Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

NAME	**:**	**Jayesh Kavedia**
SAP	**:**	**60004190053**
DIV	**:**	**A/A4**
BRANCH	**:**	**Computer Engineering**
A.Y.	**:**	**2021-22**
SEM	**:**	**VI**
SUB	**:**	**Information Security**

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Experiment 1

Date of Performance : 22-03-2022 **Date of Submission:** 23-03-2022

SAP Id: 60004190053 **Name :** Jayesh Kavedia

Div: A **Batch :** A4

Aim of Experiment

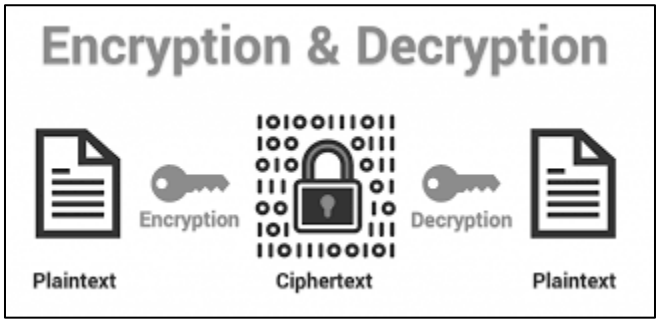
Design and Implement Encryption and Decryption Algorithm for

a. Caesar cipher cryptographic algorithm by considering letter [A..Z] and digits [0..9]. Create two functions Encrypt() and Decrypt(). Apply Brute Force Attack to reveal secret. Create Function BruteForce(). Demonstrate the use of these functions on any paragraph.
b. Hill Cipher. Your Program Must Input Image in Gray Scale. Choose keys according to Gray Scale Intensity level. Create two functions Encrypt() and Decrypt(). Make sure to have Multiplicative Inverse Exists for one of the Key in selected Key pair of Affine Cipher.

(CO1)

Theory / Algorithm / Conceptual Description

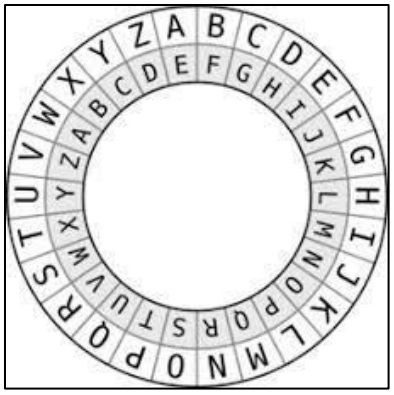
Encryption and Decryption



Encryption is the process of converting a readable message into an unreadable form so that it cannot be read by unauthorized parties. The process of converting an encrypted message back to its original (readable) format is known as decryption. The plaintext message is the original message. The ciphertext message is the encrypted message. Digital encryption techniques function by mathematically modifying the digital content of a plaintext message and producing a ciphertext version of the message using an encryption algorithm and a digital key. If the sender and recipient are the only ones who have access to the key, they can communicate safely.

Caesar Cipher Encryption Algorithm

The Caesar cipher, often known as the Shift cipher, is one of the oldest and most basic ciphers. It's a mono-alphabetic substitution cipher in which each letter in the plaintext is 'shifted' down the alphabet a fixed number of times. With a shift of one, for example, A would be replaced by B, B by C, and so on. If the attacker is aware that Caesar cipher is being used for encryption, decrypting it is simple. In this situation, the attacker can use the brute force method to determine the key by applying each key value in the range of characters and selecting the most meaningful answer.



Algorithm

CipherText = (PlainText + Key) % 26 (For A-Z)

PlainText = (CipherText - Key) % 26 (For A-Z)

Advantages

1. Only one short key is used in its entire process.
2. If a system does not use complex coding techniques, it is the best method for it.
3. It requires only a few computing resources.

Disadvantages

1. The message encrypted by this method can be easily decrypted.
2. It provides very little security.

Hill Cipher Encryption Algorithm

The Hill cipher is a polygraphic substitution cipher based on Linear Algebra principles. The Hill cipher is a more mathematical cipher than others since it uses modulo arithmetic, matrix multiplication, and matrix inverses. Because the Hill cipher is also a block cipher, it can theoretically function with blocks of any size. Encryption is based on modular arithmetic and matrix multiplication of the Key matrix and the Plain Text matrix or vector. The Cipher Text matrix or vector is multiplied by the inverse of the key matrix in modular arithmetic for decryption.

Algorithm

CipherText = (Key * PlainText) mod 26 (For A-Z)

PlainText = (Key^{-1} * CipherText) mod 26 (For A-Z)

where, Key^{-1} = ((Multiplicative Inverse of |D|) % 26 * (Adjoint (Key)) % 26) % 26 (For A-Z)

Limitation of Hill Cipher

Hill's cipher is susceptible to a known plaintext attack. If we had a set of n matching plaintext/ciphertext pairs, it is a straightforward process to recover the encryption matrix.

Program

A)

```
characters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789'
# Encryption
PT = 'UkraineIsACountryInEasternEurope'
PT = PT.upper()
CT = ''
key = 5
for char in PT:
  temp = (characters.find(char) + key)%36
  CT += characters[temp]
print('Cipher Text for given Plain Text : ',CT)
# Brute Force for Decryption
for k in range(36):
  PT = ''
  for char in CT:
    diff = characters.find(char) - k
    if diff >= 0:
      PT += characters[diff]
    else:
      PT += characters[diff + 36]
  print(f'Plain Text for Cipher Text with Key {k} is {PT}')
```

Output

Encryption
Cipher Text for given Plain Text : ZPWFNSJNXFHTZSYW3NSJFXYJWSJZWTUJ

Decryption Brute Force
Plain Text for Cipher Text with Key 0 is ZPWFNSJNXFHTZSYW3NSJFXYJWSJZWTUJ
Plain Text for Cipher Text with Key 1 is YOVEMRIMWEGSYRXV2MRIEWXIVRIYVSTI
Plain Text for Cipher Text with Key 2 is XNUDLQHLVDFRXQWU1LQHDVWHUQHXURSH
Plain Text for Cipher Text with Key 3 is WMTCKPGKUCEQWPVT0KPGCUVGTPGWTQRG
Plain Text for Cipher Text with Key 4 is VLSBJOFJTBDPVOUSZJOFBTUFSOFVSPQF
Plain Text for Cipher Text with Key 5 is UKRAINEISACOUNTRYINEASTERNEUROPE
Plain Text for Cipher Text with Key 6 is TJQ9HMDHR9BNTMSQXHMD9RSDQMDTQNOD
Plain Text for Cipher Text with Key 7 is SIP8GLCGQ8AMSLRPWGLC8QRCPLCSPMNC
Plain Text for Cipher Text with Key 8 is RHO7FKBFP79LRKQOVFKB7PQBOKBROLMB
Plain Text for Cipher Text with Key 9 is QGN6EJAEO68KQJPNUEJA6OPANJAQNKLA
Plain Text for Cipher Text with Key 10 is PFM5DI9DN57JPIOMTDI95NO9MI9PMJK9
Plain Text for Cipher Text with Key 11 is OEL4CH8CM46IOHNLSCH84MN8LH8OLIJ8
Plain Text for Cipher Text with Key 12 is NDK3BG7BL35HNGMKRBG73LM7KG7NKHI7
Plain Text for Cipher Text with Key 13 is MCJ2AF6AK24GMFLJQAF62KL6JF6MJGH6
Plain Text for Cipher Text with Key 14 is LBI19E59J13FLEKIP9E51JK5IE5LIFG5

Plain Text for Cipher Text with Key 15 is KAH08D48I02EKDJHO8D40IJ4HD4KHEF4
Plain Text for Cipher Text with Key 16 is J9GZ7C37HZ1DJCIGN7C3ZHI3GC3JGDE3
Plain Text for Cipher Text with Key 17 is I8FY6B26GY0CIBHFM6B2YGH2FB2IFCD2
Plain Text for Cipher Text with Key 18 is H7EX5A15FXZBHAGEL5A1XFG1EA1HEBC1
Plain Text for Cipher Text with Key 19 is G6DW4904EWYAG9FDK490WEF0D90GDAB0
Plain Text for Cipher Text with Key 20 is F5CV38Z3DVX9F8ECJ38ZVDEZC8ZFC9AZ
Plain Text for Cipher Text with Key 21 is E4BU27Y2CUW8E7DBI27YUCDYB7YEB89Y
Plain Text for Cipher Text with Key 22 is D3AT16X1BTV7D6CAH16XTBCXA6XDA78X
Plain Text for Cipher Text with Key 23 is C29S05W0ASU6C5B9G05WSABW95WC967W
Plain Text for Cipher Text with Key 24 is B18RZ4VZ9RT5B4A8FZ4VR9AV84VB856V
Plain Text for Cipher Text with Key 25 is A07QY3UY8QS4A397EY3UQ89U73UA745U
Plain Text for Cipher Text with Key 26 is 9Z6PX2TX7PR39286DX2TP78T62T9634T
Plain Text for Cipher Text with Key 27 is 8Y5OW1SW6OQ28175CW1SO67S51S8523S
Plain Text for Cipher Text with Key 28 is 7X4NV0RV5NP17064BV0RN56R40R7412R
Plain Text for Cipher Text with Key 29 is 6W3MUZQU4MO06Z53AUZQM45Q3ZQ6301Q
Plain Text for Cipher Text with Key 30 is 5V2LTYPT3LNZ5Y429TYPL34P2YP52Z0P
Plain Text for Cipher Text with Key 31 is 4U1KSXOS2KMY4X318SXOK23O1XO41YZO
Plain Text for Cipher Text with Key 32 is 3T0JRWNR1JLX3W207RWNJ12N0WN30XYN
Plain Text for Cipher Text with Key 33 is 2SZIQVMQ0IKW2V1Z6QVMI01MZVM2ZWXM
Plain Text for Cipher Text with Key 34 is 1RYHPULPZHJV1U0Y5PULHZ0LYUL1YVWL
Plain Text for Cipher Text with Key 35 is 0QXGOTKOYGIU0TZX4OTKGYZKXTK0XUVK

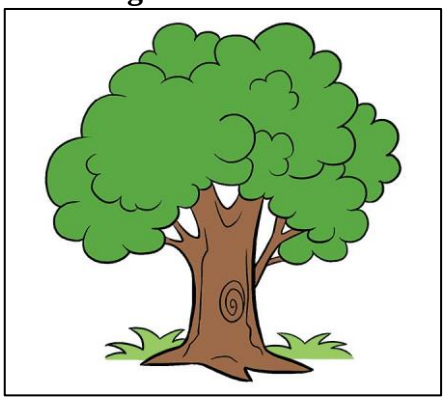
Program

B)

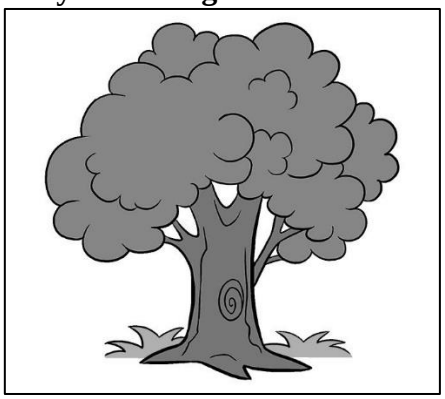
```
import cv2
import numpy as np
img = cv2.imread('tree.jpg',cv2.IMREAD_GRAYSCALE)
cv2.imwrite('gray.jpg',img)
def generate_key_matrix(n): #Creating a Involutory Matrix
  Mod = 256
  k = 23
  d = np.random.randint(256, size = (int(n/2),int(n/2)))
  I = np.identity(int(n/2))
  a = np.mod(-d,Mod)

  b = np.mod((k * np.mod(I - a,Mod)),Mod)
  k = np.mod(np.power(k,127),Mod)
  c = np.mod((I + a),Mod)
  c = np.mod(c * k, Mod)

  A1 = np.concatenate((a,b), axis = 1)
  A2 = np.concatenate((c,d), axis = 1)
  A = np.concatenate((A1,A2), axis = 0)
  return A
key = generate_key_matrix(img.shape[0])
encrypted_image = np.mod(np.matmul(key,img),256) #Encryption
cv2.imwrite('encrypted.jpg',encrypted_image)
key_inv = key # For a involutory matrix the matrix is its own inverse
decrypted_image = np.mod(np.matmul(key_inv,encrypted_image),256) #Decryption
cv2.imwrite('decrypted.jpg',decrypted_image)
```

Output

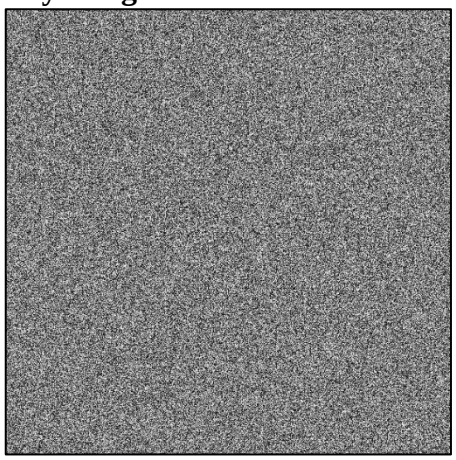
Real Image



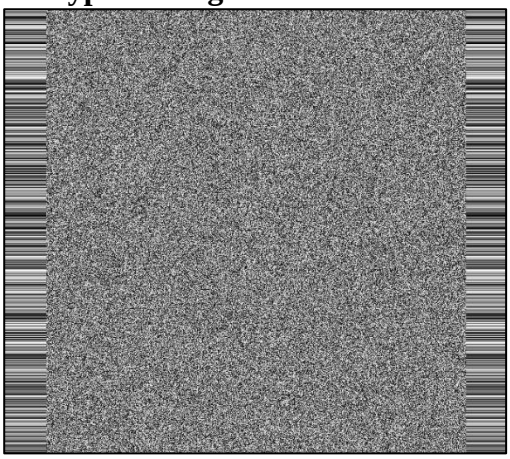
Grayscale Image



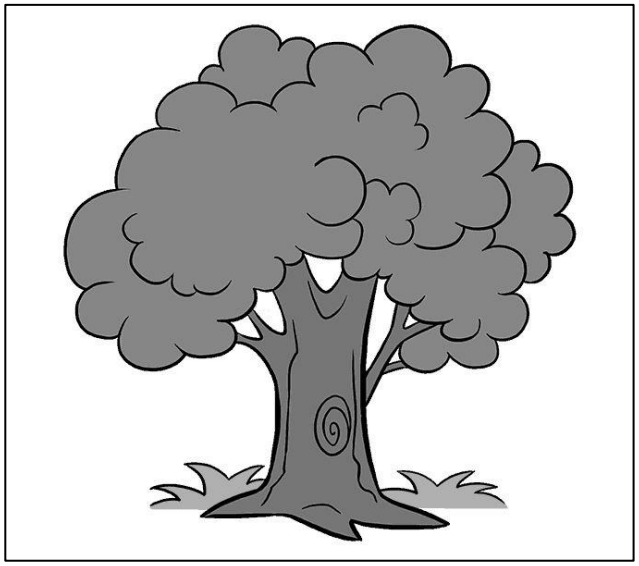
Key Image



Encrypted Image



Decrypted Image



Conclusion

Information Security of messages can be ensured by using Encryption and Decryption techniques and a common key between the sender and the receiver. Caesar Cipher Encryption Technique is the simplest substitution mono-alphabetic encryption technique. However, as it is easy to encrypt it provides very little security and can be decrypted easily. Hill Cipher is a Polygraphic substitution encryption algorithm which is more related to mathematical concepts to encrypt the data. It provides security to some extent but if set of PT's are mapped to set of CT's the key matrix can be easily determined and the data can be decrypted.

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Experiment 2

Date of Performance : 29-03-2022 **Date of Submission:** 02-04-2022

SAP Id: 60004190053 **Name :** Jayesh Kavedia

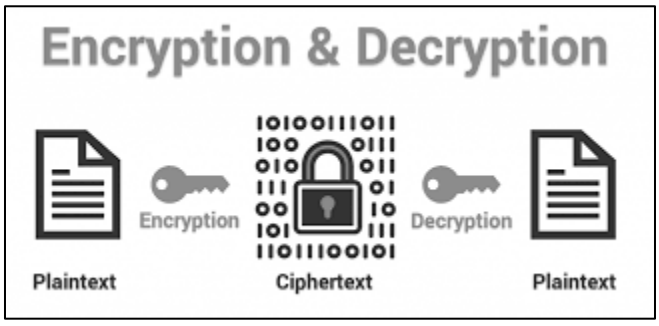
Div: A **Batch :** A4

Aim of Experiment

Design and Implement Encryption and Decryption algorithm using Simple Columnar Transposition cipher technique.

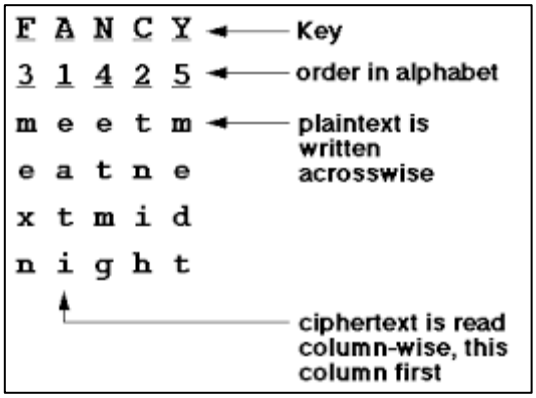
Theory / Algorithm / Conceptual Description

Encryption and Decryption



Encryption is the process of converting a readable message into an unreadable form so that it cannot be read by unauthorized parties. The process of converting an encrypted message back to its original (readable) format is known as decryption. The plaintext message is the original message. The ciphertext message is the encrypted message. Digital encryption techniques function by mathematically modifying the digital content of a plaintext message and producing a ciphertext version of the message using an encryption algorithm and a digital key. If the sender and recipient are the only ones who have access to the key, they can communicate safely.

Single Columnar Transposition Encryption Algorithm



A transposition cipher is an encryption method in which the positions of plaintext are shifted according to a regular pattern, resulting in the ciphertext being a permutation of the plaintext. Transposition ciphers include the Columnar Transposition Cipher. Columnar Transposition entails writing plaintext in rows and then reading the ciphertext one by one in columns. To decipher it, the recipient must divide the message length by the key

length to determine the column lengths. Then rewrite the message in columns, reformatting the key phrase to reorder the columns.

Algorithm

Encryption

1. The message is written in rows of a fixed length, and then read out again column by column, and the columns are chosen in some order.
2. Width of the rows and the permutation of the columns are usually defined by a keyword.
3. For example, the word LAZY is of length 4 (so the rows are of length 4), and the permutation is defined by the alphabetical order of the letters in the keyword. In this case, the order would be “2 1 4 3”.
4. Any spare spaces are filled with nulls or left blank or placed by a character (Example: *).
5. Finally, the message is read off in columns, in the order specified by the keyword.

Decryption

1. Recipient finds the column lengths by dividing the message length by the key length.
2. Write the received message in columns again, re-order the columns by reforming the key word.

Advantages

1. It is easy to understand and implement but still complex to break by brute force attack or cryptanalysis.
2. It enhances in performance as the key size is increased and by re applying the same technique.
3. Impossible to decrypt the encrypted cipher text without the knowledge of the key and the length of it and the no of columnar transpositions that were made.

