Experiment No. 1

Implementation of a product cipher using Substitution and Transposition

Course Outcome [CSL602.1]: Implement classical encryption techniques

Aim: Design and Implementation of a product cipher using Substitution and Transposition

Objectives:

• To understand the encryption and decryption fundamentals.

• To understand the concepts of the product cipher.

Outcomes: The learner will be able to

• Understand the principles and practices of cryptographic techniques

Hardware / Software Required: C/C++/JAVA/python

Theory:

Substitution cipher is a method of encryption by which units of plaintext are replaced with ciphertext according to a regular system; the "units" may be single letters (the most common), pairs of letters, triplets of letters, mixtures of the above, and so forth. The receiver deciphers the text by performing an inverse substitution.

Transposition cipher is a method of encryption by which the positions held by units of plaintext (which are commonly characters or groups of characters) are shifted according to a regular system, so that the ciphertext constitutes a permutation of the plaintext. That is, the order of the units is changed.

Substitution ciphers can be compared with Transposition ciphers. In a transposition cipher, the units of the plaintext are rearranged in a different and usually quite complex order, but the units themselves are left unchanged. By contrast, in a substitution cipher, the units of the plaintext are retained in the same sequence in the ciphertext, but the units themselves are altered.

1. Caesar Cipher: In cryptography, a Caesar cipher, also known as a Caesar's cipher, the shift cipher, Caesar's code or Caesar shift, is one of the simplest and most widely known encryption techniques.

It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. For example, with a shift of 3, A would be

replaced by D, B would become E, and so on. The method is named after Julius Caesar, who used it to communicate with his generals.

Example:

1.The transformation can be represented by aligning two alphabets; the cipher alphabet is the plain alphabet rotated left or right by some number of positions. For instance, here is a Caesar cipher using a left rotation of three places (the shift parameter, here 3, is used as the key):

Plain: ABCDEFGHIJKLMNOPQRSTUVWXYZ

Cipher: DEFGHIJKLMNOPQRSTUVWXYZABC

2.Rail Fence Transposition:

The rail fence cipher (sometimes called zigzag cipher) is a transposition cipher that jumbles up the order of the letters of a message using a basic algorithm.

The rail fence cipher works by writing your message on alternate lines across the page, and then reading off each line in turn.

For example, let's consider the plaintext "This is a secret message".

To encode this message we will first write over two lines (the "rails of the fence") as follows:

Note that all white spaces have been removed from the plain text.

The ciphertext is then read off by writing the top row first, followed by the bottom row:

Implementation :

```
package anproduct;
import java.io.*;
public class Anproduct {
public static void main(String[] args)throws IOException
{
System.out.println("Enter choice");
System.out.append("1.Encryption\n2.Decryption");
BufferedReader obj = new BufferedReader(new InputStreamReader(System.in));
```

```
int ch = Integer.parseInt(obj.readLine());
if(ch==1)
{
enc e = new enc();
e.enc();
}
else if(ch==2)
{
dec d = new dec();
d.dec();
}
else
{
System.out.println("Invalid Choice");
}
}
}
class enc
{
public void enc()throws IOException
{
BufferedReader obj = new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter IP:");
String ip = obj.readLine();
System.out.println("Enter Key 1:");
int k1 = Integer.parseInt(obj.readLine());
System.out.println("Enter Key 2:");
int k2 = Integer.parseInt(obj.readLine());
String op = "";
```

```
op = additive(ip,k1);
op = tansposition(op,k2);
System.out.println(op);
}
String additive(String ip, int k){
StringBuilder s = new StringBuilder();
int len = ip.length();
char temp;
int t1;
for(int i=0;i<len;i++)
{
temp = ip.charAt(i);
if(Character.isUpperCase(temp))
{
t1 = (int)temp - (int)'A';
t1 = (t1 + k)%26;
t1 = t1 + (int)'A';
temp = (char)t1;
s.append(temp);
}
else if(Character.isLowerCase(temp))
{
t1 = (int)temp - (int)'a';
t1 = (t1 + k)%26;
t1 = t1 + (int)'a';
temp = (char)t1;
s.append(temp);
}
else
```

```
{
s.append(temp);
}
}
String op = s.toString();
return op;
}
String tansposition(String ip,int m_row)
{
char op[][]=new char[100][100];
int len = ip.length();
String op2="";
int i1,i2,i;
//calculate columns from rows
int m_col = (int)Math.ceil((float)len/m_row);
for(i=0,i1=0,i2=0;i<len;i++)
{
op[i2][i1]=ip.charAt(i);
i2++;
if(i2==m_row)
{
i2=0;
i1++;
}
}
for(i1=0;i1<m_row;i1++)
{
for(i2=0;i2<m_col;i2++)
{
```

```
op2 = op2+op[i1][i2];
}
}
return (op2);
}
}
class dec
{
public void dec()throws IOException
{
BufferedReader obj = new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter IP:");
String ip = obj.readLine();
System.out.println("Enter Key 1:");
int k1 = Integer.parseInt(obj.readLine());
System.out.println("Enter Key 2:");
int k2 = Integer.parseInt(obj.readLine());
String op = "";
op = additive(ip,k1);
op = tansposition(op,k2);
System.out.println(op);
}
String additive(String ip, int k){
StringBuilder s = new StringBuilder();
int len = ip.length();
char temp;
int t1;
for(int i=0;i<len;i++)
{
```

```
temp = ip.charAt(i);
if(Character.isUpperCase(temp))
{
t1 = (int)temp - (int)'A';
t1 = (t1 - k +26)%26;
t1 = t1 + (int)'A';
temp = (char)t1;
s.append(temp);
}
else if(Character.isLowerCase(temp))
{
t1 = (int)temp - (int)'a';
t1 = (t1 - k+26)%26;
t1 = t1 + (int)'a';
temp = (char)t1;
s.append(temp);
}
else
{
s.append(temp);
}
}
String op = s.toString();
return op;
}
String tansposition(String ip,int m_row)
{
char op[][]=new char[100][100];
int len = ip.length();
```

```
String op2="";
int i1,i2,i;
//calculate columns from rows
int m_col = (int)Math.ceil((float)len/m_row);
for(i=0,i1=0,i2=0;i<len;i++)
{
op[i1][i2]=ip.charAt(i);
i2++;
if(i2==m_col)
{
i2=0;
i1++;
}
}
for(i1=0;i1<m_col;i1++)
{
for(i2=0;i2<m_row;i2++)
{
op2 = op2+op[i2][i1];
}
}
return (op2);
}
}
```

Output:

Enter choice

1.Encryption

2.Decryption1

Enter IP:

Ankur Mhatre

Enter Key 1:

1

Enter Key 2:

5

B soNfli vb su

Enter choice

1.Encryption

2.Decryption2

Enter IP:

B soNfli vb su

Enter Key 1:

1

Enter Key 2:

5

Ankur Mhatre

Conclusion : A product cipher is a composite of two or more elementary ciphers with the goal of producing a cipher which is more secure that any of the individual components. In product cipher substitution and transposition are applied to create confusion and diffusion in the text message.