DELHI TECHNOLOGICAL UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
(Formerly Delhi College of Engineering)
Bawana Road, Delhi- 110042

CO-306 Computer Networks
Project Report

Submitted to
Dr. Pawan Singh Mehra

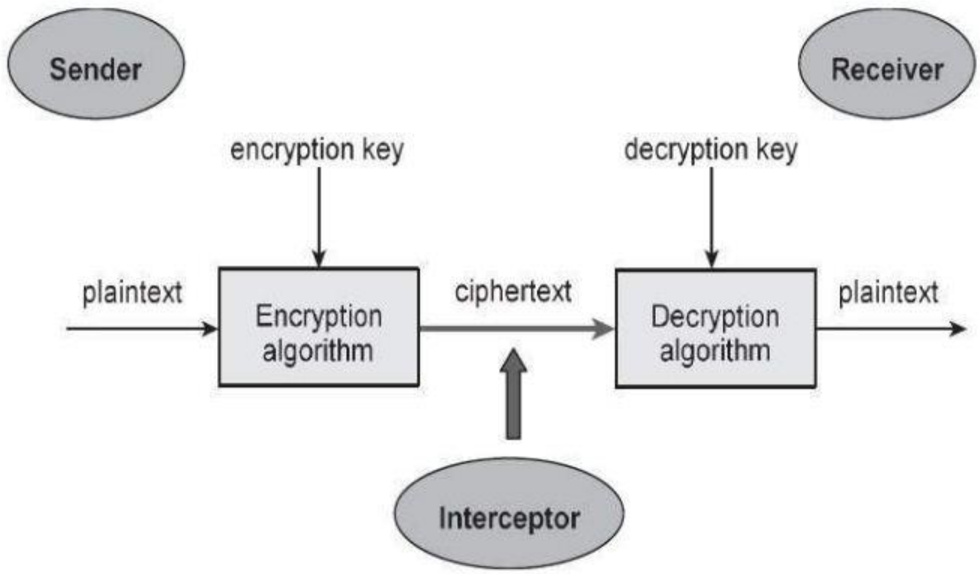
Submitted by

SOURABH
(2K18/CO/355)
Batch-A6 G1
3rd Year

SUNIL SAHU
(2K18/CO/363)
Batch-A6 G1
3rd Year

Cryptosystems

It is an execution of cryptographic practice and their associated communications to offer information safety measures. It is also known as a cipher system.



The figure demonstrates a dispatcher who needs to transport some insightful information to a recipient in such a move so as to any party interrupting on the communication conduit cannot take out the data.
The purpose of this straightforward cryptosystem is so as to at the end of the course, only the dispatcher and the recipient will be acquainted with the plaintext.

Components of a Cryptosystem

The range of mechanism of a fundamental cryptosystem is −

- **Plaintext.** The information to be sheltered throughout communication.

- **Encryption Algorithm.** It is a mathematical process that produces a cipher text for any given plaintext and encryption key. It is a cryptographic algorithm that takes plaintext and an encryption key as input and produces a cipher text.
- **Cipher text.** It is the scrambled version of the plaintext produced by the encryption algorithm using a specific the encryption key. The cipher text is not guarded. It flows on public channel. It can be intercepted or compromised by anyone who has access to the communication channel.
- **Decryption Algorithm,** It is a mathematical process, that produces a unique plaintext for any given cipher text and decryption key. It is a cryptographic algorithm that takes a cipher text and a decryption key as input, and outputs a plaintext. The decryption algorithm essentially reverses the encryption algorithm and is thus closely related to it.
  - **Encryption Key.** It is a value that is known to the sender. The sender inputs the encryption key into the encryption algorithm along with the plaintext in order to compute the cipher text.
  - **Decryption Key.** It is a value that is known to the receiver. The decryption key is related to the encryption key, but is not always identical to it. The receiver inputs the decryption key into the decryption algorithm along with the cipher text in order to compute the plaintext.

For a given cryptosystem, a collection of all possible decryption keys is called a key space.

An interceptor (an attacker) is an unauthorized entity who attempts to determine the plaintext. He can see the cipher text and may know the decryption algorithm. He, however, must never know the decryption key.

Types of Cryptosystems

Essentially, there are two categories of cryptosystems supported on the method in which encryption-decryption is agreed in the system −

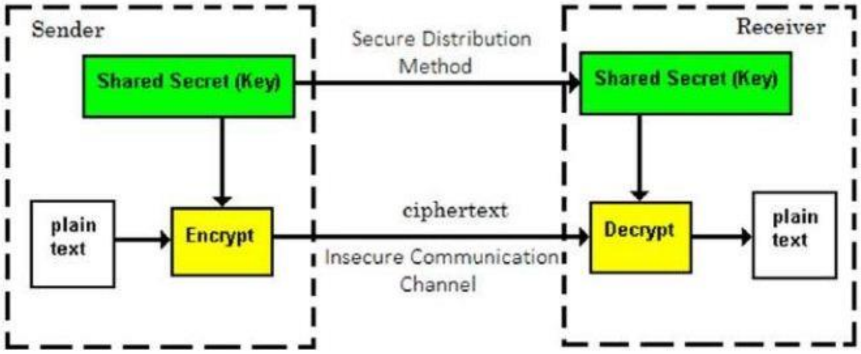
- **Symmetric Key Encryption**
- **Asymmetric Key Encryption**

The main difference between these cryptosystems is the relationship between the encryption and the decryption key.

