

LABORATORY MANUAL

INFORMATION AND NETWORK SECURITY

SUBJECT CODE: 2170709

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

B.E. 7th SEMESTER

NAME:
ENROLLMENT NO:
BATCH NO:
YEAR:

Amiraj College of Engineering and Technology,

Nr.Tata Nano Plant, Khoraj, Sanand, Ahmedabad.

COLLEGE OF ENGINEERING &	R TECHNOLOGY
Amiraj College of Engineering	and Technology,
Nr.Tata Nano Plant, Khoraj, Sana	nd, Ahmedabad.
<u>CERTIFICATE</u>	
This is to certify that Mr. / Ms Of class Enrolment No	has
Satisfactorily completed the course in by the Gujarat Technological University for Yea Science and Enaineerina in the Academic year	as ar (B.E.) semester of Computer
Date of Submission:-	
Faculty Name and Signature Paras Narkhede	Head of Department Foram Patel



COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

B.E. 7th SEMESTER

SUBJECT: INOFRMATION AND NETWORK SECURITY

SUBJECT CODE: 2170709

List Of Experiments

Sr. No.	Experiments	Date of Performance	Date of submission	Sign	Remarks
1	Implement Caesar cipher encryption- decryption.				
2	Implement Playfair cipher encryption- decryption.				
3	Implement Polyalphabetic cipher encryption-decryption.				
4	Implement Hill cipher encryption- decryption.				
5	Implementation of Transposition cipher.				

6	To implement Simple DES or AES.		
7	Implement Diffi-Hellmen Key exchange Method.		
8	Implement RSA encryption- decryption algorithm.		
9	Perform various encryption- decryption techniques with cryptool.		
10	Study and use the Wireshark for the various network protocols.		

PRACTICAL-1

OBJECTIVE:

Implement CAESER cipher encryption- decryption

THEORY:

The Caesar cipher is one of the earliest known and simplest ciphers. It is a type of substitution cipher in which each letter in the plaintext is 'shifted' a certain number of places down the alphabet. For example, with a shift of 1, A would be replaced by B, B would become C, and so on. The method is named after Julius Caesar, who apparently used it to communicate with his generals. The Caesar cipher involves replacing each letter of the alphabet with the letter standing three places further down the alphabet.

For example,

plain: meet me after the toga party cipher: PHHW PH DIWHU WKH WRJD SDUWB

Note that the alphabet is wrapped around, so that the letter following Z is A. We can define the transformation by listing all possibilities, as follows:

plain: a b c d e f g h i j k l m n o p q r s t u v w x y z cipher: D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

Let us assign a numerical equivalent to each letter:

a	b	с	d	e	f	g	h	i	j	k	1	m
0	1	2	3	4	5	6	7	8	9	10	11	12
n	0	р	q	r	S	t	u	V	W	Х	У	Z
13	14	15	16	17	18	10	20	21	22	22	24	25

Then the algorithm can be expressed as follows. For each plaintext letter, substitute the cipher text letter C₂:

$$C = E(3, p) = (p + 3) \mod 26$$

A shift may be of any amount, so that the general Caesar algorithm is

 $C = E(k, p) = (p + k) \mod 26$

Where takes on a value in the range 1 to 25. The decryption algorithm is simply

 $p = D(k, C) = (C - k) \mod 26$

RSULTS / OBSERVATIONS:

```
#include<stdio.h>
#include<conio.h>
void main(){
        char pt[50], ct[50], dt[50];
       int key, len, i;
       clrscr();
        printf("Enter message to be encrypted : ");
        scanf("%s", &pt);
        len = strlen(pt);
        again :
        printf("\nEnter Key : ");
        scanf("%d", &key);
        if(key > 26){
               printf("\nInvalid key. Please enter valid key.\n");
               goto again;
        for(i = 0; i < len; i++){
               ct[i] = pt[i] + key;
               if(ct[i] > 122)
                       ct[i] = ct[i] - 26;
        }
        ct[i] = \langle 0';
        printf("\nEncrypted message is, %s", ct);
        for(i = 0; i < len; i++) {
               dt[i] = ct[i] - key;
               if(dt[i] < 97)
                        dt[i] = dt[i] + 26;
        }
        dt[i] = ' 0';
        printf("\n\nDecrypted message is, %s", dt);
        getch();
```

}

Output:-

🛗 DOSBox 0.74, Cpu sj	peed: max 100% cycle	es, Frameskip (), Program:	тс	
Enter message to be	e encrypted : vru	ındagandh i			
Enter Key : 5					
Encrypted message i	is, awzsiflfsimm				
Decrypted message i	is, vrundagandhi_	-			

PRACTICAL-2

OBJECTIVE :

Implement Playfair cipher encryption-decryption.

THEORY:

The Playfair cipher was the first practical digraph substitution cipher. The scheme was invented in 1854 by Charles Wheatstone, but was named after Lord Playfair who promoted the use of the cipher.

The technique encrypts pairs of letters (digraphs), instead of single letters as in the simple substitution cipher. The Playfair is significantly harder to break since the frequency analysis used for simple substitution ciphers does not work with it. Frequency analysis can still be undertaken, but on the 25*25=625 possible digraphs rather than the 25 possible monographs. Frequency analysis thus requires much more ciphertext in order to work.

The best-known multiple-letter encryption cipher is the Playfair, which treats digrams in the plaintext as single units and translates these units into ciphertext digrams. The Playfair algorithm is based on the use of a 5×5 matrix of letters constructed using a keyword. Here is an example, solved by Lord Peter Wimsey in Dorothy Sayers's *Have His Carcase:*

Μ	0	Ν	Α	R
С	Η	Y	В	D
E	F	G	I/J	K
L	Р	Q	S	Т
U	V	W	Х	Ζ

In this case, the keyword is *monarchy*. The matrix is constructed by filling in the letters of the keyword (minus duplicates) from left to right and from top to bottom, and then filling in the remainder of the matrix with the remaining letters in alphabetic order. The letters I and J count as one letter. Plaintext is encrypted two letters at a time, according to the following rules:

1. Repeating plaintext letters that are in the same pair are separated with a filler letter, such as x, so that balloon would be treated as ba lx lo on.

- 2. Two plaintext letters that fall in the same row of the matrix are each replaced by the letter to the right, with the first element of the row circularly following the last. For example, ar is encrypted as RM.
- 3. Two plaintext letters that fall in the same column are each replaced by the letter beneath, with the top element of the column circularly following the last. For example, mu is encrypted as CM.
- Otherwise, each plaintext letter in a pair is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter. Thus, hs becomes BP and ea becomes IM (or JM, as the encipherer wishes).

