SRI SIDDHARTHA INSTITUTE OF TECHNOLOGY, TUMAKURU.

(A Constituent College of Sri Siddhartha Academy of Higher Education, Agalakote, Tumakuru)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

LABORATORY MANUAL

for

B.E

VII SEMESTER

18CS705: CRYPTOGRAPHY & NETWORK SECURITY LAB

Prepared by:

1. Dr. Sujatha S R
Associate Professor
Dept of CSE, SSIT

2. Prof. Nirmala G
Assistant Professor
Dept of CSE, SSIT

3. Prof. Vishwas R
Assistant Professor
Dept of CSE, SSIT

PREFACE

Networking is a key area in the field of computers that deals with the physical connectivity of computers which is coordinated and monitored by a combination of special hardware’s and software’s. Cryptography is the study of secure communications techniques that allow only the sender and intended recipient of a message to view its contents on a network.

Cryptography techniques used to protect information are obtained from mathematical concepts and a set of rule-based calculations known as algorithms to convert messages in ways that make it hard to decode it. These algorithms are used for cryptographic key generation, digital signing, verification to protect data privacy, web browsing on internet and to protect confidential transactions. This manual is prepared for the lab course Cryptography and Network Security Lab of VII semester.

Dr. M Siddappa,
Professor & Head,
Dept. of CSE, SSIT

INDEX

VISION OF THE INSTITUTE

To carve technically competent, confident and socially responsible engineers.

MISSION OF THE INSTITUTE

- To impart fundamental knowledge in science and technology.
- To create a conducive ambience for better learning and to bring out creativity in the students.
- To instil managerial, entrepreneurial and soft skills.
- To evolve as trusted destination for quality technical education.
- Positive contribution to meet societal needs.
- To inculcate a spirit of enquiry, make learning perceptive and rational.

VISION OF THE DEPARTMENT

To craft professionally skilled engineers with research orientation, innovative insights and a passion for life-long learning to meet the needs of Industry and Society.

MISSION OF THE DEPARTMENT

M1: To offer need-based curriculum in collaboration with industry.

M2: To inculcate professional skills with innovative thinking to address societal problems of multidisciplinary nature.

M3: To provide a congenial environment to learn and exhibit soft skills.

M4: To promote research culture and the need for life-long learning.

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

PEO1: Excel in professional career and higher education by acquiring knowledge in mathematical, computing and engineering principles.

PEO2: Analyse societal problems and provide technically competent solutions.

PEO3: Possess academic excellence through innovative insights, soft skills and life-long learning.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: Demonstrate the uses of knowledge by writing programs and integrate them with hardware/software products in multidisciplinary environment.

PSO2: Participate in planning and implementation of solutions to cater the industry specific requirements.

PROGRAM OUTCOMES

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization for the solution of complex engineering problems

PO2: Problem analysis: Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for public health and safety, and cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools, including prediction and modeling to complex engineering activities, with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess Societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with the society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

GUIDELINES FOR CONTINUOUS INTERNAL EVALUATION AND SUGGESTED LAB RUBRICS

Laboratory Course Evaluation:

The distribution of marks for laboratory courses evaluation is shown in the following table:

Assessment tools			Marks	Total marks	Weightage for CO attainment
CIE	Lab Test1		10	50	50%
	Lab Test2		10		
	Weekly internal evaluation		30		
SEE	Lab End-term examination	Procedure write up	10	50	50%
		Experiment conduction and result analysis	30		
		Viva voce	10		
	Total Marks			100	100%

Table 1. Laboratory Course Evaluation

Students' performance in Laboratory course is evaluated using rubrics defined by the program. Rubrics for evaluating a lab test in programming lab are displayed in the following table.

