SECURE STORAGE OF CRIME DATE USING HYBRID CRYPTOGRAPHY

TEAM MEMBERS:

1.M. RAGHU VAMSI (18BCE0006)
2.S. MAHENDRA (18BCE0009)
3.S. MOHITH (18BCE0013)
4.T. ADITYA REDDY (18BCE0045)
5. P. SAI TEJA (17BCE0976)

Report submitted for the
Final Project Review of
Course Code: CSE4003 – Cyber Security

Slot: C1+ TC1

Professor: MURUGAN.K

1. Abstarct:

Our main aim of the project is securing the data.In order to keep the data secured we propose a solution Hybrid Cryptography. When Sender tries to send the crime data to receiver, the data will be encrypted through symmetric encryption using symmetric key. Generally recipient will get symmetric key which he uses to decrypt the data. But we use hybrid cryptography to enhance security even more by encrypting the symmetric key through the asymmetric encryption and both encrypted symmetric key and encrypted crime data is transmitted to recipient. Recipient who has private key decrypts the encrypted symmetric key and further decrypts encrypted crime data using decrypted symmetric key and thus secured crime data reaches recipient safely.

2. Problem Statement:

In Police departments and other government organisations, data of criminals such as their background, record, and information related to their prison details and officers who handled them is stored in cloud which is highly confidential. But for the purpose of department, that data is transferred from one department to another. In this process that data can hacked by supporters of the criminals and can be corrupted. In order to keep this data secured we propose a solution Hybrid Cryptography.

3. Introduction:

Curiosity is one of the most common human traits, matched by the wish to conceal private information. Spies and the military all resort to information hiding to pass messages securely, sometimes deliberately including misleading information. Steganography, a mechanism for hiding information in apparently innocent pictures, may be used on its own or with other methods. Encryption fundamentally consists of scrambling a message so that its contents are not readily accessible while decryption is the reversing of that process. These processes depend on particular algorithms, known as ciphers. Suitably scrambled text is known as cipher text while the original is, not surprisingly, plain text. Readability is neither a necessary nor sufficient condition for something to be plain text. The original might well not make any obvious sense when read, as would be the case, for example, if something already encrypted were being further encrypted. It's also quite possible to construct a mechanism whose output is readable text but which actually bears no relationship to the unencrypted original. A key is used in conjunction with a cipher to encrypt or decrypt text. The key might appear meaningful, as would be the case with a character string used as a password, but this transformation is irrelevant, the functionality of a key lies in its being a string of bits determining the mapping of the plain text to the cipher text.

Cryptography is a technique to hide the data over communication channel. Securing Crime data over communication channel is very essential and it needs to be secured. At present, Police department stores data of crime records in cloud database. Although the cloud database improves the efficiency of use, it also poses a huge impact and challenge to the secure storage of data. We try to implement combination of Symmetric encryption and Asymmetric encryption to safely send the data over channel so that no crime data gets leaked.

In Police departments and other government organisations, data of criminals such as their background, record, and information related to their prison details and officers who handled them is stored in cloud which is highly confidential. But for the purpose of department, that data is transferred from one department to another. In this process that data can hacked by supporters of the criminals and can be corrupted. In order to keep this data secured we propose a solution Hybrid Cryptography.

4.Literature Review

[1] In this usage of cloud computing in data storage and data retrieval is described. The main issue described is the security of data stored in cloud. Cryptography and steganography techniques are more popular now a day’s for data security. Use of a single algorithm is not effective for high level security to data in cloud computing. In this paper the authors have introduced new security mechanism using symmetric key cryptography algorithm and steganography. In this proposed system AES, blowfish, RC6 and BRA algorithms are used to provide block wise security to data. All algorithm key size is 128bit. LSB steganography technique is introduced for key information security. Key information contains which part of file is encrypted using by which algorithm and key. File is split into eight parts. Each and every part of file is encrypted using different algorithm.

[2] As the cloud storage services are not secure by nature here a secure solution is provided. The risks are (a)data exposure(confidentiality), (b)data tampering(integrity) and (c)denial of access to data(availability). The solution provided is a service-oriented solution for provisioning secure storage service in the hybrid cloud environment, called Trust Store. The system is suitable to facilitate individual as well as collaborative data storage and access. The Trust Store system allows users to securely store and collaboratively share sensitive data in untrusted, public cloud storage environments. It does so by fragmenting, encrypting, and signing the data before uploading it to storage. This allows the system to store large data volumes cheaply with public storage providers and only requires trusted storage for very small data volumes to store keys and signatures.

[3] In the cloud environment, resources are shared among all of the servers, users and individuals. As a result, it is very easy for an intruder to access, misuse and destroy the original form of data. In case of compromise at any cost; entrusting cloud is of no use. A need for “practically strong and infeasible to get attacked” technique becomes vital. This paper presents the file security model which uses the concept of hybrid encryption scheme to meet security needs. In the proposed model, encryption and decryption of files at cloud servers

done using blowfish and modified version of RSA. Further, it is tested in cloud environment: Open Nebula. The above mentioned model is fruitful in data as a service, which can be extended in other service models of cloud. Also it is tested in cloud environment like Open Nebula, in future this can be deployed in other cloud environments and the best among of all can be chosen.

[4] As we know data stored in the cloud is increasingly gaining popularity for all users including personal, institutions and business purposes. The data is usually highly protected, encrypted and replicated depending on the security and scalability needs. Despite the advances in technology, the practical usefulness and longevity of cloud storage is limited in today's systems. This paper provides a solution to the problem of securely storing the client's data by maintaining the confidentiality and integrity of the data within the cloud. A data encryption model that is in charge of storing data in an encrypted format in the cloud is designed. To improve the efficiency of the designed architecture, the service in form of the model designed allows the users to choose the level of security of the data and according to this level different encryption algorithms are used. The hybrid of AES and Blowfish gives the properties of both algorithms thus making the formed hybrid algorithm much stronger to threats. This makes the formed hybrid system secure by increasingly adding the complexity functionalities.

[5] To improve the strength of these security algorithms, a new security protocol for on line transaction can be designed using combination of both symmetric and asymmetric cryptographic techniques. This protocol provides three cryptographic primitives such as integrity, confidentiality and authentication. These three primitives can be achieved with the help of Elliptic Curve Cryptography, Dual-RSA algorithm and Message Digest MD5. That is it uses Elliptic Curve Cryptography for encryption, Dual-RSA algorithm for authentication and MD-5 for integrity. This new security protocol has been designed for better security with integrity using a combination of both symmetric and asymmetric cryptographic techniques.