Assume one wants to encrypt the digraph OR. There are three general cases:

An example encryption, "we are discovered, save yourself" using the key square shown at the beginning of this section:

```
plaintext: wearediscoveredsaveyourselfx
```

```
ciphertext: ugrmkcsxhmufmkbtoxgcmvatluiv
```

RSULTS / OBSERVATIONS:

#include<stdio.h>
#include<conio.h>

void main(){

```
int n,i,j,k,temp,len,r1,r2,c1,c2, flag, index = 0, defaultValue = 97, count = 0;
char pt[50], ct[50], dt[50], key[10], keyMatrix[5][5], comparedValue, tempText[26];
clrscr();
printf("Enter message to be encrypted : ");
scanf("%s", &pt);
n=strlen(pt);
len=n;
printf("\nEnter Keyword : ");
scanf("%s", &key);
for(i = 0; i < strlen(key); i++){
       if(key[i] == 'i'){
               key[i] = 'j';
        }
}
for(i = 0; i < 5; i++){
       for(j = 0; j < 5; j + +){
               if(index < strlen(key)){
                       comparedValue = key[index];
                       index++;
                }else{
                       if(defaultValue == 105)
                               defaultValue = 106;
                       }
                       comparedValue = defaultValue;
                       defaultValue++;
                ł
               flag = 0;
               for(k = 0; k < \text{count}; k++){
                       if(comparedValue == tempText[k]){
                               flag = 1;
                               break;
                       }
               if (flag != 0){
                 j--;
                 if(j < 0){
                       i--;
```

```
i = 4;
                  }
                }else{
                  keyMatrix[i][j] = comparedValue;
                 tempText[count] = keyMatrix[i][j];
                  count++;
                }
        }
printf("\nKeymatrix is, \n\n");
for(i = 0; i < 5; i++){
       for(j = 0; j < 5; j + +){
                printf("%c ", keyMatrix[i][j]);
       printf("\n");
}
for(i = 0; i < n; i = (i+2))
       if(pt[i] == 'i')
                pt[i] = 'j';
       if(pt[i+1] == 'i')
                pt[i+1] = 'j';
       if(pt[i] == pt[i+1]){
                temp = pt[i+1];
                pt[i+1] = 'x';
                len++;
                for(j = (n-1); j > (i+1); j--)
                        pt[j+1] = pt[j];
                pt[j+1] = temp;
        }
if((len % 2) != 0){
       pt[len] = 'z';
       len++;
}
for(i = 0; i < len; i = (i+2))
       for(j = 0; j < 5; j + +){
                for(k = 0; k < 5; k++){
                        if(pt[i] == keyMatrix[j][k]){
                                r1 = j;
                                c1 = k;
                        if(pt[i+1] == keyMatrix[j][k]){
                                r^2 = j;
                                c2 = k;
                        }
                }
```

```
if(r1 == r2){
               c1++;
               c2++;
               if(c1 > 4)
                       c1 = 0;
               if(c2 > 4)
                       c2 = 0;
               ct[i] = keyMatrix[r1][c1];
               ct[i+1] = keyMatrix[r2][c2];
        else if(c1 == c2)
               r1++;
               r2++;
               if(r1 > 4)
                       r1 = 0;
               if(r_2 > 4)
                       r^2 = 0;
               ct[i] = keyMatrix[r1][c1];
               ct[i+1] = keyMatrix[r2][c2];
        }else{
               ct[i] = keyMatrix[r1][c2];
               ct[i+1] = keyMatrix[r2][c1];
        }
ł
printf("\nRearranged message is, ");
for(i = 0; i < len; i++){
       printf("%c",pt[i]);
}
printf("\n\nEncrypted message is, ");
for(i = 0; i < len; i++)
       printf("%c",ct[i]);
}
for(i = 0; i < len; i = (i+2))
       for(j = 0; j < 5; j + +){
               for(k = 0; k < 5; k++){
                       if(ct[i] == keyMatrix[j][k]){
                               r1 = j;
                               c1 = k;
                       }
                       if(ct[i+1] == keyMatrix[j][k]){
                               r^2 = j;
                               c2 = k;
                       }
                }
        }
       if(r1 == r2){
```

```
c1--;
               c2--;
               if(c1 < 0)
                       c1 = 4;
               if(c2 < 0)
                       c2 = 4;
               dt[i] = keyMatrix[r1][c1];
               dt[i+1] = keyMatrix[r2][c2];
        else if(c1 == c2)
               r1--;
               r2--;
               if(r1 < 0)
                       r1 = 4;
               if(r2 < 0)
                       r^2 = 4;
               dt[i] = keyMatrix[r1][c1];
               dt[i+1] = keyMatrix[r2][c2];
        }else{
               dt[i] = keyMatrix[r1][c2];
               dt[i+1] = keyMatrix[r2][c1];
        }
}
printf("\n\nDecrypted message is, ");
for(i = 0; i < len; i++){
       printf("%c",dt[i]);
}
getch();
```

Output:-

}

INFORMATION AND NETWORK SECURITY (2170709)

🚟 DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program:	тс	
Enter message to be encrypted : vrunda		
Enter Keyword : monarchy		
Keymatrix is,		
monar chybd efgjk lpqst uvwxz		
Rearranged message is, vrunda		
Encrypted message is, zowmbr		
Decrypted message is, vrunda		

PRACTICAL-3

OBJECTIVE :

Implement Polyalphabetic (Vigenere) cipher encryption-decryption.

THEORY:

The Vigenère cipher, was invented by a Frenchman, Blaise de Vigenère in the 16th century. It is a polyalphabetic cipher because it uses two or more cipher alphabets to encrypt the data. In other words, the letters in the Vigenère cipher are shifted by different amounts, normally done using a word or phrase as the encryption key.

Unlike the mono-alphabetic ciphers, polyalphabetic ciphers are not susceptible to frequency analysis, as more than one letter in the plaintext can be represented by a single letter in the encryption. One of the main problems with simple substitution ciphers is that they are so vulnerable to *frequency analysis*.

Given a sufficiently large cipher text, it can easily be broken by mapping the frequency of its letters to the know frequencies of, say, English text. Therefore, to make ciphers more secure, cryptographers have long been interested in developing enciphering techniques that are immune to frequency analysis. One of the most common approaches is to suppress the normal frequency data by using more than one alphabet to encrypt the message.

A *polyalphabetic substitution cipher* involves the use of two or more cipher alphabets. Instead of there being a one-to-one relationship between each letter and its substitute, there is a one-to-many relationship between each letter and its substitutes.

