VIT®

Vellore Institute of Technology

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

School of Information Technology& Engineering

M.Tech Software Engineering

SWE3002 – Information & System Security

Slot – F1+TF1

TITLE

COMPARISION OF AES ,DES, AND RSA USING TEXT ENCRYPTION

REVIEW 3

7-11-2019

DONE BY

K.KARTHIKEYAN - 16MIS0102

P.GOVINDAN - 16MIS0465

MOHAN V - 16MIS0522

Faculty Incharge:

Prof JEYANTHI N

Abstract

We are going to do compare encryption algorithims like AES,DES,and RSA for that we are using text encryption and decryption using that algorithims so We are going to do compare encryption algorithims like AES,DES,and RSA for that we are using text encryption and decryption using that algorithims so Encryption is the process of encoding information or data in order to prevent unauthorized access. These days we need to secure the information that is stored in our computer or is transmitted via internet against attacks. There are different types of cryptographic methods that can be used. Basically, the selecting cryptographic method depends on the application demands such as the response time, bandwidth, confidentiality and integrity. However, each of cryptographic algorithms has its own weak and strong points. In this paper, we will present the result of the implementation and analysis that applied on several cryptographic algorithms such as DES AES, RSA . Also, we will show the comparisons between the previous cryptographic techniques in terms of performances, weaknesses and strengths. Encryption is the process of encoding information or data in order to prevent unauthorized access.

Introduction

In recent years, many applications based on internet are developed such as on-line shopping, internet banking and electronic bill payment etc. Such transactions, over wire or wireless public networks demand end-to-end secure connections, should be private, to ensure data authentication, accountability and privacy, integrity and availability, also known as CIA triad

Algorithms

RSA has been used in most digital data, information and telephone security applications. The RSA has it's advantages of being a reliable and safe system but it also has the disadvantage of being very slow in data calculating. For this reason it is used in hybrid cryptographic systems that simultaneously use symmetric algorithms (AES) for the communication and data encryption phase and public key algorithms (RSA) for the safe delivery of the symmetric key (or session key) that is necessary for encrypting and decrypting the message. There are different levels of encryption in telephone cryptography.

Data Encryption Standard (DES)

It uses block cipher. It encrypts the data in block size of 64 bits each. Same algorithm and key encryption and decryption. Key is 56 bits of 8, 16,24,32,40,48,56,64 are discarded. two fundamental attributes of cryptography Diffusion (Substitution) and Confusion (Permutation) rounds. In each round key and data bits are shifted, permuted, XORed and sent through, 8 sround 64 bit plaintext is handed to initial permutation(IP).Then IP generates two halves left plaintext(LPT)and right plaintext(RPT).Each LPT and

RPT goes through 16 rounds. At the last LPT and RPT are rejoined. Decryption is same process perform rounds in reverse order.

Advanced encrytion Standard [AES]:

AES is based on a design principle known as a substitution–permutation network, and is efficient in both software and hardware.[9] Unlike its predecessor DES, AES does not use a Feistel network. AES is a variant of Rijndael which has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits. By contrast, Rijndael per se is specified with block and key sizes that may be any multiple of 32 bits, with a minimum of 128 and a maximum of 256 bits. AES operates on a 4 × 4 column-major order array of bytes, termed the state. Most AES calculations are done in a particular finite field. The key size used for an AES cipher specifies the number of transformation rounds that convert the input, called the plaintext, into the final output, called the ciphertext. The number of rounds are as follows: ϖ 10 rounds for 128-bit keys. ϖ 12 rounds for 192-bit keys. ϖ 14 rounds for 256-bit keys. Each round consists of several processing steps, including one that depends on the encryption key itself. A set of reverse rounds are applied to transform cipher text back into the original plaintext using the same encryption key.

Literature Survey

RSA ALGORITHIM:

Title	Advantages	Disadvantages
1.A STUDY AND PERFORMANCE ANALYSIS OF RSA ALGORITHM	The RSA provides highest security to the business application. Moreover, this scheme can be used for encryption of long messages without employing the hybrid and symmetric encryption.	algorithm requires more time for encryption decryption
2.RSA Public-key Cryptosystem	In hardware, RSA can be found in secure telephones, on Ethernet network cards, and on smart cards. In addition, RSA is incorporated into all of the ma jor protocols for secure Internet communications.	The security of RSA is related to the assumption that factoring is dicult. An easy factoring method or some other feasible attack would break RSA.
3. Image Cryptography Using RSA Algorithm in Network Security	Here, the image encryption algorithm proposed efficient and highly securable with high level of security and less computation.	All the techniques are in a real-time image encryption could only find a low level of security.

4. RSA authentication in Internet of Things	Is it possible to use the RSA algorithm for authentication when using a device with limited computational capabilities	This paper are not able to conclude that a disparity exists between the results and the expectation in the industry due to a limited number of respondents. There is a disparity in the actual performance measured and the answers given in the survey. This could be due to di↵erent interpretations of the scenario provided in the survey
5. STUDY ON IMPROVEMENT IN RSA ALGORITHM AND ITS IMPLEMENTATION	RSA variants which can improve the performance of the decryption and signature .The performance of the RSA algorithm will be improved by reducing the modulus and private exponents. At the same time it will get higher security and higher speedup based on current multicore devices.	In this cryptosystem, if the private key is lost then all received message cannot be decrypted but security wise , it's great.
6. A study of public key 'e' in RSA algorithm	the modified RSA algorithm has slightly increased in time complexity in the encryption process. In the decryption process, the Modified RSA Algorithm is almost the same of time consumed compare to the original RSA Algorithm.	Still this paper using The implementation of this study uses the small list of public key 'e'
7. Improved RSA cryptosystem based on the study of number theory and public key cryptosystems	Though it is hard to find out the factors of n and get p and q, two large prime numbers, therefore brute force attack is more difficult in our proposed algorithm as the encryption keys are sent separately , not at once.	The proposed RSA is used for system that needs high security but with less speed.

8. Image Encryption/Decryption Using RSA Algorithm	With the implementation of RSA algorithm using 2 bit rotation, we reach a conclusion that for better security of any text or image.	Low computational speed
9. Data Encryption and Decryption Using RSA Algorithm in a Network Environment	An eavesdropper that breaks into the message will return a meaningless message. Obviously encryption and decryption is one of the best ways of hiding the meanings of a message from intruders in a network environment.	An incorrect private key will still decrypt the encrypted message but to a form different from the original message.
10. A New Approach for Image Encryption in the Modified RSA Cryptosystem Using MATLAB	Therefore, this approach produces a new and secure strategy to encrypt any gray or color image using the modified RSA cryptosystem programmed by MATLAB and gives us the confidence to transmit these images thru any network even if it was not very secure.	any attacks in the transmission of images in all agencies

AES ALGORITHIM

Title	**Advantages**	**Disadvantages**
1. Image encryption using AES alogorithm.	The image can only be viewed by the receiver as the image is encrypted using AES and the key is only known to the sender and receiver.	The file size to be transmitted becomes large since it contains encrypted data.

2. Implementation of secure payment system using AES algorithm.	Most secured sysem built ever.	Implementation cost is higher and scaling is difficult.
3. FPGA Implementation of AES Algorithm using Spartan6.	The encryption module is implemented with 128 bits key length and the Key Schedule module run with 10 rounds. For each round the Key Schedule module will generate a new round key.	Spartan6 FPGA Project Kit doesn't support both of the modules in the same time, so the encryption.bit file from encryption respectively decryption.bit file from decryption should be loaded sequentially because of fact that each file occupies ~60% from FPGA resources.
4. Symmetric Key Encryption by AES.	The AES algorithm is very efficient, which can encrypt 1M of data in just a few milliseconds on my computer.	: Yes, Java has a built-in support for AES by its "javax.crypto.Cipher" class. In Microsoft .NET, it has a similar support. Thus it can be easily encrypted and decrypted.
5. Implementation of AES ENCRYPTION in HDL MATLAB Project.	Cryptography algorithms available, Rijndael algorithm was chosen as the Advanced Encryption Standard (AES) because of its ease of implementation and high performance.	Algorithm is dependent on the key size.
6. Review on Image Encryption and Decryption using AES Algorithm	The usage of 256 bit cipher key to achieve the high security, because 256 bit cipher key is difficult to break	Low speed

7. An image encryption and decryption using AES algorithm	It has shown that the algorithms have extremely large security key space and can withstand most common attacks such as the brute force attack, cipher attacks and plaintext attacks.	The biggest problem with AES symmetric key encryption is that you need to have a way to get the key to the party with whom you are sharing data
8. SECURE IMAGE PROCESSING USING AES ALGORITHM	In this paper detailed study of AES algorithm has been presented also provides compassion of different cryptographic algorithms.	it is essential to protect the confidential image data from unauthorized access, Image security has become a critical issue here.
9. A Review on Various Image Encryption Technique using AES and Random RGB Substitution	the use of AES encryption algorithm as fast in nature for encryption and decryption along with some pixel shuffling technique so this hybrid combination make our system more secure to transmit out secret information over a network or computer network this system will result the more strength and less information loss after encryption and decryption process.	Recommended some existing security system
10. IMAGE ENCRYPTION AND DECRYPTION USING AES ALGORITHM	The proposed algorithim offers, encryption quality. Even AES-128 offers a sufficiently large number of possible keys, making an exhaustive search impractical for many decades	Many AES attacks are based upon the simplicity of this key schedule and it is possible that one day an attack will be created to break AES encryption.

DES ALGORITHM:-

TITLE	ADVANTAGES	DISADVANTAGES
1).Enchancing the security of des algorithm using transposition cryptography techniques.	1).The main advantage of using enhanced DES algorithm the security is very tight and approximately impossible to crack and break the enchanced DES algorithm 2).The brute force attack is weak against the enhanced DES because the intruder required breaking the DES and simpler columnar approach both.He required extra time to hack the algorithm.	1).The only drawback is extra computation is needed.But this is not so crucial because our aim is to provide tighter security. 2).The second disadvantage is to generate the random number of columns and to send it on the network.This is also not so hard at today's time because the speed of machines is very high.
2)Enhancement of DES algorithm with multi state logic	1).The advantage of using enhancement of DES algorithm with multi state logic is they got a good avalanche effect and also they solve cryptanalysis attack while comparing proposed enhanced multi state DES algorithm with original des algorithm. 2).Here increasing the number of states for presenting the information,will increasing the number of combination of information so it will be hard for the intruder to detect the actual information.	1).The main drawback is we cannot a expect a 100 % security for the data information. 2).The volume of information exchanged by electronic means such as internet,wireless phones,fax,etc. is increasing very faster.It is very serious that information through internet,an enormous computer network,is vulnerable to hackers and that privacy of wireless phones without security is invaded.so,at this perspective they should improve the cryptosystems to provide greater security.
3).New approach of data encryption standard algorithm.	1).By adding additional key,modified S-Box design,modifies function implementation and replacing the old XOR by a new operation as proposed by these it gives a more robustness to DES algorithm and make it stronger against any kind of intruding. 2).DES encryption with two keys instead of one key already will	1).At present 21st century DES algorithm is considered to be insecure for some applications like banking system. 2).There are also some analytical results which demonstrates that theoretical weakness in the cipher.

	increase the efficiency of cryptography.	
4).Image Steganography using Triple DES	1).The image can only viewed by the receiver as the image is encrypted by using triple DES and key known to sender and receiver 2).Since image encrypted using triple DES ,it is more secure than the DES.	1).The file size to be transmitted becomes large since it contains encrypted data. 2).Since the file size is huge it can be suspected to contain some critical information.
5).Design and implement dynamic key generation to enhance DES algorithm	1).The main advantage is due to the dynamic key generation unit secrecy of the key get increased. 2).By using proposed design we can achieve high speed and reduced logic complexity which gives enhanced DES algorithm. 3)Enhanced DES algorithm has broad application area in secure data communication and transmission.	1).As per the research done DES algorithm is weak due to its weak key generation.so,that the key generation can be configured.
6).A review paper on AES and DES cryptographic algorithms	1).Hardware implementation of DES are very fast. 2).Even now no real weakness has been found. 3).It needs less processing time. 4)It has less rounds of communication compared with AES .	1).DES is complex when compared to AES. 2)The most efficient attack is brute force.The 56 bit key size is the biggest defect. 3)In some times the DES is inefficient and insecure for some applications.
7)Expanded 128-bit Data Encryption Standard	1).Because of expanded 128-bit data encryption standard,brute force attacks should take approximately twice as long,considering the additional steps that would be needed to decrypt the modified block cipher . 2).Due to this,it is satisfactory to say that the original aim,which was to improve the defense of DES against brute force attacks,of which it was weak against,has been met.	1).Weak keys:the key that is selected on the rounds are a problem.During splitting of keys to two half and swapping them might throw up the same result if they have continuous 1's and 0's.Thins ends up in using the same key through out the 16-cycles. 2).There can be same output from the s-boxes on different inputs on permutation.These are called semi weak keys.

8).Design and simulation DES algorithm of encryption for Information security	1).DES uses 16-bits keys generated from a master 56-master keys(64 bits if we consider also parity bits) 2).Reduce cipher complexity. 3).Weak keys can be avoided at key generation. 4)Algorithms consumes least encryption time	1).DES has 4 weak keys. 2).Weak keys:keys make the subkeys to be generated in more than one round. 3).DES consumes maximum encryption time. 4).From this research papers,they concluded that the algorithms are far bette than DES algorithms.
9).Data Encryption standard algorithm for secure data transmission	1).DES algorithm is a 64 bit block cipher with keys of 56 bits. 2)DES technique is used for secure data transmission while maintaining the authenticity and integrity of the message. 3).In this paper,encryption and decryption of data is done by using data encryption standard algorithm.	1).DES is considered to be an insecure technique of encryption for some applications like banking system. 2)There are some analytical results which demonstrate theoretical weakness in the cipher. 3)So it becomes very important to augment this algorithm by adding new level of security to it.
10).A modified simplified data encryption standard algorithm	1)They done a modification in simplified Des algorithm to secure data. 2).They use random number generation keys to make more secure the S-DES algorithm 3).They modified S-DES key generation process through pseudo random number key generation (PRNKG).PRNKG use system current time as a seed and perform function with pass pharse to generate 16bits of keys. 4)This 16bits of key is very difficult to guess by attackers.	1)Data is not secure because of attackers and intruders. 2).Simplified DES is insecure algorithm because of its 8 bits static keys.

RSA Methodology

(1) Select p and q both prime number, p is not equal to q.

(2) Calculate n = p x q.

(3) Calculate ø (n) = (p -1) x (q-1).

(4) Select integer e whose gcd (Ø (n), e) = 1; 1 < e <Ø (n).

(5) Calculate private key d = e^{-1} (mod Ø (n)).

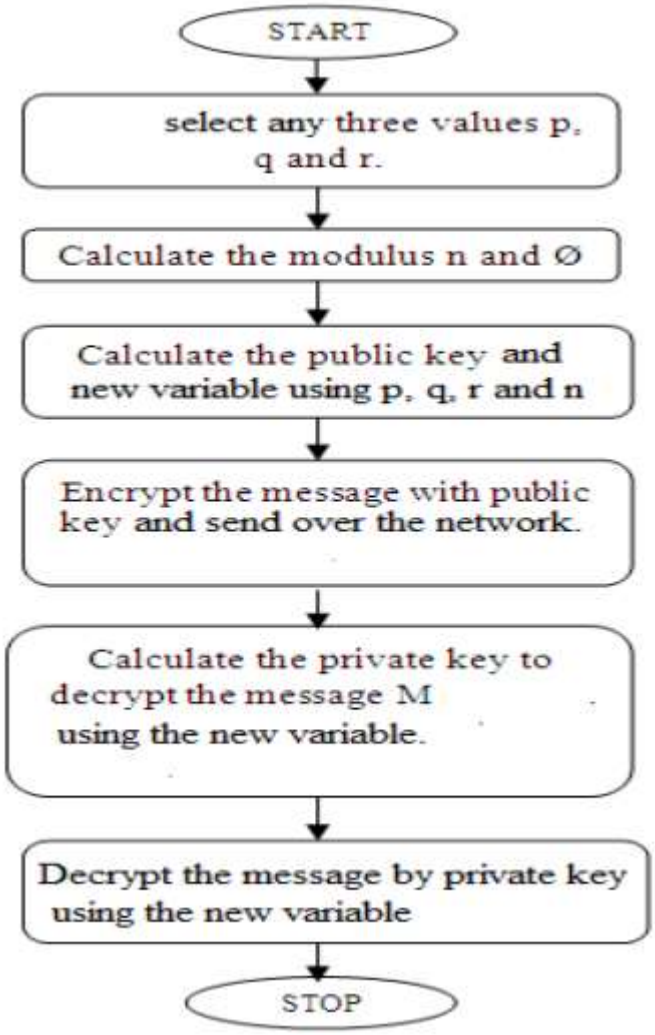
(6) Public key PU = {e, n}.

(7) Private Key PR = {d, n}.

B. *Encryption Procedure*

Plaintext- Message (M)
Cipher text- C = M^e mod n.

Our proposed System


START
select any three values p, q and r.
Calculate the modulus n and Ø
Calculate the public key and new variable using p, q, r and n
Encrypt the message with public key and send over the network.
Calculate the private key to decrypt the message M using the new variable.
Decrypt the message by private key using the new variable
STOP

Output

Our method with 3 prime

```
import java.math.BigInteger;
import java.util.Random;

public class RSAA
{
    private BigInteger p;//prime number 1
    private BigInteger q;//prime number 2
     private BigInteger my;//prime number 3



    private BigInteger N;
```

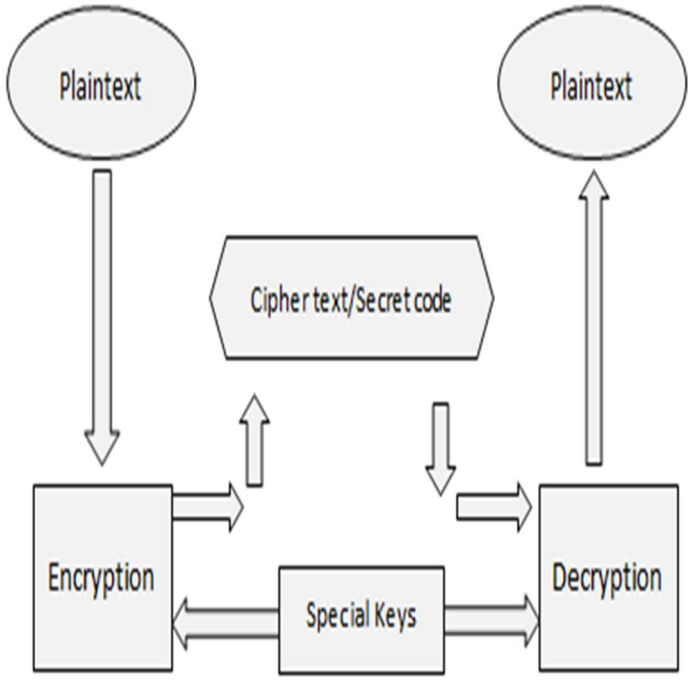
```
aa.RSAA
t - RSAA (run) ×
run:
Enter the plain text:
ninja
Encrypting String: ninja
String in Bytes: 110105110106697
Decrypting Bytes: 110105110106697
Decrypted String: ninja
BUILD SUCCESSFUL (total time: 5 seconds)
```

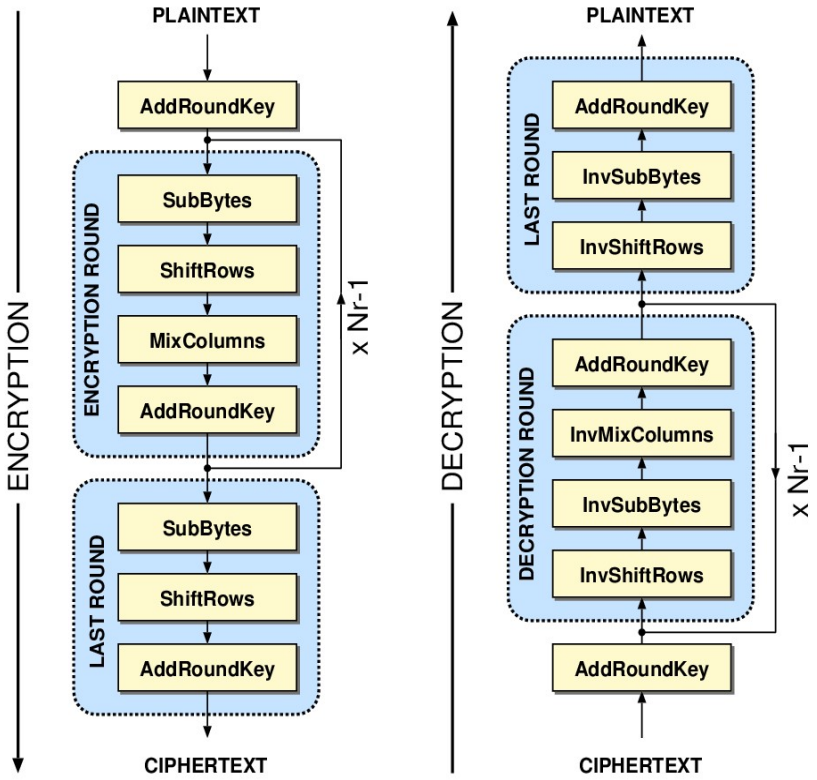
With 2 prime

```
import java.io.DataInputStream;
import java.io.IOException;
import java.math.BigInteger;
import java.util.Random;

public class RSA
{
    private BigInteger p;//prime number 1
    private BigInteger q;// prime numbe 2
```

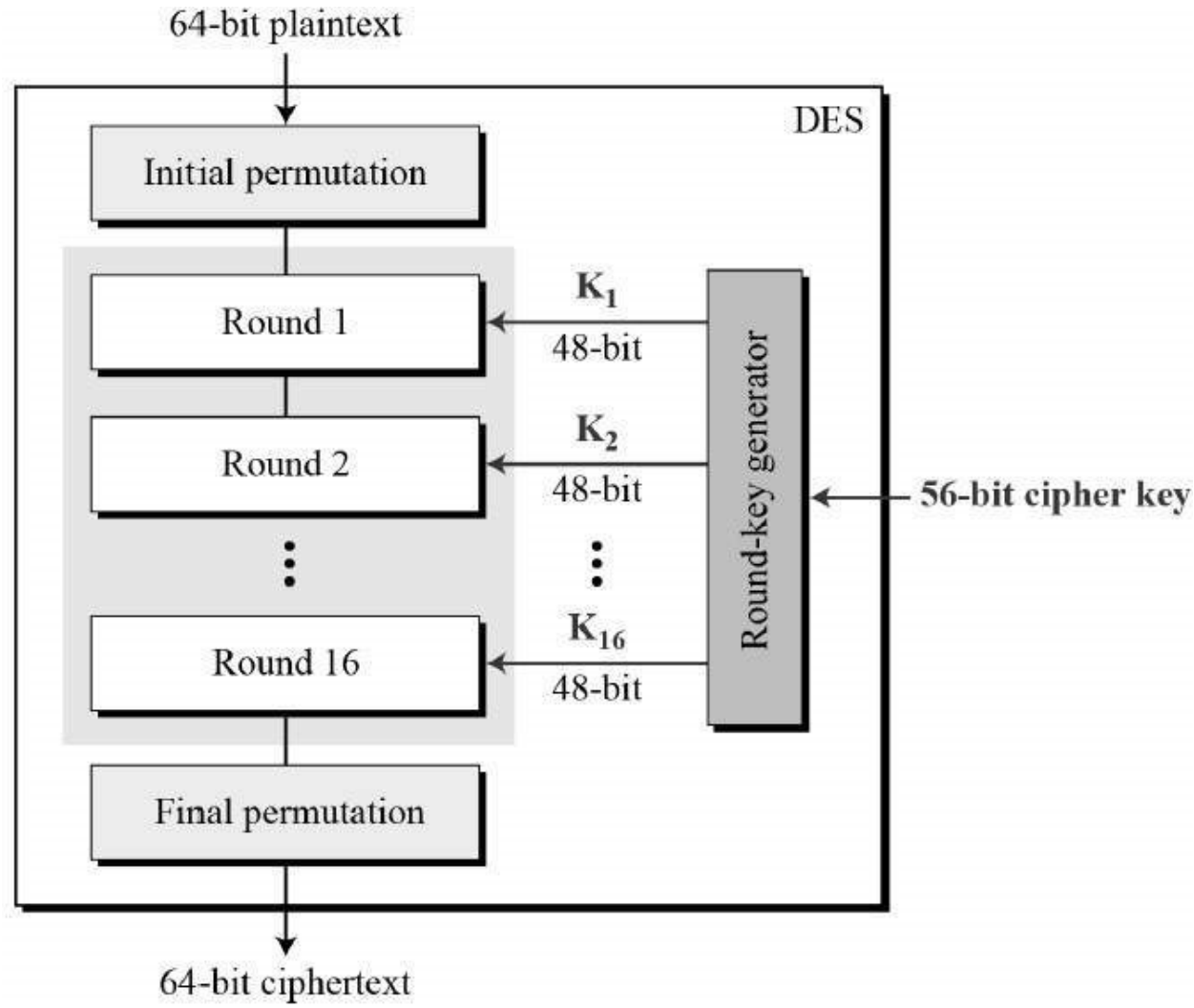
```
rsa.RSA
put - RSA (run) ×
run:
Enter the plain text:
ninja
Encrypting String: ninja
String in Bytes: 110105110106697
Decrypting Bytes: 110105110106697
Decrypted String: ninja
BUILD SUCCESSFUL (total time: 9 seconds)
```

AES Diagram





DES Algorithim



Implementation Code

Code for Modified RSA

```
import java.io.*;
import java.io.DataInputStream;
import java.io.IOException;
import java.math.BigInteger;
import java.util.Random;


public class RSAA
{
```

```
private BigInteger p;//prime number 1
private BigInteger q;// prime numbe 2
private BigInteger k;//prime numner 3



private BigInteger N;
        private BigInteger NN;
private BigInteger phi;
private BigInteger e;
private BigInteger d;
private int  bitlength = 1024;
private Random     r;

public RSAA()
{
   r = new Random();
   p = BigInteger.probablePrime(bitlength, r);
   q = BigInteger.probablePrime(bitlength, r);
        k = BigInteger.probablePrime(bitlength, r);
   NN = p.multiply(q);
                    N=NN.multiply(k);
   phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE)).multiply(k.subtract(BigInteger.ONE));
   e = BigInteger.probablePrime(bitlength / 2, r);
   while (phi.gcd(e).compareTo(BigInteger.ONE) > 0 && e.compareTo(phi) < 0)
   {
      e.add(BigInteger.ONE);
   }
   d = e.modInverse(phi);
}
```

```
public RSAA(BigInteger e, BigInteger d, BigInteger N)
{
   this.e = e;
   this.d = d;
   this.N = N;
}

@SuppressWarnings("deprecation")
public static void main(String[] args) throws IOException
{
   RSAA rsa = new RSAA();
   DataInputStream in = new DataInputStream(System.in);
   String teststring;
   System.out.println("Enter the plain text:");
   teststring = in.readLine();
   System.out.println("Encrypting String: " + teststring);
   System.out.println("String in Bytes: "
         + bytesToString(teststring.getBytes()));
   // encrypt
   byte[] encrypted = rsa.encrypt(teststring.getBytes());
   // decrypt
   byte[] decrypted = rsa.decrypt(encrypted);
   System.out.println("Decrypting Bytes: " + bytesToString(decrypted));
   System.out.println("Decrypted String: " + new String(decrypted));
}

private static String bytesToString(byte[] encrypted)
{
   String test = "";
   for (byte b : encrypted)
   {
```

```
        test += Byte.toString(b);
      }
      return test;
    }

    // Encrypt message
    public byte[] encrypt(byte[] message)
    {
      return (new BigInteger(message)).modPow(e, N).toByteArray();
    }

    // Decrypt message
    public byte[] decrypt(byte[] message)
    {
      return (new BigInteger(message)).modPow(d, N).toByteArray();
    }
}
```

Code for AES text encryption

```
import java.awt.Frame;
import java.awt.Toolkit;
import java.awt.datatransfer.StringSelection;
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Arrays;
import java.util.Base64;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
```

```
/**
 *
 * @author MH Habib
 */
public class EDcrypt extends javax.swing.JFrame {

    /**
     * Creates new form EDcrypt
     */
    private static SecretKeySpec secretKey;
    private static byte[] key;
    public static void setKey(String myKey)
    {
        MessageDigest sha = null;
        try {
            key = myKey.getBytes("UTF-8");
            sha = MessageDigest.getInstance("SHA-1");
            key = sha.digest(key);
            key = Arrays.copyOf(key, 16);
            secretKey = new SecretKeySpec(key, "AES");
        }
        catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
    }
    public EDcrypt() {
        initComponents();
    }
```

```
/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jScrollPane1 = new javax.swing.JScrollPane();
    text1 = new javax.swing.JTextArea();
    jScrollPane2 = new javax.swing.JScrollPane();
    text2 = new javax.swing.JTextArea();
    jScrollPane3 = new javax.swing.JScrollPane();
    text3 = new javax.swing.JTextArea();
    jScrollPane4 = new javax.swing.JScrollPane();
    text4 = new javax.swing.JTextArea();
    msg1 = new javax.swing.JTextField();
    msg2 = new javax.swing.JTextField();
    encrypt = new javax.swing.JButton();
    decrypt = new javax.swing.JButton();
    copyencrypt = new javax.swing.JButton();
    copydecrypt = new javax.swing.JButton();
    jLabel2 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    jLabel1 = new javax.swing.JLabel();
    jLabel4 = new javax.swing.JLabel();
    jLabel5 = new javax.swing.JLabel();
    jLabel6 = new javax.swing.JLabel();
    message1 = new javax.swing.JLabel();
```

```
message2 = new javax.swing.JLabel();
mainsection = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setTitle("Encryption and Decryption ");
setAlwaysOnTop(true);
setUndecorated(true);
setResizable(false);
getContentPane().setLayout(null);

text1.setColumns(20);
text1.setFont(new java.awt.Font("Dialog", 0, 14)); // NOI18N
text1.setRows(5);
text1.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(102, 102, 102), 2));
jScrollPane1.setViewportView(text1);

getContentPane().add(jScrollPane1);
jScrollPane1.setBounds(80, 80, 300, 120);

text2.setColumns(20);
text2.setFont(new java.awt.Font("Dialog", 0, 14)); // NOI18N
text2.setRows(5);
text2.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(102, 102, 102), 2));
jScrollPane2.setViewportView(text2);

getContentPane().add(jScrollPane2);
jScrollPane2.setBounds(80, 322, 300, 120);

text3.setColumns(20);
text3.setFont(new java.awt.Font("Dialog", 0, 14)); // NOI18N
text3.setRows(5);
```

```
text3.setToolTipText("");
text3.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(102, 102, 102), 2));
jScrollPane3.setViewportView(text3);

getContentPane().add(jScrollPane3);
jScrollPane3.setBounds(470, 80, 320, 120);

text4.setColumns(20);
text4.setFont(new java.awt.Font("Dialog", 0, 14)); // NOI18N
text4.setRows(5);
text4.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(102, 102, 102), 2));
jScrollPane4.setViewportView(text4);

getContentPane().add(jScrollPane4);
jScrollPane4.setBounds(470, 320, 320, 120);

msg1.setHorizontalAlignment(javax.swing.JTextField.CENTER);
msg1.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      msg1ActionPerformed(evt);
   }
});
getContentPane().add(msg1);
msg1.setBounds(200, 220, 180, 30);

msg2.setHorizontalAlignment(javax.swing.JTextField.CENTER);
getContentPane().add(msg2);
msg2.setBounds(595, 220, 190, 30);

encrypt.setBackground(new java.awt.Color(0, 51, 51));
encrypt.setFont(new java.awt.Font("Dialog", 1, 14)); // NOI18N
```

```
encrypt.setForeground(new java.awt.Color(255, 255, 255));
encrypt.setText("Encrypt");
encrypt.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      encryptActionPerformed(evt);
   }
});
getContentPane().add(encrypt);
encrypt.setBounds(80, 275, 90, 30);

decrypt.setBackground(new java.awt.Color(0, 51, 51));
decrypt.setFont(new java.awt.Font("Dialog", 1, 14)); // NOI18N
decrypt.setForeground(new java.awt.Color(255, 255, 255));
decrypt.setText("Decrypt");
decrypt.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      decryptActionPerformed(evt);
   }
});
getContentPane().add(decrypt);
decrypt.setBounds(470, 275, 90, 30);

copyencrypt.setBackground(new java.awt.Color(102, 0, 0));
copyencrypt.setFont(new java.awt.Font("Dialog", 1, 12)); // NOI18N
copyencrypt.setForeground(new java.awt.Color(255, 255, 255));
copyencrypt.setText("Copy Encryption");
copyencrypt.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      copyencryptActionPerformed(evt);
   }
});
```

```
getContentPane().add(copyencrypt);
copyencrypt.setBounds(240, 275, 140, 30);

copydecrypt.setBackground(new java.awt.Color(102, 0, 0));
copydecrypt.setFont(new java.awt.Font("Dialog", 1, 12)); // NOI18N
copydecrypt.setForeground(new java.awt.Color(255, 255, 255));
copydecrypt.setText("Copy Decryption");
copydecrypt.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      copydecryptActionPerformed(evt);
   }
});
getContentPane().add(copydecrypt);
copydecrypt.setBounds(633, 275, 150, 30);

jLabel2.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
jLabel2.addMouseListener(new java.awt.event.MouseAdapter() {
   public void mousePressed(java.awt.event.MouseEvent evt) {
      jLabel2MousePressed(evt);
   }
});
getContentPane().add(jLabel2);
jLabel2.setBounds(810, 5, 30, 20);

jLabel3.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
jLabel3.addMouseListener(new java.awt.event.MouseAdapter() {
   public void mousePressed(java.awt.event.MouseEvent evt) {
      jLabel3MousePressed(evt);
   }
});
getContentPane().add(jLabel3);
```

```
jLabel3.setBounds(780, 5, 30, 20);

jLabel1.setFont(new java.awt.Font("Dialog", 1, 14)); // NOI18N
jLabel1.setForeground(new java.awt.Color(204, 51, 0));
jLabel1.setText("Encryption Key");
jLabel1.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(102, 102, 102)));
getContentPane().add(jLabel1);
jLabel1.setBounds(80, 220, 110, 30);

jLabel4.setFont(new java.awt.Font("Dialog", 1, 14)); // NOI18N
jLabel4.setForeground(new java.awt.Color(204, 51, 0));
jLabel4.setText("Decryption Key");
jLabel4.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(102, 102, 102)));
getContentPane().add(jLabel4);
jLabel4.setBounds(470, 220, 110, 30);

jLabel5.setFont(new java.awt.Font("Dialog", 1, 18)); // NOI18N
jLabel5.setForeground(new java.awt.Color(0, 0, 51));
jLabel5.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel5.setText("Message to Decrypt");
getContentPane().add(jLabel5);
jLabel5.setBounds(470, 50, 300, 30);

jLabel6.setFont(new java.awt.Font("Dialog", 1, 18)); // NOI18N
jLabel6.setForeground(new java.awt.Color(0, 0, 51));
jLabel6.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel6.setText("Message to Encrypt ");
getContentPane().add(jLabel6);
jLabel6.setBounds(80, 50, 300, 30);

message1.setForeground(new java.awt.Color(204, 0, 0));
```

```
    message1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
    getContentPane().add(message1);
    message1.setBounds(80, 450, 300, 20);

    message2.setForeground(new java.awt.Color(204, 0, 0));
    message2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
    getContentPane().add(message2);
    message2.setBounds(470, 450, 320, 20);

    mainsection.setForeground(new java.awt.Color(153, 0, 0));
    mainsection.setText("AES");
    mainsection.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
    getContentPane().add(mainsection);
    mainsection.setBounds(10, -220, 850, 500);

    setSize(new java.awt.Dimension(850, 499));
    setLocationRelativeTo(null);
}// </editor-fold>

private void jLabel2MousePressed(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    System.exit(0);

}

private void jLabel3MousePressed(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    this.setState(Frame.ICONIFIED);
}

private void encryptActionPerformed(java.awt.event.ActionEvent evt) {
```

```
    // TODO add your handling code here:
    String strToEncrypt;
    String secret;
    try
    {
        strToEncrypt=text1.getText();
        secret=msg1.getText();
        setKey(secret);
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
        cipher.init(Cipher.ENCRYPT_MODE, secretKey);
        text2.setText(Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes("UTF-8"))));

    }
    catch (Exception e)
    {
        text2.setText("Please fill up the right secret key");

    }
}

private void copyencryptActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Toolkit.getDefaultToolkit().getSystemClipboard().setContents(new StringSelection(text2.getText()),null);
    message1.setText("Your encryption result is copied!");
    message2.setText("");
}

private void decryptActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String secret;
    String strToDecrypt;
```

```
        try
        {
            secret=msg2.getText();
            strToDecrypt=text3.getText();
            setKey(secret);
            Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5PADDING");
            cipher.init(Cipher.DECRYPT_MODE, secretKey);
            text4.setText(new String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt))));
            //return new String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));
        }
        catch (Exception e)
        {
            text4.setText("Please fill up the right secret key");
        }
    }

    private void copydecryptActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Toolkit.getDefaultToolkit().getSystemClipboard().setContents(new StringSelection(text4.getText()),null);
        message2.setText("Your decryption result is copied!");
        message1.setText("");
    }
```

Code for DES

```
import javax.swing.*;
import java.security.SecureRandom;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.util.Random ;
```

```
class DES {
byte[] skey = new byte[1000];
String skeyString;
static byte[] raw;
String inputMessage,encryptedData,decryptedMessage;
public DES() {
try {
generateSymmetricKey();
inputMessage=JOptionPane.showInputDialog(null,"Enter message to encrypt");
byte[] ibyte = inputMessage.getBytes();
byte[] ebyte=encrypt(raw, ibyte);
String encryptedData = new String(ebyte);
System.out.println("Encrypted message "+encryptedData);
JOptionPane.showMessageDialog(null,"Encrypted Data "+"\n"+encryptedData);
byte[] dbyte= decrypt(raw,ebyte);
String decryptedMessage = new String(dbyte);
System.out.println("Decrypted message "+decryptedMessage);
JOptionPane.showMessageDialog(null,"Decrypted Data "+"\n"+decryptedMessage);}
catch(Exception e) {
System.out.println(e);
}
}
void generateSymmetricKey() {
try {
Random r = new Random();
int num = r.nextInt(10000);
String knum = String.valueOf(num);
byte[] knumb = knum.getBytes();
skey=getRawKey(knumb);
skeyString = new String(skey);
System.out.println("DES Symmetric key = "+skeyString);
```

```
}
catch(Exception e) {
System.out.println(e);}
}
private static byte[] getRawKey(byte[] seed) throws Exception {
KeyGenerator kgen = KeyGenerator.getInstance("DES");
SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");
sr.setSeed(seed);
kgen.init(56, sr);
SecretKey skey = kgen.generateKey();
raw = skey.getEncoded();
return raw;
}
private static byte[] encrypt(byte[] raw, byte[] clear) throws Exception {
SecretKeySpec skeySpec = new SecretKeySpec(raw, "DES");
Cipher cipher = Cipher.getInstance("DES");
cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
byte[] encrypted = cipher.doFinal(clear);
return encrypted;
}
private static byte[] decrypt(byte[] raw, byte[] encrypted) throws Exception {
SecretKeySpec skeySpec = new SecretKeySpec(raw, "DES");
Cipher cipher = Cipher.getInstance("DES");
cipher.init(Cipher.DECRYPT_MODE, skeySpec);
byte[] decrypted = cipher.doFinal(encrypted);
return decrypted;
}
public static void main(String args[]) {
DES des = new DES();
}
```

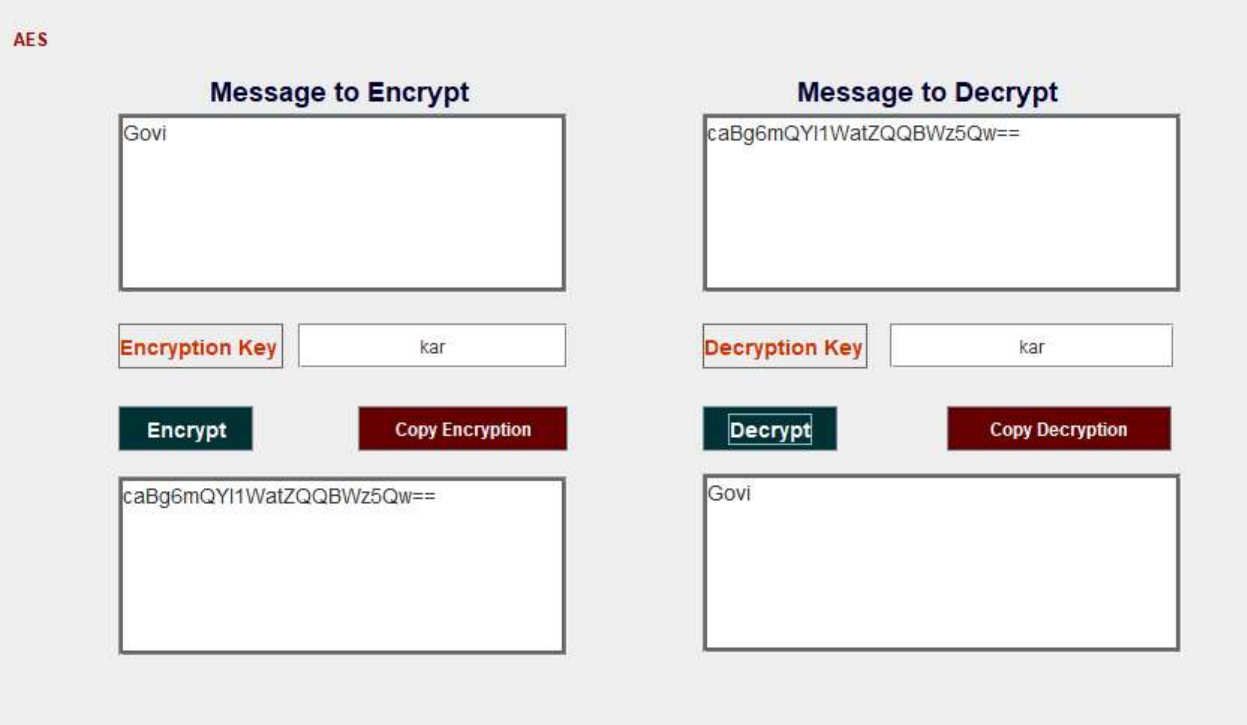
Outputs

Text encryption and decryption using RSA output

Command Prompt

```
Microsoft Windows [Version 10.0.18362.418]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\dell>D:

D:\>javac RSAA.java

D:\>java RSAA
Enter the plain text:
k.karthikeyan
Encrypting String: k.karthikeyan
String in Bytes: 1074610797114116104105107101121971l0
Decrypting Bytes: 1074610797114116104105107101121971l0
Decrypted String: k.karthikeyan

D:\>
```

Text encryption and decryption using AES output



Text encryption and decryption using DES output

ALGORITHM COMPARISON

Encryption time

the time taken to convert plaintext to ciphertext is encryption time. Encryption time depends upon key size, plaintext block size and mode. In our experiment we have measured encryption time in milliseconds. Encryption time impacts performance of the system. Encryption time must be less making the system fast and responsive.

Encryption Time:

ALGORITHM	TIME
RSA	1.5 seconds
DES	1.5 - 2 seconds
AES	1.5 - 3 seconds

Memory used

Different encryption techniques require different memory size for implementation. This memory requirement depends on the number of operations to be done by the algorithm, key size used, initialization vectors used and type of operations. The memory used impacts cost of the system. It is desirable that the memory required should be as small as possible.

Memory usage:

Algorithm	Memory used
RSA	31.5KB
DES	18.2KB
AES	14.7KB

Conclusion

So from our conclusion each algorihtim have their own advantages from our comparision so When system need high data bandwidth we can use DES and in banking transaction we use RSA because within some session the transaction can happen so big prime numbers can use and for system needs confidentiality and integrity we use AES so under our assumption AES made best but still it too hackable and in future may be some Advanced RSA may came .

References

1. 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

2. 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
3. A study and performance of RSA algorithm, IJCSMC, Vol. 2, Issue. 6, June 2013, pg.126 – 139, ISSN 2320– 088X
4. 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

Youtube refernce links

https://youtu.be/BCqW5XwtJxY

https://youtu.be/xIx_zI1tqAg

http://www.ijiset.com/v1s10/IJISET_V1_I10_106.pdf