IMPLEMENTATION OF CRYPTOGRAPHY AND STEGANOGRAPHY FOR TEXT ON COVER IMAGE USING AES AND F5 ALGORITHM

Ratna Astuti Nugrahaeni[1], R. Rumani M.[2], Surya Michrandi Nasution[3]

[1,2,3]School of Electrical Engineering, Telkom University
[1]ratnaan@gmail.com, [2]rumani@telkomuniversity.ac.id, [3]michrandi@telkomuniversity.ac.id

Abstract

This journal explains about implementation that combine both cryptography and steganography method for texton cover image to increase the security level. Text will be encrypted with AES algorithm, and then it will be embedded to the cover image using F5 algorithm. The implemented AES algorithm has a good performance, with Avalanche Effect value ranges from 0.43 – 0.59. The resulting image, or stego image, has a very similar histogram with the original image, so there is no significant difference between the two of them. However, the file size change about 1.25 – 3.25 times larger than theoriginal image. If noise or disruption is given to stego image, the information can not be extracted.

Keywords: cryptography, steganography, AES, F5

1. Introduction

Communication has been used since a long time ago, either to exchange information or just simply communicating. Everyone who communicate might have different needs, and sometimes they want to communicate credential information. A cryptography method can be used to encrypt the information using a specific key so that it will not be easily interrupted by third party. However, this method is very flashy, thus may arouse suspiciousness from everyone who sees it.

To solve this problem, there is a steganography method which use a cover media to hide the information.The media can be image, audio, or even a video file. This method is made so that when other people see it, they will not realize that there is an information hidden inside.

The writer made an implementation to make the steganography method more secure, by using encryption process before embedding information to the cover image. The cryptography algorithm is AES-128, and the steganography algorithm is F5.

2. AES Cryptography Algorithm

AES or *Advanced Encryption Standard* is asymmetric algorithm. This standard can be used with three key lengths: 128 bits, 192 bits, and 256 bits. The block size of this algorithm is 128 bits, that can be seen on Figure 1. AES is the first standard to be approved by NSA (*National Security Agency*) for secret information changing [6].

The key length will determine the number of transformation cycle from plain text into cipher text, as seen below:

a. 10 cycle for 128 bit key length;
b. 12 cycle for 192 bit key length;
c. 14 cycle for 256 bit key length;

Generally, the algorithm consists of these steps:

a. Initial Round, doing Add Round Key, which is doing an XOR process between the plaintext and cipher key.
b. Cycle of sub-processes: Sub Bytes, substituting the data with S-Box; Shift Rows, shifting the data by rows; Mix Columns, scrambling the data on each array state; and Add Round Key, XOR the data with cipher key.
c. Final Round, which consists of: Sub Bytes, Shift Rows, and Add Round Key.

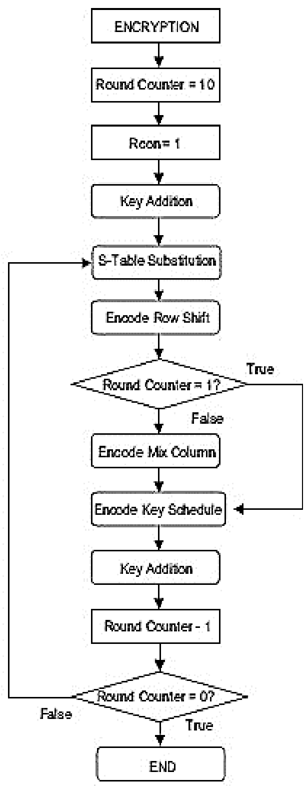


Figure 1. Encryption Process of AES-128 [8]

3. Steganography

Steganography is a technique to hide the informationon a cover media, which can be image, audio, or video file [4].This methods is used so that beside the sender and the recipient, no one will know the existence of the information.

There are some criteria on steganography:

a. Imperceptibility, the existence of information can not be seen visually.
b. Fidelity, the quality of cover media does not change significantly.
c. Recovery, the embedded information can be extracted Steganography method uses a cover media and the information to be hide, or hidden text [7].

The process of steganography can be seen on the block diagram on Figure 2.

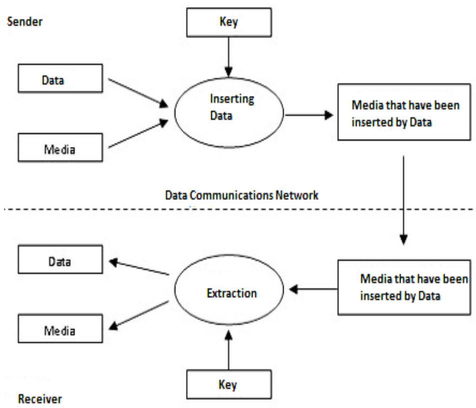


Figure 2. Block Diagram of Steganography Process [2]

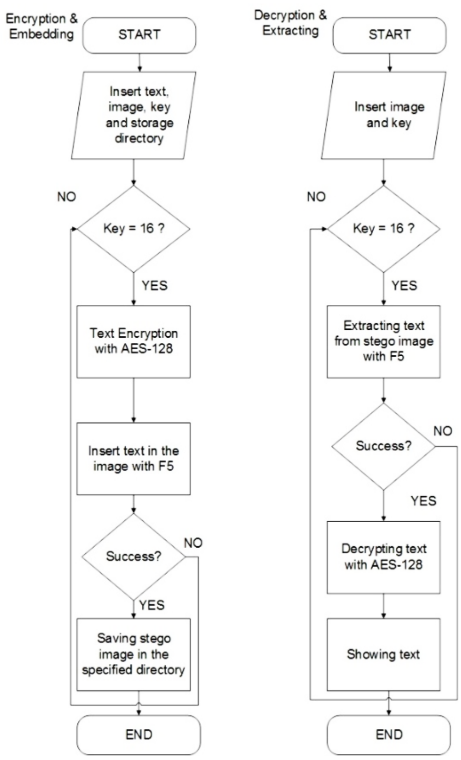


Figure 3. Flowchart of the System

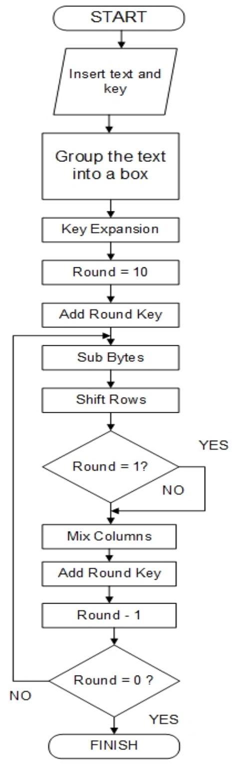


Figure 4. Flowchart of AES-128 Encryption Process

4. F5 Steganography Algorithm

The use of image as the cover media on steganography process has a limited capacity. To prevent a detection or attack, the information has to be embedded evenly on the cover.

F5 algorithm uses permutative straddling, which randomize the coefficient by permutation, and then embed the information. Permutation depends on the key derivated from the password. The coefficients will then be processed by Huffman Coder, so that the authorized recipient will be able to redo the permutation and get the information. F5 algorithm also uses matrix encoding, so that the embedding process can be more efficient.

5. System Design

If the implementation is finished, the system will have these specifications:

a. Can do cryptography process, either encryption or decryption on the text inputted by the user;
b. Can do steganography process, either to embed or to extract the information from the inputted stego image.

Implementation is done by making an application where the user can specified the cover image, text to be hidden, and the key. The output is a stego image which has a good performance; which can be seen by calculating the MSE (Mean Square Error) and PSNR (Peak Signal to Noise Ratio) value.

The parameter used to see the stego image performance is it’s resistance of Salt and Pepper noise, and disruption like cropping and compression process. The MSE shows the average square error, and PSNR shows the comparison between the maximum signal value and the noise affecting the signal. The MSE and PSNR equations are as shown below:

$$MSE = \frac{1}{mn}\sum_{i}^{m}\sum_{j}^{n} \| I(i,j) - K(i,j) \|^2 \tag{1}$$

$$PSNR = 10 \cdot \log\left(\frac{MAX_I^2}{MSE}\right) = 20 \cdot \log\left(\frac{MAX_I}{\sqrt{MSE}}\right) \tag{2}$$

The design includes flowchart of the system, system modelling, and the interface design. The system in general can be seen in Figure 3. The encryption process of AES-128 algorithm can be seen in Figure 4 and Figure 5, meanwhile the decryption process of AES-128 can be seen in Figure 6 and the extraction process of F5 algorithm can be seen in Figure 7.

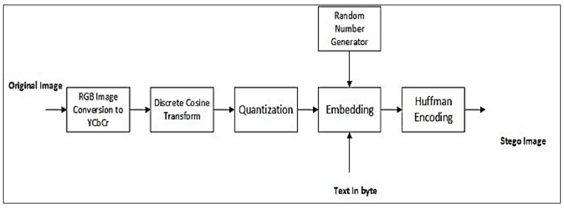


Figure 5. Diagram of F5 Embedding Process

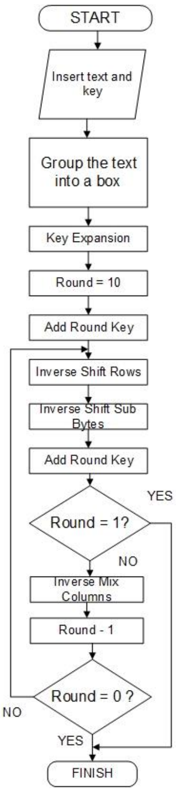


Figure 6. Flowchart of AES-128 Decryption Process

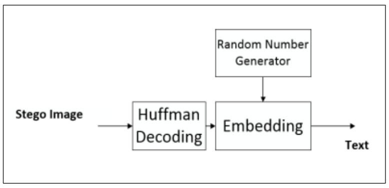


Figure 7. Block Diagram of F5 Extraction Process

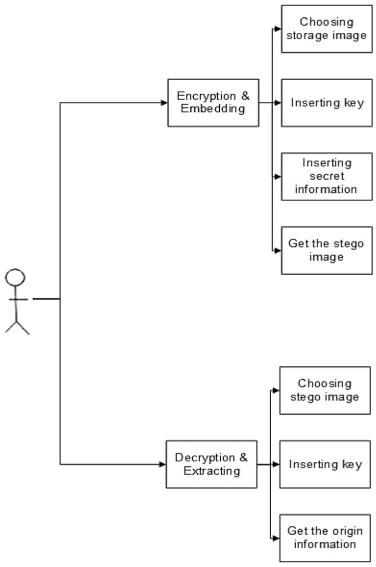


Figure 8. Activity Diagram

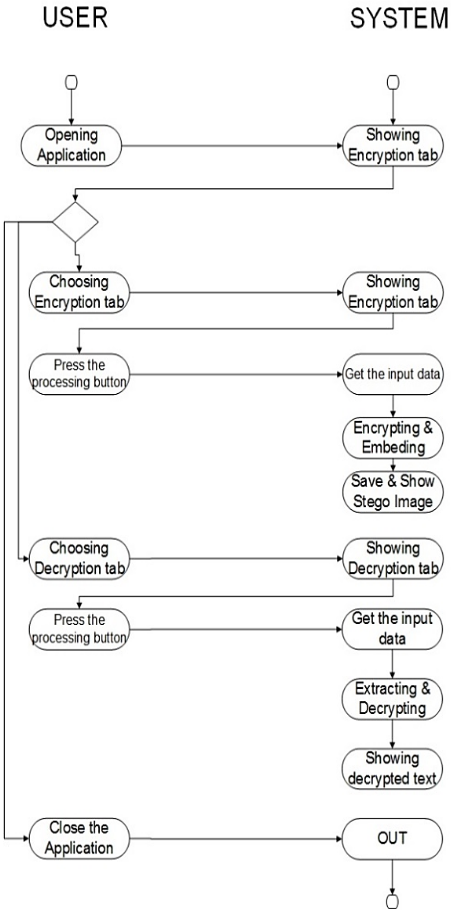


Figure 9. Class Diagram

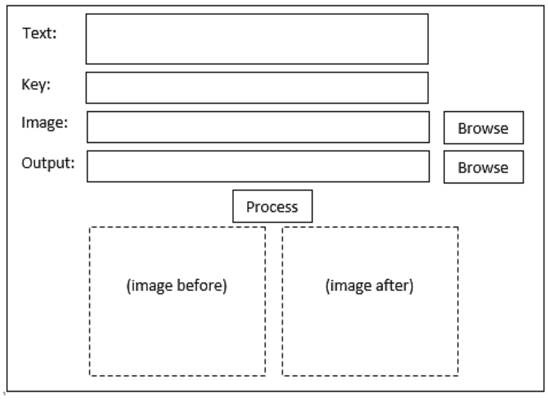


Figure 10. Interface Design of *Encryption and Embedding*

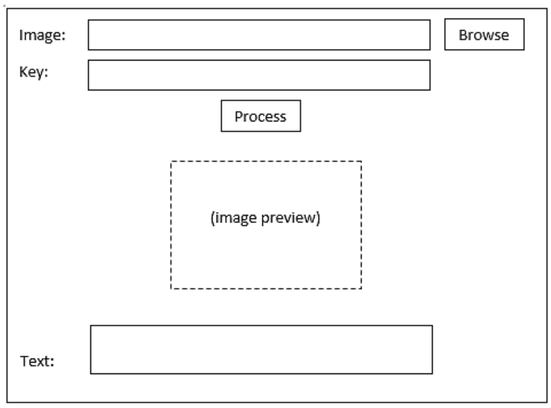


Figure 11. Interface Design of *Extraction and Decrypting*

Table 1. Encryption and Embedding Testing

Case and Result (Normal Data)	
Data Input	*Text*: "program studi sistem komputer angkatan 2009"; *Key*: "programstudicesk" (16 karakter); *Image*: orchid.jpg; *Output*: (saving directory)
Expected Result	After inputting the text, key, image, and output directory, theuser can start the process by clicking the Process button. Theprocess will start by encrypting the text and embedding it intothe cover image. If the text is not more than 500 characters, andthe key is exactly 16 characters, the stego image will be savedto the output directory.
Observation	Stego image is saved into the output directory with a goodquality.
Conclusion	Accepted.
Case and Result (Wrong Data)	
Data Input	*Text*: (none); *Key*: "programstudicesk"; *Image*: orchid.jpg; *Output*: (output directory)
Expected Result	Showing error message.
Observation	The process can not be done because the text to be embeddedhas not been specified. An error message "Text is empty!"occurred.
Conclusion	Accepted.

6. System Modelling

The application is made in one package, using 10 classes as seen in Figure 9. Based on the input and output requirements, the interface design can be seen in Figure 10 and Figure 11 (with the information on Table 1 and 2).

Table 2. Extracting and Decryption Testing

Case and Result (Normal Data)	
Data Input	*Image*: orchidstego.jpg; *Key*: "programstudicesk" (16 characters)
Expected Result	After inputting the stego image and key, user can start theprocess by clicking the Process button. The process will start byextracting information from the stego image, and then it will bedecrypted using the key. If the key is exactly 16 characters, theextracted text will be shown.
Observation	Text is successfully extracted, and is shown in the text box.
Conclusion	Accepted.
Case and Result (Wrong Data)	
Data Input	*Image*: orchidstego.jpg; *Key*: "programstudiskce" (16 characters, different than theencryption key)
Expected Result	After inputting the stego image and key, user can start theprocess by clicking the Process button. The process will start byextracting information from the stego image, and then it will bedecrypted using the key. If the key is right, the extracted text willbethe same like the original one.
Observation	Text is successfully extracted and is shown in the text box, butdifferent than the original text.
Conclusion	Accepted.
Case and Result (Wrong Data)	
Data Input	*Image*: orchid.jpg; (bukan citra stego) *Key*: "programstudicesk"
Expected Result	Showing error message.
Observation	An error message saying that there is no message extracted isshown.
Conclusion	Accepted.

Figure 12. Interface of *Encryption and Embedding*

Figure 13. Interface of *Decryption and Extracting*

6.1. System Implementation

Based on the system design, the implementation needs a hardware and software which can run Java desktop application. The implementation is made using Eclipse Juno on a Windows 8 operating system, with 2.2 GHz processor. The use lower specification of hardware and software can be done, with the minimum requirements as follow:

6.1.1. Minimum Hardware Requirements

The hardware that can be used should met the minimum requirement as follow: 1.5 GHz processor; 1 GB RAM; and 300 MB hard disk.

6.1.2. Minimum Software Requirements

The software that can be used should met the minimum requirement as follow:

a. Windows XP operating system;
b. JDK 1.6.0 (Java Development Kit);
c. JRE 1.6.0 (Java Runtime Environment).

6.2. Interface

The system is made as a desktop application with the interface as shown in Figure 12 and 13.

7. Testing

To see the performance of the implemented system, some testing have been done to see the functionality of the system, the resistance of stego image, and the cryptography algorithm performance.

7.1. Black-box Testing

This testing focused on the system functionality, to see whether both the input and output are the same as the expected result. Based on the equation (1) and (2), the MSE and PSNR values are shown on Table 5.

Table 3. Comparison of Original and Stego Image

No.	Original image	Stego image	File name
1.			Orchid.jpg
2.			Beach.jpg
3.			Hutsk.jpg

Table 4. Image's Resolution and Size

No	File name	File resolution		File size	
		Ori img	Stego img	Ori img	Stego img
1.	Orchid.jpg	1920×1825	1920×1825	383 KB	917 KB
2.	Beach.jpg	1600×1200	1600×1200	399 KB	882 KB
3.	Hutsk.jpg	720×720	720×720	64.8 KB	211 KB

Table 5. MSE and PSNR Value

No	File name	MSE	PSNR
1.	Orchid.jpg	0.8504	48.86 dB
2.	Beach.jpg	3.9491	42.19 dB
3.	Hutsk.jpg	1.2822	48.08 dB

Table 6. Salt and Pepper Test

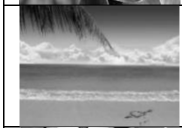
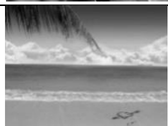
Stego image	Stego image+noise	Extracted text
		(nothing)
		(nothing)
		(nothing)

Table 7. Cropping Test

Stego image	Stego image+noise	Extracted text
		(nothing)
		(nothing)
		(nothing)

Table 8. Compression Test

Stego image	Stego image+noise	Extracted text
		(nothing)
		(nothing)
		(nothing)

The result of PSNR ranges from 42 dB – 49 dB. Standard PSNR value for image with bit depth of 8 bits is 30 dB – 40 dB or more, as seen on Table 5. Thus, the resulting image has a good performance.

7.1.1. Image's Resistance Testing

7.1.1.1 Salt and Pepper Noise

This testing is done by adding noise to stego image,which is a black and/or white pixel randomly on the entire image, as seen on Table 6.

7.1.1.2 Cropping Process

This testing is done after cropping the image by 50%, as seen on Table 7.

7.1.1.3. Compression Process

This testing is done after doing a compression to the stego image with the quality of 70, as seen on Table 8.

From the Salt and Pepper testing, we can see that if there's noise on the stego image, the coefficient's value will change and text can not be extracted. In the cropping test, some of the pixel is cropped and some embedded information might have lost with it. The amount of coefficients also reduced due to the cropping and it will cause an extraction failure. From the compression testing, the text can not be extracted because the compression process will change the coefficient's value.

7.1.2. White-box Testing

This testing is done using test case.

	User has to input the text, key, image, and outputdirectory. If one of them is not inputted, an error message will appear. Error message if text is empty: Error message if key is not 16 characters: Error message if there is no image selected: Error message if there is no output directoryspecified:

(a)

	If all the parameters are inputted correctly, the processwill be done. Stego image and the elapsed process timewill be shown.

(b)

	User has to input the stego image and key. If one ofthem is not inputted, an error message will appear. Error message if the key is not 16 characters: Error message if there is no image selected: If the inputted stego image is a stego image with noiseor disruption, an error message will appear and thep rocess will be stopped.

(c)

	If all the parameters are inputted correctly, the processwill be done. Stego image and the elapsed process time will be shown.

(d)

Figure 14. White-box Testing. (a) Encryption and Embedding (Error message), (b) Encryption and Embedding, (c)Extracting and Decryption (Error message), and (d) Extracting and Decryption

7.2. Avalanche Effect Testing

In cryptography, the result is very unique, differentthan the inputted data. A little change on the input data,the result will change drastically. This is called Avalanche Effect.

From the test, just one bit changes in the input will change the output from 56 – 76 bits. This shows that the implemented AES-128 algorithm has a good performance, as seen on Table 9. It changes half of the cipher text which is the best proportion, rather than changing too little or too much.

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	FF	D8	FF	E0	00	10	4A	46	49	46	00	01	01	01	00	60
00000010	00	60	00	00	FF	E1	1C	13	45	78	69	66	00	00	49	49
00000020	2A	00	08	00	00	00	0C	00	0E	01	02	00	1E	00	00	00
00000030	9E	00	00	00	0F	01	02	00	10	00	00	00	BC	00	00	00
00000040	10	01	02	00	0A	00	00	00	CC	00	00	00	12	01	03	00
00000050	01	00	00	00	00	00	00	00	1A	01	05	00	01	00	00	00
00000060	D6	00	00	00	1B	01	05	00	01	00	00	00	DE	00	00	00
00000070	28	01	03	00	01	00	00	00	02	00	00	00	31	01	02	00
00000080	0B	00	00	00	E6	00	00	00	32	01	02	00	14	00	00	00
00000090	F2	00	00	00	13	02	03	00	01	00	00	00	01	00	00	00
000000A0	69	87	04	00	01	00	00	00	82	01	00	00	A5	C4	07	00
000000B0	7C	00	00	00	06	01	00	00	28	04	00	00	53	4F	4E	59
000000C0	20	44	53	43	20	20	20	20	20	20	20	20	20	20	20	20
000000D0	20	20	20	20	20	20	20	20	20	00	53	4F	4E	59	20	20
000000E0	20	20	20	20	20	20	20	20	20	00	44	53	4C	52	2D	41
000000F0	31	30	30	00	48	00	00	00	01	00	00	00	48	00	00	00
00000100	01	00	00	00	50	69	63	61	73	61	20	33	2E	30	00	00
00000110	32	30	30	38	3A	30	34	3A	30	34	20	32	31	3A	35	35
00000120	3A	31	34	00	50	72	69	6E	74	49	4D	00	30	33	30	30
00000130	00	00	00	06	00	01	00	16	00	16	00	02	01	00	00	00
00000140	00	03	00	00	00	34	01	00	05	00	00	00	01	01	00	00

(a)

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	FF	D8	FF	E0	00	10	4A	46	49	46	00	01	01	00	00	01
00000010	00	01	00	00	FF	DB	00	84	00	01	01	01	01	01	01	01
00000020	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
00000030	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
00000040	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
00000050	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
00000060	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
00000070	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
00000080	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
00000090	01	01	01	01	01	01	01	01	01	01	FF	C0	00	11	08	05
000000A0	05	07	80	03	01	22	00	02	11	01	03	11	01	FF	C4	01
000000B0	A2	00	00	01	05	01	01	01	01	01	01	00	00	00	00	00
000000C0	00	00	00	01	02	03	04	05	06	07	08	09	0A	0B	10	00
000000D0	02	01	03	03	02	04	03	05	05	04	04	00	00	01	7D	01
000000E0	02	03	00	04	11	05	12	21	31	41	06	13	51	61	07	22
000000F0	71	14	32	81	91	A1	08	23	42	B1	C1	15	52	D1	F0	24
00000100	33	62	72	82	09	0A	16	17	18	19	1A	25	26	27	28	29
00000110	2A	34	35	36	37	38	39	3A	43	44	45	46	47	48	49	4A
00000120	53	54	55	56	57	58	59	5A	63	64	65	66	67	68	69	6A
00000130	73	74	75	76	77	78	79	7A	83	84	85	86	87	88	89	8A
00000140	92	93	94	95	96	97	98	99	9A	A2	A3	A4	A5	A6	A7	A8

(b)

Figure 15. Comparison of Hexa Decimal from (a) orchid.jpg and (b) orchidstego.jpg

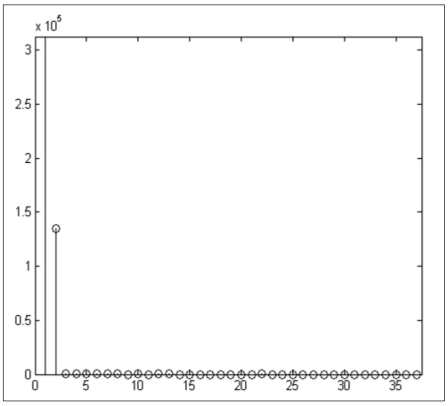


Figure 16. Histogram's Difference from orchid.jpg and orchidstego.jpg

Table 9. Avalanche Effect Testing

Plain text	Cipher text	Avalanche Effect
3333 33 33 33 33 33 33 33 33 33 33 33 33 33 33	D3 F5 51 F6 88 38 65 C5 8E 72 CE 20 F7 6D C4 8C	0.59375 (76 bit change)
3233 33 33 33 33 33 33 33 33 33 33 33 33 33 33	8B C7 45 D9 75 C6 DA 1A A8 28 00 15 20 56 31 9A	
41 4E 47 4B 41 54 414E 20 32 30 30 39 67 75 65	F0 E1 F1 A4 2F 9D 27 28 A3 92 09 F3 22 97 6A 72	0.4375 (56 bit change)
41 4E 47 4B 41 54 414D 20 32 30 30 39 67 75 65	46 FF E3 C1 FF 9C 82 AB E9 E7 9D 73 90 B0 96 69	

8. Conclusion

From the implementation and testing result, the conclusions are as follow:

a. Changes in file size depends on the color intensity. Image with the smalles size changes is orchid2.jpg with 500 – 2000 pixel for each color intensity value. Image with the biggest size change is hutsk.jpg with 1000 – 7000 pixel for each color intensity value.
b. Stego image size is 1.25 – 3.25 times larger than the original image, which can be the result of Huffman Encoding, because when image is being written, the Huffman Table is included, which size is based on the encoding result.
c. Stego image is very sensitive to noise or disruption. A slight crop (0.002%) from the stego image can cause extraction failure.
d. The cryptography algorithm has a good performance, with the resulting changes from 56 – 76 bit for 1 bit changes from the input.

References

[1] Ariyus, Dony, "*Pengantar Ilmu Kriptografi: Teori, Analisis, dan Implementasi*", Yogyakarta, Penerbit Andi, 2008.

[2] Batarius, P., and M. Maslim, "*Perbandingan Metode Dalam Teknik Steganografi*", Semantik 2012.

[3] Cheddad, Abbas, "*Digital Image Steganography*", VDM Publishing, 2009.

[4] Cox, Ingemar J., "*Digital Watermarking and Steganography*", Burlington, Morgan Kaufmann Publisher, 2008.

[5] Fridrich, Jessica, "*Steganography in Digital Media: Principles, Algorithms, and Applications*", Cambridge University Press, 2009.

[6] Hulme, F. Edward, "*Cryptography: Or, the History, Principles, and Practice of Cipher-Writing*", Biblio Bazaar, 2010.

[7] Singh, Ram Kumar and Amit Asthana, "*Steganography*", Lambert Academic Publishing, 2012.

[8] Surian, D., "*Algoritma Kriptografi AES Rijndael*", TESLA Jurnal Teknik Elektro UNTAR, 8(2), pp-97, 2009.