THE VIGENÈRE SQUARE:

Blaise de Vigenère developed a square to help encode messages. Reading along each row, you can see that it is a really a series of Caesar ciphers the first has a shift of 1, the second a shift of 2 and

so.

	A	в	С	D	E	F	G	н		J	ĸ	L	M	N	0	P	Q	R	S	т	U	V	vv	×	Y	z
A	A	в	С	D	E	F	G	н	1	J	к	L	M	Ζ	0	Р	Q	R	s	т	U	\sim	w.	×	Y	Z
в	в	C	D	E	F	G	н	1	J	ĸ	L	M	N	0	P	Q	R	S	т	U	\sim	\sim	×	Y	Z	A
C	С	D	E	F	G	н	1	J	ĸ	L	M	N	0	P	Q	R	s	т	U	\sim	\sim	×	Y	Z	A	в
D	D	E	F	G	н	1	J	ĸ	L	M	N	0	Р	Q	R	S	т	U	\sim	\sim	×	Y	Z	A	в	С
E	E	F	G	н	1	J	ĸ	L	M	Ζ	0	P	0	R	0	т	U	<	Ş	×	Y	Z	A	в	С	D
F	F	G	н	1	J	ĸ	L	M	Ν	0	Р	Ø	R	0	Т	U	<	~	×	Y	Z	A	в	C	D	E
G	G	н	1	J	ĸ	L	M	Z	0	P	0	R	S	т	U	<	~	×	Y	Z	A	в	C	D	E	F
н	н	1	J	ĸ	L	M	N	0	P	Ø	R	ŝ	т	C	>	\sim	×	Y	Ν	A	в	С		E	F	G
	1	J	ĸ	L	M	N	0	P	Ø	R	S	Т	U	>	≷	×	Y	Z	4	в	0	D	E	F	G	н
J	J	ĸ	L	M	N	0	P	Q	R	S	т	υ	\sim	ž	×	Y	Z	A	в	C	D	E	F	G	н	1
ĸ	ĸ	L	M	Ν	0	P	Q	R	S	т	U	$\mathbf{>}$	\sim	×	Y	Z	A	в	C	D	E	F	G	н	1	J
L	L	M	Ν	0	P	Q	R	S	Т	U	\sim	Š	\times	Y	Z	<u>A</u>	в	C	D	E	F	G	н	1	J	ĸ
M	M	N	0	Р	Q	R	S	т	υ	>	\sim	×	Y	Ν	<	в	С	D	E	F	G	н	1	J	ĸ	L.
N	N	0	P	Q	R	S	т	U	\sim	\sim	\times	\mathbf{Y}	Z	<u> </u>	в	C	D	E	F	G	н	1	J	ĸ	L	M
0	0	P	Q	R	S	т	U	\sim	\sim	\times	\mathbf{Y}	Z	<u> </u>	в	C	D	E	F	G	н	1	J	ĸ	L	M	N
P	P	Q	R	S	т	U	\sim	\sim	\times	\mathbf{Y}	Z	<u> </u>	в	С	D	E	F	G	н	1	J	ĸ	L	M	N	0
Q	Q	R	S	т	U	\sim	\sim	×	Y	Z	A	в	С	D	E	F	G	н	1	J	к	L	M	N	0	P
R	R	S	Т	U	\sim	\sim	×	Y	Z	A	в	C	D	E	F	G	н	1	J	ĸ	L	M	N	0	P	Q
S	S	Т	U	\sim	\sim	×	Y	Z	A	в	C	D	E	F	G	н	1	J	ĸ	L	M	N	\circ	P	Q	R
T	т	U	\sim	\sim	×	Y	Z	A	в	C	D	E	F	G	н	1	J	ĸ	L	M	N	0	P	Q	R	S
U	U	\sim	w	×	Y	Z	A	в	С	D	E	F	G	н	1	J	ĸ	L	M	N	0	P	Q	R	S	т
V	\sim	\sim	\times	Y	Z	A	в	C		E	F	G	н	1	J	ĸ	L	M	N	0	P	Q	R	S	Т	U
vv	\sim	\times	Y	Z	A	в	C	D	E	F	G	н	1	J	ĸ	L	M	N	0	P	Q	R	S	Т	U	D
×	\times	Y	Z	A	в	C	D	E	F	G	н	1	J	ĸ	L	M	N	0	Р	Q	R	S	T	U	\mathbf{v}	C
Y	Y	Z	A	в	C	D	E	F	G	н	1	J	ĸ	L	M	N	0	P	Q	R	S	т	U	\sim	\sim	в,
Z	Z		в	C	D	E	F	G	н	1	J	ĸ	L	M	N	0	P	Q	R	S	Т	U		W	×	Y

Vigenère cipher uses this table in conjunction with a key to encipher a message. So, if we were to encode a message using the key COUNTON, we write it as many times as necessary above our message. To find the encryption, we take the letter from the intersection of the Key letter row, and the Plaintext letter column.

Key	С	0	U	N	Т	0	N	С	0	U	Ν	Т	0	Ν
Plaintext	V	Ι	G	Е	Ν	Е	R	Е	С	Ι	Ρ	Н	Е	R
Encryption	Х	W	Α	R	G	S	Е	G	Q	С	С	Α	S	Е

To decipher the message, the recipient needs to write out the key above the cipher text and reverse the process. The maths behind the Vigenère cipher can be written as follows:

To encrypt a message: $Ca = Ma + Kb \pmod{26}$

To decrypt a message: $Ma = Ca - Kb \pmod{26}$

(Where C = Code, M = Message, K = Key, and where a = the ath character of the message bounded

by the message, and b is the bth character of the Key bounded by the length of the key.)

RSULTS / OBSERVATIONS:

```
#include<conio.h>
#include<stdio.h>
void main(){
       char pt[50], o[26], ct[50], dt[50], k, key[15], inversekey[15];
       int i, j = 0, ptlen, keylen;
       clrscr();
       for(i = 0; i < 26; i + +)
               o[i] = i + 97;
        printf("Enter message to be encrypted : ");
       scanf("%s", &pt);
        ptlen = strlen(pt);
       printf("\nEnter Key : ");
       scanf("%s", &key);
       keylen = strlen(key);
       for(i = 0; i < ptlen; i++)
               for(k = 0; k < 26; k++){
                       if(pt[i] == o[k])
                       break;
                ł
               ct[i] = key[j] + k;
               if(ct[i] > 122)
                       ct[i] = ct[i] - 26;
               if(j > (keylen - 2))
                       i = 0;
               else
                       j++;
        }
       ct[i] = \langle 0';
       printf("\nEncrypted message is, %s", ct);
       i = 0;
       for(i = 0; i < keylen; i++){
               for(k = 0; k < 26; k++){
                       if(key[i] == o[k])
                       break;
               inversekey[i] = 123 - k;
               if(inversekey[i] > 122)
                       inversekey[i] = inversekey[i] - 26;
        }
```

```
inversekey[i] = '\0';
for(i = 0; i < ptlen; i++){
       for(k = 0; k < 26; k++){
                if(ct[i] == o[k])
                break;
        }
       if((inversekey[j] + k) > 122)
                dt[i] = (inversekey[j] + k) - 26;
       else
               dt[i] = inversekey[j] + k;
       if(j > (keylen - 2))
                j = 0;
       else
                j++;
}
dt[i] = ' 0';
printf("\n\nDecrypted message is, %s", dt);
getch();
```

Output:-

}



PRACTICAL-4

OBJECTIVE :

Implement Hill cipher encryption-decryption.

