EXPERIMENT 1

Aim: To implement firewall through App to login into bank-site to implement E-commerce, debit card transaction through payment gateway.

Language used PHP, MySQL

Description

Firewall: is a dedicated hardware, software or a combination of both which inspects network traffic passing through it, and denies or permits passage based on a set of rules

Firewall Characteristics

→ A firewall defines a single chokepoint that keeps unauthorised users out of the protected network

→ A firewall provides a location for monitoring security related events. Audits and alarms can be implemented on the firewall system.

→ It is a convenient platform for several Internet functions that are not security related.

→ A firewall can serve as a platform for IPSec. Using tunnel mode capability, the firewall can be used to implement VPN.

Firewall limitations

-> The firewall can't protect against attacks that bypass the firewall
-> The firewall doesn't protect against internal threats
-> The firewall cant ~~protect at the application~~ against the transfer of virus-infected programs or files

Design Goals

-> All traffic from inside to outside and vice versa, must pass through firewall
-> Only authorized traffic, as defined by local security policy will be allowed to pass.
-> The firewall itself is immune to penetration. This implies the use of a trusted system with a secure operating system.

Methods of Control in firewall

User Control : only authorized users are having access to the other side of the firewall.

Access Control : The access over the firewall is restricted to certain services

Behaviour Control : An application, the allowed usage scenarios are known.

Direction Control : Different rules for traffic into the Intranet and outgoing traffic to Internet can be defined.

DELTA Pg No.
Date / /

Types of Firewall

Packet Filter Firewall : As each packet passes through the firewall it is examined and information contained in the header is compared to a pre-configured set of rules. An allow or deny entry is made based on the results of the comparison. Each packet is examined individually without regard to other packets that are part of the same connection.

Application Gateways: it acts as an intermediary between two endpoints. This packet screening method actually breaks the client/server model in that two connections are required : one from source to the gateway and one from gateway to destination. The endpoints can only communicate by going through the gateway.

Circuit level Gateway: it monitors TCP or UDP sessions. Once a session is established, it leaves the port open to allow all other packets belonging to that session to pass. The port is closed when the session is terminated.

CODE

a) For the database

```
define('DB_SERVER', 'localhost:3036');
define('DB_USERNAME', 'root');
define('DB_PASSWORD', 'rootpassword');
define('DB_DATABASE', 'database');
$db = mysqli_connect(DB_SERVER,DB_USERNAME,DB_PASSWORD,DB_DATABASE);
?>
<?php
include("config.php");
session_start();

if($_SERVER["REQUEST_METHOD"] == "POST") {
// username and password sent from form
    $result = mysqli_query($db,$sql);
    $row = mysqli_fetch_array($result,MYSQLI_ASSOC);
    $active = $row['active'];
    $count = mysqli_num_rows($result);
    // If result matched $myusername and $mypassword, table row must be 1 row
    if($count == 1) {
        session_register("myusername");
        $_SESSION['login_user'] = $myusername;
        header("location: welcome.php");
    }else {
        $error = "Your Login Name or Password is invalid";
    }
```

b) For the login page

```
<div align="center">
<h3>LOGIN FORM</h3>
    <form id="login-form" method="post" action="authen_login.php" >
        <table border="0.5" >
            <tr>
                <td><label for="user_id">User Name</label></td>
                <td><input type="text" name="user_id" id="user_id"></td>
            </tr>
            <tr>
                <td><label for="user_pass">Password</label></td>
                <td><input type="password" name="user_pass”
id="user_pass"></input></td>
            </tr>
            <tr>
                <td><input type="submit" value="Submit" />
            </tr>
        </table>
    </form>
</div>
```

c) Session:

```
   $ses_sql = mysqli_query($db,"select username from admin where username =
'$user_check' ");
   $row = mysqli_fetch_array($ses_sql,MYSQLI_ASSOC);
   $login_session = $row['username'];
   if(!isset($_SESSION['login_user'])){
      header("location:login.php");
   }
?>
```

OUTPUT

DATE / /
PAGE NO

EXPERIMENT 2

Aim Write a program to implement Transposition Cipher Language C++

Theory

In cryptography a transposition cipher is a method of encryption by which the position held by units of plain text are shifted according to a regular system, so that the cipher text constitutes of a permutation of plain text.

