CRYPTOGRAPHY

Akash Poudel
210047
11491817

Practical Cryptography
ST6051CEM

Content

Abstract

Basic encryption of an image. First, the image to be encrypted is parsed as a binary and then an alphanumeric key is taken from the user. SHA256 hashing is used to increase the security of data by creation of a checksum purposed to represent private information. The process works by passing information as input to a hash function and using a returned hash string to represent the encrypted key. We then create the AES cipher and use it along with the hash string to encrypt the image.

The block encryptor may be utilized as a stream cipher thanks to the AES cipher's CFB (Cipher FeedBack) method of operation. Also, for AES encryption using pycrypto, we had to ensure that the data is a multiple of 16-bytes in length. Pad the buffer if it is not and include the size of the data at the beginning of the output, so the receiver can decrypt properly.

Introduction

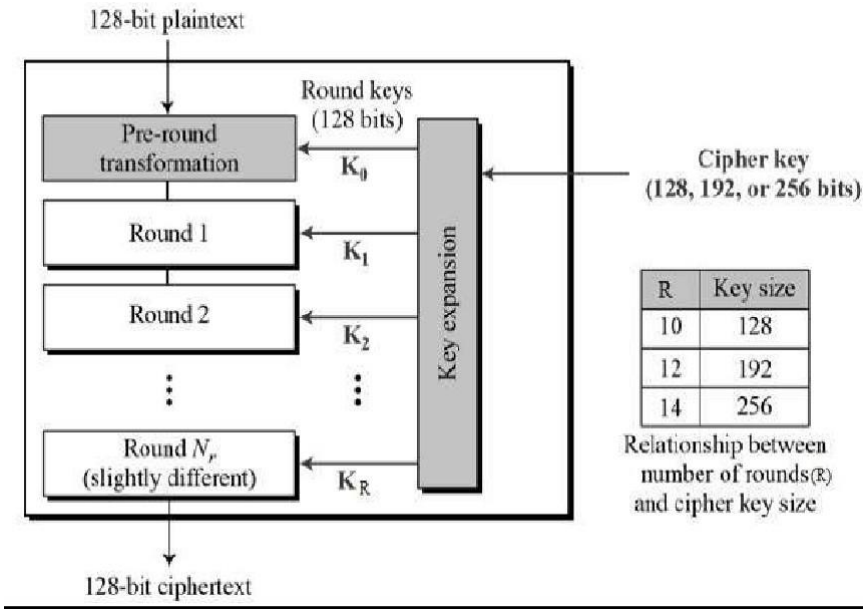
AES is a block encryption method that is often used. Back in 2001, five modes of operation of the AES algorithm were standardized: ECB (Electronic Code Book), CBC (Cipher Block Chaining), CFB (Cipher FeedBack), OFB (Output FeedBack) and CTR (Counter).The block ciphers are schemes for encryption or decryption where a block of plaintext is treated as a single block and is used to obtain a block of ciphertext with the same size. AES (Advanced Encryption Standard) is an among the most popular block encryption techniques today. It has been standardized by the NIST (National Institute of Standards and Technology) in 2001, in order to replace DES and 3DES which were used for encryption in that period.
AES blocks are 128 bits in size, but encryption keys might be 128, 192, or 256 bits in size. In each of the stages of encryption, four functions are applied: substitution of bytes, permutation, arithmetic operations over finite fields and an XOR operation with the encryption key. The size of the AES block provides efficiency, but also sufficient security. It is found at least six time faster than triple DES.

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

The features of AES are as follows −

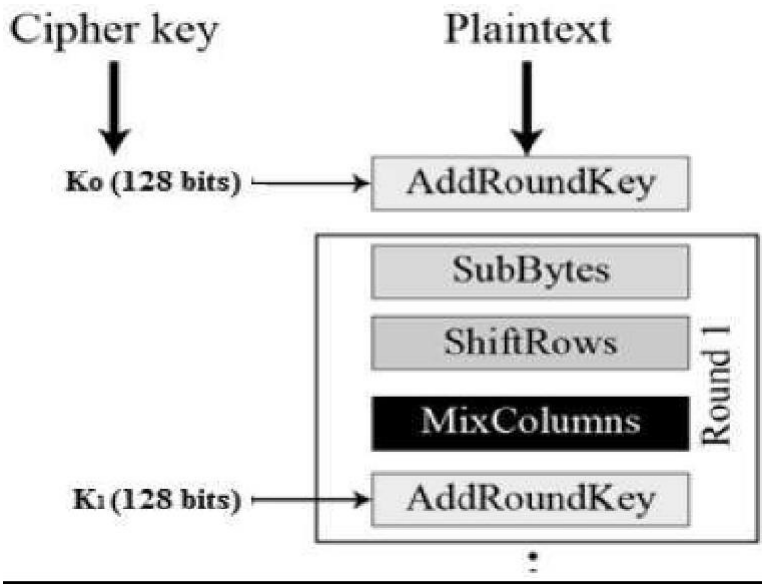
- Symmetric key symmetric block cipher
- 128-bit data, 128/192/256-bit keys
- Stronger and faster than Triple-DES
- Provide full specification and design details
- Software implementable in C and Java