THEORY:

THE HILL Cipher is an encryption algorithm takes **m** successive plaintext letters and substitutes for them **m** cipher text letters. The substitution is determined by **m** linear equations in which each character is assigned a numerical value ($\mathbf{a} = 0, \mathbf{b} = 1... \mathbf{z} = 25$). For **m** =3, the system can be described as;

 $c1 = (k11p1 + k12p2 + k13p3) \mod 26$ $c2 = (k21p1 + k22p2 + k23p3) \mod 26$ $c3 = (k31p1 + k32p2 + k33p3) \mod 26$

This can be expressed in terms of row vectors and matrices:

$$(c_1 \ c_2 \ c_3) = (p \ p_2 \ p_3) \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix} \mod 26$$

 $\mathbf{C} = \mathbf{PK} \mod 26$

Where **C** and **P** are row vectors of length 3 representing the plaintext and cipher text, and **K** is a 3 x 3 matrix representing the encryption key. Operations are performed mod 26.

RESULTS / OBSERVATIONS:

```
#include<stdio.h>
#include<conio.h>
void main()
ł
       char s[26]={'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z'};
       char p[100],ency[100],decy[100];
      int
1,8,21, {21,12,8};
      clrscr();
       printf(" The Key is=\n ");
       for (i=0;i<3;i++)
       {
             for(j=0;j<3;j++)
                     printf(" %d ",a[i][j]);
              printf("\n");
       printf("Enter The plaintext=\n");
       scanf("%s",p);
       length=strlen(p);
       for (i=0;i<length;i++)
       {
             for (j=0; j<26; j++)
                     if (s[j]==p[i])
                            num[i]=j;
              }
       for (i=0;i<3;i++)
             c[i]=0;
             for (j=0;j<3;j++)
               c[i]=c[i]+(a[i][j]*num[j]);
       }
       c[i] = 0';
       printf("The Encryptrd Message is=\n");
       for (i=0;i<3;i++)
             ency[i]=s[c[i]%26];
       ency[i]='0';
       puts(ency);
       printf(" The Inverse Key is=\n ");
```

INFORMATION AND NETWORK SECURITY (2170709)

```
for (i=0;i<3;i++)
{
       for(j=0;j<3;j++)
               printf(" %d ",b[i][j]);
        printf("\n");
for (i=0;i<length;i++)
{
       for (j=0;j<26;j++)
        {
               if (s[j]==ency[i])
                       num[i]=j;
        }
}
for (i=0;i<3;i++)
{
       c[i]=0;
       for (j=0; j<3; j++)
         c[i]=c[i]+(b[i][j]*num[j]);
}
c[i]='\0';
printf("The Decryptrd Message is=\n");
for (i=0;i<3;i++)
       decy[i]=s[c[i]%26];
decy[i]='\0';
puts(decy);
getch();
```

Output :-

}

INFORMATION AND NETWORK SECURITY (2170709)

CSE 7TH Semester

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program:	тс	
The Keu is=		
6 24 1		
13 16 10		
20 17 15		
Enter The plaintext=		
pay		
The Encryptrd Message is=		
ktk		
The Inverse Key is=		
8 5 10		
21 8 21		
21 12 8		
The Decryptrd Message is=		
рау		
_		

PRACTICAL-5

OBJECTIVE :

Implementation of Transposition cipher.

THEORY:

Transposition cipher, simple data encryption scheme in which plaintext characters are shifted in some regular pattern to form cipher text. In manual systems transpositions are generally carried out with the aid of an easily remembered mnemonic. For example, a popular schoolboy cipher is the "rail fence," in which letters of the plaintext are written alternating between rows and the rows are then read sequentially to give the cipher. In a depth-two rail fence (two rows) the message WE ARE DISCOVERED SAVE YOURSELF would be written;

WAEICVRDAEOREF E RDSOEESVYUSL

OI

WAEICVRDAEOREFERDSOEESVYUSL.

Simple frequency counts on the cipher text would reveal to the cryptanalyst that letters occur with precisely the same frequency in the cipher as in an average plaintext and, hence, that a simple rearrangement of the letters is probable.

The rail fence is the simplest example of a class of transposition ciphers, known as route ciphers, which enjoyed considerable popularity in the early history of cryptology. In general, the elements of the plaintext (usually single letters) are written in a prearranged order (route) into a geometric array (matrix)—typically a rectangle—agreed upon in advance by the transmitter and receiver and then read off by following another prescribed route through the matrix to produce the cipher. The key in a route cipher consists of keeping secret the geometric array, the starting point, and the routes. Clearly both the matrix and the routes can be much more complex than in this example; but even so, they provide little security. One form of transposition (permutation) that was widely used depends on an easily remembered key word for identifying the route in which the columns of a rectangular matrix are to be read. For example, using the key word AUTHOR and ordering the columns by the lexicographic order of the letters in the key word

A	. U 6	Т 5	Н 2	03	R 4
V	VΕ	A	R	E	D
1.1	I S	C	0	V	Е
F	εE	D	S	A	V
E	ĽΥ	0	U	R	S
I E	ΕL	F	A	В	C

yields the cipher

WIREEROSUAEVARBDEVSCACDOFESEYL.

In decrypting a route cipher, the receiver enters the cipher text symbols into the agreed-upon matrix according to the encryption route and then reads the plaintext according to the original order of entry. A significant improvement in crypto-security can be achieved by re-encrypting the cipher obtained from one transposition with another transposition. Because the result (product) of two transpositions is also a transposition, the effect of multiple transpositions is to define a complex route in the matrix, which in itself would be difficult to describe by any simple mnemonic. In the same class also fall systems that make use of perforated cardboard matrices called grilles;

descriptions of such systems can be found in older books on cryptography. In contemporary cryptography, transpositions serve principally as one of several encryption steps in forming a compound or product cipher.

RSULTS / OBSERVATIONS:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
       int i,length,j,k,l,len1,len2,temp1,temp2,q;
       char
a[100],text1[100],text2[100],encrypt[100],decrypttext1[100],decrypttext2[100],decrypt[100];
       clrscr();
       printf("Enter The PlainText=\n");
       gets(a);
       length=strlen(a);
       j=0;
       k=0;
       for (i=0;i<length;i++)
       {
               if (i%2==0)
               {
                       text1[j]=a[i];
                       j++;
               }
               else
               {
                       text2[k]=a[i];
                       k++;
       for (i=0;i<j;i++)
               encrypt[i]=text1[i];
       len1=i;
       for (l=0;l<k;l++)
       ł
               encrypt[i]=text2[l];
               i++;
       len2=l;
       encrypt[i] = \langle 0';
       printf("The Encrypted Message is=\n");
```

```
printf("%s\n",encrypt);
       for (i=0;i<len1;i++)
       ł
               decrypttext1[i]=encrypt[i];
       decrypttext1[i]='0';
       for (j=0;j<len2;j++)
       {
               decrypttext2[j]=encrypt[i];
               i++;
       decrypttext2[j]='\0';
       q=0;
       printf("Decrypted Message is=\n");
       for (i=0;i<strlen(encrypt);i++)
       {
          if (i<=len1)
          {
          printf("%c",decrypttext1[i]);
           }
           if (i<=len2)
           {
          printf("%c",decrypttext2[i]);
           }
       }
       decrypt[q]='0';
getch();
```



ł

INFORMATION AND NETWORK SECURITY (2170709)

CSE 7TH Semester



PRACTICAL-6

OBJECTIVE :

To implement Simple DES.

THEORY:

The **Data Encryption Standard** is a previously predominant symmetric-key algorithm for the encryption of electronic data. It was highly influential in the advancement of modern cryptography in the academic world. Developed in the early 1970s at IBM and based on an earlier design by Horst Feistel, the algorithm was submitted to the National Bureau of Standards (NBS) following the agency's invitation to propose a candidate for the protection of sensitive, unclassified electronic government data. In 1976, after consultation with the National Security Agency (NSA), the NBS eventually selected a slightly modified version, which was published as an official Federal Information Processing Standard (FIPS) for the United States in 1977. The publication of an NSA-approved encryption standard simultaneously resulted in its quick international adoption and widespread academic scrutiny. Controversies arose out of classified design elements, a relatively short key length of the symmetric-key block cipher design, and the involvement of the NSA, nourishing suspicions about a backdoor. The intense academic scrutiny the algorithm received over time led to the modern understanding of block ciphers and their cryptanalysis.

DES FEATURES:

- Block size = 64 bits
- Key size = 56 bits (in reality, 64 bits, but 8 are used as parity-check bits for error control)
- Number of rounds = 16
- 16 intermediary keys, each 48 bits



KEY LENGTH IN DES:

- In the DES specification, the key length is 64 bit.
- 8 bytes; in each byte, the 8th bit is a parity-check bit.



Each parity-check bit is the XOR of the previous 7 bits.



PROCEDURE:

In this excercize you can use the **DES**-component to encrypt an arbitrary text entered in the **Input message**-component on the left side. The resulting encrypted text is displayed in the **Output message**component on the right side after hitting the Play-button. The **DES**-component works on binary values, i.e. bytes. Thus, the inputed text is first converted to bytes with the **Message decoder**-component. With the current settings, it is interpreted as ASCII. The resulting bytes are then encrypted with **DES**, yielding another sequence of bytes. These bytes are then simply printed as hexadecimal values with the help of the **Message encoder**-component. Note that you can also decrypt messages with this template. To do so, you first copy the encrypted hexadecimal values to the **Input message**. Then you have to change the following:

- Set *Input format* of the **Message decoder** to *Hexadecimal*
- Set *Action* of **DES** to *Decrpyt*;
- Set *Format* of the **Message encoder** to *Text* and the *encoding* to *ASCII*.

RESULTS / OBSERVATIONS:

Step 1: Enter the Plaintext

INFORMATION AND NETWORK SECURITY (2170709)

CSE 7TH Semester



Step 2 : Enter the Key Value

Key Entry: DES (ECB)										
Enter the key using hexadecimal Key length: 64 bits (effectively	characters (09, AF). 56 bits)									
11 25 69 02 36 58 5	11 25 69 02 36 58 56 45									
Encrypt	Decrypt	Cancel								

Step 3 : Generated Encrypted Text (Cipher Text)

CrypTool 1.4.31 Beta 6b [VS2008] - DES (ECB) encryption of <cry-unnamed6>, key <11 25 69 02 36 💷 💷 🔤</cry-unnamed6>	3
le <u>E</u> dit <u>V</u> iew En <u>c</u> rypt/Decrypt Digital Signatures/ <u>P</u> KI <u>I</u> ndiv. Procedures <u>A</u> nalysis <u>O</u> ptions <u>W</u> indow <u>H</u> elp	
H DES (ECB) encryption of <cry-unnamed6>, key <11 25 69 02 36 58 56 45></cry-unnamed6>	
$ \begin{array}{ c c c c c c c c c c c c c c c c c c c$	
ess F1 to obtain help. L:4 C:14 P:56 OVR NUM	

Step 4: For Decryption Process Enter Cipher Text and the same key value

INFORMATION AND NETWORK SECURITY (2170709)

CSE 7TH Semester

CrypTool 1.4.31 Beta 6b [VS2008]	- Unnamed8					
<u>File Edit View Encrypt/Decrypt</u>	Digital Signatures/ <u>P</u> KI	Indiv. Procedures	<u>A</u> nalysis	<u>O</u> ptions	<u>W</u> indow	<u>H</u> elp
□≥≤€₩₩	J 🕾 🤋 💦 👘					
C [#] Unnamed8						
ACCASSOCIELISSANNRA ,(EIE)æTzñDÓŒ_>-DC2	₩ĸ₽yq ı∆U:s ı¦Gu sıs W-İ!{òŸĔû	SOIT: GAIN; /~ SO +1	vµj so x			Ţ
😚 des 📾 💷 🔀 😚 de	S 🗗 🗆 🔀					
Press F1 to obtain help.		L:1 C:65 P:65			NUM	

Key Entry: DES (ECB)		×
Enter the key using hexadecimal chara Key length: 64 bits (effectively 56 bits	acters (09, AF).	
11 25 69 02 36 58 56 48	5	6
Encrypt	Decrypt	<u>C</u> ancel

Step 5 : Generated Plain Text

CrypTool 1.4.31 Beta 6b [VS2008] - [[DES (ECB) decryption of <unnamed8>, key <11 25 69</unnamed8>	02 36 58 🕒 💷 💌
GI Eile Edit View Encrypt/Decrypt Help Help	nt Digital Signatures/ <u>P</u> KI Indiv. Procedures <u>A</u> nalys	is <u>O</u> ptions <u>W</u> indow
	<u>\$</u> 723 ? №	
00000000 1 7 65 6C 63 6F 6 00000011 72 6C 64 20 6F 6 00000022 68 79 20 61 6E 6 00000033 63 75 72 69 74 7	D 65 20 74 6F 20 74 68 65 20 57 6F 6 20 43 72 79 70 74 6F 67 72 61 70 4 20 4E 65 74 77 6F 72 6B 20 53 65 9 00 00 00 00 00 00 00	Welcome to the Wo rld of Cryptograp hy and Network Se curity
Press F1 to obtain help.	L:1 C:1 P:1	OVR NUM

PRACTICAL-7

OBJECTIVE :

Implement Diffi-Hellmen Key exchange Method.

THEORY:

first public-key type scheme proposed

- By Diffie & Hellman in 1976 along with the exposition of public key concepts
- Note: now know that Williamson (UK CESG) secretly proposed the concept in 1970 is a practical method for public exchange of a secret key
- Used in a number of commercial products
- Public-key distribution scheme cannot be used to exchange an arbitrary message Rather it can establish a common key

known only to the two participants

• Value of key depends on the participants (and their private and public key information) based on exponentiation in a finite (Galois) field (modulo a prime or a polynomial)

Program:

```
/* This program calculates the Key for two persons
using the Diffie-Hellman Key exchange algorithm */
#include<stdio.h>
#include<math.h>
// Power function to return value of a ^ b mod 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);
}
//Driver program
int main() {
    longlongint P, G, x, a, y, b, ka, kb;
    // Both the persons will be agreed upon the
       // public keys G and P
    P = 23; // A prime number P is taken
    printf("The value of P : %lld\n", P);
    G = 9; // A primitve root for P, G is taken
    printf("The value of G : lld\n, G;
```



```
// Alice will choose the private key a
    a = 4; // a is the chosen private key
   printf("The private key a for Alice : %lld\n", a);
    x = power(G, a, P); // gets the generated key
    // Bob will choose the private key b
   b = 3; // b is the chosen private key
   printf("The private key b for Bob : %lld\n\n", b);
    y = power(G, b, P); // gets the generated key
    // Generating the secret key after the exchange
       // of keys
    ka = power(y, a, P); // Secret key for Alice
    kb = power(x, b, P); // Secret key for Bob
   printf("Secret key for the Alice is : %lld\n", ka);
   printf("Secret Key for the Bob is : %lld\n", kb);
   return 0;
Result:
```

The value of P : 23

}

The value of G:9

The private key a for Alice : 4

The private key b for Bob : 3

Secret key for the Alice is : 9

Secret Key for the Bob is : 9

PRACTICAL-8

OBJECTIVE :

Implement RSA encryption-decryption algorithm.

THEORY:

by Rivest, Shamir & Adleman of MIT in 1977

- best known & widely used public-key scheme
- based on exponentiation in a finite (Galois) field
- over integers modulo a prime
- exponentiation takes O((log n)3) operations (easy)
- uses large integers (eg. 1024 bits)
- security due to cost of factoring large numbers
- factorization takes O(e log n log log n) operations (hard)to encrypt a message M the sender:

RSA Example - Key Setup

- 1. Select primes: p=17 & q=11
- 2. Compute *n* = *pq* =17 x 11=187
- 3. Compute $\phi(n) = (p-1)(q-1) = 16 \ge 10 = 160$
- 4. Select e: gcd(e,160)=1; choose *e*=7
- 5. Determine d: $de=1 \mod 160$ and d < 160
- Value is d=23 since 23x7=161= 10x160+1
- 6. Publish public key $PU=\{7,187\}$
- 7. Keep secret private key PR={23,187}

RESULTS / OBSERVATIONS:

Step 1: Enter Plain Text to Decrypt

CrypTool 1.4.31 Beta 6b [VS2008] -	cry2.org					
<u>File Edit View Encrypt/Decrypt</u>	Digital Signatures/ <u>P</u> KI	Indiv. Procedures	<u>A</u> nalysis	<u>Options</u>	<u>W</u> indow	<u>H</u> elp
	3 TH 🤋 院 📂					
G ^a cry2.org						
Welcome to the World of Infor	rmation and Network 3	Security				
Press F1 to obtain help.		L:1 C:57 P:57			NUM	

Step 2 : Selection of Key

Selection of a key Choose the recip	for RSA encry	ption of <cry2< th=""><th>l.org></th><th></th><th>×</th></cry2<>	l.org>		×						
Last name SideChannelAt	First name Bob	Key type RSA-512	Key identifier PIN=1234	Created 06.07.2006 15:21:34	Internal ID no.						
i Note: Here only names are displayed, which have an RSA key.											
Display encrypt	ption time				<u>C</u> ancel						

Step 3: Encrypted Text (Cipher Text)

C Cryp	pTool 1.4.31 Be	eta 6b	[VS200	8] - R	SA en	crypt	ion c	of <c< th=""><th>ry2.</th><th>org</th><th>> for</th><th><bc< th=""><th>b Si</th><th>deCh</th><th>annel</th><th>Attac</th><th>k></th><th></th><th></th><th></th></bc<></th></c<>	ry2.	org	> for	<bc< th=""><th>b Si</th><th>deCh</th><th>annel</th><th>Attac</th><th>k></th><th></th><th></th><th></th></bc<>	b Si	deCh	annel	Attac	k>			
<u>F</u> ile <u>f</u>	<u>E</u> dit <u>V</u> iew E	n <u>c</u> rypt	/Decryp	ot D	igital	Signa	ature	s/ <u>P</u> K	II	Indiv	z. Pro	oced	ures	An	alysis	Ор	tions	Wi	ndow	<u>H</u> elp
	s 🛋 🖪 🕾	*	B 6	8 d		2	<u> </u> N2													
	🚰 RSA encryp	tion of	f < cry2.	org>	for <e< td=""><td>Bob S</td><td>ideC</td><td>hann</td><td>nelA</td><td>ttac</td><td>k></td><td></td><td></td><td></td><td></td><td></td><td>-</td><td></td><td>—×</td><td></td></e<>	Bob S	ideC	hann	nelA	ttac	k>						-		— ×	
	000000000 00000000E 00000001C 00000002A 000000038	8F 46 43 1 9A 1 6B	97 A6 87 7D E9 D8 EB 24 2F 85	81 4E 54 C6 D8	7D 1 36 71 D9 A3 1	F5 6 99 4 71 6 1A 2 D3 1	E E E E E E E E E E E E E E E E E E E	SE E 21 E 22 2 45 C 2C	34 33 27 E	F7 D8 98 17	58 AA 9F 0F	1E 60 D5 31	CB 37 16 36	D5 C8 17 76	·F C ·k	} .}N6 Tq .\$. on . N! E	X .	7. 	
																			1	
Press F1	to obtain help	o.				_	_			L:1	1 C:1	L P:1					OVR		NUM	

Step 4: Decrypt Cipher Text to Plain Text

CrypTool 1.4.31 Beta 6b [VS2008] - Unnamed5	
<u>File Edit View Encrypt/Decrypt Digital Signatures/PKI Indiv. Procedures Analysis Options Window Help</u>	
Cir Unnamed5	
-¦}ãon′÷X (\$S ËŐF‡}N6™N!®Ø*'7ÈCéØTqqn SIX ″ŸŐ SYN(⊒IB šë\$ÆÙ SUB .EÎ (⊒IBS) 16vk/Ø£ ⊋ Ó IDS ,	
Press F1 to obtain help. L:1 C:65 P:65 NUM	1

Step 5 : Decrypt Text Using Key

Last name	First name	Key type	Key identifier	Created	Internal ID no.					
SideUhannelAt	Вор	HSA-512	PIN=1234	05.07.2005 15:21:34	1152179494					
Note: Only PSEs containing an RSA key are shown. PIN code:										

Step 6: Generated Plain Text

J C	урТо	ol 1	.4.31 B	eta 6t	o [VS	2008	8] - R	SA d	ecry	ptior	n of •	<unr< th=""><th>name</th><th>d5></th><th></th><th></th><th></th><th></th><th></th><th>-</th><th></th><th>- 0</th><th></th></unr<>	name	d5>						-		- 0	
<u>F</u> ile	Edit	: ⊻ ∽l	iew I	En <u>c</u> ryp	ot/De	ecryp	et E	Digita	al Sig	natu	res/	<u>p</u> kī	Indi	v. Pr	oced	ures	<u>A</u> n	alysis	<u>O</u> pti	ons	<u>W</u> in	dow	Help
	1		H	8 36				5 4	9	81	~~		_	_	_	_	_	_	_	_	_	_	_
ſ	8 1	RSA	decryp	otion o	of < L	Jnna	med	5>														×	D
			000 00E 01C 02A 038	57 20 72 74 00	65 57 6D 77 00	6C 6F 6F 6F 00	63 72 74 72 00	6F 6C 69 6B 00	6D 64 6F 20 00	65 20 6E 53 00	20 6F 20 65	74 66 61 63	6F 20 6E 75	20 49 64 72	74 6E 20 69	68 66 4E 74	65 6F 65 79	Ve V rm tw	lcom orld atio ork	e t of n a Sec	o th Inf nd M urit	ie o Ve y	
L	F1 40	aht	nin hel											1.0	1 0.1					VB		DUIDA]
ress	F1 to	obt	ain hei	р.						_			L:	1 C:	1 P:1					VR		MOM	

PRACTICAL-9

OBJECTIVE:

Perform various encryption-decryption techniques with Cryptool.

INTRODUCTION:

The Cryptool 2 is supplied free program with which each person installing the computer to experiment with encryption. Create their own provisions, to test the algorithms and study the results.

BASIC DESCRIPTION:

More specifically, a description of the basic steps are the following: Select an algorithm determines what kind of work will be performed (encryption - decryption), place the input data (text - file, picture , etc) appropriate type keys according to the algorithm , we form connections in the interface and the output connectors where the result will look and if everything is as it should be - if not, the program does not allow the completion of connections and not running - and then displayed after the execution result. The results obtained are thereafter studied, and compared or used in new encryption or decryption.

RESULTS / OBSERVATIONS: A. AES Encryption Technique

Step 1 : Enter Plain Text to Decrypt



Step 2 : Enter the Key Value

INFORMATION AND NETWORK SECURITY (2170709)

CSE 7TH Semester

Key Entry: Rijndael (AES)		×
Enter the key using hexadecimal cha	aracters (09, AF).	
Key length: 128 bits	•	
12 36 58 74 56 98 32 3	14 63 25 47 96 50 25 5	79 34
1		
<u>E</u> ncrypt	Decrypt	<u>C</u> ancel

Step 3 : Generated Cipher Text

С СгурТо	ol 1.4.31 Beta 6	ь [VS2008] -	Rijndael (AE	S) encryptic	on of <cry_13>, key -</cry_13>	<12 36 58 74 56 98 3	
<u>File</u> <u>E</u> dit	<u>V</u> iew En <u>c</u> ry	pt/Decrypt	Digital Sign	atures/ <u>P</u> KI	Indiv. Procedures	<u>A</u> nalysis <u>O</u> ptions <u>W</u>	<u>(</u> indow <u>H</u> elp
	- B - B	6 🖻 📾	8 Ti 1	· • • •			
Sr o	Crv 13						
W	🚰 Rijndael (Al	ES) encryptio	n of <cry_13< td=""><td>>, key <12</td><td>36 58 74 56 98 32 14 6</td><td>3 25 47 96 50 2</td><td></td></cry_13<>	>, key <12	36 58 74 56 98 32 14 6	3 25 47 96 50 2	
	00000000 0000000E 0000001C 0000002A 00000028	85 A3 7 B7 03 7 5A A9 0 48 89 7 4A 9C F	6 F3 B2 (5 51 4B 8 2B 5D 7 4A 57 (0 94 C3	56 9E 22 78 FB AD 98 E5 C0 0A 60 31 90 E2 08	74 42 A7 EC C BB AF 8D 01 D 01 BA 8B D5 2 F8 4E AA 89 A	7 6Bvf." 8 93uQKx D 1C Z+] C 46 H.wJW.`1 J	tBk
Press F1 to	obtain help.				L:1 C:1 P:1	OVR	NUM //

Step 4 : For Decryption use Cipher Text and Key

CrypTool 1.4.31 Beta 6b [VS2008] - Unnamed14 <u>Eile Edit View Encrypt/Decrypt Digital Signatures/PKI Indiv. Procedures Analysis</u> D D D D D D D D D D D D D D D D D D D	Options Window Help
Kit Crv 13 W Sit Unnamed14 Image: Sot Sit Sit Sit Sit Sit Sit Sit Sit Sit Si	
Press F1 to obtain help. L:2 C:17 P:66	NUM
Key Entry: Rijndael (AES)	

Key Entry: Rijndael (AES)								
Enter the key using hexadecimal characters (09, AF).								
Key length: 128 bits	-							
12 36 58 74 56 98 32	14 63 25 47 96 50 25 '	79 34						
Encrypt	Decrypt	Cancel						

Step 5 : Generated Plain Text

CrypTool 1.4.31 Beta 6b [VS2008] - Rijndael (AES)	decryption of <unnamed14>, key <12</unnamed14>	36 58 74 5
<u>File Edit View Encrypt/Decrypt</u> Digital Signate	ures/ <u>P</u> KI <u>I</u> ndiv. Procedures <u>A</u> nalysis	Options Window Help
	₩?	
Rijndael (AES) decryption of <unnamed14>, key</unnamed14>	/ <12 36 58 74 56 98 32 14 63 25 47 9	
