AN ADVANCED

ENCRYPTION AND DECRYPTION

Abhay Chaudhary
19BCE7290

With consideration from different Algorithms !

Introduction to Cryptography
(CSE-1007)

15 November 2020

Abhay Chaudhary
(19BCE7290)
A+TA

Dr. Garima Singh
School of Computer Science and Engineering
(SCOPE)

Vellore Institute of Technology, Amravati
2020

All the source code, files, input files, output, videos everything is available on the GitHub for reference as this is a multiple files project.

The link to the GitHub repository is

https://github.com/Abhayindia/Message-Encription

Please have a look at the video for better understanding so as to how the second phase of the project works.

INDEX

Encryption and Decryption of Files and Messages

Abhay Chaudhary
Department of Computer Science and Engineering
Vellore Institute of Technology, Amaravati, AP, India

Abstract:

Cryptography is the science of secret codes. Previously we used DES algorithms in order to secure but cannot encrypt completely. Thus, we referred AES Algorithms to create a ciphertext in encryption and is given as an input in decryption. In present scenario, everyone sharing their data in online using internet also online transactions like e-banking for money transfers, in shopping malls, restaurants, and many more. While transferring a huge amount or any confidential data there are many chances to hack the data. Encryption is one the most effective approach to achieve data security and privacy. The Encryption techniques hide the original content of a data in such a way that the original information is recovered only through using a key known as decryption process. The objective of the encryption is to secure or protect data from unauthorized access in term of viewing or modifying the data. Encryption can be implemented occurs by using some substitute technique, shifting technique, or mathematical operations. Several symmetric key base algorithms have been developed in the past year. In paper an efficient reliable symmetric key based algorithm to encrypt and decrypt the text data has proposed. Send a quick message with simple text encryption to your WhatsApp contact and , basically in ROT13 with new multi encryption based algorithm on Symbols Substitution and ASCII. The other half-design here is to encrypt and decrypt any form of data in any given format based on 256 bit AES algorithm in OFB mode. The proposed method is easy to implement.

Keywords: Encryption, Decryption, Symmetric Method, Key Size and File or Message Size.

INTRODUCTION

1.1. Advanced Encryption Standard (AES) Cipher

In the Advanced Encryption Standards, the Rijndael is a Block cipher, which works on fixed length group of bits, called blocks. An input is taken a certain size, usually 128 bits, the transformation requires a second input, the secret key. The secret key can be of any size depending on the cipher used while AES supports only three different key sizes of 128,192 and 256 bits.

The AES algorithm is a symmetric block cipher that operates onfixed block of data size 128 bits and key sizes is 128, 192 and 256 bits depending on 10, 12 and 14 rounds respectively. The AES encryption process operates on four different operations such as Substitution byte, Shift row, Mix-column and Addround key. The decryption process also has four operations are Inverse substitution byte, Inverse shift row, Inverse Mix-column and Inverse add round key. The 128bits plaintext contains 16 bytes i.e., (b0,b1,b2,….,b15).

A. AES ENCRYPTION

In this operation the plaintext is converted into the ciphertext format using the secret key.

a. Sub-bytes Transformation: Every byte in the state is replaced by another one using the Rijndael S-box given in Table 1.
b. Shift row: Every row in the 4x4 array is shifted a certain amount to the left.
c. Mix-column: A linear transformation on the columns of the state.
d. Add round key: Each byte of the state is XOR with a round key, which is a different key for each round and derived from the Rijndael key Schedule.

Table 1 Substitutional box table

		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	1	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	4	c7	23	c3	18	96	5	9a	7	12	80	e2	eb	27	b2	75
	4	9	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	0	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	2	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
X	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	6	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	bd	d8	4e	a9	6c	56	f4	ea	65	78	ae	8
	c	ba	78	25	2e	1c	a5	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	3	f6	0e	61	35	57	b9	85	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	9f	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

B. AES DECRYPTION

Decryption is the reverse operation of encryption operation i.e., the ciphertext is converted into the plaintext.

Table 2 Inverse Substitutional box table

		Y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
X	0	52	9	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	8	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	0	8c	bc	d3	0a	f7	e4	58	5	b8	b3	45	6
	7	do	2c	1e	8f	ca	3f	0f	2	c1	af	bd	3	1	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	1a
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	7	c7	31	b1	12	10	59	27	80	6c	5f
	d	60	51	7f	a9	19	b5	4a	od	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	4	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

a. Inverse Sub-Byte: Each byte in the state matrix is replaced with inverse S-box table given in Table 2.
b. Inverse Shift row: Every row in the 4x4 array is shifted a certain amount to right.
c. Inverse Mix-column: This is inverse operation of mix column operation. This operates on the state matrix column by column and each column is treated as a four-term polynomial.
*d. Inverse Add round key:*Inverse XOR operation is performed with each byte.

III. ENCRYPTION AND DECRYPTION ALGORITHM

The Cryptography algorithm performs two different operations, Encryption and Decryption. In Encryption, the plaintext is converted into ciphertext using a secret key and in decryption, the ciphertext is again converted into plaintext with the help of the same secret key. In AES encryption algorithm, the 128 bits size of a key consists of 10 rounds as given in Figure 1. The operations that are applied on the state during each round are Sub byte transformation, Shift Row operation, Mix column transformation and Add round key operation. The Mix-column operation is omitted in the final round.

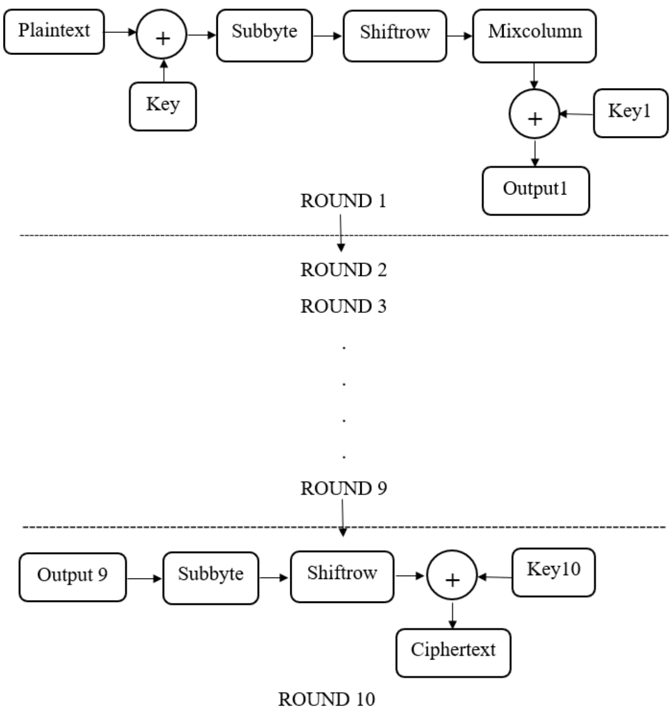


Figure 1: The 128 Bits Size of a Key Consists of 10 Rounds in AES Encryption Algorithm

AES Decryption algorithm is just the reverse operation of encryption algorithm. A key of 128 bits consists of 10 rounds as given in Figure 2.

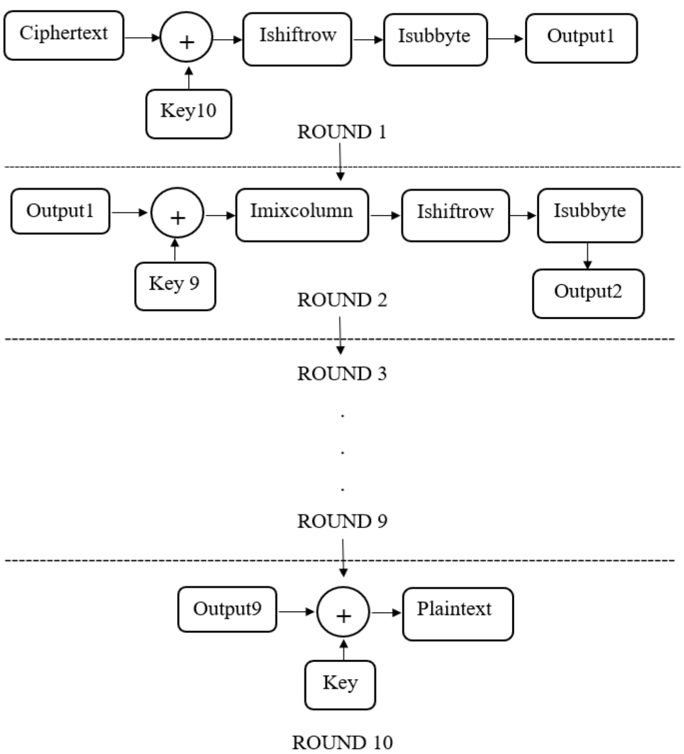


Figure 2: The 128 Bits Size of a Key Consists of 10 Rounds in AES Decryption Algorithm

The operations applied on the state during each round are inverse Sub-byte transformation, inverse shift row operation, inverse Mix-column transformation and inverse Add round key operation. In the first round Inverse Mix-column transformation operation is omitted.

IV. MODES OF OPERATION

The different modes of operation of block ciphers in AES are configuration methods that allowed to process with large data streams also without the risk of compromising the security provided. Here we provide some existing ways to blur the cipher text as a result the intruder can be avoided to break the cipher. Such modifications are known as Modes of block cipher operations.

1. ECB— (ELECTRONIC CODEBOOK) MODE

This is the simplest mode of encryption. Each plaintext block is encrypted separately. Similarly, each ciphertext block is decrypted separately. The ECB encryption and decryption given in Figure 3(a) and 3(b) respectively. In this mode, we can encrypt and decrypt by using many threads simultaneously. The only disadvantage is the created ciphertext is not blurred.

