VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SECURE FILE STORAGE USING HYBRID CRYPTOGRAPHY ALGORITHM

A PROJECT REPORT

Submitted by

Rashi Maheshwari (19BDS0006)

Shubham Kaushik (19BCE2154)

Course Code: CSE3501

Course Title: Information Security Management

Under the guidance of

Dr. Kakelli Anil Kumar

Associate Professor

SCOPE, VIT, Vellore.

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

April -2022

INDEX

CHAPTER NO.	TITLE	PAGE NO.
1.	**Abstract**	**3**
2.	**Introduction**	**3**
3.	**Literature review**	**4**
4.	**Overview of the work**	**7**
	4.1. Working of the model	7
	4.2. Design Description	8
	4.3. Algorithm	8
	4.3.2. Blowfish Algorithm	8
	4.3.2. SRNN Algorithm	11
5.	**Implementation**	**12**
	5.1. Code	12
	5.2. Screenshots	14
6.	**Results**	**15**
7.	**Conclusion and Future Scope**	**15**
8.	**References**	**15**

1. ABSTRACT

Cloud Computing is increasingly becoming popular as many enterprise applications and data are moving into cloud platforms. It allows us to share data among many clients, servers and people. It is one of the most exciting technologies due to its ability to reduce costs associated with computing while increasing flexibility and scalability for computer processes. During the past few years, cloud computing has grown from being a promising business idea to one of the fastest growing parts of the IT industry. However, a major barrier of cloud computing is that the security of information present in the cloud is not guaranteed and so, it is simple for an intruder to access and demolish the information. In this project, we will discuss a security model to give a productive answer for the fundamental issue of security in cloud condition. In this model, half breed encryption is utilized where records are scrambled by blowfish combined with document part and SRNN (modified RSA) is utilized for the secured correspondence amongst clients and the servers. So, there is a requirement of some plain key which will help us to do cross breed encryption and protect the data.

2. INTRODUCTION

Cloud computing originated from earlier large-scale distributed computing technology. NIST defines Cloud computing as a model for enabling convenient, on demand network access to a shared pool of configurable computing resources (e.g.,networks, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction".

In Cloud computing, both files and software are not fully contained on the user's computer. File security concerns arise because both users application and program are residing in provider premises. The cloud provider can solve this problem by encrypting the files by using an encryption algorithm. Here we discussed a security model to give aproductive answer for the fundamental issue of security in cloud condition. In this model, half breed encryption is utilized where records are scrambled by blowfish combined with document part and SRNN (modified RSA) is utilized for the secured correspondence amongst clients and the servers.

3. LITERATURE REVIEW

3.1. A File Encryption System Based on Attribute Based EncryptionA File Encryption System Based on Attribute Based Encryption (Tong, Jie Long, Yihong, Liu, Quan)

Attribute Based Encryption (ABE) is a kind of public-key cryptosystem that is commonly used to protect file data security and achieve the fine-grained sharing of files. However, the existing attribute-based encryption algorithms are very complex and involve a large amount of additional cryptographic data, which is not convenient to implement. To solve this problem, an attribute-based data encryption scheme is proposed in this paper to achieve secure storage and access control of files. This system is convenient to implement and reliable in operation. In this file encryption system, the attribute private key is divided into a server-side secret share and a user-side secret share. When a user is authorized to access a file, the client and the server decrypt the encrypted file by collaborative computation using the user-side secret shares and server-side secret shares of the user's attribute private keys.

3. 2. Designing a New XTS-AES Parallel Optimization Implementation Technique for Fast File Encryption (An, S.Seo, S.C.)

XTS-AES is a disk encryption mode of operation that uses the block cipher AES. Several studies have been conducted to improve the encryption speed using XTS-AES according to the increasing disk size. Among them, there are researches on parallel encryption of XTS-AES using GPU. Although these studies focus on parallel encryption of AES, optimization for the entire XTS mode has not been performed. The reason is that the $\alpha ^{j}$ computation process included in XTS mode is not suitable for parallel operation. Therefore, in this paper, we proposed several techniques for high-speed encryption in GPU by modifying XTS-AES into a form that is advantageous for parallel operation. The core idea is to pre-calculate the $\alpha ^{j}$ calculation on the CPU into a form that is easy to operate on the GPU. To achieve this goal, we analyzed the $\alpha ^{j}$ calculation process and present the parts that can be optimized.