1) The message is written out in rows of a fixed length, and then read out again column by column, & the columns are chosen in some scrambled order.
2) Width of the rows & the permutation of the column are generally defined by a keyword.
3) For example, the word HACK is of length 4 and the permutation is defined by the alphabetical orders of the letters in the keyword. In this case, the order would be 3124
4) Any spare spaces are filled with nulls or left blanks or placed by a character
5) Finally the message is read off in columns, in the order specified by the keyword.

Decryption

1) The recepient has to work out the column lengths by dividing the message length by the key length.
2) Then, write the message out in columns again, then reorder the columns by informing the keyword.

Be Positive...

Sign.....

CODE

```
#include<bits/stdc++.h>
using namespace std;
string const key = "HACK";
map<int,int> keyMap;
void setPermutationOrder() {
    for(int i=0; i < key.length(); i++) {
        keyMap[key[i]] = i;
    }
}

string encryptMessage(string msg) {
    int row,col,j;
    string cipher = "";
    col = key.length();
    row = msg.length()/col;
    if (msg.length() % col)
        row += 1;
    char matrix[row][col];
    for (int i=0,k=0; i < row; i++) {
        for (int j=0; j<col; ) {
            if(msg[k] == '\0') {
                /* Adding the padding character '_' */
                matrix[i][j] = '_';
                j++;
            }
            if( isalpha(msg[k]) || msg[k]==' ') {
                /* Adding only space and alphabet into matrix*/
                matrix[i][j] = msg[k];
                j++;
            }
            k++;
        }
    }
    for (map<int,int>::iterator ii = keyMap.begin(); ii≠keyMap.end(); ++ii) {
        j=ii→second;
        for (int i=0; i<row; i++) {
            if( isalpha(matrix[i][j]) || matrix[i][j]==' ' || matrix[i][j]=='_')
                cipher += matrix[i][j];
        }
    }

    return cipher;
}

// Decryption
string decryptMessage(string cipher) {
    int col = key.length();

    int row = cipher.length()/col;
    char cipherMat[row][col];

    for (int j=0,k=0; j<col; j++)
        for (int i=0; i<row; i++)
            cipherMat[i][j] = cipher[k++];

    int index = 0;
    for( map<int,int>::iterator ii=keyMap.begin(); ii≠keyMap.end(); ++ii)
```

```
        ii->second = index++;

    char decCipher[row][col];
    map<int,int>::iterator ii=keyMap.begin();
    int k = 0;
    for (int l=0,j; key[l]!='\0'; k++) {
        j = keyMap[key[l++]];
        for (int i=0; i<row; i++) {
            decCipher[i][k]=cipherMat[i][j];
        }
    }

    string msg = "";
    for (int i=0; i<row; i++) {
        for(int j=0; j<col; j++) {
            if(decCipher[i][j] != '_')
                msg += decCipher[i][j];
        }
    }
    return msg;
}

int main(void) {
    string msg = "Work hard, play harder";
    setPermutationOrder();
    string cipher = encryptMessage(msg);
    cout << "Encrypted Message: " << cipher << endl;
    cout << "Decrypted Message: " << decryptMessage(cipher) << endl;
    return 0;
}
```

OUTPUT

```
kunal@Kunal-Notebook:~/Documents/CollegeSem7/IS
$ ./a.out
Encrypted Message: oh yr_rap d_W daarkrlhe_
Decrypted Message: Work hard play harder
```

DATE / /
PAGE NO.

EXPERIMENT 3

Aim: To implement RSA algorithm Language Used Java, JVM

Description: RSA algorithm is an asymmetric cryptography algorithm. Asymmetric means that it works on two different keys i.e Public Key and Private Key. As name suggests Public key is given to everyone and private key is kept private.

Algorithm:

1 -> Choose 2 prime numbers p & q.
2 -> Calculate $n = p * q$
3 -> Calculate $\phi(n) = (p-1) \times (q-1)$
4 -> Choose e such that $gcd(e, \phi(n)) = 1$
5 -> Calculate d such that $(k * \phi(n) + 1)/e$ where k is an integer
6 -> Public Key [e,n] Private key [d, n]
7 -> Cipher text $C = P^e \bmod n$, where P = plain text
8 -> For decryption $D = C \bmod n$ will give back the plain text.

An Example of Asymmetric Cryptography

1) A client sends its public key to the sender and requests for some data
2) The sender encrypts the data using clients public key & sends the encrypted data.
3) Client receives this data & decrypts it.