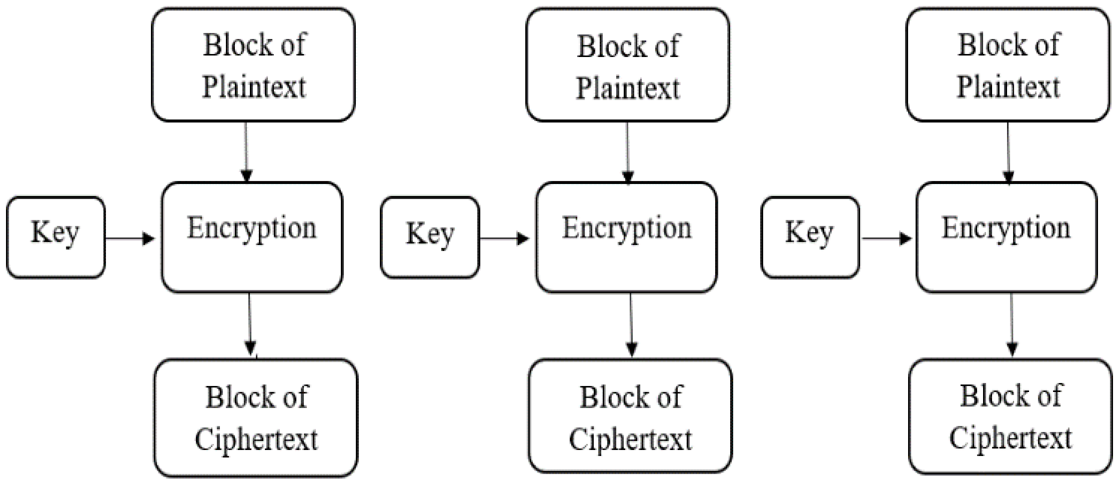


Figure 3(a). Electronic Codebook Encryption

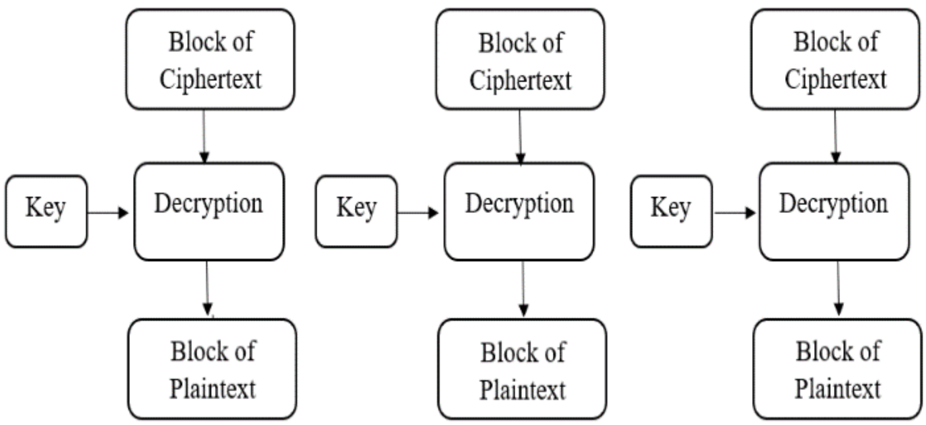


Figure 3(b). Electronic Codebook Decryption

2. PCBC— (PROPAGATING OR PLAINTEXT CIPHERBLOCK CHAINING) MODE

This mode adds XOR to the plaintext and then encrypts the data. The first plaintext block is XOR with Initialization Vector (IV). The IV has the same block size as plaintext. During decryption, the decrypted data is XOR with IV. The PCBC encryption and decryption are given in fig. 4(a) and 4(b) respectively. In this mode, both encryption and decryption can be performed using only one thread at a time. If any single ciphertext bit is damaged, the next plaintext and all subsequent blocks will be damaged and unable to decrypt correctly.

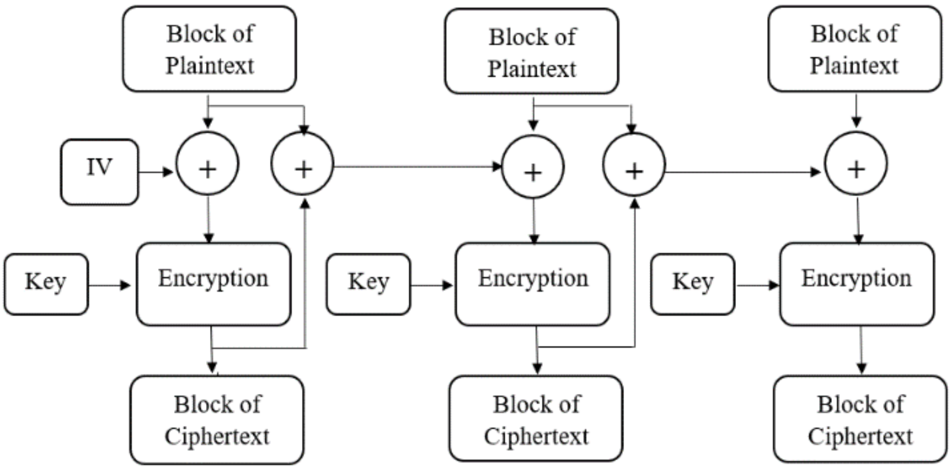


Figure 4(a). Propagating Or Plaintext Cipherblock Chaining Encryption

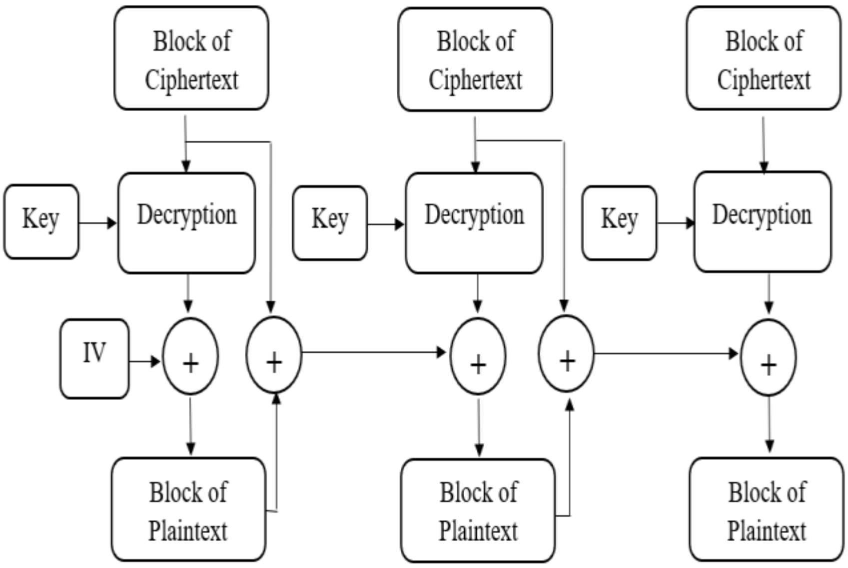


Figure 4(b). Propagating Or Plaintext Cipherblock Chaining Decryption

3. CFB— (CIPHER FEEDBACK) MODE

This is also similar as PCBC mode, except that one should encrypt cipher data from previous round, not the plaintext. The CFB encryption and decryption are given in Figure 5(a) and 5(b) respectively. Encryption in CFB mode can be performed only by using one thread and Decryption can be performed using many threads simultaneously.

