

**IMPLEMENTASI STEGANOGRAFI TEKS PADA GAMBAR  
MENGUNAKAN ALGORITMA GOST DAN XOR LSB DENGAN PRNG**

**Laporan Tugas Besar Kriptografi**



**Disusun Oleh**

Jaya Megelar Cakrawarty	119140227
Andhika Wibawa Bhagaskara	119140218
Perdana Raga Winata	119140087

**PROGRAM STUDI TEKNIK INFORMATIKA  
JURUSAN TEKNIK ELEKTRO, INFORMATIKA, DAN SISTEM FISIKA  
INSTITUT TEKNOLOGI SUMATERA  
LAMPUNG SELATAN**

**2022**

## DAFTAR ISI

<b>DAFTAR ISI</b>	<b>2</b>
<b>BAB I PENDAHULUAN</b>	<b>4</b>
1.1 Latar Belakang	4
1.2 Rumusan Masalah	5
1.3 Tujuan Penelitian	5
<b>BAB II LANDASAN TEORI</b>	<b>6</b>
2.1 Kriptografi	6
2.2 Government Standard (GOST)	6
2.3 Steganografi	6
2.4 Least Significant Bit (LSB)	7
2.5 MSE dan PSNR	7
2.6 Fungsi Hash dan SHA256	8
<b>BAB III METODE DAN PEMBAHASAN</b>	<b>9</b>
3.1 Metodologi Penelitian	9
3.2 Studi Pustaka dan Literatur	9
3.3 Analisis Masalah	9
3.4 Implementasi	10
3.5 Pengujian	10
3.6 Maintenance	11
<b>BAB IV HASIL DAN ANALISIS</b>	<b>12</b>
4.1 Hasil Pengujian	13
<b>BAB V KESIMPULAN</b>	<b>17</b>
<b>DAFTAR PUSTAKA</b>	<b>18</b>
<b>LOG SHEET</b>	<b>21</b>

# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

Keamanan dalam komunikasi merupakan hal yang sangat penting untuk diperhatikan tiap-tiap orang. Meski begitu, seringkali kita lupa akan hal tersebut ketika sedang menggunakan berbagai produk dan jasa yang tersedia di internet. Informasi vital yang ada di internet seperti data bisnis, kesehatan, pendidikan, dan pemerintahan memiliki risiko yang tinggi sebagai target dari kejahatan dan pengambilan informasi secara ilegal (*cyber crime*) [1]. Bentuk dari *cyber crime* yang ada di internet bermacam-macam, mulai dari penyadapan, pencurian, perubahan, dan lain lain [2]. Data yang telah diambil ini kemudian dapat dimanfaatkan oleh pihak yang tidak bertanggung jawab dalam berbagai hal, seperti personifikasi, pengancaman, dan/atau untuk dijual kembali secara ilegal.

Dari banyaknya data dan informasi yang tersedia saat ini, setidaknya dapat dibagi menjadi 4 bentuk dasar, yaitu gambar, teks, audio, dan video. Diantara keempat bentuk data tersebut, data teks merupakan salah satu jenis data yang sering dicuri dan dilakukan manipulasi sehingga keaslian data menjadi tidak terjaga. Dengan adanya internet dan teknologi yang ada saat ini, tindakan ilegal tersebut dapat terjadi secara cepat dan semakin lumrah bila tidak diawasi dengan baik. Oleh karena itu, untuk mengurangi dan mencegah terjadinya hal tersebut, terdapat beberapa teknik yang bisa digunakan untuk mengamankan data atau informasi. Kriptografi merupakan teknik pengamanan data melalui proses enkripsi plainteks menjadi suatu cipherteks yang sulit dipahami orang lain [3]. Lalu, steganografi merupakan teknik menyembunyikan data ke dalam suatu *file* atau media penampung [4].

Pada penelitian ini, penulis mencoba menggabungkan 2 teknik pengamanan data (teks ke dalam gambar) menggunakan 2 algoritma yaitu GOST (untuk kriptografi) dan LSB (untuk steganografi) untuk dianalisis keefektifannya. GOST (*Government Standard*) merupakan algoritma alternatif dari DES (*Data Encryption Standard*) yang banyak digunakan *software* (di masa lampau) [5], sedangkan LSB merupakan penyisipan pesan pada citra digital dengan memodifikasi bit-bit terakhir (Least Significant Bit) pada piksel warna gambar [4]. Berbagai serangan terhadap GOST telah dilakukan pada penelitian-penelitian sebelumnya, dimana serangan terbaik membutuhkan  $2^{32}$  data dan  $2^{36}$  memori dengan kompleksitas waktu yaitu  $2^{224}$

[6]. Untuk mengatasi kelemahan GOST dan juga LSB, dilakukan pula modifikasi lebih lanjut terhadap LSB menggunakan fungsi logika XOR dan PRNG [8].

## **1.2 Rumusan Masalah**

Berdasarkan latar belakang tersebut, dapat ditarik beberapa rumusan masalah yaitu:

1. Apakah algoritma GOST dan XOR LSB dengan PRNG dapat digunakan untuk menyisipkan teks (steganografi) pada gambar?
2. Bagaimana proses penggunaan algoritma GOST dan XOR LSB dengan PRNG?
3. Bagaimana keefektifan steganografi teks pada gambar dengan menggabungkan algoritma GOST dan XOR LSB dengan PRNG?

## **1.3 Tujuan Penelitian**

Adapun tujuan akhir dari diberlakukannya penelitian ini yaitu:

1. Untuk mengetahui apakah algoritma GOST dan XOR LSB dengan PRNG dapat digunakan untuk menyisipkan teks pada gambar.
2. Untuk mengetahui proses kerja algoritma GOST dan XOR LSB dengan PRNG.
3. Untuk mengetahui efektivitas dari steganografi teks pada gambar dengan menggabungkan algoritma GOST dan XOR LSB dengan PRNG.

## **BAB II**

### **LANDASAN TEORI**

#### **2.1 Kriptografi**

Kriptografi merupakan suatu teknik untuk mencapai kerahasiaan dalam mengirim dan menerima pesan [9]. Kata kriptografi sendiri berasal dari bahasa Yunani, *crypto* dan *graphia* yang bila digabungkan memiliki arti “penulisan rahasia” (*secret writing*) [10]. Kriptografi modern yang digunakan saat ini beragam dan umum digunakan untuk tujuan keamanan data informasi seperti autentifikasi, kerahasiaan, dan integritas data. Dalam kriptografi, terdapat proses enkripsi dimana data akan diacak menggunakan kunci sehingga menjadi suatu susunan data baru yang sulit dibaca (bila tanpa didekripsi terlebih dahulu) [3]. Proses dekripsi kemudian digunakan untuk mendapatkan kembali data asli tersebut. Rahasia dalam teknik kriptografi umumnya terletak dalam *parameter*, dimana terdapat kunci yang harus dirahasiakan [11].

#### **2.2 Government Standard (GOST)**

GOST merupakan salah satu algoritma yang termasuk ke dalam *feistel network block cipher*, dimana data akan dipecah menjadi beberapa blok kecil untuk diproses pada tiap *round* (putaran) dalam algoritmanya [12]. Algoritma ini termasuk ke dalam kriptografi modern yang berorientasi bit karena dilakukan menggunakan media komputer dalam melakukan penyandiannya [13]. Dalam proses enkripsi algoritma GOST, terdapat proses 32 *round* (putaran) dengan menggunakan 64 bit *block cipher* (sebagai teks) dan 256 bit sebagai kunci [14]. Algoritma GOST yang diimplementasikan pada penelitian ini didasarkan pada publikasi oleh Taroni Zebua (2014) [15] dan Tonni Limbong, dkk (2018) [7].

#### **2.3 Steganografi**

Steganografi merupakan teknik untuk mengirimkan data rahasia yang tersembunyi dibalik data/media lainnya [16]. Kata steganografi (*steganography*) sendiri berasal dari bahasa Yunani yaitu *steganos* (tersembunyi) dan *graphein* (menulis) yang bila digabungkan berarti tulisan yang tersembunyi [17]. Dalam steganografi, hampir semua jenis *file* dapat digunakan untuk penyembunyian pesan, tetapi jenis file yang umum untuk digunakan dalam steganografi adalah yang memiliki tingkat *redundancy* tinggi. Redundancy merupakan jumlah bit yang

berlebih dalam suatu objek sehingga dapat disisipkan banyak pesan dengan akurasi yang tinggi [17]. Dalam steganografi juga terdapat 2 istilah penting lainnya yaitu *cover* dan *stego*. *Cover* merupakan media yang masih murni sehingga hasil akhir dari proses steganografi adalah media *stego* (yang telah disisipkan pesan) [16].

## 2.4 Least Significant Bit (LSB)

*Least Significant Bit* merupakan salah satu metode steganografi yang paling mudah digunakan untuk menyembunyikan informasi dengan lompatan tertentu dalam suatu media *cover* [18]. Metode ini menyembunyikan informasi dengan cara menyisipkan pesan pada bit paling kanan pada suatu piksel gambar (atau *file*) [11]. Pada nilai biner 1001000 misalnya, angka satu paling depan merupakan MSB (*Most Significant Bit*) dan angka nol paling belakang merupakan LSB [14]. Untuk memperkuat keamanan dari algoritma LSB konvensional, juga digunakan fungsi logika XOR dan PRNG sehingga jumlah bit pesan yang disisipkan pada suatu piksel warna menjadi acak dan lebih sulit untuk dibaca [8].

## 2.5 MSE dan PSNR

*Mean Square Error* (MSE) dapat diartikan sebagai nilai *error* kuadrat rata-rata antara piksel yang telah diubah dengan piksel aslinya [19]. MSE merupakan metode yang paling simpel dan banyak digunakan untuk mengukur kualitas gambar. MSE juga umumnya dihitung bersama *Peak Signal-to-Noise Ratio* (PSNR) yang tidak lain adalah perbandingan nilai rasio antara warna piksel dengan MSE (dalam bentuk logaritma) [20]. Nilai PSNR yang kecil memiliki arti bahwa kualitas dari suatu citra tersebut buruk, sedangkan semakin besar nilai PSNR maka semakin baik juga kualitas dari suatu citra [21]. Hubungan antara MSE dan PSNR adalah saling berkebalikan, dimana semakin besar nilai PSNR maka akan semakin kecil nilai MSE-nya. Rumus dari MSE dan PSNR dapat dilihat di bawah ini dengan (H adalah tinggi dan W adalah lebar):

$$MSE = \frac{1}{H \times W} \sum_{i=1}^H \sum_{j=1}^W (x(i, j) - y(i, j))^2$$

$$PSNR = 20 \times \log \frac{255}{\sqrt{MSE}}$$

## 2.6 Fungsi Hash dan SHA256

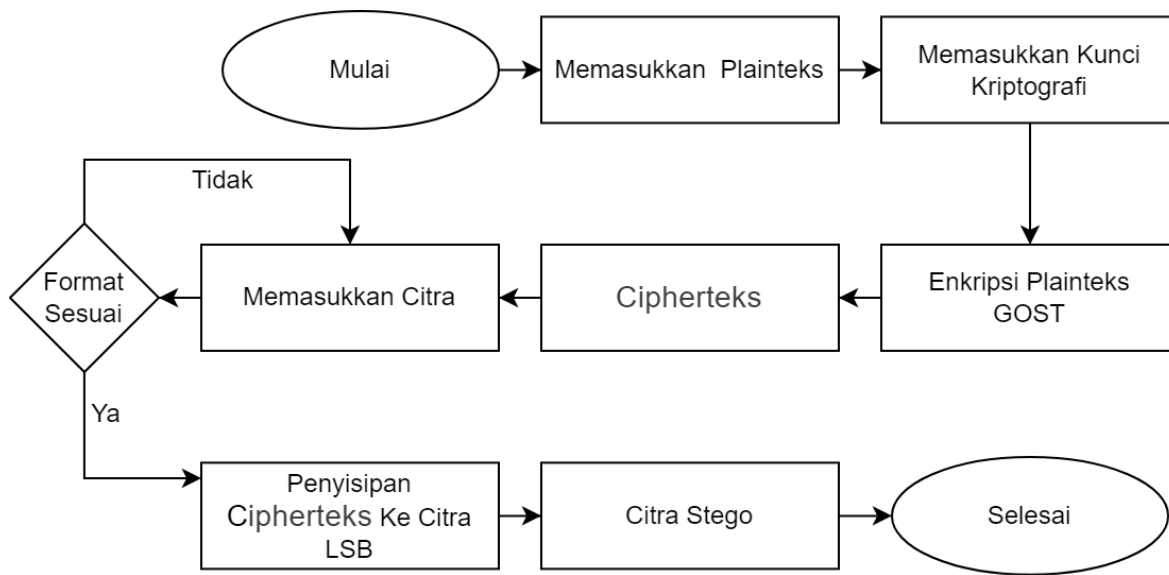
*Hash* merupakan salah satu metode kriptografi satu arah (tanpa kunci) dimana suatu pesan dengan panjang yang bervariasi akan selalu menghasilkan *hash code* dengan panjang yang tetap [22]. Hasil berupa *hash code* atau *checksum* ini kemudian dapat dimanfaatkan untuk mengecek keaslian suatu data atau *file*. SHA-2 merupakan salah fungsi *hash* yang dirancang National Security Agency (NSA) dan dipublikasi oleh National Institute of Standard and Technology (NIST) [23]. Fungsi *hash* SHA sendiri terdiri dari 4 varian yaitu SHA-0, SHA-1, SHA-2, dan SHA-3. Algoritma SHA-2 kemudian mengalami pembaruan dengan panjang *output* 256 bit (SHA256) atau 512 bit (SHA512) [24]. Struktur kedua fungsi *hash* ini memiliki perbedaan pada jumlah putaran dan panjang *hash code* yang dihasilkan saja [23].

## BAB III

### METODE DAN PEMBAHASAN

#### 3.1 Metodologi Penelitian

Pada penelitian ini, kriptografi pada teks diimplementasikan melalui algoritma GOST menggunakan bahasa Python dengan spesifikasi sistem yaitu Manjaro Linux 21.2.6, Intel Core i5 8250U, HDD 1 TB SATA 5400 RPM, dan SDRAM 12 GB DDR3 2133 MHz. Untuk meningkatkan keamanan dari algoritma GOST tersebut, teks kemudian disisipkan ke gambar menggunakan konsep steganografi XOR LSB dengan PRNG. Secara garis besar, proses kriptografi dan steganografi dapat digambarkan seperti di bawah ini:



Gambar 1. Alur proses algoritma

#### 3.2 Metode Kriptografi GOST

Algoritma GOST menerima kunci sebanyak 256 bit serta teks (plaintext/ciphertext) per blok-nya sebanyak 64 bit. Bila terdapat teks yang lebih panjang dari 64 bit, maka teks tersebut akan dipecah menjadi beberapa blok dan ditambahkan *padding* jika diperlukan. Untuk dapat melakukan proses enkripsi dan dekripsi, algoritma GOST menggunakan 32 *round* (putaran) dimana dilakukan berbagai proses seperti XOR, *rotate left shift* (11 bit), dan substitusi nilai biner dengan S-box.



### 3.2.1 Kotak Substitusi (S-Box)

Kotak substitusi atau S-box merupakan tabel yang digunakan sebagai acuan penukaran (substitusi) suatu nilai masukan menjadi nilai keluaran yang baru. Spesifikasi tabel S-box yang digunakan pada algoritma GOST ini dapat dilihat pada tabel berikut:

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

Tabel 1. Kotak S-box untuk algoritma GOST

Demonstrasi penggunaan S-box untuk suatu angka biner masukan “1111000000010011” adalah sebagai berikut:

1. Pecah angka biner menjadi per 4 digit sehingga dihasilkan 4 buah angka secara berurutan yaitu “1111” (15), “0000” (0), “0001” (1), dan “0011” (3)
2. Untuk urutan pertama yaitu “1111” (15), maka cari perpotongan kolom 15 dan baris 1 sehingga dihasilkan “0001” (1)
3. Untuk urutan kedua yaitu “0000” (0), maka cari perpotongan kolom 0 dan baris 2 sehingga dihasilkan “0110” (6), dan seterusnya untuk biner urutan ketiga dan keempat...
4. Hasil akhir dari masukan biner berupa “1111000000010011” akan menghasilkan keluaran berupa biner “0001011000110001”

### 3.2.2 Rotate Left Shift (RLS)

*Rotate Left Shift* atau *Left Circular Shift* merupakan cara perputaran biner secara sederhana dimana posisi biner paling depan (MSB) akan dihapus dan diatur ulang ke posisi biner paling belakang (LSB). Pada algoritma GOST, proses RLS digunakan hanya untuk 11 bit biner pertama saja. Misalkan suatu biner bernilai “11111111 11100000 00000000 00000000”, maka setelah dilakukan RLS 11 bit hasil keluarannya adalah “00000000 00000000 00000111 11111111”.

### 3.2.3 Enkripsi GOST

Proses enkripsi GOST dilakukan secara bertahap. Tahapan-tahapan ini yaitu:

1. Konversi plainteks (64 bit/8 karakter) dan kunci (256 bit/32 karakter) ke dalam bentuk biner
2. Pecah biner kunci menjadi 8 bagian dengan aturan K0 = bit ke 32 sampai 1, K1 = bit ke 64 sampai 33, dan seterusnya hingga K7 = bit ke 256 sampai 225
3. Pecah biner plainteks menjadi 2 bagian (R0 dan L0) dengan aturan R0 = bit ke 32 sampai 1, dan L0 = bit ke 64 sampai 33
4. Untuk setelahnya, proses terdiri dari 32 *round* (putaran) dengan *pseudocode* untuk tiap *round* adalah sebagai berikut:

```
# Nilai mula-mula
K = [ K0, K1, K2, K3, K4, K5, K6, K7]
R = [ R0 ]
L = [ L0 ]

# j = Indeks kunci yang sedang digunakan (K0-K7)
# i = Round atau putaran saat ini (0-31)

j = 0
for i in range(32):
    if i < 24:
        # Jika belum 24 round maka urutan kunci dari K0-K7
        hasil = (R[i] + K[j]) mod (2 ** 32)
        if j == 7: j = 0
        else: j = j + 1
    else:
        # Jika sudah 24 round maka urutan kunci dari K7-K0
        if i == 24: j = 7
        hasil = (R[i] + K[j]) mod (2 ** 32)
        j = j - 1

    hasil = sbbox(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]

    # Hasil akhir
    R[i+1] = reverse(R[i+1])
    L[i+1] = reverse(L[i+1])
    return R[i+1] merged with L[i+1]

```

### 3.2.4 Dekripsi GOST

Proses dekripsi GOST dilakukan secara bertahap dan hanya memiliki sedikit perbedaan dengan proses enkripsi sebelumnya. Tahapan-tahapan ini yaitu:

1. Konversi cipherteks (64 bit/8 karakter) dan kunci (256 bit/32 karakter) ke dalam bentuk biner
2. Pecah biner kunci menjadi 8 bagian dengan aturan K0 = bit ke 32 sampai 1, K1 = bit ke 64 sampai 33, dan seterusnya hingga K7 = bit ke 256 sampai 225
3. Pecah biner cipherteks menjadi 2 bagian (R0 dan L0) dengan aturan R0 = bit ke 32 sampai 1, dan L0 = bit ke 64 sampai 33
5. Untuk setelahnya, proses terdiri dari 32 *round* (putaran) dengan *pseudocode* untuk tiap *round* adalah sebagai berikut:

```

# Nilai mula-mula
K = [ K0, K1, K2, K3, K4, K5, K6, K7 ]
R = [ R0 ]
L = [ L0 ]

# j = Indeks kunci yang sedang digunakan (K0-K7)
# i = Round atau putaran saat ini (0-31)

j = 0
for i in range(32):
    if i < 8:
        # Jika belum 8 round maka urutan kunci dari K0-K7
        hasil = (R[i] + K[j]) mod (2 ** 32)
        if j < 7: j = j + 1
    else:
        # Jika sudah 8 round maka urutan kunci dari K7-K0
        hasil = (R[i] + K[j]) mod (2 ** 32)
        if j == 0: j = 7
        else: j = j - 1

    hasil = sbbox(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]

    # Hasil akhir

```

```

R[i+1] = reverse(R[i+1])
L[i+1] = reverse(L[i+1])
return R[i+1] merged with L[i+1]

```

### 3.3 Metode Steganografi XOR LSB dengan PRNG

Algoritma LSB merupakan salah satu algoritma steganografi yang paling mudah untuk diimplementasikan. Meski begitu, sebagai ganti dari kemudahannya, algoritma ini juga sangat rentan terhadap modifikasi dan pesan yang disisipkannya mudah untuk dideteksi. Untuk mengatasi kelemahan dari algoritma LSB biasa, PRNG dan fungsi logika XOR dapat digunakan untuk mengacak jumlah bit pesan yang dimasukkan per piksel gambar (3-6 bit pesan per piksel). Algoritma XOR LSB dengan PRNG ini didasarkan pada publikasi oleh Chandra Kumar Deo, dkk (2020) [8].

#### 3.3.1 Penyisipan Pesan

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

1. Pilih gambar *cover* yang akan disisipkan pesan (menjadi gambar *stego*)
2. Minta pesan yang ingin disisipkan dan konversi ke dalam bentuk biner (bila masih dalam bentuk plainteks/cipherteks)
3. Minta *seed* atau kunci untuk PRNG (dapat disamakan dengan kunci GOST)
4. Gunakan PRNG untuk menghasilkan nomor acak dengan rentang 1 sampai 6. Bila dihasilkan nomor 1 atau 4, maka *channel* yang dijadikan indikasi adalah warna merah; bila nomor 2 atau 5 maka *channel* warna hijau, dan bila nomor 3 atau 6 maka *channel* warna biru
5. Berdasarkan *channel* indikator yang terpilih sebelumnya (merah, hijau, atau biru), lihat nilai 3-bit MSB yang ada pada *channel* tersebut. 3 bit MSB ini akan dijadikan acuan penyisipan pesan dengan aturan sebagai berikut:

Nilai MSB Pada Channel Indikator	Jumlah Penyisipan Bit (Channel Merah)	Jumlah Penyisipan Bit (Channel Hijau)	Jumlah Penyisipan Bit (Channel Biru)
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

Tabel 2. Aturan penyisipan pesan pada algoritma LSB

6. Lakukan operasi XOR antara bit pesan yang ingin disisipkan saat ini dengan bit *channel* warna pada urutan hasil PRNG sebelumnya. Misal hasil PRNG adalah 6, maka lakukan operasi bit pesan XOR bit *channel* biru urutan ke 6. Untuk kejelasan pada langkah selanjutnya, sebut saja bit *channel* biru urutan ke 6 ini sebagai bit indikator
7. Hasil XOR dari langkah 6 sebelumnya kemudian disisipkan ke LSB salah satu *channel* dengan aturan seperti pada langkah 5. Ulangi langkah 6 dan 7 dengan bit pesan dan *channel* warna yang berbeda-beda. Bit indikator yang di-XOR dengan bit pesan selalu tetap selama belum berganti piksel
8. Lakukan langkah 3-5 secara berulang-ulang selama masih ada bit pesan yang harus disisipkan. Lalu, simpan semua perubahan bit ke gambar *stego*

### 3.3.2 Ekstraksi Pesan

Alur dari ekstraksi pesan tidak jauh berbeda dengan penyisipan pesan. Adapun alur dari ekstraksi pesan yaitu:

1. Pilih gambar *stego* yang akan diambil pesannya
2. Minta *seed* atau kunci PRNG yang digunakan pada proses penyisipan sebelumnya
3. Gunakan PRNG untuk menghasilkan nomor acak dengan rentang 1 sampai 6. Bila dihasilkan nomor 1 atau 4, maka *channel* yang dijadikan indikasi adalah warna merah; bila nomor 2 atau 5 maka *channel* warna hijau, dan bila nomor 3 atau 6 maka *channel* warna biru
4. Berdasarkan *channel* indikator yang terpilih sebelumnya (merah, hijau, atau biru), lihat nilai 3-bit MSB yang ada pada channel tersebut. 3 bit MSB ini akan dijadikan acuan pengambilan pesan dengan aturan sebagai berikut:

Nilai MSB Pada Channel Indikator	Jumlah Pengambilan Bit (Channel Merah)	Jumlah Pengambilan Bit (Channel Hijau)	Jumlah Pengambilan Bit (Channel Biru)
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

Tabel 3. Aturan pengambilan pesan pada algoritma LSB

- Lakukan operasi XOR antara LSB *channel* warna saat ini (sesuai ketentuan pada langkah 4) dengan bit *channel* warna pada urutan hasil PRNG sebelumnya. Misal hasil PRNG adalah 6, maka lakukan operasi LSB *channel* warna saat ini XOR bit *channel* biru urutan ke 6. Untuk kejelasan pada langkah selanjutnya, sebut saja bit *channel* biru urutan ke 6 ini sebagai bit indikator
- Hasil XOR dari langkah 5 sebelumnya kemudian disimpan sebagai bit pesan (dan digabungkan dengan bit pesan lainnya). Ulangi langkah 5 dan 6 dengan LSB dan *channel warna* yang berbeda-beda. Bit indikator yang di-XOR dengan LSB selalu tetap selama belum berganti piksel
- Lakukan langkah 3-5 secara berulang-ulang selama masih ada bit pesan yang harus disisipkan. Lalu, simpan semua perubahan bit ke gambar stego

### 3.4 Metode Pengujian

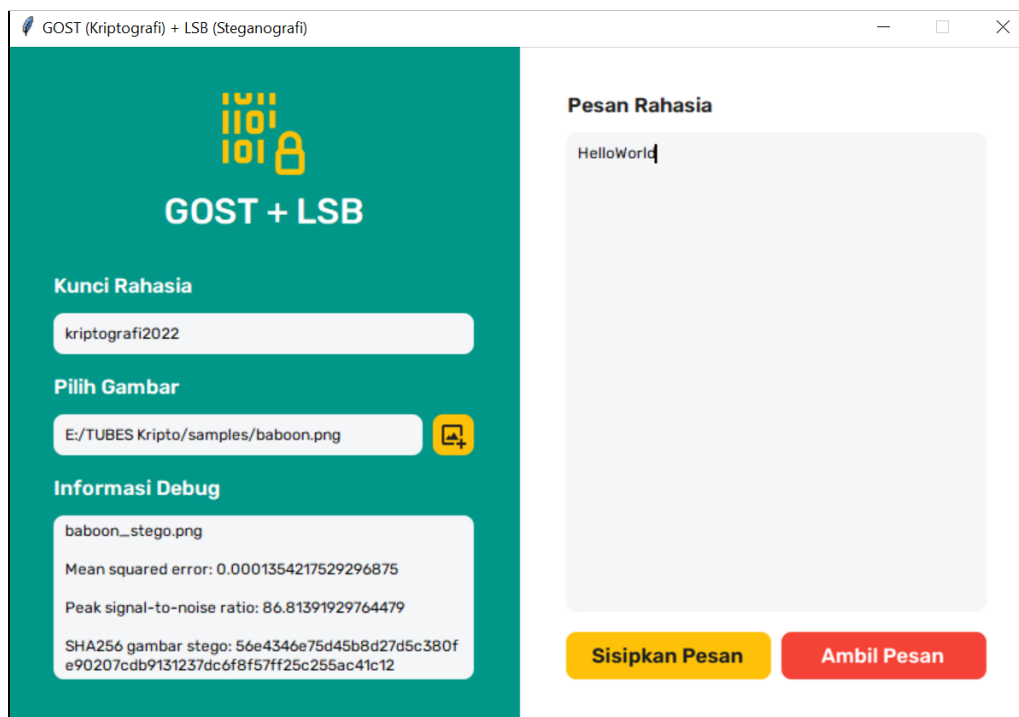
Pengujian dilakukan untuk melihat tingkat keberhasilan dari penyisipan pesan teks pada citra *cover* menggunakan algoritma GOST dan XOR LSB dengan PRNG. Salah satu tingkat keberhasilan didasarkan pada kualitas (*fidelity*) MSE dan PSNR yang didapatkan setiap kali penyisipan pesan dilakukan. Lalu, terdapat juga pengetesan-pengetesan lainnya yang didasarkan pada beberapa aspek dalam kriptografi dan steganografi. Beberapa aspek ini yaitu:

1. *Fidelity*: Kualitas antara gambar *cover* dengan gambar stego tidak jauh berubah. Dapat dibuktikan melalui nilai MSE dan PSNR seperti yang telah dijelaskan sebelumnya
2. *Imperceptibility*: Perubahan sulit untuk dideteksi setidaknya secara kasat mata. Dapat dibuktikan dengan histogram dan survei melalui beberapa orang responden
3. *Capacity*: Semakin besar kapasitas pesan yang dapat disisipkan, maka akan semakin bagus pula algoritma steganografi yang digunakan
4. *Speed*: Kecepatan dalam melakukan proses komputasi algoritma. Semakin cepat maka akan semakin baik
5. *Recovery*: Pesan yang telah disisipkan dapat diambil atau diekstraksi kembali
6. *Robustness*: Kekuatan atau keamanan bila dilakukan berbagai serangan atau modifikasi terhadap gambar stego

## BAB IV

### HASIL DAN ANALISIS

Pada penelitian ini, digunakan algoritma GOST untuk proses enkripsi/dekripsi pesan dan XOR LSB dengan PRNG untuk menyisipkan/mengambil pesan pada gambar. Proses pengujian ini sebagian besar diujikan pada 3 gambar yang berbeda dengan ukuran masing-masing yaitu 512 x 512 piksel (“baboon.png”, “lena.png” dan “peppers.png”). Pesan yang disisipkan yaitu “HelloWorld” dengan kunci yang digunakan untuk GOST dan *seed* PRNG yaitu “kriptografi2022”. Adapun gambar mengenai tampilan program dapat dilihat di bawah ini:






Gambar 2. Tampilan GUI program

Terdapat 4 komponen utama yang terlihat pada program ini, yaitu “Kunci Rahasia”, “Pilih Gambar”, “Informasi Debug”, dan “Pesan Rahasia”. “Kunci Rahasia” merupakan bagian untuk memasukkan kunci yang akan digunakan pada GOST dan PRNG. Lalu, “Pilih Gambar” digunakan untuk memilih gambar yang akan disisipkan atau diambil pesannya. “Informasi Debug” adalah bagian penting lainnya yang akan menunjukkan nilai MSE, PSNR, *checksum* SHA256, dan informasi dimana gambar *stego* disimpan. Terakhir, “Pesan Rahasia” adalah bagian untuk menulis/mengambil pesan pada gambar.



#### 4.1 Pengujian Fidelity dan Imperceptibility

Setelah melakukan proses penyisipan pesan “HelloWorld” dengan kunci “kriptografi2022” pada masing-masing gambar “baboon.png”, “lena.png” dan “peppers.png”, didapatkan hasil berupa gambar *stego*, MSE, dan PSNR. Kemudian, hasil MSE dan PSNR (GOST + XOR LSB dengan PRNG) dibandingkan dengan penelitian-penelitian sebelumnya yaitu menggunakan metode 1-bit LSB, 2-bit LSB, *random bit substitution* [25], dan XOR LSB dengan PRNG (tanpa enkripsi GOST) [8]. Nilai MSE dan PSNR dari semua metode tersebut diberikan masing-masing pada tabel 4, 5, dan 6.

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

Tabel 4. Gambar stego beserta nilai MSE dan PSNR-nya

Gambar Stego	1-bit LSB	2-bit LSB	Random bit substitution	XOR LSB dengan PRNG	Metode saat ini
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
<b>Rata-Rata</b>	0.0000504	0.0001372	0.00676600	0.0000906	0.0001691

Tabel 5. Perbandingan MSE antar metode

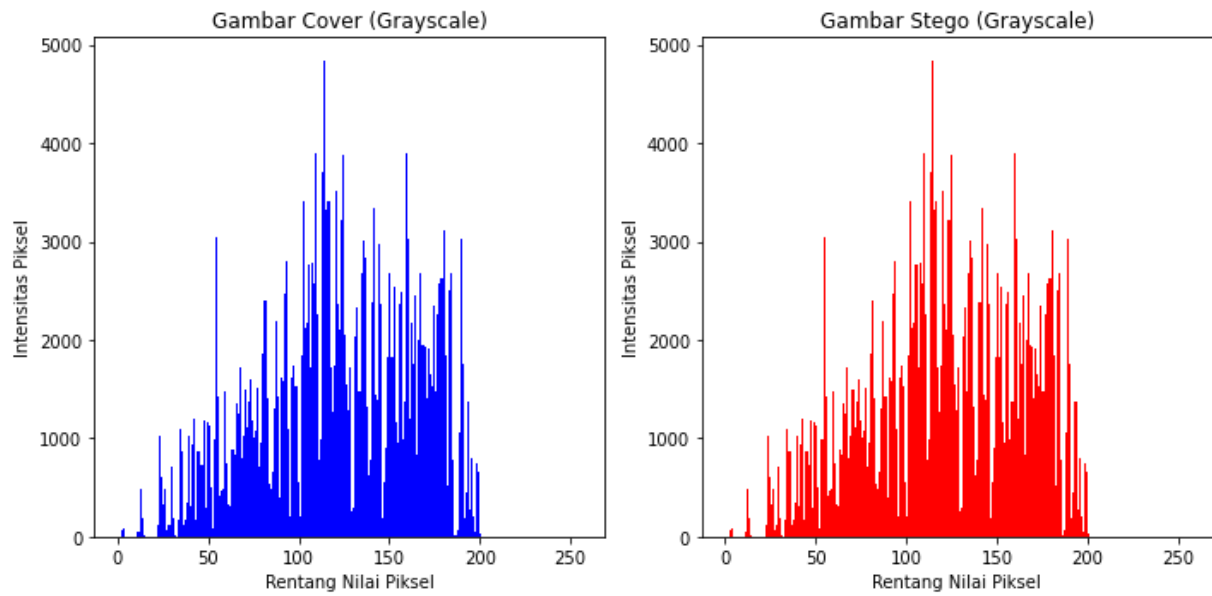
Gambar Stego	1-bit LSB	2-bit LSB	Random Bit Substitution	XOR LSB dengan PRNG	Metode Saat Ini
Baboon.png	90.85	87.26	61.48	89.025	87.230
Lena.png	90.65	86.56	90.18	87.896	85.326
Peppers.png	91.90	86.48	90.09	88.826	85.261
<b>Rata-Rata</b>	91.13	86.76	80.58	88.582	85.939

Tabel 6. Perbandingan PSNR antar metode

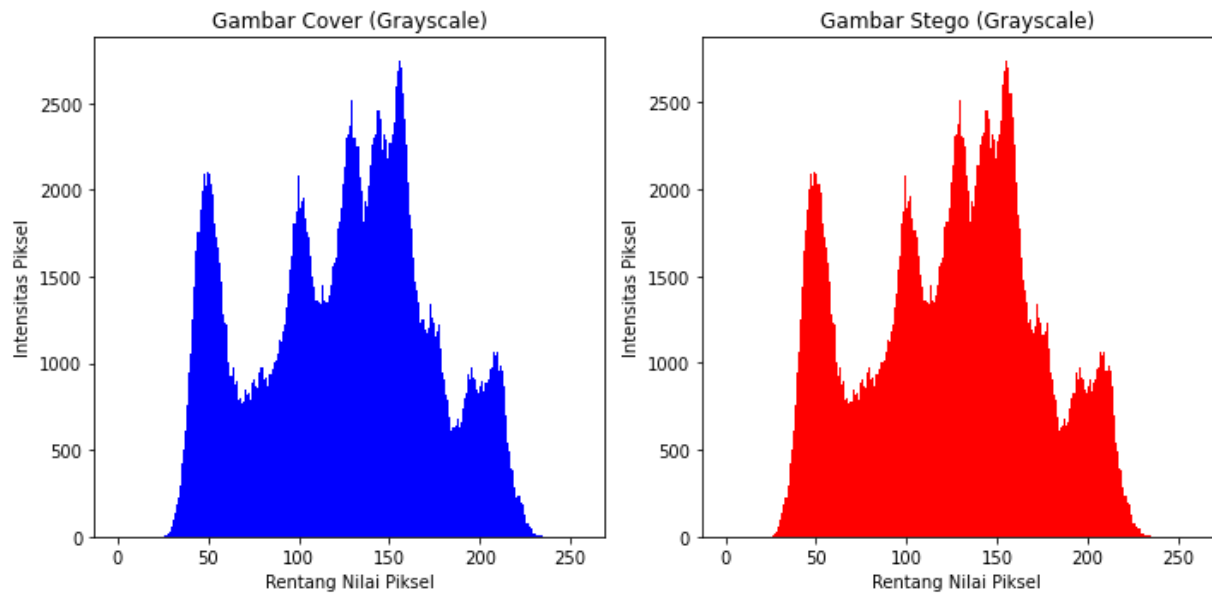
Berdasarkan data hasil eksperimen tersebut, 1-bit LSB (91.13) memiliki PSNR tertinggi jika dibandingkan metode GOST + XOR LSB dengan PRNG (85.939), sedangkan *random bit substitution* (80.58) memiliki PSNR terendah. Hal ini berarti GOST + XOR LSB dengan PRNG memiliki kualitas citra lebih baik dibandingkan metode *random bit substitution*, dan sedikit lebih buruk jika dibandingkan dengan metode lainnya.

Lalu, dilakukan juga pengujian *imperceptibility* terhadap masing-masing gambar *cover* dan *stego* untuk melihat tanda-tanda adanya perbedaan yang signifikan terhadap kedua jenis gambar. Pengujian dilakukan melalui perbandingan histogram (dengan mengkonversi gambar menjadi *grayscale*) serta metode survei dengan responden sebanyak 5 orang. Hasil histogram serta jawaban dari tiap-tiap responden dapat dilihat pada tabel 7 dan 8 di bawah ini.

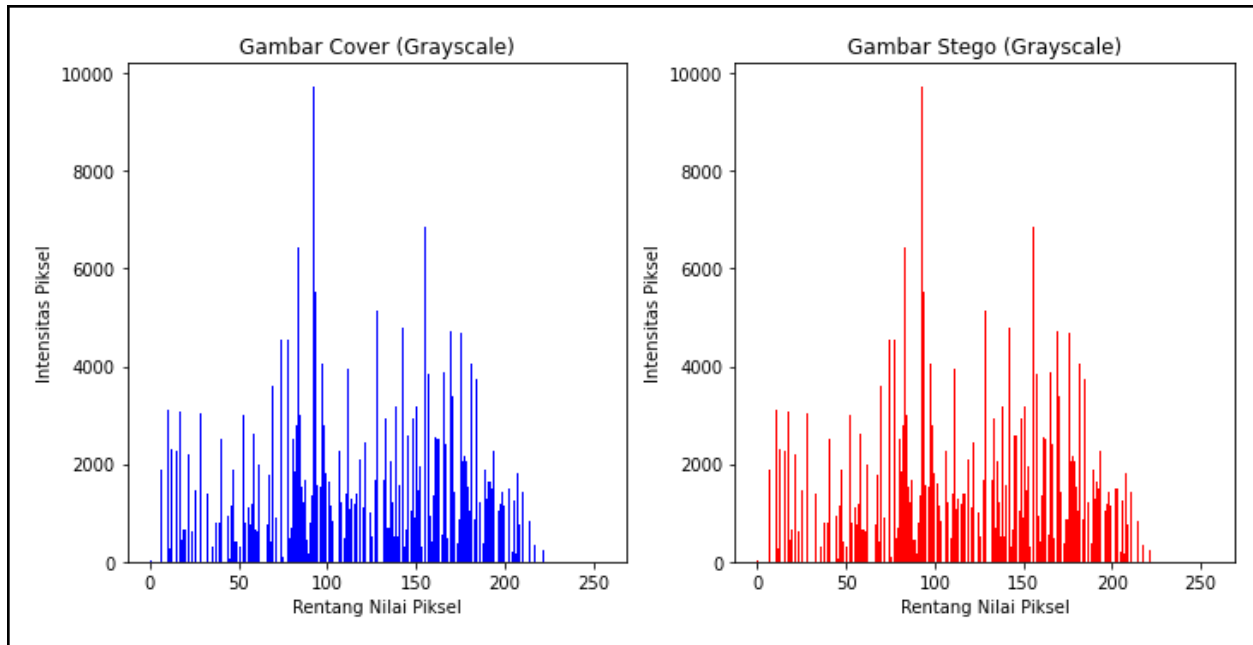
### Histogram Gambar Baboon.png



### Histogram Gambar Lena.png



### Histogram Gambar Peppers.png



Tabel 7. Perbandingan histogram antara gambar cover dan stego

Gambar	Perbedaan Gambar Cover dan Stego
Baboon.png	5 responden merasa tidak ada perbedaan
Lena.png	5 responden merasa tidak ada perbedaan
Peppers.png	5 responden merasa tidak ada perbedaan

Tabel 8. Hasil survei *imperceptibility* terhadap gambar cover dan stego

Berdasarkan histogram-histogram tersebut, terlihat bahwa tidak terdapat perubahan nilai yang signifikan antara piksel-piksel antara gambar cover dengan gambar stego. Lalu, jika dilihat dari hasil survei *imperceptibility*, kelima responden juga konsisten merasa tidak ada perbedaan antara gambar *cover* dan stego untuk file “baboon.png”, “lena.png”, dan “peppers.png”.

## 4.2 Pengujian Capacity

Pengujian yang dilakukan selanjutnya yaitu pengujian kapasitas penyimpanan pesan pada tiap-tiap gambar. Secara teori, metode 1-bit LSB dan *random bit substitution* hanya dapat menyimpan 3 bit pesan per piksel, sedangkan 2-bit LSB dapat menyimpan 6 bit pesan per piksel. Dengan demikian, tersisa 2 metode lainnya yaitu XOR LSB dengan PRNG dan GOST + XOR LSB dengan PRNG yang masing-masing dapat menyimpan 3-6 bit piksel per gambar. Hasil dari pengujian kapasitas terlihat seperti berikut:

Pesan Per X Piksel	1-bit LSB	2-bit LSB	Random Bit Substitution	XOR LSB dengan PRNG	Metode Saat Ini
Pesan Per Piksel Gambar (Teori)	3 bit	6 bit	3 bit	3-6 bit	3-6 bit
Pesan Per 10 Piksel Gambar Lena.png	30 bit	60 bit	30 bit	48 bit	48 bit
Pesan Per 10 Piksel Gambar Baboon.png	30 bit	60 bit	30 bit	38 bit	38 bit
Pesan Per 10 Piksel Gambar Peppers.png	30 bit	60 bit	30 bit	44 bit	44 bit
<b>Rata-Rata</b>	30 bit	60 bit	30 bit	43.33 bit	43.33 bit

Tabel 9. Perbandingan kapasitas bit pesan per piksel antar metode

Berdasarkan hasil pengujian tersebut, terlihat bahwa tidak ada perbedaan antara metode XOR LSB dengan PRNG dan GOST + XOR LSB dengan PRNG (sama-sama dapat menyimpan rata-rata 43.33 bit per 10 piksel gambar). Lalu, dibandingkan 3 metode lainnya (yang dapat menampung masing-masing 30 bit, 60 bit, dan 30 bit), kedua metode XOR LSB dengan PRNG ini menempati posisi median yang berarti tidak terlalu buruk ataupun baik.

#### 4.3 Pengujian Kecepatan

Lalu, bila dilihat dari segi kecepatan algoritmanya, metode GOST + XOR LSB dengan PRNG ini masih terbilang cukup cepat karena dapat memproses penyisipan pesan yang cukup panjang dengan waktu kurang dari 1 detik (dengan rata-rata 0.485 detik). Pengetesan dilakukan melalui bantuan ekstensi “Code Runner” yang dijalankan di Visual Studio Code dengan gambar yang dipakai yaitu “peppers.png”. Detail mengenai kecepatan algoritma dapat dilihat pada tabel 10 di bawah ini.

Karakter Pesan	GOST	GOST + XOR LSB dengan PRNG	XOR LSB dengan PRNG (Hasil Selisih)
----------------	------	-------------------------------	--

1 karakter pesan	0.046 detik	0.481 detik	0.435 detik
10 karakter pesan	0.079 detik	0.493 detik	0.414 detik
50 karakter pesan	0.083 detik	0.473 detik	0.390 detik
100 karakter pesan	0.087 detik	0.478 detik	0.391 detik
500 karakter pesan	0.070 detik	0.476 detik	0.406 detik
1000 karakter pesan	0.071 detik	0.507 detik	0.436 detik
<b>Rata-Rata</b>	0.073 detik	0.485 detik	0.412 detik

Tabel 10. Perbandingan kecepatan algoritma saat ini per karakter pesan

#### 4.4 Pengujian Recovery dan Robustness

Terakhir, dilihat dari segi *recovery* dan *robustness*-nya, ekstraksi pesan kembali pada algoritma GOST + XOR LSB dengan PRNG setelah dilakukan modifikasi pada gambar stego menunjukkan bahwa algoritma LSB ini cukup baik dalam menyisipkan pesan ke format gambar *lossless* (PNG, TIFF, TGA, dll) namun sangat buruk untuk format yang bersifat *lossy* seperti JPEG. Lalu, berdasarkan pengujian modifikasi gambar (rotasi dan *resizing*) juga menunjukkan bahwa LSB bersifat sangat lemah jika dilakukan berbagai modifikasi pada gambar stego. Hasil dari pengujian recovery dan robustness dapat dilihat pada tabel 11 dan 12.

Gambar Stego	MSE	PSNR	Ekstraksi Pesan
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	✗

<b>Median</b>	0.0001935	85.261	-
---------------	-----------	--------	---

Tabel 11. Pengujian ekstraksi pesan pada berbagai format gambar

Gambar Stego	Ekstraksi Pesan (Rotasi)			Ekstraksi Pesan (Resize)			
	90°	180°	270°	10%	30%	50%	80%
Baboon.png	X	X	X	X	X	X	X
Lena.png	X	X	X	X	X	X	X
Peppers.png	X	X	X	X	X	X	X

Tabel 12. Pengujian ekstraksi pesan setelah modifikasi gambar stego

Seperti yang terlihat pada tabel-tabel di atas, hasil ekstraksi pesan gagal pada format GIF dan JPEG dengan PSNR masing-masing yaitu 51.203 dan 33.878. Dari 9 format gambar yang dites, 7 sisanya dapat diekstraksi pesan dengan baik dengan median MSE yaitu 0.0001935 dan median PSNR yaitu 85.261. Lalu, dari segi rotasi, semua rotasi gambar dari 90°, 180°, dan 270° gagal untuk diekstraksi pesan dan dari segi resizing semua ukuran dari 10%, 30%, 50%, dan 80% juga gagal untuk diekstraksi pesan.

## **BAB V**

### **KESIMPULAN**

Setelah melakukan beberapa pengujian hasil dan analisis pada tahapan sebelumnya, dapat ditarik beberapa kesimpulan yaitu:

1. Kombinasi GOST dan XOR LSB dengan PRNG cukup cocok untuk digunakan bersama-sama dalam mengamankan teks ke dalam media gambar yang bersifat *lossless* namun tidak cocok untuk gambar yang bersifat *lossy*.
2. Secara garis besar, proses pada algoritma GOST dan XOR LSB dengan PRNG ini dimulai dari memasukkan plainteks dan kunci, enkripsi plainteks menjadi cipherteks, dan memasukkan cipherteks ke dalam gambar *cover* hingga menjadi gambar *stego* yang telah diisi pesan tersembunyi.
3. Algoritma GOST dan XOR LSB dengan PRNG cukup baik dari segi kualitas gambar, kapasitas pesan, kecepatan penyisipan, dan *imperceptibility*-nya (tidak dapat dilihat secara kasat mata). Meski begitu, algoritma ini masih sangat lemah jika dilakukan berbagai serangan/modifikasi pada gambar *stego* yang dihasilkan (misalnya yaitu rotasi dan *resizing*).



## DAFTAR PUSTAKA

- [1] D. Darwis, “Implementasi Teknik Steganografi Least Significant Bit (LSB) Dan Kompresi Untuk Pengamanan Data Pengiriman Surat Elektronik,” *Jurnal Teknoinfo*, vol. 10, no. 2, hlm. 32–38, Jul 2016, doi: 10.33365/jti.v10i2.8.
- [2] B. Nabila dan L. T. Sianturi, “Implementasi Fungsi Hash Untuk Mendeteksi Orisinalitas File Audio Menggunakan Metode Gost,” *Informasi dan Teknologi Ilmiah (INTI)*, vol. 8, no. 2, hlm. 48–52, Feb 2021, Diakses: 26 Mei 2022. [Daring]. Tersedia pada: <http://ejurnal.stmik-budidarma.ac.id/index.php/inti/article/view/2842>
- [3] S. G. M. Siregar, “Implementasi Metode Enhanced Audio Steganografi (EAS) Untuk Penyembunyian Text Terenkripsi Algoritma Gost,” *KLIK: Kajian Ilmiah Informatika dan Komputer*, vol. 2, no. 1, hlm. 20–27, Agustus 2021.
- [4] M. Diana, “Implementasi Metode GOST (Government Standard) dan LSB-1 (Least Significant Bit) Untuk Mengamankan Teks,” *TIN: Terapan Informatika Nusantara*, vol. 1, no. 7, hlm. 363–382, 2020.
- [5] N. A. Simatupang dan N. A. Hasibuan, “KEAMANAN FILE TEKS MENGGUNAKAN ALGORITMA GOVERNMENT STANDARD (GOST),” *Pelita Informatika: Informasi dan Informatika*, vol. 7, no. 2, hlm. 253–257, Okt 2018, Diakses: 26 Mei 2022. [Daring]. Tersedia pada: <http://ejurnal.stmik-budidarma.ac.id/index.php/pelita/article/view/1091>
- [6] I. Dinur, O. Dunkelman, dan A. Shamir, “Improved Attacks on Full GOST,” dalam *Fast Software Encryption*, Berlin, Heidelberg, 2012, hlm. 9–28. doi: 10.1007/978-3-642-34047-5\_2.
- [7] T. Limbong dkk., “The implementation of computer based instruction model on Gost Algorithm Cryptography Learning,” *IOP Conf. Ser.: Mater. Sci. Eng.*, vol. 420, hlm. 012094, Okt 2018, doi: 10.1088/1757-899X/420/1/012094.
- [8] C. K. Deo, A. Singh, D. K. Singh, dan N. K. Soni, “Developing a Highly Secure and High Capacity LSB Steganography Technique using PRNG,” dalam *2020 International Conference on Computational Performance Evaluation (ComPE)*, Jul 2020, hlm. 136–140. doi: 10.1109/ComPE49325.2020.9200077.

- [9] A. M. Qadir dan N. Varol, "A Review Paper on Cryptography," dalam *2019 7th International Symposium on Digital Forensics and Security (ISDFS)*, Jun 2019, hlm. 1–6. doi: 10.1109/ISDFS.2019.8757514.
- [10] L. Anggriani, "Penerapan Metode Gost Untuk Mendeteksi Keaslian File Dokumen," *TIN: Terapan Informatika Nusantara*, vol. 1, no. 9, hlm. 445–450, 2021.
- [11] J. I. Sari, S. M.kom, dan H. T. Sihotang, "Implementasi Penyembunyian Pesan Pada Citra Digital Dengan Menggabungkan Algoritma Hill Cipher Dan Metode Least Significant Bit (LSB)," *Jurnal Mantik Penusa*, vol. 1, no. 2, Nov 2017, Diakses: 26 Mei 2022. [Daring]. Tersedia pada: <https://e-jurnal.pelitanusantara.ac.id/index.php/mantik/article/view/253>
- [12] G. Hatzivasilis, K. Fysarakis, I. Papaefstathiou, dan C. Manifavas, "A review of lightweight block ciphers," *J Cryptogr Eng*, vol. 8, no. 2, hlm. 141–184, Jun 2018, doi: 10.1007/s13389-017-0160-y.
- [13] L. Pangaribuan, L. J. Pangaribuan, dan D. Sitanggang, "A KEAMANAN PESAN WHATSAPP MENGGUNAKAN KRIPTOGRAFI ALGORITMA GOVERNMENT STANDARD (GOST)," *SKYLANDSEA PROFESIONAL Jurnal Ekonomi, Bisnis dan Teknologi*, vol. 2, no. 1, hlm. 210–217, Jan 2022, Diakses: 26 Mei 2022. [Daring]. Tersedia pada: <https://jurnal.yappsu.org/index.php/skylandsea/article/view/76>
- [14] A. Riski, H. Purwantoro, dan A. Kamsyakawuni, "PENYEMBUNYIAN CIPHERTEXT ALGORITMA GOST PADA CITRA KE DALAM AUDIO DENGAN METODE LEAST SIGNIFICANT BIT," *Jurnal Ilmiah Matematika dan Pendidikan Matematika*, vol. 10, no. 2, hlm. 49–62, Des 2018, doi: 10.20884/1.jmp.2018.10.2.2844.
- [15] Taronisokhi Zebua, "GOST Algorithm (Encryption and Decryption)," 2014, doi: 10.13140/RG.2.2.35980.21125.
- [16] S. Dhawan dan R. Gupta, "Analysis of various data security techniques of steganography: A survey," *Information Security Journal: A Global Perspective*, vol. 30, no. 2, hlm. 63–87, Mar 2021, doi: 10.1080/19393555.2020.1801911.
- [17] L. P. Malese, "Penyembunyian Pesan Rahasia Pada Citra Digital dengan Teknik Steganografi Menggunakan Metode Least Significant Bit (LSB )," *Jurnal Ilmiah Wahana Pendidikan*, vol. 7, no. 5, hlm. 343–354, Sep 2021, doi: 10.5281/zenodo.5563416.

- [18] O. Rachael, S. Misra, R. Ahuja, A. Adewumi, F. Ayeni, dan R. Mmaskeliunas, "Image Steganography and Steganalysis Based on Least Significant Bit (LSB)," dalam *Proceedings of ICETIT 2019*, Cham, 2020, hlm. 1100–1111. doi: 10.1007/978-3-030-30577-2\_97.
- [19] U. Sarah, M. Akter, dan M. S. Uddin, "Image quality assessment through FSIM, SSIM, MSE and PSNR—a comparative study," *Journal of Computer and Communications*, vol. 7, no. 3, hlm. 2019, 8-18.
- [20] Tb. Ai Munandar, Maria Adelvin L, dan Alb. Joko Santoso, "ANALISA PSNR, RASIO KOMPRESI WARNA DAN MSE TERHADAP KOMPRESI IMAGE MENGGUNAKAN 31 FUNGSI WAVELET," *DIGITAL INFORMATION & SYSTEM CONFERENCE*, 2011.
- [21] I. Maulana dan P. N. Andono, "Analisa Perbandingan Adaptif Median Filter Dan Median Filter Dalam Reduksi Noise Salt & Pepper," *CogITO Smart Journal*, vol. 2, no. 2, hlm. 157–166, Des 2016, doi: 10.31154/cogito.v2i2.26.157-166.
- [22] D. Rachmawati, J. T. Tarigan, dan A. B. C. Ginting, "A comparative study of Message Digest 5(MD5) and SHA256 algorithm," *J. Phys.: Conf. Ser.*, vol. 978, hlm. 012116, Mar 2018, doi: 10.1088/1742-6596/978/1/012116.
- [23] Z. Panjaitan, E. F. Ginting, dan Y. Yusnidah, "Modifikasi SHA-256 dengan Algoritma Hill Cipher untuk Pengamanan Fungsi Hash dari Upaya Decode Hash," *J. Sains Manaj. Inform. dan Komput.*, vol. 19, no. 1, hlm. 53, Feb 2020, doi: 10.53513/jis.v19i1.225.
- [24] R. K. Ibrahim, R. A. J. Kadhim, dan A. S. H. Alkhalid, "Incorporating SHA-2 256 with OFB to realize a novel encryption method," dalam *2015 World Symposium on Computer Networks and Information Security (WSCNIS)*, Sep 2015, hlm. 1–6. doi: 10.1109/WSCNIS.2015.7368295.
- [25] U. A. Md. Ehasn Ali, Md. Sohrawordi, dan Md. Palash Uddin, "A Robust and Secured Image Steganography using LSB and Random Bit Substitution ," *American Journal of Engineering Research (AJER)*, vol. 8, no. 2, hlm. 39–44, 2019.

## LOG SHEET

Tanggal	Nama	NIM	Kegiatan
24/04/2022	-	-	Meet Pertama untuk Menentukan Algoritma yang Akan Dipakai dan Dibuat
24/04/2022	Jaya Megelar Cakrawarty	119140227	Review Jurnal dan Menyusun Laporan Awal
24/04/2022	Andhika Wibawa Bhagaskara	119140218	Review Jurnal dan Membuat Kerangka Awal Program GOST
24/04/2022	Perdana Raga Winata	119140087	Review Jurnal dan Membuat Kerangka Awal Program LSB
29/04/2022	Andhika Wibawa Bhagaskara	119140218	Membuat Program GOST Bagian Input Plainteks dan Kunci menjadi Biner, R0, L0, dsb
03/05/2022	Andhika Wibawa Bhagaskara	119140218	Review Algoritma GOST dan Fungsi Pembantu dalam Enkripsi dan Dekripsi (S-Box, Rotate Left Shift, dsb)
04/05/2022	Andhika Wibawa Bhagaskara	119140218	Implementasi Algoritma Enkripsi pada GOST Beserta Fungsi Pembantu
05/05/2022	Andhika Wibawa Bhagaskara	119140218	Melanjutkan Implementasi Algoritma Enkripsi dan Dekripsi pada GOST

05/05/2022	Jaya Megelar Cakrawarty	119140227	Menyusun Laporan Tugas Besar Kriptografi (Pendahuluan)
07/05/2022	Jaya Megelar Cakrawarty	119140227	Menyusun Laporan Tugas Besar Kriptografi (Pendahuluan Lanjutan, Landasan Teori)
08/05/2022	Jaya Megelar Cakrawarty	119140227	Menyusun Laporan Tugas Besar Kriptografi (Metode dan Pembahasan)
12/05/2022	Perdana Raga Winata	119140087	Menyelesaikan Algoritma Steganografi LSB Versi Awal
14/05/2022	-	-	Meet Bersama untuk Membahas Kelanjutan Pengerjaan Tugas Besar
17/05/2022	Andhika Wibawa Bhagaskara	119140218	Review Jurnal LSB dan Menyusun Ulang Algoritma LSB dengan XOR dan PRNG
18/05/2022	Andhika Wibawa Bhagaskara	119140218	Membuat GUI Program dan Menyesuaikan Algoritma GOST dengan LSB
22/05/2022	Andhika Wibawa Bhagaskara	119140218	Memperbaiki GUI Program serta Menambahkan Fungsi Pengecekan MSE, PSNR, dan Checksum (Hash)
25/05/2022	Perdana Raga Winata	119140087	Menyusun Laporan Tugas Besar Kriptografi (Rumusan, Tujuan, Hasil dan Analisis, Kesimpulan)

26/05/2022	-	-	Meet Bersama untuk Mengerjakan Komponen Tugas Besar
26/05/2022	Andhika Wibawa Bhagaskara	119140218	Melakukan Pengujian Pada Gambar Stego (Kecepatan, Serangan, dan Kapasitas)
26/05/2022	Perdana Raga Winata	119140087	Memulai Pembuatan PowerPoint
26/05/2022	Jaya Megelar Cakrawarty	119140227	Melengkapi Laporan, Membuat Jurnal dan PowerPoint
27/05/2022	Andhika Wibawa Bhagaskara	119140218	Melengkapi Laporan Bagian Hasil Pengujian dan Kesimpulan
27/05/2022	Perdana Raga Winata	119140087	Membuat Video Demo Program
28/05/2022	Andhika Wibawa Bhagaskara	119140218	Revisi Metode Pencarian Penanda Pesan Pada LSB Untuk Menghilangkan Limitasi Panjang Pesan Diatas 1024 Karakter
01/06/2022	Andhika Wibawa Bhagaskara	119140218	Melakukan Pengetesan Histogram dan Efektivitas Ekstraksi Pesan Terhadap Berbagai Format Gambar
03/06/2022	Andhika Wibawa Bhagaskara	119140218	Melengkapi Bagian Hasil dan Analisis Untuk Mengakomodir Berbagai Pengetesan Tambahan
04/06/2022	Andhika Wibawa Bhagaskara	119140218	Memperbaiki Bagian Metode Untuk GOST dan LSB
04/06/2022	-	-	Mengerjakan Jurnal dan Finalisasi Tugas Besar