Sl. No.	Evaluation Parameters	Excellent	Very Good	Good	Satisfactory
1	Writing Program **[Max. Marks 5]**	5 Marks	4 Marks	3 Marks	2 Marks
		Completeness of code, consistent variable naming and formatting, well commented, uses existing skills in new ways/learns new skills to solve the experimental problem.	Completeness of code, consistent variable naming and formatting, lack of comments, uses existing skills in new ways/learns new skills to solve the experimental problem.	Completeness of code, inconsistent variable naming and formatting, lacks clarity in commenting, uses existing skills to solve the experimental problem.	Lack of completeness of code, improper variable naming and not formatted, lacks comments, uses existing skills to partially solve the experimental problem.
2	Program Execution **[Max. Marks 5]**	5 Marks	4 Marks	3 Marks	1Marks
		Program is free of errors and output is well formatted. Demonstrates excellent problem solving and creativity skills.	Program is free of errors and output is well formatted. Demonstrates better problem solving and creativity skills.	Program is free of errors and output is not properly formatted. Demonstrates a clear understanding of the concepts relevant to the experiment.	Program contains few logical errors and output is not formatted. Demonstrates partial understanding of the concepts relevant to the experiment.
3	Program Modification and verify the results **[Max. Marks 5]**	5Marks	4Marks	3Marks	1 Marks
		Able to modify the changes specified and test the output.	Able to modify the changes specified with help could test the output.	Able to modify the changes specified with help could test the output.	Unable to modify the changes specified and unable to test the output.
4	Viva-Voce **[Max. Marks 5]**	5Marks	4Marks	3Marks	1 Marks
		Answers all the viva questions.	Answer 80% of the viva questions.	Answer 50% of the viva questions.	Couldn't answer viva questions properly.

Table 2. Rubrics for Lab Test Evaluation

Students' ability in developing programs, and testing the program, documenting the work done through lab record, etc are evaluated continuously during all lab sessions.

Sl. No.	**Evaluation Component**	**Excellent**	**Good**	**Fair**	**Satisfactory**
1	Writing Program [Max. Marks 6]	6 Marks	5 Marks	4 Marks	3 Marks
		Completeness of code, consistent variable naming and formatting, well commented, uses existing skills in new ways/learns new skills to solve the experimental problem.	Completeness of code, consistent variable naming and formatting, lack of comments, uses existing skills in new ways/learns new skills to solve the experimental problem.	Completeness of code, inconsistent variable naming and formatting, lacks clarity in commenting, uses existing skills to solve the experimental problem.	Lack of completeness of code, improper variable naming and not formatted, lacks comments, uses existing skills to partially to solve the experimental problem.
2	Program Execution [Max. Marks 6]	6 Marks	5 Marks	4-3 Marks	2-1 Marks
		Program is free of errors and output is well formatted. Demonstrates excellent problem solving and creativity skills.	Program is free of errors and output is well formatted. Demonstrates better problem solving and creativity skills.	Program is free of errors and output is not properly formatted. Demonstrates a clear understanding of the concepts relevant to the experiment.	Program contains few logical errors and output is not formatted. Demonstrates partial understanding of the concepts relevant to the experiment.
3	Program Testing and observation [Max. Marks 6]	6 Marks	5 Marks	4-3 Marks	2-1 Marks
		Able to give different inputs and record the output.	Able to give inputs and with help could record the output.	Able to give inputs and with help could record the output.	With help gives inputs and unable to record the output.
4	Lab report assessment [Max. Marks 6]	6 Marks	5 Marks	4-3 Marks	2-1 Marks
		Neatly written the record and regular to lab.	Neatly written the record, some data are missing in the record. Student is regular to lab.	Lack of neatness in the record, some data are missing in the record. Student is regular to lab..	Data are missing & not written the record neatly and irregular to lab
5	Viva [Max. Marks 6]	6 Marks	5 Marks	4-3 Marks	2-1 Marks
		Answers all the viva questions.	Answer 80% of the viva questions.	Answer 50% of the viva questions	Couldn't answer viva questions.

Table 3. Rubrics for Continuous Evaluation in Lab

Syllabus for the Academic Year – 2021 - 2022

Department: Computer Science and Engineering **Semester: VII**

Subject Name: Cryptography & Network Security Lab

Subject Code: 18CS705 **L-T-P-C: 0-0-2-1**

Course Objectives:

Sl. No	Course Objectives
1	Understand the concepts of different encryption algorithms like Caesar, Playfair, Hill cipher.
2	Get the overview of symmetric algorithm like DES.
3	Understand and implement asymmetric algorithm like RSA and Diffie-Hellman.
4	Develop Digital Signature Scheme for signing and verification of messages.

Course Outcomes:

Course outcome	Descriptions
CO1	Develop and execute various symmetric ciphers.
CO2	Implement various asymmetric encryption algorithms.
CO3	Implement Digital Signature algorithm for signing and verification of messages.

