1. Implement any two Substitution Techniques. AIM: Write a Java program to perform encryption and decryption using the  
   following algorithms:  
   a) **Ceaser Cipher**

ALGORITHM

1. In Ceaser Cipher each letter in the plaintext is replaced by a letter some

fixed number of positions down the alphabet.

2. For example, with a left shift of 3, D would be replaced by A, E would

become B, and so on.

3. The encryption can also be represented using modular arithmetic by first

transforming the letters into numbers, according to the scheme, A = 0, B = 1,Z = 25.

4. Encryption of a letter x by a shift n can be described mathematically as,

En(x) = (x + n) mod26

5. Decryption is performed similarly,

Dn (x)=(x - n) mod26

PROGRAM:

class caesarCipher {

public static String encode(String enc, int offset) {

offset = offset % 26 + 26;

StringBuilder encoded = new StringBuilder();

for (char i : enc.toCharArray()) {

if (Character.isLetter(i)) {

if (Character.isUpperCase(i)) {

encoded.append((char) ('A' + (i - 'A' + offset) % 26));

} else {

encoded.append((char) ('a' + (i - 'a' + offset) % 26));

}

} else {

encoded.append(i);

}

}

return encoded.toString();

}

public static String decode(String enc, int offset) {

return encode(enc, 26 - offset);

}

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

String msg = "LBRCE CSE";

System.out.println("Simulating Caesar Cipher\n------------------------");

System.out.println("Input : " + msg);

System.out.printf("Encrypted Message : ");

System.out.println(caesarCipher.encode(msg, 3));

System.out.printf("Decrypted Message : ");

System.out.println(caesarCipher.decode(caesarCipher.encode(msg, 3), 3));

}

}

Output:

Simulating Caesar Cipher

------------------------Input : LBRCE CSE

Encrypted Message : OEUFH FVH

Decrypted Message : LBRCE CSE

**RESULT:**

Thus the program for ceaser cipher encryption and decryption algorithm has

been implemented and the output verified successfully.

b) **Playfair Cipher**

**AIM:**

To implement a program to encrypt a plain text and decrypt a cipher text

using play fair Cipher substitution technique.

**ALGORITHM:**

1. To encrypt a message, one would break the message into digrams (groups of

2 letters)

2. For example, "HelloWorld" becomes "HE LL OW OR LD".

3. These digrams will be substituted using the key table.

4. Since encryption requires pairs of letters, messages with an odd number of

characters usually append an uncommon letter, such as "X", to complete the

final digram.

5. The two letters of the digram are considered opposite corners of a rectangle

in the key table. To perform the substitution, apply the following 4 rules, in

order, to each pair of letters in the plaintext:

**PROGRAM:**

**playfairCipher.java**

import java.awt.Point;

class playfairCipher {

private static char[][] charTable;

private static Point[] positions;

private static String prepareText(String s, boolean chgJtoI) {

s = s.toUpperCase().replaceAll("[^A-Z]", "");

return chgJtoI ? s.replace("J", "I") : s.replace("Q", "");

}

private static void createTbl(String key, boolean chgJtoI) {

charTable = new char[5][5];

positions = new Point[26];

String s = prepareText(key + "ABCDEFGHIJKLMNOPQRSTUVWXYZ",

chgJtoI);

int len = s.length();

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

char c = s.charAt(i);

if (positions[c - 'A'] == null) {

charTable[k / 5][k % 5] = c;

positions[c - 'A'] = new Point(k % 5, k / 5);

k++;

}

}

}

private static String codec(StringBuilder txt, int dir) {

int len = txt.length();

for (int i = 0; i < len; i += 2) {

char a = txt.charAt(i);

char b = txt.charAt(i + 1);

int row1 = positions[a - 'A'].y;

int row2 = positions[b - 'A'].y;

int col1 = positions[a - 'A'].x;

int col2 = positions[b - 'A'].x;

if (row1 == row2) {

col1 = (col1 + dir) % 5;

col2 = (col2 + dir) % 5;

} else if (col1 == col2) {

row1 = (row1 + dir) % 5;

row2 = (row2 + dir) % 5;

} else {

int tmp = col1;

col1 = col2;

col2 = tmp;

}

txt.setCharAt(i, charTable[row1][col1]);

txt.setCharAt(i + 1, charTable[row2][col2]);

}

return txt.toString();

}

private static String encode(String s) {

StringBuilder sb = new StringBuilder(s);

for (int i = 0; i < sb.length(); i += 2) {

if (i == sb.length() - 1) {

sb.append(sb.length() % 2 == 1 ? 'X' : "");

} else if (sb.charAt(i) == sb.charAt(i + 1)) {

sb.insert(i + 1, 'X');

