DAYANANDA SAGAR UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SCHOOL OF ENGINEERING

KUDLU GATE

BANGALORE - 560068

MINOR PROJECT REPORT ON

"Securing File On Cloud Using Hybrid Cryptography"

SUBMITTED TO THE 5TH SEMESTER MINOR PROJECT COURSE

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

Submitted by:

Chatterjee Keshav Kanchan(ENG19CS0071)

Under the supervision of

Dr Bondu Venkateswarlu

Associate Professor, DSU

DAYANANDA SAGAR UNIVERSITY

School of Engineering, Kudlu Gate, Bangalore-560068

CERTIFICATE

This is to certify that Chatterjee Keshav Kanchan USN ENG19CS0071 respectively have satisfactorily completed his/her Mini Project as prescribed by the University for the 5th semester B.Tech. programme in Computer Science & Engineering for Minor Project during the year 2021 at the School of Engineering, Dayananda Sagar University, Bangalore.

Date:______________ __________________________

Signature of faculty in-charge

Signature of Chairman
Department of Computer Science & Engineering

DECLARATION

We hereby declare that the work presented in this minor project entitled -"* Securing File On Cloud Using Hybrid Cryptography*", has been carried out by us and it has not been submitted for the award of any degree, diploma or the minor project of any other college or university.

Chatterjee Keshav Kanchan
(ENG19CS0071)

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of a task would be incomplete without the mention of the people who

made it possible and whose constant guidance and encouragement crown all the efforts with success.

We are especially thankful to our Dean & Chairman Dr. Srinivas A, Department of Computer Science & Engineering, for providing necessary departmental facilities, moral support and encouragement.

We are very much thankful to Dr Bondu Venkateswarlu , Department of Computer Science & Engineering, for providing help and suggestions in completion of this minor project successfully.

We have received a great deal of guidance and co-operation from our friends and we wish to thank all that have directly or indirectly helped us in the successful completion of this project work.

Chatterjee Keshav Kanchan
(ENG19CS0071)

ABSTRACT

Cloud is used in various fields like industry, military, college, etc. for various services and storage of huge amount of data. Data stored in this cloud can be accessed or retrieved on the users request without direct access to the server computer. But the major concern regarding storage

of data online that is on the cloud is the Security. This Security concern can be solved using various ways, the most commonly used techniques are cryptography and stegnography . Here we use Cryptography techniques.

TABLE OF CONTENTS

TABLE OF FIGURES

SL.NO	TITLE	PAGE NO.

CHAPTER 1

1. INTRODUCTION

- To store huge amount of data cloud computing is used now a day. We can retrieve any data from cloud, when the user request for it.

- Many issues are faced while storing the data, the solution for these issues we are using hybrid cryptography technique.

- In this system, we use two cryptography algorithm for providing block wise security to the data.

CLOUD COMPUTING

- Cloud computing is the delivery of on-demand computing service from application to storage and processing power typically over the internet.

- Data security refers to protective digital privacy measures that are applied to prevent unauthorized access to computers, databases and websites.
- The cloud provider can solve this problem by encrypting the files by using encryption algorithm. This project presents a file security model to provide an efficient solution

DATA SECURITY ISSUES

- Data Misuse
- Locality
- Access
- Integrity
- Confidentiality
- Breaches
- Storage

1.1 PROBLEM STATEMENT

In cloud the data is stored and handled by unknown servers, these servers can be sometimes accessed by unauthorized persons thus leading to violation of data integrity and security, and also user has no control over the data. In this project we are required to make a secure platform for file storage on cloud using hybrid cryptography algorithm.

1.2 PROJECT PURPOSE

In this project we aim to make a platform where user can

- Store any multimedia file into cloud and secure the file using hybrid cryptography algorithm.
- The multimedia files can be of any type text file, picture file, audio file, video file.
- It provides a secure platform for the user in which file is stored in encrypted format on cloud server and can be decrypted easily by the one who's having the required key access to the file.

1.3 PROJECT FEATURES

- It takes input (files) from users, asks for mode of cryptography and then encrypt or decrypt a file and stores it to the cloud.

CHAPTER 2

2. LITERATURE SURVEY

SL no.	TITLE	YEAR	METHODOLOGY	KEY FEATURES	DRAWBACKS
1.	Implementation of Secure File Storage on Cloud with owner defined Attributes for Encryption	2018	SHA And BLOWFISH	User Can give their own key for encryption	The problem is that if a user who knows the attributes can decrypt the file easily
2.	approach hybrid crypto environment	2019	RSA	The proposed security mechanisms will prevent confidential data from being misusedmaking the system more reliable	Works only if encrypt is higher then 2048 bits
3.	Cloud Oriented Distributed and Encrypted File Storage (CODE-FS)	2020	AES	The file is first encrypted and then uploaded to the server where it is divided into shards and these shards are distributed over multiple cloud nodes.	Single Algo Used for both Encrypt and Decrypt
4.	Secure File storage in Cloud Computing using Hybrid Cryptography Algorithm	2016	LSB STEGNOGRAPHY	It proposes the way of splitting files into different parts and then applying the hybrid cryptography for each part.	Easily decrypt from outside

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FUNCTIONAL REQUIREMENTS

- The user signs in if already registered, or signs up to register themselves by providing their details such as name, email id, phone number, password for account etc.
- The user should select the file that is to be uploaded by browsing from local storage.
- The user should select the encryption algorithm that they want to use. The proposed system provides the choice of rounds between using a combination of RSA or DES.
- The selected file gets uploaded after getting encrypted using the selected encryption algorithm combination or Rounds.
- The user also has the option of viewing the files that they have uploaded or have access to and decrypted them.
- On selecting a file to decrypted it, the user should give the decryption key
- Using this key, the user can decrypt encrypted original file.

3.2 Hardware Requirements & Software Requirements

- Operating Systems: Windows 10, Windows Server 2012 or greater.
- Processor: 1 GHz or greater.
- Hard Disk Space: 16GB available disk space.
- Memory: 2 GB (32-bit), 4 GB (64-bit).