CODE

```
import java.io.DataInputStream;
import java.io.IOException;
import java.math.BigInteger;
import java.util.Random;

public class RSA{
    private BigInteger P;
    private BigInteger Q;
    private BigInteger N;
    private BigInteger M;
    private BigInteger e;
    private BigInteger d;
    private int maxLength = 104;
    private Random R;

    public RSA(){
        R = new Random();
        P = BigInteger.probablePrime(maxLength, R);
        Q = BigInteger.probablePrime(maxLength, R);
        N = P.multiply(Q);
        M = P.subtract(BigInteger.ONE).multiply(  Q.subtract(BigInteger.ONE));
        e = BigInteger.probablePrime(maxLength / 2, R);
        while (M.gcd(e).compareTo(BigInteger.ONE) > 0 && e.compareTo(M) < 0)
        {
            e.add(BigInteger.ONE);
        }
        d = e.modInverse(M);
        System.out.println("P: "+P);
        System.out.println("Q: "+Q);
        System.out.println("E: "+e);
        System.out.println("D: "+d);
    }

    public static void main (String [] arguments) throws IOException{
        RSA rsa = new RSA();
        DataInputStream input = new DataInputStream(System.in);
        String inputString;
        System.out.println("Enter message you wish to send.");
        inputString = input.readLine();
        System.out.println("Encrypting the message: " + inputString);
        System.out.println("The message in bytes is:: "
                + bToS(inputString.getBytes()));
        // encryption
        byte[] cipher = rsa.encryptMessage(inputString.getBytes());
        // decryption
        byte[] plain = rsa.decryptMessage(cipher);
        System.out.println("Decrypting Bytes: " + bToS(plain));
        System.out.println("Plain message is: " + new String(plain));
    }

    private static String bToS(byte[] cipher){
        String temp = "";
        for (byte b : cipher){
            temp += Byte.toString(b);
        }
        return temp;
    }

    // Encrypting the message
    public byte[] encryptMessage(byte[] message){
```

```
        return (new BigInteger(message)).modPow(e, N).toByteArray();
    }

    // Decrypting the message
    public byte[] decryptMessage(byte[] message){
        return (new BigInteger(message)).modPow(d, N).toByteArray();
    }
}
```

OUTPUT

```
/home/kunal/Apps/idea-IC-192.6262.58/jbr/bin/java -javaagent:/home/kunal/Apps/idea-
P: 184969215489914250060562574639O3
Q: 17925161543951720519378667515459
E: 2443566578507711
D: 561611421109645700107469826139843603104366573868623768159104ll
Enter message you wish to send.
Hello help!
Encrypting the message: Hello help!
The message in bytes is:: 72101108108111321041011081123З
Decrypting Bytes: 72101108108111321041011081123З
Plain message is: Hello help!

Process finished with exit code 0
```

DATE / /
PAGE NO

EXPERIMENT 4

Aim: Write a program to implement DES algorithm

Language Used:

Theory:

Data Encryption Standard (DES) is a block cipher algorithm that takes plain text in blocks of 64 bits and converts them to cipher text using keys of 48 bits.

→ Its a symmetric key algorithm, meaning, the same key is used for encrypting and decrypting data.

→ Generating keys: There are 16 rounds of encryption in this algorithm, of a different key is used for each round.

→ Bits are labelled from 1 to 64 starting from the most significant bit and going to the least significant bit.

Result: Successfully implemented DES algorithm in Java

Be Positive....

