ENKRIPSI CITRA DIGITAL MENGGUNAKAN ALGORITMA *MODIFIED RIVEST CODE* 6

SKRIPSI

Disusun oleh :

POMPY H

NIM. 0710960020

PROGRAM STUDI INFORMATIKA / ILMU KOMPUTER

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2013

ENKRIPSI CITRA DIGITAL MENGGUNAKAN ALGORITMA *MODIFIED RIVEST CODE* 6

SKRIPSI

Sebagai salah satu syarat untuk memperoleh
Gelar Sarjana dalam bidang Ilmu Komputer

Disusun oleh :
POMPY H
NIM. 0710960020

PROGRAM STUDI INFORMATIKA / ILMU KOMPUTER
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2013

ENKRIPSI CITRA DIGITAL MENGGUNAKAN ALGORITMA *MODIFIED RIVEST CODE* 6

SKRIPSI

Sebagai salah satu syarat untuk memperoleh
Gelar Sarjana dalam bidang Ilmu Komputer

Disusun oleh :
POMPY H
NIM. 0710960020

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I,	**Dosen Pembimbing II,**
Drs. Marji, MT	**Nurul Hidayat. S.Pd, M.Sc**
NIP. 196708011992031001	**NIP. 196804302002121001**

LEMBAR PENGESAHAN SKRIPSI

ENKRIPSI CITRA DIGITAL MENGGUNAKAN ALGORITMA *MODIFIED RIVEST CODE* 6

SKRIPSI

Sebagai salah satu syarat untuk memperoleh

Gelar Sarjana dalam bidang Ilmu Komputer

Disusun oleh:

POMPY H

NIM. 0710960020

Skripsi ini telah diuji dan dinyatakan lulus
pada tanggal 22 Juli 2013
dan dinyatakan memenuhi syarat untuk memperoleh
gelar Sarjana dalam bidang Ilmu Komputer

Penguji,

Lailil Muflikhah, S.Kom., M.Sc
NIP. 19741113 200501 2 001

Penguji,

Budi Darma Setiawan, S.Kom., M.Cs
NIK. 841015 06 1 1 0090

Penguji,

Issa Arwani, S.Kom., M.Sc
NIP. 830922 06 1 1 0074

Mengetahui
Ketua Program Studi Teknik Informatika

Drs. Marji, M.T.
NIP. 19670801 199203 1 001

PERNYATAAN ORISINALITAS SKRIPSI

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

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

Malang, 18 Juli 2013

Mahasiswa,

Pompy H

NIM. 0710960020

KATA PENGANTAR

Puji syukur ke hadirat Tuhan YME, hanya dengan rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan skripsi yang berjudul "Enkripsi Citra Digital Menggunakan Algoritma *Modified Rivest Code* 6".

Skripsi ini merupakan salah satu syarat untuk memenuhi persyaratan akademis untuk menyelesaikan studi di program Sarjana Ilmu Komputer Universitas Brawijaya.

Pada kesempatan ini penulis mengucapkan banyak terima kasih atas segala bantuan dan dedikasi moral maupun material dalam rangka penyusunan skripsi ini.

1. Drs. Marji, MT., selaku dosen pembimbing I dan Ketua Program Studi Ilmu Komputer di program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya yang telah membimbing dengan bijaksana dan sabar dalam membimbing dengan baik penyusunan skripsi ini.
2. Nurul Hidayat, S.Pd., MSc., selaku dosen pembimbing II yang telah membimbing dengan bijaksana dan sabar dalam membimbing dengan baik penyusunan skripsi ini.
3. Drs. Muh. Arif Rahman, M.Kom., selaku dosen pembimbing akademik atas nasehat, bimbingan, saran, dukungan yang diberikan selama penulis menuntut ilmu di Program Studi Ilmu Komputer.
4. Ir. Sutrisno, M.T., selaku Ketua Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
5. Segenap Bapak dan Ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada Penulis selama menempuh pendidikan di Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya.
6. Segenap staf dan karyawan di Program Teknik Informatika Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya yang telah banyak membantu Penulis dalam pelaksanaan penyusunan skripsi ini.
7. Ayah, Ibu, yang terkasih, terima kasih atas segala dukungan baik materi dan non materi serta doanya dan motivasinya.

8. Ns. Dee yang telah setia memberikan doa, semangat dan kasih dalam menyelesaikan skripsi ini.
9. Teman-teman Ilmu Komputer 2007 B, teristimewa John, Rahmi dan Riza atas segala bantuan, motivasi dan doanya.
10. Seluruh pihak yang tidak dapat disebut secara langsung yang telah memberikan bantuan demi terselesaikannya skripsi ini.

Penulis menyadari bahwa masih banyak kekurangan dalam penulisan laporan ini yang disebabkan oleh keterbatasan kemampuan dan pengalaman. Oleh karena itu, Penulis sangat menghargai saran dan kritik yang sifatnya membangun demi perbaikan penulisan dan mutu isi penelitian ini.

Penulis berharap semoga laporan ini dapat memberikan manfaat kepada pembaca dan bisa diambil manfaatnya untuk pengembangan selanjutnya.

Malang, 18 Juli 2013

Penulis

ABSTRAK

Pompy H. 2013. Enkripsi Citra Digital Menggunakan Algoritma *Modified Rivest Code* 6.

Dosen Pembimbing: Drs. Marji, MT dan Nurul Hidayat, S.Pd, M.Sc.

Enkripsi adalah salah satu cara membuat pesan yang awalnya dapat dibaca menjadi tidak dapat dibaca, pesan hasil dari enkripsi tersebut dikenal dengan istilah *cipher*. Dekripsi merupakan proses kebalikan dari enkripsi, yaitu membuat *cipher* menjadi data *plain* yang tak lain adalah pesan asli yang mengandung informasi tersembunyi. Terdapat banyak algoritma yang telah ada untuk melakukan enkripsi dan dekripsi. Salah satunya adalah algoritma MRC6.

Algoritma *MRC6* termasuk ke dalam kategori blok cipher, yaitu teknik dengan menggunakan pembagian blok-blok bit. Pembagian blok-blok bit ini dapat diterapkan juga terhadap citra. Setelah citra terbagi-bagi menjadi blok-blok bit, dapat dilakukan enkripsi maupun dekripsi citra yang memakai operasi-operasi primitif, diantaranya penjumlahan ”+”, pengurangan ”-”, *XOR* ”⊕” dan rotasi ”<<< atau >>>”, serta menggunakan operasi perkalian 32 bit yang telah diimplementasikan secara efisien dalam prosesor saat ini.