1.	Implement the following Substitution and Transposition techniques: a) Caesar Cipher b) Playfair Cipher c) Hill Cipher d) Rail fence Cipher e) Mono-alphabetic Cipher
2.	Implement the following algorithms: a) DES b) RSA c) Diffie-Hellman e) SHA-1
3.	Implement the Signature Scheme - Digital Signature Standard

CO -PO MAPPING:

	PO-1	**PO-2**	**PO-3**	**PO-4**	**PO-5**	**PO-6**	**PO-7**	**PO-8**	**PO-9**	**PO-10**	**PO-11**	**PO-12**
CO1	2	3										
CO2	2	3		2	3							3
CO3	2	3		2	2							3

LIST OF EXPERIMENTS

Sl. No	Experiment
1.	Caesar Cipher
2.	Playfair Cipher
3.	Hill Cipher
4.	Rail fence Cipher
5.	Mono-alphabetic Cipher
6.	DES Algorithm
7.	RSA Algorithm
8.	Diffie-Hellman Key Exchange Mechanism
9.	SHA-1 Algorithm
10.	Digital Signature Standard

1. CAESAR CIPHER

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
#include<math.h>
void main()
{
char plain[20], cipher[20];
int key, i, length, result;
printf("Enter the plain text\n");
scanf("%s",plain);
printf("Enter the key value\n");
scanf("%d",&key);
printf("Plain text is %s", plain);
printf("Encrypted text:\n");
for(i=0;length=strlen(plain);i<length;i++)
{
cipher[i]=plain[i]+key;
if(isupper(plain[i])&&(cipher[i]>'Z'))
cipher[i]=cipher[i]-26;
if(islower(plain[i])&&(cipher[i]>'z'))
cipher[i]=cipher[i]-26;
printf("%c",cipher[i]);
}
printf("After Decryption\n");
for(i=0;i<length;i++)
{
plain[i]=cipher[i]-key;
if(isupper(cipher[i])&&plain[i]<'A')
plain[i]=plain[i]+26;
if(islower(cipher[i])&&cipher[i]<'A')
plain[i]=plain[i]+26
printf("%c\n",plain[i]);
}
}
```

Output:

