Image Steganography Analysis Using GOST Algorithm and PRNG Based on LSB

Ilham Firman Ashari[1], Andhika Wibawa Bhagaskara[2], Jaya Megelar Cakrawarty[3] , Perdana Raga Winata[4]
[1,2,3,4] Program Studi Teknik Informatika, Institut Teknologi Sumatera
E-mail: [1]firman.ashari@if.itera.ac.id

Abstract

Security in communication is very important for everyone to pay attention to. To prevent data leakage and illegal retrieval of data, cryptography and steganography can be used. In this study, GOST and XOR LSB with PRNG were used to inject text into images. The final result shows that the combination of these two methods is quite good in terms of image quality, message capacity, embedding speed, and imperceptibility (which cannot be seen with the naked eye). Of the 9 image formats tested, 7 images containing messages could be extracted properly with a median MSE of 0.0001935 and a median PSNR of 85.261. Then, in terms of rotation, all im-age rotations from 90°, 180°, and 270° fail to extract messages and in terms of resizing all sizes from 10%, 30%, 50%, and 80% also fail to extract messages. The average time required for encryption with test data 6 test data is 0.073 seconds with the GOST algorithm, using GOST + XOR LSB with PNRG is 0.485 seconds. By using the XOR combination method LSB with PRNG can store an average of 43.3 bits. From the imperceptibility test using 25 respondents and visualization using an RGB histogram, it can be seen that there is no difference between the cover image and the stego image.

Keywords: GOST, PRNG, Steganography, LSB, Image

1. INTRODUCTION

The development of information technology has an impact on convenience for everyone, especially in terms of communicating [1]. Security in communication is very important for everyone to pay attention to. Even so, we often forget about this when we are using various products and services available on the internet. Vital information on the internet such as business, health, education, and government data has a high risk of being a target for crime and illegal retrieval of information (cybercrime) [2]. The forms of cybercrime that exist on the internet vary, ranging from wiretapping, theft, alteration, and others [3]. The data that has been taken can then be used by irresponsible parties in various ways, such as personification, threats, and/or illegal resale.

From the amount of data and information available today, it can be divided into at least 4 basic forms, namely images, text, audio, and video [4]. Among the four forms of data, text data is one type of data that is often stolen and manipulated so that the authenticity of the data is not maintained [5]. With the internet and today's technology, illegal acts can occur quickly and are becoming more common if not properly monitored. Therefore, to reduce and prevent this from happening, several techniques can be used to secure data or information.

Cryptography and steganography are one of the most widely used data security techniques today [6][7]. Cryptography is a data security technique through the plaintext encryption process into a ciphertext that is difficult for others to understand [8]. The word cryptography itself comes from the Greek, crypto, and graphia which when combined have the meaning of "secret writing" [9]. Steganography is a technique of hiding data into a file or storage media [10]. The word steganography (steganography) itself comes from the Greek words steganos (hidden) and graphein (writing) which when combined mean hidden writing [11].

The word steganography (steganography) itself comes from the Greek words steganos (hidden) and graphein (writing) which when combined mean hidden writing [12]. GOST is included in the Feistel network block cipher, where data will be split into several small blocks to be processed in each round in the algorithm [13]. LSB is the insertion of a message in a digital image by modifying the last bits (Least Significant Bit) in the color pixels of the image [14]. This method hides information by inserting a

message at the rightmost bit of an image pixel (or file) [12]. Various attacks against GOST have been carried out in previous studies, where the best attack requires 2^{32} data and 2^{36} memory with a time complexity is 2^{224} [13][15]. To overcome the weaknesses of GOST and LSB, further modifications were made to the LSB using the XOR and PRNG logic functions [16].

2. RESEARCH METHODS

In this study, cryptography in the text is implemented through the GOST algorithm using the Python language with system specifications, namely Mac OS Monterey, M1 pro, SSD NvME 512 GB, and RAM 16 GB. To increase the security of the GOST algorithm, the text is then inserted into the image using the concept of XOR LSB steganography with PRNG. Broadly speaking, the cryptographic and steganographic processes can be described as follows:

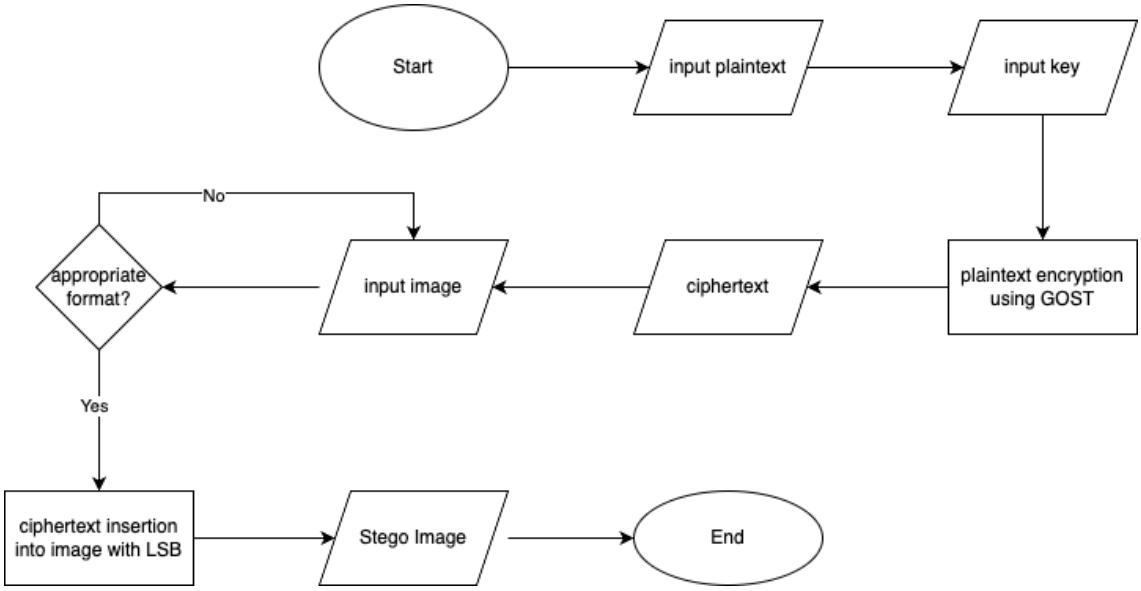


Figure 1. Algorithm process flow

2.1 GOST Algorithm

The GOST algorithm accepts a 256-bit key and 64-bit text (plaintext/ciphertext) per block [15]. If there is a text that is longer than 64 bits, then the text will be broken into several blocks and added padding if needed. To be able to perform the encryption and decryption process, the GOST algorithm uses 32 rounds where various processes are carried out such as XOR, rotating left shift (11 bits), and substitution of binary values with S-boxes.

2.2 Substitution Box (S-Box)

The substitution box or S-box is a table that is used as a reference for the exchange (substitution) of an input value into a new output value. The specification of the S-box table used in the GOST algorithm can be seen in the following table:

Table 1. S-box for GOST algorithm

GOST R 34.12-2015 S-box																
#	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
1	12	4	6	2	10	5	11	9	14	8	13	7	0	3	15	1
2	6	8	2	3	9	10	5	12	1	14	4	7	11	13	0	15
3	11	3	5	8	2	15	10	13	14	1	7	4	12	9	6	0
4	12	8	2	1	13	4	15	6	7	0	10	5	3	14	9	11
5	7	15	5	10	8	1	6	13	0	9	3	14	11	4	2	12
6	5	13	15	6	9	2	12	10	11	7	8	1	4	3	14	0
7	8	14	2	5	6	9	1	12	15	4	11	0	13	10	3	7
8	1	7	14	13	0	5	8	3	4	15	10	6	9	12	11	2

The demonstration of using the S-box for a binary number input "1111000000010011" is as follows:

1. Break the binary number into 4 digits so that 4 digits are produced sequentially, namely “1111” (15), “0000” (0), “0001” (1), and “0011” (3)
2. For the first order, namely "1111" (15), then look for the intersection of column 15 and row 1 so that "0001" is produced (1)
3. For the second order, namely "0000" (0), then look for the intersection of column 0 and row 2 so that it produces "0110" (6), and so on for the third and fourth binary sequences...
4. The final result of the binary input in the form of "1111000000010011" will produce a binary output of "0001011000110001"

2.3 Rotate Left Shift (RLS)

Rotate Left Shift or Left Circular Shift is a simple binary rotation method where the leading binary position (MSB) is deleted and reset to the last binary position (LSB). In the GOST algorithm, the RLS process is used only for the first 11 binary bits. Suppose a binary has the value "11111111 11100000 00000000 00000000", then after the 11-bit RLS the output is "00000000 00000000 00000111 11111111".

2.4 GOST Encryption

The GOST encryption process is carried out in stages. These stages are:

1. Convert plaintext (64 bit/8 characters) and key (256 bit/32 characters) into binary form
2. Split the key binary into 8 parts with the rule K0 = 32 to 1 bit, K1 = 64 to 33 bit, and so on until K7 = 256 to 225 bit
3. Split the plaintext binary into 2 parts (R0 and L0) with the rules R0 = 32 to 1 bit, and L0 = 64 to 33 bits
2. After that, the process consists of 32 rounds (rounds) with the pseudocode for each round as follows:

```
# Initial value
K = [ K0, K1, K2, K3, K4, K5, K6, K7]
R = [ R0 ]
L = [ L0 ]

j = 0
for i in range(32):
    if i < 24:
        # If not 24 rounds then the key sequence is from K0-K7
        hasil = (R[i] + K[j]) mod (2 ** 32)
        if j == 7: j = 0
        else: j = j + 1
    else:
        # If it is 24 rounds then the key sequence from K7-K0
        if i == 24: j = 7
        hasil = (R[i] + K[j]) mod (2 ** 32)
        j = j - 1

    hasil = sbox(hasil)

    if i < 32:
        R[i+1] = RLS(hasil) xor L[i]
        L[i+1] = R[i]
    else:
        R[i+1] = R[i]
        L[i+1] = RLS(hasil) xor L[i]

        # Final result
        R[i+1] = reverse(R[i+1])
        L[i+1] = reverse(L[i+1])
        return R[i+1] merged with L[i+1]
```

2.5 GOST Decryption

The GOST decryption process is carried out in stages and has only a slight difference from the previous encryption process. These stages are:

1. Convert ciphertext (64 bit/8 characters) and key (256 bit/32 characters) into binary form
3. Split the key binary into 8 parts with the rules K0 = 32 to 1 bit, K1 = 64 to 33 bits, and so on until K7 = 256 to 225 bits
4. Split the binary ciphertext into 2 parts (R0 and L0) with the rules R0 = 32 to 1 bit, and L0 = 64 to 33 bits
5. After that, the process consists of 32 rounds (rounds) with the pseudocode for each round as follows:

```
# Initial Value
K = [ K0, K1, K2, K3, K4, K5, K6, K7]
R = [ R0 ]
L = [ L0 ]

j = 0
for i in range(32):
    if i < 8:
        # If not 8 rounds then the key sequence from K0-K7
        hasil = (R[i] + K[j]) mod (2 ** 32)
        if j < 7: j = j + 1
    else:
        # If it's 8 rounds then the key sequence from K7-K0
        hasil = (R[i] + K[j]) mod (2 ** 32)
        if j == 0: j = 7
        else: j = j - 1

    hasil = sbox(hasil)

    if i < 31:
        R[i+1] = RLS(hasil) xor L[i]
        L[i+1] = R[i]
    else:
        R[i+1] = R[i]
        L[i+1] = RLS(hasil) xor L[i]
```

```
        # Final Result
        R[i+1] = reverse(R[i+1])
        L[i+1] = reverse(L[i+1])
        return R[i+1] merged with L[i+1]
```

2.6 XOR LSB Steganography Method with PRNG

The LSB algorithm is one of the easiest steganographic algorithms to implement [17]. However, in exchange for its simplicity, this algorithm is also very susceptible to modification and the messages it inserts are easy to detect. To overcome the weakness of the usual LSB algorithm, PRNG and logical XOR functions can be used to randomize the number of message bits entered per image pixel (3-6 message bits per pixel) [8]. The XOR LSB algorithm with PRNG is based on the publication by Chandra Kumar Deo, et al (2020) [16].

2.7 Message Insertion

Adapun alur dari proses *embedding* (penyisipan pesan) pada algoritma ini yaitu:

1. Select the stego image to which the message will be retrieved
2. Request the seed or PRNG key that was used in the previous insertion process
3. Use PRNG to generate a random number with a range of 1 to 6. If a number 1 or 4 is generated, then the indicated channel is red; if number 2 or 5 then the channel is green, and if number 3 or 6 then the channel is blue
4. Based on the previously selected indicator channel (red, green, or blue), look at the 3-bit MSB value that is on that channel. This 3-bit MSB will be used as a reference for message retrieval with the following rules:

Table 2. Message embedding rules in the LSB algorithm

MSB Value on Channel Indicators	Number of Insert Bits (Red Channel)	Number of Insert Bits (Green Channel)	Number of Insert Bits (Blue Channel)
000	1	1	1
001	1	1	2
010	1	2	1
011	1	2	2
100	2	1	1
101	2	1	2
110	2	2	1
111	2	2	2

5. Perform an XOR operation between the message bits you want to insert at this time and the color channel bits in the previous PRNG result sequence. For example, if the result of PRNG is 6, then perform the message bit operation XOR the blue channel bit of the 6th sequence. For clarity in the next step, call this 6th sequence blue channel bit as an indicator bit
6. The XOR results from the previous step 6 are then inserted into the LSB of one of the channels with the rules as in step 5. Repeat steps 6 and 7 with different message bits and channel colors. The indicator bit XORed with the message bit is always the same as long as it hasn't changed pixels

7. Perform steps 3-5 repeatedly as long as there are still message bits that must be inserted. Then, save all the bit changes to a stego image

2.8 Message Extraction

The flow of message extraction is not much different from message insertion. The flow of message extraction is:

1. Select the stego image to which the message will be retrieved
2. Request the seed or PRNG key that was used in the previous insertion process
3. Use PRNG to generate a random number with a range of 1 to 6. If a number 1 or 4 is generated, then the indicated channel is red; If it's number 2 or 5, then the channel is green, and if it's number 3 or 6, the channel is blue
4. Based on the previously selected indicator channel (red, green, or blue), look at the 3-bit MSB value that is on that channel. This 3-bit MSB will be used as a reference for message retrieval with the following rules:

Table 3. Message retrieval rules on the LSB algorithm

MSB value on indicator channel	Number of Bit Capture (Red Channel)	Number of Bit Capture (Green Channel)	Number of Bit Capture (Blue Channel)
000	1	1	1
001	1	1	2
010	1	2	1
011	1	2	2
100	2	1	1
101	2	1	2
110	2	2	1
111	2	2	2

5. Perform an XOR operation between the current LSB color channel (as defined in step 4) and the color channel bit in the previous PRNG result sequence. For example, if the result of PRNG is 6, then perform the LSB operation on the current color channel XOR the blue channel bit of the 6th sequence. For clarity in the next step, call this 6th sequence blue channel bit as an indicator bit
6. The XOR result from the previous step 5 is then stored as message bits (and combined with other message bits). Repeat steps 5 and 6 with different color LSBs and channels. The indicator bit XORed with LSB is always the same as long as it hasn't changed pixels
7. Perform steps 3-5 repeatedly as long as there are still message bits that must be inserted. Then, save all the bit changes to a stego image

2.9 Testing Methods

Tests are carried out to see the success rate of inserting text messages on cover images using the GOST and XOR LSB algorithms with PRNG. One success rate is based on the MSE and PSNR values in the image. Mean Square Error (MSE) can be interpreted as the average squared error value between the modified pixels and the original pixels [18]. MSE is also generally calculated with the Peak Signal-to-Noise Ratio (PSNR) which is nothing but the ratio of the pixel color ratio value to the MSE (in loga-rhythm form) [19]. A small PSNR value means that the quality of an image is bad, while the greater the PSNR value, the better the quality of an image [20]. Then, there are also other tests based on several aspects of cryptography and steganography. Some of these aspects are:

1. Fidelity: The quality between the cover image and the stego image has not changed much. It can be proven through the MSE and PSNR values as previously described
2. Imperceptibility: Changes are difficult to detect at least with the naked eye. It can be proven by histograms and surveys from several respondents
3. Capacity: The larger the message capacity that can be inserted, the better the steganographic algorithm used

4. Speed: Speed in performing the algorithm computation process. The sooner the better [21].
5. Recovery: Messages that have been pasted can be retrieved or extracted again
6. Robustness: Strength or security when carried out various attacks or modifications to the stego image.

3.RESULT AND DISCUSSION

In this study, the GOST algorithm is used for message encryption/decryption and XOR LSB with PRNG to insert/retrieve messages in images. This testing process was mostly tested on 3 different images with each size of 512 x 512 pixels (“baboon.png”, “lena.png”, and “peppers.png”). The inserted message is "HelloWorld" with the key used for GOST and the PRNG seed "crypto-grafi2022". The image of the program display can be seen below:

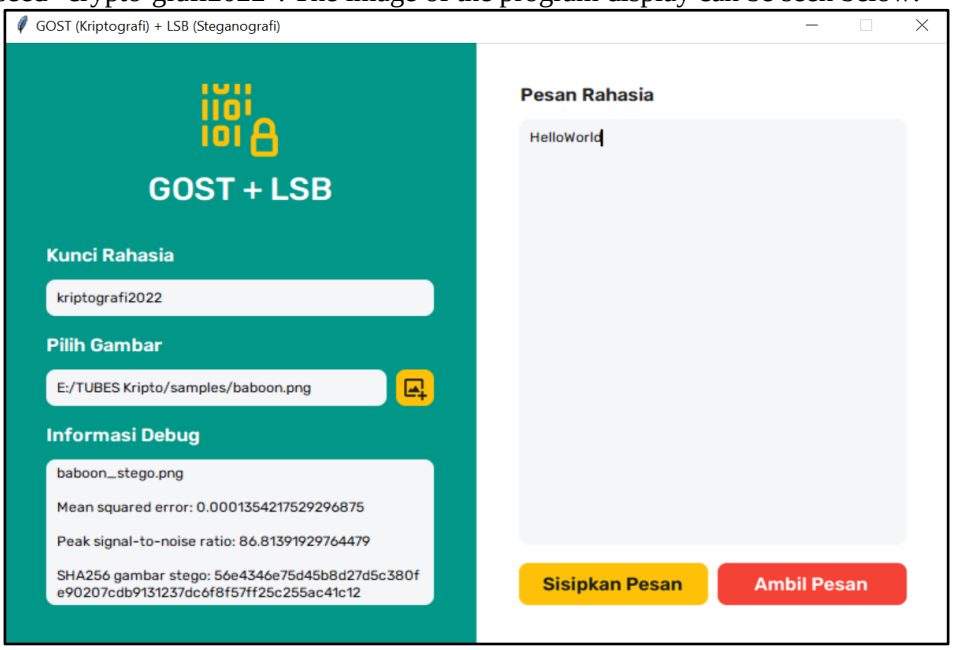


Figure 2. GUI Application

There are 4 main components visible in this program, namely “Secret Key”, “Select Image”, “Debug Information”, and “Secret Message”. “Secret Key” is the part to enter the key that will be used in GOST and PRNG. Then, “Select Image” is used to select an image to insert or retrieve the message. “De-bug Information” is another important section that will show the MSE value, PSNR, SHA256 checksum, and information where the stego image is stored. Finally, “Secret Message” is the section for writing/retrieving messages on images.

3.1 Fidelity and Imperceptibility Testing

After inserting the message "HelloWorld" with the key "cryptography2022" on each image "baboon.png", "lena.png" and "peppers.png", the results obtained are stego, MSE, and PSNR images. Then, the results of MSE and PSNR (GOST + XOR LSB with PRNG) were compared with previous studies using the 1-bit LSB, 2-bit LSB, random bit substitution methods [22], and XOR LSB with PRNG (no GOST encryption) [23]. The MSE and PSNR values of all these methods are given in Tables 4, 5, and 6.

Table 4. Image of stego along with its MSE and PSNR values

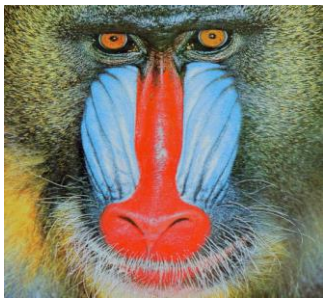


Image Name: baboon.png MSE: 0.0001230 PSNR: 87.230	Image Name: lena.png MSE: 0.0001907 PSNR: 85.326	Image Name: peppers.png MSE: 0.0001935 PSNR: 85.261

Table 5. MSE comparison between methods

Image Stego	1-bit LSB	2-bit LSB	Random bit substitution	XOR LSB With PRNG	Current Method
Baboon.png	0.0000534	0.0001220	0.02022934	0.0000814	0.0001230
Lena.png	0.0000559	0.0001436	0.00004196	0.0001055	0.0001907
Peppers.png	0.0000419	0.0001462	0.00002670	0.0000851	0.0001935
Rata-Rata	0.0000504	0.0001372	0.00676600	0.0000906	0.0001691

Table 6. PSNR comparison between methods

Image Stego	1-bit LSB	2-bit LSB	Random Bit Substitution	XOR LSB With PRNG	Current Method
Baboon.png	90.85	87.26	61.48	89.025	87.230
Lena.png	90.65	86.56	90.18	87.896	85.326
Peppers.png	91.90	86.48	90.09	88.826	85.261
Average	91.13	86.76	80.58	88.582	85.939

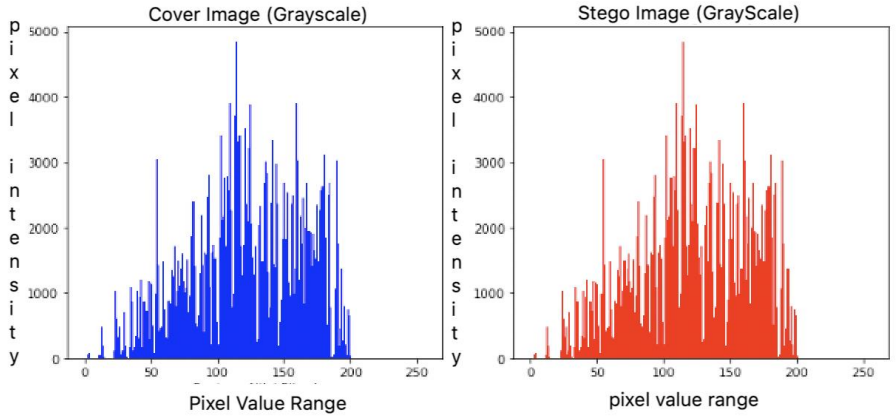
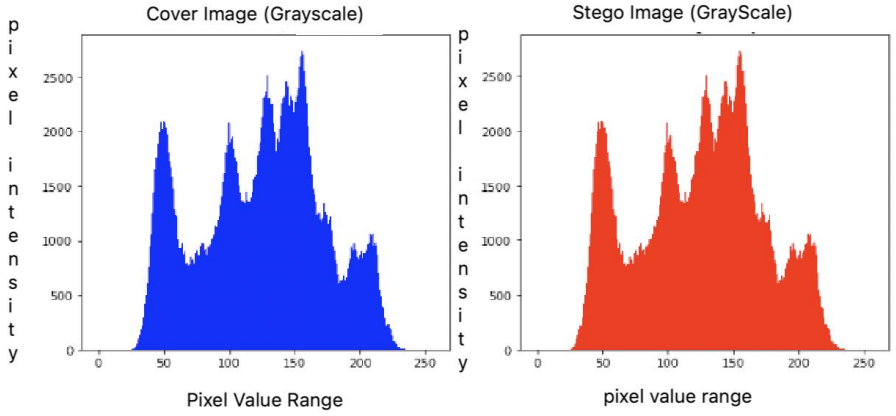
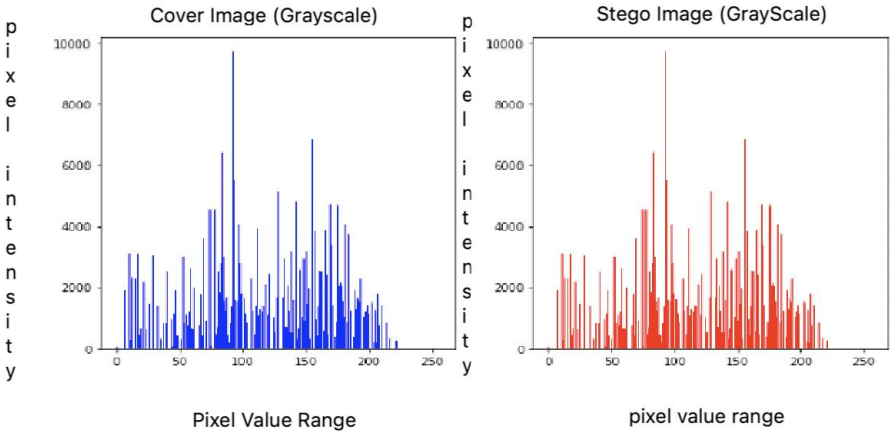
Based on the experimental data, 1-bit LSB (91.13) has the highest PSNR compared to the GOST + XOR LSB method with PRNG (85.939), while random bit substitution (80.58) has the lowest PSNR. This means that GOST + XOR LSB with PRNG has better image quality than the random bit substitution method, and slightly worse than other methods.

Then, an imperceptibility test was also carried out on each cover and stego image to see signs of significant differences between the two types of images. The test was carried out through a histogram comparison (by converting the image to grayscale) as well as a survey method with 25 respondents. The answers from each respondent and histogram results can be seen in tables 7 and 8 below.

Table 7. The results of the imperceptibility survey on cover and stego images

Image	The Difference Between Cover and Stego Images
Baboon.png	25 respondents feel there is no difference
Lena.png	25 respondents feel there is no difference
Peppers.png	25 respondents feel there is no difference

Table 8. Histogram comparison between the cover and stego images

Histogram from Baboon.png

Histogram from Lena.png

Histogram from Peppers.png


Based on these histograms, it can be seen that there is no significant change in the value between the pixels between the cover image and the stego image. Then, when viewed from the imperceptibility survey results, 25 respondents also consistently felt that there was no difference between the cover image and stego for the “baboon.png”, “lena.png”, and “peppers.png” files.

3.2 Capacity Testing

The next test is testing the message storage capacity of each image. In theory, the 1-bit LSB and random bit substitution methods can only store 3 message bits per pixel, while 2-bit LSB can store 6 message bits per pixel. Thus, there are 2 other methods, namely XOR LSB with PRNG and GOST + XOR LSB with PRNG, each of which can store 3-6 bit pixels per image. The results of the capacity test look like the following:

Table 9. Comparison of message bit capacities per pixel between methods

Message Per X Piksel	**1-bit LSB**	**2-bit LSB**	**Random Bit Substitution**	**XOR LSB with PRNG**	**Current Method**
Message Per 10 Piksel in Image (Lena.png)	30 bits	60 bits	30 bits	48 bits	48 bits
Message Per 10 Piksel in Image (Baboon.png)	30 bits	60 bits	30 bits	38 bits	38 bits
Message Per 10 Piksel in Image (Peppers.png)	30 bits	60 bits	30 bits	44 bits	44 bits
Average	30 bits	60 bits	30 bits	43.33 bit	43.33 bit

Based on the test results, there is no difference between the XOR LSB method with PRNG and GOST + XOR LSB with PRNG (both can store an average of 43.33 bits per 10 image pixels). Then, compared to the other 3 methods (which can accommodate 30 bits, 60 bits, and 30 bits respectively), these two XOR LSB methods with PRNG occupy the median position which means they are neither too bad nor good.

3.3 Computation Testing Time

Computational time testing is needed to determine the speed level of the algorithm in processing data [24][25]. Then, when viewed in terms of the speed of the algorithm, the GOST + XOR LSB method with PRNG is still quite fast because it can process the insertion of a fairly long message in less than 1 second (with an average of 0.485 seconds). The test was carried out through the help of the "Code Runner" extension which was run in Visual Studio Code with the image used "peppers.png". Details regarding the speed of the algorithm can be seen in table 10 below.

Table 10. Comparison of the current algorithm speed per message character

Length of Message	**GOST**	**GOST + XOR LSB with PRNG**	**XOR LSB with PRNG (Difference Result)**
1 character	0.046 seconds	0.481 seconds	0.435 seconds
10 characters	0.079 seconds	0.493 seconds	0.414 seconds
50 characters	0.083 seconds	0.473 seconds	0.390 seconds
100 characters	0.087 seconds	0.478 seconds	0.391 seconds
500 characters	0.070 seconds	0.476 seconds	0.406 seconds
1000 characters	0.071 seconds	0.507 seconds	0.436 seconds
Average	0.073 seconds	0.485 seconds	0.412 seconds

3.4 Recovery and Robustness Testing

Finally, in terms of recovery and robustness, message extraction back to the GOST + XOR LSB algorithm with PRNG after modifications to the stego image shows that the LSB algorithm is quite good at inserting messages into lossless image formats (PNG, TIFF, TGA, etc.) but is very bad for a lossy format like JPEG. Then, based on the image modification test (rotation and resizing) it also shows that the LSB is very weak if various modifications are made to the stego image. The results of the recovery and robustness tests can be seen in tables 11 and 12.

Table 11. Message extraction test on various image formats

Stego Image	**MSE**	**PSNR**	**Message Extraction**
Baboon.png	0.0001230	87.230	✓
Lena.png	0.0001907	85.326	✓
Peppers.png	0.0001935	85.261	✓
Airplane.bmp	0.0002962	83.413	✓
Corona.tga	0.0001420	86.604	✓
House.gif	0.4928525	51.203	✗
Peacock.webp	0.0001805	85.564	✓
Sailboat.tiff	0.0002390	84.345	✓
Sunflower.jpg	26.620347	33.878	✗
Median	0.0001935	85.261	-

Table 12. Message extraction test after stego image modification

Stego Image	**Message Extraction (Rotating)**			**Message Extraction (Resizing)**			
	90°	**180°**	**270°**	**10%**	**30%**	**50%**	**80%**
Baboon.png	✗	✗	✗	✗	✗	✗	✗
Lena.png	✗	✗	✗	✗	✗	✗	✗
Peppers.png	✗	✗	✗	✗	✗	✗	✗

As can be seen in tables 11 dan 12, message extraction failed in GIF and JPEG formats with PSNRs of 51,203 and 33,878, respectively. Of the 9 tested image formats, the remaining 7 messages can be extracted well with the median MSE 0.0001935 and median PSNR 85.261. Then, in terms of rotation, all image rotations from 90°, 180°, and 270° fail to extract messages and in terms of resizing all sizes from 10%, 30%, 50%, and 80% also fail to extract messages.

4. CONCLUSION

After doing some testing of the results and analysis in the previous stages, several conclusions can be drawn, namely:

1. The GOST and XOR LSB algorithms with PRNG are quite good in terms of image quality, message capacity, embedding speed, and imperceptibility (cannot be seen with the naked eye). Even so, this algorithm is still very weak if carried out various attacks/modifications on the resulting stego image (eg rotation and resizing).
2. Testing on the robustness aspect, from 9 tested image formats, 7 images containing messages can be extracted properly with a median MSE of 0.0001935 and a median PSNR of 85.261. Then, in terms of rotation, all image rotations from 90°, 180°, and 270° fail to extract messages and in terms of resizing all sizes from 10%, 30%, 50%, and 80% also fail to extract messages.
3. Testing the computational time, the average time required for encryption with test data 6 test data is 0.073 seconds with the GOST algorithm, using GOST + XOR LSB with PNRG is 0.485 seconds.
4. Testing on the aspect of capacity, using the XOR LSB combination method with PRNG can store an average of 43.3 bits.
2. Testing on the imperceptibility aspect using 25 respondents and visualization using an RGB histogram showed no difference between the cover image and the stego image.
3. Testing on the fidelity aspect of the 3 tested images, 1-bit LSB (91.13) has the highest PSNR when compared to the GOST + XOR LSB method with PRNG (85.939), while random bit substitution (80.58) has the lowest PSNR. This means that GOST + XOR LSB with PRNG has better image quality than the random bit substitution method, and slightly worse than other methods.

REFERENCES

[1] I. F. Ashari, "Implementation of Cyber-Physical-Social System Based on Service Oriented Architecture in Smart Tourism Case Study : Bandung Natural Tourism," *J. Appl. Informatics Comput.*, vol. 4, no. 1, pp. 66–73, 2020.

[2] D. Darwis, "Implementasi Teknik Steganografi Least Significant Bit (LSB) Dan Kompresi Untuk Pengamanan Data Pengiriman Surat Elektronik," *J. Teknoinfo*, vol. 10, no. 2, p. 32, 2016, doi: 10.33365/jti.v10i2.8.

[3] B. Nabila and L. T. Sianturi, "Implementasi Fungsi Hash Untuk Mendeteksi Orisinalitas File Audio Menggunakan Metode Gost," *... dan Teknol. Ilm.*, vol. 8, no. 2, pp. 48–52, 2021, [Online]. Available: http://ejurnal.stmik-budidarma.ac.id/index.php/inti/article/view/2842.

[4] I. F. Ashari, A. J. Aryani, and A. M. Ardhi, "DESIGN AND BUILD INVENTORY MANAGEMENT INFORMATION SYSTEM," vol. 9, no. 1, pp. 27–35, 2022.

[5] I. F. Ashari, M. Alfarizi, M. N. K, and M. A. H, "Vulnerability Analysis and Proven On The neonime . co Website Using OWASP ZAP 4 and XSpear," *J. Teknol. Komput. dan Sist. Inf.*, vol. 5, no. 2, pp. 75–81, 2022.

[6] S. G. M. Siregar, "Implementasi Metode Enhanced Audio Steganogafi (EAS) Untuk Penyembunyian Text Terenkripsi Algoritma Gost," *KLIK Kaji. Ilm. Inform. dan Komput.*, vol. 2, no. 1, pp. 20–27, 2021, [Online]. Available: http://www.djournals.com/klik/article/view/220%0Ahttp://www.djournals.com/klik/article/download/220/158.

[7] I. F. Ashari, "Graph Steganography Based On Multimedia Cover To Improve Security and Capacity," in *2018 International Conference on Applied Information Technology and Innovation (ICAITI)*, 2018, no. April 2019, pp. 194–201.

[8] L. Anggriani, "Penerapan Metode Gost Untuk Mendeteksi Keaslian File Dokumen," *TIN Terap. Inform. Nusant.*, vol. 1, no. 9, pp. 445–450, 2021, [Online]. Available:

https://ejurnal.seminar-id.com/index.php/tin/article/view/625.

[9] M. Diana, "Implementasi Metode GOST (Government Standard) dan LSB-1 (Least Significant Bit) Untuk Mengamankan Teks TIN : Terapan Informatika Nusantara," *Terap. Inform. Nusant.*, vol. 1, no. 7, pp. 363–382, 2020.

[10] L. P. Malese, "Penyembunyian Pesan Rahasia Pada Citra Digital dengan Teknik Steganografi Menggunakan Metode Least Significant Bit (LSB)," *J. Ilm. Wahana Pendidik. https//jurnal.unibrah.ac.id/index.php/JIWP*, vol. 6, no. 3, pp. 295–307, 2020, doi: 10.5281/zenodo.5563416.

[11] N. A. Simatupang and N. A. Hasibuan, "Keamanan File Teks Menggunakan Algoritma Government Standard (GOST)," *Pelita Inform. …*, vol. 7, pp. 253–257, 2018, [Online]. Available: https://www.ejurnal.stmik-budidarma.ac.id/index.php/pelita/article/view/1091.

[12] J. I. Sari, H. T. Sihotang, and T. Informatika, "Implementasi Penyembunyian Pesan Pada Citra Digital Dengan Menggabungkan Algoritma Hill Cipher Dan Metode Least Significant Bit (LSB)," *J. Mantik Penusa*, vol. 1, no. 2, pp. 1–8, 2017, [Online]. Available: http://e-jurnal.pelitanusantara.ac.id/index.php/mantik/article/view/253.

[13] I. Dinur, O. Dunkelman, and A. Shamir, "Improved attacks on full GOST," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7549 LNCS, pp. 9–28, 2012, DOI: 10.1007/978-3-642-34047-5_2.

[14] I. F. Ashari, "Aplikasi steganografi pesan teks pada media audio mp3 menggunakan metode penyisipan least significant bit dan advanced encryption standard skripsi," pp. 1–114, 2015.

[15] A. Riski, H. Purwantoro, and A. Kamsyakawuni, "PENYEMBUNYIAN CIPHERTEXT ALGORITMA GOST PADA CITRA KE DALAM AUDIO DENGAN METODE LEAST SIGNIFICANT BIT Abduh," *J. Ilm. Mat. dan Pendidik. Mat.*, vol. 10, no. 2, pp. 49–62, 2018.

[16] C. K. Deo, A. Singh, D. K. Singh, and N. K. Soni, "Developing a Highly Secure and High Capacity LSB Steganography Technique using PRNG," *2020 Int. Conf. Comput. Perform. Eval. ComPE 2020*, pp. 136–140, 2020, DOI: 10.1109/ComPE49325.2020.9200077.

[17] O. Rachael, S. Misra, R. Ahuja, A. Adewumi, F. Ayeni, and R. Mmaskeliunas, "Image Steganography and Steganalysis Based on Least Significant Bit (LSB)," *Lect. Notes Electr. Eng.*, vol. 605, no. March 2020, pp. 1100–1111, 2020, DOI: 10.1007/978-3-030-30577-2_97.

[18] U. Sara, M. Akter, and M. S. Uddin, "Image Quality Assessment through FSIM, SSIM, MSE, and PSNR—A Comparative Study," *J. Comput. Commun.*, vol. 07, no. 03, pp. 8–18, 2019, DOI: 10.4236/jcc.2019.73002.

[19] T. Ai Munandar, M. Adelvin L, and A. J. Santoso, "Analisa Psnr, Rasio Kompresi Warna Dan Mse Terhadap Kompresi Image Menggunakan 31 Fungsi Wavelet," no. October, 2011.

[20] I. Maulana and P. N. Andono, "Analisa Perbandingan Adaptif Median Filter Dan Median Filter Dalam Reduksi Noise Salt & Pepper," *CogITo Smart J.*, vol. 2, no. 2, p. 157, 2016, doi: 10.31154/cogito.v2i2.26.157-166.

[21] I. F. Ashari, R. Banjarnahor, and D. R. Farida, "Application of Data Mining with the K-Means Clustering Method and Davies Bouldin Index for Grouping IMDB Movies," vol. 6, no. 1, pp. 7–15, 2022.

[22] T. Limbong *et al.*, "The implementation of computer-based instruction model on Gost Algorithm Cryptography Learning," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 420, no. 1, 2018, DOI: 10.1088/1757-899X/420/1/012094.

[23] G. Hatzivasilis, K. Fysarakis, I. Papaefstathiou, and C. Manifavas, "A review of lightweight block ciphers," *J. Cryptogr. Eng.*, vol. 8, no. 2, pp. 141–184, 2018, DOI: 10.1007/s13389-017-0160-y.

[24] I. F. Ashari and V. Adhelia, "Expert System and IoT for Diagnose of Feline

Panleukopenia Virus Using Certainty Factor," *Matrik J. Manajemen, Tek. Inform. dan Rekayasa Komput.*, vol. 21, no. 2, pp. 451–462, 2022, doi: 10.30812/matrik.v21i2.1517.

[25] I. F. Ashari, M. D. Satria, and M. Idris, "Parking System Optimization Based on IoT using Face and Vehicle Plat Recognition via Amazon Web Service and ESP-32 CAM (Case Study : Institut Teknologi Sumatera)," vol. 11, no. 2, pp. 137–153, 2022.