can be used to protect electronic data. It was publish by National Institute of Standard and Technology (NIST) in 2001 developed by Joan Daemen and Vincent Rijmen, an algorithm called Rijdae.

1.4.1 AES SPECIFICATION

AES algorithm is of three types i.e. AES-128, AES-192 and AES-256. This classification is done on the bases of the key used in the algorithm for encryption and decryption process. The numbers represent the size of key in bits. This key size determines the security level as the size of key increases the level of security increases. The AES algorithm uses a round function that is composed of four different byte-oriented transformations.

For encryption purpose four rounds consist of:

- Substitute byte
- Shift row
- Mix columns
- Add round key

While the decryption process is the reverse process of the encryption which consists of:

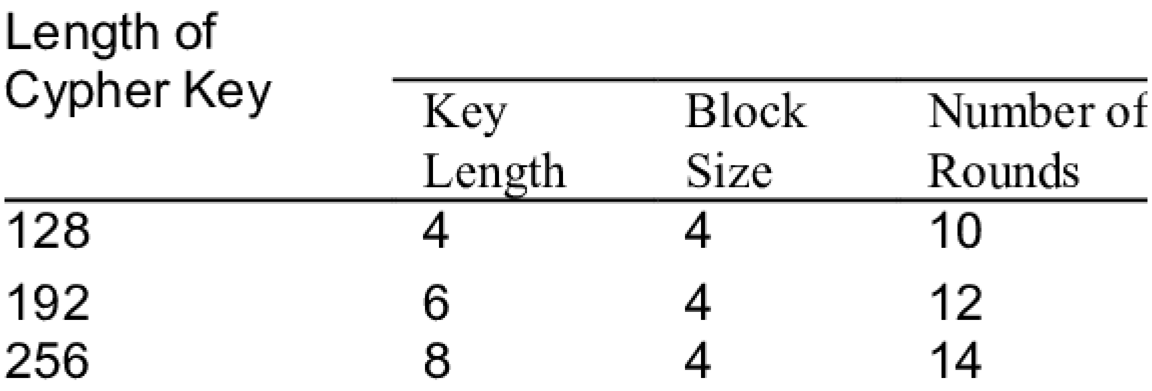
• Inverse shift row

• Inverse substitute byte

• Add round key

• Inverse mix columns

There is a number of round present of key and block in the algorithm. The number of rounds depends on the length of key use for Encryption and Decryption.

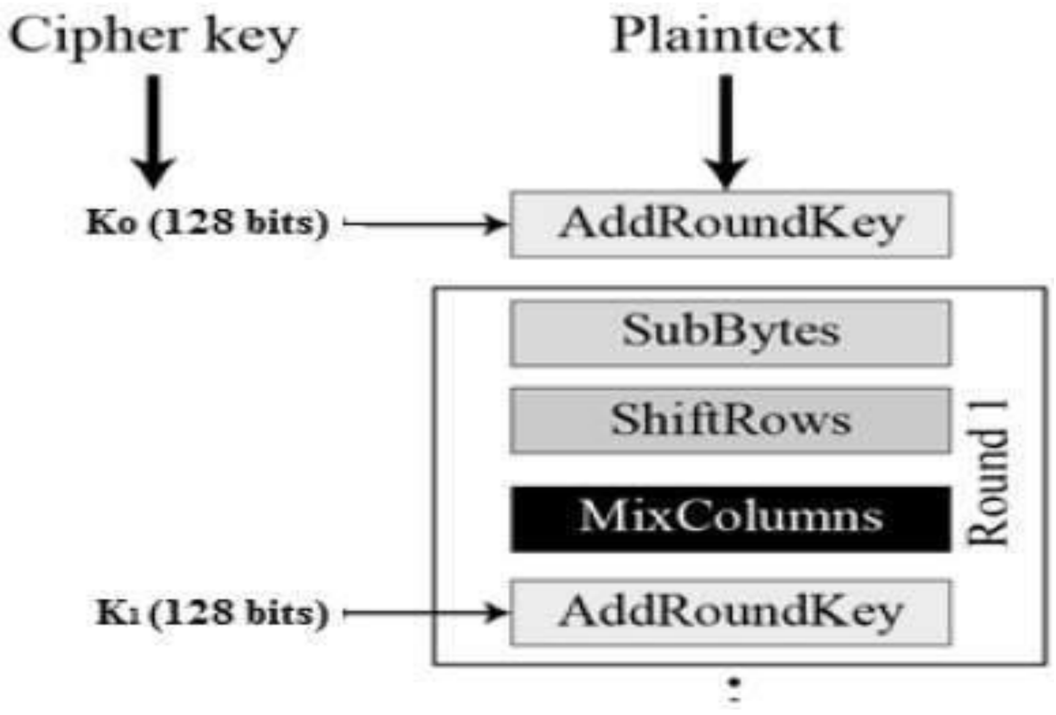


Length of Cypher Key	Key Length	Block Size	Number of Rounds
128	4	4	10
192	6	4	12
256	8	4	14