Disadvantages

1. It has the disadvantage of a transposition cipher that if the plain text is very small then the attacker can very easily decrypt the encrypted text via brute force attack.

Program

```
# Importing Libraries
from math import ceil
import cv2
from google.colab.patches import cv2_imshow
import numpy as np

# Text Based Columnar Transposition
plaintext = list('meetmeattheboatclubcanteen')
key = 'monarchy'

def encryption(message, key):
  ciphertext = []
  # message_list = list(message)
  message_length = len(message)
  sorted_key = sorted(list(key))
  column = len(key)
  row = int(ceil(message_length/column))
  dummy_characters = (row*column) - message_length
  message.extend('*'*dummy_characters)
  matrix = list()
  counter = 0
  for i in range(row):
    temp = list()
    for j in range(column):
      temp.append(message[counter])
      counter += 1
    matrix.append(temp)
  counter = 0
  for j in range(column):
    index = key.index(sorted_key[counter])
    # ciphertext += ''.join([row[index] for row in matrix])
    for row in matrix:
      ciphertext.append(row[index])
    counter += 1
  return ciphertext
ciphertext = encryption(plaintext,key)
print(f'For given plaintext : {plaintext} the corresponding ciphertext is :
{ciphertext}')

def decryption(ciphertext, key):
  plaintext = []
  message_length = len(ciphertext)
  # message = list(ciphertext)
  sorted_key = sorted(list(key))
  column = len(key)
  row = int(message_length/column)
  counter = 0
  matrix = list()
  for j in range(row):
```

```
    matrix.append(['']*column)
  for j in range(column):
    index = key.index(sorted_key[j])
    for i in range(row):
      matrix[i][index] = ciphertext[counter]
      counter+=1
  for i in range(row):
    for j in range(column):
      if matrix[i][j] != '*':
        plaintext.append(matrix[i][j])
  return plaintext
plaintext = decryption(ciphertext,key)
print(f'For given ciphertext : {ciphertext} the corresponding plaintext is
: {plaintext}')

# Image Encryption using Columnar Transposition
key = "authorized"
image = cv2.imread("baboon.jpg",0)
image = np.array(image,dtype=np.int64)
cv2_imshow(image)
shape = image.shape
flat_image = list(image.flatten())
encrypted = encryption(flat_image,key)
encrypted = np.array(encrypted,dtype=np.int64)
encrypted = encrypted.reshape(shape)
cv2_imshow(encrypted)
encrypted_flat_image = list(encrypted.flatten())
decrypted = decryption(encrypted_flat_image,key)
decrypted = np.array(decrypted,dtype=np.int64)
decrypted = decrypted.reshape(shape)
cv2_imshow(decrypted)
```

Output

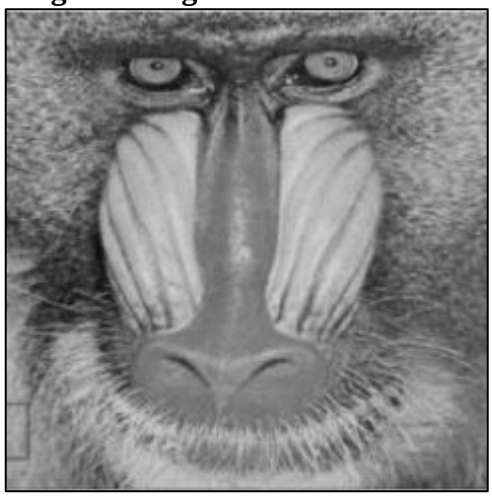
Text Based Columnar Transposition

Encryption
For given plaintext : meetmeattheboatclubcanteen****** the corresponding ciphertext is :tbc*ean*att*mtleeeb*ehunmoa*tce*

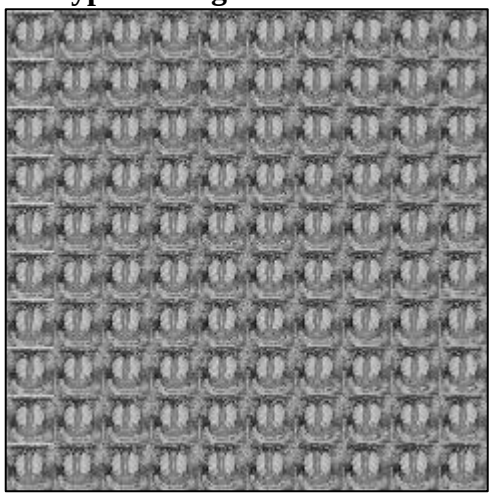
Decryption
For given ciphertext : tbc*ean*att*mtleeeb*ehunmoa*tce* the corresponding plaintext is :meetmeattheboatclubcanteen

Image Encryption using Columnar Transposition

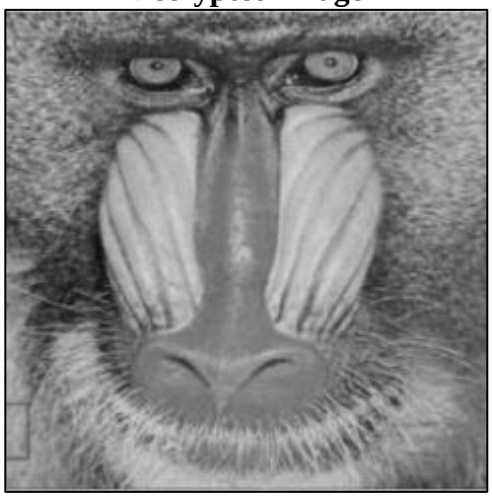
Original Image



Encrypted Image



Decrypted Image



Conclusion

Transposition Ciphers encrypt data by shuffling the original plaintext by some order and converting to a scrambled ciphertext. On receivers end the ciphertext is unscrambled using same order and converted to original plain text. Single Columnar Transposition is a type of Transposition technique where the order is decided by the alphabetical sequence of the key used. It is easy to understand but is complex to break through brute force attack. However it can be susceptible to brute force attack if the length of the message is too small.

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Experiment 3

Date of Performance : 29-03-2022

Date of Submission: 02-04-2022

SAP Id: 60004190053

Name : Jayesh Kavedia

Div: A

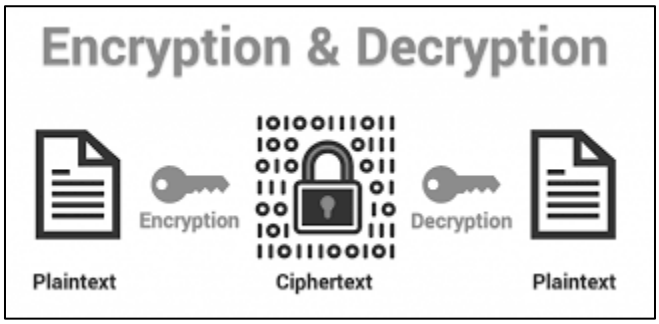
Batch : A4

Aim of Experiment

Design and Implement Encryption and Decryption Algorithm for Simplified Data Encryption Standard (S-DES).

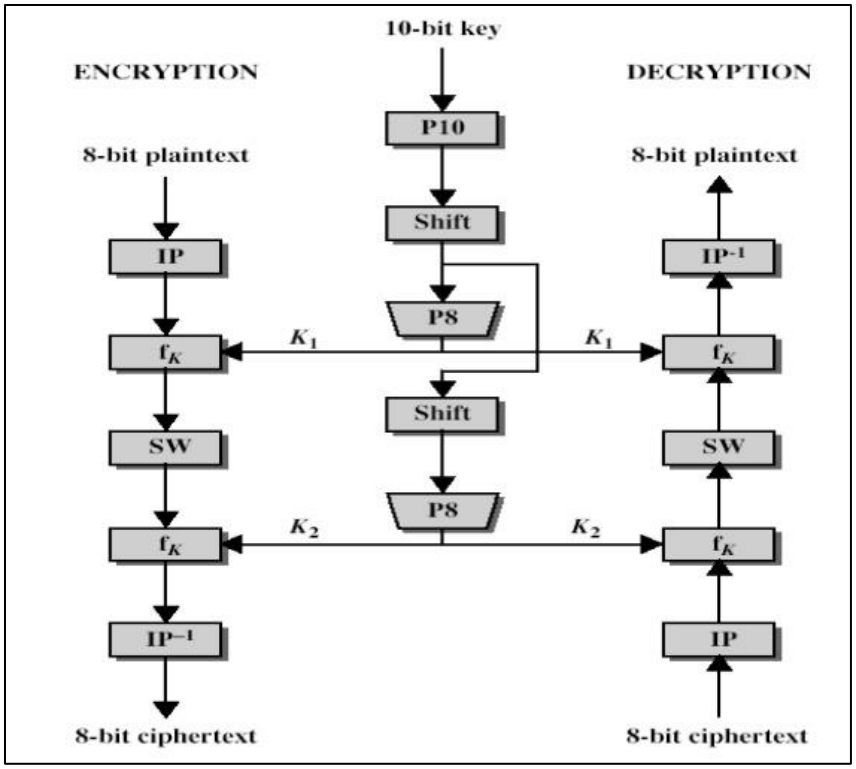
Theory / Algorithm / Conceptual Description

Encryption and Decryption



Encryption is the process of converting a readable message into an unreadable form so that it cannot be read by unauthorized parties. The process of converting an encrypted message back to its original (readable) format is known as decryption. The plaintext message is the original message. The ciphertext message is the encrypted message. Digital encryption techniques function by mathematically modifying the digital content of a plaintext message and producing a ciphertext version of the message using an encryption algorithm and a digital key. If the sender and recipient are the only ones who have access to the key, they can communicate safely.

Simplified Data Encryption Standard (S-DES)



The DES Algorithm is simplified in the Simplified Data Encryption Standard (S-DES). It's comparable to the DES algorithm, however it's a smaller version with less parameters. It is a block cipher that turns plaintext into ciphertext. It requires an 8-bit block. It's a symmetric key cipher, which means the encryption and decryption keys are the same.

The algorithm consists of five functions: an initial permutation (IP), a complex function labelled f_k that combines permutation and substitution operations and is dependent on a key input, a simple permutation function that switches (SW) the two halves of the data, the function f_k once more, and a permutation function (IP^{-1}) that is the inverse of the initial permutation. The function f_k accepts an 8-bit key as well as the data travelling through the encryption process. Two eight-bit subkeys are created from a ten-bit key. A permutation test is performed on the key initially (P_{10}). After that, a shift operation is carried out. The shift operation's output is then sent through a permutation function, which generates an 8-bit output (P_8) for the first subkey (K_1). The second subkey is created by feeding the output of the shift operation into another shift and another instance of P_8 (K_2).

Algorithm

Ciphertext = IP^{-1} (f_{K2} (SW (f_{k1} (IP (Plaintext)))))

Plaintext = IP^{-1} (f_{K1} (SW (f_{k2} (IP (Ciphertext)))))

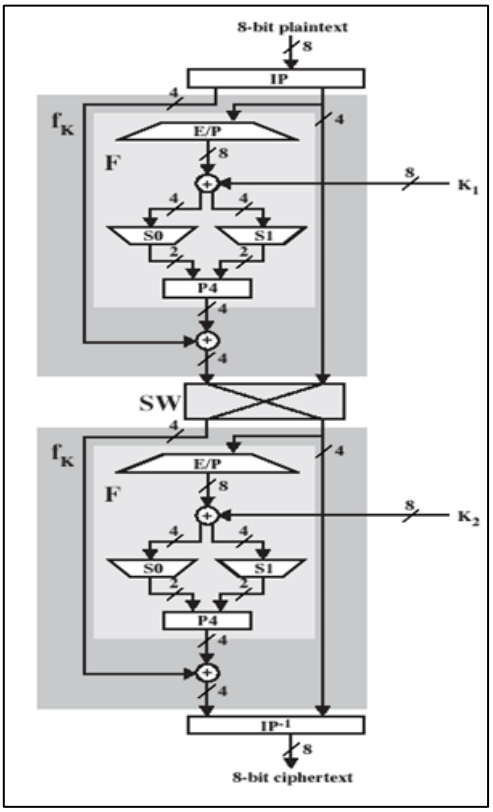
Where,

$K_1 = P_8$ (Shift (P_{10} (Key)))

$K_2 = P_8$ (Shift (Shift (P_{10} (Key))))

Complex Function f_k

The complex function f_k receives input in two halves : left half contains 4 bits as well as the right half contains 4 bits, the right 4 bits are provided as input to expansion block which provides output of 8 bits. These 8 bits are Ex-ORed with input key, the output is again divided into 2 halves. The decimal value of 1st and 4th bits of the 4 bits provide row number whereas the decimal value of 2nd and 3rd bits provide column number. The calculated row number and column number are looked into the respective substitution matrix and the value represented by the row number and column number in the matrix is converted to 2-bit binary number and provided as output. The two, two-bit outputs are concatenated and provided as input to the P_4 block, the output is then Ex-ORed with the initial left 4-bits. The output of Ex-OR is the left half of the function output whereas the right half of the function output are the initial right 4-bits.



Program

```
import numpy as np

#INITIALIZATION

initial_key = np.array([1,0,1,1,0,0,1,1,0,1]) #Example 1
# initial_key = np.array([1,0,1,0,1,0,1,1,0,1]) #Example 2
p10 = np.array([3,5,2,7,4,10,1,9,8,6])
p8 = np.array([6,3,7,4,8,5,10,9])
p4 = np.array([2,4,3,1])
ip = np.array([2,6,3,1,4,8,5,7])
ip_inv = np.array([4,1,3,5,7,2,8,6])
ep = np.array([4,1,2,3,2,3,4,1])

#KEY GENERATION"

key = initial_key[p10 - 1]
left = np.roll(key[:5],-1)
right = np.roll(key[5:], -1)
key1 = np.concatenate((left,right))
```

```
key1 = key1[p8 - 1]
print('Key 1 : ',key1)
left = np.roll(left,-2)
right = np.roll(right, -2)
key2 = np.concatenate((left,right))
key2 = key2[p8 - 1]
print('Key 2 : ',key2)

#SUBSTITUTION MATRIX

s0 = np.array([[1,0,3,2],[3,2,1,0],[0,2,1,3],[3,1,3,2]])
s1 = np.array([[0,1,2,3],[2,0,1,3],[3,0,1,0],[2,1,0,3]])

#COMPLEX FUNCTION

def complex_function(left,right,key,ep,p4,s0,s1):
  intermediate_text = right[ep - 1]
  intermediate_text = np.bitwise_xor(intermediate_text,key)
  left_s = intermediate_text[:4]
  right_s = intermediate_text[4:]
  s0_op_row = left_s[[0,3]]
  s0_op_row = [val.item() for val in s0_op_row]
  s0_op_row = int(''.join(list(map(str,s0_op_row))),2)
  s0_op_col = left_s[[1,2]]
  s0_op_col = [val.item() for val in s0_op_col]
  s0_op_col = int(''.join(list(map(str,s0_op_col))),2)
  s1_op_row = right_s[[0,3]]
  s1_op_row = [val.item() for val in s1_op_row]
  s1_op_row = int(''.join(list(map(str,s1_op_row))),2)
  s1_op_col = right_s[[1,2]]
  s1_op_col = [val.item() for val in s1_op_col]
  s1_op_col = int(''.join(list(map(str,s1_op_col))),2)
  s0_op = bin(s0[s0_op_row][s0_op_col]).replace('0b','').zfill(2)
  s1_op = bin(s1[s1_op_row][s1_op_col]).replace('0b','').zfill(2)
  intermediate_text = list(s0_op + s1_op)
  intermediate_text = np.array(intermediate_text,dtype=np.int64)
  intermediate_text = intermediate_text[p4 - 1]
  intermediate_text = np.bitwise_xor(intermediate_text,left)
  return intermediate_text, right

#ENCRYPTION
pt = 'CA' #Example1
# pt = 'FC' #Example2
```

```
print('Encryption :\nPlain Text to be Encrypted : ',pt)
pt = list(bin(int(pt,16)).replace('0b','').zfill(8))
pt = np.array(pt,dtype=np.int64)
print('8-bit Plain Text : ',pt)
intermediate_text = pt[ip - 1]
left = intermediate_text[:4]
right = intermediate_text[4:]
left,right = complex_function(left,right,key1,ep,p4,s0,s1)
left,right = right,left
left,right = complex_function(left,right,key2,ep,p4,s0,s1)
intermediate_text = np.concatenate((left,right))
ct = intermediate_text[ip_inv - 1]
print('8-bit Cipher Text : ',ct)
ct = [val.item() for val in ct]
ct = int(''.join(list(map(str,ct))),2)
ct = hex(ct).replace('0x','')
print('Encrypted Text / Cipher Text : ',ct.upper())

#DECRYPTION
ct = '2D' #Example 1
# ct = '28' #Example 2
print('Decryption :\nCipher Text to be Decrypted : ',ct)
ct = list(bin(int(ct,16)).replace('0b','').zfill(8))
ct = np.array(ct,dtype=np.int64)
print('8-bit Cipher Text : ',ct)
intermediate_text = ct[ip - 1]
left = intermediate_text[:4]
right = intermediate_text[4:]
left,right = complex_function(left,right,key2,ep,p4,s0,s1)
left,right = right,left
left,right = complex_function(left,right,key1,ep,p4,s0,s1)
intermediate_text = np.concatenate((left,right))
pt = intermediate_text[ip_inv - 1]
print('8-bit Plain Text : ',pt)
pt = [val.item() for val in pt]
pt = int(''.join(list(map(str,pt))),2)
pt = hex(pt).replace('0x','')
print('Decrypted Text / Plain Text : ',pt.upper())
```

Output

Example 1
Key 1 : [1 1 0 1 1 1 1 0]
Key 2 : [1 1 0 0 1 0 0 1]
Encryption :
Plain Text to be Encrypted : CA
8-bit Plain Text : [1 1 0 0 1 0 1 0]
8-bit Cipher Text : [0 0 1 0 1 1 0 1]
Encrypted Text / Cipher Text : 2D
Decryption :
Cipher Text to be Decrypted : 2D
8-bit Cipher Text : [0 0 1 0 1 1 0 1]
8-bit Plain Text : [1 1 0 0 1 0 1 0]
Decrypted Text / Plain Text : CA

Example 2
Key 1 : [1 1 0 0 1 1 1 0]
Key 2 : [1 1 0 1 1 0 0 1]
Encryption :
Plain Text to be Encrypted : FC
8-bit Plain Text : [1 1 1 1 1 1 0 0]
8-bit Cipher Text : [0 0 1 0 1 0 0 0]
Encrypted Text / Cipher Text : 28
Decryption :
Cipher Text to be Decrypted : 28
8-bit Cipher Text : [0 0 1 0 1 0 0 0]
8-bit Plain Text : [1 1 1 1 1 1 0 0]
Decrypted Text / Plain Text : FC

Conclusion

DES/SDES are block cipher encryption algorithm. DES encrypts blocks of 64 bits with help of 56-bit keys over 16 rounds. SDES is simplified version of DES which encrypts 8-bit blocks with help of 8-bit keys over 2 rounds. The SDES algorithm was made for understanding the DES algorithm. Since the size of key is 56 bits in DES, it takes 2^{56} attempts in brute force method also the algorithm is complex which increases the complexity to crack it.

