LAMPIRAN

DAFTAR LAMPIRAN

Lampiran 1 : *Source Code Program*

a. *Membangkitan keypair & proses enkrip dekrip.cs*

```
 /*
 * To change this license header, choose License Headers in
Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package com.socket;

/**
 *
 * @author gamers
 */
import java.awt.EventQueue;
import java.io.*;
import java.math.BigInteger;
import java.util.ArrayList;
import java.util.Random;
import java.util.Scanner;


/**
 * Quick and dirty implementation of the RSA algorithm
 * Read through main() for a breakdown on the RSA workings
 */
public class Coba {

      /**
       * @param args
       */
      public static void main(String[] args) {
            // 1. Find large primes p and q
            BigInteger p = largePrime(512);
            BigInteger q = largePrime(512);

            // 2. Compute n from p and q
            // n is mod for private & public keys, n bit length
is key length
            BigInteger n = n(p, q);

            // 3. Compute Phi(n) (Euler's totient function)
            // Phi(n) = (p-1)(q-1)
            // BigIntegers are objects and must use methods for
algebraic operations
            BigInteger phi = getPhi(p, q);

            // 4. Find an int e such that 1 < e < Phi(n)    and
gcd(e,Phi) = 1
            BigInteger e = genE(phi);

            // 5. Calculate d where  d ≡ e^(-1) (mod Phi(n))
            BigInteger d = extEuclid(e, phi)[1];
```

```
            // Print generated values for reference
            System.out.println("p: " + p);
            System.out.println("q: " + q);
            System.out.println("n: " + n);
            System.out.println("Phi: " + phi);
            System.out.println("e: " + e);
            System.out.println("d: " + d);

            // encryption / decryption example
            String message = "wahono";
            // Convert string to numbers using a cipher
            BigInteger cipherMessage = stringCipher(message);
            // Encrypt the ciphered message
            BigInteger encrypted = encrypt(cipherMessage, e, n);
            // Decrypt the encrypted message
            BigInteger decrypted = decrypt(encrypted, d, n);
            // Uncipher the decrypted message to text
            String restoredMessage = cipherToString(decrypted);

            System.out.println("Original message: " + message);
            System.out.println("Ciphered: " + cipherMessage);
            System.out.println("Encrypted: " + encrypted);
            System.out.println("Decrypted: " + decrypted);
            System.out.println("Restored: " + restoredMessage);
      }


      /**
       * Takes a string and converts each character to an ASCII
decimal value
       * Returns BigInteger
       */
      public static BigInteger stringCipher(String message) {
      message = message.toUpperCase();
            String cipherString = "";
            int i = 0;
            while (i < message.length()) {
                  int ch = (int) message.charAt(i);
                  cipherString = cipherString + ch;
                  i++;
            }
            BigInteger cipherBig = new
BigInteger(String.valueOf(cipherString));
            return cipherBig;
      }


      /**
       * Takes a BigInteger that is ciphered and converts it
back to plain text
       *    returns a String
       */
      public static String cipherToString(BigInteger message) {
            String cipherString = message.toString();
            String output = "";
            int i = 0;
            while (i < cipherString.length()) {
                  int temp =
Integer.parseInt(cipherString.substring(i, i + 2));
```

```
                  char ch = (char) temp;
                  output = output + ch;
                  i = i + 2;
            }
            return output;
      }


      /** 3. Compute Phi(n) (Euler's totient function)
       *  Phi(n) = (p-1)(q-1)
       *    BigIntegers are objects and must use methods for
algebraic operations
       */
      public static BigInteger getPhi(BigInteger p, BigInteger
q) {
            BigInteger phi =
(p.subtract(BigInteger.ONE)).multiply(q.subtract(BigInteger.ONE)
);
            return phi;
      }

      /**
       * Generates a random large prime number of specified
bitlength
       *
       */
      public static BigInteger largePrime(int bits) {
            Random randomInteger = new Random();
            BigInteger largePrime =
BigInteger.probablePrime(bits, randomInteger);
            return largePrime;
      }


      /**
       * Recursive implementation of Euclidian algorithm to find
greatest common denominator
       * Note: Uses BigInteger operations
       */
      public static BigInteger gcd(BigInteger a, BigInteger b) {
            if (b.equals(BigInteger.ZERO)) {
                  return a;
            } else {
                  return gcd(b, a.mod(b));
            }
      }


      /** Recursive EXTENDED Euclidean algorithm, solves
Bezout's identity (ax + by = gcd(a,b))
       * and finds the multiplicative inverse which is the
solution to ax ≡ 1 (mod m)
       * returns [d, p, q] where d = gcd(a,b) and ap + bq = d
       * Note: Uses BigInteger operations
       */

        //mencari privat key
      public static BigInteger[] extEuclid(BigInteger a,
BigInteger b) {
```

```
            if (b.equals(BigInteger.ZERO)) return new
BigInteger[] {
                  a, BigInteger.ONE, BigInteger.ZERO
            }; // { a, 1, 0 }
            BigInteger[] vals = extEuclid(b, a.mod(b));
            BigInteger d = vals[0];
            BigInteger p = vals[2];
            BigInteger q =
vals[1].subtract(a.divide(b).multiply(vals[2]));
            return new BigInteger[] {
                  d, p, q
            };
      }


      /**
       * generate e by finding a Phi such that they are coprimes
(gcd = 1)
       *
       */
      public static BigInteger genE(BigInteger phi) {
            Random rand = new Random();
            BigInteger e = new BigInteger(1024, rand);
            do {
                  e = new BigInteger(1024, rand);
                  while (e.min(phi).equals(phi)) { // while phi
is smaller than e, look for a new e
                        e = new BigInteger(1024, rand);
                  }
            } while (!gcd(e, phi).equals(BigInteger.ONE)); // if
gcd(e,phi) isnt 1 then stay in loop
            return e;
      }

      public static BigInteger encrypt(BigInteger message,
BigInteger e, BigInteger n) {
            return message.modPow(e, n);
      }

      public static BigInteger decrypt(BigInteger message,
BigInteger d, BigInteger n) {
            return message.modPow(d, n);
      }

      public static BigInteger n(BigInteger p, BigInteger q) {
            return p.multiply(q);

      }
}
```

Lampiran 2 :Hasil Kuisioner

a. Hasil Kuisioner 1

Kuisioner Evaluasi Desain dan Implementasi Algoritma Kriptografi RSA pada Telkom Bojonegoro untuk Meningkatkan Keamanan Sistem Jaringan

Nama* : Priyanto

NIP : 646018

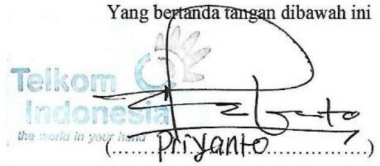
Beri tanda silang (√) pada pilihan yang sesuai.

Sangat Setuju/SS (Skor 5), Setuju/S (Skor 4), Cukup Setuju/CS (Skor 3), Tidak Setuju/TS (Skor 2) dan Sangat Tidak Setuju/STS (Skor 1).

NO	URAIAN	SKORE				
		1	2	3	4	5
1	Tampilan sistem nyaman dan sesuai keinginan				√	
2	Kesesuaian nama tombol dan layanan				√	
3	Mudah memahami sistem dan kegunaanya					√
4	Keamanan sistem teruji dengan baik					√
5	Hasil data dekrip sesuai dengan plaintext				√	
6	Apakah dengan adanya sistem ini dapat membantu perusahaan anda				√	

Bojonegoro ….,……

Yang bertanda tangan dibawah ini



(……Priyanto……)

*Wajib diisi berdasarkan identitas pegawai asli

b. Hasil Kuisioner 2

Kuisioner Evaluasi Desain dan Implementasi Algoritma Kriptografi RSA pada Telkom Bojonegoro untuk Meningkatkan Keamanan Sistem Jaringan

Nama* : Zainul Fanani

NIP : 18860001

Beri tanda silang (√) pada pilihan yang sesuai.

Sangat Setuju/SS (Skor 5), Setuju/S (Skor 4), Cukup Setuju/CS (Skor 3), Tidak Setuju/TS (Skor 2) dan Sangat Tidak Setuju/STS (Skor 1).

NO	URAIAN	SKORE				
		1	2	3	4	5
1	Tampilan sistem nyaman dan sesuai keinginan			√		
2	Kesesuaian nama tombol dan layanan				√	
3	Mudah memahami sistem dan kegunaanya				√	
4	Keamanan sistem teruji dengan baik					√
5	Hasil data dekrip sesuai dengan plaintext					√
6	Apakah dengan adanya sistem ini dapat membantu perusahaan anda				√	

Bojonegoro ...,.....

Yang bertanda tangan dibawah ini

Telkom Indonesia

(Zainul Fanani)

*Wajib diisi berdasarkan identitas pegawai asli