Graphics: Support for DirectX 9 graphics with minimum 128MB RAM. ● Eclipse IDE for JAVA developers 2020-12.

CHAPTER 4

4. DESIGN METHODS

4.1 Modules

Main function, DES function class, RSA function class, Helper function

Encryption

In encryption, user is asked to enter details of the file to be encrypted. File name is given to the encrypted file
Mode of encryption is asked and executed.
The file will then be encrypted and will be stored at the desired location.

Decryption

In decryption, the user is asked for the details of the file to be decrypted.
Generated key from encryption is used.
Name and extension in which the file is to be converted is given by user.
Mode of decryption is asked and executed.
The file is then decrypted and stored in desired location.

4.2 DES

- The Data Encryption Standard is a symmetric-key algorithm for the encryption of digital data.
- DES is an algorithm that takes a fixed-length string of plaintext bits and transforms it through a series of complicated operations into another cipher text bit string of the same length.

DES is based on the two fundamental attributes of cryptography: ● Substitution (or Confusion) and Transposition (or Diffusion).

- DES consists of 16 steps,
- Initial Permutation (IP) function.
- The initial permutation performed on plain text.
- Left Plain Text (LPT) and Right Plain Text (RPT).
- LPT and RPT to go through 16 rounds of encryption process.
- LPT and RPT are re-joined and a Final Permutation (FP) is performed on the combined block
- The result of this process produces 64-bit cipher text.

4.3RSA

- RSA algorithm is a public key encryption technique and is considered as the most secure way of encryption.
- It was invented by Rivest, Shamir and Adleman in year 1978 and hence name RSA algorithm.

The RSA algorithm holds the following features −

- RSA algorithm is a popular exponentiation in a finite field over integers including prime numbers.
- The integers used by this method are sufficiently large making it difficult to solve.

There are two sets of keys in this algorithm: private key and public key.

- Step 1: Generate the RSA modulus
- Step 2: Derived Number (e)
- Step 3: Generate Public key
- Public key (e, n) ,where :1<=e<=(p-1) (q-1) ,
- e is co-prime with (p-1) (q-1)
- Step 4: Generate Private Key
- The mathematical relationship between the numbers is as follows: ed = 1 mod (p-1) (q-1)
- private key is (d, n)

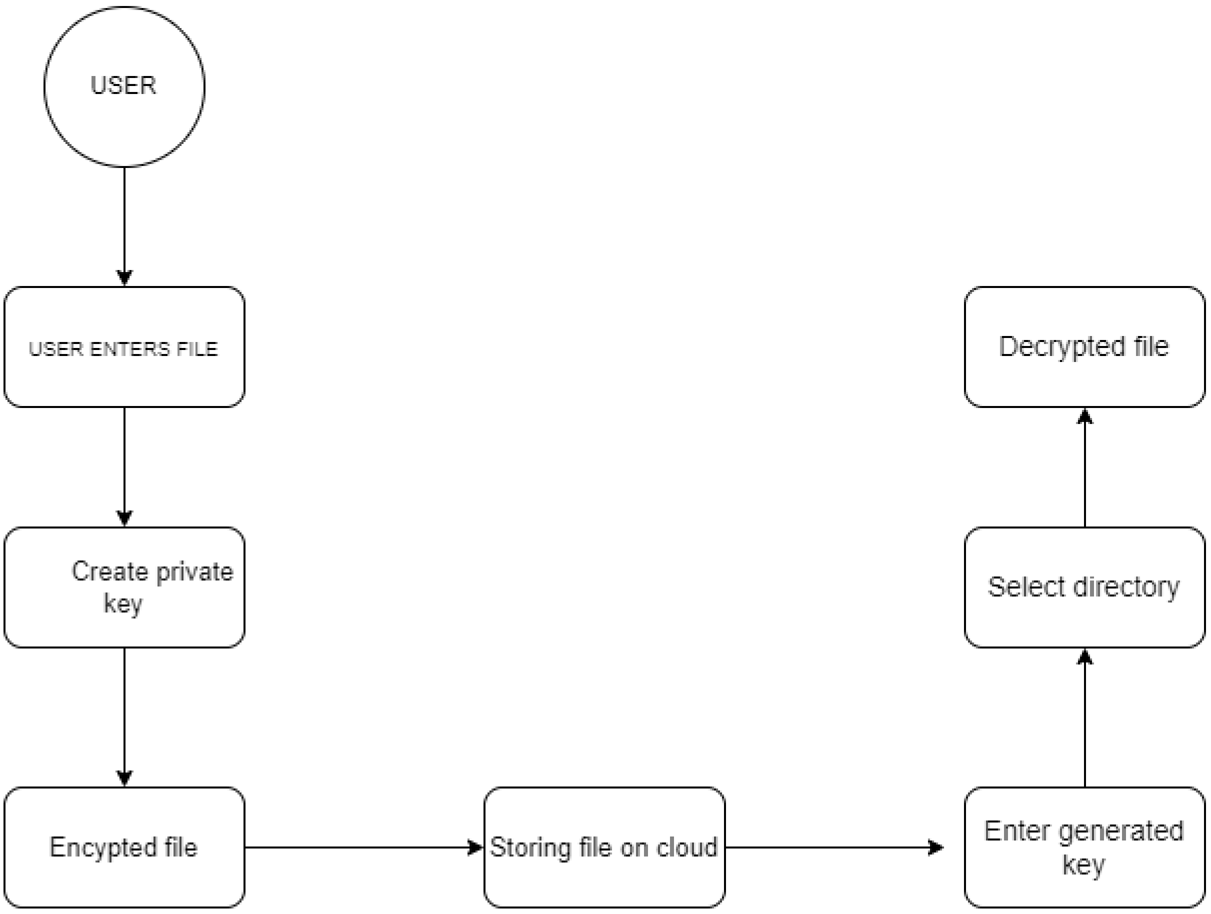
Encryption Formula

Consider a sender who sends the plain text message to someone whose public key is (e, n). To encrypt the plain text message in the given scenario, use the following syntax C = $\boldsymbol{P^e}$ mod n Decryption Formula