Playfair Image Encryption

1st Jayesh Kavedia
Department of Computer Engineering
Dwarkadas J. Sanghvi college of Engineering
Mumbai, India
kavediajayesh26@gmail.com

2nd Jigar Shah
Department of Computer Engineering
Dwarkadas J. Sanghvi college of Engineering
Mumbai, India
shahjigar185@gmail.com

3rd Jazib Dawre
Department of Computer Engineering
Dwarkadas J. Sanghvi college of Engineering
Mumbai, India
jazib980@gmail.com

4th Dr. Ramchandra Mangrulkar
Department of Computer Engineering
Dwarkadas J. Sanghvi college of Engineering
Mumbai, India
ramchandra.mangrulkar@djsce.ac.in

***Abstract*—In today's social networking era, it's no surprise that information security breaches are at an all-time high, especially when it comes to texts and photographs. As a result, for information to be communicated across an insecure channel, integrity is required. As a result, image/text encryption is an important preprocessing step in today's data transmission. Encryption of images has been demonstrated to be a viable technique of communicating sensitive information, resulting in a plethora of procedures. Here the method employed is modified Playfair technique for image encryption consisting of three stages stage 1 consist of key generation stage 2 consist of constructing Playfair matrix stage 3 consist of encrypting pixel value of each image. The approach uses a 160 bit key which requires 2^{160} attempts for a brute force attack, which is practically infeasible. The approach discussed here can be used for secured encryption of images in various multimedia systems.**

***Index Terms*—Cryptography, Image Encryption, Block Cipher, Playfair Cipher, Internet Security**

I. Introduction

Communication and data are at the center of our world in the twenty-first century. The data contains confidential information and needs to be secured. Cryptography is the field where the information is converted to an unreadable format using various techniques to preserve confidentiality of data. Images play a critical role in the millions of multimedia transfers that occur on a daily basis. In terms of the safe transfer of private information including images, image cryptography has proven to be a very important domain.

Encryption and data-hiding techniques such as steganography are some of the methods for securing images and data privacy. Various researchers have contributed a range of distinct image encryption algorithms, each with their own concepts, efficiency, working scopes, and application fields in mind. Various standard techniques that apply to photos and videos in order to encrypt them efficiently are already available. Image encryption has applications in corporate world, health care, military operations, and multimedia systems.

The paper introduces and investigates a novel image encryption technique that is both efficient and secure. The approach uses concepts of Playfair Cipher, XOR operation, permutation blocks and CBC mode of operation to encrypt images. The modified Playfair cipher is a symmetric digraph substitution cipher with a 4x4 combination where each cell value consists of a unique hexadecimal value. Each pixel value is divided into two halves of 4 bit converted to hex values and sent to Playfair cipher for encryption. The technique is capable of encryption and decryption of all three planes of an image. The algorithm provides protection to brute force attacks, good confusion and diffusion which makes it a secured algorithm. [1]

The rest of the paper is organized as follows : Section 2 contains Literature Survey for the various image encryption techniques and cryptographic principles. Section 3 contains the proposed methodology with the algorithm for the approach and security features of the algorithm. Section 4 contains the experimental setup used to perform encryption and decryption as well as the results achieved from them are discussed in this section. Section 5 concludes the paper and various references are provided.

II. Literature Survey

Playfair cipher: The Playfair cipher is a message encryption technique. Instead of a single letter, it encrypts a digraph (a pair of two characters). It starts by generating a 5*5 matrix key-table. The matrix contains alphabets that serve as the plaintext's encryption key. No alphabet should be used twice. There are 26 alphabets and only 25 blocks to fit a letter into. As a result of the surplus letter, a letter (typically J) will be omitted from the matrix. The message is encrypted by digraph with the Playfair cipher. As a result, the Playfair cipher might be considered a digraph substitution cipher. [2]

DES: DES is a block cipher that encrypts data in 64-bit blocks. Encryption and decryption employ the same algorithm and key, with slight variations. DES uses a 56-bit key. The initial key is 64 bits long. DES is built on two cryptographic fundamentals: substitution (also known as confusion) and transposition (also called diffusion). For transposition it uses

Permutations blocks and for substitution it uses Substitution blocks. DES is made up of 16 steps, each of which is referred to as a round. Substitution and transposition are performed in each round. [3]

Block cipher: A block cipher uses a cryptographic key and algorithm to encrypt data in blocks to produce ciphertext. In contrast to a stream cipher, which encrypts data one bit at a time, the block cipher processes fixed-size blocks at the same time. The majority of current block ciphers encrypt data in fixed-size blocks of 64 or 128 bits. [4]

CBC (Cipher block chaining): Cipher Block Chaining Mode (CBC) is used when we have similar Plain Text and want to produce different cipher Text for the same. [5]

Image Encryption: "Ref. [6]" uses a 16 x 16 matrix for the Playfair matrix as opposed to the traditional 5 x 5 matrix, where each cell of the matrix represents a unique pixel value between 0 - 255 for image encryption. In case of the same values of Plain Text the approach chooses to select the right diagonal value as the Cipher Text.

"Ref. [7]" modifies the traditional Playfair by using a trigraph substitution instead of digraph substitution which means instead of taking 2 values at a time it uses three values at a time. 16 4x4 matrices are used for creating the Playfair key matrix consisting values between 0 -255 for image encryption.

The drawback of these two approaches is the key matrix size, one uses a 16 x 16 2 Dimensional matrix whereas the other uses 16 4 x 4 matrix.

III. Proposed Methodology

This section introduces a new block cipher method to encrypt images using Playfair Cipher with some modifications to the traditional one. The Playfair Image Encryption technique can be divided into three stages to encrypt all three planes of an image as shown in Fig. 1. It uses key generation, Playfair matrix and xor operation to encrypt all the pixels of an image. The first stage consists of key generation, the second stage consists of generating a Playfair matrix from the given round key and the final stage consists of using the round key and the Playfair matrix to encrypt the pixels of the image.

A. Encryption

Stage 1 : Stage 1 consists of key generation, as seen in Fig. 2, an initial key of 160 bits is randomly generated. 10 consecutive bits of the key are used to produce 16 round keys for the encryption process. Each 10 consecutive bits are passed to P10 and P8 permutation blocks to generate 16 8-bit round keys for the encryption process.

Stage 2 : Stage 2 consists of constructing the Playfair matrix which is to be used for the encryption process. The Playfair matrix is a 4 x 4 matrix as opposed to the traditional 5 x 5 matrix. Each cell of the Playfair matrix consists of a unique hex value from all the hexadecimal digits. For each round a single Playfair matrix is used which is constructed using the round key corresponding to the round. The round key is

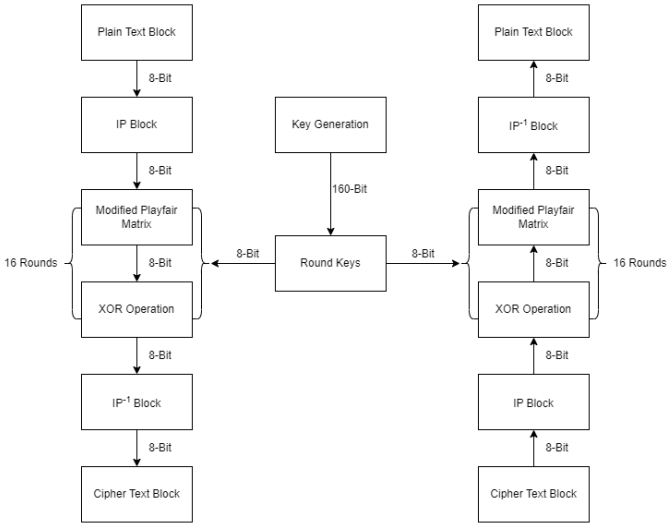


Fig. 1: Block Diagram of Proposed Methodology

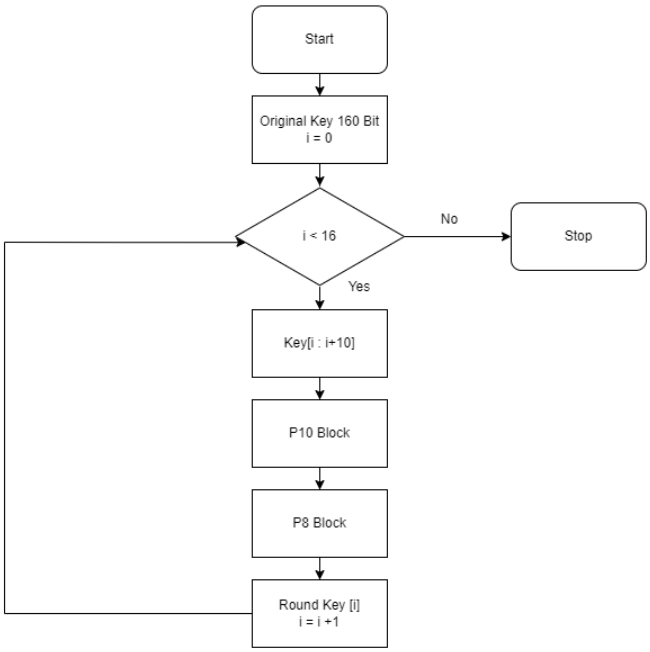


Fig. 2: Block Diagram of Key Generation

divided into two halves and converted to hex and placed in the matrix and all remaining hex values are placed after them. This constructed Playfair matrix is then used for encryption as well as the decryption process.

Stage 3 : Stage 3 is the stage where the pixel value of an image gets encrypted, the encryption process starts by considering each plane of an image (R,G,B). The pixel value from each plane is considered and converted to binary equivalent of 8 bits. The preprocessing step is to pass each pixel value to the IP (Initial Permutation) block. The binary equivalent is then converted to hex representation and passed to the Playfair matrix for encryption. Playfair encryption is

similar to the traditional Playfair encryption except that in case of the same plain text values instead of adding a dummy character it chooses the bottom value as the cipher text value. The hex value from the Playfair encryption is again converted to binary equivalent and is exored with the round key. This process is continued till 16 rounds. The output from the 16^{th} round is passed as input to IP^{-1} (Inverse Initial Permutation) block to get the cipher text value of the pixel.

B. Decryption

The Decryption process is the exact reverse of the encryption process except the position of IP and IP^{-1} blocks are swapped. The same round keys and Playfair matrices are used which were used for the encryption process. The Cipher text value of the pixel is passed to the IP block. The output is processed for 16 rounds of XOR operation and Playfair Decryption process. The Playfair decryption is similar to the traditional one except that for the same values of cipher text the decryption process chooses the corresponding top value as the plain text value. The output of the 16^{th} round is passed to the IP^{-1} block, the output of which gives the plain text value of the pixel.

C. Cipher Block Chaining Mode

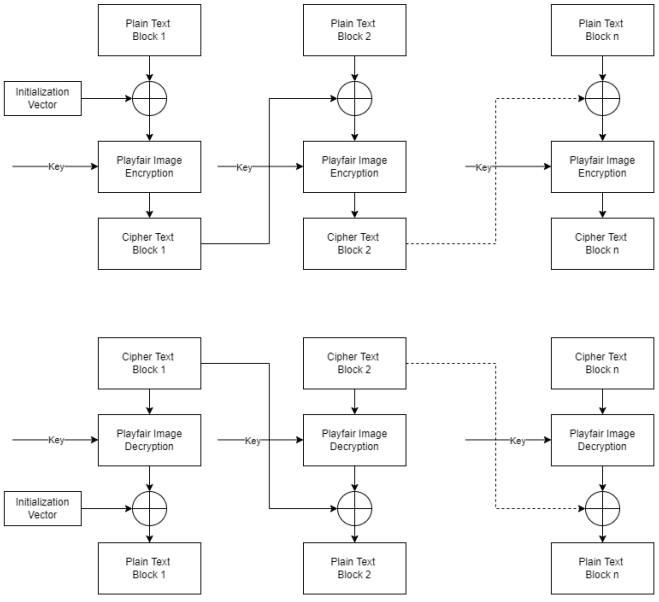


Fig. 3: Block Diagram of Cipher Block Chaining Module

If the same plain text values are passed to the block cipher then the corresponding ciphertext values are also the same. This can help the attacker to find patterns and attack the encryption process. In case of images there can be multiple similar pixel values, the approach uses Cipher Block Chaining Mode (CBC), Fig. 3, to prevent any such pattern being formed. CBC mode passes the Cipher text value of the current plain text to the next plain text value, where they are exored and passed to the encryption algorithm. The decryption is exactly the reverse of the encryption. With the help of CBC mode for the same plain text value corresponding ciphertext value will be different, thus confusing the attacker.

D. Security

The Playfair Image Encryption Technique uses a 160 bit key for the encryption and decryption process. By applying a brute force attack to the technique an attacker will require 2^{160} attempts to crack the technique which is large enough to protect the brute force attack. The technique uses Playfair substitution cipher which achieves the confusion principle of secured cryptography. It uses permutation blocks which achieves the diffusion principle of secured cryptography. Further CBC mode is used to ensure the same plain text values don't generate the same ciphertext values.

IV. Algorithmic Modules

A. Key Generation

Input : 160 bit key

1) Initialize P10, P8 blocks, RoundKeys.
2) For i : 1 to n ; step = 10 do
    a) RoundKey = key [i : i+10]
    b) RoundKey = RoundKey[P10]
    c) RoundKey = RoundKey[P8]
    d) Append RoundKey to RoundKeys

Output : RoundKeys of length 16

B. Constructing Playfair Matrix

Input : Key

1) Initialize KeyMatrix
2) KeyMatrix[1][1] = Key[1]
3) If Key [2] not in KeyMatrix then KeyMatrix[1][2] = Key[2]
4) Fill Matrix by the remaining hexadecimal values not present in KeyMatrix.

Output : KeyMatrix (4 x 4)

C. Playfair Encryption

Input : PlainText, Playfair Matrix

1) If PlainText[1] and PlainText[2] lie in the same row then return immediate values to the right in a circular manner.
2) If PlainText[1] and PlainText[2] lie in the same column then return immediate values to the bottom in a circular manner.
3) If PlainText[1] and PlainText[2] lie in the same cell then return immediate values to the bottom in a circular manner.
4) If PlainText[1] and PlainText[2] lie in the different row and column then return intersection values of the row and column.

Output : CipherText

D. Image Encryption

Input : Image, Round Keys

1) For each plane in Image (i: 1 to 3)
    a) For k : 1 to 16
        i) Key = RoundKeys[k]
        ii) Matrix = ConstructPlayfairMatrix(Key)
        iii) For i : 1 to length(rows)
            A) For j : 1 to length(columns)
            B) Pixel = Image[i][j]
            C) For Round 1 : Pixel = Pixel[IP]
            D) Pixel = PlayfairEncryption(Pixel,Matrix)
            E) Pixel = Pixel XOR Key
            F) For Round 16 : Pixel = Pixel[IP^{-1}]

Output : EncryptedImageMatrix

V. Experimental Setup

Google Colab was used to run the algorithm on images. It is a cloud environment for running of python scripts and was used for collaboration and consistent common environments. Specifications of Colab are as follows:

OS: Ubuntu 18.04.5 LTS x86_64
Host: Google Compute Engine
Kernel: 5.4.188+
Shell: bash 4.4.20
Terminal: jupyter-notebook
CPU: Intel Xeon (2) @ 2.199 GHz
Memory: 12986 MiB

Programming language used to code the algorithm was Python as it is relatively easy to code and has great community support. Version of Python used was 3.8

Numpy library was used to perform operations on multidimensional arrays as it has faster execution of arrays. Version of Numpy used : 1.21.0

The OpenCV library was used to work with images as it is efficient. Version of OpenCV used : 4.5.5.

VI. Results

The Playfair Image Encryption Technique was used to encrypt and decrypt 5 RGB images of 200 x 200. The following results were achieved :

a) Analysis of Images based on Original and Decrypted Images: The original and decrypted images were compared and the difference between the images were calculated and a new image was created showing the difference. The results can be seen in Fig. 4. The complete black image of difference represents no data was lost from the original image after decryption. Thus the image encryption process is smooth and without any loss of data.

b) Analysis of Image Histograms: Histograms of images represent the number of pixels at each intensity level. The original image generally has a non uniform distributed intensity, whereas the encrypted image has the same or nearly similar distribution of the intensity values which makes it a secure encrypted image. The histograms of various original images and encrypted images are plotted and the result can be seen in the Fig. 5.

VII. Conclusion

The paper introduced a new image encryption technique using modified Playfair Encryption technique and Block Cipher. It uses a three stage approach to encrypt an image : Key Generation, Matrix Construction, Image Encryption. The approach uses CBC mode of operation. The 160 bit key used makes the algorithm robust to brute force attacks as it requires 2^{160} attempts. It also provides good confusion and diffusion which makes it suitable for usage in real life applications for Image Encryption in multimedia systems, military operations etc.

References

[1] Oad, A., Yadav, H. and Jain, A., 2014. A review: image encryption techniques and its terminologies. International Journal of Engineering and Advanced Technology (IJEAT) ISSN, pp.2249-8958. https://www.ijeat.org/wp-content/uploads/papers/v3i4/D2998043414.pdf

[2] Deepthi, R., 2017. A survey paper on playfair cipher and its variants. Int. Res. J. Eng. Technol, 4(4), pp.2607-2610. https://www.irjet.net/archives/V4/i4/IRJET-V4I4642.pdf

[3] Singh, S., Maakar, S.K. and Kumar, D.S., 2013. Enhancing the security of DES algorithm using transposition cryptography techniques. International Journal of Advanced Research in Computer Science and Software Engineering, 3(6), pp.464-471. https://www.researchgate.net/publication/310828708_Enhancing_the_security_of_des_algorithm_using_transposition_cryptography_techniques

[4] RadiHamade, F., 2016. Survey: Block cipher Methods. International Journal of Advancements in Research & Technology, 5(11). https://www.researchgate.net/publication/309478984_Survey_Block_cipher_Methods

[5] Pavan, A.C. and Magadum, S.S., CIPHER BLOCK CHAINING MODE USING AES. https://www.ijisrt.com/assets/upload/files/IJISRT20MAR059_(1).pdf

[6] Adam, F.A.E., 2018. Enhanced Three Dimension Playfair Algorithm for Image Encryption (Doctoral dissertation, Sudan University of Science and Technology). http://repository.sustech.edu/bitstream/handle/123456789/22845/Enhanced%20Three%20Dimension%20...%20.pdf?sequence=1&isAllowed=y

[7] Chakravarthy, S., Venkatesan, S.P., Anand, J.M. and Ranjani, J.J., 2016. Enhanced playfair cipher for image encryption using integer wavelet transform. Indian Journal of Science and Technology, 9(39), pp.1-12. https://sciresol.s3.us-east-2.amazonaws.com/IJST/Articles/2016/Issue-39/Article33.pdf

[8] Bhat, K., Mahto, D., Yadav, D.K., Azad, C. (2021). Image Security Using Hyperchaos and Multidimensional Playfair Cipher. In: Stănică, P., Gangopadhyay, S., Debnath, S.K. (eds) Security and Privacy. Lecture Notes in Electrical Engineering, vol 744. Springer, Singapore. https://link.springer.com/chapter/10.1007/978-981-33-6781-4_8

[9] C. Pradhan, B. J. Saha, K. K. Kabi and Arun, "Comparative analysis of digital watermarking scheme using enhanced playfair cipher in DCT & DWT," Fifth International Conference on Computing, Communications and Networking Technologies (ICCCNT), 2014, pp. 1-6, doi: 10.1109/ICCCNT.2014.6963114.

