School of Information Technology and Engineering

Educational Application Using Various Cryptographic Algorithms

A project submitted
in partial fulfillment of the requirements for the degree of
M.Tech. (SE)

By

Sirigiri Sri Sai Sharanya (19MIS0266)

Shakthivel RK (19MIS0313)

Course Instructor

Dr. Selva Rani B

Associate Professor

Dec 2021

UNDERTAKING

This is to declare that the project entitled “Project Title” is an original work done by undersigned, in partial fulfillment of the requirements for the degree *M.Tech. (Software Engineering)* at School of Information Technology and Engineering, VIT, Vellore.

All the analysis, design and system development have been accomplished by the undersigned.

Sirigiri Sri Sai Sharanya 19MIS0266
Shakthivel RK 19MIS0313

CONTENT

ABSTRACT

Due to the increasing complexity of computational technology, encryption has become extremely relevant. This project undertakes and explains the working behind various security standards and techniques through an android application which allows for encryption, decryption and hashing of text using a myriad of encryption techniques such as AES, 3DES and SHA-512. The algorithms used ensure data privacy for the text a user might wish to encrypt; thereby, allowing the application to serve as, not only an educational but also a functional tool.

1. Introduction

The Android application helps the students to learn Encryption and Hashing algorithms with ease. Many students perceive that the algorithms are tricky and hard to acknowledge. The app has both encryption and decryption modules of various algorithms with step-by-step approach.

Objectives:

• Help the student to understand.

• Deliver step-wise solving technique.

• Make the algorithm code fragment stable for the application.

2. Literature Survey

Paper1:

Title: Comparison of Various Encryption Algorithms and Techniques for improving secured data Communication

Authors: Soheila Omer AL Faroog Mohammed Koko, 2Dr.Amin Babiker A/Nabi Mustafa

Overview: This paper is about information security using cryptography technique. The authors have done a study about the network security using cryptography. The authors also made an attempt to compare various algorithms with respect to many factors.

Conclusion:

A breif introudction of data security and its neccessity has been presented. The performace of various algorithms in comparsion of encryption times for various common symmetric encryption algorithms has been presented.

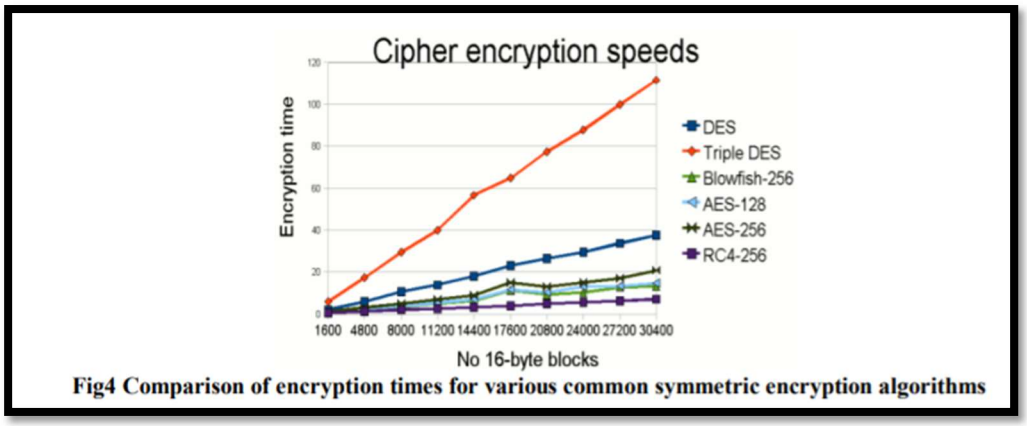


Fig4 Comparison of encryption times for various common symmetric encryption algorithms

Characteristics such as key size, speed and security of RC4, Blowfish, AES, DES and triple des are described. The discussion on types of cryptography has also been made explaining symmetric, asymmetric, modern cryptographic algorithms. The origin and patent related issues of various cryptographic algorithms is also included. The criteria for selecting a cryptographic algorithm are presented where they discussed about factors such as Level of protection, key management, overheads related to cryptographic algorithms. Finally, they mentioned about the hacker attacks possible and methods to encounter them.

Paper 2:

Title: A New Hybrid Data Encryption and Decryption Technique to Enhance Data Security in Communication Networks: Algorithm Development

Authors: Adedeji Kazeem B. and Ponnle Akinlolu A.

Overview: In this paper, the authors have given an explanation about the most used algorithms DES and RSA and also a hybrid approach which they developed from these algorithms. They have also presented the reasons they choose the algorithms DES and RSA. The performance of the developed hybrid algorithm is in detail compared to the DES algorithm and then briefly with the other algorithms.

Conclusion:

In Introduction they stated about various types of encryptions and hashing techniques present in cryptography, and also about various previous studies which have been done on developing a hybrid cryptographic algorithm and their limitations. They next presented the design methodology where they gave detailed description of the DES and RSA algorithm and as well as the hybrid technique implementation they have developed. Here is the block diagram of the hybrid technique they developed.