AES algorithm uses a round function for both its Cipher and Inverse Cipher. This function is composed of four different byte-oriented transformations

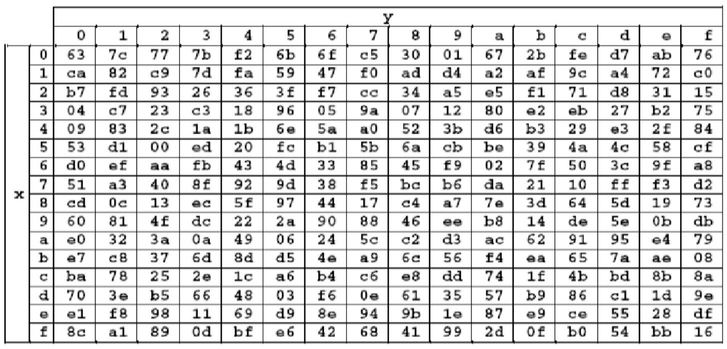
1.4.1 ENCRYPTION PROCESS

Here, we restrict to description of a typical round of AES encryption. Each round comprise of four sub-processes. The first round process is depicted below:-



Substitute Byte Transformation :

The Substitute bytes transformation is a non-linear byte substitution that operates independently on each byte of the State using a substitution table S-box.



x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

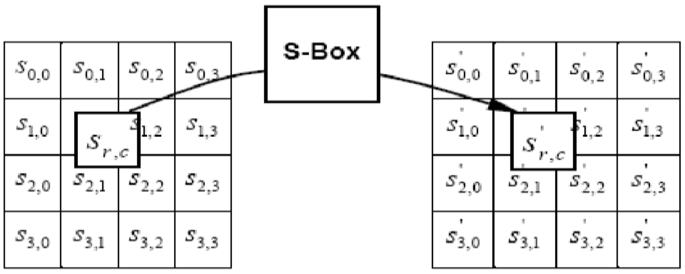
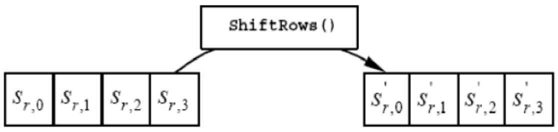
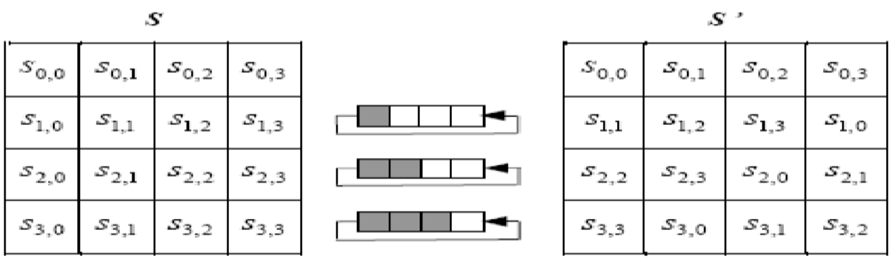

S-Box
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{r,c}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{r,c}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

Fig shows Operation of substitute byte

Shift rows transformation:

In the Shift Rows transformation, the bytes in the last three rows of the State are cyclically shifted over different numbers of bytes. The first row, r = 0, is not shifted. This has the effect of moving bytes to "lower" positions in the row while the "lowest" bytes wrap around into the "top" of the row.


ShiftRows()
$s_{r,0}$ $s_{r,1}$ $s_{r,2}$ $s_{r,3}$
$s'_{r,0}$ $s'_{r,1}$ $s'_{r,2}$ $s'_{r,3}$

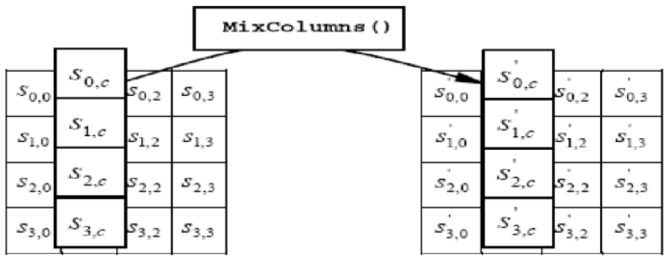

S
S '
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,1}$ $s_{1,2}$ $s_{1,3}$ $s_{1,0}$
$s_{2,2}$ $s_{2,3}$ $s_{2,0}$ $s_{2,1}$
$s_{3,3}$ $s_{3,0}$ $s_{3,1}$ $s_{3,2}$

Mix columns transformation

The Mix Columns transformation operates on the State column-by-column, treating each column as a four-term polynomial. The columns are considered as polynomials over GF(2^8) and multiplied modulo x 4 + 1 with a fixed polynomial a(x), given by

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} .$$

The resultant columns are shown in the figure below. This is the operation of mix columns.