```
Enter the Plain Text:
ABCDE
Enter the key value:
3
Plain text is
ABCDE
Encrypted value is
D
E
F
G
H
After Decryption
A
B
C
D
E
```

2. PLAYFAIR CIPHER

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
int removerepeated(int size,int a[]);
int insertelementat(int position,int a[],int size);
main()
{
 int i,j,k,numstr[100],numcipher[100],numkey[100],lenkey,templen,tempkey[100],flag=-
1,size,cipherkey[5][5],lennumstr,row1,row2,col1,col2;
 char str[100],key[100];
 printf("Enter a string\n");
 gets(str);
 //converting entered string to Capital letters
 for(i=0,j=0;i<strlen(str);i++)
 {
  if(str[i]!=' ')
  {
   str[j]=toupper(str[i]);
   j++;
  }
 }
 str[j]='\0';
 printf("Entered String is %s\n",str);
 //Storing string in terms of ascii and to restore spaces I used -20
 size=strlen(str);
 for(i=0;i<size;i++)
 {
  if(str[i]!=' ')
  numstr[i]=str[i]-'A';
 }
```

```
lennumstr=i;
//Key processing
printf("Enter the key (Non repeated elements if possible)\n");
gets(key);
//converting entered key to Capital letters
for(i=0,j=0;i<strlen(key);i++)
{
 if(key[i]!=' ')
 {
  key[j]=toupper(key[i]);
  j++;
 }
}
key[j]='\0';
printf("%s\n",key);
//Storing key in terms of ascii
k=0;
for(i=0;i<strlen(key)+26;i++)
{
 if(i<strlen(key))
 {
  if(key[i]=='J')
  {
   flag=8;
   printf("%d",flag);
  }
      numkey[i]=key[i]-'A';
 }
 else
 {
   if(k!=9 && k!=flag)//Considering I=J and taking I in place of J except when J is there in key ignoring I
   {
          numkey[i]=k;
    }
    k++;
 }
}
templen=i;
lenkey=removerepeated(templen,numkey);
printf("Entered key converted according to Play Fair Cipher rule\n");
for(i=0;i<lenkey;i++)
{
   printf("%c",numkey[i]+'A');
}
printf("\n");
//Arranging the key in 5x5 grid
k=0;
for(i=0;i<5;i++)
{
 for(j=0;j<5;j++)
 {
  cipherkey[i][j]=numkey[k];
  k++;
 }
}
printf("Arranged key\n");
for(i=0;i<5;i++)
{
 for(j=0;j<5;j++)
 {
  printf("%c ",cipherkey[i][j]+'A');
 }
 printf("\n");
}
  //Message Processing
  for(i=0;i<lennumstr;i+=2)
  {
     if(numstr[i]==numstr[i+1])
     {
      insertelementat(i+1,numstr,lennumstr);
      lennumstr++;
     }
  }
  if(lennumstr%2!=0)
  {
   insertelementat(lennumstr,numstr,lennumstr);
   lennumstr++;
  }
  printf("Entered String/Message After Processing according to Play fair cipher rule\n");
  for(i=0;i<lennumstr;i++)
  {
   printf("%c",numstr[i]+'A');
  }
  for(k=0;k<lennumstr;k+=2)
  {
   for(i=0;i<5;i++)
   {
    for(j=0;j<5;j++)
    {
     if(numstr[k]==cipherkey[i][j])
   {
      row1=i;
      col1=j;
   }
   if(numstr[k+1]==cipherkey[i][j])
   {
      row2=i;
      col2=j;
   }
  }
 }
//Only change between Ecryption to decryption is changing + to -
//If negative add 5 to that row or column
if(row1==row2)
{
 col1=(col1-1)%5;
 col2=(col2-1)%5;
 if(col1<0)
 {
  col1=5+col1;
 }
 if(col2<0)
 {
  col2=5+col2;
 }
 numcipher[k]=cipherkey[row1][col1];
 numcipher[k+1]=cipherkey[row2][col2];
}
if(col1==col2)
{
     row1=(row1-1)%5;
     row2=(row2-1)%5;
      if(row1<0)
     {
      row1=5+row1;
     }
     if(row2<0)
     {
      row2=5+row2;
     }
     numcipher[k]=cipherkey[row1][col1];
     numcipher[k+1]=cipherkey[row2][col2];
    }
    if(row1!=row2&&col1!=col2)
    {
     numcipher[k]=cipherkey[row1][col2];
     numcipher[k+1]=cipherkey[row2][col1];
    }
   }
   printf("\nCipher Text is\n");
   for(i=0;i<lennumstr;i++)
   {
    if((numcipher[i]+'A')!='X')//Should remove extra 'X' which were created during Encryption
      printf("%c",numcipher[i]+'A');
   }
   printf("\n");
}
int removerepeated(int size,int a[])
{
 int i,j,k;
 for(i=0;i<size;i++)
  {
 for(j=i+1;j<size;)
 {
    if(a[i]==a[j])
    {
     for(k=j;k<size;k++)
     {
      a[k]=a[k+1];
     }
         size--;
        }
    else
    {
      j++;
     }
 }
 }
return(size);
}
int insertelementat(int position,int a[],int size)
{
       int i,insitem=23,temp[size+1];
    for(i=0;i<=size;i++)
        {
        if(i<position)
        {
            temp[i]=a[i];
        }
        if(i>position)
        {
         temp[i]=a[i-1];
        }
        if(i==position)
        {
            temp[i]=insitem;
        }
        }
        for(i=0;i<=size;i++)
        {
         a[i]=temp[i];
        }
}
```

3. HILL CIPHER

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int **matrixMultiply(int**a,int r1,int c1,int **b,int r2,int c2)
{
 int **resultMatrix;
 int i,j,k,r,c;
 r=r1;c=c2;
 resultMatrix=(int**)malloc(sizeof(int*)*r);
 for(i=0;i<r;i++)
  resultMatrix[i]=(int*)malloc(sizeof(int)*c);
 for(i=0;i<r;i++)
 {
  for(j=0;j<c;j++)
  {
   resultMatrix[i][j]=0;
   for(k=0;k<c1;k++)
    resultMatrix[i][j]+=a[i][k]*b[k][j];
  }
 }
 return resultMatrix;
}
void printMatrix(int**matrix,int r,int c)
{
 int i,j;
 for(i=0;i<r;i++)
 {
  for(j=0;j<c;j++)
   printf("%d ",matrix[i][j]);
  printf("\n");
 }
}


int plainTextToCipherText(char plainText[],int**matrix)
{
 int len,**plainTextMatrix,**resultMatrix,i,j;
 // The matrix will be of dimensions strlen(plainText) by strlen(plainText)
 char *cipherText;
 len=strlen(plainText);
 cipherText=(char*)malloc(sizeof(char)*1000);

 // plainTextMatrix should be of dimension strlen(plainText) by 1
 // allcating memory to plainTextMatrix
 plainTextMatrix=(int**)malloc(sizeof(int*)*len);
 for(i=0;i<len;i++)
  plainTextMatrix[i]=(int*)malloc(sizeof(int)*1);