Symmetric Key Encryption

The encryption procedure where similar keys are employed for encrypting and decrypting the data is recognized as Symmetric Key Encryption. Some examples of symmetric key encryption methods are − Digital Encryption Standard (DES), Triple-DES (3DES), IDEA, and BLOWFISH.

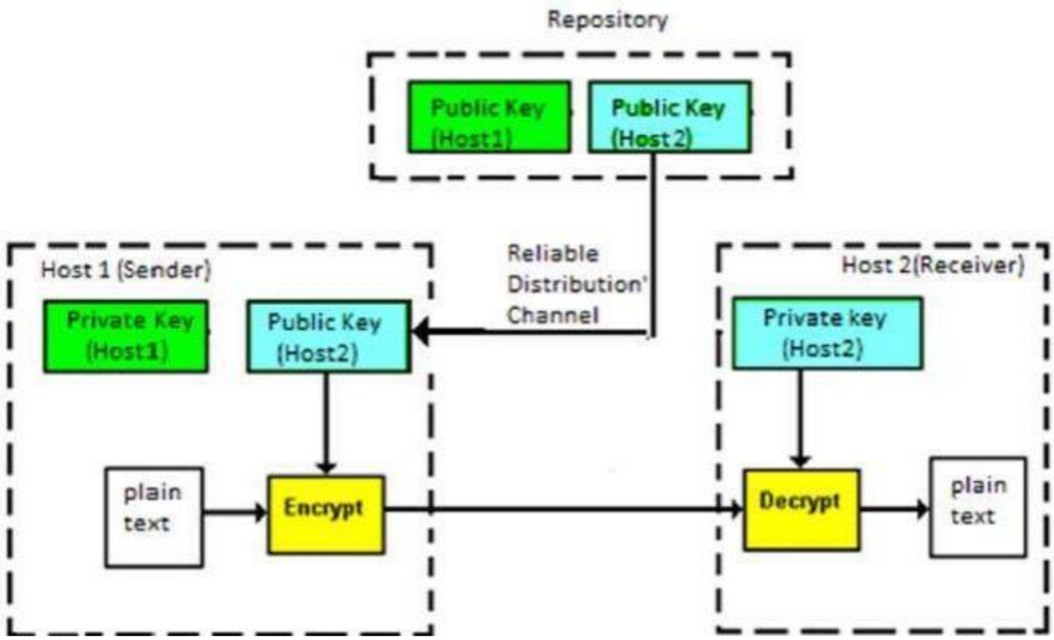
Shift cipher is an instance of a symmetric cryptosystem because the encryption shifts the alphabet through a specified issue which is then reversed for decryption. AES encryption is a further intricate example of symmetric encryption. They are used for subsequent locked key swaps in bilateral connections or in the majority frequent case of AES for individual encryption or encryption within an organization where all of the associates may bear the encryption (and therefore decryption) key so that they could read the inside documents but they stay sheltered as of exterior assaults.



Asymmetric Key Encryption

The encryption process where different keys are used for encrypting and decrypting the information is known as Asymmetric Key Encryption. Though the keys are different, they are mathematically related and hence, retrieving the plaintext by decrypting cipher text is feasible.

Public keys are only possible in the case of asymmetric cryptosystems, which are also hence called public-key cryptosystems. These are cryptosystems where the encryption key differs from the decryption key. These public-key or asymmetric cryptosystems scale well since anyone who wishes to receive messages simply needs to publish their public key and reference the encryption system they are using and secure conversations can then begin. Examples of asymmetric cryptosystems include the RSA and ElGamal cryptosystems.



What formulates cryptosystems as protected: One-way functions

The sanctuary of cryptosystems comes from the one-way nature of the encryption occupation. They could be mathematical utilities in way that it is competent to compute the assessment of the function specified as an input, except calculating the inverse (i.e. retrieving the input given the output) is practically infeasible or impossible.