000000000 ■7 65 6C 63 6F 6D 65 20 00000000E 20 57 6F 72 6C 64 20 6F 0000001C 74 6F 67 72 DC F4 65 C4 0000002Å 41 F0 81 6B 67 6C 52 48 00000038 14 26 C8 93 2E 19 CC 56 00000046 B7 6D EE 73 C9 BB DF FC	74 6F 20 74 68 65 Welcor 66 20 43 72 79 70 World 13 0C FE C2 3F 9C togr. 4B 88 B2 B6 AA 6B Akgl 03 E2 35 27 40 0B .& EC .m.s.	ne to the 1 of Cryp .e?. LRHKk V5'@.
Press F1 to obtain help.	L:1 C:1 P:1	OVR NUM

B. Public Key Infrastructure

Step 1: Open PKI for Key Generation

CrypTool 1.4.31 Beta 6b [VS2008]	- Cry_13	Microsof
File Edit View Encrypt/Decrypt	Digital Signatures/PKI Indiv. Procedures Analysis Options	Window Help
	PKI >	Generate/Import Keys
	Sign Document	Display/Export Keys
	Verify Signature	- Nu
	Extract Signature	ler & Foot
Cry_13	Signature Demonstration (Signature Generation)	
VVelcome to the World o	of Cryptography and Network security	
Generate (or import from PKCS#12) a	key pair for different asymm L:1 C:1 P:1	NUM

Step 2: Filled up User data

age	orithm			User data		
œ	BSA Bit length of RSA modulus:	1024	-	The key pair will be the name shown be protected by your F	a put in an encrypted PSE wit slow. The key pair will be PIN code.	th
0	DSA Bit length of DSA prime number:	1024	~	Last name: First name:	william Bob	
\sim	Elliptic curves			Key identifier (optional):	wim	
	Identifier (bit length and	prime239v1	-	PIN:		_
	carve parametery.			PIN verification:	[nnnn]	_
ne d Para	lomain parameter of the selected el ameters Value of the parameter	liptic curve will b	e shown be	low.	Bit len	
Para	Iomain parameter of the selected el ameters Value of the parameter	liptic curve will b	e shown be	low.	Bit len	

INFORMATION AND NETWORK SECURITY (2170709)



Step 3: Generating New Key Pair

CrypTool	×
i	The parameters chosen by you and the new key pair have been successfully saved. The assigned key identifier is '[william][Bob][RSA-1024][1472802036][wlm]'. Elapsed time while creating key pair: 21.839 seconds.
	ОК

Step 4 : Show Available Asymmetric Key Pairs

ailable Asymme	tric Key Pairs			-				
The list below shows the asymmetric key pairs that are available. Select the desired name by clicking its row with the left mouse button.								
Last name	First name	Key type	Key identifier	Created	Internal ID no.			
hello HybridEncrypti SideChannelAt	hi Bob Bob	RSA-1024 EC-prime239∨1 RSA-512	opt PIN=1234 PIN=1234	02.09.2016 11:19:22 09.05.2007 14:51:14 06.07.2006 15:21:34	1472795362 1178702474 1152179494			
Listed key types:		[
🔽 RSA key 🔽 DSA key	is	Sh	ow certificate	<u>Export P</u>	SE (PKCS#12)			



Public paramete	ers of: Bob william
Exponent:	177759871594987213848200618142632679837681863106686019714684 111168509046045854200241744066913005195457830823254098569872 473131253949844640342977193894470408207228245848535362556647
Modulus:	65537
Base for pre	sentation of numbers © <u>D</u> ecimal O <u>H</u> exadecimal Back

Step 6 : Certificate Generation

Certificate Data	
Version: 2 (X.509v3-1996)	
SubjectName: CN=Bob william [1472802036], DC=cryptool	
IssuerName: CN=CrypTool CA 2, DC=cryptool, DC=org	
SerialNumber: A2:5F:14:15:5B:51:61:8B	
Validity - NotBefore: Fri Sep 02 13:10:58 2016 (160902074058Z)	
NotAfter: Sat Sep 02 13:10:58 2017 (170902074058Z)	
Public Key Fingerprint: 8825 CAAD 89EE F855 D032 4038 9AAE 8883	
SubjectKey: Algorithm rsa (OID 2.5.8.1.1), Keysize =	
Public modulus (no. of bits = 1024):	
U FDZ3/219 542AADB1 5585/B10 1/2E5/46	
10 D5D50775 C/62A660 27361AA4 CC7CF7A6 20 D0D5D756 0054510C 40C1544D 02001072	
20 DUDED/F0 075AFIUC A0CISAOD 027017/3 20 ED7AD02E EA7012DC 0D0E2E27 16DE1207	
40 0B5A5F9A FAF96787 25FDBAAA 9D6B35C3	
50 5C851D29 36CF3B0F F0FF01DF 589BFB19	
60 AC9FA7BF 157416F4 410D7AA6 F7CFF0DD	
70 80FBA8C1 7F1FAC56 6F5FB3C7 F6002DE7	
Public exponent (no, of bits = 17):	
· · · · · · · · · · · · · · · · · · ·	
,	
Close	

PRACTICAL-10

OBJECTIVE:

Study and use the Wireshark for the various network protocols.

THEORY:

A packet sniffer, sometimes referred to as a network monitor or network analyzer, can be used by a network or system administrator to monitor and troubleshoot network traffic. Using the information captured by the packet sniffer an administrator can identify erroneous packets and use the data to pinpoint bottlenecks and help maintain efficient network data transmission. In its simple form a packet sniffer simply captures all of the packets of data that pass through a given network interface. By placing a packet sniffer on a network in promiscuous mode, a Malicious intruder can capture and analyze all of the network traffic. Wireshark is a network packet analyzer. A network packet analyzer will try to capture network packets and tries to display that packet data as detailed as possible. Download and install wireshark network analyzer.