3.3. File Encryption System Based on the Hybrid LPC Coefficient and AES Algorithm

At present, academic research on information security is getting deeper and wider. In this paper, the LPC coefficient and the speech waveform characteristics are mixed to form the

hybrid LPC coefficient. The hybrid LPC is then used as the secret key of the AES algorithm in the file encryption system. The system is designed based on VS2012, all the files can be encrypted and decrypted by the system using the voice hybrid LPC key, and the system performance is relatively stable.

3.4. Research on One Encryption Technology of Video File

Based on content-associated key technology, the paper designed and implemented an encryption algorithm for MP4 files. The algorithm first parses the MP4 file format to determine the position of the video data frame in the file, and uses the PWLCM chaotic mapping to determine the key extraction position in frames (or frame sets). In order to achieve better encryption effect, the Key frame / non-key frame hybrid extraction method is used to extract the corresponding data of each frame in the file, and to synthesize the key of the entire file. According to the playback effect of encrypted MP4 video, the algorithm is effective.

3.5. Encryption of Text File Using a User Controlled Automatically-Generated Key

There are many different approaches that can be taken in order to generate automatically secret keys, however, for high security applications, end users may require control over the key generation process rather than leaving the process to third parties. In this paper, both the decryption and the encryption keys are obtained using the evolutionary computing tool Eureqa to model pseudo-random input data. The secret keys found using this technique when applied to encryption and decryption data files are validated using a range of tests, encryption and encryption time, throughput, entropy analyses and key sensitivity. The results obtained from the experiments show that the proposed encryption and decryption approaches are both reliable and secure when applied to text files and have the potential to be applied in high-security text communication applications.

3.6. File Encryption and Decryption Using DNA Technology

Cryptography is the method of transforming the original texted message into an unknown form and in reverse also. It is the process of hiding and forwarding the data in an appropriate form so that only authorized persons can know and can process it. Cryptographic process secures the data from hijacking or transmutation, it is mainly used for users data security.

This paper justifies the encryption and decryption using DNA(Deoxyribo Nucleic Acid) sequence. This process includes several intermediate steps, the perception of binary-coded form and generating of arbitrary keys is used to encrypt the message. A common key should be established between the sender and receiver for encryption and decryption process. The common key provides more security to the sequence. In this paper, both the process of binary-coded form and generating of arbitrary keys are used to encrypt the message. It is widely used in an institution and by every individual to hide their data from the muggers and hijackers and provides the data securely, and confidentially over the transmission of information.

3.7. Symmetric Jumbling-Salting Encryption Algorithm for Files

This paper presents an improvised approach for file encryption in the server's database using jumbling-salting algorithm. One of the most import concepts used here is randomization using a randomly generated character set. Dictionary attacks and brute force attacks while sending a file from the client to the server are common ways of security breaches. Here, the Jumbling-Salting algorithm has been used to augment the security aspect by dividing the file at hand into chunks of a set size (explicitly by the client) and then encrypting the chunks individually. The length of the cipher text is considerably increased in this process. The algorithm defined in the paper defines a character set from where it selects a random character and integrates it with the plaintext using a modulus function.

3.8. A Practical One-Time File Encryption Protocol for IoT Devices

Security and privacy issues of the Internet of Things (IoT in short, hereafter) attracts the hot topic of researches through these years. As the relationship between user and server become more complicated than before, the existing security solutions might not provide exhaustive securities in IoT environment and novel solutions become new research challenges, e.g., the solutions based on symmetric cryptosystems are unsuited to handle with the occasion that decryption is only allowed in specific time range. In this paper, a new scalable one-time file encryption scheme combines reliable cryptographic techniques, which is named OTFEP, is proposed to satisfy specialized security requirements. One of OTFEP's key features is that it offers a mechanism to protect files in the database from arbitrary visiting from system manager or third-party auditors.

3.9. A File Encryption Algorithm Based on Dynamic Block Out of order Matrix Mapping

In order to solve the security of content transfer during file transfer and fast encryption of large files. This paper proposes a method to quickly encrypt large files and protect the contents of file information, also prevent files from being intercepted and decrypted before use. The algorithm of file encryption based on dynamic block out of order matrix mapping. First, dynamically block the files according to a certain ratio, extract the raw data and key information of each block, then store the out of order data into a two-dimensional table of matrix after calculation and processing. Second, use Base64 encoding on this matrix mapping table. Then the matrix mapping table is to be a unique decryption key of this file. After that, the data in the spare block of the file is filled with the MD5 data, the MD5 data is generated by UUID and block with the same size as the number of dynamic block.

3.10. Enhanced SPK Encryption Algorithm for File Encryption Using Java

The encryption process is the method such that a normal plain text is coded in various ways using different algorithms so that the data gets unreadable and can only be read by decoding the data using decryption methods. Encryption is necessary for obtaining data integrity and safety. It helps with secure data flow in any network. So during this encryption obviously we will be experiencing some overhead. This paper focuses on the different encryption methods and algorithms and how each algorithm behaves with different kinds of files such as text file, MP3, PNG etc. The algorithms used were AES, DES, RSA, and another encryption algorithm which was formulated by ourselves. The execution time and the difference in the file size was measured. Graphs were plotted accordingly and the measured quantities were compared.

4. OVERVIEW OF THE WORK

4.1. Working of the model

In order to ensure file security on cloud, the above hybrid cryptosystem is deployed on cloud. We assume a cloud server as trusted but in order to prevent tampering/misuse of data by intruder or data leakage or other security concerns, the data is stored at server in the encrypted form. First the client registers himself/herself in order to upload and view his/her

files to/from the cloud server. After successful registration the client is registered with the IP address of the corresponding VM of the cluster. node. The encrypted files, encrypted blowfish keys, and public SRNN keys are stored at his registered VM in the cloud server. When the client wants to upload his files, he first authenticates himself and uploads his files. The uploaded files are encrypted using the hybrid cryptosystem. When the client wants to download the files, he will provide the SR private key. The SRNN private key will be decrypted to get the blowfish key. By using this blowfish key the encrypted file will be decrypted.

4.2. Design Description

The hybrid cryptosystem used to maintain security of the files has two phases:

1) Encryption Phase: In the encryption phase, the file is encrypted using the Blowfish key provided by the user. The key will be encrypted using the SRNN public key.

2) Decryption Phase: At the decoding phase, the client will give the SRNN private key. Blowfish key is decoded at the server end utilizing the SRNN private key.

4.3. Algorithm

The hybrid cryptosystem uses a combination of:

1) Blowfish Algorithm

2) SRNN Algorithm

4.3.1. Blowfish Algorithm

Blowfish is a keyed, symmetric cryptographic block cipher designed by BruceSchneier in 1993 and placed in the public domain which uses a Fiesta network.There are 16 rounds of iterative encryption and decryption functional design. The block size used is of 64 bits and key size can vary from any length to 448. Blowfish cipher uses 18 sub arrays each of 32-bit commonly known as P - boxes and four Substitution boxes each of 32-bit, each having 256 entries.

It consists of two phases: Key Expansion phase and Data Encryption phase. In the Key expansion phase, the key is converted into several sub-keys. The algorithm keeps two subkey arrays: the 18-entry P-array and four 256-entry S-boxes. In the Data Encryption phase, encryption occurs via 16-round networks. Each round consists of a key dependent permutation and a key and data-dependent substitution.

Blowfish is one of the fastest block ciphers in use, except when changing keys. Blowfish is not subject to any patents and is therefore freely available for anyone to use. This has contributed to its popularity in cryptographic software.

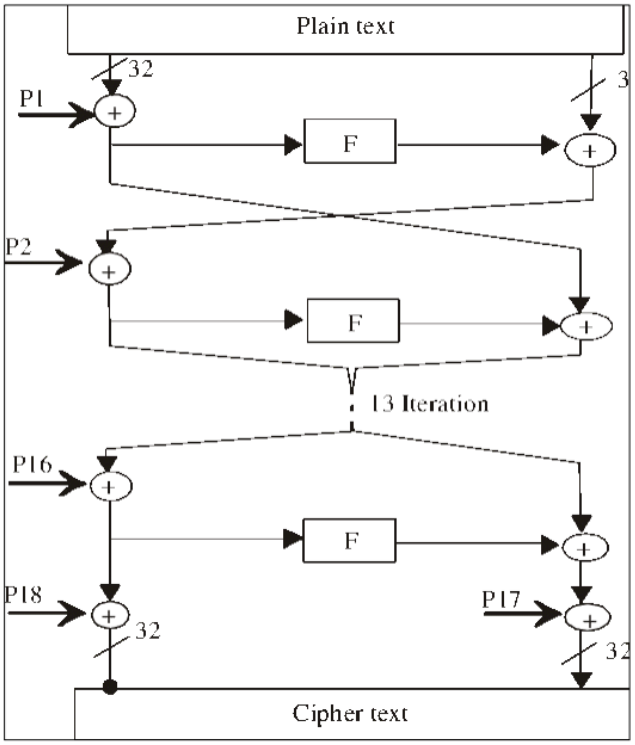


Figure: Blowfish Algorithm

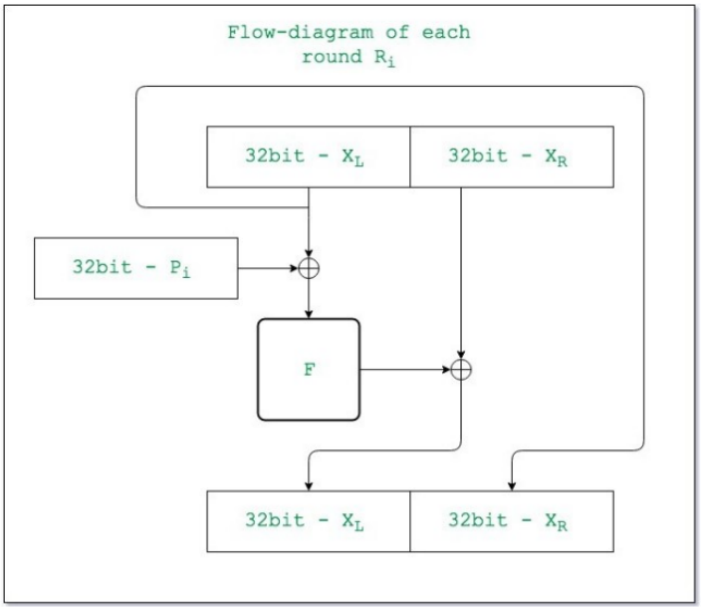


Figure: Flow Diagram Of Each Round

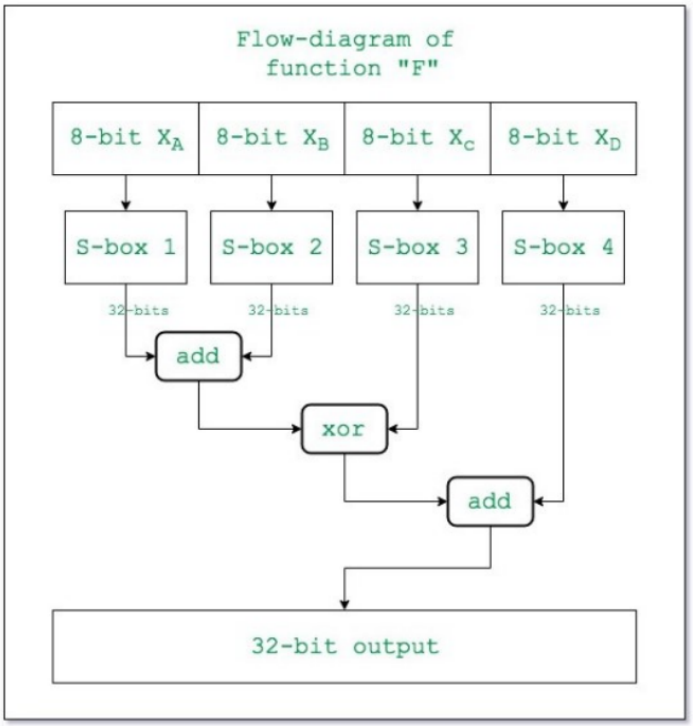


Figure: Flow diagram of each F (function)

4.3.2. SRNN Algorithm

The SRNN calculation is an open key cryptography calculation with some changes. In this calculation, an amazingly vast number having two prime components is utilized. Notwithstanding, two short range characteristic numbers in combination of keys are utilized. This change expands the security of the cryptosystem. SRNN is utilized for secure correspondence amongst client and cloud servers. At the decoding end, the client will give n SRNN private keys.

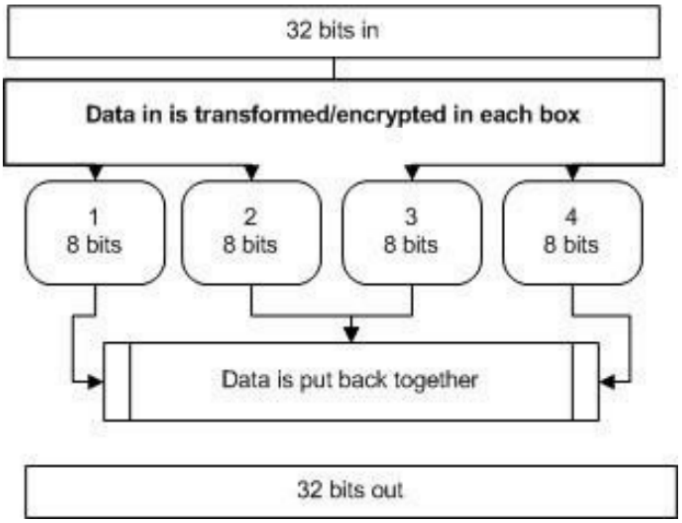


Figure: SRNN Encryption

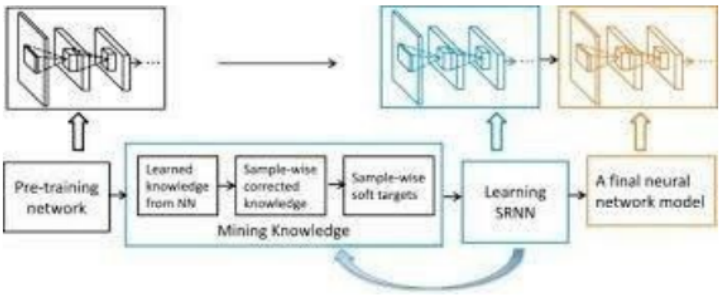


Figure: SRNN Decryption

5. IMPLEMENTATION

5.1. Code

TestFileEncryption.java 7

Users > shubhamkaushik > Documents > ism > TestFileEncryption.java > ...

```java
import java.math.BigInteger;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.security.Key;
import java.time.format.TextStyle;
import java.util.*;
import java.io.*;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

public class TestFileEncryption{
    private static final String ALGORITHM = "Blowfish";
    private static String keyString;
    public static void encrypt(File inputFile, File outputFile)throws Exception {
    doCrypto(Cipher.ENCRYPT_MODE, inputFile, outputFile);
    System.out.println("Fileencrypted successfully!");
    }
    public static void decrypt(File inputFile, File outputFile)throws Exception {
    doCrypto(Cipher.DECRYPT_MODE, inputFile, outputFile);
        System.out.println("File decrypted successfully!");
    }
    private static void doCrypto(int ciphe String keyString ile,File outputFile) throws Exception {
        Key secretKey = new SecretKeySpec(keyString.getBytes(),ALGORITHM);
        Cipher cipher = Cipher.getInstance(ALGORITHM);
        cipher.init(cipherMode, secretKey);
        FileInputStream inputStream = new FileInputStream(inputFile);
        byte[] inputBytes = new byte[(int) inputFile.length()];
        inputStream.read(inputBytes);
        byte[] outputBytes = cipher.doFinal(inputBytes);
        FileOutputStream outputStream = new FileOutputStream(outputFile);
        outputStream.write(outputBytes);
        inputStream.close();
        outputStream.close();
    }
    Run | Debug
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter the blowfish key to Encrypt thefile:"); String s1=in.nextLine();
        int num;
        BigInteger d,n,phi,u,a,e1,p,q,ua,b,c,v;
        int j;
        String encrypt="";
        BigInteger t=new BigInteger("1");
        p=new BigInteger("11");
        q=new BigInteger("7");
```

Ln 1, Col 1 Spaces: 4 UTF-8 LF Java Go Live JavaSE-17

TestFileEncryption.java 7

Users > shubhamkaushik > Documents > ism > TestFileEncryption.java > ...

```java
        File decryptedFile =new File("/Users/shubhamkaushik/Documents/ism/DecryptedFile.docx");
        try{
            TestFileEncryption.keyString=s1;
            TestFileEncryption.encrypt(inputFile, encryptedFile);
            for(j=0;j<s1.length();j++)
            {
                ekey[j]=BigInteger.valueOf((int)s1.charAt(j));//System.out.println(ekey[j]);
            }
            System.out.println("Your Encrypted key usingSRNN is:");
            for(j=0;j<s1.length();j++)
            {
                b=(ekey[j].multiply(ua)).pow(7);
                enkey[j]=b.mod(n);
                System.out.println(enkey[j]);
            }
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
        System.out.println("To decrypt your file Enter the SRNN keygenerated:");
        v=(u.pow(phi.intValue()-a.intValue())).mod(n);
        System.out.println("Enter the number of keys to decrypt thefile:");
        num=in.nextInt();
        System.out.println("Enter the values:");
        for(j=0;j<num;j++){
            arr[j]=in.nextBigInteger();
        }
        for(j=0;j<num;j++)
        {
            dearr[j]=(((v.pow(7)).multiply(arr[j])).pow(43)).mod(n);
            encrypt=encrypt+(char)dearr[j].intValue();
        }
        System.out.println("Your blowfish key is:"+encrypt);
        try {
            TestFileEncryption.keyString = encrypt;
            TestFileEncryption.decrypt(encryptedFile, decryptedFile);
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

Ln 1, Col 1 Spaces: 4 UTF-8 LF Java Go Live JavaSE-17

TestFileEncryption.java 7

Users > shubhamkaushik > Documents > ism > TestFileEncryption.java > ...

```java
        v=(u.pow(phi.intValue()-a.intValue())).mod(n);
        System.out.println("Enter the number of keys to decrypt thefile:");
        num=in.nextInt();
        System.out.println("Enter the values:");
        for(j=0;j<num;j++){
            arr[j]=in.nextBigInteger();
        }
        for(j=0;j<num;j++)
        {
            dearr[j]=(((v.pow(7)).multiply(arr[j])).pow(43)).mod(n);
            encrypt=encrypt+(char)dearr[j].intValue();
        }
        System.out.println("Your blowfish key is:"+encrypt);
        try {
            TestFileEncryption.keyString = encrypt;
            TestFileEncryption.decrypt(encryptedFile, decryptedFile);
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL

Java Process Console

```
123456789
Fileencrypted successfully!
Your Encrypted key usingSRNN is:
49
17
76
51
47
57
11
42
52
To decrypt your file Enter the SRNN keygenerated:
Enter the number of keys to decrypt thefile:
```

Ln 1, Col 1 Spaces: 4 UTF-8 LF Java Go Live JavaSE-17

TestFileEncryption.java 7

Users > shubhamkaushik > Documents > ism > TestFileEncryption.java > ...

```java
        v=(u.pow(phi.intValue()-a.intValue())).mod(n);
        System.out.println("Enter the number of keys to decrypt thefile:");
        num=in.nextInt();
        System.out.println("Enter the values:");
        for(j=0;j<num;j++){
            arr[j]=in.nextBigInteger();
        }
        for(j=0;j<num;j++)
        {
            dearr[j]=(((v.pow(7)).multiply(arr[j])).pow(43)).mod(n);
            encrypt=encrypt+(char)dearr[j].intValue();
        }
        System.out.println("Your blowfish key is:"+encrypt);
        try {
            TestFileEncryption.keyString = encrypt;
            TestFileEncryption.decrypt(encryptedFile, decryptedFile);
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

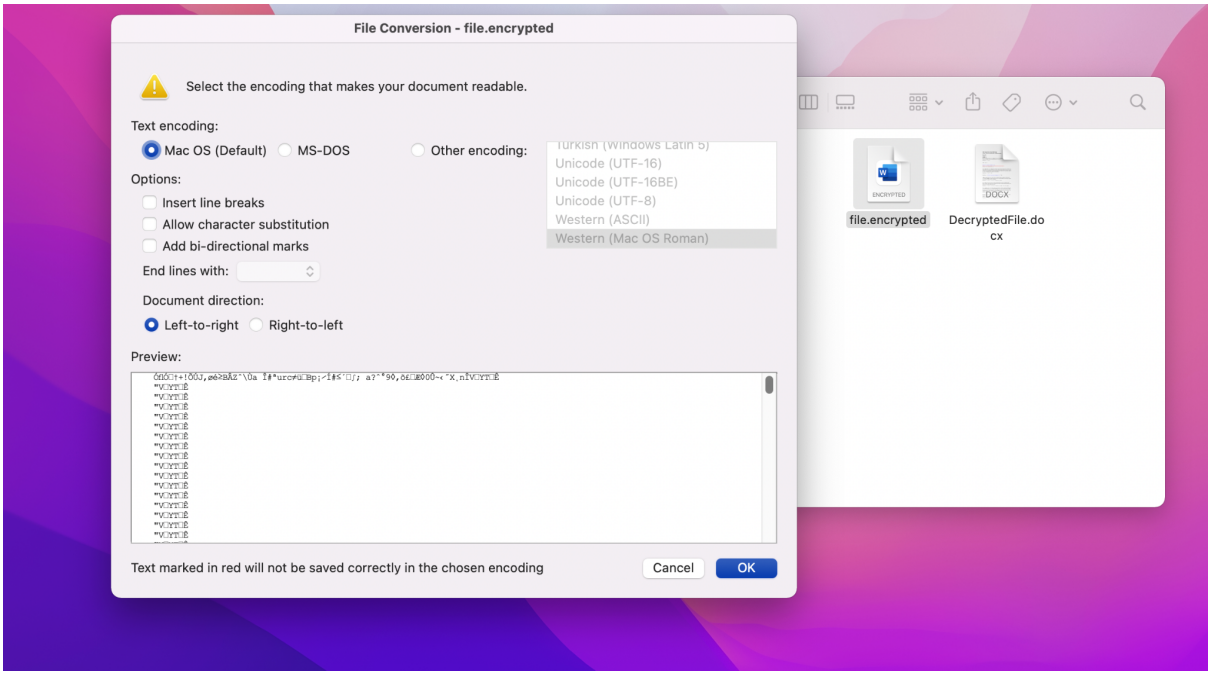
PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL

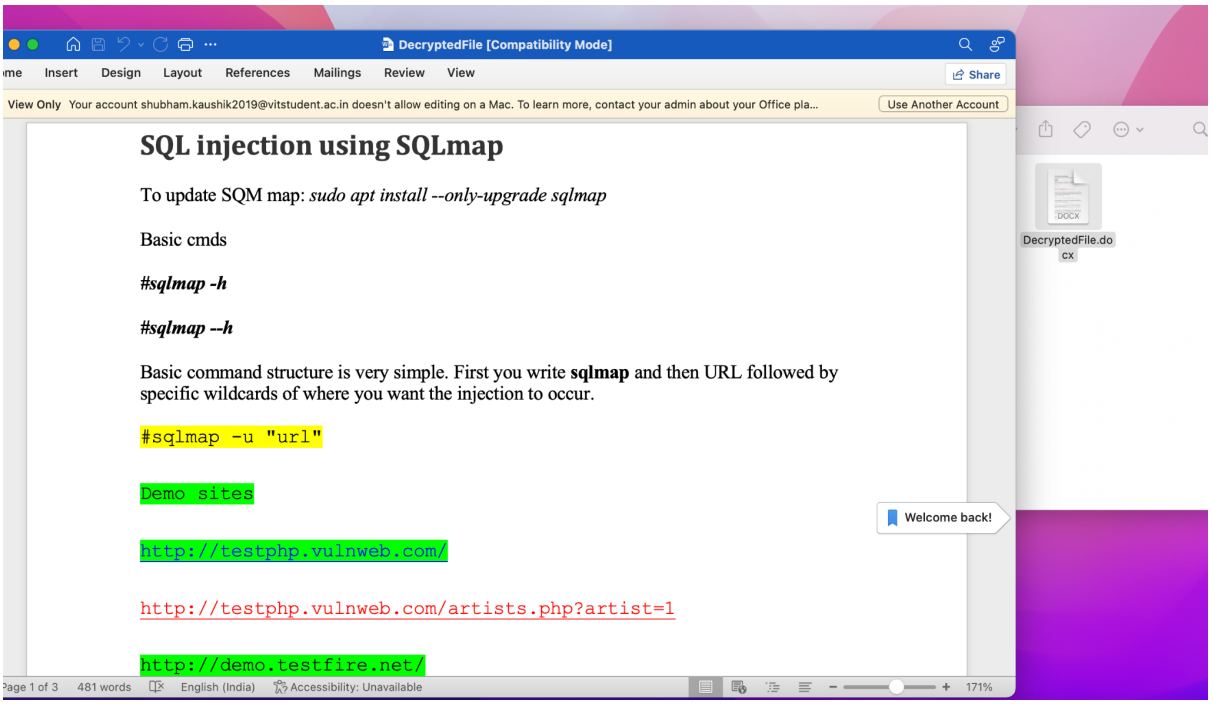
Java Process Console

```
42
52
To decrypt your file Enter the SRNN keygenerated:
Enter the number of keys to decrypt thefile:
9
Enter the values:
49
17
76
51
47
11
42
52
```

Ln 1, Col 1 Spaces: 4 UTF-8 LF Java Go Live JavaSE-17

5.2 Screenshots





6. RESULTS

The proposed model is liable to meet the required security needs of the datacenter of the cloud. Blowfish used for the encryption of file slices takes minimum time and has maximum throughput for encryption and decryption from other symmetric algorithms. Modified RSA (SRNN) has increased security than RSA. The idea of using blowfish key and SRNN to encrypt the keys meet the principle of data security. The hybrid approach when deployed in a cloud environment makes the remote server more secure and thus, helps the cloud providers to fetch more trust of their users. For data security and privacy protection issues, the fundamental challenge of separation of sensitive data and access control is fulfilled.

The various benefits are as summarized: The public key cryptography used helps to facilitate authorization of users for each file. The need for a more light and secure encryption system forfile information preserving systems on cloud is satisfied. The file splitting and merging makes the model unfeasible to get attacked. The file splitting and merging makes the model unfeasible to get attacked.

7. CONCLUSION AND FUTURE SCOPE

As indicated by benefit conveyance models and organization models of the cloud, information security and protection assurance are the essential issues that should be tackled. Information Security and protection issues exist in all levels in SPI benefit conveyance models. The previously mentioned demonstration is productive in information as an administration, which can be stretched out in other administration models of cloud.

8. REFERENCES

1) Peter Mel and Tim Grace, "The NIST Definition of Cloud Computing", NIST, 2010.

2) Achill Buhl, "Rising Security Challenges in Cloud Computing", in Proc. of World on Information and correspondence Technologies ,pp. 217-222, Dec. 2011.

3) Srinivasarao D et al., "Breaking down the Superlative symmetricCryptosystem Encryption Algorithm", Journal of Global Research inComputer Science, vol. 7, Jul. 2011.