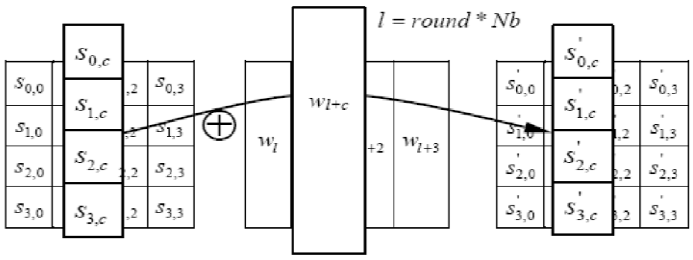

MixColumns()
$s_{0,0}$ $s_{0,c}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,c}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,c}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,c}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,c}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,c}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,c}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,c}$ $s'_{3,2}$ $s'_{3,3}$

Mix Column Operation

Add Round Rey Transformation :

In the Add Round Key transformation, a Round Key is added to the State by a simple bitwise XOR operation. The Round Key is derived from the Cipher key by means of key schedule process. The State and Round Key are of the same size and to obtain the next State an XOR operation is done per element:

$$b(i, j) = a(i, j) \oplus k(i, j)$$

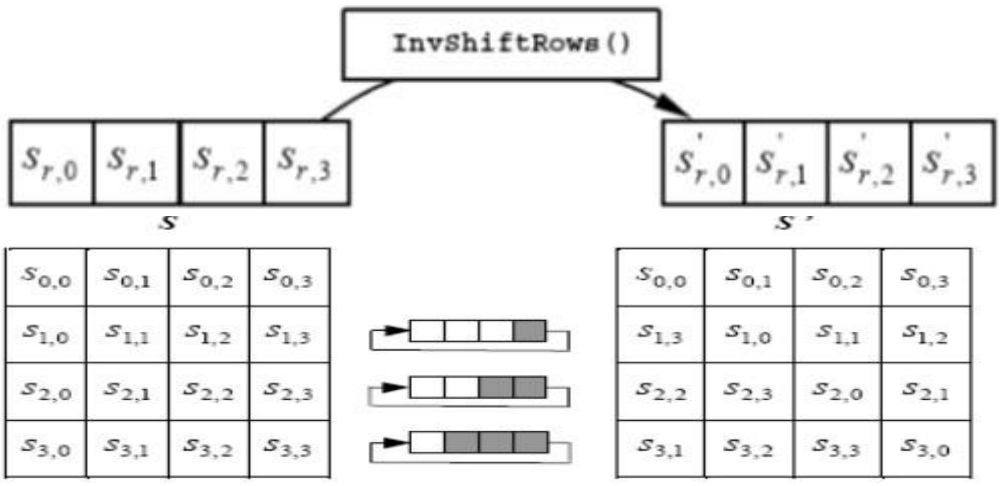

$l = round * Nb$
$s_{0,0}$ $s_{0,c}$ $s_{0,3}$
$s_{1,0}$ $s_{1,c}$ $s_{1,3}$
$s_{2,0}$ $s_{2,c}$ $s_{2,3}$
$s_{3,0}$ $s_{3,c}$ $s_{3,3}$
$\oplus$
w_l w_{l+c} w_{l+3}
$s'_{0,0}$ $s'_{0,c}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,c}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,c}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,c}$ $s'_{3,3}$

add round key operation

1.4.2 DECRYPTION PROCESS

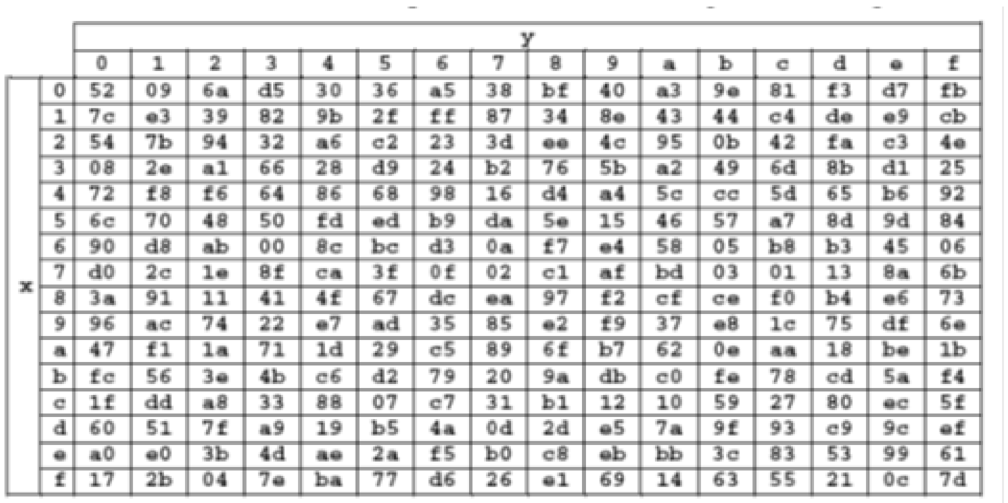
Inverse shift row transformation :