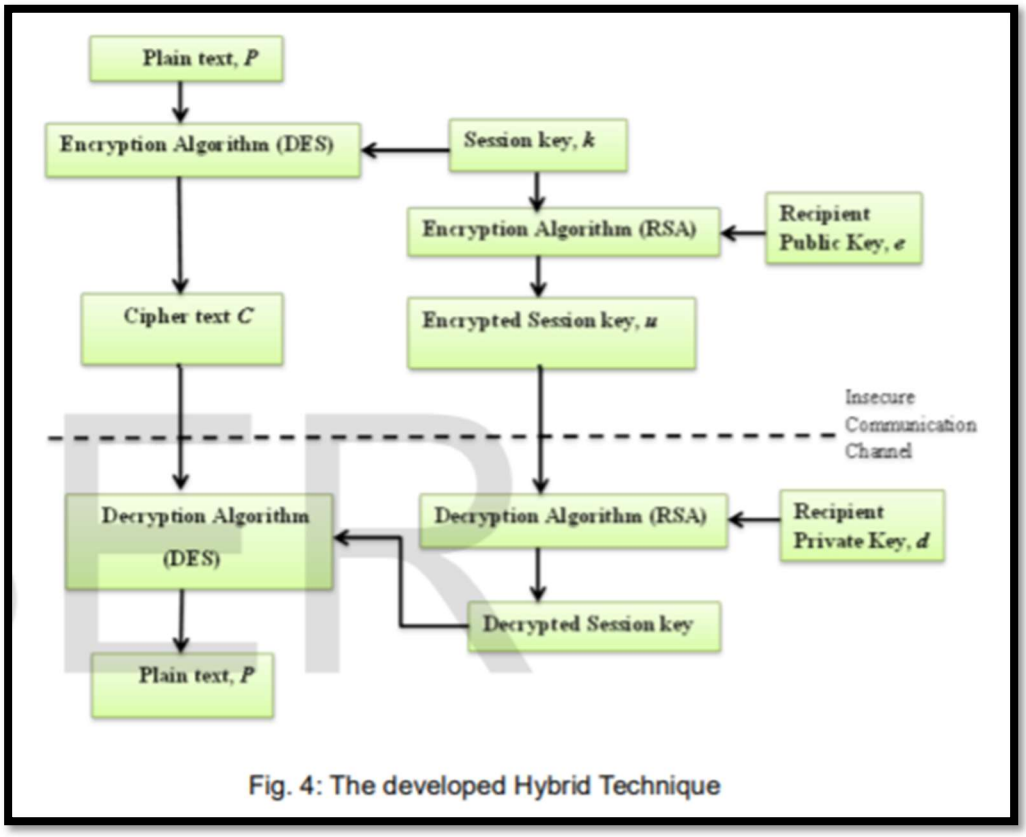


Fig. 4: The developed Hybrid Technique

After explaining the working mechanism of the hybrid technique, the next discussion was about the different metrics that were used to evaluate the performance of the system, such as encryption and decryption time, throughput, average data rate and central processing unit power consumption.

The results from those tests conducted while comparing DES and hybrid technique whereas the results of all other algorithms were presented briefly in the document. Finally, the authors concluded with the result that the developed hybrid (DES-RSA) technique has a better throughput

than the other hybrid techniques and why it is suitable choice for encrypting data in modern applications.

Paper 3:

Title: Encryption and Decryption of Text using AES Algorithm

Authors: Roshni Padate, Aamna Patel

Overview: This paper completely involves the discussion related to AES algorithm, the means of its origin, it's limitations, advantages as well as the future work that can be done on this algorithm.

Conclusion:

The introduction was about cryptography, the purpose of cryptography and specially encryption in detail. The origin of the AES was discussed and the evaluation criteria using which the AES was selected was described. They were Security, Cost and Algorithm and implementation characteristics. Difference between AES and DES was presented along with the characteristics of the AES and its parameters. The Rijndael Algorithm which plays a major role in AES algorithm was discussed and it is also mentioned that this algorithm was the fastest algorithm in terms of the critical path. The encryption and decryption techniques of the AES algorithm was presented and this article is concluded with mentioning how we can utilize AES algorithm for future work such as Image encryption and decryption.

Paper 4:

Title: A Study of Encryption Algorithms (RSA, DES, 3DES and AES) for Information Security

Authors: Gurpreet Singh, Supriya

Overview: This paper was the summary work of various another pervious done research on comparing different algorithms in different perspectives and it also gave a detailed explanation of the DES, AES, 3DES, RSA algorithms.

Conclusion:

The introduction was about how encryption is the principal means to guarantee security of sensitive information. The next section was about related works done in the field of encryption especially focusing on comparing different algorithms.

1. The first was on the performance of different security algorithms on a cloud network and also on a single processor for different input sizes. It was aimed on finding quantitative terms like speed-up ratio.

Speed up – ration = mean processing time on single Processor / mean processing time on cloud.

The results reported in this paper conclude that the algorithms implemented on cloud environment (i.e., Google App) are more efficient than using them on single system. For

both uniprocessor (local) as well as cloud (Appengine) environment, RSA is the most time consuming and MD5 is the least.

2. The second was comparative analysis of three algorithms; RSA, DES and AES while considering certain parameters such as computation time, memory usage and output byte.

And it concluded that DES algorithm consumes least encryption time and AES algorithm has least memory usage while encryption time difference is very minor in case of AES and DES algorithm. RSA consume longest encryption time and memory usage is also very high but output byte is least in case of RSA algorithm.

There are various research works that are mentioned. One of them also mentioned about Avalanche Effect. Avalanche effect is the property of any encryption algorithm in which a small change in either the key or the plaintext should produce a significant change in the cipher text.

Avalanche Effect = number of flipped bits in ciphered text/ number of bits in ciphered text.

This study related to avalanche effect mentioned that Avalanche effect is very high for AES as compared to DES whereas memory requirement and simulation time for DES is greater than that of AES.

The detailed description about generation, modification and transportation ok keys have been mentioned. Finally, the conclusion of the document was presenting that the asymmetric algorithms are slower than that of Symmetric algorithms and RSA is the least secure algorithm when compared to DES, 3DES and AES.

Table 1. Comparision of RSA, DES, 3DES and AES

Factors	RSA	DES	3DES	AES
Created By	Ron Rivest, Adi Shamir, and Leonard Adleman In 1978	IBM in 1975	IBM IN 1978	Vincent Rijmen, Joan Daemen in 2001
Key Length	Depends on number of bits in the modulus n where n=p*q	56 bits	168 bits (k1, k2 and k3) 112 bits (k1 and k2)	128, 192, or 256 bits
Round(s)	1	16	48	10 - 128 bit key,12 - 192 bit key,14 - 256 bit key
Block Size	Variable	64 bits	64 bits	128 bits
Cipher Type	Asymmetric Block Cipher	Symmetric Block Cipher	Symmetric Block Cipher	Symmetric Block Cipher
Speed	Slowest	Slow	Very Slow	Fast
Security	Least Secure	Not Secure Enough	Adequate Security	Excellent Security

Paper 5:

Authors: Diaa Salama Abd Elminaam , Hatem Mohamed Abdual Kader, and Mohiy Mohamed Hadhoud

Title: Evaluating the Performance of Symmetric Encryption Algorithms

Overview: In this paper, the author presented the analysis of various symmetric encryption algorithms with respect to many factors such as different packet size, different formats such as audio files and video files and their effects on power consumption and throughput.

Conclusion:

The algorithms taken for consideration were AES (Rijndael), DES, 3DES, RC2, Blowfish, and RC6. The performance measures were conducted in terms of energy, changing data types such as text or document, audio data, video data, power consumption, changing packet size and changing key size.

Many conclusions which were deducted from previous related work are also stated. In the experimental design the authors stated that the comparison all the experiments is done in two different coding bases, hexadecimal base encoding and base 64 encoding.

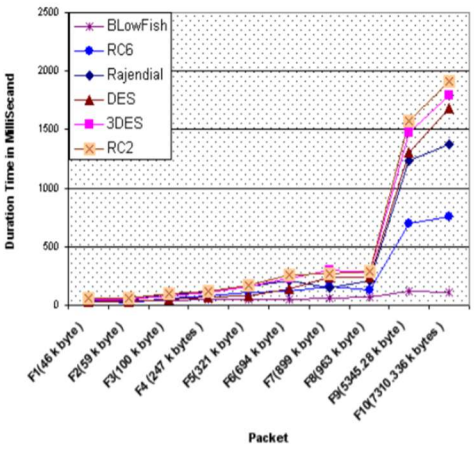


Figure 2: Time consumption of encryption algorithm (base 64 encoding)

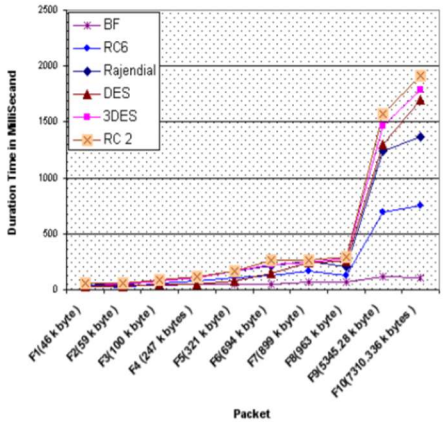


Figure 3: Time consumption of encryption algorithm (Hexadecimal encoding)

The first study was on the effect of changing packet size at power consumption during throughput for each selected cryptography algorithm.

The encryption and decryption results are presented as follows.

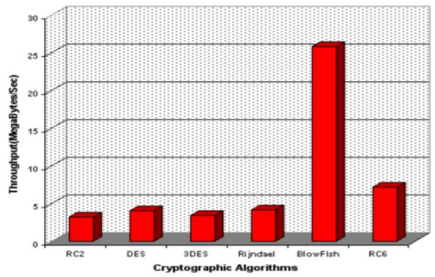


Figure 4: Throughput of each encryption algorithm (Megabyte/Sec)

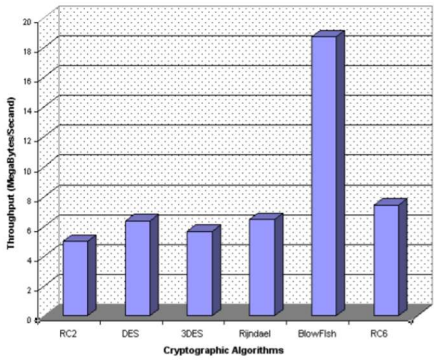


Figure 5: Throughput of each decryption algorithm (Megabyte/Sec)

In similar manner the results of encryption of different audio files of different sizes, decryption of different audio files and as well as video files were inferred. Finally, the effect of changing key size of AES, and RC6 on power consumption is discussed.

The conclusions that we were able to draw from this paper were

1. There is no significant difference when the results are displayed either in hexadecimal base encoding or in base 64 encoding.
2. In the case of changing packet size, it was concluded that Blowfish has better performance than other common encryption algorithms used, followed by RC6.
3. 3DES still has low performance compared to algorithm DES.
4. RC2, has disadvantage over all other algorithms in terms of time consumption.
5. AES has better performance than RC2, DES, and 3DES.
6. In the case of audio and video files we found the result as the same as in text and document.
7. Changing key size - it can be seen that higher key size leads to clear change in the battery and time consumption.

Paper 6:

Authors: E. Thambiraja, G. Ramesh, Dr. R. Umarani.

Title: A Survey on Various Most Common Encryption Techniques.

Overview: In this paper, the author explains about cryptography used in networking technology.

Conclusion: In the introduction of this paper, the author explains about the basic terms of cryptography and the different types of cryptography.

The purpose of the cryptography is also explained. In the previous work sub-section, the authors discussed about the most common algorithm implementation for both software and hardware approaches. The metrics taken into consideration are processing speed, throughput, power consumption, packet size and data types.

In the case of changing packet size with and without transmission, it was concluded that Blowfish has better performance than other common encryption algorithms used.

In the case of data type such as audio and video files the result as the same as in text and document.

In the case of image instead of text, it was found that RC2, RC6 and Blowfish has disadvantage over other algorithms in terms of time consumption.

Many results like there are presented in quite a number.

Finally, it was concluded in this paper that every encryption technique is useful in real-time encryption and each technique is unique in its own way and it was also stated that everyday new encryption technique is evolving hence fast and secure conventional encryption techniques will always work out.

Paper 7:

Authors: Rajdeep Bhanot and Rahul Hans

Title: A Review and Comparative Analysis of Various Encryption Algorithms

Conclusion:

Network security is highly based on cryptography. The strength of each encryption algorithm depends upon the key management, type of cryptography, number of keys, number of bits used in a key. The longer the key length and data length more will be the power consumption that will lead to more heat dissipation. All the keys are based upon mathematical properties and their strength decreases over time. The keys having a greater number of bits need more computation time which simply indicates that the system takes more time to encrypt the data.

Paper 8:

Authors: Performance Analysis of Cryptography Algorithms

Title: Rishabh Arora and Sandeep Sharma

Conclusion:

It is recommended to replace the algorithms that consume more energy, modification in the algorithms and to implement new design of algorithms. In Encrypting and Decrypting. EXE files, Blowfish has outstanding performance than other algorithms. In Encrypting. DOC files, AES is better to other algorithms and in Decrypting .DOC files, Blowfish is exceptional. AES and DESX both have the same performance in encrypting. WMV files, Blowfish is superior to DESX and AES. AES has the advantage over DESX in terms of throughput whereas, in Decrypting WMV files, Blowfish is better to other algorithms. In Encrypting and Decrypting AVI files, Blowfish is better to other algorithms. AES has the advantage over DESX in terms of throughput.

Paper 9:

Authors: Nishtha Mathura and Rajesh Bansode

Title: AES Based Text Encryption Using 12 Rounds with Dynamic Key Selection

Conclusion:

AES is a symmetrical encryption algorithm that uses a series of table lookups to increase its efficiency of performance. The cache hits and misses are common during the encryption process which causes various look-up times. The Cache Timing Attack correlates the timing details for encryption using a known key and also with an unknown key to infer the unknown key. To increase the efficiency of the encryption of data, AES will be implemented having the key size of 192 bit and with 12 rounds.

Paper 10:

Authors: D. Asir Antontony Gnana Singh and R.Priyadharshini

Title: Performance Analysis of Data Encryption Algorithms for Secure Data Transmission

Conclusion:

Selection of the suitable data encryption algorithm is based normally on its key length, data size and its performance. AES algorithm is regarded to be highly compromising data encryption standard for the data present in the network. Adding a higher block size to the system we can be able to increase the security and the integrity of the system.

3. Modules and Descriptions

The important modules present in this project are

1. Encryption module

Encryption module is used to present all the encryption techniques that are available in the application. The encryption techniques avalaible are again presented as sub-modules.

The available techniques are :

1. Playfair Cipher
2. Triple DES
3. Ceaser Ciper
4. Vigenre Ciper
5. AES

2. Decryption module

Decryption module is used to present all the encryption techniques that are available in the application. The decryption techniques avalaible are again presented as sub-modules.

The available techniques are :

1. Playfair Cipher
2. Triple DES
3. Ceaser Cipher
4. Vigenre Cipher
5. AES

3. Hashing module

Hashing module is used to present all the encryption techniques that are available in the application. The Hashing techniques avalaible are again presented as sub-modules.

The available hashing modules are:

1. SHA- 256
2. SHA512
3. MD5

4. Implementation

Description: AES – Encryption

```
public class AES {
    public String AESencrypt ( byte[] key, byte[] clear)
throws Exception {

        MessageDigest md =
MessageDigest.getInstance("md5");
        byte[] digestOfPassword = md.digest(key);

        SecretKeySpec skeySpec = new
SecretKeySpec(digestOfPassword, "AES");
        Cipher cipher =
Cipher.getInstance("AES/ECB/PKCS7Padding");
        cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
        byte[] encrypted = cipher.doFinal(clear);
        return Base64.encodeToString(encrypted,
Base64.DEFAULT);

    }
    public String AESdecrypt (String key,byte[] encrypted)
throws Exception {
        MessageDigest md =
MessageDigest.getInstance("md5");
        byte[] digestOfPassword =
md.digest(key.getBytes("UTF-16LE"));

        SecretKeySpec skeySpec = new
SecretKeySpec(digestOfPassword, "AES");
        Cipher cipher =
Cipher.getInstance("AES/ECB/PKCS7Padding");
        cipher.init(Cipher.DECRYPT_MODE, skeySpec);
        byte[] decrypted = cipher.doFinal(encrypted);
        return new String(decrypted, "UTF-16LE");
    }
}
```

Description: Ceaser-Encryption

```
public String caesarcipherEnc (String message,int key){
        String encryptedMessage = "";
        int n = 1;
        for (int i = 0; i < message.length(); i++) {
            n = 1;
            ch = message.charAt(i);
            for (int j = 0; j < NumbTest.length; j++)
            {
                if (ch == NumbTest[j])
                {

                    if((char)key+ch>'9')
                        break;
                    ch = (char) (ch + key);
                    encryptedMessage += ch;
                    n = 0;
                    break;
                }
            }
            if (n == 0)
            {
                continue;
            } else
                if (ch >= 'a' && ch <= 'z')
                {
                ch = (char) (ch + key);

                if (ch > 'z') {
                    ch = (char) (ch - 'z' + 'a' - 1);
                }

                encryptedMessage += ch;
            } else if (ch >= 'A' && ch <= 'Z') {
                ch = (char) (ch + key);

                if (ch > 'Z') {
                    ch = (char) (ch - 'Z' + 'A' - 1);
                }

                encryptedMessage += ch;
            } else
                encryptedMessage += ch;
        }
        return encryptedMessage;
    }
```

Description: TDES_Encryption

```
public class DES {
    public String encrypt ( byte[] key, byte[] clear)
throws Exception {

        MessageDigest md =
MessageDigest.getInstance("md5");
        byte[] digestOfPassword = md.digest(key);

        SecretKeySpec skeySpec = new
SecretKeySpec(digestOfPassword, "DESede");
        Cipher cipher =
Cipher.getInstance("DESede/CBC/PKCS5Padding");
        cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
        byte[] encrypted = cipher.doFinal(clear);
        return Base64.encodeToString(encrypted,
Base64.DEFAULT);

    }
public String decrypt (String key,byte[] encrypted) throws
Exception {

        MessageDigest md =
MessageDigest.getInstance("md5");

        byte[] digestOfPassword =
md.digest(key.getBytes("UTF-16LE"));



        SecretKeySpec skeySpec = new
SecretKeySpec(digestOfPassword, "DESede");

        Cipher cipher =
Cipher.getInstance("DESede/CBC/PKCS5Padding");

        cipher.init(Cipher.DECRYPT_MODE, skeySpec);

        byte[] decrypted = cipher.doFinal(encrypted);

        return new String(decrypted, "UTF-16LE");

    }
```

```
Description: Vigenère Encryption
public class Vigenere {
    public String Vigenereencrypt (String text, String key)
    {

        String res = "";
        text = text.toUpperCase();
        key = key.toUpperCase();
        for (int i = 0, j = 0; i < text.length(); i++) {
            char c = text.charAt(i);
            if (c < 'A' || c > 'Z') continue;
            res += (char) ((c + key.charAt(j) - 2 * 'A') %
26 + 'A');
            j = ++j % key.length();
        }
        return res;

    }
    public String Vigeneredecrypt (String text, String key)
    {
        String res = "";
        text = text.toUpperCase();
        key = key.toUpperCase();
        for (int i = 0, j = 0; i < text.length(); i++) {
            char c = text.charAt(i);
            if (c < 'A' || c > 'Z') continue;
```

```
            res += (char) ((c - key.charAt(j) + 26) % 26 +
'A');
            j = ++j % key.length();
        }

        return res;

    }
}
```

Description: Playfair Encryption

```
public class Basic {
        String allChar = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

        boolean indexOfChar(char c) {
            for (int i = 0; i < allChar.length(); i++) {
                if (allChar.charAt(i) == c)
                    return true;
            }
            return false;
        }
    }

    Basic b = new Basic();
    char keyMatrix[][] = new char[5][5];

    boolean repeat(char c) {
        if (!b.indexOfChar(c)) {
            return true;
        }
        for (int i = 0; i < keyMatrix.length; i++) {
            for (int j = 0; j < keyMatrix[i].length; j++) {
                if (keyMatrix[i][j] == c || c == 'J')
                    return true;
            }
```

```
    }
    return false;
}

public void insertKey(String key) {
    key = key.toUpperCase();
    key = key.replaceAll("J", "I");
    key = key.replaceAll(" ", "");
    int a = 0, b = 0;

    for (int k = 0; k < key.length(); k++) {
        if (!repeat(key.charAt(k))) {
            keyMatrix[a][b++] = key.charAt(k);
            if (b > 4) {
                b = 0;
                a++;
            }
        }
    }

    char p = 'A';

    while (a < 5) {
        while (b < 5) {
            if (!repeat(p)) {
```

```
            keyMatrix[a][b++] = p;

        }
        p++;
    }
    b = 0;
    a++;
}

for (int i = 0; i < 5; i++) {

    for (int j = 0; j < 5; j++) {

    }
}
t1=t1.concat("\n");
for(int i=0;i < 5;i++)
{
    for(int j=0;j < 5;j++)
        t1 = t1 +"   "+ keyMatrix[i][j];

         t1=t1.concat("\n");

    }
}
```

```
int rowPos(char c) {
    for (int i = 0; i < keyMatrix.length; i++) {
        for (int j = 0; j < keyMatrix[i].length; j++) {
            if (keyMatrix[i][j] == c)
                return i;
        }
    }
    return -1;
}

int columnPos(char c) {
    for (int i = 0; i < keyMatrix.length; i++) {
        for (int j = 0; j < keyMatrix[i].length; j++) {
            if (keyMatrix[i][j] == c)
                return j;
        }
    }
    return -1;
}

public String encryptChar(String plain) {
    plain = plain.toUpperCase();
```

```
char a = plain.charAt(0), b = plain.charAt(1);
String cipherChar = "";
int r1, c1, r2, c2;
r1 = rowPos(a);
c1 = columnPos(a);
r2 = rowPos(b);
c2 = columnPos(b);

if (c1 == c2) {
    ++r1;
    ++r2;
    if (r1 > 4)
        r1 = 0;

    if (r2 > 4)
        r2 = 0;
    cipherChar += keyMatrix[r1][c2];
    cipherChar += keyMatrix[r2][c1];
} else if (r1 == r2) {
    ++c1;
    ++c2;
    if (c1 > 4)
        c1 = 0;

    if (c2 > 4)
```

```
            c2 = 0;
        cipherChar += keyMatrix[r1][c1];
        cipherChar += keyMatrix[r2][c2];

    } else {
        cipherChar += keyMatrix[r1][c2];
        cipherChar += keyMatrix[r2][c1];
    }
    return cipherChar;
}


public String Encrypt(String plainText, String key) {
    insertKey(key);
    String cipherText = "";
    plainText = plainText.replaceAll("j", "i");
    plainText = plainText.replaceAll(" ", "");
    plainText = plainText.toUpperCase();
    int len = plainText.length();

    if (len / 2 != 0) {
        plainText += "X";
        ++len;
    }
```

```
        for (int i = 0; i < len - 1; i = i + 2) {
            cipherText +=
encryptChar(plainText.substring(i, i + 2));
            cipherText += " ";
        }
        return cipherText;
    }


    public String decryptChar(String cipher) {
        cipher = cipher.toUpperCase();
        char a = cipher.charAt(0), b = cipher.charAt(1);
        String plainChar = "";
        int r1, c1, r2, c2;
        r1 = rowPos(a);
        c1 = columnPos(a);
        r2 = rowPos(b);
        c2 = columnPos(b);

        if (c1 == c2) {
            --r1;
            --r2;
            if (r1 < 0)
                r1 = 4;

            if (r2 < 0)
```

```
            r2 = 4;
        plainChar += keyMatrix[r1][c2];
        plainChar += keyMatrix[r2][c1];
    } else if (r1 == r2) {
        --c1;
        --c2;
        if (c1 < 0)
            c1 = 4;

        if (c2 < 0)
            c2 = 4;
        plainChar += keyMatrix[r1][c1];
        plainChar += keyMatrix[r2][c2];

    } else {
        plainChar += keyMatrix[r1][c2];
        plainChar += keyMatrix[r2][c1];
    }
    return plainChar;
}

public String Decrypt(String cipherText, String key) {
    String plainText = "";
    cipherText = cipherText.replaceAll("j", "i");
```

```
        cipherText = cipherText.replaceAll(" ", "");
        cipherText = cipherText.toUpperCase();
        int len = cipherText.length();
        for (int i = 0; i < len - 1; i = i + 2) {
            plainText +=
decryptChar(cipherText.substring(i, i + 2));
            plainText += " ";

        }
        return plainText;
    }
```

Description: Hashing

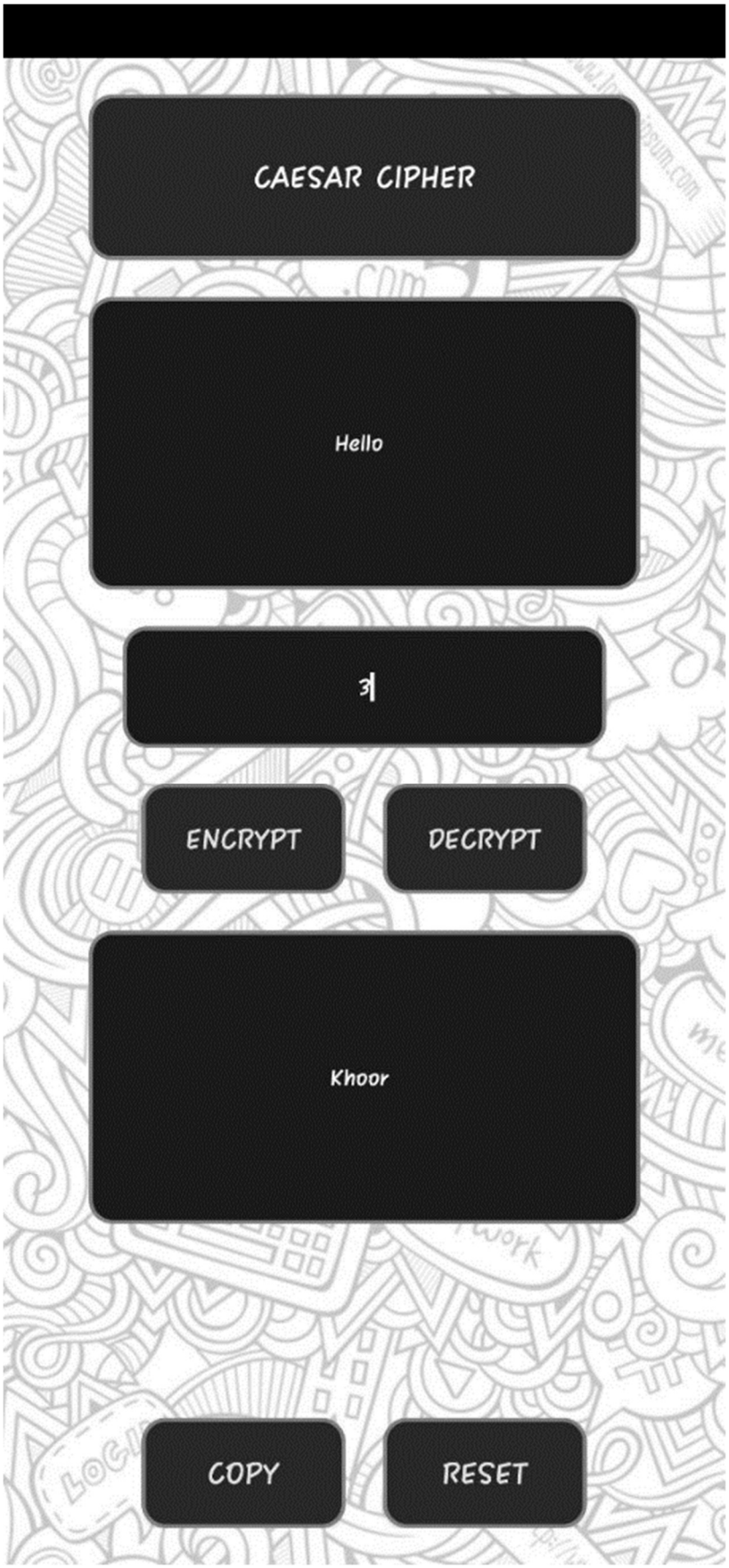
```
public  String hashText(String algo,String salt, String
plainText)
            throws NoSuchAlgorithmException {
        MessageDigest m = MessageDigest.getInstance(algo);
        m.reset();
        if (salt.length() != 0) {
            m.update(salt.getBytes());
        }
        m.update(plainText.getBytes());
        byte[] digest = m.digest();
        BigInteger bigInt = new BigInteger(1,digest);
        String hashtext = bigInt.toString(16);
        // Now we need to zero pad it if you actually want
the full 32 chars.
        while(hashtext.length() < 32 ){
            hashtext = "0"+hashtext;
        }
        return hashtext;
    }




    private  byte[] getRandomSalt() throws
NoSuchAlgorithmException, NoSuchProviderException
    {
```

```
        //Always use a SecureRandom generator
        SecureRandom sr =
SecureRandom.getInstance("SHA1PRNG", "SUN");
        //Create array for salt
        byte[] salt = new byte[16];
        //Get a random salt
        sr.nextBytes(salt);
        //return salt
        return salt;
    }



    public void reset(View view) {
        Textfield_Text.setText("");
        Textfield_salt.setText("");
        Answer.setText("");
        if(view!=null)
        Toast.makeText(view.getContext(), "All data has
been deleted", Toast.LENGTH_SHORT).show();
    }



    public void copyToClipboard(View view) {
            String copyText =
String.valueOf(Answer.getText());
```

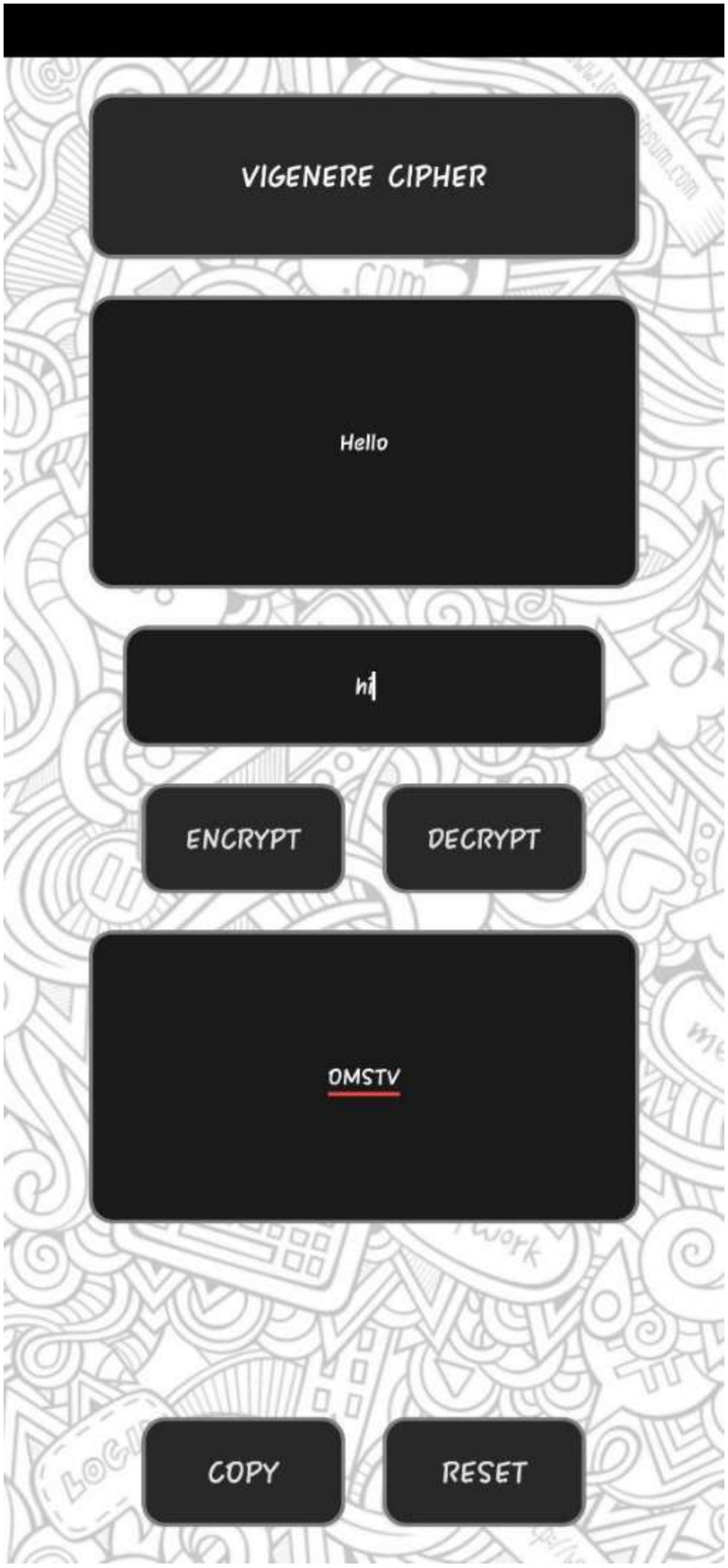
```
            if (Answer.length() == 0) {
                Toast.makeText(view.getContext(), "There is
no message to copy", Toast.LENGTH_SHORT).show();
                return;
            }

            int sdk = android.os.Build.VERSION.SDK_INT;
            if (sdk <
android.os.Build.VERSION_CODES.HONEYCOMB) {
                android.text.ClipboardManager clipboard =
(android.text.ClipboardManager)

view.getContext().getSystemService(Context.CLIPBOARD_SERVIC
E);
                clipboard.setText(copyText);
            } else {
                android.content.ClipboardManager clipboard
= (android.content.ClipboardManager)

view.getContext().getSystemService(Context.CLIPBOARD_SERVIC
E);
                android.content.ClipData clip =
android.content.ClipData
                        .newPlainText("Your message :",
copyText);
                clipboard.setPrimaryClip(clip);
            }
            Toast.makeText(view.getContext(),
                    "Your message has be copied",
Toast.LENGTH_SHORT).show();}
```

5. Results

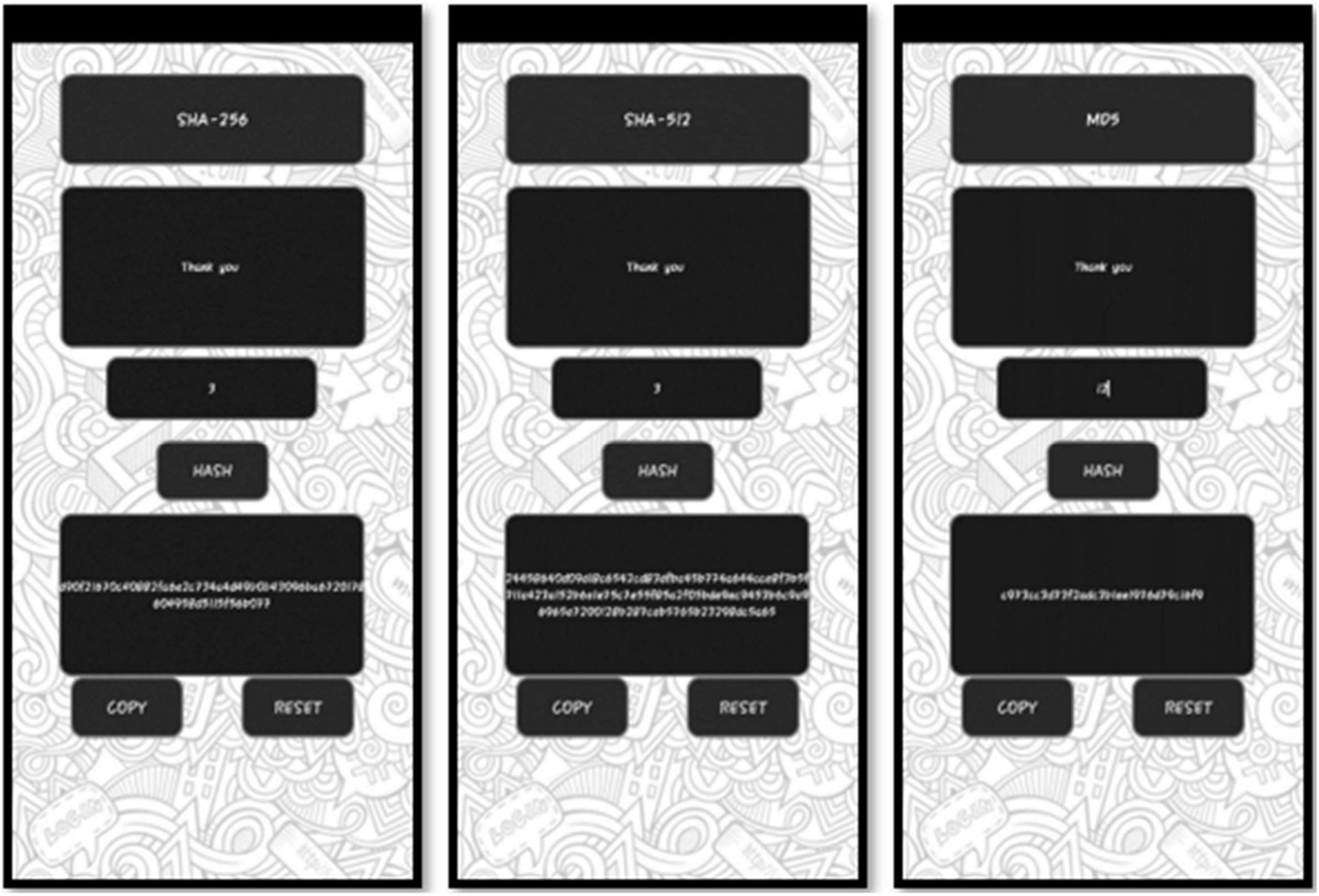

ADVANCED ENCRYPTION STANDARD
Hello
3
ENCRYPT
DECRYPT
Y8LCu5N/aSJgje6i6KPp/w==
COPY
RESET


CAESAR CIPHER
Hello
3
ENCRYPT
DECRYPT
Khoor
COPY
RESET


TRIPLE DATA ENCRYPTION STANDARD
Hello
3
ENCRYPT
DECRYPT
BwlleqQZwWSl74QZpimq2A==
COPY
RESET


VIGENERE CIPHER
Hello
hi
ENCRYPT
DECRYPT
OMSTV
COPY
RESET


PLAY FAIR
Hello
3
ENCRYPT
DECRYPT
A B C D E
F G H I K
L M N O P
Q R S T U
V W X Y Z
KC QQ NY
COPY
RESET


SHA-256
Thank you
3
HASH
COPY
RESET
SHA-512
Thank you
3
HASH
COPY
RESET
MD5
Thank you
3
HASH
COPY
RESET

6. Conclusion

In this project, we have dealt with the concepts of security and digital data encryption. The modern encryption techniques used prioritizes the privacy of the user by encrypting the data required.

The application has ensured that its users not only implement but also understand the workings of the algorithms. This project has aimed to explain complex concepts of encryption in an easy to use, simple application allowing its user a convenient gateway into personal security.

7. Team Members' Contribution

Team Leader (Reg.No. & Name) : Sirigiri Sri Sai Sharanya

Register Number	Name	Contribution / Role in this Project
19MIS0266	Sirigiri Sri Sai Sharanya	Hashing
19MIS0313	Shakthivel RK	Encryption

References

Paper 1:

https://d1wqtxts1xzle7.cloudfront.net/47096569/L017136269.pdf?1467952216=&response-content-disposition=inline%3B+filename%3DComparison_of_Various_Encryption_Algorit.pdf&Expires=1633604449&Signature=FHBgR1U01GVIfHtRY8GmzaSLQIWAl4tL5pDdoZwYLYa0H~HBjaAVY~P2rmFdG-eLx-RJpWJES3UYS49oXibgAS2zcO9J1SCs0zpYWbjl7OnESig6vpFuT8~Dz57NaI1-18jjsd1q1G3faek63NwToI-B-hq097EXcuI6YsBTOElsZcnYs5v~YMNPnIP5Q~SVQWdc0VyzMdmGQ~AlRzMRE52XB4RPil9g1oqCZC75IsB9NiC-FvTmOYWgq2tGtYMK0Ncgky~-d1UcyyxgSfRoZGOMVD7A6EjSlq9TJtPfDEr-CQXz4ciyt~mVF5Ge4ZnH2xIfK91oAyPrGYbv~7a0UA__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

Paper 2:

https://www.researchgate.net/profile/Kazeem-Adedeji/publication/303497916_A_New_Hybrid_Data_Encryption_and_Decryption_Technique_to_Enhance_Data_Security_in_Communication_Networks_Algorithm_Development/links/5745979c08aea45ee855c818/A-New-Hybrid-Data-Encryption-and-Decryption-Technique-to-Enhance-Data-Security-in-Communication-Networks-Algorithm-Development.pdf

Paper 3:

https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.639.7316&rep=rep1&type=pdf

Paper 4:

https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.403.5601&rep=rep1&type=pdf

Paper 5:

http://isrc.ccs.asia.edu.tw/ijns/contents/ijns-v10-n3/ijns-2010-v10-n3-p213-219.pdf

Paper 6:

https://d1wqtxts1xzle7.cloudfront.net/36508209/V2I600106-with-cover-page-v2.pdf?Expires=1633604548&Signature=Z9Qj0iheD5HMy~6XtjZO~1ndDVelLlf~FWnB0wPBLlFPmUs67mt7NOTkWvdAdI86UOG~AppXN6Pc8ayWCL~hIY4C53N3cg0IxWu3tJYPNdB2xAx~Qx6qrzare7dBeNPZxcHrpNWwVhkIT4THqSHQYAXAEzmbcWqN4OeHuurHU6C3ThuM2JVI-RDUhgz-8vqVIn3SopM49QevI~VecOuCpWOuTHWP1hcd-L3zbe9dxd7IodbucX1ExFfVu~1IWzo9FqPyVZi9PkPhnoJAKqEu3GohIIJFNQ7SEhz52y0yuVi9i-XpbacLh9hTqdd8rFkzpIewzTOVX7WQFTm9SE53sA__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

Paper 8:

https://www.researchgate.net/profile/Asir-Antony-Danasingh/publication/316789478_Performance_Analysis_of_Data_Encryption_Algorithms_for_Secure_Data_Transmission/links/5911cb7aaca27200fe391193/Performance-Analysis-of-Data-Encryption-Algorithms-for-Secure-Data-Transmission.pdf

Paper 7:

http://article.nadiapub.com/IJSIA/vol9_no4/27.pdf

Paper 9:

http://isrc.ccs.asia.edu.tw/ijns/contents/ijns-v10-n3/ijns-2010-v10-n3-p213-219.pdf

Paper 10:

https://www.researchgate.net/profile/Asir-Antony-Danasingh/publication/316789478_Performance_Analysis_of_Data_Encryption_Algorithms_for_Secure_Data_Transmission/links/5911cb7aaca27200fe391193/Performance-Analysis-of-Data-Encryption-Algorithms-for-Secure-Data-Transmission.pdf

www.google.com