DES Algorithm

The Data Encryption Standard (DES) was developed on the basis of a cryptographic algorithm (LUCIFER algorithm (Venus algorithm)) proposed by IBM researchers Horst Feistel and Walter Tuchman in the mid-1970s The DES algorithm is a typical Feistel structured block cipher algorithm. It's plaintext block length is 64 bits, and the key length is 64 bits. Among them, 8 bits of the key are parity, so the effective key length is 56 bits. DES algorithm encryption It uses the same process as decryption, and its security depends on a valid key. The DES algorithm first divides the plaintext that needs to be encrypted into each 64-bit data block, and encrypts each 64-bit data block with a 56-bit effective key. After each encryption performs 16 rounds of substitution and permutation on the input 64-bit plaintext data, the output is completely different from the plaintext 64-bit cipher text. It is suitable for software implementation on most computers, and it is also suitable for implementation on dedicated chips.

IMPLEMENTATION

1. Generating keys

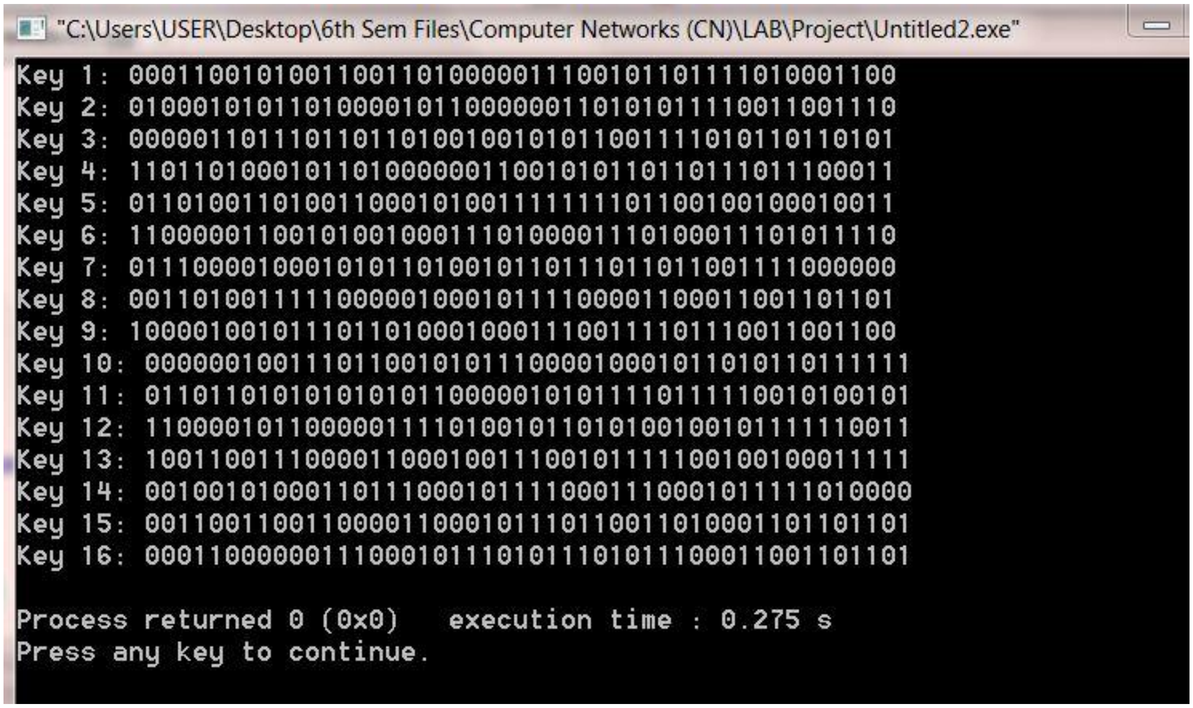
The algorithm engages sixteen rounds of encryption, with each round using a different key. Therefore, 16 keys are generated.

```
#include <iostream>
#include <string>
using namespace std;

string rk[16];
```

```
string SL(string KC){
   string sft="";
      for(int p = 1; p < 28; p++){
         sft += KC[p];
      }
      sft += KC[0];
   return sft;
}

string SL_T(string KC){
   string sft="";
   for(int p = 0; p < 2; p++){
      for(int q = 1; q < 28; q++){
         sft += KC[q];
      }
      sft += KC[0];
      KC= sft;
      sft ="";
   }
   return KC;
}
void GK(string key){
          // The P_C_1 table
          int P_C_1[56] = {
          57,49,41,33,25,17,9,
          1,58,50,42,34,26,18,
          10,2,59,51,43,35,27,
          19,44, 56,87,89,43,65,
          56, 7,8,9,1,2,4,6,78,89
          67,43,67,89,09,76,47,89,
          33,56,78,76,54,67,98,99,
          12,34,56,81,14,14,67,89,99,

          };
          // The P_C_2 table
          int P_C_2[48] = {
          14,17,11,24,1,5,
          3,28,15,6,21,10,
          23,19,12,4,26,8,
          56, 7,8,9,1,2,4,6,78,89
          67,43,67,89,09,76,47,89,
          33,56,78,76,54,67,98,99,
          12,34,56,81,14,14,67,89,99,
          };
```

```
	string PK ="";
	for(int p = 0; p < 56; p++){
		PK+= key[P_C_1[p]-1];
	}

	string left= PK.subr(0, 28);
	string right= PK.subr(28, 28);

	for(int p=0; p<16; p++){
		// 3.1. For rounds 1, 2, 9, 16 the KCs
		// are sft by one.
		if(p == 0 || p == 1 || p==8 || p==15 ){
			left= SL(left);
			right= SL(right);
		}

		else{
			left= SL_T(left);
			right= SL_T(right);
		}

	string c_k = left + right;
	string R_K = "";

	for(int p = 0; p < 48; p++){
		R_K += c_k[P_C_2[p]-1];
	}
	rk[p] = R_K;
		cout<<"Key "<<p+1<<": "<<rk[p]<<endl;
	}

}
int main(){
	string key = "1010101010111011000010010001100000100111001 1"
	"01101100110011011101";
	GK(key);

}
```

