VIT®

Vellore Institute of Technology

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

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING (SCOPE)

October 2019

CSE4003 - CYBER SECURITY

FINAL PROJECT REPORT

Secure File Storage in Cloud Computing using Hybrid Cryptography Algorithm

Under the guidance of,
Prof. Pramod Kumar Maurya

By

Kaushik (17BCE0398)
Divam Kesharwani (17BCE0541)

SLOT: A2 + TA2

TABLE OF CONTENTS

1. Abstract	2
2. Keywords	2
3. Introduction	3
4. Literature survey	4
5. Proposed Algorithms with Pseudo-Code	6
6. Performance evaluation	15
7. Conclusion	18
8. References	20
9. Appendix	22

1. ABSTRACT

In Cloud Computing, we share data among many clients, servers and people. So the security of information present in cloud is not guaranteed. Thus it is simple for an intruder to access and demolish the first type of information.

Symmetric Key Cryptography is quick and productive. Anyway, key trade keeps on being prevention towards its ideal use. The individual who scrambles the message and the individual, who unscrambles the message in Symmetric Key Cryptography utilize a similar key and consequently keeping up the security of the regular key, without it coming into the learning of others, is an extreme inquire. Uneven Key Cryptography is gainful in annihilating this issue. Here each imparting gathering utilizes two keys to shape a key pair - one key is made open (and henceforth called open key) that is utilized to scramble the message.

So, there is a requirement of some plainly key which help us to do cross breed encryption and protect the data. In the proposed project, 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 servers.

2. KEYWORDS

Cloud Computing, Data Security, Hybrid Cryptosystem, Blowfish, RSA, SRNN, Symmetric Key Cryptography, Short Range Natural Number,

3. INTRODUCTION

Cloud computing is 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 user's application and program are residing in provider premises. The cloud provider can solve this problem by encrypting the files by using encryption algorithm. Here we discussed 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.

In existing system single algorithm is used for data encode and decode purposes. But use of single algorithm is not accomplish high level security. If we use single symmetric key cryptography algorithm than we have to face security problem because in this type of algorithm applies a single key for data encode and decode. So key transmission problems occur while sharing key into multiuser environment. Public key cryptography algorithms accomplish high security but maximum delay is needed for data encode and decode. To solve above issues we have introduced new security mechanism.

4. LITERATURE SURVEY

In today's applications storing the data in cloud is the most needed requirement. Every application requires information like chat messages, images, videos, user's authentication information to be stored in the cloud. Social Media is the most popular and often overlooked application of cloud computing. Facebook, LinkedIn, MySpace, Twitter, and many other social networking sites use cloud computing.

So before storing the information or data in the cloud they need to be encrypted so as to protect his/her data from other users (especially from hackers). There are many methods to encrypt the data. We have made a survey and analysed the results as follows:

On the survey of the cloud, the traditional security mechanisms are no longer suitable for applications and data in cloud. Some issues are:

- Due to dynamic scalability, service and location transparency features of cloud computing model, all kinds of application and data of the cloud platform have no fixed infrastructure and security boundaries. In the event of a security breach, it is difficult to isolate a particular resource that has a threat or has been compromised.

- Due to the receptiveness of cloud and sharing virtualized assets by multitenant, client information might be gotten to by other unapproved clients.

- According to service delivery models of Cloud computing, resources and cloud services may be owned by multiple providers. As there is a conflict of interest, it is difficult to deploy a unified security measure.

Hybrid cryptography algorithm present by author A. Shahade. AES and RSA algorithms are used into hybrid algorithm. AES algorithm requires a single key. In hybrid algorithm three keys are used. For data upload on cloud mandatory keys are AES secret key and RSA public key. Private key of RSA and AES secret key are essential to download data from cloud. Whenever use makes an effort to upload data on cloud first that file stored onto directory for short time. In encryption process first AES algorithm is applied on file after that RSA algorithm is applied on encrypted data. Reverse process is followed for decryption. After applying keys that file convert into encoded form and stored on cloud server. Advantages of hybrid algorithm are data integrity, security, confidentiality and

availability. Disadvantage of RSA algorithm is large amount time essential for data encode and decode.[8]

In security model symmetric algorithm uses chunk level encryption and decryption of data in cloud computing. Key size is 256 bit .Key is rotated to achieve high level security. For data integrity purpose hash value is generated. Hash values are generated after encryption and decryption. If both hash values matches than that data is in the correct form. In this security model only valid users can access data from cloud. Advantages of security model are integrity, security and confidentiality.[9]

Three algorithms are used for implementation of hybrid algorithm. User authentication purpose digital signature is used. Blowfish algorithm is used to produce high data confidentiality .It is symmetric algorithm .It uses single key .Blowfish algorithm need least amount of time for encode and decode. Sub key array concept is used into blowfish algorithm. It is block level encryption algorithm. The main aim of this hybrid algorithm is achieve high security to data for upload and download from cloud. Hybrid algorithm solves the security, confidentiality and authentication issues of cloud. [10]

5. PROPOSED ALGORITHM WITH PSEUDO CODE

In the proposed algorithm, we will have two phases - Encryption Phase and Decryption Phase. In the encryption phase, we will encrypt the file using Blowfish Algorithm, and then, the slices will be made. Each slice would be encrypted using SRNN Algorithm, and by doing so, we will obtain different SRNN public keys for different slices. In the Decryption phase, we will provide the SRNN keys, which will be used to decrypt the Blowfish key, which has encrypted the file. After entering the SRNN keys, the Blowfish key will be decrypted and the file deciphered.

Blowfish Algorithm

BlowFish is a symmetric block cipher which uses a Fiestal network, 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 .The algorithm design is shown in figure. It consists of two phases: one is Key Expansion phase another is Data Encryption phase. In Key expansion phase, key is converted into several sub-keys and in Data Encryption phase, encryption occurs via 16-round networks. Each round consists of a key dependent permutation and a key and data-dependent substitution.

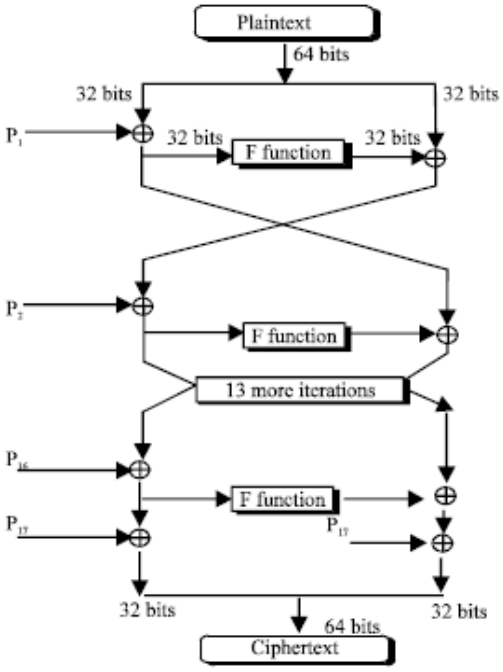


Fig 1: Diagram of Blowfish Process

Encryption-
Encryption with Blowfish has two primary stages: sixteen cycles of the **round function** and **yield operation**.

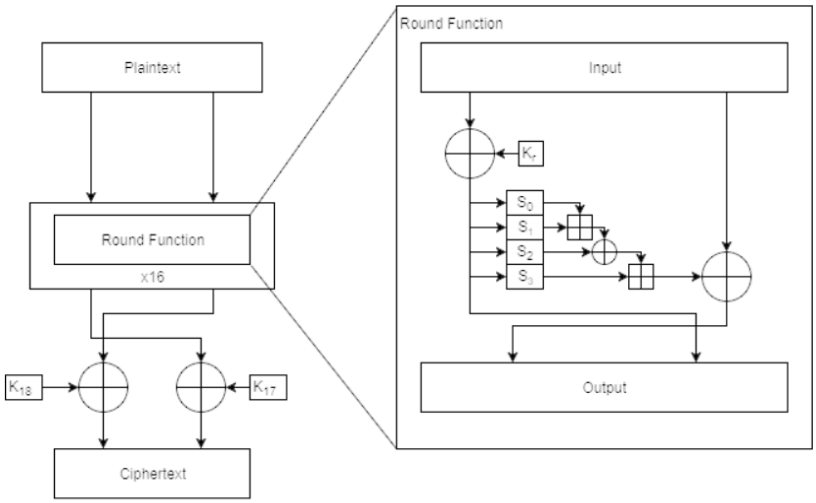


Fig 2: Diagram of Blowfish Encryption

accepting that the given round keys and the estimation of the S-boxes. Subtleties of how the round keys are produced and S-encloses instated is secured the key timetable segment.

Blowfish round function-
The round capacity in Blowfish encryption has four phases:

1. Key brightening of the left half of the contribution with the rth round key
2. Use of the S-Boxes and combination of their outcomes
3. Elite or of the correct side of the contribution with the yield of the F work (key brightening, S-Boxes and blend of S-Box yield)
4. Swapping the sides of the yield

In the **key-whitening stage**, the left half of the information is exclusive-or with the round key for the given round.
The **S-Boxes** play out a 8-piece to 32-piece mapping. The S-Boxes are set as a component of the key age calculation. The yield of a S-Box for a contribution of n is the nth incentive in the S-Box.
The yields of the S-Boxes are joined through a blend of expansion and selective or. The yields of the initial two S-Boxes are included modulo 2^{32}. The outcome is exclusive-or to

the yield of the third S-Box and the consequence of that is added modulo 2^{32} to the yield of the fourth S-Box.

Blowfish output function-

The last phase of the Blowfish figure includes two stages: turning around the last swap and performing yield brightening. In yield brightening, the correct side of the yield (subsequent to being swapped) is exclusive or with the seventeenth round key and the left side is exclusive or with the eighteenth round key. The aftereffect of this is the Blowfish ciphertext.

Blowfish Key Schedule (and S-box generation)-

Initial values

The Blowfish key timetable depends vigorously on the Blowfish encryption calculation depicted in the past segment. The key calendar utilizes a worth, P, comprising of eighteen words which contain (all together) the initial eighteen expressions of the parallel portrayal of the partial piece of pi. For instance, the hexadecimal portrayal of pi starts with 3.243F6A8885A308D313198A2E037073, in this manner P1=0x243F6A88, P2=0x85A308D3, and so on. This worth, P, will turn into the round keys utilized in encryption.

Next, set the underlying estimations of the S-Boxes, in a similar way, starting with the nineteenth expression of the fragmentary piece of pi. The requesting ought to be that the whole first S-Box is dispatched all together before proceeding onward to the following, etc.

Since P contains 18 words and the S-Boxes each contain 256 words, a total of 18 + 4*256 = 1042 pi words are used, each 32-bit in size.

Generating Round Keys and S-box-

Age of the round key is performed in rounds where each round produces two round key qualities. The procedure is as per the following:

1. Instate P and S-Boxes as depicted previously

2. Select or P1 with the initial 32 key bits, P2 with the following 32 bits, etc until the majority of the key has been exclusive-or (since the key is shorter than P, portions of it will be utilized on different occasions to cover all of P)

3. Set the underlying contribution to zero

4. Encode the information utilizing the present adaptation of P as the round keys

5. Set the initial two unreplaced estimations of P to the estimation of the ciphertext from stage 4

6. Set the contribution to the ciphertext from stage 4

7. Rehash stages 4 through 6 until all of P has been supplanted

8. Utilize the subsequent estimation of P as the round keys in encryption

9. Rehash stages 4 through 6, supplanting estimations of the S-Boxes two at once until all S-Box esteems have been supplanted.

Since P contains 18 words and the S-Boxes each contain 256 words, there is an aggregate of 18 + 4*256 = 1042 qualities to supplant, which will make 521 cycles of strides 4 through 6 of the above calculation to finish.

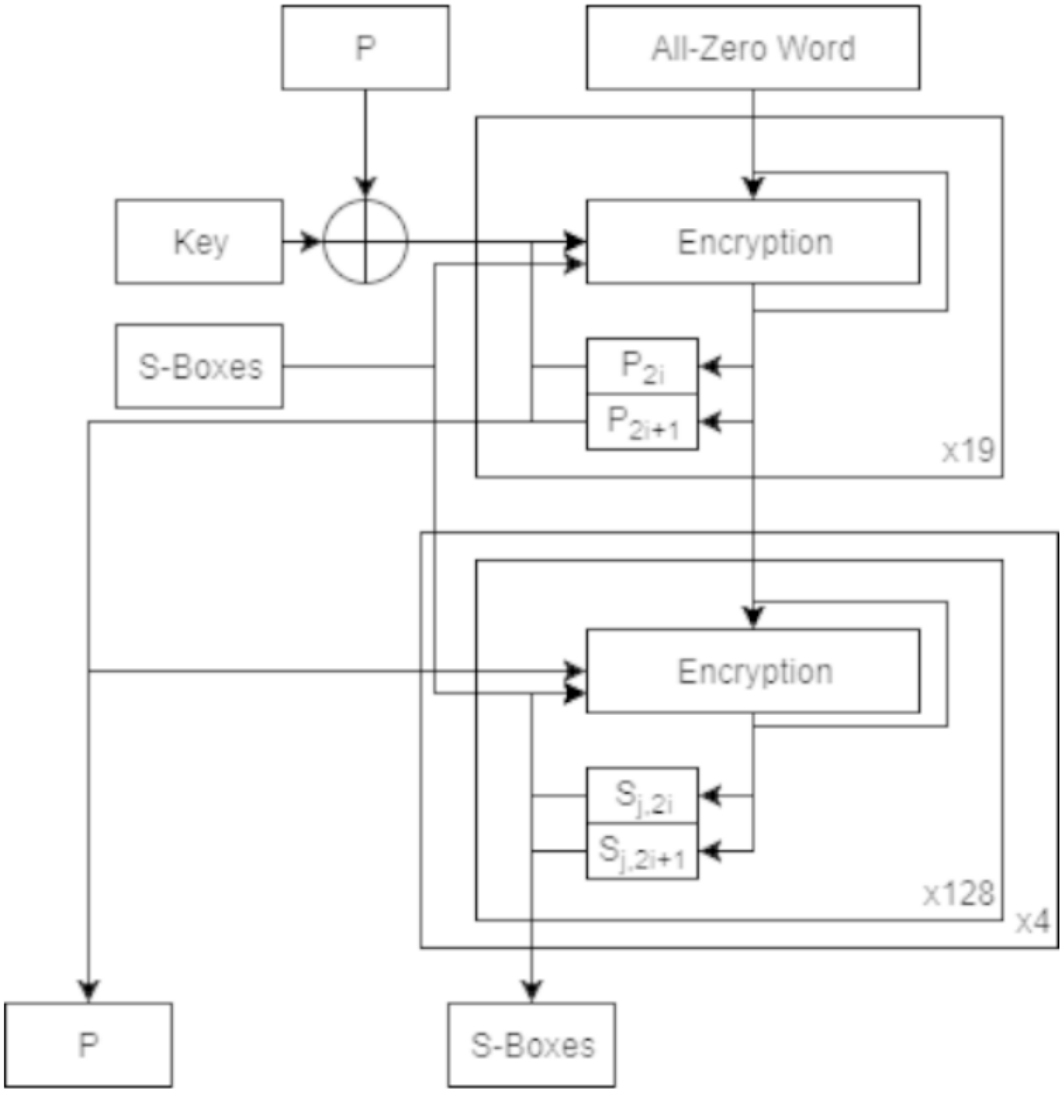


Fig3: Diagram of Blowfish Encryption operation

RSA Algorithm

RSA (Rivest–Shamir–Adleman) is one of the first public-key cryptosystems and is widely used for secure data transmission. In such a cryptosystem, the encryption key is public and it is different from the decryption key which is kept secret (private). In RSA, this asymmetry is based on the practical difficulty of the factorization of the product of two large prime numbers, the "factoring problem". The acronym RSA is made of the initial letters of the surnames of Ron Rivest, Adi Shamir, and Leonard Adleman, who first publicly described the algorithm in 1977.
A client of RSA makes and afterward distributes an open key dependent on two huge prime numbers, alongside a helper esteem. The prime numbers must be stayed quiet. Anybody can utilize the open key to scramble a message, however just somebody with information of the prime numbers can decipher the message. Breaking RSA encryption is known as the RSA issue.

Key generation

Choose two distinct prime numbers p and q.

- For security purposes, the integers p and q should be chosen at random, and should be similar in magnitude but differ in length by a few digits to make factoring harder. Prime integers can be efficiently found using a primality test.
- p and q are kept secret.

Compute $n = pq$.

- n is used as the modulus for both the public and private keys. Its length, usually expressed in bits, is the key length.
- n is released as part of the public key.

Choose an integer e such that $1 < e < \lambda(n)$ and $\gcd(e, \lambda(n)) = 1$; that is, e and $\lambda(n)$ are coprime.

- e having a short bit-length and small Hamming weight results in more efficient encryption – the most commonly chosen value for e is $2^{16} + 1 = 65{,}537$. The smallest (and fastest) possible value for e is 3, but such a small value for e has been shown to be less secure in some settings.
- e is released as part of the public key.

Determine d as $d \equiv e^{-1} \pmod{\lambda(n)}$; that is, d is the modular multiplicative inverse of e modulo $\lambda(n)$.

- This means: solve for d the equation $d \cdot e \equiv 1 \pmod{\lambda(n)}$. d can be computed efficiently by using the Extended Euclidean algorithm.
- d is kept secret as the *private key exponent*.

The public key is (n,e) and the private key (d,p,q). Keep all the values d, p, q and ϕ secret. [Sometimes the private key is written as (n,d) because you need the value of n when using d. Other times we might write the key pair as ((N,e),d).]

Encryption

1. Obtains the recipient B's public key (n,e).
2. Represents the plaintext message as a positive integer m 1<m<n
3. Computes the ciphertext c=me(mod)n
4. Sends the ciphertext c to B.

Decryption

Uses his private key (n,d) to compute $m=c^d(\text{mod})n$.

Algorithm: Generate an RSA key pair.
INPUT: Required modulus bit length, k.
OUTPUT: An RSA key pair ((N,e),d) where N is the modulus, the product of two primes (N=pq) not exceeding k bits in length; e is the public exponent, a number less than and coprime to (p-1)(q-1); and d is the private exponent such that ed≡1mod(p-1)(q-1).

1. Select a value of e from 3,5,17,257,65537
2. Repeat
3. p ← genprime(k/2)
4. until (pmode)≠1
5. Repeat
6. q ← genprime(k - k/2)
7. until (qmode)≠1
8. N ← pq
9. L ← (p-1)(q-1)
10. d ← modinv(e, L)

11. Return (N,e,d)

SRNN Algorithm

SRNN algorithm is similar with RSA with some modification. SRNN algorithm is also a Public key cryptography algorithm. In this algorithm we have extremely large number that has two prime factors. In addition to this, we have used two short range natural numbers in pair of keys. One key (public key) for encryption and other corresponding key (private key) for decryption. This modification increases the security of the cryptosystem. So its name is short range natural number public key algorithm.

Key generation Process

- Generate two large random prime p, q.
- Compute n=p*q
- Compute phi=(p-1)*(q-1)
- Choose an integer e, 1 < e < phi, such that gcd(e,phi) = 1 And compute u^a
- Find d, such that e*d mod(phi) =1
- Pick short range natural number u randomly such that u < phi-1
- Pick another short range natural number a randomly such that u < a < phi, and compute u^a
- Public key is (n, e, u^a)
- Private Key is (d, a, u)
  p, q, phi should also be kept secret.

Encryption Process

- Obtains the recipient's public key (n, e, u^a)
- Represent the plaintext message as positive integer M
- Computes the cipher text C=$(m*u^a)^e$ mod n
  Send the cipher text C to recipient.

Decryption Process

- Use Recipient private key(d, a , u)
- compute M=$(v^e*c)^d$ mod n where v= u^{phi-a} mod n
- Extracts the plaintext from the integer representative M

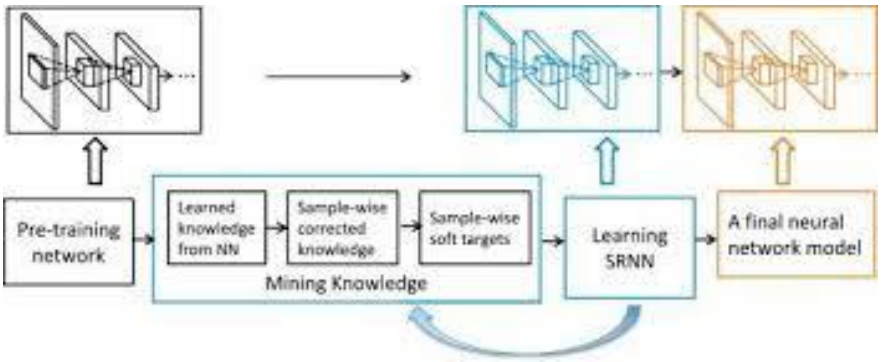


Fig 4: Diagram of SRNN Encryption operation

PSEUDO CODE

```
Class encryption{
        ALGORITHM <- “Blowfish”
        Keystring
        Function encrypt(inputfile, outputfile){
                Encrypt the file}
        Function decrypt(inputfile, outputfile){
                decrypt the file}
        Function secret_key_formation{
                Create the secret key from the byte array for both inputfilestream and
outputfilestream
        Function main{
                String s1 <- input of the blowfish key
                Encrypt <- empty string
                BigInteger p,q,phi,n,u,a <- initialize them for use in forming SRNN keys
                BigInteger arr, dearr <- arrays for the encryption and decryption of the file
streams
                BigInteger ekey, enkey <- arrays for keys
                File file <- new file object of created file
                File enfile< - new file object of the encrypted file
                File defile< - new file object of the decrypted file
                Try{    Calculate the srnn keys from the given blowfish keys (encryption)}
                Catch {         return exception if error}
                Try{    Calulate the blowfish keys from the obtained srnn keys
(decryption)}
                Catch{          return exception if error or if the decryption is not done
properly}
```

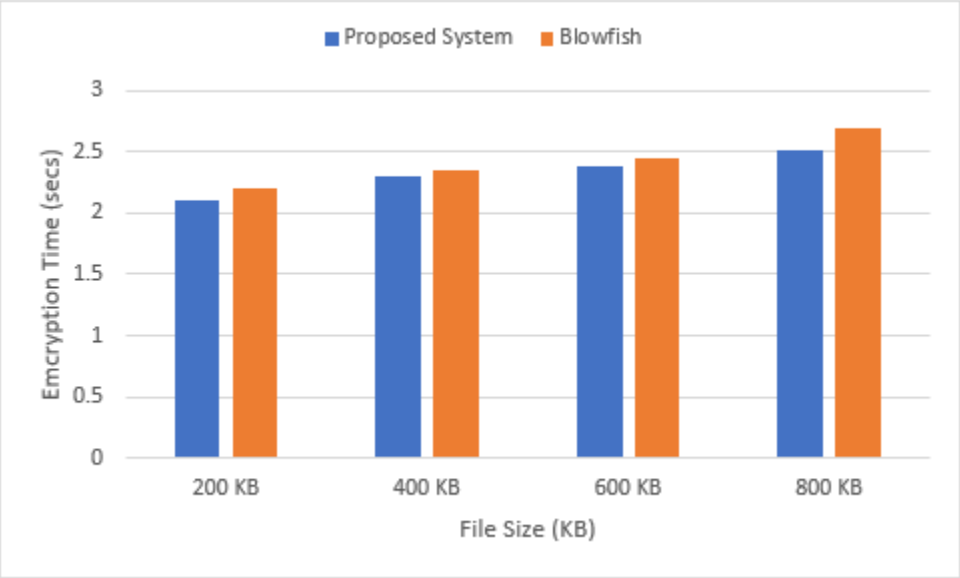
6. PERFORMANCE EVALUATION

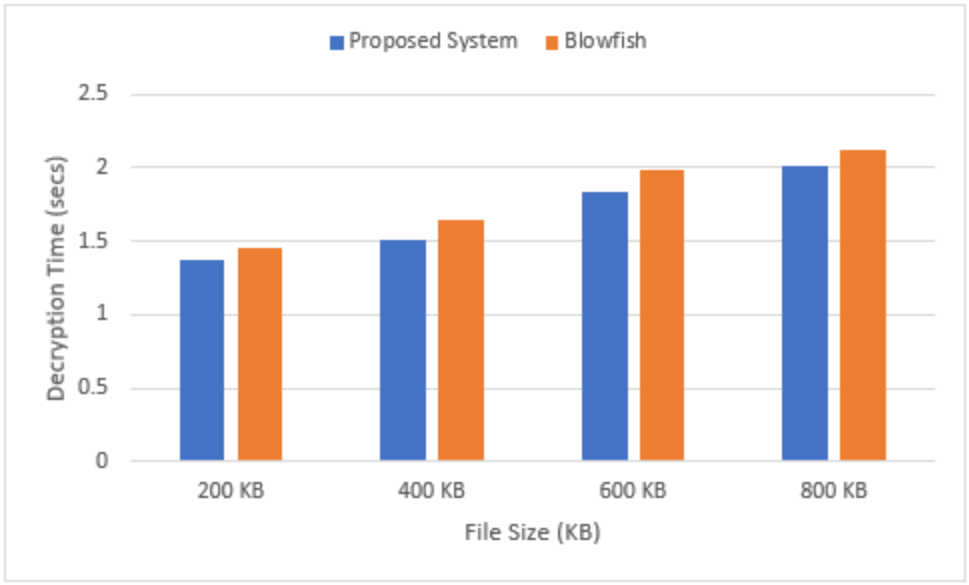
The following graphs have been used to evaluate the performance of the proposed hybrid system with its constituent algorithms i.e. RSA algorithm and Blowfish algorithm

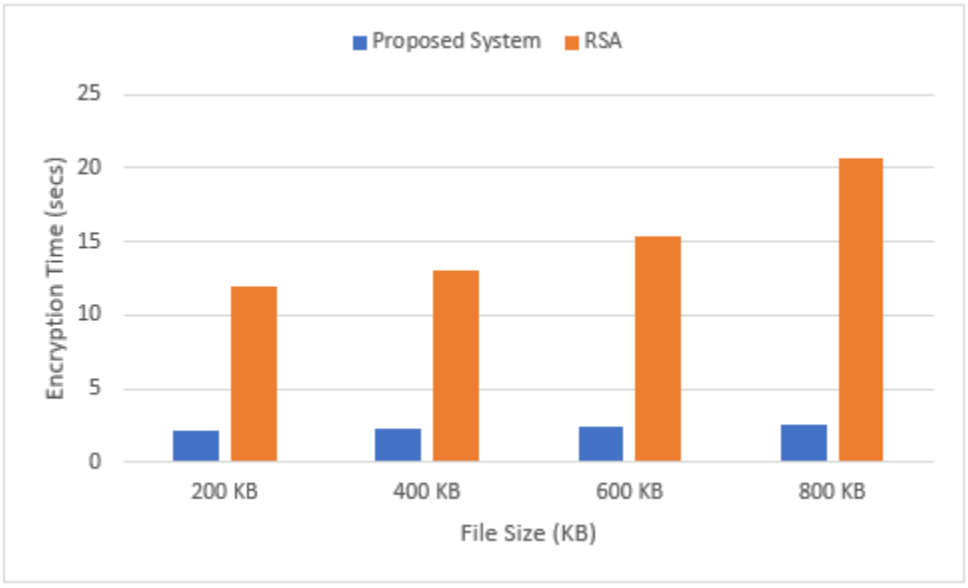
The parameters used for this are

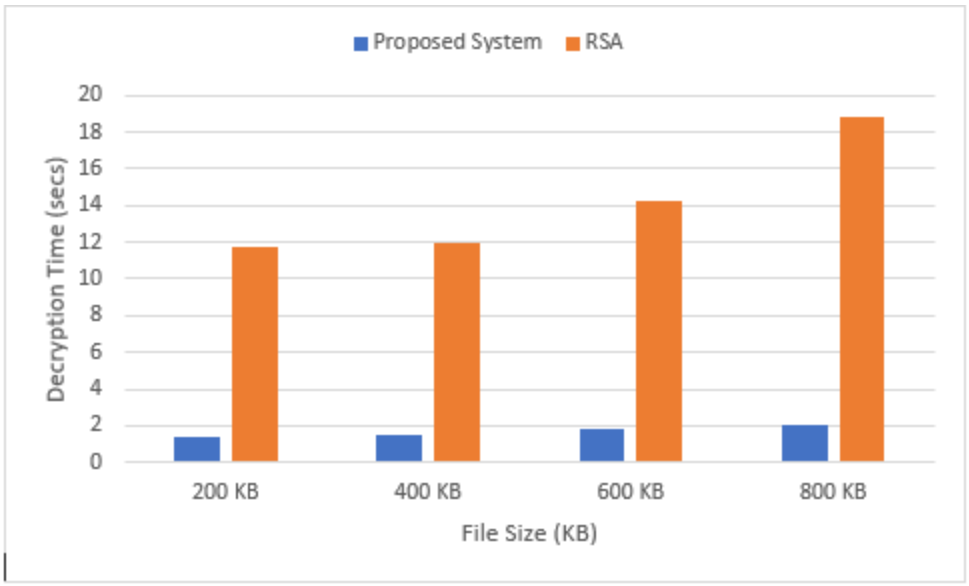
1. Encryption Time
2. Decryption Time

All the evaluation has been done on .docx files with file size of 200, 400, 600 and 800 KB.




Proposed System
Blowfish
2.5
2
1.5
1
0.5
0
Decryption Time (secs)
200 KB
400 KB
600 KB
800 KB
File Size (KB)


Proposed System
RSA
25
20
15
10
5
0
Encryption Time (secs)
200 KB
400 KB
600 KB
800 KB
File Size (KB)



The graphs show that there is negligible difference in the encryption and decryption times of proposed System and Blowfish System, but there is a significant difference in RSA, due to big values of prime numbers and the random number generators.

The proposed system is faster than the Blowfish due to usage of SRNN algorithm, which helps in the key changing at a faster rate, which lacks in the Blowfish algorithm.

7. CONCLUSION

The proposed model is liable to meet the required security needs of data center of 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 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 user for each file. The need of more light and secure encryption system for file information preserving system on cloud is satisfied.

The file splitting and merging makes the model unfeasible to get attacked.

As indicated by benefit conveyance models and organization models of 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 demonstrate is productive in information as an administration, which can be stretched out in other administration models of cloud. Likewise it is tried in cloud condition like Open Nebula[7] , in future this can be conveyed in other cloud situations and the best among of all can be picked.

OUTPUT

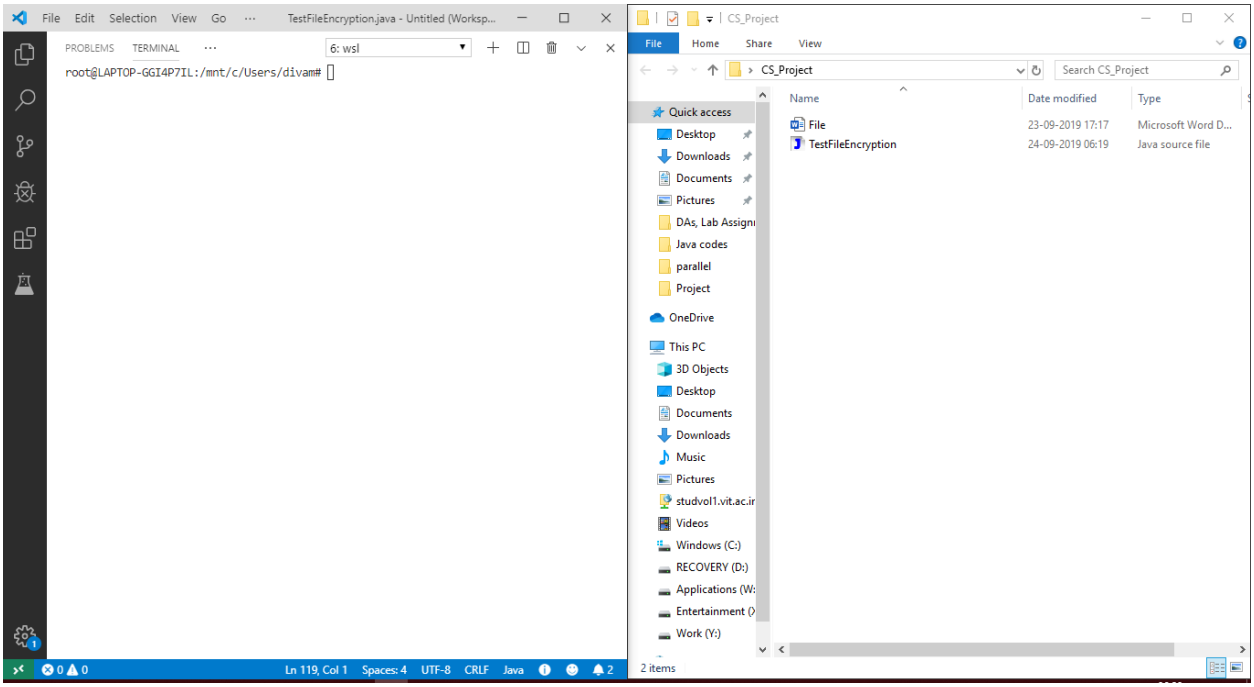


Fig 5: File Encryption

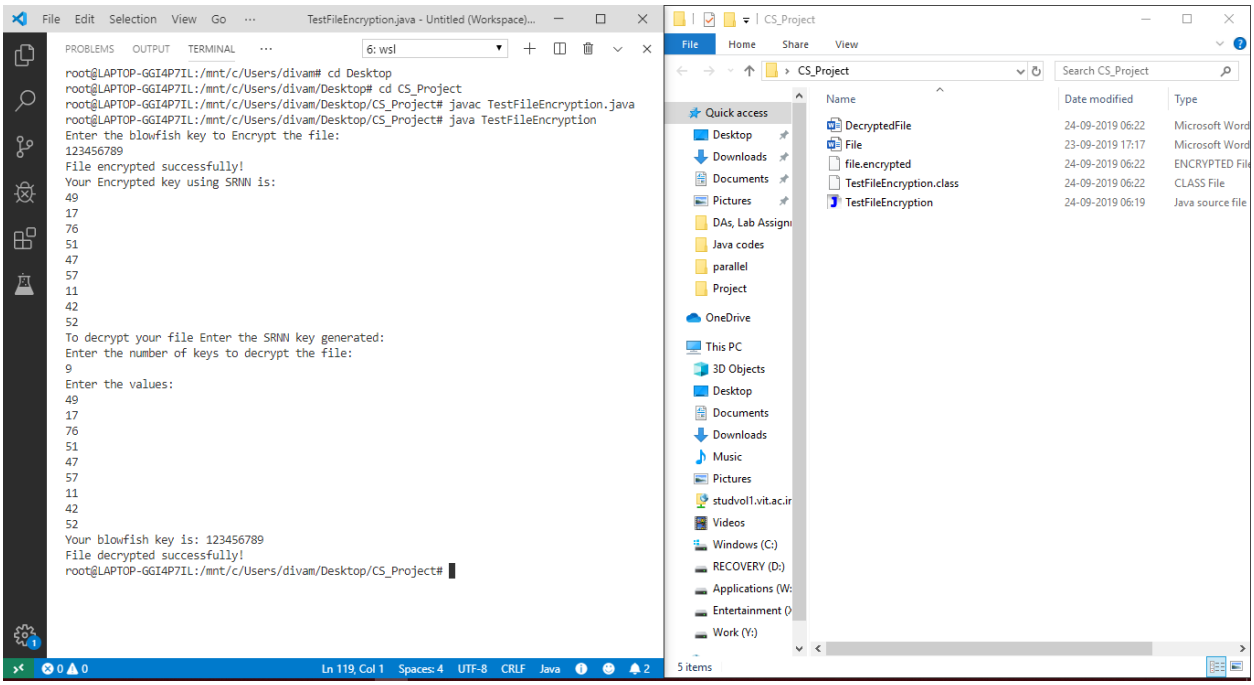


Fig 6: File Decryption

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 Congress on Information and correspondence Technologies ,pp. 217-222, Dec. 2011.

[3]Srinivasarao D et al., "Breaking down the Superlative symmetric Cryptosystem Encryption Algorithm", Journal of Global Research in Computer Science, vol. 7, Jul. 2011

[4] Tingyuan Nye and Tang Zhang "An investigation of DES and Blowfish encryption algorithm" , in Proc. IEEE Region 10 Conference, pp. 1-4 ,Jan. 2009.

[5] Jitendra Singh Adam et al.," Modified RSA Public Key Cryptosystem Using Short Range Natural Number Algorithm" , International Journal of Advanced Research in Computer Science and Software Engineering ,vol. 2,Aug. 2012.

[6] Manikandan.G et al., "A changed cryptographic plan improving information", Journal of Theoretical and Applied Information Technology, vol. 35, no.2, Jan. 2012.

[7] Niles Maintain and Subhead Bhingarkar, " The examination and Judgment of Nimbus, Open Nebula and Eucalyptus", International Journal of Computational Biology , vol. 3, issue 1, pp 44-47, 2012.

[8] V.S. Mahalle , A. K. Shahade, “Enhancing the Data Security in Cloud by Implementing Hybrid (Rsa & Aes) Encryption Algorithm”, IEEE , INPAC,pp 146-149,Oct .2014.

[9] Inder Singh, M. Prateek,” “Data Encryption and Decryption Algorithms using Key Rotations N. Sharma ,A.Hasan, “A New Method Towards Encryption Schemes (Name-BasedEncryption Algorithm)”,IEEE, International Conference on Reliability, Optimization and Information Technology,pages 310-313,Feb 2014.

[10] Jasleen K., S.Garg[,“Security in Cloud Computing using Hybrid of Algorithms”,IJERJS, Volume 3, Issue 5, ISSN 2091-2730,pages 300-305, September-October, 2015.

[11] Comparison of symmetric and asymmetric cryptography with existing vulnerabilities and countermeasures byYogesh Kumar, Rajiv Munjal, and Harsh ,(IJAFRC) Volume 1, Issue 6, June 2014. ISSN 2348 - 4853

[12] Comparative analysis of performance efficiency and security measures of some encryption algorithms by AL.Jeeva, Dr.V.Palanisamy, K.Kanagaram compares symmetric and asymmetric cryptography algorithms ISSN: 2248-9622

[13] New Comparative Study Between DES, 3DES and AES within Nine Factors Hamdan.O.Alanazi, B.B.Zaidan, . A.Zaidan, Hamid A.Jalab, M.Shabbir and Y. Al-Nabhani JOURNAL OF COMPUTING, VOLUME 2, ISSUE 3, MARCH 2010, ISSN 2151-9617

[14] Comparative Study of Symmetric and Asymmetric Cryptography Techniques by Ritu Tripathi, Sanjay Agrawal compares Symmetric and Asymmetric Cryptography Techniques using throughput, key length, tunability, speed, encryption ratio and security attacks. IJCSMS International Journal of Computer Science and Management Studies, Vol. 11, Issue 03, Oct 2011 ISSN (Online): 2231-5268

[15] Evaluation of Blowfish Algorithm based on Avalanche Effect by Manisha Mahindrakar gives a new performance measuring metric avalanche effect. International Journal of Innovations in Engineering and Technology (IJIET) 2014

[16] Efficient Implementation of AES, RituPahal, Vikaskumar, Volume 3, Issue 7, July 2013 ISSN: 2277 128X, © 2013, IJARCSSE

[17] Superiority of blowfish Algorithm ,Pratap Chandra Mandal , Volume 2, Issue 9, September 2012 ISSN: 2277 128X International Journal of Advanced Research in Computer Science and Software Engineering

[18] A study and performance of RSA algorithm, IJCSMC, Vol. 2, Issue. 6, June 2013, pg.126 – 139, ISSN 2320– 088X

[19] Data encryption and decryption by using triple DES and performance analysis of crypto system, Karthik .S ,Muruganandam .A, ISSN (Online): 2347-3878 Volume 2 Issue 11, November 2014

9. APPENDIX

```
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("File encrypted 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 cipherMode, File inputFile, 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();
    }

    public static void main(String[] args)
    {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter the blowfish key to Encrypt the file:");
```

```
        String s1 = in.nextLine();
        int num;
        BigInteger d, n, phi, u, a, e, p, q, ua, b, c, v;
        int j;
        String encrypt = "";
        BigInteger t = new BigInteger("1");
        p = new BigInteger("11");
        q = new BigInteger("7");
        n = p.multiply(q);
        phi = (p.subtract(t)).multiply(q.subtract(t));
        e = new BigInteger("7");
        d = new BigInteger("43");
        u = new BigInteger("3");
        a = new BigInteger("31");
        ua = u.pow(a.intValue());
        BigInteger[]arr = new BigInteger[100];
        BigInteger[]dearr = new BigInteger[100];
        BigInteger[]ekey = new BigInteger[100];
        BigInteger[]enkey = new BigInteger[100];
        int i1, i2;

        File inputFile = new File("File.docx");
        File encryptedFile = new File("file.encrypted");
        File decryptedFile = new File("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 using SRNN is:");
            for(j=0; j<s1.length(); j++)
            {
                b = (ekey[j].multiply(ua)).pow(e.intValue());
                enkey[j] = b.mod(n);
                System.out.println(enkey[j]);
            }
        }
        catch (Exception z)
        {
            z.printStackTrace();
        }

        System.out.println("To decrypt your file Enter the SRNN key
generated:");
        v = (u.pow(phi.intValue() - a.intValue())).mod(n);
        System.out.println("Enter the number of keys to decrypt the
file:");
        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(e.intValue())).multiply(arr[j])).pow(d.intValue())).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 z)
        {
            z.printStackTrace();
        }
    }
}
```