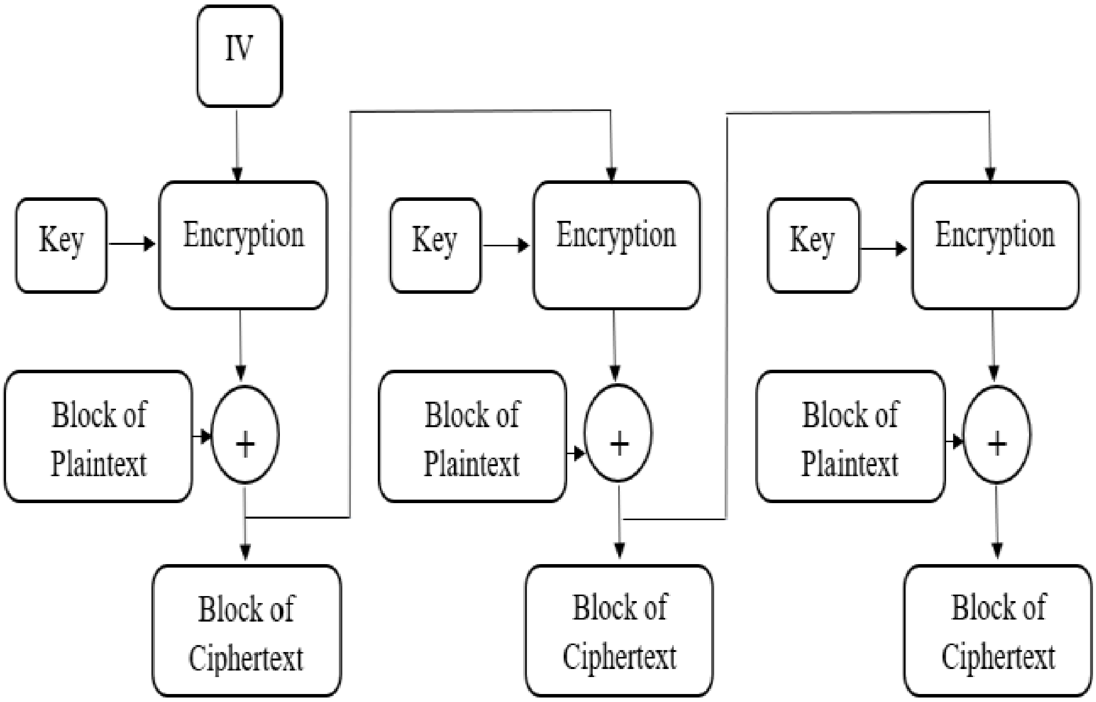


Figure 5(a). Cipher Feedback Encryption

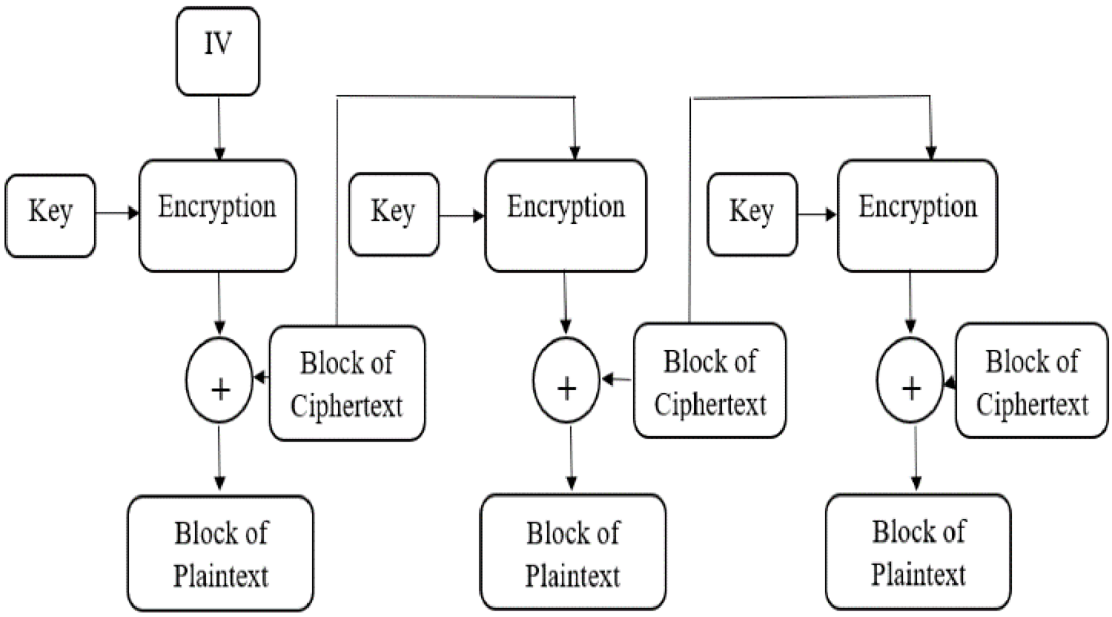


Figure 5(b). Cipher Feedback Decryption

4. OFB— (OUTPUT FEEDBACK) MODE

This creates keystream bits that are used for encrypting subsequent data blocks. In this regard, the way of working of cipher becomes similar the way of working of typical stream cipher. The OFB encryption and decryption are given in Figure 6(a) and 6(b) respectively. In OFB mode we can perform both encryption and decryption using only one thread at a time.

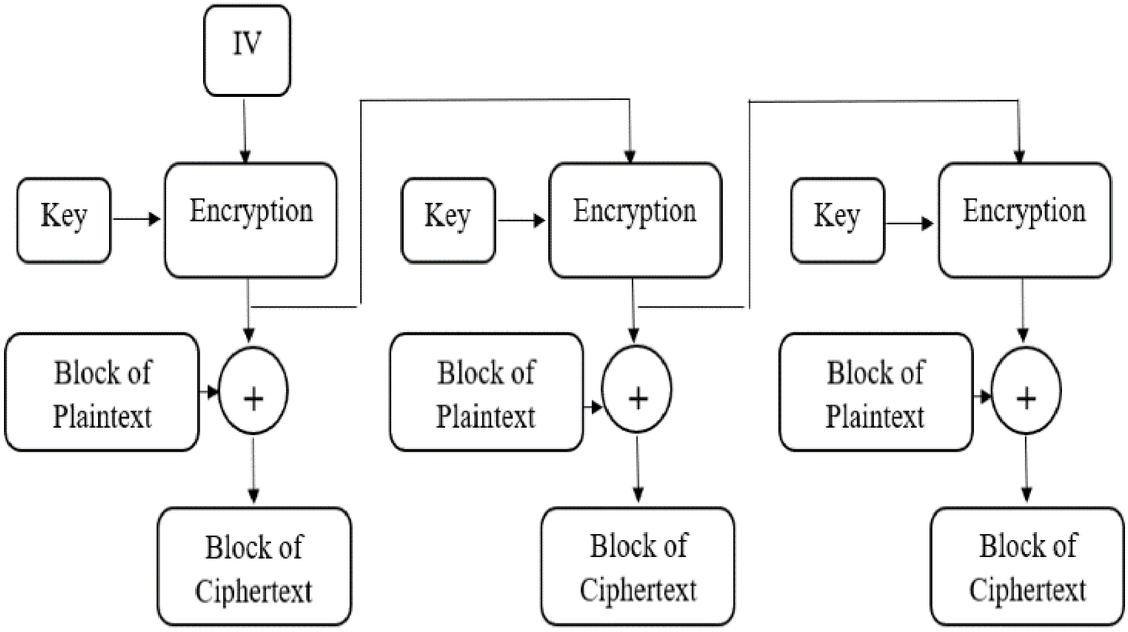


Figure 6(a) Output Feedback Encryption

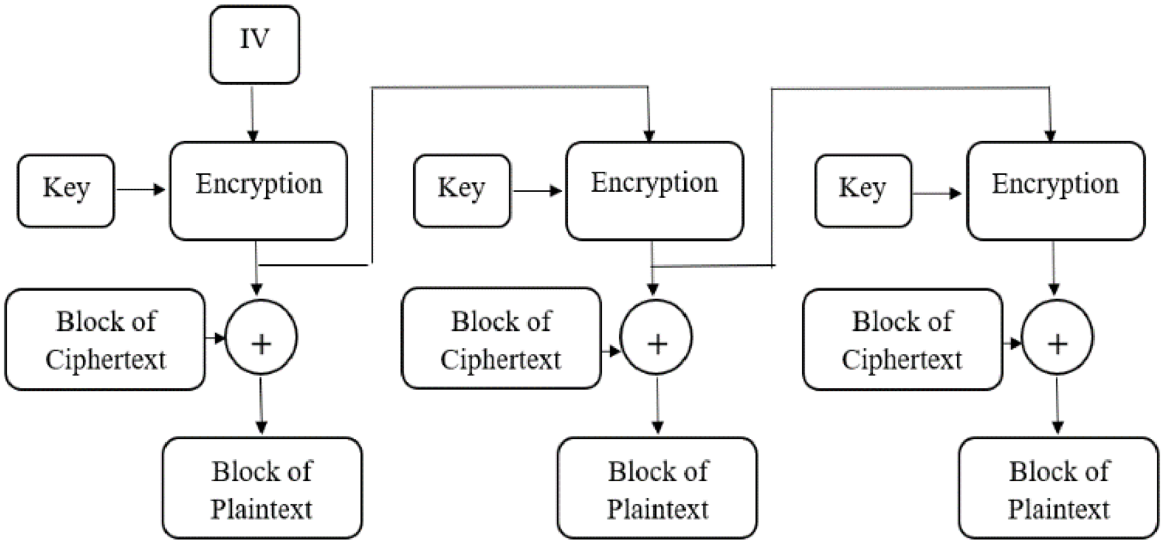


Fig. 6(b) Output Feedback Decryption

5. CTR— (COUNTER) MODE

This is the most popular block cipher modes of operation. The CTR encryption and decryption are given in Figure 7(a) and 7(b) respectively. In this mode, Both the encryption and decryption can be performed using many threads at a time. The nonce is a unique number used once. It plays the same role as IV. The subsequent values of an increasing counter are added to nonce. CTR mode is also known as SIC (Segment Integer Counter) mode. If one plaintext bit iscorrupted, then only one corresponding output bit is damaged.

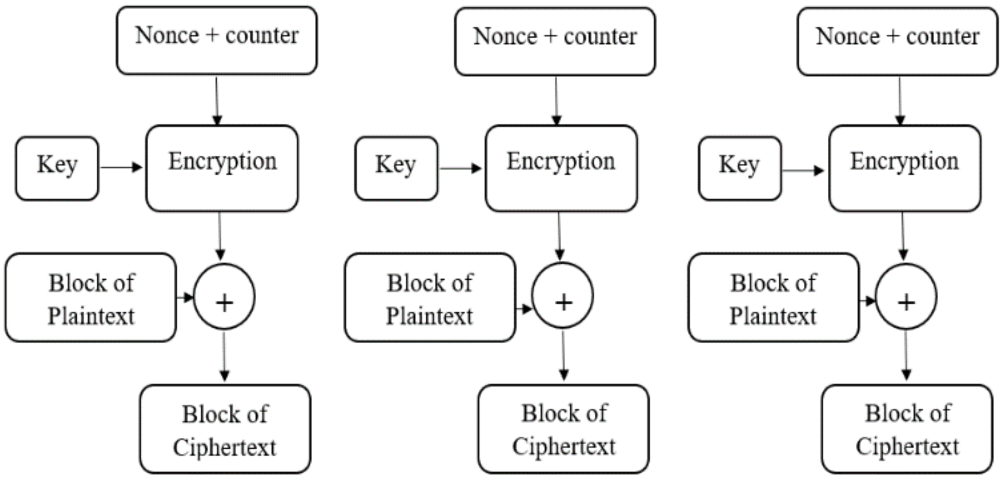


Figure 7(a) Counter Encryption

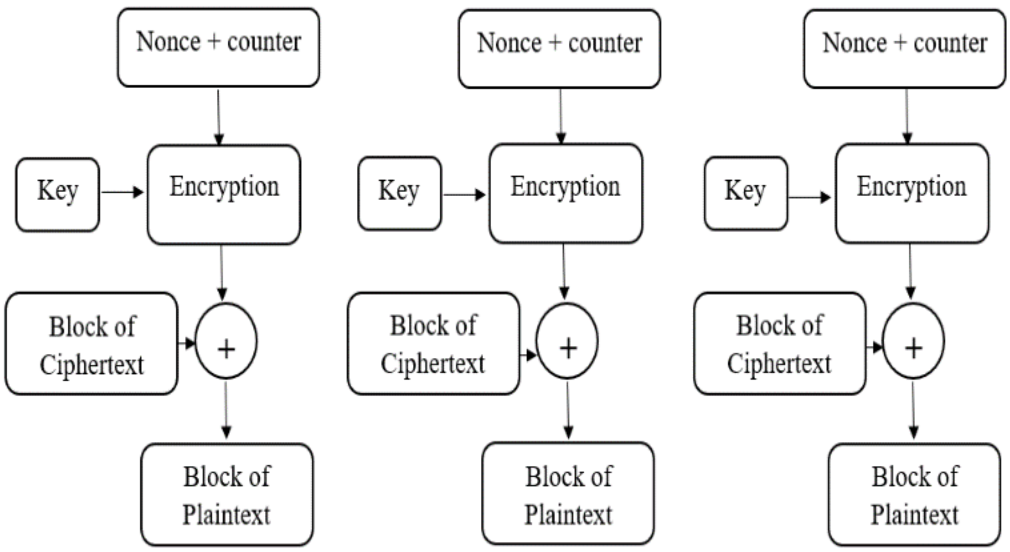


Figure 7(b) Counter Decryption

First Half of the Project: Encryption and Decryption of Files of any Format

Libraries Used:

1. Tqdmm
2. Termcolor

Input Data:

File type of any format that the user inputs into the system.

