Jurnal Infomedia Vol. 5 No. 1 Juni 2020 | P-ISSN: 2527-9858
E-ISSN: 2548-1180

Penerapan *Least Significant Bit* untuk Penyisipian Penanda Pada Gambar

Damian Victor Putra Rape Tupen[1], Wahyu Eko Sridaryanto [2], Lukman Hakim[3]

Jurusan Teknik Informatika Universitas Bunda Mulia

[1]murcielagon99@gmail.com, [2]wahyuekosridaryanto@gmail.com, [3]lhakim2710@gmail.com

Abstrak— Gambar menjadi salah satu cara untuk mendokumentasikan suatu informasi yang mudah untuk dilakukan. Penanda atau *tag* menjadi salah satu variabel yang berguna untuk melakukan penyaringan gambar supaya gambar dengan penanda atau *tag* yang bersangkutan akan muncul ketika dilakukan pencarian. Namun saat ini, penanda atau tag pada gambar dapat dilihat dengan mudah oleh siapapun sehingga informasi yang terkandung didalamnya dapat diketahui oleh orang tidak bertanggung jawab. Oleh karena itu, dibuat sebuah aplikasi galeri gambar berbasis *android*. Tujuan dari aplikasi tersebut adalah penerapan *Least Significant Bit* (LSB) untuk melakukan penyisipan penanda atau *tag* dengan pembuktian keberhasilan penyisipan melalui pencarian gambar dengan penanda tertentu pada aplikasi galeri berbasis *android*. *Least Significant Bit* (LSB) diterapkan dalam proses steganografi pada saat melakukan *encode* dan *decode*. Penggunaan *Least Significant Bit* bertujuan untuk meneliti tingkat efektifitas dalam pengimplementasian pada sebuah aplikasi berbasis *android*. Aplikasi ini dibatasi hanya untuk dapat memproses file dengan format PNG. Dari hasil pengujian aplikasi sebanyak 30 kali dengan 30 gambar dengan format PNG yang berbeda dibutuhkan waktu rata-rata 8,17 detik dengan akurasi pencarian sebesar 100% dan saat dilakukannya proses *encode tag* kedalam gambar maka ada kenaikan pada ukuran file 0,2 – 0,6 *KB*. Penelitian yang telah dilakukan dengan penerapan *Least Significant Bit* (LSB) ini diharapkan dapat membantu memberikan informasi tentang efektifitas dari penggunaan metode *Least Significant Bit* (LSB) dalam proses *encode*, *decode* dan *searching* gambar dalam sebuah aplikasi berbasis *android* dan membuka peluang untuk pengembangan aplikasi ini menjadi lebih efektif dan lebih fleksibel.
Kata kunci—gambar, Least Significant Bit, efektifitas, tag, penerapan, steganografi.

Abstract— Images become one of the way to document information that is easy to do. Markers or tags become one of the variables that are useful for filtering images so that images with the relevant markers or tags will appear when searching. But at the moment, markers or tags on images can be seen easily by anyone so that the information contained within them can be known by people malicious intent. Therefore, an android-based image gallery application was created. The purpose of the application is to implement the Least Significant Bit (LSB) to insert markers or tags with proof of the success of insertion through searching images with certain markers on the Android-based gallery application. Least Significant Bit (LSB) is applied in the steganography process when encoding and decoding. The use of Least Significant Bit aims to examine the level of effectiveness in implementing an android-based application. This application is limited only to being able to process files in PNG format. From the application test results as many as 30 times with 30 images with different PNG formats, it takes an average time of 8.17 seconds with a search accuracy of 100% and when the encoding process is inserted into the image there is an increase in the file size of 0.2 - 0, 6 KB. Research that has been done with the application of Least Significant Bit (LSB) is expected to help provide information about the effectiveness of using the Least Significant Bit (LSB) method in the process of encoding, decoding and searching images in an android-based application and opening opportunities for the development of this application to more effective and more flexible.
Keywords—image, Least Significant Bit, effectiveness, tags, application, steganography.

I. PENDAHULUAN

Image atau gambar yang lebih dikenal dengan citra dalam hal ilmu pengetahuannya merupakan gambar dua dimensi yang dihasilkan dari gambar analog dua dimensi yang diteruskan menjadi gambar diskrit melalui proses sampling. Alasan untuk menggunakan gambar dalam presentasi atau publikasi multimedia adalah karena lebih menarik perhatikan dan dapat mengurangi kebosanan dibandingkan dengan teks. Gambar dapat meringkas dan menyajikan data kompleks dengan cara yang baru dan lebih berguna. Sering dikatakan bahwa sebuah gambar mampu menyampaikan seribu kata. Tapi, itu hanya berlaku ketika menampilkan gambar yang diinginkan saat diperlukan. Grafis seringkali muncul sebagai *background* (latar belakang) suatu teks untuk menghadirkan kerangka yang mempermanis teks[1].