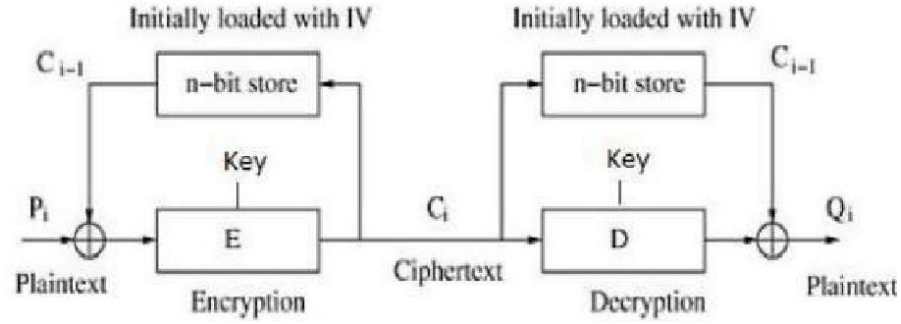
Block Diagram And Design

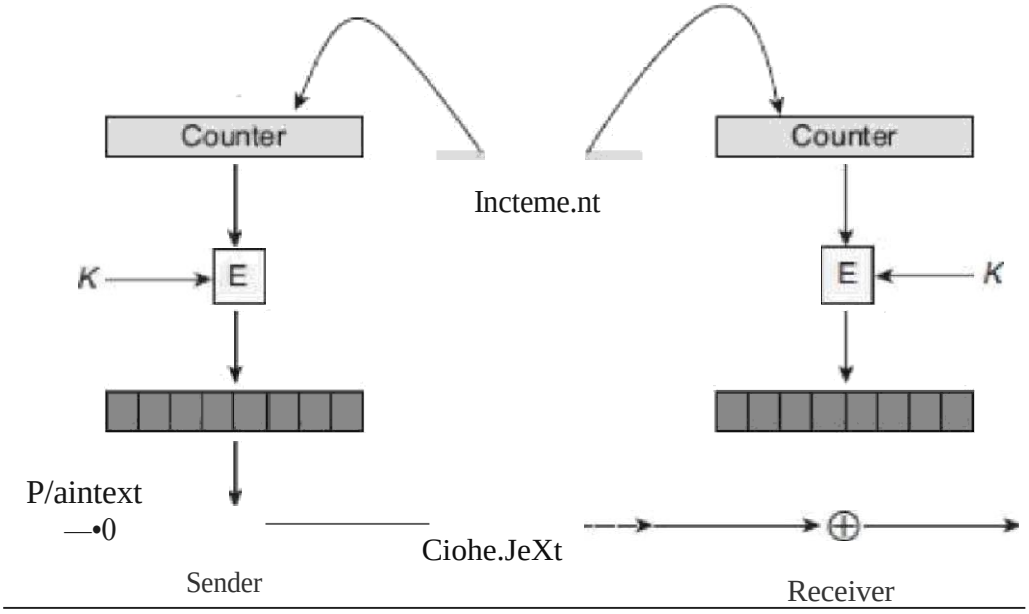


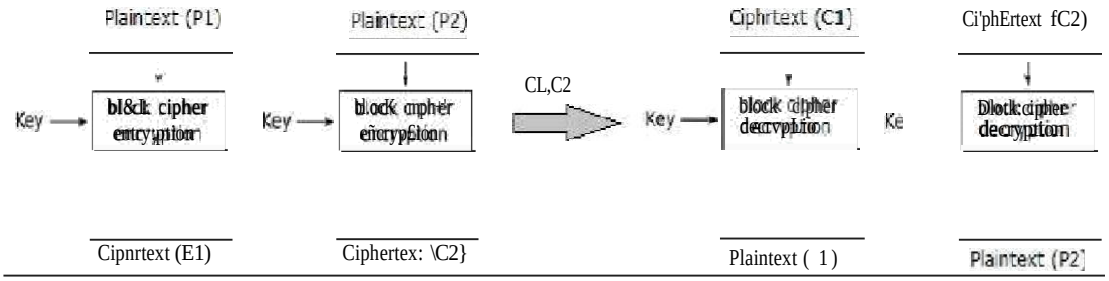
R	Key size
10	128
12	192
14	256

Relationship between number of rounds(R) and cipher key size






Counter
Incteme.nt
Counter
K
E
E
K
P/aintext
—•0
Ciohe.JeXt
Sender
Receiver


Plaintext (P1)
Plaintext (P2)
Ciphrtext (C1)
Ci'phErtext fC2)
Key
Key
CL,C2
Key
Ke
Cipnrtext (E1)
Ciphertex: \C2}
Plaintext (1)
Plaintext (P2)

AES-Algorithm

AES algorithm is of three types i.e. AES-128, AES-192 and AES-256. The key utilized for the encryption and decryption procedure is the basis for this categorization. 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 .

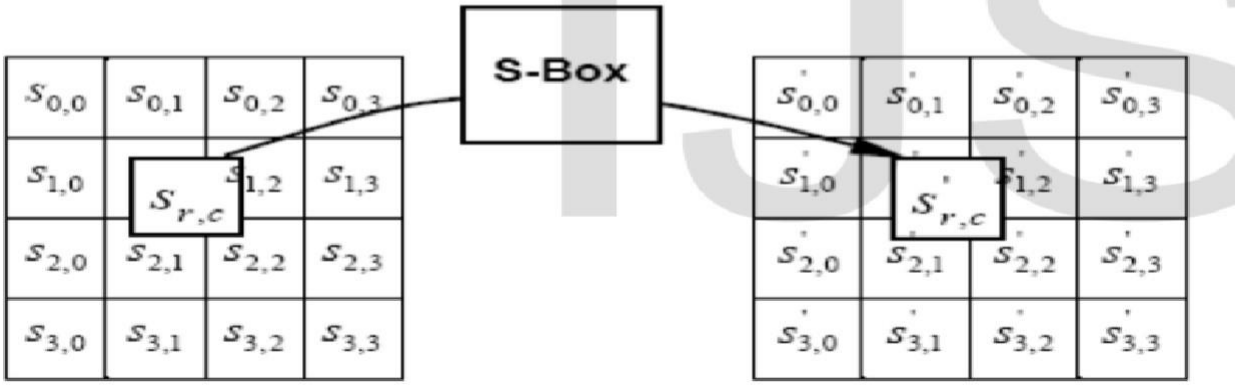
There is a number of round present of key and block in the algorithm. The number of rounds depends on the length of key use for Encryption and Decryption. AES algorithm uses a round function for both its Cipher and Inverse Cipher. This function is composed of four different byte- oriented transformations.

1. Encryption process

1.1 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. The operation of substitute byte is shown in figure 1.

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



1.2 Shift rows transformation

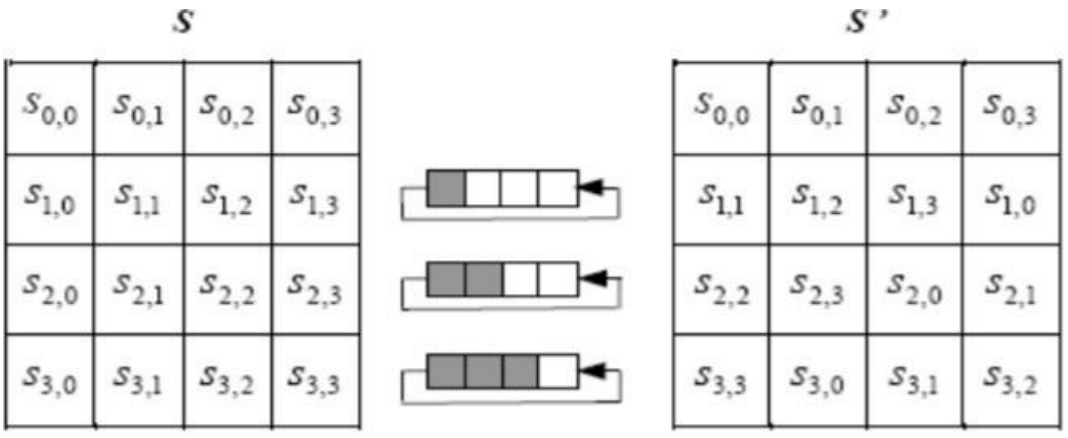
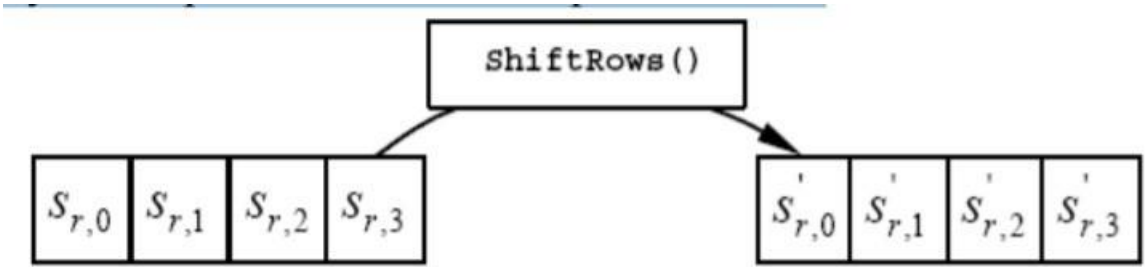


Figure.2.Cyclic shift row operation



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.

1.3 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 result of columns is shown in the figure below. This is the operation of mix columns.

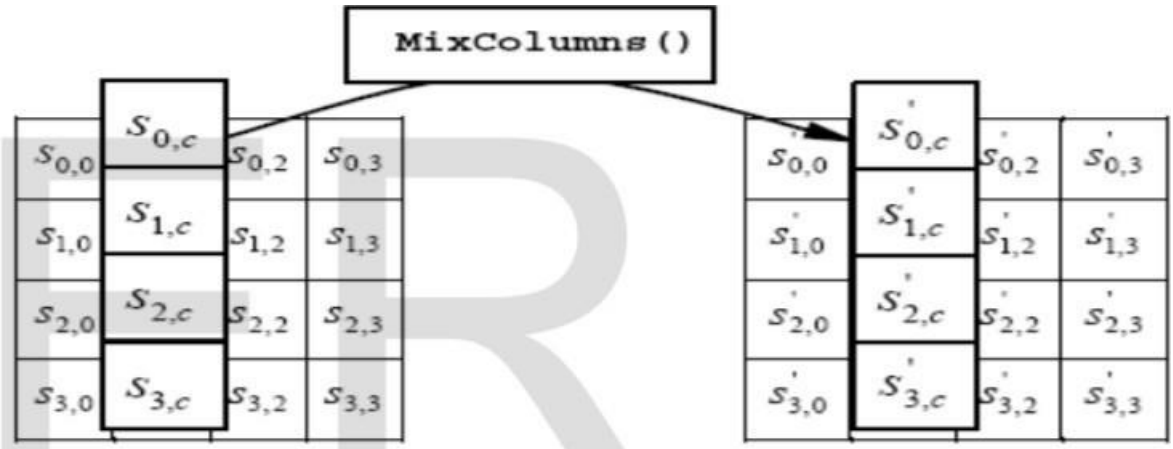


Figure.3.Mix columns operation

1.4 Add round key 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 andRound 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) ⊕ k (i, j)

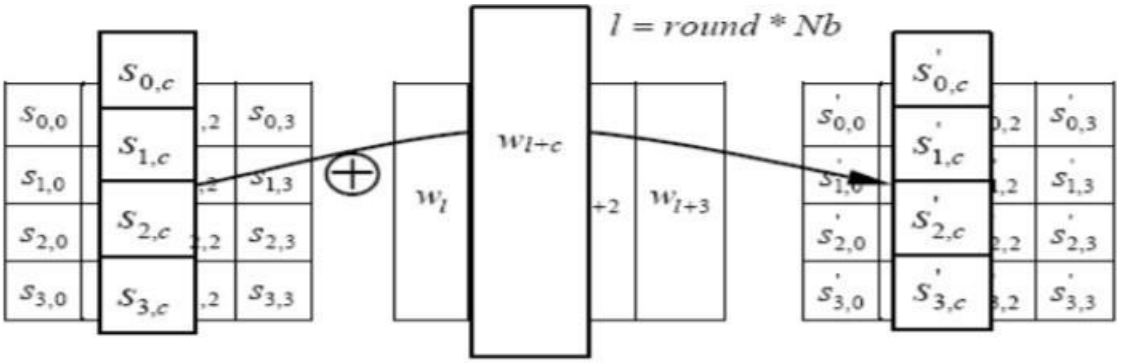
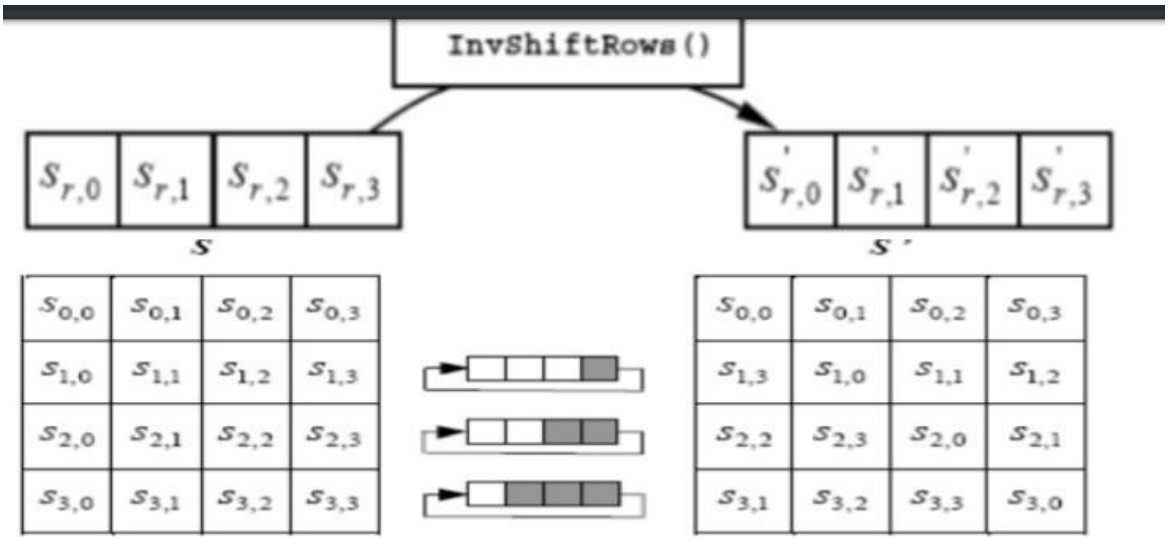


Figure 4. Add round key operation

2) Decryption Process



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.

2.2 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

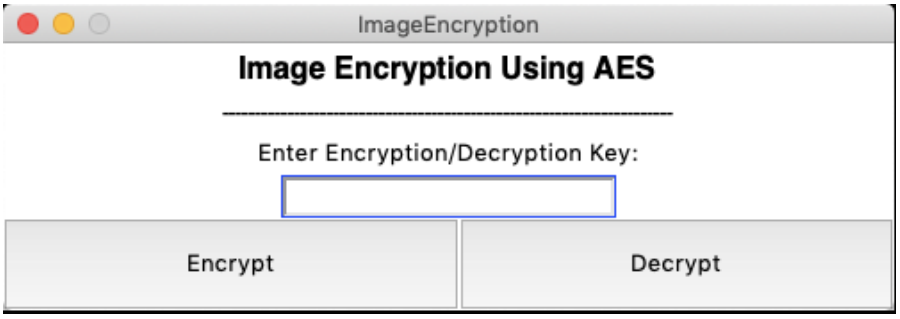
2.3 Inverse mix columns transformation

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}

Implementation

Encryption-

1. Generating a key:



The user enters the key first and that key is thrown into the hashlib sha256 function,

```
def encrypt():

    global file_path_e
    enc_pass = passg.get()
    if enc_pass == "":
        pass_alert()
    else:
        filename = tkinter.filedialog.askopenfilename()
        file_path_e = os.path.dirname(filename)

        hash=hashlib.sha256(enc_pass.encode())
```

To point data of any size to data of certain size the function is used. The values returned by a hash function are called hash values.

2. Initialization Vector:

```
p = hash.digest()
key = p
iv = p.ljust(16)[:16]
print("Encoding key is: ",key)
```

The digest is the output of the hash function.For example, sha256 has a digest of 256 bits, i.e. its digest has a length of 32 bytes.

An initialization vector (iv) is an arbitrary number that can be used along with a secret key for data encryption. This number, also called a nonce, is employed only one time in any session.

The use of an iv prevents repetition in data encryption, making it more difficult for a hacker using a dictionary attack to find patterns and break a cipher. For example, a sequence might appear twice or more within the body of a message. If there are repeated sequences in encrypted data, an attacker could assume that the corresponding sequences in the message were also identical. The iv prevents the appearance of corresponding duplicate character sequences in the ciphertext.

The ideal iv is a random number that is made known to the destination computer to facilitate decryption of the data when it is received. The iv can be agreed on in advance, transmitted independently or included as part of the session setup prior to exchange of the message data. The length of the iv (the number of bits or bytes it contains) depends on the method of encryption.
The size of the iv is often equivalent to the size of the block or encryption key of the chosen cipher. We can also use:
import base64
from Crypto import Random
iv = Random.new().read(AES.block_size)

4. Loading the image:

```
input_file = open(filename,'rb')
input_data = input_file.read()
input_file.close()
```

Reading the input image as binary. The open() function opens a file in text format by default.....Hence the "rb" mode opens the file in binary format for reading.

5. Encrypting with AES:

```
def enc_image(input_data,key,iv,filepath):
    cfb_cipher = AES.new(key, AES.MODE_CFB, iv) # Mode Cipher FeedBack object
    enc_data = cfb_cipher.encrypt(input_data)

    enc_file = open(filepath+"/encrypted.enc", "wb")
    enc_file.write(enc_data)
    enc_file.close()
```

We now create the AES cipher in the CFB mode of operation, at the beginning (at the first block) the encryption (utilizes an encryptor pointed with Encrypt) is handled by using an “iv” and an encryption key “key”. After that, the XOR operation between the encryption result (the output form the encryptor) and the plaintext block (P_1) is performed. For all the other blocks, the encryption is performed over the result of the encryption of the previous blocks accordingly (C_1, C_2, ...). Then an XOR is performed with the corresponding plaintext block (P_2, P_3,.....). In the beginning, the “iv” is placed in a shift register, the size of which can be e.g. 64 bits. The result of the encryption of the “iv” is again 64 bits. But, the XOR is applied to only a few bits s (for example, s=8) of the encrypted “iv” with also s bits from the plaintext P_1. The least significant bits from the iv that will

not be used are discarded. The result C_1 from the XOR operation is then placed at the rightmost position in the shift register from the next block, and the operation is repeated in the same manner. The encryption and decryption operations in the CFB mode of operation are the same operations. Also, an error in one block will propagate to the next block, which is manifested in the process of decryption.

Decryption-

The encryption key used to encrypt the data must also be used to decrypt it. The recipient additionally requires the initialization vector regarding the key. Plain text may be utilized to transmit this; encryption is not necessary.

```
def decrypt():

    global file_path_e
    enc_pass = passg.get()
    if enc_pass == "":
        pass_alert()
    else:
        filename = tkinter.filedialog.askopenfilename()
        file_path_e = os.path.dirname(filename)

        hash=hashlib.sha256(enc_pass.encode())
        p = hash.digest()
        key = p
        iv = p.ljust(16)[:16]
        input_file = open(filename,'rb')
        input_data = input_file.read()
        input_file.close()
        enc_script.dec_image(input_data,key,iv,file_path_e)
```