Output:

Encrypted Files of various format which has been done under the process of 256 bit AES algorithm in OFB Mode.

The project source code can be viewed on GitHub at:

https://github.com/Abhayindia/Message-Encription

SOURCE CODE:

```
# CRYPTOGRAPHIC TOOL BASED ON AES-
256 (OFB MODE) - A Symmetric Cryptographic Encryption and Decryption in Python
# Submitted by - Abhay Chaudhary
# Submitted to - Dr. Garima Singh
# CSE1007 Introduction to Cryptography (SLOT-A+TA)
# Python v3.9.0
# imports

import os
import sys
from tqdm import tqdm
from termcolor import colored,cprint

class Encryption:

    def __init__(self,filename):    # Constructor
        self.filename = filename

    def encryption(self): # Allows us to perfrom file operation

        try:
            original_information = open(self.filename,'rb')

        except (IOError, FileNotFoundError):
```

```
            cprint('File with name {} is not found.'.format(self.filename), co
lor='red',attrs=['bold','blink'])
            sys.exit(0)

        try:

            encrypted_file_name = 'cipher_' + self.filename
            encrypted_file_object = open(encrypted_file_name,'wb')

            content = original_information.read()
            content = bytearray(content)


            key = 192
            cprint('Encryption Process is in progress...!',color='green',attrs
=['bold'])
            for i,val in tqdm(enumerate(content)):
                content[i] = val ^ key

            encrypted_file_object.write(content)


        except Exception:
            cprint('Something went wrong with {}'.format(self.filename),color=
'red',attrs=['bold','blink'])
        finally:
            encrypted_file_object.close()
            original_information.close()

class Decryption:

    def __init__(self,filename):
        self.filename = filename

    def decryption(self):   # produces the original result

        try:
            encrypted_file_object = open(self.filename,'rb')

        except (FileNotFoundError,IOError):
            cprint('File with name {} is not found'.format(self.filename),colo
r='red',attrs=['bold','blink'])
            sys.exit(0)

        try:

            decrypted_file = input('Enter the filename for the Decryption file
 with extension:') # Decrypted file as output
```

```
            decrypted_file_object = open(decrypted_file,'wb')

            cipher_text = encrypted_file_object.read()

            key = 192

            cipher_text = bytearray(cipher_text)

            cprint('Decryption Process is in progress...!',color='green',attrs
=['bold'])

            for i,val in tqdm(enumerate(cipher_text)):
                cipher_text[i] = val^key

            decrypted_file_object.write(cipher_text)


        except Exception:
                cprint('Some problem with Ciphertext unable to handle.',color=
'red',attrs=['bold','blink'])

        finally:
            encrypted_file_object.close()
            decrypted_file_object.close()

space_count = 30 * ' '
cprint('{} Encription and Decription of Files in AES-
256 (OFB MODE). {}'.format(space_count, space_count), 'red')
cprint('{} {}'.format(space_count + 3 * ' ','Programmed by Abhay Chaudhary 19B
CE7290.'),'green')
while True:
        cprint('1. Encryption',color='magenta')
        cprint('2. Decryption',color='magenta')
        cprint('3. Exit', color='red')
        # cprint('Enter your choice:',color='cyan',attrs=["bold"])
        cprint('~Python3:',end=' ', color='green')
        choice = int(input())

        if choice == 1:
            logo = '''  ___                        _   _
 | __|_ _  __ _ _ _  _ _ __| |_(_)___ _ _
 | _|| ' \/ _| '_| || | '_ \  _| / _ \ ' \
 |___|_||_\__|_|  \_, | .__/\__|_\___/_||_|
                  |__/|_|                   '''
            cprint(logo,color='red',attrs=['bold'])
            cprint('Enter the filename for Encryption with proper extension:',
end=' ',color='yellow',attrs=['bold'])
```

```
            file = input()
            E1 = Encryption(file)
            E1.encryption()
            cprint('{} Encryption is done Sucessfully...!'.format(file), color
='green',attrs=['bold'])
            cprint('Do you want to do it again (y/n):',end = ' ', color='red',
attrs=['bold','blink'])
            again_choice  = input()
            if (again_choice.lower() == 'y'):
                continue
            else:
                break

        elif choice == 2:
            logo = '''  ___                         _   _
 |   \ ___ __ _ _ _  _ _ __| |_(_)___ _ _
 | |) / -_) _| '_| || | '_ \  _| / _ \ ' \
 |___/\___\__|_|  \_, | .__/\__|_\___/_||_|
                  |__/|_|                  '''
            cprint(logo,color='red',attrs=['bold'])
            cprint('Enter the Encrypted filename with proper extension:',end='
 ',color='yellow',attrs=['bold'])
            file = input()
            D1 = Decryption(file)
            D1.decryption()
            cprint('{} Decryption is done Sucessfully...!'.format(file),color=
'green',attrs=['bold'])
            cprint('Do you want to do it again (y/n):',end = ' ', color='red',
attrs=['bold','blink'])
            again_choice  = input()
            if (again_choice.lower() == 'y'):
                continue
            else:
                break
        elif choice == 3:
            sys.exit(0)
        else:
            print('Your choice of selection is not available. Sorry to see you
 again.')
```

OUTPUT:

```
Microsoft Windows [Version 10.0.18363.1198]
(c) 2019 Microsoft Corporation. All rights reserved.

D:\Vellore Institue of Technology\Semester wise Material\Semester 3(Fall Semester)(2020-2021)\CSE-1007 Introduction to Cryptography Slot A\Project\AES-Cryptographic-Tool-ma
ster>python Script.py
                              Encription and Decription of Files in AES-256 (OFB MODE).
                                 Programmed by Abhay Chaudhary 19BCE7290.
1. Encryption
2. Decryption
3. Exit
~Python3: 1
Enter the filename for Encryption with proper extension: Test Image.jpg
Encryption Process is in progress...!
902216it [00:00, 2594284.18it/s]
Test Image.jpg Encryption is done Sucessfully...!
Do you want to do it again (y/n): y
1. Encryption
2. Decryption
3. Exit
~Python3: 2
Enter the Encrypted filename with proper extension: cipher_Test Image.jpg
Enter the filename for the Decryption file with extension:decrypt_Image.jpg
Decryption Process is in progress...!
902216it [00:00, 2634348.72it/s]
cipher_Test Image.jpg Decryption is done Sucessfully...!
Do you want to do it again (y/n): n

D:\Vellore Institue of Technology\Semester wise Material\Semester 3(Fall Semester)(2020-2021)\CSE-1007 Introduction to Cryptography Slot A\Project\AES-Cryptographic-Tool-ma
ster>
```