Penanda atau *Tag* adalah label yang dilampirkan pada seseorang atau sesuatu untuk tujuan identifikasi atau untuk memberikan informasi lain. Selain itu media gambar juga dapat memperjelas penyajian pesan dan informasi sehingga dapat memperlancar dan meningkatkan proses dan hasil belajar[2]. Penanda atau *tag* ini akan disisipkan kedalam gambar demi menyembunyikan informasi yang dimiliki dari pihak yang tidak bertanggung jawab [3].

Berdasarkan hal tersebut, dilakukan penelitian dengan judul "Penerapan *Least Significant Bit* untuk Penyisipan Penanda pada Gambar". Penelitian ini akan dibuat aplikasi berbasis *Android*. *Android* adalah sebuah sistem operasi untuk perangkat mobile berbasis linux yangmencakup sistem operasi, *middleware*, dan aplikasi. *Android* menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka[4]. Dan Secara istilah pengertian aplikasi adalah suatu program yang siap untuk digunakan yang dibuat untuk melaksanankan suatu fungsi bagi pengguna jasa aplikasi serta penggunaan aplikasi lain yang dapat digunakan oleh suatu sasaran yang akan dituju. Menurut kamus *computer* eksekutif, aplikasi mempunyai arti yaitu pemecahan masalah yang menggunakan salah satu teknik pemrosesan data aplikasi yang biasanya berpacu pada sebuah komputansi yang diinginkan atau diharapkan maupun pemrosesan data yang di harapkan. Pengertian aplikasi menurut Kamus Besar Bahasa Indonesia, "Aplikasi adalah penerapan dari rancang sistem untuk

mengolah data yang menggunakan aturan atau ketentuan bahasa pemrograman tertentu"[5].

Sebuah algoritma pencarian image dapat menggunakan *tag* sebagai salah satu sumber referensi informasi untuk membandingkan informasi yang dimiliki oleh *image* tersebut dengan informasi image yang ingin dicari oleh pengguna. Tetapi *tag* ini dapat menjadi sumber masalah karena informasi tersebut dapat dengan mudah dilihat oleh orang lain. Oleh karena itu penulis merasa tertantang untuk mencoba membuat sebuah aplikasi yang dapat melakukan pencarian image berdasarkan *tag* yang disteganografikan kedalam *image* menggunakan algoritma LSB (*Least Significant Bit*). Steganografi dapat membantu orang-orang untuk menyembunyikan informasi yang terkandung dalam *tag* didalam *image* sehingga tidak mudah untuk dilihat oleh orang yang tidak memiliki akses terhadap informasi tersebut.

Steganografi berarti "tulisan terselubung" yang menyembunyikan keberadaan pesan itu sendiri. Steganografi digital menyediakan potensi untuk komunikasi pribadi dan aman yang telah menjadi kebutuhan sebagian besar aplikasi di dunia saat ini. Berbagai operator multimedia seperti audio, teks, video, gambar dapat bertindak sebagai media sampul untuk membawa informasi rahasia.

Steganografi (*Steganography*) berasal dari bahasa Yunani steganos (*hidden*) dan gráphein (*writing*). Jadi, steganografi berarti *hidden writing* (tulisan tersembunyi). Steganografi adalah seni dan ilmu menyembunyikan pesan ke dalam sebuah media dengan suatu cara sehingga selain pengirim dan penerima, tidak ada seorang pun yang mengetahui atau menyadari bahwa sebenarnya ada suatu pesan rahasia. Pada steganografi modern, arti steganografi berkembang menjadi penyembunyian informasi pada sebuah media *file* digital, bisa berupa media gambar, suara ataupun video. Pengguna pertama (pengirim pesan) dapat mengirim media yang telah disisipi informasi rahasia tersebut melalui jalur komunikasi publik, hingga dapat diterima oleh pengguna kedua (penerima pesan). Penerima pesan dapat mengekstraksi informasi rahasia yang ada di dalamnya. Penyembunyian data rahasia ke dalam media digital mengubah kualitas media tersebut[6].

Pembuatan aplikasi dibantu dengan *Android studio* adalah IDE (*Integrated Development Environment*) resmi untuk pengembangan aplikasi *Android* dan bersifat *open source* atau gratis. Peluncuran *Android studio* ini diumumkan oleh *Google* pada 13 mei 2013 pada event *Google I/O Conference* untuk tahun 2013/ Sejak saat itu, *Android Studio* menggantikan *Eclipse* sebagai IDE resmi untuk mengembangkan aplikasi *Android*. *Android studio* sendiri dikembangkan berdasarkan IntelliJ IDEA yang mirip dengan *Eclipse* disertai dengan *ADT plugin* (*Android Development Tools*)[5].

