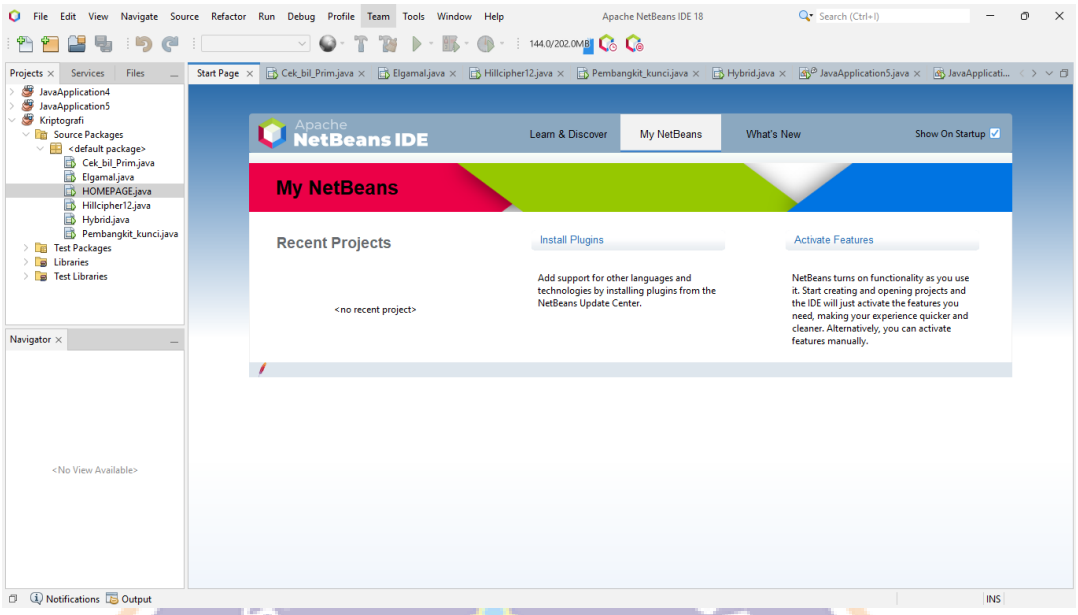
LAMPIRAN

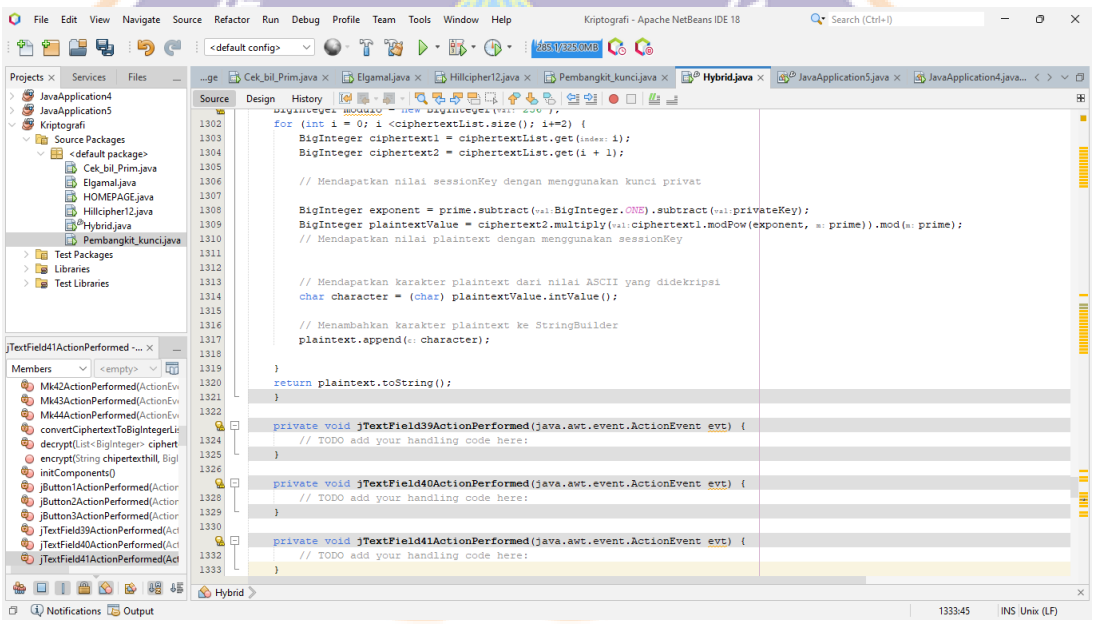
Lampiran 1. Kode ASCII - 256 Character (American Standard Code for Information Interchange)

Dec	Hex	Char
0	00	NUL
1	01	SOH
2	02	STX
3	03	ETX
4	04	EOT
5	05	ENQ
6	06	ACK
7	07	BEL
8	08	BS
9	09	HT
10	0A	LF
11	0B	VT
12	0C	FF
13	0D	CR
14	0E	SO
15	0F	SI
16	10	DLE
17	11	DC1
18	12	DC2
19	13	DC3
20	14	DC4
21	15	NAK
22	16	SYN
23	17	ETB
24	18	CAN
25	19	EM
26	1A	SUB
27	1B	ESC
28	1C	FS
29	1D	GS
30	1E	RS
31	1F	US
32	20	space
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	DEL

Dec	Hex	Char
128	80	□
129	81	□
130	82	□
131	83	□
132	84	□
133	85	□
134	86	□
135	87	□
136	88	□
137	89	□
138	8A	□
139	8B	□
140	8C	□
141	8D	□
142	8E	□
143	8F	□
144	90	□
145	91	□
146	92	□
147	93	□
148	94	□
149	95	□
150	96	□
151	97	□
152	98	□
153	99	□
154	9A	□
155	9B	□
156	9C	□
157	9D	□
158	9E	□
159	9F	□
160	A0	
161	A1	¡
162	A2	¢
163	A3	£
164	A4	¤
165	A5	¥
166	A6	¦
167	A7	§
168	A8	¨
169	A9	©
170	AA	ª
171	AB	«
172	AC	¬
173	AD	
174	AE	®
175	AF	¯
176	B0	°
177	B1	±
178	B2	²
179	B3	³
180	B4	´
181	B5	µ
182	B6	¶
183	B7	·
184	B8	¸
185	B9	¹
186	BA	º
187	BB	»
188	BC	¼
189	BD	½
190	BE	¾
191	BF	¿
192	C0	À
193	C1	Á
194	C2	Â
195	C3	Ã
196	C4	Ä
197	C5	Å
198	C6	Æ
199	C7	Ç
200	C8	È
201	C9	É
202	CA	Ê
203	CB	Ë
204	CC	Ì
205	CD	Í
206	CE	Î
207	CF	Ï
208	D0	Ð
209	D1	Ñ
210	D2	Ò
211	D3	Ó
212	D4	Ô
213	D5	Õ
214	D6	Ö
215	D7	×
216	D8	Ø
217	D9	Ù
218	DA	Ú
219	DB	Û
220	DC	Ü
221	DD	Ý
222	DE	Þ
223	DF	ß
224	E0	à
225	E1	á
226	E2	â
227	E3	ã
228	E4	ä
229	E5	å
230	E6	æ
231	E7	ç
232	E8	è
233	E9	é
234	EA	ê
235	EB	ë
236	EC	ì
237	ED	í
238	EE	î
239	EF	ï
240	F0	ð
241	F1	ñ
242	F2	ò
243	F3	ó
244	F4	ô
245	F5	õ
246	F6	ö
247	F7	÷
248	F8	ø
249	F9	ù
250	FA	ú
251	FB	û
252	FC	ü
253	FD	ý
254	FE	þ
255	FF	ÿ

Lampiran 2. Tampilan JAVA NetBeans18





Lampiran 3. Source Code HomePage.java

```
private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
        Elgamal elgml = new Elgamal();
            elgml.setVisible(true);
                this.dispose();     }

    private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
        Hillcipher12 HC = new Hillcipher12();
            HC.setVisible(true);
                this.dispose();// TODO add your handling code here: }

    private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {
        Cek_bil_Prim cekbilprim = new Cek_bil_Prim();
             cekbilprim.setVisible(true);
                this.dispose();
    }

    private void jButton8ActionPerformed(java.awt.event.ActionEvent evt) {
        Pembangkit_kunci PK = new Pembangkit_kunci();
            PK.setVisible(true);
                this.dispose();// TODO add your handling code here:
    }
    private void jButton9ActionPerformed(java.awt.event.ActionEvent evt) {
        Hybrid hyb=new Hybrid();
            hyb.setVisible(true);
                this.dispose();// TODO add your handling code here:
    }
    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(HOMEPAGE.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(HOMEPAGE.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(HOMEPAGE.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(HOMEPAGE.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>
        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new HOMEPAGE().setVisible(true); }
        });
    }
```

Lampiran 4. Source Code Cek_bil_Prim.java

```
import static java.lang.Integer.parseInt;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;
public class Cek_bil_Prim extends javax.swing.JFrame {
    /**
     * Creates new form Cek_bil_Prim
     */
    public Cek_bil_Prim() {
        initComponents();
    }
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
        HOMEPAGE hmpge = new HOMEPAGE();
            hmpge.setVisible(true);
                this.dispose();// TODO add your handling code here:
    }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        String input = numberField.getText();
        int bil;
        try{bil =Integer.parseInt(input);}
        catch (NumberFormatException e){JOptionPane.showMessageDialog(this,"input yang anda masukan salah, masukan
bilangan bulat yang benar");
        return;
        }
        if (isPrime(bil)){
        jTextField2.setText("Bilangan Ini merupakan Bilangan Prima");}
        else{
         jTextField2.setText("Bilangan Ini bukan merupakan Bilangan Prima");
        }
    }
      public static boolean isPrime(int number) {
        if (number <= 1) {
            return false;
        }

        for (int i = 2; i <= Math.sqrt(number); i++) {
            if (number % i == 0) {
                return false;
            }
        }

        return true;
    }
```

Lampiran 5. Source Code enkripsi Pembangkit_kunci.java

```
import java.util.Random;
import javax.swing.JOptionPane;
import java.math.BigInteger;
public class Pembangkit_kunci extends javax.swing.JFrame {
   /**
    * Creates new form Pembangkit_kunci
    */
   public Pembangkit_kunci() {
      initComponents();
   }
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
      String input = p.getText();
      int bil;

      try{bil =Integer.parseInt(input);}

      catch (NumberFormatException e){JOptionPane.showMessageDialog(this,"input yang anda masukan salah, masukan
bilangan bulat yang benar");
      return;
      }

        int upperlimit=bil;
        int b = upperlimit-2;
            Random random = new Random();
            int randomNumber=random.nextInt(upperlimit);
            int randomNumber1=random.nextInt(b);
            BigInteger bil1=new BigInteger(String.valueOf(upperlimit));
            jTextField1.setText(String.valueOf(randomNumber));
            jTextField2.setText(String.valueOf(randomNumber1));
             BigInteger RN = new BigInteger(String.valueOf(randomNumber));
            BigInteger y1 = RN.pow(randomNumber1);
            BigInteger y = y1.remainder(bil1);
            jTextField3.setText(String.valueOf(y));
   }

   private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
         HOMEPAGE hmpg = new HOMEPAGE();
            hmpg.setVisible(true);
               this.dispose();
```

Lampiran 6. Source Code dekripsi Hillcipher12.java

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
String input_1 = Mk11.getText();
String input_2 = Mk12.getText();
String input_3 = Mk13.getText();
String input_4 = Mk14.getText();
String input_5 = Mk21.getText();
String input_6 = Mk22.getText();
String input_7 = Mk23.getText();
String input_8 = Mk24.getText();
String input_9 = Mk31.getText();
String input_10 = Mk32.getText();
String input_11 = Mk33.getText();
String input_12 = Mk34.getText();
String input_13 = Mk41.getText();
String input_14 = Mk42.getText();
String input_15 = Mk43.getText();
String input_16 = Mk44.getText();

int M11, M12, M13, M14, M21, M22, M23, M24, M31, M32, M33, M34, M41, M42, M43, M44;

try {
   M11 = Integer.parseInt(input_1);
   M12 = Integer.parseInt(input_2);
   M13 = Integer.parseInt(input_3);
   M14 = Integer.parseInt(input_4);
   M21 = Integer.parseInt(input_5);
   M22 = Integer.parseInt(input_6);
   M23 = Integer.parseInt(input_7);
   M24 = Integer.parseInt(input_8);
   M31 = Integer.parseInt(input_9);
   M32 = Integer.parseInt(input_10);
   M33 = Integer.parseInt(input_11);
   M34 = Integer.parseInt(input_12);
   M41 = Integer.parseInt(input_13);
   M42 = Integer.parseInt(input_14);
   M43 = Integer.parseInt(input_15);
   M44 = Integer.parseInt(input_16);
} catch (NumberFormatException e) {
   JOptionPane.showMessageDialog(this, "Input yang Anda masukkan salah, masukkan bilangan bulat yang benar");
   return;
}
if (M11 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 1 kolom 1");
   return;
}
if (M12 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 1 kolom 2");
   return;
}
if (M13 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 1 kolom 3");
   return;
}
if (M14 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 1 kolom 4");
   return;
}
if (M21 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 2 kolom 1");
   return;
}
if (M22 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 2 kolom 2");
   return;
}
if (M23 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 2 kolom 3");
   return;
}
```

```
if (M24 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 2 kolom 4");
   return;
}
if (M31 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 3 kolom 1");
   return;
}
if (M32 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 3 kolom 2");
   return;
}
if (M33 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 3 kolom 3");
   return;
}
if (M34 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 3 kolom 4");
   return;
}
if (M41 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 4 kolom 1");
   return;
}
if (M42 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 4 kolom 2");
   return;
}
if (M43 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 4 kolom 3");
   return;
}
if (M44 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 4 kolom 4");
   return;
}
int C11 = (M22 * M33 * M44 + M23 * M34 *M42 + M24 * M32 * M43 - M42 * M33 * M24 - M43 * M34 * M22 - M44
* M32 * M23);
int C12 = (M21 * M33 * M44 + M23 * M34 *M41 + M24 * M31 * M43 - M41 * M33 * M24 - M43 * M34 * M21 - M44
* M31 * M23);
int C13 = (M21 * M32 * M44 + M22 * M34 *M41 + M24 * M31 * M42 - M41 * M32 * M24 - M42 * M34 * M21 - M44
* M31 * M22);
int C14 = (M21 * M32 * M43 + M22 * M33 *M41 + M23 * M31 * M42 - M41 * M32 * M23 - M42 * M33 * M21 - M43
* M31 * M22);
int det = M11*C11 - M12*C12 + M13*C13 - M14*C14;
jTextField1.setText(String.valueOf(det));

if (det == 0) {
   JOptionPane.showMessageDialog(this, "Matriks Kunci tidak memiliki invers");
   return;
}
if (det % 2 == 0) {
   JOptionPane.showMessageDialog(this, "Matriks Kunci determinan bernilai genap, silahkan masukan matriks kunci yang
determinannya bernilai ganjil");
   return;
}
if (det < 0) {
   JOptionPane.showMessageDialog(this, "Matriks Kunci determinan bernilai negatif, silahkan masukan matriks kunci
yang determinannya bernilai ganjil");
   return;
}
int[][] keyMatrix = new int[4][4];
keyMatrix[0][0] = M11;
keyMatrix[0][1] = M12;
keyMatrix[0][2] = M13;
keyMatrix[0][3] = M14;
keyMatrix[1][0] = M21;
keyMatrix[1][1] = M22;
keyMatrix[1][2] = M23;
keyMatrix[1][3] = M24;
keyMatrix[2][0] = M31;
keyMatrix[2][1] = M32;
keyMatrix[2][2] = M33;
```

```
keyMatrix[2][3] = M34;
keyMatrix[3][0] = M41;
keyMatrix[3][1] = M42;
keyMatrix[3][2] = M43;
keyMatrix[3][3] = M44;
// Mengukur waktu eksekusi
String plainText = jTextArea1.getText();
int blockSize = 4;
int remainder = plainText.length() % blockSize;
int padLength = remainder == 0 ? 0 : blockSize - remainder;
if (padLength != 0) {
   StringBuilder paddedText = new StringBuilder(plainText);
   for (int i = 0; i < padLength; i++) {
      paddedText.append("X");
   }
   plainText = paddedText.toString();
}
List<List<Integer>> asciiBlocks = new ArrayList<>();

for (int i = 0; i < plainText.length(); i += blockSize) {
   List<Integer> asciiBlock = new ArrayList<>();
   for (int j = i; j < i + blockSize; j++) {
      char c = plainText.charAt(j);
      int asciiValue = (int) c;
      asciiBlock.add(asciiValue);
   }
   asciiBlocks.add(asciiBlock);
}
long startTime = System.nanoTime();
StringBuilder encryptedText = new StringBuilder();
StringBuilder encryptedAsciiText = new StringBuilder();

for (List<Integer> asciiBlock : asciiBlocks) {
   List<Integer> encryptedBlock = new ArrayList<>();

   for (int i = 0; i < asciiBlock.size(); i += blockSize) {
      int[] v = new int[blockSize];
      for (int j = 0; j < blockSize && (i + j) < asciiBlock.size(); j++) {
         v[j] = asciiBlock.get(i + j);
      }

      int[] result = new int[blockSize];
      for (int row = 0; row < blockSize; row++) {
         for (int col = 0; col < blockSize; col++) {
            result[row] += keyMatrix[row][col] * v[col];
         }
         result[row] %= 256;
      }

      for (int value : result) {
         encryptedBlock.add(value);
          encryptedAsciiText.append(value).append(" ");
      }
   }

   for (int value : encryptedBlock) {
      String character = Character.toString((char) value);
      encryptedText.append(character);
   }
}
 long endTime = System.nanoTime();
      long executionTime = endTime - startTime;

jTextArea3.setText(encryptedText.toString());
jTextArea2.setText(encryptedAsciiText.toString());
jTextField34.setText( + executionTime + " ns");
   }

   private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
      HOMEPAGE hmpge = new HOMEPAGE();
         hmpge.setVisible(true);
            this.dispose();
```

Lampiran 7. Source Code dekripsi Hillcipher12.java

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
String input_1 = A11.getText();
String input_2 = A12.getText();
String input_3 = A13.getText();
String input_4 = A14.getText();
String input_5 = A21.getText();
String input_6 = A22.getText();
String input_7 = A23.getText();
String input_8 = A24.getText();
String input_9 = A31.getText();
String input_10 = A32.getText();
String input_11 = A33.getText();
String input_12 = A34.getText();
String input_13 = A41.getText();
String input_14 = A42.getText();
String input_15 = A43.getText();
String input_16 = A44.getText();
int A11, A12, A13, A14, A21, A22, A23, A24, A31, A32, A33, A34, A41, A42, A43, A44;
try {
   A11 = Integer.parseInt(input_1);
   A12 = Integer.parseInt(input_2);
   A13 = Integer.parseInt(input_3);
   A14 = Integer.parseInt(input_4);
   A21 = Integer.parseInt(input_5);
   A22 = Integer.parseInt(input_6);
   A23 = Integer.parseInt(input_7);
   A24 = Integer.parseInt(input_8);
   A31 = Integer.parseInt(input_9);
   A32 = Integer.parseInt(input_10);
   A33 = Integer.parseInt(input_11);
   A34 = Integer.parseInt(input_12);
   A41 = Integer.parseInt(input_13);
   A42 = Integer.parseInt(input_14);
   A43 = Integer.parseInt(input_15);
   A44 = Integer.parseInt(input_16);
} catch (NumberFormatException e) {
   JOptionPane.showMessageDialog(this, "Input yang Anda masukkan salah, masukkan bilangan bulat yang benar");
   return;
}
 //mencari ADjoin matriks kunci
int C11 = A22 * A33 * A44 + A23 * A34 *A42 + A24 * A32 * A43 - A42 * A33 * A24 - A43 * A34 * A22 - A44 * A32 *
A23;
int C12 = A21 * A33 * A44 + A23 * A34 *A41 + A24 * A31 * A43 - A41 * A33 * A24 - A43 * A34 * A21 - A44 * A31 *
A23;
int C13 = A21 * A32 * A44 + A22 * A34 *A41 + A24 * A31 * A42 - A41 * A32 * A24 - A42 * A34 * A21 - A44 * A31 *
A22;
int C14 = A21 * A32 * A43 + A22 * A33 *A41 + A23 * A31 * A42 - A41 * A32 * A23 - A42 * A33 * A21 - A43 * A31 *
A22;
int C21 = A12 * A33 * A44 + A13 * A34 *A42 + A14 * A32 * A43 - A42 * A33 * A14 - A43 * A34 * A12 - A44 * A32 *
A13;
int C22 = A11 * A33 * A44 + A13 * A34 *A41 + A14 * A31 * A43 - A41 * A33 * A14 - A43 * A34 * A11 - A44 * A31 *
A13;
int C23 = A11 * A32 * A44 + A12 * A34 *A41 + A14 * A31 * A42 - A41 * A32 * A14 - A42 * A34 * A11 - A44 * A31 *
A12;
int C24 = A11 * A32 * A43 + A12 * A33 *A41 + A13 * A31 * A42 - A41 * A32 * A13 - A42 * A33 * A11 - A43 * A31 *
A12;
int C31 = A12 * A23 * A44 + A13 * A24 *A42 + A14 * A22 * A43 - A42 * A23 * A14 - A43 * A24 * A12 - A44 * A22 *
A13;
int C32 = A11 * A23 * A44 + A13 * A24 *A41 + A14 * A21 * A43 - A41 * A23 * A14 - A43 * A24 * A11 - A44 * A21 *
A13;
int C33 = A11 * A22 * A44 + A12 * A24 *A41 + A14 * A21 * A42 - A41 * A22 * A14 - A42 * A24 * A11 - A44 * A21 *
A12;
int C34 = A11 * A22 * A43 + A12 * A23 *A41 + A13 * A21 * A42 - A41 * A22 * A13 - A42 * A23 * A11 - A43 * A21 *
A12;
```

```
int C41 = A12 * A23 * A34 + A13 * A24 *A32 + A14 * A22 * A33 - A32 * A23 * A14 - A33 * A24 * A12 - A34 * A22 *
A13;
int C42 = A11 * A23 * A34 + A13 * A24 *A31 + A14 * A21 * A33 - A31 * A23 * A14 - A33 * A24 * A11 - A34 * A21 *
A13;
int C43 = A11 * A22 * A34 + A12 * A24 *A31 + A14 * A21 * A32 - A31 * A22 * A14 - A32 * A24 * A11 - A34 * A21 *
A12;
int C44 = A11 * A22 * A33 + A12 * A23 *A31 + A13 * A21 * A32 - A31 * A22 * A13 - A32 * A23 * A11 - A33 * A21 *
A12;

if (A11 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 1 kolom 1");
   return;
}
if (A12 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 1 kolom 2");
   return;
}
if (A13 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 1 kolom 3");
   return;
}
if (A14 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 1 kolom 4");
   return;
}
if (A21 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 2 kolom 1");
   return;
}
if (A22 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 2 kolom 2");
   return;
}
if (A23 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 2 kolom 3");
   return;
}
if (A24 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 2 kolom 4");
   return;
}
if (A31 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 3 kolom 1");
   return;
}
if (A32 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 3 kolom 2");
   return;
}
if (A33 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 3 kolom 3");
   return;
}
if (A34 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 3 kolom 4");
   return;
}
if (A41 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 4 kolom 1");
   return;
}
if (A42 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 4 kolom 2");
   return;
}
if (A43 < 0) {
   JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 4 kolom 3");
```

```
    return;
}
if (A44 < 0) {
    JOptionPane.showMessageDialog(this, "Masukan bilangan Bulat positif pada Baris 4 kolom 4");
    return;
}

int Z11 = (A22 * A33 * A44 + A23 * A34 *A42 + A24 * A32 * A43 - A42 * A33 * A24 - A43 * A34 * A22 - A44 * A32 *
A23);
int Z12 = (A21 * A33 * A44 + A23 * A34 *A41 + A24 * A31 * A43 - A41 * A33 * A24 - A43 * A34 * A21 - A44 * A31 *
A23);
int Z13 = (A21 * A32 * A44 + A22 * A34 *A41 + A24 * A31 * A42 - A41 * A32 * A24 - A42 * A34 * A21 - A44 * A31 *
A22);
int Z14 = (A21 * A32 * A43 + A22 * A33 *A41 + A23 * A31 * A42 - A41 * A32 * A23 - A42 * A33 * A21 - A43 * A31 *
A22);

int det = Z11*A11 - Z12*A12 + Z13*A13 - Z14*A14;
jTextField2.setText(String.valueOf(det));

//int det = A11*C11 - A12*C12 + A13*C13 - A14*C14;
//jTextField17.setText(String.valueOf(det));

if (det == 0 ) {
    JOptionPane.showMessageDialog(this, "Matriks Kunci tidak memiliki invers");
    return;
}
if (det % 2 == 0) {
    JOptionPane.showMessageDialog(this, "Matriks Kunci determinan bernilai genap, silahkan masukan matriks kunci yang
determinannya bernilai ganjil");
    return;
}
if (det < 0) {
    JOptionPane.showMessageDialog(this, "Matriks Kunci determinan bernilai negatif, silahkan masukan matriks kunci
yang determinannya bernilai ganjil");
    return;
}
 int detInverse = -1;
for (int i = 0; i < 256; i++) {
    if ((det * i) % 256 == 1) {
        detInverse = i;
        break;
    }
}
//jTextArea4.setText(String.valueOf(detInverse));
int[][] keyMatrix = new int[4][4];
keyMatrix[0][0] = (C11 * detInverse);
keyMatrix[0][1] = (-C21 * detInverse);
keyMatrix[0][2] = (C31 * detInverse);
keyMatrix[0][3] = (-C41 * detInverse);
keyMatrix[1][0] = (-C12 * detInverse);
keyMatrix[1][1] = (C22 * detInverse);
keyMatrix[1][2] = (-C32 * detInverse);
keyMatrix[1][3] = (C42 * detInverse);
keyMatrix[2][0] = (C13 * detInverse);
keyMatrix[2][1] = (-C23 * detInverse);
keyMatrix[2][2] = (C33 * detInverse);
keyMatrix[2][3] = (-C43 * detInverse);
keyMatrix[3][0] = (-C14 * detInverse);
keyMatrix[3][1] = (C24 * detInverse);
keyMatrix[3][2] = (-C34 * detInverse);
keyMatrix[3][3] = (C44 * detInverse);
// Mengukur waktu eksekusi

String encryptedText = jTextArea3.getText();
int blockSize = 4;
List<List<Integer>> asciiBlocks = new ArrayList<>();
```

```
for (int i = 0; i < encryptedText.length(); i += blockSize) {
    List<Integer> asciiBlock = new ArrayList<>();
    for (int j = i; j < i + blockSize && j < encryptedText.length(); j++) {
        char c = encryptedText.charAt(j);
        int asciiValue = (int) c;
        asciiBlock.add(asciiValue);
    }
    asciiBlocks.add(asciiBlock);
}
long startTime = System.nanoTime();
StringBuilder decryptedText = new StringBuilder();

for (List<Integer> asciiBlock : asciiBlocks) {
    List<Integer> decryptedBlock = new ArrayList<>();

    for (int i = 0; i < asciiBlock.size(); i += blockSize) {
        int[] v = new int[blockSize];
        for (int j = 0; j < blockSize && (i + j) < asciiBlock.size(); j++) {
            v[j] = asciiBlock.get(i + j);
        }

        int[] result = new int[blockSize];

        for (int row = 0; row < blockSize; row++) {
            for (int col = 0; col < blockSize; col++) {
                result[row] += keyMatrix[row][col] * v[col];
            }
            result[row] %= 256;
            if (result[row] < 0) {
                result[row] += 256;
            }
        }

        for (int value : result) {
            decryptedBlock.add(value);
        }
    }
    for (int value : decryptedBlock) {
        String character = Character.toString((char) value);
        decryptedText.append(character);
    }
}
long endTime = System.nanoTime();
 long executionTime = endTime - startTime;

jTextArea4.setText(decryptedText.toString());
jTextField17.setText(+ executionTime + " ns");
```

Lampiran 8. Source Code Enkripsi ElGamal.java

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        String plaintext = jTextArea1.getText();
           BigInteger prime = new BigInteger(jTextField3.getText());
           BigInteger generator = new BigInteger(jTextField2.getText());
           BigInteger publicKey = new BigInteger(jTextField1.getText());
           BigInteger Modulo = new BigInteger("256");
// Mengukur waktu eksekusi
    long startTime = System.nanoTime();
           List<BigInteger> encryptedText = encrypt(plaintext, prime, generator, publicKey);
           // Menampilkan hasil ciphertext pada textarea
           StringBuilder ciphertextOutput = new StringBuilder();
           //mengubah cipertext pada decimal ascii
           for (BigInteger ciphertext1 : encryptedText) {
            ciphertextOutput.append(ciphertext1.toString()).append(" ");
           }
        long waktuakhir = System.nanoTime();
        long executionTime = waktuakhir - startTime;
       jTextArea3.setText(ciphertextOutput.toString());
       jTextField17.setText("Execution Time: " + executionTime + " ns");
       StringBuilder ciphertextOutput1 = new StringBuilder();
            for (int i=0; i < encryptedText.size();i++){
            BigInteger ciphertexts = encryptedText.get(i).mod(Modulo);
              int asciiValue = ciphertexts.intValue();
              char character = (char) asciiValue;
             ciphertextOutput1.append(character).append(" ");
            }
         //  long waktuakhir = System.nanoTime();
      // long executionTime = waktuakhir - startTime;
    //  jTextField17.setText("Execution Time: " + executionTime + " ns");
            jTextArea2.setText(ciphertextOutput1.toString());
}
   public List<BigInteger> encrypt(String plaintext, BigInteger prime, BigInteger generator, BigInteger publicKey) {
       SecureRandom secureRandom = new SecureRandom();
       List<BigInteger> encryptedText = new ArrayList<>();


       for (int i = 0; i < plaintext.length(); i++) {
           char c = plaintext.charAt(i);
           BigInteger asciiValue = BigInteger.valueOf((int) c);
           // Generate random value for the session key

           //BigInteger sessionKey = new BigInteger(prime.bitLength() - 1, secureRandom).mod(prime);
           BigInteger maxSessionKey = prime.subtract(BigInteger.ONE); // p-1
           BigInteger sessionKeyRange = maxSessionKey.subtract(BigInteger.ONE); // p-2
           BigInteger sessionKey = new BigInteger(sessionKeyRange.bitLength(), secureRandom);

           // Tambahkan 1 untuk memastikan berada dalam rentang [1, p-1]
           sessionKey = sessionKey.add(BigInteger.ONE);
            System.out.println("Session Key: " + sessionKey);
           // Compute the ciphertext
           BigInteger ciphertext1 = generator.modPow(sessionKey, prime);
           BigInteger ciphertext2 = asciiValue.multiply(publicKey.modPow(sessionKey, prime)).mod(prime);

           encryptedText.add(ciphertext1);
           encryptedText.add(ciphertext2);


       }

       return encryptedText;
```

Lampiran 9. Source Code Dekripsi ElGamal.java

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
 String ciphertext = jTextArea3.getText();
BigInteger privateKey = new BigInteger(jTextField5.getText().trim());
BigInteger prime = new BigInteger(jTextField6.getText().trim());
long startTime = System.nanoTime();
List<BigInteger> ciphertextList = convertCiphertextToBigIntegerList(ciphertext);
String decryptedText = decrypt(ciphertextList, prime, privateKey);
long waktuakhir = System.nanoTime();
      long executionTime = waktuakhir - startTime;
      // Menampilkan hasil plaintext pada textarea
      jTextArea4.setText(decryptedText);
     jTextField18.setText( + executionTime + " ns");
   }

// Metode untuk mengonversi ciphertext menjadi daftar BigInteger
private List<BigInteger> convertCiphertextToBigIntegerList(String ciphertext) {
   List<BigInteger> ciphertextList = new ArrayList<>();
   String[] ciphertextArray = ciphertext.split(" ");
   for (String ciphertextNum : ciphertextArray) {
      if (!ciphertextNum.isEmpty()) {
         try {
            BigInteger num = new BigInteger(ciphertextNum.trim());
            ciphertextList.add(num);
         } catch (NumberFormatException e) {
            e.printStackTrace();
         }
      }
   }
   return ciphertextList;
}

// Metode untuk mendekripsi ciphertext
private String decrypt(List<BigInteger> ciphertextList, BigInteger prime, BigInteger privateKey) {
   StringBuilder plaintext = new StringBuilder();


   for (int i = 0; i <ciphertextList.size(); i+=2) {
      BigInteger ciphertext1 = ciphertextList.get(i);
      BigInteger ciphertext2 = ciphertextList.get(i + 1);

      // Mendapatkan nilai sessionKey dengan menggunakan kunci privat

      BigInteger exponent = prime.subtract(BigInteger.ONE).subtract(privateKey);
      BigInteger plaintextValue = ciphertext2.multiply(ciphertext1.modPow(exponent, prime)).mod(prime);
      // Mendapatkan nilai plaintext dengan menggunakan sessionKey


      // Mendapatkan karakter plaintext dari nilai ASCII yang didekripsi
      char character = (char) plaintextValue.intValue();

      // Menambahkan karakter plaintext ke StringBuilder
      plaintext.append(character);
   }

   return plaintext.toString();
   }
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
      HOMEPAGE hmpge = new HOMEPAGE();
         hmpge.setVisible(true);
            this.dispose();
```

Lampiran 10. Source Code Enkripsi Hybrid.java

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
 long startTime2 = System.nanoTime();
String input_1 = Mk11.getText();
String input_2 = Mk12.getText();
String input_3 = Mk13.getText();
String input_4 = Mk14.getText();
String input_5 = Mk21.getText();
String input_6 = Mk22.getText();
String input_7 = Mk23.getText();
String input_8 = Mk24.getText();
String input_9 = Mk31.getText();
String input_10 = Mk32.getText();
String input_11 = Mk33.getText();
String input_12 = Mk34.getText();
String input_13 = Mk41.getText();
String input_14 = Mk42.getText();
String input_15 = Mk43.getText();
String input_16 = Mk44.getText();

int M11, M12, M13, M14, M21, M22, M23, M24, M31, M32, M33, M34, M41, M42, M43, M44;

try {
   M11 = Integer.parseInt(input_1);
   M12 = Integer.parseInt(input_2);
   M13 = Integer.parseInt(input_3);
   M14 = Integer.parseInt(input_4);
   M21 = Integer.parseInt(input_5);
   M22 = Integer.parseInt(input_6);
   M23 = Integer.parseInt(input_7);
   M24 = Integer.parseInt(input_8);
   M31 = Integer.parseInt(input_9);
   M32 = Integer.parseInt(input_10);
   M33 = Integer.parseInt(input_11);
   M34 = Integer.parseInt(input_12);
   M41 = Integer.parseInt(input_13);
   M42 = Integer.parseInt(input_14);
   M43 = Integer.parseInt(input_15);
   M44 = Integer.parseInt(input_16);
} catch (NumberFormatException e) {
   JOptionPane.showMessageDialog(this, "Input yang Anda masukkan salah, masukkan bilangan bulat yang benar");
   return;
}
/*int det1 = M11*M22*M33*M44 - M12*M23*M34*M41 + M13*M24*M31*M42 - M14*M21*M32*M43
      - M42*M33*M24*M11 + M43*M34*M21*M12 - M44*M31*M22*M13 + M41*M32*M23*M14;
int det2 = -M11*M22*M34*M43 + M12*M23*M31*M44 - M13*M24*M32*M41 + M14*M21*M33*M42
      +M11*M24*M32*M43 - M12*M21*M33*M44 + M13*M22*M34*M41 - M14*M23*M31*M42;
int det3 = M11*M23*M34*M42 - M12*M24*M31*M43 + M13*M21*M32*M44 - M14*M22*M33*M41
      - M11*M23*M32*M44 + M12*M24*M33*M41 - M13*M22*M34*M42 + M14*M22*M31*M43;
int det = det1+det2+det3;*/
int C11 = (M22 * M33 * M44 + M23 * M34 *M42 + M24 * M32 * M43 - M42 * M33 * M24 - M43 * M34 * M22 - M44
* M32 * M23);
int C12 = (M21 * M33 * M44 + M23 * M34 *M41 + M24 * M31 * M43 - M41 * M33 * M24 - M43 * M34 * M21 - M44
* M31 * M23);
int C13 = (M21 * M32 * M44 + M22 * M34 *M41 + M24 * M31 * M42 - M41 * M32 * M24 - M42 * M34 * M21 - M44
* M31 * M22);
int C14 = (M21 * M32 * M43 + M22 * M33 *M41 + M23 * M31 * M42 - M41 * M32 * M23 - M42 * M33 * M21 - M43
* M31 * M22);

int det = M11*C11 - M12*C12 + M13*C13 - M14*C14;
jTextField2.setText(String.valueOf(det));

if (det == 0) {
   JOptionPane.showMessageDialog(this, "Matriks Kunci tidak memiliki invers");
   return;
```

```
}
if (det % 2 == 0) {
   JOptionPane.showMessageDialog(this, "Matriks Kunci determinan bernilai genap, silahkan masukan matriks kunci yang determinannya bernilai ganjil");
   return;
}
if (det < 0) {
   JOptionPane.showMessageDialog(this, "Matriks Kunci determinan bernilai negatif, silahkan masukan matriks kunci yang determinannya bernilai ganjil");
   return;
}
int[][] keyMatrix = new int[4][4];
keyMatrix[0][0] = M11;
keyMatrix[0][1] = M12;
keyMatrix[0][2] = M13;
keyMatrix[0][3] = M14;
keyMatrix[1][0] = M21;
keyMatrix[1][1] = M22;
keyMatrix[1][2] = M23;
keyMatrix[1][3] = M24;
keyMatrix[2][0] = M31;
keyMatrix[2][1] = M32;
keyMatrix[2][2] = M33;
keyMatrix[2][3] = M34;
keyMatrix[3][0] = M41;
keyMatrix[3][1] = M42;
keyMatrix[3][2] = M43;
keyMatrix[3][3] = M44;
// Mengukur waktu eksekusi
    String plainText = jTextArea1.getText();
int blockSize = 4;
int remainder = plainText.length() % blockSize;
int padLength = remainder == 0 ? 0 : blockSize - remainder;
if (padLength != 0) {
   StringBuilder paddedText = new StringBuilder(plainText);
   for (int i = 0; i < padLength; i++) {
      paddedText.append("X");
   }
   plainText = paddedText.toString();
}
List<List<Integer>> asciiBlocks = new ArrayList<>();

for (int i = 0; i < plainText.length(); i += blockSize) {
   List<Integer> asciiBlock = new ArrayList<>();
   for (int j = i; j < i + blockSize; j++) {
      char c = plainText.charAt(j);
      int asciiValue = (int) c;
      asciiBlock.add(asciiValue);
   }
   asciiBlocks.add(asciiBlock);
}
StringBuilder encryptedText = new StringBuilder();

for (List<Integer> asciiBlock : asciiBlocks) {
   List<Integer> encryptedBlock = new ArrayList<>();

   for (int i = 0; i < asciiBlock.size(); i += blockSize) {
      int[] v = new int[blockSize];
      for (int j = 0; j < blockSize && (i + j) < asciiBlock.size(); j++) {
         v[j] = asciiBlock.get(i + j); }
      int[] result = new int[blockSize];
      for (int row = 0; row < blockSize; row++) {
         for (int col = 0; col < blockSize; col++) {
            result[row] += keyMatrix[row][col] * v[col];
         }
         result[row] %= 256;
      }
```

```
        for (int value : result) {
            encryptedBlock.add(value);
        }
    }
    for (int value : encryptedBlock) {
        String character = Character.toString((char) value);
        encryptedText.append(character);
    }
jTextArea2.setText(encryptedText.toString());
}
  String chipertexthill = jTextArea2.getText();
          BigInteger prime = new BigInteger(jTextField39.getText());
          BigInteger generator = new BigInteger(jTextField40.getText());
          BigInteger publicKey = new BigInteger(jTextField41.getText());
        List<BigInteger> encryptedText2 = encrypt(chipertexthill, prime, generator, publicKey);
          // Menampilkan hasil ciphertext pada textarea
          StringBuilder ciphertextOutput2 = new StringBuilder();
          //mengubah cipertext pada decimal ascii
          for (BigInteger ciphertext2 : encryptedText2) {
           ciphertextOutput2.append(ciphertext2.toString()).append(" ");
          }
          long endTime2 = System.nanoTime();
        long executionTime2 = endTime2 - startTime2;
        jTextArea2.setText(ciphertextOutput2.toString());
        jTextArea3.setText(ciphertextOutput2.toString());
        jTextField34.setText( + executionTime2 + " ns");
    }
    public List<BigInteger> encrypt(String chipertexthill, BigInteger prime, BigInteger generator, BigInteger publicKey) {
        SecureRandom secureRandom = new SecureRandom();
        List<BigInteger> encryptedText2 = new ArrayList<>();
for (int i = 0; i < chipertexthill.length(); i++) {
          char c = chipertexthill.charAt(i);
          BigInteger asciiValue = BigInteger.valueOf((int) c);
          BigInteger Modulo = new BigInteger("256");
          // Generate random value for the session key
          //BigInteger sessionKey = new BigInteger(prime.bitLength() - 1, secureRandom).mod(prime);
          BigInteger sessionKey;
    do {
    sessionKey = new BigInteger(prime.bitLength() - 1, secureRandom);
    } while (sessionKey.compareTo(BigInteger.ONE) < 0 || sessionKey.compareTo(prime.subtract(BigInteger.ONE)) > 0);

          System.out.println("Session Key: " + sessionKey);
          // Compute the ciphertext
          BigInteger ciphertext1 = generator.modPow(sessionKey, prime);
          BigInteger ciphertext2 = asciiValue.multiply(publicKey.modPow(sessionKey, prime)).mod(prime);

          encryptedText2.add(ciphertext1);
          encryptedText2.add(ciphertext2);
        }
        return encryptedText2;
```

Lampiran 11. Source Code Dekripsi Hybrid.java

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
String ciphertext = jTextArea3.getText();
BigInteger privateKey = new BigInteger(jTextField42.getText().trim());
BigInteger prime = new BigInteger(jTextField43.getText().trim());
long startTime2 = System.nanoTime();
List<BigInteger> ciphertextList = convertCiphertextToBigIntegerList(ciphertext);
String decryptedText = decrypt(ciphertextList, prime, privateKey);
// Menampilkan hasil plaintext pada textarea
jTextArea4.setText(decryptedText);
String input_1 = A_11.getText();
String input_2 = A_12.getText();
String input_3 = A_13.getText();
String input_4 = A_14.getText();
String input_5 = A_21.getText();
String input_6 = A_22.getText();
String input_7 = A_23.getText();
String input_8 = A_24.getText();
String input_9 = A_31.getText();
String input_10 = A_32.getText();
String input_11 = A_33.getText();
String input_12 = A_34.getText();
String input_13 = A_41.getText();
String input_14 = A_42.getText();
String input_15 = A_43.getText();
String input_16 = A_44.getText();
int A11, A12, A13, A14, A21, A22, A23, A24, A31, A32, A33, A34, A41, A42, A43, A44;
try {
   A11 = Integer.parseInt(input_1);
   A12 = Integer.parseInt(input_2);
   A13 = Integer.parseInt(input_3);
   A14 = Integer.parseInt(input_4);
   A21 = Integer.parseInt(input_5);
   A22 = Integer.parseInt(input_6);
   A23 = Integer.parseInt(input_7);
   A24 = Integer.parseInt(input_8);
   A31 = Integer.parseInt(input_9);
   A32 = Integer.parseInt(input_10);
   A33 = Integer.parseInt(input_11);
   A34 = Integer.parseInt(input_12);
   A41 = Integer.parseInt(input_13);
   A42 = Integer.parseInt(input_14);
   A43 = Integer.parseInt(input_15);
   A44 = Integer.parseInt(input_16);
} catch (NumberFormatException e) {
   JOptionPane.showMessageDialog(this, "Input yang Anda masukkan salah, masukkan bilangan bulat yang benar");
   return;
}
//mencari ADjoin matriks kunci
int C11 = A22 * A33 * A44 + A23 * A34 *A42 + A24 * A32 * A43 - A42 * A33 * A24 - A43 * A34 * A22 - A44 * A32 *
A23;
int C12 = A21 * A33 * A44 + A23 * A34 *A41 + A24 * A31 * A43 - A41 * A33 * A24 - A43 * A34 * A21 - A44 * A31 *
A23;
int C13 = A21 * A32 * A44 + A22 * A34 *A41 + A24 * A31 * A42 - A41 * A32 * A24 - A42 * A34 * A21 - A44 * A31 *
A22;
int C14 = A21 * A32 * A43 + A22 * A33 *A41 + A23 * A31 * A42 - A41 * A32 * A23 - A42 * A33 * A21 - A43 * A31 *
A22;
int C21 = A12 * A33 * A44 + A13 * A34 *A42 + A14 * A32 * A43 - A42 * A33 * A14 - A43 * A34 * A12 - A44 * A32 *
A13;
int C22 = A11 * A33 * A44 + A13 * A34 *A41 + A14 * A31 * A43 - A41 * A33 * A14 - A43 * A34 * A11 - A44 * A31 *
A13;
int C23 = A11 * A32 * A44 + A12 * A34 *A41 + A14 * A31 * A42 - A41 * A32 * A14 - A42 * A34 * A11 - A44 * A31 *
A12;
int C24 = A11 * A32 * A43 + A12 * A33 *A41 + A13 * A31 * A42 - A41 * A32 * A13 - A42 * A33 * A11 - A43 * A31 *
A12;
```

```
int C31 = A12 * A23 * A44 + A13 * A24 *A42 + A14 * A22 * A43 - A42 * A23 * A14 - A43 * A24 * A12 - A44 * A22 *
A13;
int C32 = A11 * A23 * A44 + A13 * A24 *A41 + A14 * A21 * A43 - A41 * A23 * A14 - A43 * A24 * A11 - A44 * A21 *
A13;
int C33 = A11 * A22 * A44 + A12 * A24 *A41 + A14 * A21 * A42 - A41 * A22 * A14 - A42 * A24 * A11 - A44 * A21 *
A12;
int C34 = A11 * A22 * A43 + A12 * A23 *A41 + A13 * A21 * A42 - A41 * A22 * A13 - A42 * A23 * A11 - A43 * A21 *
A12;
int C41 = A12 * A23 * A34 + A13 * A24 *A32 + A14 * A22 * A33 - A32 * A23 * A14 - A33 * A24 * A12 - A34 * A22 *
A13;
int C42 = A11 * A23 * A34 + A13 * A24 *A31 + A14 * A21 * A33 - A31 * A23 * A14 - A33 * A24 * A11 - A34 * A21 *
A13;
int C43 = A11 * A22 * A34 + A12 * A24 *A31 + A14 * A21 * A32 - A31 * A22 * A14 - A32 * A24 * A11 - A34 * A21 *
A12;
int C44 = A11 * A22 * A33 + A12 * A23 *A31 + A13 * A21 * A32 - A31 * A22 * A13 - A32 * A23 * A11 - A33 * A21 *
A12;
int det = A11*C11 - A12*C12 + A13*C13 - A14*C14;
jTextField1.setText(String.valueOf(det));
if (det == 0) {
   JOptionPane.showMessageDialog(this, "Matriks Kunci tidak memiliki invers");
   return;
}
if (det < 0) {
   JOptionPane.showMessageDialog(this, "Matriks Kunci determinan bernilai negatif, silahkan masukan matriks kunci
yang determinannya bernilai ganjil");
   return;
}
if (det % 2 == 0) {
   JOptionPane.showMessageDialog(this, "Matriks Kunci determinan bernilai genap, silahkan masukan matriks kunci yang
determinannya bernilai ganjil");
   return;
}
 int detInverse = -1;
for (int i = 0; i < 256; i++) {
   if ((det * i) % 256 == 1) {
      detInverse = i;
      break;
   }
}
//jTextArea4.setText(String.valueOf(detInverse));
int[][] keyMatrix = new int[4][4];
keyMatrix[0][0] = (C11 * detInverse);
keyMatrix[0][1] = (-C21 * detInverse);
keyMatrix[0][2] = (C31 * detInverse);
keyMatrix[0][3] = (-C41 * detInverse);
keyMatrix[1][0] = (-C12 * detInverse);
keyMatrix[1][1] = (C22 * detInverse);
keyMatrix[1][2] = (-C32 * detInverse);
keyMatrix[1][3] = (C42 * detInverse);
keyMatrix[2][0] = (C13 * detInverse);
keyMatrix[2][1] = (-C23 * detInverse);
keyMatrix[2][2] = (C33 * detInverse);
keyMatrix[2][3] = (-C43 * detInverse);
keyMatrix[3][0] = (-C14 * detInverse);
keyMatrix[3][1] = (C24 * detInverse);
keyMatrix[3][2] = (-C34 * detInverse);
keyMatrix[3][3] = (C44 * detInverse);
// Mengukur waktu eksekusi
String encryptedText = jTextArea4.getText();
int blockSize = 4;
List<List<Integer>> asciiBlocks = new ArrayList<>();
for (int i = 0; i < encryptedText.length(); i += blockSize) {
   List<Integer> asciiBlock = new ArrayList<>();
   for (int j = i; j < i + blockSize && j < encryptedText.length(); j++) {
      char c = encryptedText.charAt(j);
      int asciiValue = (int) c;
      asciiBlock.add(asciiValue);
```

```
    }
    asciiBlocks.add(asciiBlock);
}
StringBuilder decryptedText1 = new StringBuilder();
for (List<Integer> asciiBlock : asciiBlocks) {
    List<Integer> decryptedBlock = new ArrayList<>();

    for (int i = 0; i < asciiBlock.size(); i += blockSize) {
        int[] v = new int[blockSize];
        for (int j = 0; j < blockSize && (i + j) < asciiBlock.size(); j++) {
            v[j] = asciiBlock.get(i + j);
        }
        int[] result = new int[blockSize];
        for (int row = 0; row < blockSize; row++) {
            for (int col = 0; col < blockSize; col++) {
                result[row] += keyMatrix[row][col] * v[col];
            }
            result[row] %= 256;
            if (result[row] < 0) {
                result[row] += 256;} }
        for (int value : result) {
            decryptedBlock.add(value);
        }
    }
    for (int value : decryptedBlock) {
        String character = Character.toString((char) value);
        decryptedText1.append(character);
    }
     long endTime2 = System.nanoTime();
        long executionTime = endTime2 - startTime2;
jTextField17.setText( + executionTime + " ns");
}
jTextArea4.setText(decryptedText1.toString());
    }
// Metode untuk mengonversi ciphertext menjadi daftar BigInteger
private List<BigInteger> convertCiphertextToBigIntegerList(String ciphertext) {
    List<BigInteger> ciphertextList = new ArrayList<>();
    String[] ciphertextArray = ciphertext.split(" ");
    for (String ciphertextNum : ciphertextArray) {
        if (!ciphertextNum.isEmpty()) {
            try {
                BigInteger num = new BigInteger(ciphertextNum.trim());
                ciphertextList.add(num);
            } catch (NumberFormatException e) {
                e.printStackTrace(); } }
    }
    return ciphertextList;
}
// Metode untuk mendekripsi ciphertext
private String decrypt(List<BigInteger> ciphertextList, BigInteger prime, BigInteger privateKey) {
    long startTime1 = System.currentTimeMillis();
    StringBuilder plaintext = new StringBuilder();
    BigInteger modulo = new BigInteger("256");
    for (int i = 0; i <ciphertextList.size(); i+=2) {
        BigInteger ciphertext1 = ciphertextList.get(i);
        BigInteger ciphertext2 = ciphertextList.get(i + 1);
        // Mendapatkan nilai sessionKey dengan menggunakan kunci privat
        BigInteger exponent = prime.subtract(BigInteger.ONE).subtract(privateKey);
        BigInteger plaintextValue = ciphertext2.multiply(ciphertext1.modPow(exponent, prime)).mod(prime);
        // Mendapatkan nilai plaintext dengan menggunakan sessionKey
        // Mendapatkan karakter plaintext dari nilai ASCII yang didekripsi
        char character = (char) plaintextValue.intValue();
        // Menambahkan karakter plaintext ke StringBuilder
        plaintext.append(character);
       }
    return plaintext.toString();
    }
```

Lampiran 12. Source Code Menghitung Nilai Entropi

```
package javaapplication4;
import java.util.HashMap;
import java.util.Map;

/**
 *
 * @author LENOVO
 */
public class JavaApplication4 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        String ciphertext = "68 1 30 36 92 164 241 160 249 190 234 97 241 201 140 126 4 86 184 28 222 43 32 73 30 10 4
204 34 52 140 21 73 53 17 246.";
        double entropy = calculateEntropy(ciphertext);
        System.out.println("Entropy: " + entropy);
    }

    public static double calculateEntropy(String ciphertext) {
        String[] numbers = ciphertext.split(" ");
        int totalNumbers = numbers.length;
        Map<String, Integer> frequencyMap = new HashMap<>();

        // Hitung frekuensi masing-masing angka
        for (String number : numbers) {
            frequencyMap.put(number, frequencyMap.getOrDefault(number, 0) + 1);
        }
    double entropy = 0.0;
     // Hitung entropi berdasarkan frekuensi
        for (Map.Entry<String, Integer> entry : frequencyMap.entrySet()) {
            double probability = (double) entry.getValue() / totalNumbers;
            entropy -= probability * (Math.log(probability) / Math.log(2));
        }

        return entropy;
    }
}
```

Lampiran 13. Source Code Memeriksa Kesamaan Ciphertext Menggunakan Aplikasi dan Perhitungan Manual

```
package javaapplication5;

/**
 *
 * @author LENOVO
 */
public class JavaApplication5 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        String CPProgram = "160 151 7 176 65 136 136 93 151 232 139 20 136 31 200 "
                + "121 122 238 125 13 177 105 17 146 36 114 14 246 223 90 255 192 10 115 6 17 65 85 102 19 42 "
                + "227 172 230 176 235 184 243 170 1 124 242 108 223 144 183 87 11 116 151 130 154 11 104 208 "
                + "166 58 196 177 162 195 205 23 207 212 133 108 173 83 101 45 207 160 91 90 247 117 25";
        String CPManual = "160 151 7 176 65 136 136 93 151 232 139 20 136 31 200 121 122 238 125 13 177 105 17 "
                + "146 36 114 14 246 223 90 255 192 10 115 6 17 65 85 102 19 42 227 172 230 176 235 184 243 170 "
                + "1 124 242 108 223 144 183 87 11 116 151 130 154 11 104 208 166 58 196 177 162 195 205 23 207 "
                + "212 133 108 173 83 101 45 207 160 91 90 247 117 25";

    if (CPProgram == CPManual)
        System.out.println("Sama");
        else
        System.out.println("tidak sama");
```

Lampiran 14. Riwayat Hidup

RIWAYAT HIDUP

Faizar Rusyadi lahir di Negara pada tanggal 29 Maret 2000. Penulis lahir dari pasangan suami istri Bapak Nur Muhammad dan Ibu Misriati. Penulis berkebangsaan Indonesia dan beragama Islam. Kini penulis beralamat di Jalan Gilimanuk-Denpasar Km.23 Desa Banyubiru, Penulis menyelesaikan pendidikan dasar di SD Negeri 4 Banyubiru dan lulus pada tahun 2012. Kemudian penulis melanjutkan di SMP Negeri 4 Negara dan lulus pada tahun 2015. Pada tahun 2018, penulis lulus dari MAN Negara yang sekarang berganti nama menjadi MAN 1 Jembrana jurusan Ilmu Pengetahuan Alam dan melanjutkan ke Strata I prodi Matematika Jurusan Matematika di Universitas Pendidikan Ganesha. Pada semester pertengahan tahun 2024 penulis telah menyelesaikan Tugas Akhir yang berjudul "Kriptografi Hybrid Metode Hill Cipher dan ElGamal dalam Meningkatkan Keamanan Pesan Rahasia". Selanjutnya, mulai tahun 2018 sampai dengan penulisan skripsi ini, penulis masih terdaftar sebagai mahasiswa Program S1 Matematika di Universitas Pendidikan Ganesha.