Figure 8. Test Image.jpg

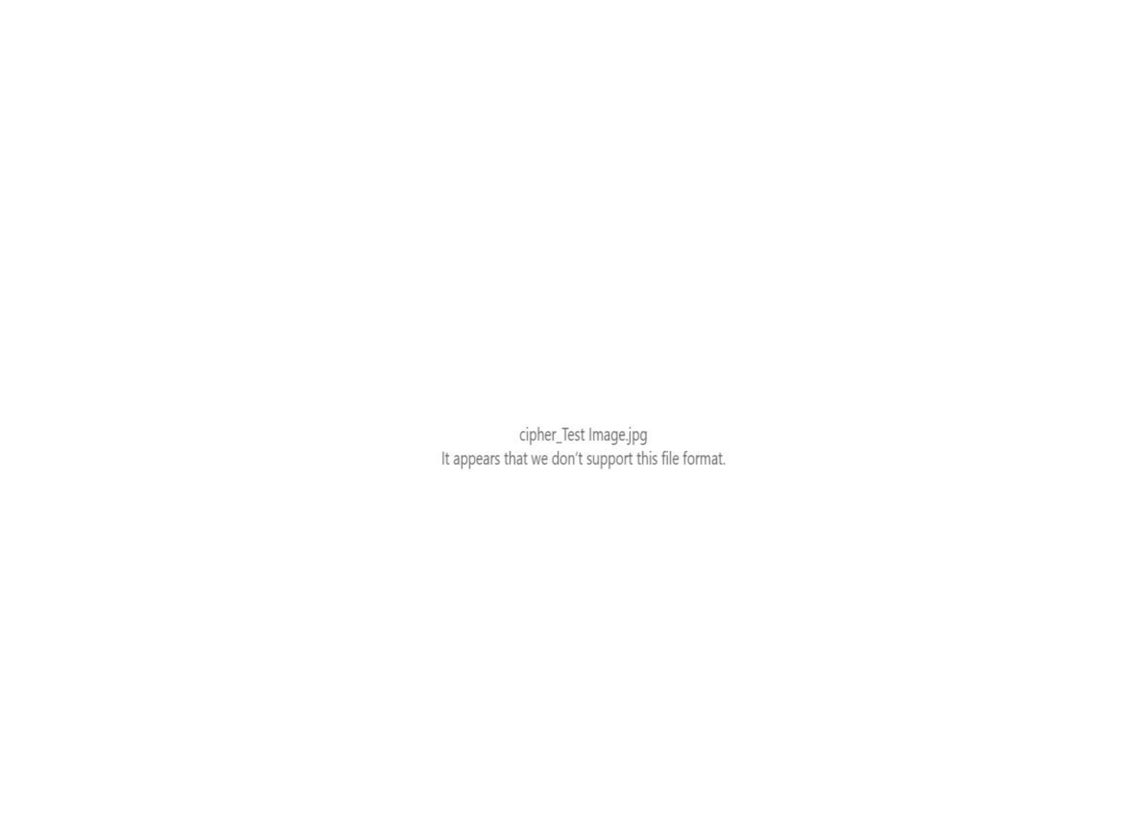


Figure 9. cipher_Test Image.jpg

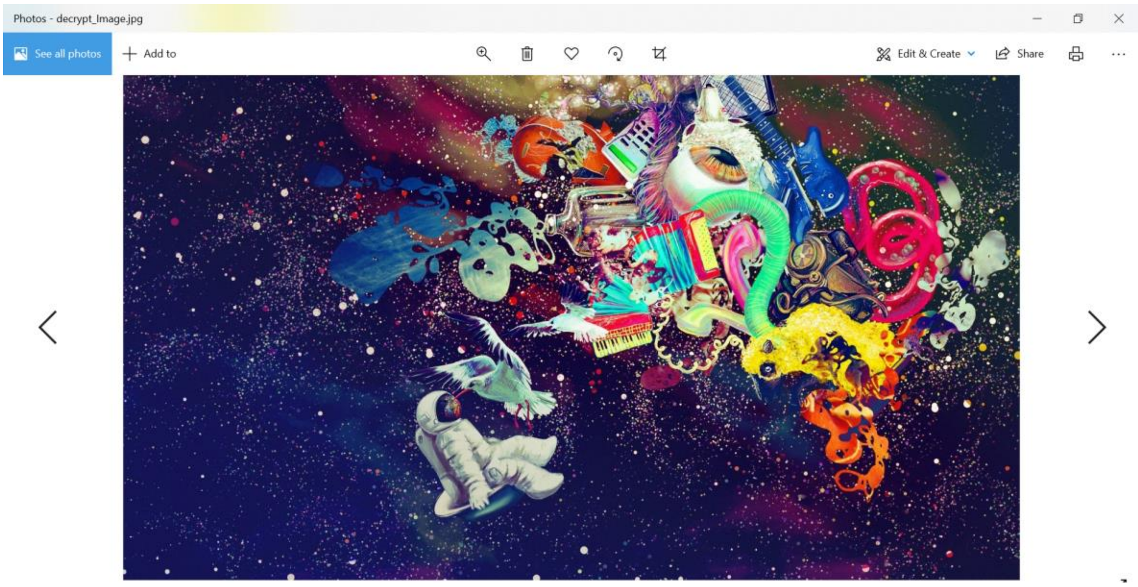


Figure 10. decrypt_Image.jpg

The whole process of Encryption and Decryption of files has been successfully performed using AES 256 bit in OFB Mode for several file formats.

1.2. *Message Encryption to WhatsApp Contacts*

ROT 13

ROT13 ("rotate by 13 places", sometimes hyphenated ROT-13) is a simple letter substitution cipher that replaces a letter with the 13th letter after it in the alphabet. ROT13 is a special case of the Caesar cipher which was developed in ancient Rome.

Because there are 26 letters (2×13) in the basic Latin alphabet, ROT13 is its own inverse; that is, to undo ROT13, the same algorithm is applied, so the same action can be used for encoding and decoding. The algorithm provides virtually no cryptographic security, and is often cited as a canonical example of weak encryption. ROT13 is used in online forums as a means of hiding spoilers, punchlines, puzzle solutions, and offensive materials from the casual glance. ROT13 has inspired a variety of letter and word games online, and is frequently mentioned in newsgroup conversations.

Applying ROT13 to a piece of text merely requires examining its alphabetic characters and replacing each one by the letter 13 places further along in the alphabet, wrapping back to the beginning if necessary. A becomes N, B becomes O, and so on up to M, which becomes Z, then the sequence continues at the beginning of the alphabet: N becomes A, O becomes B, and so on to Z, which becomes M. Only those letters which occur in the English alphabet are affected; numbers, symbols, whitespace, and all other characters are left unchanged. Because there are 26 letters in the English alphabet and 26 = 2 × 13, the ROT13 function is its own inverse:

In other words, two successive applications of ROT13 restore the original text (in mathematics, this is sometimes called an *involution*; in cryptography, a *reciprocal cipher*).

The transformation can be done using a lookup table, such as the following:

Input	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
Output	NOPQRSTUVWXYZABCDEFGHIJKLMnopqrstuvwxyzabcdefghijklm

For example, in the following joke, the punchline has been obscured by ROT13:

Why did the chicken cross the road?

Gb trg gb gur bgure fvqr!

Transforming the entire text via ROT13 form, the answer to the joke is revealed:

Jul qvq gur puvpxra pebff gur ebnq?
To get to the other side!

A second application of ROT13 would restore the original.

ASCII

ASCII was developed from telegraph code. Its first commercial use was as a seven-bit teleprinter code promoted by Bell data services. Work on the ASCII standard began on October 6, 1960, with the first meeting of the American Standards Association's (ASA) (now the American National Standards Institute or ANSI) X3.2 subcommittee. The first edition of the standard was published in 1963, underwent a major revision during 1967, and experienced its most recent update during 1986. Compared to earlier telegraph codes, the proposed Bell code and ASCII were both ordered for more convenient sorting (i.e., alphabetization) of lists, and added features for devices other than teleprinters.

The use of ASCII format for Network Interchange was described in 1969. That document was formally elevated to an Internet Standard in 2015.

Originally based on the English alphabet, ASCII encodes 128 specified characters into seven-bit integers as shown by the ASCII chart above. Ninety-five of the encoded characters are printable: these include the digits 0 to 9, lowercase letters a to z, uppercase letters A to Z, and punctuation symbols. In addition, the original ASCII specification included 33 non-printing control codes which originated with Teletype machines; most of these are now obsolete, although a few are still commonly used, such as the carriage return, line feed and tab codes.

For example, lowercase i would be represented in the ASCII encoding by binary 1101001 = hexadecimal 69 (i is the ninth letter) = decimal 105.

Second Half of the Project: Encryption and Decryption of Messages

Language Used:
Shell
Input Data:
Text which shall be encrypted or Decrypted.
Output:
Encrypted Message will be sent to the WhatsApp contact which is given.
WhatsApp API used:
Rapiwha (https://panel.rapiwha.com/landing/)

SOURCE CODE:

The project source code can be viewed on GitHub at:

https://github.com/Abhayindia/Message-Encription

WhatsApp API Configuration:

```
# Getting API :
# Register in here https://panel.rapiwha.com/landing/?utm_source=www.apiwha.co
m
# Use your mail or temp mail :P

# Setup API KEY
# Example :
#      - api="----------------------"
api=" "
```

Encryption :

```
elif [[ $case1 == "set" ]];then
                if [[ $case2 == "file" ]] && [[ ! $case3 == "" ]]
                    then
                        file=$case3
                elif [[ $case2 == "style" ]] && [[ ! $case3 == "" ]]
                then
                    if [[ $case3 == "1" ]];then
                      type1=template/0x1.key
                      type=$type1
                    elif [[ $case3 = "2" ]]; then
                      type1=template/0x2.key
                      type=$type1
                    elif [[ $case3 = "3" ]]; then
                      type1=template/0x3.key
                      type=$type1
```