II. METODOLOGI PENELITIAN

A. *Least Significant Bit Algorithm*

Metode LSB: Ini menggantikan bit paling signifikan dari objek penutup dengan pesan rahasia. Ini adalah teknik paling populer dan sederhana ketika berhadapan dengan gambar. Ini memiliki kompleksitas komputasi yang rendah dan kapasitas embedding yang tinggi. Memodulasi LSB tidak menghasilkan perbedaan yang dapat dilihat manusia karena amplitudo perubahan kecil. Oleh karena itu, bagi mata manusia, gambar stego yang dihasilkan akan terlihat identik dengan gambar penutup.

Hal ini memungkinkan transparansi persepsi LSB yang tinggi. Meskipun ini adalah teknik yang sangat sederhana tetapi rentan terhadap kompresi *lossy* dan manipulasi gambar seperti penskalaan, rotasi, pemangkasan dll, dan di samping itu, dari *noise* atau kompresi *lossy*, *stego-image* akan menghancurkan pesan juga. Ini bekerja paling baik ketika *file* gambar lebih besar dari *file* pesan dan jika gambar *grayscale* dengan perubahan gradasi dalam nuansa. LSB dapat berupa bit tipe tetap dan bit variabel[7].

B. *Least Significant Bit Embedding*

LSB adalah bit signifikan terendah dalam nilai *byte* piksel gambar. Steganografi gambar berbasis LSB menanamkan rahasia dalam bit paling tidak signifikan dari nilai piksel dari gambar sampul (CVR). Konsep *Embedding* LSB sederhana. Ini mengeksploitasi fakta bahwa tingkat presisi dalam banyak format gambar jauh lebih besar daripada yang dapat dilihat oleh rata-rata penglihatan manusia. Oleh karena itu, gambar yang diubah dengan sedikit variasi warna akan tidak dapat dibedakan dari aslinya oleh manusia, hanya dengan melihatnya. Dalam teknik LSB konvensional, yang membutuhkan delapan byte piksel untuk menyimpan 1 *byte* data rahasia tetapi dalam teknik LSB yang diusulkan, hanya empat byte piksel yang cukup untuk menampung satu *byte* pesan. Sisa bit dalam piksel tetap sama. Berikut ini menunjukkan interpretasi level bit dari algoritma[8].

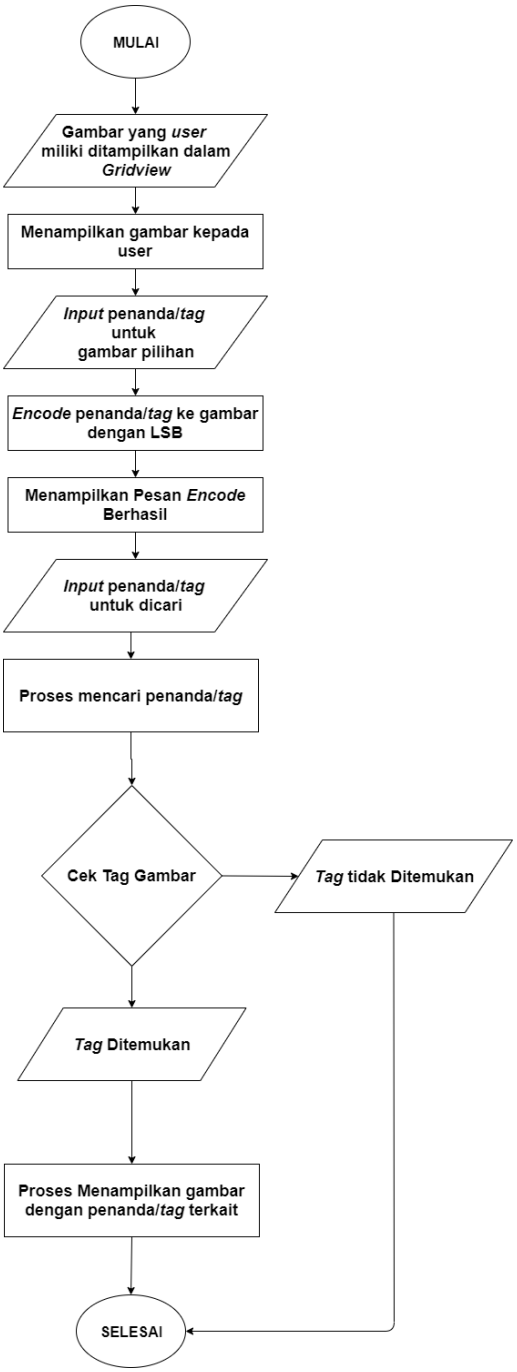
C. Perancangan Aplikasi

1. *Flowchart Diagram* Aplikasi

   Flowchart dapat diartikan sebagai suatu alat atau sarana yang menunjukkan langkah-langkah yang harus dilaksanakan dalam menyelesaikan suatu permasalahan untuk komputasi dengan cara mengekspresikannya ke dalam serangkaian simbol-simbol grafis khusus. Manfaat yang akan diperoleh bila menggunakan *flowchart* dalam pemecahan masalah komputasi[9]:

   a. Terbiasa berfikir secara sistematis dan terstruktur.
   b. Mudah mengecek dan menemukan bagian-bagian prosedur yang tidak valid dan berteletele.
   c. Prosedur akan mudah dikembangkan.

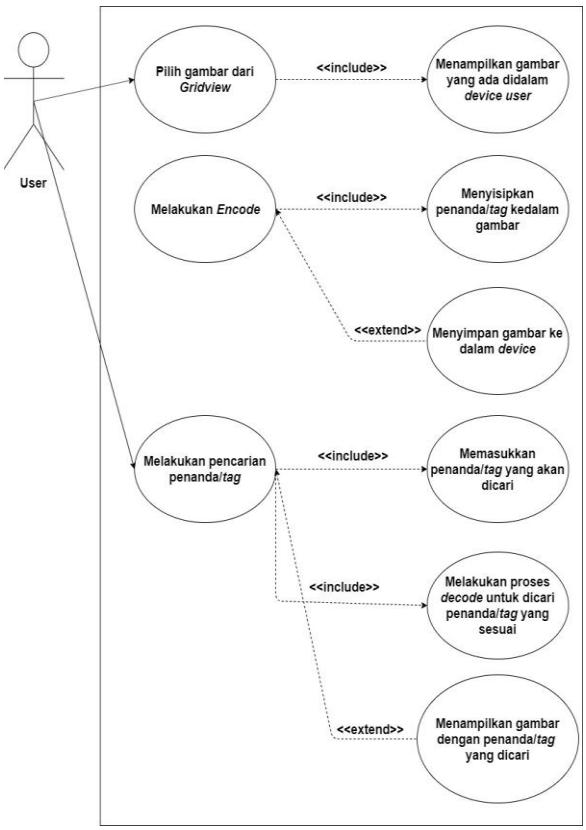
   Flowchart diagram dari aplikasi ini ditujukan pada Gambar 1 yang menjelaskan alur kerja aplikasi secara detail.



Gambar 1. *Flowchart* Aplikasi

2. *Use Case Diagram*

Salah satu diagram penting yang digunakan untuk mengilustrasikan kebutuhan (*requirements*) dari sistem adalah *use case* (UC) diagram, yang menjelaskan secara visual konteks dari interaksi antara aktor dengan sistem[10]. Gambar 2 menjelaskan apa saja yang dapat dilakukan oleh pengguna didalam aplikasi :



Gambar 2. *Use Case Diagram* Aplikasi

Seperti yang terlihat pada Gambar 2, interaksi antara pengguna (*user*) dengan aplikasi adalah sebagai berikut:

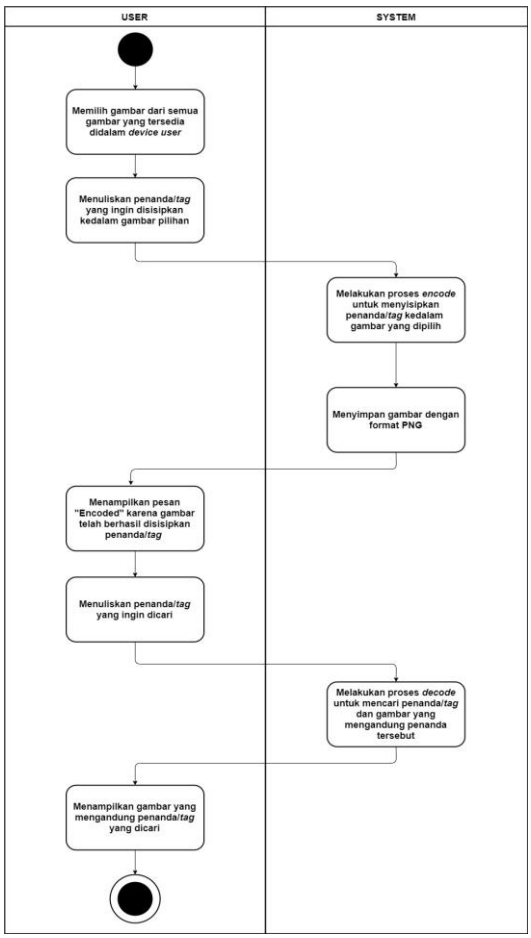
1. Pengguna akan memilih gambar dari *gridview*, dimana semakin banyak gambar semakin lama pula proses menampilkan gambar.
2. Pengguna dapat melakukan *encode* dengan memilih salah satu gambar dari *gridview*, lalu input *tag* yang ingin di masukan ke gambar, kemudian Eksekusi fungsi *encode* dengan menekan tombol *encode*. Dimana proses *encode* akan menanamkan message berupa *tag* pada gambar sehingga gambar tersebut menyimpan informasi *tag* ketika *encode* selesai, gambar disimpan kembali dalam bentuk PNG.
3. Pengguna dapat melakukan searching dengan mengetik/*input tag* yang telah di buat, jika yang diketikan adalah *tag* yang belum ada, maka proses akan tetap berlangsung namun tidak akan menghasilkan keluaran apapun selain message "tag tidak ditemukan". Eksekusi perintah *search* dengan memencet tombol search. Maka aplikasi akan secara otomatis menjalankan fungsi *decode* pada seluruh gambar yang ada pada perangkat, semakin banyak gambar semakin lama pula proses dilakukan. Ketika *tag* sudah di temukan, gambar yang mengandung *tag* tersebut akan di simpan informasi lokasi dan namanya untuk nanti di tampilkan pada *gridview*. Apabila pencarian sudah selesai, aplikasi akan menampilkan seluruh

gambar yang mengandung *tag* tersebut kedalam *gridview*.

4. Pengguna dapat melakukan *decode* dengan memilih suatu gambar yang sudah di-*encode* dan memasukkan key atau kunci saat melakukan *encode* sehingga dapat terlihat *tag* yang di-*encode* kedalam *image* terkait.

3. *Activity Diagram*

Diagram aktivitas dapat digunakan untuk memodelkan alur kerja sistem atau perilaku kompleks sistem atau operasi. Teknik yang diusulkan berfokus pada aktivitas UML diagram yang memodelkan alur kerja dan operasi sistem yang sedang dikembangkan untuk menghasilkan kasus uji [11]. Gambar 3 ini bertujuan untuk menggambarkan aktivitas-aktivitas yang terjadi di dalam sistem atau aplikasi secara sistematis.



Gambar 3. Activity Diagram *user*

Aktivitas yang dilakukan oleh *user* dalam penggunaan aplikasi digambarkan dengan activity diagram yang ditunjukkan pada Gambar 3.

1. Ketika *user* menggunakan aplikasi maka, *user* akan pertama kali memilih gambar yang akan di-*encode* dari semua gambar yang tersedia didalam device.
2. Lalu *user* dapat memilih untuk melakukan *encode* atau *decode*.
3. Jika memilih *encode* maka *user* akan menginput *tag* untuk gambar pilihan dan sebuah *key*.
4. Sehingga sistem dapat melakukan *encode tag* kedalam gambar pilihan dan gambar akan disimpan kedalam format PNG.
5. Jika memilih *decode* maka *user* akan menginput key yang telah dibuat pada gambar yang telah di-*encode* supaya *decode* berhasil dan *Tag* dalam gambar tersebut dapat terlihat.
6. *User* dapat menampilkan pesan gambar tersimpan jika proses *encode* berhasil.
7. *User* dapat menampilkan *Tag* dari gambar yang telah ter-*decode*.
8. *User* dapat menampilkan gambar kedalam *gridview*.
9. *User* dapat menginput *Tag* yang telah dibuat untuk melakukan *searching*.
10. Sistem akan melakukan search ke semua gambar dengan melakukan *decode* setiap gambar dan mencari *tag* yang sama dengan yang dicari. Sehingga semakin banyak gambar yang dimiliki akan semakin lama proses ini berjalan.
11. Setelah proses selesai maka gambar dengan *tag* sesuai akan ditampilkan kepada *user*.

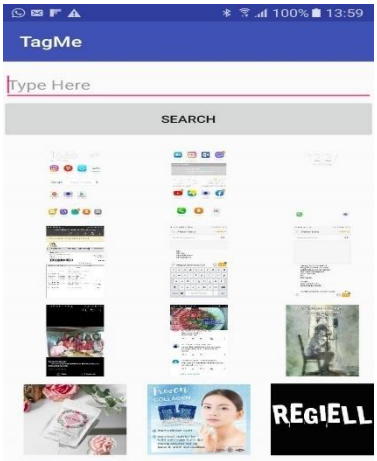
III. HASIL DAN PEMBAHASAN

A. Implementasi Aplikasi

Implementasi aplikasi merupakan tahapan untuk memenuhi kebutuhan *user*, dalam berinteraksi dengan media aplikasi. Fasilitas antar muka yang baik akan membantu *user* untuk melakukan navigasi aplikasi dengan mudah, sehingga *user* dapat memahami proses kerja aplikasi.

1. Tampilan Utama Aplikasi

Ketika *user* menjalankan aplikasi, ditampilkan tampilan awal aplikasi yang akan dijalankan oleh *user*. Pada tampilan aplikasi, *button* ”Search” berguna untuk mencari gambar berdasarkan *Tag* yang telah di-*encode* kedalam gambar yang dipilih. Untuk melakukan *encode user* dapat mengklik gambar yang ingin di-*encode* dan menuliskan isi daripada *Tag* dan *Key* untuk gambar tersebut.



Gambar 4.Tampilan Utama Aplikasi

2. Tampilan Aplikasi saat ingin melakukan *Encode Tag*
Ketika *user* menjalankan aplikasi dan ingin melakukan *encode tag* pada gambar 5. Maka pada tampilan aplikasi ada *button* "ENCODE" yang berguna untuk melakukan *encode Tag* pada gambar menggunakan metode *Least Significant Bit.*

Gambar 5.Tampilan Aplikasi saat akan *encode tag*

3. Tampilan Hasil Pencarian *Tag* Gambar
Ketika *user* menjalankan aplikasi dan telah melakukan proses pencarian gambar menggunakan *tag* gambar tersebut sesuai gambar 6. Maka pada aplikasi ada button "Search" yang berguna untuk menjalankan fungsi search aplikasi.

Gambar 6. Tampilan Aplikasi saat melakukan *search*

B. Pengujian Algoritma
Dalam pengujian ini aplikasi menggunakan *Key* : "12345" maka hasil pengujian algoritma pada aplikasi adalah sebagai berikut :

Sebelum *encode*	Sesudah *encode*	Penanda (Tag)	Hasil gambar	Tipe gambar	Format gambar
712kb	713kb	Sick	Baik	Warna	PNG
455kb	455kb	Hati	Baik	Warna	PNG
320kb	321kb	Gunung	Baik	Warna	PNG
550kb	550kb	Snow	Baik	Warna	PNG
237kb	238kb	Manga	Baik	Warna	PNG
430kb	430kb	Bunga	Baik	Warna	PNG
1.39mb	1.39mb	Anjing	Baik	Warna	PNG
1.44mb	1.44mb	Senja	Baik	Warna	PNG
743kb	744kb	Night	Baik	Warna	PNG
758kb	767kb	Tsuki	Baik	Warna	PNG
1.67mb	1.67mb	Iroha	Baik	Warna	PNG
1.72mb	1.76mb	Bloodlust	Baik	Warna	PNG
1.91mb	1.89mb	Assasin	Rusak	Putih	PNG
590kb	519kb	Putih	Rusak	Hitam	PNG
1.61mb	1.62mb	Kakatua	Baik	Warna	PNG
1.14mb	1.14mb	Sad	Baik	Hitam putih	PNG
434kb	437kb	Kuda	Baik	Warna	PNG
376kb	364kb	Sword	Rusak	Warna	PNG
1.65mb	1.65mb	Litch	Baik	Warna	PNG
679kb	680kb	Bulan	Baik	Hitam putih	PNG
833kb	835kb	Universe	Baik	Warna	PNG
142kb	142kb	Cat	Baik	Hitam putih	PNG
24.96kb	26.54kb	Riot	Rusak	putih	PNG
274kb	274kb	Lubu	Baik	Hitam putih	PNG
182kb	184kb	Sendal	Baik	Hitam putih	PNG
45.38kb	45.52kb	Boom	Baik	Hitam putih	PNG
1.21mb	1.22kb	Nackrot	Baik	Warna	PNG
1.04mb	1.04kb	Zombie	Baik	Warna	PNG
456kb	457kb	Gamma	Baik	Warna	PNG
331kb	331kb	Demon	Baik	Warna	PNG

Tabel 1. Penyisipan penanda(*tag*) pada gambar

Dari pengujian algortima ini ditemukan bahwa gambar berjenis render, akan berubah atau rusak. Contoh seperti gambar 7 dan gambar 8.

Gambar 7. *Render* Sebelum *Encode*

Gambar 8. *Render* sesudah *Encode*

Warna abu-abu pada *background* gambar merupakan pengganti background transparan supaya text didalamnya dapat terlihat. Kotak bewarna hitam pada Gambar 8 menunjukkan matriks pixel yang terambil ketika proses *encode* hal ini yang menyebabkan ukuran gambar sebelum *encode* lebih besar daripada gambar setelah *encode*. Kesimpulan yang dapat ditarik dari percobaan ini adalah apabila gambar yang di proses berformat JPEG, maka ukurannya akan semakin besar karena diubah menjadi PNG, sementara itu bagi gambar yang sudah berformat PNG,

sizenya tidak akan bertambah terlalu banyak. Lain halnya apabila gambar tersebut berjenis render, gambar tersebut akan rusak dan memiliki ukuran yang lebih kecil, hal ini dikarenakan, pada proses LSB, membutuhkan sejumlah matriks *pixel* yang pada gambar berjenis render, tidak ada. Sehingga gambar pun menjadi rusak karena algoritma secara otomatis menciptakan matriks *pixel* baru untuk menggantikan matriks yang tidak ada tersebut.

Berikut adalah hasil pengujian fungsi search dengan menggunakan *Tag* atau penanda yang telah disisipkan sebelumnya ditunjukan oleh Tabel 2.

Penanda(*Tag*)	Lama Pencarian (Detik)	Status Pencarian
Sick	8,39	Ditemukan
Hati	8,12	Ditemukan
Gunung	8,34	Ditemukan
Snow	7,97	Ditemukan
Manga	8,66	Ditemukan
Bunga	8,43	Ditemukan
Anjing	8,14	Ditemukan
Senja	8,97	Ditemukan
Night	7,73	Ditemukan
Tsuki	9,21	Ditemukan
Iroha	7,43	Ditemukan
Bloodlust	8,93	Ditemukan
Assasin	8,75	Ditemukan
Putih	8,47	Ditemukan
Kakatua	9,34	Ditemukan
Sad	7,53	Ditemukan
Kuda	9,42	Ditemukan
Sword	7,89	Ditemukan
Litch	8,11	Ditemukan
Bulan	8,24	Ditemukan
Universe	8,13	Ditemukan
Cat	9,32	Ditemukan
Riot	8,57	Ditemukan
Lubu	8,31	Ditemukan
Sendal	8,63	Ditemukan
Boom	7,71	Ditemukan
Nackrot	8,46	Ditemukan
Zombie	9,17	Ditemukan
Gamma	9,29	Ditemukan
Demon	8,44	Ditemukan

Tabel 2. Hasil Pengujiian fungsi *Search* pada aplikasi

Berdasarkan Tabel 2 dapat disimpulkan bahwa proses rata-rata waktu penyisipan penanda yang telah berhasil dilakukan kepada setiap gambar pada saat uji coba adalah 8,17 detik. Status pencarian dari 30 kali pencarian gambar yang berbeda adalah "Ditemukan" ini menjadi pembuktian bahwa gambar-gambar tersebut berhasil disisipkan sebuah penanda atau *tag* sehingga hasil yang tercatat pada tabel 2 merupakan bukti penguat dari keberhasilan proses penyisipan penanda karena penanda atau tag yang disisipkan dapat dengan sukses digunakan sebagai parameter pencarian gambar.

C. Hasil Pengujian Algoritma pada Aplikasi

Dari pengujian yang dilakukan waktu yang dibutuhkan oleh aplikasi untuk memproses 35 gambar rata-rata selama 8,47 detik dengan akurasi pencarian sebesar 100%. Alasan dibatasinya jumlah gambar yang diproses tertulis pada tabel 3 karena saat aplikasi diuji untuk memproses lebih dari 35 terjadi kesalahan yang dijelaskan pada tabel 3 yang merupakan hasil pengujian dengan jumlah tertentu dan status dari hasil pengujian.

No	Jumlah gambar	Status	Penyebab status muncul
1	100 gambar	*Failed*	*Caused by: java.lang.OutOfMemoryError: Failed to allocate a 1048588 byte allocation with 993376 free bytes and 970KB until OOM*
2	90 gambar	*Failed*	*Caused by: java.lang.OutOfMemoryError: Failed to allocate a 8294412 byte allocation with 845176 free bytes and 825KB until OOM*
3	70 gambar	*Failed*	*Caused by: java.lang.OutOfMemoryError: Failed to allocate a 6553612 byte allocation with 317792 free bytes and 310KB until OOM*
4	50 gambar	*Failed*	*Caused by: java.lang.OutOfMemoryError: Failed to allocate a 1440012 byte allocation with 634288 free bytes and 619KB until OOM*
5	45 gambar	*Failed*	*Caused by: java.lang.OutOfMemoryError: Failed to allocate a 1740800byte allocation with 527384 free bytes and 515KB until OOM*
6	35 gambar	*Success*	Algoritma dalam aplikasi berhasil dijalankan tetapi batas maksimum gambar yang dapat diproses adalah 35 gambar. Jika aplikasi memproses lebih dari 35 gambar maka *OutOfMemoryError akan muncul.*

Tabel 3. Hasil pengujian aplikasi untuk jumlah gambar tertentu

IV. KESIMPULAN

Berdasarkan hasil dan pembahasan, dapat disimpulkan bahwa metode *Least Significant Bit* dapat digunakan pada aplikasi tersebut tapi kurang efektif karena terlalu banyak mengeksekusi atau memproses *file* yang tidak dibutuhkan sehingga saat memproses gambar dengan jumlah yang banyak yaitu diatas 35 gambar dengan format PNG maka akan terjadi error yang disebabkan oleh perangkat tidak dapat mengekseskusi perintah *decode* yang terlalu banyak sehingga terjadi *OutOfMemoryError*. Dimana memori yang akan digunakan untuk mengeksekusi perintah melebihi memori yang tersedia. Hal ini tetap terjadi meski telah ditambahkan fungsi *LargeHeap* dimana ada *manifest* yang memungkinkan aplikasi menjalankan perintah yang tergolong besar atau berat. Hal ini berarti pengeksekusian *decode* pada aplikasi ini tergolong besar

dan karena banyaknya jumlah data atau file yang diproses, menyebabkan proses lebih besar dan lebih berat dari yang seharusnya.

Meskipun ada kekurangan yang dimiliki oleh metode ini tetapi ada keunggulan yang ditawarkan juga. Akurasi pencarian yang dihasilkan oleh aplikasi dengan menggunakan algoritma LSB sebagai algoritma untuk memasukan *tag* kedalam gambar adalah 100%. Dimana tidak mengubah kualitas gambar yang digunakan jika bukan merupakan gamber berjenis render, tidak mengubah ukuran gambar secara spesifik (ukuran gambar berubah hanya sebanyak 0,2 – 0,6 KB) kecuali pada gambar gambar tertentu dan gambar dengan format JPEG karena harus diconvert menjadi format PNG terlebih dahulu dan *tag* sulit diketahui karena ada metode steganografi yang menyembunyikan *tag* dan informasinya didalam gambar.

REFERENSI

[1] J. T. Santoso, "Analysis and Implementation Seo (Search Engine Optimization)," 2010.https://www.neliti.com/publications/249703/analisis-dan-penerapan-metode-seo-search-engine-optimization-image-untuk-meningk

[2] A. Amir, D. P. Matematika, and I. Padangsidimpuan, "Penggunaan Media Gambardalam Pembelajaran Matematika," *J. Eksakta*, vol. 2, no. 1, pp. 34–40, 2016. http://jurnal.umtapsel.ac.id/index.php/eksakta/article/view/184

[3] S. Rohayah, G. W. Sasmito, and O. Somantri, "Aplikasi Steganografi Untuk Penyisipan Pesan," *J. Inform.*, vol. 9, no. 1, pp. 975–981, 2015.Https://Www.Researchgate.Net/Publication/283641542_Aplikasi_Steganografi_Untuk_Penyisipan_Pesan

[4] E. W. Putra, D. W., Nugroho, A. P., & Puspitarini, "Game Edukasi Berbasis Android Sebagai Media Pembelajaran Untuk Anak Usia Dini Dian," *J. Inform. Merdeka Pasuruan*, vol. 1, no. 1, pp. 46–58, 2016.https://www.neliti.com/publications/264570/game-edukasi-berbasis-android-sebagai-media pembelajaran-untuk-anak-usia-dini

[5] A. Juansyah, "Pembangunan Aplikasi Child Tracker Berbasis Assisted – Global Positioning System (A-Gps) Dengan Platform Android Jurnal Ilmiah Komputer Dan Informatika (Komputa)," *J. Ilm. Komput. Dan Inform.*, Vol. 1, No. 1, Pp. 1–8, 2015. Https://Elib.Unikom.Ac.Id/Files/Disk1/673/Jbptunikompp-Gdl-Andijuansy-33648-11-20.Unik-A.Pdf

[6] M. Sitorus, "Teknik Steganography Dengan Metode Least Significant Bit (Lsb)," *J. Ilm. Fak. Tek. Limit's*, Vol. 11, No. 9, Pp. 1689–1699, 2015. Https://Www.Researchgate.Net/Publication/308610177_Teknik_Steganography_Dengan_Metode_Least_Significant_Bit_Lsb

[7] S. Kaur, S. Bansal, and R. K. Bansal, "Steganography and classification of image steganography techniques," *2014 Int. Conf. Comput. Sustain. Glob. Dev. INDIACom 2014*, pp. 870–875, 2014. https://ieeexplore.ieee.org/abstract/document/6828087

[8] A. Agarwal, "LSB(Least Significant Bit) Embedding," 2018. [Online]. Available: https://github.com/aagarwal1012/Image Steganography-Library-Android#lsbleast-significant-bit-embedding. [Accessed: 22-Sep-2019].

[9] R. Nuraini, "Desain Algorithma Operasi Perkalian Matriks Menggunakan Metode Flowchart," *J. Tek. Komput. Amik Bsi*, vol. 1, no. 1, p. 146, 2015. http://garuda.ristekdikti.go.id/documents/detail/534268

[10] T. A. Kurniawan, "Pemodelan Use Case (UML): Evaluasi Terhadap beberapa Kesalahan dalam Praktik," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 5, no. 1, p. 77, 2018. http://jtiik.ub.ac.id/index.php/jtiik/article/view/610

[11] O. Oluwagbemi and H. Asmuni, "Automatic generation of test cases from activity diagrams for UML based testing (UBT)," *J. Teknol.*, vol. 77, no. 13, pp. 37–48, 2015. https://jurnalteknologi.utm.my/index.php/jurnalteknologi/article/view/6358