|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ex. No** | **Date** | **Title** | **Page No** | **Signature** |
| 1 |  | Implement substitution and translation techniques |  |  |
| 2 |  | Implement the Hashing Algorithms |  |  |
| 3 |  | Write a program to implement a set of values to combine the control of Bell lapadula with the integrating controls of the Biba model |  |  |
| 4 |  | Installation of rootkits and study about various options |  |  |
| 5 |  | Implement hacking windows – windows login and password |  |  |
| 6 |  | Implement hacking windows – accessing restricted drivers |  |  |
| 7 |  | Implement cross site scripting and implement XSS |  |  |
| 8 |  | Implement SQL injection attack |  |  |
| 9 |  | Implement Buffer overflow attack |  |  |
| 10 |  | Understanding malware working principles, detection and prevention |  |  |
| 11 |  | Setup honeypot and monitor the honeypot network |  |  |
| 12 |  | Demonstrate intrusion detection system using any tool |  |  |

**INDEX**

**Ex.No. 1**

**Implement the following substitution and translation techniques of the following Caesar Cipher, Playfair Cipher, Hill Cipher, ABCD Vigenere Cipher, Railfence Row and Column transformation**

**CASESAR CIPHER**

**Aim:**

To implement Caesar Substitution Cipher.

**Algorithm:**

Caesar Cipher is one of the earliest and the simplest method of substitution technique. Each letter of a given text is replaced by a letter some fixed number of positions down the alphabet.

**Input:**

1. A string of lower case letters, called Plain Text.
2. An integer denoting the required shift, called Key.

**Steps:**

1. Key = key % 26.
2. Traverse the given text one character at a time.
3. For each character, transform the given character as per the rule, depending on whether encryption or decryption is done.
   1. **Rule for encryption**

Cipher text = (Plain text + Key) % 26.

* 1. **Rule for decryption**

Cipher text = (Plain text + Key) % 26.

1. Return the generated string.

**Program:**

import java.io.\*;

public class Caesar{

private static BufferedReader br = null;

public static void main(String[] args) throws Exception{

br = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter the text to encrypt...... enter q to quit");

String text = br.readLine();

System.out.println("Enter the shift");

int shift = Integer.parseInt(br.readLine());

while(!text.equals("q")){

System.out.println("\*\*\*\*\*\*ENCRYPTION\*\*\*\*\*\*");

String encrypt = "";

shift %= 26;

for(int i=0; i<text.length(); i++){

int s = text.charAt(i) + shift;

if(text.charAt(i) >= 'A' && text.charAt(i) <= 'Z')

if(s > 'Z') s -= 26;

else if(text.charAt(i) >= 'a' && text.charAt(i) <= 'z')

if(s > 'z') s -= 26;

encrypt += (char)s;}

System.out.println("Encrypted text is : " + encrypt);

System.out.println("\*\*\*\*\*\*DECRYPTION\*\*\*\*\*\*");

String decrypt = "";

for(int i=0; i<encrypt.length(); i++){

int s = encrypt.charAt(i) - shift;

if(encrypt.charAt(i) >= 'A' && encrypt.charAt(i) <= 'Z')

if(s < 'A') s += 26;

else if(encrypt.charAt(i) >= 'a' && encrypt.charAt(i) <= 'z')

if(s < 'a') s += 26;

decrypt += (char)s;

}

System.out.println("Decrypted text is : " + decrypt + "\n");

System.out.println("Enter the text to encrypt...... enter q to quit");

text = br.readLine();

if(text.equals("q")) break;

System.out.println("Enter the shift");

shift = Integer.parseInt(br.readLine());

}

}}

**Output:**

Enter the text to encrypt...... enter q to quit

Gomathy

Enter the shift

3

\*\*\*\*\*\*ENCRYPTION\*\*\*\*\*\*

Encrypted text is : Jrpdwkb

\*\*\*\*\*\*DECRYPTION\*\*\*\*\*\*

Decrypted text is : Gomathy

**PLAYFAIR CIPHER**

**Aim:**

To implement Playfair Cipher.

**Algorithm:**

1. The key for playfair cipher is generally a word.
2. Any sequence of 25 letters can be used as a key.
3. Remove any punctuations/characters not part of grid.
4. Split the plain text into pairs.
5. Identify any double letter in the plain text and insert ‘z’
6. If there is an odd number of letters add ‘z’.
7. Break the plain text into pairs of letters.
8. Now apply the following rules.
   1. If both the letters are in the same column take letter below each one.
   2. If both the letters are in the same row take letter right to each one.
   3. If neither of the rules apply, form a rectangle with two letters and take letters of horizontal opposite.

**Program:**

import java.io.\*;

import java.util.\*;

public class PlayFair{

private static BufferedReader br = null;

public static char[][] matCreate(String key){

HashMap<Character, Integer> m = new HashMap<Character, Integer>();

char[][] mat = new char[5][5];

for(char i='a'; i<='z'; i++)

m.put(i, 1);

m.put('j', 0);

int row = 0, col = 0;

for(int i=0; i<key.length(); i++){

if(m.get(key.charAt(i)) == null) continue;

if(m.get(key.charAt(i)) == 1){

m.put(key.charAt(i), 0);

if(col == 5){

col = 0;

row++;

}

mat[row][col] = key.charAt(i);

col++;

}

}

char alpha = 'a';

if(col < 5){

for(int j=col; j<5; j++){

if(m.get(alpha) == 1){

mat[row][j] = alpha;

m.put(alpha, 0);

}

alpha++;

}

row++;

}

for(int i=row; i<5; i++){

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

if(m.get(alpha) == 1){

mat[i][j] = alpha;

m.put(alpha, 0);

}

else{

while(m.get(alpha) == 0 && alpha <= 'z'){

alpha++;

}

mat[i][j] = alpha;

m.put(alpha, 0);

}

alpha++;

}

}

return mat;

}

public static int[] find(char c, char[][] mat){

int[] a = new int[2];

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

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

if(mat[i][j] == c){

a[0] = i;

a[1] = j;

return a;

}

}

}

return a;

}

public static String encryptText(String t, char[][] mat){

int[] a = find(t.charAt(0), mat);

int[] b = find(t.charAt(1), mat);

String s = "";

if(a[0] == b[0]){

s += mat[a[0]][(a[1]+1)%5];

s += mat[b[0]][(b[1]+1)%5];

}

else if(a[1] == b[1]){

s += mat[(a[0]+1)%5][a[1]];

s += mat[(b[0]+1)%5][b[1]];

}

else{

s += mat[a[0]][b[1]];

s += mat[b[0]][a[1]];

}

return s;

}

public static String decryptText(String t, char[][] mat){

int[] a = find(t.charAt(0), mat);

int[] b = find(t.charAt(1), mat);

String s = "";

if(a[0] == b[0]){

int c = a[1]-1;

if(c < 0)

c += 5;

s += mat[a[0]][c%5];

c = b[1]-1;

if(c < 0)

c += 5;

s += mat[b[0]][c%5];

}

else if(a[1] == b[1]){

int r = a[0]-1;

if(r < 0)

r += 5;

s += mat[r%5][a[1]];

r = b[0]-1;

if(r < 0)

r += 5;

s += mat[r%5][b[1]];

}

else{

s += mat[a[0]][b[1]];

s += mat[b[0]][a[1]];

}

return s;

}

public static void main(String[] args) throws Exception{

br = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter the text to encrypt...... enter q to quit");

String text = br.readLine();

System.out.println("Enter the key");

String key = br.readLine();

while(!text.equals("q")){

System.out.println("\*\*\*\*\*\*ENCRYPTION\*\*\*\*\*\*");

char[][] mat = matCreate(key);

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

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

System.out.print(mat[i][j] + " ");

System.out.println();

}

if(text.length()%2 == 1) text += "z";

String[] tokens = new String[text.length()/2];

for(int i=0, j=0; i<text.length(); i++, j++){

if(text.charAt(i) == ' ') i++;

tokens[j] = String.valueOf(text.charAt(i));

i++;

if(text.charAt(i) == ' ') i++;

tokens[j] += String.valueOf(text.charAt(i));

}

String encrypt = "";

for(int i=0; i<text.length()/2; i++)

encrypt += encryptText(tokens[i], mat);

System.out.println("Encrypted text = " + encrypt);

System.out.println("\*\*\*\*\*\*DECRYPTION\*\*\*\*\*\*");

tokens = new String[encrypt.length()/2];

for(int i=0,j=0; i<encrypt.length(); i++, j++){

tokens[j] = String.valueOf(encrypt.charAt(i));

i++;

tokens[j] += String.valueOf(encrypt.charAt(i));

}

String decrypt = "";

for(int i=0; i<encrypt.length()/2; i++)

decrypt += decryptText(tokens[i], mat);

System.out.println("Decrypted text = " + decrypt + "\n");

System.out.println("Enter the text to encrypt...... enter q to quit");

text = br.readLine();

if(text.equals("q"))

break;

System.out.println("Enter the shift");

key = br.readLine();

}

}

}

**Output:**

Enter the text to encrypt...... enter q to quit

gomathy

Enter the key

goms

\*\*\*\*\*\*ENCRYPTION\*\*\*\*\*\*

g o m s a

b c d e f

h i k l n

p q r t u

v w x y z

Encrypted text = omsgplzv

\*\*\*\*\*\*DECRYPTION\*\*\*\*\*\*

Decrypted text = gomathyz

**HILL CIPHER**

**Aim:**

To implement Hill cipher.

**Algorithm:**

1. Generate the key matrix, where each character is represented by its equivalent number.
2. Turn the plain text into digraphs and generate the column vector.
3. Perform matrix multiplication modulo 26.
4. These letters are then converted back to produce cipher text.

**Program:**

import java.io.\*;

import java.util.\*;

public class Hill{

private static BufferedReader br = null;

public static String mult(double[][] mat, double[] a, int n){

String encrypt = "";

for(int i=0; i<n; i++){

int sum = 0;

for(int j=0; j<n; j++){

sum += a[j] \* mat[i][j];

}

sum = (sum % 26) + 'a';

encrypt += (char)sum;

}

return encrypt;

}

public static double[][] matCreate(String key, int n){

double[][] mat = new double[n][n];

for(int i=0; i<n; i++){

for(int j=0; j<n; j++){

mat[i][j] = (key.charAt((i\*n)+j) - 'a') % 26 ;

}

}

return mat;

}

public static double[][] getCofactor(double[][] mat, int p, int q, int n){

double temp[][] = new double[n][n];

int i = 0, j = 0;

for (int row = 0; row < n; row++){

for (int col = 0; col < n; col++){

if (row != p && col != q){

temp[i][j++] = mat[row][col];

if (j == n - 1){

j = 0;

i++;

}

}

}

}

return temp;}

public static double determinant(double[][] mat, int n){

int d = 0;

if(n == 1) return mat[0][0];

double temp[][] = new double[n][n];

int sign = 1;

for (int f = 0; f < n; f++){

temp = getCofactor(mat, 0, f, n);

d += sign \* mat[0][f] \* determinant(temp, n - 1);

sign = -sign;

}

return d;

}

public static double[][] adjoint(double[][] mat, int n){

double adj[][] = new double[n][n];

if (n == 1){

adj[0][0] = 1;

return adj;

}

int sign = 1;

double temp[][] = new double[n][n];

for (int i=0; i<n; i++){

for (int j=0; j<n; j++){

temp = getCofactor(mat, i, j, n);

sign = ((i+j)%2==0)? 1: -1;

adj[j][i] = (sign)\*(determinant(temp, n-1));

}

}

return adj;

}

public static double[][] inverse(double[][] mat, int n){

double imat[][] = new double[n][n];

double det = determinant(mat, n);

double[][] adj = adjoint(mat, n);

for (int i=0; i<n; i++)

for (int j=0; j<n; j++){

imat[i][j] = adj[i][j] % 26;

if(imat[i][j] < 0) imat[i][j] += 26;

}

return imat;

}

public static void main(String[] args) throws Exception{

br = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter the text to encrypt...... enter q to quit");

String text = br.readLine();

System.out.println("Enter the key");

String key = br.readLine();

while(!text.equals("q")){

System.out.println("\*\*\*\*\*\*ENCRYPTION\*\*\*\*\*\*");

double[][] mat = matCreate(key, text.length());

double[] a = new double[text.length()];

for(int i=0; i<text.length(); i++)

a[i] = (text.charAt(i) - 'a') % 26;

for(int i=0; i<text.length(); i++){

for(int j=0; j<text.length(); j++)

System.out.print(mat[i][j] + " ");

System.out.println();

}

System.out.println("\*\*\*\*\*\*DECRYPTION\*\*\*\*\*\*");

String encrypt = mult(mat, a, text.length());

System.out.println("Encrypted text = " + encrypt);

mat = inverse(mat, encrypt.length());

for(int i=0; i<text.length(); i++){

for(int j=0; j<text.length(); j++)

System.out.print(mat[i][j] + " ");

System.out.println();

}

String decrypt = mult(mat, a, text.length());

System.out.println("Decrypted text = " + text);

System.out.println("Enter the text to encrypt...... enter q to quit");

text = br.readLine();

if(text.equals("q")) break;

System.out.println("Enter the shift");

key = br.readLine();

}

}

}

**Output:**

Enter the text to encrypt...... enter q to quit

goms

Enter the key

welcometomyworld

\*\*\*\*\*\*ENCRYPTION\*\*\*\*\*\*

22.0 4.0 11.0 2.0

14.0 12.0 4.0 19.0

14.0 12.0 24.0 22.0

14.0 17.0 11.0 3.0

\*\*\*\*\*\*DECRYPTION\*\*\*\*\*\*

Encrypted text = ssao

22.0 10.0 1.0 10.0

16.0 18.0 10.0 10.0

24.0 18.0 0.0 0.0

22.0 2.0 8.0 0.0

Decrypted text = goms

**VIGENERE CIPHER**

**Aim:**

To implement ABCD Vigenere Cipher.

**Algorithm:**

1. Construct a 26 x 26 matrix containing all 26 alphabets.
2. Multiply the keyword to match the length of plain text.
3. Pair the text and the keyword and find the character in the matrix.

**Program:**

import java.io.\*;

import java.util.\*;

public class Hill{

private static BufferedReader br = null;

public static String mult(double[][] mat, double[] a, int n){

String encrypt = "";

for(int i=0; i<n; i++){

int sum = 0;

for(int j=0; j<n; j++)

sum += a[j] \* mat[i][j];

sum = (sum % 26) + 'a';

encrypt += (char)sum;

}

return encrypt;

}

public static double[][] matCreate(String key, int n){

double[][] mat = new double[n][n];

for(int i=0; i<n; i++){

for(int j=0; j<n; j++)

mat[i][j] = (key.charAt((i\*n)+j) - 'a') % 26 ;

return mat;

}

public static double[][] getCofactor(double[][] mat, int p, int q, int n){

double temp[][] = new double[n][n];

int i = 0, j = 0;

for (int row = 0; row < n; row++){

for (int col = 0; col < n; col++){

if (row != p && col != q){

temp[i][j++] = mat[row][col];

if (j == n - 1){

j = 0;

i++;

}

}

}

}

return temp;

}

public static double determinant(double[][] mat, int n){

int d = 0;

if(n == 1) return mat[0][0];

double temp[][] = new double[n][n];

int sign = 1;

for (int f = 0; f < n; f++){

temp = getCofactor(mat, 0, f, n);

d += sign \* mat[0][f] \* determinant(temp, n - 1);

sign = -sign;

}

return d;

}

public static double[][] adjoint(double[][] mat, int n){

double adj[][] = new double[n][n];

if (n == 1){

adj[0][0] = 1;

return adj;

}

int sign = 1;

double temp[][] = new double[n][n];

for (int i=0; i<n; i++){

for (int j=0; j<n; j++){

temp = getCofactor(mat, i, j, n);

sign = ((i+j)%2==0)? 1: -1;

adj[j][i] = (sign)\*(determinant(temp, n-1));

}

}

return adj;

}

public static double[][] inverse(double[][] mat, int n){

double imat[][] = new double[n][n];

double det = determinant(mat, n);

double[][] adj = adjoint(mat, n);

for (int i=0; i<n; i++)

for (int j=0; j<n; j++){

imat[i][j] = (adj[i][j]) % 26;

if(imat[i][j] < 0)

imat[i][j] += 26;

}

return imat;

}

public static void main(String[] args) throws Exception{

br = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter the text to encrypt...... enter q to quit");

String text = br.readLine();

System.out.println("Enter the key");

String key = br.readLine();

while(!text.equals("q")){

System.out.println("\*\*\*\*\*\*ENCRYPTION\*\*\*\*\*\*");

double[][] mat = matCreate(key, text.length());

double[] a = new double[text.length()];

for(int i=0; i<text.length(); i++)

a[i] = (text.charAt(i) - 'a') % 26;

for(int i=0; i<text.length(); i++){

for(int j=0; j<text.length(); j++)

System.out.print(mat[i][j] + " ");

System.out.println();

}

String encrypt = mult(mat, a, text.length());

System.out.println("Encrypted text = " + encrypt);

System.out.println("\*\*\*\*\*DECRYPTION\*\*\*\*\*");

mat = inverse(mat, encrypt.length());

for(int i=0; i<text.length(); i++){

for(int j=0; j<text.length(); j++)

System.out.print(mat[i][j] + " ");

System.out.println();

}

String decrypt = mult(mat, a, text.length());

System.out.println("Decrypted text = " + decrypt);

System.out.println("Enter the text to encrypt...... enter q to quit");

text = br.readLine();

if(text.equals("q")) break;

System.out.println("Enter the shift");

key = br.readLine();

}

}

}

**Output:**

Enter the text to encrypt...... enter q to quit

gomathy

Enter the key

goms

\*\*\*\*\*\*ENCRYPTION\*\*\*\*\*\*

New Key = gomsgom

Encrypted text = mcyszvk

\*\*\*\*\*\*DECRYPTION\*\*\*\*\*\*

Decrypted text = gomathy

**RAILFENCE CIPHER**

**Aim:**

To implement Railfence row and column transformation

**Algorithm:**

1. The plain text is written downward diagonally on successive rails of a fence.
2. On reaching bottom we traverse upwards diagonally.
3. After building the fence the cipher is formed by writing the text row wise.

**Program:**

import java.io.\*;

import java.util.\*;

public class RailFence{

private static BufferedReader br = null;

public static char[][] matCreate(String text, int key){

int n = text.length();

char[][] mat = new char[key][n];

int factor = 1;

for(int i=0, j=0, k=0; k<n; k++){

mat[i][j] = text.charAt(k);

i += factor;

j++;

if(i == key || i < 0){

factor \*= -1;

i += factor;

i += factor;

}

}

return mat;

}

public static char[][] deMatCreate(String text, int key){

int n = text.length();

char[][] mat = new char[key][n];

int factor = 1;

for(int i=0, j=0, k=0; k<n; k++){

mat[i][j] = '\*';

i += factor;

j++;

if(i == key || i < 0){

factor \*= -1;

i += factor;

i += factor;

}

}

int k =0;

for(int i=0; i<key; i++){

for(int j=0; j<text.length(); j++)

if(mat[i][j] == '\*')

mat[i][j] = text.charAt(k++);

}

return mat;

}

public static String decryptText(String text, int key){

char[][] mat = new char[key][text.length()];

int z = 0;

mat = deMatCreate(text, key);

for(int i=0; i<key; i++){

for(int j=0; j<text.length(); j++)

System.out.print(mat[i][j] + " ");

System.out.println();

}

String decrypt = "";

for(int j=0; j<text.length(); j++){

for (int i=0; i<key; i++)

if(mat[i][j] != 0)

decrypt += mat[i][j];

}

return decrypt;

}

public static void main(String[] args) throws Exception{

br = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter the text to encrypt...... enter q to quit");

String text = br.readLine();

System.out.println("Enter the key");

int key = Integer.parseInt(br.readLine());

while(!text.equals("q")){

System.out.println("\*\*\*\*\*\*ENCRYPTION\*\*\*\*\*\*");

char[][] mat = matCreate(text, key);

for(int i=0; i<key; i++){

for(int j=0; j<text.length(); j++)

System.out.print(mat[i][j] + " ");

System.out.println();

}

String encrypt = "";

for(int i=0; i<key; i++){

for (int j=0; j<text.length() ; j++)

if(mat[i][j] != 0)

encrypt += mat[i][j];

}

System.out.println("Encrypted text = " + encrypt);

System.out.println("\*\*\*\*\*\*DECRYPTION\*\*\*\*\*\*");

System.out.println("Decypted Text = " + decryptText(encrypt, key));

System.out.println("Enter the text to encrypt...... enter q to quit");

text = br.readLine();

if(text.equals("q")) break;

System.out.println("Enter the shift");

key = Integer.parseInt(br.readLine());

}

}

}

**Output:**

Enter the text to encrypt...... enter q to quit

gomathy nagarajan

Enter the shift

3

\*\*\*\*\*\*ENCRYPTION\*\*\*\*\*\*

g t n r n

o a h a a a a

m y g j

Encrypted text = gtnrnoah aaaamygj

\*\*\*\*\*\*DECRYPTION\*\*\*\*\*\*

g t n r n

o a h a a a a

m y g j

Decypted Text = gomathy nagarajan

**Result:**

Thus translation and substitution ciphers have been implemented successfully.

**Ex.No. 2**

**Implement the following algorithms DES, RSA, Diffie, MD5, SHA1**

**DATA ENCRYPTION STANDARD**

**Aim:**

To implement Data Encryption Standard algorithm.

**Algorithm:**

The Data Encryption Standard (DES) is a symmetric-key block cipher.

**DES Encryption:**

1. Plaintext is broken into blocks of length 64 bits. Encryption is block wise.
2. A message block is first gone through an initial permutation IP, then divided into two parts L0,where L0 is the left part of 32 bits and R0 is the right part of the 32 bits
3. Round i has input Li-1,Ri-1 and output Li, Ri.

        Li = Ri-1, Ri = Li-1 ⊕ f(Ri-1,Ki)

        and Ki is the subkey for the 'i'th where 1 ≤ i ≤ 16

      L1 = R0,    R1 = L0 ⊕ f(R0,K1)

      L2 = R1,    R2 = L1 ⊕ f(R1,K2)

      L3 = R2,    R3 = L2 ⊕ f(R2,K3)

      ................         ..........................

      ................         ..........................

      ................         ..........................

      L16 = R15,    R16 = L15 ⊕ f(R15,K16)

1. After round 16, L16 and R16 are swapped, so that the decryption algorithm has the same structure as the encryption algorithm.
2. Finally, the block is gone through the inverse the permutation IP-1 and then output

**DES Decryption:**

1. In encryption, we have

      Li = Ri-1, Ri = Li-1 ⊕ f(Ri-1, Ki)

1. Ki is the subkey for the 'i'th round. Hence

    Ri-1 = Li, Li-1 = Ri ⊕ f(Li,Ki) for each 'i'

1. Due to swap operation after the 16th round encryption, the output of encryption is IP-1(R16,L16)
2. Equation(1) as follows:

      R15 = L16,    L15 = R16 ⊕ f(L16,K16)

      R14 = L15,    L14 = R15 ⊕ f(L15,K15)

      R13 = L14,    L13 = R14 ⊕ f(L14,K14)

      ................         ..........................

      ................         ..........................

      ................         ..........................

      R1 = L2,    L1 = R2 ⊕ f(L2,K2)

1. If we give IP-1(R16,L16) as the input for the same algorithm with round sub keys(K16,K15,......K1),then the output is IP-1(L0,R0),the original message block
2. Decryption is performed using the same algorithm, except the K16 is used as the first round,K15 in the second, and so on, with K1 used in the 16th round.

**Program:**

import java.util.\*;

public 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], 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:

0123456789ABCDEF

Enter the key as a 16 character hexadecimal value:

FEDCBA9876543210

**+++ ENCRYPTION +++**

L0 = cc00ccff R0 = f0aaf0aa

*Round 1:*

Round key = f4fd9864b65a L = f0aaf0aa R = dd840dda

*Round 2:*

Round key = 9659a6da95d9 L = dd840dda R = 1fb1e4ce

*Round 3:*

Round key = ba2b754bd72d L = 1fb1e4ce R = 128e50f5

*Round 4:*

Round key = 8d762d5a7da8 L = 128e50f5 R = 8a8fc403

*Round 5:*

Round key = c317fce8593d L = 8a8fc403 R = c5c92ca4

*Round 6:*

Round key = dcdae1c37aba L = c5c92ca4 R = e7a273b1

*Round 7:*

Round key = 93fb6af51b39 L = e7a273b1 R = 74ecf977

*Round 8:*

Round key = a877c7931a7e L = 74ecf977 R = 7d5487fb

*Round 9:*

Round key = 3f3616d947c6 L = 7d5487fb R = 038409a7

*Round 10:*

Round key = 6e1cf89ce28d L = 038409a7 R = 903ffb1f

*Round 11:*

Round key = dee07cf276c5 L = 903ffb1f R = 22a78279

*Round 12:*

Round key = 8ecf1abaa3ab L = 22a78279 R = f34f1a1e

*Round 13:*

Round key = 6e3b2fb67f03 L = f34f1a1e R = aab320aa

*Round 14:*

Round key = abbc497e2372 L = aab320aa R = f1f9f27c

*Round 15:*

Round key = 496efaf5e94a L = f1f9f27c R = d5735710

*Round 16:*

Round key = 35c2fc478fcd L = d5735710 R = bd7ee107

Encrypted text: ED39D950FA74BCC4

**+++ DECRYPTION +++**

L0 = bd7ee107 R0 = d5735710

*Round 1:*

Round key = 35c2fc478fcd L = d5735710 R = f1f9f27c

*Round 2:*

Round key = 496efaf5e94a L = f1f9f27c R = aab320aa

*Round 3:*

Round key = abbc497e2372 L = aab320aa R = f34f1a1e

*Round 4:*

Round key = 6e3b2fb67f03 L = f34f1a1e R = 22a78279

*Round 5:*

Round key = 8ecf1abaa3ab L = 22a78279 R = 903ffb1f

*Round 6:*

Round key = dee07cf276c5 L = 903ffb1f R = 038409a7

*Round 7:*

Round key = 6e1cf89ce28d L = 038409a7 R = 7d5487fb

*Round 8:*

Round key = 3f3616d947c6 L = 7d5487fb R = 74ecf977

*Round 9:*

Round key = a877c7931a7e L = 74ecf977 R = e7a273b1

*Round 10:*

Round key = 93fb6af51b39 L = e7a273b1 R = c5c92ca4

*Round 11:*

Round key = dcdae1c37aba L = c5c92ca4 R = 8a8fc403

*Round 12:*

Round key = c317fce8593d L = 8a8fc403 R = 128e50f5

*Round 13:*

Round key = 8d762d5a7da8 L = 128e50f5 R = 1fb1e4ce

*Round 14:*

Round key = ba2b754bd72d L = 1fb1e4ce R = dd840dda

*Round 15:*

Round key = 9659a6da95d9 L = dd840dda R = f0aaf0aa

*Round 16:*

Round key = f4fd9864b65a L = f0aaf0aa R = cc00ccff

Decrypted text: 0123456789ABCDEF

**RSA ALGORITHM**

**Aim:**

To implement RSA asymmetric cryptographic algorithm.

**Algorithm:**

1. **Generate public key:**
   1. Select two prime numbers, p and q.
   2. Public key is composed of n and e, where e is an integer, n = p \* q;
   3. e is not a factor of n, 1 < e < theta(n).
2. **Generate private key:**
   1. Theta(n) = (p-1) \* (q-1)
   2. Private key d = (k \* theta(n) + 1) / e, k is an integer.
3. **Encryption and Decryption:**
   1. CT = pow (PT, e) mod n.
   2. PT = pow (CT, d) mod n.

**Program:**

import java.io.\*;

import java.util.\*;

import java.lang.Math.\*;

public class RSA{

private static BufferedReader br = null;

public static double gcd(double a, double h){

double temp;

while (true){

temp = a%h;

if (temp == 0) return h;

a = h;

h = temp;

}

}

public static void main(String[] args) throws Exception{

br = new BufferedReader(new InputStreamReader(System.in));

double p, q;

System.out.println("Enten the prime number P");

p = Double.parseDouble(br.readLine());

System.out.println("Enten the prime number Q");

q = Double.parseDouble(br.readLine());

double n = p\*q;

int k = 2;

double e = 2;

double phi = (p-1)\*(q-1);

while(e < phi){

if(gcd(e, phi) == 1)

break;

else e++;

}

double d = (k\*phi+1)/e;

System.out.println("Enter the message");

double msg = Double.parseDouble(br.readLine());

double encrypt = Math.pow(msg, e) % n;

System.out.println("d = " + d + " Encrypted message = " + encrypt);

double decrypt = 1;

while(d > 0){

decrypt \*= encrypt % n;

decrypt %= n;

d--;

}

System.out.println("Decrypted message = " + decrypt);

}

}

**Output:**

Enten the prime number P

53

Enten the prime number Q

59

Enter the message

89

d = 2011.0 Encrypted message = 1394.0

Decrypted message = 89.0

**DIFFIE HELLMAN ALGORITHM**

**Aim:**

To implement Diffie Hellman Algorithm.

**Algorithm:**

1. **Sender side**
   1. Public key = P, G
   2. Private key = a
   3. Key generated, x = pow(G, a) mod P
   4. Exchange generated keys.
   5. Key received = y
   6. Generated secret key, ka = pow(y, a) mod P
2. **Receiver side**
   1. Public keys = P, G
   2. Private key = b
   3. Key generated, x = pow(G, b) mod P
   4. Exchange generated keys.
   5. Key received = x
   6. Generated secret key, kb = pow(x, b) mod P

**Program:**

import java.io.\*;

import java.math.BigInteger;

public class Diffie{

public static void main(String[]args)throws IOException {

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter prime number P:");

BigInteger p=new BigInteger(br.readLine());

System.out.print("Enter prime number G:");

BigInteger g=new BigInteger(br.readLine());

System.out.println("Enter value x:");

BigInteger x=new BigInteger(br.readLine());

BigInteger R1=g.modPow(x,p);

System.out.println("R1="+R1);

System.out.print("Enter value y:");

BigInteger y=new BigInteger(br.readLine());

BigInteger R2=g.modPow(y,p);

System.out.println("R2="+R2);

BigInteger k1=R2.modPow(x,p);

System.out.println("Key calculated at Sender side:"+k1);

BigInteger k2=R1.modPow(y,p);

System.out.println("Key calculated at Receiver side:"+k2);

}

}

**Output:**

Enter prime number P: 23

Enter prime number G: 9

Enter value x: 4

R1=6

Enter value y: 3

R2=16

Key calculated at Sender side: 9

Key calculated at Receiver side: 9

**MD5 ALGORITHM**

**Aim:**

To implement MD5 algorithm.

**Algorithm:**

1. Divide the input text into blocks of 512 bits each.
2. 64 bits are inserted at the end of last block.
3. These 64 bits record the length of original input.
4. MD5 helper function
   1. **The Buffer**

MD5 uses a buffer that is made up of 4 words that are 32 bits long.

* 1. **The Table**

MD5 uses a table k that has 64 elements. Each number i is indicated as Ki

Ki = abs (sin (i + 1)) \* 2^32

* 1. **Four auxiliary function**

F(X, Y, Z) = (X && Y) || (!(X) && Z)

G(X, Y, Z) = (X && Y) || (Y && !(Z))

H(X, Y, Z) = (X ^ Y ^ Z)

I(X, Y, Z) = Y ^ (X && !(Z))

**Program:**

public class MD5{

private static final int INIT\_A = 0x67452301, INIT\_B = (int) 0xEFCDAB89L;

private static final int INIT\_C = (int) 0x98BADCFEL, INIT\_D = 0x10325476;

private static final int[] SHIFT\_AMTS={ 7, 12, 17, 22, 5, 9, 14, 20, 4, 11, 16, 23, 6, 10, 15, 21};

private static final int[] TABLE\_T = new int[64];

static{

for (int i = 0; i < 64; i++)

TABLE\_T[i] = (int) (long) ((1L << 32) \* Math.abs(Math.sin(i + 1)));

}

public static byte[] computeMD5(byte[] message){

int messageLenBytes = message.length;

int numBlocks = ((messageLenBytes + 8) >>> 6) + 1;

int totalLen = numBlocks << 6;

byte[] paddingBytes = new byte[totalLen - messageLenBytes];

paddingBytes[0] = (byte) 0x80;

long messageLenBits = (long) messageLenBytes << 3;

for (int i = 0; i < 8; i++){

paddingBytes[paddingBytes.length - 8 + i] = (byte) messageLenBits;

messageLenBits >>>= 8;

}

int a = INIT\_A;

int b = INIT\_B;

int c = INIT\_C;

int d = INIT\_D;

int[] buffer = new int[16];

for (int i = 0; i < numBlocks; i++){

int index = i << 6;

for (int j = 0; j < 64; j++, index++)

buffer[j >>> 2] = ((int) ((index < messageLenBytes) ? message[index] : paddingBytes[index - messageLenBytes]) << 24) | (buffer[j >>> 2] >>> 8);

int originalA = a, originalB = b, originalC = c, originalD = d;

for (int j = 0; j < 64; j++){

int div16 = j >>> 4;

int f = 0;

int bufferIndex = j;

switch (div16){

case 0:

f = (b & c) | (~b & d);

break;

case 1:

f = (b & d) | (c & ~d);

bufferIndex = (bufferIndex \* 5 + 1) & 0x0F;

break;

case 2:

f = b ^ c ^ d;

bufferIndex = (bufferIndex \* 3 + 5) & 0x0F;

break;

case 3:

f = c ^ (b | ~d);

bufferIndex = (bufferIndex \* 7) & 0x0F;

break;

}

int temp = b + Integer.rotateLeft(a + f + buffer[bufferIndex] + TABLE\_T[j],

SHIFT\_AMTS[(div16 << 2) | (j & 3)]);

a = d;

d = c;

c = b;

b = temp;

}

a += originalA;

b += originalB;

c += originalC;

d += originalD;

}

byte[] md5 = new byte[16];

int count = 0;

for (int i = 0; i < 4; i++){

int n = (i == 0) ? a : ((i == 1) ? b : ((i == 2) ? c : d));

for (int j = 0; j < 4; j++){

md5[count++] = (byte) n;

n >>>= 8;

}

}

return md5;

}

public static String toHexString(byte[] b){

StringBuilder sb = new StringBuilder();

for (int i = 0; i < b.length; i++) sb.append(String.format("%02X", b[i] & 0xFF));

return sb.toString();

}

public static void main(String[] args){

String[] testStrings = { "", "Gomathy", "Gomathy Nagarajan", "abcdefghijklmnopqrstuvwxyz"};

for (String s : testStrings)

System.out.println("0x" + toHexString(computeMD5(s.getBytes())) + " <== \"" + s + "\"");

return;

}

}

**Output:**

0xD41D8CD98F00B204E9800998ECF8427E <== ""

0x5DDBA4B789A1DFB9E6A24430C2611BC0 <== "Gomathy"

0x966368B9D7A14A050D10F8ACC1A3E167 <== "Gomathy Nagarajan"

0xC3FCD3D76192E4007DFB496CCA67E13B <== "abcdefghijklmnopqrstuvwxyz"

**SHA1 ALGORITHM**

**Aim:**

To implement SHA1 algorithm.

**Algorithm:**

**Input:** Any input whose length is less than 2^64 bits.

**Output:** Output 160 bits

1. Padding of bits.
2. Append length.
3. Divide the input into blocks of 512 bits.
4. Initialize chaining variables.

A = 01 23 45 67

B = 89 AB CD EF

C = FE DC BA 98

D = 76 53 32 98

E = C3 D2 E1 F0

1. Process Blocks
   1. Copy chaining variables into a-e variables.
   2. Divide current 512 bit block into 16 sub-blocks of 32 bits.
   3. SHA has 4 rounds, each 20 steps, 3 inputs, 512 bit block, register abcde.

|  |  |
| --- | --- |
| **ROUND** | **VALUE OF t** |
| 1 | 1 and 19 |
| 2 | 20 and 39 |
| 3 | 40 and 59 |
| 4 | 60 and 79 |

* 1. abcde = (e + process p + s^5 (a) + w[t], k[t]), a, s^30(b), c, d.
  2. S^t = circular shift left of 32 bit block by t bit.

|  |  |
| --- | --- |
| **ROUND** | **PROCESS p** |
| 1 | (b & c) | (~b & d) |
| 2 | b ^ c ^ d |
| 3 | (b & c) | (b & d) | (c & d) |
| 4 | b ^ c ^d |

**Program:**

import java.io.\*;

import java.lang.Math.\*;

import java.security.\*;

public class SHA{

public static void main(String arg[]) throws Exception {

System.out.println(encrypt("abirami"));

}

public static byte[] encrypt(String x) throws Exception {

MessageDigest d = null;

d = MessageDigest.getInstance("SHA1");

d.reset();

d.update(x.getBytes());

return d.digest();

}

}

**Output:**

CBA6d06d69c

**Result:**

This DES, RSA, Diffie Hellman, MD5 and SHA1 algorithms have been implemented successfully.

**Ex.No. 3**

**Write a program to implement a set of values to combine the control of Bell lapadula with the integrating controls of the Biba model**

**Aim:**

To write a program to implement a set of values to combine the control of Bell lapadula with the integrating controls of the Biba model.

**Algorithm:**

**Bell lapadula**

1. State machine model that describes a set of access control rules which use security labels on objects and clear access for subjects.
2. Security labels.
   1. Top secret.
   2. Secret
   3. Confidentiality.
3. Properties.
   1. No read up
   2. No write down
   3. Discretionary security property

**Biba model**

1. Biba model has a lattice structure.
2. Defined on mathematical basis that allows security level dedicated by Bella lapdula.
3. Biba policy uses three defining properties to protect objects from illegitimately modified.
   1. Simply integrity.
   2. Star integrity.
   3. Invocation.

**Program:**

import java.io.\*;

import java.util.\*;

public class BellBiba{

public int number;

public String[] user;

public int[] prior,stored;

public String[][] written;

public BellBiba(int number) {

this.number=number;

user=new String[number];

prior=new int[number];

written=new String[number][10];

stored=new int[number];

}

public void BellLaPadula(){

System.out.println("Bell LaPadula\nFiles which a user can read");

for(int i=0;i<number;i++){

System.out.println(user[i]);

for(int j=prior[i]+1;j<number;j++){

int l=0;

for(int ii=0;ii<number;ii++)

if(prior[ii]==j){

l=ii;

break;

}

for(int k=0;k<stored[l];k++)

System.out.println("-->"+written[l][k]);

}

System.out.println();

}

System.out.println("Users and their accepted receipients");

for(int i=0;i<number;i++){

System.out.println(user[i]);

for(int j=prior[i]-1;j>=0;j--){

int l=0;

for(int ii=0;ii<number;ii++)

if(prior[ii]==j)

{

l=ii;

break;

}

System.out.println("-->"+user[l]);

}

System.out.println();

}

}

public void Biba(){

System.out.println("Biba Model\nFiles which a user can read");

for(int i=0;i<number;i++){

System.out.println(user[i]);

for(int j=prior[i]-1;j>=0;j--){

int l=0;

for(int ii=0;ii<number;ii++)

if(prior[ii]==j){

l=ii;

break;

}

for(int k=0;k<stored[l];k++)

System.out.println("-->"+written[l][k]);

}

System.out.println();

}

System.out.println("Users and their accepted receipients");

for(int i=0;i<number;i++){

System.out.println(user[i]);

for(int j=prior[i]+1;j<number;j++){

int l=0;

for(int ii=0;ii<number;ii++)

if(prior[ii]==j) {

l=ii;

break;

}

System.out.println("-->"+user[l]);

}

System.out.println();

}

}

public static void main(String[] argv) throws Exception {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter the number of users");

int n = Integer.parseInt(br.readLine());

BellBiba bb=new BellBiba(n);

System.out.println("Enter the user names");

for(int i=0; i<n; i++) bb.user[i]=br.readLine();

System.out.println("Enter the user prior");

for(int i=0; i<n; i++) bb.prior[i]=Integer.parseInt(br.readLine());

System.out.println("Enter the sub-user");

for(int i=0; i<n; i++){

bb.stored[i]=Integer.parseInt(br.readLine());

}

bb.written[0][0]="Article1.pdf";

bb.written[0][1]="Article2.pdf";

bb.written[1][0]="Article3.docx";

bb.written[1][1]="Article4.pdf";

bb.written[1][2]="Article5.pdf";

bb.written[2][0]="Article6.pdf";

bb.written[3][0]="Article7.docx";

bb.written[3][1]="Article8.rtf";

bb.written[3][2]="Article9.pdf";

bb.written[4][0]="Article10.pdf";

bb.written[4][1]="Article11.pdf";

bb.BellLaPadula();

bb.Biba();

}

}

**Output:**

Enter the number of users 5

Enter the user names

abi

goms

abirami

gomathy

abigoms

Enter the user prior 5 4 3 2 1

Enter the sub-user 3 2 1 5 4

Bell LaPadula

Files which a user can read

abi -->

goms -->

abirami --> Article3.docx, Article4.pdf,

gomathy --> Article6.pdf, Article3.docx, Article4.pdf,

abigoms --> Article7.docx, Article8.rtf, Article9.pdf, null, null, Article6.pdf, Article3.docx, Article4.pdf,

Users and their accepted receipients

abi-->goms, abirami, gomathy, abigoms, abi,

goms-->abirami, gomathy, abigoms, abi,

abirami-->gomathy, abigoms, abi,

gomathy-->abigoms, abi,

abigoms-->abi,

Biba Model

Files which a user can read

abi-->Article3.docx, Article4.pdf, Article6.pdf, Article7.docx, Article8.rtf, Article9.pdf, null, null, Article10.pdf, Article11.pdf, null, null, Article1.pdf, Article2.pdf, null,

goms-->Article6.pdf, Article7.docx, Article8.rtf, Article9.pdf, null, null, Article10.pdf, Article11.pdf, null, null, Article1.pdf, Article2.pdf, null,

abirami-->Article7.docx, Article8.rtf, Article9.pdf, null, null, Article10.pdf,Article11.pdf, null, null, Article1.pdf, Article2.pdf, null,

gomathy-->Article10.pdf, Article11.pdf, null, null, Article1.pdf, Article2.pdf, null,

abigoms-->Article1.pdf, Article2.pdf, null,

Users and their accepted receipients

abi-->

goms-->

abirami-->goms,

gomathy-->abirami, goms,

abigoms-->gomathy, abirami, goms,

**Result:**

Thus a program to implement a set of values to combine the control of Bell laPadula with the integrating controls of the Biba models has been implemented successfully.

**Ex.No. 4**

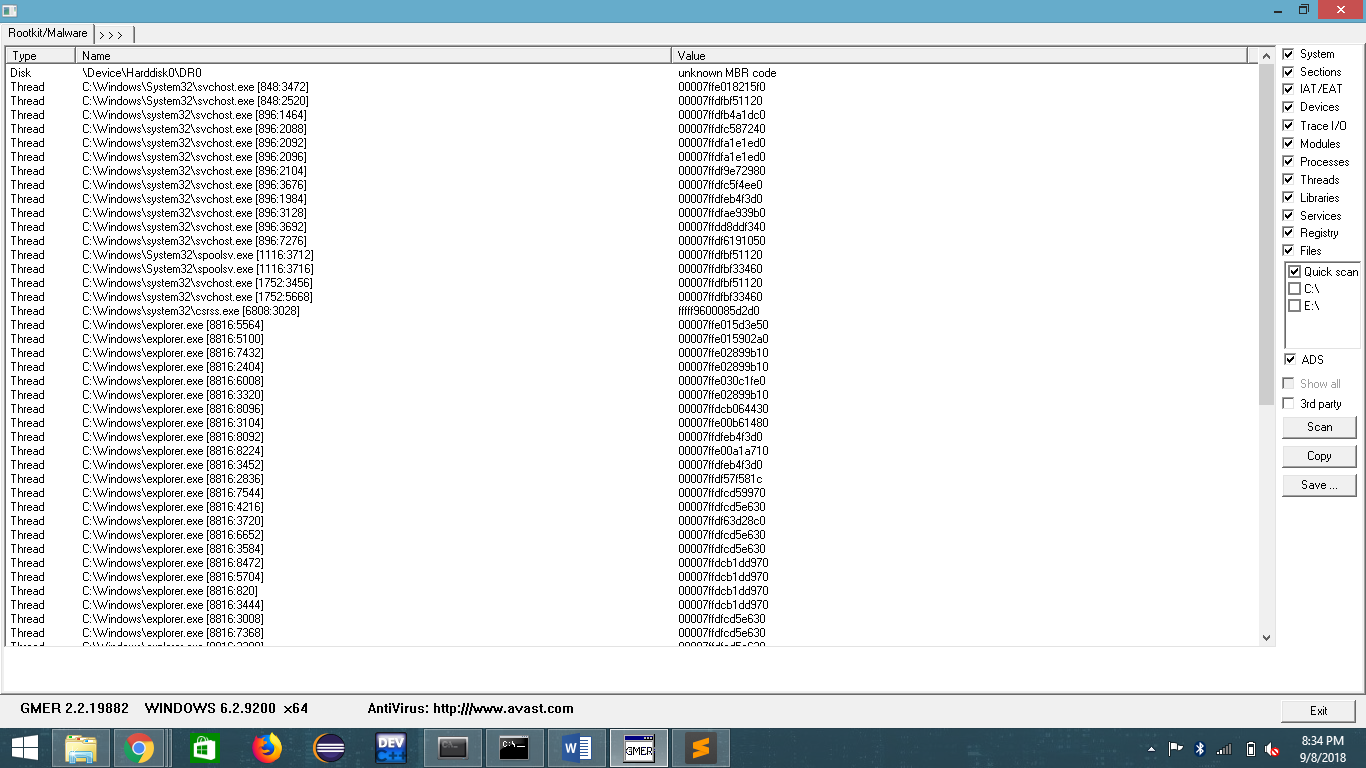
**Installation of rootkits and the study about various options**

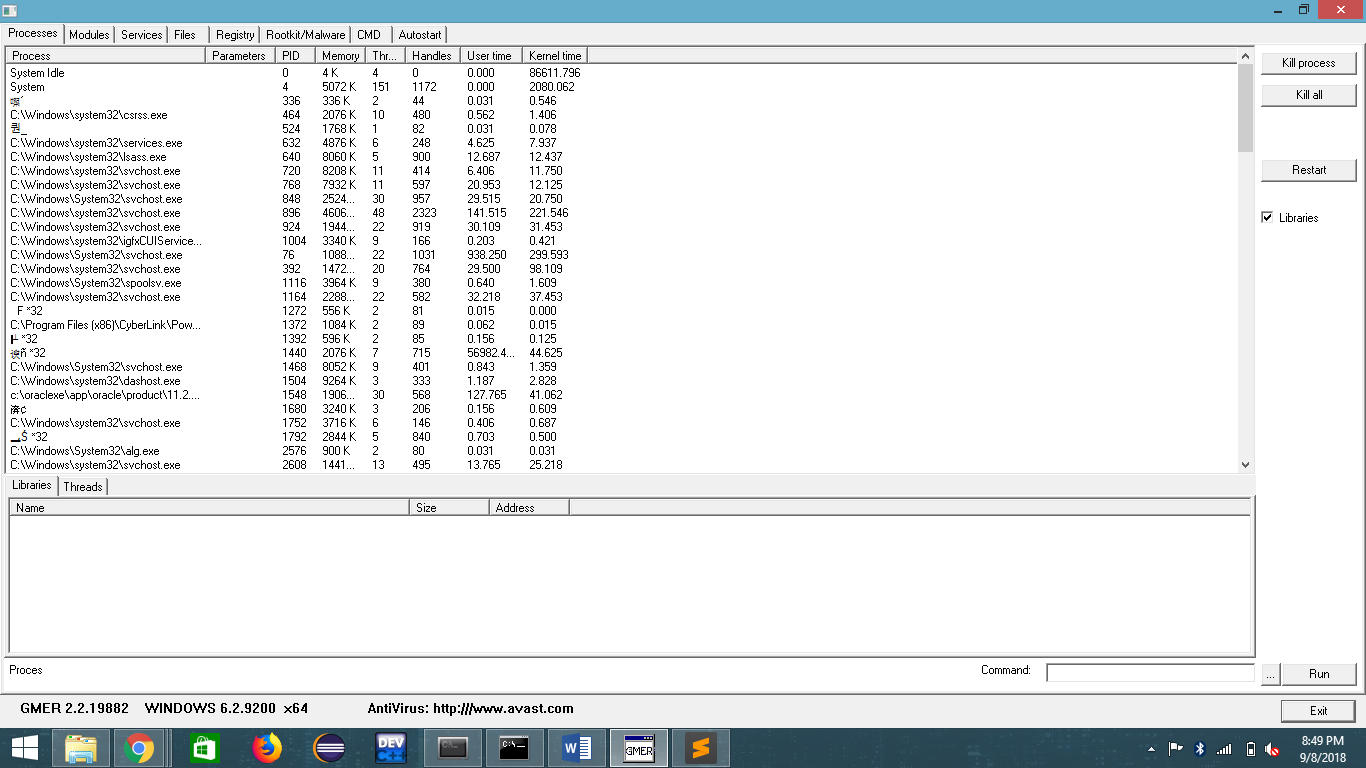
**Aim:**

To install rootkits and study the various options available.

**Procedure:**

1. Download the rootkit tool from the GMER website.
2. This tool displays the following options, Processes, Modules, Services, Files, Registry, Rootkit/Malware, Auto start, cmd of localhost.
3. Select Processes menu and kill any unwanted processes.
4. Modules menu display the various system files like .sys, .dll.
5. Services menu display the complete services running with Autostart, Enable, Disable, System, Boot.
6. Files menu display full files on hard-disk volumes.
7. Rootkits/Malware scans the local drivers selected.
8. Autostart displays the registry base Autostart applications.
9. CMD allows the user to interact wth command line utilities or registry.

**Output:**

**Result:**

Thus implementation of rootkits and the study about various experiments has been implemented successfully.

**Ex.No. 5**

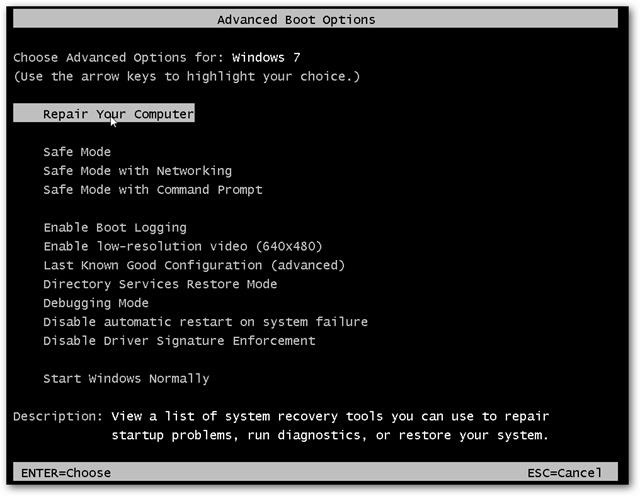
**Implement hacking windows – windows login and password**

**Aim:**

To implement hacking windows to hack windows login and password.

**Procedure:**

1. Any Linux live CD/USB.
2. Insert the live CD/USB and boot from it.
3. Locate the driver where Windows is installed.
4. Rename the file named cmd.exe to cmd0.exe.
5. Rename the file named sethc.exe to cmd.exe.
6. Rename the file named cmd0.exe to sethcexe.
7. Shut down and boot into windows.
8. Press shift key 5 times.
9. Type net user to display the list of active users.
10. Type net user <account name> \*.
11. It will ask for new password, enter the new password.
12. Done.

**Output:**



**Result:**

Thus hacking windows login and password has been implemented successfully.

**Ex.No.6**

**Implement hacking windows – Accessing restricted drivers**

**Aim:**

To implement hacking windows to access restricted drivers.

**Procedure:**

Task Manager

* If task manager can't be opened with the Ctrl-Alt-Delete shortcut, then you may want to try the Ctrl-Shift-Escape shortcut. It can open hundreds of error messages in a few seconds if this shortcut is blocked.
* There are other ways of opening Task Manager or similar tools. The blocked computers in most of the cases block both task manager shortcuts as well as installing and running new software, but they left Microsoft Access installed with full Visual Basic for Applications functionality, and many places on the Internet have codes for creating your own task manager or process list.

Remote Control

One way of opening a lot of blocked file formats or viewing blocked websites is by remotely controlling another machine that doesn't have these restrictions. One way is to use [TeamViewer](http://www.teamviewer.com/en/index.aspx?cdsplit=C) where you can install it on removable media, run it without installation on computers that stop you installing new software, or you can even use it through a web browser if you can't run .exe files.

Website Blocking

* The easiest way is to use a circumventor like [stupidcensorship.com](http://www.stupidcensorship.com/), although these are often blocked quite quickly as well, so sign up to their mailing list to receive emails whenever a new circumventing site is created.
* Check that the sites aren't blocked locally. Check this by going into your browser's "tools" section or its equivalent on whatever browser you use, click on security or a similar section and look for a restricted or blocked sites section, then view these sites and remove any you want to access if they're there.
* Accessing the blocked website using its IP address, you find this by opening the command prompt on any Windows computer (by typing cmd in the run box) and typing:

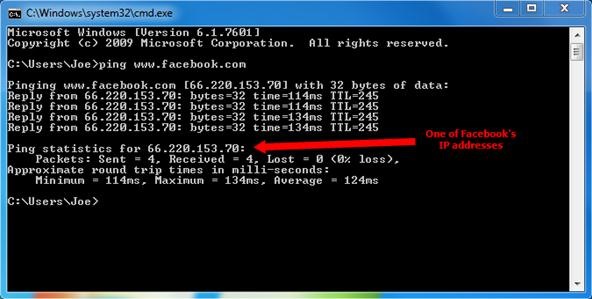
*ping www.example.com*

Installing Software

* There are several ways to find admin passwords, often the default passwords will still work, or you can use a password cracker like Ophcrack to find local admin passwords.
* If you have access to the command prompt on a Windows computer, then you can use the "*Net user*" and "*Net Group*" commands to find out details on each account, and if you know enough about command prompts, you may even be able to change the passwords.
* Boot Windows in safe mode by pressing F8 as Windows boots and trying to install the software then. Safe mode is a good way to bypass a lot of security software and restrictions. For example, my school had an "RM Login" screen that I could only skip by using safe mode.

Running Software

* Using safe mode like in the last section, Use Notepad and make some kind of runnable file. Here is a list of some of the most common runnable formats on Windows: .exe, .bat, .vbs, .cmd
* Used is booting or running from an external device. I've used USB flash drives, CDs, DVDs and an external hard drive and have booted both Linux and Windows operating systems. Most operating systems can be installed on removable media. Once you've booted from this device, there are often no restrictions at all on the system other than Internet-based ones which force you to connect to the Internet through a proxy server that blocks websites, but like I've said, they're not hard to get around.

**Output:**

**Result:**

Thus accessing restricted drivers has been implemented successfully.