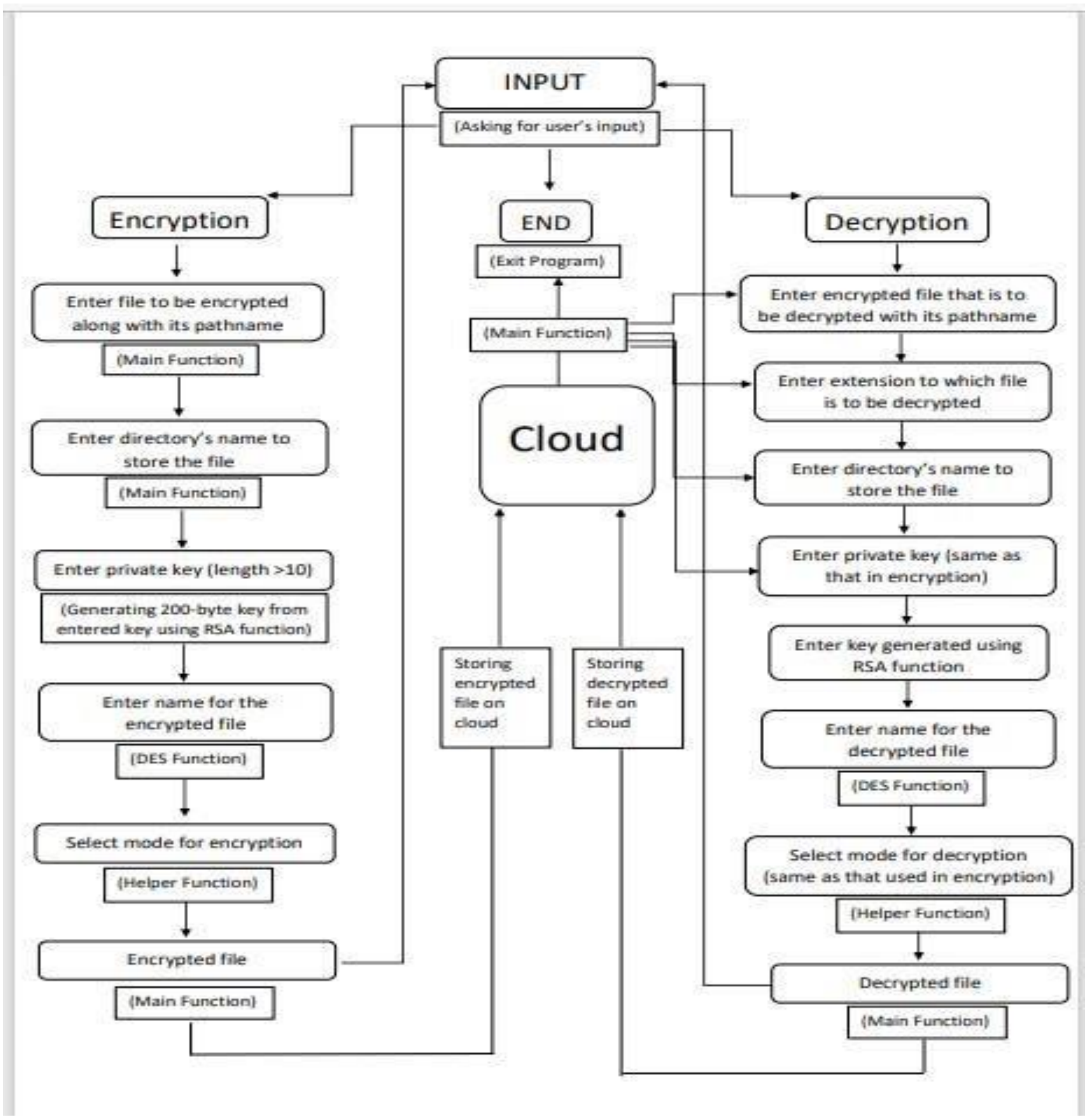
The decryption process is very straightforward and includes analytics for calculation in a systematic approach with private key (d, n). Considering receiver C has the private key d, the result modulus will be calculated as Plaintext =$\boldsymbol{C^d}$ mod n

.
.

4.2 ARCHITECTURE DIAGRAM



4.3 FLOWCHART DIAGRAM


INPUT
(Asking for user's input)
END
(Exit Program)
(Main Function)
Cloud
Encryption
Enter file to be encrypted along with its pathname
(Main Function)
Enter directory's name to store the file
(Main Function)
Enter private key (length >10)
(Generating 200-byte key from entered key using RSA function)
Enter name for the encrypted file
(DES Function)
Select mode for encryption
(Helper Function)
Encrypted file
(Main Function)
Storing encrypted file on cloud
Storing decrypted file on cloud
Decryption
Enter encrypted file that is to be decrypted with its pathname
Enter extension to which file is to be decrypted
Enter directory's name to store the file
Enter private key (same as that in encryption)
Enter key generated using RSA function
Enter name for the decrypted file
(DES Function)
Select mode for decryption (same as that used in encryption)
(Helper Function)
Decrypted file
(Main Function)

CHAPTER 5

5. PROJECT BREAKDOWN

1. Securing On cloud : We stores the files on cloud which will be secure

2. Hybrid Encryption: We use hybrid methods to encrypt and decrypt the files including test,jpg,png,gif etc.
3. More Secure: More secure way to send files to send files through cloud without intruders.