 // populating the plainTextMatrix
 for(i=0;i<len;i++)
  for(j=0;j<1;j++)
   plainTextMatrix[i][j]=plainText[i]-'a';

 resultMatrix=matrixMultiply(matrix,len,len,plainTextMatrix,len,1);

 // taking mod 26 of each element of the result matrix
 for(i=0;i<len;i++)
  for(j=0;j<1;j++)
   resultMatrix[i][j]%=26;

 // Printing the cipher text
 printf("The cipher text is as follows : ");
 for(i=0;i<len;i++)
  for(j=0;j<1;j++)
   printf("%c",resultMatrix[i][j]+'a');
 printf("\n");
 //printMatrix(resultMatrix,len,1);
}

int main()
{
 int len,i,j,**matrix;
 char plainText[1000];
 printf("Enter the word to be encrypted : ");
 scanf(" %s",plainText);
 len=strlen(plainText);
 // allocating memory to matrix
 matrix=(int**)malloc(sizeof(int*)*len);
 for(i=0;i<len;i++)
  matrix[i]=(int*)malloc(sizeof(int)*len);
 printf("Enter the matrix of %d by %d to be used in encryption process:\n",len,len);
 for(i=0;i<len;i++)
  for(j=0;j<len;j++)
   scanf("%d",&matrix[i][j]);
 plainTextToCipherText(plainText,matrix);
 return 0;

}
```

Output:

```
Enter the word to be encrypted : ssit
Enter the matrix of 4 by 4 to be used in encryption process:
10 20 30 40
50 60 70 80
90 100 101 102
103 104 112 117

The cipher text is as follows : geeh
```

4. RAIL FENCE TECHNIQUE

```
#include<stdio.h>
#include<string.h>

void encryptMsg(char msg[], int key){
    int msgLen = strlen(msg), i, j, k = -1, row = 0, col = 0;
    char railMatrix[key][msgLen];

    for(i = 0; i < key; ++i)
        for(j = 0; j < msgLen; ++j)
            railMatrix[i][j] = '\n';

    for(i = 0; i < msgLen; ++i){
        railMatrix[row][col++] = msg[i];

        if(row == 0 || row == key-1)
            k= k * (-1);

        row = row + k;
    }

    printf("\nEncrypted Message: ");

    for(i = 0; i < key; ++i)
        for(j = 0; j < msgLen; ++j)
            if(railMatrix[i][j] != '\n')
                printf("%c", railMatrix[i][j]);
}

void decryptMsg(char enMsg[], int key){
```

```
int msgLen = strlen(enMsg), i, j, k = -1, row = 0, col = 0, m = 0;
char railMatrix[key][msgLen];

for(i = 0; i < key; ++i)
    for(j = 0; j < msgLen; ++j)
        railMatrix[i][j] = '\n';

for(i = 0; i < msgLen; ++i){
    railMatrix[row][col++] = '*';

    if(row == 0 || row == key-1)
        k= k * (-1);

    row = row + k;
}

for(i = 0; i < key; ++i)
    for(j = 0; j < msgLen; ++j)
        if(railMatrix[i][j] == '*')
            railMatrix[i][j] = enMsg[m++];

row = col = 0;
k = -1;

printf("\nDecrypted Message: ");

for(i = 0; i < msgLen; ++i){
    printf("%c", railMatrix[row][col++]);

    if(row==0||row==key-1)
```

```
            k= k*(-1);

        row = row + k;
    }
}

int main(){
    char msg[] = "Hello World";
    char enMsg[] = "Horel ollWd";
    int key = 3;

    printf("Original Message: %s", msg);

    encryptMsg(msg, key);
    decryptMsg(enMsg, key);

    return 0;
}
```

Output

```
Original Message: Hello World
Encrypted Message: Horel ollWd
Decrypted Message: Hello World
```

5. DES ALGORITHM