OUTPUT:



2. Encrypting plain text to obtain cipher text

The complete algorithm is put into practice. The plain text has been transposed, separated into two halves, and undergone sixteen rounds of encryption. It has been shared and transposed once more, which undoes the consequence of the first transposed function to get hold of the cipher text.

```
#include <iostream>
#include <string>
#include <cmath>
using namespace std;

string rk[16];

string plain;

string DEC_BIN(int DEC)
{
        string BIN;
   while(DEC != 0) {
                BIN = (DEC % 2 == 0 ? "0" : "1") + BIN;
```

```
                DEC = DEC/2;
        }
        while(BIN.length() < 4){
                BIN = "0" + BIN;
        }
    return BIN;
}

int BIN_DEC(string BIN)
{
    int DEC = 0;
        int ctr = 0;
        int sz = BIN.length();
        for(int p = sz-1; p >= 0; p--)
        {
        if(BIN[i] == '1'){
        DEC += pow(2, ctr);
        }
    ctr++;
        }
        return DEC;
}

string SL(string KC){
    string sft="";
        for(int p = 1; p < 28; p++){
            sft += KC[p];
        }
        sft += KC[0];
    return sft;
}

string SL_T(string KC){
    string sft="";
    for(int p = 0; p < 2; p++){
        for(int q = 1; q < 28; q++){
            sft += KC[q];
        }
        sft += KC[0];
        KC= sft;
        sft ="";
    }
    return KC;
}
```

```
string X(string m, string n){
	string result = "";
	int sz = n.sz();
	for(int p = 0; p < sz; p++){
		if(m[p] != n[p]){
			result += "1";
		}
		else{
			result += "0";
		}
	}
	return result;
}

void GK(string key){
	// The P_C_1 table
	int P_C_1[56] = {
	57,49,41,33,25,17,9,
	1,58,50,42,34,26,18,
	56, 7,8,9,1,2,4,6,78,89
	67,43,67,89,09,76,47,89,
	33,56,78,76,54,67,98,99,
	12,34,56,81,14,14,67,12

	};
	// The P_C_2 table
	int P_C_2[48] = {
	14,17,11,24,1,5,
	57,49,41,33,25,17,9,
	1,58,50,42,34,26,18,
	56, 7,8,9,1,2,4,6,78,89
	67,43,67,89,09,76,47,89,
	33,56,78,76,54,67,98,99,
	12,34,56,81,14,14,67,12
	44,49,39,56,34,53,
	46,42,50,36,29,32
	};
	// 1. Compressing the key using the P_C_1 table
	string PK ="";
	for(int p = 0; p < 56; p++){
		PK+= key[P_C_1[p]-1];
	}
	// 2. Dividing the key into two equal halves
```

```
        string left= PK.subr(0, 28);
        string right= PK.subr(28, 28);
        for(int p=0; p<16; p++){
                // 3.1. For rounds 1, 2, 9, 16 the KCs
                // are sft by one.
                if(p == 0 || p == 1 || p==8 || p==15 ){
                        left= SL(left);
                        right= SL(right);
                }
                // 3.2. For other rounds, the KCs
                // are sft by two
                else{
                        left= SL_T(left);
                        right= SL_T(right);
                }
                // Combining the two chunks
                string c_k = left + right;
                string R_K = "";
                // Finally, using the P_C_2 table to transpose the key bits
                for(int p = 0; p < 48; p++){
                        R_K += c_k[P_C_2[p]-1];
                }
                rk[p] = R_K;
        }

}
// Implementing the algorithm
string DES(){
        // The initial permutation table
        int ini_perm[64] = {
        58,50,42,34,26,18,10,2,
        60,52,44,36,28,20,12,4,
        57,49,41,33,25,17,9,
        1,58,50,42,34,26,18,
        56, 7,8,9,1,2,4,6,78,89
        67,43,67,89,09,76,47,89,
        33,56,78,76,54,67,98,99,
        12,34,56,81,14,14,67,12
        63,55,47,39,31,23,15,7
        };
        // The expansion table
        int ET[48] = {
        32,1,2,3,4,5,4,5,
        6,7,8,9,8,9,10,11,
```