Authors and Year (Reference)	*Title (Study)*	*Concept / Theoretical model/ Framework*	*Methodology used/ Implementation*	*Dataset details/ Analysis*	*Relevant Finding*	*Limitations / Future Research/ Gaps identified*
Maitri, P. V., & Verma, A. (2016, March)	Cloud computing using hybrid cryptography algorithm	AES-3DES	AES and 3DES algorithms merging	Provided Data is stored using cloud	Usage of hybrid cryptography and DES	Less security in cloud
Nepal, S., Friedrich, C., Henry, L., & Chen, S. (2011,	Storage of file Service in the Hybrid Cloud	Hybrid cloud model	Key Management Service (KMS) Integrity Management	Random Data is stored using cloud	Related dataset	Trusted key and signature storage can be hacked

Dec)			Service (IMS)			
Swarna, C., & Eastaff (2013, April)	Implementation of Hybrid Cryptography Algorithm	Blowfish - merging	Blowfish Algorithm coupled with File Splitting and Merging mechanism	Some random set	It is fruitful in data as a service	Maximum run time
Mata, F., Kimwele, M., & Okeyo (2015, March)	Enhanced Data Storage in Cloud Computing Using Hybrid Cryptography	AES-Blowfish	Hybrid cryptography	Data stored in the cloud	Non vulnerable to threats.	Performing the same experiments using audio and video as well.
Subasree, S., & Sakthivel, N. K. (2010)	Design of a new Security Protocol using Hybrid Cryptography	DUAL RSA-CRT	Hybrid Protocol Architecture	Own data is taken	Security strength	Performance level

Table 1. Literature survey table

Innovation component in the project:

Our project itself is an innovative idea where security is strengthened when we used both symmetric encryption and asymmetric encryption. Normally we think of using one algorithm for encryption/decryption. But in our project, at first encryption of message is done using symmetric encryption (DES Algorithm) and the symmetric key used to encrypt the message is also encrypted using asymmetric encryption (RSA Algorithm) by receiving public key from receiver who wants to access the file. Now both encrypted message and encrypted symmetric key are sent to receiver. Receiver decrypts the encrypted symmetric key using RSA algorithm by allowing his private key to decrypt it. With obtained symmetric key, receiver further decrypts the encrypted message and he gets access to the message.

5. Implementation Platform/Tools:

Software Requirements:

Code Blocks

Windows/ Ubuntu OS

Terminal (for Ubuntu/Linux)

Hardware Requirements:

Processor: i3 or above

Quadcore

Dataset used:

a. **Where from you are taking your dataset?**
   No dataset is taken from internet or from some other sites because crime data is not officially allowed to access via sites. Thus we created our own dataset to execute the program.

b. **Is your project based on any other reference project (Stanford Univ. or MIT)?**
   Our project is not completely based on some other projects. Only Hybrid cryptography is similar but the combination of algorithms is different.

c. **How does your project differ from the reference project?**
   Hybrid cryptography is combination of symmetric encryption and asymmetric encryption. Other projects have used some other different combination of symmetric and asymmetric algorithms. But we used combination of DES and RSA algorithm.

Tools used:

We didn’t require any special tools to do this project except we needed some software requirements like code blocks and some hardware requirements.

6.System Model and Methods:

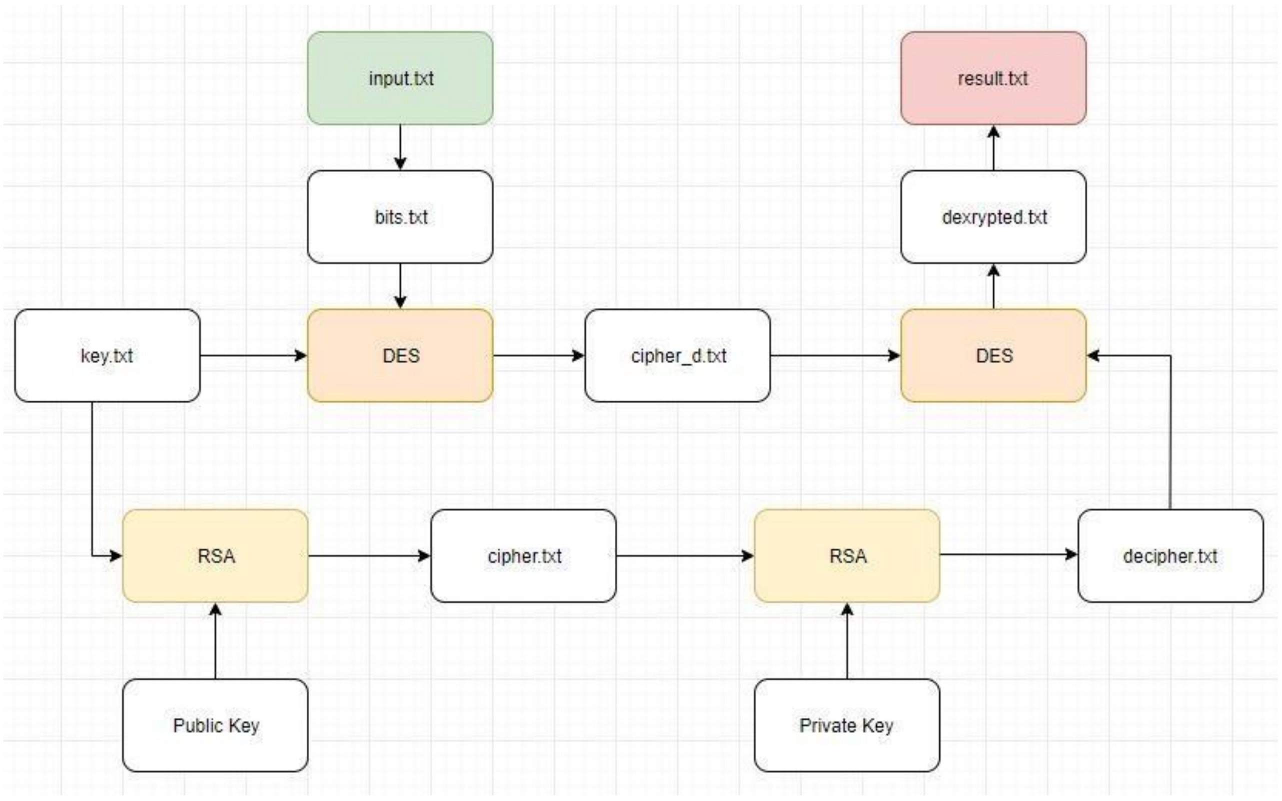
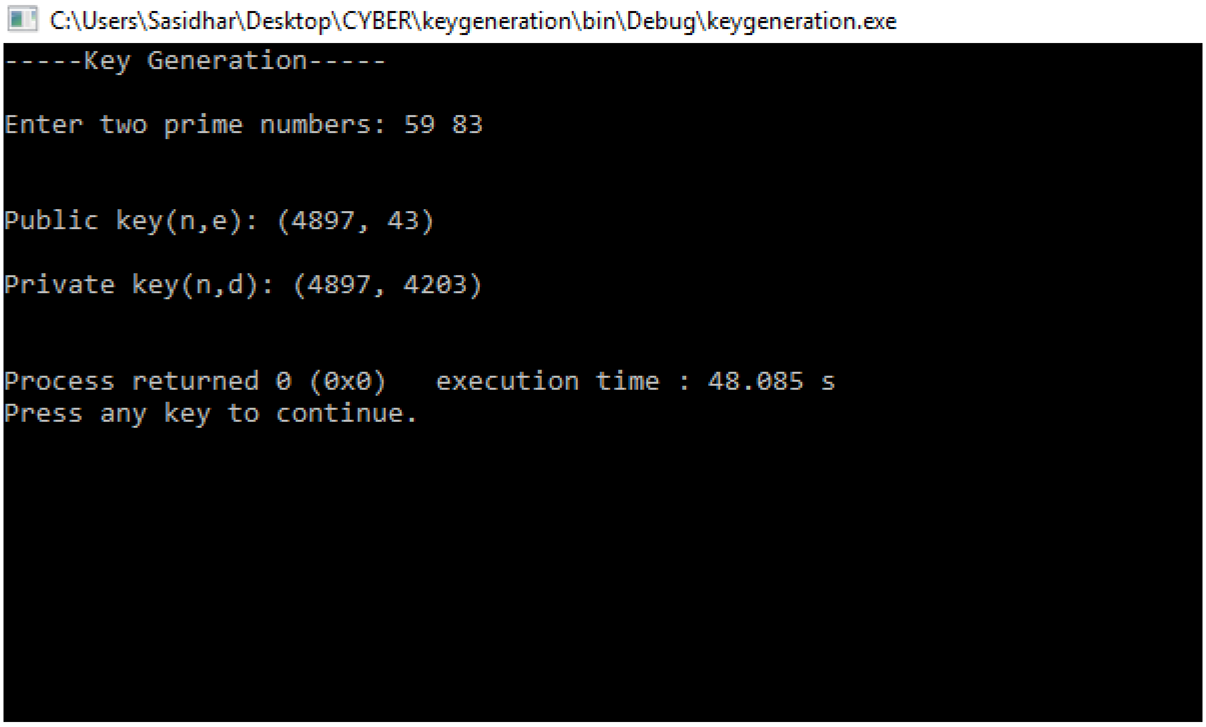