```
#include<stdio.h>
 #include<stdlib.h>
 #include<math.h>

 void sboxAccess(int[8][4][16],int[48],int*);
 void decimalToBinary(int,int*);

 int main()
{

    int sboxes[8][4][16] = {
         // S1
        { 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},
         //S2
         {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},
         //S3
         {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},
         //S4
         {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},
         //S5
         {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},
         //S6
         {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},
         //S7
         {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},
         //S8
         {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}
  };

  int i,j,k;

   for(k=0;k<8;k++)
     {
  for(i=0;i<4;i++)
{
for(j=0;j<16;j++)
{
printf("%d",sboxes[k][i][j]);
}
printf("\n");
}
printf("\n");
}

int input[48]=
{0,1,1,0,0,0,0,1,0,0,0,1,0,1,1,1,1,0,1,1,1,0,1,0,1,0,1,0,0,0,0,1,1,0,0,1,1,0,0,1,0,1,0,0,1,0};

int output[32];

sboxAccess(sboxes,input,output);

printf("\nS-box output");

for(i=0;i<32;i++)
{
if(i%4==0)
printf("\n");
printf("%d",output[i]);
}

printf("\n Permutted output");

int permutationTable[32]=
{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};
int permutedoutput[32];

for(i=0;i<32;i++)
```

```
{
permutedoutput[i]=output[permutationTable[i]-1];
if(i%4==0)
printf("\n");
printf("%d",permutedoutput[i]);
}

return 0;
}

void sboxAccess(int sboxes[8][4][16],int input[48],int output[32])
{
int i,j,k;
int numberInput[6],row,column,binaryVersion[4];

for(i=0;i<8;i++)
{
printf("%d:",i);
j=i*6;
for(k=0;k<6;k++)
{
numberInput[k]=input[j+k];
}

row=(numberInput[0]*pow(2,1))+(numberInput[5]*pow(2,0));

column=(numberInput[1]*pow(2,3))+(numberInput[2]*pow(2,2))+(numberInput[3]*pow(2,1))+(num
berInput[4]*pow(2,0));
printf("\n Number in sbox %d,%d,%d=%d\n",i,row,column,sboxes[i][row][column]);
decimalToBinary(sboxes[i][row][column],binaryVersion);
for(k=0;k<4;k++)
{
output[(i*4)+k]=binaryVersion[k];
}
}
}

void decimalToBinary(int number,int *binary)
{
int bin[4]={0,0,0,0};
int i=3;
if(number!=0){
while(number!=1)
{
bin[i--]=number%2;
number/=2;
}
bin[i]=number;
}
for(i=0;i<4;i++)
{
binary[i]=bin[i];
}

}
```

Output:

```
1441312151183106125907
0157414213110612119538
4114813621115129731050
1512824917511314100613

1518146113497213120510
3134715281412011069115
0147111041315812693215
1381013154211671205149

1009146315511312711428
1370934610285141211151
1364981530111212510147
1101306987415143115212

7131430691012851112415
1381156150347212110149
1069012117131513145284
3150610113894511127214

2124171011685315130149
1411212471315015103986
4211110137815912563014
1181271142136150910453

1211015926801334147511
1015427129561131401138
9141552812370410113116
4321295151011141760813

4112141508133129751061
1301174911014351221586
1411131237141015680592
6111381410795015142312
```

```
13284615111109314501127
11513810374125611101492
71141912142061013153 58
21147410813151290356 11

0:
 Number in sbox 0,0,12=5
1:
 Number in sbox 1,1,8=12
2:
 Number in sbox 2,0,15=8
3:
 Number in sbox 3,2,13=2
4:
 Number in sbox 4,2,4=10
5:
 Number in sbox 5,1,12=0
6:
 Number in sbox 6,3,2=13
7:
 Number in sbox 7,0,4=6

S-box output
0101
1100
1000
0010
1010
0000
1101
0110
 Permutted output
0000
0011
0101
1010
1000
0001
1011
0011
```

6. RSA Algorithm

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
long int e,d,n;
long int val[50];
char decode(long int ch)
{
 int i;
 long int temp=ch;
 for(i=1;i<d;i++)
 ch=(temp*ch)%n;
 return ch;
}
int gcd(long int a,long int b)
{
 long int temp;
 while(b!=0)
 {
  temp=b;
  b=a%b;
  a=temp;
 }
return a;
}
int prime(int a)
{
 int i;
 for(i=2;i<a;i++)
 {
  if((a%i)==0)
  return 0;
 }
```

```
return 1;
}
int encode(char ch)
{
 int i;
 long int temp;
 temp=ch;
 for(i=1;i<e;i++)
 temp=(temp*ch)%n;
 return temp;
}
int main()
{
int i;
long int p;
long int q,phi,c[50];
char text[50],ctext[50];
system("clear");
printf("\nEnter the text to be encoded\n");
scanf("%s",text);
do
{
p=rand()%30;
}while(!prime(p));
do
{
q=rand()%30;
}while(!prime(q));
n=p*q;
phi=(p-1)*(q-1);
printf("\np=%d\tq=%d\tn=%d\tphi=%d\n",p,q,n,phi);
do
{
e=rand()%phi;
}while(!gcd(e,phi));
do
{
d=rand()%phi;
}while(((d*e)%phi)!=1);
printf("\n***************Encoding Message***************\n");
sleep(3);
for(i=0;text[i]!='\0';i++)
val[i]=encode(text[i]);
val[i]=-999;
printf("Encode Message:\n");
for(i=0;val[i]!=-999;i++)
printf("%ld",val[i]);
printf("\n***************Decoding encrypted Message*******************\n");
sleep(3);
for(i=0;val[i]!=-999;i++)
ctext[i]=decode(val[i]);
ctext[i]='\0';
printf("Decoded message is:%s\n\n",ctext);
}
```