}

}

return codec(sb, 1);

}

private static String decode(String s) {

return codec(new StringBuilder(s), 4);

}

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

String key = "CSE";

String txt = "Security Lab"; /\* make sure string length is even \*/ /\* change J

to I \*/

boolean chgJtoI = true;

createTbl(key, chgJtoI);

String enc = encode(prepareText(txt, chgJtoI));

System.out.println("Simulating Playfair Cipher\n----------------------");

System.out.println("Input Message : " + txt);

System.out.println("Encrypted Message : " + enc);

System.out.println("Decrypted Message : " + decode(enc));

}

}

**OUTPUT:**

Simulating Playfair Cipher

----------------------

Input Message : Security Lab

Encrypted Message : EABPUGYANSEZ

Decrypted Message : SECURITYLABX

**RESULT:**

Thus the program for playfair cipher encryption and decryption algorithm

has been implemented and the output verified successfully.

2. Implement any two Transposition Techniques

a) Rail Fence Cipher Transposition Technique

**AIM:**

To implement a program for encryption and decryption using rail fence

transposition technique.

**ALGORITHM:**

1. In the rail fence cipher, the plaintext is written downwards and diagonally on

successive "rails" of an imaginary fence, then moving up when we reach the

bottom rail.

2. When we reach the top rail, the message is written downwards again until

the whole plaintext is written out.

3. The message is then read off in rows.

**PROGRAM:**

**railFenceCipher.java**

class railfenceCipherHelper {

int depth;

String encode(String msg, int depth) throws Exception {

int r = depth;

int l = msg.length();

int c = l / depth;

int k = 0;

char mat[][] = new char[r][c];

String enc = "";

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

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

if (k != l) {

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

} else {

mat[j][i] = 'X';

}

}

}

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

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

enc += mat[i][j];

}

}

return enc;

}

String decode(String encmsg, int depth) throws Exception {

int r = depth;

int l = encmsg.length();

int c = l / depth;

int k = 0;

char mat[][] = new char[r][c];

String dec = "";

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

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

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

}

}

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

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

dec += mat[j][i];

}

}

return dec;

}

}

class railFenceCipher {

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

railfenceCipherHelper rf = new railfenceCipherHelper();

String msg, enc, dec;

msg = "INFORMATION SECURITY";

int depth = 2;

enc = rf.encode(msg, depth);

dec = rf.decode(enc, depth);

System.out.println("Simulating Railfence Cipher\n-------------------------");

System.out.println("Input Message : " + msg);

System.out.println("Encrypted Message : " + enc);

System.out.printf("Decrypted Message : " + dec);

}

}

**OUTPUT:**

Simulating Railfence Cipher

-------------------------

Input Message : INFORMATION SECURITY

Encrypted Message : IFRAINSCRTNOMTO EUIY

Decrypted Message : INFORMATION SECURITY

**RESULT:**

Thus the java program for Rail Fence Transposition Technique has been

implemented and the output verified successfully.

**b) Columnar Transformation Technique**

**AIM:**

To implement a program for encryption and decryption by using row and

column transformation technique.

**ALGORITHM:**

1. Consider the plain text hello world, and let us apply the simple columnar

transposition technique as shown below

|  |  |  |  |
| --- | --- | --- | --- |
| h | e | l | l |
| o | w | o | r |
| l | d |  |  |

2. The plain text characters are placed horizontally and the cipher text is

created with vertical format as: holewdlo lr.

3. Now, the receiver has to use the same table to decrypt the cipher text to

plain text.

**PROGRAM:**

**TransCipher.java**

import java.util.\*;

class TransCipher {

public static void main(String args[]) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the plain text");

String pl = sc.nextLine();

sc.close();

String s = "";

int start = 0;

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

if (pl.charAt(i) == ' ') {

s = s + pl.substring(start, i);

start = i + 1;

}

}

s = s + pl.substring(start);

System.out.print(s);

System.out.println();

// end of space deletion

int k = s.length();

int l = 0;

int col = 4;

int row = s.length() / col;

char ch[][] = new char[row][col];

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

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

if (l < k) {

ch[i][j] = s.charAt(l);

l++;

} else {

ch[i][j] = '#';

}

}

}

// arranged in matrix

char trans[][] = new char[col][row];

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

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

trans[j][i] = ch[i][j];

}

}

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

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

System.out.print(trans[i][j]);

}

}

// display

System.out.println();

}

}

**OUTPUT:**

Enter the plain text

information security

informationsecurity

irienmocfanuotsr

**RESULT:**

Thus the java program for Row and Column Transposition Technique has been implemented and the output verified successfully.

1. Implement any two Symmetric algorithms.