Inverse Shift Rows is the inverse of the Shift Rows transformation. The bytes in the last three rows of the State are cyclically shifted over different numbers of bytes. The first row, r = 0, is not shifted. The bottom three rows are cyclically shifted by Nb-shift(r, Nb) bytes, where the shift value shift(r,Nb) depends on the row number


InvShiftRows()
$s_{r,0}$ $s_{r,1}$ $s_{r,2}$ $s_{r,3}$
s
$s'_{r,0}$ $s'_{r,1}$ $s'_{r,2}$ $s'_{r,3}$
s'
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,3}$ $s_{1,0}$ $s_{1,1}$ $s_{1,2}$
$s_{2,2}$ $s_{2,3}$ $s_{2,0}$ $s_{2,1}$
$s_{3,1}$ $s_{3,2}$ $s_{3,3}$ $s_{3,0}$

Inverse Substitute Byte Transformation:

Inverse Substitute Bytes is the inverse of the byte substitution transformation, in which the inverse S-box is applied to each byte of the State. It is reverse process of Substitute byte transform. This is obtained by applying the inverse of the affine transformation followed by taking the multiplicative inverse in GF (2^8). There is an inverse s-box table for substitute the value.



		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

—

Inverse Mix Columns Transformation:

Inverse Mix Columns is the inverse of the Mix Columns transformation. Inverse Mix Columns operates on the State column-by-column, treating each column as a four-term polynomial. The columns are considered as polynomials over GF(2^8) and multiplied modulo x^ 4 + 1 with a fixed polynomial (x), given by

$$a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}$$

1.4.3 AES ANALYSIS

In present day cryptography, AES is widely adopted and supported in both hardware and software. Till date, no practical cryptanalytic attacks against AES has been discovered. Additionally, AES has built-in flexibility of key length, which allows a degree of 'future-proofing' against progress in the ability to perform exhaustive key searches.

However, just as for DES, the AES security is assured only if it is correctly implemented and good key management is employed.

CHAPTER 2
PROJECT

2.1 REQUIREMENT ANALYSIS

Python 3 or Above

Any IDE like PyCharm , Jupiter Notebook , Anaconda

We can also use the Visual studio or any other platform to run the code which supports the python language.

2.2 SOFTWARE SPECIFICATION

PyCharm is the most popular IDE used for Python scripting language.

Install python
Install PyCharm ide

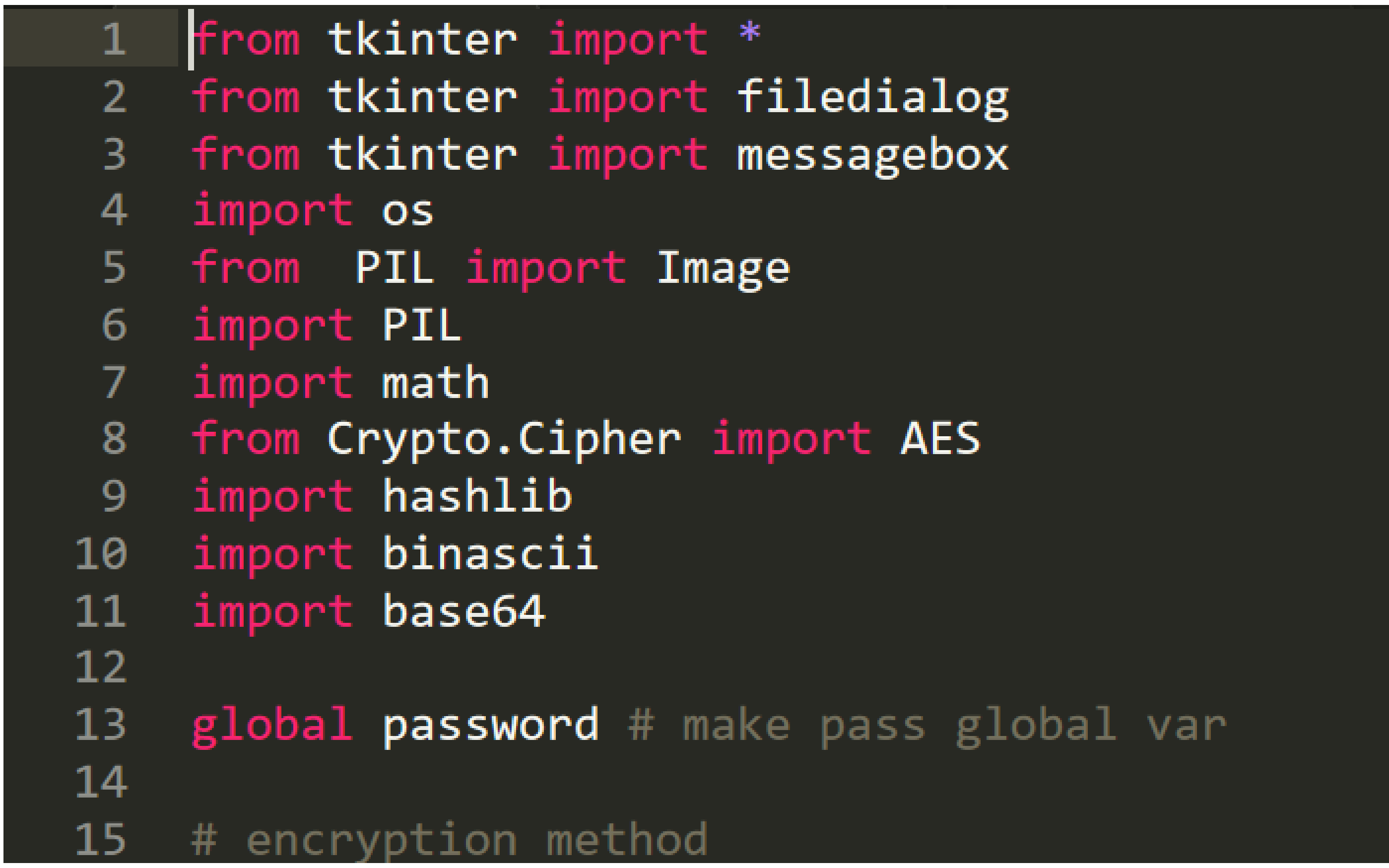
After that install all the libraries which will used for project.