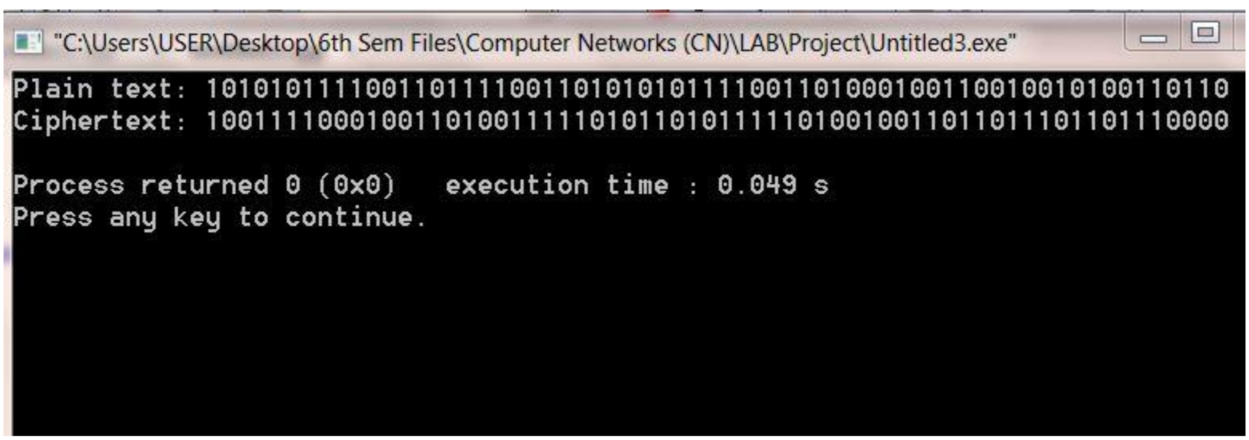
```
        57,49,41,33,25,17,9,
        1,58,50,42,34,26,18,
        56, 7,8,9,1,2,4,6,78,89
        67,43,67,89,09,76,47,89,
        33,56,78,76,54,67,98,99,
        12,34,56,81,14,14,67,12
        };


        int S_T[8][4][16]=
        {{
    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
},
{
    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
},
{
    10,0,9,14,6,3,15,5,1,13,12,7,11,4,2,8,
    13,17,01,91,13,14,16,11,21,81,51,4,2
    34,29,22,25,32,2,11,92,77,52,12,24,72,
    14,41,43,40,46,4,84,75,4,71,14,31,11,
},
{
    13,17,01,91,13,14,16,11,21,81,51,4,2
    34,29,22,25,32,2,11,92,77,52,12,24,72,
    14,41,43,40,46,4,84,75,4,71,14,31,11,
    3,15,0,6,10,1,13,8,9,4,5,11,12,7,2,14
},
{
    2,12,4,1,7,10,11,6,8,5,3,15,13,0,14,9,
    13,17,01,91,13,14,16,11,21,81,51,4,2
    34,29,22,25,32,2,11,92,77,52,12,24,72,
    14,41,43,40,46,4,84,75,4,71,14,31,11,

},
{
    12,1,10,15,9,2,6,8,0,13,3,4,14,7,5,11,
        13,17,01,91,13,14,16,11,21,81,51,4,2
```

```
    34,29,22,25,32,2,11,92,77,52,12,24,72,
    14,41,43,40,46,4,84,75,4,71,14,31,11,

  },
  {
   13,17,01,91,13,14,16,11,21,81,51,4,2
    34,29,22,25,32,2,11,92,77,52,12,24,72,
    14,41,43,40,46,4,84,75,4,71,14,31,11,
    6,11,13,8,1,4,10,7,9,5,0,15,14,2,3,12
  },
  {
    13,2,8,4,6,15,11,1,10,9,3,14,5,0,12,7,
   13,17,01,91,13,14,16,11,21,81,51,4,2
    34,29,22,25,32,2,11,92,77,52,12,24,72,
    14,41,43,40,46,4,84,75,4,71,14,31,11,

  }};

        int PER_T[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 inv_p[64]= {
        40,8,48,16,56,24,64,32,
        39,7,47,15,55,23,63,31,
        38,6,46,14,54,22,62,30,
        57,49,41,33,25,17,9,
        1,58,50,42,34,26,18,
        56, 7,8,9,1,2,4,6,78,89
        67,43,67,89,09,76,47,89,
        33,56,78,76,54,67,98,99,
        12,34,56,81,14,14,67,12
,
        33,1,41,9,49,17,57,25
        };

        string perm = "";
        for(int p = 0; p < 64; p++){
                perm += plain[ini_perm[p]-1];
        }
```

```
    string left = perm.subr(0, 32);
    string right = perm.subr(32, 32);
        for(int p=0; p<16; p++) {
    string R_E = "";

    for(int p = 0; p < 48; p++) {
        R_E += right[ET[p]-1];
};      string xed = X(rk[p], R_E);
        string res = "";


        for(int p=0;p<8; p++){

        string R1= xed.subr(p*6,1) + xed.subr(p*6 + 5,1);
        int row = BIN_DEC(R1);
        string C1 = xed.subr(p*6 + 1,1) + xed.subr(p*6 + 2,1) + xed.subr(p*6 + 3,1) +
xed.subr(p*6 + 4,1);;
            int column = BIN_DEC(C1);
            int value = S_T[p][row][column];
            res += DEC_BIN(value);
        }

        string P2 ="";
        for(int p = 0; p < 32; p++){
            P2 += res[PER_T[p]-1];
        }
                xed = X(P2, left);

        left = xed;
        if(p < 15){
            string tem = right;
            right = xed;
            left = tem;
        }
    }

    string com_t = left + right;
    string cip_txt ="";

    for(int p = 0; p < 64; p++){
        cip_txt+= com_t[inv_p[p]-1];
    }

    return cip_txt;
```

```
}
int main(){
			string key=
"1010101010111011000010010001100000100111001101101100110011011101";
			plain=
"1010101111001101111001101010101111001101000100110010010100110110";
			GK(key);
   cout<<"Plain text: "<<plain<<endl;
   string ct= DES();
   cout<<"Cip_txt: "<<ct<<endl;
}
```