Fig.1. Methodology used

7.Pre and Post Processing:

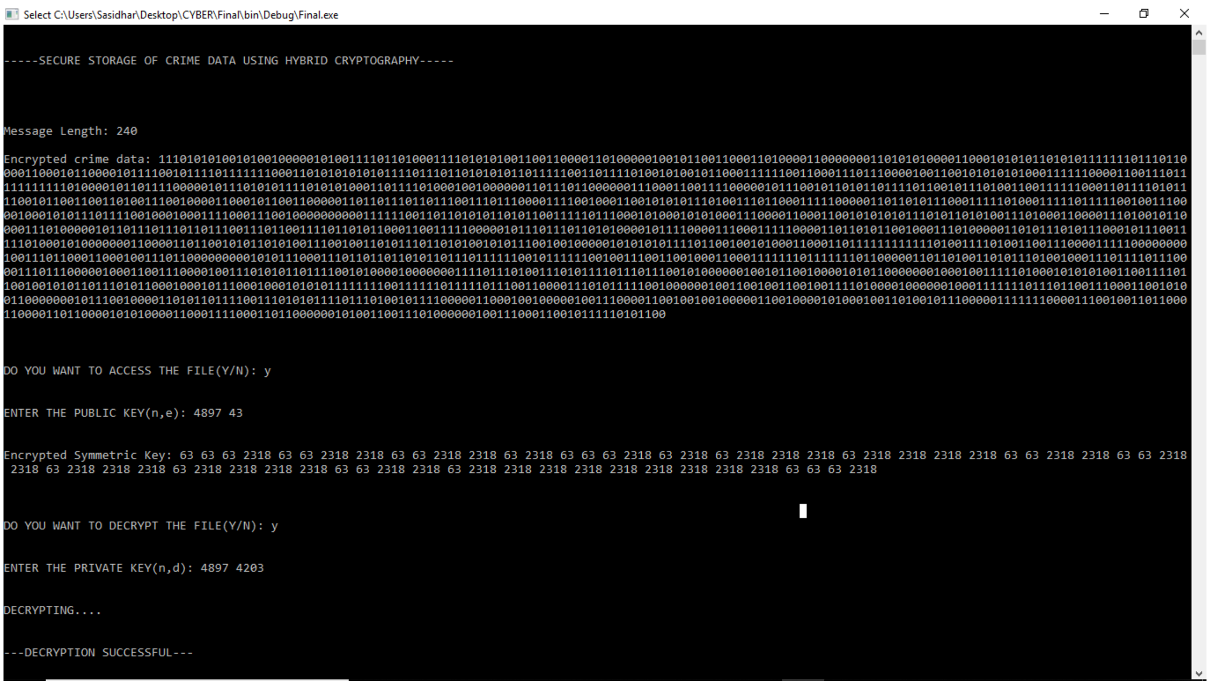
Receiver runs keygeneration.c to generate public key and private key



```
C:\Users\Sasidhar\Desktop\CYBER\keygeneration\bin\Debug\keygeneration.exe
-----Key Generation-----

Enter two prime numbers: 59 83


Public key(n,e): (4897, 43)

Private key(n,d): (4897, 4203)


Process returned 0 (0x0)   execution time : 48.085 s
Press any key to continue.
```