Sign.....

```
import java.util.*;

class DES {
    private static final byte[] 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
    };

    private static final byte[] 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
    };

    private static final byte[] 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
    };

    private static final byte[] rotations = {
            1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1
    };

    private static final byte[] E = {
            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
    };

    private static final byte[][] S = {{
            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
    }};

    private static final byte[] 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
    };

    private static final byte[] FP = {
        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
    };


    private static int[] C = new int[28];
    private static int[] D = new int[28];

    private static int[][] subkey = new int[16][48];

    public static void main(String args[]) {
       System.out.println("Enter the input as a 16 character hexadecimal value:");
       String input = new Scanner(System.in).nextLine();
       int inputBits[] = new int[64];

       for (int i = 0; i < 16; i++) {
          String s = Integer.toBinaryString(Integer.parseInt(input.charAt(i) + "", 16));

          while (s.length() < 4) {
             s = "0" + s;
          }
          for (int j = 0; j < 4; j++) {
             inputBits[(4 * i) + j] = Integer.parseInt(s.charAt(j) + "");
          }
       }

       System.out.println("Enter the key as a 16 character hexadecimal value:");
       String key = new Scanner(System.in).nextLine();
       int keyBits[] = new int[64];
       for (int i = 0; i < 16; i++) {
          String s = Integer.toBinaryString(Integer.parseInt(key.charAt(i) + "", 16));
          while (s.length() < 4) {
             s = "0" + s;
          }
          for (int j = 0; j < 4; j++) {
             keyBits[(4 * i) + j] = Integer.parseInt(s.charAt(j) + "");
          }
       }

       System.out.println("\n+++ ENCRYPTION +++");
       int outputBits[] = permute(inputBits, keyBits, false);
       System.out.println("\n+++ DECRYPTION +++");
       permute(outputBits, keyBits, true);
    }
```

```
private static int[] permute(int[] inputBits, int[] keyBits, boolean isDecrypt) {
    int newBits[] = new int[inputBits.length];
    for (int i = 0; i < inputBits.length; i++) {
        newBits[i] = inputBits[IP[i] - 1];
    }

    int L[] = new int[32];
    int R[] = new int[32];
    int i;

    for (i = 0; i < 28; i++) {
        C[i] = keyBits[PC1[i] - 1];
    }
    for (; i < 56; i++) {
        D[i - 28] = keyBits[PC1[i] - 1];
    }

    System.arraycopy(newBits, 0, L, 0, 32);
    System.arraycopy(newBits, 32, R, 0, 32);
    System.out.print("\nL0 = ");
    displayBits(L);
    System.out.print("R0 = ");
    displayBits(R);
    for (int n = 0; n < 16; n++) {
        System.out.println("\n-------------");
        System.out.println("Round " + (n + 1) + ":");

        int newR[] = new int[0];
        if (isDecrypt) {
            newR = fiestel(R, subkey[15 - n]);
            System.out.print("Round key = ");
            displayBits(subkey[15 - n]);
        } else {
            newR = fiestel(R, KS(n, keyBits));
            System.out.print("Round key = ");
            displayBits(subkey[n]);
        }

        int newL[] = xor(L, newR);
        L = R;
        R = newL;
        System.out.print("L = ");
        displayBits(L);
        System.out.print("R = ");
        displayBits(R);
    }

    int output[] = new int[64];
    System.arraycopy(R, 0, output, 0, 32);
    System.arraycopy(L, 0, output, 32, 32);
    int finalOutput[] = new int[64];

    for (i = 0; i < 64; i++) {
        finalOutput[i] = output[FP[i] - 1];
    }

    String hex = new String();
    for (i = 0; i < 16; i++) {
        String bin = new String();
        for (int j = 0; j < 4; j++) {
            bin += finalOutput[(4 * i) + j];
        }
        int decimal = Integer.parseInt(bin, 2);
        hex += Integer.toHexString(decimal);
    }
    if (isDecrypt) {
        System.out.print("Decrypted text: ");

    } else {
        System.out.print("Encrypted text: ");
    }
    System.out.println(hex.toUpperCase());
    return finalOutput;
}

private static int[] KS(int round, int[] key) {
```

```
    int C1[] = new int[28];
    int D1[] = new int[28];

    int rotationTimes = (int) rotations[round];

    C1 = leftShift(C, rotationTimes);
    D1 = leftShift(D, rotationTimes);
    int CnDn[] = new int[56];
    System.arraycopy(C1, 0, CnDn, 0, 28);
    System.arraycopy(D1, 0, CnDn, 28, 28);

    int Kn[] = new int[48];
    for (int i = 0; i < Kn.length; i++) {
        Kn[i] = CnDn[PC2[i] - 1];
    }

    subkey[round] = Kn;
    C = C1;
    D = D1;
    return Kn;
}

private static int[] fiestel(int[] R, int[] roundKey) {

    int expandedR[] = new int[48];
    for (int i = 0; i < 48; i++) {
        expandedR[i] = R[E[i] - 1];
    }
    int temp[] = xor(expandedR, roundKey);

    int output[] = sBlock(temp);
    return output;
}

private static int[] xor(int[] a, int[] b) {
    int answer[] = new int[a.length];
    for (int i = 0; i < a.length; i++) {
        answer[i] = a[i] ^ b[i];
    }
    return answer;
}

private static int[] sBlock(int[] bits) {
    int output[] = new int[32];
    for (int i = 0; i < 8; i++) {
        int row[] = new int[2];
        row[0] = bits[6 * i];
        row[1] = bits[(6 * i) + 5];
        String sRow = row[0] + "" + row[1];
        int column[] = new int[4];
        column[0] = bits[(6 * i) + 1];
        column[1] = bits[(6 * i) + 2];
        column[2] = bits[(6 * i) + 3];
        column[3] = bits[(6 * i) + 4];
        String sColumn = column[0] + "" + column[1] + "" + column[2] + "" + column[3];
        int iRow = Integer.parseInt(sRow, 2);
        int iColumn = Integer.parseInt(sColumn, 2);
        int x = S[i][(iRow * 16) + iColumn];
        String s = Integer.toBinaryString(x);
        while (s.length() < 4) {
            s = "0" + s;
        }
        for (int j = 0; j < 4; j++) {
            output[(i * 4) + j] = Integer.parseInt(s.charAt(j) + "");
        }
    }

    int finalOutput[] = new int[32];
    for (int i = 0; i < 32; i++) {
        finalOutput[i] = output[P[i] - 1];
    }
    return finalOutput;
}

private static int[] leftShift(int[] bits, int n) {
    int answer[] = new int[bits.length];
    System.arraycopy(bits, 0, answer, 0, bits.length);
```

```
        for (int i = 0; i < n; i++) {
            int temp = answer[0];
            for (int j = 0; j < bits.length - 1; j++) {
                answer[j] = answer[j + 1];
            }
            answer[bits.length - 1] = temp;
        }
        return answer;
    }

    private static void displayBits(int[] bits) {
        for (int i = 0; i < bits.length; i += 4) {
            String output = new String();
            for (int j = 0; j < 4; j++) {
                output += bits[i + j];
            }
            System.out.print(Integer.toHexString(Integer.parseInt(output, 2)));
        }
        System.out.println();
    }
}
```