OUTPUT:



3. Decrypting cipher text to obtain plain text

To decrypt the cipher text, overturn the order of the keys (i.e., key 16 becomes key 1, and so on) and apply the DES() function again.

```
#include <iostream>
#include <string>
#include <cmath>
using namespace std;

string rk[16];

string plain;
```

```
string DEC_BIN(int DEC)
{
        string BIN;
  while(DEC != 0) {
                BIN = (DEC % 2 == 0 ? "0" : "1") + BIN;
                DEC = DEC/2;
        }
        while(BIN.length() < 4){
                BIN = "0" + BIN;
        }
  return BIN;
}

int BIN_DEC(string BIN)
{
  int DEC = 0;
        int ctr = 0;
        int sz = BIN.length();
        for(int p = sz-1; p >= 0; p--)
        {
        if(BIN[p] == '1'){
        DEC += pow(2, ctr);
        }
  ctr++;
        }
        return DEC;
}

string SL(string KC){
  string sft="";
    for(int p = 1; p < 28; p++){
      sft += KC[p];
    }
    sft += KC[0];
  return sft;
}

string SL_T(string KC){
  string sft="";
  for(int p = 0; p < 2; p++){
    for(int q = 1; q < 28; q++){
      sft += KC[q];
    }
    sft += KC[0];
```

```
        KC= sft;
        sft ="";
    }
    return KC;
}

string X(string m, string n){
        string result = "";
        int sz = n.sz();
        for(int i = 0; i < sz; i++){
                if(m[i] != n[i]){
                        result += "1";
                }
                else{
                        result += "0";
                }
        }
        return result;
}

void GK(string key){
        // The P_C_1 table
        int P_C_1[56] = {
        57,49,41,33,25,17,9,
        1,58,50,42,34,26,18,
        57,49,41,33,25,17,9,
        1,58,50,42,34,26,18,
        56, 7,8,9,1,2,4,6,78,89
        67,43,67,89,09,76,47,89,
        33,56,78,76,54,67,98,99,
        12,34,56,81,14,14,67,12
        };
        // The P_C_2 table
        int P_C_2[48] = {
        14,17,11,24,1,5,
        3,28,15,6,21,10,
        23,19,12,4,26,8,
        57,49,41,33,25,17,9,
        1,58,50,42,34,26,18,
        56, 7,8,9,1,2,4,6,78,89
        67,43,67,89,09,76,47,89,
        33,56,78,76,54,67,98,99,
        12,34,56,81,14,14,67,12
        };
```

```
	string PK ="";
	for(int p = 0; p < 56; p++){
		PK+= key[P_C_1[p]-1];
	}

	string left= PK.subr(0, 28);
	string right= PK.subr(28, 28);
	for(int p=0; p<16; p++){

		if(p == 0 || p == 1 || p==8 || p==15 ){
			left= SL(left);
			right= SL(right);
		}

		else{
			left= SL_T(left);
			right= SL_T(right);
		}

		string c_k = left + right;
		string R_K = "";

		for(int p = 0; p < 48; p++){
			R_K += c_k[P_C_2[p]-1];
		}
		rk[p] = R_K;
	}

}

string DES(){

	int initial_permutation[64] = {
	58,50,42,34,26,18,10,2,
	60,52,44,36,28,20,12,4,
	57,49,41,33,25,17,9,
	1,58,50,42,34,26,18,
	56, 7,8,9,1,2,4,6,78,89
	67,43,67,89,09,76,47,89,
	33,56,78,76,54,67,98,99,
	12,34,56,81,14,14,67,12
	63,55,47,39,31,23,15,7
	};
```