Source Code

```
“from tkinter import*
from tkinter import ttk
import tkinter as tk
from tkinter.filedialog import *
import tkinter.messagebox from
PIL import Image,ImageTk
import hashlib
import enc_script

def pass_alert():
  tkinter.messagebox.showinfo("Password Alert","Please enter a password.") def

encrypt():

   global file_path_e
   enc_pass = passg.get() if
   enc_pass == "":
      pass_alert()
   else:
      #LOAD THE IMAGE
      filename = tkinter.filedialog.askopenfilename()
      file_path_e = os.path.dirname(filename)

      #GENERATE KEY & INITIALIZATION VECTOR
      hash=hashlib.sha256(enc_pass.encode()) p =
      hash.digest()
      key = p
      iv = p.ljust(16)[:16] print("Encoding
      key is: ",key)

      input_file = open(filename,'rb')
      input_data = input_file.read()
      input_file.close()
      enc_script.enc_image(input_data,key,iv,file_path_e)
      tkinter.messagebox.showinfo("Encryption Alert","Encryption ended successfully. File stored as: rypted.enc")
enc

def decrypt(): global

   file_path_e
   enc_pass = passg.get() if
   enc_pass == "":
      pass_alert()
   else:
      filename = tkinter.filedialog.askopenfilename()
      file_path_e = os.path.dirname(filename)

      hash=hashlib.sha256(enc_pass.encode()) p =
      hash.digest()
      key = p
      iv = p.ljust(16)[:16]
      input_file = open(filename,'rb')
      input_data = input_file.read()
      input_file.close()
      enc_script.dec_image(input_data,key,iv,file_path_e)
```

```
tkinter.messagebox.showinfo("Decryption Alert","Decryption ended successfully File Stored as: output.png")


# GUI STUFF
top=tk.Tk()
top.geometry("500x150")
top.resizable(0,0)
top.title("ImageEncryption")

title="Image Encryption Using AES"
msgtitle=Message(top,text=title)
msgtitle.config(font=('helvetica',17,'bold'),width=300)
msgtitle.pack()

sp="------------------------------------------------------------------ "
sp_title=Message(top,text=sp)
sp_title.config(font=('arial',12),width=650) sp_title.pack()


passlabel = Label(top, text="Enter Encryption/Decryption Key:")
passlabel.pack()
passg = Entry(top, show="*", width=20)
passg.config(highlightthickness=1,highlightbackground="blue") passg.pack()

encrypt=Button(top,text="Encrypt",width=28,height=3,command=encrypt)
encrypt.pack(side=LEFT)
decrypt=Button(top,text="Decrypt",width=28,height=3,command=decrypt)
decrypt.pack(side=RIGHT)

top.mainloop()

from Crypto.Cipher import AES

def enc_image(input_data,key,iv,filepath):
        cfb_cipher = AES.new(key, AES.MODE_CFB, iv) enc_data =
        cfb_cipher.encrypt(input_data)

        enc_file = open(filepath+"/encrypted.enc", "wb")
        enc_file.write(enc_data)
        enc_file.close()


def dec_image(input_data,key,iv,filepath):
        cfb_decipher = AES.new(key, AES.MODE_CFB, iv)
        plain_data = cfb_decipher.decrypt(input_data)

        output_file = open(filepath+"/output.png", "wb")
        output_file.write(plain_data)
        output_file.close()
"
```

5. CONCLUSION

The AES technique is used for picture encryption and decryption to protect the image data from unwanted access. One of the finest encryption and decryption standards now in the industry is the effective execution of the symmetric key AES algorithm. AES algorithm architecture is generated and modeled for image encryption and decryption with the aid of Python scripting. 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.

REFERENCES

[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] Manoj .B,Manjula N Harihar (2012, June). "Image Encryption and Decryption using AES", International Journal of Engineering and Advance Technology (IJEAT) volume-1, issue-5, pp. 290-294.

[4] KundankumarRameshwarSaraf, Vishal PrakashJagtap, Amit Kumar Mishra, (2014, May- June)."Text and Image Encryption Decryption Using Advance Encryption Standard", International Journal of Emerging Trends and Technology in computer science(IJETTCS) volume-3, issue-3, pp. 118-126.

[5] VedkiranSaini, ParvinderBangar, Harjeet Singh Chauhan, (2014, April)."Study and Literature Survey of Advanced Encryption Algorithm for Wireless Application", International Journal of Emerging Science and Engineering (IJESE) volume-2, issue-6, pp.33-37.

[6] Sourabh Singh, Anurag Jain, (2013, May). "An Enhanced Text to Image Encryption Technique using RGB Substitution and AES", International Journal of Engineering Trends and Technology (IJETT) volume-4,issue-5,pp.2108-2112.

[7] R.Gopinath, M.Sowjanya, (2012, October)."Image Encryption for Color Images Using Bit Plane and Edge Map Cryptography Algorithm", International Journal of Engineering Research and Technology (IJERT) volume-1, issue-8, pp.1-4.

[8] Kundankumar R. Saraf,Sunita P. Ugale, "Implementation of text encryption and decryption in Advance Encryption Standard", ASM'S International E-journal of ongoing Research in Management and IT.

[9] Hamdan.O.Alanazi, B.B.Zaidan, A.A.Zaidan, Hamid A.Jalab, M.Shabbir and Y. Al- Nabhani, (2010, March), "New Comparative Study Between DES, 3DES and AES within Nine Factors", Journal of computing,volume2-,issue-3,pp.152-157