Fig.2. key generation output

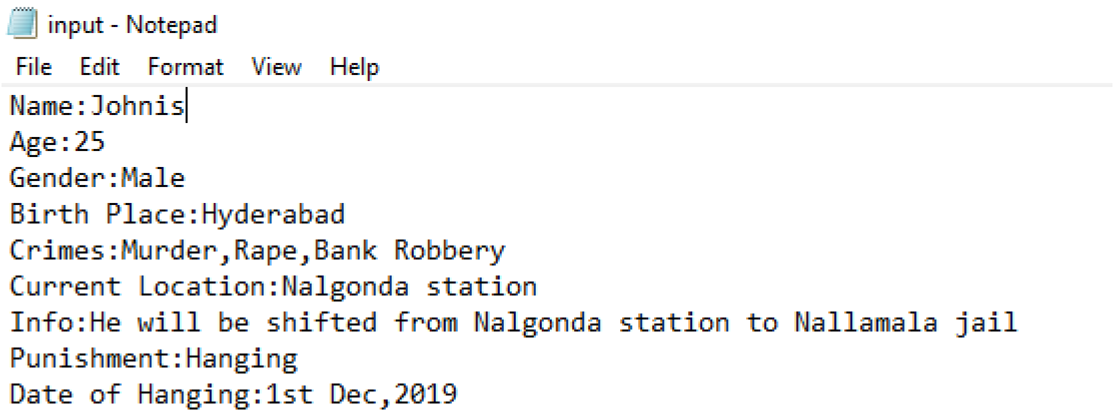
Screenshot of whole procedure of Hybrid Cryptography



```
Select C:\Users\Sasidhar\Desktop\CYBER\Final\bin\Debug\Final.exe

-----SECURE STORAGE OF CRIME DATA USING HYBRID CRYPTOGRAPHY-----



Message Length: 240

Encrypted crime data: 1110101010010100100000101001111011010001111010101001100110000110100000010010110011000110100001100000001101010100001100010101011010101111111101110110
0001100010110000101111001011110111111100011010101010101111011101101010101101111100110111110100101001011000111111001100011101110000100110010101010100011111100001100111011
1111111110100001011011110000010111010101111010101000110111101000100100000011011101100000001110001100111100000101110010110101101111011001011101001100111111000110111101011
1100101100110011010011100100001100010110011000001101101110110111001110111000011110010001100101010111010011101100011111000000110110101110001111101000111110111110010011100
0010001010111011110010001000111100011100100000000000111111001101101010110101100111110111000101000101010001110000110001100101010101110101101010011101000110000111010010110
0001110100000010110111011101101110011101100111101101011000110011111000001011101110110101000010111100001110001111100001101101011001000111010000011010111010111000101110011
1110100010100000000110000110110010101101010011100100110101110110101001010111001001000000101010101011110110010010100011000110111111111111101001111010011001110000111110000000000
1001110110001100010011101100000000010101110001110110110101101011101111110010111111001001110011001000110001111111011111111011000000110110100110101110100100011101111011100
0011101110000010001100111000010011101010110111100101000010000000011110111010011101011110111011100101000000010010110010000101011000000001000100111110100010101010011001111 01
1001001010110111010110001000101110001000101010111111110011111101111101110011000011101011111001000000010011001001100100111101000010000000100011111110111011001110001100101 0
0110000000101110010000110101101111001110101011110111010010111100000110001001000000100111000011001001001000000110010000101000100110100101110000001111110000111001001101100 0
110000110110000101010000110001111000110110000001010011001110100000001001110001100101111101 01100


DO YOU WANT TO ACCESS THE FILE(Y/N): y


ENTER THE PUBLIC KEY(n,e): 4897 43


Encrypted Symmetric Key: 63 63 63 2318 63 63 2318 2318 63 63 2318 2318 63 2318 63 63 63 2318 63 2318 63 2318 2318 2318 63 2318 2318 2318 2318 63 63 2318 2318 63 63 2318
 2318 63 2318 2318 2318 63 2318 2318 2318 2318 63 63 2318 2318 63 2318 2318 2318 2318 2318 2318 2318 2318 2318 63 63 63 2318


DO YOU WANT TO DECRYPT THE FILE(Y/N): y


ENTER THE PRIVATE KEY(n,d): 4897 4203


DECRYPTING....


---DECRYPTION SUCCESSFUL---
```

Fig.3. Final output

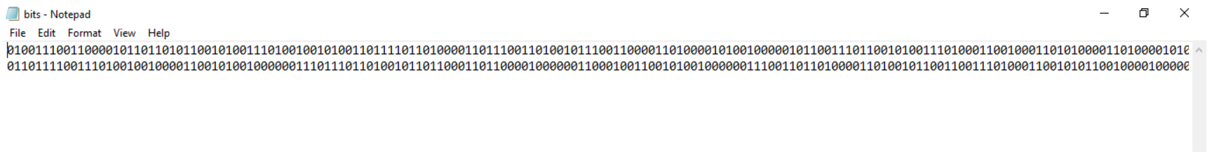
Input message (crime data) is given in input.txt



```
Name:Johnis
Age:25
Gender:Male
Birth Place:Hyderabad
Crimes:Murder,Rape,Bank Robbery
Current Location:Nalgonda station
Info:He will be shifted from Nalgonda station to Nallamala jail
Punishment:Hanging
Date of Hanging:1st Dec,2019
```

Fig.4. input.txt

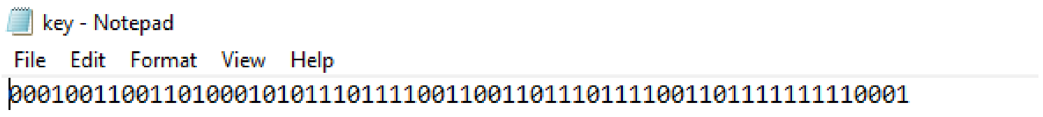
Binary form of input.txt is stored in bits.txt:



```
010011100110000101101101011001010011101001001010011011110110100001101110011010010111001100001101000010100100000101100111011001010011101000110010001101010000110100001010
011011110011101001001000011001010010000001110111011010010110110001101100001000000110001001100101001000000111001101101000011010010110011001110100011001010110010000100000
```

Fig.5. bits.txt

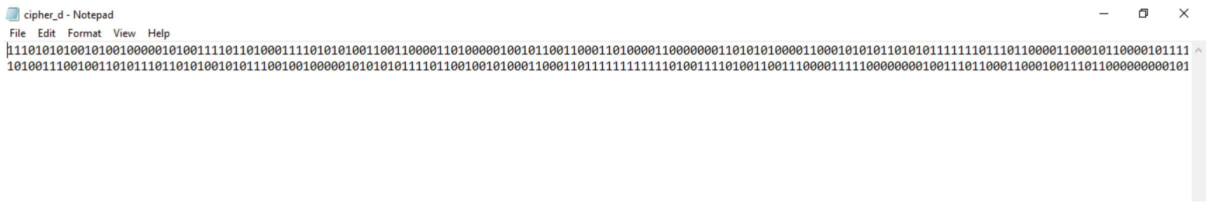
Symmetric key used to encrypt bits.txt is stored initially in key.txt



```
0001001100110100010101110111100110011011101111001101111111110001
```

Fig.6. key.txt

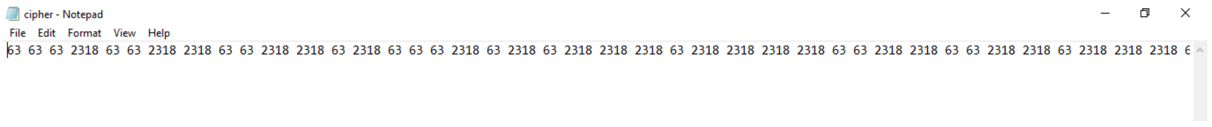
After DES encryption, encrypted bits.txt is stored in cipher_d.txt



```
1110101010010100100000101001111011010001111010101001100110000110100000100101100110001101000011000000011010101000011000101010110101011111110111011000011000101100001011111
10100111001001101011101101001010111001001000001010101011110110010010100011000110111111111111010011110100110011100001111100000000010011101100011000100111011000000000101
```

Fig.7.cipher_d.txt

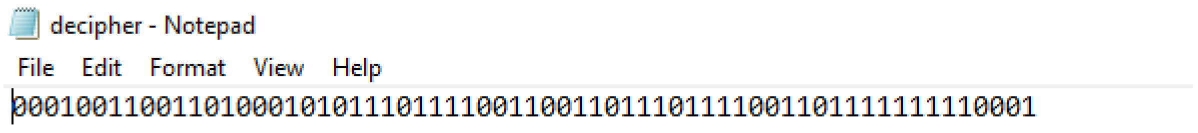
Using public key obtained from receiver, key.txt is encrypted (RSA encryption) and stored in cipher.txt



```
63 63 63 2318 63 63 2318 2318 63 63 2318 2318 63 2318 63 63 63 2318 63 2318 63 2318 2318 2318 63 2318 2318 2318 2318 63 63 2318 2318 63 63 2318 2318 63 2318 2318 2318 6
```

Fig.8. cipher.txt

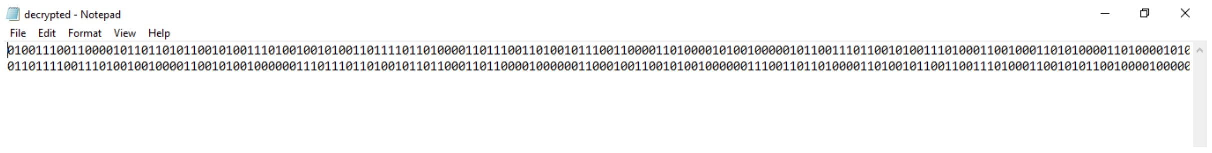
At receiver end, cipher.txt is decrypted using private key and stored in decipher.txt



```
0001001100110100010101110111100110011011101111001101111111110001
```

Fig.9. decipher.txt

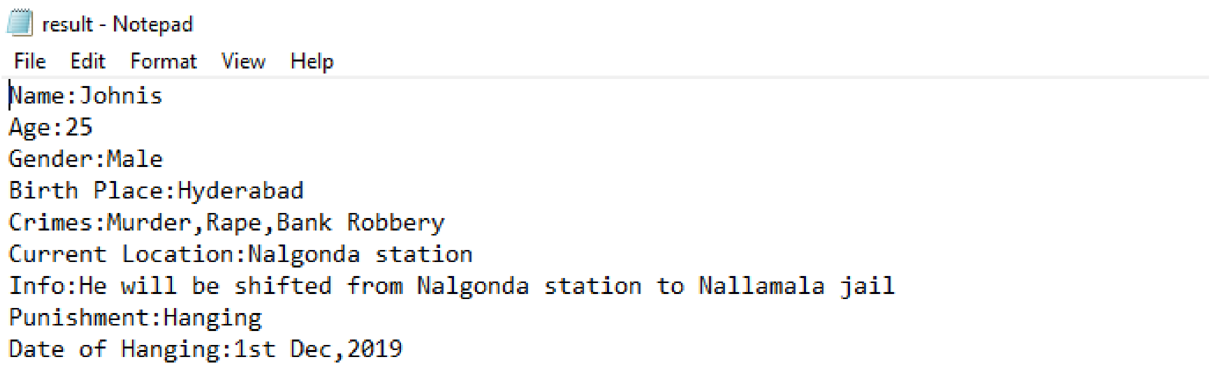
Using symmetric key from decipher.txt, cipher_d.txt is decrypted and stored in decrypted.txt



```
0100111001100001011011010110010100111010010010100110111101101000011011100110100101110011000011010000101001000000101100111011001010011101000110010001101010000110100001010
0110111100111010010010000110010100100000011101110110100101101100011011000010000000110001001100101001000000111001101101000011010010110011001110100011001010110010000100000
```

Fig.10. decrypted.txt

Finally, decrypted.txt is converted from binary to characters and stored in result.txt



```
Name:Johnis
Age:25
Gender:Male
Birth Place:Hyderabad
Crimes:Murder,Rape,Bank Robbery
Current Location:Nalgonda station
Info:He will be shifted from Nalgonda station to Nallamala jail
Punishment:Hanging
Date of Hanging:1st Dec,2019
```

Fig.11. result.txt

6. Implementation:

Final Code:

```
#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <math.h>

#include <time.h>
```

```
int IP[] =
{
        58, 50, 42, 34, 26, 18, 10, 2,
        60, 52, 44, 36, 28, 20, 12, 4,
        62, 54, 46, 38, 30, 22, 14, 6,
        64, 56, 48, 40, 32, 24, 16, 8,
        57, 49, 41, 33, 25, 17, 9, 1,
        59, 51, 43, 35, 27, 19, 11, 3,
        61, 53, 45, 37, 29, 21, 13, 5,
        63, 55, 47, 39, 31, 23, 15, 7
};

int E[] =
{
        32, 1, 2, 3, 4, 5,
         4, 5, 6, 7, 8,  9,
         8, 9, 10, 11, 12, 13,
        12, 13, 14, 15, 16, 17,
        16, 17, 18, 19, 20, 21,
        20, 21, 22, 23, 24, 25,
        24, 25, 26, 27, 28, 29,
        28, 29, 30, 31, 32,  1
};

int P[] =
{
        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 FP[] =
{
		40, 8, 48, 16, 56, 24, 64, 32,
		39, 7, 47, 15, 55, 23, 63, 31,
		38, 6, 46, 14, 54, 22, 62, 30,
		37, 5, 45, 13, 53, 21, 61, 29,
		36, 4, 44, 12, 52, 20, 60, 28,
		35, 3, 43, 11, 51, 19, 59, 27,
		34, 2, 42, 10, 50, 18, 58, 26,
		33, 1, 41, 9, 49, 17, 57, 25
};

int S1[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
};
```

```
int S2[4][16] =
{
        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
};

int S3[4][16] =
{
        10,  0,  9, 14,  6,  3, 15,  5,  1, 13, 12,  7, 11,  4,  2,  8,
        13,  7,  0,  9,  3,  4,  6, 10,  2,  8,  5, 14, 12, 11, 15,  1,
        13,  6,  4,  9,  8, 15,  3,  0, 11,  1,  2, 12,  5, 10, 14,  7,
         1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12
};

int S4[4][16] =
{
         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
};

int S5[4][16] =
{
         2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9,
            14, 11,  2, 12,  4,  7, 13,  1,  5,  0, 15, 10,  3,  9,  8,  6,
             4,  2,  1, 11, 10, 13,  7,  8, 15,  9, 12,  5,  6,  3,  0, 14,
            11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3
};

int S6[4][16] =
{
             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
};

int S7[4][16]=
{
             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
};

int S8[4][16]=
{
              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 PC1[] =
{
        57, 49, 41, 33, 25, 17,  9,
         1, 58, 50, 42, 34, 26, 18,
        10,  2, 59, 51, 43, 35, 27,
        19, 11,  3, 60, 52, 44, 36,
        63, 55, 47, 39, 31, 23, 15,
         7, 62, 54, 46, 38, 30, 22,
        14,  6, 61, 53, 45, 37, 29,
        21, 13, 5, 28, 20, 12, 4
};

int PC2[] =
{
        14, 17, 11, 24,  1,  5,
         3, 28, 15,  6, 21, 10,
        23, 19, 12,  4, 26,  8,
        16,  7, 27, 20, 13,  2,
        41, 52, 31, 37, 47, 55,
        30, 40, 51, 45, 33, 48,
        44, 49, 39, 56, 34, 53,
        46, 42, 50, 36, 29, 32
};

int SHIFTS[] = { 1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1 };

FILE *inp,*out,*in,*fptr;
```

```
int LEFT[17][32], RIGHT[17][32];
int IPtext[64];
int EXPtext[48];
int XORtext[48];
int X[8][6];
int X2[32];
int R[32];
int key56bit[56];
int key48bit[17][48];
int CIPHER[64];
int ENCRYPTED[64];

long int e, d, n;


void expansion_function(int pos, int text)
{
   int i;
          for (i = 0; i < 48; i++)
                    if (E[i] == pos + 1)
                              EXPtext[i] = text;
}

int initialPermutation(int pos, int text)
{
          int i;
          for (i = 0; i < 64; i++)
                    if (IP[i] == pos + 1)
```

```
                break;
        IPtext[i] = text;
}

int F1(int i)
{
        int r, c, b[6],j;
        for (j = 0; j < 6; j++)
                b[j] = X[i][j];

        r = b[0] * 2 + b[5];
        c = 8 * b[1] + 4 * b[2] + 2 * b[3] + b[4];
        if (i == 0)
                return S1[r][c];
        else if (i == 1)
                return S2[r][c];
        else if (i == 2)
                return S3[r][c];
        else if (i == 3)
                return S4[r][c];
        else if (i == 4)
                return S5[r][c];
        else if (i == 5)
                return S6[r][c];
        else if (i == 6)
                return S7[r][c];
        else if (i == 7)
                return S8[r][c];
```

```
}

int XOR(int a, int b)
{
        return (a ^ b);
}

void ToBits(int value)
{
        int k, j, m;
        static int i;
        if (i % 32 == 0)
                i = 0;
        for (j = 3; j >= 0; j--)
        {
                m = 1 << j;
                k = value & m;
                if (k == 0)
                        X2[3 - j + i] = '0' - 48;
                else
                        X2[3 - j + i] = '1' - 48;
        }
        i = i + 4;
}

void SBox(int XORtext[])
{
        int k = 0,i,j;
        for (i = 0; i < 8; i++)
                for (j = 0; j < 6; j++)
                        X[i][j] = XORtext[k++];

        int value;
        for (i = 0; i < 8; i++)
        {
                value = F1(i);
                ToBits(value);
        }
}

void PBox(int pos, int text)
{
        int i;
        for (i = 0; i < 32; i++)
                if (P[i] == pos + 1)
                        break;
        R[i] = text;
}

void cipher(int Round, int mode)
{
    int i;
        for (i = 0; i < 32; i++)
                expansion_function(i, RIGHT[Round - 1][i]);

        for (i = 0; i < 48; i++)
        {
                if (mode == 0)
                        XORtext[i] = XOR(EXPtext[i], key48bit[Round][i]);
                else
                        XORtext[i] = XOR(EXPtext[i], key48bit[17 - Round][i]);
        }

        SBox(XORtext);

        for (i = 0; i < 32; i++)
                PBox(i, X2[i]);
        for (i = 0; i < 32; i++)
                RIGHT[Round][i] = XOR(LEFT[Round - 1][i], R[i]);
}

void finalPermutation(int pos, int text)
{
        int i;
        for (i = 0; i < 64; i++)
                if (FP[i] == pos + 1)
                        break;
        ENCRYPTED[i] = text;
}

void convertToBinary(int n)
{
        int k, m,i;
        for (i = 7; i >= 0; i--)
        {
                m = 1 << i;
                k = n & m;
                if (k == 0)
                        fprintf(out, "0");
                else
                        fprintf(out, "1");
        }
}

int convertCharToBit(long int n)
{
        inp = fopen("input.txt", "rb");
        out = fopen("bits.txt", "wb+");
        char ch;
        int i = n * 8;
        while (i)
        {
                ch = fgetc(inp);
                if (ch == -1)
                        break;
                i--;
                convertToBinary(ch);
        }
        fclose(out);
        fclose(inp);
}
```

```
void Encryption(long int plain[])
{
   int i,k;
          out = fopen("cipher_d.txt", "ab+");
          for (i = 0; i < 64; i++)
                    initialPermutation(i, plain[i]);

          for (i = 0; i < 32; i++)
                    LEFT[0][i] = IPtext[i];
          for (i = 32; i < 64; i++)
                    RIGHT[0][i - 32] = IPtext[i];

          for (k = 1; k < 17; k++)
          {
                    cipher(k, 0);

                    for (i = 0; i < 32; i++)
                              LEFT[k][i] = RIGHT[k - 1][i];
          }

          for (i = 0; i < 64; i++)
          {
                    if (i < 32)
                              CIPHER[i] = RIGHT[16][i];
                    else
                              CIPHER[i] = LEFT[16][i - 32];
                    finalPermutation(i, CIPHER[i]);
          }
```

```
        for (i = 0; i < 64; i++)
                fprintf(out, "%d", ENCRYPTED[i]);
        fclose(out);
}

long int findFileSize()
{
        inp = fopen("input.txt", "rb");
        long int size;
        if (fseek(inp, 0L, SEEK_END))
                perror("fseek() failed");
        else // size will contain no. of chars in input file.
                size = ftell(inp);
        fclose(inp);

        return size;
}

long int findFileSize2()
{
        inp = fopen("cipher_d.txt", "rb");
        long int size;
        if (fseek(inp, 0L, SEEK_END))
                perror("fseek() failed");
        else // size will contain no. of chars in input file.
                size = ftell(inp);
        fclose(inp);
        return size;
}

long int findFileSize3()
{
        inp = fopen("decrypted.txt", "rb");
        long int size;
        if (fseek(inp, 0L, SEEK_END))
                perror("fseek() failed");
        else // size will contain no. of chars in input file.
                size = ftell(inp);
        fclose(inp);

        return size;
}

void Decryption(long int plain[])
{
  int i,k;
        out = fopen("decrypted.txt", "ab+");
        for (i = 0; i < 64; i++)
                initialPermutation(i, plain[i]);

        for (i = 0; i < 32; i++)
                LEFT[0][i] = IPtext[i];

        for (i = 32; i < 64; i++)
```

```
                RIGHT[0][i - 32] = IPtext[i];


        for (k = 1; k < 17; k++) {
                cipher(k, 1);


                for (i = 0; i < 32; i++)
                        LEFT[k][i] = RIGHT[k - 1][i];
        }
        for (i = 0; i < 64; i++)
        {
                if (i < 32)
                        CIPHER[i] = RIGHT[16][i];
                else
                        CIPHER[i] = LEFT[16][i - 32];
                finalPermutation(i, CIPHER[i]);
        }
        for (i = 0; i < 64; i++)
        fprintf(out, "%d", ENCRYPTED[i]);
        fclose(out);
}


void convertToBits(int ch[])
{
        int value = 0,i;
        for (i = 7; i >= 0; i--)
                value += (int)pow(2, i) * ch[7 - i];
        fprintf(out, "%c", value);
}
```

```
void bittochar()
{
    int i;
            out = fopen("result.txt", "ab+");
            for (i = 0; i < 64; i = i + 8)
                        convertToBits(&ENCRYPTED[i]);
            fclose(out);
}

void key56to48(int round, int pos, int text)
{
            int i;
            for (i = 0; i < 56; i++)
                        if (PC2[i] == pos + 1)
                                    break;
            key48bit[round][i] = text;
}

void key64to56(int pos, int text)
{
            int i;
            for (i = 0; i < 56; i++)
                        if (PC1[i] == pos + 1)
                                    break;
            key56bit[i] = text;
}
```

```
void key64to48(unsigned int key[])
{
        int k, backup[17][2];
        int i,x,j;
        int CD[17][56];
        int C[17][28], D[17][28];

        for (i = 0; i < 64; i++)
                key64to56(i, key[i]);

        for (i = 0; i < 56; i++)
                if (i < 28)
                        C[0][i] = key56bit[i];
                else
                        D[0][i - 28] = key56bit[i];

        for (x = 1; x < 17; x++)
        {
                int shift = SHIFTS[x - 1];

                for (i = 0; i < shift; i++)
                        backup[x - 1][i] = C[x - 1][i];
                for (i = 0; i < (28 - shift); i++)
                        C[x][i] = C[x - 1][i + shift];
                k = 0;
                for (i = 28 - shift; i < 28; i++)
                        C[x][i] = backup[x - 1][k++];
```

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

                backup[x - 1][i] = D[x - 1][i];

            for (i = 0; i < (28 - shift); i++)

                D[x][i] = D[x - 1][i + shift];

            k = 0;

            for (i = 28 - shift; i < 28; i++)

                D[x][i] = backup[x - 1][k++];

        }


        for (j = 0; j < 17; j++)

        {

            for (i = 0; i < 28; i++)

                CD[j][i] = C[j][i];

            for (i = 28; i < 56; i++)

                CD[j][i] = D[j][i - 28];

        }


        for (j = 1; j < 17; j++)

            for (i = 0; i < 56; i++)

                key56to48(j, i, CD[j][i]);

    }


    void decrypt(long int n)

    {

        in = fopen("cipher_d.txt", "rb");

        long int plain[n * 64];

        int i = -1;

        char ch;
        while (!feof(in))
        {
                ch = getc(in);
                plain[++i] = ch - 48;
        }

        for (i = 0; i < n; i++)
        {
                Decryption(plain + i * 64);
                bittochar();
        }
        fclose(in);
}

void encrypt(long int n)
{
        in = fopen("bits.txt", "rb");

        long int plain[n * 64];
        int i = -1;
        char ch;

        while (!feof(in))
        {
                ch = getc(in);
                plain[++i] = ch - 48;
        }
        for (i = 0; i < n; i++)
                Encryption(plain + 64 * i);

        fclose(in);
}

void create16Keys()
{
        FILE* pt = fopen("key.txt", "rb");
        unsigned int key[64];
        int i = 0, ch;

        while (!feof(pt))
        {
                ch = getc(pt);
                key[i++] = ch - 48;
        }

        key64to48(key);
        fclose(pt);
}
void create16Keys_d()
{
        FILE* pt = fopen("decipher.txt", "rb");
        unsigned int key[64];
        int i = 0, ch;
        while (!feof(pt))
        {
                ch = getc(pt);
                key[i++] = ch - 48;
        }

        key64to48(key);
        fclose(pt);
}

int gcd(int a, int b)
{
        int q, r1, r2, r;

        if (a > b) {
                r1 = a;
                r2 = b;
        }
        else {
                r1 = b;
                r2 = a;
        }

        while (r2 > 0) {
                q = r1 / r2;
                r = r1 - q * r2;
                r1 = r2;
                r2 = r;
        }

        return r1;
}

int PrimarityTest(int a, int i)
{
        int n = i - 1;
        int k = 0;
        int j, m, T;

        while (n % 2 == 0) {
                k++;
                n = n / 2;
        }

        m = n;
        T = FindT(a, m, i);

        if (T == 1 || T == i - 1)
                return 1;

        for (j = 0; j < k; j++) {
                T = FindT(T, 2, i);
                if (T == 1)
                        return 0;
                if (T == i - 1)
                        return 1;
        }
        return 0;
}

int FindT(int a, int m, int n)
{
        int r;
        int y = 1;

        while (m > 0) {
                r = m % 2;
                FastExponention(r, n, &y, &a);
                m = m / 2;
        }
        return y;
}

int FastExponention(int bit, int n, int* y, int* a)
{
        if (bit == 1)
                *y = (*y * (*a)) % n;

        *a = (*a) * (*a) % n;
}

int inverse(int a, int b)
{
        int inv;
        int q, r, r1 = a, r2 = b, t, t1 = 0, t2 = 1;

        while (r2 > 0) {
                q = r1 / r2;
                r = r1 - q * r2;
                r1 = r2;
                r2 = r;

                t = t1 - q * t2;
                t1 = t2;
                t2 = t;
        }

        if (r1 == 1)
                inv = t1;

        if (inv < 0)
                inv = inv + a;

        return inv;
}

int Encryption_r(int value, FILE* out)
{
        int cipher;
        cipher = FindT(value, e, n);
        fprintf(out, "%d ", cipher);
}
```

```
void Decryption_r(int value, FILE* out)
{
        int decipher;
        decipher = FindT(value, d, n);
        fprintf(out, "%c", decipher);
}

int gcd2(int a, int b)
{
    if (a == 0)
        return b;
    return gcd(b % a, a);
}

int phi(unsigned int v)
{
    int r;
    unsigned int result = 1;
    for (r= 2; r < v; r++)
        if (gcd2(r, v) == 1)
            result++;
    return result;
}

int main()
{
```

```
            // destroy contents of these files (from previous runs, if any)

            out = fopen("result.txt", "wb+");

            fclose(out);

            out = fopen("decrypted.txt", "wb+");

            fclose(out);

            out = fopen("cipher_d.txt", "wb+");

            fclose(out);

    out = fopen("cipher.txt", "w+");

            fclose(out);

            out = fopen("decipher.txt", "w+");

            fclose(out);

            printf("\n\n-----SECURE    STORAGE    OF    CRIME    DATA    USING    HYBRID
CRYPTOGRAPHY ---- \n\n\n");

            create16Keys();


            long int y = findFileSize() / 8;


            if(findFileSize()%8!=0)

    {

        printf("\nMessage length must be multiple of 8\n\n");

        return 0;