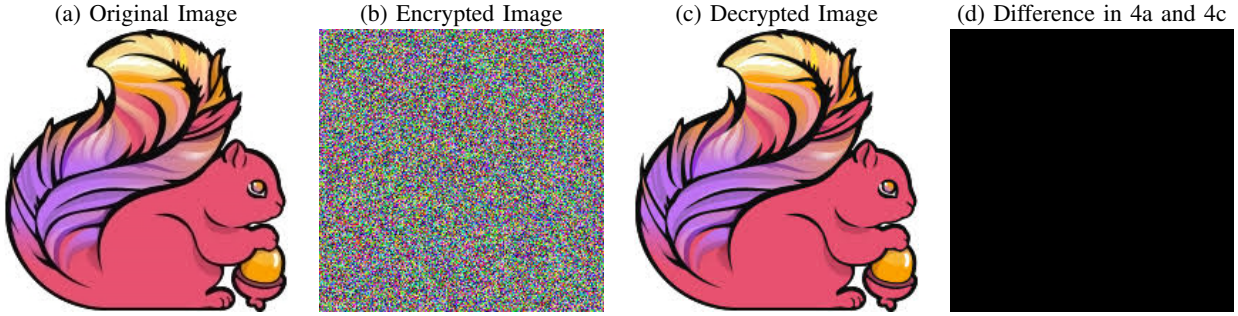
[10] S. Hans, R. Johari and V. Gautam, "An extended Playfair Cipher using rotation and random swap patterns," 2014 International Conference on Computer and Communication Technology (ICCCT), 2014, pp. 157-160, doi: 10.1109/ICCCT.2014.7001485.

[11] Y. Wang, "A Classical Cipher-Playfair Cipher and Its Improved Versions," 2021 International Conference on Electronic Information Engineering and Computer Science (EIECS), 2021, pp. 123-126, doi: 10.1109/EIECS53707.2021.9587989.

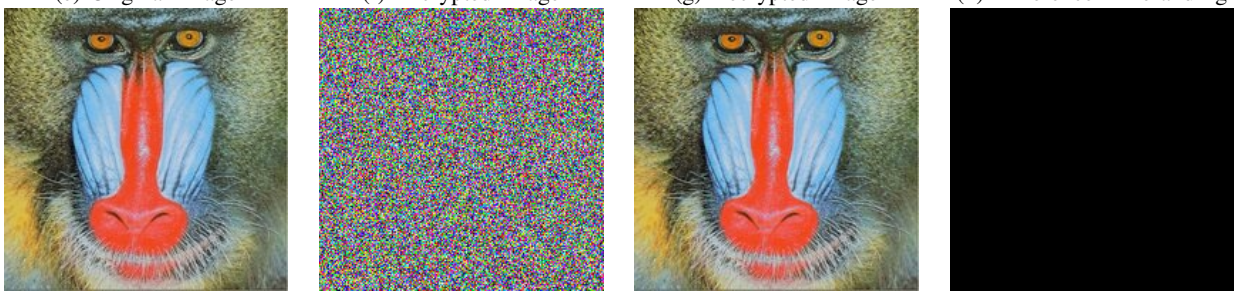
[12] A. Hyder, M. Dhande, K. Dharod, M. Lohar and K. Makan, "Text to Image Encryption Using 16 16 Playfair Cipher and Dynamic Color Channel Selection Technique," 2018 3rd International Conference for Convergence in Technology (I2CT), 2018, pp. 1-4, doi: 10.1109/I2CT.2018.8529718.

[13] P. Goyal, G. Sharma and S. S. Kushwah, "A New Modified Playfair Algorithm Using CBC," 2015 International Conference on Computational Intelligence and Communication Networks (CICN), 2015, pp. 1008-1012, doi: 10.1109/CICN.2015.200.

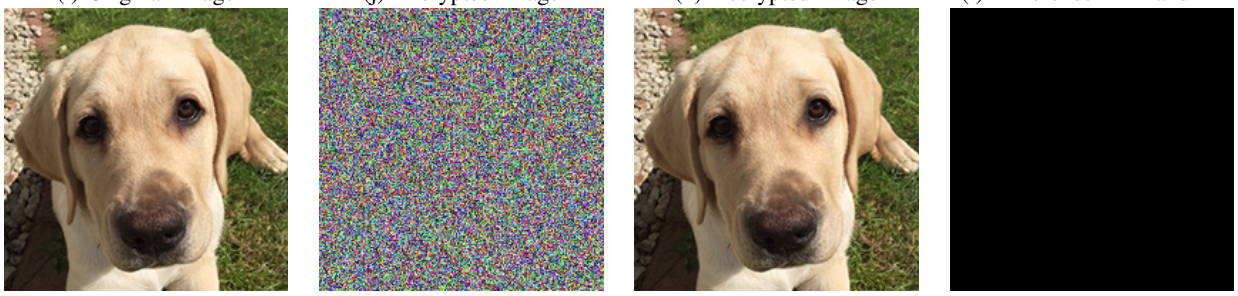
(a) Original Image (b) Encrypted Image (c) Decrypted Image (d) Difference in 4a and 4c



(e) Original Image (f) Encrypted Image (g) Decrypted Image (h) Difference in 4e and 4g



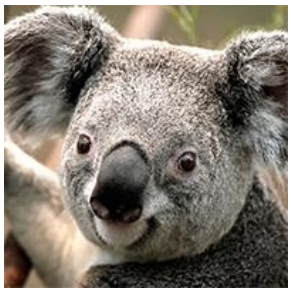
(i) Original Image (j) Encrypted Image (k) Decrypted Image (l) Difference in 4i and 4k



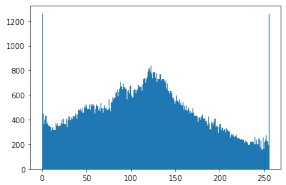
(m) Original Image (n) Encrypted Image (o) Decrypted Image (p) Difference in 4m and 4o

(q) Original Image (r) Encrypted Image (s) Decrypted Image (t) Difference in 4q and 4s

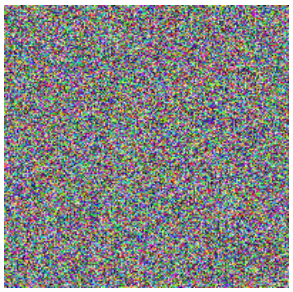
Fig. 4: Analysis of images based on original and decrypted images



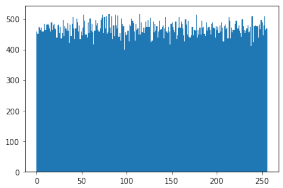
(a) Original Image



(b) Histogram for 5a

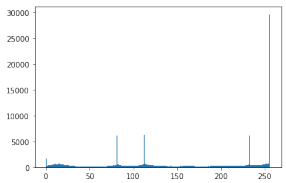


(c) Encrypted Image

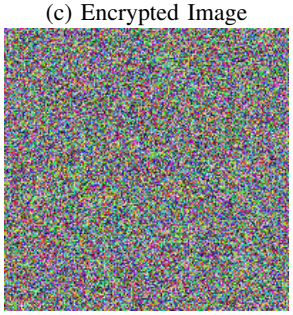


(d) Histogram for 5d

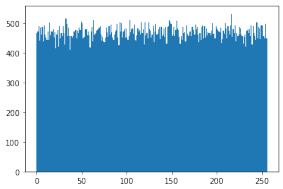
(e) Original Image



(f) Histogram for 5e

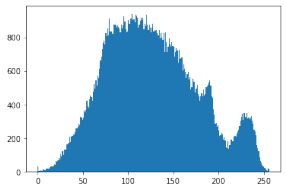


(g) Encrypted Image

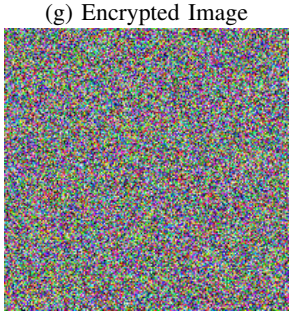


(h) Histogram for 5g

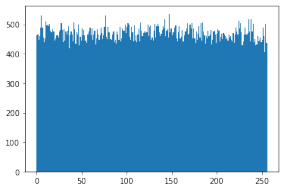
(i) Original Image



(j) Histogram for 5i

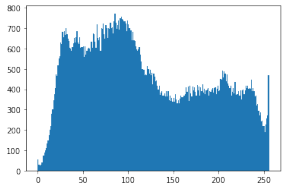


(k) Encrypted Image

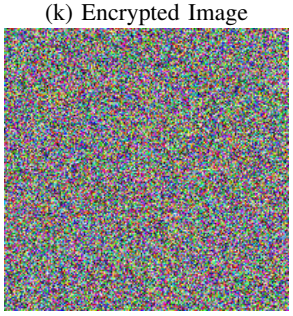


(l) Histogram for 5k

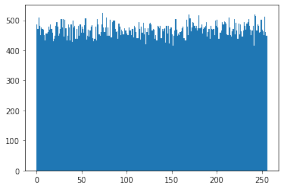
(m) Original Image



(n) Histogram for 5m

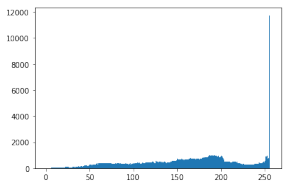


(o) Encrypted Image

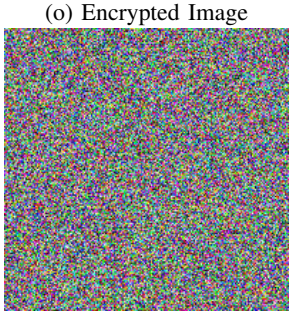


(p) Histogram for 5o

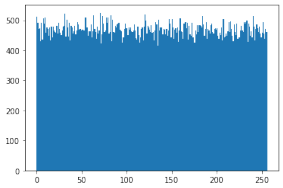
(q) Original Image



(r) Histogram for 5q



(s) Encrypted Image



(t) Histogram for 5s

Fig. 5: Histogram analysis of encrypted and original images

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Experiment 5

Date of Performance : 12-04-2022

Date of Submission: 13-04-2022

SAP Id: 60004190053

Name : Jayesh Kavedia

Div: A

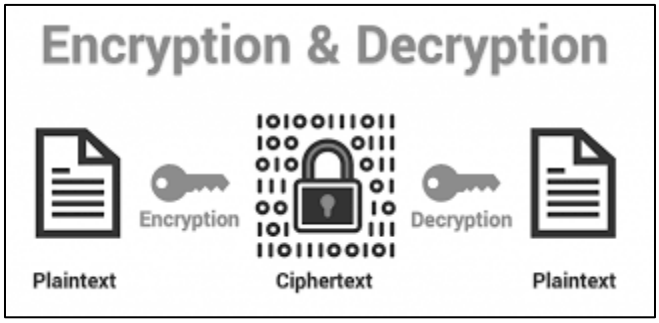
Batch : A4

Aim of Experiment

Implement RSA Cryptosystem using RSA Algorithm.

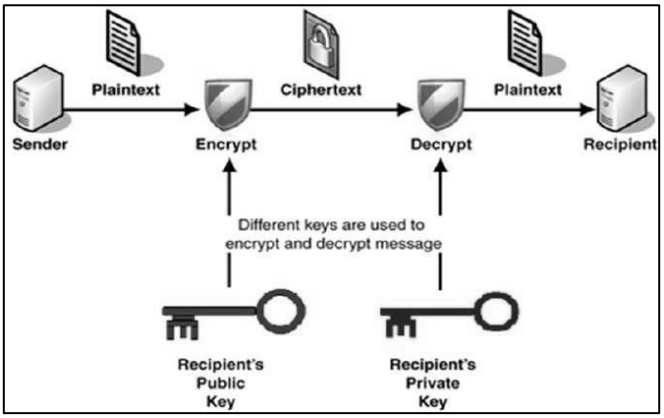
Theory / Algorithm / Conceptual Description

Encryption and Decryption



Encryption is the process of converting a readable message into an unreadable form so that it cannot be read by unauthorized parties. The process of converting an encrypted message back to its original (readable) format is known as decryption. The plaintext message is the original message. The ciphertext message is the encrypted message. Digital encryption techniques function by mathematically modifying the digital content of a plaintext message and producing a ciphertext version of the message using an encryption algorithm and a digital key. If the sender and recipient are the only ones who have access to the key, they can communicate safely.

Public Key Cryptography



Asymmetric cryptography, sometimes known as public-key cryptography, is a type of encryption that uses pairs of keys. A public key and a private key make up each pair. Cryptographic procedures based on mathematical problems known as one-way functions are used to generate such key pairs. The private key must be kept private for effective security; the public key can be freely distributed without jeopardizing security. In such a system, anyone can encrypt a message using the intended receiver's public key, but only the receiver's private key

can decrypt the message. This allows a server software to construct a cryptographic key for a suitable symmetric-key cryptography, then encrypt that freshly generated symmetric key using a clients openly provided public key. The server can then deliver this encrypted symmetric key to the client through an unsafe channel; only the client's private key can decrypt it. Because the client and server share the same symmetric key, they can safely communicate over otherwise insecure channels using symmetric key encryption.

RSA Algorithm

RSA is a frequently used public-key cryptosystem for secure data transmission. The initials "RSA" stand for Ron Rivest, Adi Shamir, and Leonard Adleman, who first published the algorithm in 1977. An RSA user generates and distributes a public key using two huge prime integers and an auxiliary value. The prime numbers are a closely guarded secret. Messages can be encrypted by anyone using the public key, but only those who know the prime numbers can decode them. The RSA algorithm is a slow one. As a result, it is rarely used to encrypt customer data directly. RSA is frequently used to transmit symmetric-key cryptography shared keys, which are then utilised for bulk encryption–decryption.

Algorithm

1. Choose two large prime numbers (p and q)
2. Calculate n = p*q and z = (p-1)(q-1)
3. Choose a number e where 1 < e < z such that e is coprime to z.
4. Calculate d such that e*d = 1 mod z or d is multiplicative inverse of e over mod z.
5. Private Key : d
6. Public key : (e, n)
7. If the plaintext is M, ciphertext = M^e mod n.
8. If the ciphertext is C, plaintext = C^d mod n.

Advantages

1. No Key Sharing: RSA encryption depends on using the receiver’s public key, so you don’t have to share any secret key to receive messages from others.
2. Proof of Authenticity: Since the key pairs are related to each other, a receiver can’t intercept the message since they won’t have the correct private key to decrypt the information.
3. Faster Encryption: The encryption process is faster than that of the DSA algorithm.

Disadvantages

1. It has slow data transfer rate due to large numbers involved.
2. It requires third party to verify the reliability of public keys sometimes.
3. High processing is required at receiver’s end for decryption.

Program

```
# Import Libraries
import numpy as np
import cv2
from google.colab.patches import cv2_imshow

# Defining the Prime Numbers
p = 19
q = 17
M = 255
n = p * q

# Calculate GCD of 2 numbers
def gcd(a,b):
  if b == 0:
    return a
  return gcd(b,a%b)

# Calculate Euler's Totient Function
def phi(p,q):
  return (p-1)*(q-1)
pn = phi(p,q)

# Calculating Public Key e
def public_key(pn):
  for k in range(2,pn):
    if gcd(pn,k) == 1:
      break
  return k
e = public_key(pn)

# Extended Euclidean Algorithm to find Multiplicative Inverse
def multiplicative_inverse(e,pn):
  x2,y2 = 1,0
  x1,y1 = 0,1
  r = -1
  while(r!=1):
    r = pn % e
    q = pn // e
    pn = e
    e = r
    x = (x2) - (q*x1)
    y = (y2) - (q*y1)
    x2,y2 = x1,y1
    x1,y1 = x,y
  return x,y
multiplicative_inverse(e,pn)
```

```
# Calculating Private Key d
def private_key(e,pn):
  x,y = multiplicative_inverse(e,pn)
  if y<0:
    e = y + pn
  else:
    e = y
  return e
d = private_key(e,pn)

# Encryption
def rsa_encrypt(M,e,n):
  return (pow(int(M),e))%n
C = rsa_encrypt(M,e,n)

# Decryption
def rsa_decrypt(C,d,n):
  return (pow(int(C),d))%n
P = rsa_decrypt(C,d,n)

# Image Encryption
image = cv2.imread('tree.jpg',0)
image = np.array(image,dtype=np.int64)
cv2_imshow(image)

# Encrypt
def image_encryption(image):
  encrypted_image = np.copy(image)
  for i,row in enumerate(image):
    for j in range(len(row)):
      encrypted_image[i][j] = rsa_encrypt(encrypted_image[i][j],e,n)
  return encrypted_image
encrypted = image_encryption(image)
cv2_imshow(encrypted)

# Decrypt
def image_decryption(encrypted_image):
  decrypted_image = np.copy(encrypted_image)
  for i,row in enumerate(encrypted_image):
    for j in range(len(row)):
      decrypted_image[i][j] = rsa_decrypt(decrypted_image[i][j],d,n)
  return decrypted_image
decrypted = image_decryption(encrypted)
cv2_imshow(decrypted)
```

Output

RSA Encryption and Decryption

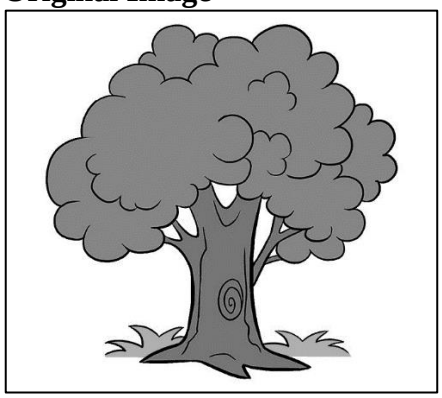
P = 19
Q = 17
M = 255
N = 323
Phi = 288
E = 5
D = 173

Encryption
221

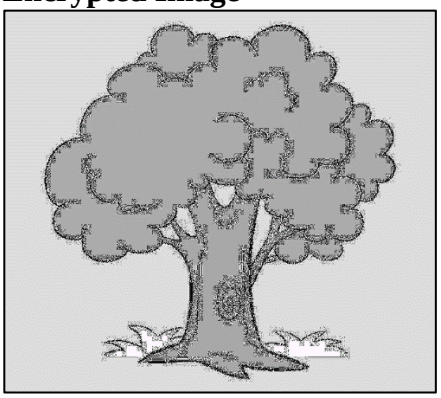
Decryption
255

Image Encryption using RSA

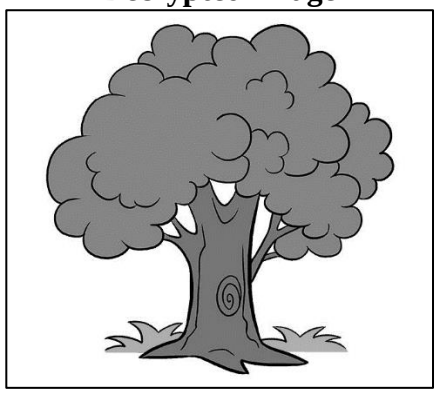
Original Image



Encrypted Image



Decrypted Image



Conclusion

RSA stands for Rivest, Shamir and Adleman, which is a public key cryptography algorithm. Public Key algorithm uses a pair of keys : public and private key. In RSA, these private and public keys are generated from two large prime numbers. The public key is then used for encrypting the data and the private key is used by the receiver for decrypting the data. RSA is a slower algorithm as it deals with large number computations, thus it is not used for sharing large data, generally it is used for sharing the symmetric key which can then be used for actual data encryption.