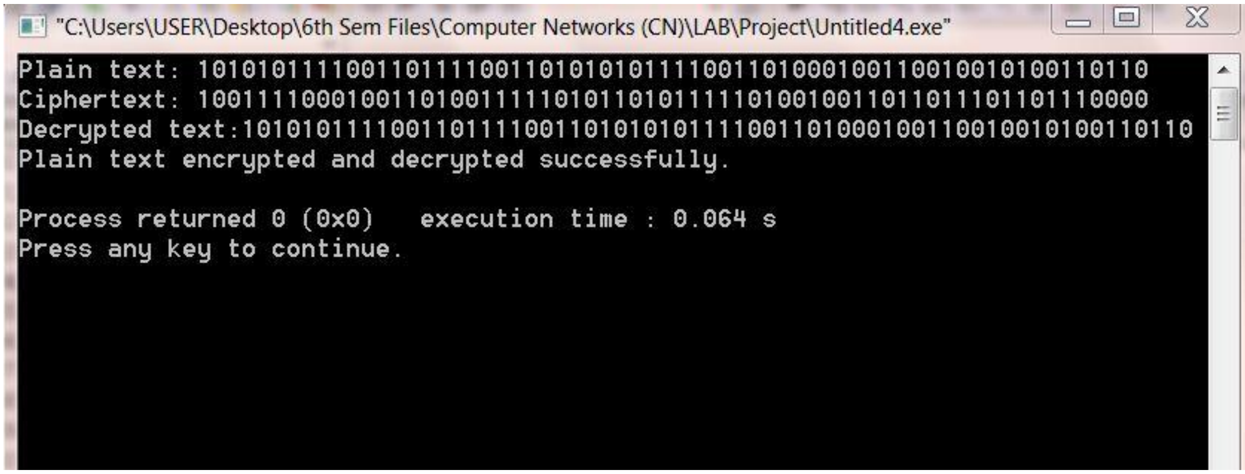
```
        int ET[48] = {
        32,1,2,3,4,5,4,5,
        57,49,41,33,25,17,9,
        1,58,50,42,34,26,18,
        56, 7,8,9,1,2,4,6,78,89
        67,43,67,89,09,76,47,89,
        33,56,78,76,54,67,98,99,
        12,34,56,81,14,14,67,12
        28,29,28,29,30,31,32,1
        };

        int S_T[8][4][16]=
        {{
    33,17,01,91,13,14,16,11,21,81,51,4,44
    34,29,22,25,32,2,11,92,77,52,12,24,72,
    14,41,43,40,46,4,84,75,4,71,14,31,11,
    15,12,8,2,4,9,1,7,5,11,3,14,10,0,6,13
  },
  {
   53,67,01,91,13,14,16,11,21,81,51,4,21
    34,29,22,25,32,2,11,92,77,52,12,24,72,
    14,41,43,40,46,4,84,75,4,71,14,31,11,
    13,8,10,1,3,15,4,2,11,6,7,12,0,5,14,9
  },
  {
    10,0,9,14,6,3,15,5,1,13,12,7,11,4,2,8,
     23,27,01,91,13,14,16,11,21,81,51,40,13
    34,29,22,25,32,2,11,92,77,52,12,24,72,
    14,41,43,40,46,4,84,75,4,71,14,31,11,

  },
  {
    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
  },
  {
    2,12,4,1,7,10,11,6,8,5,3,15,13,0,14,9,
     13,17,01,91,13,14,16,11,21,81,51,4,2
    34,29,22,25,32,2,11,92,77,52,12,24,72,
    14,41,43,40,46,4,84,75,4,71,14,31,11,
```

```
},
{
    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
},
{
    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
},
{
    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 PER_T[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 inv_p[64]= {
        40,8,48,16,56,24,64,32,
        39,7,47,15,55,23,63,31,
        38,6,46,14,54,22,62,30,
        57,49,41,33,25,17,9,
        1,58,50,42,34,26,18,
        56, 7,8,9,1,2,4,6,78,89
        67,43,67,89,09,76,47,89,
        33,56,78,76,54,67,98,99,
        12,34,56,81,14,14,67,12
        33,1,41,9,49,17,57,25
        };

        string perm = "";
        for(int p = 0; p < 64; p++){
                perm += plain[in_perm[p]-1];
        }
```

```
    string left = perm.subr(0, 32);
    string right = perm.subr(32, 32);

    for(int p=0; p<16; p++) {
    string R_E = "";

    for(int p = 0; p < 48; p++) {
        R_E += right[ET[p]-1];
 };     string xed = X(rk[p], R_E);
        string res = "";


        for(int p=0;p<8; p++){

        string R1= xed.subr(p*6,1) + xed.subr(p*6 + 5,1);
        int row = BIN_DEC(R1);
        string C1 = xed.subr(p*6 + 1,1) + xed.subr(p*6 + 2,1) + xed.subr(p*6 + 3,1) +
xed.subr(p*6 + 4,1);;
            int column = BIN_DEC(C1);
            int value = S_T[p][row][column];
            res += DEC_BIN(value);
        }

        string P2 ="";
        for(int p = 0; p < 32; p++){
            P2 += res[PER_T[p]-1];
        }

        xed = X(P2, left);

        left = xed;
        if(p < 15){
            string tem = right;
            right = xed;
            left = tem;
        }
    }

    string com_t = left + right;
    string cip_txt ="";

    for(int p = 0; p < 64; p++){
        cip_txt+= com_t[inv_p[p]-1];
```

```
	}

	return cip_txt;
}
int main(){
	string key=
"1010101010111011000010010001100000100111001101101100110011011101";
	// A block of plain text of 64 bits
	plain=
"1010101111001101111001101010101111001101000100110010010100110110";
	string aplain = plain;

	GK(key);
   cout<<"Plain text: "<<plain<<endl;

   string ct= DES();
   cout<<"Cip_txt: "<<ct<<endl;
	// Reversing the rk array for decryption
	int p = 15;
	int q = 0;
	while(p > q)
	{
		string temp = rk[p];
		rk[p] = rk[q];
		rk[q] = temp;
		p--;
		q++;
	}
	plain = ct;
	string decrypted = DES();
	cout<<"Decrypted text:"<<decrypted<<endl;

	if (decrypted == aplain){
		cout<<"Plain text encrypted and decrypted successfully."<<endl;
	}
}
```