Libraries used:

Tkinter (GUI)	hashlib
Binascii	math
Image	base64
PIL	os
Crypto	messagebox,filedialog

CHAPTER 3

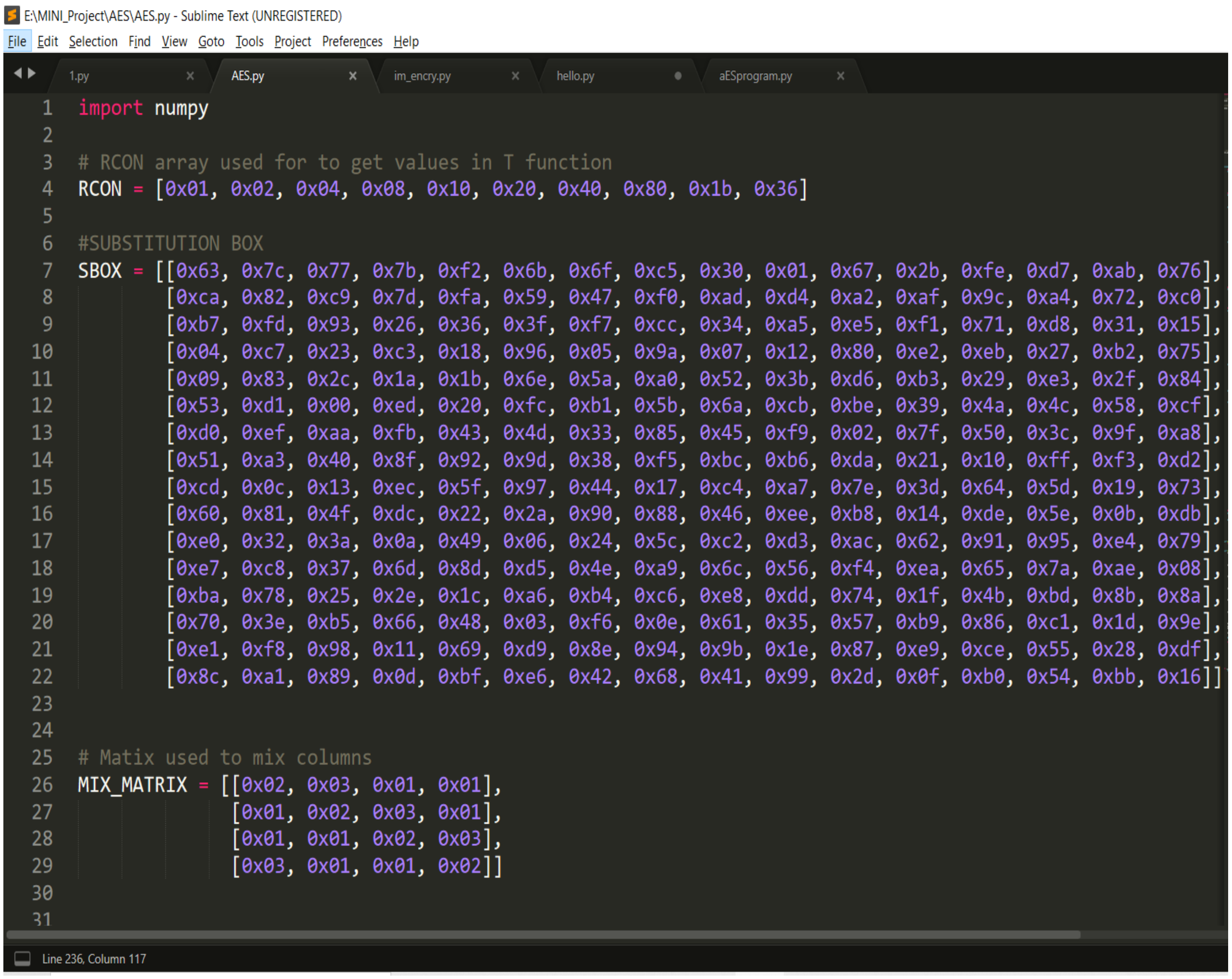
SNAPSHOT OF PROJECT



```
from tkinter import *
from tkinter import filedialog
from tkinter import messagebox
import os
from  PIL import Image
import PIL
import math
from Crypto.Cipher import AES
import hashlib
import binascii
import base64

global password # make pass global var

# encryption method
```