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Experiment 6

Date of Performance : 19-04-2022 **Date of Submission:** 20-04-2022

SAP Id: 60004190053 **Name :** Jayesh Kavedia

Div: A **Batch :** A4

Aim of Experiment

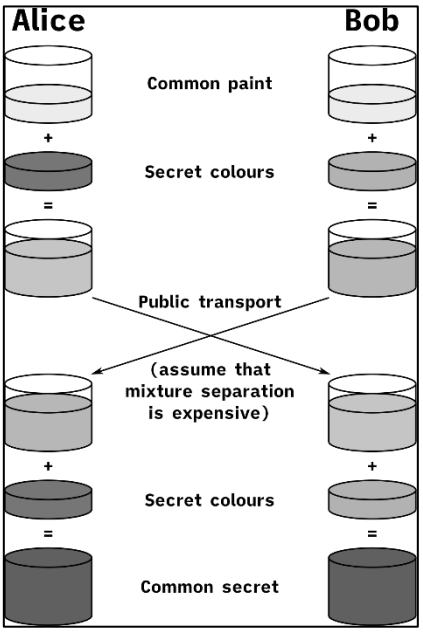
Implement Diffie Hellman Key Exchange Protocol.

Theory / Algorithm / Conceptual Description

Key Exchange Problem

Key exchange is a method in cryptography by which cryptographic keys are exchanged between two parties, allowing use of a cryptographic algorithm. If the cipher is a symmetric key cipher, both will need a copy of the same key. If it is an asymmetric key cipher with the public/private key property, both will need the other's public key. Key exchange is done either in-band or out-of-band.

Diffie Hellman Key Exchange Protocol



For the exchange of private information, symmetric encryption has long been a solid type of cryptography. The difficulty in communicating the required secret key with the message receiver has always been a major issue. It allows hackers to intercept any key sent via an unsecured channel and decipher encrypted ciphertexts using the same key. The Diffie Hellman method solves this problem by utilising one-way functions that only the sender and receiver can decrypt using a secret key. One-way functions use an algorithm that allows you to calculate an output for each input. However, deriving the appropriate input from a random output is logically impossible. The Diffie-Hellman algorithm is a means of securely transferring cryptographic keys via insecure channels without jeopardising data transmission security and integrity. Martin Hellman and Whitefield Diffie created and released it in 1976. The method starts with Alice and Bob openly agreeing on an arbitrary starting colour that does not need to be kept secret. In addition, each person chooses a secret hue that they will keep to themselves. The most important step in the process is for Alice and Bob to mix their own secret colour with their mutually shared colour, then publicly exchange the two combined colours. Finally, each of them combines the colour from their partner with their

own personal colour. The end outcome is a final colour mixing that is identical to that of their companion.

Algorithm

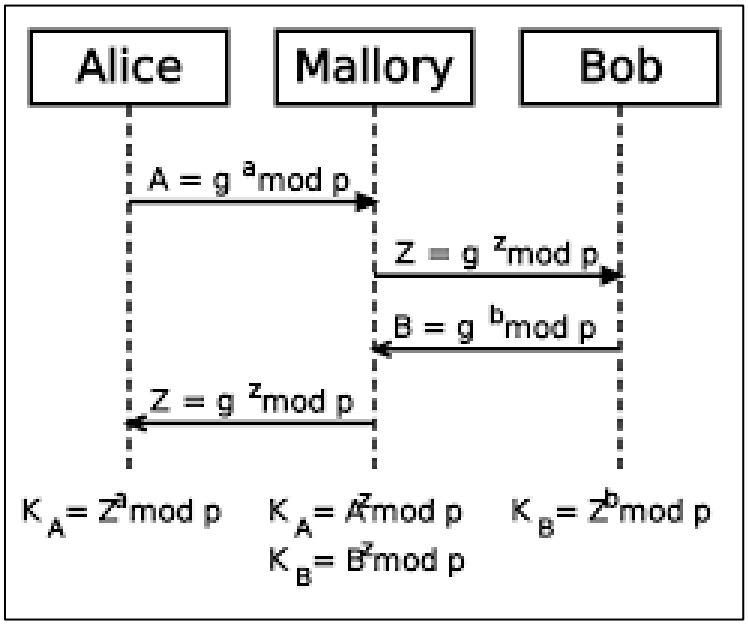
1. Choose a prime number q and select a primitive root of q as α.
2. Assume the private key for sender as Xa where Xa < q. The public key can be calculated as $Ya = \alpha^{xa} \bmod q$. So, the key pair for sender becomes {Xa, Ya}. Assume the private key for the receiver to be Xb where Xb < q. The public key for the receiver is calculated as $Yb = \alpha^{xb} \bmod q$. For the receiver, the key pair becomes {Xb, Yb}.
3. To generate the final secret key. The formula to calculate the key for sender is $K = (Yb)^{Xa} \bmod q$. The formula to calculate the secret key for receiver is $K = (Ya)^{Xb} \bmod q$. If both the values of K generated are equal, the Diffie-Hellman key exchange algorithm is complete.

Applications

1. Public Key Infrastructure
2. SSL/TLS Handshake
3. Secure Shell Access (SSH)

Disadvantages

1. Susceptible to Discrete Logarithmic Attack.
2. Susceptible to Man in the Middle Attack.



Man in the Middle Attack.

Program

```
# Importing Libraries
import numpy as np
from random import randint

# Selecting a Prime Number
q = 11
print("Selected Prime Number : ", q)

# Calculating Primitive Roots
def calculate_primitive_root(q):
  roots = []
  for i in range(2,q):
    flag = True
    unique_list = []
    for j in range(1,q):
      if pow(i,j) % q in unique_list:
        flag = False
        break
      else:
        unique_list.append(pow(i,j) % q)
    if flag == True:
      roots.append(i)
  return roots
calculate_primitive_root(q)

# Selecting a Primitive Root
alpha = 7
print(f"Primitive Root of {q} : {alpha}")

# Selecting Private Keys
xa = randint(1,q-1)
xb = randint(1,q-1)
print("Private Key Selected by Alice : ", xa)
print("Private Key Selected by Bob : ", xb)

# Computing Public Keys
ya = pow(alpha,xa) % q
yb = pow(alpha,xb) % q
print("Shared Public Key by Alice : ", ya)
print("Shared Public Key by Bob : ", yb)

# Computing Shared Secret Key
k1 = pow(yb,xa) % q
k2 = pow(ya,xb) % q
print("Key Computed by Alice : ", k1)
print("Key Computed by Bob : ", k2)
```

Output

```
Selected Prime Number :  11

Primitive Roots : [2, 6, 7, 8]

Primitive Root of 11 : 7

Private Key Selected by Alice :  3
Private Key Selected by Bob :  9

Shared Public Key by Alice :  2
Shared Public Key by Bob :  8

Key Computed by Alice :  6
Key Computed by Bob :  6
```

Conclusion

Key Exchange Protocols define a way by which a cryptographic key is exchanged between 2 parties over an insecure channel. Diffie Hellman is a Key Exchange Protocol which uses concept of prime number and primitive root to exchange key between 2 parties. It is used generally for Symmetric Key Exchanging. It was best protocol for Symmetric exchange however it is susceptible to Discrete Logarithmic attack as well as Man in the Middle attack.

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Experiment 7

Date of Performance : 26-04-2022

Date of Submission: 27-04-2022

SAP Id: 60004190053

Name : Jayesh Kavedia

Div: A

Batch : A4

Aim of Experiment

Study the use of network reconnaissance tools like WHOIS, dig, traceroute, nslookup to gather information about networks and domain registrars.

Theory

WHOIS

WHOIS (pronounced as the phrase "who is") is a query and response protocol that is widely used for querying databases that store the registered users or assignees of an Internet resource, such as a domain name, an IP address block or an autonomous system, but is also used for a wider range of other information. The protocol stores and delivers database content in a human-readable format. To get the information about the registered domain, simply issue the whois command with the domain name. It will retrieve domain data including availability, ownership, creation, expiration details, name servers, etc.

Output

```
C:\Users\DELL\Downloads\WhoIs>whois amazon.com

Whois v1.21 - Domain information lookup
Copyright (C) 2005-2019 Mark Russinovich
Sysinternals - www.sysinternals.com

Connecting to COM.whois-servers.net...

WHOIS Server: whois.markmonitor.com
   Registrar URL: http://www.markmonitor.com
   Updated Date: 2019-05-07T20:09:37Z
   Creation Date: 1994-11-01T05:00:00Z
   Registry Expiry Date: 2024-10-31T04:00:00Z
   Registrar: MarkMonitor Inc.
   Registrar IANA ID: 292
   Registrar Abuse Contact Email: abusecomplaints@markmonitor.com
   Registrar Abuse Contact Phone: +1.2086851750
   Domain Status: clientDeleteProhibited https://icann.org/epp#clientDeleteProhibited
   Domain Status: clientTransferProhibited https://icann.org/epp#clientTransferProhibited
   Domain Status: clientUpdateProhibited https://icann.org/epp#clientUpdateProhibited
   Domain Status: serverDeleteProhibited https://icann.org/epp#serverDeleteProhibited
   Domain Status: serverTransferProhibited https://icann.org/epp#serverTransferProhibited
   Domain Status: serverUpdateProhibited https://icann.org/epp#serverUpdateProhibited
   Name Server: NS1.P31.DYNECT.NET
   Name Server: NS2.P31.DYNECT.NET
   Name Server: NS3.P31.DYNECT.NET
   Name Server: NS4.P31.DYNECT.NET
   Name Server: PDNS1.ULTRADNS.NET
   Name Server: PDNS6.ULTRADNS.CO.UK
   DNSSEC: unsigned
   URL of the ICANN Whois Inaccuracy Complaint Form: https://www.icann.org/wicf/
>>> Last update of whois database: 2022-06-02T15:38:19Z <<<

For more information on Whois status codes, please visit https://icann.org/epp

NOTICE: The expiration date displayed in this record is the date the
registrar's sponsorship of the domain name registration in the registry is
currently set to expire. This date does not necessarily reflect the expiration
date of the domain name registrant's agreement with the sponsoring
registrar.  Users may consult the sponsoring registrar's Whois database to
view the registrar's reported date of expiration for this registration.

TERMS OF USE: You are not authorized to access or query our Whois
database through the use of electronic processes that are high-volume and
automated except as reasonably necessary to register domain names or
modify existing registrations; the Data in VeriSign Global Registry
Services' ("VeriSign") Whois database is provided by VeriSign for
information purposes only, and to assist persons in obtaining information
about or related to a domain name registration record. VeriSign does not
```

```
guarantee its accuracy. By submitting a Whois query, you agree to abide
by the following terms of use: You agree that you may use this Data only
for lawful purposes and that under no circumstances will you use this Data
to: (1) allow, enable, or otherwise support the transmission of mass
unsolicited, commercial advertising or solicitations via e-mail, telephone,
or facsimile; or (2) enable high volume, automated, electronic processes
that apply to VeriSign (or its computer systems). The compilation,
repackaging, dissemination or other use of this Data is expressly
prohibited without the prior written consent of VeriSign. You agree not to
use electronic processes that are automated and high-volume to access or
query the Whois database except as reasonably necessary to register
domain names or modify existing registrations. VeriSign reserves the right
to restrict your access to the Whois database in its sole discretion to ensure
operational stability.  VeriSign may restrict or terminate your access to the
Whois database for failure to abide by these terms of use. VeriSign
reserves the right to modify these terms at any time.

The Registry database contains
Connecting to whois.markmonitor.com...

WHOIS Server: whois.markmonitor.com
Registrar URL: http://www.markmonitor.com
Updated Date: 2019-08-26T19:19:56+0000
Creation Date: 1994-11-01T05:00:00+0000
Registrar Registration Expiration Date: 2024-10-30T07:00:00+0000
Registrar: MarkMonitor, Inc.
Registrar IANA ID: 292
Registrar Abuse Contact Email: abusecomplaints@markmonitor.com
Registrar Abuse Contact Phone: +1.2083895770
Domain Status: clientUpdateProhibited (https://www.icann.org/epp#clientUpdateProhibited)
Domain Status: clientTransferProhibited (https://www.icann.org/epp#clientTransferProhibited)
Domain Status: clientDeleteProhibited (https://www.icann.org/epp#clientDeleteProhibited)
Domain Status: serverUpdateProhibited (https://www.icann.org/epp#serverUpdateProhibited)
Domain Status: serverTransferProhibited (https://www.icann.org/epp#serverTransferProhibited)
Domain Status: serverDeleteProhibited (https://www.icann.org/epp#serverDeleteProhibited)
Registry Registrant ID:
Registrant Name: Hostmaster, Amazon Legal Dept.
Registrant Organization: Amazon Technologies, Inc.
Registrant Street: P.O. Box 8102
Registrant City: Reno
Registrant State/Province: NV
Registrant Postal Code: 89507
Registrant Country: US
Registrant Phone: +1.2062664064
Registrant Phone Ext:
Registrant Fax: +1.2062667010
Registrant Fax Ext:
Registrant Email: hostmaster@amazon.com
Registry Admin ID:
```

```
Admin Name: Hostmaster, Amazon Legal Dept.
Admin Organization: Amazon Technologies, Inc.
Admin Street: P.O. Box 8102
Admin City: Reno
Admin State/Province: NV
Admin Postal Code: 89507
Admin Country: US
Admin Phone: +1.2062664064
Admin Phone Ext:
Admin Fax: +1.2062667010
Admin Fax Ext:
Admin Email: hostmaster@amazon.com
Registry Tech ID:
Tech Name: Hostmaster, Amazon Legal Dept.
Tech Organization: Amazon Technologies, Inc.
Tech Street: P.O. Box 8102
Tech City: Reno
Tech State/Province: NV
Tech Postal Code: 89507
Tech Country: US
Tech Phone: +1.2062664064
Tech Phone Ext:
Tech Fax: +1.2062667010
Tech Fax Ext:
Tech Email: hostmaster@amazon.com
Name Server: pdns6.ultradns.co.uk
Name Server: pdns1.ultradns.net
Name Server: ns4.p31.dynect.net
Name Server: ns1.p31.dynect.net
Name Server: ns3.p31.dynect.net
Name Server: ns2.p31.dynect.net
DNSSEC: unsigned
URL of the ICANN WHOIS Data Problem Reporting System: http://wdprs.internic.net/
>>> Last update of WHOIS database: 2022-06-02T15:39:45+0000 <<<

For more information on WHOIS status codes, please visit:
  https://www.icann.org/resources/pages/epp-status-codes

If you wish to contact this domainΓÇÖs Registrant, Administrative, or Technical
contact, and such email address is not visible above, you may do so via our web
form, pursuant to ICANNΓÇÖs Temporary Specification. To verify that you are not a
robot, please enter your email address to receive a link to a page that
facilitates email communication with the relevant contact(s).

Web-based WHOIS:
  https://domains.markmonitor.com/whois

If you have a legitimate interest in viewing the non-public WHOIS details, send
your request and the reasons for your request to whoisrequest@markmonitor.com
```

```
and specify the domain name in the subject line. We will review that request and
may ask for supporting documentation and explanation.

The data in MarkMonitorΓÇÖs WHOIS database is provided for information purposes,
and to assist persons in obtaining information about or related to a domain
nameΓÇÖs registration record. While MarkMonitor believes the data to be accurate,
the data is provided "as is" with no guarantee or warranties regarding its
accuracy.

By submitting a WHOIS query, you agree that you will use this data only for
lawful purposes and that, under no circumstances will you use this data to:
  (1) allow, enable, or otherwise support the transmission by email, telephone,
or facsimile of mass, unsolicited, commercial advertising, or spam; or
  (2) enable high volume, automated, or electronic processes that send queries,
data, or email to MarkMonitor (or its systems) or the domain name contacts (or
its systems).

MarkMonitor reserves the right to modify these terms at any time.

By submitting this query, you agree to abide by this policy.

MarkMonitor Domain Management(TM)
Protecting companies and consumers in a digital world.

Visit MarkMonitor at https://www.markmonitor.com
Contact us at +1.8007459229
In Europe, at +44.02032062220
----


Domain Name: amazon.com
Registry Domain ID: 281209_DOMAIN_COM-VRSN
Registrar WHOIS Server: whois.markmonitor.com
Registrar URL: http://www.markmonitor.com
Updated Date: 2019-08-26T19:19:56+0000
Creation Date: 1994-11-01T05:00:00+0000
Registrar Registration Expiration Date: 2024-10-30T07:00:00+0000
Registrar: MarkMonitor, Inc.
Registrar IANA ID: 292
Registrar Abuse Contact Email: abusecomplaints@markmonitor.com
Registrar Abuse Contact Phone: +1.2083895770
Domain Status: clientUpdateProhibited (https://www.icann.org/epp#clientUpdateProhibited)
Domain Status: clientTransferProhibited (https://www.icann.org/epp#clientTransferProhibited)
Domain Status: clientDeleteProhibited (https://www.icann.org/epp#clientDeleteProhibited)
Domain Status: serverUpdateProhibited (https://www.icann.org/epp#serverUpdateProhibited)
Domain Status: serverTransferProhibited (https://www.icann.org/epp#serverTransferProhibited)
Domain Status: serverDeleteProhibited (https://www.icann.org/epp#serverDeleteProhibited)
Registry Registrant ID:
Registrant Name: Hostmaster, Amazon Legal Dept.
```

```
Registrant Organization: Amazon Technologies, Inc.
Registrant Street: P.O. Box 8102
Registrant City: Reno
Registrant State/Province: NV
Registrant Postal Code: 89507
Registrant Country: US
Registrant Phone: +1.2062664064
Registrant Phone Ext:
Registrant Fax: +1.2062667010
Registrant Fax Ext:
Registrant Email: hostmaster@amazon.com
Registry Admin ID:
Admin Name: Hostmaster, Amazon Legal Dept.
Admin Organization: Amazon Technologies, Inc.
Admin Street: P.O. Box 8102
Admin City: Reno
Admin State/Province: NV
Admin Postal Code: 89507
Admin Country: US
Admin Phone: +1.2062664064
Admin Phone Ext:
Admin Fax: +1.2062667010
Admin Fax Ext:
Admin Email: hostmaster@amazon.com
Registry Tech ID:
Tech Name: Hostmaster, Amazon Legal Dept.
Tech Organization: Amazon Technologies, Inc.
Tech Street: P.O. Box 8102
Tech City: Reno
Tech State/Province: NV
Tech Postal Code: 89507
Tech Country: US
Tech Phone: +1.2062664064
Tech Phone Ext:
Tech Fax: +1.2062667010
Tech Fax Ext:
Tech Email: hostmaster@amazon.com
Name Server: pdns6.ultradns.co.uk
Name Server: pdns1.ultradns.net
Name Server: ns4.p31.dynect.net
Name Server: ns1.p31.dynect.net
Name Server: ns3.p31.dynect.net
Name Server: ns2.p31.dynect.net
DNSSEC: unsigned
URL of the ICANN WHOIS Data Problem Reporting System: http://wdprs.internic.net/
>>> Last update of WHOIS database: 2022-06-02T15:39:45+0000 <<<

For more information on WHOIS status codes, please visit:
  https://www.icann.org/resources/pages/epp-status-codes
```

```
  https://www.icann.org/resources/pages/epp-status-codes

If you wish to contact this domainΓÇÖs Registrant, Administrative, or Technical
contact, and such email address is not visible above, you may do so via our web
form, pursuant to ICANNΓÇÖs Temporary Specification. To verify that you are not a
robot, please enter your email address to receive a link to a page that
facilitates email communication with the relevant contact(s).

Web-based WHOIS:
  https://domains.markmonitor.com/whois

If you have a legitimate interest in viewing the non-public WHOIS details, send
your request and the reasons for your request to whoisrequest@markmonitor.com
and specify the domain name in the subject line. We will review that request and
may ask for supporting documentation and explanation.

The data in MarkMonitorΓÇÖs WHOIS database is provided for information purposes,
and to assist persons in obtaining information about or related to a domain
nameΓÇÖs registration record. While MarkMonitor believes the data to be accurate,
the data is provided "as is" with no guarantee or warranties regarding its
accuracy.

By submitting a WHOIS query, you agree that you will use this data only for
lawful purposes and that, under no circumstances will you use this data to:
  (1) allow, enable, or otherwise support the transmission by email, telephone,
or facsimile of mass, unsolicited, commercial advertising, or spam; or
  (2) enable high volume, automated, or electronic processes that send queries,
data, or email to MarkMonitor (or its systems) or the domain name contacts (or
its systems).

MarkMonitor reserves the right to modify these terms at any time.

By submitting this query, you agree to abide by this policy.

MarkMonitor Domain Management(TM)
Protecting companies and consumers in a digital world.

Visit MarkMonitor at https://www.markmonitor.com
Contact us at +1.8007459229
In Europe, at +44.02032062220
----


C:\Users\DELL\Downloads\WhoIs>
```

DIG

DIG is a network administration command-line tool for querying the Domain Name System (DNS). DIG is useful for network troubleshooting and for educational purposes. It can operate based on command line option and flag arguments, or in batch mode by reading requests from an operating system file. When a specific name server is not specified in the command

invocation, it uses the operating system's default resolver, usually configured in the file resolv.conf. Without any arguments it queries the DNS root zone.

Output

```
C:\Users\DELL>dig amazon.com

; <<>> DiG 9.16.29 <<>> amazon.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6915
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 6, ADDITIONAL: 13

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;amazon.com.                    IN      A

;; ANSWER SECTION:
amazon.com.             48      IN      A       176.32.103.205
amazon.com.             48      IN      A       54.239.28.85
amazon.com.             48      IN      A       205.251.242.103

;; AUTHORITY SECTION:
amazon.com.             588     IN      NS      ns3.p31.dynect.net.
amazon.com.             588     IN      NS      pdns6.ultradns.co.uk.
amazon.com.             588     IN      NS      ns2.p31.dynect.net.
amazon.com.             588     IN      NS      ns4.p31.dynect.net.
amazon.com.             588     IN      NS      ns1.p31.dynect.net.
amazon.com.             588     IN      NS      pdns1.ultradns.net.

;; ADDITIONAL SECTION:
ns1.p31.dynect.net.     31925   IN      A       108.59.161.31
ns1.p31.dynect.net.     31925   IN      AAAA    2600:2000:2210::31
ns3.p31.dynect.net.     181     IN      A       108.59.163.31
ns3.p31.dynect.net.     181     IN      AAAA    2600:2000:2230::31
ns2.p31.dynect.net.     31925   IN      A       108.59.162.31
ns2.p31.dynect.net.     31925   IN      AAAA    2600:2000:2220::31
ns4.p31.dynect.net.     181     IN      A       108.59.164.31
ns4.p31.dynect.net.     181     IN      AAAA    2600:2000:2240::31
pdns1.ultradns.net.     3600    IN      A       204.74.108.1
pdns1.ultradns.net.     3600    IN      AAAA    2001:502:f3ff::1
pdns6.ultradns.co.uk.   1803    IN      A       204.74.115.1
pdns6.ultradns.co.uk.   1803    IN      AAAA    2610:a1:1017::1

;; Query time: 6 msec
;; SERVER: 192.168.1.1#53(192.168.1.1)
;; WHEN: Thu Jun 02 21:36:23 India Standard Time 2022
;; MSG SIZE  rcvd: 500


C:\Users\DELL>
```

```
C:\Users\DELL>dig

; <<>> DiG 9.16.29 <<>>
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 52564
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;.                              IN      NS

;; ANSWER SECTION:
.                       86621   IN      NS      m.root-servers.net.
.                       86621   IN      NS      b.root-servers.net.
.                       86621   IN      NS      c.root-servers.net.
.                       86621   IN      NS      d.root-servers.net.
.                       86621   IN      NS      e.root-servers.net.
.                       86621   IN      NS      f.root-servers.net.
.                       86621   IN      NS      g.root-servers.net.
.                       86621   IN      NS      h.root-servers.net.
.                       86621   IN      NS      a.root-servers.net.
.                       86621   IN      NS      i.root-servers.net.
.                       86621   IN      NS      j.root-servers.net.
.                       86621   IN      NS      k.root-servers.net.
.                       86621   IN      NS      l.root-servers.net.

;; Query time: 9 msec
;; SERVER: 192.168.1.1#53(192.168.1.1)
;; WHEN: Thu Jun 02 21:36:36 India Standard Time 2022
;; MSG SIZE  rcvd: 239


C:\Users\DELL>
```

```
C:\Users\DELL>dig 176.32.103.205

; <<>> DiG 9.16.29 <<>> 176.32.103.205
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 57395
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;176.32.103.205.                        IN      A

;; AUTHORITY SECTION:
.                       86394   IN      SOA     a.root-servers.net. nstld.verisign-grs.com. 2022060200 1800 900 604800 86400

;; Query time: 0 msec
;; SERVER: 192.168.1.1#53(192.168.1.1)
;; WHEN: Thu Jun 02 21:37:17 India Standard Time 2022
;; MSG SIZE  rcvd: 118


C:\Users\DELL>
```

NSLOOKUP

nslookup (from name server lookup) is a network administration command-line tool for querying the Domain Name System (DNS) to obtain the mapping between domain name and IP address, or other DNS records. nslookup operates in interactive or non-interactive mode. When used interactively by invoking it without arguments or when the first argument is - (minus sign) and the second argument is a hostname or Internet address of a name server, the user issues parameter configurations or requests when presented with the nslookup prompt (>). When no arguments are given, then the command queries the default server. The - (minus sign) invokes subcommands which are specified on the command line and should precede nslookup commands. In non-interactive mode, i.e. when the first argument is a name or Internet address of the host being searched, parameters and the query are specified as command line arguments in the invocation of the program. The non interactive mode searches the information for a specified host using the default name server.

Output

Nslookup followed by the domain name will display the "A Record" (IP Address) of the domain.

```
C:\Users\DELL\Downloads>nslookup amazon.com
Server:  UnKnown
Address:  192.168.1.1

Non-authoritative answer:
Name:    amazon.com.iballbatonwifi.com
Addresses:  45.79.19.196
          45.56.79.23
          96.126.123.244
          72.14.185.43
          45.33.18.44
          173.255.194.134
          45.33.30.197
          72.14.178.174
          198.58.118.167
          45.33.23.183
          45.33.2.79
          45.33.20.235


C:\Users\DELL\Downloads>
```

SOA record (start of authority), provides the authoritative information about the domain, the e-mail address of the domain admin, the domain serial number, etc.

```
C:\Users\DELL\Downloads>nslookup -type=soa amazon.com
Server:  UnKnown
Address:  192.168.1.1

Non-authoritative answer:
amazon.com.iballbatonwifi.com
        primary name server = ns1.securetrafficrouting.com
        responsible mail addr = admin.mytrafficmanagement.com
        serial  = 2013020401
        refresh = 10800 (3 hours)
        retry   = 3600 (1 hour)
        expire  = 604800 (7 days)
        default TTL = 3600 (1 hour)

iballbatonwifi.com      nameserver = ns2.securetrafficrouting.com
iballbatonwifi.com      nameserver = ns1.securetrafficrouting.com
ns1.securetrafficrouting.com    internet address = 192.53.171.69
ns1.securetrafficrouting.com    internet address = 194.195.216.90
ns1.securetrafficrouting.com    internet address = 194.195.219.162
ns1.securetrafficrouting.com    internet address = 45.79.46.24
ns1.securetrafficrouting.com    internet address = 96.126.118.145
ns1.securetrafficrouting.com    internet address = 172.104.196.157
ns1.securetrafficrouting.com    internet address = 192.46.217.211
ns1.securetrafficrouting.com    internet address = 192.46.218.200
ns2.securetrafficrouting.com    internet address = 194.195.219.162
ns2.securetrafficrouting.com    internet address = 45.79.46.24
ns2.securetrafficrouting.com    internet address = 96.126.118.145
ns2.securetrafficrouting.com    internet address = 172.104.196.157
ns2.securetrafficrouting.com    internet address = 192.46.217.211
ns2.securetrafficrouting.com    internet address = 192.46.218.200
ns2.securetrafficrouting.com    internet address = 192.53.171.69
ns2.securetrafficrouting.com    internet address = 194.195.216.90

C:\Users\DELL\Downloads>
```

NS (Name Server) record maps a domain name to a list of DNS servers authoritative for that domain. It will output the name servers which are associated with the given domain.

```
C:\Users\DELL\Downloads>nslookup -type=ns amazon.com
Server:  UnKnown
Address:  192.168.1.1

Non-authoritative answer:
amazon.com.iballbatonwifi.com   nameserver = ns2.securetrafficrouting.com
amazon.com.iballbatonwifi.com   nameserver = ns1.securetrafficrouting.com

ns1.securetrafficrouting.com    internet address = 192.46.218.200
ns1.securetrafficrouting.com    internet address = 192.53.171.69
ns1.securetrafficrouting.com    internet address = 194.195.216.90
ns1.securetrafficrouting.com    internet address = 194.195.219.162
ns1.securetrafficrouting.com    internet address = 45.79.46.24
ns1.securetrafficrouting.com    internet address = 96.126.118.145
ns1.securetrafficrouting.com    internet address = 172.104.196.157
ns1.securetrafficrouting.com    internet address = 192.46.217.211
ns2.securetrafficrouting.com    internet address = 192.53.171.69
ns2.securetrafficrouting.com    internet address = 194.195.216.90
ns2.securetrafficrouting.com    internet address = 194.195.219.162
ns2.securetrafficrouting.com    internet address = 45.79.46.24
ns2.securetrafficrouting.com    internet address = 96.126.118.145
ns2.securetrafficrouting.com    internet address = 172.104.196.157
ns2.securetrafficrouting.com    internet address = 192.46.217.211
ns2.securetrafficrouting.com    internet address = 192.46.218.200

C:\Users\DELL\Downloads>
```

TRACEROUTE

In computing, traceroute and tracert are computer network diagnostic commands for displaying possible routes (paths) and measuring transit delays of packets across an Internet Protocol (IP) network. The history of the route is recorded as the round-trip times of the packets received from each successive host (remote node) in the route (path); the sum of the mean times in each hop is a measure of the total time spent to establish the connection. Traceroute proceeds unless all (usually three) sent packets are lost more than twice; then the connection is lost and the route cannot be evaluated.

```
C:\Users\DELL\Downloads\WhoIs>tracert amazon.com

Tracing route to amazon.com [176.32.103.205]
over a maximum of 30 hops:

  1     5 ms     2 ms     2 ms  192.168.1.1
  2     5 ms     3 ms     3 ms  24-5-224-103.vasaicable.co.in [103.224.5.24]
  3     *        *        4 ms  17-5-224-103.vasaicable.co.in [103.224.5.17]
  4     5 ms     5 ms     5 ms  static-253.92.248.49-tataidc.co.in [49.248.92.253]
  5    46 ms     6 ms     6 ms  10.0.10.209
  6    30 ms     6 ms     7 ms  10.124.253.101
  7     *        *        *     Request timed out.
  8     6 ms     5 ms     5 ms  115.113.165.21.static-mumbai.vsnl.net.in [115.113.165.21]
  9     6 ms     6 ms     5 ms  172.23.78.237
 10     6 ms     6 ms     6 ms  ix-ae-0-100.tcore1.mlv-mumbai.as6453.net [180.87.38.5]
 11   236 ms   200 ms   221 ms  if-ae-2-2.tcore2.mlv-mumbai.as6453.net [180.87.38.2]
 12     *        *        *     Request timed out.
 13   216 ms   209 ms     *     if-ae-66-7.tcore3.nto-newyork.as6453.net [80.231.130.23]
 14   201 ms     *      219 ms  if-be-2-2.ecore1.n75-newyork.as6453.net [66.110.96.62]
 15   210 ms   202 ms   202 ms  if-ae-57-2.tcore1.n75-newyork.as6453.net [66.110.96.58]
 16     *      205 ms   211 ms  66.110.96.157
 17     *        *        *     Request timed out.
 18     *        *        *     Request timed out.
 19     *        *        *     Request timed out.
 20     *        *        *     Request timed out.
 21     *        *        *     Request timed out.
 22     *        *        *     Request timed out.
 23     *        *        *     Request timed out.
 24     *        *        *     Request timed out.
 25     *        *        *     Request timed out.
 26     *        *        *     Request timed out.
 27     *        *        *     Request timed out.
 28     *        *        *     Request timed out.
 29     *        *        *     Request timed out.
 30     *        *        *     Request timed out.

Trace complete.

C:\Users\DELL\Downloads\WhoIs>
```

Conclusion

WHOIS is used to get domain name, IP address and various details about a registered domain. DIG is used for querying DNS servers. NSLOOKUP is used to obtain mapping between IP Address and Domain Name. TRACEROUTE is used to find possible paths and transit delays in an IP network. Thus we have successfully studied the use of network reconnaissance tools like WHOIS, dig, traceroute, nslookup to gather information about networks and domain registrars.

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Experiment 8

Date of Performance : 10-05-2022 **Date of Submission:** 11-05-2022

SAP Id: 60004190053 **Name :** Jayesh Kavedia

Div: A **Batch :** A4

Aim of Experiment

Study of packet sniffer tools Wireshark, : Download and install Wireshark and capture ICMP, TCP, and HTTP packets in promiscuous mode. Explore how the packets can be traced based on different filters.

Theory / Algorithm / Conceptual Description

Wireshark is a free and open-source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education. Wireshark lets the user put network interface controllers into promiscuous mode (if supported by the network interface controller), so they can see all the traffic visible on that interface including unicast traffic not sent to that network interface controller's MAC address. Port mirroring or various network taps extend capture to any point on the network.

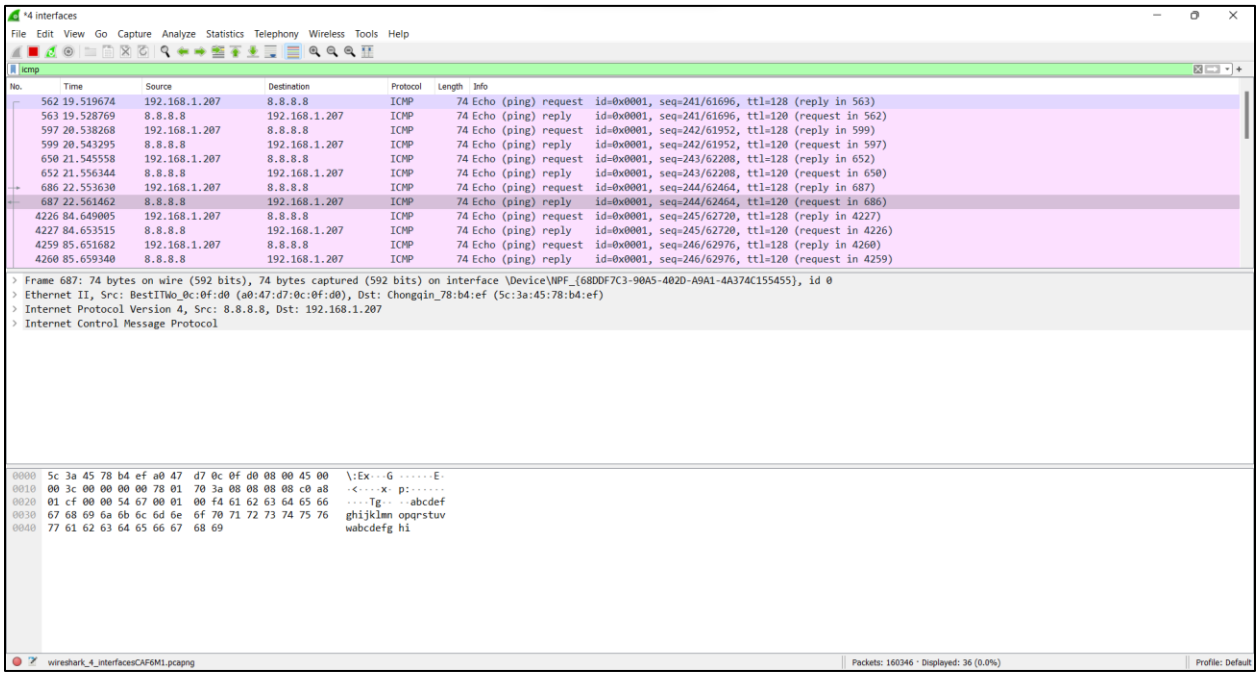
Features :

1. Data can be captured "from the wire" from a live network connection or read from a file of already-captured packets.
2. Live data can be read from different types of networks, including Ethernet, IEEE 802.11, PPP, and loopback.
3. Captured network data can be browsed via a GUI, or via the terminal (command line) version of the utility, TShark.
4. Captured files can be programmatically edited or converted via command-line switches to the "editcap" program.
5. Data display can be refined using a display filter.
6. Plug-ins can be created for dissecting new protocols.
7. VoIP calls in the captured traffic can be detected. If encoded in a compatible encoding, the media flow can even be played.
8. Raw USB traffic can be captured.
9. Wireless connections can also be filtered as long as they traverse the monitored Ethernet.
10. Various settings, timers, and filters can be set to provide the facility of filtering the output of the captured traffic.

Output

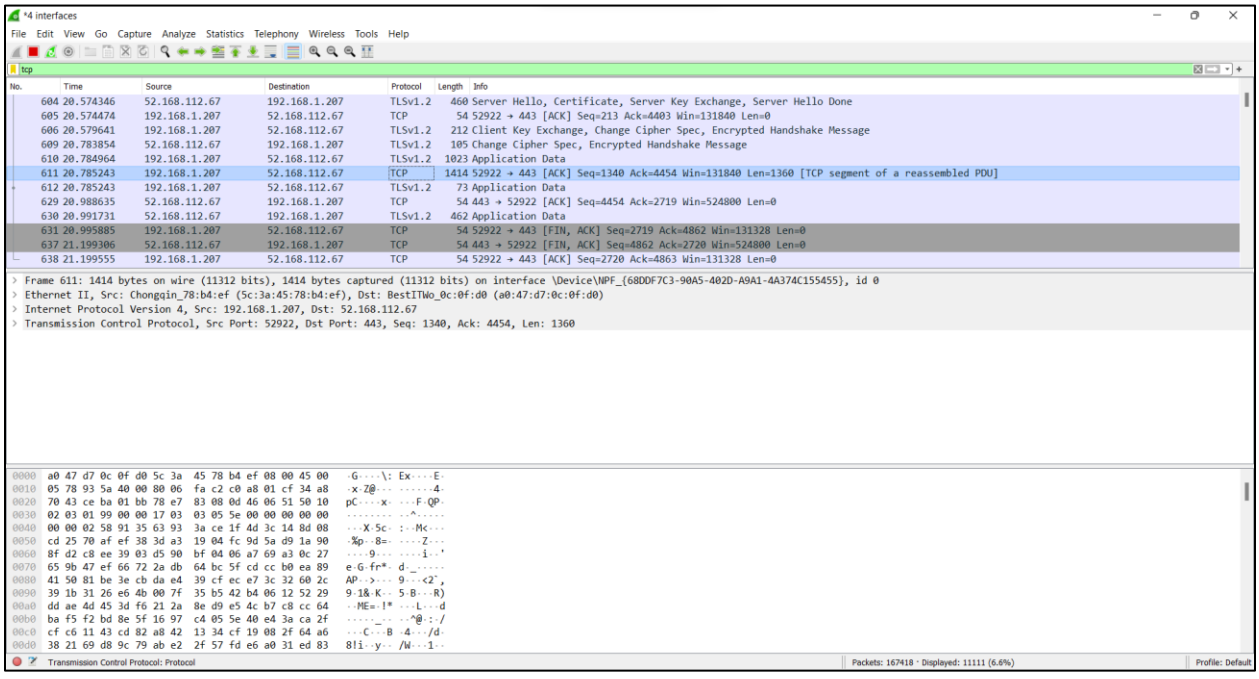
ICMP Packets

The Internet Control Message Protocol (ICMP) is used by network devices, including routers, to send error messages and operational information indicating success or failure when communicating with another IP address.



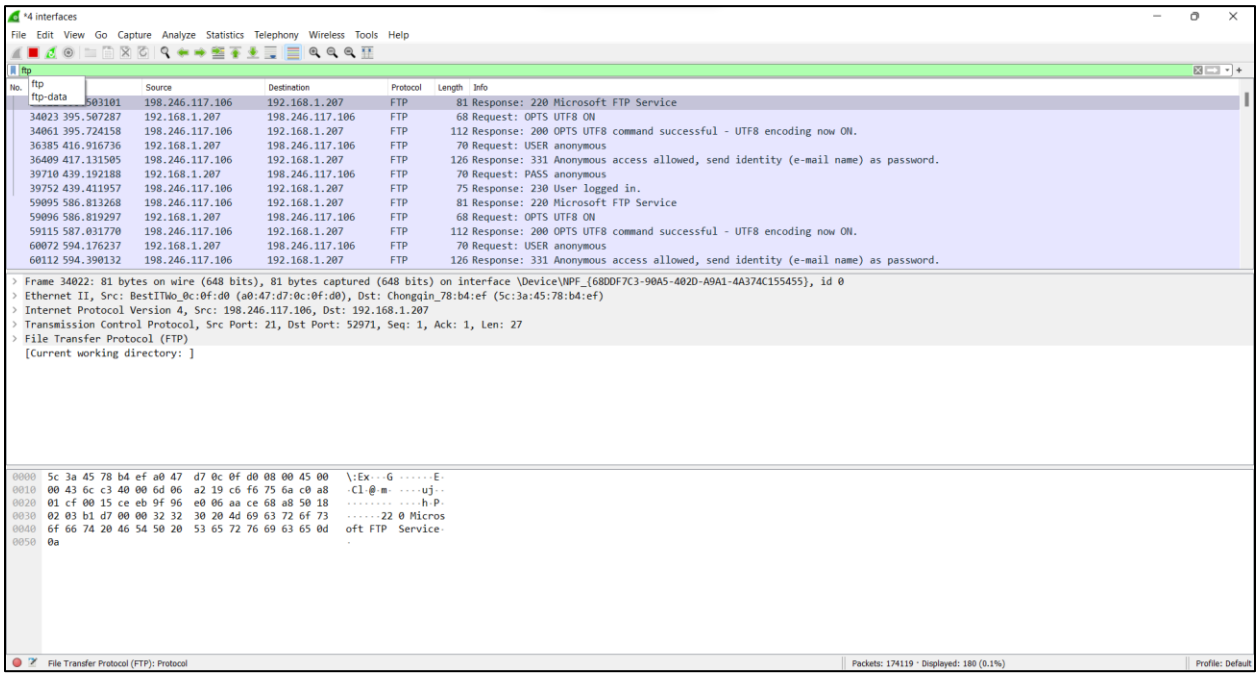
TCP Packets

The transmission control protocol (TCP) is the internet standard ensuring the successful exchange of data packets between devices over a network.



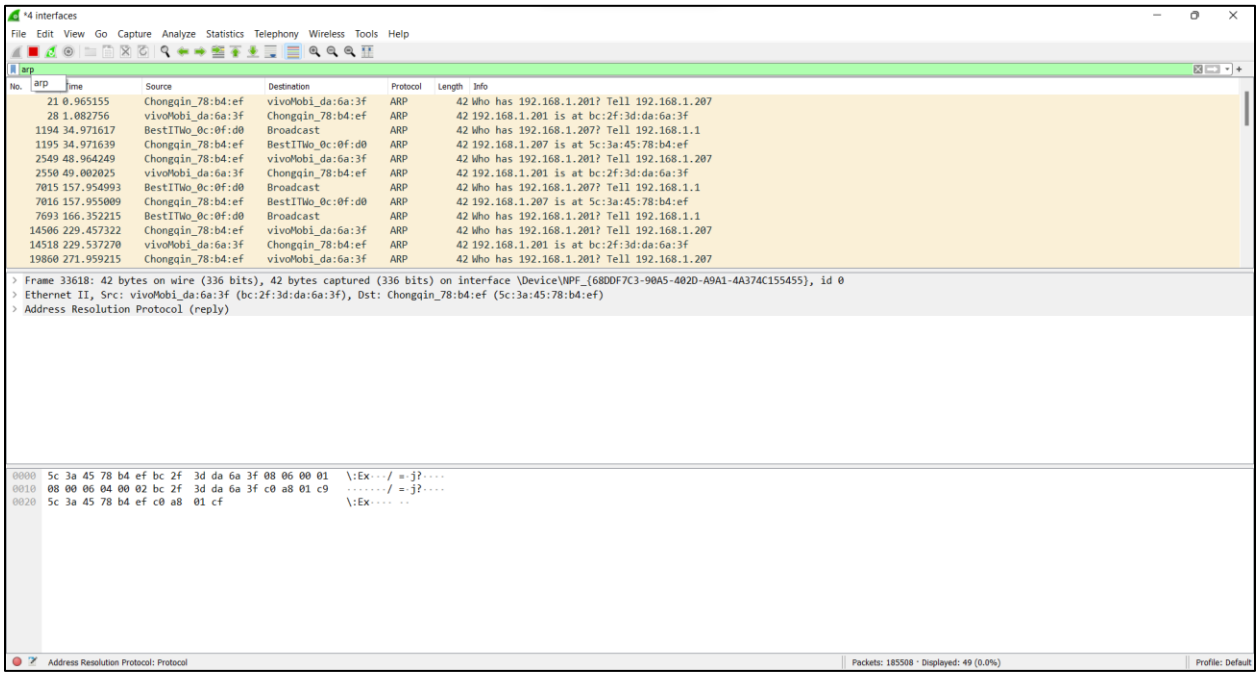
FTP Packets

The File Transfer Protocol (FTP) is a standard communication protocol used for the transfer of computer files from a server to a client on a computer network.

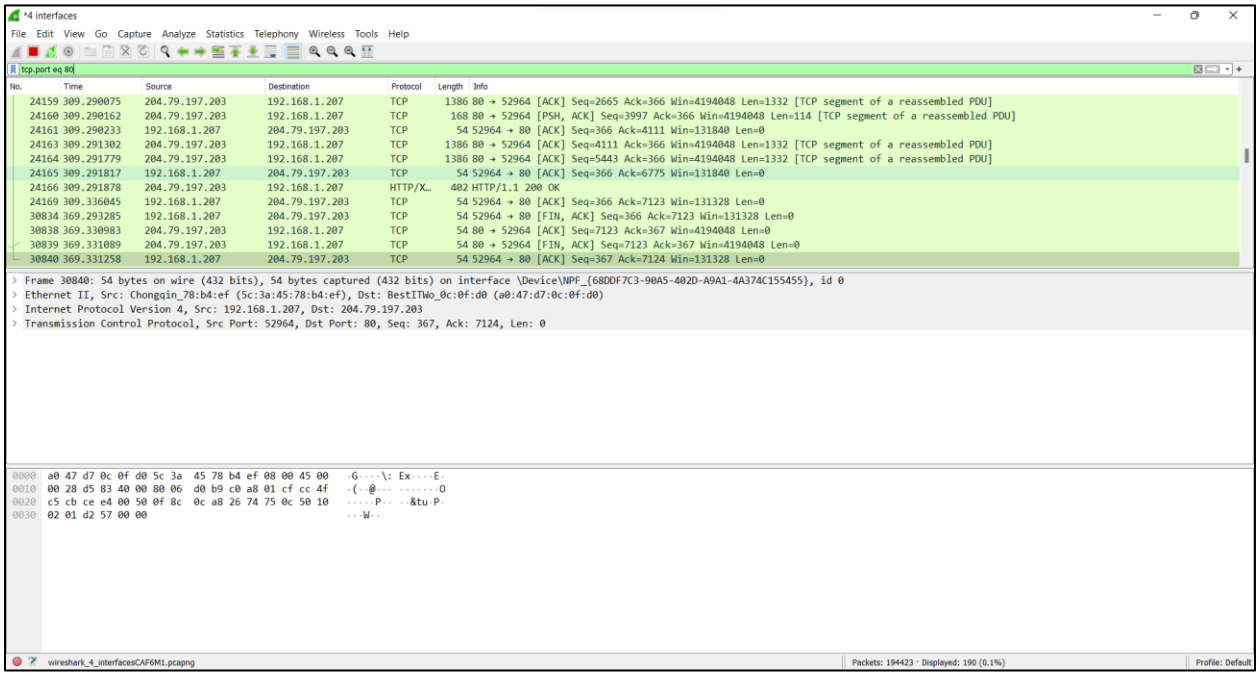


ARP Packets

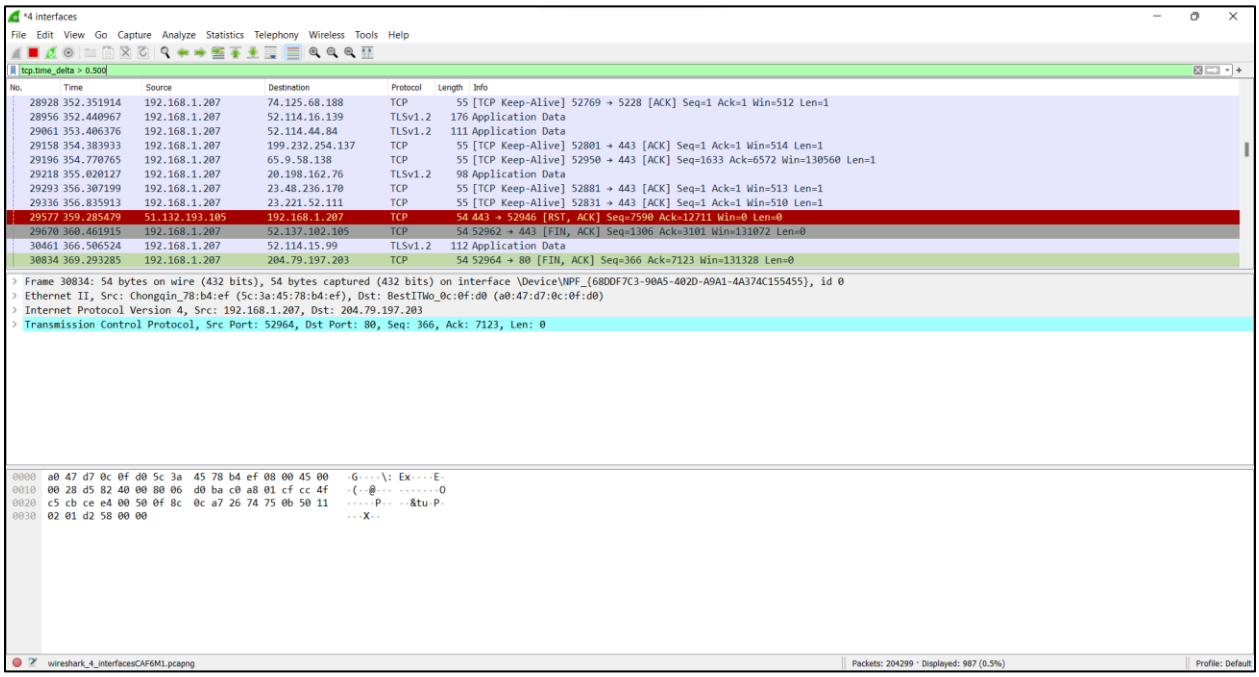
The address resolution protocol (ARP) uses a basic message format that contains either address resolution request or address resolution response.



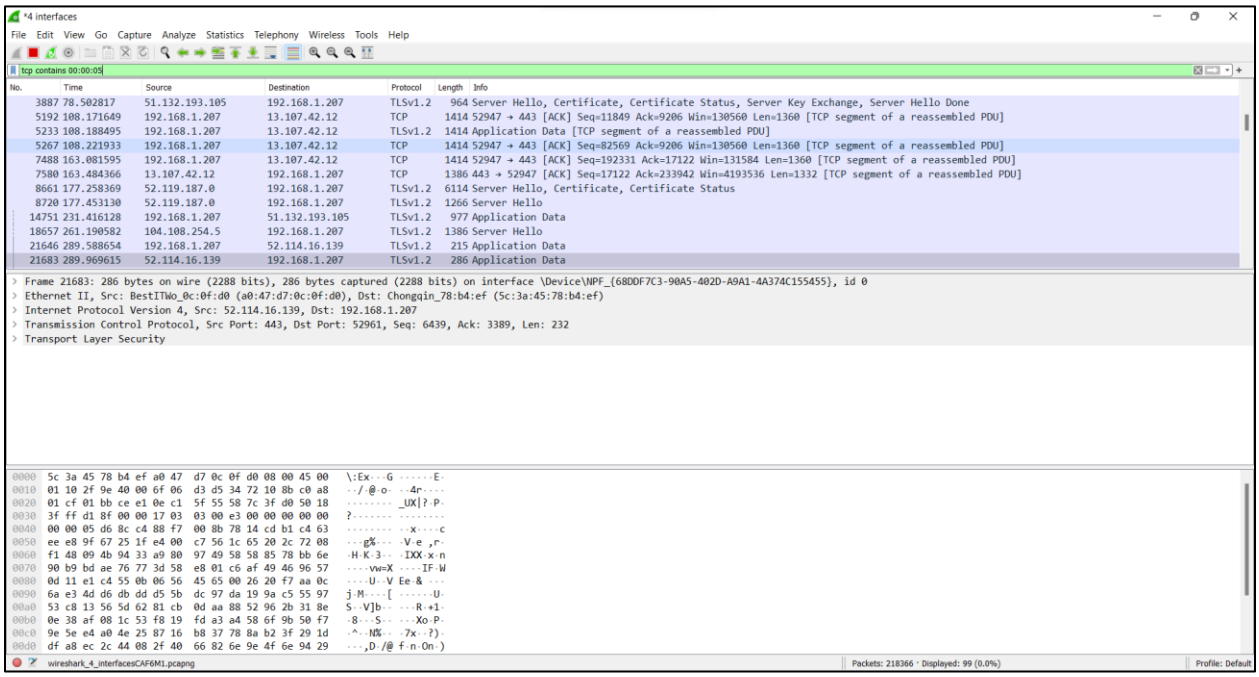
Filter 1 : Trace all packets related to Port 80



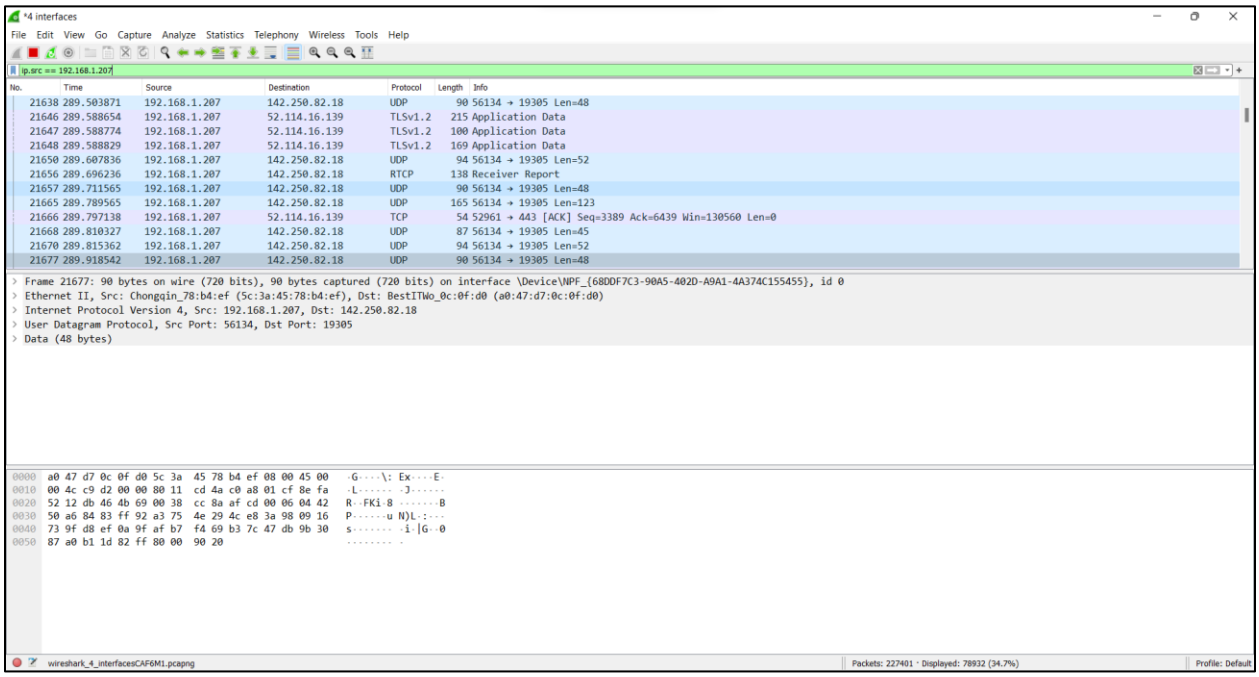
Filter 2 : Trace TCP Packets having delta time > 0.500 sec



Filter 3 : Trace Packets with a certain byte sequence



Filter 4 : Trace Packets with a given source IP Address



Conclusion

Wireshark is a powerful tool for packet analyzer. It allows to capture packets which are being sent over the live network. It has a beautiful GUI which shows the packets being captured. It allows various filters to search for particular packets being captured. Thus, we studied packet sniffer tool : Wireshark and captured ICMP, TCP, and HTTP packets in promiscuous mode and traced them based on different filters.

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Experiment 9

Date of Performance : 17-05-2022 **Date of Submission:** 18-05-2022

SAP Id: 60004190053 **Name :** Jayesh Kavedia

Div: A **Batch :** A4

Aim of Experiment

Implementation of Network Intrusion Detection System using NMAP, SNORT and IPTABLE.

Theory / Algorithm / Conceptual Description

IPTABLES

iptables is a user-space utility program that allows a system administrator to configure the IP packet filter rules of the Linux kernel firewall, implemented as different Netfilter modules. The filters are organized in different tables, which contain chains of rules for how to treat network traffic packets. Different kernel modules and programs are currently used for different protocols; iptables applies to IPv4, ip6tables to IPv6, arptables to ARP, and ebtables to Ethernet frames.

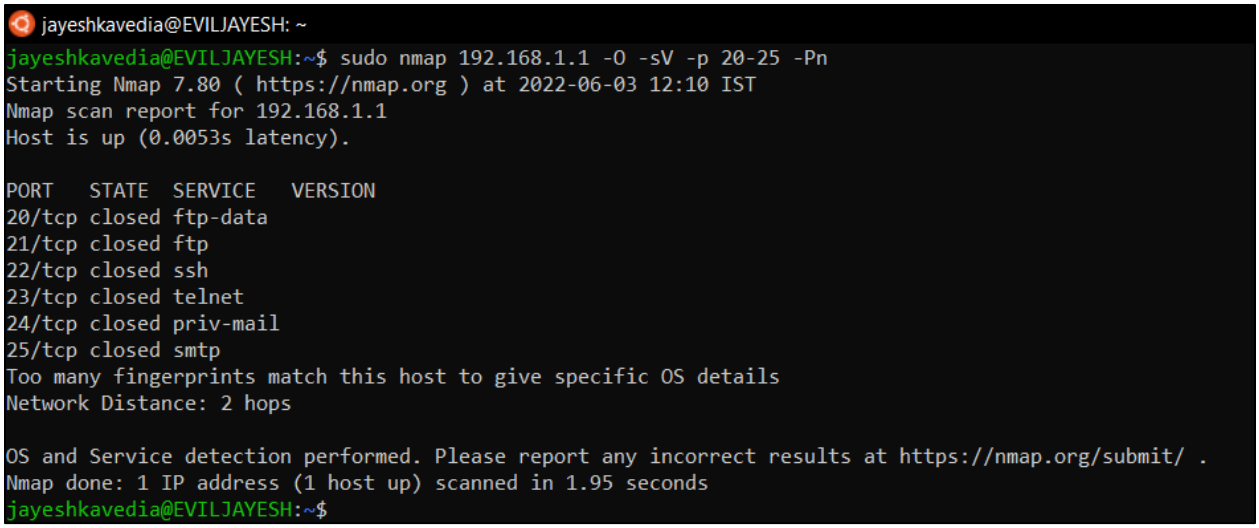
NMAP

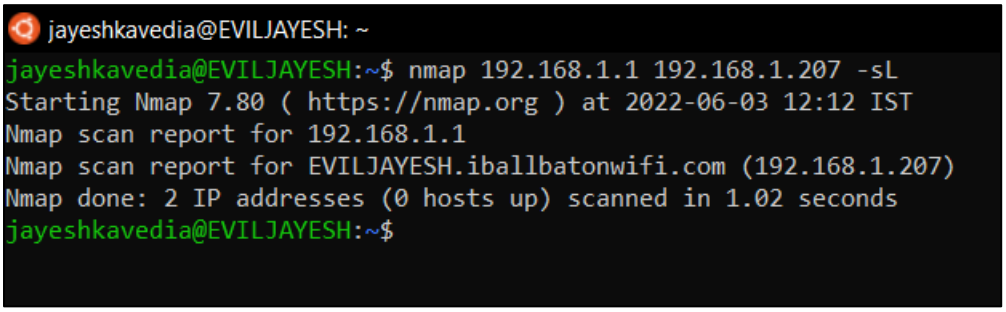
Nmap (Network Mapper) is a network scanner. Nmap is used to discover hosts and services on a computer network by sending packets and analyzing the responses. Nmap provides a number of features for probing computer networks, including host discovery and service and operating system detection. These features are extensible by scripts that provide more advanced service detection, vulnerability detection, and other features. Nmap can adapt to network conditions including latency and congestion during a scan.

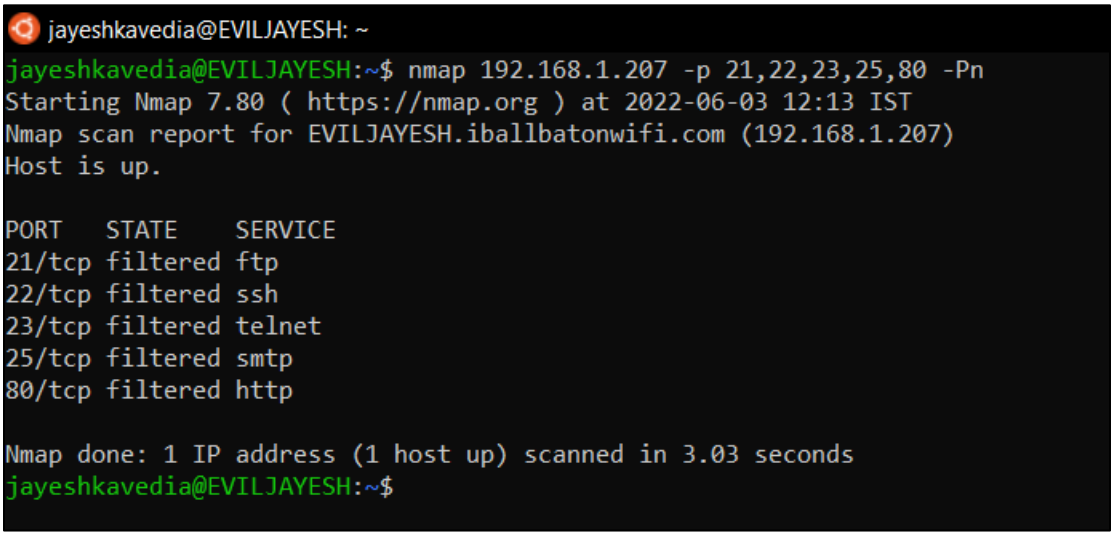
Features :

1. Host Discovery
2. Port Scanning
3. Version Detection
4. TCP/IP stack fingerprinting.

Output







SNORT

Snort is the foremost Open Source Intrusion Prevention System (IPS) in the world. Snort IPS uses a series of rules that help define malicious network activity and uses those rules to find packets that match against them and generates alerts for users. Snort can be deployed inline to stop these packets, as well. Snort has three primary uses: As a packet sniffer like tcpdump, as a packet logger — which is useful for network traffic debugging, or it can be used as a full-blown network intrusion prevention system.

root@EVILJAYESH: /etc/snort

```
root@EVILJAYESH:/etc/snort# snort -dev
Running in packet dump mode

        --== Initializing Snort ==--
Initializing Output Plugins!
pcap DAQ configured to passive.
Acquiring network traffic from "eth0".
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.7.0 GRE (Build 149)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.9.1 (with TPACKET_V3)
           Using PCRE version: 8.39 2016-06-14
           Using ZLIB version: 1.2.11

Commencing packet processing (pid=6116)
WARNING: No preprocessors configured for policy 0.
06/03-13:11:33.895227 00:15:5D:D1:22:83 -> 01:00:5E:7F:FF:FA type:0x800 len:0xD8
172.22.176.1:58882 -> 239.255.255.250:1900 UDP TTL:1 TOS:0x0 ID:14030 IpLen:20 DgmLen:202
Len: 174
4D 2D 53 45 41 52 43 48 20 2A 20 48 54 54 50 2F  M-SEARCH * HTTP/
31 2E 31 0D 0A 48 4F 53 54 3A 20 32 33 39 2E 32  1.1..HOST: 239.2
35 35 2E 32 35 35 2E 32 35 30 3A 31 39 30 30 0D  55.255.250:1900.
0A 4D 41 4E 3A 20 22 73 73 64 70 3A 64 69 73 63  .MAN: "ssdp:disc
6F 76 65 72 22 0D 0A 4D 58 3A 20 31 0D 0A 53 54  over"..MX: 1..ST
3A 20 75 72 6E 3A 64 69 61 6C 2D 6D 75 6C 74 69  : urn:dial-multi
73 63 72 65 65 6E 2D 6F 72 67 3A 73 65 72 76 69  screen-org:servi
63 65 3A 64 69 61 6C 3A 31 0D 0A 55 53 45 52 2D  ce:dial:1..USER-
41 47 45 4E 54 3A 20 47 6F 6F 67 6C 65 20 43 68  AGENT: Google Ch
72 6F 6D 65 2F 31 30 32 2E 30 2E 35 30 30 35 2E  rome/102.0.5005.
36 33 20 57 69 6E 64 6F 77 73 0D 0A 0D 0A        63 Windows....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Conclusion

NMAP is useful for port scanning and SNORT is a powerful tool for Network Intrusion Detection and Prevention. Thus, we successfully implemented Network Intrusion Detection System using NMAP, SNORT and IPTABLE.

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Experiment 10

Date of Performance : 24-05-2022 **Date of Submission:** 25-05-2022

SAP Id: 60004190053 **Name :** Jayesh Kavedia

Div: A **Batch :** A4

Aim of Experiment

Implement Buffer Overflow Attack.

Theory / Algorithm / Conceptual Description

A buffer is a temporary area for data storage. When more data (than was originally allocated to be stored) gets placed by a program or system process, the extra data overflows. It causes some of that data to leak out into other buffers, which can corrupt or overwrite whatever data they were holding. In a buffer-overflow attack, the extra data sometimes holds specific instructions for actions intended by a hacker or malicious user; for example, the data could trigger a response that damages files, changes data or unveils private information.

Attacker would use a buffer-overflow exploit to take advantage of a program that is waiting on a user's input. There are two types of buffer overflows: stack-based and heap-based. Heap-based, which are difficult to execute and the least common of the two, attack an application by flooding the memory space reserved for a program. Stack-based buffer overflows, which are more common among attackers, exploit applications and programs by using what is known as a stack: memory space used to store user input.

Ollydbg

OllyDbg is an x86 debugger that emphasizes binary code analysis, which is useful when source code is not available. It traces registers, recognizes procedures, API calls, switches, tables, constants and strings, as well as locates routines from object files and libraries. OllyDbg is often used for reverse engineering of programs. It is often used by crackers to crack software made by other developers. It is also useful for programmers to ensure that their program is running as intended, and for malware analysis purposes.

Splint

Splint, short for Secure Programming Lint, is a programming tool for statically checking C programs for security vulnerabilities and coding mistakes. With minimal effort, Splint can be used as a better lint. If additional effort is invested adding annotations to programs, Splint can perform stronger checking than can be done by any standard lint. Splint is particularly good at checking type checking of variable and function assignments, efficiency, unused variables and

function identifiers, unreachable code and possible memory leaks. There are many useful options to help control splint.

Cppcheck

Cppcheck is a static code analysis tool for the C and C++ programming languages. It is a versatile tool that can check non-standard code. Cppcheck supports a wide variety of static checks that may not be covered by the compiler itself. These checks are static analysis checks that can be performed at a source code level. The program is directed towards static analysis checks that are rigorous, rather than heuristic in nature. Some of the checks that are supported include:

1. Automatic variable checking
2. Bounds checking for array overruns
3. Classes checking
4. Usage of deprecated or superseded functions according to Open Group
5. Exception safety checking, for example usage of memory allocation and destructor checks
6. Memory leaks
7. Resource leaks
8. Invalid usage of Standard Template Library functions and idioms
9. Dead code elimination using unused Function option
10. Miscellaneous stylistic and performance errors

Program

A. Buffer Overflow Attack

```
#include <stdio.h>
#include <string.h>

#define UP_MAXLEN 20
#define UP_PAIR_COUNT 3

int main()
{
    int flag;
    char termBuf;
    char username[UP_MAXLEN];
    char cpass[UP_MAXLEN];
    char npass[UP_MAXLEN];
    char keys[UP_PAIR_COUNT][2][UP_MAXLEN] = {
        {"Admin", "pass1234"},
        {"Max", "Hasten"},
        {"Sally", "Monarch"}};
    while (1)
    {
        flag = 0;
        printf("Change Password \n");
        printf("Enter Username: ");
```

```
        gets(username);
        printf("Enter Current Password: ");
        gets(cpass);
        for (int i =0;i<UP_PAIR_COUNT;i++){
            if(strcmp(keys[i][0],username) == 0 && strcmp(keys[i][1],cpass)
== 0){
                printf("Enter New Password: ");
                gets(npass);
                strcpy(&keys[i][1][0],npass);
                for(int j=0;j<UP_PAIR_COUNT;j++){
                    printf("%s | %s\n",keys[j][0],keys[j][1]);
                }
                printf("Password Changed!\n");
                printf("Continue? Y/N: ");
                gets(&termBuf);
                if (termBuf != 'Y') return 0;
                else flag = 1;
            }
        }
        if (flag == 1) continue;
        printf("Incorrect Username and Password. Enter Y to continue.\n");
        gets(&termBuf);
        if(termBuf != 'Y') return 0;
    }
}
```

B. Fixing Buffer Overflow Attack

```
#include <stdio.h>
#include <string.h>

#define UP_MAXLEN 20
#define UP_PAIR_COUNT 3

int main()
{
    int flag;
    char termBuf;
    char username[UP_MAXLEN];
    char cpass[UP_MAXLEN];
    char npass[UP_MAXLEN];
    char keys[UP_PAIR_COUNT][2][UP_MAXLEN] = {
        {"Admin", "pass1234"},
        {"Max", "Hasten"},
        {"Sally", "Monarch"}
    };
```

```
    while(1){
        flag = 0;
        printf("Change Password \n");
        printf("Enter Username: ");
        fgets(username,UP_MAXLEN,stdin);
        username[strcspn(username, "\r\n")] = 0;
        printf("Enter Current Password: ");
        fgets(cpass,UP_MAXLEN,stdin);
        cpass[strcspn(cpass,"\r\n")] = 0;
        for (int i =0;i<UP_PAIR_COUNT;i++){
            if(strcmp(keys[i][0],username) == 0 && strcmp(keys[i][1],cpass)
== 0){
                printf("Enter New Password: ");
                fgets(npass,UP_MAXLEN,stdin);
                npass[strcspn(npass,"\n")] = 0;
                strcpy(&keys[i][1][0],npass);
                for(int j=0;j<UP_PAIR_COUNT;j++){
                    printf("%s | %s\n",keys[j][0],keys[j][1]);
                }
                printf("Password Changed!\n");
                printf("Continue? Y/N: ");
                scanf("%c",&termBuf);
                if (termBuf != 'Y') return 0;
                else flag = 1;
                while((termBuf = getchar()) != '\n' && termBuf != EOF);
            }
        }
        if (flag == 1) continue;
        printf("Incorrect Username and Password. Enter Y to continue.\n");
        scanf("%c",&termBuf);
        if(termBuf != 'Y') return 0;
        while((termBuf = getchar()) != '\n' && termBuf != EOF);
    }
}
```

Output

A. Buffer Overflow Attack

```
PS C:\Users\DELL>  & 'c:\Users\DELL\.vscode\extensions\ms-vscode.cpptools-
tdout=Microsoft-MIEngine-Out-nezyavgs.igl' '--stderr=Microsoft-MIEngine-Er
ter=mi'
Change Password
Enter Username: Admin
Enter Current Password: pass1234
Enter New Password: qwertyuiopasdfghjklzBUFFEROVERFLOWATTACKpassword
Admin | qwertyuiopasdfghjklzBUFFEROVERFLOWATTACKpassword
BUFFEROVERFLOWATTACKpassword | password
Sally | Monarch
Password Changed!
Continue? Y/N: Y
Change Password
Enter Username: BUFFEROVERFLOWATTACKpassword
Enter Current Password: password
Enter New Password: hacked
Admin | qwertyuiopasdfghjklzBUFFEROVERFLOWATTACKhacked
BUFFEROVERFLOWATTACKhacked | hacked
Sally | Monarch
Password Changed!
Continue? Y/N: N
PS C:\Users\DELL>
```

B. Solving Buffer Overflow Attack

```
PS C:\Users\DELL>  & 'c:\Users\DELL\.vscode\extensions\ms-vscode.cpptools-1.
tdout=Microsoft-MIEngine-Out-smrlyms5.2o5' '--stderr=Microsoft-MIEngine-Erro
ter=mi'
Change Password
Enter Username: Admin
Enter Current Password: pass1234
Enter New Password: qwertyuiopasdfghjklzBUFFEROVERFLOWATTACKpassword
Admin | qwertyuiopasdfghjkl
Max | Hasten
Sally | Monarch
Password Changed!
Continue? Y/N:
PS C:\Users\DELL>
```

C. Splint Output for Vulnerability

```
C:\splint-3.1.1\bin>set include=C:\TDM-GCC-64\bin

C:\splint-3.1.1\bin>splint -type -retvalother -predboolint "D:\SEM-VI\IS\EXPERIMENTS\bo.c"
Splint 3.1.1 --- 12 April 2003

D:\SEM-VI\IS\EXPERIMENTS\bo.c: (in function main)
D:\SEM-VI\IS\EXPERIMENTS\bo.c(23,9): Use of gets leads to a buffer overflow
                                     vulnerability.  Use fgets instead: gets
  Use of function that may lead to buffer overflow. (Use -bufferoverflowhigh to
  inhibit warning)
D:\SEM-VI\IS\EXPERIMENTS\bo.c(25,9): Use of gets leads to a buffer overflow
                                     vulnerability.  Use fgets instead: gets
D:\SEM-VI\IS\EXPERIMENTS\bo.c(26,17):
    Parse Error. (For help on parse errors, see splint -help parseerrors.)
*** Cannot continue.

C:\splint-3.1.1\bin>
```

D. Splint output for fixed code

```
C:\splint-3.1.1\bin>splint -type -retvalother -predboolint "D:\SEM-VI\IS\CODE\bo_solved.c"
Splint 3.1.1 --- 12 April 2003

D:\SEM-VI\IS\CODE\bo_solved.c(28,17):
    Parse Error. (For help on parse errors, see splint -help parseerrors.)
*** Cannot continue.

C:\splint-3.1.1\bin>
```

Conclusion

Buffer Overflow attack typically involves violating programming languages and overwriting the bounds of the buffers they exist on. We have successfully demonstrated how a buffer overflow attack can occur in C programming language using gets() function. Also, we demonstrated how we can fix the buffer overflow attack in C using fgets() function which performs bound checking. Also we verified the vulnerabilities of corrupt and fixed code using Splint tool.