    }

    else

    {



            printf("\n\nMessage Length: %d\n\n",findFileSize());


            convertCharToBit(y);
            encrypt(y);

            //KeyGeneration();
    printf("Encrypted crime data: ");
    char t;
            fptr = fopen("cipher_d.txt", "r");
    t = fgetc(fptr);
    while (t != EOF)
    {
        printf ("%c", t);
        t = fgetc(fptr);
    }
    printf("\n\n");
    fclose(fptr);
    char o;
    printf("\n\nDO YOU WANT TO ACCESS THE FILE(Y/N): ");
    scanf("%c",&o);
    if(o=='y'||o=='Y')
    {
            printf("\n\nENTER THE PUBLIC KEY(n,e): ");
            scanf("%d",&n);
            scanf("%d",&e);
    int phi_n=phi(n);

            inp = fopen("key.txt", "r+");
            out = fopen("cipher.txt", "w+");
```

```
          // encryption starts
          while (1) {
                    char ch = getc(inp);
                    if (ch == -1)
                              break;
                    int value = toascii(ch);
                    Encryption_r(value, out);
          }
          fclose(inp);
          fclose(out);
          printf("\n\nEncrypted Symmetric Key: ");
          fptr = fopen("cipher.txt", "r");
   t = fgetc(fptr);
   while (t != EOF)
   {
      printf ("%c", t);
      t = fgetc(fptr);
   }
   printf("\n\n");
   fclose(fptr);
          // decryption starts
   char xq;
   scanf("%c",&xq);
   printf("\n\nDO YOU WANT TO DECRYPT THE FILE(Y/N): ");
   scanf("%c",&xq);
   if(xq=='y'||xq=='Y')
   {
   printf("\n\nENTER THE PRIVATE KEY(n,d): ");
```

```
          scanf("%d",&n);
          scanf("%d",&d);
printf("\n\nDECRYPTING ... \n");
          inp = fopen("cipher.txt", "r");
          out = fopen("decipher.txt", "w+");
          while (1) {
                    int cip;
                    if (fscanf(inp, "%d", &cip) == -1)
                              break;
                    Decryption_r(cip, out);
          }
          fclose(inp);
          fclose(out);
create16Keys_d();
y = findFileSize() / 8;
          decrypt(y);
printf("\n\n---DECRYPTION SUCCESSFUL---\n\n");
          return 0;
}
else
{
   printf("\n\n---DECRYPTION UNSUCCESSFUL -- \n\n");
   return 0;
}
}
else
{
   printf("\n\n---THANKS FOR USING THE PORTALS --\n\n");
```

```
        return 0;
    }
    }
}
```

Code for Keygeneration:

```
#include <iostream>
#include<cstdlib>
using namespace std;
int FastExponention(int bit, int n, int* y, int* a)
{
        if (bit == 1)
                *y = (*y * (*a)) % n;

        *a = (*a) * (*a) % n;
}
int FindT(int a, int m, int n)
{
        int r;
        int y = 1;
        while (m > 0) {
                r = m % 2;
                FastExponention(r, n, &y, &a);
                m = m / 2;
        }
        return y;
}
int gcd(int a, int b)
{
        int q, r1, r2, r;

        if (a > b) {
                r1 = a;
                r2 = b;
        }
        else {
                r1 = b;
                r2 = a;
        }

        while (r2 > 0) {
                q = r1 / r2;
                r = r1 - q * r2;
                r1 = r2;
                r2 = r;
        }

        return r1;
}
int PrimarityTest(int a, int i)
{
        int n = i - 1;
        int k = 0;
        int j, m, T;
        while (n % 2 == 0) {
                k++;
                n = n / 2;
        }
        m = n;
        T = FindT(a, m, i);

        if (T == 1 || T == i - 1)
                return 1;

        for (j = 0; j < k; j++) {
                T = FindT(T, 2, i);
                if (T == 1)
                        return 0;
                if (T == i - 1)
                        return 1;
        }
        return 0;
}
int inverse(int a, int b)
{
        int inv;
        int q, r, r1 = a, r2 = b, t, t1 = 0, t2 = 1;

        while (r2 > 0) {
                q = r1 / r2;
                r = r1 - q * r2;
                r1 = r2;
                r2 = r;

                t = t1 - q * t2;
```

```
                        t1 = t2;

                        t2 = t;

                }


                if (r1 == 1)

                        inv = t1;


                if (inv < 0)

                        inv = inv + a;


                return inv;

        }

        int main()

        {

          int p,q,e,n,d;

          int phi_n;

          cout<<"-----Key Generation ---- \n\nEnter two prime numbers: ";

          cin>>p;

          cin>>q;

                n = p * q;

                phi_n = (p - 1) * (q - 1);

                do

                        e = rand() % (phi_n - 2) + 2;

                while (gcd(e, phi_n) != 1);

          cout<<"\n\nPublic key(n,e): ("<<n<<", "<<e<<")";

                d = inverse(phi_n, e);

          cout<<"\n\nPrivate key(n,d): ("<<n<<", "<<d<<")\n\n";

        }
```

8. Limitations:

Usage of two algorithms increases run time. Even though performance makes hybrid cryptography efficient, run time disagrees with it. Our Future research is going to be in that area. We try to reduce the run time and increase the security strength for Hybrid Cryptography.

9. Results discussion and Conclusion

Thus using Hybrid cryptography, we can double the security level and enhance the protection of information. Because encryption and decryption using one algorithm doesn't provide security level provided by encryption and decryption using combination of symmetric and asymmetric algorithms. The proposed hybrid protocol tries to trap the intruder by splitting the plain text and then applies two different techniques. First, it takes the advantages of the combination of both Symmetric and Asymmetric cryptographic techniques using both DES and RSA algorithms. Second, combination of two algorithms is used since it is more robust and cannot be easily attacked. The attractiveness of the proposed protocol, compared to other existing security protocols, is that it appears to offer better security for a shorter encryption and decryption time, and smallest cipher text size. There by, reducing processing overhead and achieving lower memory consumption that is appropriate for all applications.

10. References

[1] Maitri, P. V., & Verma, A. (2016, March). Secure file storage in cloud computing using hybrid cryptography algorithm. In *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)* (pp. 1635-1638). IEEE.

[2] Nepal, S., Friedrich, C., Henry, L., & Chen, S. (2011, December). A secure storage service in the hybrid cloud. In *2011 Fourth IEEE International Conference on Utility and Cloud Computing* (pp. 334-335). IEEE.

[3] Swarna, C., & Eastaff, M. S. Secure File Storage in Cloud Computing Using Hybrid Cryptography Algorithm.

[4] Mata, F., Kimwele, M., & Okeyo, G. Enhanced Secure Data Storage in Cloud Computing Using Hybrid Cryptographic Techniques (AES and Blowfish).

[5] Subasree, S., & Sakthivel, N. K. (2010). Design of a new security protocol using hybrid cryptography algorithms. IJRRAS, 2(2), 95-103.