Enkripsi menggunakan algoritma *MRC6* menghasilkan perubahan terhadap nilai korelasi dan entropy dari citra semula. Penurunan nilai korelasi dan peningkatan nilai entropy pada hasil proses enkripsi menunjukkan kualitas *cipherimage* yang baik. Berdasarkan hasil analisis dan implementasi, didapatkan nilai korelasi dengan rata-rata penurunan 0,769357 dan nilai entropy dengan rata-rata peningkatan 0,523834 untuk 6 kali pengujian

Kata kunci : Enkripsi citra, dekripsi citra, *cipherimage*, algoritma MRC6, *Entropy*, Korelasi, *block cipher.*

ABSTRACT

Pompy H. 2013. Digital Image Encryption using *Modified Rivest Code* 6 Algorithm.
Advisors: Drs. Marji, MT dan Nurul Hidayat, S.Pd, M.Sc.

Encryption is one method to transform a readable message into unreadable, the result is known as *cipher*. Decryption is the opposite of the encryption. It works by transforming *cipher* into *plain* data. *Plain* data is the original message that contains hidden information. There are many existing algorithms for encryption and decryption. One of the them is the *MRC6* algorithm.

MRC6 algorithm is a block cipher, it is a technique using distribution of bit blocks. Distribution of bit blocks can be applied to images. After the image is divided into blocks of bit, it is able to do image encryption or decryption which is using primitive operations that are addition "+", subtraction "-", XOR " ⊕ "and rotation "<<<" or ">>>". The 32 bit multiplication operation has been implemented efficiently in processors nowadays.

Encryption using *MRC6* algorithm produces a change values of correlation and entropy of the original image. Decrease of correlation values and increase of the entropy values in the encryption process result showed a good quality *cipherimage*. Based on the result of analysis and implementation that has been done from encryption process, it yields an average correlation reduction of 0,7440061 and the entropy values with an average increase of 0,519829 at 6 times test procedures.

Keywords: image encryption, image decryption, *cipherimage*, MRC6 algorithm, entropy, correlation, blockcipher.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR *SOURCECODE*

BAB I
PENDAHULUAN

1.1. Latar Belakang Masalah

Seiring dengan pesatnya kemajuan teknologi, arus komunikasi dan informasi dapat dengan mudah dilakukan dan terjadi secara deras. Pertukaran data merupakan hal yang semakin lumrah. Data-data yang dipertukarkan seringkali berupa data digital yang berupa dokumen maupun berkas multimedia. Seluruh arus komunikasi data ini dapat terjadi antar organisasi seperti lembaga atau institusi pendidikan, pelayanan publik, pemerintahan, organisasi, maupun antar perusahaan.

Salah satu data yang sering ditukar adalah data multimedia berupa gambar. Data jenis ini paling mudah ditemui di dalam kehidupan sehari dan juga sering memegang peranan penting pada beberapa institusi ataupun organisasi yang memiliki kepentingan pengambilan, pengolahan dan penarikan kesimpulan dari suatu pengetahuan yang terekam dalam wujud gambar.

Masalah yang sering ditemui pada pertukaran informasi adalah keamanan. Pada data gambar, aspek keamanan menjadi penting dalam kondisi tertentu misalnya pada lingkungan pemerintahan di saat sebuah data pencitraan wilayah digunakan sebagai berkas pendukung strategis pertahanan keamanan negara yang bersifat rahasia. Contoh lain di kehidupan sehari-hari misalnya saat bertukar data gambar, yang mana koneksi yang digunakan rawan diinterupsi dan data yang ditukar ingin dijaga kerahasiaannya hingga sampai ke tujuan (penerima). Aspek keamanan tersebut dapat diwujudkan salah satunya dengan menggunakan teknik enkripsi.

Enkripsi merupakan sebuah proses yang melakukan pengubahan sebuah kode dari yang bisa dimengerti menjadi sebuah kode yang tidak bisa dimengerti. [WAH-03]. Enkripsi digunakan untuk menyandikan data-data atau informasi sehingga tidak dapat dibaca oleh orang yang tidak berhak.[RAH-99]. Enkripsi memanfaatkan sebuah *key* (kunci) untuk melakukan proses *encrypt* (penyandian) dan *decrypt* (pembukaan sandi). Pada tugas akhir ini digunakan algoritma enkripsi *Modified Rivest Code 6* (MRC6).

Algoritma *MRC6* merupakan hasil pengembangan dari *RC6*. Pengembangan ini dilakukan oleh Nawal El-Fishawy, Talat El-Danaf dan Osama Abu Said. Dasar algoritma ini adalah penggunaan register dan *word* dengan ukuran empat kali lebih besar daripada algoritma *RC6*. Algoritma RC6 sendiri adalah salah satu algoritma AES yang mampu menawarkan tingkat keamanan relatif tinggi dibandingkan ragam algoritma AES lain. Keuntungan yang didapatkan dari peningkatan jumlah register dan *word* tersebut adalah jumlah rotasi yang mungkin dilakukan menjadi lebih sedikit dibanding algoritma *RC6*, namun dengan tingkat keamanan yang lebih besar.[ELF-05].

Dengan kelebihan ini, maka penulis menggunakan algoritma *MRC6* pada tugas akhir sebagai dasar untuk melakukan enkripsi dan dekripsi pesan berupa citra digital.

1.2. Perumusan Masalah

Perumusan masalah dalam Tugas Akhir ini adalah:

1. Bagaimana menerapkan teknik kriptografi dengan algoritma MRC6 untuk pengamanan citra.
2. Berapa perubahan nilai korelasi sebelum dan sesudah terenkripsi dari segi *pixel* dan *entropy* nya.
3. Bagaimana ketahanan *cipherimage* terhadap manipulasi citra.

1.3. Pembatasan Masalah

Dalam perancangan sistem ini, perlu dibatasi ruang lingkupnya agar pemecahan masalah dapat tepat sasaran dan tidak melenceng dari tujuan awal. Hal-hal yang dibatasi adalah sebagai berikut :

1. Enkripsi yang dilakukan terbatas pada penanganan file secara perangkat lunak (software). Enkripsi dari segi perangkat keras (hardware) tidak termasuk dalam pembahasan.
2. Jenis file yang akan melalui proses enkripsi-dekripsi hanya file berjenis citra (image) dan tidak dilakukan terhadap file selainnya.
3. File citra yang digunakan adalah berformat *bitmap* (.bmp)

4. Kualitas kelayakan didasarkan pada perubahan nilai korelasi dan *entropy* hasil enkripsi citra.

1.4. Tujuan

Berdasarkan masalah yang telah diidentifikasikan, maka tujuan yang hendak dicapai adalah:

1. Menghasilkan perangkat lunak untuk pengamanan data citra dengan menggunakan teknik kriptografi algoritma MRC6.
2. Menghitung perubahan nilai korelasi sebelum dan sesudah terenkripsi dari segi *pixel* dan *entropy* nya sebagai parameter kelayakan algoritma kriptografi MRC6.
3. Menguji ketahanan *cipherimage* terhadap manipulasi citra.

1.5. Manfaat

Manfaat penelitian ini adalah membantu menyediakan metode pengamanan data berupa citra digital kepada pihak-pihak yang memerlukan dengan cara melakukan enkripsi dan dekripsi, sehingga diharapkan mampu menyimpan informasi yang ada pada citra tersebut tanpa takut informasi yang ada menjadi bocor.

1.6. Metodologi Penelitian

Metodologi penelitian yang digunakan dalam menyusun skripsi ini adalah sebagai berikut :

1. Studi Literatur

   Dengan mempelajari teori dari berbagai sumber literatur berupa buku, artikel, dan jurnal yang berhubungan dengan konsep Kriptografi dan algoritma *RC6*.

2. Analisis dan Perancangan Sistem

   Berdasarkan literatur yang diperoleh, melakukan analisis dan membuat rancangan dalam bentuk perangkat lunak.

3. Implementasi dan Pengujian Sistem

   Melakukan pengujian terhadap hasil dari perangkat lunak yang merupakan implementasi dari rancangan sistem.

4. Pembuatan Laporan

Pembuatan dokumentasi sistem dan analisis berdasarkan sistematika penulisan skripsi.

1.7. Sistematika Penulisan

BAB I PENDAHULUAN

Mencakup latar belakang permasalahan perancangan sistem, perumusan masalah, pembatasan masalah, tujuan dan sistematika penyusunan Tugas Akhir.

BAB II KAJIAN PUSTAKA DAN DASAR TEORI

Mencakup landasan teori yang digunakan dalam penyusunan tugas akhir.

BAB III METODE PENELITIAN/ PERANCANGAN

Mencakup metode-metode yang diaplikasikan dalam perancangan sistem termasuk pembuatan diagram alir (flowchart)

BAB IV IMPLEMENTASI

Merupakan lingkungan perangkat keras dan lunak yang akan digunakan untuk membangun aplikasi sebagai implementasi teori dan perancangan.

BAB V PENGUJIAN DAN ANALISIS

Merupakan hasil dari sistem yang telah dirancang. Berisi pengujian terhadap implementasi yang telah dikerjakan dan analisis terhadap hasil pengujian.

BAB VI PENUTUP

Berisi kesimpulan dari Tugas Akhir, mencakup sistem yang telah dirancang dan saran yang mungkin dapat digunakan untuk penelitian dan pengembangan sistem di masa depan agar sistem dapat teraplikasikan secara lebih sempurna.

BAB II
KAJIAN PUSTAKA DAN DASAR TEORI

Bab dua berisi landasan teori yang digunakan. Pada bab ini akan dibahas dasar-dasar teori yang digunakan dalam pembuatan tugas akhir, meliputi konsep dasar perancangan perangkat lunak, konsep kemanan data, definisi dan konsep kriptografi dan enkripsi, *block cipher* dan algoritma MRC6.

2.1. Rekayasa Perangkat Lunak

Rekayasa perangkat lunak merupakan aplikasi praktis dari pengetahuan ilmiah dalam desain dan pembuatan program komputer beserta dokumentasi yang dibutuhkan untuk mengembangkan, mengoperasikan dan memelihara program komputer tersebut [BOE-1997]. Rekayasa perangkat lunak menyediakan fasilitas yang bertujuan menghasilkan perangkat lunak yang memiliki kualitas tinggi. Kualitas yang dijadikan tolak ukur terbagi menjadi 2, yaitu kualitas desain dan kualitas konformasi.

Kualitas desain mengacu pada karakteristik yang ditentukan oleh perancang terhadap suatu item tertentu. Nilai material, toleransi dan spesifikasi kinerja. Semua hal tersebut memberikan kontribusi terhadap kualitas desain. Kualitas desain berbanding lurus terhadap spesifikasi, semakin tinggi spesifikasi yang terpenuhi dalam suatu produk, maka kualitas desain dari suatu produk akan bertambah pula. Dalam pemenuhan spesifikasi yang diberikan, hal yang menentukan adalah nilai dari material yang digunakan, toleransi yang lebih ketat serta kinerja yang lebih baik.

Kualitas konformasi merupakan tingkat di mana spesifikasi desain terus diikuti selama pembuatan. Semakin tinggi tingkat konformasi, semakin tinggi pula tingkat kualitas konformasi.

Dalam pengembangan perangkat lunak, kualitas desain mencakup syarat, spesifikasi dan desain sistem. Kualitas konformasi adalah suatu masalah yang difokuskan pada implementasi. Bila implementasi mengikuti desain dan sistem yang dihasilkan memenuhi persyaratan serta tujuan kinerja, maka kualitas

konformasi menjadi tinggi. Aktivitas dalam pengembangan perangkat lunak mencakup [ROB-1999]:

4. Requirement analysis and specification

   Mengetahui dan mengerti apa yang dilakukan oleh sistem saat itu (pokok permasalahan) dan mengekspresikannya dalam bentuk yang jelas dan detail.
5. System and software design

   Menentukan solusi yang tepat untuk pokok permasalahan dan mengekspresikannya dalam bentuk yang mudah untuk diterjemahkan ke dalam suatu bentuk program.
6. Implementation/ coding
7. Testing

   Melakukan uji coba perangkat lunak agar sesuai dengan kebutuhan yang diinginkan dan bebas dari kesalahan (error/bug).
8. Operation and maintenance

   mengoperasikan dan memelihara perangkat lunak yang telah selesai dibuat.

2.2. Keamanan Data

Keamanan dan kerahasiaan dari suatu data merupakan hal yang harus dipenuhi jika suatu data tidak ingin sembarangan diakses pihak lain. Suatu informasi menjadi kehilangan arti pentingnya atau kerahasiaannya dari sisi pengirim apabila sampai jatuh ke pihak lain yang tidak diinginkan untuk menerima informasi tersebut. Untuk menjaga agar pihak yang tidak diinginkan tak memiliki akses diperlukan pengamanan data. Selain untuk meningkatkan keamanan data, pengamanan data juga berfungsi untuk:

1. Melindungi data agar tidak terbaca oleh pihak-pihak yang tidak berhak.
2. Mencegah pihak tersebut menyisipkan atau menghapus data.

Aspek yang berkaitan dengan ancaman keamanan data antara lain:

1. *Interruption*.

   Interruption merupakan usaha perusakan terhadap ketersediaan data. Contoh adalah harddisk yang dirusak, memotong saluran komunikasi, menghapus data secara sengaja, dan lain-lain (Gambar 2.1).

Gambar 2.1 Perusakan ketersediaan data

2. *Interception.*

   Interception merupakan tindakan di mana pihak yang tidak berhak, berhasil mendapatkan akses informasi dari dalam sistem. Misalnya dengan memotong arus komunikasi data pada jaringan public WiFi (Gambar 2.2).

Gambar 2.2 Pencurian data

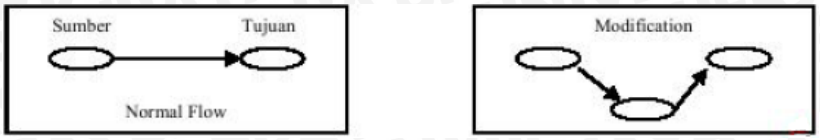
3. *Modification*.

   Modification merupakan pengubahan pada data. Pihak yang tidak berkepentingan dapat melakukan tindakan masuk ke dalam sistem tanpa izin lalu melakukan usaha-usaha untuk memodifikasi data yang ada dengan tujuan merusak (gambar 2.3).

Gambar 2.3 Modification

4. *Fabrication.*

   Fabrication merupakan tindakan meniru, menambahkan record palsu atau memalsukan suatu file dengan tujuan merusak atau mengacaukan keberadaan file asal (gambar 2.4). [WAH-03]



Gambar 2.4 Fabrication

Bentuk-bentuk kegiatan yang pernah dilakukan dalam menunjang keamanan data adalah:

1. Mengatur *access control*

   Kontrol terhadap akses suatu data adalah salah satu hal yang mungkin dilakukan untuk mencegah akses suatu data terhadap pihak-pihak yang tak berkepentingan. Cara yang mungkin dilakukan adalah dengan mengatur user level pada sistem operasi atau dengan menambahkan *password*.

2. Memasang Pelindung

   Tindakan pencegahan dari seseorang yang tidak memiliki hak akses dan kewenangan terhadap suatu data dengan cara memblokir tepat di pintu masuk menuju akses data. Prosedurnya dilakukan pada tingkat software dengan setting tertentu.

3. Sistem Monitoring

   Sistem monitoring digunakan untuk mengetahui adanya suatu serangan. Sistem monitoring akan langsung mendeteksi dan melaporkan sebuah ancaman terhadap integritas data yang datang melalui serangan kepada administrator.

4. Backup Data

   Backup terhadap data bertujuan agar data yang tersimpan tidak hilang bergitu saja jika sewaktu-waktu terjadi hal yang tidak diinginkan.

Secara umum, sebuah atau sekelompok data dapat dikatakan aman apabila telah memenuhi aspek kerahasiaan, integritas serta jaminan bahwa data yang tersimpan maupun hendak dikirim merupakan data yang asli. [RAH-99]

2.3. Kriptografi

2.3.1. Konsep Dasar Kriptografi

Kriptografi merupakan cabang ilmu pengetahuan dalam bidang komputasi yang berhubungan dengan dengan teknik mengamankan informasi. Kriptografi berasal dari bahasa Yunani, "*crypto"* yang berarti rahasia, dan "*grafos"* yang berarti menulis [RAH-1999]. Teknik yang digunakan pada kriptografi pada dassarnya menggunakan pengetahuan matematis. Kriptografi

bukanlah satu-satunya metode dalam menjaga keamanan, namun kriptogtafi menyediakan sekumpulan teknik untuk menjaga keamanan dokumen.

Secara umum, kriptografi dibagi ke dalam dua jenis, yaitu kriptografi klasik dan kriptografi modern. Pada kriptografi klasik, algoritmanya bekerja pada mode karakter. Sedangkan pada algoritma modern algoritmanya bekerja pada mode bit. Teknik kriptografi modern dibagi menjadi dua jenis, yaitu algoritma kriptografi kunci simetris dan algoritma kriptografi kunci publik.

2.3.2. Algoritma Kunci Simetris

Pada algoritma kunci simetris, kunci yang digunakan untuk proses enkripsi dan dekripsi merupakan satu kunci saja. Satu kunci tersebut dapat pula diturunkan dari kunci yang lainnya. Kelebihan algoritma kriptografi simetris adalah:

1. Algoritma ini dirancang sehingga proses enkripsi/dekripsi membutuhkan waktu yang singkat.
2. Ukuran kunci relatif lebih pendek.
3. Algoritmanya bisa menghasilkan *cipher* yang lebih kuat.
4. Autentikasi pengiriman pesan langsung diketahui dari *ciphertext* yang diterima, karena kunci hanya diketahui oleh pengirim dan penerima pesan saja.

Kelemahan algoritma kriptografi simetris adalah:

1. Kunci harus dikirim melalui saluran yang aman. Kedua entitas yang berkomunikasi harus menjaga kerahasiaan kunci ini.
2. Kunci harus sering diubah, mungkin pada setiap sesi komunikasi [MUN-04].

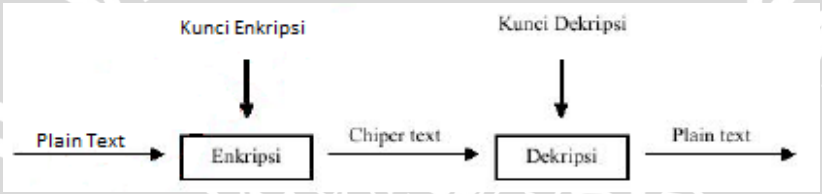
2.3.3. Algoritma Kunci Asimetris

Proses enkripsi dan dekripsi pada algoritma kunci publik menggunakan dua kunci yang berbeda. Satu kunci merupakan sebuah kunci publik (*public key*) yang dapat dipublikasikan. Kunci yang lain disebut kunci privat (*private key*) yang merupakan kunci yang rahasia. Saat pengirim pesan hendak mengirimkan pesannya, pesan dapat disandikan dengan menggunakan kunci publik penerima pesan. Saat penerima pesan hendak membaca pesan,

penerima harus mendekripsikan pesan tersebut dengan menggunakan kunci privatnya sendiri.

2.4. Enkripsi

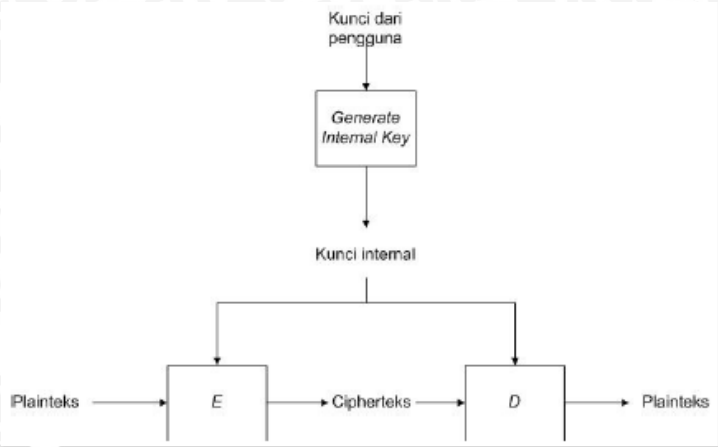
Enkripsi merupakan suatu teknik untuk menyembunyikan suatu informasi dari pihak yang tak berkepentingan melalui proses penyandian (*encryption*). Proses ini memerlukan bantuan kunci (*key*) untuk melakukan baik proses penyandian maupun pembukaan kembali informasi yang telah disandikan sebelumnya (*information retrieval & decryption*). Proses penyandian informasi hingga membuka sandi untuk membaca informasi dapat dijabarkan pada gambar 2.5.



Gambar 2.5 Proses Enkripsi-Dekripsi

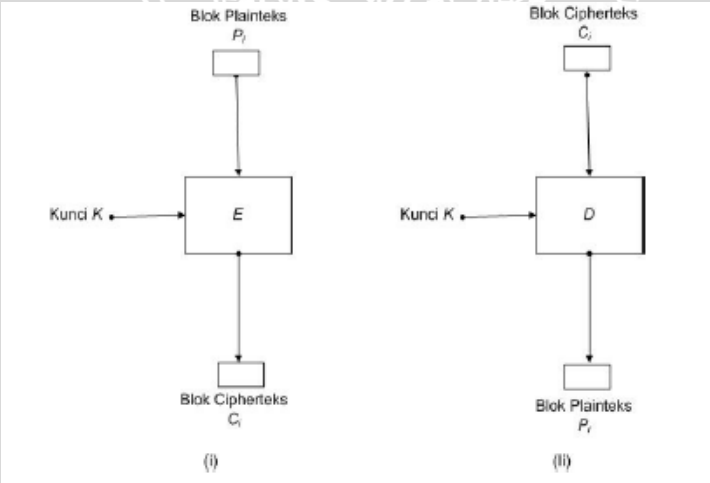
2.5. Block Cipher

Block cipher merupakan algoritma kriptografi kunci simetris yang mengubah plaintext yang dibagi dalam blok sama panjang, menjadi ciphertext dengan panjang sama pula. Ukuran panjang blok dapat bervariasi tergantung kepada algoritma yang dipergunakan. Secara umum cara kerja block cipher dapat digambarkan pada gambar 2.6. Kunci internal yang terbentuk digunakan untuk mengenkripsi data ke dalam *cipher* sekaligus mendekripsikan kembali ke dalam mode *plain.*



Gambar 2.6. Proses Kerja Block Cipher

Block cipher memerlukan teknik dalam pengoperasiannya. Mode pengoperasian yang sederhana dan paling sering digunakan adalah mode *Electronic Code Book* (ECB). Pada mode ini, setiap blok pada plaintext dienkripsi satu persatu secara terpisah. Hasil enkripsi pada masing-masing blok tidak mempengaruhi blok yang lain. Untuk proses dekripsinya juga terjadi secara terpisah. Proses enkripsis dan dekripsi dari mode ECB digambarkan pada gambar 2.7.



Gambar 2.7. (i) Proses Enkripsi ECB dan (ii) proses Dekripsi ECB

Dalam konstruksi *block cipher*, perlu diperhatikan beberapa ketentuan agar hasil yang diperoleh baik, kerahasiaan terjaga serta tidak terjadi error. Di antaranya:

1. Aturan *Confusion* dan *Diffusion* Shannon

   Confusion berarti menyamarkan pola yang mungkin ditangkap oleh pembuka pesan terhadap plaintext, ciphertext dan key. Modifikasi

terhadap satu bit plaintext berpengaruh terhadap ciphertext yang pasti akan ikut berubah, prinsip ini merupakan dasar dari *diffusion* di mana aturan dasarnya menyebarkan pengaruh pengubahan satu saja bit pada plaintext atau key ke sebanyak mungkin ciphertext sehingga akan menambah bobot kesulitan dekripsi pihak yang tidak berkepentingan [RAH-1999].

2. *Iterated Cipher*

   Iterated cipher merupakan penambahan iterasi pada pemrosesan tiap-tiap blok, untuk setiap iterasi dilakukan rotasi. Pada setiap hasil rotasi dilakukan fungsi transformasi yang sama dengan key yang berbeda. Key khusus yang berbeda ini disebut key internal. Key internal merupakan key awal yang dimasukkan oleh pengguna namun telah dikomputasi menggunakan fungsi tertentu.

3. *Weak key*

   Weak key merupakan hal umum yang harus dihindari. *Weak key* adalah kunci yang berpotensi menghasilkan ciphertext yang masih memiliki derajat kemiripan tinggi dengan plaintext.

2.6. Algoritma MRC6

2.6.1. Deskripsi

Algoritma MRC6 merupakan algoritma *block cipher*. Algoritma ini merupakan pengembangan dari algoritma RC6 yang telah ada sejak tahun 1998 dan dikembangkan oleh tim dari RSA laboratories. Algoritma RC6 awalnya dirancang untuk menjadi AES (*Advance Encryption Standard*) namun tidak terpilih [DAE-1999].

Algoritma MRC6 merupakan algoritma dengan parameter penuh. Algoritma ini dispesifikasikan dengan menggunakan notasi *w/r/b* di mana notasi *w* adalah ukuran *word* dalam bit; notasi *r* adalah jumlah putaran non-negatif; notasi *b* adalah panjang key enkripsi/dekripsi dalam byte. Pada dasarnya menggunakan parameter-parameter yang sama dengan RC6 [ELF-2005]. Algoritma MRC6 memiliki jumlah register sebanyak 16 buah, 4 kali lebih besar dibanding RC6 dan masing-masing register memiliki ukuran 32 bit. Cara kerja algoritma MRC6 adalah menggunakan prinsip *Iterated Block*

Cipher yang menggunakan iterasi, untuk pembentukan kunci internal dan prosedur enkripsi-dekripsi pada dasarnya mirip dengan prosedur algoritma RC6, hanya salah satu yang membedakan ukuran word pembentuk *keytable* (S) yang digunakan pada MRC6 adalah t=8r+16, di mana *t* adalah ukuran dari tabel (S). Menyediakan keamanan yang lebih tinggi daripada RC6.

2.6.2. Pembentukan Kunci Internal Pada MRC6

Tujuan dari proses pembangunan kunci internal adalah membangun suatu tabel *expanded key* dengan array S[0...,t – 1], yang terdiri dari ukuran *key* t=8r+16 w=bit word. Array tersebut akan digunakan dalam proses enkripsi dan dekripsi.[ELF-05].

Dalam pembuatan kunci internal diperlukan dua buah konstanta yang disebut *magic constant*. *Magic constant* tersebut dinotasikan sebagai P_w dan Q_w [RAG-2001]. Pembentukan *magic constant* untuk masing-masing P_w dan Q_w dapat dijabarkan pada persamaan 2.1. dan 2.2.

$$P_w = \text{Odd}((e-2)2^w) \qquad (2.1)$$

$$Q_w = \text{Odd}((\emptyset-1)2^w) \qquad (2.2)$$

Di mana:

e = 2.7182818284859 (basis logaritma alami)

$\emptyset$ = 1.618022988749 (golden ratio)

Nilai P_w dan Q_w dalam heksadesimal untuk nilai w =16,32 dan 64 adalah:

P_{16} = *b*7*e*1

Q_{16} = 9*e*37

P_{32} = *b*7*e*15163

Q_{32} = 9*e*3779*b*9

P_{64} = *b*7*e*151628*aed*2*a*6*b*

Q_{64} = 9*e*3779*b*97*f*4*a*7*c*15

Odd(x) merupakan integer ganjil paling dekat terhadap x, apabila x adalah bilangan genap maka diambil integer yang ganjil setelah x.

Untuk pembangunan key, terdiri dari tiga tahapan:

1. Konversi *secret key* dari bytes ke words

Sebelum dilakukan konversi, *secret key* K[0..b-1] perlu disalin dalam sebuah array L[0..c-1], untuk c= pembulatan ke atas(*b*/u) dan u = *w*/8, penyalinan dilakukan dengan *little endian*. Untuk semua posisi byte pada L yang kosong diberi nilai nol. Untuk kasus di mana *b* = 0 , maka *c*= 1 dan L[0]=0.

2. Inisialisasi *array* S

   Langkah kedua adalah melakukan inisialisasi array S agar array tersebut menjadi berpola *pseudo*-random bit tertentu. Caranya ialah menggunakan aritmatika modulo 2^w yang ditentukan dengan P_w dan Q_w.

3. Mencampurkan L dan S

   Tahapan yang terakhir adalah mencampurkan kunci rahasia dari pengguna yang sudah tersimpan dalam L dan S sebanyak 3 kali iterasi.

Kunci yang dihasilkan oleh proses pembentukan kunci ini memiliki sifat satu arah [RIV-1998], sehingga pembentukan kunci ini dapat digunakan sebagai fungsi *hash* satu arah. Dengan sifat satu arah tersebut, maka kunci internal akan sangat berbeda dengan kunci yang dimasukkan oleh pengguna sehinnga hasilnya adalah ciphertext yang tidak mudah dipecahkan.

2.6.3. Dasar Enkripsi dan Dekripsi

Algoritma MRC6 memiliki dasar enkripsi dan dekripsi yang sama dengan algoritma RC6. Pada algoritma RC6, digunakan empat register yang masing-masing berukuran *w*-bit. Register-register tersebut berisi plaintext yang selanjutnya akan dienkripsi. *Byte* pertama dari plaintext atau ciphertext disimpan pada MSB (*most significant byte*) dari D. Proses enkripsi dan dekripsi algoritma RC6 menggunakan enam buah operasi dasar, yaitu.

a + b = penjumlahan integer modulo 2^w

a – b = pengurangan integer modulo 2^w

a $\oplus$ b= operasi bitwise exclusive-or sebesar *w*-bit *words*

a * b = perkalian integer modulo 2^w

a<<<b= rotasi sejumlah *w*-bit *word* ke kiri sebanyak jumlah yang diberikan oleh *least significant* logaritma basis 2 *w* bit dari b

a>>>b = rotasi sejumlah *w*-bit word ke kanan sebanyak jumlah yang diberikan oleh *least significant* logaritma basis 2 *w* bit dari b

Enkripsi dengan *RC6-w/r/b*

Input :- *Plainteks* disimpan dalam empat w-bit *register* A,B,C,D
- Pengulangan sejumlah r putaran
- Kunci W-bit putaran S[0..2r+3]

Output : *Cipherteks* dalam A,B,C,D

Procedure :
```
B = B + S[0]
D = D + S[1]
for i = 1 to r do
{
t = (B × (2B + 1))<<<lg w
u = (D × (2D + 1))<<<lg w
A = ((A + t)<<<u) ⊕S[2i]
C = ((C + u)<<<t) ⊕ S[2i+ 1]
(A,B,C,D) = (B,C,D,A)
}
A = A + S[2r + 2]
C = C + S[2r + 3]
```

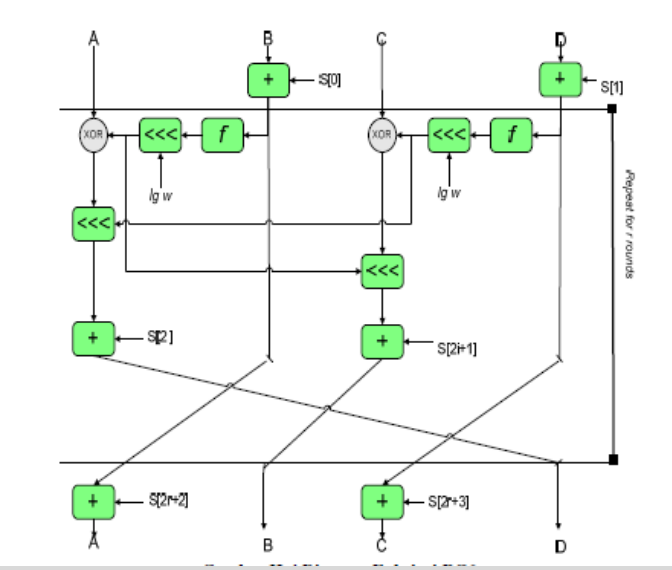
Algoritma MRC6 memiliki perbedaan yaitu memiliki jumlah *w*-bit register empat kali lebih banyak yang disimbolkan dengan register 1 hingga 16 (R1-R16). Prosedurnya adalah

```
R2 = R2 + S[0],      R4 = R4 + S[1]
R6 = R6 + S[2],      R8 = R8 + S[3]
R10 = R10 + S[4],    R12 = R12 + S[5]
R14 = R14 + S[6],    R16 = R16 + S[7]

for i = 1 to r do
{
k = (R2 × (2R2 + 1))<<<lg w
l = (R4 × (2R4 + 1))<<<lg w
m = (R6 × (2R6 + 1))<<<lg w
n = (R8 × (2R8 + 1))<<<lg w
t = (R10 × (2R10 + 1))<<<lg w
u = (R12 × (2R12 + 1))<<<lg w
v = (R14 × (2R14 + 1))<<<lg w
z = (R16 × (2R16 + 1))<<<lg w

R1 = ((R1 + k)<<<l) ⊕S[8i]
R3 = ((R3 + l)<<<k) ⊕ S[8i+1]
```

```
R5 = ((R5 + m)<<<n) ⊕S[8i+2]
R7 = ((R7 + n)<<<m) ⊕S[8i+3]
R9 = ((R9 + t)<<<u) ⊕S[8i+4]
R11 = ((R11 + u)<<<t) ⊕S[8i+5]
R13 = ((R13 + v)<<<z) ⊕S[8i+6]
R15 = ((R15 + z)<<<v) ⊕S[8i+7]

(R1,R2,...,R15,R16) = (R2,R3,...,R16,R1)
}
R1 = R1 + S[8r + 8],     R3 = R3 + S[8r + 9]
R5 = R5 + S[8r + 10],    R7 = R7 + S[8r + 11]
R9 = R9 + S[8r + 12],    R11 = R11 + S[8r + 13]
R13 = R13 + S[8r + 14], R15 = R15 + S[8r + 15] [ELF-05]
```

Proses enkripsi dapat dilihat pada gambar 2.8. Persamaan yang digunakan dalam proses enkripsi maupun dekripsi dapat dilihat pada persamaan 2.3.

$$f(x)=x * (2x+1) \qquad (2.3)\ [\text{BUU-06}]$$



Gambar 2.8. diagram enkripsi dengan *f(x)=x * (2x+1)*

Algoritma *RC6* menggunakan sub kunci yang dibangkitkan dari kunci dan dinamakan dengan S[0] hingga S[43]. Masing-masing sub kunci panjangnya 32 bit. Proses enkripsi pada algoritma *RC6* dimulai dan diakhiri dengan proses *whitening* yang bertujuan untuk menyamarkan iterasi yang pertama dan yang terakhir dari proses enkripsi dan dekripsi. Pada proses *whitening* awal, nilai B

akan dijumlahkan dengan S[0] dan nilai D dijumlahkan dengan S[i]. Pada masing-masing iterasi pada *RC6* menggunakan dua (2) buah sub kunci. Sub kunci pada iterasi yang pertama menggunakan sub-sub kunci lanjutannya. Setelah iterasi ke-20 selesai, dilakukan proses *whitening* akhir dimana nilai A dijumlahkan dengan S[42], dan nilai C dijumlahkan dengan S[43].

Setiap iterasi pada algoritma *RC6* mengikuti aturan sebagai berikut, nilai B dimasukkan ke dalam fungsi f, yang didefinisikan sebagai f(x) = x(2x + 1), kemudian diputar ke kiri sejauh 5 bit. Hasil yang didapat pada proses ini dimisalkan sebagai u. Nilai u kemudian di XOR dengan C dan hasilnya menjadi nilai C.

Nilai t juga digunakan sebagai acuan bagi C untuk memutar nilainya ke kiri. Begitu pula dengan nilai U, juga digunakan sebagai acuan bagi nilai A untuk melakukan proses pemutaran ke kiri.

Kemudian sub kunci S[2i] pada iterasi dijumlahkan dengan A, dan sub kunci S[2i+1] dijumlahkan dengan C. Keempat bagian dari blok kemudian akan dipertukarkan dengan mengikuti aturan bahwa nilai A ditempatkan pada D, nilai B ditempatkan pada A, nilai C ditempatkan pada B, dan nilai (asli) D ditempatkan pada C. Demikian iterasi tersebut terus berlangsung hingga dua puluh (20) kali [RUD-07].

