RANCANG BANGUN APLIKASI *IMAGE WATERMARKING* (LSB-IW) DENGAN METODE *LEAST SIGNIFICANT BIT* PADA *PLATFORM* ANDROID

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar sarjana komputer

Disusun oleh:

Suryawan Pratama

NIM: 105060803111011

TEKNIK INFORMATIKA

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2016

PENGESAHAN

RANCANG BANGUN APLIKASI *IMAGE WATERMARKING* (LSB-IW) DENGAN METODE *LEAST SIGNIFICANT BIT* PADA *PLATFORM* ANDROID

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :

Suryawan Pratama
NIM: 105060803111011

Skripsi ini telah diuji dan dinyatakan lulus pada
19 Januari 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Dr.Eng. Herman Tolle, ST., MT.	Imam Cholissodin, S.Si, M.Kom
NIP. 19740823 200012 1 001	NIK. 850719 16 1 1 0268

Mengetahui
Ketua Program Studi Teknik Informatika

Drs. Marji, MT.
NIP: 196708011992031001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 19 Januari 2016

Suryawan Pratama

NIM: 105060803111011

KATA PENGANTAR

Puji dan syukur penulis panjatkan kehadirat Tuhan yang Maha Esa, yang telah melimpahkan berkat dan anugerah-Nya kepada penulis sehingga penulis dapat menyelesaikan skripsi ini yang berjudul "**Rancang Bangun Aplikasi *Image Watermarking* (LSB-IW) dengan metode *Least Significant Bit* pada *Platform* Android**".

Pada penyusunan Skripsi ini tidak semata-mata hasil kerja penulis sendiri, melainkan juga berkat dan bimbingan dan dorongan dari pihak-pihak yang telah membantu, baik secara materi maupun non materi. Maka dari itu penulis ingin mengucapkan banyak terima kasih yang tak terhingga serta penghargaan yang setinggi-tingginya kepada orang-orang yang telah membantu penulis secara langsung maupun tidak langsung kepada yang terhormat:

1. Bapak Dr. Eng. Herman Tolle, S.T., M.T. selaku dosen pembimbing 1.
2. Bapak Imam Cholissodin, S.Si, M.Kom selaku dosen pembimbing 2.
3. Bapak Drs. Marji, MT. selaku ketua program studi teknik informatika.
4. Orang tua dan anggota keluarga penulis yang telah memberikan dukungan moral dan spiritual.
5. Teman–teman teknik informatika angkatan 2010 yang telah memberikan masukan dan inspirasi kepada penulis selama menyelesaikan skripsi ini.

Penulis menyadari bahwa penulisan skripsi ini jauh dari kata sempurna. Oleh karena itu, penulis mengharapkan masukan berupa saran dan kritik dari semua pihak demi tercapainya kesempurnan dalam skripsi ini. Akhir kata semoga penulisan skripsi ini dapat memberikan manfaat bagi semua pihak.

Malang, 19 Januari 2016

Penulis

105060803111011@mail.ub.ac.id

ABSTRAK

Pertukaran informasi selalu mengalami kenaikan, Kenaikan dari lalu lintas data dan pengguna perangkat bergerak menuntut kinerja yang lebih baik lagi khususnya dalam hal keamanan, terutama dalam pengiriman dan penerimaan informasi. Hak cipta menjadi suatu masalah yang muncul ditengah maraknya distribusi suatu konten digital secara bebas. Sulitnya untuk mengawasi peredaran produk digital melalui media elektronik tentu menjadikan pemegang hak cipta sebagai orang yang paling dirugikan. *Digital Watermarking* mencoba untuk memberikan solusi pada permasalahan ini. Digital watermarking adalah suatu proses untuk meletakkan data yang seringkali disebut watermark atau digital signature pada suatu produk digital. Data ini biasanya bertujuan untuk memverifikasi dan mengidentifikasi pemilik dari suatu konten digital, membuktikan keaslian dari produk itu sendiri, dan informasi tambahan lainnya. Pemilihan metode yang tepat menjadi hal yang penting, karena sangat berpengaruh dalam kualitas file citra yang dihasilkan. Pada penelitian ini, metode *Least Significant Bit Watermarking* adalah metode yang akan digunakan. Metode ini meletakkan data pada *byte* yang kurang berpengaruh dalam suatu konten, hal ini membuat watermark menjadi sulit terdeteksi dan meminimalisir terjadinya perubahan kualitas pada konten digital. LSB mengubah nilai *byte* paling kanan satu lebih tinggi atau satu lebih rendah dari nilai sebelumnya. Proses dari watermark akan menghasilkan *encoded-image*, yaitu *file* citra yang sudah disisipkan *digital signature*.

Kata kunci: Watermark, Steganografi, *Least Significant Bit*

ABSTRACT

The Exchange of information is always increase, increase in data traffic and mobile device users demanding better performance, especially in terms of security, especially in sending and receiving information. Copyright becomes a problem that arises amid rampant distribution of a digital content freely. it is difficult to oversee the distribution of digital products via electronic media would make copyright holders as the most disadvantaged person. Digital Watermarking try to provide a solution to this problem. Digital watermarking is a process for putting the data that is often called a digital watermark or signature on a digital product. This data is typically aims to verify and identify the owner of a digital content, proving the authenticity of the product itself, and other additional information. Selection of appropriate methods becomes important, because it influences the quality of the resulting image file. In this study, the method of Least Significant Bit Watermarking is the method to be used. This puts the data at byte that is less influential in a content, it makes the watermark becomes difficult to detect and minimize the occurrence of changes in the quality of digital content. LSB change rightmost byte value one higher or one lower than the previous value. The process of the watermark will produce the encoded-image, the image file is inserted digital signature.

Keywords: Watermark, Steganography, Least Significant Bit

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR PERSAMAAN

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Dalam beberapa tahun terakhir, pertukaran informasi selalu mengalami kenaikan, khususnya pada lalu lintas data perangkat bergerak. Pada akhir tahun 2014 disebutkan bahwa lalu lintas data perangkat bergerak di dunia mencapai angka 2.5 *exabytes* per bulan, angka ini mengalami kenaikan dari 1.5 *exabytes* per bulan pada akhir tahun 2013. Selain itu, hampir setengah triliun (497 miliar) perangkat bergerak dan koneksi yang ditambahkan pada tahun 2014. Perangkat bergerak dan koneksi yang ada di dunia pada tahun 2014 telah mencapai angka 7.4 triliun, mengalami kenaikan dari 6.9 triliun pada tahun 2013 [CIS-15]. Kenaikan dari lalu lintas data dan pengguna perangkat bergerak menuntut kinerja yang lebih baik lagi khususnya dalam hal keamanan, terutama dalam pengiriman dan penerimaan informasi.

Pengiriman dan penerimaan informasi produk digital seperti *text*, *image*, audio, dan video telah banyak digunakan secara luas. Hal ini menyebabkan mudahnya untuk memperoleh konten digital baik secara legal maupun ilegal. Hak cipta menjadi suatu masalah yang muncul ditengah maraknya distribusi suatu konten digital secara bebas. Sulitnya untuk mengawasi peredaran produk digital melalui media elektronik tentu menjadikan pemegang hak cipta sebagai orang yang paling dirugikan karena ia tidak mendapat royalti dari semua tindakan yang terjadi, hal ini juga menimbulkan kekhawatiran untuk membagikan produk atau konten digital melalui media elektronik.

Digital Watermarking mencoba untuk memberikan solusi pada permasalahan ini. *Digital watermarking* adalah suatu proses untuk meletakkan data yang seringkali disebut *watermark* atau *digital signature* pada suatu produk digital. Data ini biasanya bertujuan untuk memverifikasi dan mengidentifikasi pemilik dari suatu konten digital, membuktikan keaslian dari produk itu sendiri, dan informasi tambahan lainnya. Tentu saja semua informasi ini tidak akan tampak bila dilihat secara langsung oleh mata manusia [MUN-04].

Metode yang akan dipakai adalah *Least Significant Bit Watermarking*. Metode ini meletakkan data pada *byte* yang kurang berpengaruh dalam suatu konten, hal ini membuat *watermark* menjadi sulit terdeteksi dan meminimalisir terjadinya perubahan kualitas pada konten digital. LSB mengubah nilai *byte* paling kanan satu lebih tinggi atau satu lebih rendah dari nilai sebelumnya [MUN-04].

Berdasarkan alasan yang sudah dijelaskan di atas, pada penelitian ini dibuat sebuah aplikasi "*Least Significant Bit Image Watermarking*" (LSB-IW) pada platform *Android*. Pemilihan platform *Android* dikarenakan saat ini *Android* diminati oleh sebagian besar pemakai *smartphone*, bersifat *open source* dan terus melakukan perkembangan yang sangat cepat tiap tahunnya.

1.2 Rumusan Masalah

Dari penjelasan yang terdapat di latar belakang, maka dibuat rumusan masalah sebagai berikut:

1. Bagaimana mengimplementasikan *watermarking* dengan metode *Least Significant Bit* pada *file* citra?
2. Bagaimana rancangan aplikasi *watermarking* pada platform *Android*?
3. Bagaimana analisis hasil *watermarking* meliputi ukuran *file* dan kualitas citra antara sebelum dan sesudah disisipkan?

1.3 Batasan Masalah

Agar masalah yang akan diteliti tidak meluas, maka penelitian ini dibatasi oleh hal-hal sebagai berikut:

1. Pembuatan aplikasi menggunakan *software Android Studio*.
2. Aplikasi bekerja pada *OS Android* versi 4.4 *KitKat* (API 19) atau yang lebih baru.
3. *Watermarking* menggunakan metode *Least Significant Bit*.
4. Media yang digunakan adalah *file* citra.
5. Aplikasi hanya dapat melakukan penyisipan berupa *file* citra.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut:

1. Membuat aplikasi *watermarking* yang dapat melindungi hak cipta *file* citra pada platform *Android*.
2. Mengetahui cara mengimplementasikan *watermarking* dengan metode *Least Significant Bit* pada *file* citra.
3. Mendapatkan hasil analisis *watermarking* meliputi ukuran *file* dan kualitas citra.

1.5 Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini adalah sebagai berikut:

1. Pengguna mampu melindungi hak cipta *file* citra yang dimiliki.
2. Pengguna dapat membuktikan *file* citra yang dimiliki melalui proses verifikasi *watermark*.
3. Pengguna dapat menjaga integritas *file* citra yang dimiliki untuk menghindari upaya pemalsuan *file* citra.

1.6 Sistematika Penyusunan Laporan

Untuk mencapai tujuan yang diharapkan, maka sistematika penulisan yang disusun dalam skripsi ini adalah sebagai berikut:

BAB 1 Pendahuluan

Bab ini berisi tentang latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.

BAB 2 Landasan Kepustakaan

Membahas kajian pustaka pada penelitian sebelumnya dan teori-teori yang mendukung dalam perancangan dan pembangunan aplikasi *Least Significant Bit Image Watermarking* pada platform *Android*.

BAB 3 Metodologi

Membahas tentang metode yang digunakan dalam penulisan yang terdiri dari studi literatur, perancangan perangkat lunak, implementasi perangkat lunak, pengujian dan analisis.

BAB 4 Perancangan

Membahas tentang perancangan aplikasi *Least Significant Bit Image Watermarking* pada platform *Android* dan hasil perancangan sistem

BAB 5 Implementasi

Membahas tentang implementasi sistem aplikasi.

BAB 6 Pengujian dan Analisis

Memuat tentang hasil pengujian dan analisis terhadap aplikasi yang telah dibangun.

BAB 7 Penutup

Memuat kesimpulan yang diperoleh dari pembuatan dan pengujian aplikasi yang dikembangkan dalam skripsi ini, beserta saran yang bersifat membangun.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini terdiri dari kajian pustaka dan dasar teori. Kajian pustaka adalah membahas penelitian yang telah ada dan yang diusulkan. Dasar teori membahas teori yang diperlukan untuk menyusun penelitian yang diusulkan.

2.1 Kajian Pustaka

Kajian pustaka pada penelitian ini adalah membandingkan penelitian *watermark* sebelumnya yang berjudul "*Analysis of Image Watermarking Using Least Significant Bit Algorithm* [SHA-12]" dengan "LSB *Based Digital Image Watermarking for Gray Scale Image* [CHO-12]". Selain itu pada bagian ini juga melakukan perbandingan terhadap metode yang akan diimplementasikan yaitu "*Analysis of Image Watermarking*: LSB *Modification and Spread-Spectrum Technique* [SHU-12]" dengan "Perbandingan Steganografi Metode *Spread Spectrum* dan *Least Significant Bit* (LSB) Antara Waktu Proses dan Ukuran File Gambar [PAK-10]". Perbedaan antara implementasi digambarkan pada Tabel 2.1.

Tabel 2.1 Kajian Pustaka

No.	Judul	Objek	Metode	Hasil
1.	*Analysis of Image Watermarking Using Least Significant Bit Algorithm* [SHA-12].	Citra	*Least Significant Bit*	*Watermark* ditanamkan pada data citra hanya dengan mengganti LSB dan mendapatkan citra *watermark* tanpa adanya distorsi yang terlihat jelas. Akan tetapi ketika *watermark* ditanamkan pada bit kedua dan seterusnya, mulai terjadi distorsi pada data citra.
2.	LSB *Based Digital Image Watermarking for Gray Scale Image* [CHO-12].	Citra	*Least Significant Bit*	Pada penelitian ini dilakukan beberapa macam serangan untuk menguji kekuatan dan kualitas citra *watermark* yang menggunakan metode LSB. Dengan menggunakan metrik *Mean Square Error* (MSE) dan *Peak Signal to Noise Ratio* (PSNR) didapat hasil bahwa citra *watermark* tidak mengalami distorsi atau penurunan kualitas yang berarti. Salah satu contohnya pada pola serangan *Salt and Paper Noise*, terjadi penurunan

				pada PSNR dan MSE dari 141.40 dan 142.25 menjadi 26.63 dan 140.33.
3.	*Analysis of Image Watermarking*: LSB *Modification and Spread-Spectrum Technique* [SHU-12].	Citra	*Least Significant Bit, Spread Spectrum*	1. LSB merupakan metode yang digunakan oleh kebanyakan program untuk menyembunyikan pesan pada 24-bit, 8-bit citra berwarna dan hitam putih. Pada umumnya perubahan yang terjadi pada LSB warna tidak bisa dideteksi karena *noise* yang selalu ada pada citra digital. 2. *Spread Spectrum* bekerja dengan cara mengubah data yang ingin disembunyikan menjadi *noise* dalam sebuah citra digital dan diperlukan sebuah kode untuk mengubah kembali *noise* menjadi data.
4.	Perbandingan Steganografi Metode *Spread Spectrum* dan *Least Significant Bit* (LSB) Antara Waktu Proses dan Ukuran File Gambar [PAK-10].	Citra	*Least Significant Bit, Spread Spectrum*	1. Dari hasil pengujian, perbandingan ukuran *file* citra dengan menggunakan metode LSB dan *Spread Spectrum* adalah sama. 2. Dari hasil pengujian, perbandingan waktu *embedding* dan ekstraksi menggunakan metode LSB dan *Spread Spectrum* menunjukkan hasil bahwa metode LSB lebih cepat daripada metode *Spread Spectrum*. Pada gambar BMP, waktu *embedding Spread Spectrum* adalah 40.83 detik dan waktu *embedding* LSB adalah 16.14 detik sedangkan waktu ekstraksi *Spread Spectrum* adalah 40.63 detik dan waktu ekstraksi LSB adalah 14.30 detik.

Pada proses *watermark* citra, metode *Least Significant Bit* (LSB) lebih sering digunakan jika dibandingkan dengan *Spread Spectrum*. Metode LSB memiliki proses *embedding* dan proses ekstraksi hingga kurang lebih 24 detik lebih cepat daripada *Spread Spectrum* pada gambar BMP. Meskipun demikian, metode LSB memiliki tingkat keamanan lebih rendah dibandingkan dengan *Spread Spectrum* yang memiliki tingkat kompleksitas lebih tinggi dalam perhitungannya. Kelemahan lain dari metode LSB adalah ketidakmampuan dalam menyimpan data dalam ukuran besar, yaitu maksimal hanya seperdelapan dari ukuran *cover object* [HAK-13].

2.2 Data dan Informasi

Data adalah kumpulan dari fakta yang harus dikelola untuk menghasilkan suatu informasi [IND-11]. Data dapat diperoleh dari hasil pengukuran atau pengamatan yang bentuknya padat berupa angka, kata-kata, maupun fakta. Dalam dunia komputer, data adalah segala sesuatu yang dapat disimpan dalam memori dengan format tertentu.

Informasi adalah data yang diolah sehingga menghasilkan pengetahuan yang bermanfaat bagi penggunanya. Hasil pengolahan data dapat meliputi hasil gabungan, hasil analisis, dan hasil penyimpulan. Informasi juga bisa dihasilkan melalui proses komputasi. Sarana yang digunakan dalam melakukan proses pengumpulan, penyimpanan, dan penyebaran informasi disebut teknologi informasi.

2.3 *File*

File atau berkas adalah sekumpulan data (informasi) yang berhubungan dan memiliki ekstensi. Ekstensi *file* berfungsi sebagai jenis berkas. Ekstensi ditulis setelah nama *file* dipisahkan dengan sebuah tanda titik. Ekstensi *file* citra dapat berupa JPG/JPEG, PNG, BMP, maupun GIF [KAD-11]. Berikut adalah penjelasan dari beberapa ekstensi *file* citra:

a. *Bitmap* (BMP)

Tipe *file* BMP umum digunakan pada sistem operasi Windows. Kelebihan tipe *file* BMP adalah dapat dibuka oleh hampir semua program pengolah citra. Baik *file* BMP yang terkompresi maupun tidak terkompresi, *file* BMP memiliki ukuran yang jauh lebih besar daripada tipe-tipe yang lain.

File BMP cocok digunakan untuk:

1. *Desktop background* di Windows.
2. Sebagai gambar sementara sebelum dilakukan pengeditan ulang.

File BMP tidak cocok digunakan untuk:

1. *Web* atau *blog*, perlu dikonversi menjadi JPG, GIF, atau PNG.
2. Aplikasi *chat* Android yang melakukan kompresi pada *file* citra, seperti Blackberry *Messenger*, Whatsapp, atau LINE.

b. *Portable Network Graphics* (PNG)

Tipe *file PNG* merupakan solusi kompresi yang *powerful* dengan warna yang lebih banyak (24 *bit RGB* + *alpha*). Berbeda dengan JPEG yang menggunakan teknik kompresi yang menghilangkan data, *file* PNG menggunakan kompresi yang tidak menghilangkan data (*lossless compression*). Kelebihan *file* PNG adalah adanya warna transparan dan *alpha*. Warna *alpha* memungkinkan sebuah gambar transparan, tetapi gambar tersebut masih dapat dilihat mata seperti samar-samar atau bening. *File* PNG dapat diatur jumlah warnanya 64 *bit* (*true color* + *alpha*) sampai indeks *color* 1 *bit*. Dengan jumlah warna yang sama, kompresi *file* PNG lebih baik daripada *GIF*, tetapi memiliki ukuran *file* yang lebih besar daripada JPG.

File PNG cocok digunakan untuk:

1. Gambar yang memiliki banyak warna.
2. Mengedit gambar tanpa menurunkan kualitas.

c. *Graphics Interchange Format* (GIF)

Tipe *file* GIF memungkinkan penambahan warna transparan dan dapat digunakan untuk membuat animasi sederhana. *File* ini menggunakan kompresi yang tidak menghilangkan data (*lossless compression*).

File GIF cocok digunakan untuk:

1. Gambar dengan jumlah warna sedikit.
2. Gambar yang memerlukan perbedaan warna yang tegas seperti logo tanpa gradien.
3. Gambar animasi sederhana.

File GIF tidak cocok digunakan untuk:

1. Gambar yang memiliki banyak warna seperti pemandangan.
2. Gambar yang di dalamnya terdapat warna *gradient* atau gradasi.

d. *Joint Photographic Experts Group* (JPEG)

Tipe *file* JPEG sangat sering digunakan untuk *web* atau *blog*. *File* JPEG menggunakan teknik kompresi yang menyebabkan kualitas gambar turun (*lossy compression*). Setiap kali menyimpan ke tipe JPEG dari tipe lain, ukuran gambar biasanya mengecil, tetapi kualitasnya turun dan tidak dapat dikembalikan lagi. Ukuran *file* dapat turun mencapai sepersepuluhnya setelah dikonversi menjadi JPEG. Meskipun dengan penurunan kualitas gambar, pada gambar-gambar tertentu (misalnya pemandangan), penurunan kualitas gambar hampir tidak terlihat mata.

File JPEG cocok digunakan untuk:

1. Gambar yang memiliki banyak warna, misalnya foto wajah dan pemandangan.
2. Gambar yang memiliki gradien, misalnya perubahan warna yang perlahan-lahan dari merah ke biru.

File JPEG tidak cocok digunakan untuk:

1. Gambar yang hanya memiliki warna sedikit.
2. Gambar yang memerlukan ketegasan garis seperti logo.

2.4 Citra Digital

Citra digital merupakan sebuah susunan yang berisi nilai-nilai *real* maupun kompleks yang direpresentasikan dengan deretan *bit* tertentu [PUT-10]. Citra digital (diskrit) dihasilkan dari citra analog (kontinu) melalui digitalisasi. Digitalisasi citra analog terdiri atas *sampling* dan kuantisasi *(quantization)*. *Sampling* adalah pembagian citra ke dalam elemen-elemen diskrit *(pixel)*, sedangkan kuantisasi adalah pemberian nilai intensitas warna pada setiap pixel dengan nilai yang berupa bilangan bulat.

Berdasarkan warna-warna penyusunnya, citra digital dapat dibagi menjadi tiga macam [PUT-10] yaitu:

a. Citra biner, yaitu citra yang hanya terdiri atas dua warna, yaitu hitam dan putih. Oleh karena itu, setiap *pixel* pada citra biner cukup direpresentasikan dengan 1 *bit*. Berikut adalah contoh gambar citra biner yang ditunjukkan pada Gambar 2.1.

Gambar 2.1 Citra biner

Sumber: [PUT-10]

b. Citra *grayscale*, yaitu citra yang hanya memiliki satu nilai pada setiap *pixel* dan digunakan untuk menunjukkan tingkat intensitas. Nilai intensitas paling rendah merepresentasikan warna hitam dan nilai intensitas paling tinggi merepresentasikan warna putih. Berikut adalah contoh gambar citra *grayscale* yang ditunjukkan pada Gambar 2.2.

Gambar 2.2 Citra *grayscale*

Sumber: [PUT-10]

c. Citra berwarna, yaitu citra yang nilai *pixel*-nya merepresentasikan warna tertentu. Banyaknya warna yang mungkin digunakan bergantung kepada kedalaman *pixel* citra yang bersangkutan. Citra berwarna direpresentasikan dalam beberapa kanal (*channel)* yang menyatakan komponen-komponen warna penyusunnya. Berikut adalah contoh gambar citra berwarna yang ditunjukkan pada Gambar 2.3.

Gambar 2.3 Citra berwarna

Sumber: [PUT-10]

2.5 Steganografi

Kata Steganografi berasal dari Bahasa Yunani. *Steganos* yang artinya 'tersembunyi/ terselubung', dan *graphien* berarti 'menulis' sehingga kurang lebih artinya "menulis (tulisan) terselubung" [SUK-02]. Steganografi merupakan suatu teknik untuk menyembunyikan informasi yang bersifat rahasia dengan sesuatu yang hasilnya akan tampak seperti informasi normal lainnya.

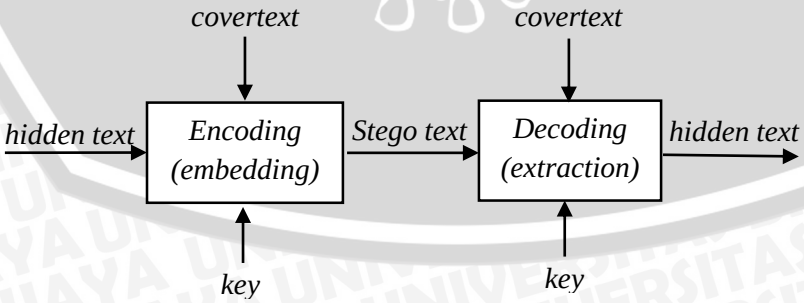
Teknik ini membuat orang lain tidak sadar bahwa informasi penting yang dikirim telah tersembunyi di dalam media lain, seperti citra, audio, maupun video. Seandainya informasi yang telah disembunyikan pada suatu media dicuri, oknum pencuri belum tentu bisa mengetahui informasi yang ada di dalamnya. Hal tersebut disebabkan adanya sandi yang dibutuhkan untuk membuka informasi yang terkandung dalam media, sedangkan sandi hanya diketahui oleh pengirim dan penerima.

Menurut Munir [MUN-06], kriteria steganografi yang baik adalah:

1. *Imperceptible*, yaitu keberadaan pesan tidak dapat dipersepsi oleh indera manusia.
2. *Fidelity*, Mutu citra penampung tidak jauh berubah. Setelah penambahan data rahasia, citra hasil steganografi masih terlihat dengan baik. Pengamat tidak mengetahui kalau di dalam citra tersebut terdapat data rahasia.
3. *Robustness*, pesan yang disembunyikan harus tahan terhadap manipulasi yang dilakukan pada citra penampung (seperti pengubahan kontras, penajaman, kompresi, rotasi, perbesaran gambar, pemotongan (*cropping*), enkripsi, dan sebagainya). Bila pada citra dilakukan operasi pengolahan citra, maka pesan yang disembunyikan tidak rusak.
4. *Recovery*. Pesan yang disembunyikan harus dapat diungkapkan kembali (*recovery*). Karena tujuan steganografi adalah *data hiding*, maka sewaktu-waktu data rahasia di dalam citra penampung harus dapat diambil kembali untuk digunakan lebih lanjut.

Tujuan dari steganografi adalah merahasiakan keberadaaan sebuah pesan tersembunyi atau informasi. Teknik ini umumnya membuat perubahan tipis pada media digital sehingga tidak menarik perhatian orang yang melihatnya. *File* yang telah disisipi steganografi tidak menunjukkan perubahan yang nyata dalam kualitas dan struktur *file* semula.

Pada Steganografi terdapat empat komponen utama [MUN-06], yaitu seperti ditunjukkan pada Gambar 2.4.



Gambar 2.4 Komponen Steganografi

Sumber: [MUN-06]

Keterangan:

1. *Embedded message (hidden text)*, yaitu pesan yang disembunyikan (dapat berupa teks, audio, video).
2. *Cover-object*, yaitu pesan yang digunakan untuk menyembunyikan *embedded message.*
3. *Stego-object,* pesan yang sudah berisi *embedded message*.
4. *Stego-key,* kunci yang digunakan untuk menyisipkan pesan dan meng-ekstraksi pesan yang disembunyikan dari *stegotext*.

Kelebihan steganografi daripada kriptografi adalah informasi yang disembunyikan tidak menarik perhatian orang lain. Pesan berkode pada kriptografi, walaupun tidak dapat dipecahkan akan menimbulkan kecurigaan karena menghasilkan karakter aneh atau susunan huruf yang sulit dibaca. Oleh karena itu, untuk meningkatkan keamanan informasi yang disembunyikan, seringkali digunakan steganografi dan kriptografi secara bersamaan.

2.6 Metode Steganografi

Terdapat banyak metode yang digunakan dalam melakukan penyembunyian data ke dalam data lainnya. Metode-tersebut tersebut mampu menyembunyikan pesan rahasia tanpa mengubah kualitas dan ukuran *file* citra, sehingga pesan yang disembunyikan tidak akan tampak. Berikut adalah penjelasan mengenai beberapa metode yang digunakan dalam steganografi pada *file* citra.

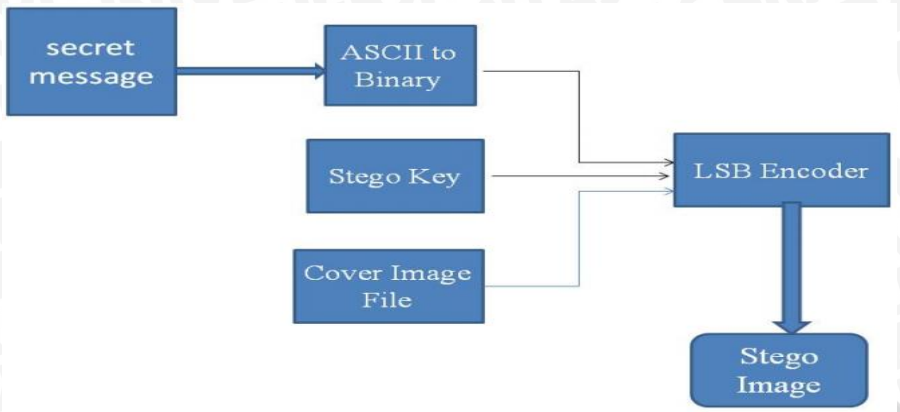
a. *Least Significant Bit Insertion (LSB)*

Penyembunyian data dilakukan dengan mengganti *bit-bit* data yang tidak terlalu berpengaruh di dalam segmen citra dengan *bit-bit* data rahasia. Pada susunan *bit* di dalam sebuah *Byte* (1 *Byte* = 8 *bit*), ada *bit* yang paling berarti (*most significant bit* atau *MSB*) dan *bit* yang paling kurang berarti (*least significant bit* atau *LSB*).

Metode ini menyisipkan pesan rahasia pada *bit* rendah atau *bit* yang paling kanan. Teknik Steganografi Modifikasi LSB dilakukan dengan memodifikasi *bit-bit* yang termasuk *bit* LSB pada setiap *Byte* warna pada sebuah *pixel*. *Bit-bit* LSB akan dimodifikasi dengan menggantikan setiap LSB yang ada dengan *bit-bit* informasi lain yang ingin disembunyikan. Setelah semua *bit* informasi lain menggantikan *bit* LSB di dalam *file* tersebut, maka informasi telah berhasil disembunyikan.

Ketika informasi rahasia ingin kembali dibuka, maka *bit-bit* LSB yang telah ada diambil satu per satu kemudian disatukan kembali menjadi sebuah informasi yang utuh seperti semula. Penentuan *bit-bit* LSB dilakukan secara berurutan, mulai dari *Byte* awal sampai *Byte* terakhir sesuai panjang dari data rahasia yang akan disembunyikan. Mengubah *bit* LSB hanya mengubah nilai *Byte* satu lebih tinggi atau satu lebih rendah dari nilai sebelumnya sehingga tidak berpengaruh terhadap persepsi visual [MAR-12].

Langkah-langkah penyisipan dari *Least Significant Bit Steganography* ditunjukkan pada Gambar 2.5.



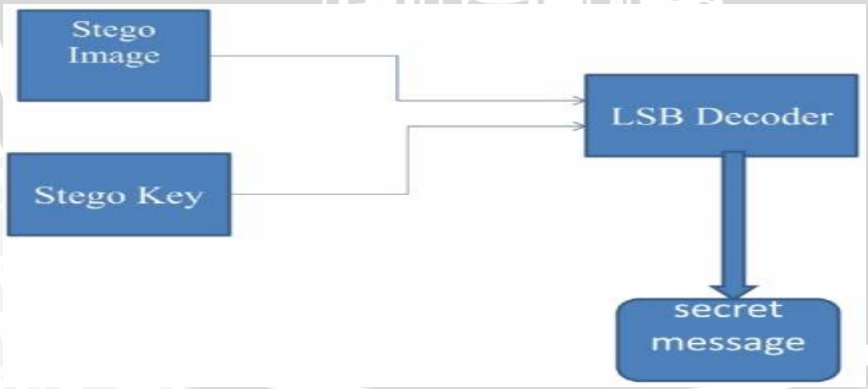
Gambar 2.5 Tahapan LSB *Steganography*

Sumber: [DEV-13]

Keterangan:

1. *Secret message* : Pesan yang akan disembunyikan.
2. *ASCII to Binary* : Merubah pesan dari kode standar Amerika menjadi biner.
3. *Stego Key* : Kunci yang digunakan pada proses enkripsi.
4. *Cover Image File* : Citra yang akan disisipkan pesan.
5. *LSB Encoder* : Algoritma penyisipan pesan pada LSB.
6. *Stego Image* : Citra yang berisi pesan yang disembunyikan.

Langkah-langkah ekstraksi pesan dari *Least Significant Bit Steganography* ditunjukkan pada Gambar 2.6.



Gambar 2.6 Tahapan Ekstraksi Pesan LSB

Sumber: [DEV-13]

Keterangan:

1. *Stego Image* : Citra yang berisi pesan yang disembunyikan.
2. *Stego Key* : Kunci yang digunakan pada proses enkripsi.

3. *LSB Encoder* : Algoritma ekstraksi pesan pada LSB.
4. *Secret Message* : Pesan yang disembunyikan.

b. *Algorithms and Transformation Compression*

Algorithms and Transformation Compression adalah metode steganografi dengan menyembunyikan data dalam fungsi matematika. Dua fungsi tersebut adalah *Discrete Cosine Transformation* (DCT) dan *Wavelet Transformation.* Fungsi DCT dan *Wavelet* yaitu mentransformasi data dari satu tempat *(domain)* ke tempat *(domain)* yang lain. Fungsi DCT yaitu mentransformasi data dari tempat spasial *(spatial domain)* ke tempat frekuensi *(frequency domain)* [MAR-12].

c. *Redundant Pattern Encoding*

Redundant Pattern Encoding adalah sebuah metode dengan menggambar pesan kecil pada kebanyakan gambar. Keuntungan dari metode ini adalah dapat bertahan dari *cropping*, kekurangannya yaitu tidak dapat menggambar pesan yang lebih besar [NUG-10].

d. *Spread Spectrum*

Spread Spectrum adalah sebuah teknik transmisi data dengan menggunakan *pseudonoise code* sebagai *modulator* bentuk gelombang untuk menyebarkan (*spreading*) energi sinyal dalam sebuah jalur komunikasi. Oleh penerima, sinyal dikumpulkan kembali menggunakan replika *pseudonoise code* tersinkronisasi [PAD-13]. Berdasarkan definisi, dapat dikatakan bahwa steganografi menggunakan metode *Spread Spectrum* menambahkan derau semu (*pseudonoise*) ke dalam *cover-object*. Karena sifatnya yang *noise-like*, *Spread Spectrum* sulit untuk terdeteksi, dipotong, atau termodulasi [PRA-12]. Proses penyisipan pesan menggunakan metode *Spread Spectrum* ini terdiri dari tiga proses, yaitu modulasi, *spreading*, dan penyisipan pesan ke citra digital. Sedangkan proses ekstraksi pesan menggunakan metode *Spread Spectrum* juga terdiri dari tiga proses, yaitu pengambilan pesan dari matriks frekuensi, *despreading*, dan demodulasi.

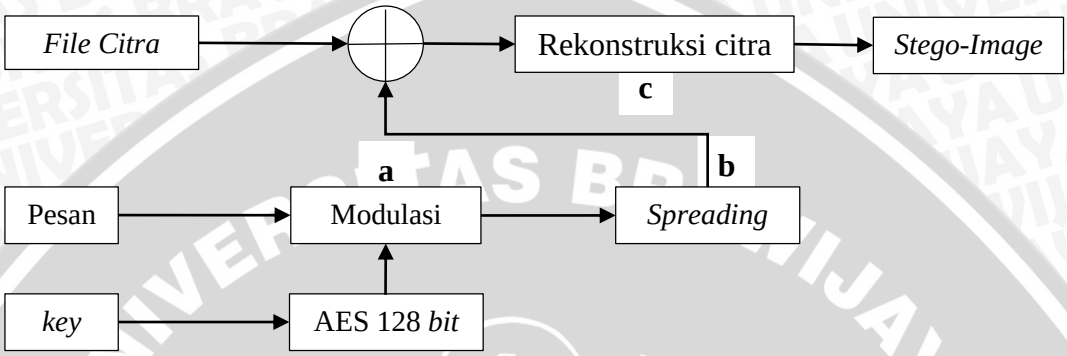
Pada implementasinya, terdapat tiga langkah utama dalam pembentukan *stego-image* [BON-03].

a. Langkah pertama pada *Spread Spectrum Image Steganography* adalah melakukan modulasi, yaitu proses pengacakan pesan melalui enkripsi pada pesan teks dengan algoritma AES 128 *bit*. Setelah terenkripsi, pesan diubah ke dalam bentuk *bit* agar dapat disisipkan ke dalam *file* citra.
b. Langkah selanjutnya adalah proses *spreading*, yaitu menyebarkan *bit-bit* pesan ke dalam *channel* dari *file* citra. Sebagai contoh penyisipan pada sebuah *file* (*.png*) dengan kedalaman 32 *bit* terdiri dari 4 *channel* dimana setiap *channel* merupakan kumpulan dari 8 *bit* atau 1 *Byte* (yang bernilai antara 0 sampai 255 atau dalam format biner antara 00000000 sampai 11111111) yang merepresentasikan nilai intensitas cahaya yang membentuk warna dasar yaitu *Alpha-Red-Green-Blue* atau ARGB. Sebelum *bit-bit* pesan disebar, dilakukan

terlebih dahulu penentuan wilayah penyisipan. Agar tidak mempengaruhi nilai *pixel* dari *cover object* secara drastis, hasil modulasi ditempatkan pada *bit-bit* paling rendah dari *file* citra.

c. Setelah proses penyisipan semua *bit-bit* pesan ke dalam *file* citra, dilakukan proses rekonstruksi citra sehingga menghasilkan *stego-image*.

Langkah-langkah penyisipan dari *Spread Spectrum Image Steganography* ditunjukkan pada Gambar 2.7.



Gambar 2.7 Tahapan *Spread Spectrum Image Steganography*

Sumber: [BON-03]

Keterangan:

1. Pesan : Informasi rahasia yang akan dienkripsi kemudian disisipkan pada *file* citra, format pesan berupa teks.
2. AES 128 *bit* : Algoritma dalam proses enkripsi, menggunakan *Advanced Encryption Standard* (AES) 128 *bit*.
3. *key* : Kunci yang digunakan pada proses enkripsi.
4. Modulasi : Proses pengacakan pesan dengan cara enkripsi.
5. *Spreading* : Proses penyebaran *bit-bit* pesan dan penentuan wilayah penyisipan.
6. *File Citra* : *Cover-object* berupa *file* citra yang akan disisipkan pesan.
7. Rekonstruksi citra : Proses pembentukan *file* citra baru setelah melalui tahap steganografi.
8. *Stego Image* : *File* citra yang telah disisipkan pesan di dalamnya.

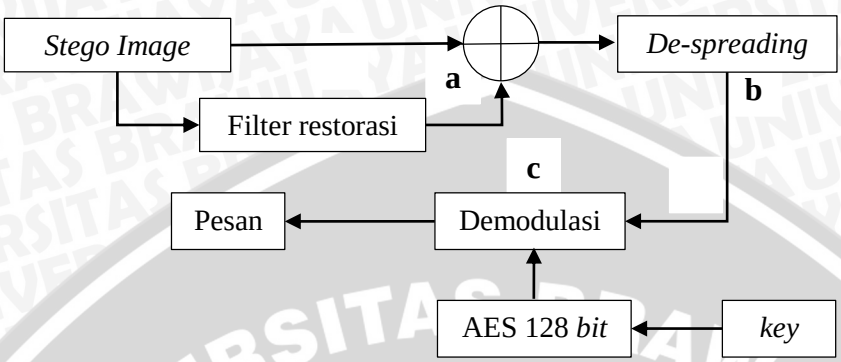
Pada proses ekstraksi terdapat tiga langkah dalam pencarian pesan yang disisipkan:

a. Langkah pertama adalah melakukan penyaringan (*filtering)* pada *channel* dari komponen warna *stego-image*.

b. Langkah selanjutnya adalah melakukan *de-spreading*, atau substraksi (pengambilan *bit-bit*) *stego-image* untuk mendapatkan pesan yang teracak.

c. Langkah terakhir ekstraksi adalah demodulasi, yaitu penyusunan kembali pesan teracak dengan melakukan dekripsi sehingga diperoleh pesan asli.

Langkah-langkah ekstraksi pesan dari *Spread Spectrum Image Steganography* digambarkan lebih jelas pada Gambar 2.8.



Gambar 2.8 Tahapan Ekstraksi Pesan *Spread Spectrum*

Sumber: [BON-03]

Keterangan:

1. Pesan : Pesan asli yang akan diekstrak.
2. AES128 *bit* : Algoritma dalam dekripsi, menggunakan *Advanced Encryption Standard* (AES) 128 *bit*.
3. *key* : Kunci yang digunakan dalam proses dekripsi.
4. *De-spreading* : Pengembalian *bit-bit* dari *stego-image*.
5. Demodulasi : Proses penyusunan kembali pesan yang telah teracak dengan cara melakukan proses dekripsi
6. *Stego Image* : *File* citra yang disisipkan pesan terenkripsi dan telah melalui proses steganografi.
7. *Filter* restorasi : Proses penyaringan *bit-bit* pesan yang telah disisipkan pada *stego-image*.

2.7 *Watermark*

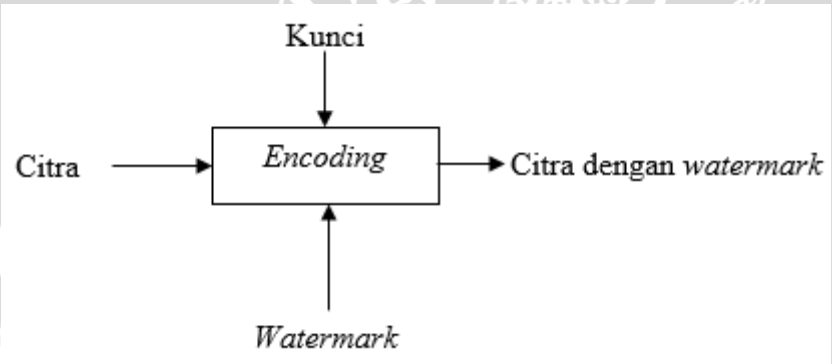
Watermarking sudah ada sejak 700 tahun yang lalu. Pada akhir abad 13, pabrik kertas di Fabriano, Italia, membuat kertas yang diberi *watermark* atau tanda-air dengan cara menekan bentuk cetakan gambar atau tulisan pada kertas yang baru setengah jadi. Ketika kertas dikeringkan terbentuklah suatu kertas yang ber-*watermark*. Kertas ini biasanya digunakan oleh seniman atau sastrawan untuk menulis karya mereka. Kertas yang sudah dibubuhi tanda-air tersebut sekaligus dijadikan identifikasi bahwa karya seni di atasnya adalah milik mereka. Setelah muncul dokumen digital, Berikut adalah contoh *watermark* yang ditunjukkan pada Gambar 2.9.

Gambar 2.9 *Watermark* kasat mata

Sumber: [SAE-10]

Teknik *watermarking* adalah proses menambahkan kode identifikasi secara permanen ke dalam data digital. Kode identifikasi tersebut dapat berupa teks, gambar, suara, atau video. Selain tidak merusak data digital produk yang akan dilindungi, kode yang disisipkan seharusnya memiliki ketahanan (*robustness*) dari berbagai pemrosesan lanjutan seperti pengubahan, transformasi geometri, kompresi, enkripsi, dan sebagainya. Sifat *robustness* berarti data *watermark* tidak rusak akibat pemrosesan lanjutan tersebut [MUN-04].

Proses penyisipan *watermark* ke dalam citra disebut *encoding* seperti ditunjukkan pada Gambar 2.10.



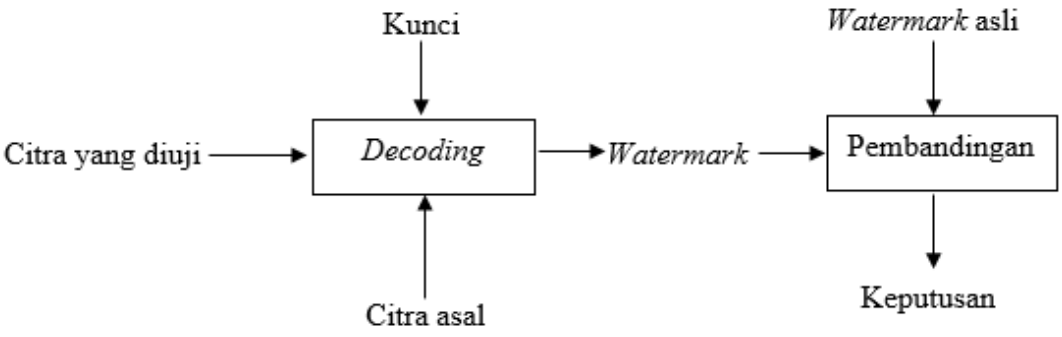
Gambar 2.10 Proses penyisipan *watermark*

Sumber: [MUN-04]

Keterangan:

1. *Encoding* : Proses penyisipan *watermark*.
2. Citra : Gambar yang ingin dilindungi.
3. Kunci : Kunci yang digunakan dalam proses enkripsi.
4. *Watermark* : Informasi yang disisipkan ke dalam data multimedia.
5. Citra *watermark* : Citra yang sudah memiliki sidik digital.

Proses untuk verifikasi *watermark* dilakukan dengan ekstraksi *watermark* dan pembandingan. Proses ekstraksi *watermark* disebut *decoding* dan bertujuan mengungkap *watermark* dari dalam citra. Proses pembandingan bertujuan untuk membandingkan *watermark* yang diungkap dengan *watermark* asli. Proses verifikasi *watermark* ditunjukkan pada Gambar 2.11.



Gambar 2.11 Proses verifikasi *watermark*

Sumber: [MUN-04]

Keterangan:

1. Citra yang diuji : Citra yang memiliki *watermark*.
2. Citra asal : Citra asli sebagai pembanding.
3. *Decoding* : Proses ekstraksi *watermark*.
4. Kunci : Kunci yang digunakan dalam proses dekripsi.
5. *Watermark* : Sidik digital yang dimasukkan pada *file* citra yang terekstraksi.
6. Pembandingan : Proses pembandingan *watermark*.
7. *Watermark* asli : Sidik digital asli.
8. Keputusan : Hasil akhir verifikasi.

Menurut Munir [MUN-04], selain untuk tujuan pelabelan hak cipta, *watermarking* juga dimanfaatkan untuk tujuan lain sebagai berikut:

1. *Tamper-proofing*. *Watermarking* digunakan sebagai alat untuk mengidentifikasi atau menunjukkan bahwa data digital telah mengalami perubahan dari aslinya.
2. *Feature location*. *Watermarking* digunakan untuk mengidentifikasi isi dari data digital pada lokasi-lokasi tertentu.
3. *Annotation / caption. Watermarking* digunakan hanya sebagai keterangan tentang data digital itu sendiri.

2.8 Perbedaan Steganografi dan *Watermark*

Watermarking merupakan aplikasi dari steganografi, namun ada perbedaan antara keduanya. Jika pada steganografi informasi rahasia disembunyikan di dalam media digital dimana media penampung tidak berarti apa-apa, maka pada *watermarking* justru media digital tersebut yang akan dilindungi kepemilikannya dengan pemberian label hak cipta. Meskipun steganografi dan *watermarking* tidak sama, namun secara prinsip proses penyisipan informasi ke dalam data digital tidak jauh berbeda

2.9 *Peak Signal to Noise Ratio*

Peak Signal to Noise Ratio (PSNR) adalah perbandingan antara nilai maksimum dari sinyal yang diukur dengan besarnya *noise* yang berpengaruh pada sinyal tersebut. PSNR diukur dalam satuan desibel [SHA-12]. Pada penelitian ini, PSNR digunakan untuk mengetahui perbandingan kualitas citra *cover* sebelum dan sesudah disisipkan pesan. Untuk menentukan PSNR, terlebih dahulu harus ditentukan MSE (*Mean Squared Error*).

MSE *(Mean Squared Error)* adalah nilai *error* kuadrat rata-rata antara citra *cover* dengan citra hasil proses steganografi, secara matematis dapat dirumuskan seperti persamaan 2.1 [SHA-12]:

$$MSE = \frac{1}{MN} \sum_{y=1}^{M} \sum_{x=1}^{N} [I(x,y) - I'(x,y)]^2 \quad \text{.....(2-1)}$$

MSE = Nilai *Mean Squared Error* I (x,y) = Nilai *pixel* dari citra c*over*

M = Panjang Citra *Stego* I'(x,y) = Nilai *pixel* dari citra *stego*

N = Lebar Citra *Stego* x,y = koordinat *pixel* citra

Setelah diperoleh nilai MSE maka nilai PSNR dapat dihitung dari kuadrat nilai maksimum dibagi dengan MSE. Secara matematis, nilai PSNR dirumuskan sebagai berikut seperti persamaan 2.2:

$$PSNR = 10.\log(\frac{MAXi^2}{MSE}) \quad \text{.....} \quad (2\text{-}2)$$

MSE = nilai *MSE*, *MAXi* = nilai maksimum dari *pixel* citra, yaitu berisi konstanta bernilai 255.

Semakin rendah nilai MSE maka akan semakin baik, dan semakin besar nilai PSNR maka semakin baik kualitas citra steganografi.

2.10 Aplikasi *Mobile*

Aplikasi *mobile* (perangkat bergerak) adalah suatu aplikasi yang dibuat secara khusus untuk berjalan pada *mobile device. Mobile device* ini pada umumnya dikelompokkan berdasarkan *platform*, beberapa kategori *platform* ini adalah:

- Blackberry
- Symbian
- Windows Mobile

- Android
- iPhone

Pada umumnya *Mobile Application* secara spesifik dikembangkan berdasarkan masing-masing *platform* sesuai kebutuhan [NAT-14].

1. ***Native Application***

Aplikasi yang di buat dan diinstal langsung didalam *device* menggunakan bahasa pemrograman yang digunakan untuk membuat aplikasi tersebut. Sebagai contoh, untuk membuat Android apps menggunakan *Java* dan SDK milik Android, dan iOS menggunakan *Objective-C* dan SDK iOS.

2. ***Mobile Web Application***

Aplikasi *mobile* yang dijalankan menggunakan *browser* yang ada di handphone, menggunakan bahasa pemrograman *web* seperti PHP dan HTML5.

3. ***Hybrid Application***

Aplikasi *mobile* yang dibuat dan diinstal langsung di dalam *device*, menggunakan bahasa pemrograman *website* yang digabung dengan bahasa pemrograman yang digunakan untuk membuat aplikasi pada *device* yang dituju. Aplikasi dibuat dengan bahasa pemrograman *web*, sedangkan untuk beberapa fitur yang tidak bisa dijalankan menggunakan bahasa pemrograman *web* akan ditulis dalam bahasa pemrograman yang sesuai dengan *device* tujuan.

2.11 Java

Java adalah Bahasa pemrograman tingkat tinggi yang dikembangkan oleh perusahaan Sun Microsystem. Definisi Java menurut Sun Microsystem adalah nama dari sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer [ASR-11].

Java berdiri diatas sebuah mesin *interpreter* yang diberi nama Jav*a Virtual Machine* (JVM). JVM berfungsi untuk membaca *Bytecode* dalam *file .class* dari suatu program sebagai representasi langsung program yang berisi bahasa mesin. Bahasa Java disebut sebagai bahasa pemrograman yang *portable* karena dapat dijalankan pada berbagai sistem operasi, jika terdapat JVM dalam sistem operasi tersebut. Java merupakan bahasa pemrograman objek murni karena semua kode programnya dibungkus dalam *class*.

2.12 Android

OS Android adalah sistem operasi yang berbasis Linux untuk *smartphone* dan komputer tablet yang awalnya dikembangkan oleh Android Inc [SUB-13]. Google *Inc.* membeli Android *Inc.* yang merupakan pendatang baru yang membuat peranti lunak untuk ponsel. Pada saat perilisan perdana Android, 5 November 2007, Android bersama *open handset alliance* menyatakan mendukung pengembangan *open source* pada perangkat *mobile*. Di lain pihak, Google merilis kode-kode Android di bawah lisensi Apache. Android dibuat berdasarkan kernel Linux yang

dimodifikasi. Aplikasi Android ditulis dengan bahasa Java dan menggunakan Java *Core Libraries*. Aplikasi Android dijalankan di atas VM bernama *Dalvik Virtual Machine*. Gambar 2.12 menunjukkan logo dari Android.

Gambar 2.12 Logo Android

Sumber: [DEV-15]

2.12.1 Versi Android

OS Android adalah salah satu sistem operasi yang berkembang pesat sejak beberapa tahun terakhir. Hal tersebut tercermin dari perkembangan versi *android*, yang akan dijelaskan pada Tabel 2.2.

Tabel 2.2 Versi-versi Sistem Operasi Android

Android *Version*	**API Level**	***Codename***
Android 1.0	1	
Android 1.1	2	
Android 1.5	3	Cupcake
Android 1.6	4	Donut
Android 2.0	5	Éclair
Android 2.0.1	6	Éclair
Android 2.1	7	Éclair
Android 2.2	8	Froyo *(Frozen Yogurt)*
Android 2.3	9	Gingerbread
Android 2.3.3	10	Gingerbread
Android 3.0	11	Honeycomb
Android 3.1	12	Honeycomb
Android 3.2	13	Honeycomb
Android 4.0	14	Ice Cream Sandwich
Android 4.0.3	15	Ice Cream Sandwich

Android 4.1	16	Jelly Bean
Android 4.2	17	Jelly Bean
Android 4.3	18	Jelly Bean
Android 4.4	19	Kitkat
Android 4.4	20	Kitkat
Android 5.0	21	Lollipop
Android 5.1	22	Lollipop

Sumber: [NAK-14]

2.12.2 Android SDK

Android *Software Development Kit* (SDK) adalah *tools* bagi para *programmer* yang ingin mengembangkan aplikasi berbasis Google Android. Android SDK mencakup seperangkat alat pengembangan yang komprehensif. Android SDK terdiri dari *debugger*, *libraries*, *handset emulator*,dokumentasi, contoh kode, dan *tutorial* [FEL-12].

2.12.3 Android Studio

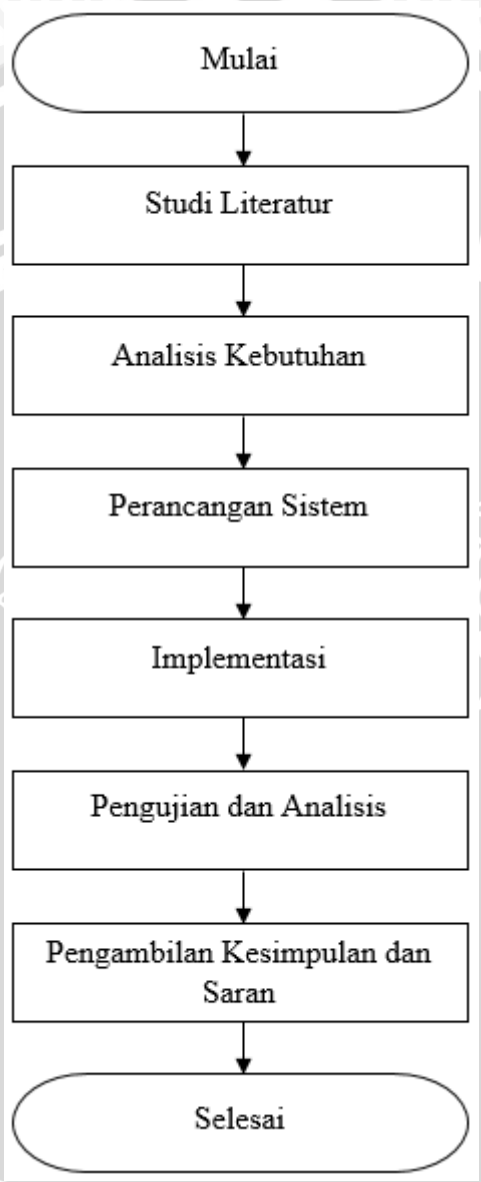
Android Studio adalah sebuah *Integrated Development Environtment* (IDE) untuk pengembangan aplikasi di p*latform* Android sama seperti kombinasi antara Eclipse dan Android *Developer Tools* (ADT) [DEV-15]. Android Studio menggunakan IDE yang berasal dari Intellij IDEA Community Edition. Android Studio menggunakan Gradle untuk manajemen proyeknya. Gradle adalah *build automation tool* yang dapat dikonfigurasi melalui *Digital Subscriber Line* (DSL) berbasis Groovy. Penggunaan DSL berbasis Groovy menyebabkan Gradle lebih fleksibel dan dapat diprogram dengan mudah. Berikut adalah contoh *launcher logo* dari Android Studio seperti ditunjukkan pada Gambar 2.13.

Gambar 2.13 Android Studio

Sumber: [DEV-15]

BAB 3 METODOLOGI

Bab ini membahas metode yang digunakan dalam penelitian yang terdiri dari studi literatur, perancangan sistem, implementasi, pengujian dan analisis serta pengambilan kesimpulan dan saran. Berikut adalah diagram alir dari metodologi penelitian yang dilakukan seperti terlihat pada Gambar 3.1.



Gambar 3.1 Diagram Alir Penelitian

3.1 Studi Literatur

Metode ini digunakan untuk mendapatkan dasar teori sebagai sumber acuan untuk penulisan skripsi dan pengembangan aplikasi. Teori dan pustaka yang berkaitan dengan tugas akhir ini meliputi:

1. Steganografi
    - *Least Significant Bit*
2. *Watermark*
3. MSE *(Mean Squared Error)*
4. PSNR *(Peak Signal to Noise Ratio)*
5. Android Studio
6. Sistem Operasi pada Android
7. Bahasa Pemrograman Java

Selain melakukan studi mengenai penelitian, penulis juga melakukan pengkajian terhadap penelitian sebelumnya yang berkaitan dengan *watermark* dan metode *Least Significant Bit*. Adapun penelitian sebelumnya yang dikaji adalah sebagai berikut:

1. *Analysis of Image Watermarking Using Least Significant Bit Algorithm*.
2. *LSB Based Digital Image Watermarking for Gray Scale Image*.
3. *Analysis of Image Watermarking: LSB Modification and Spread-Spectrum Technique*.
4. Perbandingan Steganografi Metode *Spread Spectrum* dan *Least Significant Bit* (LSB) Antara Waktu Proses dan Ukuran *File* Gambar.

3.2 Analisis Kebutuhan

Kegiatan analisis kebutuhan perangkat lunak meliputi analisis spesifikasi perangkat lunak. Metode analisis menggunakan bahasa pemodelan UML *(Unified Modeling Language). Use Case Diagram* digunakan untuk mendeskripsikan kebutuhan-kebutuhan dan fungsionalitas sistem dari perspektif *user*. Analisis kebutuhan dilakukan dengan mengidentifikasi semua kebutuhan (*requirements*) sistem yang kemudian akan dimodelkan dalam diagram *use case.* Kebutuhan fungsional yang nantinya akan disediakan oleh aplikasi ini antara lain adalah:

1. Sistem harus menyediakan operasi *encode* dan *decode* pada menu utama.
2. Sistem dapat mengimplementasikan *watermark*.
3. Aplikasi dapat melakukan fungsi penyisipan dan ekstraksi *file* citra dengan metode *Least Significant Bit*.

3.3 Perancangan Sistem

Perancangan arsitektur sistem yang digunakan dalam penelitian ini membahas tentang analisis kebutuhan sistem, perancangan sistem, implementasi, pengujian, serta analisis dari aplikasi *watermark* pada *file* citra. Metode yang digunakan adalah *Least Significant Bit*. Perancangan dilakukan dengan enam tahap yaitu perancangan umum sistem, analisis kebutuhan, perancangan modul *encoder*,

perancangan modul *decoder*, perancangan perangkat lunak dan perancangan PSNR.

3.4 Implementasi

Implementasi dilakukan dengan mengacu kepada perancangan aplikasi. Aplikasi dibangun menggunakan *software* Android Studio. Pembuatan aplikasi dikembangkan secara spesifik pada sistem operasi Android yang di buat dan dipasang langsung dalam perangkat *mobile.*

3.5 Pengujian dan Analisis

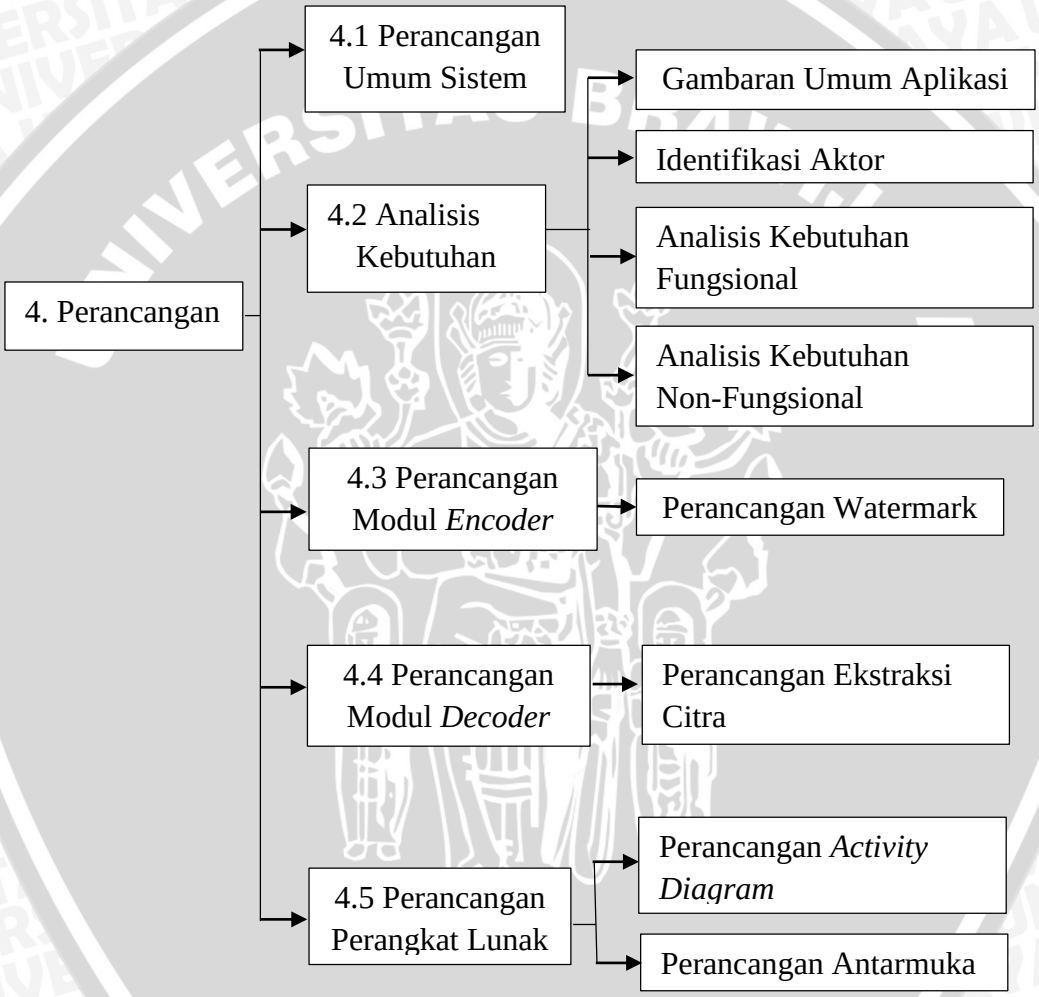
Pengujian dari perangkat bergerak berkaitan dengan pengujian sistem. Proses pengujian dilakukan melalui tahapan pengujian validitas, pengujian kompatibilitas, pengujian performansi, dan pengujian MSE dan PSNR. Pada pengujian tersebut akan digunakan teknik pengujian *black box*. Analisis sistem dilakukan dengan mengevaluasi hasil dari tiap-tiap proses pengujian.

3.6 Pengambilan Kesimpulan

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan, implementasi dan pengujian sistem aplikasi perangkat bergerak telah selesai dilakukan dan didasarkan pada kesesuaian antara teori dan praktek. Kesimpulan diambil untuk menjawab rumusan masalah yang telah ditetapkan sebelumnya. Tahap terakhir dari penulisan adalah saran yang dimaksudkan untuk memperbaiki kesalahan-kesalahan yang terjadi dan menyempurnakan penulisan serta untuk memberikan pertimbangan atas pengembangan aplikasi perangkat bergerak selanjutnya.

BAB 4 PERANCANGAN

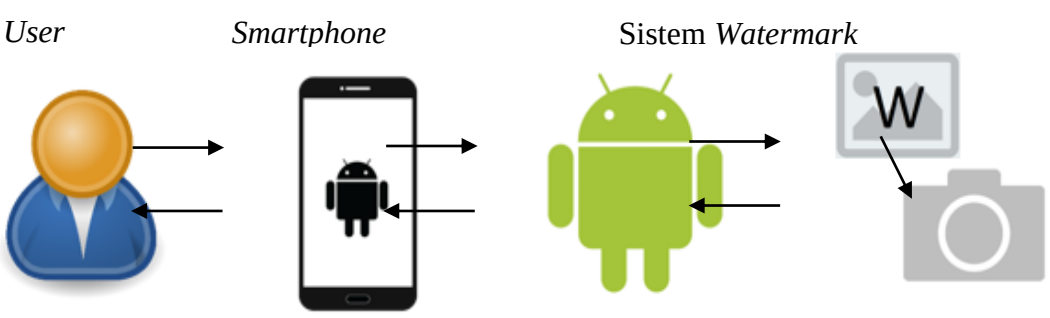
Bab perancangan membahas tentang analisis kebutuhan sistem, perancangan sistem, implementasi, pengujian, serta analisis dari aplikasi *Image watermarking* pada *platform Android*. Perancangan aplikasi menggunakan *watermark* dengan metode *Least Significant Bit*. Perancangan dilakukan melalui lima tahap, yaitu perancangan umum sistem, analisis kebutuhan, perancangan modul *encoder*, perancangan modul *decoder* dan perancangan perangkat lunak. Pohon perancangan ditunjukkan pada Gambar 4.1.



Gambar 4.1 Pohon Perancangan

4.1 Perancangan Umum Sistem

Perancangan umum sistem merupakan tahap awal dari perancangan perangkat lunak. Perancangan sistem dilakukan dengan merepresentasikan arsitektur sistem secara umum. Gambar 4.2 berikut menunjukkan perancangan umum dari sistem.



Gambar 4.2 Perancangan Umum Sistem

Pada Gambar 4.2 *user* menjalankan aplikasi *watermark* pada *smartphone*. *User* lalu memilih *file* citra yang ingin diberi *watermark* beserta dengan *file* citra sebagai *digital signature*, sistem akan melakukan proses *watermark*. Hasil dari *watermark* akan dikembalikan lagi pada *user*.

4.2 Analisis Kebutuhan Aplikasi

Tujuan dari proses analisis kebutuhan aplikasi adalah untuk mengetahui sifat dari kebutuhan sistem sehingga mempermudah dalam perancangan. Proses analisis kebutuhan aplikasi diawali dengan penjabaran gambaran umum aplikasi, identifikasi aktor, penjabaran tentang kebutuhan fungsional yang akan dimodelkan dalam bentuk *use case diagram* serta kebutuhan non fungsional. Analisis kebutuhan ini bertujuan untuk menggambarkan kebutuhan-kebutuhan yang harus disediakan oleh sistem agar dapat membantu kebutuhan pengguna. Analisis masalah juga akan dilakukan untuk mengetahui permasalahan yang mungkin terjadi pada saat pembangunan aplikasi *watermark*.

4.2.1 Analisis Masalah

Analisis masalah dilakukan untuk mengetahui masalah-masalah apa saja yang terjadi dalam pembangunan aplikasi *watermark*. Masalah yang terjadi ketika pembangunan *watermark* pada *file* citra yaitu kapasitas citra yang dapat disisipkan informasi kecil, kualitas *file* citra setelah disisipkan informasi, dan keamanan informasi yang disisipkan kedalam citra tidak terjamin. Jika letak dari informasi yang disisipkan pada *file* citra diketahui, maka informasi akan langsung dapat diketahui.

4.2.2 Gambaran Umum Aplikasi

Pembahasan gambaran umum aplikasi *Image Watermarking* pada *platform Android* terdiri atas dua bagian, yaitu deskripsi umum aplikasi dan lingkungan aplikasi.

a. Deskripsi Aplikasi

Aplikasi *Image Watermarking* pada *platform* Android adalah aplikasi yang digunakan untuk melindungi hak milik intelektual atas produk multimedia dengan

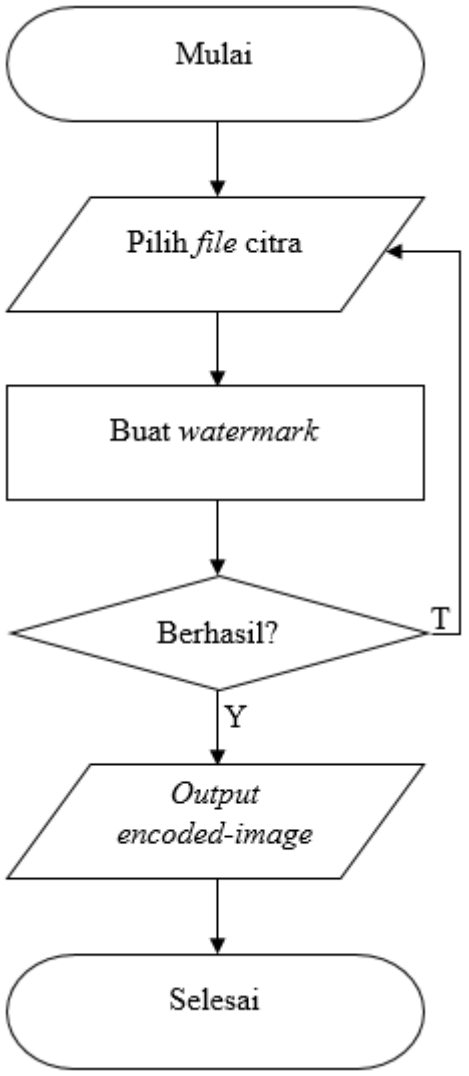
cara memberikan tanda tangan digital pada *file* citra. tanda tangan atau stempel digital ini berupa *file* citra.

Pada aplikasi ini *file* citra yang akan digunakan sebagai stempel digital akan disisipkan pada *file* citra yang ingin dilindungi. Metode *watermark* yang digunakan adalah *Least Significant Bit,* yaitu dengan cara menyisipkan *file* citra ke dalam *bit* yang kurang berpengaruh ke dalam sebuah gambar.

Pada modul *encoder*, aplikasi akan melakukan proses *Least Significant Bit* untuk memasukkan *file* citra. Sedangkan pada modul *decoder*, aplikasi akan melakukan proses ekstraksi stempel digital.

Tahapan-tahapan yang dilakukan pada setiap prosesnya ditunjukkan dengan diagram alir sebagai berikut:

1. Proses Penyisipan



Gambar 4.3 Diagram Alir Proses Penyisipan citra

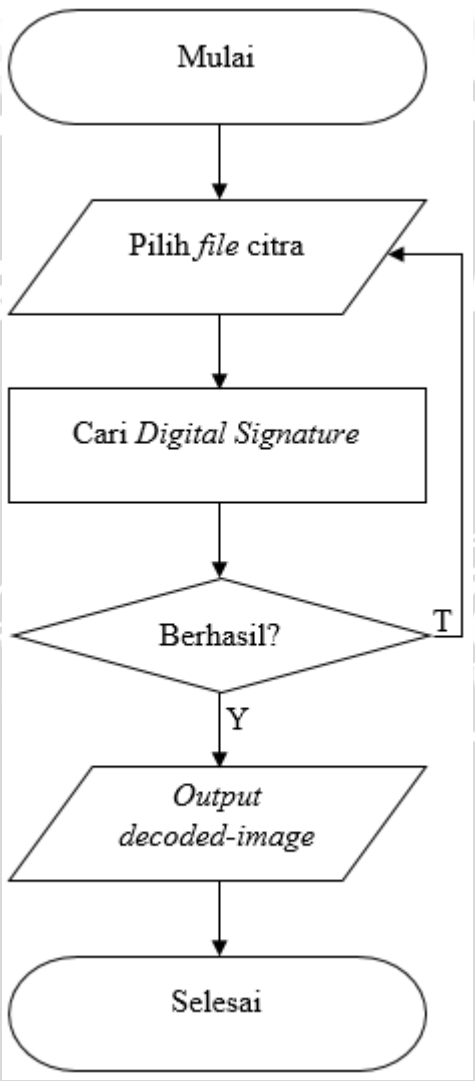
Pada Gambar 4.3 menunjukkan diagram alir dari proses penyisipan stempel digital dengan langkah-langkah dijelaskan sebagai berikut:

a. *User* memilih *file* citra yang akan dijadikan sebagai media penyisipan.

b. User melakukan pembuatan *watermark* menggunakan metode *Least Significant Bit*.

c. Hasil penyisipan disimpan sebagai *encoded-image*.

2. Proses Ekstraksi



Gambar 4.4 Diagram Alir Proses Ekstraksi Citra

Pada Gambar 4.4 menunjukkan diagram alir dari proses penyisipan stempel digital dengan langkah-langkah dijelaskan sebagai berikut:

a. *User* memilih *encoded-image*.

b. *User* melakukan pencarian stempel digital.

c. Hasil ekstraksi disimpan sebagai *decoded-image*.

b. Lingkungan Aplikasi

Aplikasi *Image Watermarking* pada *platform* Android membutuhkan suatu lingkungan (*environment*) yang digunakan sebagai tempat berjalannya aplikasi. Secara keseluruhan aplikasi perangkat bergerak ini berbasis *native mobile development*, sehingga membutuhan sebuah *device* untuk menjalankan aplikasi tersebut. Dengan penggunaan teknologi *mobile smartphone* Android, aplikasi perangkat bergerak ini dapat dijalankan dengan baik di semua *device* Android dengan versi minimal 4.1 OS Jelly Bean.

4.2.3 Identifikasi Aktor

Tahap Identifikasi aktor bertujuan untuk melakukan identifikasi terhadap aktor yang berinteraksi langsung dengan sistem. Aplikasi *Image Watermarking* pada *platform* Android memiliki satu aktor, yaitu *User*. *User* berperan dalam proses *watermark*. *User* juga dapat berperan dalam proses ekstraksi *encoded-image*. Pada Tabel 4.1 dijelaskan deskripsi aktor yang terlibat pada sistem.

Tabel 4.1 Deskripsi Aktor pada Aplikasi

No	Aktor	Deskripsi
1.	*User*	*User* dapat melakukan operasi *Encode* yaitu penyisipan stempel digital untuk membuat *encoded-image*. Selain itu *user* juga dapat melakukan operasi *Decode* berupa ekstraksi *decoded-image*.

4.2.4 Analisis Kebutuhan Fungsional

Daftar kebutuhan sistem terdiri dari sebuah kolom yang menguraikan kebutuhan yang harus disediakan oleh sistem. Pada kolom lain akan menunjukkan nama *use case* yang menunjukkan fungsionalitas masing-masing kebutuhan tersebut. Daftar kebutuhan sistem tersebut dapat dilihat pada Tabel 4.2 menggunakan SRS (*Software Requirement Specification)*.

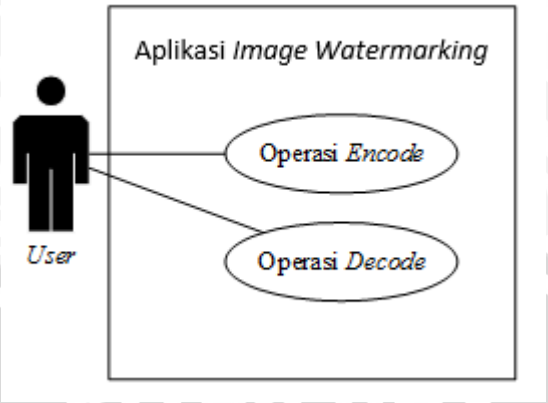
Tabel 4.2 Spesifikasi Kebutuhan Fungsional

No. SRS	Kebutuhan	Aktor	Nama *Use Case*
SRS_001	Aplikasi harus menyediakan fasilitas penyisipan stempel digital pada *file* citra.	*User*	Operasi *Encode*
SRS_002	Aplikasi harus menyediakan fasilitas ekstraksi *decoded-image*.	*User*	Operasi *Decode*

4.2.4.1 *Use Case* Diagram

Use Case Diagram merupakan salah satu model UML yang digunakan untuk mendeskripsikan kebutuhan fungsional sebuah sistem dari perspektif *end user.*

Use case juga menunjukkan aktifitas-aktifitas yang dilakukan oleh *user.* Berikut adalah *use case diagram* dari aplikasi seperti terlihat pada Gambar 4.5.



Gambar 4.5 *Use Case* Diagram

4.2.4.2 Skenario *Use Case*

Secara lebih detail masing-masing *use case* yang terdapat pada *use case* diagram aplikasi dijabarkan dalam skenario *use case*. Di dalam skenario *use case*, akan diberikan uraian nama *use case*, aktor yang berhubungan dengan *use case* tersebut, tujuan dari *use case,* deskripsi umum tentang *use case,* kondisi awal yang harus dipenuhi dan kondisi akhir yang diharapkan setelah berjalannya kebutuhan fungsional dari *use case* tersebut. Selain itu juga akan diberikan ulasan yang berkaitan dengan tanggapan dari sistem atas suatu aksi yang diberikan oleh aktor.

a. Skenario *Use Case* Operasi *Encode*

Kebutuhan fungsional yang harus disediakan oleh sistem adalah kebutuhan untuk melakukan operasi *encode* yang berisi proses penyisipan stempel digital pada *file* citra. Kebutuhan tersebut direpresentasikan oleh *use case* Operasi *Encode* seperti yang terlihat pada Tabel 4.3.

Tabel 4.3 Skenario *Use Case* Operasi *Encode*

Nama	Operasi *Encode*
Kode SRS	SRS_001
Tujuan	Menyisipkan *file* citra stempel digital pada *file* citra yang ingin dilindungi.
Deskripsi	*Use Case* ini memungkinkan *User* melakukan operasi *encode* yaitu proses menyisipkan stempel digital pada *file* citra.
Aktor	*User*
Kondisi Awal	*User* harus membuka dan masuk aplikasi terlebih dahulu sebelum *use case* dimulai. *User* harus masuk pada menu Operasi *Encode*.

Flow of Events	
Alur Utama	
1. Sistem meminta *user* untuk melakukan pemilihan *file* citra yang digunakan sebagai stempel digital. 2. Sistem meminta *user* untuk melakukan pemilihan *file* citra yang ingin dilindungi. 3. Sistem meminta *user* untuk membuat *Watermark*. 4. Sistem akan melakukan proses *watermark* pada stempel digital. 5. Hasil dari proses *watermark* adalah *encoded-image*.	
Alur Bagian	
1. Sistem melakukan proses *watermark* dengan metode *Least Significant Bit*. 2. Stempel digital akan disisipkan pada *file* citra yang ingin dilindungi melalui proses *watermark*.	
Kondisi Akhir	Proses *watermark* menghasilkan *encoded image,* kemudian disimpan dalam *internal memory* pada *smartphone.*

b. Skenario *Use Case* Operasi *Decode*

Kebutuhan fungsional lain yang harus disediakan oleh sistem adalah kebutuhan untuk melakukan operasi *decode* yang berisi proses ekstraksi pada *encoded-image*. Kebutuhan tersebut direpresentasikan oleh *use case* Operasi *Decode* seperti yang terlihat pada Tabel 4.4.

Tabel 4.4 Skenario *Use Case* Operasi *Decode*

Nama	Operasi *Decode*
Kode SRS	SRS_002
Tujuan	Melakukan proses ekstraksi pada *encoded-image*.
Deskripsi	*Use Case* ini memungkinkan *User* melakukan operasi *decode* yaitu proses ekstraksi pada *encoded image*.
Aktor	*User* (*Receiver*)
Kondisi Awal	*User* harus membuka dan masuk aplikasi terlebih dahulu sebelum *use case* dimulai. *User* harus masuk pada menu Operasi *Decode*.
Flow of Events	

Alur Utama	
1. Sistem meminta *user* untuk melakukan pemilihan *encoded-image*. 2. Sistem meminta *user* untuk melakukan Operasi *Decode*. 3. Setelah user melakukan pemilihan operasi maka proses *decode* dijalankan.	
Alur Bagian	
1. Sistem melakukan proses ekstraksi pada *encoded-image*. 2. Hasil dari proses ekstraksi adalah *decoded-image*.	
Kondisi Akhir	Jika *use case* berhasil, didapatkan stempel digital yang sama dengan sebelum melalui proses *watermark*. Jika *use case* gagal, maka stempel digital gagal di ekstraksi.

4.2.5 Analisis Kebutuhan Non-Fungsional

Analisis kebutuhan non-fungsional adalah analisis untuk mengetahui spesifikasi yang dibutuhkan oleh sistem. Ada beberapa parameter dan deskripsi kebutuhan yang akan digunakan dalam pengembangan, yaitu *Compatibility* dan *Robustness*.

Tabel 4.5 Spesifikasi Kebutuhan Non-Fungsional

Parameter	Deskripsi Kebutuhan
Compatibility	Aplikasi harus dapat dijalankan di berbagai versi sistem operasi Android (Jelly Bean, KitKat, dan Lollipop).
Robustness	Aplikasi dirancang agar bekerja dengan cepat dan efisien pada perangkat bergerak dengan spesifikasi menengah kebawah yang berakibat pada daya tahan citra.

4.2.5.1 Skenario Pengujian

Skenario pengujian dibuat berdasarkan analisis kebutuhan fungional dan non-fungsional. Skenario proses pengujian akan dilakukan seperti yang terlihat pada Tabel 4.6.

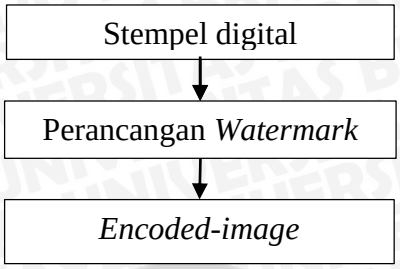
Tabel 4.6 Skenario Pengujian

Nama	Pengujian
Kode SRS	SRS_003
Tujuan	Melakukan proses pengujian untuk melihat apakah aplikasi berfungsi sesuai dengan kebutuhan.
Deskripsi	Pengujian dilakukan untuk untuk membuktikan sistem yang dibangun sesuai dengan yang

	dibutuhkan.
Aktor	*Tester*
Kondisi Awal	Perancangan aplikasi telah selesai dan dapat dijalankan.
Flow of Events	
Alur Utama	
1. Penguji melakukan pengujian validitas. 2. Penguji melakukan pengujian kompatibilitas. 3. Penguji melakukan pengujian performansi. 4. Penguji melakukan pengujian MSE dan PSNR. 5. Penguji melakukan pengujian Ukuran *file.*	
Alur Bagian	
1. Penguji mengecek apakah sistem yang dibangun sudah sesuai dengan kebutuhan. 2. Penguji mengecek apakah sistem yang dibangun dapat digunakan pada sistem operasi yang berbeda. 3. Penguji mengecek apakah sistem yang dibangun dapat bekerja secara efisien dan optimal. 4. Penguji mengecek kualitas yang dihasilkan oleh aplikasi melalui pengukuran PSNR dan MSE. 5. Penguji mengecek apakah sistem yang dibangun dapat menggunakan stempel digital dengan ukuran yang berbeda	
Kondisi Akhir	Jika proses pengujian berhasil, maka aplikasi telah bekerja sesuai dengan kebutuhan yang diharapkan.

4.3 Perancangan Modul *Encoder*

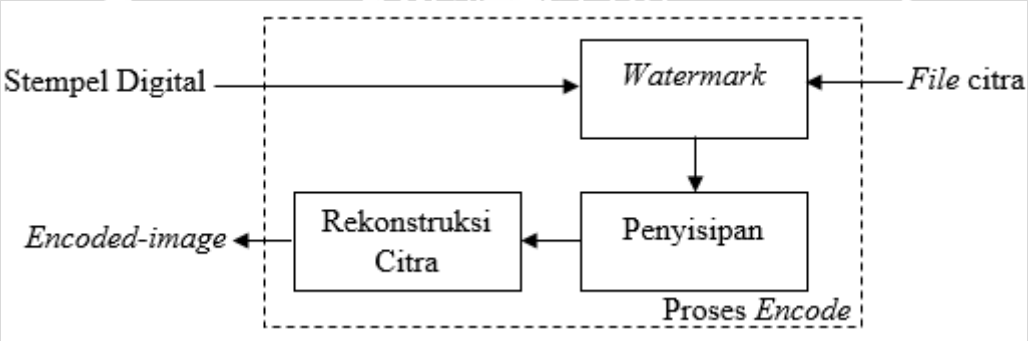
Modul *Encoder* adalah proses penyisipan stempel digital pada *file* citra untuk melindungi hak cipta intelektual *file* citra. Perancangan modul *encoder* berisi perancangan *watermark*. Proses *Watermark* menggunakan metode *Least Significant Bit*. Pada Gambar 4.6 akan menjelaskan perancangan modul *encoder*.



Gambar 4.6 Perancangan Modul *Encoder*

4.3.1 Perancangan *Watermark*

Pada *watermark* dilakukan proses penyisipan yaitu menyembunyikan stempel digital kedalam media penampung, dalam hal ini media penampung berupa *file* citra. Proses ini akan menghasilkan citra yang telah disisipkan stempel digital (*encoded-image*) yang menyerupai dengan citra sebelum disisipkan stempel digital. Proses watermark digunakan untuk menyembunyikan stempel digital di dalam *file* citra. Proses perancangan aplikasi *Image Watermarking* ditunjukkan pada Gambar 4.7.



Gambar 4.7 Perancangan Aplikasi *Image Watermarking*

Penjelasan mengenai langkah-langkah dalam pembuatan *encoded-image* dijelaskan secara urut sebagai berikut:

1. Pilih *file* citra sebagai stempel digital dari *Gallery*.
2. Pilih *file* citra yang ingin dilindungi dari *Gallery*.
3. Ambil komponen warna dari tiap *pixel file* citra, dan ubah menjadi tiga *channel* RGB (*Red*, *Green*, *Blue*).
4. Lakukan *watermark*.

   Proses *watermark* adalah menyisipkan stempel digital ke dalam *channel* RGB.
5. Penyisipan.

   Pada proses penyisipan, agar tidak mempengaruhi nilai *pixel* dari citra secara drastis, *bit* ditempatkan pada *bit* paling rendah tiap-tiap *pixel* dari *channel* RGB. Proses penyisipan dimulai secara urut dari *bit Red* → *Green* → *Blue*.
6. Pembentukan *file* citra baru.

Proses akhir dari tahapan *watermark* yaitu proses rekonstruksi dengan membuat *file* citra baru (*encoded-image*) dari *bit* modifikasi yang dihasilkan saat proses penyisipan. *Encoded-image* memiliki ekstensi (.*png*) dan nama *file* diberi awalan "*encoded*".

7. Proses *watermark* selesai.

4.3.1.1 Skenario Proses *Watermark*

Pada skenario proses *watermark*, sebagai contoh seorang *user* ingin membuat *encoded-image,* langkah yang harus dilakukan tentu saja memilih *file* citra sebagai obyek seperti yang ditunjukkan pada Gambar 4.8:

Gambar 4.8 Tampilan dari obyek *file* citra

File Citra yang dipilih : **8x8.png**

Resolusi : 8x8 *pixels*

Ukuran *file* : 824 *Bytes*

Bit depth : 24 *bit*

Langkah selanjutnya adalah memilih *file* citra yang akan digunakan sebagai *digital signature* seperti yang ditunjukkan pada Gambar 4.9:

Gambar 4.9 Tampilan dari *file* citra *digital signature*

File Citra yang dipilih : **4x4.png**

Resolusi : 4x4 *pixels*

Ukuran *file* : 785 *Bytes*

Bit depth : 24 *bit*

Setelah *input* lengkap, maka sistem akan memulai tahapan *watermark*. Berikut akan dijelaskan langkah-langkah dalam pembuatan *encoded-image*:

1. Mengambil nilai dari tiap *pixel file* citra *digital signature*. *File* **4x4.png** memiliki resolusi 4x4 *pixel* seperti yang ditunjukkan pada Gambar 4.10.

p1x1	p2x1	p3x1	p4x1
p1x2	p2x2	p3x2	p4x2
p1x3	p2x3	p3x3	p4x3
p1x4	p2x4	p3x4	p4x4

Gambar 4.10 Ilustrasi *pixel* dari 4x4.png

Dari Gambar 4.10 di atas, sebagai contoh p1x1 menunjukkan koordinat *pixel* pada kolom pertama dan baris pertama. Karena memiliki kedalaman 24 *bit*, tiap *pixel* terdiri dari komponen warna *Red, Green*, dan *Blue* yang dipetakan sebagai berikut:

Red

50	56	81	66
107	150	170	98
68	133	134	43
42	76	48	17

Green

32	38	67	52
89	132	156	84
90	155	161	70
64	98	75	44

Blue

30	36	64	49
87	130	153	81
101	166	170	79
75	109	84	53

Tiap-tiap nilai dari *pixel file* citra kemudian diubah menjadi bilangan biner 8 *bit*. Cara untuk merubah dari bilangan desimal menjadi biner 8 bit adalah dengan menggunakan perhitungan pangkat 2 di bawah nilai yang ingin diubah kemudian dilakukan pengurangan nilai hingga nilai tersebut menjadi 0. Sebagai contoh pixel *Red* pada koordinat 1x1 bernilai sebesar 50, maka dilakukan perhitungan pangkat 2 seperti yang ditunjukkan pada Gambar 4.11.

2^6	2^5	2^4	2^3	2^2	2^1	2^0
64	32	16	8	4	2	1

Gambar 4.11 Perhitungan pangkat 2

Dari perhitungan pangkat 2 diketahui bahwa 32 berada di bawah nilai 50 dan dapat dilakukan pengurangan nilai. Pengurangan nilai akan ditunjukkan pada Gambar 4.12.

50 – 32 = 18

Gambar 4.12 Proses pengurangan

Setelah berhasil dilakukan pengurangan, tulis angka 1 dibawah kotak yang tersedia dan ulangi kembali langkah di atas untuk menemukan nilai pangkat 2 yang berada di bawah nilai 18. Setiap pengurangan nilai yang berhasil akan ditulis dengan angka 1 dan setiap pengurangan yang gagal ditulis dengan angka 0 seperti yang ditunjukkan pada Gambar 4.13

50 – 32 = 18	18 – 16 = 2	2 – 8 = X	2 – 4 = X	2 – 2 = 0	0 – 1 = X
1	1	0	0	1	0

Gambar 4.13 Proses pengurangan keseluruhan

Dari Gambar 4.13, nilai pixel sebesar 50 telah diubah menjadi nilai biner 8 bit yaitu 00110010.

Bit Red = 00110010 00111000 01010001 01000010 01101011 10010110 10101010 01100010 01000100 10000101 10000110 00101011 00101010 01001100 00110000 00010001.

Bit Green = 00100000 00100110 01000011 00110100 01011001 10000100 10011100 01010100 01011010 10011011 10100001 01000110 01000000 01100010 01001011 00101100.

Bit Blue = 00011110 00100100 01000000 00110001 01010111 10000010 10011001 01010001 01100101 10100110 10101010 01001111 01001011 01101101 01010100 00110101.

Bit-bit warna ini yang akan disisipkan pada **8x8.png**

2. Mengambil nilai-nilai dari tiap *pixel file* citra obyek. *File* **8x8.png** memiliki resolusi 8x8 *pixel*, seperti yang ditunjukkan pada Gambar 4.14.

p1x1	p2x1	p3x1	p4x1	p5x1	p6x1	p7x1	p8x1
p1x2	p2x2	p3x2	p4x2	p5x2	p6x2	p7x2	p8x2
p1x3	p2x3	p3x3	p4x3	p5x3	p6x3	p7x3	p8x3
p1x4	p2x4	p3x4	p4x4	p5x4	p6x4	p7x4	p8x4
p1x5	p2x5	p3x5	p4x5	p5x5	p6x5	p7x5	p8x5
p1x6	p2x6	p3x6	p4x6	p5x6	p6x6	p7x6	p8x6

p1x7	p2x7	p3x7	p4x7	p5x7	p6x7	p7x7	p8x7
p1x8	p2x8	p3x8	p4x8	p5x8	p6x8	p7x8	p8x8

Gambar 4.14 Ilustrasi *pixel* dari 8x8.png

Dari Gambar 4.14 di atas, sebagai contoh p1x1 menunjukkan koordinat *pixel* pada kolom pertama dan baris pertama. Karena memiliki kedalaman 24 *bit,* tiap pixel terdiri dari komponen warna *Red*, *Green*, dan *Blue* yang dipetakan sebagai berikut:

Red

174	101	3	41	154	222	191	180
128	80	91	108	108	133	192	196
184	118	22	94	91	78	253	201
165	93	3	37	120	180	159	160
167	139	25	84	68	105	177	182
185	106	35	75	123	147	205	166
172	103	14	34	94	113	136	170
154	94	41	63	96	126	174	165

Green

56	47	76	158	224	220	86	24
5	20	152	216	188	142	91	39
53	45	61	192	188	111	157	42
22	12	41	137	226	222	69	5
12	55	79	202	171	141	93	37
32	22	87	189	222	181	126	29
41	27	43	121	190	155	61	36
34	20	60	138	189	172	100	32

Blue

194	158	131	131	92	85	57	46
145	133	215	200	63	15	63	60
193	163	138	193	81	0	133	62
146	117	112	139	126	112	45	23
104	132	126	188	66	31	67	54

122	99	137	179	114	68	95	37
153	127	123	131	83	31	19	33
157	133	155	159	83	40	53	25

3. Proses penyisipan dilakukan dengan meletakkan *bit-bit* warna ke dalam *file citra* obyek. Proses penyisipan *bit* warna hanya melibatkan *channel* RGB. Sebagai contoh, diambil nilai dari *pixel* pada koordinat 1x1 sampai dengan 4x1.

   Ambil nilai *pixel* 1x1, 2x1, 3x1, dan 4x1 kemudian ubah ke dalam biner 8 *bit:*

   p1x1 p2x1 p3x1 p4x1

   174 = 10101110 101 = 01100101 3 = 00000011 41 = 00101001

   56 = 00111000 47 = 00101111 76 = 01001100 158 = 10011110

   194 = 11000010 158 = 10011110 131 = 10000011 131 = 10000011

   Pada ketiga *pixel* di atas akan disisipkan 8 *bit* pertama dari *bit* warna *file* citra *digital signature*.

   Penyisipan dilakukan pada 2 *bit* terendah *pixel* dimulai dari urutan R→G→ B

   Bit Red = 00110010

   p1x1 p2x1

   174 = 10101110 >> 00 101 = 01100101 >> 11

   p3x1 p4x1

   3 = 00000011 >> 00 41 = 00101001 >> 10

   Bit Green = 00100000

   p1x1 p2x1

   56 = 00111000 >> 00 47 = 00101111 >> 10

   p3x1 p4x1

   76 = 01001100 >> 00 158 = 10011110 >> 00

   Bit Blue = 00011110

   p1x1 p2x1

   194 = 11000010 >> 00 158 = 10011110 >> 01

   p3x1 p4x1

   131 = 10000011 >> 11 131 = 10000011 >> 10

   Setelah dilakukan penyisipan akan menghasilkan:

   p1x1 p2x1 p3x1 p4x1

   172 = 10101100 103 = 01100111 0 = 00000000 42 = 00101010

   56 = 00111000 46 = 00101110 76 = 01001100 156 = 10011100

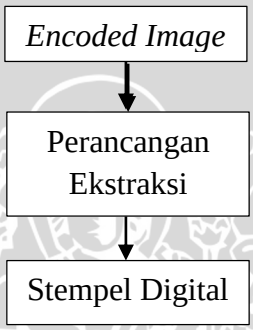
192 = 11000000 157 = 10011101 131 = 10000011 130 = 10000010

Proses tersebut berulang pada *channel* RGB hingga ditemukan *bit* penanda akhir yaitu 00000011 11100111.

4. Nilai baru dari tiap *pixel* yang diperoleh saat proses penyisipan akan dipakai untuk rekonstruksi *file* citra. *File* citra baru dibentuk dengan ekstensi (*.png*) dan nama *file* diberi akhiran "*_encoded".*

4.4 Perancangan Modul *Decoder*

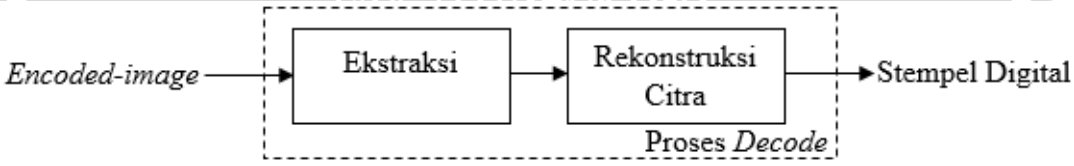
Modul *Decoder* adalah proses ekstraksi stempel digital pada *encoded-image* untuk mengambil informasi rahasia yang terselubung. Perancangan modul *decoder* berisi dengan perancangan ekstraksi. Proses ekstraksi menggunakan metode *Least Significant Bit*. Pada Gambar 4.15 akan menjelaskan perancangan modul *decoder*.



Gambar 4.15 Perancangan Modul *Decoder*

4.4.1 Perancangan Ekstraksi

Proses ekstraksi adalah proses pengambilan *file* citra yang tersembunyi pada *encoded-image*. Proses ini akan menghasilkan stempel digital yang telah disisipkan sebelumnya. Proses ekstraksi dari aplikasi *Image Watermarking* ditunjukkan pada Gambar 4.16.



Gambar 4.16 Proses Ekstraksi

Penjelasan mengenai langkah-langkah pada proses ekstraksi adalah:

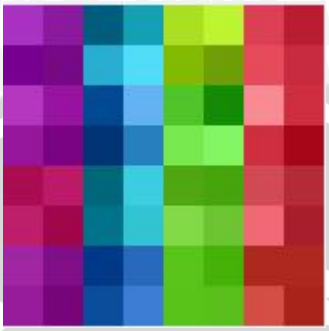
1. Pilih *encoded-image* yang akan diekstraksi.
2. Lakukan ekstraksi.

   Ambil nilai-nilai dari tiap *pixel file* citra. Pada langkah ini dilakukan konversi citra *encoded-image* ke dalam *Byte*. *Bit encoded-image* yang diambil adalah *bit* paling rendah tiap-tiap *pixel* dari *channel* RGB. Proses ekstraksi dimulai secara urut dari *bit Red* → *Green* → *Blue*.

3. Stempel digital diperoleh.

4.4.1.1 Skenario Proses Ekstraksi

Pada skenario proses watermark, sebagai contoh seorang *user* ingin mengekstraksi *encoded-image* seperti yang ditunjukkan pada Gambar 4.17.



Gambar 4.17 Tampilan *encoded-image*

Encoded-image : **8x8_encoded.png**

Resolusi : 8x8 *pixels*

Ukuran *file* : 283 *Bytes*

Bit depth : 24 *bit*

Setelah *input* lengkap, maka sistem akan memulai tahapan ekstraksi. Berikut akan dijelaskan langkah-langkah dalam ekstraksi dari *encoded-image*.

1. Ambil nilai-nilai dari tiap *pixel file* citra. *File* **8x8_encoded.png** memiliki resolusi 8x8 *pixel*, seperti yang ditunjukkan pada Gambar 4.18.

p1x1	p2x1	p3x1	p4x1	p5x1	p6x1	p7x1	p8x1
p1x2	p2x2	p3x2	p4x2	p5x2	p6x2	p7x2	p8x2
p1x3	p2x3	p3x3	p4x3	p5x3	p6x3	p7x3	p8x3
p1x4	p2x4	p3x4	p4x4	p5x4	p6x4	p7x4	p8x4
p1x5	p2x5	p3x5	p4x5	p5x5	p6x5	p7x5	p8x5
p1x6	p2x6	p3x6	p4x6	p5x6	p6x6	p7x6	p8x6
p1x7	p2x7	p3x7	p4x7	p5x7	p6x7	p7x7	p8x7
p1x8	p2x8	p3x8	p4x8	p5x8	p6x8	p7x8	p8x8

Gambar 4.18 Ilustrasi *pixel* dari 8x8_encoded.png

2. Nilai *pixel* dari **8_encoded.png** dipetakan menjadi sebagai berikut:

Red							
172	103	0	42	154	222	191	180
128	80	91	108	108	133	192	196

184	118	22	94	91	78	253	201
165	93	3	37	120	180	159	160
167	139	25	84	68	105	177	182
185	106	35	75	123	147	205	166
172	103	14	34	94	113	136	170
154	94	41	63	96	126	174	165

Green

56	46	76	156	224	220	86	24
5	20	152	216	188	142	91	39
53	45	61	192	188	111	157	42
22	12	41	137	226	222	69	5
12	55	79	202	171	141	93	37
32	22	87	189	222	181	126	29
41	27	43	121	190	155	61	36
34	20	60	138	189	172	100	32

Blue

192	157	131	130	92	85	57	46
145	133	215	200	63	15	63	60
193	163	138	193	81	0	133	62
146	117	112	139	126	112	45	23
104	132	126	188	66	31	67	54
122	99	137	179	114	68	95	37
153	127	123	131	83	31	19	33
157	133	155	159	83	40	53	25

3. Dari pemetaan **8x8_encoded.png** dilakukan konversi ke dalam biner 8 *bit* kemudian dilakukan ekstraksi pada 2 *bit* terakhir *channel* RGB. Proses untuk merubah dari bilangan desimal menjadi biner 8 bit adalah dengan menggunakan perhitungan pangkat 2 di bawah nilai yang ingin diubah kemudian dilakukan pengurangan nilai hingga nilai tersebut menjadi 0. Sebagai contoh pixel *Red* pada koordinat 1x1 bernilai sebesar 172, maka dilakukan perhitungan pangkat 2 seperti yang ditunjukkan pada Gambar 4.19.

2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
256	128	64	32	16	8	4	2	1

Gambar 4.19 Perhitungan pangkat 2

Dari perhitungan pangkat 2 diketahui bahwa 128 berada di bawah nilai 172 dan dapat dilakukan pengurangan nilai. Pengurangan nilai akan ditunjukkan pada Gambar 4.20.

172 – 128 = 44

Gambar 4.20 Proses pengurangan

Setelah berhasil dilakukan pengurangan, tulis angka 1 dibawah kotak yang tersedia dan ulangi kembali langkah di atas untuk menemukan nilai pangkat 2 yang berada di bawah nilai 44. Setiap pengurangan nilai yang berhasil akan ditulis dengan angka 1 dan setiap pengurangan yang gagal ditulis dengan angka 0 seperti yang ditunjukkan pada Gambar 4.21

172 – 128 = 44	1
44 – 64 = X	0
44 – 32 = 12	1
12 – 16 = X	0
12 – 8 = 4	1
4 – 4 = 0	1
0 – 2 = X	0
0 – 1 = X	0

Gambar 4.21 Proses pengurangan keseluruhan

Dari Gambar 4.21, nilai pixel sebesar 172 telah diubah menjadi nilai biner 8 bit yaitu 10101100. Proses ekstraksi dilakukan dari *channel* R → G → B pada tiap *pixel*. Sebagai contoh dilakukan ekstraksi pada *pixel* 1x1 sampai dengan 4x1 dengan nilai sebagai berikut:

p1x1	p2x1	p3x1	p4x1
172 = 10101100	**103 = 01100111**	**0 = 00000000**	**42 = 00101010**
56 = 00111000	**46 = 00101110**	**76 = 01001100**	**156 = 10011100**
192 = 11000000	**157 = 10011101**	**131 = 10000011**	**130 = 10000010**

Dari *nilai pixel* di atas diambil 2 *bit* terendah, sehingga diperoleh *bit* warna sebagai berikut:

Bit Red = 00110010

Bit Green = 00100000

Bit Blue = 00011110

4. Lakukan ekstraksi sampai dengan *bit* penanda akhir yaitu 00000011 11100111 ditemukan.
5. Nilai yang didapat dari tiap *pixel* yang diperoleh saat proses ekstraksi akan dipakai untuk rekonstruksi *file* citra. *File* citra baru dibentuk dengan ekstensi (*.png*) dan nama *file* diberi akhiran "*_decoded*".

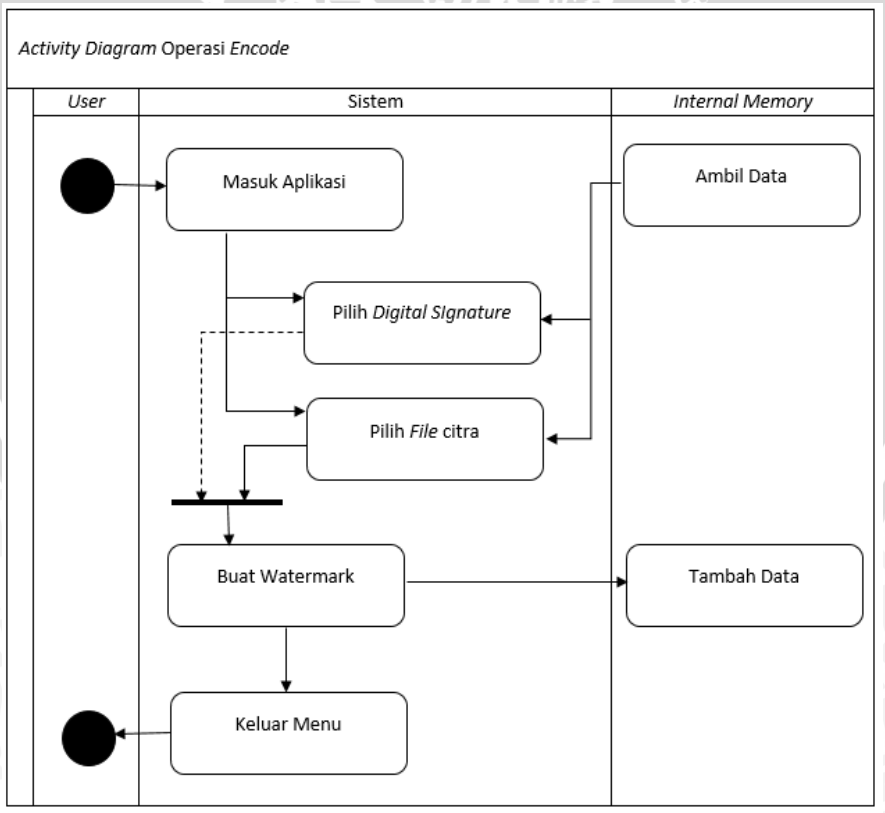
4.5 Perancangan Perangkat Lunak

Perancangan aplikasi dilakukan dalam dua tahap, yaitu perancangan *activity diagram* dan perancangan antarmuka. Perancangan aplikasi dibuat pada sistem operasi Android dengan versi minimal 4.1 Jelly Bean (API 16). Perancangan antarmuka dibuat menggunakan 2 buah *layout .xml*.

4.5.1 Perancangan *Activity Diagram*

Activity Diagram adalah diagram untuk memodelkan aktifitas antara pengguna dan sistem yang berjalan berdasarkan skenario *use case*. Terdapat 2 Activity Diagram yang digunakan yaitu Activity Diagram operasi encode dan Activity Diagram operasi decode. Activity Diagram operasi encode menjelaskan tentang model aktifitas yang terjadi dalam proses encode sedangkan Activity Diagram operasi decode menjelaskan tentang model aktifitas yang terjadi dalam proses decode.

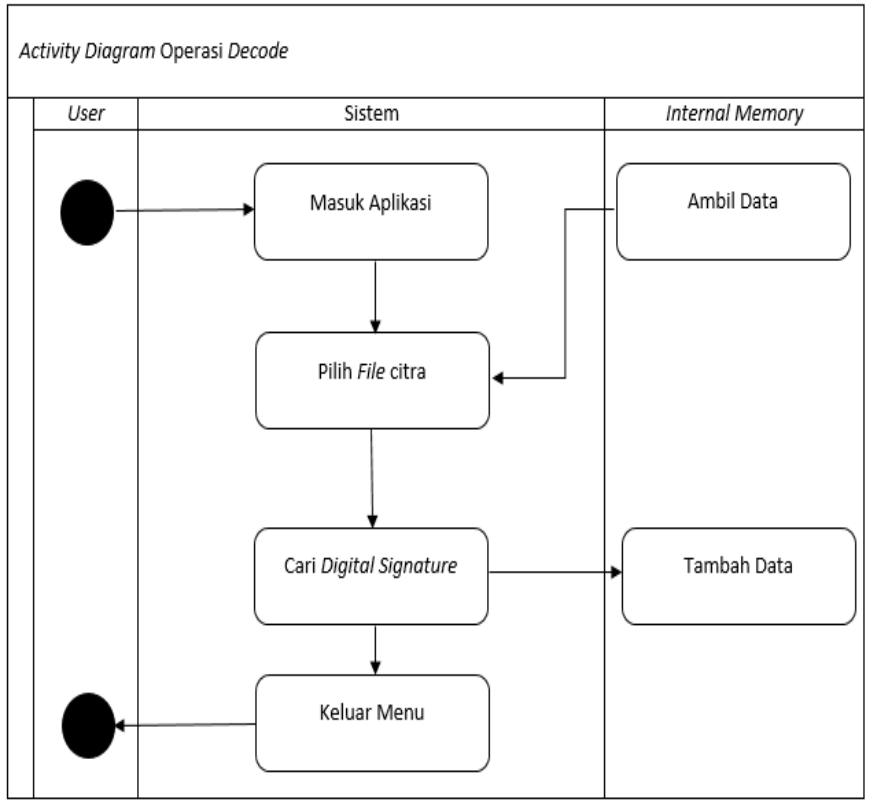
1. *Activity Diagram* Operasi *Encode*



Gambar 4.22 *Activity Diagram* Operasi *Encode*

Pada Gambar 4.22 ditunjukkan aktifitas yang dilakukan oleh *user, user interface*, dan *internal memory*. Deskripsi *activity diagram* sesuai dengan skenario pada *use case* Operasi *Encode* pada Tabel 4.3. *User* masuk aplikasi, kemudian memilih *Digital signature* atau stempel digital dan *file* citra yang ingin dilindungi. Selanjutnya *user* memilih untuk membuat *watermark* dan proses *Image Watermarking* dijalankan dan menghasilkan *encoded-image* yang kemudian disimpan dalam *memory*.

2. *Activity Diagram* Operasi *Decode*



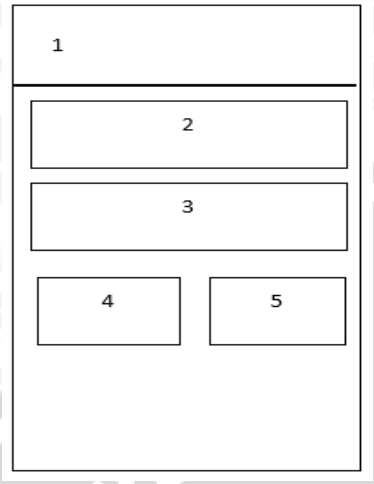
Gambar 4.23 *Activity Diagram* Operasi *Decode*

Pada Gambar 4.23 ditunjukkan aktifitas yang dilakukan oleh *user, user interface*, dan *internal memory*. Deskripsi *activity diagram* sesuai dengan skenario pada *use case* Operasi *Decode* pada Tabel 4.4. *User* masuk aplikasi, kemudian memilih *file* citra yang berisi *digital signature*. Selanjutnya *user* memilih untuk mencari *Digital Signature* dan proses dekripsi serta ekstraksi dijalankan. Hasil dari proses ini adalah *digital signature* yang kemudian disimpan dalam memory.

4.5.2 Perancangan Antarmuka

Pada bagian ini akan dijelaskan mengenai perancangan antarmuka aplikasi *Image Watermarking*. Aplikasi ini digunakan oleh *user* untuk melindungi hak cipta intelektual atas *file* citra yang dimiliki. Berikut adalah detail antarmuka aplikasi yang ditunjukkan pada Gambar 4.24, Gambar 4.25, dan Gambar 4.26.

a. Halaman *Home*

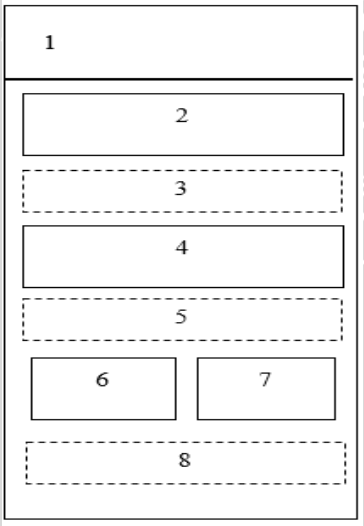


Keterangan:

1. *Image Watermarking* (nama aplikasi)
2. Pilih *Digital Signature* (*Button*)
3. Pilih *Image* untuk *Watermark* (*Button*)
4. Buat *Watermark* (*Button*)
5. Cari *Digital Signature* (*Button*)

Gambar 4.24 Halaman *Home*

b. Halaman *Encode*

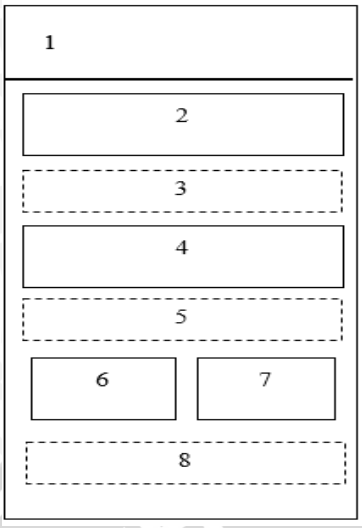


Keterangan:

1. *Image Watermarking* (nama aplikasi)
2. Pilih *Digital Signature* (*Button*)
3. Tampilan *file* citra *Digital Signature*
4. Pilih *Image* untuk *Watermark* (*Button*)
5. Tampilan *File* citra obyek
6. Buat *Watermark* (*Button*)
7. Cari *Digital Signature* (*Button*)
8. Hasil operasi *Watermark*

Gambar 4.25 Halaman *Encode*

c. Halaman *Decode*



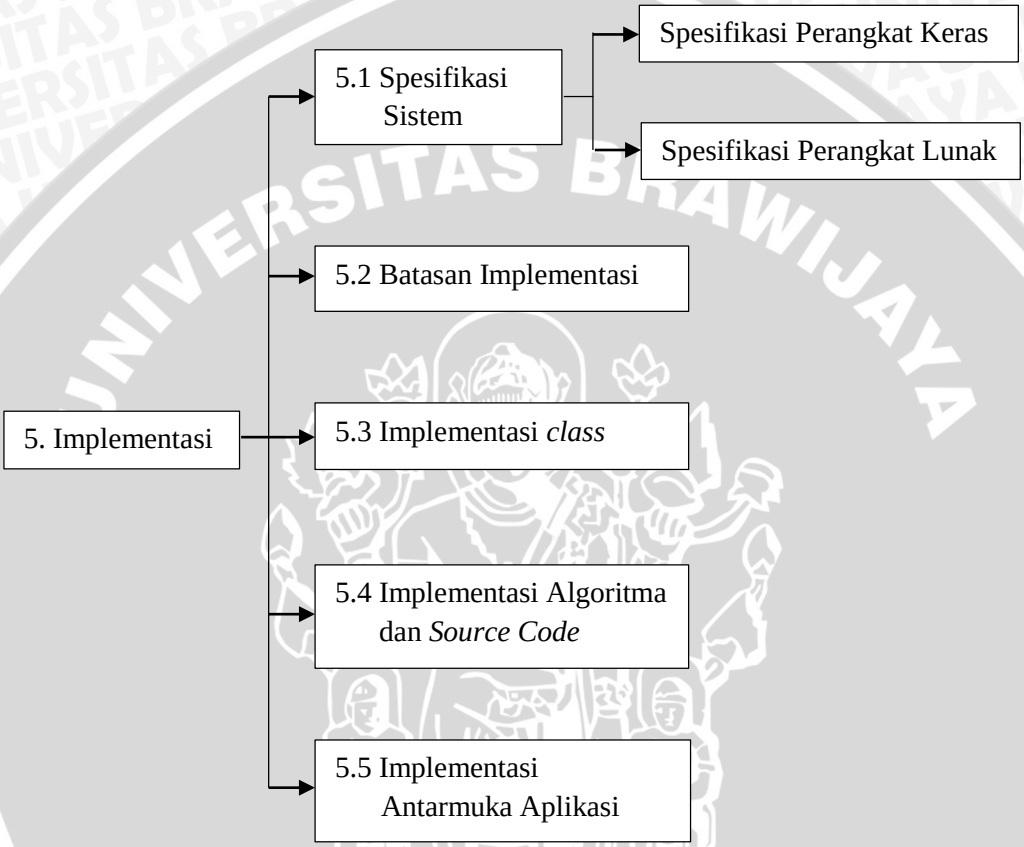
Keterangan:

1. *Image Watermarking* (nama aplikasi)
2. Pilih *Digital Signature* (*Button*)
3. Tampilan file citra digital signature
4. Pilih *Image* untuk *watermark* (*Button*)
5. Tampilan Image berisi *watermark*
6. Buat *watermark* (*Button*)
7. Cari *Digital Signature* (*Button*)
8. Pesan Decode berhasil

Gambar 4.26 Halaman *Decode*

BAB 5 IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi aplikasi berdasarkan hasil yang telah didapatkan dari perancangan. Pembahasan terdiri dari penjelasan tentang spesifikasi sistem, batasan-batasan dalam implementasi, implementasi tiap *class* pada *file* program, implementasi algoritma dan implementasi antarmuka aplikasi. Pohon implementasi ditunjukkan pada Gambar 5.1.



Gambar 5.1 Pohon Implementasi

5.1 Spesifikasi Sistem

Hasil dari analisis kebutuhan dan perancangan aplikasi yang telah dijelaskan pada tahap analisis kebutuhan dan tahap perancangan sistem menjadi dasar untuk dilakukan implementasi menjasi sebuah aplikasi Android yang dapat berfungsi sesuai kebutuhan. Spesifikasi sistem diimplementasikan pada spesifikasi perangkat keras dan perangkat lunak.

5.1.1 Spesifikasi Perangkat Keras

Dalam pengembangan aplikasi *Image Watermarking* menggunakan sebuah komputer dengan spesifikasi *processor, memory,* dan *Graphics Processing Unit* (*GPU)* yang akan dijelaskan pada Tabel 5.1 berikut.

Tabel 5.1 Spesifikasi Perangkat Keras

Nama Komponen	Spesifikasi
Processor	Intel® Core™ i3-4160 Processor (3M Cache, 3.60 GHz)
Memory	8192 MB RAM
GPU	PowerColor PCS+ HD7870 Myst. Edition 2GB GDDR5

Dalam melakukan proses implementasi dan pengujian, perangkat yang digunakan adalah *emulator* Android MEmu dengan spesifikasi yang ditunjukkan pada Tabel 5.2 berikut.

Tabel 5.2 Spesifikasi Perangkat Keras Memu

Nama Komponen	Spesifikasi
System Model	MEmu
System Resolution	1920 x 1080 *(288dpi)*
Operating System	Android OS v4.2.2 Jelly Bean *emulator with* Google APIs (API *Level* 17)

5.1.2 Spesifikasi Perangkat Lunak

Dalam proses pengembangan aplikasi *Image Watermarking* menggunakan perangkat lunak dengan spesifikasi yang akan dijelaskan pada Tabel 5.3 berikut.

Tabel 5.3 Spesifikasi Perangkat Lunak

Nama Komponen	Spesifikasi
Operating System	Windows 8.1 Pro 64 *bit*
Bahasa Pemrograman	Java
IDE (*Integrated Development Environment)*	Android Studio versi 1.5.0 dengan *bundle* SDK *Tools,*
Build Tool	Gradle v2.9
Operating System	Android OS v6.0 Marshmallow (API Level 23)

Pada proses pengembangan aplikasi *Image Watermarking* menggunakan *emulator* Google Nexus 5 dengan spesifikasi perangkat lunak yang akan dijelaskan pada Tabel 5.4 berikut.

Tabel 5.4 Spesifikasi Perangkat Lunak *Emulator*

Nama Komponen	Spesifikasi
Operating System	Android OS v6.0 Marshmallow *emulator with* Google APIs (API Level 23)

5.2 Batasan-Batasan Implementasi

Pada proses implementasi perangkat lunak aplikasi *Image Watermarking* terdapat batasan-batasan dalam prosesnya yaitu sebagai berikut:

1. Aplikasi *Image Watermarking* dirancang untuk dijalankan pada *smartphone* Android menggunakan konsep *native mobile development*.
2. Aplikasi dapat dijalankan tanpa menggunakan koneksi internet dikarenakan menggunakan konsep *offline* (*internal memory*).
3. Pembuatan *layout* pada *user interface* aplikasi menggunakan bahasa XML (*Extensible Markup Language*).
4. Pada proses *watermarking* menggunakan metode *Least Significant Bit*.

5.3 Implementasi *Class*

Setiap aktifitas *class* yang telah dirancang direalisasikan dengan format Java (.java), sedangkan implementasi dari *layout* menggunakan format *Extensible Markup Language* (.xml). Hasil implementasi akan ditunjukkan pada Tabel 5.5.

Tabel 5.5 Implementasi *Class*

Package	**Nama *Activity***	**Nama *Layout***
android.watermark	MainActivity.java	activity_main.xml
android.watermark	EncodeAndDecode.java	fragment_main.xml
android.watermark	TwoReturn.java	fragment_main.xml

5.4 Implementasi Algoritma dan *Source Code*

Aplikasi perangkat bergerak ini mempunyai dua menu utama yang terbagi dalam beberapa proses. Pada penulisan laporan skripsi ini hanya dicantumkan algoritma dari beberapa proses saja. Algoritma proses yang dicantumkan antara lain adalah algoritma *encode*, operasi *decode*, *Least Significant Bit* dan rekonstruksi citra.

5.4.1 Implementasi Algoritma *Encode*

Pada proses *encode* dilakukan proses *watermark* dengan *input*an berupa *file* citra dan *digital signature*. Sistem akan melakukan proses *watermark* ketika *Button* "Buat *watermark*" dijalankan. Implementasi dari algoritma *encode* ditunjukkan *source code* di bawah ini seperti yang terlihat pada Gambar 5.2.

```
1.  public static TwoReturn encodePicture(String hideThis,String
    hideIn,String writeTo){
2.
3.      Bitmap imageToHide = BitmapFactory.decodeFile(hideThis);;
4.      Bitmap placeToHide = BitmapFactory.decodeFile(hideIn);
5.
6.    int scaleFactor = 1;
7.    while(((3 * imageToHide.getWidth() * imageToHide.getHeight()) + 2
    > ((scaleFactor * scaleFactor * placeToHide.getHeight() *
    placeToHide.getWidth()))){
8.      scaleFactor += 1;
9.    }
10.
11.   placeToHide = scaleUp(placeToHide,scaleFactor);
12.   placeToHide = placeToHide.copy(Bitmap.Config.ARGB_8888, true);
13.
14.   int imageWidth = imageToHide.getWidth();
15.   int imageHeight = imageToHide.getHeight();
16.
17.   int[] imagePixels = new int[3 * imageWidth * imageHeight];
18.
19.   int[] currentPixelData = new int[3];
20.   for (int j = 0; j < imageWidth; j+= 1){
21.
22.     for (int i = 0; i < imageHeight; i += 1){
23.
24.       currentPixelData = getPixelData(imageToHide.getPixel(j,i));
25.
26.       imagePixels[3 * ((j * imageHeight) + i)] = currentPixelData[0];
27.       imagePixels[3 * ((j * imageHeight) + i) + 1] =
    currentPixelData[1];
28.       imagePixels[3 * ((j * imageHeight) + i) + 2] =
    currentPixelData[2];
```

```
29.       }
30.    }
31.
32.    int placeWidth = placeToHide.getWidth();
33.    int placeHeight = placeToHide.getHeight();
34.
35.    for(int j = 0; j < placeWidth; j += 1){
36.
37.       for (int i = 0; i < placeHeight; i += 1){
38.
39.          if((i == 0) && (j == 0)){
40.
placeToHide.setPixel(0,0,changeColor(placeToHide.getPixel(0,0),imag
eWidth));
41.          }
42.
43.          else if((i == 1) && (j == 0)){
44.
placeToHide.setPixel(0,1,changeColor(placeToHide.getPixel(0,1),imag
eHeight));
45.          }
46.
47.          else if((j * placeHeight) + i == imagePixels.length + 2){
48.
placeToHide.setPixel(j,i,changeColor(placeToHide.getPixel(j,i),999));
49.
50.             try{
51.                FileOutputStream output = new
FileOutputStream(writeTo, false);
52.
placeToHide.compress(Bitmap.CompressFormat.PNG,0,output);
53.                output.close();
54.             }
55.             catch(Exception e){
```

```
56.                 System.out.println(e);
57.             }
58.
59.             return new TwoReturn(placeToHide,null);
60.         }
61.         else{
62.
placeToHide.setPixel(j,i,changeColor(placeToHide.getPixel(j,i),imagePi
xels[(j * placeHeight) + i - 2]));
63.         }
64.
65.     }
66. }
67.  return null;
68.
69. }
```

Gambar 5.2 *Source Code* Algoritma *Encode*

Source code di atas adalah implementasi algoritma dari proses *encode*. *File* citra dan *digital signature* diubah menjadi *bitmap* dan diletakkan pada *imageToHide* dan *placeToHide* (*source code* 3-4). Setelah itu dilakukan perhitungan untuk menentukan kapasitas yang diperlukan untuk menampung seluruh informasi ke dalam *file* citra yang baru (*source code* 6-12). Langkah selanjutnya adalah menentukan tinggi dan lebar dari citra *digital signature* untuk mendapatkan jumlah *pixel* dari citra tersebut (*source code* 14-28). Proses dilanjutkan dengan menentukan tinggi dan lebar *file* citra kemudian mulai diletakkan citra *digital signature* pada *file* citra (*source code* 32-48). Setelah proses ini berhasil maka akan dibuat *file* citra baru dan akan ditampilkan melalui konfirmasi pesan (*source code* 50-59).

5.4.2 Implementasi Algoritma *Decode*

Algoritma *decode* terjadi setelah *digital signature* berhasil di ekstraksi. Algoritma ini dijalankan ketika *user* menekan *button* "Cari *Digital Signature*". Implementasi dari algoritma *decode* ditunjukkan pada Gambar 5.3.

```
1.  public static Bitmap decodeImage(String encoded,String writeTo,String
    decoded){
2.
3.      Bitmap encodedImage = BitmapFactory.decodeFile(encoded);
```

```
4.
5.    int encodedWidth = encodedImage.getWidth();
6.    int encodedHeight = encodedImage.getHeight();
7.
8.    int hiddenWidth = 0;
9.    int hiddenHeight = 0;
10.
11.   int[] hiddenPixels = null;
12.
13.   int pixelInt = 0;
14.   int pixelIndex = 0;
15.
16.   hiddenWidth = getHiddenInt(encodedImage.getPixel(0,0));
17.   hiddenHeight = getHiddenInt(encodedImage.getPixel(0,1));
18.   hiddenPixels = new int[3 * hiddenWidth * hiddenHeight];
19.   for (int j = 0; j < encodedWidth; j+= 1){
20.
21.      for (int i = 0; i < encodedHeight; i += 1){
22.
23.         if ((i == 0) && (j == 0)){
24.
25.         }
26.         else if ((j == 0) && (i == 1)){
27.
28.         }
29.         else{
30.            pixelInt = getHiddenInt(encodedImage.getPixel(
31.                  j,i));
32.
33.            if (pixelInt == 999){
34.
35.
```

```
36.
37.                 Return
    rebuildImage(hiddenPixels,hiddenWidth,hiddenHeight,writeTo);
38.
39.             }
40.             else{
41.                 hiddenPixels[pixelIndex] = pixelInt;
42.                 pixelIndex++;
43.
44.             }
45.           }
46.       }
47.    }
48.    return null;
49.
50. }
```

Gambar 5.3 Implementasi *Decode*

Source code yang ditunjukkan Gambar 5.3 adalah implementasi algoritma dari proses *decode*. *File* citra yang berisi *digital signature* akan diubah menjadi *bitmap* dan diletakkan pada *encodedImage* (*source code* 3). Langkah selanjutnya adalah menentukan tinggi dan lebar *file* citra yang berisi *digital signature* di dalamnya (*source code* 5-6) lalu membuat class *hiddenWidth*, *hiddenHeight*, dan *hiddenPixels* yang akan digunakan untuk menampung nilai tinggi, lebar, dan jumlah *pixel digital signature* (*source code* 8-11). Setelah itu dilakukan proses pencarian nilai *hiddenWidth* dan *hiddenHeight* untuk mendapatkan nilai *hiddenPixels* (*source code* 16-33). Proses rekonstruksi citra dilakukan untuk mengeluarkan *digital signature* (*source code* 37).

5.4.3 Implementasi Algoritma *Least Significant Bit*

Pada algoritma *Least Significant Bit* terdapat metode untuk penyisipan pesan. Pada proses *watermark*, *bit-bit* dari pesan akan disisipkan pada *bit* terendah dari tiap *pixel* gambar. Proses yang terjadi pada saat dilakukan *watermark* dijelaskan melalui *source code* di bawah ini.

```
1.  public static int changeColor(int currentColor,int inputNumber){
2.
3.    int[] red = intToDigits(Color.red(currentColor));
```

```
4.
5.    int[] green = intToDigits(Color.green(currentColor));
6.
7.    int[] blue = intToDigits(Color.blue(currentColor));
8.
9.    int[] number = intToDigits(inputNumber);
10.
11.
12.   if ((blue[0] == 2) && (blue[1] == 5) && (number[2] > 5)){
13.      blue[1] = 4;
14.   }
15.
16.   else if((number[2] - blue[2]) > 5){
17.      if ((blue[0] != 0) || (blue[1] != 0)){
18.         blue[1] -= 1;
19.      }
20.   }
21.
22.   else if(blue[2] - number[2] > 5){
23.      blue[1] += 1;
24.   }
25.
26.   blue[2] = number[2];
27.
28.
29.   if ((green[0] == 2) && (green[1] == 5) && (number[1] > 5)){
30.      green[1] = 4;
31.   }
32.
33.   else if((number[1] - green[2]) > 5){
34.      if ((green[0] != 0) || (green[1] != 0)){
35.         green[1] -= 1;
```

```
36.         }
37.     }
38.
39.     else if(green[2] - number[1] > 5){
40.         green[1] += 1;
41.     }
42.
43.     green[2] = number[1];
44.
45.
46.     if ((red[0] == 2) && (red[1] == 5) && (number[0] > 5)){
47.         red[1] = 4;
48.     }
49.
50.     else if((number[0] - red[2]) > 5){
51.         if ((red[0] != 0) || (red[1] != 0)){
52.             red[1] -= 1;
53.         }
54.     }
55.
56.     else if(red[2] - number[0] > 5){
57.         red[1] += 1;
58.     }
59.
60.     red[2] = number[0];
61.
62.
63.     return Color.argb(255, digitsToInt(red), digitsToInt(green),
64.             digitsToInt(blue));
65.
66. }
```

Gambar 5.4 Implementasi Algoritma *Least Significant Bit*

Pada Gambar 5.4 adalah implementasi dari *watermark*. Langkah pertama adalah dilakukan perubahan nilai warna dan angka yang akan disisipkan dari bentuk *integer* ke dalam bentuk digit (*source code* 3-9). Penyisipan dilakukan pada tiap-tiap *channel* warna secara berurutan dimulai dari warna merah, hijau, dan biru. Setiap nilai dari warna tidak akan melebihi 255 karena dapat berdampak dengan warna yang akan dihasilkan menjadi *undefined* (*source code* 12-60). Hasil dari penyisipan tersebut kemudian akan dirubah kembali dari bentuk digit ke dalam bentuk *integer* yang kemudian akan menjadi hasil output warna yang berisi angka rahasia (source code 63).

5.4.4 Implementasi Algoritma Rekonstruksi Citra

Pada algoritma rekonstruksi citra dilakukan proses pembangunan ulang citra dari data jumlah *pixel*, tinggi, dan lebar. Algoritma ini dijalankan ketika *user* menekan *button "*Cari *Digital Signature*". Langkah-langkah pada implementasi rekonstruksi citra dijelaskan melalui *source code* di bawah ini.

```
1.  public static Bitmap rebuildImage(int[] pixels,int width,int height,String writeTo){
2.
3.    int[] color  = new int[pixels.length/3];
4.    for (int i = 0; i < color.length; i += 1){
5.       color[i] = Color.argb(255, pixels[(3 * i)],pixels[(3 * i) + 1],pixels[(3 * i) + 2]);
6.    }
7.
8.    int[] rotated_color = new int[pixels.length/3];
9.
10.            int counter = 0;
11.            for (int i = 0; i < height; i++)
12.               for (int j = 0; j < width; j++){
13.                  rotated_color[counter] = color[j*height + i];
14.                  counter++;
15.               }
16.
17.            Bitmap hiddenImage = Bitmap.createBitmap(rotated_color,width,height,Bitmap.Config.ARGB_8888);
18.            try{
```

```
19.             FileOutputStream output = new FileOutputStream(writeTo);
20.
hiddenImage.compress(Bitmap.CompressFormat.PNG,0,output);
21.
22.             output.close();
23.         }
24.         catch(Exception e){
25.             System.out.println(e);
26.         }
27.         return hiddenImage;
28.
29.     }
```

Gambar 5.5 Implementasi Rekonstruksi citra

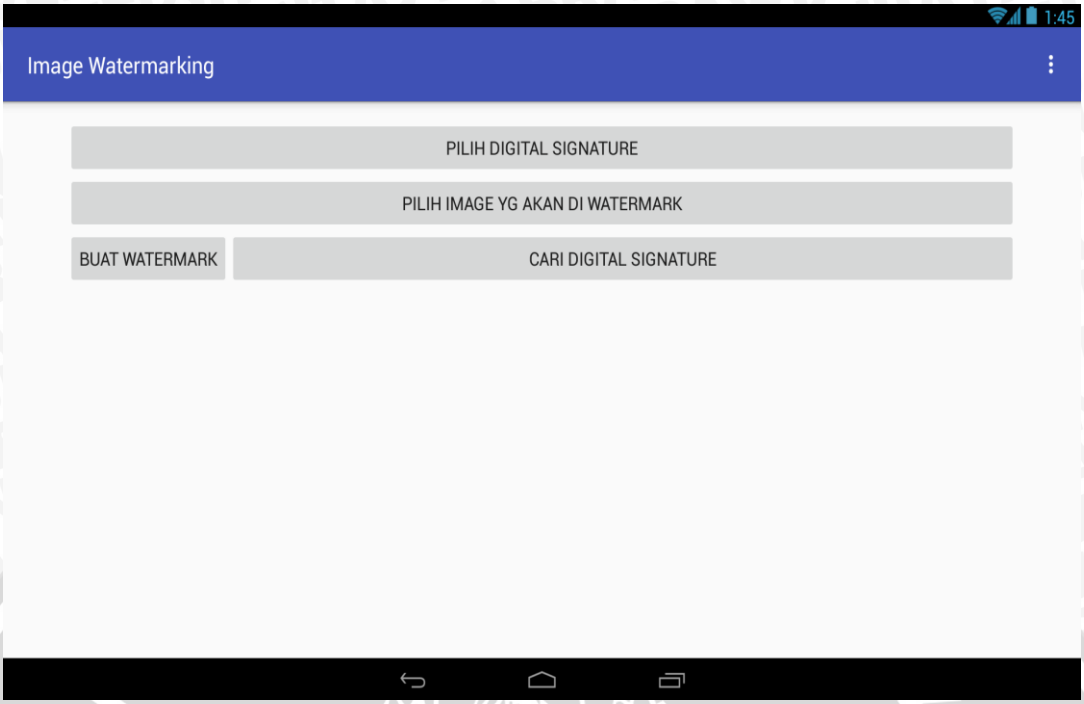
Pada Gambar 5.5, *int[] color* merepresentasikan warna dari *pixel* yang terdapat dalam *digital signature* yang disembunyikan. Setelah itu dibuat *class* baru yaitu *rotated_color* untuk mencari tinggi dan lebar dari *file* citra tersebut. Hasil yang didapat digunakan dalam proses rekonstruksi citra dalam format PNG (*source* code 17-27).

5.5 Implementasi Antarmuka Aplikasi

Pada implementasi antarmuka aplikasi akan ditampilkan hasil implementasi antarmuka aplikasi *Image Watermarking*. Aplikasi ini hanya memiliki 1 halaman antarmuka yaitu halaman *Home* yang akan dijelaskan berikut.

1. Halaman *Home*

Halaman *Home* adalah halaman utama dari aplikasi dan ditampilkan pada saat pertama kali aplikasi dijalankan. Halaman *Home* terdiri dari empat menu utama yaitu "Pilih *Digital Signature*", "Pilih *Image* yang Akan di *Watermark*", "Buat *Watermark*", dan "Cari *Digital Signature*". Halaman *Home* ditunjukkan pada Gambar 5.6.

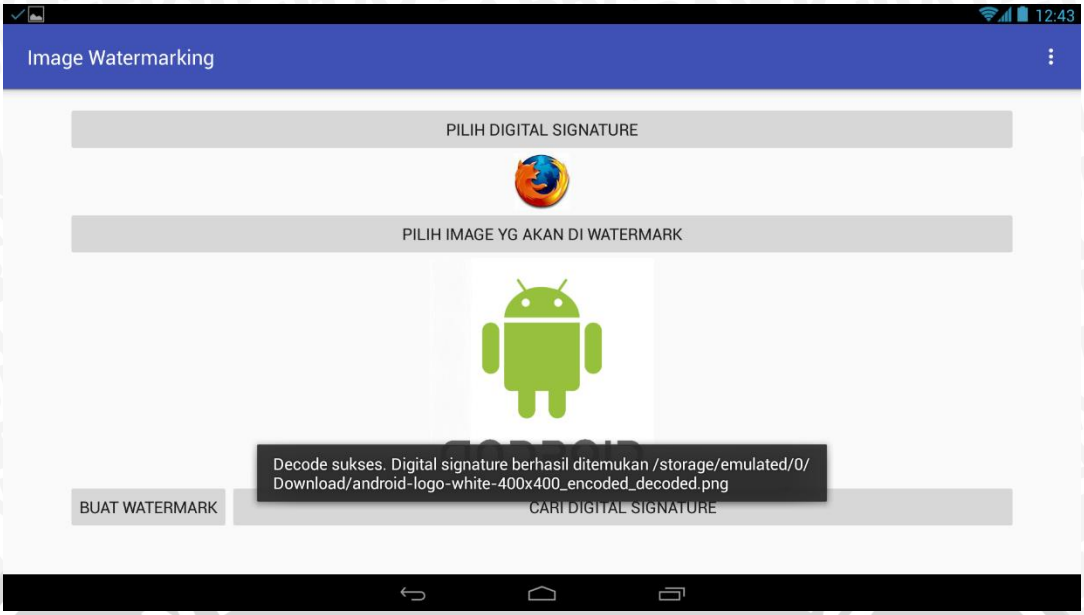


Gambar 5.6 Tampilan Halaman *Home*

Tampilan ketika menu "Buat *Watermark*" dijalankan dan ditunjukkan pada Gambar 5.7.

Gambar 5.7 Tampilan Buat *Watermark*

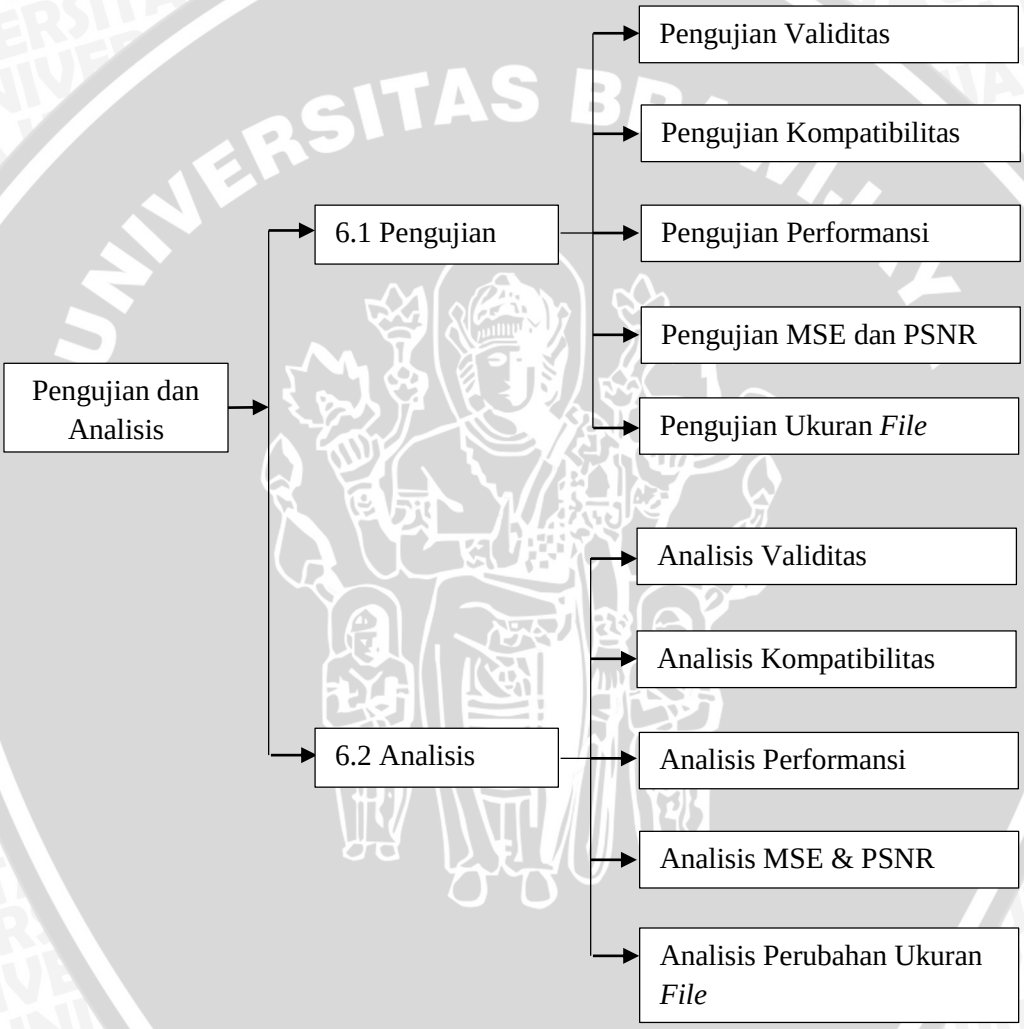
Tampilan ketika menu "Cari *Digital Signature*" dijalankan dan ditunjukkan pada Gambar 5.8.



Gambar 5.8 Tampilan Cari *Digital Signature*

BAB 6 PENGUJIAN DAN ANALISIS

Pada bab ini akan dilakukan proses pengujian dan analisis terhadap aplikasi yang telah dibangun. Proses pengujian dilakukan melalui lima tahapan yaitu pengujian validitas, pengujian kompatibilitas, pengujian performansi, pengujian MSE (*Mean Squared Error*) dan PSNR (*Peak Signal to Noise Ratio*), serta pengujian ukuran *file*. Pada proses analisis juga dilakukan sebanyak lima tahap yaitu analisis validitas, analisis kompatibilitas, analisis performansi, analisis MSE dan PSNR, serta analisis perubahan ukuran *file* seperti yang ditunjukkan pada Gambar 6.1.



Gambar 6.1 Pohon Pengujian dan Analisis

6.1 Pengujian

Pada proses pengujian aplikasi *Image Watermarking* dilakukan melalui lima tahap seperti yang dijabarkan pada skenario pengujian, yaitu pengujian validitas, pengujian kompatibilitas, pengujian performansi, pengujian MSE dan PSNR, serta

pengujian ukuran *file*. Pada subbab berikutnya akan dijelaskan tiap-tiap proses dari tahapan pengujian.

6.1.1 Pengujian Validitas

Pengujian validitas digunakan untuk mengetahui apakah sistem yang dibangun sudah benar sesuai yang dibutuhkan. Daftar kebutuhan yang telah dirumuskan akan menjadi acuan untuk melakukan validitas. Pengujian validitas menggunakan metode *Black Box*, karena tidak diperlukan konsentrasi terhadap alur jalannya algoritma program dan lebih ditekankan untuk menemukan kesesuaian antara kinerja sistem dengan daftar kebutuhan.

6.1.1.1 Kasus Uji Validitas

Pada penelitian ini dilakukan uji kasus validitas terhadap aplikasi *Image Watermarking* yang telah dibuat. Untuk mengetahui kesesuaian antara daftar daftar kebutuhan dan kinerja sistem, setiap kebutuhan fungsional dilakukan proses pengujian kasus yang ditunjukkan pada Tabel 6.1 sampai dengan Tabel 6.5.

Kasus uji validitas yang pertama adalah buka aplikasi. Pengujian ini bertujuan untuk memastikan apakah aplikasi dapat dibuka dengan baik. Kasus uji buka aplikasi ditunjukkan pada Tabel 6.1.

Tabel 6.1 Uji Validitas Buka Aplikasi

Nomor Kasus Uji	VAL_01
Nama Kasus Uji	Kasus uji buka aplikasi
Objek Uji	Kebutuhan Fungsional
Tujuan Pengujian	Untuk memastikan aplikasi dapat dibuka dengan baik.
Prosedur Uji	Tekan *icon* aplikasi pada daftar menu aplikasi *smartphone*.
Hasil yang diharapkan	Aplikasi dapat terbuka dan menampilkan halaman *home*.

Kasus uji validitas yang kedua adalah uji pilih *digital signature*. Pengujian ini untuk memastikan apakah button "Pilih *Digital Signature*" dapat berfungsi dengan baik, yaitu aplikasi dapat memilih dan mengambil *file* citra dengan baik. Kasus uji pilih *digital signature* ditunjukkan pada Tabel 6.2.

Tabel 6.2 Uji Validitas Pilih *Digital Signature*

Nomor Kasus Uji	VAL_02
Nama Kasus Uji	Kasus uji pilih *digital signature*
Objek Uji	Kebutuhan Fungsional (SRS_001)

Tujuan Pengujian	Untuk memastikan *button* "Pilih *Digital Signature*" dapat berfungsi dengan baik yaitu untuk mengambil *file* citra dari *Gallery*.
Prosedur Uji	Tekan *button* "Pilih *Digital Signature*" pada halaman *home*.
Hasil yang diharapkan	Aplikasi dapat mengambil *file* citra dari *Gallery*.

Kasus uji validitas yang ketiga adalah uji pilih *image* yang akan di watermark. Pengujian ini untuk memastikan apakah button "Pilih *Image* yang Akan di Watermark" dapat berfungsi dengan baik, yaitu aplikasi dapat memilih dan mengambil *file* citra dengan baik. Kasus uji pilih *image* yang akan di watermark ditunjukkan pada Tabel 6.3.

Tabel 6.3 Uji Validitas Pilih *Image* yang Akan di Watermark

Nomor Kasus Uji	VAL_03
Nama Kasus Uji	Kasus uji pilih *image* yang akan di watermark
Objek Uji	Kebutuhan Fungsional (SRS_001 dan SRS_002)
Tujuan Pengujian	Untuk memastikan *button* "Pilih *Image* yang Akan di Watermark" dapat berfungsi dengan baik yaitu untuk mengambil *file* citra dari *Gallery*.
Prosedur Uji	Tekan *button* "Pilih *Image* yang Akan di Watermark" pada halaman *home*.
Hasil yang diharapkan	Aplikasi dapat mengambil *file* citra dari *Gallery*.

Kasus uji validitas yang keempat adalah uji buat watermark. Pengujian ini bertujuan untuk memastikan apakah *button* "Buat Watermark" dapat berfungsi dengan baik, yaitu untuk membuat *file* citra baru yang berisi watermark di dalamnya dan menyimpan *file* tersebut ke dalam *internal memory storage*. Kasus uji buat watermark ditunjukkan pada Tabel 6.4.

Tabel 6.4 Uji Validitas Buat Watermark

Nomor Kasus Uji	VAL_04
Nama Kasus Uji	Kasus uji buat watermark
Objek Uji	Kebutuhan Fungsional (SRS_001)
Tujuan Pengujian	Untuk memastikan aplikasi dapat membuat *file* citra dengan watermark di dalamnya
Prosedur Uji	Tekan *button* "Buat Watermark" pada halaman *home*.

Hasil yang diharapkan	Aplikasi dapat membuat *file* citra yang berisi watermark di dalamnya dan menyimpan *file* citra baru ke dalam *internal memory storage*.

Kasus uji validitas yang kelima adalah uji cari *digital signature*. Pengujian ini bertujuan untuk memastikan apakah aplikasi dapat melakukan proses ekstraksi *digital signature* dan menyimpan *file* tersebut ke dalam *internal memory storage*. Kasus uji cari *digital signature* ditunjukkan pada Tabel 6.5.

Tabel 6.5 Uji Validitas Cari *Digital Signature*

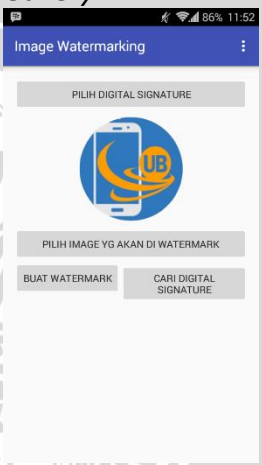
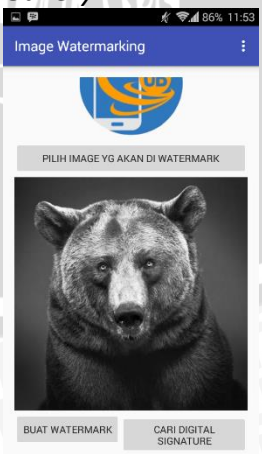
Nomor Kasus Uji	VAL_05
Nama Kasus Uji	Kasus uji cari *digital signature*
Objek Uji	Kebutuhan Fungsional (SRS_002)
Tujuan Pengujian	Untuk memastikan aplikasi dapat melakukan ekstraksi *digital signature* dari *file* citra.
Prosedur Uji	Tekan *button* "Cari *Digital Signature*" pada halaman *home*.
Hasil yang diharapkan	Aplikasi dapat melakukan proses ekstraksi *digital signature* dan menyimpan *file* citra tersebut ke dalam *internal memory storage*.

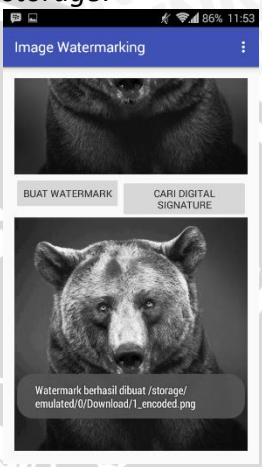
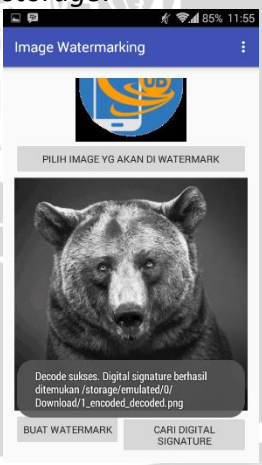
6.1.1.2 Hasil Pengujian Validitas

Berdasarkan kasus uji yang dilakukan sesuai prosedur pengujian diperoleh hasil pengujian aplikasi *Image Watermarking*. Hasil pengujian validitas dari tiap-tiap kebutuhan fungsional aplikasi ditunjukkan pada Tabel 6.6.

Tabel 6.6 Hasil Pengujian Validitas

No.	Nomor Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan	Status Validitas
1.	VAL_01	Aplikasi dapat terbuka dan menampilkan halaman *home*.	Aplikasi dapat terbuka dan menampilkan	VALID

			halaman *home*. 	
2.	VAL_02	Aplikasi dapat mengambil *file* citra dari *Gallery*.	Aplikasi dapat mengambil *file* citra dari *Gallery*. 	VALID
3.	VAL_03	Aplikasi dapat mengambil *file* citra dari *Gallery*.	Aplikasi dapat mengambil *file* citra dari *Gallery*. 	VALID

4.	VAL_04	Aplikasi dapat membuat *file* citra yang berisi watermark di dalamnya dan menyimpan *file* citra baru ke dalam *internal memory storage*.	Aplikasi dapat membuat *file* citra yang berisi watermark di dalamnya dan menyimpan *file* citra baru ke dalam *internal memory storage*. 	VALID
5.	VAL_05	Aplikasi dapat melakukan proses ekstraksi *digital signature* dan menyimpan *file* citra tersebut ke dalam *internal memory storage*.	Aplikasi dapat melakukan proses ekstraksi *digital signature* dan menyimpan *file* citra tersebut ke dalam *internal memory storage*. 	VALID

6.1.2 Pengujian Kompatibilitas

Pengujian kompatibilitas digunakan untuk mengetahui kompatibilitas aplikasi pada beberapa sistem operasi Android yang berbeda. Pengujian kompatibilitas dilakukan pada versi 4.2.2 Jelly Bean (API Level 17), versi 4.4.4 KitKat (API Level 19), dan versi 6.0 Marshmallow (API Level 23).

6.1.2.1 Pengujian Kompatibilitas Android Versi 4.2.2

Pengujian kompatibilitas sistem operasi Android versi 4.2.2 dilakukan untuk mengetahui validitas kinerja fitur-fitur yang disediakan oleh sistem terkait dengan antarmuka sistem. Pada pengujian kompatibilitas Android versi 4.2.2 dilakukan pada emulator Windroye. Tabel 6.7 menjelaskan prosedur dan hasil kasus pengujian kompatibilitas pada sistem Android versi 4.2.2.

Tabel 6.7 Kasus Uji dan Hasil Pengujian Kompatibilitas Android versi 4.2.2

Nama Kasus Uji	Pengujian kompatibilitas Android versi 4.2.2
Objek Uji	Kebutuhan nonfungsional
Tujuan Pengujian	Pengujian dilakukan untuk mengetahui validitas kinerja dari sistem dalam menyediakan fitur-fitur dan antarmuka pengguna.
Prosedur Uji	Membuka aplikasi sesuai dengan spesifikasi kebutuhan sistem.
Hasil yang diharapkan	Sistem dapat menampilkan fitur-fitur dan antarmuka sesuai dengan implementasi antarmuka sistem.
Hasil yang didapatkan	Sistem dapat menampilkan fitur-fitur dan antarmuka sesuai dengan implementasi antarmuka sistem. 
Status Validitas	Valid

6.1.2.2 Pengujian Kompatibilitas Android Versi 4.4.4

Pengujian kompatibilitas sistem operasi Android versi 4.4.4 dilakukan untuk mengetahui validitas kinerja fitur-fitur yang disediakan oleh sistem terkait dengan antarmuka sistem. Pada pengujian kompatibilitas Android versi 4.4.4 dilakukan pada perangkat bergerak Huawei Ascend P1. Tabel 6.8 menjelaskan prosedur dan hasil kasus pengujian kompatibilitas pada sistem Android versi 4.4.4.

Tabel 6.8 Kasus Uji dan Hasil Pengujian Kompatibilitas Android versi 4.4.4

Nama Kasus Uji	Pengujian kompatibilitas Android versi 4.4.4
Objek Uji	Kebutuhan nonfungsional
Tujuan Pengujian	Pengujian dilakukan untuk mengetahui validitas kinerja dari sistem dalam menyediakan fitur-fitur dan antarmuka pengguna.
Prosedur Uji	Membuka aplikasi sesuai dengan spesifikasi kebutuhan sistem.
Hasil yang diharapkan	Sistem dapat menampilkan fitur-fitur dan antarmuka sesuai dengan implementasi antarmuka sistem.
Hasil yang didapatkan	Sistem dapat menampilkan fitur-fitur dan antarmuka sesuai dengan implementasi antarmuka sistem. 
Status Validitas	Valid

6.1.2.3 Pengujian Kompatibilitas Android Versi 6.0

Pengujian kompatibilitas sistem operasi Android versi 6.0 dilakukan untuk mengetahui validitas kinerja fitur-fitur yang disediakan oleh sistem terkait dengan antarmuka sistem. Pada pengujian kompatibilitas Android versi 6.0 dilakukan pada *emulator* Android *Studio*. Tabel 6.9 menjelaskan prosedur dan hasil kasus pengujian kompatibilitas pada sistem Android versi 6.0.

Tabel 6.9 Kasus Uji dan Hasil Pengujian Kompatibilitas Android versi 6.0

Nama Kasus Uji	Pengujian kompatibilitas Android versi 5.0
Objek Uji	Kebutuhan nonfungsional
Tujuan Pengujian	Pengujian dilakukan untuk mengetahui validitas kinerja dari sistem dalam menyediakan fitur-fitur dan antarmuka pengguna.

Prosedur Uji	Membuka aplikasi sesuai dengan spesifikasi kebutuhan sistem.
Hasil yang diharapkan	Sistem dapat menampilkan fitur-fitur dan antarmuka sesuai dengan implementasi antarmuka sistem.
Hasil yang didapatkan	Sistem dapat menampilkan fitur-fitur dan antarmuka sesuai dengan implementasi antarmuka sistem. 
Status Validitas	Valid

6.1.3 Pengujian Performansi

Pengujian performansi adalah pengujian yang dilakukan untuk mengetahui kinerja dari sistem yang telah dibuat. Parameter kinerja yang dinilai meliputi perbandingan citra sebelum dan sesudah disisipkan serta perbandingan ukuran *file* citra sebelum dan sesudah disisipkan *digital signature*. Selain itu juga diuji *robustness file* citra setelah melalui proses pengiriman pada beberapa sosial media.

6.1.3.1 Uji Perbandingan Citra

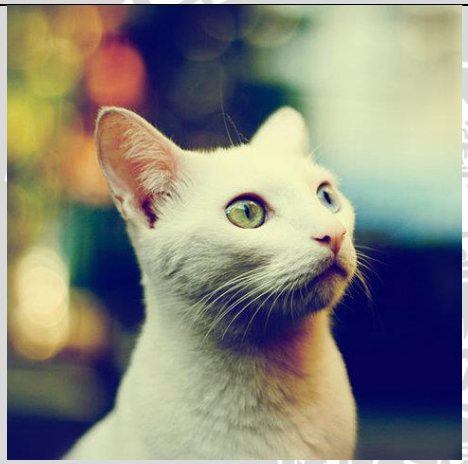
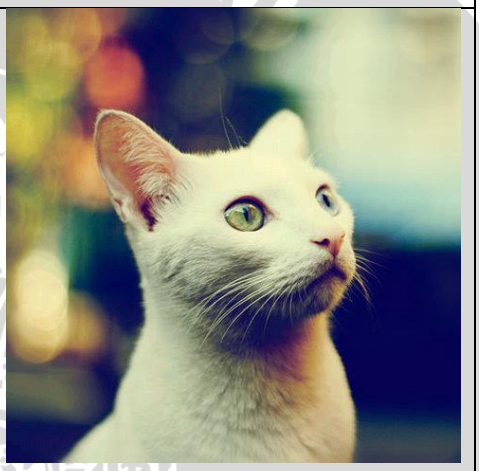
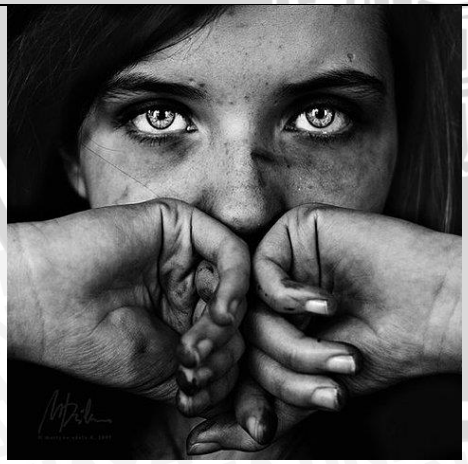
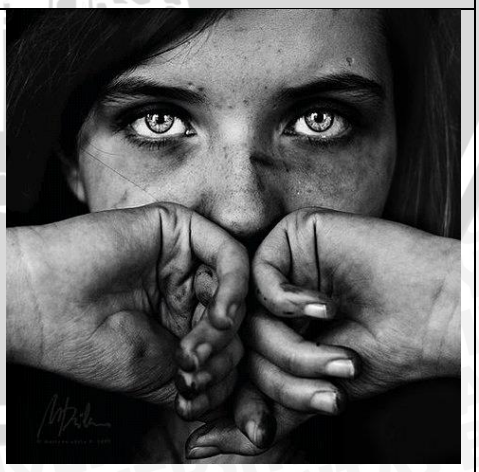
Uji Perbandingan citra dilakukan untuk mengetahui perubahan kualitas citra pada saat sebelum dan sesudah melalui proses watermark. Mula-mula semua *file* citra disisipkan dengan *digital signature* yang sama. Berikut adalah file citra *digital signature* yang akan digunakan dalam proses watermark ditunjukkan pada Gambar 6.2

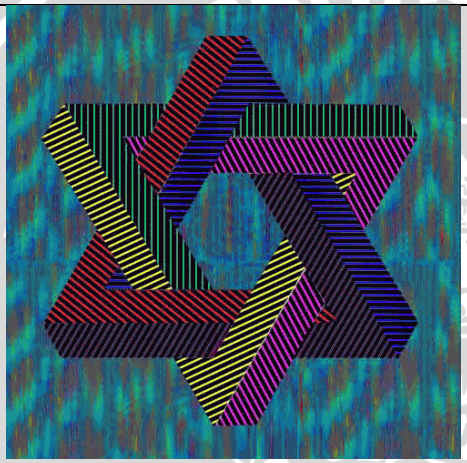
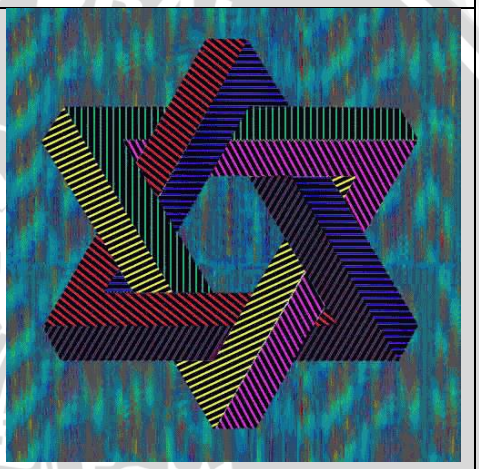
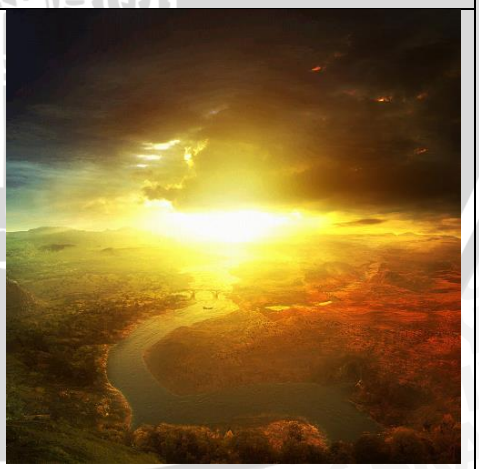
Gambar 6.2 *File* Citra Digital Signature yang Digunakan

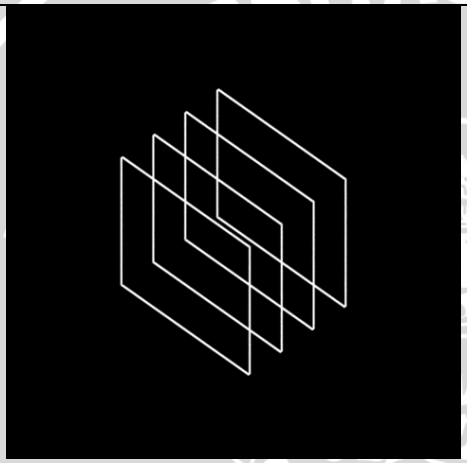
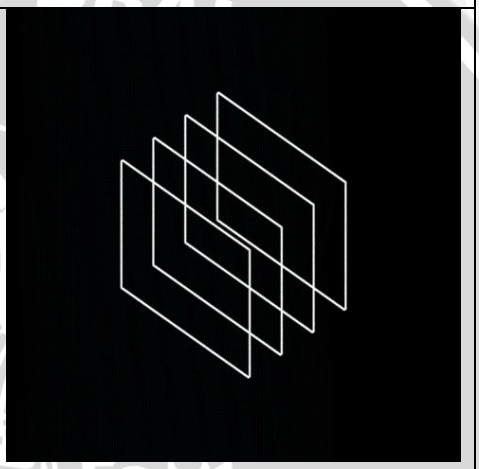
Setelah penyisipan, dilakukan pengamatan secara kasat mata terhadap *file* citra antara sebelum dan sesudah disisipi *digital signature*. Nama *file* yang telah disisipi pesan memiliki akhiran "**_encoded**" dan berformat (*.png)*. Perbandingan kasat mata dengan *input file* (*.png*), (*.gif*), (*.jpeg*), dan (*.jpg*) ditunjukkan pada Tabel 6.10.

Tabel 6.10 Perbandingan Citra pada *File*

No.	Tampilan *File* citra sebelum disisipkan	Tampilan *File* citra sesudah disisipkan
1.	 1.jpeg	 1_encoded.png
2.		

	2.png	2_encoded.png
3.		
	4.jpg	4_encoded.png
4.		
	6.jpg	6_encoded.png
5.		
	7.jpg	7_encoded.png

6.	 8.jpg	 8_encoded.png
7.	 10.gif	 10_encoded.png
8.	 11.png	 11_encoded.png

9.	 12.jpg	 12_encoded.png
10.	 14.gif	 14_encoded.png
11.	 15.jpg	 15_encoded.png

12.		
	16.jpg	16_encoded.png
13.		
	17.jpg	17_encoded.png
14.		
	18.jpg	18_encoded.png

15.	 19.jpg	 19_encoded.png
16.	 21.jpg	 21_encoded.png
17.	 22.jpg	 22_encoded.png

18.	 23.jpg	 23_encoded.png
19.	 24.jpg	 24_encoded.png
20.	 25.jpg	 25_encoded.png

6.1.3.2 Uji Perbandingan Ukuran *File* Citra

Uji perbandingan ukuran *file* citra bertujuan untuk mengukur *size* dari *file* citra sebelum dan sesudah melalui proses watermark. Selain itu juga diukur kedalaman

bit dan resolusi dari *file* citra. Perbandingan ukuran citra dengan *input file* (*.png*), (*.gif*), (*.jpeg*), dan (*.jpg*) ditunjukkan pada Tabel 6.11.

Tabel 6.11 Uji Perbandingan Ukuran *File* Citra

No.	Nama *File*	Resolusi (*pixels*)	*Bit depth*	*Size* (*Bytes*)	Selisih (*Bytes*)
1.	1.jpeg	500x500	8	37,829	341,437
	1_encoded.png	500x500	24	379,266	
2.	2.png	500x500	8	155,403	240,594
	2_encoded.png	500x500	24	395,997	
3.	4.jpg	500x500	8	126,663	298,480
	4_encoded.png	500x500	24	425,143	
4.	6.jpg	480x480	24	42,039	268,740
	6_encoded.png	480x480	24	310,779	
5.	7.jpg	480x480	8	45,117	307,631
	7_encoded.png	480x480	24	352,748	
6.	8.jpg	500x500	24	152,442	188,463
	8_encoded.png	500x500	24	340,905	
7.	10.gif	500x500	8	285,460	129,373
	10_encoded.png	500x500	24	414,833	
8.	11.png	500x500	32	378,681	3,064
	11_encoded.png	500x500	24	381,745	
9.	12.jpg	500x500	24	78,618	255,766
	12_encoded.png	500x500	24	334,384	
10.	14.gif	480x480	8	709,547	504,958
	14_encoded.png	480x480	24	204,589	
11.	15.jpg	500x500	24	73,105	236,928
	15_encoded.png	500x500	24	310,033	
12.	16.jpg	500x500	24	201,990	215,801
	16_encoded.png	500x500	24	417,791	
13.	17.jpg	500x500	24	64,234	426,055
	17_encoded.png	500x500	24	490,289	

14.	18.jpg	500x500	24	156,160	301,515
	18_encoded.png	500x500	24	457,675	
15.	19.jpg	500x500	24	61,385	238,109
	19_encoded.png	500x500	24	299,494	
16.	21.jpg	500x500	24	54,932	246,819
	21_encoded.png	500x500	24	301,751	
17.	22.jpg	500x500	24	58,048	309,644
	22_encoded.png	500x500	24	367,692	
18.	23.jpg	500x500	24	106,100	263,534
	23_encoded.png	500x500	24	369,634	
19.	24.jpg	500x500	24	17,732	253,516
	24_encoded.png	500x500	24	271,248	
20.	25.jpg	480x480	24	20,466	238,136
	25_encoded.png	480x480	24	258,602	

6.1.3.3 Uji *Robustness*

Uji *robustness* dilakukan untuk mengetahui apakah *encoded-image* yang dikirim dengan sosial media dapat diekstrak kembali. Setelah disisipi *digital signature*, dilakukan pengiriman melalui Whatsapp *Messenger*, Line *Messenger*, dan Blackberry *Messenger*, Gmail, dan Facebook. Pengujian *robustness* dilakukan sebanyak 20 kali pengiriman *encoded-image* untuk mengetahui tingkat keberhasilan dari ekstraksi pesan.

1. Pada uji *robustness* yang pertama dilakukan uji pengiriman *encoded-image* dengan aplikasi WhatsApp *Messenger* seperti yang terlihat pada Tabel 6.12.

Tabel 6.12 Uji *Robustness* dengan Media WhatsApp *Messenger*

No.	Nama *File*	*Size* Sebelum Pengiriman (*Bytes*)	*Size* Sesudah Pengiriman (*Bytes*)	Status Ekstraksi Pesan
1.	1_encoded.png	379,266	43,334	Gagal
2.	2_encoded.png	395,997	53,699	Gagal
3.	4_encoded.png	425,143	54,653	Gagal
4.	6_encoded.png	310,779	30,665	Gagal
5.	7_encoded.png	352,748	43,839	Gagal
6.	8_encoded.png	340,905	25,317	Gagal

7.	10_encoded.png	414,833	78,221	Gagal
8.	11_encoded.png	381,745	34,066	Gagal
9.	12_encoded.png	334,384	40,249	Gagal
10.	14_encoded.png	204,589	18,320	Gagal
11.	15_encoded.png	310,033	21,094	Gagal
12.	16_encoded.png	417,791	48,255	Gagal
13.	17_encoded.png	490,289	68,861	Gagal
14.	18_encoded.png	457,675	53,140	Gagal
15.	19_encoded.png	299,494	29,903	Gagal
16.	21_encoded.png	301,751	23,014	Gagal
17.	22_encoded.png	367,692	44,575	Gagal
18.	23_encoded.png	369,634	25,827	Gagal
19.	24_encoded.png	271,248	22,899	Gagal
20.	25_encoded.png	258,602	20,326	Gagal

2. Pada uji *robustness* yang kedua dilakukan uji pengiriman *encoded-image* dengan aplikasi Line *Messenger* seperti yang terlihat pada Tabel 6.13.

Tabel 6.13 Uji *Robustness* dengan Media Line *Messenger*

No.	Nama *File*	*Size* Sebelum Pengiriman (*Bytes*)	*Size* Sesudah Pengiriman (*Bytes*)	Status Ekstraksi Pesan
1.	1_encoded.png	379,266	38,860	Gagal
2.	2_encoded.png	395,997	43,612	Gagal
3.	4_encoded.png	425,143	41,858	Gagal
4.	6_encoded.png	310,779	24,513	Gagal
5.	7_encoded.png	352,748	35,634	Gagal
6.	8_encoded.png	340,905	18,658	Gagal
7.	10_encoded.png	414,833	60,803	Gagal
8.	11_encoded.png	381,745	25,658	Gagal
9.	12_encoded.png	334,384	31,286	Gagal
10.	14_encoded.png	204,589	13,556	Gagal
11.	15_encoded.png	310,033	15,529	Gagal

12.	16_encoded.png	417,791	37,888	Gagal
13.	17_encoded.png	490,289	61,785	Gagal
14.	18_encoded.png	457,675	41,710	Gagal
15.	19_encoded.png	299,494	24,526	Gagal
16.	21_encoded.png	301,751	16,830	Gagal
17.	22_encoded.png	367,692	32,781	Gagal
18.	23_encoded.png	369,634	18,598	Gagal
19.	24_encoded.png	271,248	18,704	Gagal
20.	25_encoded.png	258,602	15,912	Gagal

3. Pada uji *robustness* yang ketiga dilakukan uji pengiriman *encoded-image* dengan aplikasi Blackberry *Messenger* seperti terlihat pada Tabel 6.14.

Tabel 6.14 Uji *Robustness* dengan media Blackberry *Messenger*

No.	Nama *File*	*Size* Sebelum Pengiriman (*Bytes*)	*Size* Sesudah Pengiriman (*Bytes*)	Status Ekstraksi Pesan
1.	1_encoded.png	379,266	379,266	Berhasil
2.	2_encoded.png	395,997	395,997	Berhasil
3.	4_encoded.png	425,143	425,143	Berhasil
4.	6_encoded.png	310,779	310,779	Berhasil
5.	7_encoded.png	352,748	352,748	Berhasil
6.	8_encoded.png	340,905	340,905	Berhasil
7.	10_encoded.png	414,833	414,833	Berhasil
8.	11_encoded.png	381,745	381,745	Berhasil
9.	12_encoded.png	334,384	334,384	Berhasil
10.	14_encoded.png	204,589	204,589	Berhasil
11.	15_encoded.png	310,033	310,033	Berhasil
12.	16_encoded.png	417,791	417,791	Berhasil
13.	17_encoded.png	490,289	490,289	Berhasil
14.	18_encoded.png	457,675	457,675	Berhasil
15.	19_encoded.png	299,494	299,494	Berhasil
16.	21_encoded.png	301,751	301,751	Berhasil

17.	22_encoded.png	367,692	367,692	Berhasil
18.	23_encoded.png	369,634	369,634	Berhasil
19.	24_encoded.png	271,248	271,248	Berhasil
20.	25_encoded.png	258,602	258,602	Berhasil

4. Pada uji *robustness* yang keempat dilakukan uji pengiriman *encoded-image* dengan aplikasi Gmail seperti yang terlihat pada Tabel 6.15.

Tabel 6.15 Uji *Robustness* dengan Media Gmail

No.	Nama *File*	*Size* Sebelum Pengiriman (*Bytes*)	*Size* Sesudah Pengiriman (*Bytes*)	Status Ekstraksi Pesan
1.	1_encoded.png	379,266	379,266	Berhasil
2.	2_encoded.png	395,997	395,997	Berhasil
3.	4_encoded.png	425,143	425,143	Berhasil
4.	6_encoded.png	310,779	310,779	Berhasil
5.	7_encoded.png	352,748	352,748	Berhasil
6.	8_encoded.png	340,905	340,905	Berhasil
7.	10_encoded.png	414,833	414,833	Berhasil
8.	11_encoded.png	381,745	381,745	Berhasil
9.	12_encoded.png	334,384	334,384	Berhasil
10.	14_encoded.png	204,589	204,589	Berhasil
11.	15_encoded.png	310,033	310,033	Berhasil
12.	16_encoded.png	417,791	417,791	Berhasil
13.	17_encoded.png	490,289	490,289	Berhasil
14.	18_encoded.png	457,675	457,675	Berhasil
15.	19_encoded.png	299,494	299,494	Berhasil
16.	21_encoded.png	301,751	301,751	Berhasil
17.	22_encoded.png	367,692	367,692	Berhasil
18.	23_encoded.png	369,634	369,634	Berhasil
19.	24_encoded.png	271,248	271,248	Berhasil
20.	25_encoded.png	258,602	258,602	Berhasil

5. Pada uji *robustness* yang kelima dilakukan uji pengiriman *encoded-image* dengan aplikasi Facebook seperti yang terlihat pada Tabel 6.16.

Tabel 6.16 Uji *Robustness* dengan Media Facebook

No.	Nama *File*	*Size* Sebelum Pengiriman (*Bytes*)	*Size* Sesudah Pengiriman (*Bytes*)	Status Ekstraksi Pesan
1.	1_encoded.png	379,266	37,306	Gagal
2.	2_encoded.png	395,997	44,104	Gagal
3.	4_encoded.png	425,143	90,037	Gagal
4.	6_encoded.png	310,779	24,973	Gagal
5.	7_encoded.png	352,748	34,982	Gagal
6.	8_encoded.png	340,905	29,374	Gagal
7.	10_encoded.png	414,833	90,457	Gagal
8.	11_encoded.png	381,745	43,905	Gagal
9.	12_encoded.png	334,384	31,042	Gagal
10.	14_encoded.png	204,589	12,368	Gagal
11.	15_encoded.png	310,033	20,199	Gagal
12.	16_encoded.png	417,791	43,259	Gagal
13.	17_encoded.png	490,289	59,174	Gagal
14.	18_encoded.png	457,675	44,686	Gagal
15.	19_encoded.png	299,494	24,552	Gagal
16.	21_encoded.png	301,751	19,538	Gagal
17.	22_encoded.png	367,692	33,206	Gagal
18.	23_encoded.png	369,634	36,618	Gagal
19.	24_encoded.png	271,248	17,544	Gagal
20.	25_encoded.png	258,602	17,805	Gagal

6.1.4 Pengujian MSE dan PSNR

Pengujian *Mean Squared Error* (MSE) dan *Peak Signal to Noise Ratio* (PSNR) adalah parameter pengukuran kualitas suatu citra antara sebelum dan sesudah disisipkan watermark. Pada pengujian MSE dan PSNR dilakukan sebanyak 20 kali data uji pada masing-masing tipe *file* citra. MSE *(Mean Squared Error)* adalah nilai *error* kuadrat rata-rata antara citra asli dengan *encoded-image*. Semakin rendah nilai MSE, maka semakin baik tingkat *error* rata-rata dari suatu citra.

PSNR adalah perbandingan antara nilai maksimum dari sinyal yang diukur dengan besarnya *noise* yang berpengaruh pada sinyal tersebut. Jika semakin besar nilai PSNR, maka semakin baik kualitas suatu citra. Nilai PSNR diukur dalam satuan desibel (dB). Pada penelitian ini, PSNR digunakan untuk mengetahui perbandingan kualitas citra sebelum dan sesudah disisipkan watermark.

6.1.4.1 Normalisasi Perhitungan MSE dan PSNR

Berdasarkan normalisasi pada perhitungan MSE dan PSNR dibuatlah *source code* untuk pengujian MSE dan PSNR pada aplikasi MATLAB seperti yang ditunjukkan pada Gambar 6.3.

```
1.  function A = psnr(encoded,original)
2.
3.  encoded=(encoded);
4.  original=(original);
5.
6.      MSE=mean(mean((encoded-original).^2));
7.      MSErata=mean(MSE(:,:));
8.      disp(MSErata);
9.
10.      A=10*log10(255^2/MSErata);
11.
12.  end
```

Gambar 6.3 *Source Code* dari psnr

Penjelasan perhitungan nilai MSE dan PSNR pada Gambar 6.3, yaitu:

1. Baris 1 adalah deklarasi *function* A untuk perhitungan PSNR dengan *input encoded-image* dan *file* citra asli.
2. Baris 3-4 adalah konversi *encoded-image* dan *file* citra asli ke dalam bilangan *double.*
3. Baris 6 adalah perhitungan MSE dari tiap-tiap komponen warna RGB
4. Baris 7 adalah perhitungan MSE total
5. Baris 8 adalah menampilkan nilai MSE
6. Baris 10 adalah perhitungan PSNR dengan rumus *10*log10(255^2/MSE*

Pengujian PSNR dilakukan sebanyak 20 kali percobaan pada *file* (*.png*), (*.gif*), (*.jpeg*), dan (*.jpg*). Untuk mendapatkan nilai dari PSNR, mula-mula dihitung nilai MSE terlebih dahulu. Pada Tabel 6.17 akan ditunjukkan hasil dari perhitungan nilai MSE dan PSNR pada *file input* (*.png*), (*.gif*), (*.jpeg*), dan (*.jpg*).

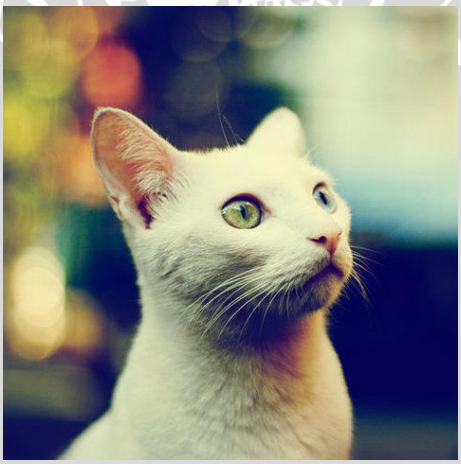
Tabel 6.17 Perhitungan Nilai MSE dan PSNR

No.	Nama *File*	Resolusi (*pixels*)	*Bit Depth (bit)*	Nilai MSE	Nilai PSNR (dB)
1.	1.jpeg	500x500	8	Error	Error
	1_encoded.png	500x500	24		
2.	2.png	500x500	8	Error	Error
	2_encoded.png	500x500	24		
3.	4.jpg	500x500	8	Error	Error
	4_encoded.png	500x500	24		
4.	6.jpg	480x480	24	2.6367	43.9203
	6_encoded.png	480x480	24		
5.	7.jpg	480x480	8	Error	Error
	7_encoded.png	480x480	24		
6.	8.jpg	500x500	24	2.5665	44.0373
	8_encoded.png	500x500	24		
7.	10.gif	500x500	8	Error	Error
	10_encoded.png	500x500	24		
8.	11.png	500x500	32	2.3646	44.3932
	11_encoded.png	500x500	24		
9.	12.jpg	500x500	24	3.5494	42.6292
	12_encoded.png	500x500	24		
10.	14.gif	480x480	8	Error	Error
	14_encoded.png	480x480	24		
11.	15.jpg	500x500	24	2.3799	44.3652
	15_encoded.png	500x500	24		
12.	16.jpg	500x500	24	4.1598	41.9400
	16_encoded.png	500x500	24		
13.	17.jpg	500x500	24	2.4476	44.2433
	17_encoded.png	500x500	24		
14.	18.jpg	500x500	24	2.5019	44.1482
	18_encoded.png	500x500	24		

15.	19.jpg	500x500	24	2.4117	44.3075
	19_encoded.png	500x500	24		
16.	21.jpg	500x500	24	2.3498	44.4206
	21_encoded.png	500x500	24		
17.	22.jpg	500x500	24	6.2291	40.1866
	22_encoded.png	500x500	24		
18.	23.jpg	500x500	24	2.3894	44.3480
	23_encoded.png	500x500	24		
19.	24.jpg	500x500	24	2.3986	44.3312
	24_encoded.png	500x500	24		
20.	25.jpg	480x480	24	2.6907	43.8322
	25_encoded.png	480x480	24		

6.1.5 Pengujian Ukuran File

Pengujian ukuran *file* bertujuan untuk mengetahui perbandingan nilai MSE dan PSNR serta ukuran citra setelah disisipkan *digital signature* dengan ukuran yang berbeda. *File* citra yang digunakan pada pengujian ukuran pesan adalah **6.jpg** dengan resolusi 500x500 *pixels* seperti yang ditunjukkan pada Gambar 6.4.



Gambar 6.4 *File* Citra yang Digunakan

Sedangkan File citra yang digunakan sebagai digital signature adalah **Picture1.png** dengan resolusi 234x244 pixels seperti yang ditunjukkan pada Gambar 6.5.

Gambar 6.5 *File* Citra yang Digunakan Sebagai *Digital Signature*

Pengujian dilakukan dengan cara menyisipkan *file* citra *digital signature* yang memiliki ukuran resolusi yang berbeda ke dalam *file* **6.jpg**, kemudian dilakukan pengamatan terhadap ukuran, MSE, dan PSNR dari *file* citra setelah penyisipan. Pada Tabel 6.18 ditunjukkan *file* citra *digital signature* dengan ukuran yang berbeda sebagai data uji.

Tabel 6.18 *File* Citra *Digital Signature* yang Digunakan sebagai Data Uji

No.	Nama	Resolusi (*pixels*)	*Bit depth*	*Size* (*Bytes*)
1.	8.png	8x8	32	369
2.	16.png	16x16	32	858
3.	24.png	24x24	32	1,557
4.	32.png	32x32	32	2,437
5.	40.png	40x40	32	3,463
6.	48.png	48x48	32	4,676
7.	56.png	56x56	32	6,023
8.	64.png	64x64	32	7,633
9.	72.png	72x72	32	9,284
10.	Picture1.png	234x244	32	87,012

Berdasarkan data uji pada Tabel 6.18 dilakukan pengujian dengan cara menyisipkan *digital signature* ke dalam **6.jpg**. Pengamatan dilakukan dengan cara menghitung nilai MSE dan PSNR dari *file* serta perubahan ukuran *file* citra sesudah melalui penyisipan. Hasil dari pengujian ukuran *file* ditampilkan pada Tabel 6.19.

Tabel 6.19 Hasil Pengujian Ukuran *File*

No.	*File* Citra *Digital Signature* yang digunakan	Perubahan ukuran *file* citra pada **6.jpg** (*Bytes*)			Nilai MSE	Nilai PSNR (dB)
		Sebelum	Sesudah	Selisih		
1.	8.png	42,039	258,361	216,322	0.0978	58.2277
2.	16.png	42,039	258,851	216,812	0.1041	57.9561
3.	24.png	42,039	259,489	217,450	0.1137	57.5717
4.	32.png	42,039	260,069	218,030	0.1313	56.9494
5.	40.png	42,039	260,721	218,682	0.1480	56.4289
6.	48.png	42,039	262,737	220,698	0.1748	55.7051
7.	56.png	42,039	263,489	221,450	0.2084	54.9424
8.	64.png	42,039	265,761	223,722	0.2429	54.2763
9.	72.png	42,039	266,997	224,958	0.2808	53.6468
10.	Picture1.png	42,039	310,779	268,740	2.6367	43.9203

6.2 Analisis

Proses analisis bertujuan untuk mendapatkan kesimpulan dari pengujian aplikasi *Image Watermarking.* Analisis dilakukan berdasarkan dari hasil pengujian aplikasi. Pada proses analisis dilakukan sebanyak empat tahapan, yaitu analisis validitas, analisis kompatibilitas, analisis performansi, analisis MSE & PSNR, dan analisis perubahan ukuran *file*.

6.2.1 Analisis Pengujian Validitas

Proses analisis terhadap hasil pengujian validitas dilakukan dengan melihat kesesuaian antara hasil kinerja sistem dengan daftar kebutuhan fungsional. Berdasarkan hasil pengujian validitas dari Tabel 6.6 dapat disimpulkan bahwa aplikasi *Image Watermarking* telah memenuhi kebutuhan yang telah dijabarkan pada tahap analisis kebutuhan. Analisis kebutuhan yang dimaksud adalah dapat menjalankan semua fitur aplikasi, yaitu pemilihan *file* citra *digital signature,* pemilihan *file* citra yang akan di watermark, pembuatan watermark, dan ekstraksi *digital signature*.

6.2.2 Analisis Pengujian Kompatibilitas

Proses analisis validitas terhadap hasil pengujian kompatibilitas dilakukan pada aplikasi *Image Watermarking* terhadap sistem operasi Android dengan versi yang berbeda. Sistem operasi yang digunakan sebagai implementasi pengujian kompatibilitas adalah Android versi 4.2.2 Jelly Bean, Android versi 4.4.4 Kitkat, dan

Android versi 6.0 Marshmallow. Pengujian kompatibilitas dilakukan dengan melihat kesesuaian antara hasil kinerja sistem dengan daftar kebutuhan.

Berdasarkan hasil pengujian kompatibilitas pada Tabel 6.7, Tabel 6.8, dan Tabel 6.9 dapat disimpulkan bahwa implementasi dan fungsional aplikasi *Image Watermarking* telah memenuhi kebutuhan yang dijabarkan pada tahap analisis kebutuhan. Analisis kebutuhan yang dimaksud adalah aplikasi dapat menjalankan semua kebutuhan fungsional dengan baik pada tiga sistem operasi Android yang berbeda.

6.2.3 Analisis Pengujian Performansi

Analisis pengujian performansi adalah suatu parameter pengukuran untuk mengetahui kinerja dari aplikasi yang dibuat. Parameter pengukuran yang dimaksud meliputi perbandingan kasat mata, ukuran *file* citra, dan *robustness*. Proses analisis performansi terhadap aplikasi *Image Watermarking* dilakukan melalui tiga tahap analisis, yaitu analisis perbandingan kasat mata dari *file* citra, analisis perbandingan ukuran *file* citra, dan analisis *robustness* dari *file* citra.

6.2.3.1 Analisis Perbandingan Citra

Pada proses analisis perbandingan citra dilakukan pengamatan fisik terhadap *file* citra saat sebelum dan sesudah penyisipan. Pengamatan bertujuan untuk melihat apakah ada perbedaan yang signifikan antara *file* citra asli dan *encoded-image* sebagai data uji. Data uji yang dipakai pada proses analisis adalah 10 *file input* dengan format (*.png*), (*gif*), (*.jpeg*), dan (*.jpg*) beserta *output*. Pengamatan dilakukan oleh 10 orang responden yang tidak mengalami gangguan penyakit buta warna. Hasil pengamatan responden terhadap data uji ditunjukkan pada Tabel 6.20.

Tabel 6.20 Analisis Perbandingan Citra Oleh Responden

No.	Nama Responden	Perbedaan pada *file* citra									
		1	2	3	4	5	6	7	8	9	10
1.	Erick Kurniawan	T	T	T	T	T	T	T	T	T	T
2.	Steady Kambodji	T	T	T	T	T	T	T	T	T	T
3.	M. Sulton	T	T	T	T	T	T	T	T	T	T
4.	Adi Anggono	T	T	T	T	T	T	T	T	T	T
5.	Nia Adina	T	T	T	T	T	T	T	T	T	T
6.	Bayu W. Harimurti	T	T	T	T	T	T	T	T	T	T
7.	David Licindo	T	T	T	T	T	T	T	T	T	T
8.	Afan Amarullah	T	T	T	T	T	T	T	T	T	T
9.	M. Taufiqur	T	T	T	T	T	T	T	T	T	T

10.	Evan Andriawan	T	T	T	T	T	T	T	T	T	T

Berdasarkan pengamatan dari Tabel 6.20 dapat disimpulkan bahwa pada pengamatan secara kasat mata *file input* (*.gif*), (*.png*), (*.jpeg*), dan (*jpg*) tidak terlihat perbedaan warna maupun bentuk gambar antara *file* citra asli dan *encoded-image*. Tidak terlihatnya perbedaan secara kasat mata menunjukkan bahwa penyisipan *digital signature* pada *file* citra dengan aplikasi *Image Watermarking* tidak dapat dideteksi oleh indera manusia.

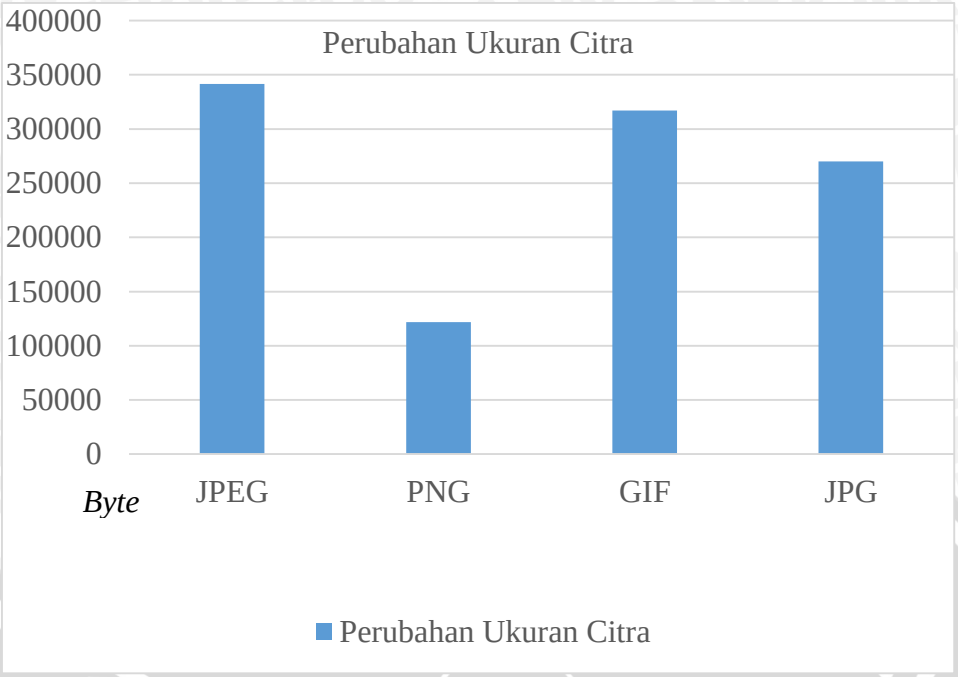
6.2.3.2 Analisis Perbandingan Ukuran *File* Citra

Pada proses perbandingan ukuran *file* citra dilakukan pengamatan terhadap resolusi, kedalaman *bit*, dan *size* dari *file* citra sebelum dan sesudah melalui penyisipan pesan. Pengujian dilakukan dengan 20 buah data uji yang terdiri dari *file input* dengan format (*.png*), (*gif*), (*.jpeg*), dan (*.jpg*) beserta *output*. Berdasarkan pengujian ukuran *file* citra pada subbab 6.1.3.2 tidak terjadi perubahan resolusi dari *file* citra, namun terjadi perubahan kedalaman *bit* dari 8 sampai 32 *bit* menjadi 24 *bit* setelah melalui proses watermark. Perubahan kedalaman *bit* disebabkan pada saat proses rekonstruksi *encoded-image* dibangun dari 8 *bit Red*, 8 *bit Green*, dan 8 *bit Blue*.

Pada pengujian *size* dari *file* citra akan dianalisis berdasarkan ekstensi format dari *file input*. Dari dua puluh data uji pada *file* citra terjadi perubahan *size* menjadi lebih besar, yaitu berkisar antara 204,589 *bytes* sampai dengan 490,289 *bytes* setelah melalui proses penyisipan pesan. Penambahan *size* paling kecil terjadi pada *file* "*11.png*" yang mula-mula sebesar 378,681 *bytes* menjadi 381,745 *bytes*. Penambahan *size* tersebut disebabkan oleh kompresi secara *lossless* (tidak menghilangkan data) dari *file input* (*.png*) menjadi (*.png*). Rata-rata perubahan ukuran pada *file* citra adalah sebesar 263,428 *bytes.*

Tabel 6.21 Perubahan ukuran *file* citra sesudah proses watermark

File input:	Perubahan ukuran rata-rata
.jpeg	**341,437** *bytes*
.png	**121,829** *bytes*
.gif	**317,166** *bytes*
.jpg	**269,943** *bytes*



Gambar 6.6 Perubahan Ukuran *File* Citra

Berdasarkan analisis data uji pada Tabel 6.21 dan Gambar 6.6 dapat disimpulkan bahwa aplikasi bekerja dengan performa terbaik pada *input file* berformat (*.png*) yang tidak terjadi perubahan ukuran secara signifikan. Jika dibandingkan *file input* dengan format (*.jpeg*), (*.gif*) dan (*.jpg*), nilai perubahan ukuran rata-rata pada *file* (*.png*) terbilang paling rendah, yaitu sebesar 121,829 *bytes*. Hal ini terjadi karena *file* dengan format *(.png)* merupakan *file* berjenis *lossless data compression* dan merupakan pengembangan dari file dengan format *(.gif)* sehingga ketika *file input* ini diubah menjadi *file* dengan format *(.png)* tidak terjadi perubahan ukuran yang signifikan karena memiliki jenis format *file* yang sama, selain itu *file* dengan format *(.jpeg)* dan *(.jpg)* yang merupakan *file* berjenis *lossy data compression* akan mengalami perubahan ukuran yang cukup besar karena *file* akan mengalami perubahan kompresi data.

6.2.3.3 Analisis *Robustness File* Citra

Berdasarkan hasil pengujian *robustness* pada subbab 6.1.3.3, dilakukan pengiriman 20 data uji dari *encoded-image* untuk diekstraksi oleh penerima. Aplikasi yang digunakan sebagai media pengiriman *encoded-image* adalah Blackberry *Messenger*, Gmail, WhatsApp *Messenger*, Line *Messenger,* dan Facebook. Pada pengiriman melalui media Blackberry *Messenger* dan Gmail *encoded-image* berhasil diekstraksi. Sedangkan pada pengiriman melalui WhatsApp *Messenger*, Line *Messenger,* dan Facebook pesan gagal diekstraksi disebabkan adanya manipulasi citra serta proses kompresi secara *lossy* (menghilangkan beberapa informasi data). Dengan kata lain, aplikasi *Image Watermarking* tidak tahan terhadap manipulasi citra.

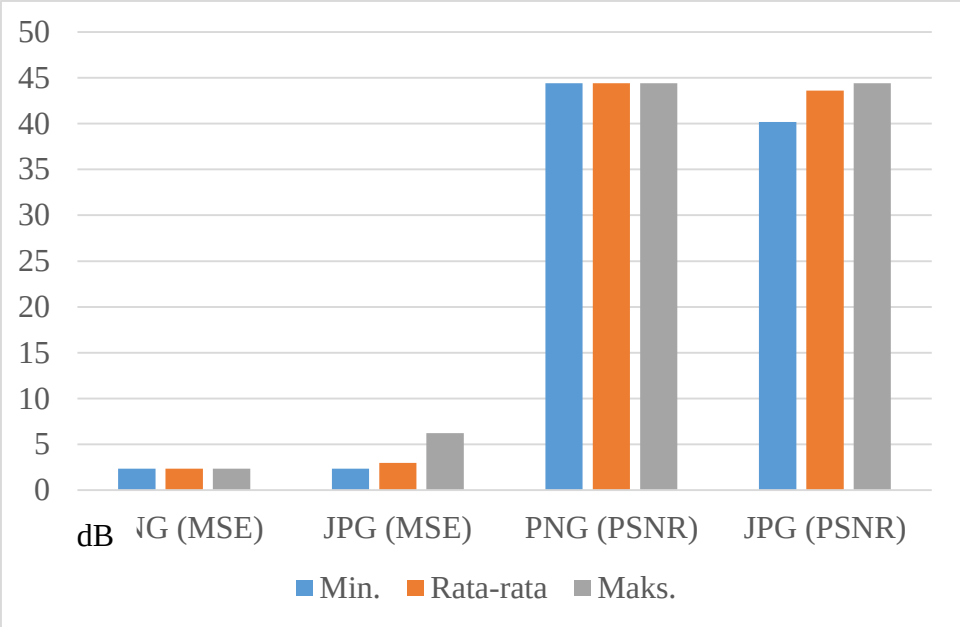
6.2.4 Analisis MSE dan PSNR

Pada analisis dari pengujian MSE dan PSNR pada subbab 6.1.4 diperoleh empat poin utama, antara lain:

1. Dari dua puluh data uji diperoleh nilai MSE tertinggi sebesar 6.2291 dan nilai MSE terendah sebesar 2.3498. Pada pengujian PSNR diperoleh nilai tertinggi 44.4206 dB dan PSNR terendah 40.1866 dB. Semakin rendah nilai MSE maka semakin baik kualitas suatu *file* citra, sebaliknya semakin tinggi nilai PSNR maka semakin baik kualitas suatu *file* citra.
2. Dari dua puluh data uji terdapat nilai error pada perhitungan PSNR dan MSE karena file citra memiliki perbedaan kedalaman bit sehingga tidak dapat dicari matriks perhitungannya. Untuk melakukan pengujian MSE & PSNR minimal suatu *file* citra harus memiliki kedalaman *bit* RGB yang sama pada *file input* dan *output*-nya.
3. Rata-rata nilai MSE pada *file input* dengan format (*.jpg*) adalah 2.9778 sedangkan nilai PSNR rata-rata pada *file input* dengan format (*.jpg*) adalah 43.5931 dB.
4. Rata-rata nilai MSE pada *file input* dengan format (*.png*) adalah 2.3646 sedangkan nilai PSNR rata-rata pada *file input* dengan format (*.png*) adalah 44.3932 dB.

Tabel 6.22 Analisis Nilai MSE dan PSNR

File input:	Nilai MSE		Nilai PSNR	
	Min.	Maks.	Min.	Maks.
.png	**2.3646**	**2.3646**	**44.3932**	**44.3932**
.jpg	**2.3498**	**6.2291**	**40.1866**	**44.4206**



Gambar 6.7 Grafik Hasil Pengujian MSE dan PSNR

Berdasarkan analisis data pada Tabel 6.22 dan Gambar 6.7 dapat disimpulkan bahwa *file input* (*.jpg*) menghasilkan rata-rata nilai MSE paling tinggi yaitu sebesar 2.9778 jika dibandingkan *file* (*.png*). Pada analisis PSNR, nilai PSNR dari *file input* (*.jpg*) lebih rendah jika dibandingkan dengan *file (.png*). Hal ini ditunjukkan dari nilai rata-rata PSNR pada *file* (*.jpg*) yang hanya mencapai 43.5931 dB, sedangkan nilai rata-rata PSNR maksimum yang dihasilkan *file input (.png*) mencapai 44.3932 dB.

6.2.5 Analisis Perubahan Ukuran *File*

Berdasarkan hasil pengujian ukuran *file* dengan ukuran *digital signature* yang berbeda pada subbab 6.1.5 dapat disimpulkan bahwa perubahan ukuran *file* tidak mempengaruhi ukuran *file* citra secara signifikan, namun mempengaruhi nilai MSE dan PSNR. Nilai PSNR berbanding terbalik dengan nilai dari MSE. Semakin besar ukuran *file* citra *digital signature* membuat nilai MSE semakin tinggi, sehingga mengakibatkan nilai PSNR menjadi semakin rendah. Nilai PSNR yang dihasilkan memiliki korelasi dengan kualitas *file* citra, yaitu semakin tinggi nilai PSNR maka semakin baik kualitas dari *file* citra.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian dari aplikasi yang dibuat, maka diambil kesimpulan sebagai berikut:

1. Aplikasi *Image Watermarking* telah dibuat sesuai perancangan dan dapat digunakan untuk menyembunyikan *digital signature* ke dalam *file* citra.
2. Aplikasi *Image Watermarking* berhasil diimplementasikan pada sistem operasi Android versi 4.2.2 (Jelly Bean), 4.4.4 (KitKat), dan 6.0 (Marshmallow).
3. Kualitas citra setelah melalui proses watermark tergolong baik, karena perbandingan citra antara sebelum dan sesudah penyisipan pesan tidak dapat terdeteksi oleh indera manusia.
4. Nilai rata-rata MSE yang dihasilkan *file input* (*.jpg*) lebih tinggi jika dibandingkan dengan *file* (*.png*) yaitu sebesar 2.9778. Pada perhitungan rata-rata PSNR yang dihasilkan oleh *file input* (*.png*) menunjukkan nilai yang lebih tinggi yaitu mencapai 44.3932 dB jika dibandingkan rata-rata *file input* (*.jpg*) yang hanya mencapai 43.5931 dB.

7.2 Saran

Aplikasi *Image Watermarking* masih mempunyai banyak kekurangan, sehingga perlu dikembangkan lagi agar menjadi lebih sempurna. Untuk meningkatkan kualitas dan fungsionalitas dari aplikasi, saran yang diberikan untuk pengembangan penelitian selanjutnya antara lain:

1. Media penampung tidak hanya berupa *file citra* saja, tetapi menggunakan media digital lain seperti *file* audio dan video. Aplikasi dapat beradaptasi dan menyesuaikan dengan format-format baru yang ada.
2. Watermark yang disisipkan tidak hanya berupa *file* citra saja, tetapi menggunakan *file* lain seperti penyisipan teks dan *file* khusus. Aplikasi dapat menyisipkan beberapa macam *file* sehingga sulit ditebak *file* apa yang disembunyikan.
3. Dibutuhkan metode steganografi lain yang tahan terhadap manipulasi citra. Dalam penggunaan aplikasi perangkat bergerak, sering kali terjadi kompresi atau perubahan format kepada data karena ada keterbatasan ruang penyimpanan dan bandwidth. Hal ini menyebabkan *file* citra rentan terhadap manipulasi baik saat *user* mengunggah maupun mengunduh *file* citra.

DAFTAR PUSTAKA

[ASR-11] A. S. Rosa dan Shalahudin, M. 2011. Rekayasa Perangkat Lunak (Terstruktur dan Berorientasi Objek). Modula, Bandung.

[BON-03] Boncelet, Jr. 2003. Spread Spectrum Image Steganography. url: http://cryptome.org/usa-patent.htm, Department of Defense, USA.

[CHO-12] Chopra, Deepshikha et al. 2012. Lsb Based Digital Image Watermarking For Gray Scale Image. IOSR Journal of Computer Engineering (IOSRJCE).

[CIS-15] Cisco. 2015. Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2014–2019. Cisco, USA.

[DEV-13] Devi, Kshetrimayum Jenita. 2013. A Sesure Image Steganography Using LSB Technique and Pseudo Random Encoding Technique. Department of Computer Science and Engineering National Institute of Technology Rourkela, India.

[DEV-15] Developer, Android. Built for Multiscreen World. url: http://developer.android.com/index.html, diakses 30 Mei 2015.

[FEL-12] Felker, Donn dan Burton M. 2012. Android Application Development for Dummies. John Wiley & Sons Inc., New Jersey.

[HAK-13] Hakim, Muhammad. 2013. Studi dan Implementasi Steganografi Metode LSB dengan Preprocessing Kompresi Data dan Ekspansi Wadah. ITB, Bandung.

[IND-11] Indrajani. 2011. Perancangan Basis Data All in 1. Elex Media Komputindo, Bogor, Jawa Barat.

[KAD-11] Kadir, Abdul. 2011. Pengenalan Teknologi Informasi dan Komunikasi untuk Remaja. Andi Publisher, Indonesia.

[MAR-12] Maradilla, Temmy. 2012. Aplikasi Steganografi untuk Penyisipan Data Teks ke dalam Citra Digital. Universitas Gunadarma, Depok.

[MUN-04] Munir, Rinaldi. 2004. Steganografi dan Watermarking. Departemen Teknologi Informatika Institut Teknologi Bandung.

[MUN-06] Munir, Rinaldi. 2006. Kriptografi. Pener*bit*: Informatika, Bandung.

[NAK-14] Nakamura, Masumi dan Gargenta Marko. 2014. Learning Android: Develop Mobile Apps Using Java And Eclipse. O'Reilly Media, Inc., USA.

[NAT-14] Native, Web and Hybrid Apps – Understanding The Difference. url: http://www.xcubelabs.com/blog/native-web-and-hybrid-apps-understanding-the-difference/.

[NUG-10] Nugraha, Erdiansyah Fajar. 2010. Meningkatkan Kapasitas Pesan yang Disisipkan dengan Metode Redundant Pattern Coding. ITB, Bandung.

[PAD-13] Padmasari B, dan Surabi M. 2013. *Spread Spectrum* Image Steganography with Encryption Key Implementation. International Journal of Advanced Search in Computer Science and Software Engineering vol 3. PRIST University, India.

[PAK-10] Pakereng, M.A. Ineke, Yos Richard Beeh, and Sonny Endrawan. 2010. Perbandingan Steganografi Metode Spread Spectrum dan Least Significant Bit (LSB) Antara Waktu Proses dan Ukuran File Gambar. JURNAL INFORMATIKA, VOLUME 6 NOMOR 1, APRIL 2010.

[PRA-12] Pratiarso, Aries. 2012. Analisa PSNR pada Teknik Steganografi Menggunakan *Spread Spectrum*. Electronic Engineering Polytechnic Institute of Surabaya, Surabaya.

[PUT-10] Putra, Darma. 2010. Pengolahan Citra Digital. C.V ANDI OFFSET, Yogyakarta.

[SAE-10] Saelan, Athia. 2010. Analisis Beberapa Teknik Watermarking dengan Domain Spasial pada Citra Digital. Makalah IF3058 Kriptografi – Sem. II Tahun 2010/2011, Bandung

[SHA-12] Sharma, Yuvraj dan Kaushik, Pooja. 2012. Comparison Of Different Image Enhancement Techniques Based Upon PSNR & MSE. International Journal of Applied Engineering Research, ISSN 0973-4562 Vol.7 No.11, India.

[SHU-12] Shukla, Ruby, Manish, and Prof. A.K. Arora. Analysis of Image Watermarking: LSB Modification and Spread-Spectrum Technique. IOSR Journal of Electronics and Communication Engineering (IOSRJECE).

[SUB-13] Subiyantoro, Eko. 2013. Arsitektur Sistem Operasi Android. url: http://www.vedcmalang.com/pppptkboemlg/index.php/menuutama/teknologi-informasi/825-arsitektur-sistem-operasi-android

[SUK-02] Sukmawan, Budi. 2002. Steganografi.

url: http://www.bdg.centrin.net.id/~budskman/ stegano.htm