CHAPTER 6 6. IMPLEMENTATION

1. MAIN Function:

```
import java.math.BigInteger;

import java.util.Scanner;

public class Main {
	static boolean flag=true;
	static String filename="",dirname="",DESKEY="";
	static int chk=0,ch=0;
	public static void main(String[] args) {
		//object of class EncDec.
		//used to encrypt/decrypt files.
		DESFunctionClass obj=new DESFunctionClass();
		Scanner chscanner =new Scanner(System.in);

		while(flag){

	System.out.println("\n===============================================================\n
1.ENCRYPT a file \t 2.DECRYPT a file \t 3.Exit
Program\n==============================================================");
				System.out.println("Enter Your Choice:\t");
	if(chscanner.hasNextInt())
					ch=chscanner.nextInt();
	switch(ch)
				{
				case 1:
					//Checks infinitely for valid file type.
					do {
				System.out.println("Enter Name of the File to be Encrypted(include path if
outside):");
						filename=chscanner.next();
						filename=filename.replaceAll("\\\\", "/");    //for windows
dir scheme
						chk= HelperFunction.check(filename);
string type to BigInteger type
						BigInteger RSAKEY=RSAFunctionClass.EncDec(k,
RSAFunctionClass.e,RSAFunctionClass.n); //RSA-Encryption of the DES-key
						String keyloc=RSAFunctionClass.WriteEncKey(RSAKEY, dirname,
filename); //write encrypted key to file for further use

						//Encrypting the File using DESKEY and DES algorithm.
						obj.encrypt(filename,dirname,DESKEY);
```

```
                                    }while(chk!=1);

                              //Checks for valid Directory type to store encrypted file.
                                    do {
                                    System.out.println("Enter Name of Directory to store Encrypted
file:");
                                    dirname=chscanner.next();
                                    dirname=dirname.replaceAll("\\\\", "/");
                                    chk= HelperFunction.check(dirname);
                                    }while(chk!=2);

                                    //Takes Valid private input key of size>10
                                    do{
                                    System.out.println("Enter Your Private Key (length>10):)");
                                    DESKEY=chscanner.nextLine();
                                    if(DESKEY.length()<10)
                                          System.out.println("\t\t--Private Key Size should be >
10!--");
                                    }while(DESKEY.length()<10);


                                    DESKEY= HelperFunction.KeyGen(DESKEY);//generating random key
from input key.

                                    BigInteger k=new BigInteger(DESKEY); //convert key from

                                    System.out.println("\nFile ENCRYPTED Successfully, Stored
at"+"'"+dirname+"'");
                                    System.out.println("ATTENTION! NOW Your Encrypted Private Key
is:"+RSAKEY+"\n\tIt is Saved for You at '"+keyloc+"'");
                              break;

                        case 2:
                                    //Checks infinitely for valid file type.
                              do{
                                          System.out.println("Enter Name of the Encrypted File
that is to be Decrypted(include path if outside):");
                                          filename=chscanner.next();
                        filename=filename.replaceAll("\\\\", "/");
                                          chk= HelperFunction.check(filename);
                                          }while(chk!=1);

                                    //Get Original Extension for Decryption
                                    System.out.println("Enter EXTENSION to which file is to be
Decrypted(e.g txt,pdf,jpg,mp3,mp4,etc):");
                                    String extname = chscanner.next();
                                    extname=extname.substring(extname.lastIndexOf(".") + 1);
      //if user provided a '.' with extension
                                    //Checks for valid Directory type to store decrypted file.
                                    do{
                                          System.out.println("Enter Name of Directory where
Decrypted file will be Stored:");
                                          dirname=chscanner.next();
```

```
                                        dirname=dirname.replaceAll("\\\\", "/");
                                        chk= HelperFunction.check(dirname);
                                }while(chk!=2);

     String regex = "[0-9]+";//Regular Expression for string to make sure key contains only
numbers

                                do{
                                        System.out.println("Enter Your Encrypted Private Key of
the file:");
                                        DESKEY=chscanner.next();
      /*if(DESKEY.length()<500||!(DESKEY.matches(regex)))            System.out.println("\t\t--
Encrypted-Key Size must be > 500 and Must only contain Numeric Values!");*/
                                }while((DESKEY.length()<500)||!(DESKEY.matches(regex)));

                                BigInteger c=new BigInteger(DESKEY);//convert to BI
                                BigInteger Deckey=RSAFunctionClass.EncDec(c,
RSAFunctionClass.d,RSAFunctionClass.n); //UNHANDLED>> make regex seq for key in EncDec
fxn
                                DESKEY=Deckey.toString();

                                obj.decrypt(filename,extname,dirname,DESKEY);


                                 System.out.println("\nFile DECRYPTED Successfully,' Stored at
"+"'"+dirname+"'");
                        break;

                        case 3:
                                flag=false;
                                System.out.println("Good Bye!");
            chscanner.close();
                                break;

                        default:
                                System.out.println("No Such Option... Try Again!");
                }
            }
      }
}
```