E:\MINI_Project\AES\AES.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
1.py
AES.py
im_encry.py
hello.py
aESprogram.py
Line 236, Column 117

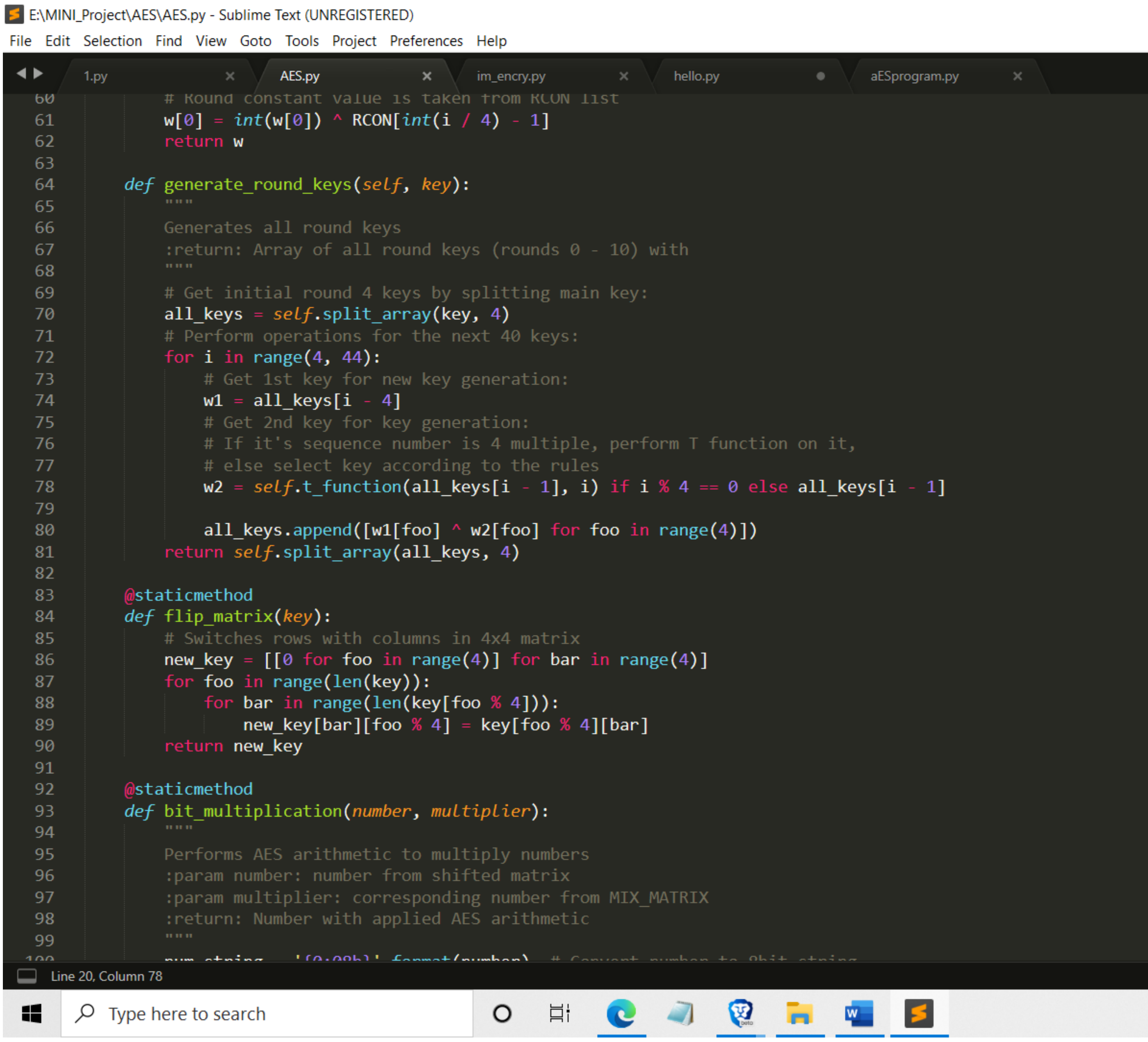
```python
import numpy

# RCON array used for to get values in T function
RCON = [0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36]

#SUBSTITUTION BOX
SBOX = [[0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76],
        [0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0],
        [0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15],
        [0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75],
        [0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84],
        [0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf],
        [0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8],
        [0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2],
        [0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73],
        [0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb],
        [0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79],
        [0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08],
        [0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a],
        [0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e],
        [0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf],
        [0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16]]


# Matix used to mix columns
MIX_MATRIX = [[0x02, 0x03, 0x01, 0x01],
              [0x01, 0x02, 0x03, 0x01],
              [0x01, 0x01, 0x02, 0x03],
              [0x03, 0x01, 0x01, 0x02]]

```


E:\MINI_Project\AES\AES.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
1.py
AES.py
im_encry.py
hello.py
aESprogram.py
Line 20, Column 78
Type here to search

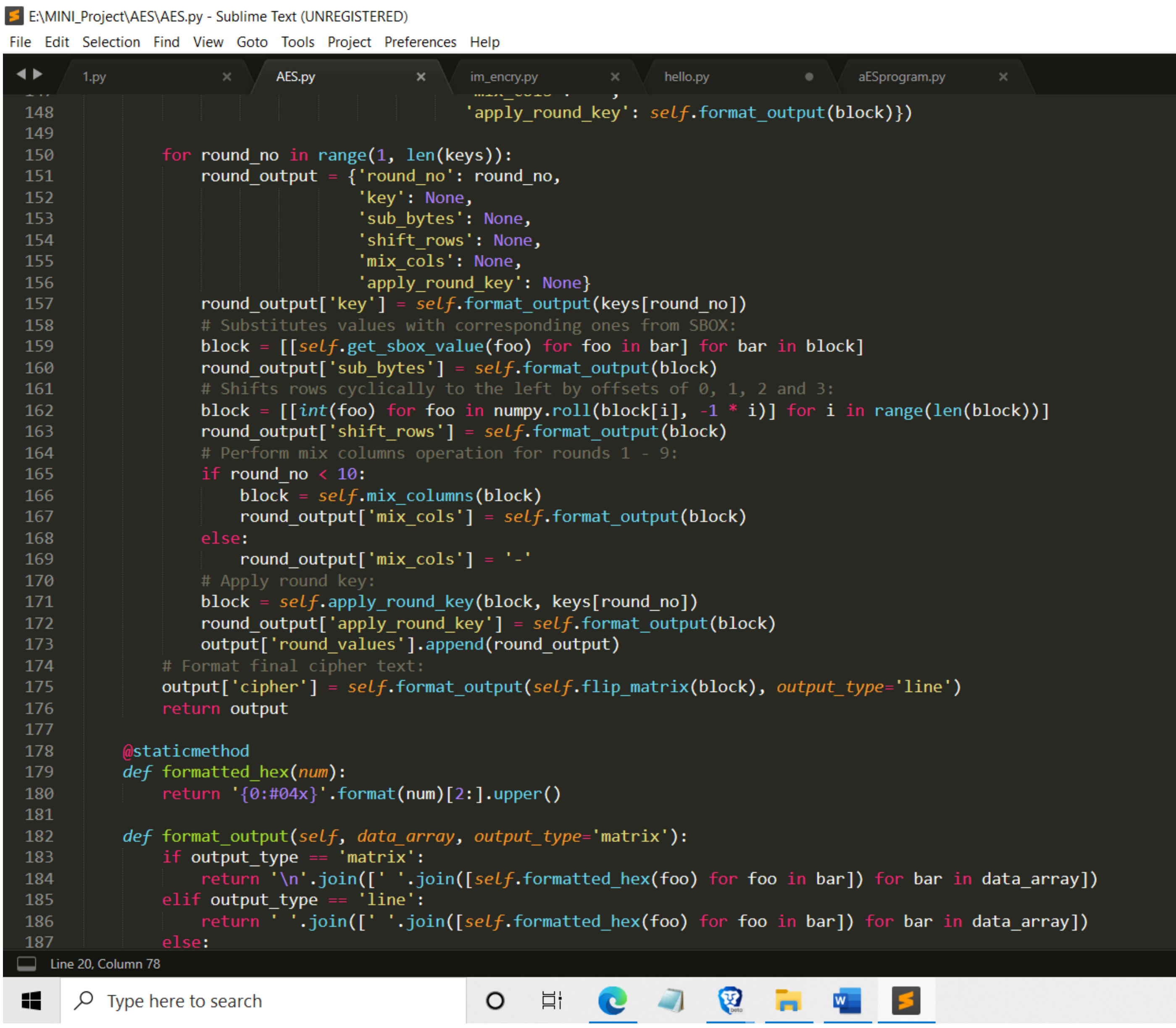
```python
        # Round constant value is taken from RCON list
        w[0] = int(w[0]) ^ RCON[int(i / 4) - 1]
        return w

    def generate_round_keys(self, key):
        """
        Generates all round keys
        :return: Array of all round keys (rounds 0 - 10) with
        """
        # Get initial round 4 keys by splitting main key:
        all_keys = self.split_array(key, 4)
        # Perform operations for the next 40 keys:
        for i in range(4, 44):
            # Get 1st key for new key generation:
            w1 = all_keys[i - 4]
            # Get 2nd key for key generation:
            # If it's sequence number is 4 multiple, perform T function on it,
            # else select key according to the rules
            w2 = self.t_function(all_keys[i - 1], i) if i % 4 == 0 else all_keys[i - 1]

            all_keys.append([w1[foo] ^ w2[foo] for foo in range(4)])
        return self.split_array(all_keys, 4)

    @staticmethod
    def flip_matrix(key):
        # Switches rows with columns in 4x4 matrix
        new_key = [[0 for foo in range(4)] for bar in range(4)]
        for foo in range(len(key)):
            for bar in range(len(key[foo % 4])):
                new_key[bar][foo % 4] = key[foo % 4][bar]
        return new_key

    @staticmethod
    def bit_multiplication(number, multiplier):
        """
        Performs AES arithmetic to multiply numbers
        :param number: number from shifted matrix
        :param multiplier: corresponding number from MIX_MATRIX
        :return: Number with applied AES arithmetic
        """
        num_string = '{0:08b}'.format(number)  # Convert number to 8bit string
```


E:\MINI_Project\AES\AES.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
1.py
AES.py
im_encry.py
hello.py
aESprogram.py
Line 20, Column 78
Type here to search

```
                                'apply_round_key': self.format_output(block)})

        for round_no in range(1, len(keys)):
            round_output = {'round_no': round_no,
                            'key': None,
                            'sub_bytes': None,
                            'shift_rows': None,
                            'mix_cols': None,
                            'apply_round_key': None}
            round_output['key'] = self.format_output(keys[round_no])
            # Substitutes values with corresponding ones from SBOX:
            block = [[self.get_sbox_value(foo) for foo in bar] for bar in block]
            round_output['sub_bytes'] = self.format_output(block)
            # Shifts rows cyclically to the left by offsets of 0, 1, 2 and 3:
            block = [[int(foo) for foo in numpy.roll(block[i], -1 * i)] for i in range(len(block))]
            round_output['shift_rows'] = self.format_output(block)
            # Perform mix columns operation for rounds 1 - 9:
            if round_no < 10:
                block = self.mix_columns(block)
                round_output['mix_cols'] = self.format_output(block)
            else:
                round_output['mix_cols'] = '-'
            # Apply round key:
            block = self.apply_round_key(block, keys[round_no])
            round_output['apply_round_key'] = self.format_output(block)
            output['round_values'].append(round_output)
        # Format final cipher text:
        output['cipher'] = self.format_output(self.flip_matrix(block), output_type='line')
        return output

    @staticmethod
    def formatted_hex(num):
        return '{0:#04x}'.format(num)[2:].upper()

    def format_output(self, data_array, output_type='matrix'):
        if output_type == 'matrix':
            return '\n'.join([' '.join([self.formatted_hex(foo) for foo in bar]) for bar in data_array])
        elif output_type == 'line':
            return ' '.join([' '.join([self.formatted_hex(foo) for foo in bar]) for bar in data_array])
        else:
```

CHAPTER 4
CONCLUSION

4.1 CONCLUSION

Image Encryption and Decryption using AES algorithm is implemented to secure the image data from an unauthorized access. A Successful implementation of symmetric key AES algorithm is one of the best encryption and decryption standard available in market. With the help of python coding implementation of an AES algorithm is synthesized and simulated for Image Encryption and Decryption. The original images can also be completely reconstructed without any distortion. It has shown that the algorithms have extremely large security key space and can withstand most common attacks such as the brute force attack, cipher attacks and plaintext attacks.

REFERENCE

1. William Stallings, “Advance Encryption Standard,” in Cryptography and Network Security, 4th Ed., India:PEARSON,pp. 134–165
2. AtulKahate, “Computer-based symmetric key cryptographic algorithm”, in Cryptography and Network Security, 3th Ed. New Delhi:McGraw-Hill, pp. 130-141..
3. WEBSITES

   Geekforgeeks

   Tutorialspoint

```
}
```