OUTPUT:


"C:\Users\USER\Desktop\6th Sem Files\Computer Networks (CN)\LAB\Project\Untitled4.exe"
Plain text: 1010101111001101111001101010101111001101000100110010010100110110
Ciphertext: 1001111000100110100111110101101011111010010011011011101101110000
Decrypted text:1010101111001101111001101010101111001101000100110010010100110110
Plain text encrypted and decrypted successfully.
Process returned 0 (0x0)   execution time : 0.064 s
Press any key to continue.

RSA Algorithm

RSA is one of the first practicable public-key cryptosystems and is widely used for secure data transmission.

RSA Algorithm is used to encrypt and decrypt data in modern computer systems and other electronic devices. RSA algorithm is an asymmetric cryptographic algorithm as it creates 2 different keys for the purpose of encryption and decryption. It is public key cryptography as one of the keys involved is made public. RSA stands for Ron Rivest, Adi Shamir and Leonard Adleman who first publicly described it in 1978.

RSA makes use of prime numbers (arbitrary large numbers) to function. The public key is made available publicly (means to everyone) and only the person having the private key with them can decrypt the original message.

IMPLEMENTATION

```
#include <iostream>
  #include <math.h>
  using namespace std;

 int var(int y, int z)
 {
   int x;
   while (1)
   {
      x = y % z;
      if (x == 0)
         return z;
      y = z;
      z = x;
   }
```

```
}

int main()
{
  double f = 13;
  double g = 11;
  double h = f * g; //calculate h
  double track;
  double fhi = (f - 1) * (g - 1); //calculate fhi

  double enc = 7;

  //for checking that 1 < enc < fhi(h) and var(enc, fhi(h)) = 1; i.e., enc and fhi(h) are coprime.
  while (enc < fhi)
  {
    track = var(enc, fhi);
    if (track == 1)
      break;
    else
      enc++;
  }


  double d1 = 1 / enc;
  double d = fmod(d1, fhi);
  double message = 99;
  double c = fow(message, enc); //encrypt the message
  double m = fow(c, d);

  c = fmod(c, h);
  m = fmod(m, h);

  cout << "Original Message = " << message;
  cout << "\h"
     << "f = " << f;
  cout << "\h"
     << "g = " << g;
  cout << "\h"
     << "h = fg = " << h;
  cout << "\h"
     << "fhi = " << fhi;
  cout << "\h"
     << "enc = " << enc;
  cout << "\h"
```

```
     << "d = " << d;
  cout << "\h"
     << "Encrypted message = " << c;
  cout << "\h"
     << "Decrypted message = " << m;

  return 0;
}
```

OUTPUT: