UNITED INSTITUTE OF TECHNOLOGY
COIMBATORE – 641020

Department of Computer Science and Engineering (Cyber Security)

CB3411 – CRYPTOGRAPHY AND CYBER SECURITY LABORATORY

(Regulation 2021)
Academic Year: 2023-2024 (Even Sem) Year/ Semester: II/ IV

Department of Computer Science and Engineering - Cyber Security

CB3411 - CRYPTOGRAPHY AND CYBER SECURITY LABORATORY

NAME:..... **ROLL NO:.....**

CLASS:..... **BRANCH:.....**

Certified bonafide record of work done by…….……

Place:

Date:

Staff In-Charge **Head of the Department**

University Register Number:……..

Submitted for the University Practical Examination held on…….……

INTERNAL EXAMINER **EXTERNAL EXAMINER**

UNITED INSTITUTE OF TECHNOLOGY, COIMBATORE

Department of Computer Science and Engineering (Cyber security)

INSTITUTION VISION:

Empowering students with competitive aspects of Engineering and Technology through Innovative Teaching-Learning, Applied Research, Nurturing their Career with Entrepreneurial Prospects and thereby moulding them to become good citizens with human values.

INSTITUTION MISSION:

- To inculcate students with knowledge in cutting edge technologies through innovative teaching-learning processes.
- To impart skills focusing on applied research-oriented learning.
- To build engineers specialized in technical skills and entrepreneurial skills.
- To develop great citizens with moral values confronting worldwide challenges.

DEPARTMENT VISION:

To provide quality technical training to employ them with technopreneur skills in a broad area of cyber threat/attack detection, analysis, prevention and forensics and become responsible citizens with human values.

DEPARTMENT MISSION:

- To equip students with state-of-the-art training and development in the broad field of cyber security.
- To impart knowledge as per present industrial requirements and prepare them for better employment.
- To cultivate and practice human values to make them good citizens to serve.
- To impart Entrepreneurial skills to become a better employer.

ANNA UNIVERSITY, CHENNAI

AFFILIATED INSTITUTIONS

B.E - COMPUTER SCIENCE AND ENGINEERING (Cyber security)

REGULATIONS – R2021

CHOICE BASED CREDIT SYSTEM

PROGRAMME OUTCOMES:

Engineering Graduates will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **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 the public health and safety, and the cultural, societal, and environmental considerations.
4. **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.
5. **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.
6. **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.
7. **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.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with 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.
11. **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.
12. **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

PROGRAM SPECIFIC OUTCOMES (PSOs):

Graduating students of Computer Science and Engineering from United Institute of Technology will have the ability to

PSO1: Exhibit design and programming skills to build and automate business solutions using cutting edge technologies.

PSO2: Strong theoretical foundation leading to excellence and excitement towards research, to provide elegant solutions to complex problems.

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs):

Bachelor of Computer Science and Engineering (Cyber security) curriculum is designed to prepare the graduates having attitude and knowledge to

1. Apply their technical competence in computer science to solve real world problems, with technical and people leadership.
2. Conduct cutting edge research and develop solutions on problems of social relevance.
3. Work in a business environment, exhibiting team skills, work ethics, adaptability and lifelong learning.

CO-PO Mapping and justifications

CO / PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	3	2	1	2	2	-	-	-	1	-	-	1
CO2	3	3	3	3	3	-	-	-	2	-	-	1
CO3	3	3	3	3	3	-	-	-	2	-	-	1
CO4	3	3	3	3	3	-	-	-	2	-	-	1
CO5	3	2	3	2	3	-	-	-	3	-	-	2

COURSE ARTICULATION MATRIX:

Contribution **1: Low** **2: Moderate** **3: High** **‘ - ’ Cannot Co-relate**

COs	**POs**	**JUSTIFICATIONS**
CO1	**PO1**	Applying engineering knowledge to cyber security concepts
	PO2	Applying problem analysis for the various encryption methods
	PO3	Design solution for model of OSI model and network security
	PO4	Design and apply research knowledge on stenography
	PO5	Demonstrate the experimental on perfect security models
	PO9	Get deep knowledge on information theory
	PO12	Need lifelong learning cryptosystem and cryptanalysis.
CO2	**PO1**	Applying Engineering knowledge to Symmetric cipher
	PO2	Applying Mathematical Knowledge on number theory
	PO3	Design solution for complex problem on DES
	PO4	Design and apply research knowledge on block cipher mode
	PO5	Demonstrate the experimental on Pseudorandom Number Generators and RC4
	PO9	Deep knowledge in Euclid‘s algorithm
	PO12	Need lifelong learning procedure on symmetric methods of cipher

CO3	**PO1**	Applying Engineering knowledge on Asymmetric cipher
	PO2	Applying Mathematical Knowledge for Factorization
	PO3	Design solution for complex problem on various Asymmetric methods.
	PO4	Design and apply research knowledge on various theorems
	PO5	Demonstrate the experimental on key distribution and management
	PO9	Deep knowledge in Diffie Hellman key exchange -– Elliptic curve arithmetic – Elliptic curve cryptography
	PO12	Need lifelong learning procedure for Asymmetric cryptography
CO4	**PO1**	Applying Engineering knowledge on integrity and authentication
	PO2	Applying Mathematical Knowledge on hash functions
	PO3	Design solution for complex problem on Digital signature and Authentication protocols
	PO4	Design and apply research knowledge on Digital Signature Scheme and ElGamal cryptosystem
	PO5	Demonstrate the experimental on Key management and distribution
	PO9	Deep knowledge in Distribution of public keys
	PO12	Need lifelong learning procedure for integrity and authentication algorithms
CO5	**PO1**	Applying Engineering knowledge on Cyber Crime and Information Security
	PO2	Applying Mathematical Knowledge on Cyber Crimes tools
	PO3	Design solution for complex problem on Password Cracking, Keyloggers, Spywares, SQL Injection
	PO4	Design and apply research knowledge on Network Access Control
	PO5	Demonstrate the experimental on Cloud Security
	PO9	Deep knowledge in Web Security
	PO12	Need lifelong learning procedure for cyber crimes and cyber security

INDEX

S.No	Name of the Program	Page No	Marks	Staff Signature
1(A)	Caesar Cipher			
1(B)	Playfair Cipher			
1(C)	Hill Cipher			
2	Rail fence – row & Column Transformation			
3	Data Encryption Standard(DES)			
4	AES Algorithm			
5	RSA Algorithm			
6	Diffiee-Hellman Algorithm			
7	MD5			
8	SHA-1			
9	Implement the Signature Scheme for Digital Signature Standard			

EX. NO: 1(A)

IMPLEMENTATION OF CAESAR CIPHER

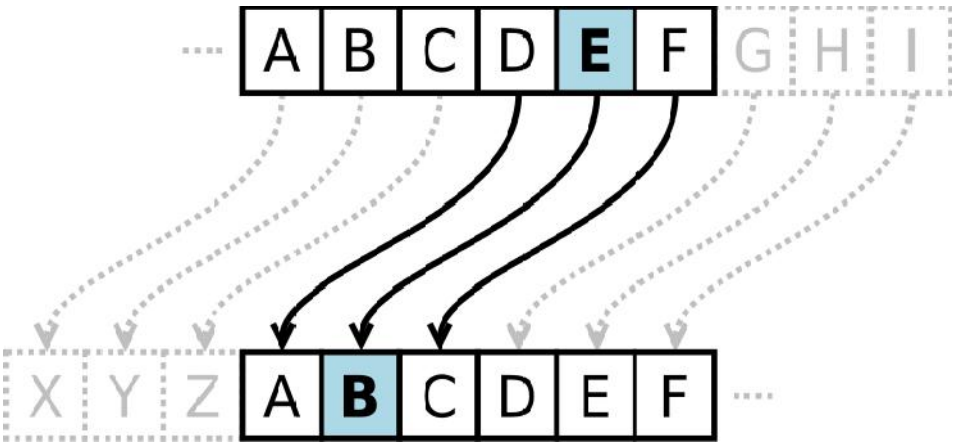
AIM:

To implement the simple substitution technique named Caesar cipher using C language.

DESCRIPTION:

To encrypt a message with a Caesar cipher, each letter in the message is changed using a simple rule: shift by three. Each letter is replaced by the letter three letters ahead in the alphabet. A becomes D, B becomes E, and so on. For the last letters, we can think of the alphabet as a circle and "wrap around". W becomes Z, X becomes A, Y bec cmes B, and Z becomes C. To change a message back, each letter is replaced by the one three before it.

EXAMPLE:



ALGORITHM:

STEP-1: Read the plain text from the user.

STEP-2: Read the key value from the user.

STEP-3: If the key is positive then encrypt the text by adding the key with each character in the plain text.

STEP-4: Else subtract the key from the plain text.

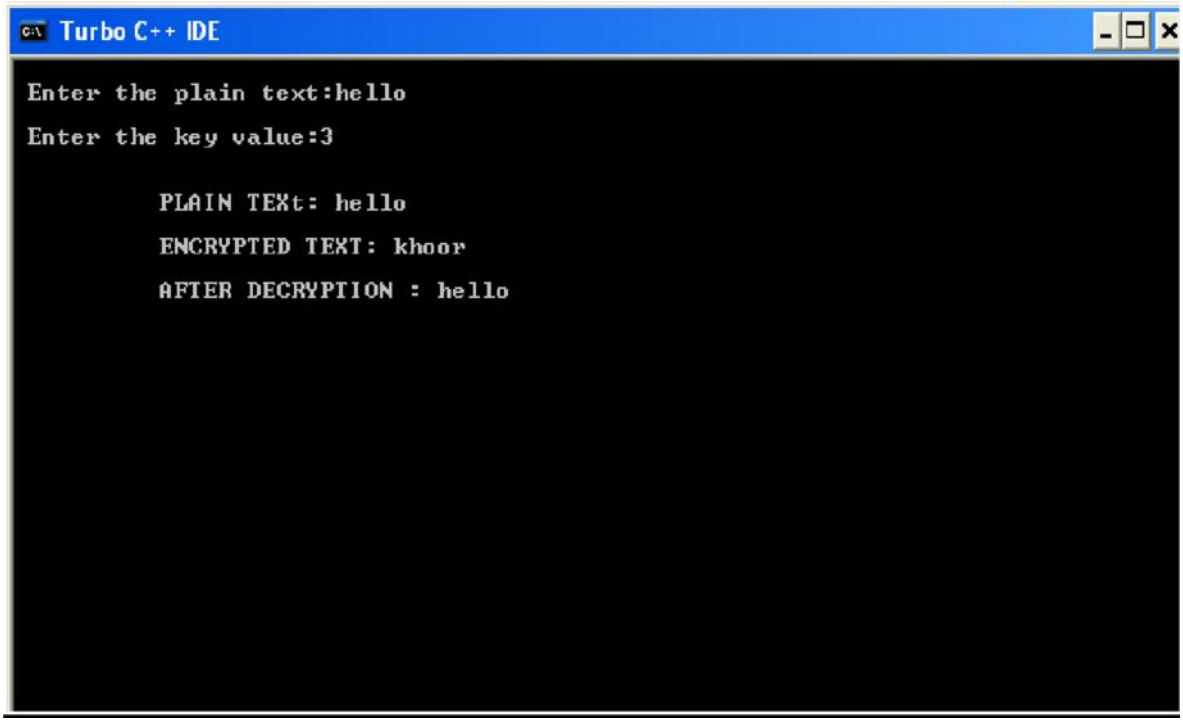
STEP-5: Display the cipher text obtained above.

PROGRAM: (Caesar Cipher)

```
#include <stdio.h>
#include <string.h>
#include<conio.h>
#include <ctype.h>
void main()
```

```
{
 char plain[10], cipher[10];
 int key,i,length;
 int result;
 clrscr();
 printf("\n Enter the plain text:");
 scanf("%s", plain);
 printf("\n Enter the key value:");
 scanf("%d", &key);
 printf("\n \n \t PLAIN TEXt: %s",plain);
 printf("\n \n \t ENCRYPTED TEXT: ");
 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("\n \n \t AFTER DECRYPTION : ");
 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])&&(plain[i]<'a'))
     plain[i]=plain[i]+26;
     printf("%c",plain[i]);
 }
getch();
}
```

OUTPUT:



RESULT:

Thus the implementation of Caesar cipher had been executed successfully.

EX. NO: 1(B)

IMPLEMENTATION OF PLAYFAIR CIPHER

AIM:

To write a C program to implement the Playfair Substitution technique.

DESCRIPTION:

The Playfair cipher starts with creating a key table. The key table is a 5×5 grid of letters that will act as the key for encrypting your plaintext. Each of the 25 letters must be unique and one letter of the alphabet is omitted from the table (as there are 25 spots and 26 letters in the alphabet).

To encrypt a message, one would break the message into digrams (groups of 2 letters) such that, for example, "HelloWorld" becomes "HE LL OW OR LD", and map them out on the key table. The two letters of the diagram are considered as the opposite corners of a rectangle in the key table. Note the relative position of the corners of this rectangle. Then apply the following 4 rules, in order, to each pair of letters in the plaintext:

1. If both letters are the same (or only one letter is left), add an "X" after the first letter
2. If the letters appear on the same row of your table, replace them with the letters to their immediate right respectively
3. If the letters appear on the same column of your table, replace them with the letters immediately below respectively
4. If the letters are not on the same row or column, replace them with the letters on the same row respectively but at the other pair of corners of the rectangle defined by the original pair.

EXAMPLE:

D. Playfair Cipher

Example1: Plaintext: CRYPTO IS TOO EASY **Key =** INFOSEC **Ciphertext:** ??

Grouped text: CR YP TO IS TO XO EA SY

Ciphertext: AQ TV YB NI YB YF CB OZ

I / J	N	F	O	S
E	C	A	B	D
G	H	K	L	M
P	Q	R	T	U
V	W	X	Y	Z

ALGORITHM:

STEP-1: Read the plain text from the user.

STEP-2: Read the keyword from the user.

STEP-3: Arrange the keyword without duplicates in a 5*5 matrix in the row order and fill the remaining cells with missed out letters in alphabetical order. Note that 'i' and 'j' takes the same cell.

STEP-4: Group the plain text in pairs and match the corresponding corner letters by forming a rectangular grid.

STEP-5: Display the obtained cipher text.

PROGRAM: (Playfair Cipher)

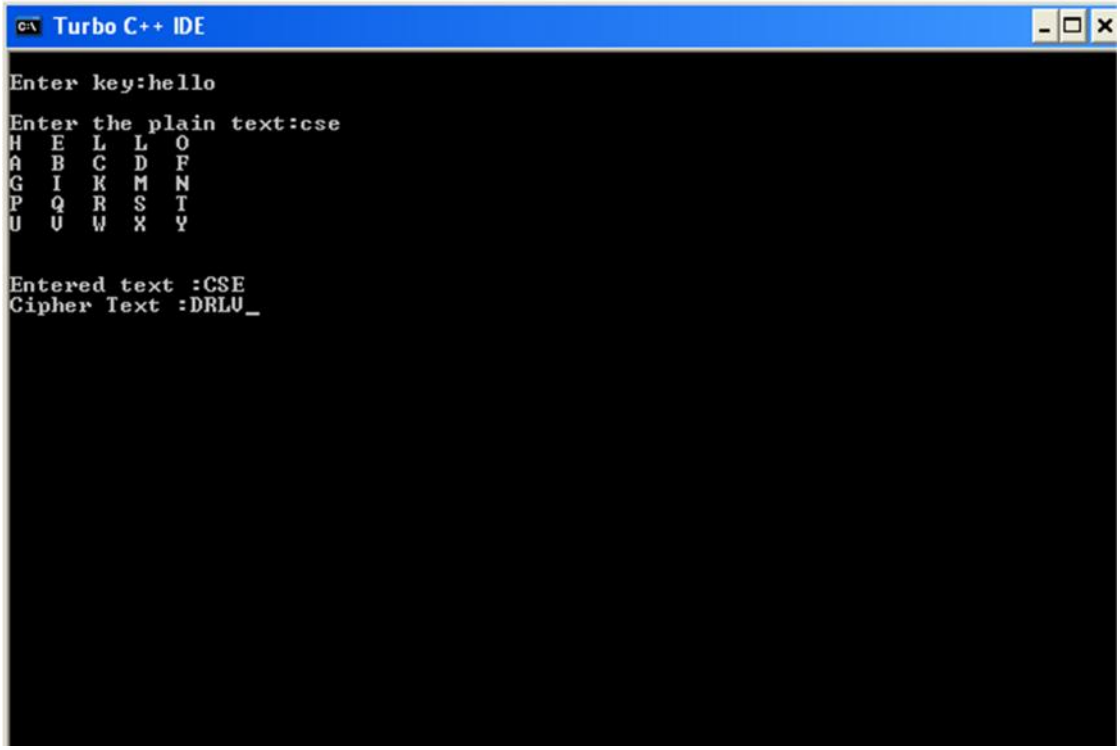
```
#include<stdio.h>
#include<conio.h>
#include<string.h
>
#include<ctype.h>
#define MX 5
void playfair(char ch1,char ch2, char key[MX][MX])
{
     int
     i,j,w,x,y,z;
     FILE *out;
if((out=fopen("cipher.txt","a+"))==NULL)
{
     printf("File Currupted.");
}
for(i=0;i<MX;i++)
{
```

```
        for(j=0;j<MX;j++)
        {
        if(ch1==key[i][j])
        {
        w=i;
        x=j;
        }
        else if(ch2==key[i][j])
        {
        y=i;
        z=j;
    }}}
    //printf("%d%d  %d%d",w,x,y,z);
    if(w==y)
    {
        x=(x+1)%5;z=(z+1)%5;
        printf("%c%c",key[w][x],key[y][z]);
        fprintf(out,  "%c%c",key[w][x],key[y][z]);
    }
    else if(x==z)
    {
```

```
        w=(w+1)%5;y=(y+1)%5;
        printf("%c%c",key[w][x],key[y][z]);
        fprintf(out,  "%c%c",key[w][x],key[y][z]);
    }
    else
    {
        printf("%c%c",key[w][z],key[y][x]);
        fprintf(out,  "%c%c",key[w][z],key[y][x]);
    }
    fclose(out);
    }
    void main()
    {
        int i,j,k=0,l,m=0,n;
        char   key[MX][MX],keyminus[25],keystr[10],str[25]={0};
        char
        alpa[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'}
        ;
        clrscr();
        printf("\nEnter key:");
        gets(keystr);
        printf("\nEnter the plain text:");
        gets(str);
        n=strlen(keystr);
        //convert the characters to uppertext
        for (i=0; i<n; i++)
        {
            if(keystr[i]=='j')keystr[i]='i';
            else  if(keystr[i]=='J')keystr[i]='I';
            keystr[i] = toupper(keystr[i]);
        }
        //convert all the characters of plaintext to uppertext
        for (i=0; i<strlen(str); i++)
        {
            if(str[i]=='j')str[i]='i';
            else  if(str[i]=='J')str[i]='I';
            str[i] = toupper(str[i]);
        }
        j=0;
        for(i=0;i<26;i++)
        {
            for(k=0;k<n;k++)
            {
            if(keystr[k]==alpa[i])
            break;
            else  if(alpa[i]=='J')
            break;
            }
            if(k==n)
            {
            keyminus[j]=alpa[i];j++;
            }
        }
```

```
        //construct key keymatrix
        k=0;
        for(i=0;i<MX;i++)
        {
             for(j=0;j<MX;j++)
             {
             if(k<n)
             {
             key[i][j]=keystr[k];
             k++;}
             else
             {
             key[i][j]=keyminus[m];m++;
             }
             printf("%c  ",key[i][j]);
             }
             printf("\n");
        }
        printf("\n\nEntered text :%s\nCipher Text :",str);
        for(i=0;i<strlen(str);i++)
        {
        if(str[i]=='J')str[i]='I';
        if(str[i+1]=='\0')
        playfair(str[i],'X',key);
        else
        {
             if(str[i+1]=='J')str[i+1]='I';
             if(str[i]==str[i+1])
             playfair(str[i],'X',key);
             else
             {
             playfair(str[i],str[i+1],key);i++;
             }}
        }
    getch();
    }
```

OUTPUT:



RESULT:

Thus the Playfair cipher substitution technique had been implemented successfully.

EX. NO: 1(C)

IMPLEMENTATION OF HILL CIPHER

AIM:

To write a C program to implement the hill cipher substitution techniques.

DESCRIPTION:

Each letter is represented by a number modulo 26. Often the simple scheme A = 0, B = 1... Z = 25, is used, but this is not an essential feature of the cipher. To encrypt a message, each block of n letters is multiplied by an invertible $n \times n$ matrix, against modulus 26. To decrypt the message, each block is multiplied by the inverse of the m trix used for encryption. The matrix used for encryption is the cipher key, and it sho ld be chosen randomly from the set of invertible $n \times n$ matrices (modulo 26).

EXAMPLE:

$$\begin{bmatrix} 2 & 4 & 5 \\ 9 & 2 & 1 \\ 3 & 17 & 7 \end{bmatrix} \begin{bmatrix} 0 \\ 19 \\ 19 \end{bmatrix} = \begin{bmatrix} 171 \\ 57 \\ 456 \end{bmatrix} \pmod{26} = \begin{bmatrix} 15 \\ 5 \\ 14 \end{bmatrix} = \text{'PFO'}$$

ALGORITHM:

STEP-1: Read the plain text and key from the user.

STEP-2: Split the plain text into groups of length three.

STEP-3: Arrange the keyword in a 3*3 matrix.

STEP-4: Multiply the two matrices to obtain the cipher text of length three.

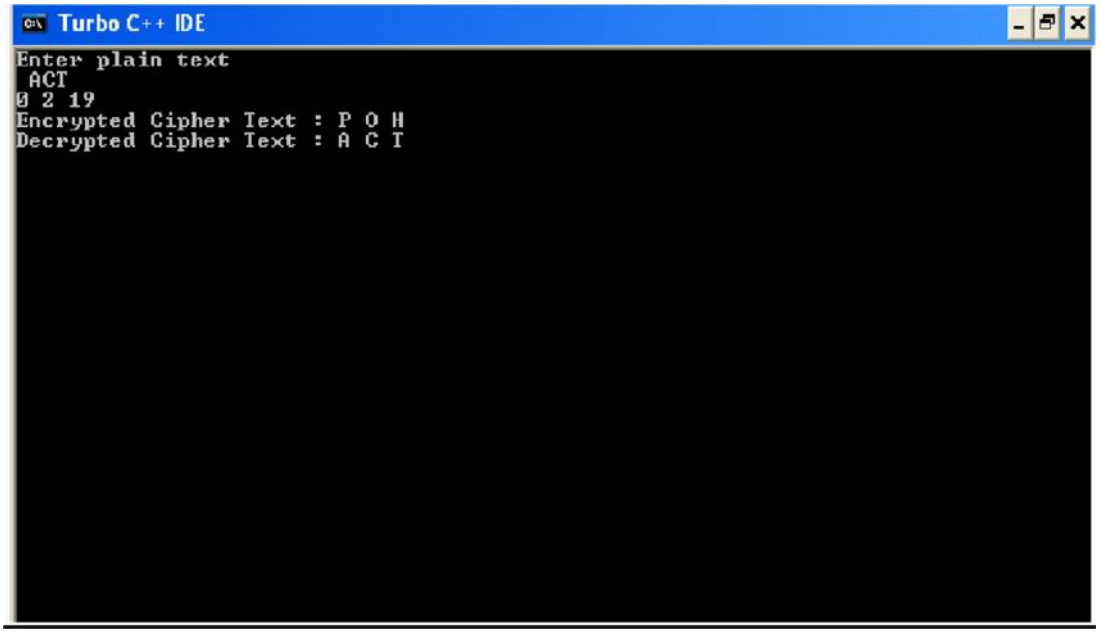
STEP-5: Combine all these groups to get the complete cipher text.

PROGRAM: (Hill Cipher)

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
int main(){
      unsigned int a[3][3]={{6,24,1},{13,16,10},{20,17,15}};
      unsigned int b[3][3]={{8,5,10},{21,8,21},{21,12,8}};
      int i,j, t=0;
      unsigned int c[20],d[20];
      char msg[20];
      clrscr();
      printf("Enter plain text\n ");
      scanf("%s",msg);
      for(i=0;i<strlen(msg);i++)
      {    c[i]=msg[i]-65;
```

```
            printf("%d ",c[i]);
      }
      for(i=0;i<3;i++)
      {   t=0;
      for(j=0;j<3;j++)
      {
            t=t+(a[i][j]*c[j]);
      }
            d[i]=t%26;
      }
printf("\nEncrypted Cipher Text :");
for(i=0;i<3;i++)
printf(" %c",d[i]+65);
for(i=0;i<3;i++)
{
      t=0;
      for(j=0;j<3;j++)
      {
            t=t+(b[i][j]*d[j]);
      }
      c[i]=t%26;
}
printf("\nDecrypted Cipher Text :");
for(i=0;i<3;i++)
printf(" %c",c[i]+65);
getch();
return 0;
}
```

OUTPUT:



RESULT:

Thus the hill cipher substitution technique had been implemented successfully in C.

EX. NO: 2

IMPLEMENTATION OF RAIL FENCE – ROW & COLUMN TRANSFORMATION TECHNIQUE

AIM:

To write a C program to implement the rail fence transposition technique.

DESCRIPTION:

In the rail fence cipher, the plain text is written downwards and diagonally on successive "rails" of an imaginary fence, then moving up when we reach the bottom rail. When we reach the top rail, the message is written downwards again until the whole plaintext is written out. The message is then read off in rows.

EXAMPLE:

```
A U T H O R
1 6 5 2 3 4
-----------
W E A R E D
 I S C O V E
R E D S A V
E Y O U R S
E L F A B C
```

yields the cipher

```
W I R E E R O S U A E V A R B D E V S C A C D O F E S E Y L .
```

ALGORITHM:

STEP-1: Read the Plain text.

STEP-2: Arrange the plain text in row columnar matrix format.

STEP-3: Now read the keyword depending on the number of columns of the plain text.

STEP-4: Arrange the characters of the keyword in sorted order and the corresponding columns of the plain text.

STEP-5: Read the characters row wise or column wise in the former order to get the cipher text.

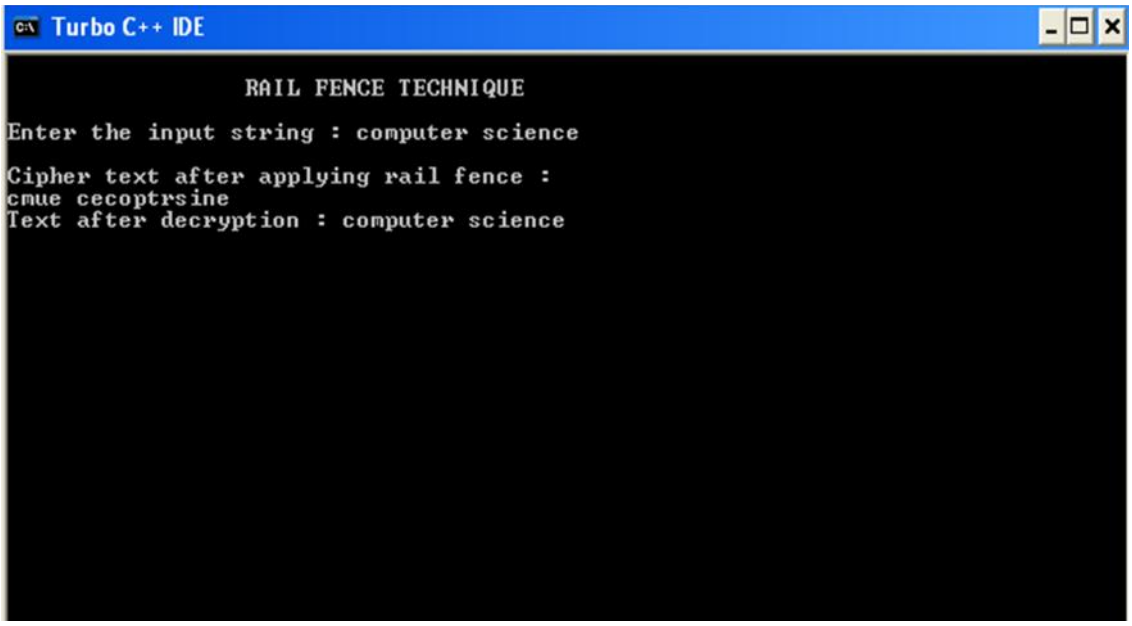
PROGRAM: (Rail Fence)

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
     int i,j,k,l;
     char  a[20],c[20],d[20];
     clrscr();
     printf("\n\t\t RAIL FENCE TECHNIQUE");
     printf("\n\nEnter the input string : ");
     gets(a);
     l=strlen(a);

/*Ciphering*/
for(i=0,j=0;i<l;i++)
{
     if(i%2==0)
     c[j++]=a[i];
}
for(i=0;i<l;i++)
{
     if(i%2==1)
     c[j++]=a[i];
}
c[j]='\0';
printf("\nCipher text after applying rail fence :");
printf("\n%s",c);

/*Deciphering*/
if(l%2==0)
     k=l/2;
else
     k=(l/2)+1;
for(i=0,j=0;i<k;i++)
{
    d[j]=c[i];
    j=j+2;
}
for(i=k,j=1;i<l;i++)
{
    d[j]=c[i];
    j=j+2;
}
d[l]='\0';
printf("\nText after decryption : ");
printf("%s",d);
getch();
}
```

OUTPUT:



RESULT:

Thus the rail fence algorithm had been executed successfully.

EX. NO: 3 **IMPLEMENTATION OF DES**

AIM:

To write a java program to implement Data Encryption Standard (DES).

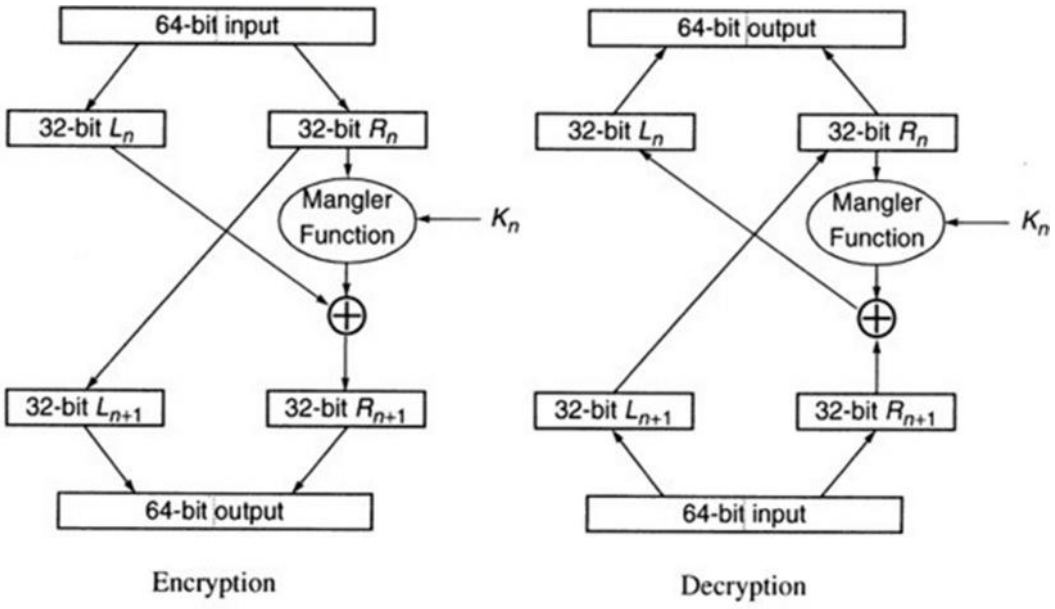
DESCRIPTION:

DES is a symmetric encryption system that uses 64-bit blocks, 8 bits of which are used for parity checks. The key therefore has a "useful" length of 56 bits, which means that only 56 bits are actually used in the algorithm. The algorithm involves carrying out combinations, substitutions and permutations between the text to be encrypted and the key, while making sure the operations can be performed in both directions. The key is ciphered on 64 bits and made of 16 blocks of 4 bits, generally denoted k_1 to k_{16}. Given that "only" 56 bits are actually used for encrypting, there can be 2^{56} different keys.

The main parts of the algorithm are as follows:

- Fractioning of the text into 64-bit blocks
- Initial permutation of blocks
- Breakdown of the blocks into two parts: left and right, named L and R
- Permutation and substitution steps repeated 16 times
- Re-joining of the left and right parts then inverse initial permutation

EXAMPLE:



ALGORITHM:

STEP-1: Read the 64-bit plain text.

STEP-2: Split it into two 32-bit blocks and store it in two different arrays.

STEP-3: Perform XOR operation between these two arrays.

STEP-4: The output obtained is stored as the second 32-bit sequence and the original second 32-bit sequence forms the first part.

STEP-5: Thus the encrypted 64-bit cipher text is obtained in this way. Repeat the same process for the remaining plain text characters.

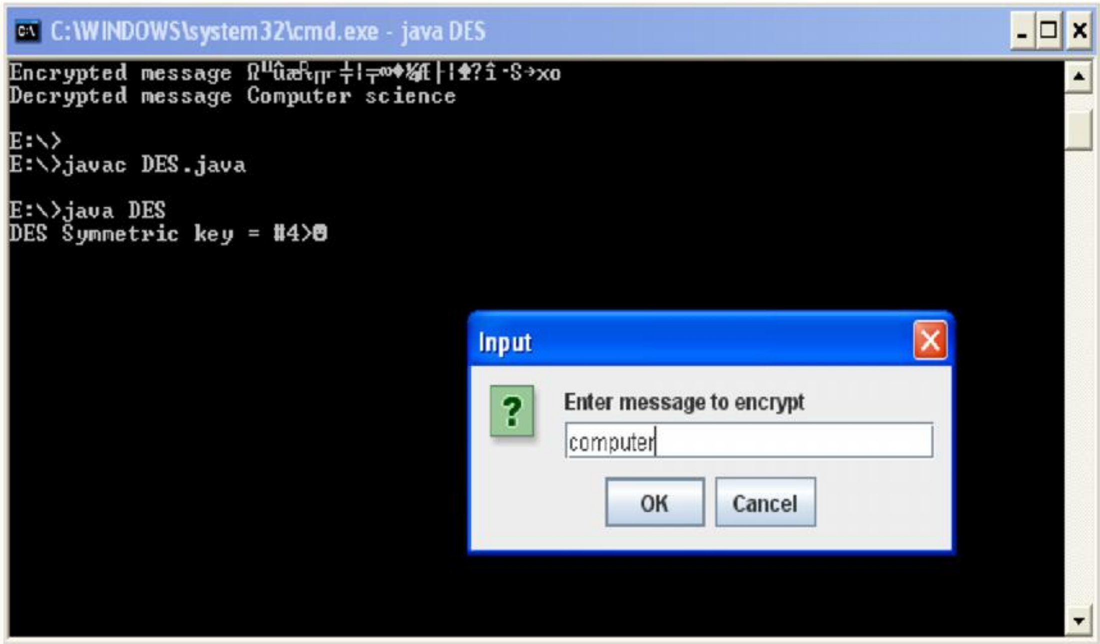
PROGRAM:

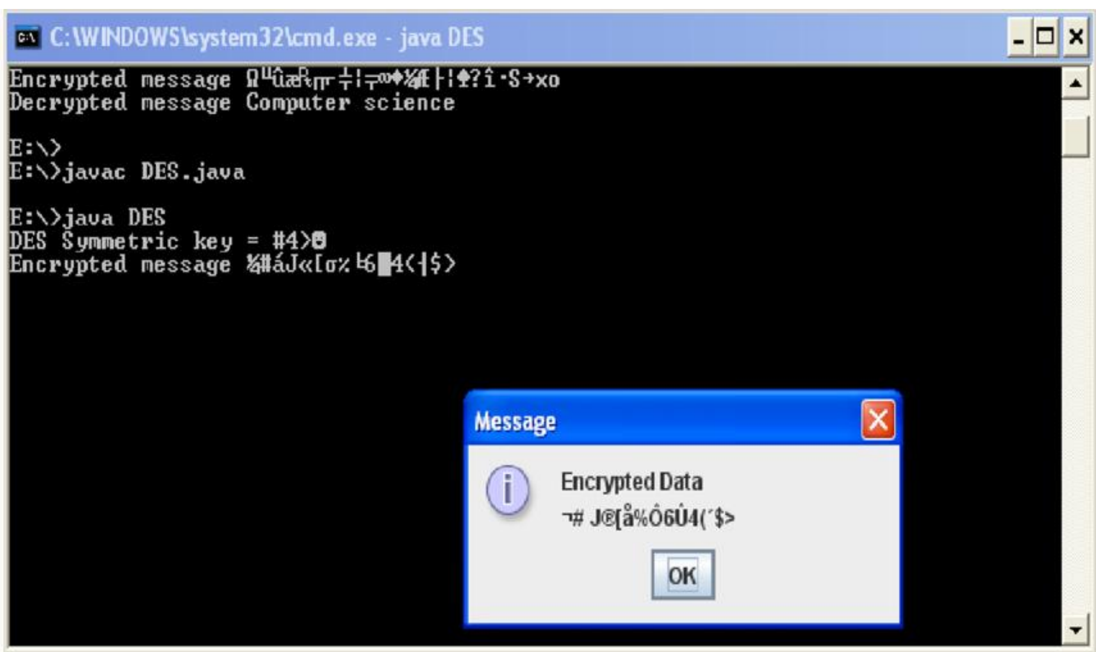
DES.java

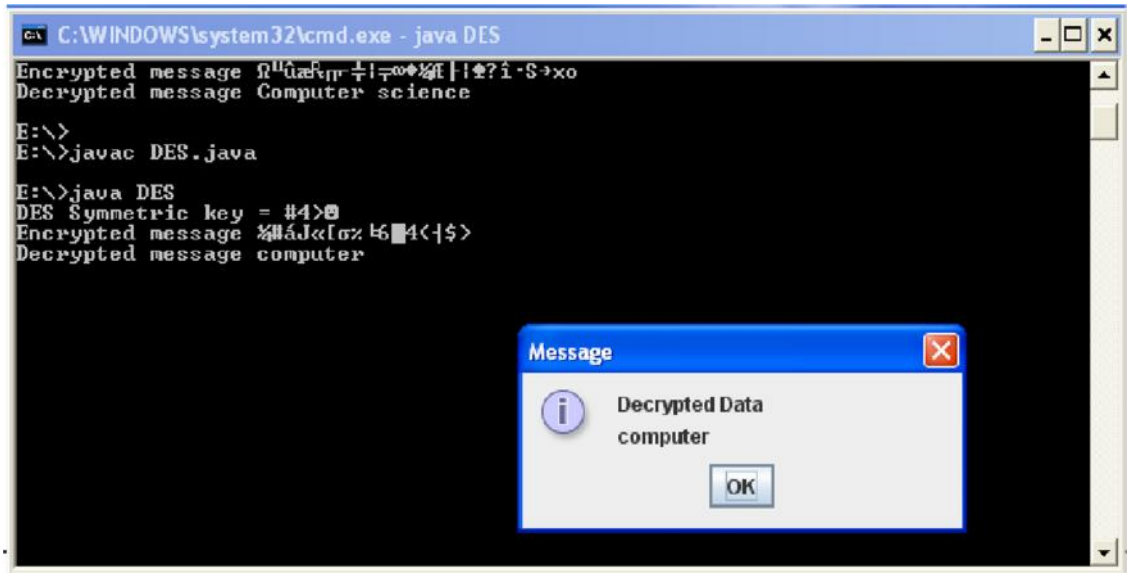
```
import javax.swing.*;
import java.security.SecureRandom;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.util.Random ;
class DES {
     byte[] skey = new byte[1000];
     String skeyString;
     static byte[] raw;
     String inputMessage,encryptedData,decryptedMessage;
public DES()
{
try
{
     generateSymmetricKey();
     inputMessage=JOptionPane.showInputDialog(null,"Enter
     message to encrypt");
     byte[] ibyte = inputMessage.getBytes();
     byte[] ebyte=encrypt(raw, ibyte);
     String encryptedData = new String(ebyte);
     System.out.println("Encrypted message "+encryptedData);
     JOptionPane.showMessageDialog(null,"Encrypted Data
     "+"\n"+encryptedData);
     byte[] dbyte= decrypt(raw,ebyte);
     String decryptedMessage = new String(dbyte);
     System.out.println("Decrypted message
     "+decryptedMessage);
     JOptionPane.showMessageDialog(null,"Decrypted Data
     "+"\n"+decryptedMessage);
}
catch(Exception e)
{
     System.out.println(e);
}
}
```

```
void generateSymmetricKey() {
try {
     Random r = new Random();
     int num = r.nextInt(10000);
     String knum = String.valueOf(num);
     byte[] knumb = knum.getBytes();
     skey=getRawKey(knumb);
     skeyString = new String(skey);
     System.out.println("DES Symmetric key = "+skeyString);
}
catch(Exception e)
{
     System.out.println(e);
}
}
private static byte[] getRawKey(byte[] seed) throws Exception
{
     KeyGenerator kgen = KeyGenerator.getInstance("DES");
     SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");
     sr.setSeed(seed);
     kgen.init(56, sr);
     SecretKey skey = kgen.generateKey();
     raw = skey.getEncoded();
     return raw;
}
private static byte[] encrypt(byte[] raw, byte[] clear) throws
Exception {
          SecretKeySpec skeySpec = new SecretKeySpec(raw,
          "DES");
          Cipher cipher = Cipher.getInstance("DES");
          cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
          byte[] encrypted = cipher.doFinal(clear);
          return encrypted;
}
private static byte[] decrypt(byte[] raw, byte[] encrypted)
throws Exception
{
          SecretKeySpec skeySpec = new SecretKeySpec(raw,
          "DES");
          Cipher cipher = Cipher.getInstance("DES");
          cipher.init(Cipher.DECRYPT_MODE, skeySpec);
          byte[] decrypted = cipher.doFinal(encrypted);
          return decrypted;
}
public static void main(String args[]) {
     DES des = new DES();
}
}
```

<u>OUTPUT:</u>


C:\WINDOWS\system32\cmd.exe - java DES
Decrypted message Computer science
E:\>
E:\>javac DES.java
E:\>java DES
Input
Enter message to encrypt
computer
OK
Cancel


C:\WINDOWS\system32\cmd.exe - java DES
Decrypted message Computer science
E:\>
E:\>javac DES.java
E:\>java DES
Message
Encrypted Data
OK



RESULT:

Thus the data encryption standard algorithm had been implemented successfully.

EX. NO: 4

IMPLEMENTATION OF AES ALGORITHM

AIM:

To write a C program to implement Advanced Encryption Standard (AES) using C++ Language.

DESCRIPTION:

AES uses a symmetric key, meaning the same key is used for both encryption and decryption.

In the provided code, a key of length 128 bits **(AES::DEFAULT_KEYLENGTH)** is used. This key is a sequence of bytes that should be kept confidential.

The Initialization Vector (IV) is a crucial element in AES encryption, especially in modes like **Cipher Block Chaining (CBC)**.

It's a random or pseudorandom value that is used alongside the key to initialize the encryption process.

The IV ensures that even if the same plaintext is encrypted multiple times, the resulting **ciphertext** will be different.

In the code, an **IV** of the same length as the block size **(AES::BLOCKSIZE)** is used.

Encryption Process:

The actual encryption process involves creating an AES encryptor object with the provided key and IV.

After that, the plaintext is processed using a **StringSource** and a **StreamTransformationFilter** to apply the encryption using the specified encryptor.

The result is the ciphertext, which is then encoded to a more human-readable form (hexadecimal) for display.

PROGRAM:

```
#include <iostream>
#include <iomanip>
#include <string>
#include <cryptopp/aes.h>
#include <cryptopp/modes.h>
#include <cryptopp/filters.h>

using namespace CryptoPP;
using namespace std;
 int main() {
 // Key and IV (Initialization Vector) for AES
 byte key[AES::DEFAULT_KEYLENGTH] = {'k', 'e', 'y', '1', '2', '3', '4', '5', '6', '7', '8', '9', '1', '0', '1', '1'
};
 byte iv[AES::BLOCKSIZE] = {'i', 'v', '1', '2', '3', '4', '5', '6', '7', '8', '9', '1', '0', '1', '1', '1'};
  // Message to be encrypted
 string plainText = "Hello, AES!";

 // Encrypt using AES in CBC mode
 CBC_Mode<AES>::Encryption encryptor(key, sizeof(key), iv);
 StringSource(plainText, true, new StreamTransformationFilter(encryptor, new StringSink(cipherTex
t)));
  // Display the encrypted message
 cout << "Encrypted Text: ";
 StringSource(cipherText, true, new HexEncoder(new StringSink(cout)));
 cout << endl;

 return 0;
}
```

OUTPUT:

Encrypted Text (Hex): CFB4DBAF82913CA4C4D68CE8B33A384C

RESULT:

Thus the C++ program to implement AES algorithm had been implementedsuccessfully.

EX. NO: 5 **IMPLEMENTATION OF RSA**

AIM:

To write a C program to implement the RSA encryption algorithm.

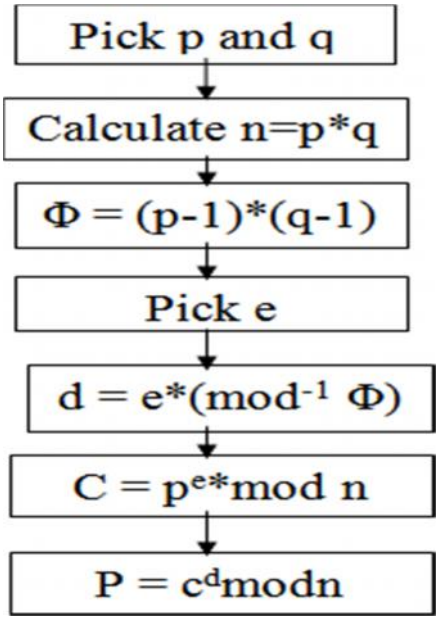
DESCRIPTION:

RSA is an algorithm used by modern computers to encrypt and decrypt messages. It is an asymmetric cryptographic algorithm. Asymmetric means that there are two different keys. This is also called public key cryptography, because one of them can be given to everyone. A basic principle behind RSA is the observation that it is practical to find three very large positive integers e, d and n such that with modular exponentiation for all integer m:

$$(m^e)^d = m \pmod n$$

The public key is represented by the integers n and e; and, the private key, by the integer d. m represents the message. RSA involves a public key and a private key. The public key can be known by everyone and is used for encrypting messages. The intention is that messages encrypted with the public key can only be decrypted in a reasonable amount of time using the private key.

EXAMPLE:



ALGORITHM:

STEP-1: Select two co-prime numbers as p and q.

STEP-2: Compute n as the product of p and q.

STEP-3: Compute (p-1)*(q-1) and store it in z.

STEP-4: Select a random prime number e that is less than that of z.

STEP-5: Compute the private key, d as e * mod^{-1}(z).

STEP-6: The cipher text is computed as $message^{e}$ * mod n.

STEP-7: Decryption is done as $cipher^{d}$mod n.

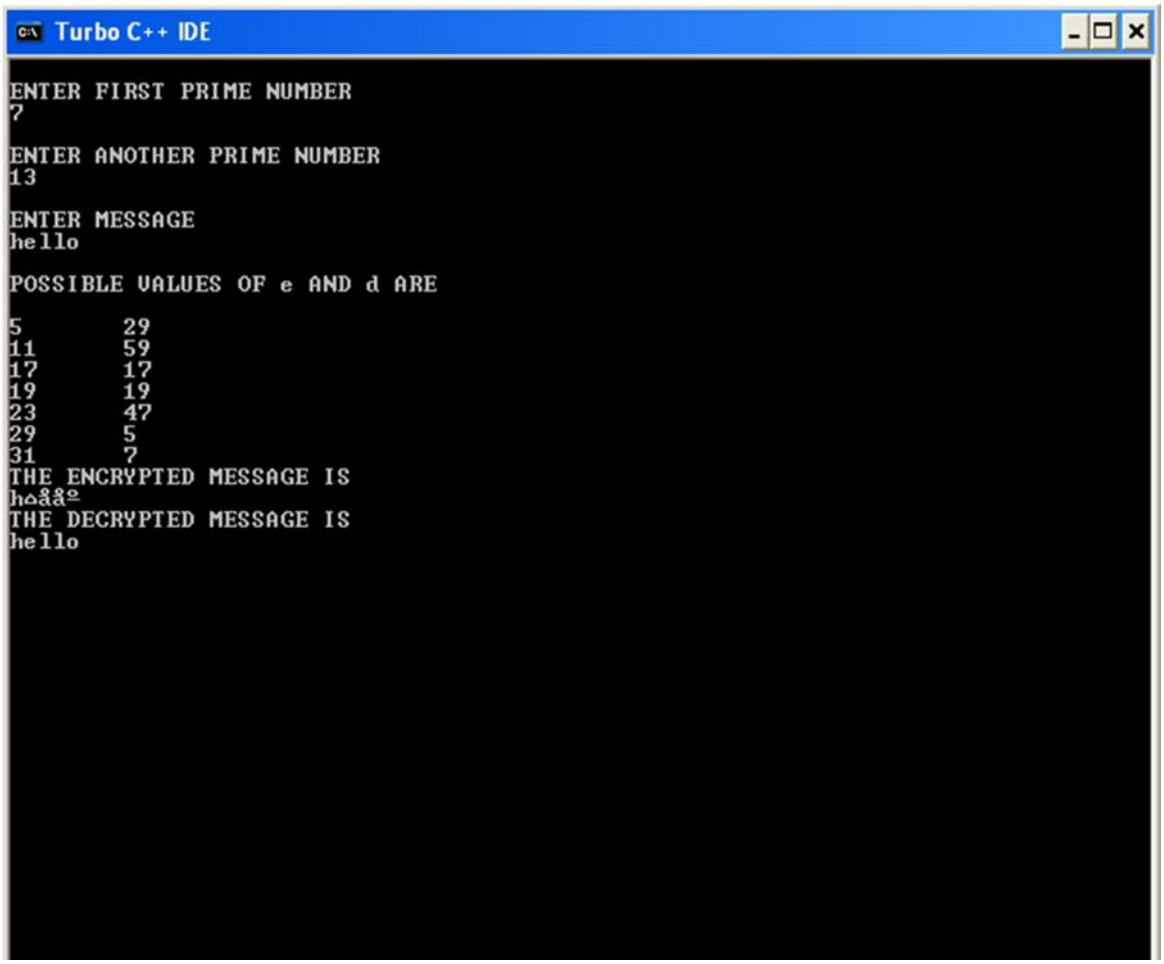
PROGRAM: (RSA)

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>
#include<string.h>
     long int
     p,q,n,t,flag,e[100],d[100],temp[100],j,m[100],en[100],i;
     char msg[100];
     int prime(long int);
     void ce();
     long int cd(long int);
     void encrypt();
     void decrypt();
void main()
{
     clrscr();
     printf("\nENTER FIRST PRIME NUMBER\n");
     scanf("%d",&p);
     flag=prime(p);
     if(flag==0)
      {
          printf("\nWRONG  INPUT\n");
          getch();
      }
      printf("\nENTER ANOTHER PRIME NUMBER\n");
      scanf("%d",&q);
      flag=prime(q);
      if(flag==0||p==q)
      {
       printf("\nWRONG  INPUT\n");
       getch();
      }
      printf("\nENTER  MESSAGE\n");
      fflush(stdin);
      scanf("%s",msg);
      for(i=0;msg[i]!=NULL;i++)
      m[i]=msg[i];
      n=p*q;
```

```
        t=(p-1)*(q-1);
        ce();
        printf("\nPOSSIBLE VALUES OF e AND d ARE\n");
        for(i=0;i<j-1;i++)
        printf("\n%ld\t%ld",e[i],d[i]);
        encrypt();
        decrypt();
        getch();
}
int prime(long int pr)
{
int  i;
j=sqrt(pr);
for(i=2;i<=j;i++)
{
if(pr%i==0)
return 0;
}
return 1;
}
void ce()
{
int k;
k=0;
for(i=2;i<t;i++)
{
 if(t%i==0)
 continue;
 flag=prime(i);
 if(flag==1&&i!=p&&i!=q)
 {
 e[k]=i;
 flag=cd(e[k]);
 if(flag>0)
 {
 d[k]=flag;
 k++;
 }
 if(k==99)
 break;
 } } }
 long int cd(long int x)
 {
 long int k=1;
 while(1)
 {
 k=k+t;
 if(k%x==0)
 return(k/x);
 } }
 void encrypt() {
 long int pt,ct,key=e[0],k,len;
 i=0;
 len=strlen(msg);
```

```
while(i!=len) {
pt=m[i];
pt=pt-96;
k=1;
for(j=0;j<key;j++)
{ k=k*pt;
k=k%n;
}
temp[i]=k;
ct=k+96;
en[i]=ct;
i++;
}
en[i]=-1;
printf("\nTHE ENCRYPTED MESSAGE IS\n");
for(i=0;en[i]!=-1;i++)
printf("%c",en[i]);
}
void decrypt()
{
long int pt,ct,key=d[0],k;
i=0;
while(en[i]!=-1)
{
ct=temp[i];
k=1;
for(j=0;j<key;j++)
{
k=k*ct;
k=k%n;
}
pt=k+96;
m[i]=pt;
i++;
}
m[i]=-1;
printf("\nTHE DECRYPTED MESSAGE IS\n");
for(i=0;m[i]!=-1;i++)
printf("%c",m[i]);
}
```

OUTPUT:



RESULT:

Thus the C program to implement RSA encryption technique had been implemented successfully.

EX. NO: 6

IMPLEMENTATION OF DIFFIE HELLMAN KEY EXCHANGE ALGORITHM

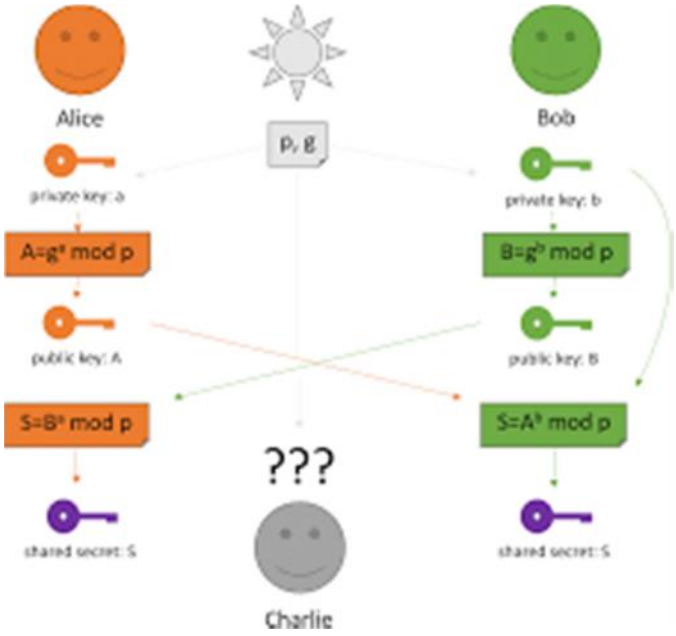
AIM:

To implement the Diffie-Hellman Key Exchange algorithm using C language.

DESCRIPTION:

Diffie–Hellman Key Exchange establishes a shared secret between two parties that can be used for secret communication for exchanging data over a public network. It is primarily used as a method of exchanging cryptography keys for use in symmetric encryption algorithms like AES. The algorithm in itself is very simple. The process begins by having the two parties, Alice and Bob. Let's assume that Alice wants to establish a shared secret with Bob.

EXAMPLE:



ALGORITHM:

STEP-1: Both Alice and Bob shares the same public keys g and p.

STEP-2: Alice selects a random public key a.

STEP-3: Alice computes his secret key A as g^a mod p.

STEP-4: Then Alice sends A to Bob.

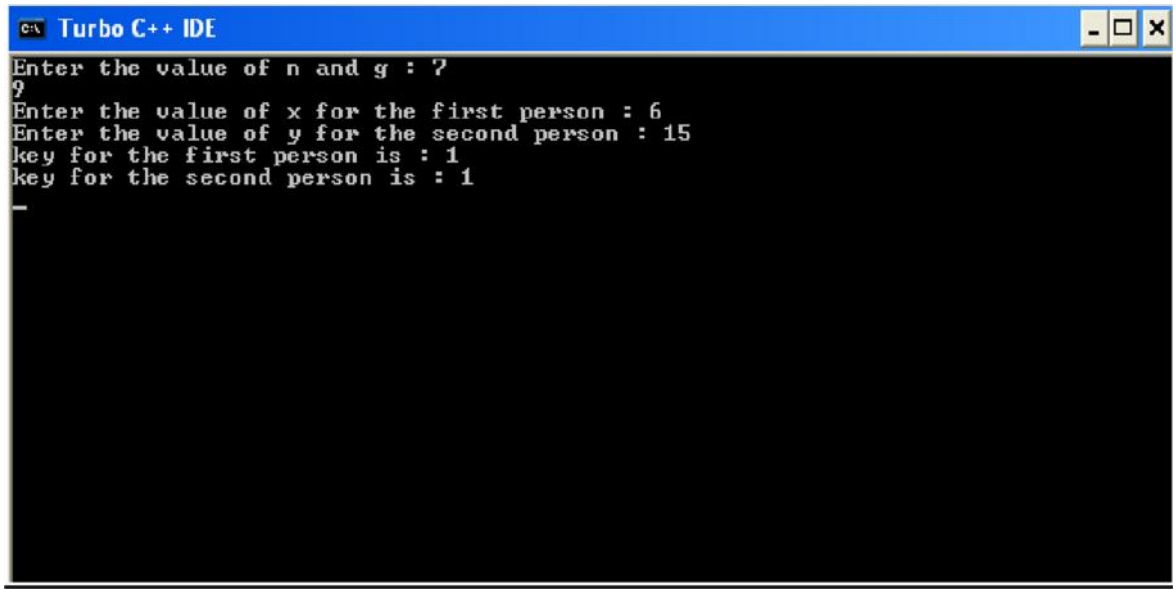
STEP-5: Similarly Bob also selects a public key b and computes his secret key as B and sends the same back to Alice.

STEP-6: Now both of them compute their common secret key as the other one's secret key power of a mod p.

PROGRAM: (Diffie Hellman Key Exchange)

```
#include<stdio.h>
#include<conio.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 int calculateKey(int a, int x, int n)
{
     return power(a,x,n);
}
void main()
{
     int n,g,x,a,y,b;
     clrscr();
     printf("Enter the value of n and g : ");
     scanf("%d%d",&n,&g);
     printf("Enter the value of x for the first person : ");
     scanf("%d",&x);
     a=power(g,x,n);
     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));
     getch();
}
```

OUTPUT:



RESULT:

Thus the Diffie-Hellman key exchange algorithm had been successfully implemented using C.

EX. NO: 7

IMPLEMENTATION OF MD5

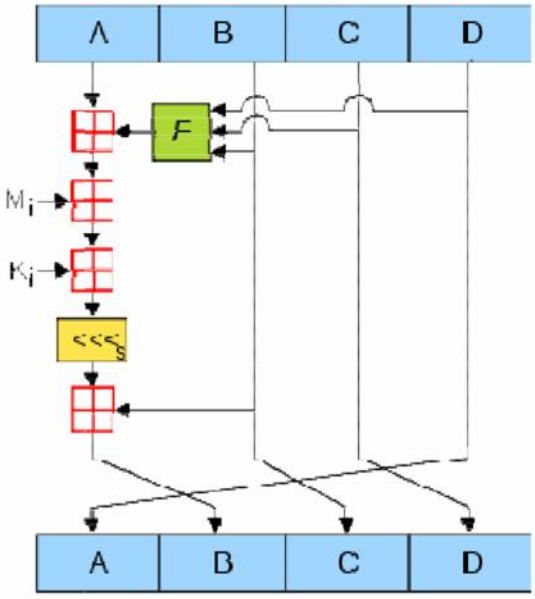
AIM:

To write a C program to implement the MD5 hashing technique.

DESCRIPTION:

MD5 processes a variable-length message into a fixed-length output of 128 bits. The input message is broken up into chunks of 512-bit blocks. The message is padded so that its length is divisible by 512. The padding works as follows: first a single bit, 1, is appended to the end of the message. This is followed by as many zeros as are required to bring the length of the message up to 64 bits less than a multiple of 512. The remaining bits are filled up with 64 bits representing the length of the original message, modulo 2^{64}.The main MD5 algorithm operates on a 128-bit state, divided into four 32-bit words, denoted A, B, C, and D. These are initialized to certain fixed constants. The main algorithm then uses each 512-bit message block in turn to modify the state.

EXAMPLE:



ALGORITHM:

STEP-1: Read the 128-bit plain text.

STEP-2: Divide into four blocks of 32-bits named as A, B, C and D.

STEP-3: Compute the functions f, g, h and i with operations such as, rotations, permutations, etc,.

STEP-4: The output of these functions are combined together as F and performed circular shifting and then given to key round.

STEP-5: Finally, right shift of 's' times are performed and the results are combined together to produce the final output.

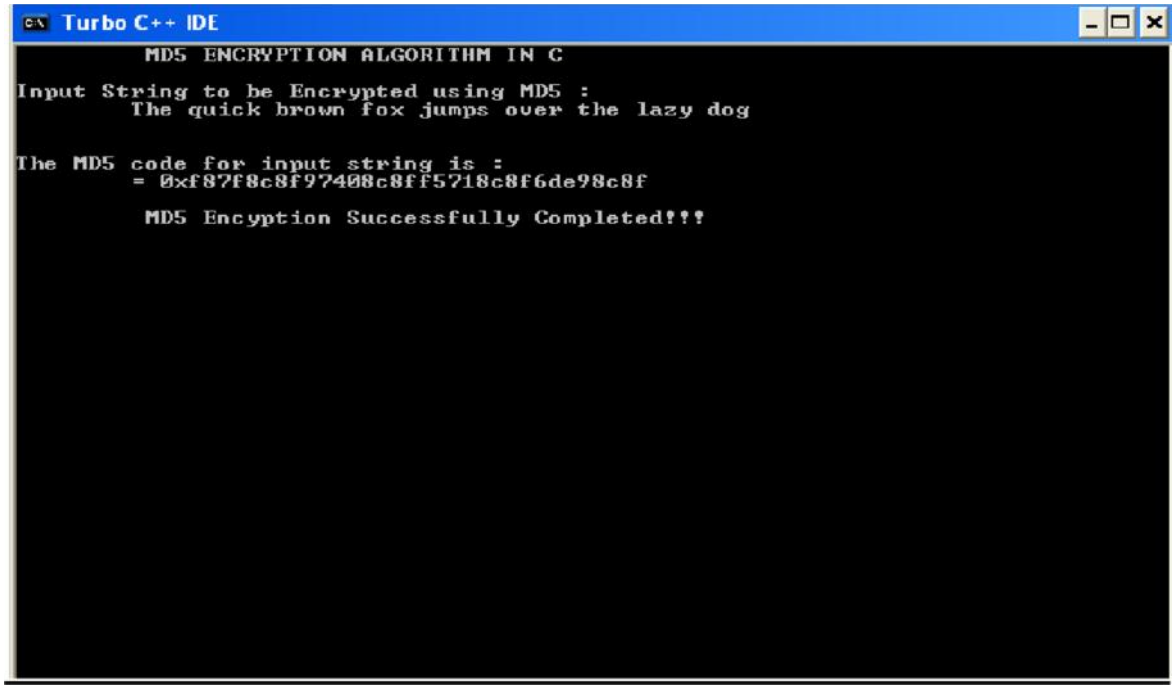
PROGRAM:(MD5)

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include<conio.h>
typedef union uwb
{
    unsigned  w;
    unsigned char b[4];
} MD5union;
     typedef unsigned DigestArray[4];
     unsigned func0( unsigned abcd[] ){
     return ( abcd[1] & abcd[2]) | (~abcd[1] & abcd[3]);}
     unsigned func1( unsigned abcd[] ){
     return ( abcd[3] & abcd[1]) | (~abcd[3] & abcd[2]);}
     unsigned func2( unsigned abcd[] ){
     return abcd[1] ^ abcd[2] ^ abcd[3];}
     unsigned func3( unsigned abcd[] ){
     return abcd[2] ^ (abcd[1] |~ abcd[3]);}
     typedef unsigned (*DgstFctn)(unsigned a[]);
unsigned *calctable( unsigned *k)
{
     double s, pwr;
     int i;
     pwr = pow( 2, 32);
for (i=0; i<64; i++)
{
     s = fabs(sin(1+i));
     k[i] = (unsigned)( s * pwr );
}
     return k;
}
unsigned rol( unsigned r, short N )
{
    unsigned  mask1 = (1<<N) -1;
    return ((r>>(32-N)) & mask1) | ((r<<N) & ~mask1);
}
```

```
unsigned *md5( const char *msg, int mlen)
{
    static DigestArray h0 = { 0x67452301, 0xEFCDAB89,
    0x98BADCFE, 0x10325476 };
    static DgstFctn ff[] = { &func0, &func1, &func2, &func3};
    static short M[] = { 1, 5, 3, 7 };
    static short O[] = { 0, 1, 5, 0 };
    static short rot0[] = { 7,12,17,22};
    static short rot1[] = { 5, 9,14,20};
    static short rot2[] = { 4,11,16,23};
    static short rot3[] = { 6,10,15,21};
    static short *rots[] = {rot0, rot1, rot2, rot3 };
    static unsigned kspace[64];
    static unsigned *k;
    static DigestArray h;
    DigestArray abcd;
    DgstFctn fctn;
    short m, o, g;
    unsigned f;
    short *rotn;
    union
    {
         unsigned w[16];
         char     b[64];
    }mm;
         int os = 0;
         int grp, grps, q, p;
         unsigned char *msg2;
         if (k==NULL) k= calctable(kspace);
    for (q=0; q<4; q++) h[q] = h0[q];   // initialize
    {
          grps = 1 + (mlen+8)/64;
          msg2 = malloc( 64*grps);
          memcpy( msg2, msg, mlen);
          msg2[mlen] = (unsigned char)0x80;
          q = mlen + 1;
     while (q < 64*grps){ msg2[q] = 0; q++ ; }
     {
          MD5union u;
          u.w = 8*mlen;
          q -= 8;
          memcpy(msg2+q, &u.w, 4 );
     }
     }
     for (grp=0; grp<grps; grp++)
     {
          memcpy( mm.b, msg2+os, 64);
```

```
        for(q=0;q<4;q++) abcd[q] = h[q];
        for (p = 0; p<4; p++)
        {
             fctn = ff[p];
             rotn = rots[p];
             m = M[p]; o= O[p];
        for (q=0; q<16; q++)
        {
             g = (m*q + o) % 16;
            f = abcd[1] + rol( abcd[0]+ fctn(abcd)+k[q+16*p]
            + mm.w[g], rotn[q%4]);
            abcd[0] = abcd[3];
             abcd[3] = abcd[2];
             abcd[2] = abcd[1];
           abcd[1] = f;
        }}
 for (p=0; p<4; p++)
 h[p] += abcd[p];
 os += 64;
 }
return h;}
void main()
{
     int j,k;
     const char *msg = "The quick brown fox jumps over
     the lazy dog";
     unsigned *d = md5(msg, strlen(msg));
     MD5union u;
     clrscr();
     printf("\t MD5 ENCRYPTION ALGORITHM IN C \n\n");
     printf("Input String to be Encrypted using MD5 :
     \n\t%s",msg);
     printf("\n\nThe MD5 code for input string is: \n");
     printf("\t= 0x");
     for (j=0;j<4; j++){
     u.w = d[j];
     for (k=0;k<4;k++) printf("%02x",u.b[k]);
}
printf("\n");
printf("\n\t MD5 Encyption Successfully
Completed!!!\n\n");
getch();
system("pause");
getch();}
```

OUTPUT:



RESULT:

Thus the implementation of MD5 hashing algorithm had been implemented successfully using C.

EX. NO: 8 **IMPLEMENTATION OF SHA-I**

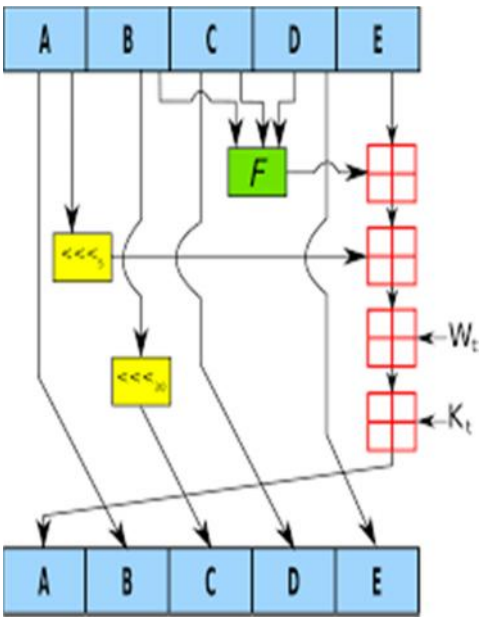
AIM:

To implement the SHA-I hashing technique using java program.

DESCRIPTION:

In cryptography, SHA-1 (Secure Hash Algorithm 1) is a cryptographic hash function. SHA-1 produces a 160-bit hash value known as a message digest. The way this algorithm works is that for a message of size < 264 bits it computes a 160-bit condensed output called a message digest. The SHA-1 algorithm is designed so that it is practically infeasible to find two input messages that hash to the same output message. A hash function such as SHA-1 is used to calculate an alphanumeric string that serves as the cryptographic representation of a file or a piece of data. This is called a digest and can serve as a digital signature. It is supposed to be unique and non-reversible.

EXAMPLE:



ALGORITHM:

STEP-1: Read the 256-bit key values.

STEP-2: Divide into five equal-sized blocks named A, B, C, D and E.

STEP-3: The blocks B, C and D are passed to the function F.

STEP-4: The resultant value is permuted with block E.

STEP-5: The block A is shifted right by ‘s’ times and permuted with the result of step-4.

STEP-6: Then it is permuted with a weight value and then with some other key pair and taken as the first block.

STEP-7: Block A is taken as the second block and the block B is shifted by ‘s’ times and taken as the third block.

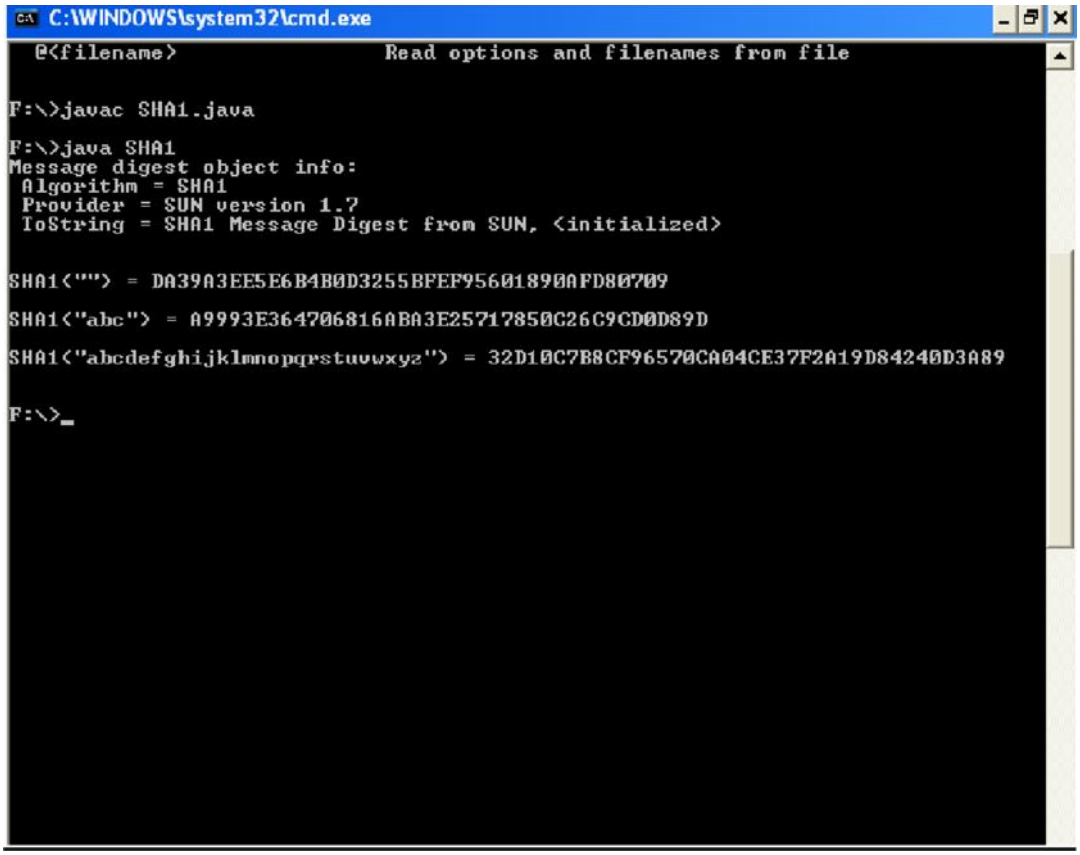
STEP-8: The blocks C and D are taken as the block D and E for the final output.

PROGRAM: (Secure Hash Algorithm)

```
import  java.security.*;
public class SHA1 {
     public static void main(String[] a) {
     try {
     MessageDigest md = MessageDigest.getInstance("SHA1");
     System.out.println("Message digest object info: ");
     System.out.println(" Algorithm = " +md.getAlgorithm());
     System.out.println(" Provider = " +md.getProvider());
     System.out.println(" ToString = " +md.toString());
     String input = "";
     md.update(input.getBytes());
     byte[] output = md.digest();
     System.out.println();
     System.out.println("SHA1(\""+input+"\") =
     +bytesToHex(output));
     input = "abc";
     md.update(input.getBytes());
     output = md.digest();
     System.out.println();
     System.out.println("SHA1(\""+input+"\") = "
     +bytesToHex(output));
     input = "abcdefghijklmnopqrstuvwxyz";
     md.update(input.getBytes());
     output = md.digest();
     System.out.println();
     System.out.println("SHA1(\"" +input+"\") = "
     +bytesToHex(output));
     System.out.println(""); }
     catch (Exception e) {
     System.out.println("Exception: " +e);
}
}
public static String bytesToHex(byte[] b)
{
     char hexDigit[] = {'0', '1', '2', '3', '4', '5', '6',
     '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};
     StringBuffer buf = new StringBuffer();
     for (int j=0; j<b.length; j++) {
```

```
        buf.append(hexDigit[(b[j] >> 4) & 0x0f]);
        buf.append(hexDigit[b[j] & 0x0f]); }
        return buf.toString(); }
}
```

OUTPUT:



RESULT:

Thus the SHA-1 hashing technique had been implemented successfully.

EX. NO: 9

IMPLEMENTATION OF DIGITAL SIGNATURE STANDARD

AIM:

To write a java program to implement the signature scheme named digital signature standard (Euclidean Algorithm).

ALGORITHM:

STEP-1: Alice and Bob are investigating a forgery case of x and y.

STEP-2: X had document signed by him but he says he did not sign that document digitally.

STEP-3: Alice reads the two prime numbers p and a.

STEP-4: He chooses a random co-primes alpha and beta and the x's original signature x.

STEP-5: With these values, he applies it to the elliptic curve cryptographic equation to obtain y.

STEP-6: Comparing this 'y' with actual y's document, Alice concludes that y is a forgery.

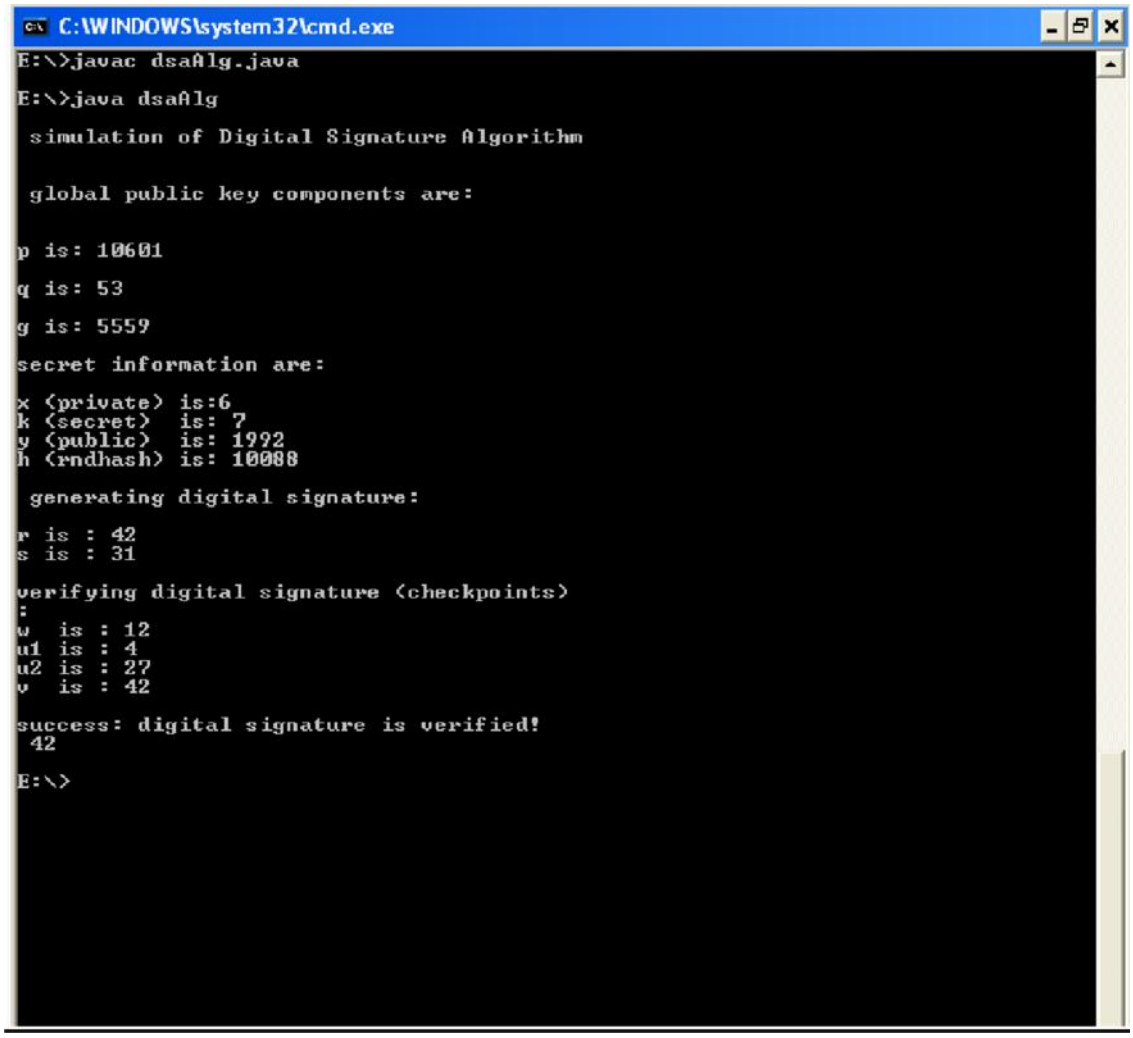
PROGRAM: (Digital Signature Standard)

```
import java.util.*;
import  java.math.BigInteger;
class dsaAlg {
     final static BigInteger one = new BigInteger("1");
     final static BigInteger zero = new BigInteger("0");
public static BigInteger getNextPrime(String ans)
{
     BigInteger test = new BigInteger(ans);
while (!test.isProbablePrime(99))
e:
{
     test = test.add(one);
}
     return test;
}
public static BigInteger findQ(BigInteger n)
{
     BigInteger start = new BigInteger("2");
while (!n.isProbablePrime(99))
{
     while  (!((n.mod(start)).equals(zero)))
     {
          start = start.add(one);
```

```
        }
        n = n.divide(start);
    }
        return n;
    }
    public static BigInteger getGen(BigInteger p, BigInteger q,
    Random r)
    {
        BigInteger h = new BigInteger(p.bitLength(), r);
        h = h.mod(p);
        return h.modPow((p.subtract(one)).divide(q), p);
    }
    public static void main (String[] args) throws
    java.lang.Exception
    {
        Random randObj = new Random();
        BigInteger p = getNextPrime("10600"); /* approximate
        prime */
        BigInteger q = findQ(p.subtract(one));
        BigInteger g = getGen(p,q,randObj);
        System.out.println(" \n simulation of Digital Signature
        Algorithm \n");
        System.out.println(" \n global public key components
        are:\n");
        System.out.println("\np is: " + p);
        System.out.println("\nq is: " + q);
        System.out.println("\ng is: " + g);
        BigInteger x = new BigInteger(q.bitLength(), randObj);
        x = x.mod(q);
        BigInteger y = g.modPow(x,p);
        BigInteger k = new BigInteger(q.bitLength(), randObj);
        k = k.mod(q);
        BigInteger r = (g.modPow(k,p)).mod(q);
        BigInteger hashVal = new BigInteger(p.bitLength(),
        randObj);
        BigInteger kInv = k.modInverse(q);
        BigInteger s = kInv.multiply(hashVal.add(x.multiply(r)));
        s = s.mod(q);
        System.out.println("\nsecret information are:\n");
        System.out.println("x (private) is:" + x);
        System.out.println("k (secret) is: " + k);
        System.out.println("y (public) is: " + y);
        System.out.println("h (rndhash) is: " + hashVal);
        System.out.println("\n generating digital signature:\n");
        System.out.println("r is : " + r);
        System.out.println("s is : " + s);
        BigInteger w = s.modInverse(q);
        BigInteger u1 = (hashVal.multiply(w)).mod(q);
        BigInteger u2 = (r.multiply(w)).mod(q);
        BigInteger v = (g.modPow(u1,p)).multiply(y.modPow(u2,p));
        v = (v.mod(p)).mod(q);
        System.out.println("\nverifying digital signature
        (checkpoints)\n:");
        System.out.println("w  is : " + w);
```

```
     System.out.println("u1 is : " + u1);
     System.out.println("u2 is : " + u2);
     System.out.println("v is : " + v);
if (v.equals(r))
{
     System.out.println("\nsuccess: digital signature is
     verified!\n " + r);
}
else
{
     System.out.println("\n error: incorrect digital
     signature\n ");
}
}
}
```

OUTPUT:



RESULT:

Thus the the signature scheme named digital signature standard (Euclidean Algorithm) had been implemented successfully.

IMPORTANT VIVA QUESTIONS

1. What is cryptocurrency?

Cryptocurrency is a digital or virtual currency that uses cryptography for security, allowing for secure and decentralized transactions without a central authority.

2. What is the purpose of cryptography?

Cryptography aims to ensure the security and confidentiality of information by encrypting or decrypting it to prevent unauthorised access.

3. What are some applications of cryptography?

Cryptography has applications in various fields, including Internet security, banking, and cybersecurity.

It allows secure data storage and transmission between parties and is crucial for protecting sensitive information.

4. What is the difference between symmetric and asymmetric key cryptography?

Symmetric key cryptography involves using the same key for encryption and decryption, while asymmetric key cryptography involves using two different keys, one for encryption and one for decryption.

5. Why is end-to-end encryption essential for protecting sensitive information?

End-to-end encryption ensures that the sender and receiver of a message are the only ones who can read and decrypt it, protecting sensitive information from interception by unauthorised parties.

6. What is blockchain technology?

Blockchain technology is a distributed ledger system that uses cryptography and hashing functions to secure data and ensure the integrity and immutability of transactions.

7. What is the Caesar cipher?

The Caesar cipher is a simple substitution cipher that shifts each letter in the plaintext by a certain number of positions down the alphabet to create the ciphertext.

8. What is a cipher?

A cipher is an algorithm used in cryptography to encrypt or decrypt data. It scrambles the data sounauthorised parties cannot read it without the decryption key.

9. What is symmetric encryption?

Symmetric encryption protects data on servers and data centres by using a single key for encryption and decryption.

This ensures messages are encrypted and decrypted correctly, preventing snooping on malicious actors.

10. What is the key exchange?

Key exchange is the process of securely exchanging encryption keys between two parties. This is done beforehand using algorithms like the Diffie-Hellman key exchange protocol to ensure that messages are encrypted and decrypted correctly.

11. What is private key cryptography?

Private key cryptography uses a single key for encryption and decryption, with both sender and receiver having a pre-shared secret key.

The secret key should not be sent along with the cipher text to avoid defeating the purpose of cryptography.

12. What are the two categories of ciphers used in public key cryptography?

The two categories of ciphers used in public key cryptography are stream ciphers and RC4, CELSA, and Panama.

Stream ciphers encrypt basic information one bit at a time, while RC4, CELSA, and Panama encrypt binary data.

13. What is a single-key structure in symmetric key cryptography?

A single-key structure in symmetric key cryptography is a simple structure that allows for easy transmission of a secret key to both sender and receiver.

This simplifies communication and maintenance, making it ideal for everyday communications.

14. What is the risk associated with using symmetric key cryptography?

A risk associated with symmetric key cryptography is key sharing, as both the sender and receiver must have a pre-shared secret key.

This cannot be easy to manage in large organizations or with many users.

15. What is asymmetric key encryption?

Asymmetric key encryption uses a double layer of protection, with a private and public key.

The public key encrypts information pre-transit, while the private key decrypts data post-transit. Both keys must belong to the receiver of the message.

16. What is a journalist's practical scenario when using symmetric key cryptography?

A journalist's practical scenario when using symmetric key cryptography would be sending encrypted data to Ryan, ensuring that the information cannot be decoded even if intercepted.

17. Who should belong to the keys in asymmetric key encryption?

The sender and receiver should belong to the keys in asymmetric key encryption. The keys must be kept private at all times to prevent unauthorized access.

18. What is the purpose of the private and public keys in asymmetric key encryption?

The private key in asymmetric encryption is used for decryption, while the public key is used for encryption.

19. What are the risks associated with symmetric key cryptography?

The risks associated with symmetric key cryptography include key sharing and the possibility of a single point of failure.

20. How does symmetric critical cryptography work?

Symmetric key cryptography relies on a single key for encryption and decryption, with both sender and receiver having a pre-shared secret key.

The private key should not be sent along with the cipher text to avoid defeating the purpose of cryptography.

21. What is symmetric encryption used for?

Symmetric encryption protects data on servers and data centres, ensuring minimal to no delay when recalled.

It plays a significant role in verifying website server authenticity, exchanging necessary encryption keys, and generating sessions for maximum security.

22. What does asymmetric key encryption provide the double layer of protection?

The double layer of protection provided by asymmetric key encryption includes pre-transit encryption using the public key and post-transit decryption using the private key.

23. How can asymmetric key encryption be used in a practical scenario?

Asymmetric key encryption can be used in a practical scenario to ensure the security of everyday communications, such as a journalist sending encrypted data to Ryan.

24. What is asymmetric cryptography?

Asymmetric cryptography is a secure method of transmitting ciphers text without a public key, allowing the receiver to decrypt it using their private key.

25. How does asymmetric cryptography solve the flaw in symmetric key cryptography?

Asymmetric cryptography solves the flaw in symmetric key cryptography by making it impossible to substitute other keys for decryption, allowing for frequent exchange of personal data using the same set of keys.

26. What is the purpose of hashing in asymmetric key cryptography?

Hashing, scrambling information beyond recognition, is another crucial advantage of asymmetric key cryptography. Hash functions generate a hash value or digest meant to be irreversible and cannot be converted back to their original value.

They are used for password storage, identity verification, and integrity checks.

27. What is the advantage of using asymmetric key cryptography over traditional symmetric encryptionmethods?

Asymmetric key cryptography offers advantages over traditional symmetric encryption methods, such as eliminating the need for a reliable key-sharing channel and ensuring confidentiality. It also

has more extensive key lengths, making it harder to break into via brute force.

28. How does asymmetric key cryptography ensure confidentiality?

Asymmetric key cryptography ensures confidentiality by using a public key for encryption and a private key for decryption, meaning only the intended recipient can read the message.

29. What is the tamper-proof characteristic of asymmetric key cryptography?

Asymmetric key cryptography has a tamper protection feature that prevents messages from being intercepted and changed without invalidating the private key used to encrypt the data.

30. What is hashing in asymmetric key cryptography?

Hashing is a crucial advantage of asymmetric key cryptography. It is the process of scrambling information beyond recognition, generating a hash value or digest.

These hashes are meant to be irreversible and cannot be converted back to their original value.

31. How is hashing used for password storage and identity verification?

Websites use hashing to store user passwords, which are then passed through the hash function when a user creates a new account.

If the hash value matches the one stored on the central server, the password is correct, preventing data breaches or hacks.

32. What is the use of asymmetric key cryptography in blockchain architecture?

Asymmetric key cryptography is used in blockchain architecture to authorize transactions and maintain the system.

If both ends approve, its two critical structures ensure changes are reflected across the network.

33. How does asymmetric key cryptography eliminate the need for a reliable key-sharing channel?

Asymmetric key cryptography eliminates the need for a reliable key-sharing channel because the public key can encrypt the message, and the private key can decrypt it.

34. How is asymmetric key cryptography used to monitor encrypted browsing sessions?

Asymmetric key cryptography is used to monitor encrypted browsing sessions by verifying websites of authenticity and exchanging encryption keys for maximum security.

35. What is a hash function?

A hash function is a set of mathematical calculations performed on two blocks of data, with the primary input divided into two blocks of similar size.

36. What is the block size in a hash function?

The block size depends on the algorithm used.

37. What are hash collisions?

Hash collisions are common issues where two users have the same password, but techniques like solving can help reduce them.

38. What is the solved value?

The solved value is a unique random keyword added to the input in the solving technique to reduce hash collisions.

39. What is pepping?

Pepping is a technique that adds a random string of data, called the paper, to input before passing it to the hash function to prevent hackers from cracking into all passwords.

40. What is the difference between DES and Crystal ciphers?

DES is a symmetric block cipher that implements substitution and permutation alternately. In contrast, Crystal ciphers are based on the Shannon structure and implement substitution and permutation alternately to combat brute force attacks.

Crystal ciphers are a backbone in the development of many symmetric block ciphers.

41. What is a public-wide competition?

A public-wide competition is a process where many people are invited to participate in a challenge or event, often to develop new technology or solutions to a problem.

In the context of encryption, public-wide competition can lead to the development of new and advanced encryption standards, such as the Vindale algorithm.

42. What is a crystal cipher?

A crystal cipher is a symmetric block cipher that implements substitution and permutation alternately, reducing redundancy and increasing complexity to combat brute force attacks.

The process involves dividing the encrypted block into two parts, using different encryption keys for each run, and reversing the decryption process to obtain the final cipher text.

This structure was a backbone in the development of many symmetric block ciphers.

43. When was the advanced encryption standard developed?

The advanced encryption standard was created in 2002. It replaced the previous encryption standard, D E S, which was deemed too slow for communication channels.

44. What is the Crystal cipher structure?

The Crystal cipher structure is a method of encryption that involves dividing plain text into two halves, passing each half through a function using a unique encryption key, and then using an XOR input with the left half of the initial plain text to decrypt the information.

45. What is the FISER cipher?

The FISER cipher is a block cipher used in the Crystal cipher structure. It is considered safest when the block size is large, but larger block sizes can slow down encryption and decryption speeds.

46. Compare symmetric and asymmetric encryption?

Symmetric encryption involves using the same key for encryption and decryption, while asymmetric encryption involves using two different keys, one for encryption and one for decryption.

47. What is the purpose of DES encryption?

DES encryption is a software application that converts 64-bit plain text into 64-bit cipher text using the Fistel cipher structure.

It is implemented in the DES algorithm and provides the cipher text.

48. What are the modes of operation for DES encryption?

DES encryption has five modes of operation for better execution speed: electronic codebooks, cipher block chaining, cipher feedback block, output feedback method, and counter method.

49. What is the dominance of DES encryption?

The dominance of DES encryption began in 2002 when the advanced encryption standard replaced it as the accepted standard.

The NIST officially withdrew the global acceptance standard in May 2005, but triple DES has been approved for sensitive government information through 2013.

50. What led to the change in the DES algorithm?

The DES algorithm was changed due to short key lengths and increased processing power of new computers.

As encryption power increases with crucial size, DES is no longer a challenge for packing computers.

51. What are some widely used encryption methods?

Some widely used encryption methods include electronic codebooks, cipher block chaining, cipher feedback block, output feedback method, and counter method.

52. What are the benefits of using a larger block size in a block cipher?

A larger block size in a block cipher provides better security but can also slow down encryption and decryption speeds.

53. What is the AES encryption algorithm?

The AES encryption algorithm is a symmetric block cipher that converts 64-bit plain text into 64-bit cipher text. It is considered a secure encryption algorithm and has been widely adopted as the standard for encryption.

It is also known as the Rijndael algorithm and runs through 10 to 256 rounds.

54. What is the NIST encryption standard?

The NIST encryption standard is a set of guidelines and specifications for encryption algorithms.

It was officially withdrawn in May 2005, but the Advanced Encryption Standard (AES) has replaced it as the accepted standard.

55. What is AES commonly used for in wireless security?

AES is commonly used in wireless security to establish a secure authentication mode between routers and clients for cryptography.

56. What is the Riddles algorithm used for?

The Riddles algorithm is widely used in securing WIFI endpoints, SSLTL S encryption, and file encryption.

57. What are the advantages of AES over DPSK?

AES has advantages in key length, block size, and complexity compared to DPSK. DPSK has a fixed number of fronts, while AES has a variable number of rounds based on the key length.

AES is considered simpler than DPSK but faster in encryption and decryption.

58. What is Private message encryption?

Private message encryption emerged as a solution to the key exchange issue with symmetric algorithms.

59. What is DSLG used for?

DSLG has been developed to address the critical exchange issue with symmetric algorithms.

60. How do digital signatures work?

Digital signatures authenticate and verify documents and data, preventing tampering and digital modification. They work on the public critical cryptography architecture, passing the original plaintext message to a hash function to create a digest.

The message is then encrypted using the sender's private key and decrypted using the public key.

The message is then interpreted and compared to the digest to verify data integrity.

61. What are the two primary algorithms for implementing digital signatures?

The two primary algorithms for implementing digital signatures are the RSL and DPSK algorithms.

62. What is the DPSK algorithm?

The DPSK algorithm is a FIPA standard proposed in 1991 and globally standardised in 1994 by the National Institute of Standards and Technology.

It uses mathematical functions to create a digital signature consisting of two 160-bit numbers generated from message digests and the private key.

63. What are the benefits of using DPSK?

DPSK offers three benefits: message authentication, integrity verification, and non-repudiation.

65. What is the difference between DES and AES?

AES is a direct successor to DES, providing key length, block size, and complexity advantages. DES has a fixed number of rounds, while AES has a variable number based on the key length.

66. What is the difference between the RSA and DSL algorithms?

The DSL algorithm is a FIPA standard proposed in 1991 and globally standardised in 1994 by the National Institute of Standards and Technology. It uses mathematical functions to create a digital signature consisting of two 160-bit numbers generated from message digests and the private key.

The RSA algorithm, on the other hand, is more difficult to forge and requires proper implementation.

67. What is the DSA algorithm?

The DSA algorithm is a robust and efficient signature verification method that uses a hash function to generate a digest H from a plain text message. It is patented, but NISD has made it available worldwide royalty-free.

68. What is the RSA algorithm?

The RSA algorithm is a cryptographic technique used for encryption and decryption of data.

It is beneficial for securing private information before being transmitted across communication challenges.

69. How does the RSA algorithm work?

The RSA algorithm generates a key pair consisting of a private key and a public key. The private key is kept secret and used for decryption, while the public key is shared and used for encryption.

Data is encrypted using the receiver's public key and decrypted using the receiver's private key.

70. What is the difference between RSA and DSA?

RSA is a symmetric encryption algorithm used for general data encryption and decryption, while DSA is an asymmetric algorithm used for signature verification.

DSA uses a hash function to generate a digest H from a plain text message, which is then encrypted using the sender's private key.

71. What is the purpose of the MD5 algorithm?

The MD5 algorithm is a one-way cryptographic function designed for secure cryptographic hashing to authenticate digital signatures. It is used to verify the integrity of data and show the sender.

72. What are the advantages of using RSA cryptography over other signature verification algorithms?

RSA cryptography offers several advantages over other signature verification algorithms, such as faster key generation, less storage space, and more robust security.

It also allows the receiver to use their public key for decryption, preventing the need for sharing a secret key.

73. What is RSA cryptography?

RSA cryptography is a method for encrypting and decrypting private information before being transmitted across communication challenges. It uses critical generation encryption and decryption functions.

The process starts with creating data and generating the key pair, which is then used for encryption and decryption.

74. What is the MD5 algorithm designed for?

The MD5 algorithm is designed to produce a 128-bit digest that appears random and is cryptographically secure.

The hash function must meet two requirements: it cannot be generated by an attacker and cannot produce two messages with the same hash value.

75. What is the plain-text requirement for the MD5 algorithm?

The plain text must be compatible with the hash function, with the size of the input string being 64-bit short of a multiple of 512.

76. What are the advantages of using the MD5 algorithm for password storage?

MD5 algorithm has a relatively low memory footprint and is easier to run on older hardware, making it suitable for storing user credentials on servers. Additionally, hashing requires less computational resources compared to other hash algorithms.

77. What are the requirements for the hash function in MD5?

The hash function must meet two requirements: it cannot be generated by an attacker and cannot produce two messages with the same hash value.

78. What is the output of the third iteration of MD5?

The output of the third iteration is added to a constant value derived from the continuous array k, which has 64 elements.

The next step involves a circular shift to create a unique digest for each input, with the output added to the buffer B and stored in the output register.

79. What is the difference between MD5 and other hash algorithms?

MD5 is a non-linear process that uses a 32-bit digest to generate a single output.

This method is more straightforward to compare and maintains computational complexity, making it more secure than other hash algorithms.

However, it exposes passwords to hackers and malicious actors, as they are stored in plain text format.

80. What are secure hash algorithms?

Secure hash algorithms are a family of cryptographic hash functions published by the National Institute of Standards and Technology and the National Security Agency.

Cryptography is an indispensable field of study for anyone pursuing cybersecurity. Cryptography uses mathematical algorithms and protocols to secure sensitive data while communicating safely over networks, and understanding its fundamental principles is vitally important in protecting personal privacy online.

This blog presents cryptography and network security lab viva questions and answers,introducingcryptography and network security interview questions with answers, from essential topics like encryption and decryption to more complex subjects such as hash functions and public key cryptography.

Cryptography is an ever-evolving field; new techniques and protocols emerge constantly, necessitating you to stay current on what’s new while expanding your knowledge base.

Therefore, stay aware of any new developments by keeping abreast of cryptographic news as it happens and keep learning!

Overall, this blog's cryptography and network security viva questionsare valuable for anyone preparing to attend a cryptography interview.

By studying and practising these concepts, you can increase your odds of success while making a meaningful contribution to cybersecurity.