7. Diffie Hellman Key Exchange

```
#include<stdio.h>
long long int power(int a,int b,int mod)
{
 long long int t;
 if(b==1)
  return a;
 t=power(a,b/2,mod);
 if(b%2==0)
  return (t*t)%mod;
 else
  return (((t*t)%mod)*a)%mod;
}
long long int calculateKey(int a,int x,int n)
{
 return power(a,x,n);
}
int main()
{
 int n,g,x,a,y,b;
 // both the persons will be agreed upon the common n and g
 printf("Enter the value of n and g : ");
 scanf("%d%d",&n,&g);

 // first person will choose the x
 printf("Enter the value of x for the first person : ");
 scanf("%d",&x);
 a=power(g,x,n);

 // second person will choose the y
 printf("Enter the value of y for the second person : ");
 scanf("%d",&y);
 b=power(g,y,n);

 printf("key for the first person is : %lld\n",power(b,x,n));
 printf("key for the second person is : %lld\n",power(a,y,n));
 return 0;
}
```

8. SHA ALGORITHM

```
 #include<stdio.h>
#include<string.h>
#include<malloc.h>
#include<math.h>
#include<stdlib.h>
#define rotateleft(x,n) ((x<<n)|(x>>(32-n)))
#define rotateright(x,n) ((x>>n)|(x<<(32-n)))

void SHA1(unsigned char * str1)
{
    unsigned int h0,h1,h2,h3,h4,a,b,c,d,e,f,k,temp;
    int i,j,m;
    h0=0x67452301;
    h1=0xEFCDAB89;
    h2=0x98BADCFE;
    h3=0x10325476;
    h4=0xC3D2E1F0;
   unsigned char * str;
    str = (unsigned char *)malloc(strlen((const char *)str1)+100);
    strcpy((char *)str,(const char *)str1);
    int current_length = strlen((const char *)str);
    int original_length = current_length;
    str[current_length] = 0x80;
    str[current_length + 1]='\0';
    char ic = str[current_length];
    current_length++;
    int ib = current_length % 64;
    if(ib<56)
    ib=56-ib;
    else
    ib=120-ib;
    for(i=0;i<ib;i++)
    {
        str[current_length]=0x00;
        current_length++;
    }
    str[current_length+1]='\0';

    for(i=0;i<6;i++)
    {
        str[current_length]=0x0;
        current_length++;
    }
    str[current_length]=(original_length*8)/0x100 ;
    current_length++;
    str[current_length]=(original_length * 8) % 0x100;
    current_length++;
    str[current_length+i]='\0';
    int number_of_chunks = current_length/64;
    unsigned long int word[80];
    for(i=0;i<number_of_chunks;i++)
    {
        for(j=0;j<16;j++)
        {
            word[j] = str[i*64 + j*4 + 0] * 0x1000000 + str[i*64 + j*4 + 1] * 0x10000 + str[i*64 + j*4 + 2] * 0x100 +
str[i*64 + j*4 + 3];
        }
        for(j=16;j<80;j++)
        {
            word[j] = rotateleft((word[j-3]^word[j-8]^word[j-14]^word[j-16]),1);
        }
        a=h0;
        b=h1;
        c=h2;
        d=h3;
        e=h4;

        for(m=0;m<80;m++)
        {
            if(m<=19)
            {
                f=(b & c)|((~b) & d);
                k=0x5A827999;
            }
            else if(m<=39)
            {
                f=b^c^d;
                k=0x6ED9EBA1;
            }
            else if(m<=59)
            {
                f=(b & c)|(b & d)|(c & d);
                k=0x8F1BBCDC;
            }
            else
            {
                f=b^c^d;
                k=0xCA62C1D6;
            }
            temp=(rotateleft(a,5)+f+e+k+word[m])&0xFFFFFFFF;
            e=d;
            d=c;
            c=rotateleft(b,30);
            b=a;
            a=temp;
         }
         h0=h0+a;
          h1=h1+b;
         h2=h2+c;
```

```
          h3=h3+d;
          h4=h4+e;
       }
     printf("\n\n");
    printf("Hash: %x %x %x %x %x",h0, h1, h2, h3, h4);
    printf("\n\n");
}
  void main()
{
    SHA1((unsigned char *)"The quick brown fox jumps over the lazy dog");

}
```