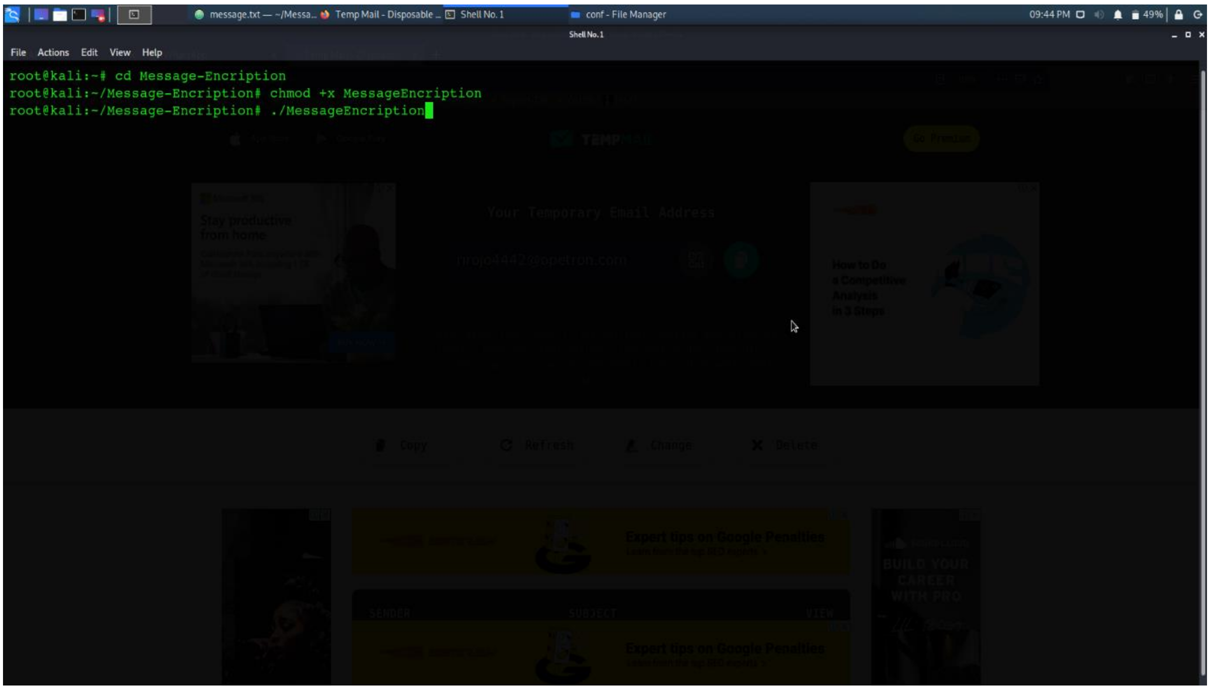
```
elif [[ $case3 = "4" ]]; then
  type1=template/0x4.key
  type=$type1
elif [[ $case3 = "5" ]]; then
  type1=template/0x5.key
  type=$type1
elif [[ $case3 = "6" ]]; then
  type1=template/0x6.key
  type=$type1
elif [[ $case3 = "7" ]]; then
  type1=template/0x7.key
  type=$type1
elif [[ $case3 = "8" ]]; then
  type1=template/0x8.key
  type=$type1
elif [[ $case3 = "9" ]]; then
  type1=template/0x9.key
  type=$type1
elif [[ $case3 = "10" ]]; then
  type1=template/0x10.key
  type=$type1
elif [[ $case3 = "11" ]]; then
  type1=template/0x11.key
  type=$type1
elif [[ $case3 = "12" ]]; then
  type1=template/0x12.key
  type=$type1
elif [[ $case3 = "13" ]]; then
  type1=template/0x13.key
  type=$type1
elif [[ $case3 = "14" ]]; then
  type1=template/0x14.key
  type=$type1
elif [[ $case3 = "15" ]]; then
  type1=template/0x15.key
  type=$type1
elif [[ $case3 = "16" ]]; then
  type1=template/0x16.key
  type=$type1
elif [[ $case3 = "17" ]]; then
  type1=template/0x17.key
  type=$type1
elif [[ $case3 = "18" ]]; then
  type1=template/0x18.key
  type=$type1
elif [[ $case3 = "19" ]]; then
  type1=template/0x19.key
  type=$type1
```

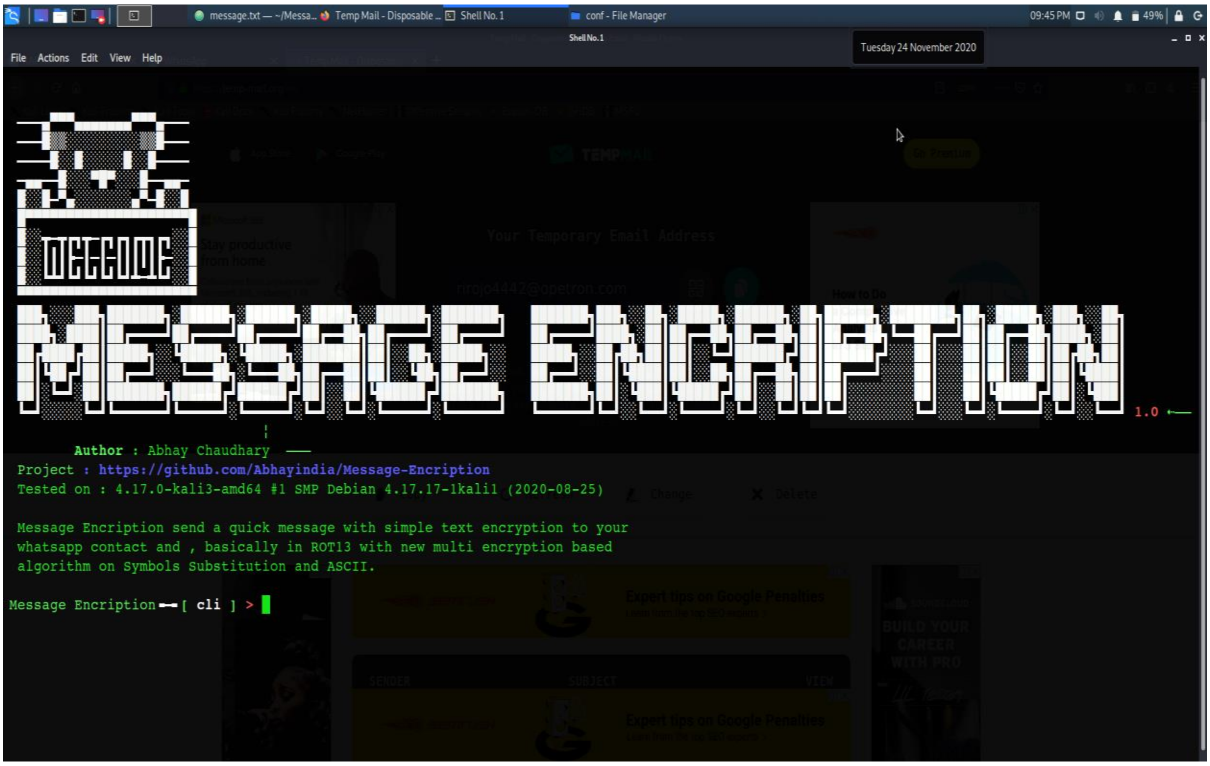
```
                    elif [[ $case3 = "20" ]]; then
                      type1=template/0x20.key
                      type=$type1
                    elif [[ $case3 = "21" ]]; then
                      type1=template/0x21.key
                      type=$type1
                    elif [[ $case3 = "22" ]]; then
                      type1=template/0x22.key
                      type=$type1
                    elif [[ $case3 = "23" ]]; then
                      type1=template/0x23.key
                      type=$type1
```

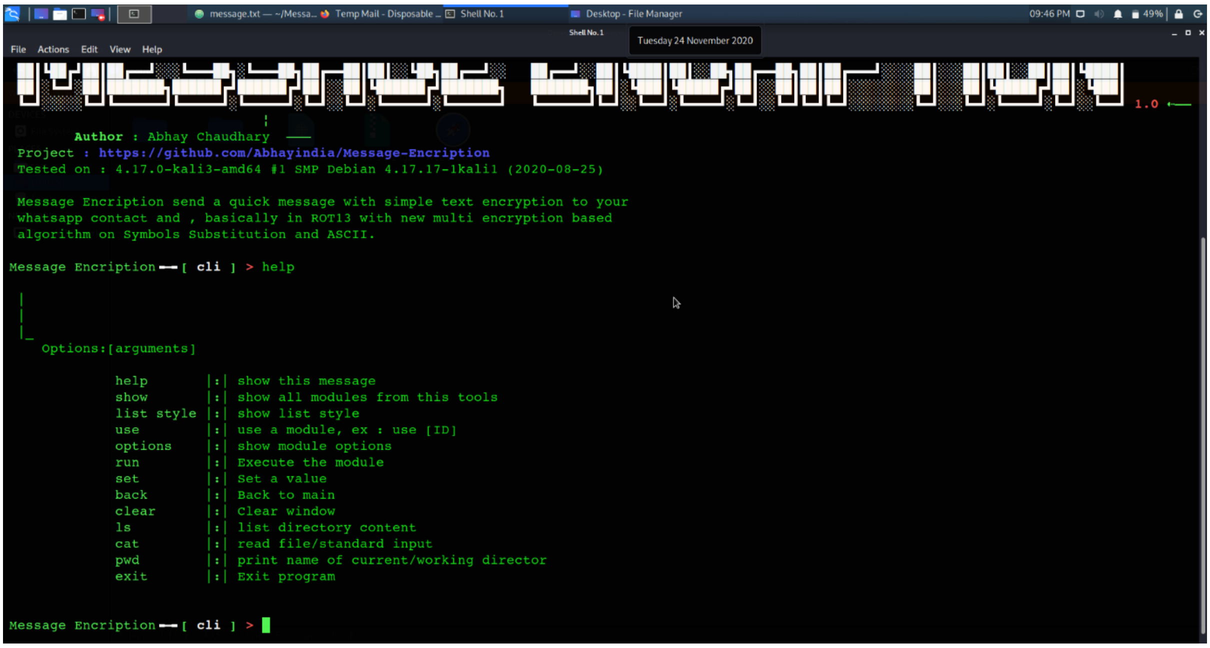
Decryption:

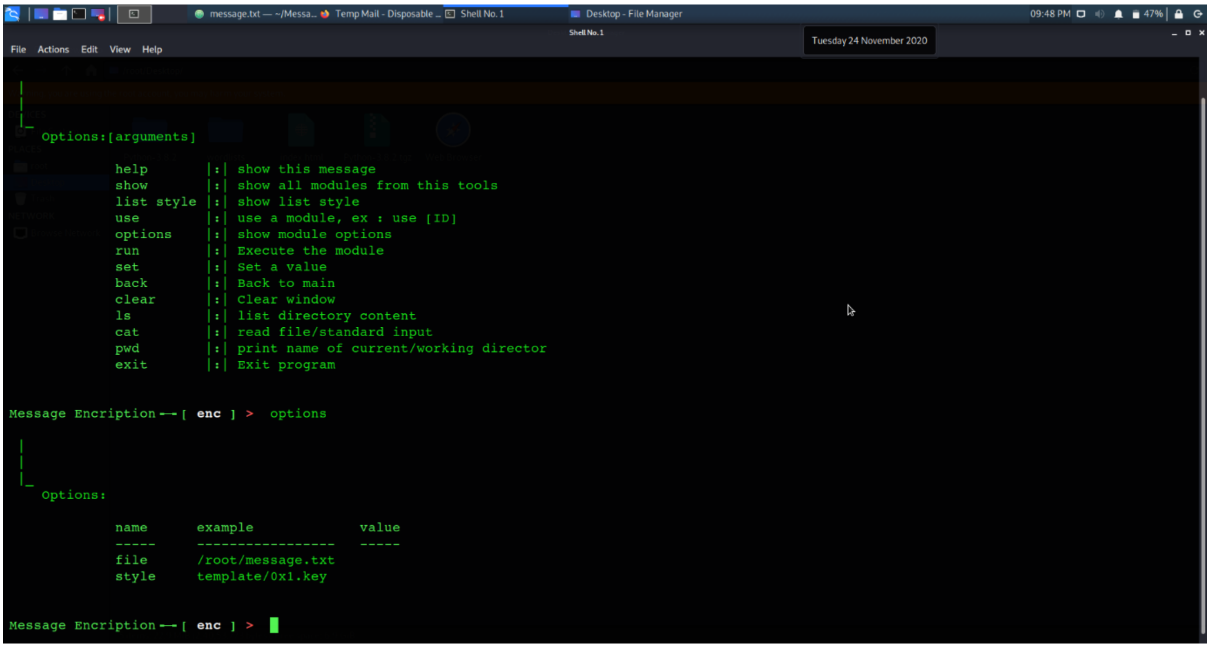
```
elif [[ $case1 == "run" ]];then #not spesific
                    if [ -f "$file" ]
                        then
                        #clear
                        echo -e "\n | "
                        echo " |_ "
                        echo -
e $okegreen"    [+]"$RESET" Process Decryption ROT13 \n "
                        echo -e $yellow"         [-] "$RESET"Process 0x1";
                        while IFS=" - " read x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x1
1 ; do
                        sed -e 's/'$x1'/N/g' -e 's/'$x2'/O/g' -
e 's/'$x3'/P/g' -e 's/'$x4'/Q/g' -e 's/'$x5'/R/g' -e 's/'$x6'/S/g' -
e 's/'$x7'/T/g' -e 's/'$x8'/U/g' -e 's/'$x9'/V/g' -e 's/'$x10'/W/g' -
e 's/'$x11'/a/g'  < $file  > $msg_dec1
                        done < $type1
                        echo -e $yellow"         [-] "$RESET"Process 0x1";
                        tr '[A-Z]' '[N-ZA-M]' < $msg_dec1 > $msg_dec2
                        value0xD2=$(<$msg_dec2);echo $value0xD2 | tr ABCDEFGHI
J 0123456789 | xxd -r -p > $msg_dec3
                        echo -e $yellow"         [-] "$RESET"Process 0x3"
                        value0xD3=$(<$msg_dec3);echo $value0xD3 | tr ABCDEFGHI
J 0123456789 | awk -
F '' '{ for(i=1; i<=NF; i+=2) {printf "%s%s ", $i,$(i+1);}}' > $msg_dec3
                        awk '{ for(i=1;i<=NF;i++) printf("%c",$i); print "";
}' < $msg_dec3 > $msg_dec4
                        echo -e $yellow"         [-
] "$RESET"Process 0x4"
                        awk '{print tolower($0)}' < $msg_dec4 > $msg_fin
                        tr '[a-z]' '[n-za-
m]' < $msg_fin > output/messageencription.decrypt
```

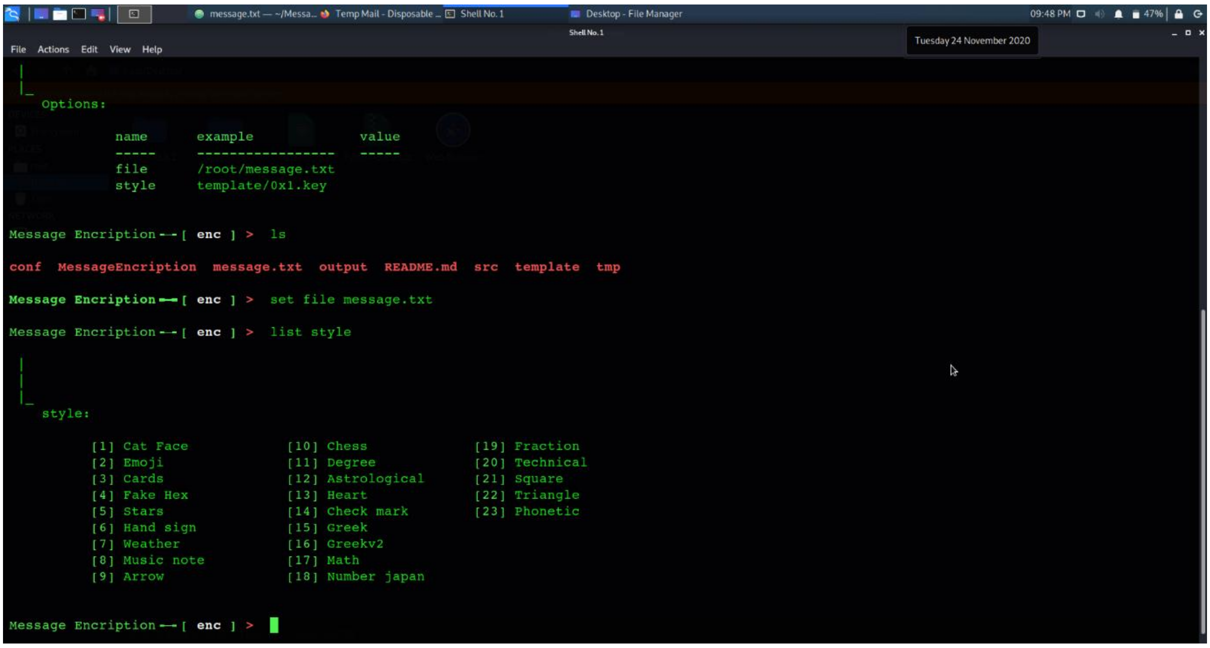
Output:

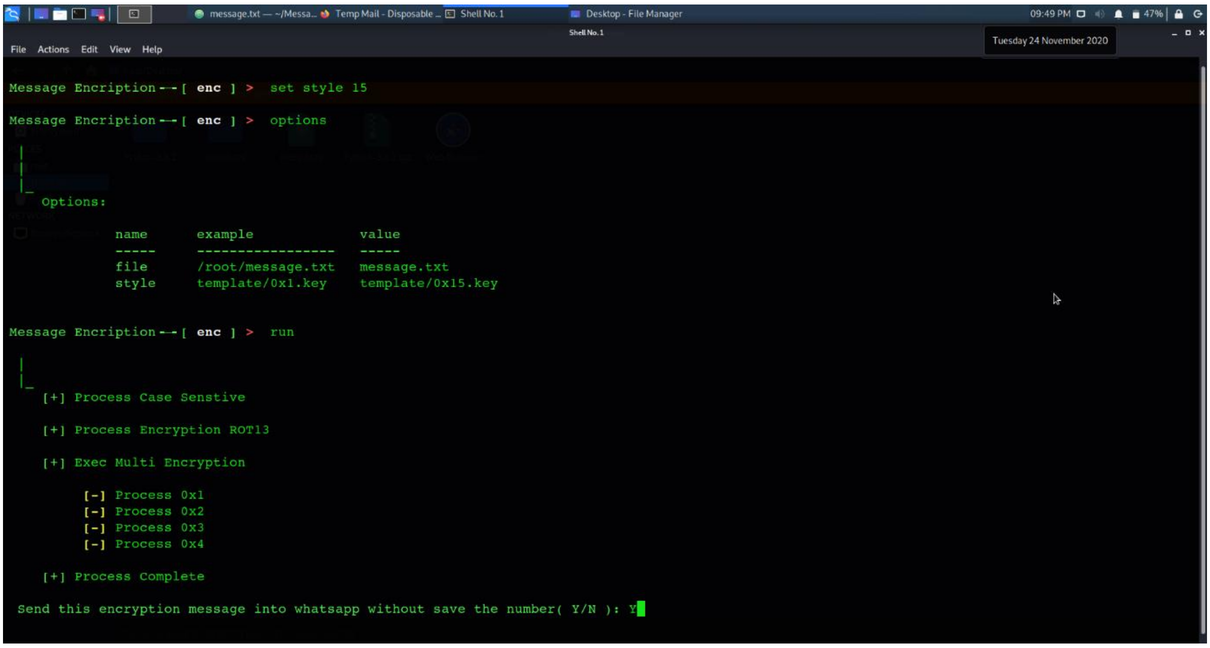

root@kali:~# cd Message-Encription
root@kali:~/Message-Encription# chmod +x MessageEncription
root@kali:~/Message-Encription# ./MessageEncription

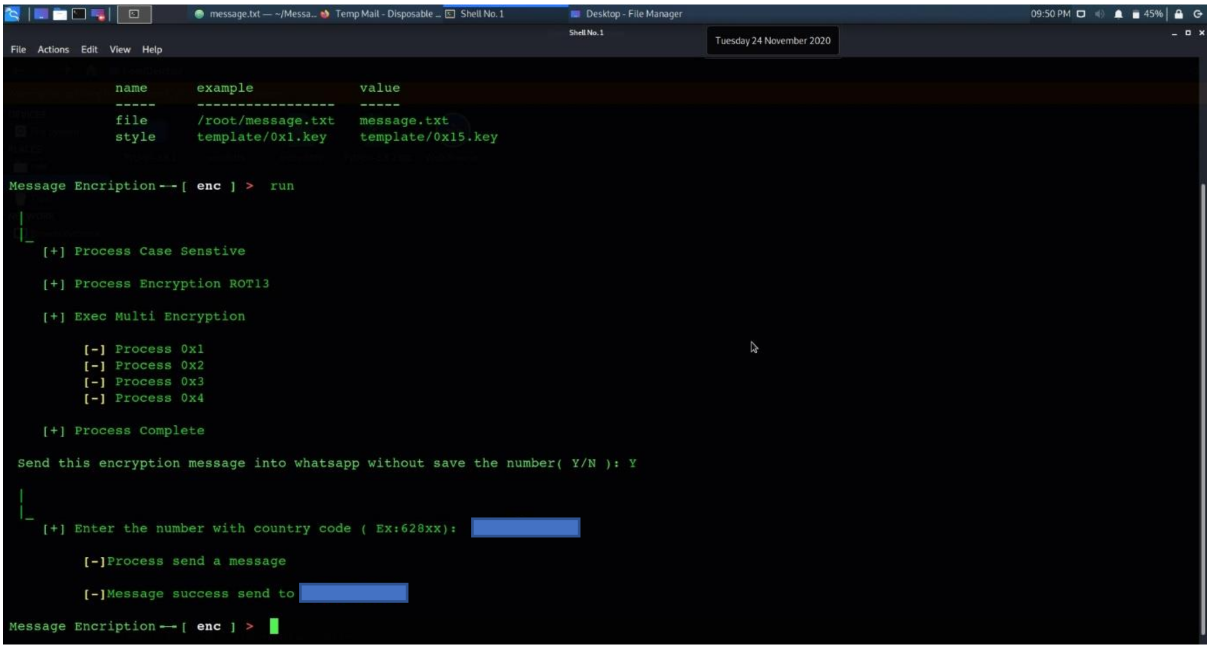

MESSAGE ENCRIPTION
1.0
Author : Abhay Chaudhary
Project : https://github.com/Abhayindia/Message-Encription
Tested on : 4.17.0-kali3-amd64 #1 SMP Debian 4.17.17-1kali1 (2020-08-25)
Message Encription send a quick message with simple text encryption to your whatsapp contact and , basically in ROT13 with new multi encryption based algorithm on Symbols Substitution and ASCII.
Message Encription [cli] >

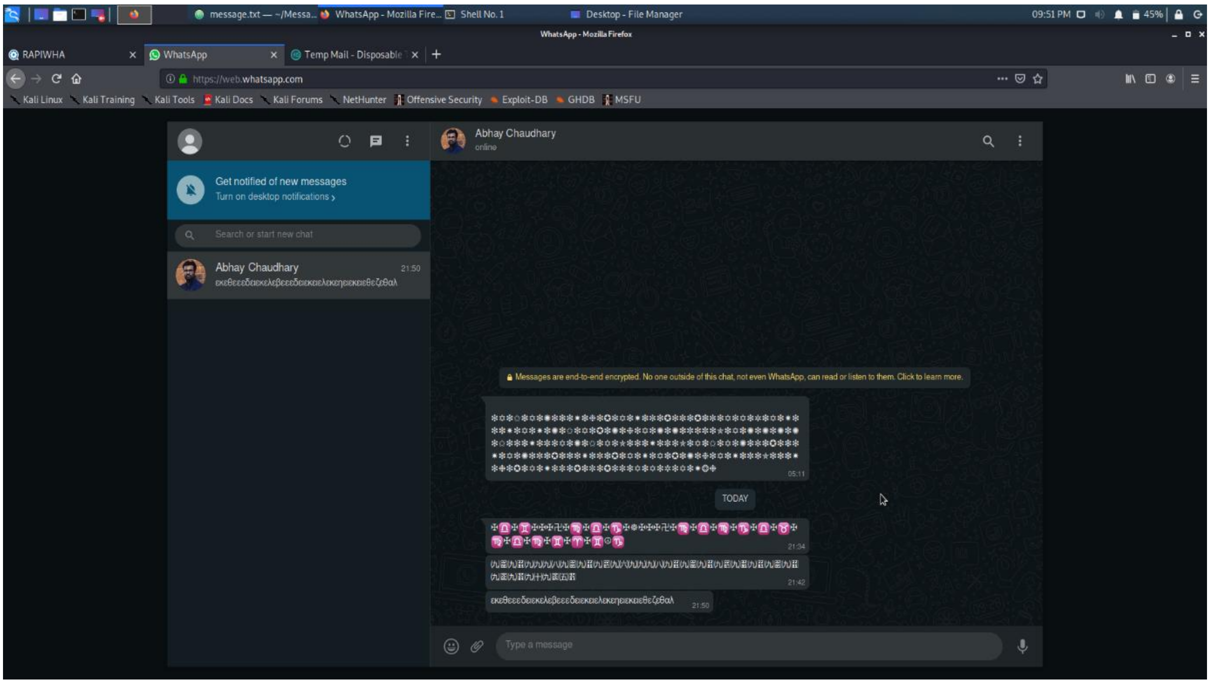

message.txt — ~/Messa...
Temp Mail - Disposable ...
Shell No. 1
Desktop - File Manager
09:46 PM
49%
Shell No. 1
Tuesday 24 November 2020
File Actions Edit View Help
1.0
Author : Abhay Chaudhary
Project : https://github.com/Abhayindia/Message-Encription
Tested on : 4.17.0-kali3-amd64 #1 SMP Debian 4.17.17-1kali1 (2020-08-25)
Message Encription send a quick message with simple text encryption to your
whatsapp contact and , basically in ROT13 with new multi encryption based
algorithm on Symbols Substitution and ASCII.
Message Encription [cli] > help
Options:[arguments]
help |:| show this message
show |:| show all modules from this tools
list style |:| show list style
use |:| use a module, ex : use [ID]
options |:| show module options
run |:| Execute the module
set |:| Set a value
back |:| Back to main
clear |:| Clear window
ls |:| list directory content
cat |:| read file/standard input
pwd |:| print name of current/working director
exit |:| Exit program
Message Encription [cli] >


message.txt — ~/Messa...
Temp Mail - Disposable ...
Shell No. 1
Desktop - File Manager
09:48 PM
47%
Shell No. 1
Tuesday 24 November 2020
File Actions Edit View Help
Options:[arguments]
help |:| show this message
show |:| show all modules from this tools
list style |:| show list style
use |:| use a module, ex : use [ID]
options |:| show module options
run |:| Execute the module
set |:| Set a value
back |:| Back to main
clear |:| Clear window
ls |:| list directory content
cat |:| read file/standard input
pwd |:| print name of current/working director
exit |:| Exit program
Message Encription [enc] > options
Options:
name example value
file /root/message.txt
style template/0x1.key
Message Encription [enc] >


File Actions Edit View Help
Tuesday 24 November 2020
Options:
name example value
file /root/message.txt
style template/0x1.key
Message Encription ─[enc] > ls
conf MessageEncription message.txt output README.md src template tmp
Message Encription ─[enc] > set file message.txt
Message Encription ─[enc] > list style
style:
[1] Cat Face [10] Chess [19] Fraction
[2] Emoji [11] Degree [20] Technical
[3] Cards [12] Astrological [21] Square
[4] Fake Hex [13] Heart [22] Triangle
[5] Stars [14] Check mark [23] Phonetic
[6] Hand sign [15] Greek
[7] Weather [16] Greekv2
[8] Music note [17] Math
[9] Arrow [18] Number japan
Message Encription ─[enc] >


File Actions Edit View Help
Tuesday 24 November 2020
Message Encription ─[enc] > set style 15
Message Encription ─[enc] > options
Options:
name example value
file /root/message.txt message.txt
style template/0x1.key template/0x15.key
Message Encription ─[enc] > run
[+] Process Case Senstive
[+] Process Encryption ROT13
[+] Exec Multi Encryption
[-] Process 0x1
[-] Process 0x2
[-] Process 0x3
[-] Process 0x4
[+] Process Complete
Send this encryption message into whatsapp without save the number(Y/N): Y


name example value
file /root/message.txt message.txt
style template/0x1.key template/0x15.key
Message Encription ─[enc] > run
[+] Process Case Senstive
[+] Process Encryption ROT13
[+] Exec Multi Encryption
[-] Process 0x1
[-] Process 0x2
[-] Process 0x3
[-] Process 0x4
[+] Process Complete
Send this encryption message into whatsapp without save the number(Y/N): Y
[+] Enter the number with country code (Ex:628xx):
[-]Process send a message
[-]Message success send to
Message Encription ─[enc] >


WhatsApp
https://web.whatsapp.com
Abhay Chaudhary
Get notified of new messages
Turn on desktop notifications
Search or start new chat
Messages are end-to-end encrypted. No one outside of this chat, not even WhatsApp, can read or listen to them. Click to learn more.
TODAY
Type a message

References:

[1] Dilna.v, C. Babu "Area Optimized and high throughput AES algorithm based on permutation data scramble approach", in International Conference on Electrical, Electronics and Optimization techniques (ICEEOT)- 2016

[2] www.crypto-it.net/eng/theory/modes-of-block-ciphers.html

[3] https://en.wikipedia.org/wiki/Advanced_Encryption_Standard
https://www.tutorialspoint.com/cryptography/advanced_encryption_standard.html

[4] Bogdanov, A., Lauridsen, M., Tischhauser, E. (2014) AES-Based Authenticated Encryption Modes in Parallel High-Performance Software, International Association for Cryptologic Research

[5] Bollapragada, V., Khalid, M., Wainner, S. (2005). *IPSec VPN Design*. Cisco Press, Indianapolis,IN, USA.

[6] Carmouche, J. (2007). *IPsec Virtual Private Network Fundamentals*, Cisco Press, Indianapolis, IN,U.S.A.

[7] CISCO (2014). Cisco 2014 Annual Security Report, available at http://www.cisco.com

[8] Calomel.org (2015). AES-NI SSL Performance a study of AES-NI acceleration using LibreSSL,

[9] OpenSSL, available at https://calomel.org/aesni_ssl_performance.html

[10] Crypto++ (2013). Crypto++ 5.6.0. benchmarks, available at http://www.cryptopp.com/

[11] Daemen, J., Rijmen, V. (2003). AES Proposal: Rijndael, National Institute of Standards and Technology

[12] Dworkin, M. (2001). Recommendations for Block Cipher Modes of Operation, National Institute of Standards and Technology, USA, available at http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf