

UNIVERSITAS GUNADARMA
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI



**PENERAPAN ALGORITMA BERNOULLI MAP DALAM
PROGRAM APLIKASI ENKRIPSI CITRA DIGITAL**

Disusun Oleh :

Nama : Nadya Sofia Laura

NPM : 15112221

Jurusan : Sistem Informasi

Pembimbing : 1. Dr. Edi Sukirman, SSi., MM.

2. Dr. Drs. Suryadi M. Thoyib, M.T.

Diajukan Guna Melengkapi Sebagai Syarat
Dalam Mencapai Gelar Sarjana Strata Satu (S1)

JAKARTA
2016

PERNYATAAN ORIGINALITAS DAN PUBLIKASI

Saya yang bertanda tangan di bawah ini,

Nama : Nadya Sofia Laura

NPM : 15112221

Judul Skripsi : Penerapan Algoritma Bernoulli Map dalam Program Aplikasi Enkripsi Citra Digital

Tanggal Sidang :

Tanggal Lulus :

Menyatakan bahwa tulisan ini adalah merupakan hasil karya sendiri dan dapat dipublikasikan sepenuhnya oleh Universitas Gunadarma. Segala kutipan dalam bentuk apa pun telah mengikuti kaidah, etika yang berlaku. Mengenai isi dan tulisan adalah merupakan tanggung jawab penulis, bukan Universitas Gunadarma.

Demikian pernyataan ini dibuat dengan sebenarnya dan dengan penuh kesadaran.

Jakarta, Agustus 2016

(Nadya Sofia Laura)

LEMBAR PENGESAHAN

KOMISI PEMBIMBING

NO	NAMA	KEDUDUKAN
1.		
2.		
3.		

Tanggal Sidang :

PANITIA UJIAN

NO	NAMA	KEDUDUKAN
1.		
2.		
3.		
4.		
5.		

Tanggal Lulus :

Mengetahui,

Pembimbing I

Pembimbing II

(Dr. Edi Sukirman, SSi., MM.)

(Dr. Drs. Suryadi M Thoyib, M. T.)

Bagian Sidang Ujian

(Dr. Edi Sukirman, SSi., MM.)

ABSTRAKSI

Nadya Sofia Laura. 15112221.

PENERAPAN ALGORITMA BERNOULLI MAP DALAM PROGRAM APLIKASI ENKRIPSI CITRA DIGITAL.

Skripsi. Jurusan Sistem Informasi. Fakultas Ilmu Komputer dan Teknologi Informasi. Universitas Gunadarma. 2016.

Kata kunci : Algoritma Enkripsi, Chaos, Citra Bitmap, Citra PNG, Bernoulli

(xiii + 88 + 67 Lampiran)

Informasi merupakan suatu data yang sudah diolah yang memiliki makna. Informasi tidak hanya berupa data teks tetapi dapat juga berupa citra yang sifatnya rahasia. Hal ini menyebabkan keamanan pada informasi merupakan aspek yang sangat penting untuk diperhatikan. Salah satu cara mengamankan dan melindungi informasi yaitu dengan melakukan teknik enkripsi dan dekripsi citra digital. Penelitian terus dilakukan untuk meningkatkan daya tahan algoritma yang digunakan pada proses enkripsi dari serangan *brute force*, salah satunya dengan cara mengimplementasikan teori *chaos*. Algoritma Bernoulli Map merupakan salah satu algoritma yang mengimplementasikan teori *chaos*. Algoritma ini mengimplementasikan teori *chaos* dengan membangkitkan deret bilangan yang bersifat acak dengan dua nilai awal. Hasil ujicoba dari penelitian ini menunjukkan bahwa algoritma ini dapat mengenkripsi sejumlah citra, baik citra *grayscale* maupun berwarna yang memiliki ekstensi .bmp dan .png. Lamanya waktu proses enkripsi dan dekripsi citra berbanding lurus dengan besarnya ukuran piksel citra. Histogram citra terenkripsi memiliki penyebaran yang merata sehingga tahan terhadap penyerang yang akan melakukan analisis frekuensi. Algoritma ini dapat memberikan keamanan yang baik dari serangan *brute force*, karena memiliki tingkat sensitivitas 10^{-13} pada kunci r dan 10^{-14} pada kunci X_n .

Daftar Pustaka (2006-2014)

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Tuhan Yesus Kristus atas segala kasih karunia-Nya, akhirnya penulis dapat menyelesaikan penyusunan skripsi yang berjudul “Aplikasi Enkripsi Citra Digital Berbasis Chaos dengan Algoritma Bernoulli Map” dengan baik. Adapun tujuan dari penulisan ini adalah untuk melengkapi sebagai syarat untuk mencapai gelar Sarjana Strata Satu (S1) jurusan Sistem Informasi, Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Gunadarma.

Penulis menyadari bahwa penulisan ini masih jauh dari kata sempurna baik dalam penulisan maupun pengembangan aplikasi, karena keterbatasan kemampuan penulis.

Dalam kesempatan ini, penulis ingin mengucapkan rasa terima kasih kepada semua pihak yang telah memberikan motivasi serta bantuan yang sangat berharga. Ucapan terima kasih penulis sampaikan kepada :

1. Ibu Prof. Dr. E. S. Margianti, SE., MM. selaku Rektor Universitas Gunadarma.
2. Bapak Prof. Dr. Ir. Bambang Suryawan, MT. selaku Dekan Fakultas Teknologi Industri, Universitas Gunadarma.
3. Bapak Dr. Ing. Adang Suhendra, SSi, SKom, MSc. selaku ketua Jurusan Teknik Informatika.
4. Bapak Dr. Edi Sukirman, SSi., MM., dan Bapak Dr. Drs. Suryadi M Thoyib, M.T, selaku Dosen Pembimbing yang telah banyak memberikan bimbingan, arahan, dukungan dan waktunya kepada penulis selama proses pembuatan penulisan skripsi ini berlangsung hingga selesai.
5. Keluarga tercinta, terutama orang tua yang telah membekalkan, mendidik dan membimbing dengan penuh kasih sayang, serta memberikan dukungan dan doanya hingga penulis dapat menyelesaikan penulisan skripsi ini.
6. Teman-teman angkatan 2012 terutama kelas 4KA23 yang tidak dapat disebutkan satu persatu dan secara tidak langsung ikut terlibat di dalam penulisan skripsi ini yang selalu memberi dukungan kepada penulis.
7. Ishak Klarissa selaku sahabat baik yang selalu memberikan semangat, menemani, membantu hingga selesainya penulisan ini.

Semua pihak yang telah membantu memberikan motivasi, ide atau apapun yang terlibat dalam pembuatan penulisan skripsi ini, penulis ucapan terima kasih yang sebesar-besarnya. Semoga Tuhan Yang Maha Esa yang akan membalas kebaikan semua nya.

Penulis merasa masih banyak kekurangan baik pada teknis penulisan maupun materi. Untuk itu kritik dan saran sangat penulis harapkan agar penulis dapat berkembang di kemudian hari. Akhirnya penulis berharap penulisan skripsi ini dapat memberikan manfaat bagi kita semua.

Jakarta, Agustus 2016

Penulis

DAFTAR ISI

BAB I	3
PENDAHULUAN	3
1.1 Latar Belakang Masalah.....	3
1.2 Rumusan Masalah	5
1.3 Tujuan Penulisan.....	5
1.4 Batasan Masalah.....	5
1.5 Metode Penelitian	6
1.6 Sistematika Penulisan	6
BAB II.....	8
LANDASAN TEORI.....	8
2.1 Konsep Kriptografi.....	8
2.2 Fungsi Chaos Bernoulli Map	12
2.3 Bernoulli map Fungsi yang tidak invertible	14
2.4 Citra Digital.....	15
2.4.1 Citra <i>Digital</i> BerformatBitmap.....	16
2.4.2 Citra <i>Digital</i> Berformat GIF	17
2.4.3 Citra <i>Digital</i> Berformat PNG.....	18
2.4.4 Perbedaan Citra Bitmap, GIF statik, dan PNG	18
2.5 Matrix Laboratory (MATLAB).	19
2.5.1 Lingkungan Kerja Matlab	20
2.5.2 GUI Designer pada Matlab (GUIDE)	20
2.5.3 Perintah pada Matlab untuk pengelolaan citra	26
2.6 Operasi XOR	26
2.7 Struktur Navigasi	27
2.8 Flowchart.....	29
BAB III	32
ANALISIS DAN PERANCANGAN.....	32
3.1 Analisis Masalah.....	32
3.2 Algoritma Enkripsi dan Dekripsi	33

3.2.1	Tahap Pembangkitan <i>Keystream</i>	33
3.2.2	Tahap Enkripsi	34
3.2.3	Tahap Dekripsi	34
3.3	Flowchart Proses Enkripsi dan Dekripsi	35
3.4	Struktur Navigasi Aplikasi Enkripsi	36
3.5	Rancangan Tampilan Aplikasi Enkripsi dan Dekripsi Citra	38
3.5.1	Rancangan Tampilan Menu Utama	38
3.5.2	Rancangan Tampilan Halaman Enkripsi Citra	39
3.5.3	Rancangan Tampilan Halaman Dekripsi Citra	41
3.5.4	Rancangan Tampilan Halaman Submenu Histogram	42
3.5.4.1	Rancangan Tampilan Halaman Histogram Citra Grayscale.....	43
3.5.4.2	Rancangan Tampilan Halaman Histogram Citra RGB.....	44
3.5.5	Rancangan Tampilan Halaman Submenu <i>Compare</i> atau Perbandingan	46
3.5.5.1	Rancangan Tampilan Halaman Perbandingan Citra Grayscale ...	47
3.5.5.2	Rancangan Tampilan Halaman Perbandingan Citra RGB	49
3.5.6	Rancangan Tampilan Halaman PSNR	51
3.5.6.1	Rancangan Tampilan Halaman PSNR Citra Grayscale	52
3.5.6.2	Rancangan Tampilan Halaman PSNR Citra RGB	54
3.5.7	Rancangan Tampilan Halaman Bantuan.....	56
3.5.8	Rancangan Tampilan Halaman Tentang	56
3.6	Pembuatan Aplikasi.....	57
3.6.1	Pembuatan Tampilan Aplikasi	57
3.6.2	Pembuatan Menu Aplikasi.....	58
3.6.3	Penulisan Kode Sumber Aplikasi.....	59
BAB IV.....		60
ANALISIS HASIL UJI COBA.....		60
4.1	Implementasi Aplikasi	60
4.2	Analisis Hasil Uji Coba	69
4.2.1	Data Citra Uji Coba	69
4.2.2	Enkripsi Citra	70

4.2.3	Analisis Waktu Enkripsi dan Dekripsi	72
BAB V		84
PENUTUP		84
5.1	Kesimpulan.....	84
5.2	Saran.....	84

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Perkembangan teknologi yang begitu pesat, memungkinkan manusia dapat berkomunikasi dan saling bertukar data atau informasi dengan jarak jauh, baik antar kota, antar wilayah, antar negara bahkan antar benua. Hal ini bukan merupakan suatu kendala lagi dalam melakukan komunikasi dan pertukaran data atau informasi. Pesatnya perkembangan teknologi menuntut sekuritas (keamanan) terhadap kerahasiaan data atau informasi yang dikirimkan sehingga menjadi hal yang sangat penting untuk diperhatikan. Banyak pengguna (*user*) seperti departemen pertahanan, suatu perusahaan, atau bahkan individu-individu yang tidak ingin data atau informasi yang disampaikannya dicuri oleh orang lain. Pengiriman dan penyimpanan data atau informasi tanpa dilakukan pengamanan akan beresiko terhadap penyadapan dan pencurian. Dengan demikian data atau informasi yang ada di dalamnya dapat mudah diketahui oleh pihak-pihak yang tidak berhak. Oleh karena itu, dikembangkanlah cabang ilmu yang mempelajari tentang cara-cara pengamanan data atau informasi yang dikenal dengan istilah kriptografi.

Kriptografi memiliki dua konsep utama yaitu enkripsi dan dekripsi. Enkripsi adalah proses dimana informasi atau data yang hendak dikirim diubah menjadi bentuk yang hampir tidak dikenali sebagai informasi awalnya dengan menggunakan algoritma tertentu. Enkripsi dilakukan dengan cara mengacak suatu data atau informasi menggunakan kunci rahasia sehingga data atau informasi tersebut menjadi tidak berarti karena tidak dapat diketahui. Dekripsi adalah kebalikan dari enkripsi yaitu mengubah data atau informasi dalam suatu bahasa sandi menjadi data atau informasi awal kembali.

Dalam perkembangannya, proses enkripsi diimplementasikan dalam berbagai bentuk data atau informasi. Data dapat berupa angka, karakter, simbol, citra digital, suara atau tanda-tanda yang dapat digunakan untuk dijadikan

informasi. Informasi dapat berupa hasil gabungan, hasil analisa, hasil penyimpulan dan juga hasil pengolahan sistem informasi komputerisasi. Salah satu bentuk data adalah citra digital. Citra merupakan istilah lain dari gambar, yang merupakan data atau informasi yang berbentuk visual. Suatu citra diperoleh dari penangkapan kekuatan sinar yang dipantulkan oleh objek. Citra Digital adalah representasi dari sebuah citra dua dimensi sebagai sebuah kumpulan nilai digital yang disebut elemen gambar atau piksel. Piksel adalah elemen terkecil yang menyusun citra dan mengandung nilai yang mewakili kecerahan dari sebuah warna pada sebuah titik tertentu.

Kriptografi pada citra digital dilakukan dengan cara mengubah warna-warna pada setiap piksel. Perubahan warna yang terjadi pada setiap piksel citra digital membuat data atau informasi yang terkandung pada citra digital tidak dapat diketahui. Banyak algoritma yang dapat digunakan untuk melakukan enkripsi, salah satunya dengan mengimplementasikan teori chaos. Teori chaos merupakan cabang dari matematika yang mempelajari cara membangkitkan suatu bilangan acak yang berguna untuk proses enkripsi.

Salah satu algoritma yang mengimplementasikan teori chaos adalah algoritma Bernoulli map. Algoritma Bernoulli map akan mengenkripsi citra digital dengan cara membangkitkan bilangan acak yang bertujuan mempersulit orang yang tidak berhak untuk mengetahui isi data atau informasinya.

Sebelumnya telah dilakukan percobaan untuk mengamankan data berupa teks dengan menggunakan algoritma RC-5. Percobaan ini dilakukan oleh Ashadi Kurniawan, Mike Yuliana dan M.Zen Samsono Hadi [1]. Pada tahap enkripsi, data yang akan di enkripsi dikembangkan menjadi dua bagian kiri dan bagian kanan. Dilakukan penjumlahan dengan *keyword* yang telah di ekspresi sebelumnya. Kemudian dilakukan operasi enkripsi dan tahap terakhir dilakukan penggabungan untuk mendapatkan data yang telah di enkripsi.

Percobaan lainnya dilakukan oleh Rinaldi Munir [2]. Dalam percobaannya beliau menggunakan algoritma Arnold Cat map dan Logistic map. Pada tahap pertama dalam proses enkripsi adalah dengan melakukan permutasi piksel-piksel di dalam citra. Kemudian ekstrakri empat bit setiap piksel dari citra. Iterasikan Logistic map untuk memperoleh nilai *keystream*. Selanjutnya menggantikan empat bit dari setiap piksel yang di enkripsi.

Berdasarkan pertimbangan diatas, akan digunakan metode yang dapat melakukan keamanan data dengan implementasi proses enkripsi dan dekripsi citra digital menggunakan metode algoritma Bernoulli map dalam pembuatan aplikasi enkripsi dan dekripsi citra digital.

1.2 Rumusan Masalah

Rumusan masalah dalam penelitian ini adalah:

1. Bagaimana membuat suatu aplikasi yang dapat mengenkripsi dan mendekripsi citra digital dengan menerapkan metode algoritma Bernoulli map?
2. Bagaimana menganalisis kinerja aplikasi proses enkripsi dan dekripsi citra digital berbasis Bernoulli map?

1.3 Tujuan Penulisan

Penulisan ini bertujuan untuk mengimplementasikan algoritma Bernoulli map dalam mengenkripsi dan mendekripsi dalam pegamanan citra digital, serta menganalisis kinerja algoritma tersebut.

1.4 Batasan Masalah

Adapun batasan masalah dari pembuatan apliasi ini adalah sebagai berikut:

1. Implementasi pemrograman menggunakan bahasa pemrograman berbasis matematika.
2. Menganalisis kinerja aplikasi yang diukur berdasarkan :

- a. Waktu yang dibutuhkan untuk proses enkripsi dan dekripsi.
- b. Sensitivitas nilai awal kunci yang digunakan.
- c. Kesamaan PSNR (*Peak Signal to Noise Ratio*).

1.5 Metode Penelitian

Metode yang digunakan dalam penulisan ini adalah sebagai berikut:

1. Perencanaan, pada tahap ini dilakukan suatu perencanaan yaitu merencanakan konsep dasar dari aplikasi yang akan dibuat serta menentukan *software* apa yang akan digunakan.
2. Analisis data, pada tahap ini dilakukan studi tentang pemrograman matlab, kriptografi (enkripsi dan dekripsi), fungsi chaos, citra digital pemrograman matlab.
3. Desain, pada tahap ini dirancang algoritma enkripsi dan dekripsi serta tampilan aplikasi seperti halaman utama, tampilan input dan output aplikasi serta merancang struktur navigasinya.
4. Implementasi, pada tahap ini diimplementasikan seluruh rancangan yang telah dibuat pada fase perancangan ke dalam kode sumber menggunakan bahasa pemrograman berbasis matematika.
5. Uji coba, pada tahap ini dilakukan analisis terhadap kinerja algoritma yang diukur berdasarkan waktu proses enkripsi dan dekripsi citra digital, sensitivitas nilai awal kunci serta kesamaan PSNR (*Peak Signal to Noise Ratio*) dari setiap data uji coba.

1.6 Sistematika Penulisan

Sistematika penulisan dalam penulisan skripsi ini terdiri dari lima bab dan didalamnya terdapat beberapa subbab. Adapun sistematika penulisan ini adalah sebagai berikut :

BABI PENDAHULUAN

Bab ini membahas latar belakang masalah, batasan masalah, tujuan penulisan, metode penelitian yang digunakan serta sistematika penulisan.

BAB II LANDASAN TEORI

Bab ini berisikan landasan teori yang digunakan dalam penulisan ini yaitu penjelasan mengenai kriptografi, teori chaos, Bernoulli Map, citra bitmap, citra png, pemrograman Matlab, konsep aljabar modular dan operator XOR, flowchart, dan struktur navigasi.

BAB III ANALISIS DAN PERANCANGAN

Bab ini akan menguraikan secara rinci tahap – tahap pembuatan aplikasi, mulai dari perancangan aplikasi, merancang algoritma enkripsi dan dekripsi, flowchart enkripsi dan dekripsi, struktur navigasi program, membuat rancangan tampilan dan pembuatan aplikasi.

BAB IV ANALISIS HASIL UJI COBA

Bab ini membahas mengenai implementasi aplikasi dan menguraikan hasil uji coba aplikasi.

BAB V PENUTUP

Bab ini menguraikan kesimpulan dari semua bahasan dalam penulisan ini disertai dengan saran dan harapan penulis yang ditujukan kepada semua pihak yang terkait dengan penulisan ini.

BAB II

LANDASAN TEORI

Pada bab ini akan dijelaskan tentang kriptografi, teori chaos, Bernoulli Map, citra digital, pemrograman Matlab, konsep aljabar modular dan operator XOR, serta struktur navigasi.

2.1 Konsep Kriptografi

Kehidupan sekarang ini semakin dilingkupi oleh kriptografi. Berbagai aktivitas seperti transaksi di bank, percakapan melalui telepon seluler, akses internet, dan sebagainya menggunakan kriptografi. Kriptografi menjadi penting sebagai suatu bentuk sistem keamanan informasi. Berbagai masalah keamanan yang dapat terjadi selama proses pengiriman pesan, seperti kerahasiaan, integritas data, otentikasi, dan penyangkalan. Oleh karena itu munculnya kriptografi tidak hanya sebagai alat tetapi juga sebagai teknik yang berguna untuk keamanan pesan.

Kriptografi berasal dari bahasa Yunani, *crypto* dan *graphia*. *Crypto* berarti *secret* (rahasia) dan *graphia* berarti *writing*(tulisan). Menurut terminologinya, kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan ketika pesan dikirim dari suatu tempat ke tempat lain.

Kriptografi adalah ilmu untuk menyimpan rahasia. Pada kenyataannya kriptografi tidak lebih dari suatu perubahan dari penyimpanan rahasia yang besar menjadi rahasia yang kecil. Dalam hal ini rahasia yang besar adalah plainteks dan rahasia yang kecil adalah kunci enkripsi.

Ada beberapa terminologi dasar dalam memahami kriptografi yaitu plainteks, cipherteks, enkripsi, dekripsi, kriptanalisis, dan kunci : [ARI,08]

1. *Plaintext* mengacu kepada berbagai jenis informasi dalam bentuk asli, dapat dibaca, tidak dalam bentuk tersandikan.
2. *Ciphertext* merupakan pesan dalam bentuk tersandikan. Sehingga informasi dalam bentuk *ciphertext* tidak jelas.
3. Enkripsi adalah proses perubahan *plaintext* menjadi *ciphertext*.
4. Dekripsi adalah proses kebalikan dari enkripsi. *Ciphertext* diubah menjadi *plaintext*

5. Kriptanalisis adalah orang yang mencoba mencari kelemahan pola enkripsi. Seorang kriptanalisis akan mencari cara untuk memecahkan suatu pola kriptografi dan kemudian, para ahli menggunakan informasi tersebut untuk membuat pola kriptografi menjadi lebih kuat.
6. Kunci adalah sesuatu yang melindungi data. Kunci dibutuhkan untuk membuka pesan terenkripsi.

Beberapa istilah lain yang juga harus diketahui adalah sebagai berikut.

1. Pengirim atau *Sender* adalah entitas yang mengirimkan pesan kepada entitas lainnya. Entitas yang menerima pesan disebut dengan *receiver*.
2. *Encryption of data in motion* mengacu pada enkripsi pesan yang di transmisikan melalui saluran komunikasi, sedangkan istilah *encryption of data at rest* mengacu pada enkripsi dokumen yang disimpan didalam *storage*. Contoh *encryption of data in motion* adalah pengiriman nomor pin dari mesin ATM ke server di bank pusat. Contoh *encryption of data at rest* adalah enkripsi *filebasis* data dalam *hard disk*.
3. Sistem kriptografi adalah kumpulan yang terdiri dari algoritma kriptografi, *plaintext*, *ciphertext*, dan kunci.
4. Penyadap atau *eavesdroper* adalah orang yang mencoba menagkap pesan selama proses pengiriman. Tujuannya untuk mendapatkan informasi agar dapat memecahkan *ciphertext*.

Serangan terhadap kriptografi dikelompokkan menjadi beberapa cara.

[MUN,06]

- a. Berdasarkan keterlibatan penyerang dalam komunikasi, serangan dibagi atas dua macam yaitu serangan pasif dan aktif.

1. Serangan pasif (*passive attack*).

Serangan ini merupakan jenis serangan dimana penyerang tidak terlibat dalam komunikasi antara pengirim dan penerima. Tujuannya adalah untuk mendapatkan informasi yang digunakan untuk kriptanalisis. Beberapa metode peyadapan data antara lain:

- a. *Wiretapping* adalah metode penyadapan dengan mencegat data yang ditransmisikan pada saluran kabel komunikasi menggunakan sambungan perangkat keras.
- b. *Electromagnetic eavesdropping* adalah pencegatan data melalui saluran *wireless*.
- c. *Acoustic eavesdropping* adalah penangkapan gelombang suara yang dihasilkan dari suara manusia.

2. Serangan aktif (*active attack*)

Serangan ini merupakan jenis serangan dimana penyerang menginterfensi komunikasi dan ikut mempengaruhi sistem untuk keuntungan dirinya. Tujuan dari serangan ini adalah untuk mendapatkan informasi berharga seperti kunci atau nilai rahasia lainnya. Contoh serangannya adalah *man in the middle attack*, yaitu penyerang mengintersepsi komunikasi antara dua pihak yang berkomunikasi kemudian menyerupai salah satu pihak.

- b. Berdasarkan banyaknya informasi yang diketahui oleh kriptanalisis, dikelompokkan menjadi lima jenis, yaitu :

1. *Ciphertext only attack*

Merupakan jenis serangan yang paling sulit karena informasi yang tersedia hanya *ciphertext* saja. Kriptanalisis memiliki beberapa *ciphertext* dari beberapa pesan yang semuanya dienkripsi dengan algoritma yang sama.

Kriptanalisis bertugas mengolah *ciphertext* tersebut sehingga menemukan *plaintext* sebanyak mungkin atau menemukan kunci untuk mendekripsikannya.

2. *Known plaintext attack*

Merupakan jenis serangan dimana kriptanalisis memiliki pasangan plainteks dan *ciphertext* yang berkoresponden.

3. *Chosen plaintext attack*

Dibandingkan dengan *known-plaintext* jenis serangan ini lebih hebat karena kriptanalisis dapat memilih *plaintext* yang dimilikinya untuk dienkripsi, yaitu *plaintext* yang lebih mengarahkan penemuan kunci.

4. *Chosen ciphertext attack*

Merupakan jenis serangan dimana kriptanalisis memilih *ciphertext* untuk didekripsi dan memiliki akses ke *plaintext* hasil dekripsi (misalnya terhadap mesin elektronik yang melakukan dekripsi secara otomatis). Jenis serangan ini biasanya dipakai pada sistem kriptografi.

5. *Chosen text attack*

Merupakan jenis serangan yang merupakan kombinasi *chosen ciphertext attack* dan *chosen plaintext attack*.

c. Berdasarkan teknik yang digunakan dalam menemukan kunci, serangan dibagi atas:

1. *Exhaustive attack* atau *brute force attack* adalah serangan untuk mengungkap *plaintext* menggunakan semua kemungkinan kunci.
2. *Analytical attack* adalah serangan dengan menganalisis kelemahan algoritma kriptografi untuk mengurangi kemungkinan kunci yang tidak mungkin ada.

Selain serangan diatas, terdapat serangan lain, yaitu:

1. *Related-key attack*. Kriptanalisis memiliki *ciphertext* yang dienkripsi dengan dua kunci berbeda. Kriptanalisis tidak mengetahui kedua kunci tersebut tetapi dia mengetahui hubungan kedua kunci.
2. *Rubber-horse cryptanalysis* adalah jenis serangan yang paling ekstrim karena penyerang mengancam, mengirim surat gelap, atau melakukan penyiksaan sampai orang yang memegang kunci memberikan kunci untuk mendekripsi pesan.

Ada empat tujuan mendasar dari ilmu kriptografi ini yang juga merupakan aspek keamanan informasi, yaitu : [ARI,08]

1. Kerahasiaan (*Confidentiality*)

Kerahasiaan adalah layanan yang digunakan untuk menjaga isi dari informasi dari siapapun kecuali yang memiliki otoritas atau kunci rahasia untuk membuka atau mengupas informasi yang telah disandi.

2. Integritas Data (*Data Integrity*).

Integritas adalah berhubungan dengan penjagaan dari perubahan data secara tidak sah. Untuk menjaga integritas data, sistem harus memiliki kemampuan untuk mendeteksi manipulasi data oleh pihak-pihak yang tidak berhak, antara lain penyisipan, penghapusan, dan pensubsitusian data lain kedalam data yang sebenarnya.

3. Otentikasi (*Autentication*).

Otentikasi adalah berhubungan dengan identifikasi atau pengenalan, baik secara kesatuan sistem maupun informasi itu sendiri. Dua pihak yang saling berkomunikasi harus saling memperkenalkan diri. Informasi yang dikirimkan melalui kanal harus diautentikasi keaslian, isi datanya, waktu pengiriman, dan lain-lain.

4. Ketiadaan Penyangkalan (*Non-repudiation*).

Ketiadaan penyangkalan adalah usaha untuk mencegah terjadinya penyangkalan terhadap pengiriman/terciptanya suatu informasi oleh yang mengirimkan atau membuat.

5. *Privacy*

Merupakan layanan yang mengarah ke data yang bersifat pribadi.

2.2 Fungsi Chaos Bernoulli Map

Teori *chaos* terkenal dengan sifat sensitivitas atau peka pada kondisi atau nilai awal. Sifat tersebut berarti perbedaan kecil pada nilai awal akan menghasilkan perbedaan yang sangat besar pada nilai fungsi setelah fungsi diiterasi beberapa kali.

Teori chaos mulai diformulasikan pada tahun 1961 oleh seorang meteorology, Edward Lorentz. Seseorang yang tercatat sebagai penemu teori *chaos* adalah Henri Poincare, yang pada tahun 1880 menemukan bahwa terdapat orbit yang bersifat nonperiodik, dalam arti tidak memiliki proses kemunculan secara tetap atau formulatif. Berawal pada tahun 1961 saat Edward Lorentz membuat

model perkiraan cuaca, model matematis cuaca diiterasikan untuk memperoleh perkiraan cuaca pada waktu yang akan datang. Dengan hanya mengubah sedikit nilai awal iterasi, perkiraan cuaca yang dihasilkan mengalami perbedaan yang besar. Teori *chaos* berasal dari teori sistem yang memperlihatkan kemunculan yang tidak teratur, meskipun sebenarnya teori ini digunakan untuk menjelaskan kemunculan data acak.

Bernoulli map merupakan salah satu fungsi untuk membuat bilangan acak yang digunakan dalam aplikasi kriptografi . Fungsinya dinyatakan sebagai :[AHM,14]

$$X_{n+1} = r \times X_n \bmod 1 \quad (2.1)$$

dengan:

1. X_n mengambil nilai dari rentang $0, 1, r \in (0, 1)$.
2. Variabel r mengambil nilai dari 0 sampai $\infty, r \in (0, \infty)$.
3. Nilai awal $X_n = 0,1$.
4. Iterasi pengulangan = 8000 untuk r bertambah 0.001.

Hasil simulasi ditunjukkan pada gambar 1 parameter r dapat dibagi menjadi dua segmen yang dapat dieksperimen pada kondisi berikut,

1. Ketika $r \in (0, 1)$ hasil perhitungan datang ke hasil yang sama setelah beberapa iterasi tanpa perilaku acak.
2. Ketika $r \in (1, \infty)$ itu menjadi sistem yang acak tanpa periodisitas.

Dari pembahasan sebelumnya kita dapat menyimpulkan bahwa:

1. Ketika $r \in (0, 1)$ titik berkonsentrasi pada beberapa nilai tidak bisa digunakan dalam gambar *cryptosystem*.
2. Ketika $r \in (1, 4.99)$ Bernoulli map memiliki perubahan kecil dalam kisaran r untuk menunjukkan perilaku acak dan karenanya memiliki ketergantungan yang sensitif sehingga dapat digunakan untuk gambar *cryptosystem* dalam kisaran kecil ini.
3. Diwajibkan tidak menggunakan nilai integer r ketika menggunakan Bernoulli map di gambar *cryptosystem*, itu harus menggunakan nilai fraksi untuk r dan ini adalah salah satu kelemahan nya .

Dalam (Kocarev dan Lian, 2011) fungsi tersebut didefinisikan dengan nama lain, yaitu Renyi map. Bernoulli map dapat ditulis sebagai,

$$f : (0,1) \rightarrow (0,1), f(x) = rx \bmod 1, r > 1$$

dengan operator modulo diperluas untuk bilangan real. Sehingga, $rX_n \bmod 1$ menyatakan bagian bilangan pecahan dari rX_n . Secara matematis dituliskan $rX_n \bmod 1 = rX_n - \lfloor X_n \rfloor$. Sehingga nilai X_{n+1} akan selalu berada pada $(0,1)$ untuk sembarang n .

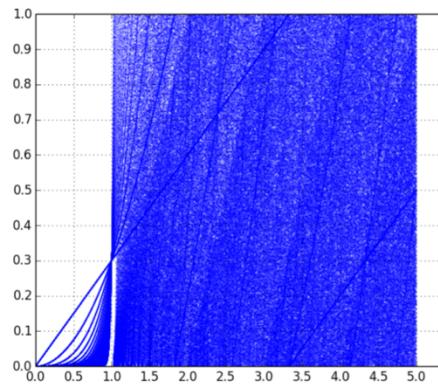
2.3 Bernoulli map Fungsi yang tidak invertible

Misalkan S dan T adalah himpunan bilangan riil. Fungsi $f : S \rightarrow T$ dikatakan fungsi berkorespondensi 1–1 atau bijektif jika f fungsi injektif dan surjektif. Misalkan S dan T adalah himpunan bilangan riil. Fungsi $f : S \rightarrow T$ dikatakan fungsi injektif jika untuk $s_1 \neq s_2$ di S , $s_1 \neq f(s_2)$ di T . Ekuivalen dengan pernyataan, f fungsi injektif jika $(s_1) = f(s_1)$ mengakibatkan $s_1 = s_2$. [HER,06]

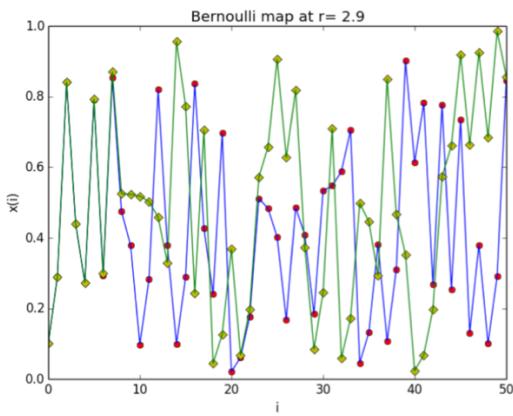
Fungsi Bernoulli map tidak memiliki fungsi invers atau tidak *invertible* karena untuk sembarang. Sehingga tidak memenuhi sifat injektif. Jadi, fungsi Bernoulli map tidak *invertible*.

Untuk $r \in (0,1)$, seiring bertambahnya n , nilai X_n akan menuju ke nol. Sedangkan untuk $r > 1$ sebaran nilai X_n merata pada $(0,1)$. Sehingga fungsi f bersifat chaos untuk $r > 1$. Gambar 2.1 menunjukkan diagram bifurkasi dari Bernoulli map untuk $r \in [0,5]$.

Sifat transitif secara topologi dapat dilihat dari diagram bifurkasi pada Gambar 2.1. Diagram bifurkasi adalah diagram yang memperlihatkan hasil pemetaan suatu fungsi ketika parameter dalam fungsi tersebut diubah-ubah.



Gambar 2.1 Bifurkasi Bernoulli map



Gambar 2.2 Plot Sebaran Nilai X_n dari Bernoulli Map untuk $i = 0,1,2,\dots,50$

Pada Gambar 2.2 menunjukkan plot dari sebaran $X_{n+1} = 2.9 X_n \bmod 1$ dengan beda sebesar 10^{-4} dan nilai awal yang digunakan adalah $X_0 = 0.1$. Terlihat bahwa nilai X_0 sampai dengan X_5 hampir sama, akan tetapi semakin besar iterasinya, nilai X_n pada kedua orbit tersebut semakin berbeda. Hal tersebut menunjukkan bahwa Bernoulli map memiliki sifat sensitif terhadap nilai awal.

2.4 Citra Digital

Citra digital merupakan representasi dari citra yang diambil oleh mesin dengan bentuk pendekatan berdasarkan sampling dan kuantitasi. Sampling menyatakan besarnya kotak-kotak yang disusun dalam baris dan kolom. Dengan kata lain sampling pada citra menyatakan besar kecilnya ukuran piksel (sebutan

untuk titik dalam sebuah citra) pada citra, sedangkan kuantitasi menyatakan besarnya nilai tingkat kecerahan yang dinyatakan dalam nilai tingkat keabuan (*grayscale*) sesuai jumlah bit biner yang digunakan oleh mesin. Dapat dikatakan kuantitasi menyatakan jumlah warna yang ada pada citra.

Nilai suatu piksel memiliki nilai dalam rentang tertentu, dari nilai minimum sampai nilai maksimum. Jangkauan yang digunakan berbeda-beda tergantung dari jenis warnanya, namun secara umum jangkauannya adalah 0 sampai dengan 255. Citra dengan penggambaran seperti ini digolongkan dalam citra integer. [DAR,10]

Berikut adalah jenis-jenis citra berdasarkan nilai pikselnya, adalah:

1. *Binary Image* atau *Black and White Image* (BW)

Merupakan jenis citra yang paling sederhana, dimana sebuah piksel hanya dapat menampung 2 buah nilai (1bit), yaitu hanya mewakili hitam atau putih. Dihasilkan dari piksel pada citra yang memiliki nilai intensitas dibawah 128 diubah menjadi hitam (bernilai 0), sedangkan yang memiliki nilai intensitas diatas 128 diubah menjadi putih (bernilai 1).

2. *Greyscale Image*

Merupakan jenis citra yang hanya memiliki satu nilai kanal pada setiap pikselnya, dengan kata lain nilai bagian RED = GREEN = BLUE. Nilai tersebut digunakan untuk menunjukkan tingkat intensitas.

3. Citra berwarna atau *RGB Image*

Merupakan jenis citra yang digunakan untuk gambar-gambar berwarna. Jenis citra berwarna ini merupakan kombinasi dari tiga warna dasar, yaitu merah, hijau, dan biru. Pada tiap-tiap pikselnya disusun oleh tiga komponen warna tersebut *Red*, *Green* dan *Blue*.

2.4.1 Citra Digital Berformat Bitmap

Gambar Bitmap merupakan rekonstruksi dari gambar asli. Gambar Bitmap adalah gambar yang tersimpan sebagai rangkaian piksel yang memenuhi bidang titik-titik dilayar komputer. Seluruh informasi gambar dinyatakan dalam piksel.

Untuk menampilkan gambar, komputer akan mengatur tiap titik dilayar sesuai dengan detail warna Bitmap. Ukuran sebuah citra Bitmap relatif besar .

Struktur umum sebuah citra Bitmap adalah :

1. Bitmap File Header merupakan *header* citra Bitmap yang berisi tipe serta ukuran citra.
2. Bitmap Info Header merupakan informasi mengenai *header* citra Bitmap. Informasi ini berupa dimensi, tipe kompresi yang digunakan (jika ada), dan format warna citra.
3. RGBQuad merupakan tabel warna yang dipakai pada citra Bitmap.
4. Byte merupakan informasi yang terperinci mengenai *bit* citra untuk setiap piksel.

2.4.2 Citra *Digital* Berformat GIF

Graphics Interchange Format (GIF) pertama kali diperkenalkan pada tahun 1987 oleh CompuServe, merupakan format citra berwarna yang menggunakan teknik kompresi secara efektif, menghasilkan ukuran citra yang lebih kecil sehingga pengiriman citra melalui internet menjadi lebih cepat. Diciptakan sebagai protokol komunikasi citra untuk operasi pertukaran data citra yang tersimpan pada sistem lokal maupun *remote*. Citra GIF tidak tergantung dengan aplikasi perangkat lunak tertentu, tetapi banyak aplikasi yang dapat mengkonversi baik ke format GIF maupun dari GIF. Citra GIF adalah citra pertama yang dapat diterima secara universal, dapat diintegrasikan pada halaman *web*, dan menjadi format utama untuk menampilkan citra secara *online* melalui *World Wide Web*.

Format awal GIF disebut dengan GIF 87a, dan kemudian berkembang menjadi GIF 89a. Perbedaan keduanya adalah GIF 89a mendukung adanya animasi sederhana dengan menampilkan beberapa citra yang disertai dengan waktu jeda, sedangkan GIF 87a merupakan GIF statik yang hanya terdiri dari satu citra.

Struktur sebuah citra GIF, terdiri atas :

1. *Header*, terdapat diawal *file* dan terdiri atas 6 *byte* penanda berisi “GIF87a” atau “GIF89a”.

2. *Global Screen Descriptor*, berisi dimensi dan warna latar belakang citra.
3. *Global Color Table*, merupakan suatu larik yang berisi nilai RGB untuk setiap piksel.
4. Satu atau beberapa citra lain yang terdiri atas *header* citra, tabel warna lokal, sekumpulan blok data, dan sebuah blok akhir.
5. *Trailer*.

2.4.3 Citra *Digital* Berformat PNG

Sebuah citra PNG terdiri atas 8 *byte* penanda PNG (*signature*), yang diikuti dengan sekumpulan *chunk*. *Chunk* merupakan suatu urutan blok biner.

1. *Signature* PNG terdiri atas 8 *byte* yang selalu terdiri atas nilai desimal 137, 80, 78, 71, 13, 10, 26, dan 10. *Signature* ini menandakan bahwa suatu citra PNG terdiri atas sekumpulan *chunk* yang dimulai dengan IHDR dan diakhiri dengan IEND. *Chunk* terdiri atas panjang, tipe, data, dan CRC (*Cyclic Redundancy Check*).
2. IHDR *chunk* berisi informasi mengenai panjang dan lebar citra, kedalaman warna, dan tipe warna.
3. PLTE *chunk* terdiri atas 256 palet warna dengan 3 *byte* RGB.
4. IDAT *chunk* terdiri atas data citra.
5. IEND *chunk* merupakan akhir dari sebuah citra PNG.

2.4.4 Perbedaan Citra Bitmap, GIF statik, dan PNG

Perbedaan citra Bitmap, GIF statik, dan PNG terlihat pada teknik kompresi yang digunakan. Bitmap biasanya tidak menggunakan teknik kompresi, namun citra ini memungkinkan untuk disimpan dengan format kompresi menggunakan *Run Length Encoding*. GIF statik menggunakan kompresi *lossless*, artinya kualitas citra tidak berkurang ketika citra disimpan. PNG, sama seperti GIF statik, menggunakan teknik kompresi *lossless* tetapi dapat mengkomprimasi citra lebih efisien dibandingkan dengan format GIF statik sehingga menghasilkan ukuran citra yang lebih kecil.

Selain itu, perbedaan juga terlihat pada mode warna yang digunakan. Bitmap memiliki mode warna RGB 1, 4, 8, 16, atau 24 *bit*. GIF statik memiliki mode warna indeks *color* 1 sampai 8 *bit*. PNG memiliki mode warna RGB 24 atau 48 *bit*, *Grayscale* 8 atau 16 *bit*, indeks *color* 1 sampai 8 *bit*, dan *bilevel* 1 *bit*.

2.5 Matrix Laboratory (MATLAB).

Bahasa pemrograman sebagai media untuk berinteraksi antara manusia dan komputer saat dibuat semakin mudah dan cepat. Sebagai contoh, dapat dilihat dari perkembangan bahasa pemrograman Pascal yang terus memunculkan varian baru sehingga akhirnya menjadi Delphi, demikian pula dengan Basic dengan Visual Basicnya serta C dengan C + + Buildernya. Pada akhirnya semua bahasa pemrograman akan semakin memberikan kemudahan pemakainya (*programmer*) dengan penambahan fungsi-fungsi baru yang sangat mudah digunakan bahkan oleh pemakai tingkat pemula.

Matlab muncul di dunia bahasa pemrograman yang cenderung dikuasai oleh bahasa yang telah mapan. Tentu saja sebagai bahasa pemrograman yang baru Matlab akan sukar mendapat hati dari pemakai. Namun Matlab hadir tidak dengan fungsi dan karakteristik yang umumnya ditawarkan bahasa pemrograman lain yang biasanya hampir seragam. Matlab dikembangkan sebagai bahasa pemrograman sekaligus alat visualisasi, yang menawarkan banyak kemampuan untuk menyelesaikan berbagai kasus yang berhubungan langsung dengan disiplin keilmuan matematika. Matlab memiliki kemampuan mengintegrasikan komputasi, visualisasi, dan pemrograman dalam sebuah lingkungan yang tunggal dan mudah digunakan. Matlab menyediakan beberapa pilihan untuk dipelajari, mempelajari metode visualisasi saja, pemrograman saja, atau kedua-duanya digunakan. Matlab menyediakan beberapa pilihan untuk dipelajari, mempelajari metode visualisasi saja, pemrograman saja, atau kedua-duanya.

Matlab adalah bahasa pemrograman level tinggi yang dikhususkan untuk komputasi teknis. Bahasa ini mengintegrasikan kemampuan

komputasi, visualisasi, dan pemrograman dalam sebuah lingkungan yang tunggal dan mudah digunakan. Matlab memberikan sistem interaktif yang menggunakan konsep array sebagai standar variabel elemennya tanpa membutuhkan pendeklarasian array seperti pada bahasa pemrograman lain.

Kehadiran Matlab sebagai bahasa pemrograman memberikan jawaban sekaligus tantangan. Matlab menyediakan beberapa pilihan untuk dipelajari, mempelajari metoda visualisasi saja, pemrograman saja, atau keduanya. Matlab memang dihadirkan bagi orang-orang yang tidak ingin disibukkan dengan rumitnya sintaks dan alur logika pemrograman, sementara pada saat yang sama membutuhkan hasil komputasi dan visualisasi yang maksimal untuk mendukung pekerjaannya. Selain itu, Matlab juga memberikan kemudahan bagi *programmer* atau *developer* program yaitu untuk menjadi pembanding yang sangat handal, hal tersebut dapat dilakukan karena kekayaannya akan fungsi matematika, fisika, statistika, dan visualisasi.

2.5.1 Lingkungan Kerja Matlab

Sebagaimana bahasa pemrograman lainnya, Matlab juga menyediakan lingkungan kerja terpadu yang sangat mendukung dalam membangun sebuah aplikasi. Pada setiap versi Matlab terbaru, lingkungan terpadunya akan semakin dilengkapi. Lingkungan tepat ini terdiri dari beberapa form yang memiliki kegunaan masing-masing. Setiap pertama kali membuka aplikasi Matlab, maka akan diperoleh beberapa form. Matlab akan menyimpan mode/setting terakhir lingkungan kerja yang digunakan sebagai *mode* atau *setting* lingkungan kerja pada saat membuka aplikasi Matlab di waktu berikutnya.

2.5.2 GUI Designer pada Matlab (GUIDE)

GUIDE atau GUI Builder merupakan sebuah Graphical User Interface (GUI) yang dibangun dengan objek grafis seperti tombol (*push button*), *edit*, *slider*, *text*, *combo*, sumbu (*axes*), maupun menu dan lain-lain untuk kita gunakan. Sebagai contoh, ketika menggerakkan *slider*, maka kita dapat

melihat perubahan sebuah nilai. Kemudian ketika kita menekan tombol OK, maka aplikasi akan dijalankan. Aplikasi yang menggunakan GUI umumnya lebih mudah dipelajari dan digunakan karena orang yang menjalankannya tidak perlu mengetahui perintah yang ada dan bagaimana perintah bekerja.

Jika membicarakan pemrograman berorientasi visual, yang paling dikenal adalah sederetan bahasa pemrograman seperti Visual Basic, Delphi, Visual C, Visual Fox Pro, dan yang lainnya yang memang didesain untuk itu. Matlab merintis ke arah pemrograman yang menggunakan GUI dimulai dari Matlab versi 5, yang terus disempurnakan hingga sekarang.

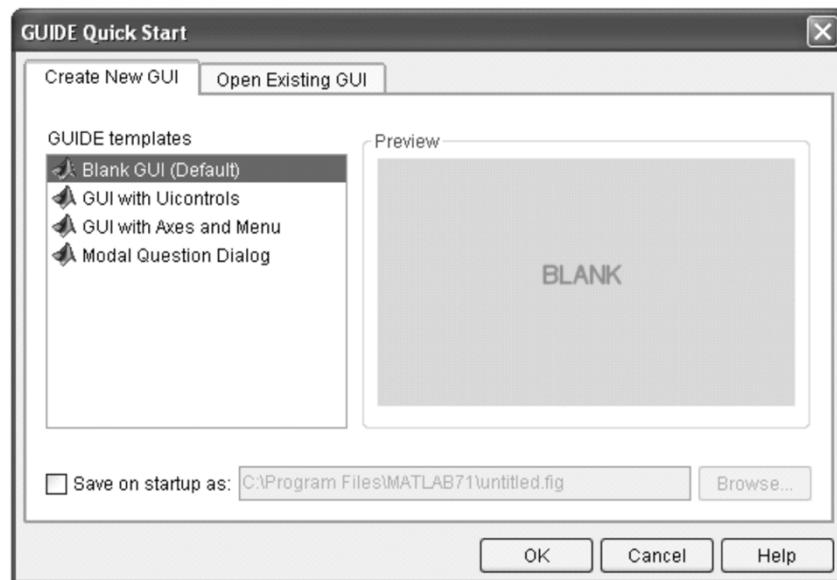
Tidak seperti bahasa pemrograman lainnya, GUIDE Matlab memiliki banyak keunggulan tersendiri, antara lain:

1. GUIDE Matlab cocok untuk aplikasi-aplikasi berorientasi sains.
2. Matlab memiliki banyak fungsi built-in yang siap digunakan dan pemakai tidak perlu repot membuatnya sendiri.
3. Ukuran file, baik Fig-file maupun M-file yang dihasilkan relatif kecil.
4. Kemampuan grafisnya cukup handal dan tidak kalah dengan bahasa pemrograman lainnya. [ARI06]

Untuk memulai penggunaan GUI Matlab, dapat dilakukan dengan dua cara yaitu:

1. Melalui command Matlab kita ketikkan >> guide, atau
2. Klik tombol *Start* pada halaman utama dan pilih Matlab, lalu pilih GUIDE atau GUI Builder.

Selanjutnya akan masuk ke sebuah kotak dialog pilihan GUIDE *Quick Start*.



Gambar 2.2. GUIDE *Quick Start*

Pada Gambar 2.2 terdapat dua pilihan, yaitu *Create new GUI* dan *Open Existing GUI*:

1. *Create New GUI*

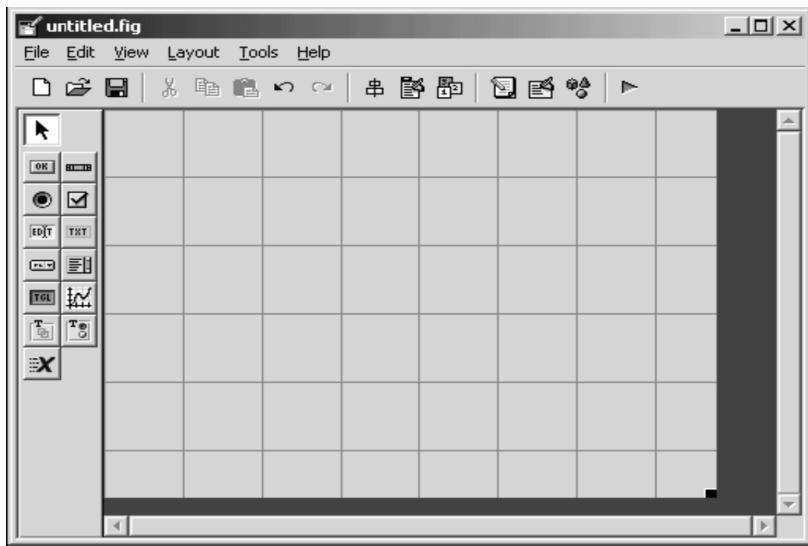
Kita dapat memilih *Create New GUI* jika memang belum pernah membuat aplikasi GUI MATLAB atau memang ingin membuat sebuah *figure* baru. Menu *Create New GUI* akan memberikan kita beberapa pilihan GUIDE template. Keuntungan menggunakan GUIDE template adalah kita dapat membuat aplikasi GUI kita menjadi lebih cepat dan mudah karena sudah tersedia beberapa bentuk (*prototype*) GUI.

Pada GUIDE template, kita dapat memilih :

a. *Blank GUI (Default)*

Blank GUI merupakan sebuah GUI dengan *figure* kosong. Kita dapat mengatur sendiri komponen yang kita butuhkan sesuai dengan aplikasi yang kita buat.

Blank GUI merupakan kondisi default dari GUIDE dan dipilih jika kita memang akan membuat sebuah aplikasi dengan komponen yang *layoutnya* tidak terdapat pada *template* lain.

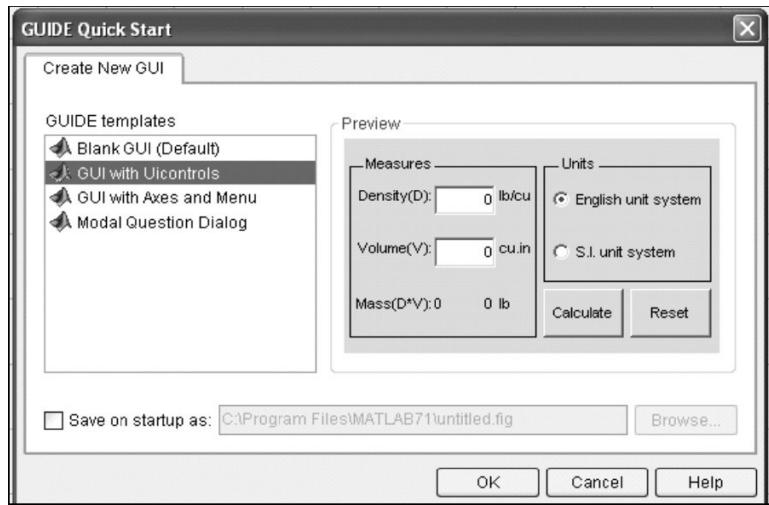


Gambar 2.3. *Blank GUI (Default)*

Setelah menjalankan *Blank GUI (default)*, maka kita akan dibawa ke halaman baru.

b. GUI with UI *controls*

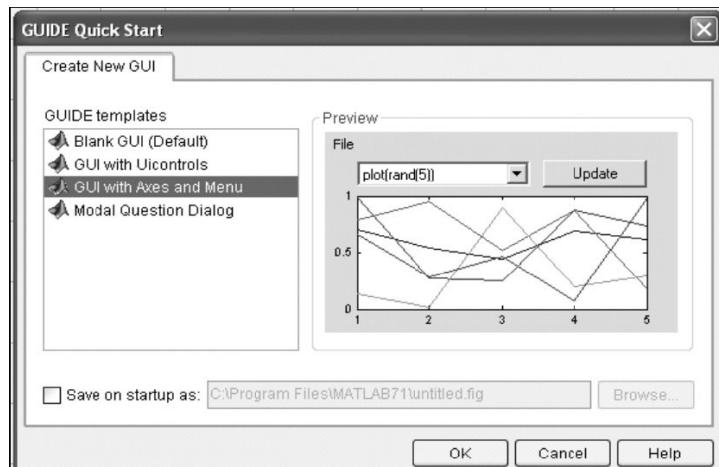
Dengan GUI with UI *controls*, kita dapat membuat sebuah aplikasi MATLAB dengan beberapa kontrol, misalnya *frame*, *radio button*, *pushbutton*, *edit text*, dan lainnya. Jika kita ingin membangun sebuah aplikasi dengan *layout* yang mirip, maka kita dapat memilih *template* ini, selanjutnya beberapa properti yang ada dapat diatur kembali sesuai aplikasi kita.



Gambar 2.4. GUI dengan *UI controls*

. c. *GUI with Axes and Menu*

Dengan *GUI with Axes and Menu*, kita lebih mudah membuat plot berbagai bentuk data yang divisualisasikan dalam sebuah *axes*.

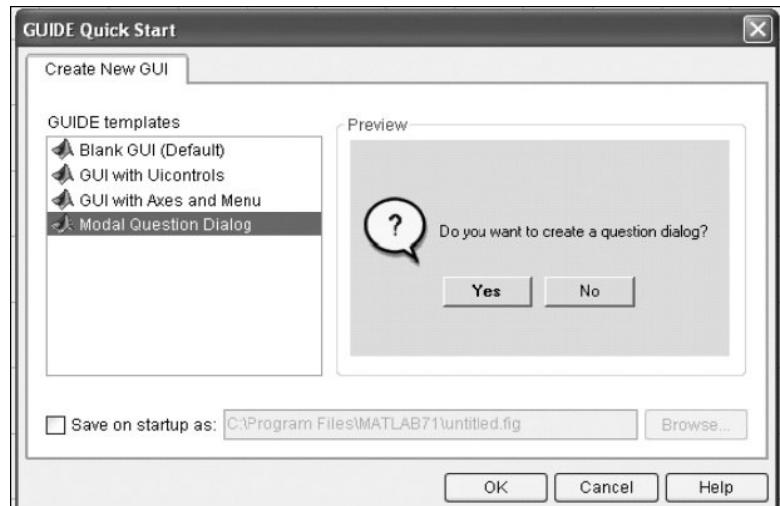


Gambar 2.5 GUI dengan *Axes* dan *Menu*

d. *Modal Question Dialog*

Template Modal Question Dialog merupakan sebuah *template* dengan kotak dialog sebagai konfirmasi untuk menutup sebuah aplikasi. Kita dapat menggunakan *template* ini untuk mengakhiri sebuah aplikasi dan untuk menghindari penutupan aplikasi secara tidak sengaja, yang

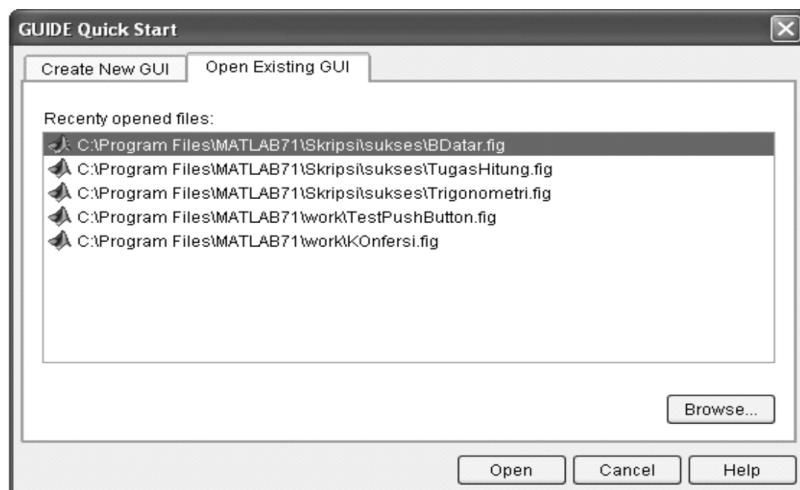
dapat membawa dampak kerugian hilangnya sebuah informasi.



Gambar 2.6. GUI Question Dialog

2. Open Existing GUI

Jika kita sudah memiliki *file figure* MATLAB atau akan memodifikasi *file figure*, maka kita dapat menggunakan pilihan *Open Existing GUI*. Kita hanya tinggal melakukan pencarian untuk mencari lokasi *file figure*.



Gambar 2.7. Open Existing GUI

2.5.3 Perintah pada Matlab untuk pengelolaan citra

Di dalam pengelolaan citra perlu beberapa perintah Matlab, seperti misalnya pembacaan citra, menampilkan citra, dan menyimpan citra. Citra dibaca dalam lingkungan Matlab menggunakan fungsi imread dengan sintaks seperti berikut
imread('namafile')

Parameter nama file adalah string yang berisi nama lengkap file dari file citra (termasuk ekstensi dari file juga harus disertakan). Berikut adalah contoh perintah imread dalam Matlab.

```
>> i = imread('bungamawar.jpg');
```

Perintah di atas merupakan pembacaan file dengan nama bungamawar dan berekstensi jpg, dan kemudian disimpan ke dalam variabel i dengan jenis matriks. Citra ditampilkan di desktop Matlab dengan menggunakan fungsi imshow, yang mempunyai sintaks seperti berikut,

imshow(f, G) dengan f adalah array citra dan G adalah jumlah level intensitas yang digunakan untuk menampilkannya. Jika G diabaikan, maka akan digunakan intensitas default, yaitu 256.

Menyimpan citra ke dalam temat penyimpanan digunakan perintah imwrite, yang sintaks dasarnya sepaerti berikut:

```
imwrite(f, 'namafile')
```

String yang digunakan untuk nama file haruslah nama yang mudah dikenali.

2.6 Operasi XOR

XOR adalah operasi *Exclusive-OR* yang dilambangkan dengan tanda “ \oplus ”. Jika meng-XOR-kan dua buah bit yang sama nilainya, maka operasi XOR akan menghasilkan nilai bit “0” (nol) dan jika meng-XOR-kan dua buah bit yang masing-masing nilainya berbeda, maka akan menghasilkan nilai bit 1 (satu). Aturan yang berlaku untuk operasi XOR dapat dilihat pada Tabel 2.1. [RIN,06]

Tabel 2.1 Aturan Operasi XOR

Masukan		Keluaran $A \oplus B$
A	B	
0	0	0
0	1	1
1	0	1
1	1	0

Berdasarkan Tabel 2.1, jika nilai A di-XOR-kan dengan nilai B sebanyak 2 (dua) kali, maka akan didapatkan nilai A kembali. Karena sifat istimewa yang dimiliki operasi XOR tersebut sehingga operasi XOR cenderung dipakai dalam proses enkripsi dan dekripsi yang memiliki algoritma yang sama.

2.7 Struktur Navigasi

Struktur navigasi digunakan sebagai penuntun alur sebuah aplikasi multimedia atau dapat pula dianalogikan sebagai diagram alur dalam perancangan bahasa pemrograman. Struktur navigasi berfungsi untuk menggambarkan dengan jelas hubungan dan rantai kerja seluruh elemen yang akan digunakan dalam aplikasi. Dengan penggambaran struktur navigasi, pembuatan sebuah aplikasi dapat sistematis dan mudah.

Jenis struktur navigasi dikelompokkan menjadi 4 (empat) struktur yang berbeda yaitu linier, hirarki, non linier dan campuran yang mempunyai perbedaan dalam bentuk rangkaianya.

1. Struktur Navigasi Linier.

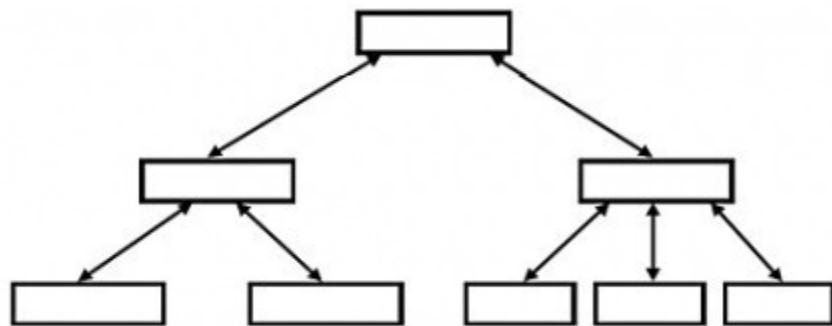
Merupakan struktur yang hanya memiliki 1 (satu) rangkaian alur yang terurut dan tidak diperkenankan adanya percabangan dan cocok digunakan untuk presentasi multimedia yang tidak terlalu membutuhkan interaktifitas. Tampilan struktur navigasi linier dapat dilihat pada Gambar 2.8



Gambar 2.8 Struktur Navigasi Linier

2. Struktur Navigasi Hirarki.

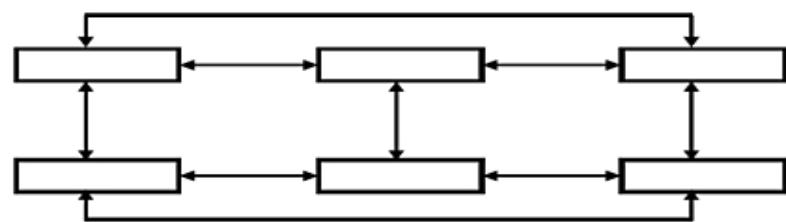
Struktur navigasi hirarki sering disebut struktur navigasi bercabang, yaitu merupakan suatu struktur yang mengandalkan percabangan untuk menampilkan data atau gambar pada layer dengan kriteria tertentu. Tampilan pada menu utama disebut master page (halaman utama satu), halaman tersebut mempunyai halaman percabangan yang disebut slave page (halaman pendukung) dan jika dipilih akan menjadi halaman kedua, begitu seterusnya. Tampilan struktur navigasi hirarki dapat dilihat pada Gambar 2.9



Gambar 2.9 Struktur Navigasi Hirarki

3. Struktur Navigasi non Linier.

Struktur navigasi non linier (tidak terurut) merupakan pengembangan dari struktur navigasi linier, hanya saja pada struktur ini diperkenankan untuk membuat percabangan. Percabangan pada struktur non linier berbeda dengan percabangan pada struktur hirarki, pada struktur ini kedudukan semua page sama, sehingga tidak dikenal adanya master atau slave page. Tampilan struktur

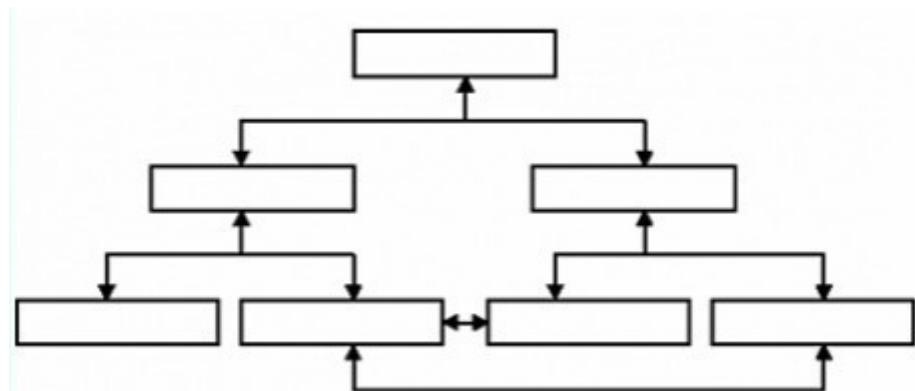


navigasi non linier dapat dilihat pada Gambar 2.10

Gambar 2.10 Struktur Navigasi Non-Linier

4. Struktur Navigasi Campuran.

Struktur navigasi campuran (composite) merupakan gabungan dari struktur sebelumnya dan disebut juga struktur navigasi bebas, maksudnya adalah jika suatu tampilan membutuhkan percabangan maka dibuat percabangan. Struktur ini paling banyak digunakan dalam pembuatan aplikasi multimedia. Tampilan struktur navigasi campuran dapat dilihat pada gambar 2.11



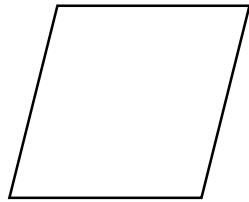
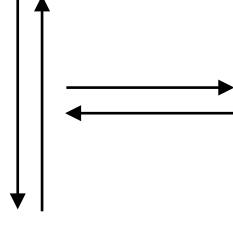
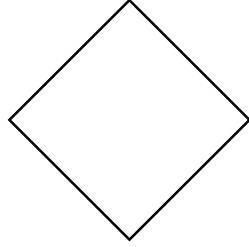
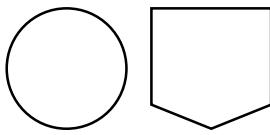
Gambar 2.11 Struktur Navigasi Campuran

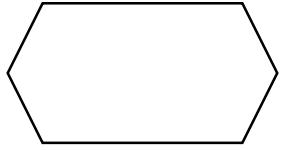
2.8 Flowchart

Flowchart adalah diagram alir merupakan sebuah diagram dengan simbol-simbol grafis yang menyatakan aliran algoritma atau proses yang menampilkan langkah-langkah yang disimbolkan dalam bentuk kotak, beserta urutannya dengan menghubungkan masing-masing langkah tersebut menggunakan tanda panah. Diagram ini bias memberi solusi selangkah demi selangkah untuk penyelesaian masalah yang ada di dalam proses atau algoritma tersebut. *Flowchart* digunakan

untuk alat bantu komunikasi dan untuk dokumentasi. Simbol *flowchart* dapat dilihat pada Tabel 2.2.

Tabel 2.2 Simbol Flowchart

Gambar	Keterangan
	Input/Output; Digunakan untuk mewakili data i/o.
	Proses; Digunakan untuk mewakili suatu proses
	Garis alir; Menunjukkan arus dari proses.
	Keputusan; Digunakan untuk suatu seleksi kondisi di dalam program.
	Penghubung; Menunjukkan penghubung ke halaman yang sama atau halaman lain.

Gambar	Keterangan
	Proses terdefinisi; Menunjukkan suatu operasi yang rinciannya ditunjukkan di tempat lain.
	Persiapan; Digunakan untuk memberi nilai awal suatu besaran
	Terminal; Menunjukkan awal dan akhir dari suatu proses

BAB III

ANALISIS DAN PERANCANGAN

Pada bab ini akan dijelaskan tentang perencanaan aplikasi, algoritma, enkripsi dan dekripsi, *flowchart* enkripsi dan dekripsi, struktur navigasi, rancangan tampilan dan pembuatan aplikasi.

3.1 Analisis Masalah

Dalam penulisan ini akan dilakukan pembuatan aplikasi untuk mengamankan data dengan mengimplementasikan algoritma *Bernoulli Map* pada proses enkripsi dan dekripsi citra digital berwarna bertipe *bitmap* (.bmp) dan *portable network graphics* (.png).

Tahap ini, analisis masalah aplikasi yang akan dibuat terdapat pada perancangan aplikasi. Pertama kali aplikasi dijalankan, maka akan muncul halaman menu utama aplikasi. Pada menu utama terdiri dari 5 menu yaitu menu enkripsi, menu dekripsi, menu perbandingan citra, menu bantuan dan menu tentang.

Pada menu enkripsi, pengguna diminta untuk memasukkan nilai X_n dan r sebagai kunci enkripsi. Menu ini digunakan untuk mengubah gambar asli menjadi gambar acak yang tidak dapat terlihat lagi. Citra dengan ekstensi .bmp atau .png dapat dipilih pada folder kerja matlab dengan menekan tombol pilih citra. Setelah citra dipilih dan kunci ditentukan, langkah selanjutnya adalah pengguna menekan tombol enkripsi supaya citra tersebut diproses. Waktu proses enkripsi akan tampil pada bagian bawah nilai X_n dan r dalam satuan detik. Citra yang sudah berhasil dienkripsi dapat disimpan dengan menggunakan tombol simpan citra.

Pada menu dekripsi, pengguna diminta untuk memasukkan nilai X_n dan r sebagai kunci enkripsi. Nilai X_n dan r yang dimasukkan dalam menu ini harus sama dengan nilai X_n dan r yang dimasukkan dalam menu enkripsi. Menu ini digunakan untuk mengubah gambar acak hasil enkripsi menjadi gambar asli seperti sebelum dienkripsi. Citra dengan ekstensi .bmp atau .png dapat dipilih pada folder kerja matlab dengan menekan tombol pilih citra. Setelah citra dipilih dan kunci ditentukan, langkah selanjutnya adalah pengguna menekan tombol enkripsi supaya citra tersebut diproses. Waktu proses enkripsi akan tampil pada bagian bawah nilai

X_n dan r dalam satuan detik. Citra yang sudah berhasil dienkripsi dapat disimpan dengan menggunakan tombol simpan citra.

Pada menu perbandingan citra, terdiri dari 3 menu yaitu menu citra *grayscale*, menu citra RGB dan menu kembali. Perbandingan citra dilakukan dengan cara menekan tombol pilih citra 1 dimana citra yang dipilih adalah citra awal atau citra aslinya. Citra awal tersebut akan dibandingkan dengan citra hasil dari enkripsi tersebut dengan cara menekan tombol pilih citra 2. Setelah kedua gambar ditampilkan, maka tekan tombol *compare*. Setelah proses *compare* dilakukan maka akan muncul histogram dari masing-masing citra beserta informasi lainnya yang terdiri dari nama *file*, ukuran *file*, lebar citra (dalam piksel), tinggi citra (dalam piksel), format *file*, kedalaman citra (dalam bit), tipe warna dan tingkat kesamaan citra.

Pada menu bantuan digunakan untuk menampilkan bantuan dalam aplikasi, sedangkan menu tentang berisi judul aplikasi dan biodata pembuat aplikasi.

3.2 Algoritma Enkripsi dan Dekripsi

Secara garis besar algoritma enkripsi dan dekripsi yang diimplementasikan dibagi menjadi tiga tahapan.

3.2.1 Tahap Pembangkitan *Keystream*

Pada tahap awal ini, pembangkitan *keystream* dilakukan dengan menggunakan algoritma Bernoulli Map seperti pada persamaan (2.1) kunci yang diminta akan digunakan sebagai parameter. Kunci nya yaitu nilai X_n dan r . Algoritma ini membangkitkan deret bilangan real, agar deret bilangan ini dapat digunakan sebagai *keystream*, maka bilangan-bilangan tersebut harus dikonversikan menjadi bilangan integer dengan rentang 0 sampai 255.

Proses tersebut dilakukan dengan cara mengabsolutkan barisan bilangan yang dihasilkan dari persamaan (2.1), lalu masing-masing dari bilangan tersebut dikalikan dengan 10000 dan dibulatkan kebawah (*floor*) untuk menghasilkan bilangan *integer*. Secara matematis fungsi konversi *integer* tersebut dapat dituliskan sebagai berikut :

$$E_n = X_n \times 10000 \quad (3.1)$$

$$F_n = |E_n| \quad (3.2)$$

Tahap selanjutnya adalah melakukan pembulatan kebawah, sehingga menghasilkan bilangan *integer* seperti yang dapat dilihat pada persamaan (3.2). Setelah didapatkan barisan bilangan integer, deret tersebut dipetakan pada rentang antara 0 sampai 255. Secara matematis fungsi pemetaan tersebut dapat dituliskan sebagai berikut :

$$K_n = F_n \bmod 256 \quad (3.3)$$

3.2.2 Tahap Enkripsi

Tahap enkripsi merupakan tahap citra semula atau *plain image* (P_n) diubah menjadi citra terenkripsi atau *chiper image* (C_n) dengan cara meng-XOR kan *pixel-pixel plain image* (P_n) terhadap *keystream* (K_n) yang telah dibangkitkan. Secara matematis fungsi enkripsi ini dapat dituliskan sebagai berikut :

$$C_n = P_n \oplus K_n \quad (3.4)$$

Keterangan :

C_n : *Cipher image* (citra terenkripsi)

P_n : *Plain image* (citra semula)

K_n : *Keystream*

3.2.3 Tahap Dekripsi

Tahap dekripsi memiliki proses yang sama dengan tahap enkripsi, hanya saja citra masukannya merupakan citra hasil enkripsi atau *chiper image* (C_n). Untuk mendekripsikan kembali citra semula dari hasil *chiper image* (C_n), maka dilakukan operasi XOR dari *pixel-pixel chiper image* (C_n) terhadap *keystream* (K_n) yang telah dibangkitkan. Secara matematis fungsi enkripsi ini dapat dituliskan sebagai berikut:

$$P_n = C_n \oplus K_n$$

$$= (P_n \oplus K_n) \oplus K_n$$

$$\begin{aligned}
 &= P_n \oplus (K_n \oplus K_n) \\
 &= P_n \oplus 0 \\
 &= P_n
 \end{aligned}$$

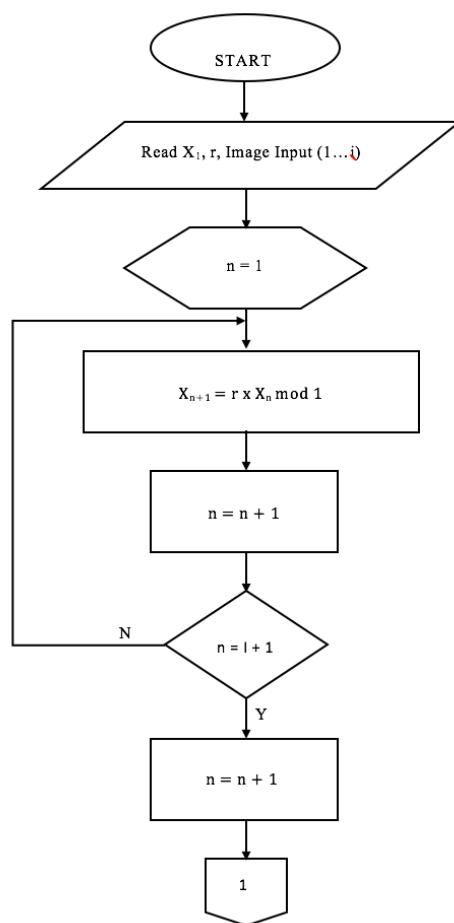
Keterangan :

C_n : *Cipher image* (citra terenkripsi)

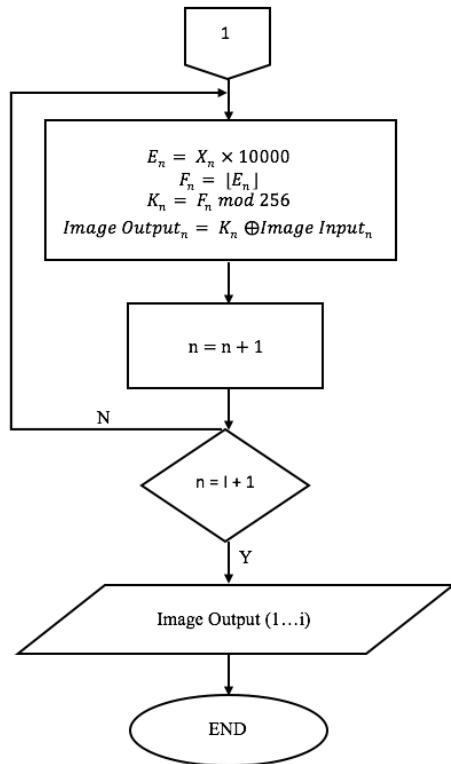
P_n : *Plain image* (citra semula)

K_n : *Keystream*

3.3 Flowchart Proses Enkripsi dan Dekripsi



Gambar 3.1 Flowchart Aplikasi Enkripsi dan Dekripsi



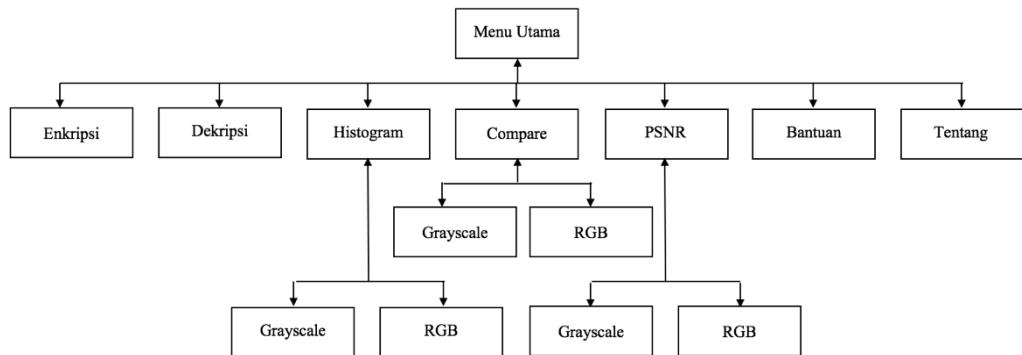
Gambar 3.1 Flowchart Enkripsi dan Dekripsi

3.4 Struktur Navigasi Aplikasi Enkripsi

Struktur navigasi merupakan bagian yang sangat penting dalam perancangan aplikasi. Struktur navigasi merupakan suatu alur yang digunakan dalam aplikasi yang dibuat yang dapat menjelaskan hubungan antar halaman pada sebuah program atau aplikasi. Struktur navigasi sangat penting karena struktur navigasi merupakan salah satu kunci untuk membuat aplikasi yang memudahkan untuk menejelajah di dalam aplikasi tersebut. Struktur navigasi disini berfungsi untuk menjelaskan alur aplikasi secara lebih singkat dan terurut.

Struktur aplikasi yang digunakan dalam aplikasi Enkripsi dan Dekripsi Citra ini adalah struktur navigasi hirarki karena mengandalkan percabangan untuk menampilkan data. Tampilan pada menu utama disebut dengan *master page* atau halaman utama satu dan mempunyai percabangan yang disebut dengan *slave page* atau halaman pendukung yang ketika dipilih akan menjadi halaman kedua dan

seterusnya. Struktur navigasi dari aplikasi ini terlihat pada Gambar 3.2.



Gambar 3.2 Struktur Navigasi Aplikasi Enkripsi

Untuk memperjelas struktur program diatas, maka secara garis besar dijelaskan langkah-langkahnya sebagai berikut: Awal tampilan aplikasi ini adalah tampilan menu utama. Pada saat pengguna masuk ke menu utama, pengguna akan menemukan pilihan menu enkripsi, menu dekripsi, menu *compare* dimana menu untuk untuk membandingkan citra di dalam menu itu terdapat *submenu grayscale* dan RGB, menu bantuan dan menu tentang.

Pada saat pengguna memilih menu enkripsi, maka pengguna diminta untuk memasukkan gambar asli yang akan dienkripsi, tetapi pengguna harus memasukkan nilai dari X_n dan r sebagai kunci enkripsi citra tersebut. Setelah itu, untuk mengenkripsi citra tersebut maka pengguna harus menekan tombol Enkripsi. Apabila proses enkripsi tersebut berhasil, langkah selanjutnya adalah menyimpan gambar hasil enkripsi tersebut dengan menekan tombol simpan.

Pada saat pengguna memilih menu dekripsi, maka pengguna diminta untuk memasukkan gambar yang telah dienkripsi lalu pengguna harus memasukkan nilai dari X_n dan r sebagai kunci sesuai dengan nilai yang dimasukkan saat proses enkripsi. Setelah itu, untuk mendekripsi citra tersebut maka pengguna harus menekan tombol Dekripsi. Apabila proses dekripsi tersebut berhasil, langkah selanjutnya adalah menyimpan gambar hasil dekripsi tersebut dengan menekan

tombol simpan.

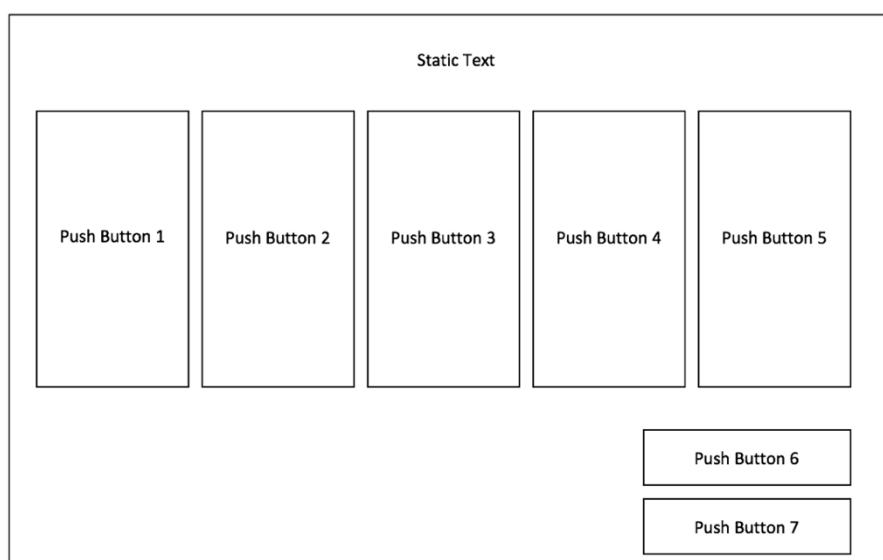
Pada saat pengguna memilih menu *compare*, maka pengguna diminta untuk memilih *submenu* dari citra yang akan dibandingkan. *_submenu* terdiri dari dua pilihan yaitu *submenu Grayscale* dan *submenu RGB*. *_submenu grayscale* berfungsi untuk membandingkan citra yang memiliki warna kelabu sedangkan *submenu RGB* berfungsi untuk membandingkan citra yang berwarna. Pada aplikasi ini setiap menu memiliki tombol kembali untuk kembali ke menu sebelumnya

3.5 Rancangan Tampilan Aplikasi Enkripsi dan Dekripsi Citra

Pada perancangan tampilan aplikasi ini, terdapat beberapa perancangan tampilan yaitu rancangan tampilan menu utama, menu enkripsi, menu dekripsi, menu *compare*, menu bantuan, dan menu tentang.

3.5.1 Rancangan Tampilan Menu Utama

Menu utama ini merupakan tampilan awal dari aplikasi yang dibuat. Didalam menu utama terdapat lima pilihan menu, yaitu : menu enkripsi, menu dekripsi, menu *histogram*, menu *compare*, menu PSNR, menu bantuan dan menu tentang. Menu utama terdiri dari satu buah *static text* dan tujuh buah *push button*. Rancangan tampilan menu utama ditunjukkan pada gambar 3.3



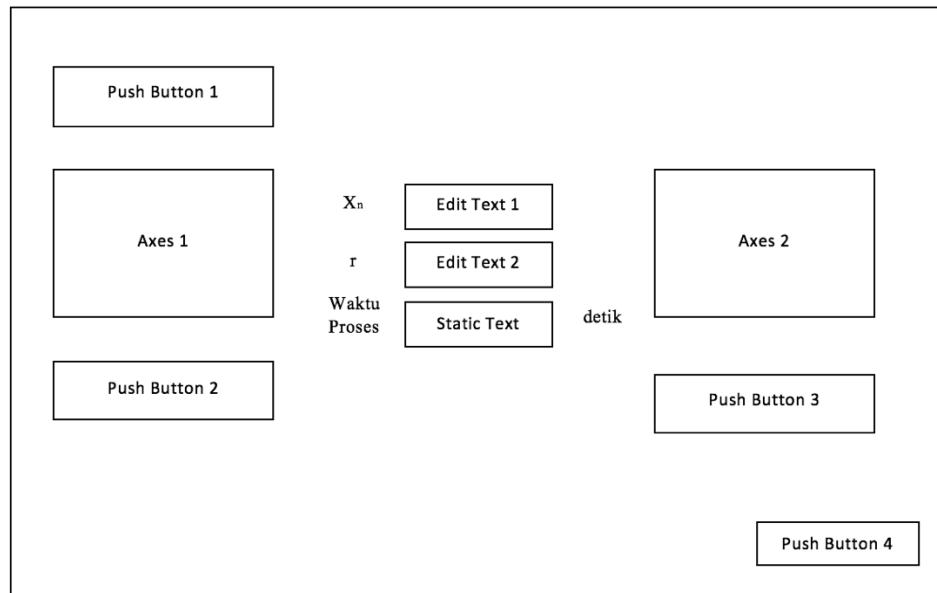
Gambar 3.3 Rancangan Tampilan Menu Utama

Berikut adalah keterangan Gambar 3.3:

- a. Static Text : Teks yang menampilkan judul dari aplikasi
- b. Push Button 1 : Tombol untuk masuk ke halaman enkripsi citra
- c. Push Button 2 : Tombol untuk masuk ke halaman dekripsi citra
- d. Push Button 3 : Tombol untuk masuk ke submenu histogram citra
- e. Push Button 4 : Tombol untuk masuk ke halaman untuk membandingkan citra (*compare*)
- f. Push Button 5 : Tombol untuk masuk ke halaman untuk membandingkan citra dengan PSNR
- g. Push Button 6 : Tombol untuk menampilkan halaman bantuan
- h. Push Button 7 : Tombol untuk menampilkan halaman tentang

3.5.2 Rancangan Tampilan Halaman Enkripsi Citra

Halaman enkripsi citra ini digunakan untuk melakukan enkripsi citra asli. Langkah pertama, pengguna harus memasukkan nilai dari X_n dan r sebagai kunci dari enkripsi citra. Setelah itu, tekan tombol enkripsi untuk melakukan proses enkripsi citra. Apabila citra telah berhasil dienkripsi, pengguna dapat menyimpan citra hasil enkripsi dengan menekan tombol simpan gambar. Halaman enkripsi citra terdiri dari dua buah *axes*, dua buah *edit text*, satu buah *static text* dan empat buah *push button*. Rancangan tampilan halaman enkripsi ditunjukkan pada gambar 3.4



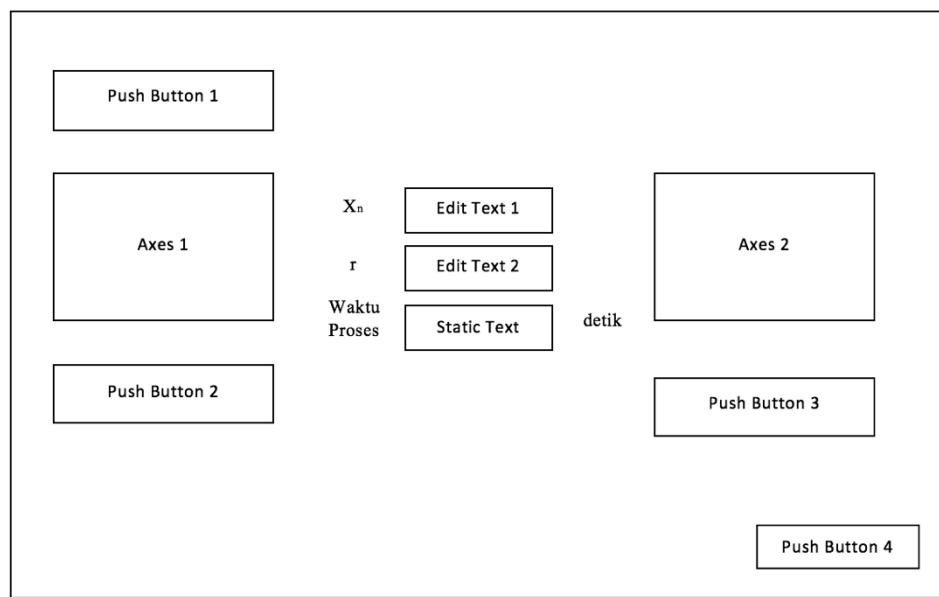
Gambar 3.4 Rancangan Tampilan Halaman Enkripsi

Berikut adalah keterangan Gambar 3.4:

- a. Axes 1 : Tempat untuk menampilkan citra sebelum proses enkripsi atau citra asli
- b. Axes 2 : Tempat untuk menampilkan citra setelah proses enkripsi atau citra hasil enkripsi
- c. Edit Text 1 : Tempat untuk memasukkan nilai kunci X_n
- d. Edit Text 2 : Tempat untuk memasukkan nilai kunci r
- e. Static Text : Tempat untuk menampilkan waktu proses yang dibutuhkan dalam proses enkripsi
- f. Push Button 1 : Tombol untuk memilih citra yang akan di enkripsi
- g. Push Button 2 : Tombol untuk melakukan proses enkripsi
- h. Push Button 3 : Tombol untuk menyimpan citra hasil proses enkripsi
- i. Push Button 4 : Tombol untuk kembali ke menu sebelumnya atau menu utama

3.5.3 Rancangan Tampilan Halaman Dekripsi Citra

Halaman dekripsi citra ini digunakan untuk melakukan proses dekripsi citra. Langkah pertama, pengguna harus membuka citra hasil enkripsi, selanjutnya pengguna harus memasukkan nilai dari X_n dan r yang sama seperti pada saat citra akan dikenkripsi. Setelah itu, tekan tombol dekripsi untuk melakukan proses dekripsi citra. Apabila citra telah berhasil didekripsi, pengguna dapat menyimpan citra hasil dekripsi dengan menekan tombol simpan gambar. Halaman dekripsi citra terdiri dari dua buah *axes*, dua buah *edit text*, satu buah *static text* dan empat buah *push button*. Rancangan tampilan halaman dekripsi ditunjukkan pada gambar 3.5



Gambar 3.5 Rancangan Tampilan Halaman Dekripsi

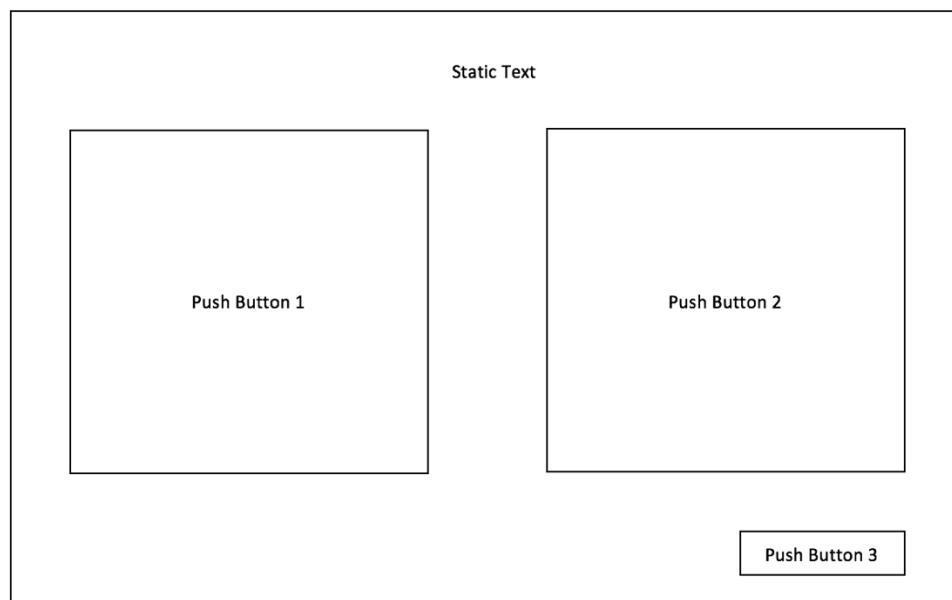
Berikut adalah keterangan Gambar 3.5:

- a. Axes 1 : Tempat untuk menampilkan citra hasil enkripsi yang akan didekripsi
- b. Axes 2 : Tempat untuk menampilkan citra yang sudah didekripsi
- c. Edit Text 1 : Tempat untuk memasukkan nilai kunci X_n
- d. Edit Text 2 : Tempat untuk memasukkan nilai kunci r

- e. Static Text : Tempat untuk menampilkan waktu proses yang dibutuhkan dalam proses dekripsi
- f. Push Button 1 : Tombol untuk memilih citra yang akan didekripsi
- g. Push Button 2 : Tombol untuk melakukan proses dekripsi
- h. Push Button 3 : Tombol untuk menyimpan citra hasil proses dekripsi
- i. Push Button 4 : Tombol untuk kembali ke menu sebelumnya atau menu utama

3.5.4 Rancangan Tampilan Halaman Submenu Histogram

Sebelum masuk ke halaman untuk melihat histogram citra, pengguna harus menentukan citra yang akan dilihat histogramnya pada submenu histogram. Terdapat dua pilihan tipe warna dari citra yaitu citra *grayscale* atau citra RGB. Halaman *submenu histogram* citra terdiri dari satu buah *static text* dan tiga buah *push button*. Rancangan tampilan halaman enkripsi ditunjukkan pada gambar 3.6



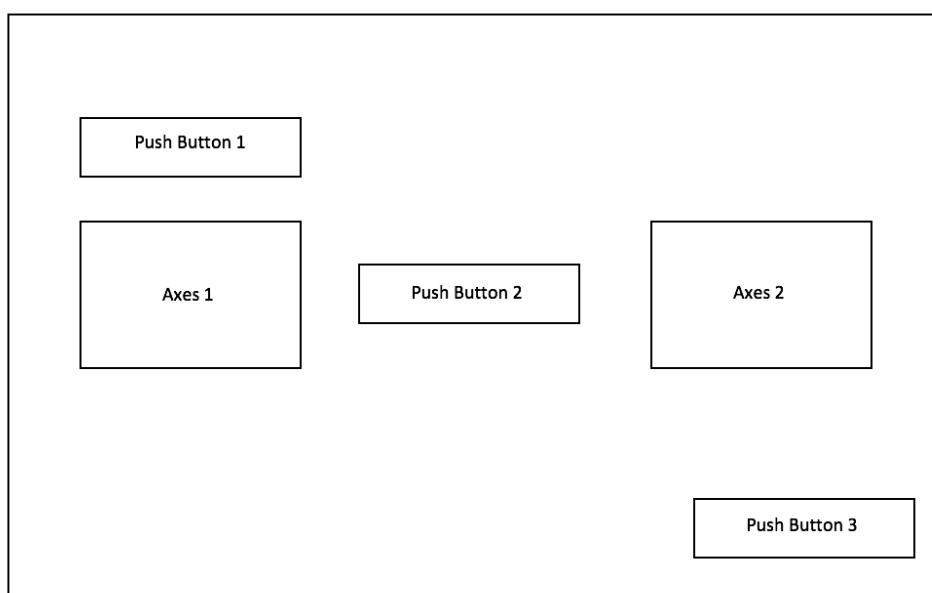
Gambar 3.6 Rancangan Tampilan Halaman Submenu Histogram

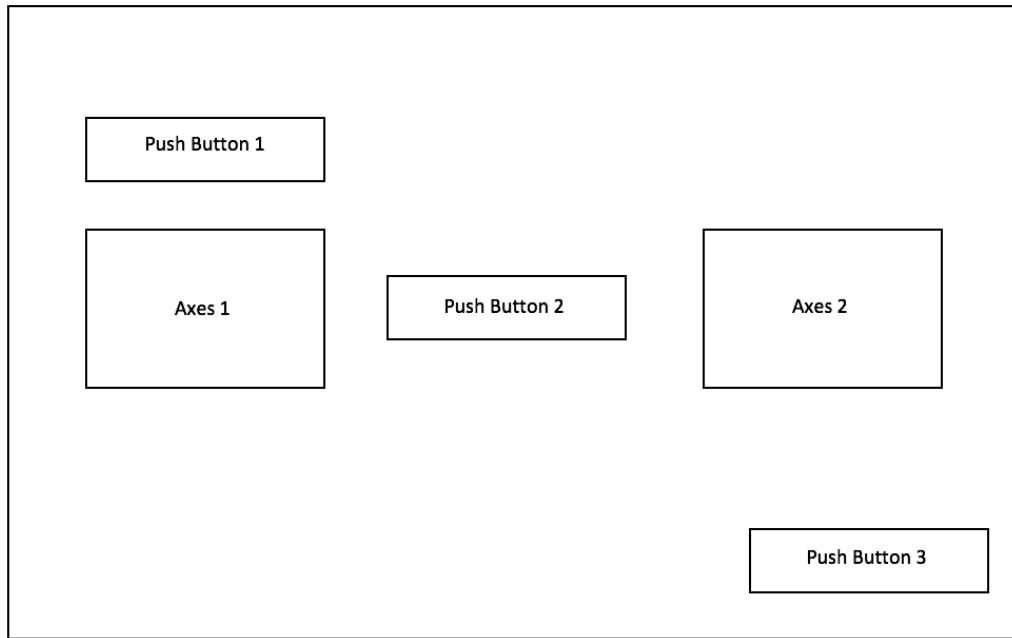
Berikut adalah keterangan Gambar 3.6:

- a. Static Text : Tempat untuk menampilkan tulisan yang berisi pertanyaan mengenai citra yang akan diketahui histogramnya
- b. Push Button 1 : Tombol untuk memilih histogram dari citra *grayscale*
- c. Push Button 2 : Tombol untuk memilih histogram dari citra RGB
- d. Push Button 3 : Tombol untuk kembali ke menu sebelumnya atau menu utama

3.5.4.1 Rancangan Tampilan Halaman Histogram Citra Grayscale

Halaman Histogram citra *grayscale* akan menampilkan citra histogram citra dua dimensi yang memiliki tipe warna *grayscale*. Langkah pertama, pengguna harus memilih citra yang akan diketahui histogramnya dengan cara menekan tombol pilih gambar. Setelah citra dipilih, maka citra yang dipilih akan tampil pada *axes1*. Untuk menampilkan histogram dari citra yang telah dipilih, pengguna harus menekan tombol cek histogram. Setelah langkah-langkah tersebut berhasil, maka histogram dari citra tersebut akan muncul pada *axes2*. Halaman histogram citra *grayscale* terdiri dari dua buah *axes* dan tiga buah *push button*. Rancangan tampilan halaman histogram citra *grayscale* ditunjukkan pada gambar 3.7





Gambar 3.7 Rancangan Tampilan Halaman Histogram Grayscale

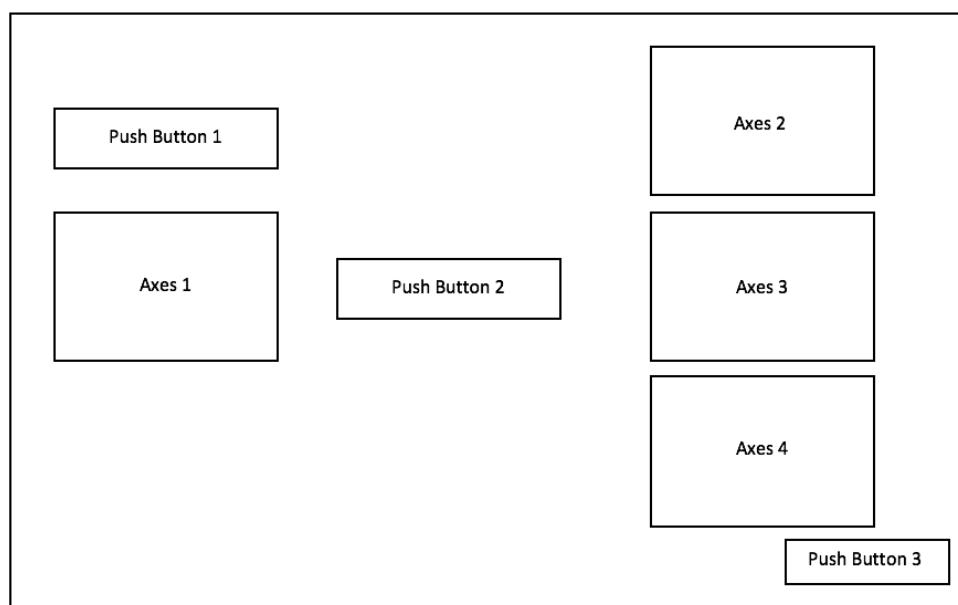
Berikut adalah keterangan Gambar 3.7:

- a. Axes 1 : Tempat untuk menampilkan citra yang akan dilihat histogramnya
- b. Axes 2 : Tempat untuk menampilkan histogram citra yang dipilih
- c. Push Button 1 : Tombol untuk memilih citra *grayscale* yang akan dilihat histogramnya
- d. Push Button 2 : Tombol untuk cek histogram dari citra yang dipilih
- e. Push Button 3 : Tombol untuk kembali ke menu sebelumnya atau *submenu* histogram

3.5.4.2 Rancangan Tampilan Halaman Histogram Citra RGB

Halaman Histogram citra RGB akan menampilkan histogram *red*, *green* dan *blue* dari citra yang memiliki tiga dimensi yang memiliki tipe warna *truecolor*. Langkah pertama, pengguna harus memilih citra yang akan diketahui histogramnya

dengan cara menekan tombol pilih gambar. Setelah citra dipilih, maka citra yang dipilih akan tampil pada *axes1*. Untuk menampilkan histogram dari citra yang telah dipilih, pengguna harus menekan tombol cek histogram. Setelah langkah-langkah tersebut berhasil, maka histogram dari citra tersebut akan muncul histogram dari lapisan *red* pada *axes2*, histogram pada lapisan *green* pada *axes3* dan histogram pada lapisan *blue* pada *axes4*. Halaman histogram citra RGB terdiri dari empat buah *axes* dan tiga buah *push button*. Rancangan tampilan halaman enkripsi ditunjukkan pada gambar 3.8



Gambar 3.8 Rancangan Tampilan Halaman Histogram RGB

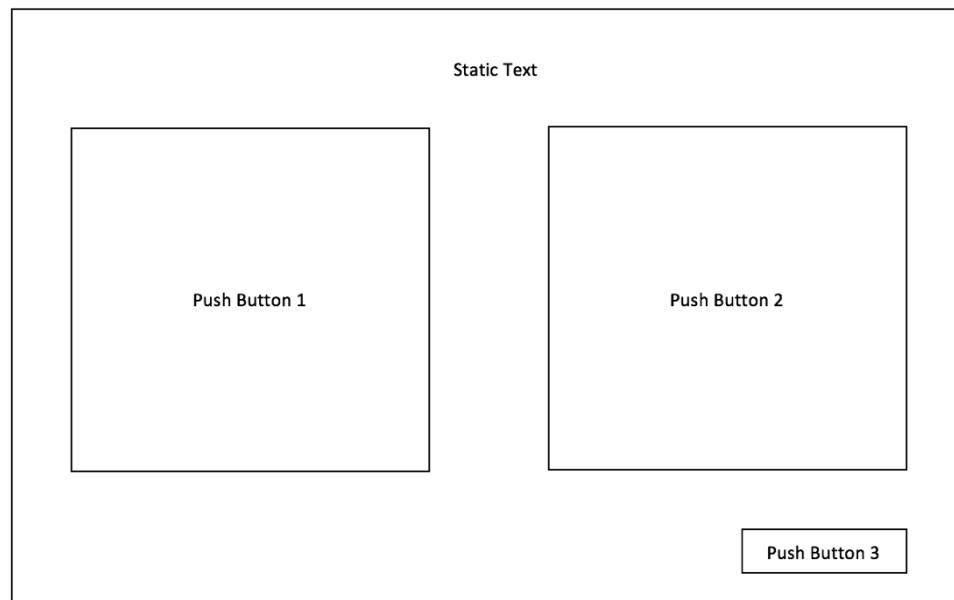
Berikut adalah keterangan Gambar 3.8:

- a. Axes 1 : Tempat untuk menampilkan citra yang akan dilihat histogramnya
- b. Axes 2 : Tempat untuk menampilkan histogram dari lapisan *red* dari citra yang dipilih
- c. Axes 3 : Tempat untuk menampilkan histogram dari lapisan *green* dari citra yang dipilih
- d. Axes 4 : Tempat untuk menampilkan histogram dari lapisan *blue* dari citra yang dipilih

- e. Push Button 1 : Tombol untuk memilih citra RGB atau citra warna yang memiliki tipe warna *truecolor* yang akan dilihat histogramnya
- f. Push Button 2 : Tombol untuk cek histogram dari citra yang dipilih
- g. Push Button 3 : Tombol untuk kembali ke menu sebelumnya atau *submenu histogram*

3.5.5 Rancangan Tampilan Halaman Submenu *Compare* atau Perbandingan

Sebelum masuk ke halaman *compare* yang berfungsi untuk melihat perbandingan citra, pengguna harus menentukan citra yang akan dibandingkan pada submenu *compare*. Terdapat dua pilihan tipe warna dari citra yaitu citra *grayscale* atau citra RGB. Halaman *submenu histogram* citra terdiri dari satu buah *static text* dan tiga buah *push button*. Rancangan tampilan halaman enkripsi ditunjukkan pada gambar 3.9



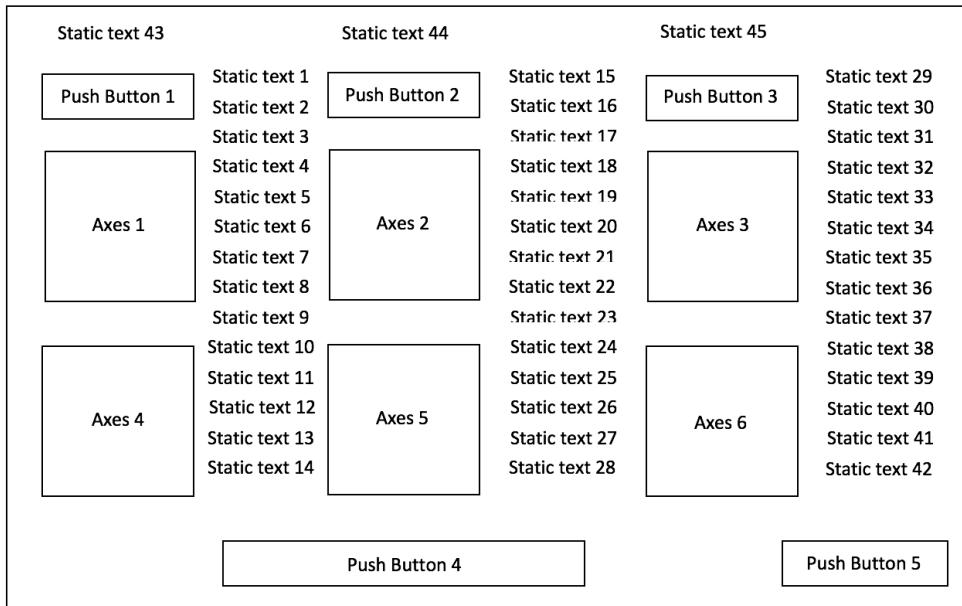
Gambar 3.9 Rancangan Tampilan Halaman Submenu *Compare*

Berikut adalah keterangan Gambar 3.9:

- a. Static Text : Tempat untuk menampilkan tulisan yang berisi pertanyaan mengenai citra yang akan dibandingkan
- b. Push Button 1 : Tombol untuk memilih citra berformat *grayscale* yang akan dibandingkan
- c. Push Button 2 : Tombol untuk memilih citra berformat *truecolor* yang akan dibandingkan
- d. Push Button 3 : Tombol untuk kembali ke menu sebelumnya atau menu utama

3.5.5.1 Rancangan Tampilan Halaman Perbandingan Citra Grayscale

Halaman perbandingan citra *grayscale* akan membandingkan tiga buah citra *grayscale* yaitu citra asli, citra hasil enkripsi dan citra hasil dekripsi. Langkah pertama, pengguna harus memilih citra asli dari citra yang telah didekripsi dengan cara menekan tombol pilih gambar pada bagian citra asli. Setelah citra dipilih, maka citra yang dipilih akan tampil pada *axes1*. Langkah kedua, pengguna harus memilih citra hasil enkripsi dengan cara menekan tombol pilih gambar pada bagian citra enkripsi. Setelah citra dipilih, maka citra yang dipilih akan tampil pada *axes2*. Langkah selanjutnya, pengguna harus memilih citra hasil dekripsi dengan cara menekan tombol pilih gambar pada bagian citra hasil dekripsi. Setelah citra dipilih, maka citra yang dipilih akan tampil pada *axes3*. Setelah kedua citra yang dipilih tampil, maka pengguna dapat menekan tombol bandingkan citra untuk menampilkan informasi dasar mengenai citra tersebut seperti nama file, ukuran, format, dan lainnya. Halaman perbandingan citra *grayscale* terdiri dari enam buah *axes*, lima buah *push button* dan empat puluh lima buah *static text*. Rancangan tampilan halaman perbandingan citra *grayscale* ditunjukkan pada gambar 3.10



Gambar 3.10 Rancangan Tampilan Halaman Perbandingan Citra Grayscale

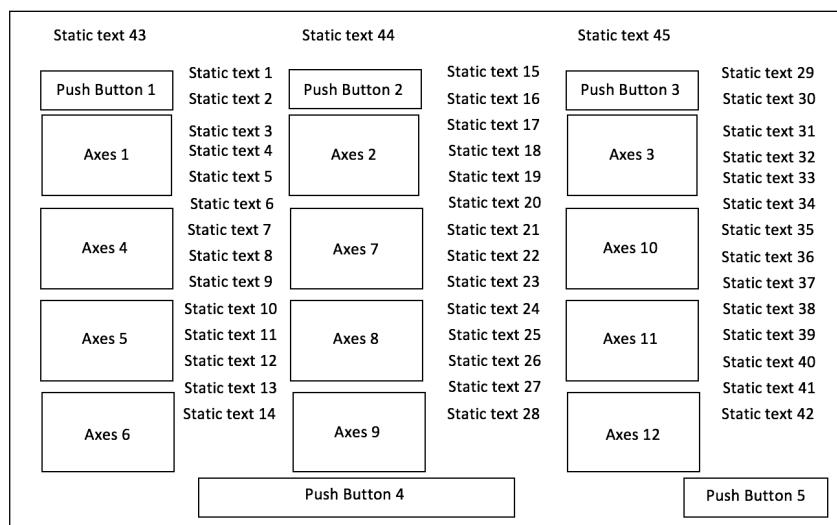
Berikut adalah keterangan Gambar 3.10:

- Push Button 1 : Tombol untuk memilih citra asli
- Push Button 2 : Tombol untuk memilih citra yang sudah dienkripsi
- Push Button 3 : Tombol untuk memilih citra hasil dekripsi
- Push Button 4 : Tombol untuk melakukan proses perbandingan citra.
- Push Button 5 : Tombol untuk kembali ke menu sebelumnya atau submenu *compare*
- Axes 1 : Tempat untuk menampilkan citra asli
- Axes 2 : Tempat untuk menampilkan citra enkripsi
- Axes 3 : Tempat untuk menampilkan citra dekripsi
- Axes 4 : Tempat untuk menampilkan histogram dari citra asli
- Axes 5 : Tempat untuk menampilkan histogram citra enkripsi
- Axes 6 : Tempat untuk menampilkan histogram citra dekripsi
- Static Text : Pada halaman perbandingan citra *grayscale* terdapat 45 (empat puluh lima) static text yang berisi

informasi dasar mengenai citra tersebut seperti nama file, ukuran, format, dan lainnya.

3.5.5.2 Rancangan Tampilan Halaman Perbandingan Citra RGB

Halaman perbandingan citra berwarna (RGB) akan membandingkan tiga buah citra berwarna (RGB) yaitu citra asli, citra hasil enkripsi dan citra hasil dekripsi. Langkah pertama, pengguna harus memilih citra asli dari citra yang telah didekripsi dengan cara menekan tombol pilih gambar pada bagian citra asli. Setelah citra dipilih, maka citra yang dipilih akan tampil pada *axes1*. Langkah kedua, pengguna harus memilih citra hasil enkripsi dengan cara menekan tombol pilih gambar pada bagian citra enkripsi. Setelah citra dipilih, maka citra yang dipilih akan tampil pada *axes2*. Langkah selanjutnya, pengguna harus memilih citra hasil dekripsi dengan cara menekan tombol pilih gambar pada bagian citra hasil dekripsi. Setelah citra dipilih, maka citra yang dipilih akan tampil pada *axes3*. Setelah ketiga citra yang dipilih tampil, maka pengguna dapat menekan tombol bandingkan citra untuk menampilkan informasi dasar mengenai citra tersebut seperti nama file, ukuran, format, dan lainnya. Halaman perbandingan citra berwarna (RGB) terdiri dari dua belas buah *axes*, lima buah *push button* dan empat puluh lima buah *static text*. Rancangan tampilan halaman perbandingan citra berwarna ditunjukkan pada gambar 3.11



Gambar 3.11 Rancangan Tampilan Halaman Perbandingan Citra Berwarna

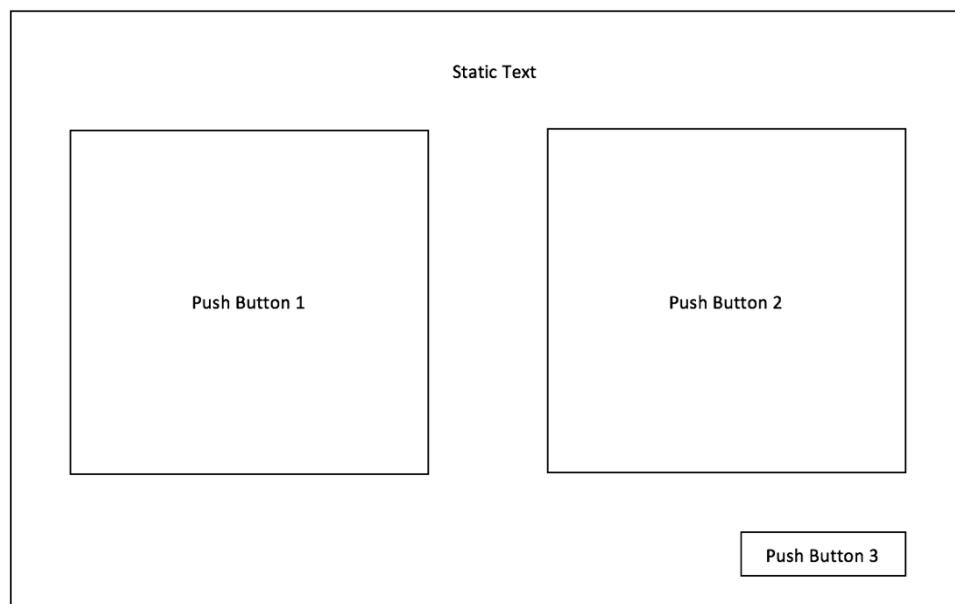
Berikut adalah keterangan Gambar 3.11:

- a. Push Button 1 : Tombol untuk memilih citra asli
- b. Push Button 2 : Tombol untuk memilih citra yang sudah dienkripsi
- c. Push Button 3 : Tombol untuk memilih citra hasil dekripsi
- d. Push Button 4 : Tombol untuk melakukan proses perbandingan citra.
- e. Push Button 5 : Tombol untuk kembali ke menu sebelumnya atau submenu *compare*
- f. Axes 1 : Tempat untuk menampilkan citra asli
- g. Axes 2 : Tempat untuk menampilkan citra enkripsi
- h. Axes 3 : Tempat untuk menampilkan citra dekripsi
- i. Axes 4 : Tempat untuk menampilkan histogram lapisan merah (*red*) citra asli
- j. Axes 5 : Tempat untuk menampilkan histogram lapisan hijau (*green*) citra asli
- k. Axes 6 : Tempat untuk menampilkan histogram lapisan biru (*blue*) citra asli
- l. Axes 7 : Tempat untuk menampilkan histogram lapisan merah (*red*) citra enkripsi
- m. Axes 8 : Tempat untuk menampilkan histogram lapisan hijau (*green*) citra enkripsi
- n. Axes 9 : Tempat untuk menampilkan histogram lapisan biru (*blue*) citra enkripsi
- o. Axes 10 : Tempat untuk menampilkan histogram lapisan merah (*red*) citra dekripsi
- p. Axes 11 : Tempat untuk menampilkan histogram lapisan hijau (*green*) citra dekripsi
- q. Axes 12 : Tempat untuk menampilkan histogram lapisan biru (*blue*) citra dekripsi
- r. Static Text : Pada halaman perbandingan citra *grayscale* terdapat 45 (empat puluh lima) static text yang berisi

informasi dasar mengenai citra tersebut seperti nama file, ukuran, format, dan lainnya.

3.5.6 Rancangan Tampilan Halaman PSNR

Sebelum masuk ke halaman *Peak Signal Noise Ratio* (PSNR) yang berfungsi untuk melihat perbandingan citra dengan *Mean Square Error* (MSE), pengguna harus menentukan citra yang akan dibandingkan pada submenu PSNR. Terdapat dua pilihan tipe warna dari citra yaitu citra *grayscale* atau citra RGB. Halaman *submenu histogram* citra terdiri dari satu buah *static text* dan tiga buah *push button*. Rancangan tampilan halaman enkripsi ditunjukkan pada gambar 3.12



Gambar 3.12 Rancangan Tampilan Halaman Submenu PSNR

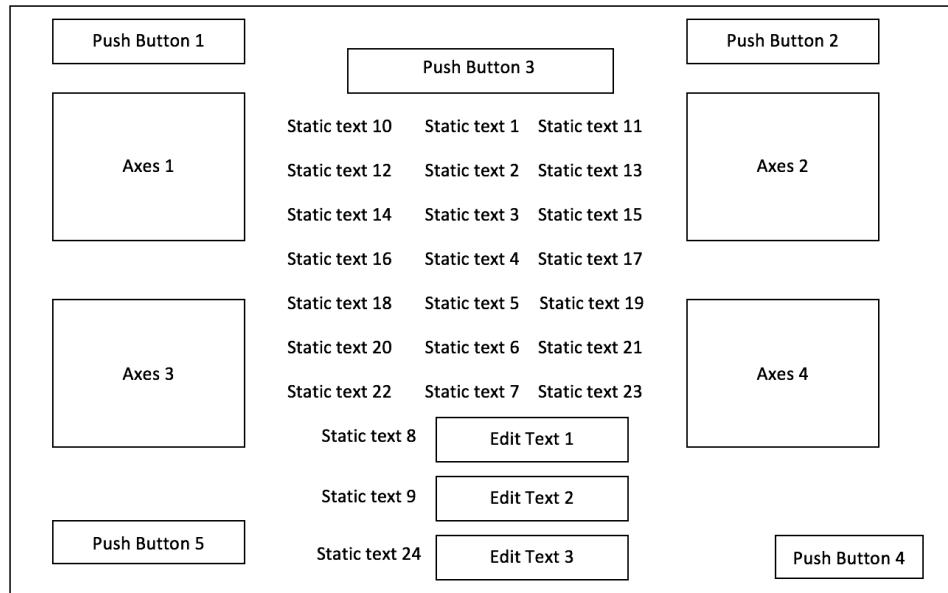
Berikut adalah keterangan Gambar 3.12:

- a. Static Text : Tempat untuk menampilkan tulisan yang berisi pertanyaan mengenai citra yang akan dibandingkan
- b. Push Button 1 : Tombol untuk memilih citra berformat *grayscale* yang akan dibandingkan
- c. Push Button 2 : Tombol untuk memilih citra berformat *truecolor* yang akan dibandingkan

- d. Push Button 3 : Tombol untuk kembali ke menu sebelumnya atau menu utama

3.5.6.1 Rancangan Tampilan Halaman PSNR Citra Grayscale

Halaman *Peak Signal Noise Ratio* (PSNR) citra *grayscale* akan memperlihatkan citra dua dimensi yang memiliki tipe warna *grayscale* melalui *Mean Square Error* (MSE). Langkah pertama, pengguna harus memilih citra asli dari citra yang telah didekripsi dengan cara menekan tombol pilih gambar 1. Setelah citra dipilih, maka citra yang dipilih akan tampil pada *axes1*. Langkah kedua, pengguna harus memilih citra yang telah didekripsi dengan cara menekan tombol pilih gambar 2. Setelah citra dipilih, maka citra yang dipilih akan tampil pada *axes2*. Setelah kedua citra yang dipilih tampil, maka pengguna dapat menekan tombol bandingkan citra untuk menampilkan informasi dasar mengenai citra tersebut seperti nama file, ukuran, format, dan lainnya serta menunjukkan hasil dari *Mean Square Error* (MSE) dan *Peak Signal Noise Ratio* (PSNR). Halaman perbandingan citra *grayscale* terdiri dari empat buah *axes*, lima buah *push button*, tiga buah *edit text* dan dua puluh empat buah *static text*. Rancangan tampilan halaman perbandingan citra *grayscale* ditunjukkan pada gambar 3.13



Gambar 3.13 Rancangan Tampilan Halaman PSNR *Grayscale*

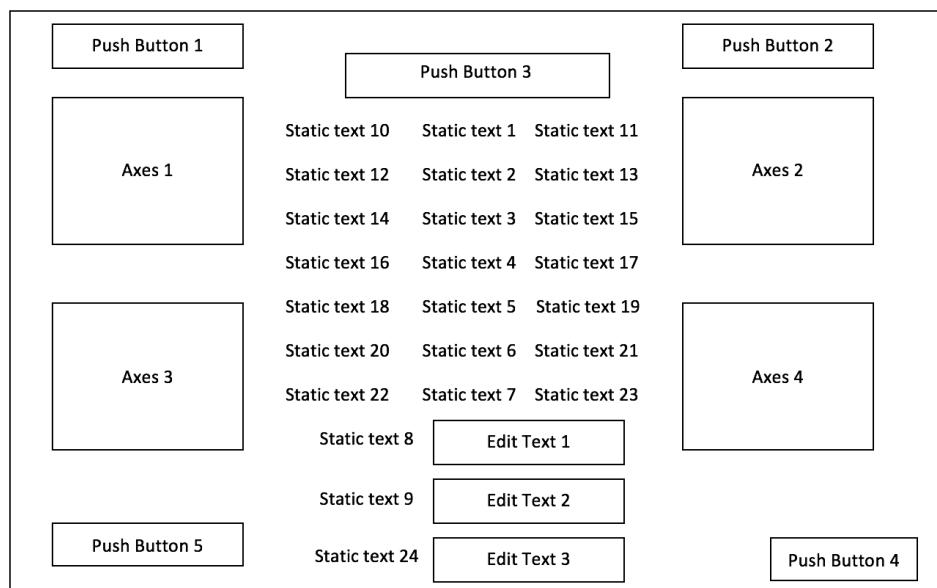
Berikut adalah keterangan Gambar 3.13:

- Push Button 1 : Tombol untuk memilih citra asli yang akan dibandingkan
- Push Button 2 : Tombol untuk memilih citra hasil dekripsi yang akan dibandingkan
- Push Button 3 : Tombol untuk melakukan proses perbandingan citra.
- Push Button 4 : Tombol untuk kembali ke menu sebelumnya atau submenu PSNR
- Push Button 5 : Tombol untuk membersihkan layar
- Axes 1 : Tempat untuk menampilkan citra asli berformat *grayscale* citra asli yang akan dibandingkan
- Axes 2 : Tempat untuk menampilkan citra hasil dekripsi berformat *grayscale* yang akan dibandingkan
- Axes 3 : Tempat untuk menampilkan hasil MSE dari citra asli
- Axes 4 : Tempat untuk menampilkan hasil MSE dari citra hasil dekripsi

- j. Static Text : Pada halaman perbandingan citra *grayscale* terdapat dua puluh tiga static text yang berisi informasi dasar mengenai citra tersebut seperti nama file, ukuran, format, dan lainnya.
- k. Edit Text 1 : Tempat untuk menampilkan hasil dari perhitungan *Mean Square Error* (MSE)
- l. Edit Text 2 : Tempat untuk menampilkan hasil dari perhitungan *Peak Signal Noise Ratio* (PSNR)
- m. Edit Text 3 : Tempat untuk menampilkan hasil dari perhitungan tingkat kesamaan berdasarkan persentasenya

3.5.6.2 Rancangan Tampilan Halaman PSNR Citra RGB

Halaman *Peak Signal Noise Ratio* (PSNR) citra berwarna sama seperti halaman *Peak Signal Noise Ratio* (PSNR) citra grayscale. Perbedaannya terletak pada citra yang akan diolah. Halaman perbandingan citra berwarna (RGB) terdiri dari empat buah *axes*, lima buah *push button*, tiga buah *edit text* dan dua puluh empat buah *static text*. Rancangan tampilan halaman perbandingan citra *grayscale* ditunjukkan pada gambar 3.14



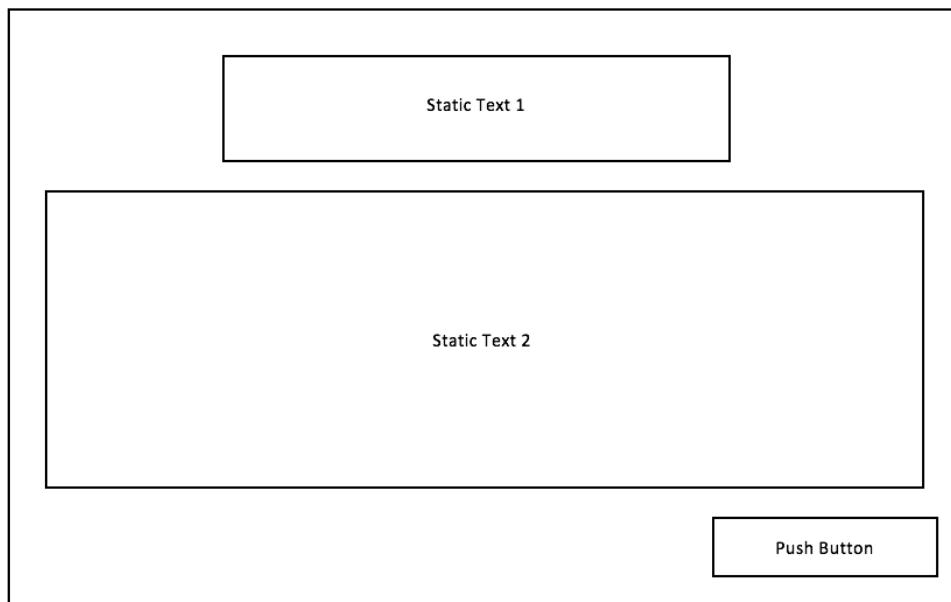
Gambar 3.14 Rancangan Tampilan Halaman PSNR Citra Berwarna

Berikut adalah keterangan Gambar 3.14:

- a. Push Button 1 : Tombol untuk memilih citra asli yang akan dibandingkan
- b. Push Button 2 : Tombol untuk memilih citra hasil dekripsi yang akan dibandingkan
- c. Push Button 3 : Tombol untuk melakukan proses perbandingan citra.
- d. Push Button 4 : Tombol untuk kembali ke menu sebelumnya atau submenu PSNR
- e. Push Button 5 : Tombol untuk membersihkan layar
- f. Axes 1 : Tempat untuk menampilkan citra asli berformat *truecolor* yang akan dibandingkan
- g. Axes 2 : Tempat untuk menampilkan citra hasil dekripsi berformat *truecolor* yang akan dibandingkan
- h. Axes 3 : Tempat untuk menampilkan hasil MSE dari citra asli
- i. Axes 4 : Tempat untuk menampilkan hasil MSE dari citra hasil dekripsi
- j. Static Text : Pada halaman perbandingan citra *grayscale* terdapat dua puluh tiga static text yang berisi informasi dasar mengenai citra tersebut seperti nama file, ukuran, format, dan lainnya.
- k. Edit Text 1 : Tempat untuk menampilkan hasil dari perhitungan *Mean Square Error* (MSE)
- l. Edit Text 2 : Tempat untuk menampilkan hasil dari perhitungan *Peak Signa Noisel Ratio* (PSNR)
- m. Edit Text 3 : Tempat untuk menampilkan hasil dari perhitungan tingkat kesamaan berdasarkan persentasenya

3.5.7 Rancangan Tampilan Halaman Bantuan

Halaman bantuan menjelaskan tentang tata cara penggunaan aplikasi. Halaman bantuan terdiri dari dua buah *static text* dan satu buah *push button*. Rancangan tampilan halaman bantuan ditunjukkan pada gambar 3.15



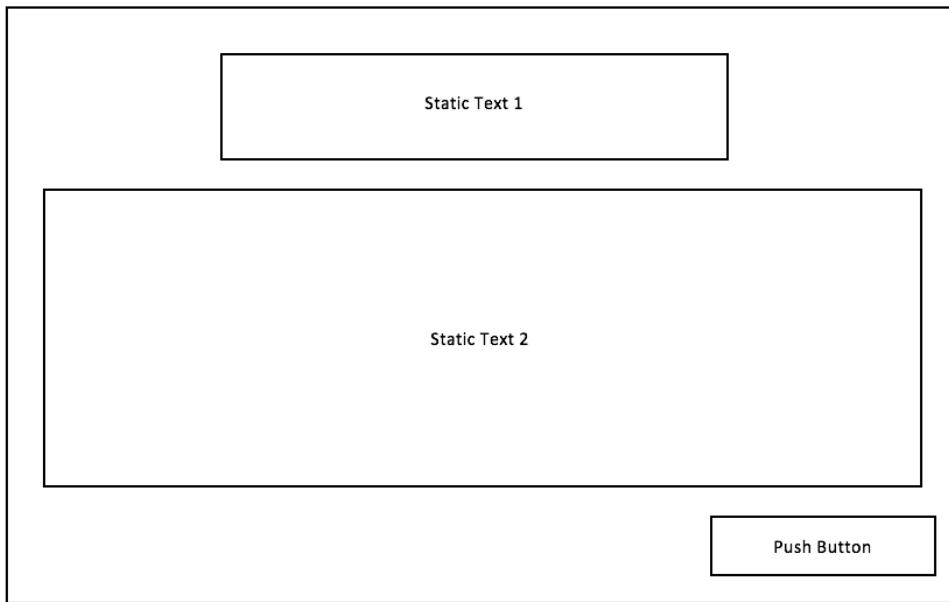
Gambar 3.15 Rancangan Tampilan Halaman Bantuan

Berikut adalah keterangan Gambar 3.15:

- a. Push Button 1 : Tombol untuk kembali ke menu sebelumnya atau menu utama
- b. Static Text 1 : Teks untuk menampilkan judul halaman atau teks bantuan
- c. Static Text 2 : Teks untuk menampilkan cara penggunaan aplikasi berdasarkan halaman-halamannya.

3.5.8 Rancangan Tampilan Halaman Tentang

Halaman tentang menjelaskan tentang aplikasi yang dibuat beserta informasi pembuat program. Halaman tentang terdiri dari dua buah *static text* dan satu buah *push button*. Rancangan tampilan halaman tentang ditunjukkan pada gambar 3.16



Gambar 3.16 Rancangan Tampilan Halaman Tentang

Berikut adalah keterangan Gambar 3.16:

- a. Push Button 1 : Tombol untuk kembali ke menu sebelumnya atau menu utama
- b. Static Text 1 : Teks untuk menampilkan judul halaman atau teks tentang
- c. Static Text 2 : Teks untuk menampilkan keterangan mengenai aplikasi

3.6 Pembuatan Aplikasi

Pembuatan aplikasi enkripsi dan dekripsi citra menggunakan perangkat lunak Matlab dengan menerapkan rancangan form aplikasi sebelumnya. Matlab merupakan suatu Bahasa pemrograman yang dapat digunakan untuk melakukan proses pengolahan citra. Terdapat tiga tahap yaitu tahap pembuatan tampilan aplikasi, menu aplikasi dan kode sumber aplikasi.

3.6.1 Pembuatan Tampilan Aplikasi

Tampilan aplikasi merupakan tampilan antarmuka (*interface*) adalah tampilan yang akan muncul ketika aplikasi dijalankan oleh pengguna, sehingga

memudahkan proses interaksi antara pengguna dengan komputer.

Pembuatan tampilan aplikasi dilakukan pada *guide* Matlab dengan ekstensi .fig dengan cara mengetikkan *guide* pada *command window*, selanjutnya dipilih *template* tampilan pada jendela *guide quick start* dan dimulai pembuatan tampilan berdasarkan rancangan tampilan. Pembuatannya dengan cara menggunakan *drag & drop tools* yang akan digunakan sesuai dengan kebutuhan aplikasi tersebut. Tampilan salah satu jendela dari aplikasi yang ditampilkan pada *guide* Matlab ditunjukkan pada Gambar 3.17



Gambar 3.17 Tampilan Aplikasi Enkripsi

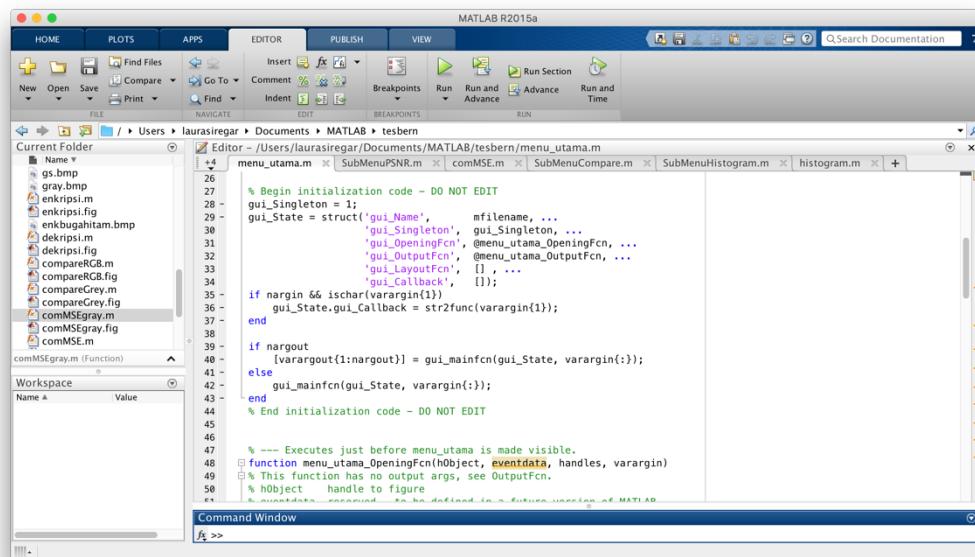
3.6.2 Pembuatan Menu Aplikasi

Menu pada aplikasi ini digunakan untuk memanggil program lain yaitu program enkripsi, program dekripsi, program *compare* atau membandingkan citra, program perbandingan citra grayscale, program perbandingan citra RGB, program bantuan, dan program tentang menggunakan *UI Control push button* yang merupakan komponen pada *guide*.

3.6.3 Penulisan Kode Sumber Aplikasi

Kode sumber atau kode program merupakan suatu rangkaian penyatuan atau deklarasi yang ditulis dalam bahasa pemrograman komputer yang melakukan suatu fungsi yang spesifik.

Penulisan kode sumber aplikasi ini dilakukan pada m-file dengan ekstensi m dengan cara mengetikkan edit diikuti dengan nama m-file pada command window, selanjutnya m-file akan ditampilkan pada editor Matlab. Tampilan salah satu m-file dari aplikasi ini yang ditampilkan pada editor Matlab ditunjukkan pada Gambar 3.18



The screenshot shows the MATLAB R2015a interface with the 'EDITOR' tab selected. The current folder browser shows files like gs.bmp, gray.bmp, and various .fig and .m files. The workspace browser shows 'comMSEgray.m (Function)'. The main editor area displays the code for 'menu_utama.m'. The code is as follows:

```
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name', ...
    'gui_Singleton', 'gui_Singleton', ...
    'gui_OpeningFcn', @menu_utama_OpeningFcn, ...
    'gui_OutputFcn', @menu_utama_OutputFcn, ...
    'gui_LayoutFcn', [], ...
    'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargin
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% ---- Executes just before menu_utama is made visible.
function menu_utama_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn
% hObject    handle to figure
%
```

Gambar 3.18 Tampilan M-File Pada *Editor Matlab*

BAB IV

ANALISIS HASIL UJI COBA

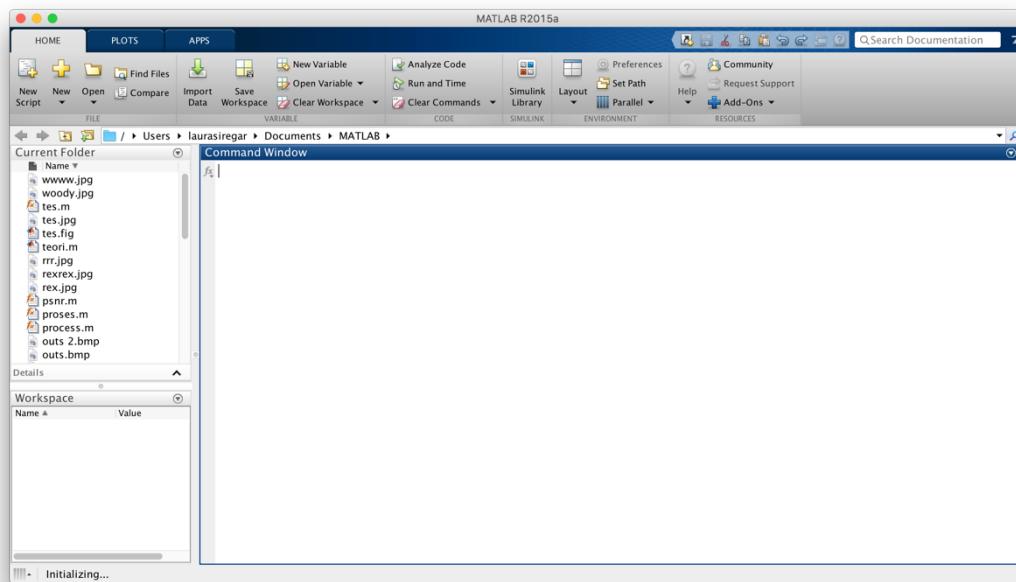
Pada bab ini akan dijelaskan tentang implementasi aplikasi dan hasil uji coba aplikasi.

4.1 Implementasi Aplikasi

Pengimplementasian aplikasi dilakukan pada komputer dengan spesifikasi perangkat keras: *processor* Intel Core i5 CPU @ 2.6GHz, kapasitas RAM 8 GB, VGA Intel Iris 1536MB, *startup disk* Macintosh HD.

Hal utama yang perlu dilakukan untuk melakukan pengimplementasian aplikasi adalah dengan menginstal bahasa pemrograman Matlab pada komputer. Setelah itu jalankan aplikasi Matlab tersebut.

Tahap pertama setelah aplikasi dijalankan, maka akan muncul jendela kerja Matlab, yang terlihat pada Gambar 4.1



Gambar 4.1 Tampilan Awal Jendela Kerja Matlab

Tahap selanjutnya jika telah membuat tampilan aplikasi dan mengetikan semua

kode sumber, pengguna dapat menjalankan program Matlab tersebut dengan cara memastikan direktori penyimpanan *file* sudah terdapat didalam daftar pencarian direktori Matlab, kemudian ketikkan nama program yang akan dieksekusi pada *command window*, pada kasus aplikasi ini nama program yang akan dieksekusi adalah menu_utama. Tampilan jendela menu utama terlihat seperti pada Gambar 4.2.

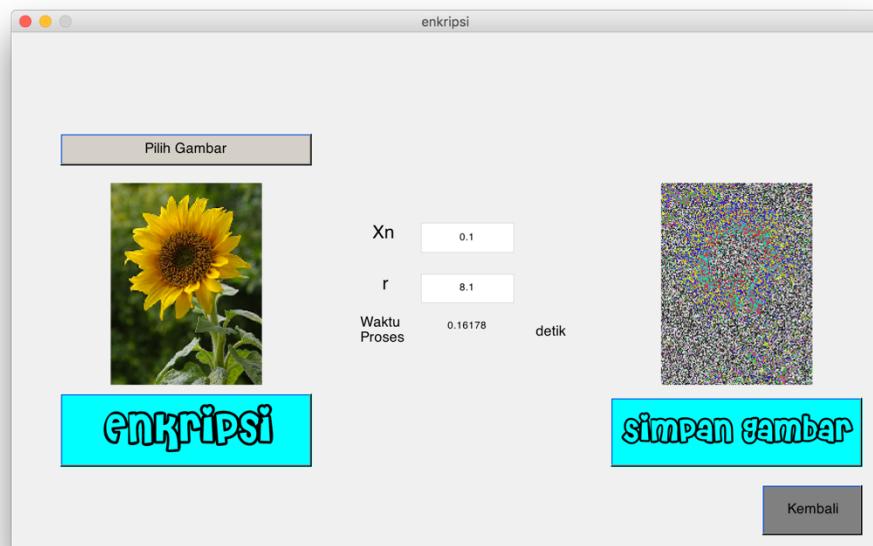


Gambar 4.2 Tampilan Menu Utama Aplikasi

Pada aplikasi ini terdapat tujuh pilihan menu, yaitu : Enkripsi, Dekripsi, Histogram, Perbandingan Citra (*compare*), *Peak Signal Noise Ratio* (PSNR), Bantuan, dan Tentang. Menu Enkripsi berfungsi untuk melakukan proses enkripsi. Menu Dekripsi berfungsi untuk melakukan proses dekripsi. Menu *Histogram* berfungsi untuk melihat grafik citra. Menu Perbandingan Citra (*compare*) berfungsi untuk membandingkan citra. Pada menu tersebut terdapat dua pilihan submenu, yaitu Citra Grayscale dan Citra Berwarna (RGB). Menu PSNR berfungsi untuk membandingkan citra dengan menggunakan *Mean Squared Error* (MSE) Pada menu PSNR juga terdapat dua pilihan submenu, yaitu Citra *Grayscale* dan Citra berwarna (RGB). Menu Bantuan berfungsi untuk menjelaskan cara menggunakan aplikasi dan Menu tentang berfungsi untuk menjelaskan aplikasi yang dibuat

beserta tentang pembuat program.

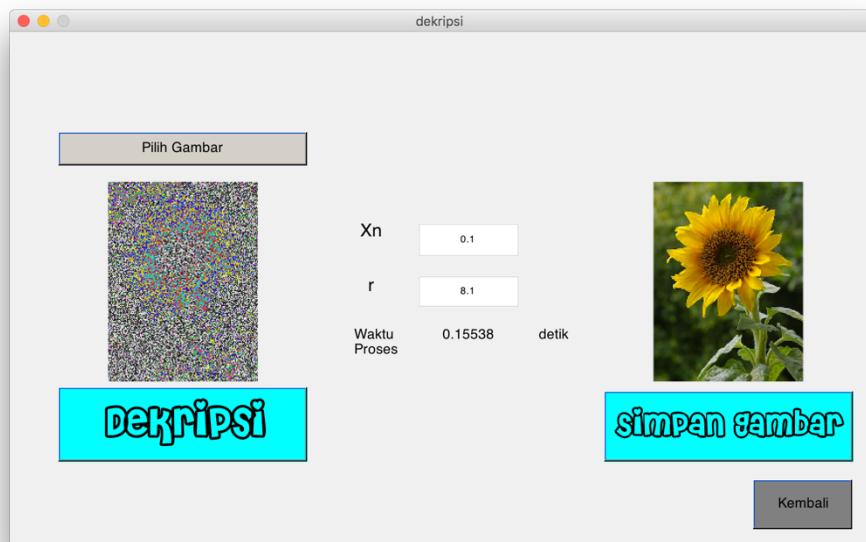
Tahap awal untuk memulai aplikasi ini, maka *user* harus memilih menu Enkripsi pada menu utama, kemudian akan muncul jendela Enkripsi seperti yang terlihat pada Gambar 4.3



Gambar 4.3 Tampilan Jendela Menu Enkripsi

Untuk melakukan proses enkripsi, *user* diminta untuk menentukan nilai kunci X_n dan r sebagai nilai awal (*initial condition*) yang digunakan pada persamaan Bernoulli Map. Selanjutnya pengguna diminta untuk membuka citra yang akan dienkripsi (*plain image*), lalu pengguna diminta menekan tombol enkripsi untuk memulai proses enkripsi, maka akan dihasilkan suatu citra dengan *pixel* acak (*chiper image*) yang didapat dengan melakukan operasi XOR terhadap *pixel plain image* dengan bit-bit kunci (*keystream*) yang dibangkitkan oleh persamaan Bernoulli Map. Kemudian *chiper image* yang telah dihasilkan dapat disimpan dengan menekan tombol simpan.

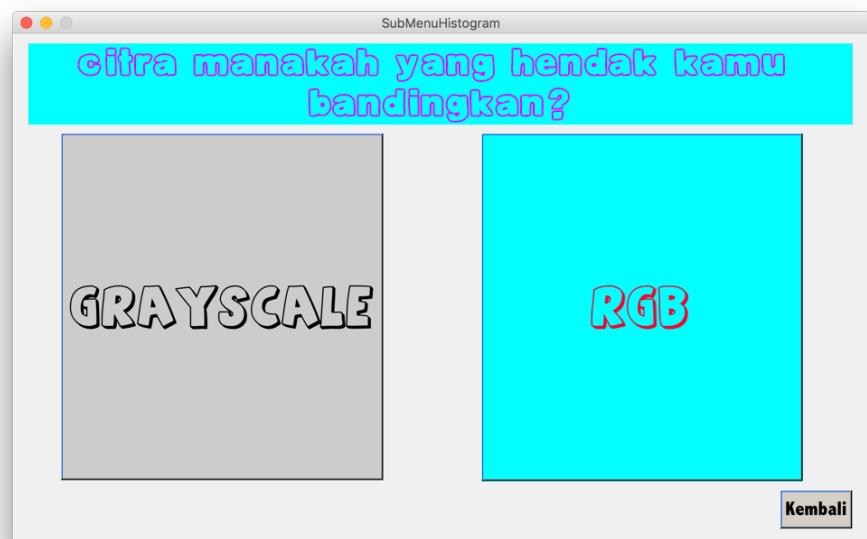
Tahap selanjutnya untuk mendekripsi *chiper image*, *user* harus memilih menu Dekripsi pada menu program utama, kemudian akan tampil jendela Dekripsi yang terlihat seperti pada Gambar 4.4.



Gambar 4.4 Tampilan Jendela Menu Dekripsi

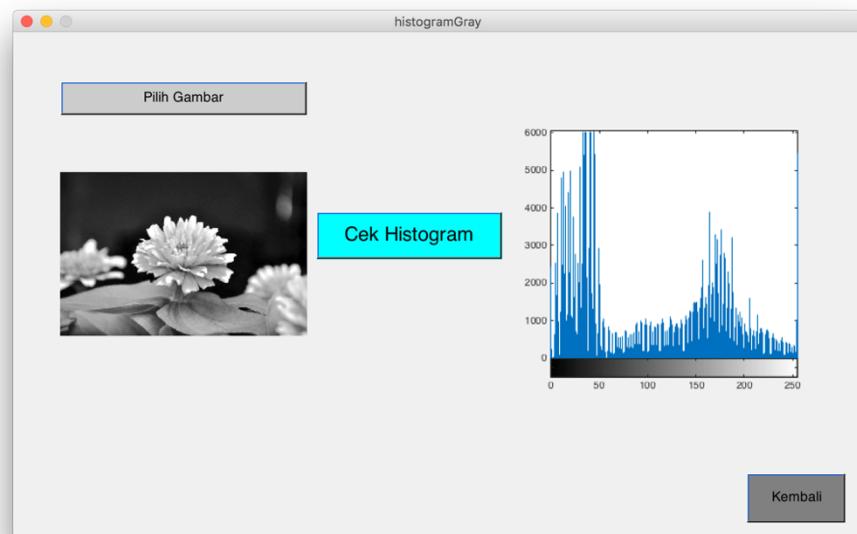
Pada tahap proses dekripsi, pengguna diminta untuk menentukan nilai kunci X_n dan r sebagai nilai awal (*initial condition*). Nilai awal yang dimasukan pada proses dekripsi ini harus sama persis dengan nilai kunci yang dimasukkan pada saat proses enkripsi, karena apabila nilai yang dimasukkan berbeda, akan menghasilkan keluaran yang berbeda dengan citra awal sebelum dienkripsi. Selanjutnya *user* diminta untuk membuka *chiper image* yang akan didekripsi, dan *user* dapat memulai proses dekripsi dengan menekan tombol dekripsi. Tahap proses ini akan menghasilkan suatu citra yang sama seperti citra awal (*plain image*) yang didapat dengan melakukan operasi XOR terhadap *pixel chiper image* dengan bit-bit kunci (*keystream*) yang dibangkitkan oleh persamaan Bernoulli Map. *Plain Image* yang telah dihasilkan akan disimpan dengan menekan tombol simpan.

Tahap selanjutnya, untuk melihat histogram, pengguna harus memilih menu *grayscale* atau *RGB* pada submenu histogram, kemudian akan tampil jendela submenu histogram seperti pada Gambar 4.5



Gambar 4.5 Tampilan Jendela Submenu Histogram

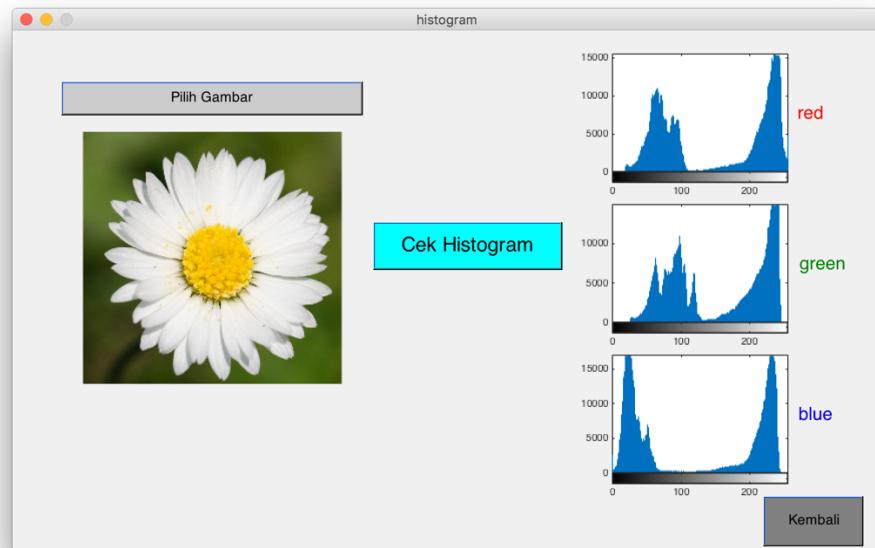
Tahap selanjutnya apabila pengguna akan mengetahui histogram dari citra 2 dimensi yang memiliki tipe warna *grayscale* maka pengguna dapat menekan tombol *grayscale*. Setelah itu akan tampil jendela histogram *grayscale* seperti terlihat pada Gambar 4.6



Gambar 4.6 Tampilan Jendela Menu Histogram Grayscale

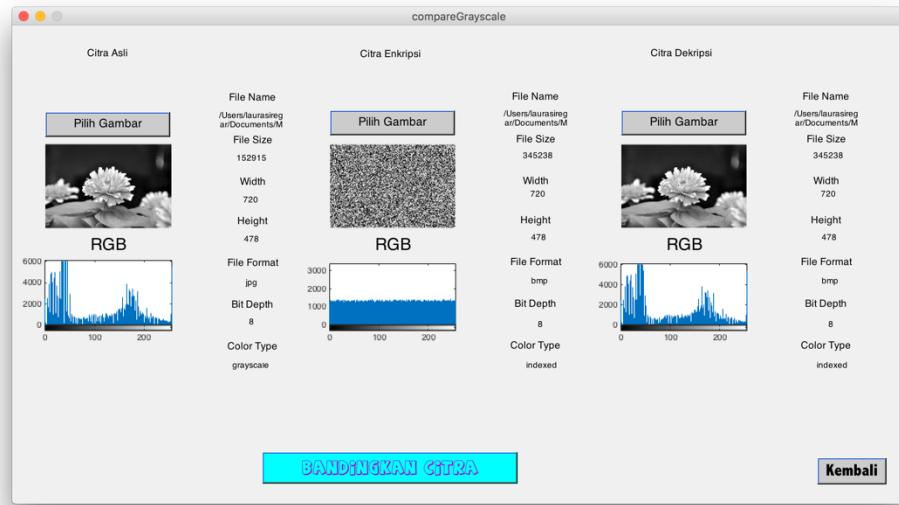
Pengguna dapat melihat histogram *red*, *green* dan *blue* dari citra, dengan cara

memilih submenu RGB untuk citra berwarna, kemudian akan tampil jendela Histogram RGB seperti pada Gambar 4.7



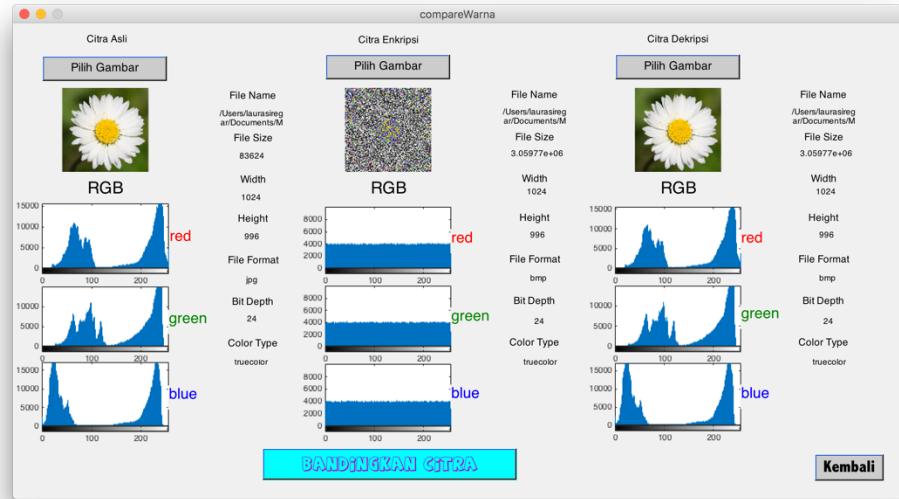
Gambar 4.7 Tampilan Jendela Menu Histogram RGB

Tahap selanjutnya untuk melihat perbandingan citra, pengguna harus memilih menu *Compare*. Pada menu *Compare* terdapat submenu untuk melihat citra *grayscale* dan citra berwarna (RGB). Pada submenu citra *grayscale* pengguna dapat membandingkan tiga buah citra *grayscale* yaitu citra asli, citra hasil enkripsi dan citra hasil dekripsi dengan cara memilih citra pada tiap-tiap bagian, setelah itu tekan tombol bandingkan citra maka akan muncul perbandingan ketiga citra seperti terlihat pada Gambar 4.8



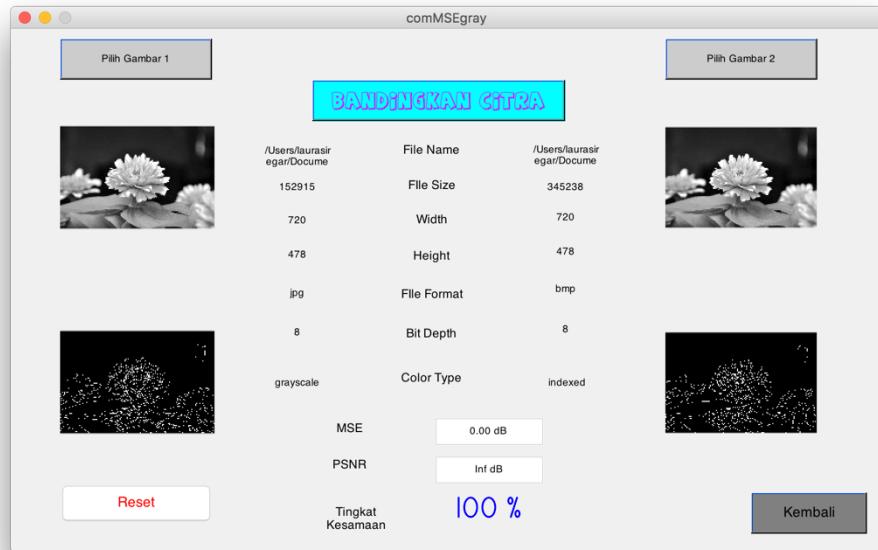
Gambar 4.8 Tampilan Jendela Menu Perbandingan Citra Grayscale

Pada submenu citra berwarna (RGB) akan membandingkan tiga buah citra berwarna (RGB) yaitu citra asli, citra hasil enkripsi dan citra hasil dekripsi. Langkah pertama, pengguna harus memilih citra asli dari citra yang telah didekripsi dengan cara menekan tombol pilih gambar pada bagian citra asli. Setelah citra dipilih, maka citra yang dipilih akan tampil pada *axes1*. Langkah kedua, pengguna harus memilih citra hasil enkripsi dengan cara menekan tombol pilih gambar pada bagian citra enkripsi. Setelah citra dipilih, maka citra yang dipilih akan tampil pada *axes2*. Langkah selanjutnya, pengguna harus memilih citra hasil dekripsi dengan cara menekan tombol pilih gambar pada bagian citra hasil dekripsi. Setelah citra dipilih, maka citra yang dipilih akan tampil pada *axes3*. Setelah ketiga citra yang dipilih tampil, maka pengguna dapat menekan tombol bandingkan citra untuk menampilkan informasi dasar mengenai citra tersebut seperti nama file, ukuran, format, dan lainnya seperti terlihat pada jendela menu perbandingan warna pada Gambar 4.9



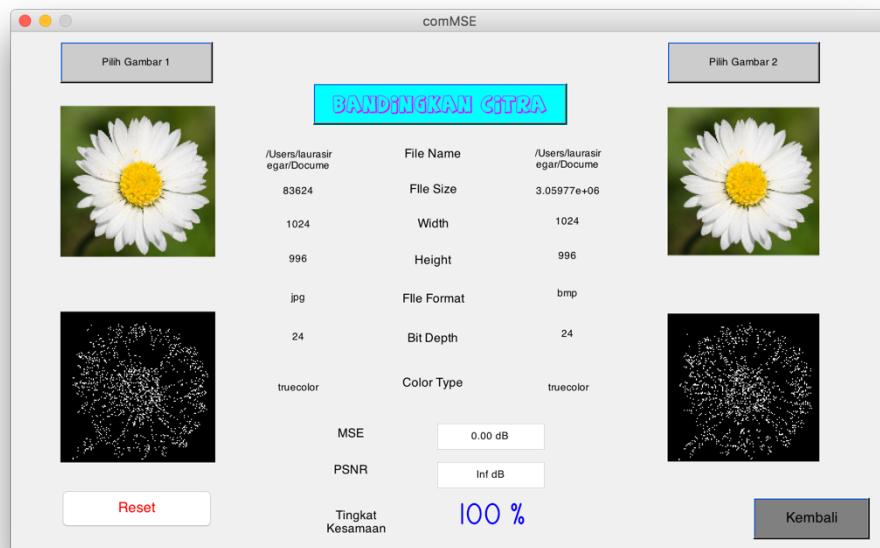
Gambar 4.9 Tampilan Jendela Perbandingan Citra Berwarna (RGB)

Pada menu *compare* atau perbandingan citra, pengguna dapat melihat perbandingan mengenai informasi dasar citra dan perbandingan yang terlihat berdasarkan histogram. Pada menu selanjutnya yaitu menu *Peak Signal to Noise Ratio* (PSNR), pengguna dapat melihat perbandingan citra berdasarkan perhitungan *Mean Squared Error* (MSE). Didalam menu PSNR juga terdapat submenu yang berisi pertanyaan mengenai citra yang hendak dibandingkan. Apabila pengguna memilih submenu *grayscale* maka rancangan tampilan terlihat seperti pada Gambar 4.10



Gambar 4.10 Tampilan Jendela PSNR Citra *Grayscale*

Apabila pengguna memilih submenu citra berwarna (RGB), maka pengguna dapat membandingkan informasi dasar kedua buah citra serta hasil *Mean Squared Error* (MSE) dari citra tersebut seperti yang terlihat pada Gambar 4.11



Gambar 4.11 Tampilan Jendela PSNR Citra Berwarna (RGB)

4.2 Analisis Hasil Uji Coba

Pada bagian ini akan diuraikan hasil uji coba dari proses enkripsi dan dekripsi citra antara lain untuk melihat kesamaan citra semula dengan citra terdekripsi, menghitung waktu proses enkripsi dan dekripsi, melihat pengaruh komposisi dan keragaman citra terhadap waktu proses enkripsi dan dekripsi, melihat sensitivitas kunci, dan melihat ruang kunci dan waktu yang dibutuhkan untuk memecahkan kunci dari serangan *brute force*.

4.2.1 Data Citra Uji Coba

Pada tahap uji coba, proses enkripsi dan dekripsi menggunakan sejumlah citra. Data citra yang digunakan dalam penelitian ini dapat dilihat pada Tabel 4.1.

Tabel 4.1 Data Citra Yang Digunakan Pada Uji Coba

Data Uji ke-	Tampilan Citra	Nama Citra	Ukuran Citra (pixel)	Ukuran File (byte)	Jenis CItra
1		abu 1.bmp	640 x 425	118 KB	Grayscale
2		abu 2.bmp	800 x 531	177 KB	Grayscale
3		abu 3.bmp	1024 x 680	253 KB	Grayscale
4		abu 4.bmp	1280 x 850	349 KB	Grayscale
5		abu 5.bmp	1627 x 1080	490 KB	Grayscale
6		matahari 1.bmp	360 x 480	36 KB	Truecolor
7		matahari 2.bmp	450 x 600	53 KB	Truecolor
8		matahari 3.bmp	576 x 768	77 KB	Truecolor
9		matahari 4.bmp	769 x 1024	121 KB	Truecolor
10		matahari 5.bmp	810 x 1080	130 KB	Truecolor

Tabel 4.1 Data Citra Yang Digunakan Pada Uji Coba

Data Uji ke-	Tampilan Citra	Nama Citra	Ukuran Citra (<i>pixel</i>)	Ukuran File (<i>byte</i>)	Jenis CItra
11		gs 1.png	480 x 480	27 KB	Grayscale
12		gs 2.png	600 x 600	34 KB	Grayscale
13		gs 3.png	720 x 720	42 KB	Grayscale
14		gs 4.png	768 x 768	46 KB	Grayscale
15		gs 5.png	1024 x 1024	64 KB	Grayscale
16		sun 1.png	493 x 480	28 KB	Truecolor
17		sun 2.png	617 x 600	41 KB	Truecolor
18		sun 3.png	790 x 768	60 KB	Truecolor
19		sun 4.png	1280x960	74 KB	Truecolor
20		sun 5.png	1110 x 1080	105 KB	Truecolor

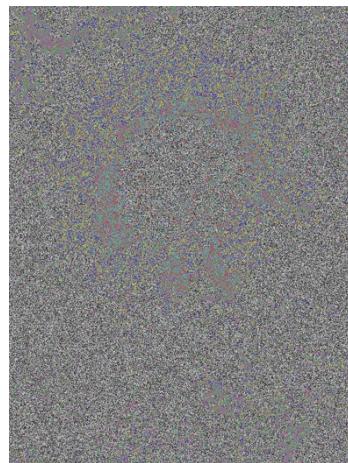
4.2.2 Enkripsi Citra

Setelah pengujian dilakukan pada sejumlah citra, baik citra *grayscale* maupun berwarna dan format .bmp maupun .png, semua citra dapat dienkripsi dengan baik menjadi citra terenkripsi (*chiper image*). Citra tersebut dapat didekripsi kembali menjadi citra semula (*plain image*) dengan memasukkan kunci yang tepat.



Gambar 4.8 Contoh Citra Uji matahari.bmp

Gambar 4.8 memperlihatkan contoh citra uji bertipe bitmap yang akan dienkripsi. Hasil enkripsi dan dekripsi contoh citra uji coba diatas akan digambarkan seperti terlihat pada Gambar 4.9 dibawah ini.



(a)



(b)

Gambar 4.9 Hasil Enkripsi dan Dekripsi Citra Uji: (a) *Chipper Image* citra matahari.bmp; (b) hasil dekripsi citra matahari.bmp

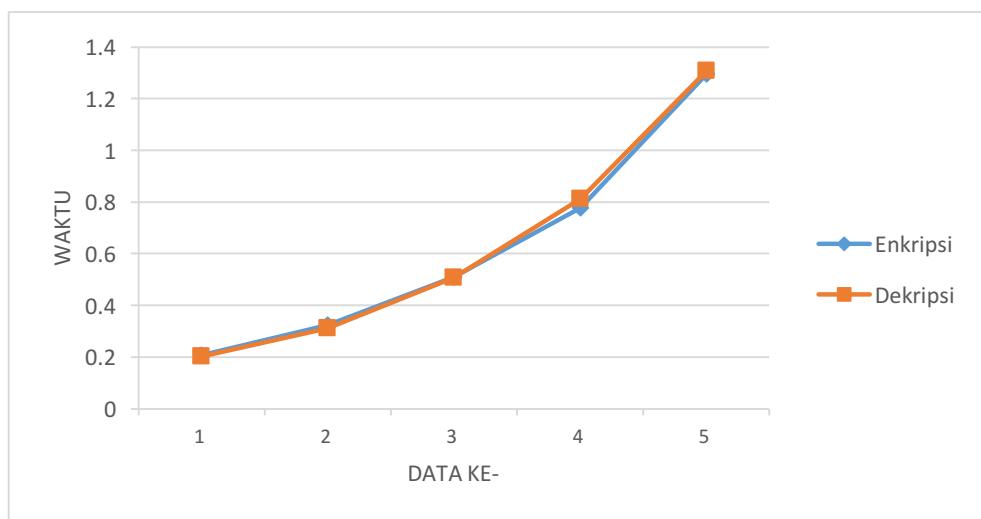
4.2.3 Analisis Waktu Enkripsi dan Dekripsi

Berikut adalah tabel perbandingan antara waktu proses rata-rata enkripsi dan dekripsi untuk tiap data uji citra dengan nilai kunci yang sama untuk setiap citra, nilai $X_n = 0.1$ dan $r = 8.1$. Pengambilan nilai waktu proses rata-rata dilakukan setelah melakukan lima kali uji coba untuk masing-masing citra. Berikut adalah data hasil uji coba pertama untuk citra grayscale dengan format .bmp beserta tampilan grafik dari data hasil uji coba tersebut.

Tabel 4.2 Tabel Waktu Enkripsi dan Dekripsi untuk Data Uji Citra *Grayscale*

Format BMP

Data ke-	Nama File	Ukuran Citra (pixel)	Waktu Rata-Rata Enkripsi (detik)	Waktu Rata-Rata Dekripsi (detik)
1	abu 1.bmp	640 x 425	0.20614	0.20246
2	abu 2.bmp	800 x 531	0.32379	0.31078
3	abu 3.bmp	1024 x 680	0.51013	0.50685
4	abu 4.bmp	1280 x 850	0.77744	0.81148
5	abu 5.bmp	1627 x 1080	1.2944	1.3082



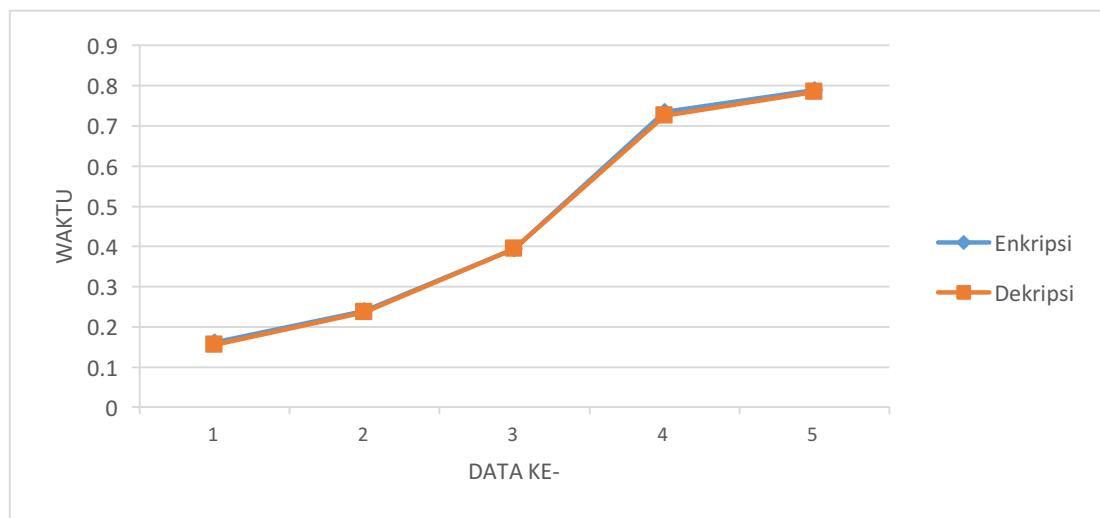
Gambar 4.10 Grafik Rata-Rata Waktu Enkripsi dan Dekripsi Data Uji Citra Grayscale Format BMP

Dari data percobaan pertama yang disuguhkan pada Tabel 4.2 dan Gambar 4.10, terlihat bahwa waktu proses enkripsi dan dekripsi mengalami kenaikan seiring dengan bertambahnya ukuran pixel citra tersebut.

Selanjutnya adalah data hasil uji untuk citra berwarna berformat bmp beserta grafik dari data hasil uji coba tersebut.

Tabel 4.3 Tabel Waktu Enkripsi dan Dekripsi untuk Data Uji Citra Berwarna (RGB) Format BMP

Data ke-	Nama File	Ukuran Citra (pixel)	Waktu Rata-Rata Enkripsi (detik)	Waktu Rata-Rata Dekripsi (detik)
1	matahari 1.bmp	360 x 480	0.16178	0.15538
2	matahari 2.bmp	450 x 600	0.23972	0.23636
3	matahari 3.bmp	576 x 768	0.39423	0.39414
4	matahari 4.bmp	769 x 1024	0.73555	0.72582
5	matahari 5.bmp	810 x 1080	0.78938	0.78508



Gambar 4.11 Grafik Rata-Rata Waktu Enkripsi dan Dekripsi Data Uji Citra Berwarna (RGB) Format BMP

Dengan menggunakan data uji coba pada Tabel 4.3 dan grafik pada Gambar

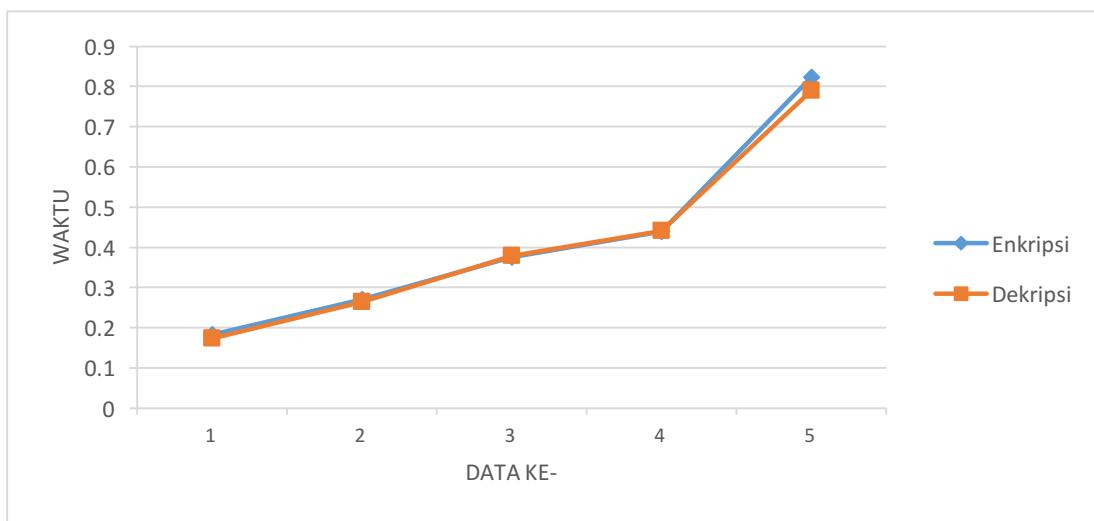
4.11, dapat dilihat bahwa waktu proses juga mengalami kenaikan seiring dengan bertambahnya ukuran citra tersebut.

Setelah melakukan uji coba dengan citra format bmp, dalam dua tipe warna yang berbeda yaitu *grayscale* dan berwarna (RGB), selanjutkan akan dilakukan uji terhadap citra format png. Pertama akan diuji citra *grayscale* format png terlebih dahulu.

Tabel 4.4 Tabel Waktu Enkripsi dan Dekripsi untuk Data Uji Citra Grayscale

Format PNG

Data ke-	Nama File	Ukuran Citra (pixel)	Waktu Rata-Rata Enkripsi (detik)	Waktu Rata-Rata Dekripsi (detik)
1	gs 1.png	480 x 480	0.18275	0.1736
2	gs 2.png	600 x 600	0.27087	0.26431
3	gs 3.png	720 x 720	0.37596	0.37861
4	gs 4.png	768 x 768	0.43902	0.441
5	gs 5.png	1024 x 1024	0.82234	0.78909



Gambar 4.12 Grafik Rata-rata Waktu Enkripsi dan Dekripsi Data Uji Citra Grayscale Format PNG

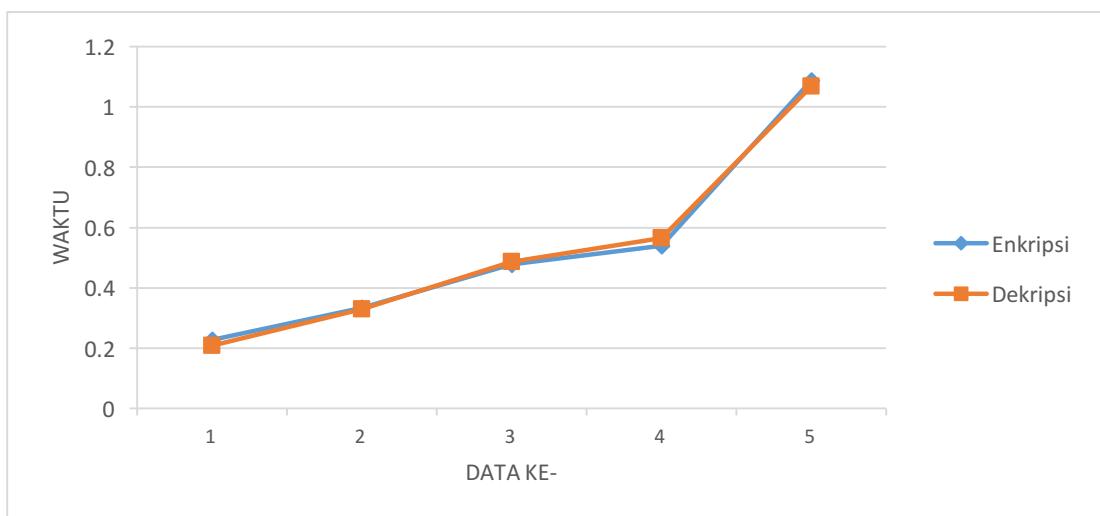
Dilihat dari Tabel 4.4 dan Gambar 4.12, hal yang sama terjadi pada citra *grayscale* berformat png. Dengan lima ukuran pixel yang berbeda dari setiap citra, waktu

proses ikut meningkat seiring dengan bertambahnya ukuran citra tersebut.

Tahap terakhir yang akan dilakukan ujicoba terhadap citra berwarna dengan fomat png. Berikut adalah hasil dan grafik dari data hasil uji coba tersebut.

Tabel 4.5 Tabel Waktu Enkripsi dan Dekripsi untuk Data Uji Citra Berwarna (RGB) Format PNG

Data ke-	Nama File	Ukuran Citra (pixel)	Waktu Rata-Rata Enkripsi (detik)	Waktu Rata-Rata Dekripsi (detik)
1	sun 1.png	493 x 480	0.22658	0.20779
2	sun 2.png	617 x 600	0.3342	0.33001
3	sun 3.png	790 x 768	0.47707	0.48597
4	sun 4.png	1280x960	0.53891	0.56581
5	sun 5.png	1110 x 1080	1.0884	1.0681



Gambar 4.13 Grafik Rata-rata Waktu Enkripsi dan Dekripsi Data Uji Citra Berwarna (RGB) Format PNG

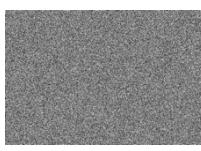
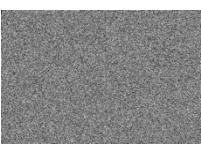
Untuk uji coba pada citra berwarna (RGB) pada format png, memiliki hasil yang sama seperti tiga percobaan sebelumnya, yaitu waktu proses meningkat seiring dengan bertambahnya ukuran citra

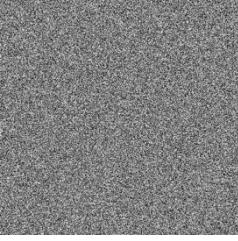
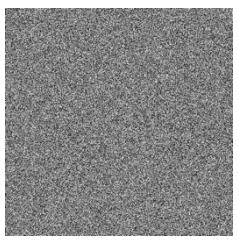
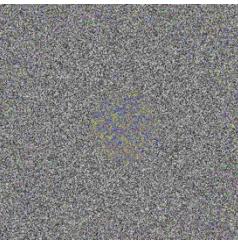
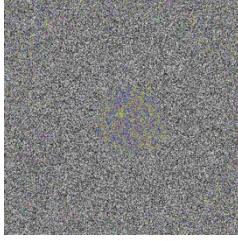
Kesimpulan dari analisis waktu ini yaitu bahwa semakin besar ukuran *pixel* pada suatu citra, baik itu citra grayscale maupun citra berwarna (RGB) dalam berbagai format (bmp dan png), waktu yang dibutuhkan untuk melakukan proses enkripsi dan dekripsi semakin lama. Jadi sudah terbukti bahwa perbedaan ukuran citra sangat berpengaruh.

4.2.4 Analisis Sensitivitas Kunci

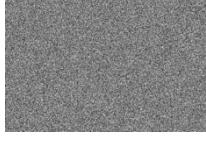
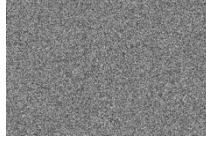
Pengujian dilakukan dengan membandingkan citra hasil dekripsi pada sejumlah citra uji coba yang telah dienkripsi dengan nilai kunci X_n sebesar 0.1 dan r sebesar 8.001, dengan perubahan yang sangat kecil dari nilai pada salah satu kunci. Hasil pengujian dapat dilihat pada Tabel 4.6 dan Tabel 4.7

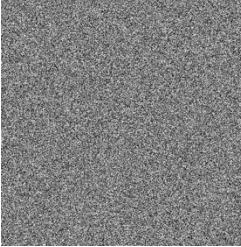
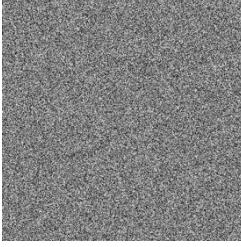
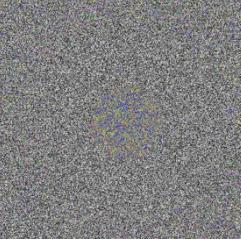
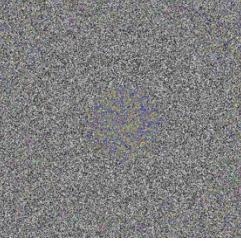
Tabel 4.6 Data Hasil Pengujian Sensitivitas Kunci Dengan Perubahan Pada Salah Satu Kunci

Nama File	Citra Hasil Enkripsi dengan Kunci $X_n = 0.1$ dan Kunci $r = 8.1$	Citra Hasil Dekripsi dengan Kunci $X_n = 0.1$ dan Kunci $r = 8.1 + 10^{-13}$	Citra Hasil Dekripsi dengan Kunci $X_n = 0.1$ dan Kunci $r = 8.1 + 10^{-14}$
abu 1.bmp			
matahari 1.bmp			
Nama File	Citra Hasil Enkripsi dengan Kunci $X_n = 0.1$ dan Kunci $r = 8.1$	Citra Hasil Dekripsi dengan Kunci $X_n = 0.1$ dan Kunci $r = 8.1 + 10^{-13}$	Citra Hasil Dekripsi dengan Kunci $X_n = 0.1$ dan Kunci $r = 8.1$

			$+10^{-14}$
gs 1.png			
sun 1.png			

Tabel 4.7 Data Hasil Pengujian Sensitivitas Kunci Dengan Perubahan Pada Kedua Kunci

Nama File	Citra Hasil Enkripsi dengan Kunci $X_n = 0.1$ dan Kunci $r = 8.1$	Citra Hasil Dekripsi dengan Kunci $X_n = 0.1 + 10^{-14}$ dan Kunci $r = 8.1 + 10^{-13}$	Citra Hasil Dekripsi dengan Kunci $X_n = 0.1 + 10^{-15}$ dan Kunci $r = 8.1 + 10^{-14}$
abu 1.bmp			
Nama File	Citra Hasil Enkripsi dengan Kunci $X_n = 0.1$ dan Kunci $r = 8.1$	Citra Hasil Dekripsi dengan Kunci $X_n = 0.1 + 10^{-14}$ dan Kunci $r = 8.1 + 10^{-13}$	Citra Hasil Dekripsi dengan Kunci $X_n = 0.1 + 10^{-15}$ dan Kunci $r = 8.1 + 10^{-14}$

			14
matahari 1.bmp			
gs 1.png			
sun 1.png			

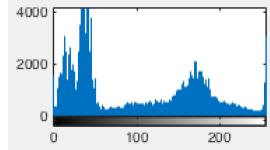
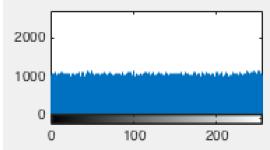
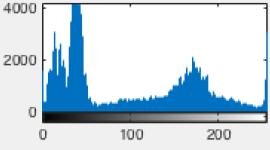
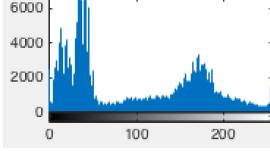
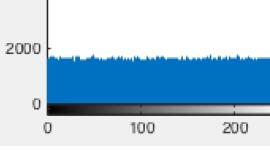
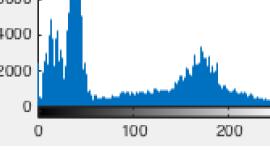
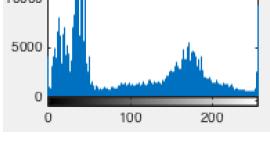
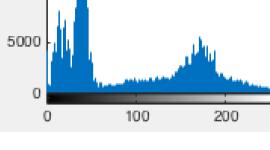
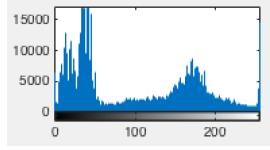
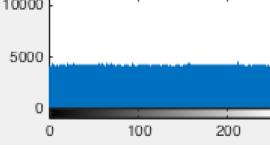
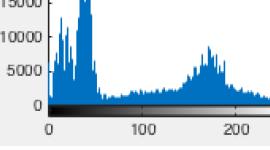
Algoritma Bernoulli Map memiliki nilai sensitivitas kunci yang cukup kecil yakni sampai 10^{-13} . Sensitivitas kunci tersebut dapat diketahui dengan dilakukan sebuah percobaan dengan mengubah nilai kunci dari algoritma Bernoulli Map dengan menambahkan nilai sebesar 10^{-13} dan ternyata menghasilkan citra yang tetap terenkripsi. Sedangkan ketika nilai kunci ditambahkan menjadi 10^{-14} , citra tersebut berhasil didekripsi. Hal ini membuktikan bahwa algoritma Bernoulli Map cukup baik untuk menghentikan serangan *brute force* dikarenakan perubahan satu bit pada kunci akan menyebabkan sebuah hasil yang berbeda.

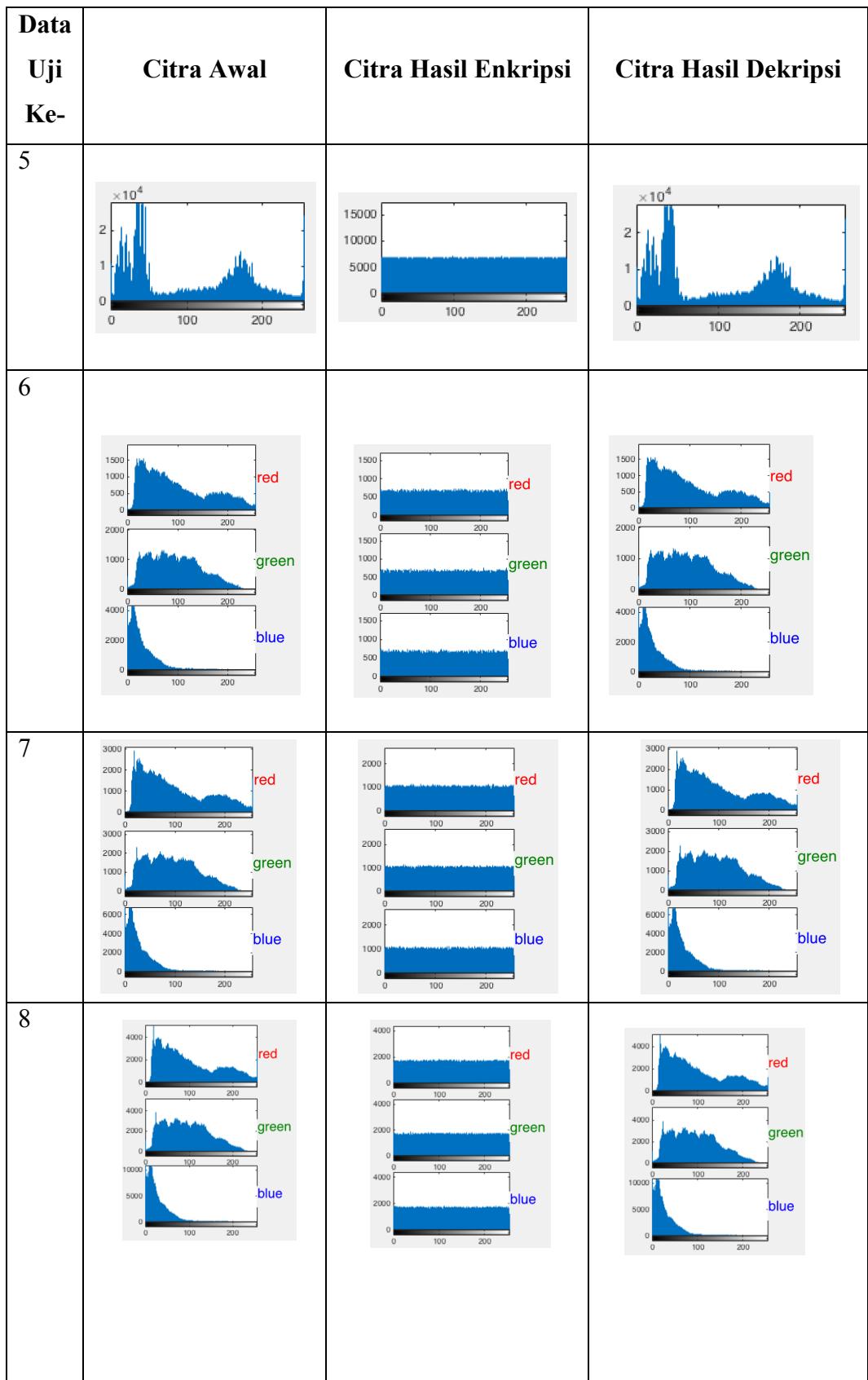
4.2.5 Analisis Kesamaan Citra

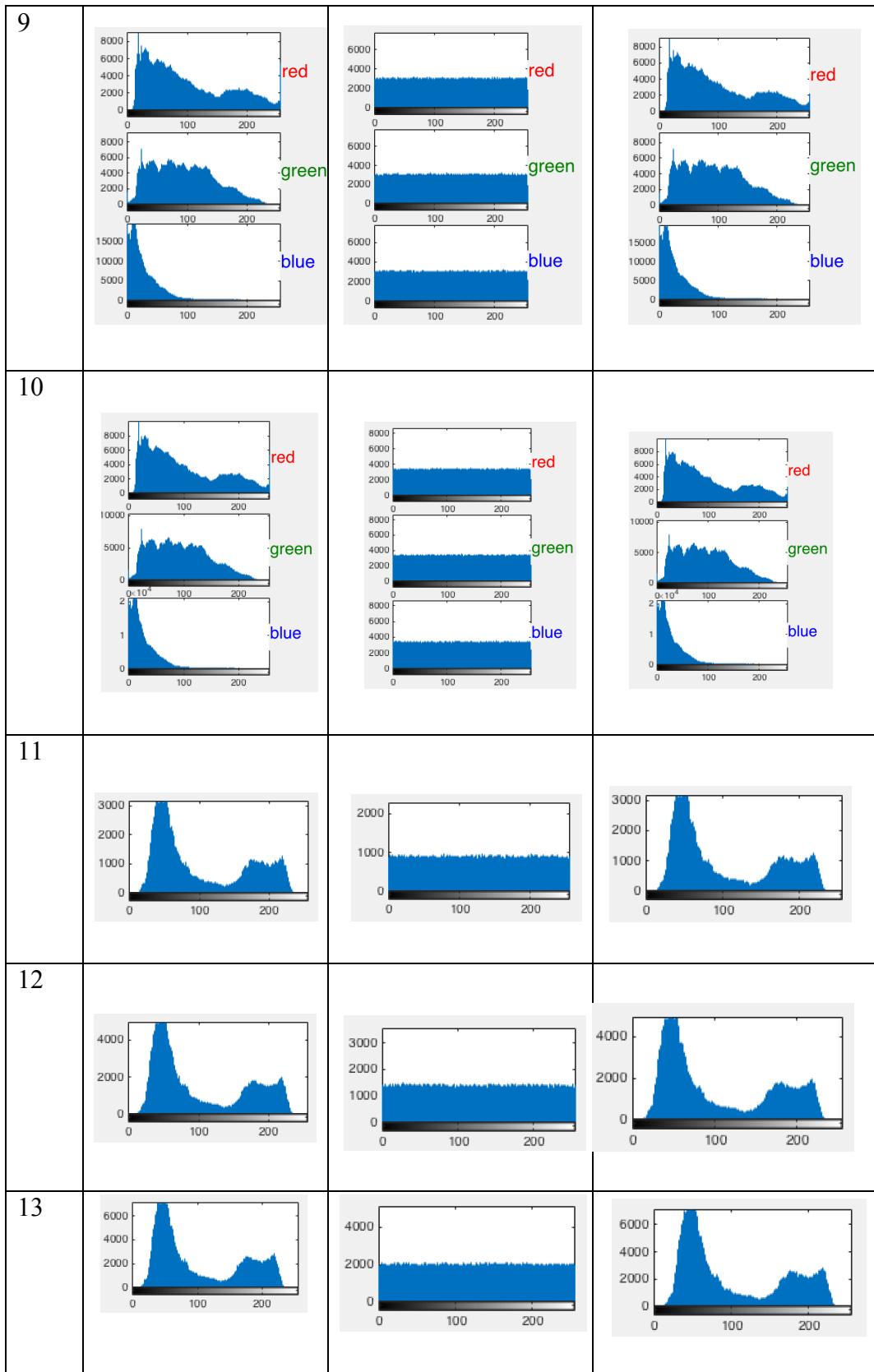
Untuk mengetahui kesamaan citra semula, citra enkripsi, dan citra dekripsi maka

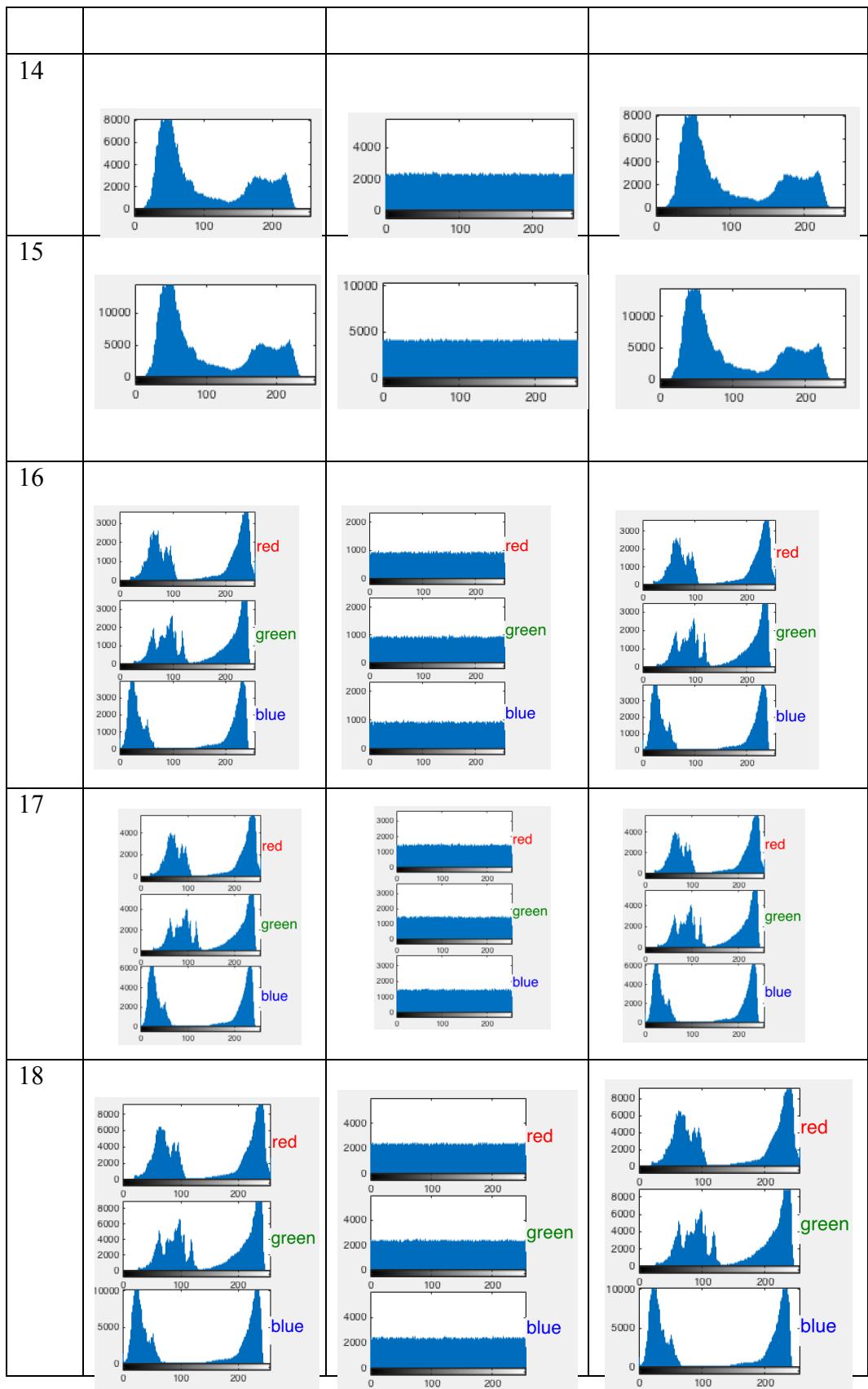
perlu dilakukan pengujian dengan membandingkan histogram dari tiap citra. Berikut tabel hasil pengujinya :

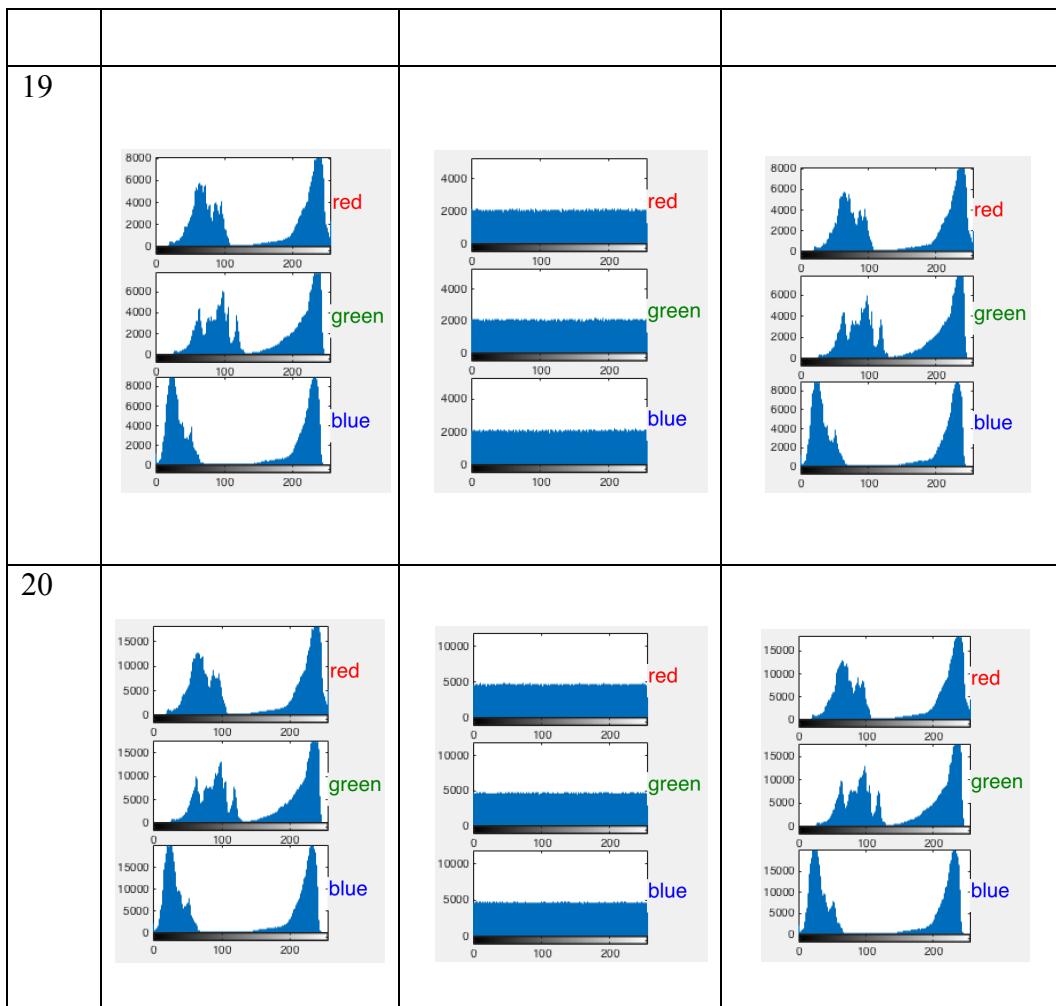
Tabel 4.8 Hasil Histogram Citra Uji Coba

Data Uji Ke-	Citra Awal	Citra Hasil Enkripsi	Citra Hasil Dekripsi
1			
2			
3			
4			









Tabel 4.8 menunjukkan histogram dari tiap-tiap citra dalam format png dan bmp, pada citra *grayscale* maupun citra warna. Dari pengujian yang dilakukan, terbukti bahwa hasil dari citra yang sudah didekripsi sama dengan citra semula atau citra asli. Sedangkan, pada histogram citra hasil enkripsi, persebaran piksel pikselnya merata atau seragam. Histogram dari citra hasil enkripsi relatif datar sehingga tahan terhadap penyerang yang akan melakukan analisis frekuensi. Oleh karena itu, histogram citra awal dan histogram citra terenkripsi seharusnya tidak memiliki kesamaan secara statistik. Persebaran piksel secara merata pada citra hasil enkripsi merupakan sebuah bukti bahwa algoritma enkripsi citra memiliki kualitas yang baik.

BAB V

PENUTUP

5.1 Kesimpulan

Pada penelitian tugas akhir skripsi ini telah dibuat program aplikasi enkripsi dan dekripsi untuk citra digital. Aplikasi ini dapat melakukan enkripsi dan dekripsi untuk citra yang memiliki tipe *grayscale* maupun berwarna. Tetapi proses enkripsi dan dekripsi citra hanya baik dilakukan pada citra dengan format bmp (bitmap) dan png (portable network graphics).

Implementasi Algoritma Bernoulli Map pada proses enkripsi dan dekripsi citra digital ini telah berhasil dilakukan pada aplikasi dan diuji cobakan pada beberapa citra, yaitu citra *grayscale* dan citra berwarna. Hasil dekripsi dari setiap citra yang dienkripsi sesuai dengan citra semula, tetapi memiliki sedikit perbedaan pada ukuran *file* citra dalam *kilobyte*. Hanya membutuhkan waktu yang singkat untuk dapat melihat hasil dari proses enkripsi dan dekripsi citra. Berdasarkan analisis yang dilakukan, muncul fakta bahwa semakin besar ukuran piksel sebuah citra, maka semakin lama waktu yang dibutuhkan untuk melakukan proses enkripsi maupun dekripsi. Perubahan pada kunci sangat sensitif sehingga akan sangat berpengaruh pada hasil proses enkripsi dan dekripsi yang mencapai 10^{-14} . Pada histogram citra terenkripsi memiliki penyebaran piksel yang merata. Hal ini membuat pihak ketiga sangat kesulitan untuk dapat membuka file citra hasil enkripsi tersebut dengan cara *brute force attack*.

5.2 Saran

Aplikasi enkripsi dan dekripsi citra digital menggunakan algoritma Bernoulli map masih memiliki kekurangan, yaitu citra yang dapat dienkripsi dan didekripsi hanya pada ekstensi tertentu yaitu bitmap dan png. Pada format file jpeg proses tetap dapat dilakukan, tetapi memiliki perbedaan antara citra asli dengan citra yang di enkripsi maupun dekripsi.

DAFTAR PUSTAKA

- [1] Ari Sugiharto. *Pemrograman GUI dengan Matlab*. ANDI. 2006
- [2] Darma Putra. *Pengolahan Citra Digital*. Andi Publisher. 2010
- [3] Dony Ariyus. *Pengantar Ilmu Kriptografi Teori, Analisis, dan Implementaso*. ANDI YOGYAKARTA. 2008
- [4] Edi Sukirman, Suryadi MT, M. Agus Mubarak. The Implementation of Henon Map Algorithm for Digital Image Encryption. *TELEKOMNIKA Vol 12 No 3*. ISSN: 1693-6930. September 2014
- [5] Gunaidi Abadia Away. *The Shortcut of Matlab Programming*. Informatika Bandung. 2006
- [6] Kasiman Peranginangan. *Pengenalan MATLAB*. ANDI. Yogyakarta. 2006
- [7] Rinaldi Munir. *Kriptografi*. Informatika. 2006
- [8] Rinaldi Munir. Algoritma Enkripsi Citra Chaos dengan Penggabungan Teknik Permutasi dan Teknik Substitusi Menggunakan Arnold Cat map dan Logistic Map. *Jurnal Nasional Pendidikan Teknik Informatika Vol 1 No 3. Desember 2014*
- [9] Suryadi MT, Eva Nurpeti, Dhian Widya. Performance of Chaos-Based Encryption Algorithm for Digital Image. *TELEKOMNIKA Vol 12 No 3. September 2014*

LAMPIRAN 1

LISTING PROGRAM

menu_utama.m

```
function varargout = menu_utama(varargin)
% MENU_UTAMA MATLAB code for menu_utama.fig
%     MENU_UTAMA, by itself, creates a new MENU_UTAMA or raises the existing
%     singleton*.
%  
%  
% H = MENU_UTAMA returns the handle to a new MENU_UTAMA or the handle to
%     the existing singleton*.
%  
%  
% MENU_UTAMA('CALLBACK', hObject, eventData, handles,...) calls the local
%     function named CALLBACK in MENU_UTAMA.M with the given input
%     arguments.
%  
%  
% MENU_UTAMA('Property','Value',...) creates a new MENU_UTAMA or raises
%     the
%  
%     existing singleton*. Starting from the left, property value pairs are
%     applied to the GUI before menu_utama_OpeningFcn gets called. An
%     unrecognized property name or invalid value makes property application
%     stop. All inputs are passed to menu_utama_OpeningFcn via varargin.
%  
%  
% *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%     instance to run (singleton)".
%  
%  
% See also: GUIDE, GUIDATA, GUIHANDLES
%  
  
% Edit the above text to modify the response to help menu_utama
%  
  
% Last Modified by GUIDE v2.5 23-Aug-2016 08:33:52
%  
  
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',         mfilename, ...
                   'gui_Singleton',    gui_Singleton, ...
                   'gui_OpeningFcn',   @menu_utama_OpeningFcn, ...
                   'gui_OutputFcn',    @menu_utama_OutputFcn, ...
                   'gui_LayoutFcn',    [], ...
                   'gui_Callback',     []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
  
  
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
%  
  
% --- Executes just before menu_utama is made visible.
```

```

function menu_utama_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin    command line arguments to menu_utama (see VARARGIN)

% Choose default command line output for menu_utama
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes menu_utama wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = menu_utama_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in enkripsi.
function enkripsi_Callback(hObject, eventdata, handles)
% hObject    handle to enkripsi (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
enkripsi;
close menu_utama;

% --- Executes on button press in bantuan.
function bantuan_Callback(hObject, eventdata, handles)
% hObject    handle to bantuan (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
bantuan;
close menu_utama;

% --- Executes on button press in dekripsi.
function dekripsi_Callback(hObject, eventdata, handles)
% hObject    handle to dekripsi (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
dekripsi;
close menu_utama;

% --- Executes on button press in compare.
function compare_Callback(hObject, eventdata, handles)

```

```
% hObject      handle to compare (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
SubMenuCompare;
close menu_utama;
```

```
% --- Executes on button press in tentang.
function tentang_Callback(hObject, eventdata, handles)
% hObject      handle to tentang (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
tentang;
```

```
% --- Executes on button press in histogram.
function histogram_Callback(hObject, eventdata, handles)
% hObject      handle to histogram (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
SubMenuHistogram;
```

```
% --- Executes on button press in btnPSNR.
function btnPSNR_Callback(hObject, eventdata, handles)
% hObject      handle to btnPSNR (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
SubMenuPSNR;
```

Enkripsi.m

```
function varargout = enkripsi(varargin)
% ENKRIPSI MATLAB code for enkripsi.fig
%   ENKRIPSI, by itself, creates a new ENKRIPSI or raises the existing
%   singleton*.
%
%   H = ENKRIPSI returns the handle to a new ENKRIPSI or the handle to
%   the existing singleton*.
%
%   ENKRIPSI('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in ENKRIPSI.M with the given input arguments.
%
%   ENKRIPSI('Property','Value',...) creates a new ENKRIPSI or raises the
%   existing singleton*. Starting from the left, property value pairs are
%   applied to the GUI before enkripsi_OpeningFcn gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to enkripsi_OpeningFcn via varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help enkripsi

% Last Modified by GUIDE v2.5 14-Aug-2016 19:41:38
```

```

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',         mfilename, ...
                   'gui_Singleton',    gui_Singleton, ...
                   'gui_OpeningFcn',   @enkripsi_OpeningFcn, ...
                   'gui_OutputFcn',    @enkripsi_OutputFcn, ...
                   'gui_LayoutFcn',    [] , ...
                   'gui_Callback',     []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before enkripsi is made visible.
function enkripsi_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to enkripsi (see VARARGIN)

% Choose default command line output for enkripsi
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes enkripsi wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = enkripsi_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in buka_gambar.
function buka_gambar_Callback(hObject, eventdata, handles)
% hObject    handle to buka_gambar (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% [handles.namafile,handles.direktori]=uigetfile({'*.bmp;*.jpg;*.png','file

```

```

citra (*.bmp, *.jpg)'; '*.bmp', 'file bitmap (*.bmp)'; '*.jpg', 'file
jpg (*.jpg)'; '.*', 'semua file (*.*)'}, 'buka file citra host/asli');
%
% Img=imread(fullfile(handles.direktori,handles.namafile));
% if size(Img,1)>1
%     set(handles.figure1,'CurrentAxes',handles.axes1);
%     set(imshow(Img));
%         set(handles.axes1,'Userdata',Img);
%     mypaths=[handles.direktori,handles.namafile];
%     set(handles.edPath1,'string',mypaths);
% end
global Img;
X = uigetfile({'*.bmp'; '*.png'; '*.jpg'}, 'pilih gambar');
Img = imread(X);
axes(handles.axes1)
imshow(Img);
guidata(hObject, handles);

% --- Executes on button press in enkripsi.
function enkripsi_Callback(hObject, eventdata, handles)
% hObject    handle to enkripsi (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Img;
global EncImg;
global key;
global A;
global B;

form = guidata(gcbo);
A=str2double(get(form.edit1,'String'));
B=str2double(get(form.edit2,'String'));

[n m k] = size(Img);
tic
key = bernoulli(Img,n*m,A,B);
EncImg = proses(Img, key);
timeEnc = toc;
axes(handles.axes2)
imshow(EncImg);
set(form.text4,'String', num2str(timeEnc)); guidata(hObject, handles);

% --- Executes on button press in simpan.
function simpan_Callback(hObject, eventdata, handles)
% hObject    handle to simpan (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global EncImg;
[filename] = uiputfile({'*.jpg'; '*.bmp'; '*.png'}, 'simpan gambar');
imwrite(EncImg,[filename]);

% --- Executes on button press in kembali.
function kembali_Callback(hObject, eventdata, handles)
% hObject    handle to kembali (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB

```

```

% handles      structure with handles and user data (see GUIDATA)
menu_utama;
close enkripsi;

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%         str2double(get(hObject,'String')) returns contents of edit1 as a
double

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%         str2double(get(hObject,'String')) returns contents of edit2 as a
double

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)

```

```

% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
% str2double(get(hObject,'String')) returns contents of edit3 as a
double

% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject handle to edit3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edPath1_Callback(hObject, eventdata, handles)
% hObject handle to edPath1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edPath1 as text
% str2double(get(hObject,'String')) returns contents of edPath1 as a
double

% --- Executes during object creation, after setting all properties.
function edPath1_CreateFcn(hObject, eventdata, handles)
% hObject handle to edPath1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

dekripsi.m

```

function varargout = comMSE(varargin)
% COMMSE MATLAB code for comMSE.fig
% COMMSE, by itself, creates a new COMMSE or raises the existing
% singleton*.
%
% H = COMMSE returns the handle to a new COMMSE or the handle to
% the existing singleton*.
%
```

```

%
%      COMMSE( 'CALLBACK', hObject, eventData, handles,...) calls the local
%      function named CALLBACK in COMMSE.M with the given input arguments.
%
%
%      COMMSE('Property','Value',...) creates a new COMMSE or raises the
%      existing singleton*. Starting from the left, property value pairs are
%      applied to the GUI before comMSE_OpeningFcn gets called. An
%      unrecognized property name or invalid value makes property application
%      stop. All inputs are passed to comMSE_OpeningFcn via varargin.
%
%
%      *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%      instance to run (singleton)".
%
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help comMSE

% Last Modified by GUIDE v2.5 23-Aug-2016 19:20:12

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',          mfilename, ...
                   'gui_Singleton',    gui_Singleton, ...
                   'gui_OpeningFcn',   @comMSE_OpeningFcn, ...
                   'gui_OutputFcn',    @comMSE_OutputFcn, ...
                   'gui_LayoutFcn',    [] , ...
                   'gui_Callback',     []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

%
% --- Executes just before comMSE is made visible.
function comMSE_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to comMSE (see VARARGIN)

%
% Choose default command line output for comMSE
handles.output = hObject;

%
% Update handles structure
guidata(hObject, handles);

%
% UIWAIT makes comMSE wait for user response (see UIRESUME)
% uiwait(handles.figure1);

```

```

% --- Outputs from this function are returned to the command line.
function varargout = comMSE_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pilihgambar1.
function pilihgambar1_Callback(hObject, eventdata, handles)
% hObject handle to pilihgambar1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
global Img1;
global info1;
Img1=[];
[filename1,pathfile1]= uigetfile({'*.bmp; *.png'},'pilih gambar');
if isequal(filename1,0)
return;
end

Img1 = imread(strcat(pathfile1,filename1));
fullpath1=fullfile(pathfile1,filename1);
info1 = imfinfo(fullpath1);
axes(handles.axes1)
imshow(Img1);

set(handles.axes1,'Userdata',Img1);
guidata(hObject,handles);

% --- Executes on button press in pilihgambar2.
function pilihgambar2_Callback(hObject, eventdata, handles)
% hObject handle to pilihgambar2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
global Img2;
global info2;
Img2=[];
[filename2,pathfile2] = uigetfile({'*.bmp; *.png'},'pilih gambar');
if isequal(filename2,0)
return;
end

Img2 = imread(strcat(pathfile2,filename2));
fullpath2=fullfile(pathfile2,filename2);
info2 = imfinfo(fullpath2);
axes(handles.axes2)
imshow(Img2);

set(handles.axes2,'Userdata',Img2);
guidata(hObject,handles);

% --- Executes on button press in btnProses.

```

```

function btnProses_Callback(hObject, eventdata, handles)
% hObject    handle to btnProses (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Img1;
global Img2;
global info1;
global info2;
cocok =0;
nilai_putih=0;
nilai_hitam=0;

Img1=get(handles.axes1,'Userdata');
Img2=get(handles.axes2,'Userdata');

if isequal(Img1,[])
warndlg('Tidak Ada Gambar 1, Silahkan Masukkan Gambar 1 !!!',[ 'PERINGATAN' ])
elseif isequal(Img2,[])
    warndlg('Tidak Ada Gambar 2, Silahkan Masukkan Gambar 2
!!!',[ 'PERINGATAN' ])
else

Img2b=rgb2gray(Img2);
deteksi2=edge(Img2b, 'canny');
axes(handles.axes3)
imshow(deteksi2);
guidata(hObject,handles);

Img1b=rgb2gray(Img1);
deteksi1=edge(Img1b, 'canny');
axes(handles.axes4)
imshow(deteksi1);
guidata(hObject,handles);

form = guidata(gcbo);

set(form.text36,'String',info1.Filename);
set(form.text37,'String',info2.Filename);

set(form.text38,'String',info1.FileSize);
set(form.text39,'String',info2.FileSize);

set(form.text40,'String',info1.Width);
set(form.text41,'String',info2.Width);
o=get(handles.text40,'String');
p=get(handles.text41,'String');
h=strcmp(o,p);

set(form.text42,'String',info1.Height);
set(form.text43,'String',info2.Height);
v=get(handles.text42,'String');
f=get(handles.text43,'String');
j=strcmp(v,f);

set(form.text44,'String',info1.Format);
set(form.text45,'String',info2.Format);

```

```

set(form.text46,'String',info1.BitDepth);
set(form.text47,'String',info2.BitDepth);

set(form.text48,'String',info1.ColorType);
set(form.text49,'String',info2.ColorType);

if (h==0 || j==0);
warndlg('Lebar atau Tinggi Gambar Tidak Sama Sehingga MSE dan PSNR Tidak Ada',[ 'PERINGATAN' ])
else
for k=1:1:256
    for l=1:1:256
        if(deteksi1(k,l)==1)
            nilai_putih = nilai_putih+1;
        else
            nilai_hitam = nilai_hitam+1;
        end
    end
end

for i = 1:1:256
    for j = 1:1:256
        if(deteksi1(i,j)==1)&&(deteksi2(i,j)==1)
            cocok=cocok+1;
        else
        end
    end
end
total_data=nilai_putih;
persen=(cocok/total_data)*100;
set(form.txtHasil,'String',[ (num2str(persen)), ' %' ]);
% % total_data=nilai_putih;
% % persen=(cocok/total_data)*100;
% % set(form.persen,'String',[ (num2str(persen,'%7.2f')), ' %' ]);

I = Img1;
Ihat = Img2;

% Read the dimensions of the image.
[rows columns ~] = size(I);

% Calculate mean square error of R, G, B.
mseRImage = (double(I(:,:,:1)) - double(Ihat(:,:,:1))) .^ 2;
mseGImage = (double(I(:,:,:2)) - double(Ihat(:,:,:2))) .^ 2;
mseBImage = (double(I(:,:,:3)) - double(Ihat(:,:,:3))) .^ 2;

mseR = sum(sum(mseRImage)) / (rows * columns);
mseG = sum(sum(mseGImage)) / (rows * columns);
mseB = sum(sum(mseBImage)) / (rows * columns);

% Average mean square error of R, G, B.

```

```

mse = (mseR + mseG + mseB)/3;

% Calculate text3 (Peak Signal to noise ratio).
psnr = 10 * log10( 255^2 / mse);
set(handles.mse,'String',num2str(mse,'%7.2f dB'));
set(handles.psnr,'String',num2str(psnr,'%7.2f dB'));
end
end

% --- Executes on button press in btnReset.
function btnReset_Callback(hObject, eventdata, handles)
% hObject    handle to btnReset (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
mau=questdlg(['Membersihkan '
get(handles.figure1,'Name')'], ['BERSIHKAN','Ya','Tidak','Ya']);
if strcmp(mau,'Tidak')
return;
else
    set(handles.axes1,'Userdata',[ ]);
    set(handles.axes2,'Userdata',[ ]);
    set(handles.text36,'String','');
    set(handles.text37,'String','');
    set(handles.text38,'String','');
    set(handles.text39,'String','');
    set(handles.text40,'String','');
    set(handles.text41,'String','');
    set(handles.text42,'String','');
    set(handles.text43,'String','');
    set(handles.text44,'String','');
    set(handles.text45,'String','');
    set(handles.text46,'String','');
    set(handles.text47,'String','');
    set(handles.text48,'String','');
    set(handles.text49,'String','');
    set(handles.mse,'String','');
    set(handles.psnr,'String','');
    set(handles.txtHasil,'String','');

    arrayfun(@cla,findall(0,'type','axes'))
    cla(handles.axes1,'reset');
    cla(handles.axes2,'reset');
    cla(handles.axes3,'reset');
    cla(handles.axes4,'reset');

    set(handles.axes1,'visible','off')
    set(handles.axes2,'visible','off')
    set(handles.axes3,'visible','off')
    set(handles.axes4,'visible','off')
hm = msgbox('Membersihkan Sukses','BERSIHKAN','help');
end

% --- Executes on button press in kembali.
function kembali_Callback(hObject, eventdata, handles)
% hObject    handle to kembali (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB

```

```

% handles      structure with handles and user data (see GUIDATA)
close commSE;
SubMenuPSNR;

function mse_Callback(hObject, eventdata, handles)
% hObject    handle to text2 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of text2 as text
%        str2double(get(hObject,'String')) returns contents of text2 as a
double

% --- Executes during object creation, after setting all properties.
function mse_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text2 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function psnr_Callback(hObject, eventdata, handles)
% hObject    handle to text3 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of text3 as text
%        str2double(get(hObject,'String')) returns contents of text3 as a
double

% --- Executes during object creation, after setting all properties.
function psnr_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text3 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

```

function persen_Callback(hObject, eventdata, handles)
% hObject    handle to persen (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of persen as text
%         str2double(get(hObject,'String')) returns contents of persen as a
double

% --- Executes during object creation, after setting all properties.
function persen_CreateFcn(hObject, eventdata, handles)
% hObject    handle to persen (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pilihgambar1.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pilihgambar1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in pilihgambar2.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pilihgambar2 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in btnProses.
function proses_Callback(hObject, eventdata, handles)
% hObject    handle to btnProses (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in btnReset.
function reset_Callback(hObject, eventdata, handles)
% hObject    handle to btnReset (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in kembali.
function pushbutton10_Callback(hObject, eventdata, handles)
% hObject    handle to kembali (see GCBO)

```

```

% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

function edit5_Callback(hObject, eventdata, handles)
% hObject handle to mse (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of mse as text
%         str2double(get(hObject,'String')) returns contents of mse as a
double

% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject handle to mse (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit6_Callback(hObject, eventdata, handles)
% hObject handle to psnr (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of psnr as text
%         str2double(get(hObject,'String')) returns contents of psnr as a
double

% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject handle to psnr (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

```

function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to persen (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of persen as text
%         str2double(get(hObject,'String')) returns contents of persen as a
double

% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to persen (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

Submenucompare.m

```

function varargout = SubMenuCompare(varargin)
% SUBMENUCOMPARE MATLAB code for SubMenuCompare.fig
%     SUBMENUCOMPARE, by itself, creates a new SUBMENUCOMPARE or raises the
existing
%     singleton*.
%
%     H = SUBMENUCOMPARE returns the handle to a new SUBMENUCOMPARE or the
handle to
%     the existing singleton*.
%
%     SUBMENUCOMPARE('CALLBACK',hObject,eventData,handles,...) calls the
local
%     function named CALLBACK in SUBMENUCOMPARE.M with the given input
arguments.
%
%     SUBMENUCOMPARE('Property','Value',...) creates a new SUBMENUCOMPARE or
raises the
%     existing singleton*. Starting from the left, property value pairs are
%     applied to the GUI before SubMenuCompare_OpeningFcn gets called. An
%     unrecognized property name or invalid value makes property application
%     stop. All inputs are passed to SubMenuCompare_OpeningFcn via
varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help SubMenuCompare

% Last Modified by GUIDE v2.5 09-Aug-2016 22:31:27

```

```

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',          mfilename, ...
                   'gui_Singleton',    gui_Singleton, ...
                   'gui_OpeningFcn',   @SubMenuCompare_OpeningFcn, ...
                   'gui_OutputFcn',    @SubMenuCompare_OutputFcn, ...
                   'gui_LayoutFcn',    [] , ...
                   'gui_Callback',     []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before SubMenuCompare is made visible.
function SubMenuCompare_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to SubMenuCompare (see VARARGIN)

% Choose default command line output for SubMenuCompare
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes SubMenuCompare wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = SubMenuCompare_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
compareGrayscale;

```

```

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
compareWarna;

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
menu_utama;
close SubMenuCompare;

```

comparewarna.m

```

function varargout = compareWarna(varargin)
% COMPAREWARNA MATLAB code for compareWarna.fig
%   COMPAREWARNA, by itself, creates a new COMPAREWARNA or raises the
existing
%   singleton*.
%
%   H = COMPAREWARNA returns the handle to a new COMPAREWARNA or the
handle to
%   the existing singleton*.
%
%   COMPAREWARNA('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in COMPAREWARNA.M with the given input
arguments.
%
%   COMPAREWARNA('Property','Value',...) creates a new COMPAREWARNA or
raises the
%   existing singleton*. Starting from the left, property value pairs are
%   applied to the GUI before compareWarna_OpeningFcn gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to compareWarna_OpeningFcn via varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help compareWarna

% Last Modified by GUIDE v2.5 24-Aug-2016 08:20:56

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',          mfilename, ...
                   'gui_Singleton',    gui_Singleton, ...
                   'gui_OpeningFcn',   @compareWarna_OpeningFcn, ...
                   'gui_OutputFcn',    @compareWarna_OutputFcn, ...
                   'gui_LayoutFcn',    [], ...
                   'gui_Callback',     []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});

```

```

end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before compareWarna is made visible.
function compareWarna_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin    command line arguments to compareWarna (see VARARGIN)

% Choose default command line output for compareWarna
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes compareWarna wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = compareWarna_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pilihgambar1.
function pilihgambar1_Callback(hObject, eventdata, handles)
% hObject    handle to pilihgambar1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Img1;
global X1;
X1 = uigetfile('*.bmp;*.png', 'Select Image');
Img1 = imread(X1);
axes(handles.axes1);
imshow(Img1);
guidata(hObject, handles);

% --- Executes on button press in compare.
function compare_Callback(hObject, eventdata, handles)
% hObject    handle to compare (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB

```

```

% handles      structure with handles and user data (see GUIDATA)
global Img1;
global Img2;
global Img3;
global red1;
global green1;
global blue1;
global red2;
global green2;
global blue2;
global red3;
global green3;
global blue3;
global X1;
global X2;
global X3;

red1 = Img1(:,:,:1);
axes(handles.axes3);
imhist(red1);

green1 = Img1(:,:,:2);
axes(handles.axes4);
imhist(green1);

blue1 = Img1(:,:,:3);
axes(handles.axes5);
imhist(blue1);

red2 = Img2(:,:,:1);
axes(handles.axes6);
imhist(red2);

green2 = Img2(:,:,:2);
axes(handles.axes7);
imhist(green2);

blue2 = Img2(:,:,:3);
axes(handles.axes8);
imhist(blue2);

red3 = Img3(:,:,:1);
axes(handles.axes10);
imhist(red3);

green3 = Img3(:,:,:2);
axes(handles.axes11);
imhist(green3);

blue3 = Img3(:,:,:3);
axes(handles.axes12);
imhist(blue3);

info1 = imfinfo(X1);
info2 = imfinfo(X2);
info3 = imfinfo(X3);

```

```

form = guidata(gcbo);

set(form.text2,'String',info1.Filename);
set(form.text3,'String',info2.Filename);
set(form.text44,'String',info3.Filename);

set(form.text4,'String',info1.FileSize);
set(form.text5,'String',info2.FileSize);
set(form.text45,'String',info3.FileSize);

set(form.text6,'String',info1.Width);
set(form.text7,'String',info2.Width);
set(form.text46,'String',info3.Width);

set(form.text8,'String',info1.Height);
set(form.text9,'String',info2.Height);
set(form.text47,'String',info3.Height);

set(form.text10,'String',info1.Format);
set(form.text11,'String',info2.Format);
set(form.text48,'String',info3.Format);

set(form.text12,'String',info1.BitDepth);
set(form.text13,'String',info2.BitDepth);
set(form.text49,'String',info3.BitDepth);

set(form.text14,'String',info1.ColorType);
set(form.text15,'String',info2.ColorType);
set(form.text50,'String',info3.ColorType);

% % prosen=analisa(Img1,Img2);
% % spersen=[num2str(prosen) ' %'];
% % set(handles.txthasil,'String',spersen);

% --- Executes on button press in pilihgambar2.
function pilihgambar2_Callback(hObject, eventdata, handles)
% hObject    handle to pilihgambar2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Img2;
global X2;
X2 = uigetfile('*.bmp;*.png','Select file');
Img2 = imread(X2);
axes(handles.axes2);
imshow(Img2);
guidata(hObject,handles);

% --- Executes on button press in kembali.
function kembali_Callback(hObject, eventdata, handles)
% hObject    handle to kembali (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
SubMenuCompare;
close compareWarna;

```

```
% --- Executes on button press in pilihgambar3.
function pilihgambar3_Callback(hObject, eventdata, handles)
% hObject    handle to pilihgambar3 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Img3;
global X3;
X3 = uigetfile('*.bmp;*.png','Select file');
Img3 = imread(X3);
axes(handles.axes9);
imshow(Img3);
guidata(hObject,handles);
```

comparegrayscale.m

```
function varargout = compareGrayscale(varargin)
% COMPAREGRAYSCALE MATLAB code for compareGrayscale.fig
%   COMPAREGRAYSCALE, by itself, creates a new COMPAREGRAYSCALE or raises
the existing
%   singleton*.
%
%   H = COMPAREGRAYSCALE returns the handle to a new COMPAREGRAYSCALE or
the handle to
%   the existing singleton*.
%
%   COMPAREGRAYSCALE('CALLBACK',hObject,eventData,handles,...) calls the
local
%   function named CALLBACK in COMPAREGRAYSCALE.M with the given input
arguments.
%
%   COMPAREGRAYSCALE('Property','Value',...) creates a new
COMPAREGRAYSCALE or raises the
%   existing singleton*. Starting from the left, property value pairs are
%   applied to the GUI before compareGrayscale_OpeningFcn gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to compareGrayscale_OpeningFcn via
varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help compareGrayscale

% Last Modified by GUIDE v2.5 24-Aug-2016 08:29:42

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',         mfilename, ...
                   'gui_Singleton',   gui_Singleton, ...
                   'gui_OpeningFcn', @compareGrayscale_OpeningFcn, ...
                   'gui_OutputFcn',  @compareGrayscale_OutputFcn, ...
                   'gui_LayoutFcn',  [], ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
```

```

    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before compareGrayscale is made visible.
function compareGrayscale_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to compareGrayscale (see VARARGIN)

% Choose default command line output for compareGrayscale
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes compareGrayscale wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = compareGrayscale_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pilihgambar1.
function pilihgambar1_Callback(hObject, eventdata, handles)
% hObject    handle to pilihgambar1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Img1;
global X1;
X1 = uigetfile('*.bmp;*.png','Select Image');
Img1 = imread(X1);
axes(handles.axes1);
imshow(Img1);
guidata(hObject,handles);

% --- Executes on button press in compare.
function compare_Callback(hObject, eventdata, handles)
% hObject    handle to compare (see GCBO)

```

```

% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
global Img1;
global Img2;
global Img3;
global X1;
global X2;
global X3;
global hist1;
global hist2;
global hist3;

% hist1 = rgb2gray(Img1);
% axes(handles.axes3)
% imhist(hist1);

hist1 = Img1;
axes(handles.axes3)
imhist(hist1);

% hist2 = rgb2gray(Img2);
% axes(handles.axes4)
% imhist(hist2);

hist2 = Img2;
axes(handles.axes6)
imhist(hist2);

% hist3 = rgb2gray(Img3);
% axes(handles.axes4)
% imhist(hist2);

hist3 = Img3;
axes(handles.axes4)
imhist(hist3);

info1 = imfinfo(X1);
info2 = imfinfo(X2);
info3 = imfinfo(X3);

form = guidata(gcbo);

set(form.text2,'String',info1.Filename);
set(form.text3,'String',info2.Filename);
set(form.text42,'String',info3.Filename);

set(form.text4,'String',info1.FileSize);
set(form.text5,'String',info2.FileSize);
set(form.text43,'String',info3.FileSize);

set(form.text6,'String',info1.Width);
set(form.text7,'String',info2.Width);
set(form.text44,'String',info3.Width);

set(form.text8,'String',info1.Height);
set(form.text9,'String',info2.Height);

```

```

set(form.text45,'String',info3.Height);

set(form.text10,'String',info1.Format);
set(form.text11,'String',info2.Format);
set(form.text46,'String',info3.Format);

set(form.text12,'String',info1.BitDepth);
set(form.text13,'String',info2.BitDepth);
set(form.text47,'String',info3.BitDepth);

set(form.text14,'String',info1.ColorType);
set(form.text15,'String',info2.ColorType);
set(form.text48,'String',info3.ColorType);

% --- Executes on button press in pilihgambar2.
function pilihgambar2_Callback(hObject, eventdata, handles)
% hObject    handle to pilihgambar2 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Img2;
global X2;
X2 = uigetfile('*.bmp;*.png','Select file');
Img2 = imread(X2);
axes(handles.axes2);
imshow(Img2);
guidata(hObject,handles);

% --- Executes on button press in kembali.
function kembali_Callback(hObject, eventdata, handles)
% hObject    handle to kembali (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
SubMenuCompare;
close compareGrayscale;

% --- Executes on button press in pilihgambar3.
function pilihgambar3_Callback(hObject, eventdata, handles)
% hObject    handle to pilihgambar3 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Img3;
global X3;
X3 = uigetfile('*.bmp;*.png','Select file');
Img3 = imread(X3);
axes(handles.axes9);
imshow(Img3);
guidata(hObject,handles);

```

submenuPSNR.m

```

function varargout = SubMenuPSNR(varargin)
% SUBMENUPSNR MATLAB code for SubMenuPSNR.fig
%      SUBMENUPSNR, by itself, creates a new SUBMENUPSNR or raises the
existing
%      singleton*.
%
```

```

%      H = SUBMENUPSNR returns the handle to a new SUBMENUPSNR or the handle
% to
%      the existing singleton*.
%
%      SUBMENUPSNR('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in SUBMENUPSNR.M with the given input
arguments.
%
%      SUBMENUPSNR('Property','Value',...) creates a new SUBMENUPSNR or
raises the
%      existing singleton*. Starting from the left, property value pairs are
% applied to the GUI before SubMenuPSNR_OpeningFcn gets called. An
% unrecognized property name or invalid value makes property application
% stop. All inputs are passed to SubMenuPSNR_OpeningFcn via varargin.
%
% *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
% instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help SubMenuPSNR

% Last Modified by GUIDE v2.5 23-Aug-2016 10:09:11

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',         mfilename, ...
                   'gui_Singleton',    gui_Singleton, ...
                   'gui_OpeningFcn',   @SubMenuPSNR_OpeningFcn, ...
                   'gui_OutputFcn',    @SubMenuPSNR_OutputFcn, ...
                   'gui_LayoutFcn',    [], ...
                   'gui_Callback',     []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before SubMenuPSNR is made visible.
function SubMenuPSNR_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to SubMenuPSNR (see VARARGIN)

% Choose default command line output for SubMenuPSNR
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

```

```

% UIWAIT makes SubMenuPSNR wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = SubMenuPSNR_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
comMSEgray;

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
comMSE;

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
menu_utama;
close SubMenuPSNR;

```

comMSEgray.m

```

function varargout = comMSEgray(varargin)
% COMMSEGRAY MATLAB code for comMSEgray.fig
%     COMMSEGRAY, by itself, creates a new COMMSEGRAY or raises the existing
%     singleton*.
%
%     H = COMMSEGRAY returns the handle to a new COMMSEGRAY or the handle to
%     the existing singleton*.
%
%     COMMSEGRAY( 'CALLBACK', hObject, eventData, handles,...) calls the local
%     function named CALLBACK in COMMSEGRAY.M with the given input
% arguments.
%
%     COMMSEGRAY( 'Property','Value',....) creates a new COMMSEGRAY or raises
% the
%     existing singleton*. Starting from the left, property value pairs are
%     applied to the GUI before comMSEgray_OpeningFcn gets called. An
%     unrecognized property name or invalid value makes property application

```

```

%
% stop. All inputs are passed to comMSEgray_OpeningFcn via varargin.
%
% *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
% instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help comMSEgray

% Last Modified by GUIDE v2.5 23-Aug-2016 09:57:27

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',          mfilename, ...
                   'gui_Singleton',    gui_Singleton, ...
                   'gui_OpeningFcn',   @comMSEgray_OpeningFcn, ...
                   'gui_OutputFcn',    @comMSEgray_OutputFcn, ...
                   'gui_LayoutFcn',    [] , ...
                   'gui_Callback',     []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

%
% --- Executes just before comMSEgray is made visible.
function comMSEgray_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to comMSEgray (see VARARGIN)

% Choose default command line output for comMSEgray
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes comMSEgray wait for user response (see UIRESUME)
% uiwait(handles.figure1);

%
% --- Outputs from this function are returned to the command line.
function varargout = comMSEgray_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

```

```

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pilihgambar1.
function pilihgambar1_Callback(hObject, eventdata, handles)
% hObject    handle to pilihgambar1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Img1;
global info1;
Img1=[];
[filename1,pathfile1]= uigetfile({'*.bmp; *.png'},'pilih gambar');
if isequal(filename1,0)
return;
end

Img1 = imread(strcat(pathfile1,filename1));
fullpath1=fullfile(pathfile1,filename1);
info1 = imfinfo(fullpath1);
axes(handles.axes1)
imshow(Img1);

set(handles.axes1,'Userdata',Img1);
guidata(hObject,handles);

% --- Executes on button press in pilihgambar2.
function pilihgambar2_Callback(hObject, eventdata, handles)
% hObject    handle to pilihgambar2 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Img2;
global info2;
Img2=[];
[filename2,pathfile2] = uigetfile({'*.bmp; *.png'},'pilih gambar');
if isequal(filename2,0)
return;
end

Img2 = imread(strcat(pathfile2,filename2));
fullpath2=fullfile(pathfile2,filename2);
info2 = imfinfo(fullpath2);
axes(handles.axes2)
imshow(Img2);

set(handles.axes2,'Userdata',Img2);
guidata(hObject,handles);

% --- Executes on button press in proses.
function proses_Callback(hObject, eventdata, handles)
% hObject    handle to proses (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Img1;
global Img2;
global info1;
global info2;

```

```

cocok =0;
nilai_putih=0;
nilai_hitam=0;

Img1=get(handles.axes1,'Userdata');
Img2=get(handles.axes2,'Userdata');

if isequal(Img1,[])
warndlg('Tidak Ada Gambar 1, Silahkan Masukkan Gambar 1 !!!',['PERINGATAN'])
elseif isequal(Img2,[])
    warndlg('Tidak Ada Gambar 2, Silahkan Masukkan Gambar 2
!!!',['PERINGATAN'])
else

Img2b=Img2;
deteksi2=edge(Img2b,'canny');
axes(handles.axes3)
imshow(deteksi2);
guidata(hObject,handles);

Img1b=Img1;
deteksil=edge(Img1b,'canny');
axes(handles.axes4)
imshow(deteksil);
guidata(hObject,handles);

form = guidata(gcbo);

set(form.text12,'String',info1.Filename);
set(form.text13,'String',info2.Filename);

set(form.text14,'String',info1.FileSize);
set(form.text15,'String',info2.FileSize);

set(form.text16,'String',info1.Width);
set(form.text17,'String',info2.Width);
o=get(handles.text16,'String');
p=get(handles.text17,'String');
h=strcmp(o,p);

set(form.text18,'String',info1.Height);
set(form.text19,'String',info2.Height);
v=get(handles.text18,'String');
f=get(handles.text19,'String');
j=strcmp(v,f);

set(form.text20,'String',info1.Format);
set(form.text21,'String',info2.Format);

set(form.text22,'String',info1.BitDepth);
set(form.text23,'String',info2.BitDepth);

set(form.text24,'String',info1.ColorType);
set(form.text25,'String',info2.ColorType);

```

```

if (h==0 || j==0);
warndlg('Lebar atau Tinggi Gambar Tidak Sama Sehingga MSE dan PSNR Tidak Ada',[ 'PERINGATAN' ])
else
for k=1:1:256
    for l=1:1:256
        if(deteksi1(k,l)==1)
            nilai_putih = nilai_putih+1;
        else
            nilai_hitam = nilai_hitam+1;
        end
    end
end

for i = 1:1:256
    for j = 1:1:256
        if(deteksi1(i,j)==1)&&(deteksi2(i,j)==1)
            cocok=cocok+1;
        else
        end
    end
end
total_data=nilai_putih;
persen=(cocok/total_data)*100;
set(form.txtHasil,'String',[ (num2str(persen)), ' %']);

% % prosen=analisa(Img1,Img2);
% % spersen=[num2str(prosen) ' %'];
% % set(handles.txthasil,'String',spersen);

I = double(Img1);
Ihat = double (Img2);

% Read the dimensions of the image.
[rows columns ~] = size(I);

% Calculate mean square error of R, G, B.
mseImage = (double(I) - double(Ihat)) .^ 2;
[rows, columns] = size (I);
mse = sum(mseImage(:)) / (rows * columns);

% Calculate text27 (Peak Signal to noise ratio).
psnr = 10 * log10( 255^2 / mse);
set(handles.mse,'String',num2str(mse,'%7.2f dB'));
set(handles.psnr,'String',num2str(psnr,'%7.2f dB'));
end
end

% --- Executes on button press in reset.
function reset_Callback(hObject, eventdata, handles)
% hObject    handle to reset (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

```

```

% handles      structure with handles and user data (see GUIDATA)
mau=questdlg(['Membersihkan '
get(handles.figure1,'Name') ''],[ 'BERSIHKAN' ],'Ya','Tidak','Ya');
if strcmp(mau,'Tidak')
return;
else
    set(handles.axes1,'Userdata',[]);
    set(handles.axes2,'Userdata',[]);
    set(handles.text12,'String','');
    set(handles.text13,'String','');
    set(handles.text14,'String','');
    set(handles.text15,'String','');
    set(handles.text16,'String','');
    set(handles.text17,'String','');
    set(handles.text18,'String','');
    set(handles.text19,'String','');
    set(handles.text20,'String','');
    set(handles.text21,'String','');
    set(handles.text22,'String','');
    set(handles.text23,'String','');
    set(handles.text24,'String','');
    set(handles.text25,'String','');
    set(handles.mse,'String','');
    set(handles.psnr,'String','');
    set(handles.txtHasil,'String','');

    arrayfun(@cla,findall(0,'type','axes'))
    cla(handles.axes1,'reset');
    cla(handles.axes2,'reset');
    cla(handles.axes3,'reset');
    cla(handles.axes4,'reset');

    set(handles.axes1,'visible','off')
    set(handles.axes2,'visible','off')
    set(handles.axes3,'visible','off')
    set(handles.axes4,'visible','off')
hm = msgbox('Membersihkan Sukses',' BERSIHKAN','help');
end

% --- Executes on button press in kembali.
function kembali_Callback(hObject, eventdata, handles)
% hObject    handle to kembali (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close comMSEgray;
SubMenuPSNR;

function mse_Callback(hObject, eventdata, handles)
% hObject    handle to mse (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of mse as text
%        str2double(get(hObject,'String')) returns contents of mse as a
double

```

```

% --- Executes during object creation, after setting all properties.
function mse_CreateFcn(hObject, eventdata, handles)
% hObject    handle to mse (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject, 'BackgroundColor'),
get(0, 'defaultUicontrolBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end


function psnr_Callback(hObject, eventdata, handles)
% hObject    handle to psnr (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of psnr as text
%        str2double(get(hObject,'String')) returns contents of psnr as a
double


% --- Executes during object creation, after setting all properties.
function psnr_CreateFcn(hObject, eventdata, handles)
% hObject    handle to psnr (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject, 'BackgroundColor'),
get(0, 'defaultUicontrolBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end


function persen_Callback(hObject, eventdata, handles)
% hObject    handle to persen (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of persen as text
%        str2double(get(hObject,'String')) returns contents of persen as a
double


% --- Executes during object creation, after setting all properties.
function persen_CreateFcn(hObject, eventdata, handles)
% hObject    handle to persen (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB

```

```

% handles      empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%        See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

comMSE.m

function varargout = comMSE(varargin)
% COMMSE MATLAB code for comMSE.fig
%   COMMSE, by itself, creates a new COMMSE or raises the existing
%   singleton*.
%
%   H = COMMSE returns the handle to a new COMMSE or the handle to
%   the existing singleton*.
%
%   COMMSE('CALLBACK', hObject, eventData, handles,...) calls the local
%   function named CALLBACK in COMMSE.M with the given input arguments.
%
%   COMMSE('Property','Value',...) creates a new COMMSE or raises the
%   existing singleton*. Starting from the left, property value pairs are
%   applied to the GUI before comMSE_OpeningFcn gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to comMSE_OpeningFcn via varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help comMSE

% Last Modified by GUIDE v2.5 23-Aug-2016 19:20:12

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',          mfilename, ...
                   'gui_Singleton',    gui_Singleton, ...
                   'gui_OpeningFcn',   @comMSE_OpeningFcn, ...
                   'gui_OutputFcn',    @comMSE_OutputFcn, ...
                   'gui_LayoutFcn',    [], ...
                   'gui_Callback',     []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

```

```

% --- Executes just before comMSE is made visible.
function comMSE_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to comMSE (see VARARGIN)

% Choose default command line output for comMSE
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes comMSE wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = comMSE_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pilihgambar1.
function pilihgambar1_Callback(hObject, eventdata, handles)
% hObject    handle to pilihgambar1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Img1;
global infol;
Img1=[];
[filename1,pathfile1]= uigetfile({'*.bmp; *.png'},'pilih gambar');
if isequal(filename1,0)
return;
end

Img1 = imread(strcat(pathfile1,filename1));
fullpath1=fullfile(pathfile1,filename1);
infol = imfinfo(fullpath1);
axes(handles.axes1)
imshow(Img1);

set(handles.axes1,'Userdata',Img1);
guidata(hObject,handles);

% --- Executes on button press in pilihgambar2.
function pilihgambar2_Callback(hObject, eventdata, handles)
% hObject    handle to pilihgambar2 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

```

```

global Img2;
global info2;
Img2=[];
[filename2,pathfile2] = uigetfile({'*.bmp; *.png'},'pilih gambar');
if isequal(filename2,0)
return;
end

Img2 = imread(strcat(pathfile2,filename2));
fullpath2=fullfile(pathfile2,filename2);
info2 = imfinfo(fullpath2);
axes(handles.axes2)
imshow(Img2);

set(handles.axes2,'Userdata',Img2);
guidata(hObject,handles);

% --- Executes on button press in btnProses.
function btnProses_Callback(hObject, eventdata, handles)
% hObject    handle to btnProses (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Img1;
global Img2;
global info1;
global info2;
cocok =0;
nilai_putih=0;
nilai_hitam=0;

Img1=get(handles.axes1,'Userdata');
Img2=get(handles.axes2,'Userdata');

if isequal(Img1,[])
warndlg('Tidak Ada Gambar 1, Silahkan Masukkan Gambar 1 !!!',['PERINGATAN'])
elseif isequal(Img2,[])
    warndlg('Tidak Ada Gambar 2, Silahkan Masukkan Gambar 2
!!!',['PERINGATAN'])
else

Img2b=rgb2gray(Img2);
deteksi2=edge(Img2b,'canny');
axes(handles.axes3)
imshow(deteksi2);
guidata(hObject,handles);

Img1b=rgb2gray(Img1);
deteksil1=edge(Img1b,'canny');
axes(handles.axes4)
imshow(deteksil1);
guidata(hObject,handles);

form = guidata(gcbo);

set(form.text36,'String',info1.Filename);
set(form.text37,'String',info2.Filename);

```

```

set(form.text38,'String',info1.FileSize);
set(form.text39,'String',info2.FileSize);

set(form.text40,'String',info1.Width);
set(form.text41,'String',info2.Width);
o=get(handles.text40,'String');
p=get(handles.text41,'String');
h=strcmp(o,p);

set(form.text42,'String',info1.Height);
set(form.text43,'String',info2.Height);
v=get(handles.text42,'String');
f=get(handles.text43,'String');
j=strcmp(v,f);

set(form.text44,'String',info1.Format);
set(form.text45,'String',info2.Format);

set(form.text46,'String',info1.BitDepth);
set(form.text47,'String',info2.BitDepth);

set(form.text48,'String',info1.ColorType);
set(form.text49,'String',info2.ColorType);

if (h==0 || j==0);
warndlg('Lebar atau Tinggi Gambar Tidak Sama Sehingga MSE dan PSNR Tidak Ada',[PERINGATAN])
else
for k=1:1:256
    for l=1:1:256
        if(deteksi1(k,l)==1)
            nilai_putih = nilai_putih+1;
        else
            nilai_hitam = nilai_hitam+1;
        end
    end
end

for i = 1:1:256
    for j = 1:1:256
        if(deteksi1(i,j)==1)&&(deteksi2(i,j)==1)
            cocok=cocok+1;
        else
            end
        end
    end
end
total_data=nilai_putih;
persen=(cocok/total_data)*100;
set(form.txtHasil,'String',[num2str(persen),'%']);
% % total_data=nilai_putih;
% % persen=(cocok/total_data)*100;
% % set(form.persen,'String',[num2str(persen,'%7.2f')),'%']);

```

```

I = Img1;
Ihat = Img2;

% Read the dimensions of the image.
[rows columns ~] = size(I);

% Calculate mean square error of R, G, B.
mseRImage = (double(I(:,:,1)) - double(Ihat(:,:,1))) .^ 2;
mseGImage = (double(I(:,:,2)) - double(Ihat(:,:,2))) .^ 2;
mseBImage = (double(I(:,:,3)) - double(Ihat(:,:,3))) .^ 2;

mseR = sum(sum(mseRImage)) / (rows * columns);
mseG = sum(sum(mseGImage)) / (rows * columns);
mseB = sum(sum(mseBImage)) / (rows * columns);

% Average mean square error of R, G, B.
mse = (mseR + mseG + mseB)/3;

% Calculate text3 (Peak Signal to noise ratio).
psnr = 10 * log10( 255^2 / mse);
set(handles.mse,'String',num2str(mse,'%7.2f dB'));
set(handles.psnr,'String',num2str(psnr,'%7.2f dB'));
end
end

% --- Executes on button press in btnReset.
function btnReset_Callback(hObject, eventdata, handles)
% hObject    handle to btnReset (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
mau=questdlg(['Membersihkan '
get(handles.figure1,'Name') ''],[ 'BERSIKHAN' ],'Ya','Tidak','Ya');
if strcmp(mau,'Tidak')
return;
else
set(handles.axes1,'Userdata',[ ]);
set(handles.axes2,'Userdata',[ ]);
set(handles.text36,'String','');
set(handles.text37,'String','');
set(handles.text38,'String','');
set(handles.text39,'String','');
set(handles.text40,'String','');
set(handles.text41,'String','');
set(handles.text42,'String','');
set(handles.text43,'String','');
set(handles.text44,'String','');
set(handles.text45,'String','');
set(handles.text46,'String','');
set(handles.text47,'String','');
set(handles.text48,'String','');
set(handles.text49,'String','');
set(handles.mse,'String','');
set(handles.psnr,'String','');

```

```

set(handles.txtHasil,'String','');

arrayfun(@cla,findall(0,'type','axes'))
cla(handles.axes1,'reset');
cla(handles.axes2,'reset');
cla(handles.axes3,'reset');
cla(handles.axes4,'reset');

set(handles.axes1,'visible','off')
set(handles.axes2,'visible','off')
set(handles.axes3,'visible','off')
set(handles.axes4,'visible','off')
hm = msgbox('Membersihkan Sukses','BERSIHKAN','help');
end

% --- Executes on button press in kembali.
function kembali_Callback(hObject, eventdata, handles)
% hObject    handle to kembali (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close comMSE;
SubMenuPSNR;

function mse_Callback(hObject, eventdata, handles)
% hObject    handle to text2 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of text2 as text
%        str2double(get(hObject,'String')) returns contents of text2 as a
double

% --- Executes during object creation, after setting all properties.
function mse_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text2 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function psnr_Callback(hObject, eventdata, handles)
% hObject    handle to text3 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of text3 as text
%        str2double(get(hObject,'String')) returns contents of text3 as a

```

```

double

% --- Executes during object creation, after setting all properties.
function psnr_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text3 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%        See ISPC and COMPUTER.
if ispc && isequal(get(hObject, 'BackgroundColor'),
get(0, 'defaultUicontrolBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end


function persen_Callback(hObject, eventdata, handles)
% hObject    handle to persen (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of persen as text
%        str2double(get(hObject,'String')) returns contents of persen as a
double

% --- Executes during object creation, after setting all properties.
function persen_CreateFcn(hObject, eventdata, handles)
% hObject    handle to persen (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%        See ISPC and COMPUTER.
if ispc && isequal(get(hObject, 'BackgroundColor'),
get(0, 'defaultUicontrolBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end


% --- Executes on button press in pilihgambar1.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pilihgambar1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in pilihgambar2.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pilihgambar2 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

```

```

% --- Executes on button press in btnProses.
function proses_Callback(hObject, eventdata, handles)
% hObject    handle to btnProses (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in btnReset.
function reset_Callback(hObject, eventdata, handles)
% hObject    handle to btnReset (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in kembali.
function pushbutton10_Callback(hObject, eventdata, handles)
% hObject    handle to kembali (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to mse (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of mse as text
%        str2double(get(hObject,'String')) returns contents of mse as a
double

% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to mse (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject, 'BackgroundColor'),
get(0, 'defaultUicontrolBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end

function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to psnr (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of psnr as text
%        str2double(get(hObject,'String')) returns contents of psnr as a

```

```

double

% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to psnr (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject, 'BackgroundColor'),
get(0, 'defaultUicontrolBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end


function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to persen (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of persen as text
%        str2double(get(hObject,'String')) returns contents of persen as a
double

% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to persen (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject, 'BackgroundColor'),
get(0, 'defaultUicontrolBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end

```

submenuhistogram.m

```

function varargout = SubMenuHistogram(varargin)
% SUBMENUHISTOGRAM MATLAB code for SubMenuHistogram.fig
%      SUBMENUHISTOGRAM, by itself, creates a new SUBMENUHISTOGRAM or raises
the existing
%      singleton*.
%
%      H = SUBMENUHISTOGRAM returns the handle to a new SUBMENUHISTOGRAM or
the handle to
%      the existing singleton*.
%
%      SUBMENUHISTOGRAM('CALLBACK', hObject, eventData, handles, ...) calls the
local
%      function named CALLBACK in SUBMENUHISTOGRAM.M with the given input

```

```

arguments.
%
%     SUBMENUHISTOGRAM('Property','Value',...) creates a new
%     SUBMENUHISTOGRAM or raises the
%     existing singleton*. Starting from the left, property value pairs are
%     applied to the GUI before SubMenuHistogram_OpeningFcn gets called. An
%     unrecognized property name or invalid value makes property application
%     stop. All inputs are passed to SubMenuHistogram_OpeningFcn via
varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help SubMenuHistogram

% Last Modified by GUIDE v2.5 23-Aug-2016 10:35:58

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',          mfilename, ...
                   'gui_Singleton',    gui_Singleton, ...
                   'gui_OpeningFcn',   @SubMenuHistogram_OpeningFcn, ...
                   'gui_OutputFcn',    @SubMenuHistogram_OutputFcn, ...
                   'gui_LayoutFcn',    [] , ...
                   'gui_Callback',     []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

%
% --- Executes just before SubMenuHistogram is made visible.
function SubMenuHistogram_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to SubMenuHistogram (see VARARGIN)

% Choose default command line output for SubMenuHistogram
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes SubMenuHistogram wait for user response (see UIRESUME)
% uiwait(handles.figure1);

```

```

% --- Outputs from this function are returned to the command line.
function varargout = SubMenuHistogram_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
histogramGray;

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
histogram;

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
menu_utama;
close SubMenuHistogram;

```

histogramgray.m

```

function varargout = histogramGray(varargin)
% HISTOGRAMGRAY MATLAB code for histogramGray.fig
%     HISTOGRAMGRAY, by itself, creates a new HISTOGRAMGRAY or raises the
existing
%     singleton*.
%
%     H = HISTOGRAMGRAY returns the handle to a new HISTOGRAMGRAY or the
handle to
%     the existing singleton*.
%
%     HISTOGRAMGRAY( 'CALLBACK',hObject,eventData,handles,...) calls the
local
%     function named CALLBACK in HISTOGRAMGRAY.M with the given input
arguments.
%
%     HISTOGRAMGRAY( 'Property','Value',...) creates a new HISTOGRAMGRAY or
raises the
%     existing singleton*. Starting from the left, property value pairs are
%     applied to the GUI before histogramGray_OpeningFcn gets called. An
%     unrecognized property name or invalid value makes property application
%     stop. All inputs are passed to histogramGray_OpeningFcn via varargin.

```

```

%
% *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
% instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help histogramGray

% Last Modified by GUIDE v2.5 23-Aug-2016 10:45:24

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',          mfilename, ...
                   'gui_Singleton',    gui_Singleton, ...
                   'gui_OpeningFcn',   @histogramGray_OpeningFcn, ...
                   'gui_OutputFcn',    @histogramGray_OutputFcn, ...
                   'gui_LayoutFcn',    [], ...
                   'gui_Callback',     []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

%
% --- Executes just before histogramGray is made visible.
function histogramGray_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to histogramGray (see VARARGIN)

% Choose default command line output for histogramGray
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes histogramGray wait for user response (see UIRESUME)
% uiwait(handles.figure1);

%
% --- Outputs from this function are returned to the command line.
function varargout = histogramGray_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure

```

```

varargout{1} = handles.output;

% --- Executes on button press in pilihgambar.
function pilihgambar_Callback(hObject, eventdata, handles)
% hObject    handle to pilihgambar (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Img1;
global X1;
X1 = uigetfile('*.bmp;*.png;*.jpg','Select Image');
Img1 = imread(X1);
axes(handles.axes5);
imshow(Img1);
guidata(hObject,handles);

% --- Executes on button press in proses.
function proses_Callback(hObject, eventdata, handles)
% hObject    handle to proses (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Img1;
% % global Img2;
global X1;
% % global X2;
global hist1;
% % global hist2;

hist1 = Img1;
axes(handles.axes6)
imhist(hist1);

% % hist2 = Img2;
% % axes(handles.axes6)
% % imhist(hist2);

% --- Executes on button press in kembali.
function kembali_Callback(hObject, eventdata, handles)
% hObject    handle to kembali (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
SubMenuHistogram;
close histogramGray;

```

histogram.m

```

function varargout = histogram(varargin)
% HISTOGRAM MATLAB code for histogram.fig
%     HISTOGRAM, by itself, creates a new HISTOGRAM or raises the existing
%     singleton*.
%
%     H = HISTOGRAM returns the handle to a new HISTOGRAM or the handle to
%     the existing singleton*.
%
%     HISTOGRAM('CALLBACK', hObject, eventData, handles, ...) calls the local
%     function named CALLBACK in HISTOGRAM.M with the given input arguments.
%
```

```

%      HISTOGRAM('Property','Value',...) creates a new HISTOGRAM or raises
the
%      existing singleton*. Starting from the left, property value pairs are
%      applied to the GUI before histogram_OpeningFcn gets called. An
%      unrecognized property name or invalid value makes property application
%      stop. All inputs are passed to histogram_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help histogram

% Last Modified by GUIDE v2.5 23-Aug-2016 08:55:32

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',          mfilename, ...
                   'gui_Singleton',    gui_Singleton, ...
                   'gui_OpeningFcn',   @histogram_OpeningFcn, ...
                   'gui_OutputFcn',    @histogram_OutputFcn, ...
                   'gui_LayoutFcn',    [] , ...
                   'gui_Callback',     []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before histogram is made visible.
function histogram_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to histogram (see VARARGIN)

% Choose default command line output for histogram
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes histogram wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = histogram_OutputFcn(hObject, eventdata, handles)

```

```

% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
global Img1;
global X1;
X1 = uigetfile('*.bmp;*.png;*.jpg','Select Image');
Img1 = imread(X1);
axes(handles.axes1);
imshow(Img1);
guidata(hObject,handles);

% --- Executes on button press in proses.
function proses_Callback(hObject, eventdata, handles)
% hObject    handle to proses (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
global Img1;
global red1;
global green1;
global blue1;
global X1;

red1 = Img1(:,:,:,1);
axes(handles.axes2);
imhist(red1);

green1 = Img1(:,:,:,:,2);
axes(handles.axes4);
imhist(green1);

blue1 = Img1(:,:,:,:,3);
axes(handles.axes5);
imhist(blue1);

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
SubMenuHistogram;
close histogram;

```

tentang.m

```

function varargout = tentang(varargin)
% TENTANG MATLAB code for tentang.fig
%     TENTANG, by itself, creates a new TENTANG or raises the existing
%     singleton*.
%
% H = TENTANG returns the handle to a new TENTANG or the handle to
% the existing singleton*.
%
% TENTANG( 'CALLBACK', hObject,eventData,handles,...) calls the local
% function named CALLBACK in TENTANG.M with the given input arguments.
%
% TENTANG('Property','Value',...) creates a new TENTANG or raises the
% existing singleton*. Starting from the left, property value pairs are
% applied to the GUI before tentang_OpeningFcn gets called. An
% unrecognized property name or invalid value makes property application
% stop. All inputs are passed to tentang_OpeningFcn via varargin.
%
% *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
% instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help tentang

% Last Modified by GUIDE v2.5 09-Aug-2016 22:55:57

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',         mfilename, ...
                   'gui_Singleton',    gui_Singleton, ...
                   'gui_OpeningFcn',   @tentang_OpeningFcn, ...
                   'gui_OutputFcn',    @tentang_OutputFcn, ...
                   'gui_LayoutFcn',    [], ...
                   'gui_Callback',     []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

%
% --- Executes just before tentang is made visible.
function tentang_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to tentang (see VARARGIN)

% Choose default command line output for tentang
handles.output = hObject;

```

```

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes tentang wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = tentang_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
menu_utama;
close bantuan;

```

Bernoulli.m

```

function[e] = bernoulli(~, N, A, B)

x = zeros(1,N);

x(1) = A;
r = B;
for i = 1:N
x(i+1) = mod(r*(x(i)),1);%rumus bernoulli
end
x(2)
x(3)
c=abs(x);
for j = 1:N+1
c(j) = c(j)*10000;
end
c=floor(c);
c=mod(c,256);
d=uint8(c);
for i = 1:N
e(i,:)=d(i);
end

```

bantuan.m

```

function varargout = bantuan(varargin)
% BANTUAN MATLAB code for bantuan.fig
% BANTUAN, by itself, creates a new BANTUAN or raises the existing
% singleton*.

```

```

%
% H = BANTUAN returns the handle to a new BANTUAN or the handle to
% the existing singleton*.
%
% BANTUAN( 'CALLBACK',hObject,eventData,handles,...) calls the local
% function named CALLBACK in BANTUAN.M with the given input arguments.
%
% BANTUAN('Property','Value',...) creates a new BANTUAN or raises the
% existing singleton*. Starting from the left, property value pairs are
% applied to the GUI before bantuan_OpeningFcn gets called. An
% unrecognized property name or invalid value makes property application
% stop. All inputs are passed to bantuan_OpeningFcn via varargin.
%
% *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
% instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help bantuan

% Last Modified by GUIDE v2.5 12-Aug-2016 11:20:32

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',          mfilename, ...
                   'gui_Singleton',    gui_Singleton, ...
                   'gui_OpeningFcn',   @bantuan_OpeningFcn, ...
                   'gui_OutputFcn',    @bantuan_OutputFcn, ...
                   'gui_LayoutFcn',    [], ...
                   'gui_Callback',     []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

%
% --- Executes just before bantuan is made visible.
function bantuan_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to bantuan (see VARARGIN)

% Choose default command line output for bantuan
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

```

```

% UIWAIT makes bantuan wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = bantuan_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in kembali.
function kembali_Callback(hObject, eventdata, handles)
% hObject handle to kembali (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
menu_utama;
close bantuan;

function edit1_Callback(hObject, eventdata, handles)
% hObject handle to edit1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
% str2double(get(hObject,'String')) returns contents of edit1 as a
double

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject handle to edit1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject, 'BackgroundColor'),
get(0, 'defaultUicontrolBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end

```

analisa.m

```

function [persen] = analisa(Img1,Img2)

[x,y,z] = size (Img1);
if (z==1);

```

```

else
    Img1 = rgb2gray(Img1);
end

[x,y,z] = size (Img2);
if (z==1);
else
    Img2 = rgb2gray(Img2);
end

deteksi_tepi_Img1 = edge(Img1,'Sobel');
deteksi_tepi_Img2 = edge(Img2,'Sobel');

cocok = 0;
nilai_putih = 0;
nilai_hitam = 0;
x=0;
y=0;
l=0;
m=0;

for a = 1:1:256
    for b = 1:1:256
        if(deteksi_tepi_Img1(a,b)==1)
            nilai_putih = nilai_putih+1;
        else
            nilai_hitam = nilai_hitam+1;
        end
    end
end

for i = 1:1:256
    for j = 1:1:256
        if(deteksi_tepi_Img1(i,j)==1)&&(deteksi_tepi_Img2(i,j)==1)
            cocok = cocok+1;
        else
            end
        end
    end
end

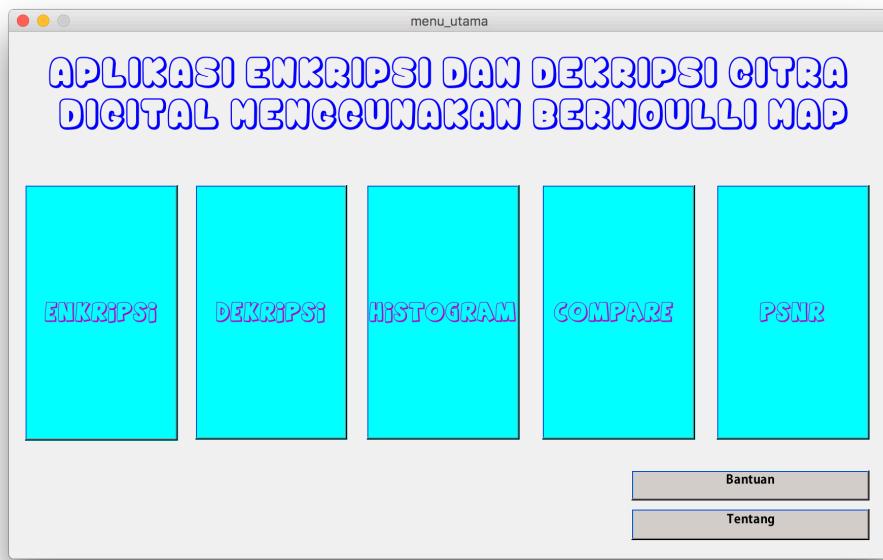
total_data = nilai_putih;
if(cocok==0)
    persen=0;
else
    persen = (cocok/total_data)*100;
end

```

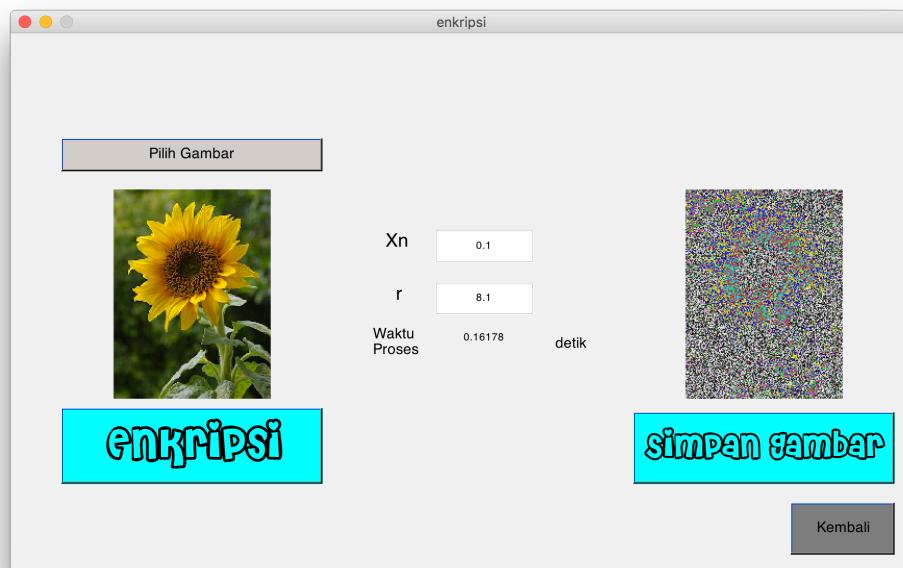
LAMPIRAN 2

OUTPUT PROGRAM

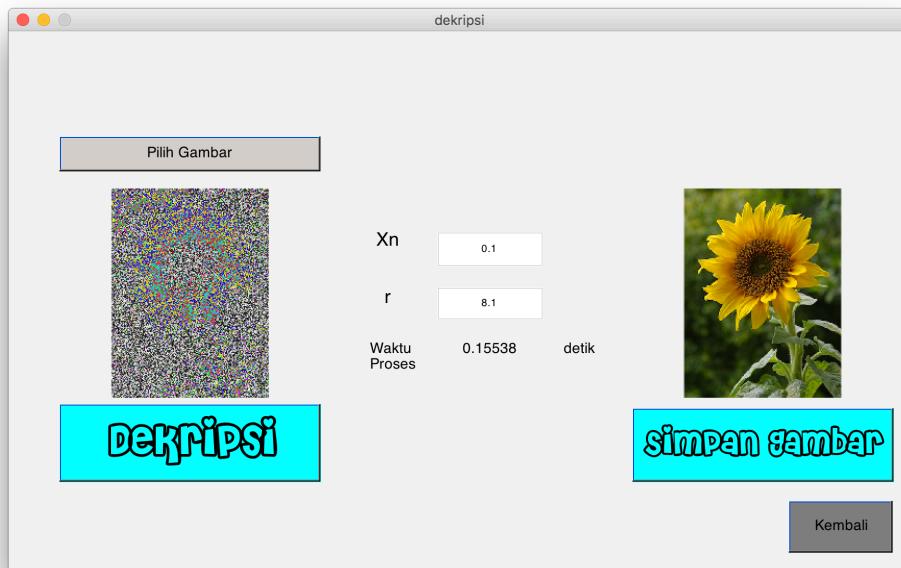
Tampilan menu utama



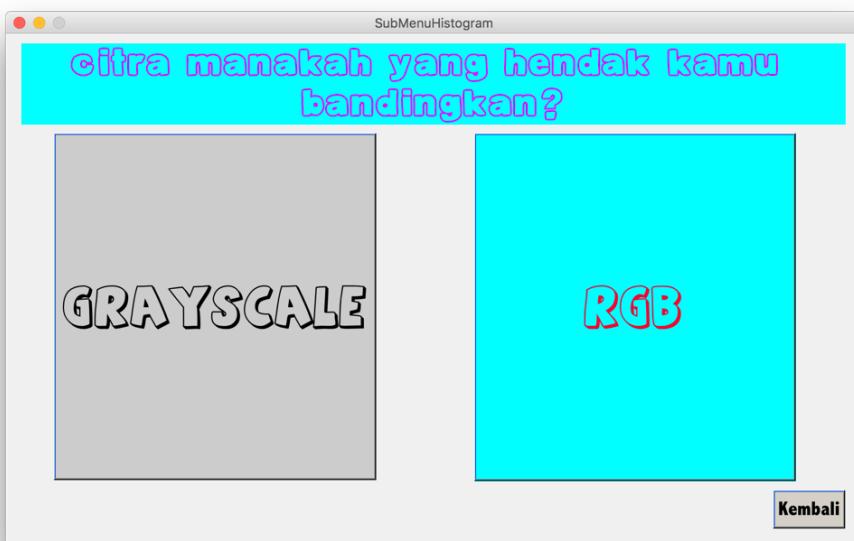
Tampilan Menu Enkripsi



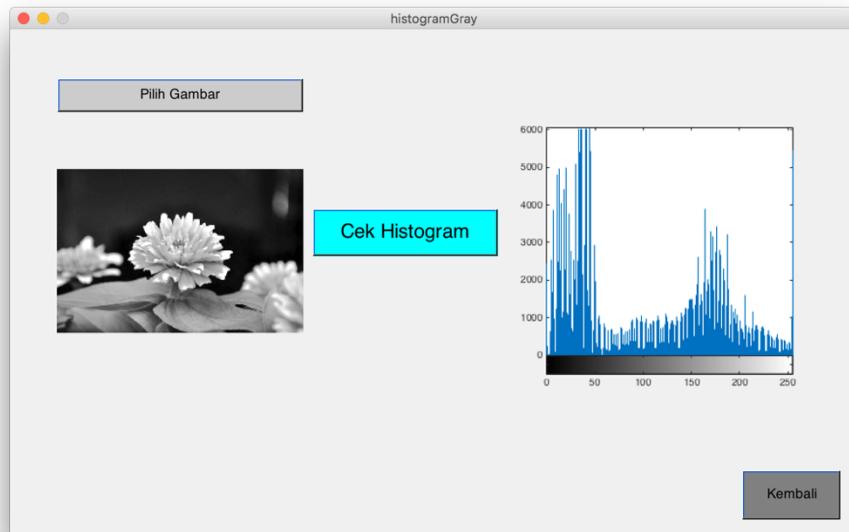
Tampilan Menu Dekripsi



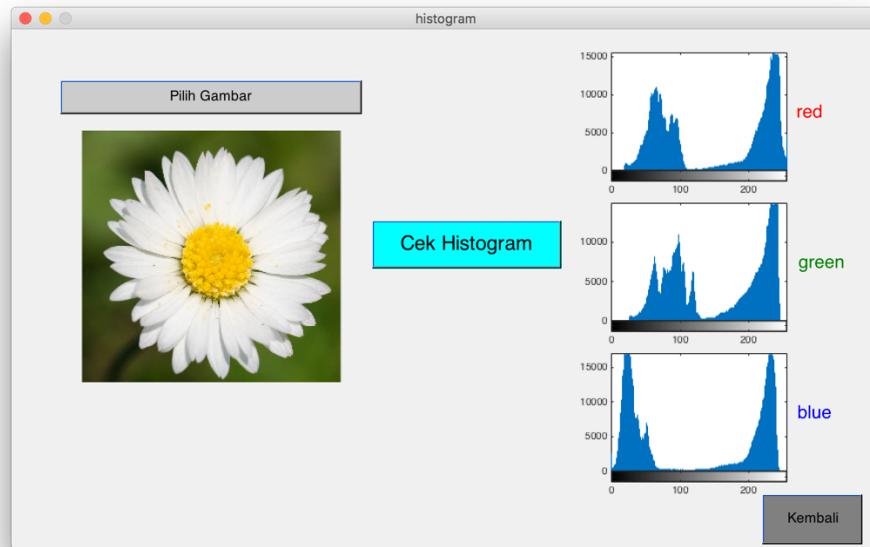
Tampilan Submenu



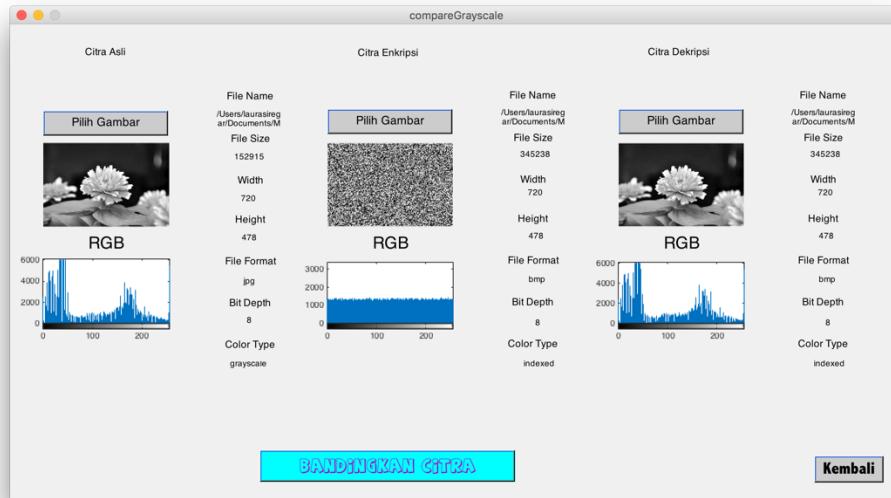
Tampilan Histogram Grayscale



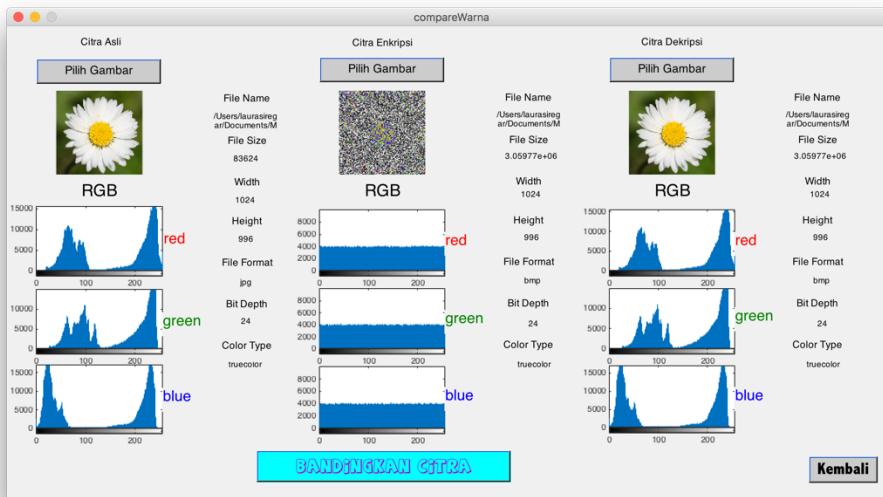
Tampilan Histogram Berwarna



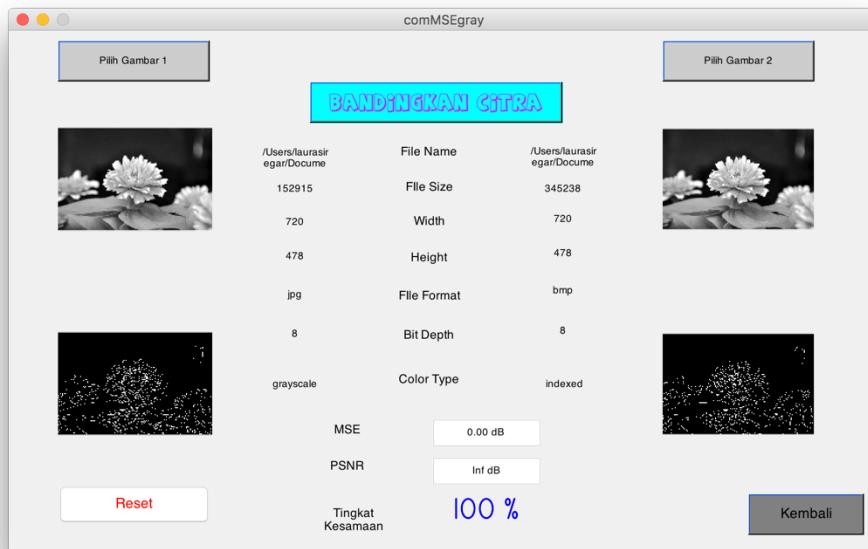
Tampilan Menu Perbandingan Grayscale



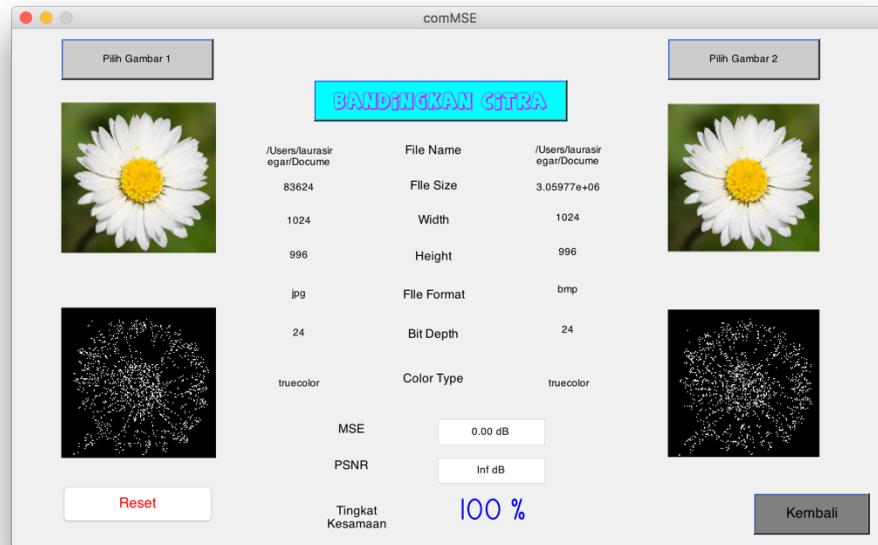
Tampilan Menu Perbandingan Citra Berwarna (RGB)



Tampilan PSNR Grayscale



Tampilan PSNR Bewarna



LAMPIRAN 3
ANALISIS

Tabel 4.2 Tabel Waktu Enkripsi dan Dekripsi untuk Data Uji Citra *Grayscale* Format BMP

Data ke-	Nama File	Ukuran Citra (pixel)	Waktu Enkripsi (detik)	Rata-Rata Dekripsi (detik)
1	abu 1.bmp	640 x 425	0.20614	0.20246
2	abu 2.bmp	800 x 531	0.32379	0.31078
3	abu 3.bmp	1024 x 680	0.51013	0.50685
4	abu 4.bmp	1280 x 850	0.77744	0.81148
5	abu 5.bmp	1627 x 1080	1.2944	1.3082

Tabel 4.3 Tabel Waktu Enkripsi dan Dekripsi untuk Data Uji Citra Berwarna (RGB) Format BMP

Data ke-	Nama File	Ukuran Citra (pixel)	Waktu Enkripsi (detik)	Rata-Rata Dekripsi (detik)
1	matahari 1.bmp	360 x 480	0.16178	0.15538
2	matahari 2.bmp	450 x 600	0.23972	0.23636
3	matahari 3.bmp	576 x 768	0.39423	0.39414
4	matahari 4.bmp	769 x 1024	0.73555	0.72582
5	matahari 5.bmp	810 x 1080	0.78938	0.78508

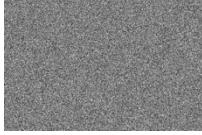
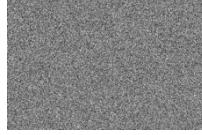
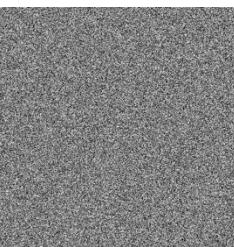
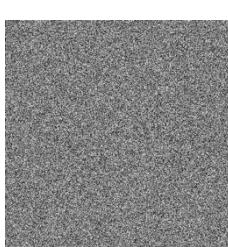
Tabel 4.4 Tabel Waktu Enkripsi dan Dekripsi untuk Data Uji Citra *Grayscale* Format PNG

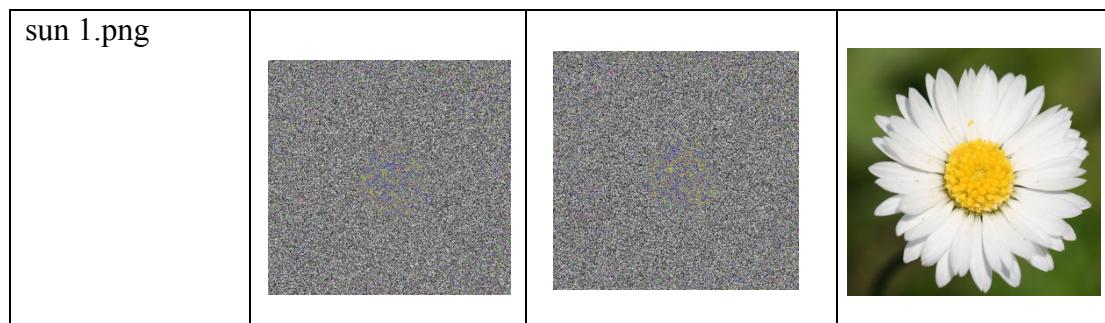
Data ke-	Nama File	Ukuran Citra (pixel)	Waktu Enkripsi (detik)	Rata-Rata Dekripsi (detik)
1	gs 1.png	480 x 480	0.18275	0.1736
2	gs 2.png	600 x 600	0.27087	0.26431
3	gs 3.png	720 x 720	0.37596	0.37861
4	gs 4.png	768 x 768	0.43902	0.441
5	gs 5.png	1024 x 1024	0.82234	0.78909

Tabel 4.5 Tabel Waktu Enkripsi dan Dekripsi untuk Data Uji Citra Berwarna (RGB) Format PNG

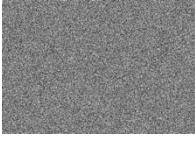
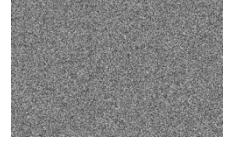
Data ke-	Nama File	Ukuran Citra (pixel)	Waktu Rata-Rata Enkripsi (detik)	Waktu Rata-Rata Dekripsi (detik)
1	sun 1.png	493 x 480	0.22658	0.20779
2	sun 2.png	617 x 600	0.3342	0.33001
3	sun 3.png	790 x 768	0.47707	0.48597
4	sun 4.png	1280x960	0.53891	0.56581
5	sun 5.png	1110 x 1080	1.0884	1.0681

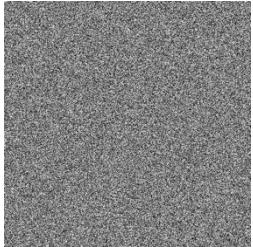
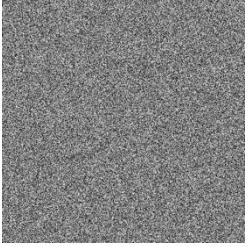
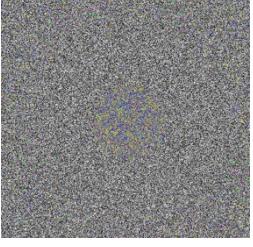
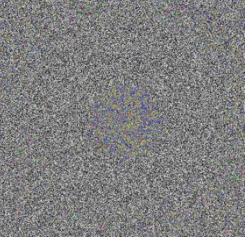
Tabel 4.6 Data Hasil Pengujian Sensitivitas Kunci Dengan Perubahan Pada Salah Satu Kunci

Nama File	Citra Hasil Enkripsi dengan Kunci $X_n = 0.1$ dan Kunci $r = 8.1$	Citra Hasil Dekripsi dengan Kunci $X_n = 0.1$ dan Kunci $r = 8.1 + 10^{-13}$	Citra Hasil Dekripsi dengan Kunci $X_n = 0.1$ dan Kunci $r = 8.1 + 10^{-14}$
abu 1.bmp			
matahari 1.bmp			
Nama File	Citra Hasil Enkripsi dengan Kunci $X_n = 0.1$ dan Kunci $r = 8.1 + 10^{-13}$	Citra Hasil Dekripsi dengan Kunci $X_n = 0.1$ dan Kunci $r = 8.1 + 10^{-14}$	Citra Hasil Dekripsi dengan Kunci $X_n = 0.1$ dan Kunci $r = 8.1 + 10^{-14}$
gs 1.png			

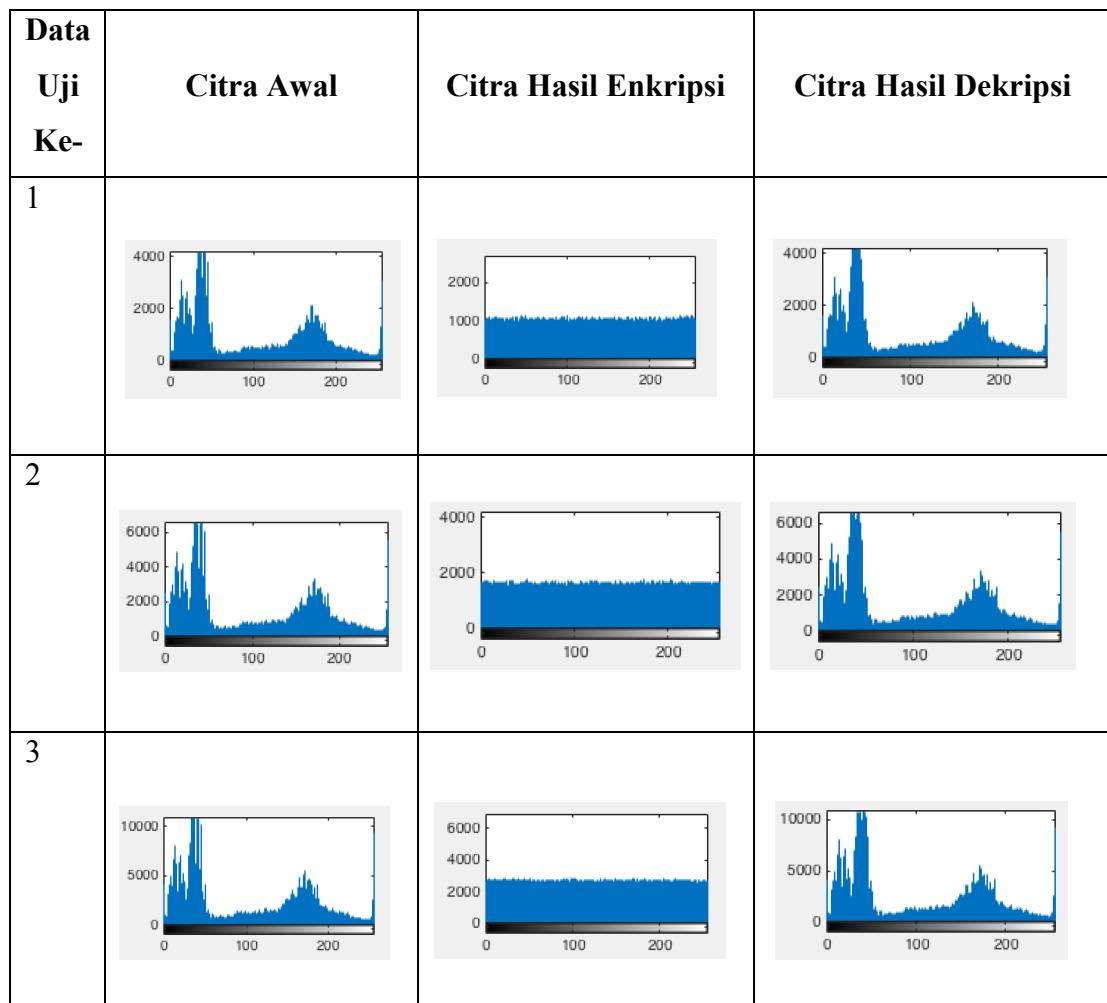


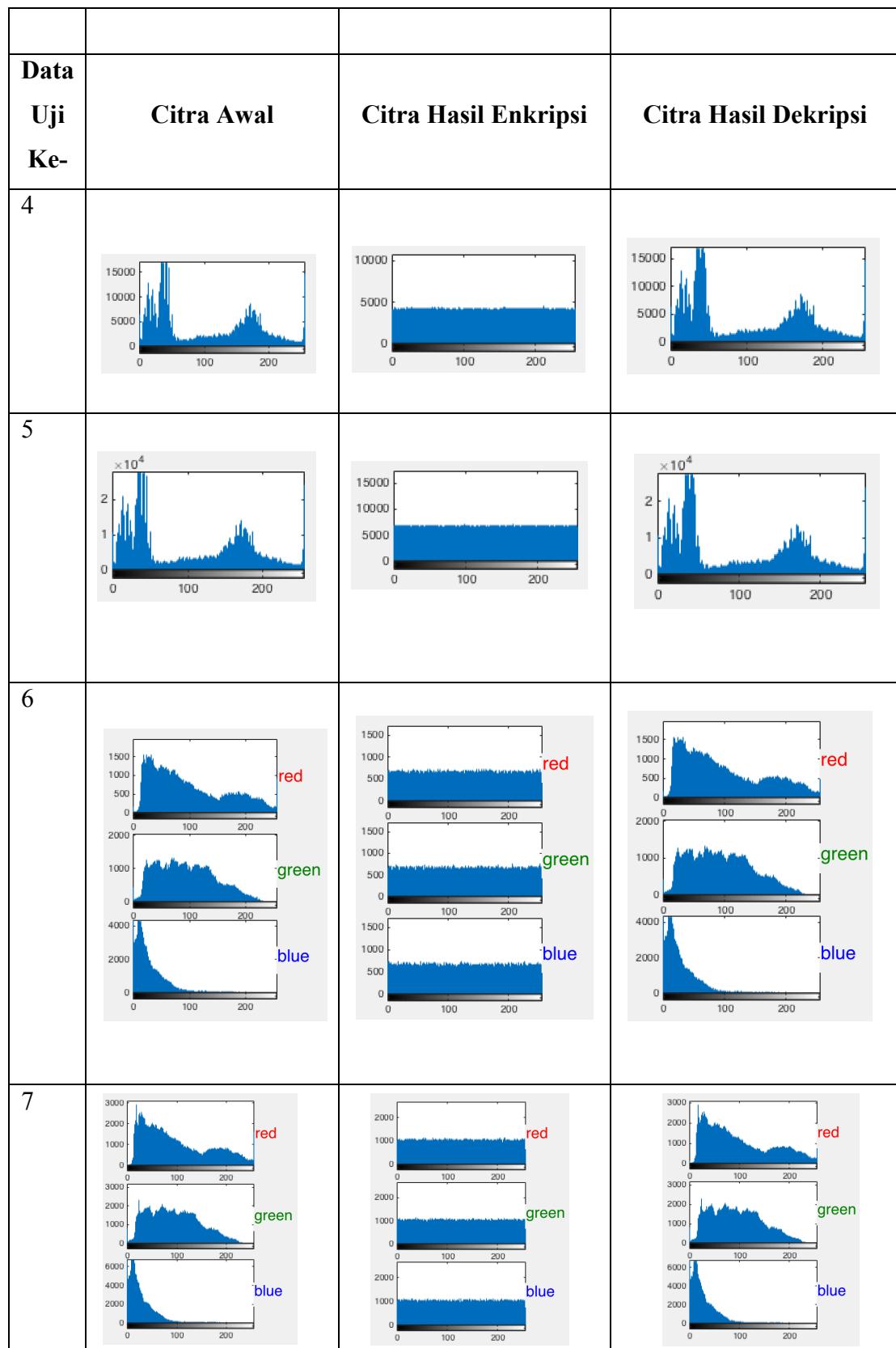
Tabel 4.7 Data Hasil Pengujian Sensitivitas Kunci Dengan Perubahan Pada Kedua Kunci

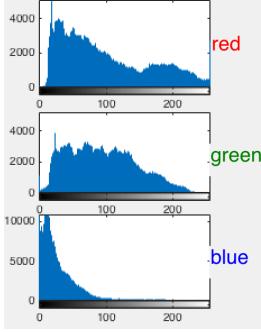
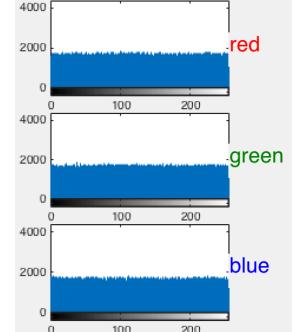
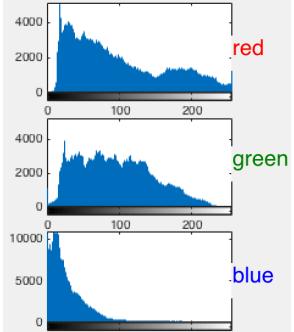
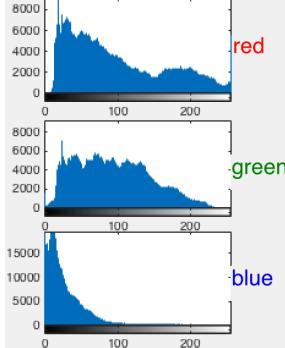
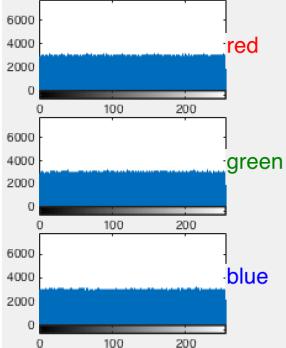
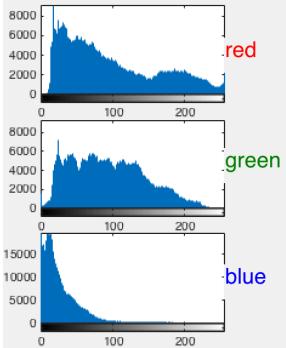
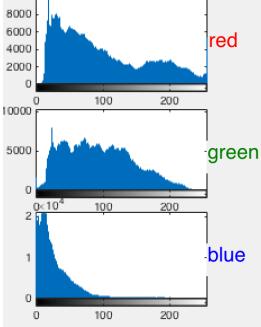
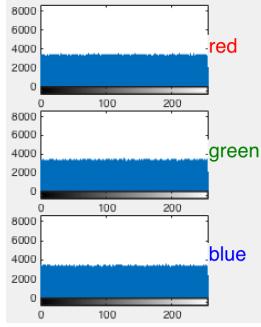
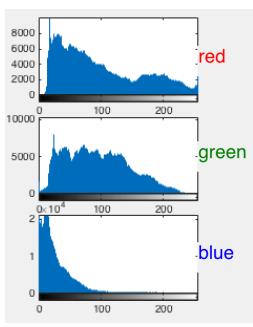
Nama File	Citra Hasil Enkripsi dengan Kunci $X_n = 0.1$ dan Kunci $r = 8.1$	Citra Hasil Dekripsi dengan Kunci $X_n = 0.1 + 10^{-14}$ dan Kunci $r = 8.1 + 10^{-13}$	Citra Hasil Dekripsi dengan Kunci $X_n = 0.1 + 10^{-15}$ dan Kunci $r = 8.1 + 10^{-14}$
abu 1.bmp			
Nama File	Citra Hasil Enkripsi dengan Kunci $X_n = 0.1$ dan Kunci $r = 8.1$	Citra Hasil Dekripsi dengan Kunci $X_n = 0.1 + 10^{-14}$ dan Kunci $r = 8.1 + 10^{-13}$	Citra Hasil Dekripsi dengan Kunci $X_n = 0.1 + 10^{-15}$ dan Kunci $r = 8.1 + 10^{-14}$
matahari 1.bmp			

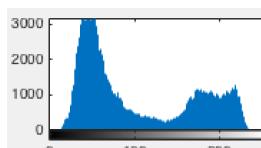
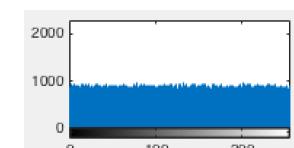
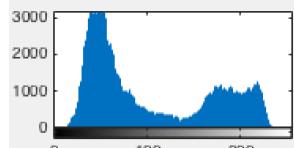
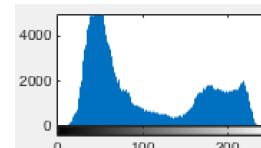
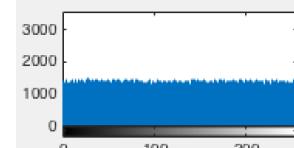
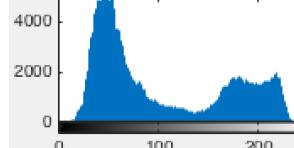
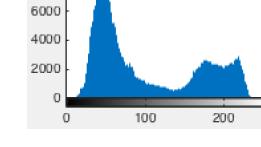
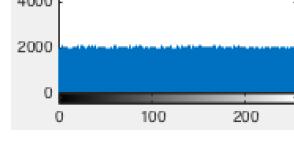
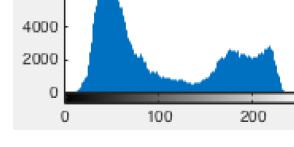
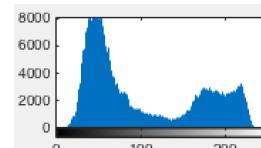
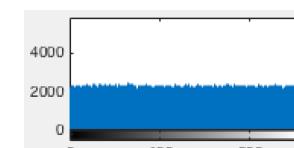
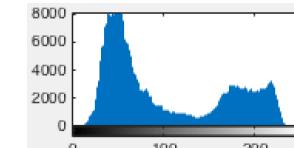
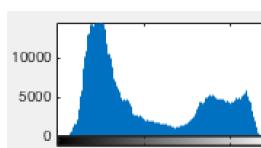
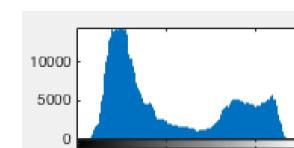
gs 1.png			
sun 1.png			

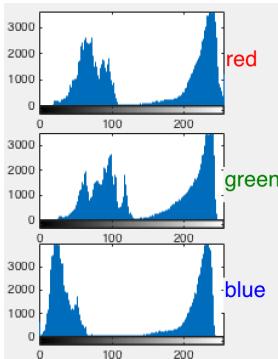
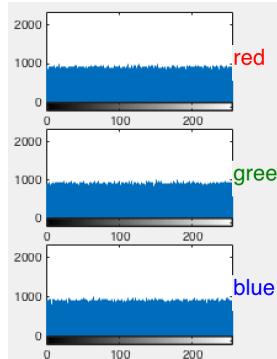
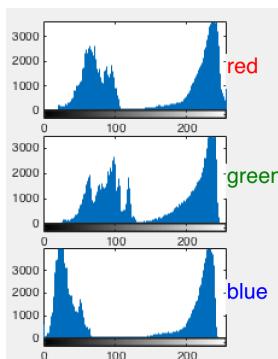
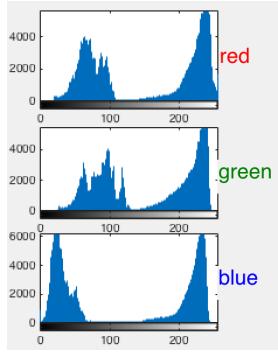
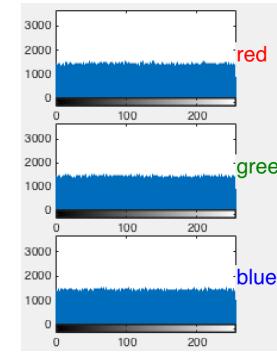
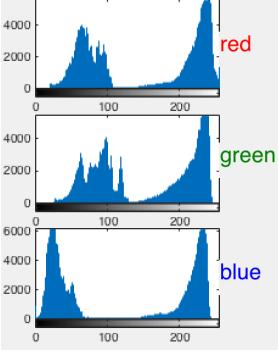
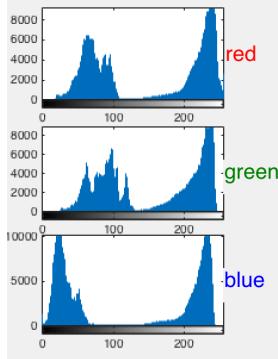
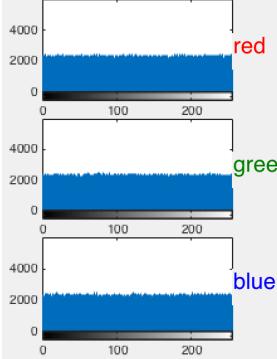
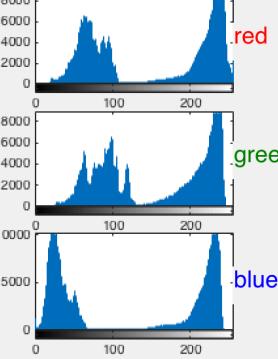
Tabel 4.8 Hasil Histogram Citra Uji Coba

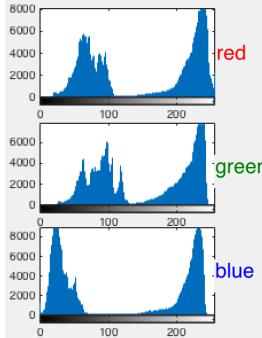
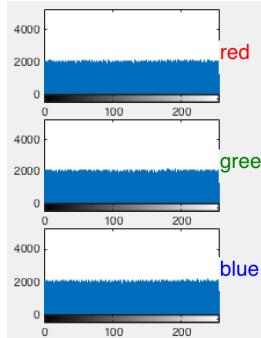
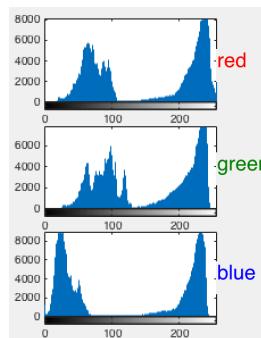




Data Uji Ke-	Citra Awal	Citra Hasil Enkripsi	Citra Hasil Dekripsi
8			
9			
10			

Data Uji Ke-	Citra Awal	Citra Hasil Enkripsi	Citra Hasil Dekripsi
11			
12			
13			
14			
15			

Data Uji Ke-	Citra Awal	Citra Hasil Enkripsi	Citra Hasil Dekripsi
16			
17			
18			

Data Uji Ke-	Citra Awal	Citra Hasil Enkripsi	Citra Hasil Dekripsi
19			
20	