<u>Output:</u>

```
Hash: d3f91 fdcc13de c5345a9d c2bf958b ca125bad
```

9. MILLER RABIN

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

long long mulmod(long long a, long long b, long long mod)

{

    long long x=0,y=a%mod;

    while(b>0)

    {

        if(b%2==1)

        {

            x=((x+y)%mod);

        }

        y=((y*2)%mod);

        b/=2;

    }

    return(x % mod);

}

long long modulo(long long base, long long exponent, long long mod)

{

    long long x = 1;

    long long y = base;

    while(exponent>0)

    {

        if(exponent%2==1)
```

```
            x=(x*y)%mod;

            y=(y*y)%mod;

        exponent=exponent/2;

    }

    return(x%mod);

}

int Miller(long long p,int iteration)

{


    int i;

    long long s;

    if(p<2)

    {

        return 0;

    }

    if((p!=2)&&(p%2==0))

    {

        return 0;

    }

     s=p-1;

    while((s%2)==0)

    {

        s/=2;
    }
    for(i=0;i<iteration;i++)
    {
        long long a=rand()%(p-1)+1,temp=s;
        long long mod=modulo(a,temp,p);
        while((temp!=p-1)&&(mod!=1)&&(mod!=p-1))
        {
            mod=mulmod(mod,mod,p);
            temp*=2;
        }
        if((mod!=p-1)&&(temp%2==0))
        {
            return 0;
        }
    }
    return 1;
}
int main()
{
    int iteration=5;
    long long num;
    printf("Enter integer to test primality: ");
    scanf("%lld",&num);
```

```
        if(Miller(num,iteration))

            printf("\n%lld is prime\n",num);

        else

            printf("\n%lld is not prime\n",num);

        return 0;

    }
```

<u>Output:</u>

```
Enter integer to test primality: 97

97 is prime


Enter integer to test primality: 93

93 is not prime
```

10. <u>Digital Signature Standard</u>

```
def euclid(m, n):
     if n == 0:
      return m
    else:
        r = m % n
        return euclid(n, r)
# Program to find
# Multiplicative inverse
def exteuclid(a, b):
  r1 = a
   r2 = b
    s1 = int(1)
    s2 = int(0)
    t1 = int(0)
    t2 = int(1)
    while r2 > 0:
       q = r1//r2
        r = r1-q * r2
        r1 = r2
        r2 = r
        s = s1-q * s2
        s1 = s2
        s2 = s
        t = t1-q * t2
        t1 = t2
        t2 = t
   if t1 < 0:
    t1 = t1 % a
   return (r1, t1)
 # Enter two large prime
# numbers p and q
p = 823
q = 953
n = p * q
Pn = (p-1)*(q-1)
# Generate encryption key
# in range 1<e<Pn
key = []
for i in range(2, Pn):
  gcd = euclid(Pn, i)
    if gcd == 1:
    key.append(i)
# Select an encryption key
# from the above list
e = int(313)
# Obtain inverse of
# encryption key in Z_Pn
r, d = exteuclid(Pn, e)
```

```
if r == 1:
   d = int(d)
   print("decryption key is: ", d)
  else:
  print("Multiplicative inverse for\
   the given encryption key does not \
    exist. Choose a different encryption key ")
  # Enter the message to be sent
M = 19070
# Signature is created by Alice
S = (M**d) % n
# Alice sends M and S both to Bob
# Bob generates message M1 using the
# signature S, Alice's public key e
# and product n.
M1 = (S**e) % n
# If M = M1 only then Bob accepts
# the message sent by Alice.
if M == M1:
 print("As M = M1, Accept the\ message sent by Alice")
else:
  print("As M not equal to M1,\ Do not accept the message\sent by Alice ")
```

Output:

```
./a.out
decryption key is:  160009 As M = M1, Accept the message sent by Alice
```