Experiment File

Information Security

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

NATIONAL INSTITUTE OF TECHNOLOGY

HAMIRPUR -177005

Submitted by:

Ashish Dhiman 17mi538
Krishan Kumar 17mi518
Kartikey Tewari 17mi551

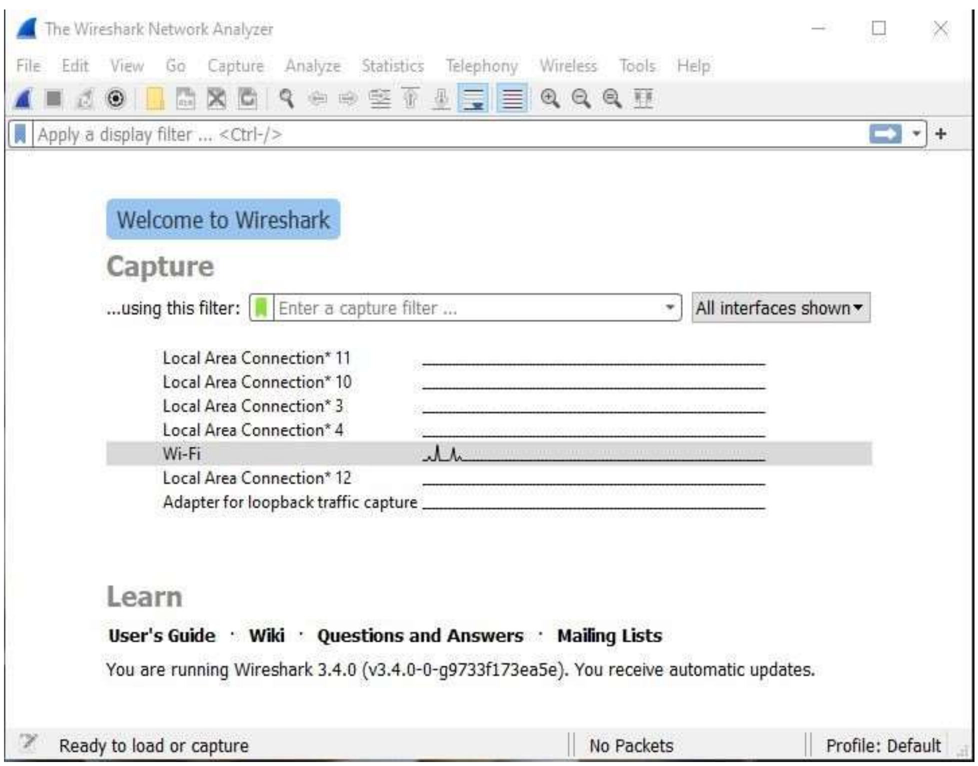
Submitted To:

Dr Narottam Chand Kaushal

Experiment 1

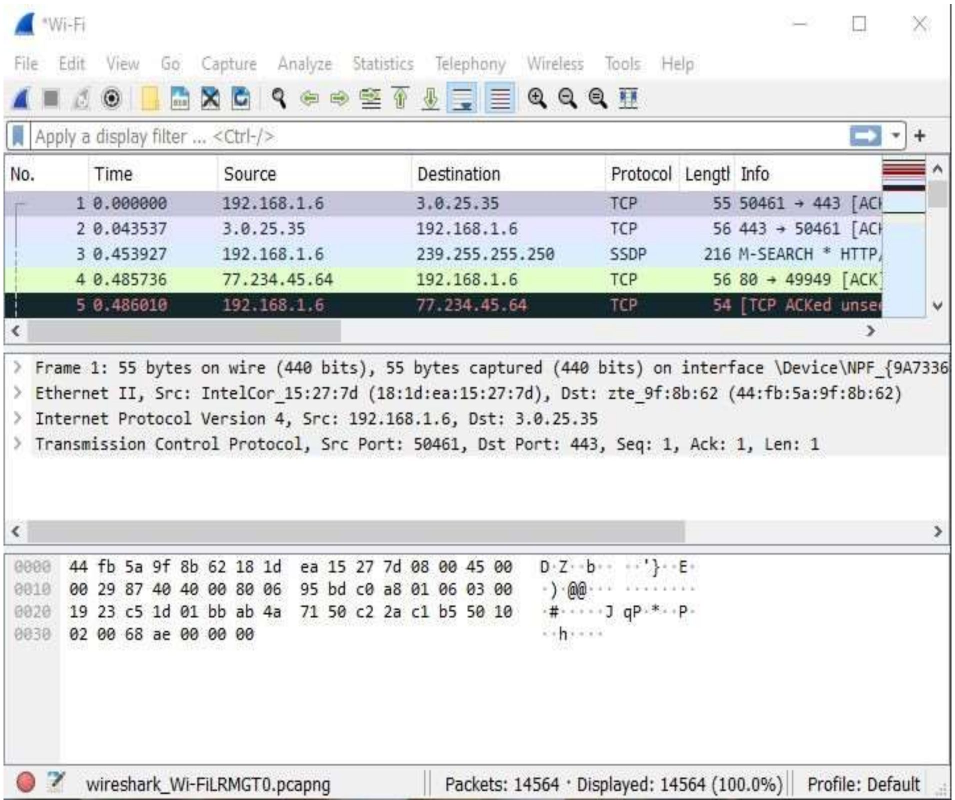
AIM : Introduction to Wireshark and implement Capture packets and Display packets in Wireshark.