OUTPUT

```
------------------------------------------------------------------------------------------------------------
Enter the input as a 16 character hexadecimal value:
1234567890123456
Enter the key as a 16 character hexadecimal value:
0987654321098765

+++ ENCRYPTION +++

L0 = 8cffc600
R0 = 104a08a5

-------------
Round 1:
Round key = 0866629080b7
L = 104a08a5
R = cba11260

-------------
Round 2:
Round key = 258214a1610c
L = cba11260
R = d3511712

-------------
Round 3:
Round key = 0610a2203282
L = d3511712
R = 3ad7eb5b

-------------
Round 4:
Round key = ba0070f40027
L = 3ad7eb5b
R = 5fc1d4fe

-------------
Round 5:
Round key = 8c4208060aca
L = 5fc1d4fe
R = b6231625

-------------
Round 6:
Round key = 02131c14b151
L = b6231625
R = 6ddc4631

-------------
Round 7:
Round key = 0c1841238460
L = 6ddc4631
```

```
R = b6b004ee

-------------
Round 8:
Round key = 03686848ad02
L = b6b004ee
R = 5b72a63d

-------------
Round 9:
Round key = 04809d149129
L = 5b72a63d
R = 52bcded2

-------------
Round 10:
Round key = 170006821c60
L = 52bcded2
R = 5327d773

-------------
Round 11:
Round key = 2a08a048ab30
L = 5327d773
R = 09977b2f

-------------
Round 12:
Round key = 98202c314c18
L = 09977b2f
R = c41f470b

-------------
Round 13:
Round key = 800618491012
L = c41f470b
R = 7587f699

-------------
Round 14:
Round key = 441a2485602c
L = 7587f699
R = 4299cabd

-------------
Round 15:
Round key = 82b840201ac4
L = 4299cabd
R = 7ba38418

-------------
Round 16:
Round key = c000175a2010
L = 7ba38418
R = f313420b
Encrypted text: F1F50883D2E0C468

+++ DECRYPTION +++

L0 = f313420b
R0 = 7ba38418

-------------
Round 1:
Round key = c000175a2010
L = 7ba38418
R = 4299cabd

-------------
Round 2:
Round key = 82b840201ac4
L = 4299cabd
R = 7587f699

-------------
```

```
Round 3:
Round key = 441a2485602c
L = 7587f699
R = c41f470b

-------------
Round 4:
Round key = 800618491012
L = c41f470b
R = 09977b2f

-------------
Round 5:
Round key = 98202c314c18
L = 09977b2f
R = 5327d773

-------------
Round 6:
Round key = 2a08a048ab30
L = 5327d773
R = 52bcded2

-------------
Round 7:
Round key = 170006821c60
L = 52bcded2
R = 5b72a63d

-------------
Round 8:
Round key = 04809d149129
L = 5b72a63d
R = b6b004ee

-------------
Round 9:
Round key = 03686848ad02
L = b6b004ee
R = 6ddc4631

-------------
Round 10:
Round key = 0c1841238460
L = 6ddc4631
R = b6231625

-------------
Round 11:
Round key = 02131c14b151
L = b6231625
R = 5fc1d4fe

-------------
Round 12:
Round key = 8c4208060aca
L = 5fc1d4fe
R = 3ad7eb5b

-------------
Round 13:
Round key = ba0070f40027
L = 3ad7eb5b
R = d3511712

-------------
Round 14:
Round key = 0610a2203282
L = d3511712
R = cba11260

-------------
Round 15:
Round key = 258214a1610c
L = cba11260
```

```
R = 104a08a5

-------------
Round 16:
Round key = 0866629080b7
L = 104a08a5
R = 8cffc600
Decrypted text: 1234567890123456

Process finished with exit code 0
```

