Jurnal Teknik Informatika Vol. 15 No. 1, April 2022 (44-51)
ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v15i1.25735

JURNAL TEKNIK INFORMATIKA

Homepage : http://journal.uinjkt.ac.id/index.php/ti

Implementation of The Advanced Encryption Standard (AES) Algorithm for Digital Image Security

Angga Aditya Permana[1], Luigi Ajeng Pratiwi[2]

[1]Informatic, Faculty of Information & Technology
[2]Technical Information, Faculty of Science & Technology
[1]University of Multimedia Nusantara
[2]Korea Advanced Institute of Science & Technology (KAIST), Daejeon
[1]Jl. Boulevard, Gading Serpong, Kel. Curug Sangereng, Kec. Kelapa Dua, Kab. Tangerang, Banten
E-mail: [1]angga.permana@umn.ac.id, [2]luigi.ajeng@kaist.ac.kr

Article:
Accepted: April 26, 2022
Revised: May 28, 2022
Issued: June 15, 2022

***Correspondence Address:**
angga.permana@umn.ac.id

ABSTRACT

Nowadays, technological advances have made it increasingly easy to obtain information, especially image data (digital images). Digital image is an interesting thing to look for information. So that misuse of data can be done for personal or public interests. Misuse of data can be avoided by adding data security systems. Cryptography is the science of securing data. Cryptography can be done using the AES (Advanced Encryption Standard) algorithm, which is an algorithm that utilizes symmetric keys. Testing is done by entering the same key in the encryption and decryption process. Encryption is the process of encoding plaintext (original text) into ciphertext (text that has been encoded). While decryption is the process of recovering the plaintext from the ciphertext. Therefore, data security is an important thing to do. This study aims to find out how encryption and decryption on the AES algorithm can be used to secure digital data. The results of this study indicate that the encryption and decryption process on the AES algorithm was successfully carried out so that it can be used to secure data on digital images.

Keywords: *Algorithm; Cryptography; AES; Digital Image; Encryption; Decryption;*

I. INTRODUCTION

Advances in technology make it easier to get information on various kinds of data, especially image data (digital images). Digital image is one of the interesting data to look for information. Through advances in information technology, personal data can be easily known by the general public. This causes a decrease in the level of privacy of a person where privacy is something that should not be known by the general public.

Digital images that are private can become public if someone manages to hack the security of the digital image. Misuse of data can be done by other parties if the data is successfully hacked. This of course violates the privacy policy and causes harm to the owner. For this reason, maintaining data security is an important thing to implement. With increasing data security, a person's level of privacy becomes more secure[1][2].

The thing that can be done to secure the digital image data is to disguise the data. With the disguise of data, the other party will not know the original image of the disguised image. Data disguise can be done with cryptography. Where cryptography [3][4][5] itself is a data security technique by disguising the data. One of the cryptographic algorithms that can be used is the AES algorithm[6][7][8], which is a symmetric ciphertext block that can perform encryption and decryption. Therefore, in this study the authors conducted research to create cryptographic applications using the AES algorithm to secure data on digital images.

II. METHODOLOGY

The system development method used in this research is the Systems development life cycle (SDLC) method by analyzing various literature studies. The method of developing this system is as follows.

Analysis
Evaluation
Testing
Implementation
Design
SDLC

Figure 1. SDLC method

Analysis

Rapid technological advances make it easier for people to exchange information with each other. Data is getting easier to disseminate. However, not everyone can maintain data security. Misuse of data can be done for personal interests that are detrimental to the owner of the original data. Therefore, making data security applications is very necessary.

Design

The design of the system application is based on the literature study that has been carried out. The design of this application consists of a main menu for encryption and decryption using the Java language in the NetBeans IDE 8.2 application.

Implementation

Implementation is in the form of an application made based on literature study and design in the previous stage.

Testing

The trial is carried out before the application is ready for use by the user. Based on the results of testing the application, its shortcomings will be evaluated for further improvement and ready to be used by users.

Evaluation

Applications that have been tested will be analyzed to get the desired results.

The algorithm used in this study is the Advanced Encryption Standard (AES) [9][10] algorithm. AES is a symmetric block cipher cryptography algorithm where the same key is required for encryption and decryption. The AES algorithm is a development of the DES algorithm whose validity period is considered over due to security factors. There are three types of key length in the AES algorithm, namely 128 bits, 192 bits, and 256 bits. Each key length will affect the number of rounds in the AES algorithm. There are two processes in the AES algorithm, namely the encryption and decryption process.

Encryption, the encryption process starts with the input, the input that has been copied in the state will undergo an AddRoundKey transformation. Then, the state will transform SubBytes, ShiftRows, Mixcolomns, and AddRoundKey repeatedly according to the key length used in the AES algorithm. The process is called a round function. The last round is different from the previous rounds because it does not undergo a Mixcolomns transformation.

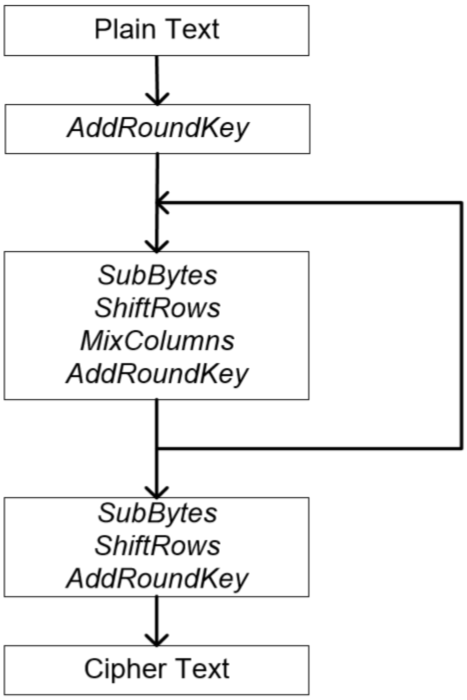


Figure 2. Flowchart Encryption AES

There are four types of byte transformation in the AES algorithm, namely AddRoundKey, SubBytes, ShiftRows, and Mixcolomns.

1. AddRoundKey is a transformation in AES that functions to combine state arrays and round keys with XOR operations.

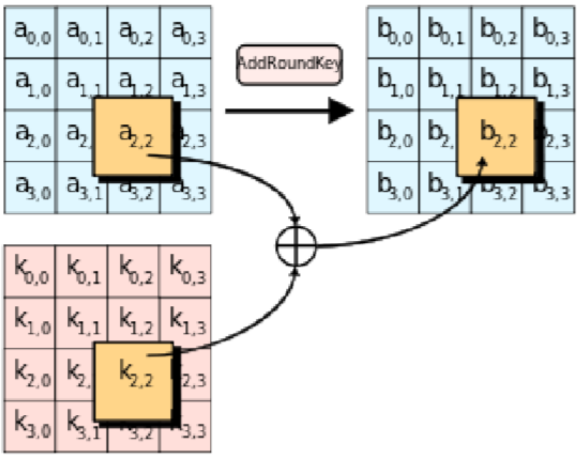
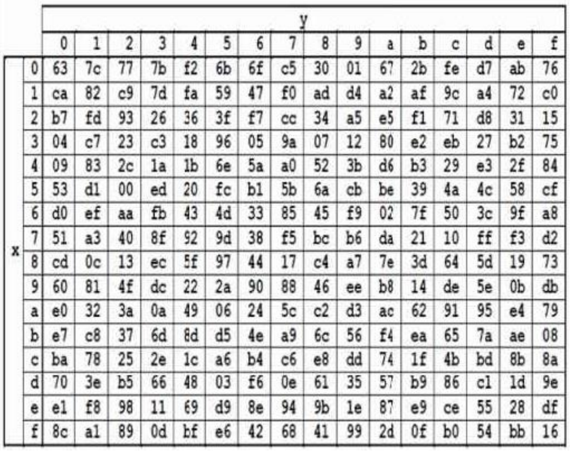

$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
AddRoundKey
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$
$k_{0,0}$ $k_{0,1}$ $k_{0,2}$ $k_{0,3}$
$k_{1,0}$ $k_{1,1}$ $k_{1,2}$ $k_{1,3}$
$k_{2,0}$ $k_{2,2}$ $k_{2,3}$
$k_{3,0}$ $k_{3,1}$ $k_{3,2}$ $k_{3,3}$

Figure 3. AddRoundKey Transform

2. SubBytes is a transformation in AES that functions to exchange the contents of the bytes using the substitution table (S-Box).

Table 1. S-Box SubBytes



		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

For each byte in the state array, for example S[r,c] = xy, in this case xy is the hexadecimal digit of the value S[r,c], then the substitution value is represented by S[r,c] is the element in the substitution table that is the intersection of row x and column y.

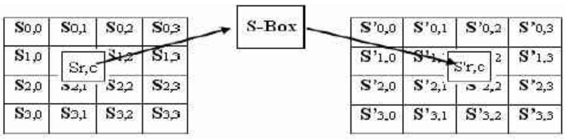

$S_{0,0}$ $S_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,0}$ $S_{1,2}$ $S_{1,3}$
$S_{r,c}$
$S_{2,0}$ $S_{2,1}$ $S_{2,2}$ $S_{2,3}$
$S_{3,0}$ $S_{3,1}$ $S_{3,2}$ $S_{3,3}$
S-Box
$S'_{0,0}$ $S'_{0,1}$ $S'_{0,2}$ $S'_{0,3}$
$S'_{1,0}$ $S'_{1,3}$
$S'_{r,c}$
$S'_{2,0}$ $S'_{2,1}$ $S'_{2,2}$ $S'_{2,3}$
$S'_{3,0}$ $S'_{3,1}$ $S'_{3,2}$ $S'_{3,3}$

Figure 4. Effect of Mapping on Each Byte in State

3. ShiftRows is the process of shifting blocks per line in the state array where the leftmost bit will be moved to the rightmost bit.

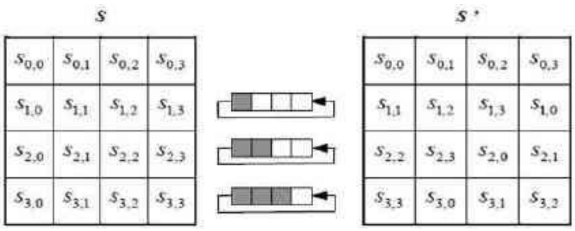

s
s'
$S_{0,0}$ $S_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,0}$ $S_{1,1}$ $S_{1,2}$ $S_{1,3}$
$S_{2,0}$ $S_{2,1}$ $S_{2,2}$ $S_{2,3}$
$S_{3,0}$ $S_{3,1}$ $S_{3,2}$ $S_{3,3}$
$S_{0,0}$ $S_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,1}$ $S_{1,2}$ $S_{1,3}$ $S_{1,0}$
$S_{2,2}$ $S_{2,3}$ $S_{2,0}$ $S_{2,1}$
$S_{3,3}$ $S_{3,0}$ $S_{3,1}$ $S_{3,2}$

Figure 5. ShiftRows Transform

4. Mixcolumns is the process of transferring data blocks in each state array. The Mixcolomns transformation can be seen in the following matrix multiplication.

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 03 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix} \quad \ldots(1)$$

The result of the matrix multiplication above can be considered as the product below.

$$S'(x) = a(x) \oplus s(x) \qquad \ldots(2)$$

$$S'_{0,c} = (\{02\}.S_{0,c}) \oplus (\{03\}.S_{1,c}) \oplus S_{2,c} \oplus S_{3,c}$$

$$S'_{1,c} = S_{0,c} \oplus (\{02\}.S_{1,c}) \oplus (\{03\}.S_{2,c}) \oplus S_{3,c}$$

$$S'_{2,c} = S_{0,c} \oplus S_{1,c} \oplus (\{02\}.S_{1,c}) \oplus (\{03\}.S_{3,c})$$

$$S'_{3,c} = (\{03\}.S_{0,c}) \oplus S_{0,c} \oplus S_{1,c} \oplus (\{02\}.S_{3,c})$$

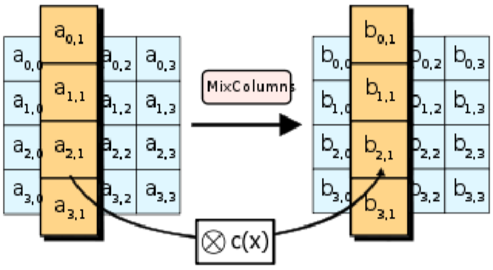


Figure 6. Mixcolumns Transformation

Decryption

The description process is done by doing an inverse of the encrypted transformation. The decryption process aims to restore data to its original form.

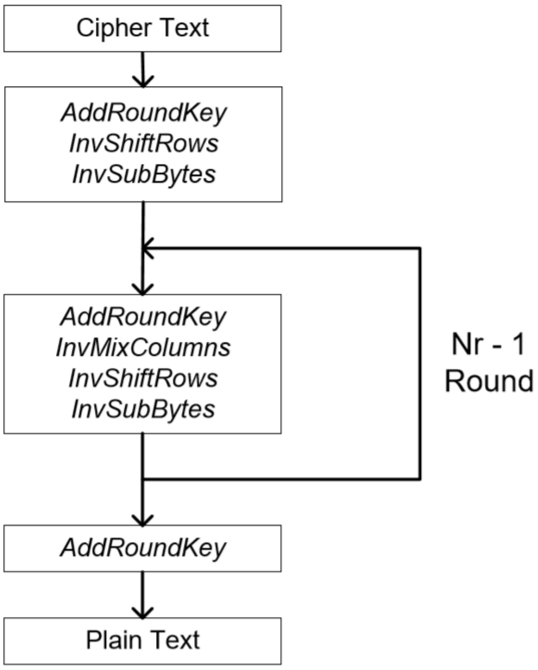


Figure 7. Flowchart Decryption AES

The block cipher byte transformations performed are InvShiftRows, InvSubBytes, InvMixClomns, and AddRoundKey.

1. InvShiftRows is doing a bit shift to the left on each line block.

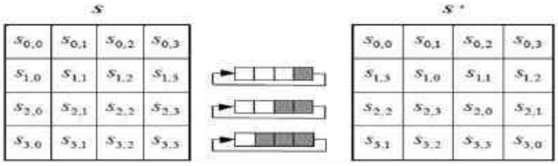
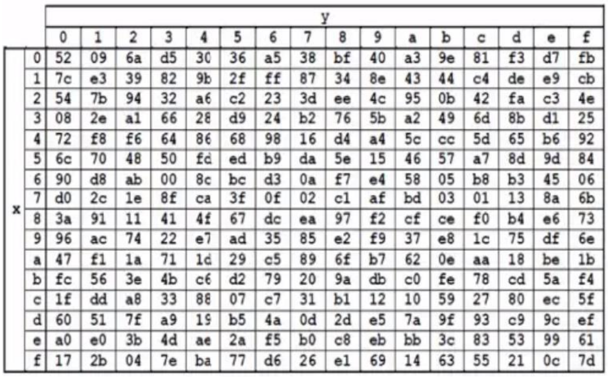


Figure 8. InvShiftRows Transform

2. InvSubBytes is each element in the state mapped by the Inverse S-Box table.

Table 2. InvSubBytes



		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

3. InvMixColumns is each column in the state multiplied by the AES matrix.

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix} \ldots(3)$$

The result of the matrix multiplication above can be considered as the product below.

$$S'_{0,c} = (\{0E\}.S_{0,c}) \oplus (\{0B\}.S_{1,c}) \oplus (\{0D\}.S_{2,c}) \oplus (\{09\}.S_{3,c})$$

$$S'_{1,c} = (\{09\}.S_{0,c}) \oplus (\{0E\}.S_{1,c}) \oplus (\{0B\}.S_{2,c}) \oplus (\{0D\}.S_{3,c})$$

$$S'_{2,c} = (\{0D\}.S_{0,c}) \oplus (\{09\}.S_{1,c}) \oplus (\{0E\}.S_{2,c}) \oplus (\{0B\}.S_{3,c})$$

$$S'_{3,c} = (\{0B\}.S_{0,c}) \oplus (\{0D\}.S_{1,c}) \oplus (\{09\}.S_{2,c}) \oplus (\{0E\}.S_{3,c})$$

… (4)

4. AddRoundKey combines state arrays and round keys with an XOR relationship.

III. RESULTS AND DISCUSSION

3.1 Software Requirements Analysis

Rapid technological advances make it easier for people to exchange information with each other. Today's society is very easy to get the information they want. Therefore, a data security application is needed to keep data safe when exchanging information.

3.2 Problem Identification

The rapid development of technology makes it easy for us to get the information we want. We can easily share images via the internet. However, when sharing images we must also be able to distinguish between personal data and public data. Because currently there are many cases of data misuse. Data on the internet can be easily misused by irresponsible parties for their personal interests. Therefore, a data security system is very necessary, one of the data security methods is the AES (Advanced Encryption Standard) cryptographic algorithm.

3.3 Analysis

This application is an implementation of the AES algorithm. AES is a cryptographic algorithm for securing data. This algorithm requires the same key for the encryption and decryption process. For this reason, the data will not be opened if the key used is different at the time of encryption and decryption. This will make the data more secure and cannot be opened by others because only those who know the key can open it. Making this application using a software development application, namely NetBeans IDE 8.2 using the Java language.

3.4 Design

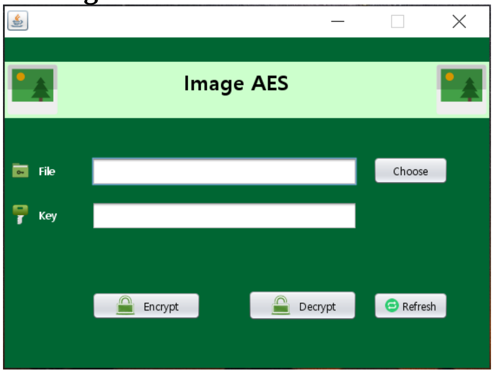


Figure 8. User Interface

In this main menu the user can select a file then enter the key. The key length that can be entered is from 1 to 16 characters. Then the user selects the action to be performed, namely encrypt, decrypt or refresh.

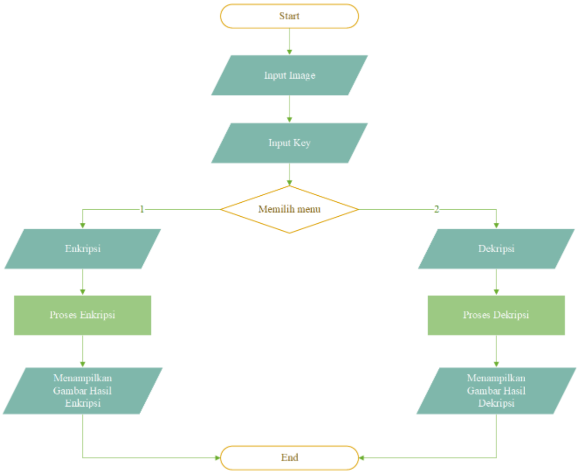
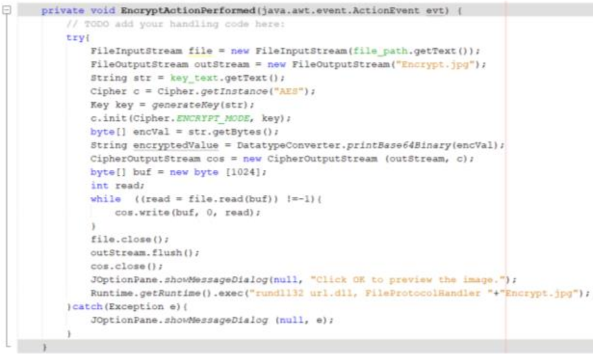
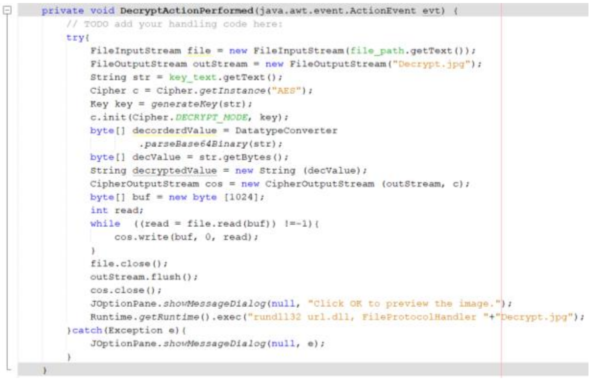


Figure 9. Program Structure Design



```
private void EncryptActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try{
        FileInputStream file = new FileInputStream(file_path.getText());
        FileOutputStream outStream = new FileOutputStream("Encrypt.jpg");
        String str = key_text.getText();
        Cipher c = Cipher.getInstance("AES");
        Key key = generateKey(str);
        c.init(Cipher.ENCRYPT_MODE, key);
        byte[] encVal = str.getBytes();
        String encryptedValue = DatatypeConverter.printBase64Binary(encVal);
        CipherOutputStream cos = new CipherOutputStream (outStream, c);
        byte[] buf = new byte [1024];
        int read;
        while  ((read = file.read(buf)) !=-1){
            cos.write(buf, 0, read);
        }
        file.close();
        outStream.flush();
        cos.close();
        JOptionPane.showMessageDialog(null, "Click OK to preview the image.");
        Runtime.getRuntime().exec("rundll32 url.dll, FileProtocolHandler "+"Encrypt.jpg");
    }catch(Exception e){
        JOptionPane.showMessageDialog (null, e);
    }
}
```

Figure 10. Encryption

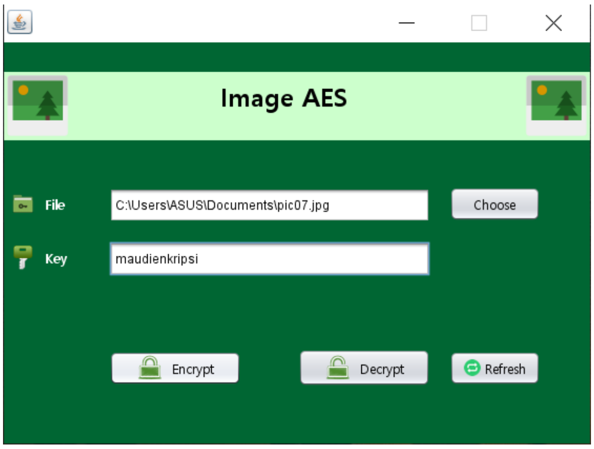


```
private void DecryptActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try{
        FileInputStream file = new FileInputStream(file_path.getText());
        FileOutputStream outStream = new FileOutputStream("Decrypt.jpg");
        String str = key_text.getText();
        Cipher c = Cipher.getInstance("AES");
        Key key = generateKey(str);
        c.init(Cipher.DECRYPT_MODE, key);
        byte[] decorderdValue = DatatypeConverter
                .parseBase64Binary(str);
        byte[] decValue = str.getBytes();
        String decryptedValue = new String (decValue);
        CipherOutputStream cos = new CipherOutputStream (outStream, c);
        byte[] buf = new byte [1024];
        int read;
        while  ((read = file.read(buf)) !=-1){
            cos.write(buf, 0, read);
        }
        file.close();
        outStream.flush();
        cos.close();
        JOptionPane.showMessageDialog(null, "Click OK to preview the image.");
        Runtime.getRuntime().exec("rundll32 url.dll, FileProtocolHandler "+"Decrypt.jpg");
    }catch(Exception e){
        JOptionPane.showMessageDialog(null, e);
    }
}
```

Figure 11. Decryption

3.5 Implementation

The implementation of the AES cryptographic algorithm was made using the NetBeans IDE 8.2 application. This application consists of encryption and decryption of image data.



File: C:\Users\ASUS\Documents\pic07.jpg

Key: maudienkripsi

Figure 12. Image Encryption Main View

In this screen, the user is asked to enter an image to be encrypted and then enter a key for encryption. After that the system will display the results of the encryption.

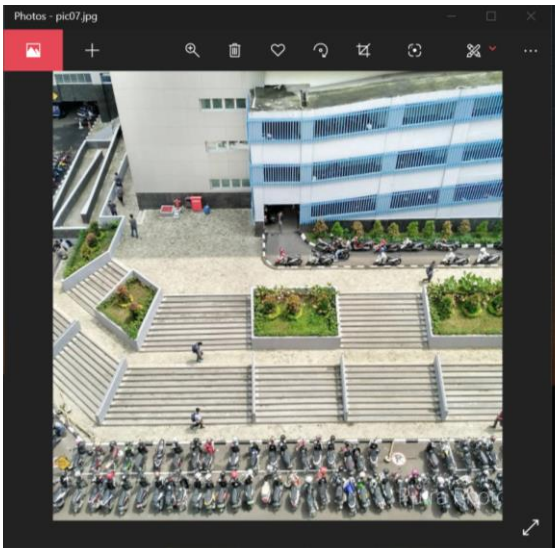
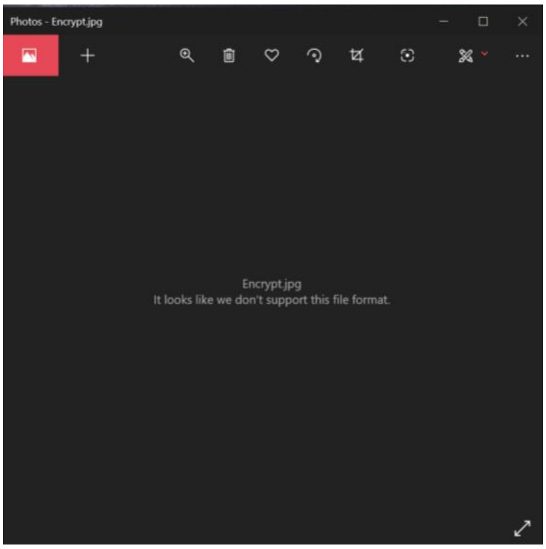
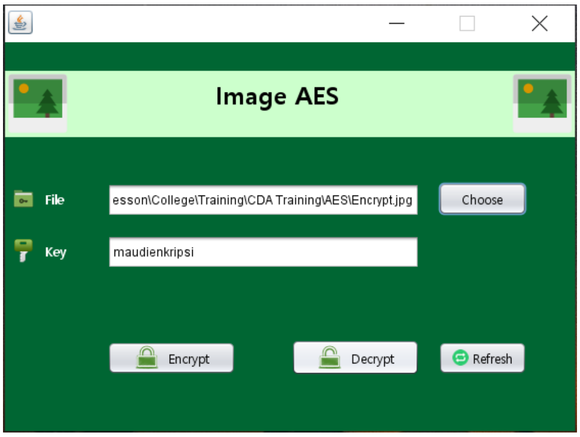


Figure 13. Files to be encrypted



Encrypt.jpg
It looks like we don't support this file format.

Figure 14. Encryption Results



File: esson\College\Training\CDA Training\AES\Encrypt.jpg

Key: maudienkripsi

Figure 15. Image Decryption Main View

In this view, the user is asked to enter an image to be decrypted and then enter the same key as the key at the time of encryption. After that the system will display the results of the decryption.

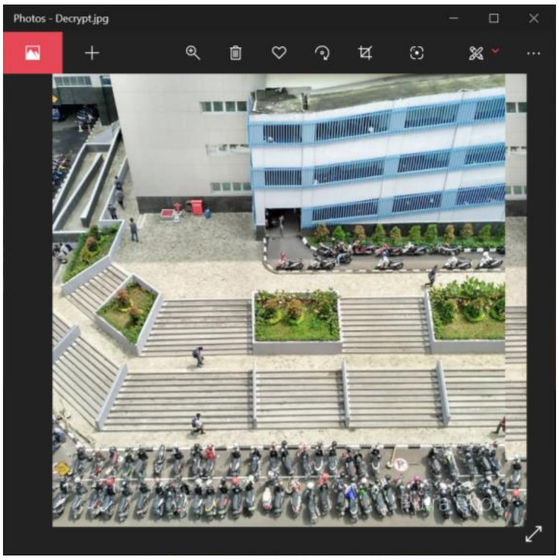


Figure 16. Decryption Result

Black Box Testing
This test is carried out to test the functional application that has been made.

Table 3. Black Box Testing

No.	Test Case	Scenario	Expected results	Information
1.	Insert image	The user selects an image from the computer as input	The system can accept image input	Valid
2.	Insert key	The user enters the key as the encryption and decryption key	The key can be used for both encryption and decryption	Valid
3.	Encrypt	Encryption with user-entered key	The system can perform encryption well	Valid
4.	Decrypt	Decrypt with user-entered key	The system can decrypt well	Valid
5.	Save image	Save the results of encryption and decryption	The system can save images	Valid
6.	Refresh	Users can delete the contents of the file and key fields	The system can refresh well	Valid

IV. CONCLUSION

The implementation of the AES (Advanced Encrypted Standard) algorithm for the encryption and decryption process in this application was successfully carried out using the NetBeans IDE 8.2 application. The same key in the encryption and decryption process will determine the success of the process. Image file format extensions that can be encrypted include .jpeg, .png, .ico, .bmp, .cur, and .gif. The original image file size is the same as the encrypted and decrypted image. This black box test shows that this application can be used properly. For further development, it is hoped that readers will be more varied in entering keys and can also add features to this application and add password characters for encryption and decryption to make it more secure.

ACKNOWLEDGEMENT

This research can be carried out properly thanks to the assistance of various parties, for that the researchers would like to thank Multimedia Nusantara University and Syarif Hidayatullah State Islamic University for the support and assistance that has been given during the collaborative process of writing this article.

BIBLIOGRAPHY

[1] A. D. Hidayat and I. Afrianto, “Sistem Kriptografi Citra Digital Pada Jaringan Intranet Menggunakan Metode Kombinasi Chaos Map Dan Teknik Selektif,” *J. Ultim.*, vol. 9, no. 1, pp. 59–66, 2017, doi: 10.31937/ti.v9i1.565.

[2] Taronisokhizebua1, “Pengamanancitradigitalberdasarkan Modifikasialgoritmarc4,” vol. 4, no. 4, 2017, doi: 10.25126/jtiik.201744474.

[3] N. M. S. Iswari, “Key generation algorithm design combination of RSA and ElGamal algorithm,” *Proc. 2016 8th Int. Conf. Inf. Technol. Electr. Eng. Empower. Technol. Better Futur. ICITEE 2016*, 2017, doi: 10.1109/ICITEED.2016.7863255.

[4] Rinaldi Munir, “Pengenalan Kriptografi dan Steganografi untuk Keamanan Informasi,” *Inst. Teknol. Bandung*, p. 53.

[5] A. Hidayatullah, Entik Insanudin, MT, “Pengenalan Kriptografi Dan Pemakaianya Sehari-Hari,” *Kriptografi*, no. May. pp. 1–7, 2016.

[6] A. A. Permana, “Penerapan Kriptografi Pada Teks Pesan dengan Menggunakan Metode Vigenere Cipher Berbasis Android,” *J. Al-AZHAR Indones. SERI SAINS DAN Teknol.*, vol. 4, no. 3, p. 110, 2018, doi: 10.36722/sst.v4i3.280.

[7] D. Nurnaningsih and A. A. Permana, “Rancangan Aplikasi Pengamanan Data Dengan Algoritma Advanced Encyption Standard (Aes),” *J. Tek. Inform.*, vol. 11, no. 2, pp. 177–186, 2018, doi: 10.15408/jti.v11i2.7811.

[8] N. Sofian, A. Wicaksana, and S. Hansun, “LSB steganography and AES encryption for multiple PDF documents,” *Proc. 2019 5th Int. Conf. New Media Stud. CONMEDIA 2019*, pp. 100–105, 2019, doi:

10.1109/CONMEDIA46929.2019.8981842.

[9] M. A. Amarullah, dan Andi Suprianto, and M. P. Enggineer Pamapersada Nusantara, “Penggunaan Algorithma AES-RIJNDAEL Pada Sistem Enkripsi Dan Dekripsi Untuk Komunikasi Data Implementasi of AES-Rijndael in Encrytion and Decryption System for Data Communication,” vol. 25, no. 2, pp. 31–39, 2015.

[10] Henry, A. H. Kridalaksana, and Z. Arifin, “Kriptografi Aes Mode Cbc Pada Citra Digital Berbasis Android,” *Semin. Ilmu Komput. dan Teknol. Inf.*, vol. 1, no. 1, pp. 45–52, 2016.