2. HELPER FUNCTION:

```
import java.io.File;
 import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException; import
java.io.RandomAccessFile; import
java.nio.channels.FileChannel;
import java.util.Random; import
java.util.Scanner; public class
HelperFunction {
      Scanner inscan=new Scanner(System.in);
      //    checks for file/Directory
      //    returns 1 for file
	//	returns 2 for Directory
	//	returns 0 if not file or directory
	public static int check(String filename){
		int flag=0;
		File filechk = new File(filename);
		if(filechk.isFile())
			flag=1;
		else if(filechk.isDirectory())
			flag=2;
		if(flag!=1&&flag!=2)
   System.out.println("'"+filename+"'"+" is not a Valid FILE/DIRECTORY!");   return flag;
	}

	//Generates Random Key from the Input Key.
	//Uses Random and Fisher Yates Shuffle.
	//Returns String.
	public static String KeyGen(String key){	//make a 200 byte key from user key
		int len=key.length(),i=0,n=1;

		do{
			int x= key.charAt(i)+n;			//more randomness by 'n'
			key+=x;
		n++;i++;

			if(i==len-1)
				i=0;

		}while(key.length()<len+100);

		key=key.substring(len);

		int [] kArray=new int[key.length()];
	for(int m =0;m<=key.length()-1;m++)
		{
			kArray[m]=key.charAt(m);
			double temp= HelperFunction.percentage(m, key.length()-1);
			if (temp%20==0.00){
				System.out.println("Generating 200 byte key from Entered key: 
"+temp+"%");
			}
		}

		int kArrLen=kArray.length;		for (int j = kArrLen - 1; j >= 1; 
j--)   //FISHER-YATES Random Shuffle
		{
			Random rand = new Random();
			// generate a random number k such that 0 <= k <= j
			int k = rand.nextInt(j + 1);

			// swap current element with randomly generated index
			int temp = kArray[j];			kArray[j] = 
kArray[k];
			kArray[k] = temp;
		}
		key="";
		for(int m =0;m<=kArrLen-1;m++)
		{
			key+=kArray[m];
		}
		System.out.println(key.length()+" Byte Key Generated Successfully!");
		return key;
	}

	//Gives Percentage of task
	public static double percentage(long no, long total){			//percent
function
		double per = (no*100.0)/total;
		per = per * 100;
	per = Math.round(per);
	per = per/100;
	return per;
	}

	//For copying files using File channel=>faster method
	public static void copyFile(String source, String dest) throws IOException {
		File src=new File(source);
		File dst=new File(dest);
		FileChannel sourceCh = null;
	FileChannel destCh = null;
		try {
			sourceCh = new FileInputStream(src).getChannel();
		destCh = new FileOutputStream(dst).getChannel();
	destCh.transferFrom(sourceCh, 0, sourceCh.size());
		}
		finally{
			sourceCh.close();destCh.close();
		}
	}

	public static void rounds(RandomAccessFile in,RandomAccessFile out,String key,int
shiftby,String mode){		//round Encryption/decryption
	HelperFunction obj=new HelperFunction();
		int round=0,ch=0;
	String roundname="";

	System.out.println("=====================================================================
=====");
		System.out.println("Enter Mode:\n1:->02 Round Enc/Dec\t\t\n2:->04 Round
Enc/Dec\t\t\n3:->08 Round Enc/Dec\t\t\n4.:->12 Round Enc/Dec\t\n5.:->16 Round
Enc/Dec\t\t\n\t\tUse Same Mode for Decryption with which the File was Encrypted!");
	 System.out.println("=====================================================================
=====");
		if(obj.inscan.hasNextInt())
			ch=obj.inscan.nextInt();

		if(ch==1){
			round=2;
			roundname="2 Round Enc/Dec";
		}
```

```
          else if(ch==2){
          round=4;
                 roundname="4 Round Enc/Dec";
          }
          else if(ch==3){
                 round=8;
                 roundname="8 Round Enc/Dec";
          }
          else if(ch==4)
          {round=12;
                 roundname="12
          Round Enc/Dec";
          }
          else if(ch==5){
          round=16;
                 roundname="16 Round Enc/Dec";
          }
          else
                 System.out.println("Invalid Option!");

          if(key.length()%2!=0)
                 round--;

          for(int i=1;i<=round;i++)        //Apply Alternate rounds
          {

                 if(i%2!=0)
                 {
                        System.out.println("\t\t\tROUND--"+i);
                 key= HelperFunction.shiftKey(key,shiftby);
          HelperFunction.xor(in, out, key,mode,"Round-"+i);
                 }
                 else
                 {
                        System.out.println("\t\t\tROUND--"+i);
                 key= HelperFunction.shiftKey(key, shiftby);
          HelperFunction.xor(out, in, key, mode,"Round-"+i);
                 }
          }
          System.out.println("\t"+roundname+ " Successfully Completed!");
   }

   //left shift by factor "shiftby'
   public static String shiftKey(String key,int shiftby){
          int keylen=key.length();
          String s1=key.substring(shiftby,keylen);
   String s2=key.substring(0,shiftby);
          key=s1+s2;
   return key;
   }

   //XOR function.
 public static void xor(RandomAccessFile in,RandomAccessFile temp,String key,String
mode,String round){
          int len_var=0;
          try
          {
                  long incount=in.length();
                  int p=0;
                  double percent;
  in.seek(0);
                  temp.seek(0);
                  for(int j=0; j<=incount-1;j++)
                  {
                          int intchr=in.read();
                          //write at beginning
                          temp.write(intchr ^ key.charAt(len_var));//XORing
          len_var++;

                          if(len_var>key.length()-1)
                                  len_var=0;

                          percent=percentage(p,incount);
                          p++;
                          if (percent%20==100.00 && p%10==0) {
                                  System.out.println("[" + round + "]" + " " + mode + "
Characters to File:" + percent + "%");
                          }
                  }
          }
          catch(Exception e){

          }
  }

  //Simple Reversing
  public static void shuffle(RandomAccessFile in,RandomAccessFile out){
          try {
                  int p=0;
                  long count=in.length();

                  for(long i=count-1;i>=0;i--)
                          {
                              //Writing on file>>
                                  in.seek(i);         //set cursor of in file at last >> i=count-
1
                                  out.seek(p);        //set cursor of output file to start >> p=0
                                  int ch =in.read();  //read() from in
                                  out.write(ch);              //write it at start of output file

                                  p=p+1;                      //increment p

                          }
          } catch (IOException e) {
                  System.out.println(e);
          }
  }
  public static void delencf(String filename, int ch){
          if(ch==1)
          {
```

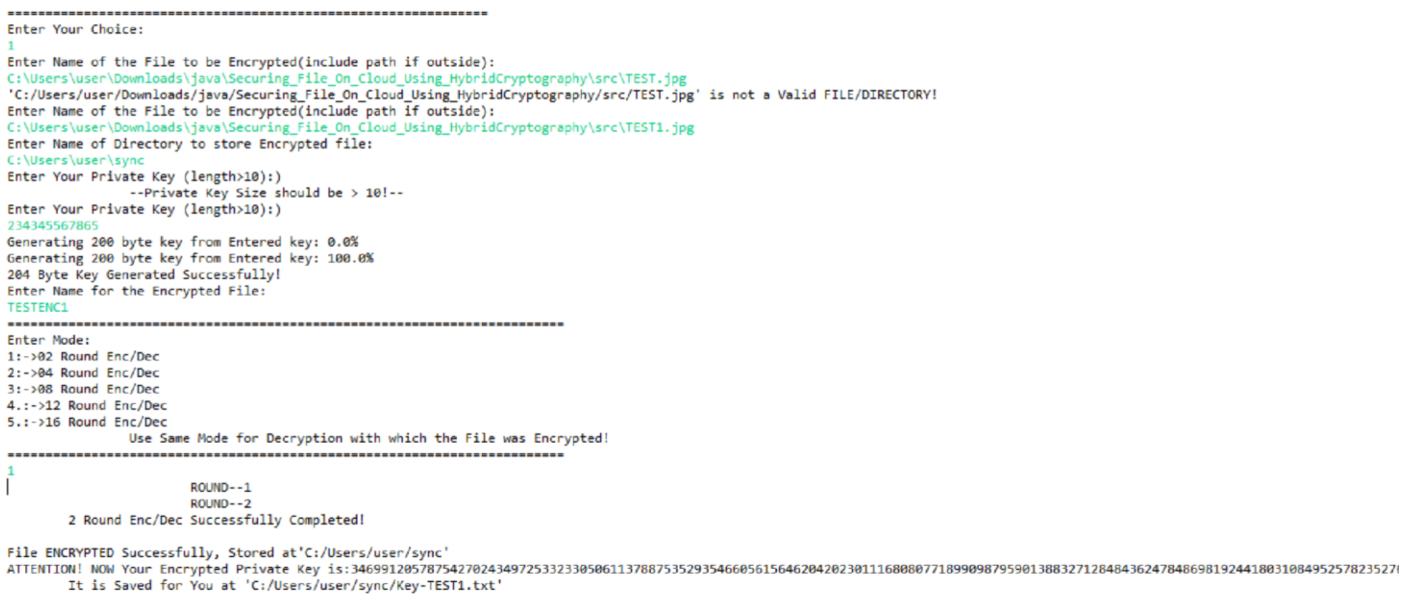
```
                        File f = new File(filename);
                if(f.delete())
                                System.out.println(filename+" deleted Successfully!");
                        else
                                System.out.println("\nCouldn't Locate "+filename+" to delete!");
                }
        }

        }
```

CHAPTER 7

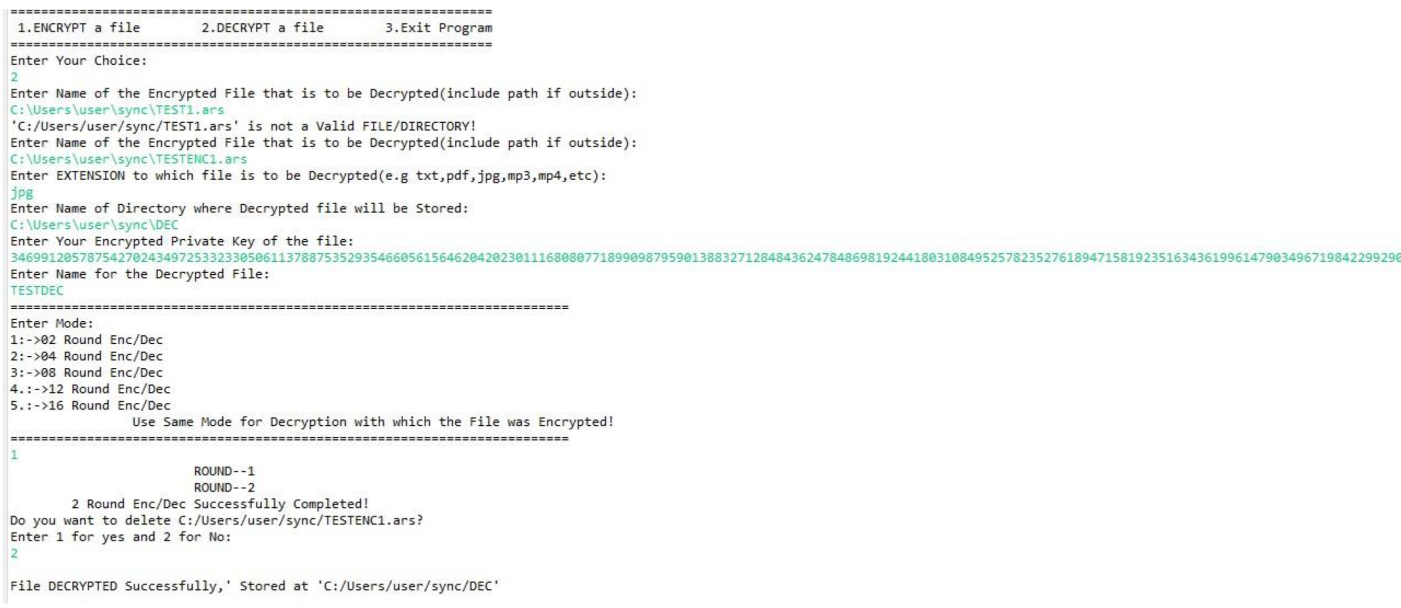
7. TEST CASE

File Type	File Size	Encryption Time	Decryption Time
TEXT	293 kb	6545 milliseconds	7554 milliseconds
IMAGE	543 kb	11439 milliseconds	13045 milliseconds
AUDIO	2065 kb	43086 milliseconds	47830 milliseconds
VIDEO	3042 kb	63294 milliseconds	70966 milliseconds



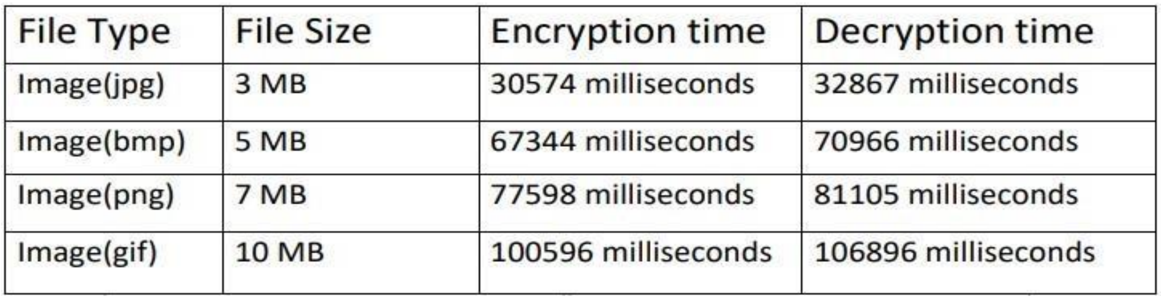
```
================================================================
Enter Your Choice:
1
Enter Name of the File to be Encrypted(include path if outside):
C:\Users\user\Downloads\java\Securing_File_On_Cloud_Using_HybridCryptography\src\TEST.jpg
'C:/Users/user/Downloads/java/Securing_File_On_Cloud_Using_HybridCryptography/src/TEST.jpg' is not a Valid FILE/DIRECTORY!
Enter Name of the File to be Encrypted(include path if outside):
C:\Users\user\Downloads\java\Securing_File_On_Cloud_Using_HybridCryptography\src\TEST1.jpg
Enter Name of Directory to store Encrypted file:
C:\Users\user\sync
Enter Your Private Key (length>10):)
                --Private Key Size should be > 10!--
Enter Your Private Key (length>10):)
234345567865
Generating 200 byte key from Entered key: 0.0%
Generating 200 byte key from Entered key: 100.0%
204 Byte Key Generated Successfully!
Enter Name for the Encrypted File:
TESTENC1
==========================================================================
Enter Mode:
1:->02 Round Enc/Dec
2:->04 Round Enc/Dec
3:->08 Round Enc/Dec
4.:->12 Round Enc/Dec
5.:->16 Round Enc/Dec
                Use Same Mode for Decryption with which the File was Encrypted!
==========================================================================
1
|                       ROUND--1
                        ROUND--2
        2 Round Enc/Dec Successfully Completed!

File ENCRYPTED Successfully, Stored at'C:/Users/user/sync'
ATTENTION! NOW Your Encrypted Private Key is:3469912057875427024349725332330506113788753529354660561564620420230111680807718990987959013883271284843624784869819244180310849525782352
        It is Saved for You at 'C:/Users/user/sync/Key-TEST1.txt'
```

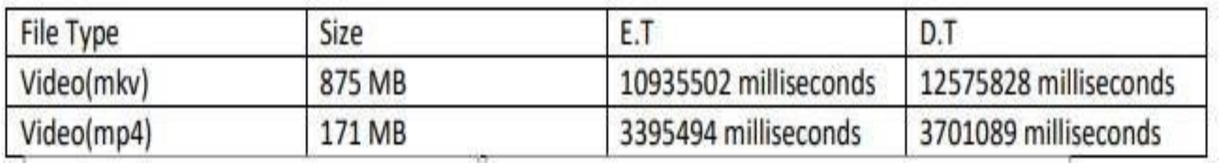


```
================================================================
 1.ENCRYPT a file        2.DECRYPT a file        3.Exit Program
================================================================
Enter Your Choice:
2
Enter Name of the Encrypted File that is to be Decrypted(include path if outside):
C:\Users\user\sync\TEST1.ars
'C:/Users/user/sync/TEST1.ars' is not a Valid FILE/DIRECTORY!
Enter Name of the Encrypted File that is to be Decrypted(include path if outside):
C:\Users\user\sync\TESTENC1.ars
Enter EXTENSION to which file is to be Decrypted(e.g txt,pdf,jpg,mp3,mp4,etc):
jpg
Enter Name of Directory where Decrypted file will be Stored:
C:\Users\user\sync\DEC
Enter Your Encrypted Private Key of the file:
3469912057875427024349725332330506113788753529354660561564620420230111680807718990987959013883271284843624784869819244180310849525782352761894715819235163436199614790349671984229929
Enter Name for the Decrypted File:
TESTDEC
==========================================================================
Enter Mode:
1:->02 Round Enc/Dec
2:->04 Round Enc/Dec
3:->08 Round Enc/Dec
4.:->12 Round Enc/Dec
5.:->16 Round Enc/Dec
                Use Same Mode for Decryption with which the File was Encrypted!
==========================================================================
1
                        ROUND--1
                        ROUND--2
        2 Round Enc/Dec Successfully Completed!
Do you want to delete C:/Users/user/sync/TESTENC1.ars?
Enter 1 for yes and 2 for No:
2

File DECRYPTED Successfully,' Stored at 'C:/Users/user/sync/DEC'
```



File Type	File Size	Encryption time	Decryption time
Image(jpg)	3 MB	30574 milliseconds	32867 milliseconds
Image(bmp)	5 MB	67344 milliseconds	70966 milliseconds
Image(png)	7 MB	77598 milliseconds	81105 milliseconds
Image(gif)	10 MB	100596 milliseconds	106896 milliseconds



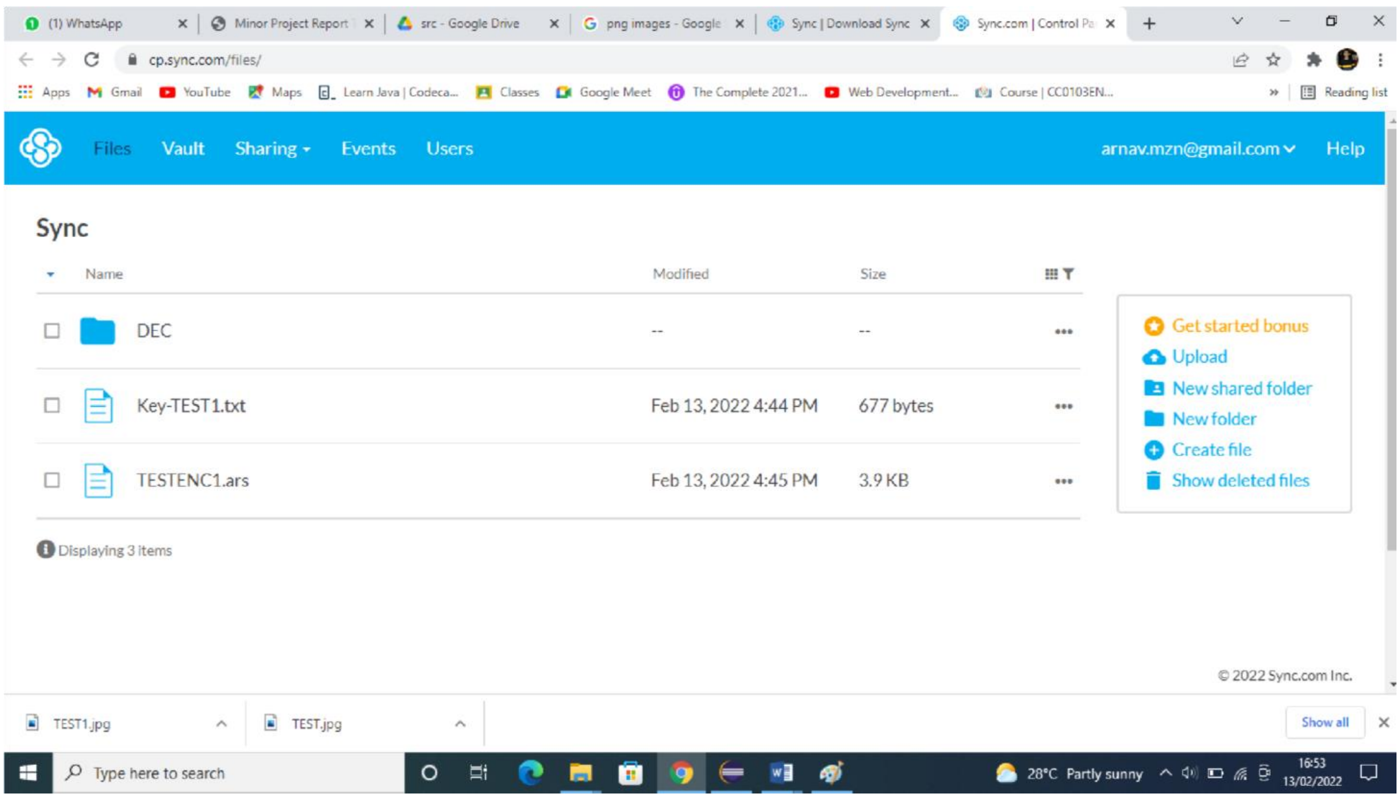
File Type	Size	E.T	D.T
Video(mkv)	875 MB	10935502 milliseconds	12575828 milliseconds
Video(mp4)	171 MB	3395494 milliseconds	3701089 milliseconds

CHAPTER 8

8. RESULTS/OUTPUT SCREENSHOTS

ENCRYPTION

DECRYPTION



Sync

Name	Modified	Size
DEC	--	--
Key-TEST1.txt	Feb 13, 2022 4:44 PM	677 bytes
TESTENC1.ars	Feb 13, 2022 4:45 PM	3.9 KB

Displaying 3 items

CLOUD :

CHAPTER 10

10. CONCLUSION & FUTURE WORK

To conclude, it can be said that the current implementation has been tested on text files, image file, audio file and video file the results obtained so far are very motivating because of the speed gains over the existing techniques. Also it provides a choice of option for the user to select how much secure they want their file to be. The implementation could be used in applications requiring encryption of multimedia data at a rapid pace. The same implementation could be used in a network to encrypt the files travelling through it for example, attachments travelling through an email could be secured using the hybrid encryption along with the existing email security provided by the mail server or using it alone.

Future Enhancement:

- Adding more cryptographic algorithms
- Can make it as combination of 3 or more algorithms
- Reduce time consumption in encryption and decryption
- Can verify the user

REFERENCES

- Ajit Singh Aarti Nandal and Swati Malik "Implementation of Ceaser Cipher with Rail Fence for enhancing data Security" International Journal of Advanced research in Computer Science and Software Engineering vol. 2 no. 12 pp. 78-82 December 2019

- Singh Inder and M. Prateek "Data Encryption and Decryption Algorithms using Key Rotations N. Sharma A. Hasan “A New Method Towards Encryption Schemes (Name-Based-Encryption Algorithm)" IEEE International Conference on Reliability Optimization and Information Technology pp. 310-313 Feb 2014.

- N. Sharma and A. Hasan "Name-Based-Encryption Algorithm" IEEE International Conference on Reliability Optimization and Information Technology pp. 310-313 Feb 2014

- Sonal Modh, DR. M. K. Rawat, “secure file using Cryptography”, IJSETR, Vol.05, Issue.06, March-2016, Pages:1140-1146

- Md.Zaheer Abbas, Dr JVR Murthy, “Secure File Storage Using Hybrid Cryptography” International Journal of Engineering Research and Applications (IJERA) 2012; volume 2(3), p.1347-1352

- Ashita Sharma, Navroz Kaur, “Implementation of DES (Data Encryption Standard) Algorithm”, International Journal for Multi-Disciplinary Engineering and Business Management, Volume-2, Issue-3, JulySeptember, 2014

- Mahavir Jain and Arpit Agrawal "Implementation of Hybrid Cryptography Algorithm" International journal of Core Engineering &amp; Management vol. 1 no. 3 pp. 1-8 June 2018.