DATE: / /
PAGE NO:

EXPERIMENT 5

Aim: To implement and study about Diffie Hellman algorithm.

Theory: It is used to established a shared secret that can be used for secret communications while exchanging data over public network using elliptic curve to generate points and get secret key using the parameters.

Example:

	USER 1 / Alice	USER 2 / Bob
PUBLIC KEYS	P, G	P, G
Private key Selected	a	b
Key Generated	$x = G^a \bmod P$	$y = G^b \bmod P$
Key Received	y	x
Generated Secret key	$k_a = y^a \bmod P$	$k_b = x^b \bmod P$

$k_a = k_b$ which can be used for encryption.

In Static - static mode, both Alice and Bob retain their private / public keys over multiple communications. Therefore the resulting shared secret will be the same everytime. In ephemerical - static mode one party will generate a new private / public key every time, thus a new shared secret will be generated.

Be Positive...

Sign.....

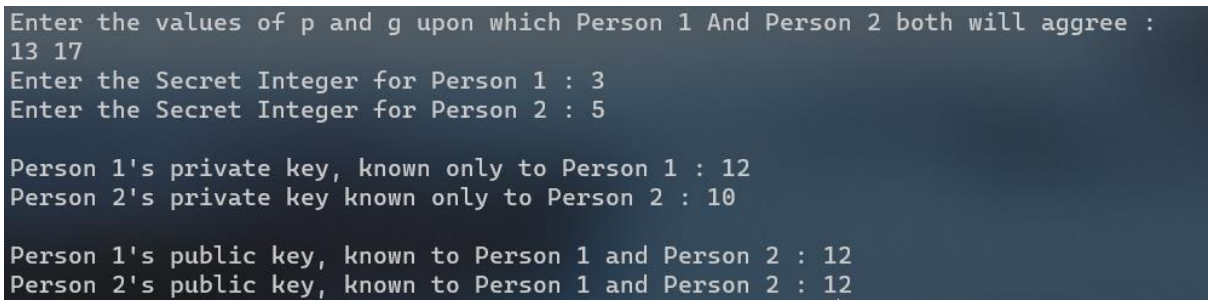
Experiment 5

Aim: To study and implement Diffie Helman

CODE:

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

using namespace std;

class DiffieHellman
{
	public:
		long long int p,g,x,a,y,b,A,B;
		DiffieHellman(long long int p1,long long int g1,long long
int x1,long long int y1)
		{
			p = p1;
			g = g1;
			x = x1;
			y = y1;
			a=power(g,x,p);
			b=power(g,y,p);
			A = power(b,x,p);
			B = power(a,y,p);
		}
		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,A,B;

	cout<<"Enter the values of p and g upon which Person 1 And Person 2
both will aggree : "<<endl;
	cin>>p>>g;

	cout<<"Enter the Secret Integer for Person 1 : ";
	cin>>x;
	cout<<"Enter the Secret Integer for Person 2 : ";
	cin>>y;
	cout<<endl;
	DiffieHellman dh(p,g,x,y);

	cout<<"Person 1's private key, known only to Person 1 :
"<<dh.a<<endl;
	cout<<"Person 2's private key known only to Person 2 :
"<<dh.b<<endl;
	cout<<endl;
	cout<<"Person 1's public key, known to Person 1 and Person 2 :
"<<dh.A<<endl;
	cout<<"Person 2's public key, known to Person 1 and Person 2 :
"<<dh.B<<endl;
	return 0;
}
```

OUTPUT


Enter the values of p and g upon which Person 1 And Person 2 both will aggree :
13 17
Enter the Secret Integer for Person 1 : 3
Enter the Secret Integer for Person 2 : 5
Person 1's private key, known only to Person 1 : 12
Person 2's private key known only to Person 2 : 10
Person 1's public key, known to Person 1 and Person 2 : 12
Person 2's public key, known to Person 1 and Person 2 : 12

Experiment 6

Aim: To study anyone simulation tool based on parameters of information security.

Theory

Nmap (Network Mapper) is a free and open-source network scanner created by Gordon Lyon (also known by his pseudonym Fyodor Vaskovich). Nmap is used to discover hosts and services on a computer network by sending packets and analyzing the responses.

Nmap provides several 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.

Nmap features include:

- Host discovery – Identifying hosts on a network. For example, listing the hosts that respond to TCP and/or ICMP requests or have a particular port open.
- Port scanning – Enumerating the open ports on target hosts.
- Version detection – Interrogating network services on remote devices to determine application name and version number.
- OS detection – Determining the operating system and hardware characteristics of network devices.
- Scriptable interaction with the target – using Nmap Scripting Engine (NSE) and Lua programming language.

Typical uses of Nmap:

- Auditing the security of a device or firewall by identifying the network connections which can be made to, or through it.
- Identifying open ports on a target host in preparation for auditing.
- Network inventory, network mapping, maintenance, and asset management.
- Auditing the security of a network by identifying new servers.
- Generating traffic to hosts on a network, response analysis and response time measurement.
- Finding and exploiting vulnerabilities in a network.
- DNS queries and subdomain search

OUTPUT

```
kunal@MSI:~$ nmap -sP google.com
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-17 20:31 IST
Nmap scan report for google.com (172.217.26.238)
Host is up (0.043s latency).
Other addresses for google.com (not scanned): 2404:6800:4009:81a::200e
rDNS record for 172.217.26.238: bom05s09-in-f14.1e100.net
Nmap done: 1 IP address (1 host up) scanned in 0.17 seconds
kunal@MSI:~$ nmap -p 1-65535 -v 172.217.26.238
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-17 20:32 IST
Initiating Ping Scan at 20:32
Scanning 172.217.26.238 [2 ports]
Completed Ping Scan at 20:32, 0.04s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 20:32
Completed Parallel DNS resolution of 1 host. at 20:32, 0.01s elapsed
Initiating Connect Scan at 20:32
Scanning bom05s09-in-f14.1e100.net (172.217.26.238) [65535 ports]
Discovered open port 443/tcp on 172.217.26.238
Discovered open port 80/tcp on 172.217.26.238
Connect Scan Timing: About 16.00% done; ETC: 20:35 (0:02:43 remaining)
Connect Scan Timing: About 38.09% done; ETC: 20:35 (0:01:39 remaining)
Connect Scan Timing: About 62.20% done; ETC: 20:34 (0:00:55 remaining)
Completed Connect Scan at 20:34, 135.30s elapsed (65535 total ports)
Nmap scan report for bom05s09-in-f14.1e100.net (172.217.26.238)
Host is up (0.056s latency).
Not shown: 65533 filtered ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 135.40 seconds
kunal@MSI:~$
```

Experiment 7

Aim: To study Intrusion Detection System

Theory:

An **Intrusion Detection System (IDS)** is a system that monitors **network traffic** for suspicious activity and issues alerts when such activity is discovered. It is a software application that scans a network or a system for harmful activity or policy breaching. Any malicious venture or violation is normally reported either to an administrator or collected centrally using a security information and event management (SIEM) system. A SIEM system integrates outputs from multiple sources and uses alarm filtering techniques to differentiate malicious activity from false alarms.

Although intrusion detection systems monitor networks for potentially malicious activity, they are also disposed to false alarms. Hence, organizations need to fine-tune their IDS products when they first install them. It means properly setting up the intrusion detection systems to recognize what normal traffic on the network looks like as compared to malicious activity.

Intrusion prevention systems also monitor network packets inbound the system to check the malicious activities involved in it and at once sends the warning notifications.

Classification of Intrusion Detection System:

IDS are classified into 5 types:

1. **Network Intrusion Detection System (NIDS):**
   Network intrusion detection systems (NIDS) are set up at a planned point within the network to examine traffic from all devices on the network. It performs an observation of passing traffic on the entire subnet and matches the traffic that is passed on the subnets to the collection of known attacks. Once an attack is identified or abnormal behavior is observed, the alert can be sent to the administrator. An example of an NIDS is installing it on the subnet where firewalls are located in order to see if someone is trying crack the firewall.
2. **Host Intrusion Detection System (HIDS):**
   Host intrusion detection systems (HIDS) run on independent hosts or devices on the network. A HIDS monitors the incoming and outgoing packets from the device only and will alert the administrator if suspicious or malicious activity is detected. It takes a snapshot of existing system files and compares it with the previous snapshot. If the analytical system files were edited or deleted, an alert is sent to the administrator to investigate. An example of HIDS usage can be seen on mission critical machines, which are not expected to change their layout.
3. **Protocol-based Intrusion Detection System (PIDS):**
   Protocol-based intrusion detection system (PIDS) comprises of a system or agent that would consistently resides at the front end of a server, controlling and interpreting the protocol between a user/device and the server. It is trying to secure the web server by regularly monitoring the HTTPS protocol stream and accept the related HTTP protocol. As HTTPS is un-encrypted and before instantly entering its web presentation layer then this system would need to reside in this interface, between to use the HTTPS.
4. **Application Protocol-based Intrusion Detection System (APIDS):**
   Application Protocol-based Intrusion Detection System (APIDS) is a system or agent that generally resides within a group of servers. It identifies the intrusions by monitoring and interpreting the communication on application specific protocols. For

example, this would monitor the SQL protocol explicit to the middleware as it transacts with the database in the web server.

5. **Hybrid Intrusion Detection System :**
   Hybrid intrusion detection system is made by the combination of two or more approaches of the intrusion detection system. In the hybrid intrusion detection system, host agent or system data is combined with network information to develop a complete view of the network system. Hybrid intrusion detection system is more effective in comparison to the other intrusion detection system. Prelude is an example of Hybrid IDS.

Example for tools

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. Snort can be downloaded and configured for personal and business use alike.

DATE: / /

PAGE NO:

EXPERIMENT 8

Aim: To study and implement Vernam Cipher.

Theory: It is a method of encrypting alphabet text. It is one of the transposition technique for converting a plain text to cipher text.

Here, we assign a number to each character of plain text like (a=1, b=1, ... z=25).

Key Generation:
We take a key to encrypt the plain text which length should be equal to the length of plain text.

Encryption Algorithm - (for decryption use reverse of this)

1. Assign a number to each character of plain text and key according to alphabetical order
2. Add both numbers corresponding to each other.
3. Subtract number from 26 if number added greater than 26.
4. Get cipher text by getting corresponding character from the number.

Experiment 8

AIM: To study and implement Vernam Cipher.

CODE:

```
#include<iostream>
using namespace std;
class vernam{
	public:
		string s,k;
		char enc[1000],dec[1000];
		vernam(string a, string b){
			s = a;
			k = b;
		}
		void encrypt(){
			int i,j=0;
			for(i=0;i<s.size();i++){
				enc[i] = s[i]^k[j];
				j++;
				if(j>=k.size()){
					j =0;
				 }
			}
		}
		void decrypt(){
			int i,j=0;
			for(i=0;i<s.size();i++)
			{
				dec[i] = enc[i]^k[j];
				j++;
				if(j>=k.size())
				{
					j =0;
				}
			}
		}
		void printenc(){
			int i;
			char c;
			for(i=0;i<s.size();i++)
			{
				c = enc[i]%26 + 48;
				cout<<c;
			}
			cout<<endl;
		}
		void printdec(){
			int i;
			for(i=0;i<s.size();i++)
			{
				cout<<dec[i];
			}
			cout<<endl;
		}
};


int main(){
	string s,k;
	cout<<"Enter the Plain Text Message"<<endl;
	getline(cin,s);
	cout<<"Enter the Key"<<endl;
	getline(cin,k);
	vernam v(s,k);
	v.encrypt();
	cout<<"Encrypted Message : ";
	v.printenc();
```

```
        cout<<endl;
        v.decrypt();
        cout<<"Decrypted Message : ";
        v.printdec();
        return 0;
}
```

OUTPUT