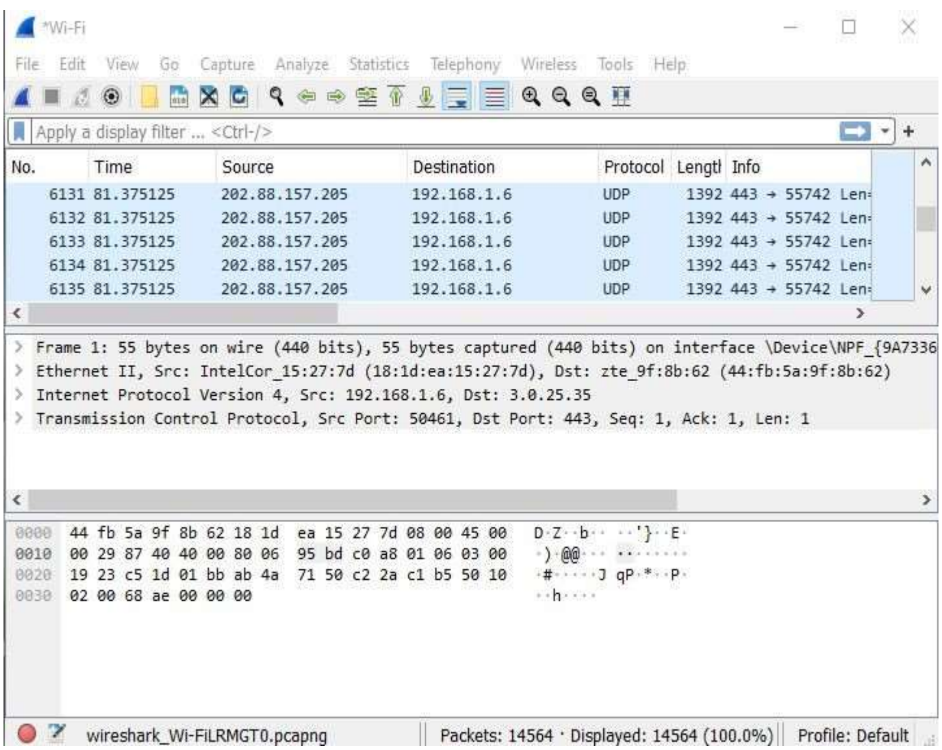
Wireshark is an open-source network protocol analysis software program started by Gerald Combs in 1998. A global organization of network specialists and software developers support Wireshark and continue to make updates for new network technologies and encryption methods. It is a free open source network protocol analyzer. It is used for network troubleshooting and communication protocol analysis. Wireshark captures network packets in real time and displays them in human-readable format. It provides many advanced features including live capture and offline analysis, three-pane packet browser, coloring rules for analysis.Wireshark is absolutely safe to use. Government agencies, corporations, non-profits, and educational institutions use Wireshark for troubleshooting and teaching purposes. There isn’t a better way to learn networking than to look at the traffic under the Wireshark microscope.

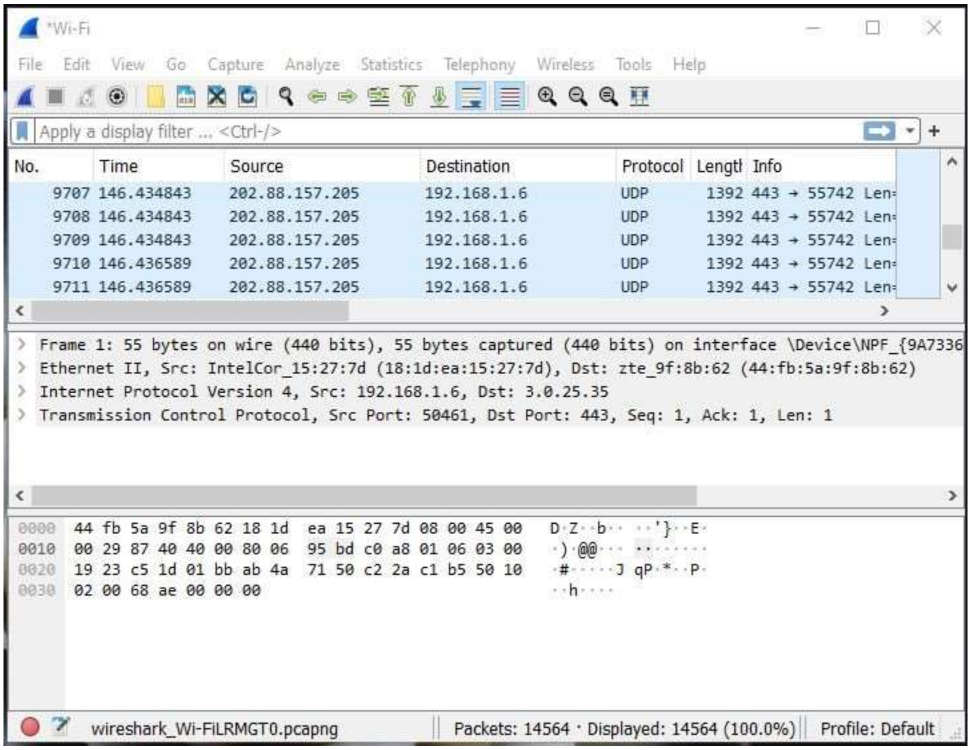


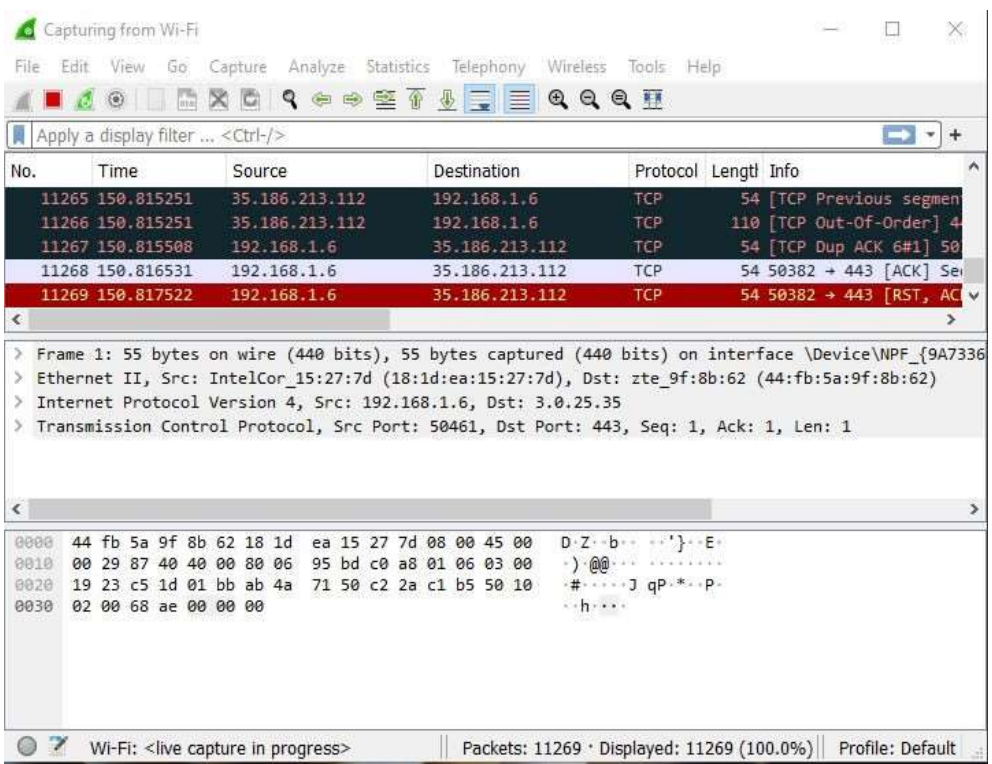
Graphic User Interface of Wireshark

Capturing Data Packets on Wireshark






*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Lengtl Info
9707 146.434843 202.88.157.205 192.168.1.6 UDP 1392 443 → 55742 Len=
9708 146.434843 202.88.157.205 192.168.1.6 UDP 1392 443 → 55742 Len=
9709 146.434843 202.88.157.205 192.168.1.6 UDP 1392 443 → 55742 Len=
9710 146.436589 202.88.157.205 192.168.1.6 UDP 1392 443 → 55742 Len=
9711 146.436589 202.88.157.205 192.168.1.6 UDP 1392 443 → 55742 Len=
Frame 1: 55 bytes on wire (440 bits), 55 bytes captured (440 bits) on interface \Device\NPF_{9A7336
Ethernet II, Src: IntelCor_15:27:7d (18:1d:ea:15:27:7d), Dst: zte_9f:8b:62 (44:fb:5a:9f:8b:62)
Internet Protocol Version 4, Src: 192.168.1.6, Dst: 3.0.25.35
Transmission Control Protocol, Src Port: 50461, Dst Port: 443, Seq: 1, Ack: 1, Len: 1
0000 44 fb 5a 9f 8b 62 18 1d ea 15 27 7d 08 00 45 00 D·Z··b·· ··'}··E·
0010 00 29 87 40 40 00 80 06 95 bd c0 a8 01 06 03 00 ·)·@@··· ········
0020 19 23 c5 1d 01 bb ab 4a 71 50 c2 2a c1 b5 50 10 ·#·····J qP·*··P·
0030 02 00 68 ae 00 00 00 ··h····
wireshark_Wi-FiLRMGT0.pcapng
Packets: 14564 · Displayed: 14564 (100.0%)
Profile: Default


Capturing from Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Lengtl Info
11265 150.815251 35.186.213.112 192.168.1.6 TCP 54 [TCP Previous segmen
11266 150.815251 35.186.213.112 192.168.1.6 TCP 110 [TCP Out-Of-Order] 4
11267 150.815508 192.168.1.6 35.186.213.112 TCP 54 [TCP Dup ACK 6#1] 50
11268 150.816531 192.168.1.6 35.186.213.112 TCP 54 50382 → 443 [ACK] Se
11269 150.817522 192.168.1.6 35.186.213.112 TCP 54 50382 → 443 [RST, AC
Frame 1: 55 bytes on wire (440 bits), 55 bytes captured (440 bits) on interface \Device\NPF_{9A7336
Ethernet II, Src: IntelCor_15:27:7d (18:1d:ea:15:27:7d), Dst: zte_9f:8b:62 (44:fb:5a:9f:8b:62)
Internet Protocol Version 4, Src: 192.168.1.6, Dst: 3.0.25.35
Transmission Control Protocol, Src Port: 50461, Dst Port: 443, Seq: 1, Ack: 1, Len: 1
0000 44 fb 5a 9f 8b 62 18 1d ea 15 27 7d 08 00 45 00 D·Z··b·· ··'}··E·
0010 00 29 87 40 40 00 80 06 95 bd c0 a8 01 06 03 00 ·)·@@··· ········
0020 19 23 c5 1d 01 bb ab 4a 71 50 c2 2a c1 b5 50 10 ·#·····J qP·*··P·
0030 02 00 68 ae 00 00 00 ··h····
Wi-Fi: <live capture in progress>
Packets: 11269 · Displayed: 11269 (100.0%)
Profile: Default

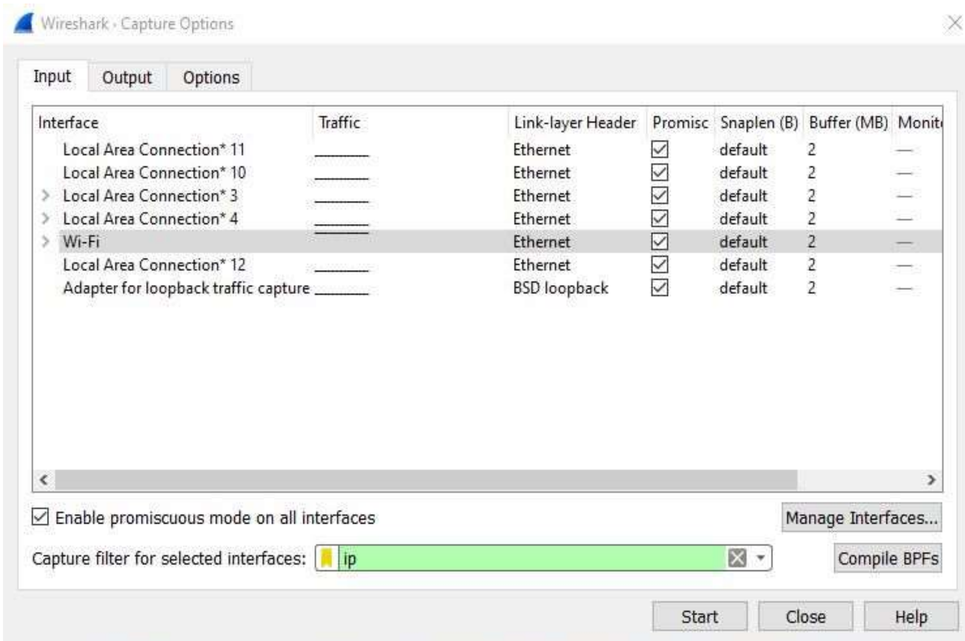
Wireshark Filters

One of the best features of Wireshark is the Wireshark Capture Filters and Wireshark Display Filters. Filters allow you to view the capture the way you need to see it so you can troubleshoot the issues at hand.

Wireshark Capture Filters

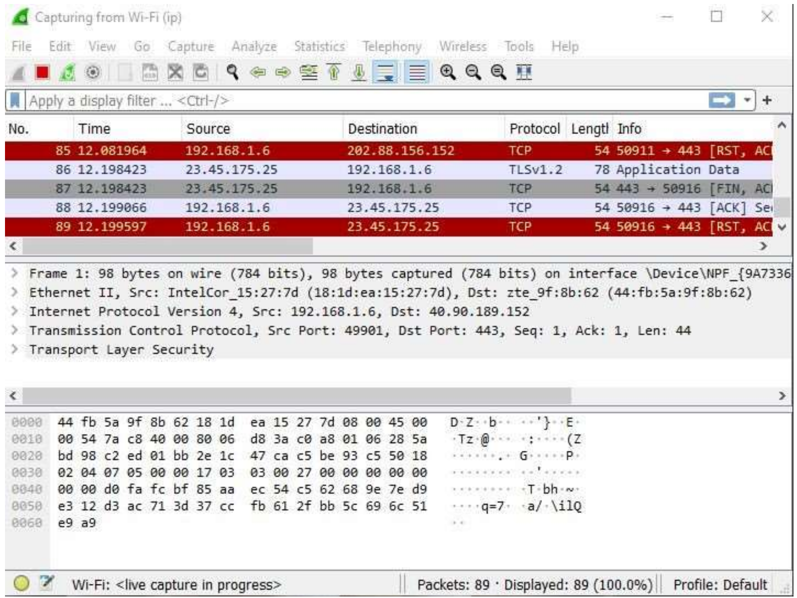
Capture filters limit the captured packets by the filter. Meaning if the packet

don’t match the filter, Wireshark won’t save them.



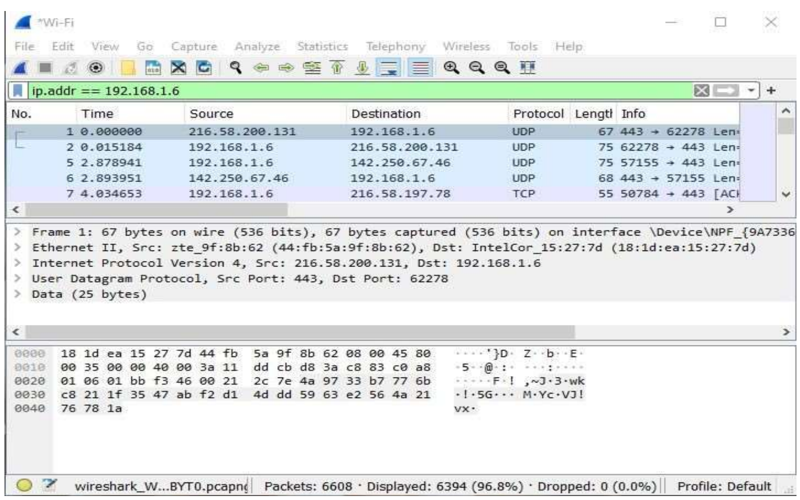
Capture only IPv4 traffic by using:

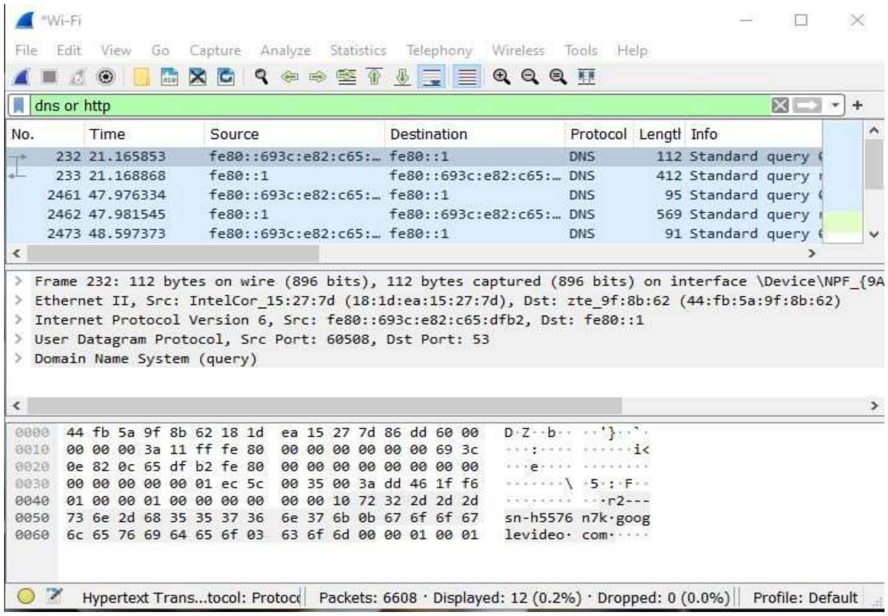
Capture Filter - ip



Wireshark Display Filters

Wireshark Display Filters change the view of the capture during analysis. After you have stopped the packet capture, you use display filters to narrow down the packets in the Packet List so you can troubleshoot your issue.





*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns or http
No. Time Source Destination Protocol Length Info
232 21.165853 fe80::693c:e82:c65:… fe80::1 DNS 112 Standard query
233 21.168868 fe80::1 fe80::693c:e82:c65:… DNS 412 Standard query
2461 47.976334 fe80::693c:e82:c65:… fe80::1 DNS 95 Standard query
2462 47.981545 fe80::1 fe80::693c:e82:c65:… DNS 569 Standard query
2473 48.597373 fe80::693c:e82:c65:… fe80::1 DNS 91 Standard query
Frame 232: 112 bytes on wire (896 bits), 112 bytes captured (896 bits) on interface \Device\NPF_{9A
Ethernet II, Src: IntelCor_15:27:7d (18:1d:ea:15:27:7d), Dst: zte_9f:8b:62 (44:fb:5a:9f:8b:62)
Internet Protocol Version 6, Src: fe80::693c:e82:c65:dfb2, Dst: fe80::1
User Datagram Protocol, Src Port: 60508, Dst Port: 53
Domain Name System (query)
0000 44 fb 5a 9f 8b 62 18 1d ea 15 27 7d 86 dd 60 00
0010 00 00 00 3a 11 ff fe 80 00 00 00 00 00 00 69 3c
0020 0e 82 0c 65 df b2 fe 80 00 00 00 00 00 00 00 00
0030 00 00 00 00 00 01 ec 5c 00 35 00 3a dd 46 1f f6
0040 01 00 00 01 00 00 00 00 00 00 10 72 32 2d 2d 2d
0050 73 6e 2d 68 35 35 37 36 6e 37 6b 0b 67 6f 6f 67
0060 6c 65 76 69 64 65 6f 03 63 6f 6d 00 00 01 00 01
sn-h5576 n7k·goog
levideo· com·
Hypertext Trans...tocol: Protoc
Packets: 6608 · Displayed: 12 (0.2%) · Dropped: 0 (0.0%)
Profile: Default

Experiment 2

AIM: Write programs to perform encryption and decryption using the following algorithms:

a) Ceasar Cipher

Encryption:

```
#include<iostream>

using namespace std;

int main()
{
	char message[100], ch;
	int i, key;

	cout << "Enter a message to encrypt:
	"; cin.getline(message, 100); cout <<
	"Enter key: "; cin >> key;

	for(i = 0; message[i] != '\0'; ++i){
		ch = message[i];

		if(ch >= 'a' && ch <= 'z'){
			ch = ch + key;

			if(ch > 'z'){
				ch = ch - 'z' + 'a' - 1;
			}

			message[i] = ch;
		}
		else if(ch >= 'A' && ch <= 'Z'){
			ch = ch + key;

			if(ch > 'Z'){
				ch = ch - 'Z' + 'A' - 1;
			}
```

```
                        message[i] = ch;
                }
        }

        cout << "Encrypted message: " << message;

        return 0;
}
```

OUTPUT:

```
Enter a message to encrypt: tutorial
Enter key: 3
Encrypted message: wxwruldo
--------------------------------
Process exited after 40.54 seconds with return value 0
Press any key to continue . . .
```

Decryption:

```
#include<iostream>

using namespace std;

int main()
{
        char message[100], ch;
        int i, key;
        cout << "Enter a message to decrypt:
        "; cin.getline(message, 100); cout <<
        "Enter key: ";
```

```
	cin >> key;

	for(i = 0; message[i] != '\0'; ++i){
		ch = message[i];

		if(ch >= 'a' && ch <= 'z'){
			ch = ch - key;

			if(ch < 'a'){
				ch = ch + 'z' - 'a' + 1;
			}

			message[i] = ch;
		}
		else if(ch >= 'A' && ch <= 'Z'){
			ch = ch - key;

			if(ch > 'a'){
				ch = ch + 'Z' - 'A' + 1;
			}

			message[i] = ch;
		}
	}

	cout << "Decrypted message: " << message;

	return 0;
}
```

OUTPUT:

```
Enter a message to decrypt: wxwruldo
Enter key: 3
Decrypted message: tutorial
--------------------------------
Process exited after 34.94 seconds with return value 0
Press any key to continue . . .
```

b) Substitution Cipher

```
import string

all_letters =
string.ascii_letters dict1 = {}
key = 4

for i in range(len(all_letters)):
   dict1[all_letters[i]] = all_letters[(i + key) % len(all_letters)]

plain_txt = "I am studying Substitution cipher"
cipher_txt = []

for char in plain_txt:    if char
in all_letters:        temp =
dict1[char]
cipher_txt.append(temp)
else:
       temp = char
       cipher_txt.append(temp)

cipher_txt = "".join(cipher_txt)
print("Cipher Text is: ", cipher_txt)

dict2 = {} for i in
range(len(all_letters)):
   dict2[all_letters[i]] = all_letters[(i - key) % (len(all_letters))]

decrypt_txt = []

for char in cipher_txt:    if char
in all_letters:        temp =
dict2[char]
decrypt_txt.append(temp)
else:
       temp = char
decrypt_txt.append(temp)

decrypt_txt = "".join(decrypt_txt)
print("Recovered plain text :", decrypt_txt)
```

OUTPUT:

```
substitution
C:\Users\Sahana\PycharmProjects\pythonProject\venv\Scripts\python.exe
Cipher Text is:  M eq wxyhCmrk Wyfwxmxyxmsr gmtliv
Recovered plain text : I am studying Substitution cipher

Process finished with exit code 0
```

c) Hill Cipher

```
#include<iostream>
#include<math.h>

using namespace std;

float encrypt[3][1], decrypt[3][1], a[3][3], b[3][3], mes[3][1], c[3][3];

void encryption();     //encrypts the message void
decryption();  //decrypts the message void getKeyMessage();
        //gets key and message from user void inverse();
        //finds inverse of key matrix

int main() {
        getKeyMessage();
        encryption();
        decryption();
}

void encryption() {
        int i, j, k;

        for(i = 0; i < 3; i++) for(j = 0; j < 1; j++) for(k = 0; k < 3; k++)
                encrypt[i][j] = encrypt[i][j] + a[i][k] * mes[k][j];

        cout<<"\nEncrypted string is: ";
        for(i = 0; i < 3; i++)
                cout<<(char)(fmod(encrypt[i][0], 26) + 97);
}
```

```
void decryption() {
	int i, j, k;

	inverse();

	for(i = 0; i < 3; i++)
		for(j = 0; j < 1; j++) for(k = 0; k < 3; k++) decrypt[i][j] =
			decrypt[i][j] + b[i][k] * encrypt[k][j];

	cout<<"\nDecrypted string is: ";
	for(i = 0; i < 3; i++)
		cout<<(char)(fmod(decrypt[i][0], 26) + 97);

	cout<<"\n";
}

void getKeyMessage() {
	int i, j;
	char msg[3];

	cout<<"Enter 3x3 matrix for key (It should be inversible):\n";

	for(i = 0; i < 3; i++) for(j =
		0; j < 3; j++) {
			cin>>a[i][j];
			c[i][j] = a[i][j];
		}

	cout<<"\nEnter a 3 letter string: ";
	cin>>msg;

	for(i = 0; i < 3; i++) mes[i][0]
		= msg[i] - 97;
}

void inverse() {
	int i, j, k;
	float p, q;

	for(i = 0; i < 3; i++)
		for(j = 0; j < 3; j++) {
			if(i == j)
				b[i][j]=1;
			else
				b[i][j]=0;
```

```
                }

        for(k = 0; k < 3; k++) {
                for(i = 0; i < 3; i++) {
                        p = c[i][k];
                        q = c[k][k];

                        for(j = 0; j < 3; j++) { if(i != k) { c[i][j] =
                                c[i][j]*q - p*c[k][j]; b[i][j] =
                                b[i][j]*q - p*b[k][j];
                                }
                        }
                }
        }

        for(i = 0; i < 3; i++)
                for(j = 0; j < 3; j++) b[i][j] =
                        b[i][j] / c[i][i];

        cout<<"\n\nInverse Matrix is:\n";
        for(i = 0; i < 3; i++) { for(j
                = 0; j < 3; j++)
                        cout<<b[i][j]<<" ";

                cout<<"\n";
        }
}
```

OUTPUT:

```
Enter 3x3 matrix for key (It should be inversible):
6 24 1
13 16 10
20 17 15

Enter a 3 letter string: act

Encrypted string is: poh

Inverse Matrix is:
0.15873 -0.777778 0.507937
0.0113379 0.15873 -0.106576
-0.22449 0.857143 -0.489796

Decrypted string is: act

--------------------------------
Process exited after 38.83 seconds with return value 0
Press any key to continue . . .
```

d) Vigenere Cipher

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

using namespace std;

int main(){
    char msg[] = "INFORMATIONSECURITY";
    char key[] = "MANUAL";
    int msgLen = strlen(msg), keyLen = strlen(key), i, j;

    char newKey[msgLen], encryptedMsg[msgLen], decryptedMsg[msgLen];

    //generating new key
    for(i = 0, j = 0; i < msgLen; ++i, ++j){
if(j == keyLen)
            j = 0;

        newKey[i] = key[j];
    }

    newKey[i] = '\0';

    //encryption
    for(i = 0; i < msgLen; ++i)
        encryptedMsg[i] = ((msg[i] + newKey[i]) % 26) + 'A';

    encryptedMsg[i] = '\0';

    //decryption
    for(i = 0; i < msgLen; ++i)         decryptedMsg[i] = (((encryptedMsg[i] -
newKey[i]) + 26) % 26) + 'A';

    decryptedMsg[i] = '\0';

    cout<<"Original Message: "<<msg;
cout<<"\nKey: "<<key;     cout<<"\nNew Generated
Key: "<<newKey;     cout<<"\nEncrypted Message:
"<<encryptedMsg;     cout<<"\nDecrypted
Message: "<<decryptedMsg;
  return 0;
}
```

OUTPUT:

```
Original Message: INFORMATIONSECURITY
Key: MANUAL
New Generated Key: MANUALMANUALMANUALM
Encrypted Message: UNSIRXMTVINDQCHLIEK
Decrypted Message: INFORMATIONSECURITY
--------------------------------
Process exited after 2.806 seconds with return value 0
Press any key to continue . . .
```

Experiment 3

AIM: To demonstrate the working of Playfair Cipher.

The Playfair cipher was the first practical digraph substitution cipher. The scheme was invented in 1854 by Charles Wheatstone but was named after Lord Playfair who promoted the use of the cipher. In playfair cipher, unlike traditional ciphe r we encrypt a pair of alphabets(digraphs) instead of a single alphabet. The Playfair Cipher Encryption Algorithm:
The Algorithm consists of 2 steps:

1.Generate the key Square(5×5):

- The key square is a 5×5 grid of alphabets that acts as the key for encrypting the plaintext. Each of the 25 alphabets must be unique and one letter of the alphabet (usually J) is omitted from the table (as the table can hold only 25 alphabets). If the plaintext contains J, then it is replaced by I.
- The initial alphabets in the key square are the unique alphabets of the key in the order in which they appear followed by the remaining letters of the alphabet in order.

2. Algorithm to encrypt the plain text: The plaintext is split into pairs of two letters (digraphs). If there is an odd number of letters, a Z is added to the last letter.

Rules for Encryption:

- If both the letters are in the same column: Take the letter below each one (going back to the top if at the bottom)
- If both the letters are in the same row: Take the letter to the right of each one (going back to the leftmost if at the rightmost position).
- If neither of the above rules is true: Form a rectangle with the two letters and take the letters on the horizontal opposite corner of the rectangle.

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

#define SIZE 30


void toLowerCase(char plain[], int ps)
{
    int i;
    for (i = 0; i < ps; i++) {          if (plain[i] > 64 && plain[i] < 91)
            plain[i] += 32;
    }
}


int removeSpaces(char* plain, int ps)
{      int i, count = 0;      for (i = 0; i < ps; i++)
        if (plain[i] != ' ')              plain[count++] = plain[i];
plain[count] = '\0';
    return count;
}


void generateKeyTable(char key[], int ks, char keyT[5][5])
{
    int i, j, k, flag = 0, *dicty;

    dicty = (int*)calloc(26, sizeof(int));
    for (i = 0; i < ks; i++) {          if (key[i] != 'j')
dicty[key[i] - 97] = 2;
    }

    dicty['j' - 97] = 1;

    i = 0;      j = 0;
      for (k = 0; k < ks; k++) {          if (dicty[key[k] - 97] == 2) {
dicty[key[k] - 97] -= 1;              keyT[i][j] = key[k];
            j++;              if (j == 5) {                  i++;
j = 0;
            }
        }
    }
```

```
      for (k = 0; k < 26; k++) {           if (dicty[k] == 0) {
keyT[i][j] = (char)(k + 97);
            j++;              if (j == 5) {                  i++;
j = 0;
            }
        }
    }
}


void search(char keyT[5][5], char a, char b, int arr[])
{      int i, j;

    if (a == 'j')          a = 'i';      else if (b == 'j')
        b = 'i';

    for (i = 0; i < 5; i++) {

        for (j = 0; j < 5; j++) {

            if (keyT[i][j] == a) {                  arr[0] = i;
arr[1] = j;
            }
            else if (keyT[i][j] == b) {                  arr[2] = i;
arr[3] = j;
            }
        }
    }
}
  int mod5(int a)
{
    return (a % 5);
}

int prepare(char str[], int ptrs)
{      if (ptrs % 2 != 0) {          str[ptrs++] = 'z';
str[ptrs] = '\0';
    }
    return ptrs;
}



void encrypt(char str[], char keyT[5][5], int ps)
{      int i, a[4];
```

```
    for (i = 0; i < ps; i += 2) {

        search(keyT, str[i], str[i + 1], a);

        if (a[0] == a[2]) {              str[i] = keyT[a[0]][mod5(a[1] + 1)];
            str[i + 1] = keyT[a[0]][mod5(a[3] + 1)];
        }
        else if (a[1] == a[3]) {              str[i] = keyT[mod5(a[0] + 1)][a[1]];
str[i + 1] = keyT[mod5(a[2] + 1)][a[1]];
        }         else {              str[i] = keyT[a[0]][a[3]];
            str[i + 1] = keyT[a[2]][a[1]];
        }
    }
}


void encryptByPlayfairCipher(char str[], char key[])
{
    char ps, ks, keyT[5][5];


    ks = strlen(key);
```

OUTPUT:

```
Key text: Monarchy
Plain text: instruments
Cipher text: gatlmzclrqtx

--------------------------------
Process exited after 3.129 seconds with return value 0
Press any key to continue . . .
```

Experiment 4

AIM: Write programs to demonstrate Data Encryption Standard (DES) and Triple Data Encryption Standard (3DES)

Data Encryption Standard (DES):

```
import java.util.*;

class Main {
    private static class DES {

        int[] IP = { 58, 50, 42, 34, 26, 18,
10, 2, 60, 52, 44, 36, 28, 20,
                12, 4, 62, 54, 46, 38,
                30, 22, 14, 6, 64, 56,
                48, 40, 32, 24, 16, 8,
                57, 49, 41, 33, 25, 17,
                9, 1, 59, 51, 43, 35, 27,
                19, 11, 3, 61, 53, 45,
                37, 29, 21, 13, 5, 63, 55,
                47, 39, 31, 23, 15, 7 };

        int[] IP1 = { 40, 8, 48, 16, 56, 24,
64,
                32, 39, 7, 47, 15, 55,
                23, 63, 31, 38, 6, 46,
                14, 54, 22, 62, 30, 37,
                5, 45, 13, 53, 21, 61,
                29, 36, 4, 44, 12, 52,
                20, 60, 28, 35, 3, 43,
                11, 51, 19, 59, 27, 34,
                2, 42, 10, 50, 18, 58,
                26, 33, 1, 41, 9, 49,
                17, 57, 25 };


        int[] PC1 = { 57, 49, 41, 33, 25,
17, 9, 1, 58, 50, 42, 34, 26,
                18, 10, 2, 59, 51, 43, 35, 27,
                19, 11, 3, 60, 52, 44, 36, 63,
                55, 47, 39, 31, 23, 15, 7, 62,
                54, 46, 38, 30, 22, 14, 6, 61,
```

```
            53, 45, 37, 29, 21, 13, 5, 28,
            20, 12, 4 };


      int[] PC2 = { 14, 17, 11, 24, 1, 5, 3,             28, 15,
6, 21, 10, 23, 19, 12,             4, 26, 8, 16, 7, 27, 20, 13,
2,
            41, 52, 31, 37, 47, 55, 30, 40,
            51, 45, 33, 48, 44, 49, 39, 56,
            34, 53, 46, 42, 50, 36, 29, 32 };


      int[] EP = { 32, 1, 2, 3, 4, 5, 4,             5, 6, 7,
8, 9, 8, 9, 10,
            11, 12, 13, 12, 13, 14, 15,
            16, 17, 16, 17, 18, 19, 20,
            21, 20, 21, 22, 23, 24, 25,
            24, 25, 26, 27, 28, 29, 28,
            29, 30, 31, 32, 1 };

      int[] P = { 16, 7, 20, 21, 29, 12, 28,
            17, 1, 15, 23, 26, 5, 18,
            31, 10, 2, 8, 24, 14, 32,
            27, 3, 9, 19, 13, 30, 6,
            22, 11, 4, 25 };

      int[][][] sbox = {
            { { 14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7 },
                  { 0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8 },
                  { 4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0 },
                  { 15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13 } },

            { { 15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10 },
                  { 3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5 },
                  { 0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15 },
                  { 13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9 } },
            { { 10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8 },
                  { 13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1 },
                  { 13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7 },
                  { 1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12 } },
            { { 7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15 },
                  { 13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9 },
                  { 10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4 },
                  { 3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14 } },
```

```
        { { 2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9 },
            { 14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6 },
            { 4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14 },
            { 11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3 } },
        { { 12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11 },
            { 10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8 },
            { 9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6 },
            { 4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13 } },
        { { 4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1 },
            { 13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6 },
            { 1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2 },
            { 6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12 } },
        { { 13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7 },
            { 1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2 },
            { 7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8 },
            { 2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11 } }
    };
    int[] shiftBits = { 1, 1, 2, 2, 2, 2, 2, 2,
        1, 2, 2, 2, 2, 2, 2, 1 };

    String hextoBin(String input)
    {
      int n = input.length() * 4;             input =
Long.toBinaryString(
          Long.parseUnsignedLong(input, 16));
      while (input.length() < n)              input =
"0" + input;             return input;
    }

    String binToHex(String input)
    {
      int n = (int)input.length() / 4;             input =
Long.toHexString(
          Long.parseUnsignedLong(input, 2));
      while (input.length() < n)              input =
"0" + input;             return input;
    }

    String permutation(int[] sequence, String input)
    {
      String output = "";             input = hextoBin(input);
for (int i = 0; i < sequence.length; i++)              output +=
input.charAt(sequence[i] - 1);             output = binToHex(output);
return output;
```

```
    }

    String xor(String a, String b)
    {

        long t_a = Long.parseUnsignedLong(a, 16);

        long t_b = Long.parseUnsignedLong(b, 16);

        t_a = t_a ^ t_b;

        a = Long.toHexString(t_a);

        while (a.length() < b.length())
            a = "0" + a;            return a;
    }


    String leftCircularShift(String input, int numBits)
    {
        int n = input.length() * 4;            int perm[] = new
int[n];            for (int i = 0; i < n - 1; i++)                perm[i] =
(i + 2);            perm[n - 1] = 1;            while (numBits-- > 0)
input = permutation(perm, input);
        return input;
    }



    String[] getKeys(String key)
    {
        String keys[] = new String[16];

        key = permutation(PC1, key);            for (int i
= 0; i < 16; i++) {                key = leftCircularShift(
                    key.substring(0, 7), shiftBits[i])                        +
leftCircularShift(key.substring(7, 14),

            keys[i] = permutation(PC2, key);
        }            return keys;
    }

    String sBox(String input)
    {
```

```
        String output = "";           input = hextoBin(input);
for (int i = 0; i < 48; i += 6) {              String temp =
input.substring(i, i + 6);
            int num = i / 6;              int row =
Integer.parseInt(
                  temp.charAt(0) + "" + temp.charAt(5), 2);               int
col = Integer.parseInt(                     temp.substring(1, 5), 2);
output += Integer.toHexString(
                  sbox[num][row][col]);
        }           return output;
    }

    String round(String input, String key, int num)
    {
        // fk
        String left = input.substring(0, 8);
        String temp = input.substring(8, 16);
        String right = temp;           // Expansion
permutation           temp = permutation(EP, temp);
// xor temp and round key           temp = xor(temp,
key);           // lookup in s-box table           temp =
sBox(temp);           // Straight D-box           temp =
permutation(P, temp);
        // xor
        left = xor(left, temp);           System.out.println("Round "
              + (num + 1) + " "
              + right.toUpperCase()
              + " " + left.toUpperCase() + " "
              + key.toUpperCase());


        return right + left;
    }

    String encrypt(String plainText, String key)
    {
        int i;

        String keys[] = getKeys(key);

        // initial permutation
        plainText = permutation(IP, plainText);
        System.out.println(
              "After initial permutation: "
```

```
                    + plainText.toUpperCase());
        System.out.println(
              "After splitting: L0="
                    + plainText.substring(0, 8).toUpperCase()
                    + " R0="
                    + plainText.substring(8, 16).toUpperCase() + "\n");          for (i =
0; i < 16; i++) {
            plainText = round(plainText, keys[i], i);
        }

        plainText = plainText.substring(8, 16)
              + plainText.substring(0, 8);


        plainText = permutation(IP1, plainText);
        return plainText;
    }

    String decrypt(String plainText, String key)
    {
```

OUTPUT:

```
Encryption:

After initial permutation: 14A7D67818CA18AD
After splitting: L0=14A7D678 R0=18CA18AD

Round 1 18CA18AD 5A78E394 194CD072DE8C
Round 2 5A78E394 4A1210F6 4568581ABCCE
Round 3 4A1210F6 B8089591 06EDA4ACF5B5
Round 4 B8089591 236779C2 DA2D032B6EE3
Round 5 236779C2 A15A4B87 69A629FEC913
Round 6 A15A4B87 2E8F9C65 C1948E87475E
Round 7 2E8F9C65 A9FC20A3 708AD2DDB3C0
Round 8 A9FC20A3 308BEE97 34F822F0C66D
Round 9 308BEE97 10AF9D37 84BB4473DCCC
Round 10 10AF9D37 6CA6CB20 02765708B5BF
Round 11 6CA6CB20 FF3C485F 6D5560AF7CA5
Round 12 FF3C485F 22A5963B C2C1E96A4BF3
Round 13 22A5963B 387CCDAA 99C31397C91F
Round 14 387CCDAA BD2DD2AB 251B8BC717D0
Round 15 BD2DD2AB CF26B472 3330C5D9A36D
Round 16 CF26B472 19BA9212 181C5D75C66D

Cipher Text: C0B7A8D05F3A829C

Decryption

After initial permutation: 19BA9212CF26B472
After splitting: L0=19BA9212 R0=CF26B472
```

Experiment 5

AIM: C++ Program to Implement the RSA Algorithm.

```
#include<iostream>
#include<math.h>

using namespace std;

int gcd(int a, int h)
{
   int temp;
   while(1)
   {
      temp =
a%h;
if(temp==0)
return h;       a =
h;       h = temp;
   }
}

int main()
{

   double p = 3;    double q =
7;    double n=p*q;    double
count;    double totient = (p-
1)*(q-1);

   double e=2;

   while(e<totient){
count = gcd(e,totient);
if(count==1)       break;
else
      e++;
   }

   double d;
```

```
    double k = 2;

    d = (1 + (k*totient))/e;
double msg = 12;
double c = pow(msg,e);
double m = pow(c,d);
c=fmod(c,n);
    m=fmod(m,n);

    cout<<"Message data = "<<msg;
    cout<<"\n"<<"p = "<<p;
cout<<"\n"<<"q = "<<q;
cout<<"\n"<<"n = pq = "<<n;
cout<<"\n"<<"totient = "<<totient;
cout<<"\n"<<"e = "<<e;
cout<<"\n"<<"d = "<<d;
    cout<<"\n"<<"Encrypted data = "<<c;
    cout<<"\n"<<"Original Message sent = "<<m;

    return 0;
}
```

OUTPUT:

```
Message data = 12
p = 3
q = 7
n = pq = 21
totient = 12
e = 5
d = 5
Encrypted data = 3
Original Message sent = 12
--------------------------------
Process exited after 2.657 seconds with return value 0
Press any key to continue . . .
```

Experiment 6

AIM: C Program to Demonstrate Deffie Hellman key exchange algorithm.

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

long long int power(long long int a, long long int b, long long int P)
{  if (b == 1)
                return a;

        else return (((long long int)pow(a, b)) % P);
}


int main()
{ long long int P, G, x, a, y, b, ka, kb;


        P = 23;
        printf("The value of P : %lld\n", P);

        G = 9;
        printf("The value of G : %lld\n\n", G);
 a = 4;  printf("The private key a for Alice : %lld\n",
        a); x = power(G, a, P); // gets the generated
        key
          b = 3;  printf("The private key b for Bob :
%lld\n\n", b); y = power(G, b, P); // gets the
generated key


        ka = power(y, a, P);
        kb = power(x, b, P);
```

```
	printf("Secret key for the Alice is : %lld\n", ka);
	printf("Secret Key for the Bob is : %lld\n", kb);

	return 0;
}
```

OUTPUT:

```
The value of P : 23
The value of G : 9

The private key a for Alice : 4
The private key b for Bob : 3

Secret key for the Alice is : 9
Secret Key for the Bob is : 9

--------------------------------
Process exited after 2.595 seconds with return value 0
Press any key to continue . . .
```

Experiment 7

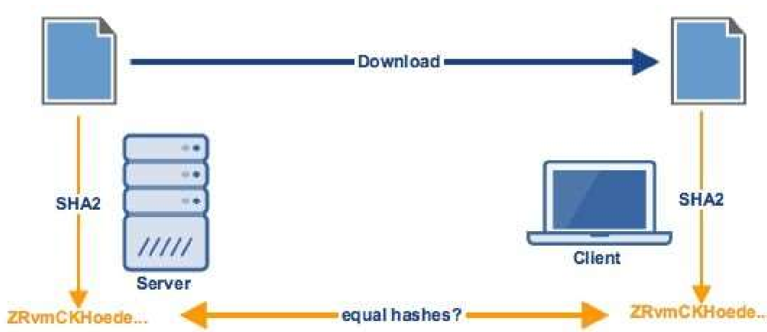
AIM: To Demonstrate the working of HMAC for secure file transfer.

HMAC algorithm stands for Hashed or Hash based Message Authentication Code. It is a result of work done on developing a MAC derived from cryptographic hash functions. HMAC is a great resistant towards cryptanalysis attacks as it uses the Hashing concept twice. HMAC consists of twin benefits of Hashing and MAC, and thus is more secure than any other authentication codes. RFC 2104 has issued HMAC, and HMAC has been made compulsory to implement in IP security. The FIPS 198 NIST standard has also issued HMAC.

How HMAC works

Let's first examine how a hash function could be used for conducting a data integrity check on a file transfer. Let's say a client application downloads a file from a remote server. It's assumed that the client and server have already agreed on a common hash function, say SHA2.

Before the server sends out the file, it first obtains a hash of that file using the SHA2 hash function. It then sends that hash along with the file itself. Upon receiving the two items (i.e. the downloaded file and the hash), the client obtains the SHA2 hash of the downloaded file and then compares it with the downloaded hash. If the two match, then that would mean the file was not tampered along the way.



If an attacker manages to intercept the downloaded file, alter the file's contents, and then forward the tampered file to the recipient, that malicious act won't go unnoticed.

That's because, once the client runs the tampered file through the agreed hash algorithm, the resulting hash won't match the downloaded hash. This will let the receiver know the file was tampered along the way.

While a hash function can establish data integrity, i.e. that the file or message wasn't altered along the way, it can't establish authenticity. How would the client know the message it received came from the legitimate source?

That's why secure file transfer protocols like FTPS, SFTP, and HTTPS use HMACs instead of just hash functions.When two parties exchange messages through those secure file transfer protocols, those messages will be accompanied by HMACs instead of plain hashes. An HMAC employs both a hash function and a shared secret key.

A shared secret key provides exchanging parties a way to establish the authenticity of the message. That is, it provides the two parties a way of verifying whether both the message and MAC (more specifically, an HMAC) they receive really came from the party they're supposed to be transacting with.

The secret key enables this capability because it's generated during key exchange, a preliminary process that requires the participation of the two parties. Only those two parties who participated in the key exchange would know what the shared secret key is. In turn, they would be the only ones who would be able to arrive at the same result if they compute the message's corresponding MAC using the shared secret key.