Dekripsi dengan *RC6*-w/r/b

Input :- *Cipherteks* disimpan dalam empat w-bit *register* A,B,C,D
- Pengulangan sejumlah r putaran
- Kunci W-bit putaran S[0..2r+3]

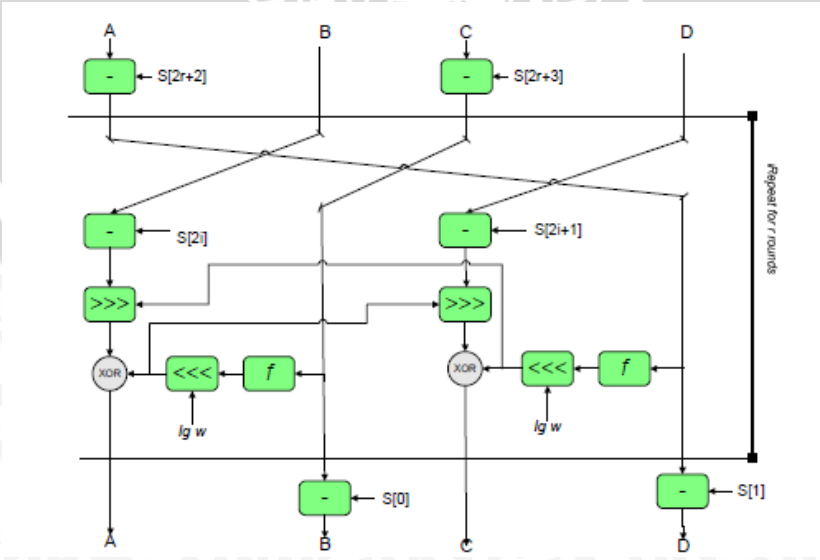
Output : *Plainteks* dalam A,B,C,D

Procedure :
```
C = C - S[2r + 3]
A = A - S[2r + 2]
for i = r downto 1 do
{
(A,B,C,D) = (D,A,B,C)
u = (D × (2D + 1))<<<lg w
t = (B × (2B + 1))<<<lg w
C = ((C - S[2i+ 1])>>>t) ⊕ u
A = ((A - S[2i])>>>) ⊕ t
}
D = D - S[1]
B = B - S[0]
```

Proses dekripsi pada MRC6 sebagai berikut

```
R15 = R15 - S[8r + 15],      R13 = R13 - S[8r + 14]
R11 = R11 - S[8r + 13],      R9 = R9 - S[8r + 12]
R7 = R7 - S[8r + 11],        R5 = R5 - S[8r + 10]
R3 = R3 - S[8r + 9],         R1 = R1 - S[8r + 8]

for i = r downto 1 do
{
(R1,R2,...,R15,R16) = (R16,R1,R2,...R15)
z = (R16 × (2R16 + 1))<<<lg w
v = (R14 × (2R14 + 1))<<<lg w
u = (R12 × (2R12 + 1))<<<lg w
t = (R10 × (2R10 + 1))<<<lg w
n = (R8 × (2R8 + 1))<<<lg w
m = (R6 × (2R6 + 1))<<<lg w
l = (R4 × (2R4 + 1))<<<lg w
k = (R2 × (2R2 + 1))<<<lg w

R15 = ((R15 - S[8i+7])>>>t) ⊕ z
R13 = ((R13 - S[8i+6])>>>) ⊕ v
R11 = ((R11 - S[8i+5])>>>) ⊕ u
R9 = ((R9 - S[8i+4])>>>) ⊕ t
R7 = ((R7 - S[8i+3])>>>) ⊕ n
R5 = ((R5 - S[8i+2])>>>) ⊕ m
R3 = ((R3 - S[8i+1])>>>) ⊕ l
R1 = ((R1 - S[8i])>>>) ⊕ k
}
R16 = R16 - S[7],      R14 = B - S[6]
R12 = R12 - S[5],      R10 = B - S[4]
R8 = R8 - S[3],        R6 = B - S[2]
R4 = R4 - S[1],        R2 = B - S[0] [ELF-05]
```

Proses dekripsi dapat dilihat pada gambar 2.9



Gambar 2.9. diagram dekripsi dengan *f(x)=x * (2x+1)*

Persamaan 2.3 digunakan karena memiliki sifat satu ke satu (one-to-one) pada aritmatik modulo 2^w dan cenderung mengubah bit yang *high-order* (dekat *most significant byte*). Fungsi tersebut termasuk fungsi kuadratik yang akan cenderung menambah digit depannya. Apabila *x* pada *f(x)* yang terdiri dari *i* bit mengalami perubahan bit pada posisi ke *j* maka *f(x)* akan berubah pada bit posisi ke *j* dan cenderung pada posisi > *j*. Sifat satu ke satu akan menmabha jumlah iterasi yang memperkuat keamanan enkripsi [RIV-1998]. Langkah terinci pada enkripsi algoritma MRC6:

1. Blok plaintext dibagi enam belas bagian yaitu R1, R2,R3,..., R16 dengan panjang masing-masing *w* bit atau panjang blok dibagi enam belas. Kemudian R2, R4 hingga R16 (register genap) dijumlahkan (dalam modulo 2^w) berturut-turut dengan kunci internal S[0], S[1], S[2] hingga S[7].

   R2 ← R2 + S[0]

   R4 ← R4 + S[1]

   ↓

   R16 ← R16 + S[7]

2. Pada setiap putaran 1 sampai *r* dilakukan XOR dan pergeseran ke kiri terhadap R1 dengan *f(x)* yang digeser ke kiri sebanyak lg *w*, di mana *f(x)* = *x*(2x+1)* dan *x* = R2. Setelah itu dilakukan penjumlahan dengan kunci internal (dengan modulo 2^w). Perlakuan sama terhadap R3 dengan *x* = R4, dan seterusnya. Kemudian dilakukan *swapping* R1 ← R2, R2 ← R3, R3 ← R4 sampai R16 ← R1.

3. Setelah iterasi selesai, langkah akhir adalah melakukan penjumlahan (dalam modulo 2^w) terhadap R1, R3 hingga R15 (register ganjil) dengan delapan kunci internal terakhir. Setelah semua blok yang terbagi menjadi 16 bagian disatukan kembali.

R1 ← R1 + S[8r + 8]

R3 ← R3 + S[8r + 9]

R15 ← R15 + S[8r + 15] [ELF-05].

2.6.4. Padding

Ada kemungkinan panjang plainteks tidak habis dibagi dengan panjang ukuran blok yang ditetapkan (misalnya 64 bit atau lainnya). Hal ini mengakibatkan blok terakhir berukuran lebih pendek dari pada blok-blok lainnya. Satu cara untuk mengatasi hal ini adalah dengan *padding*, yaitu menambahkan blok terakhir dengan pola bit yang teratur agar panjangnya sama dengan ukuran blok yang ditetapkan. Misalnya ditambahkan bit 0 semua, atau bit 1 semua, atau bit 0 dan bit 1 berselang-seling. Misalkan ukuran blok adalah 64 bit (8 *byte*) dan blok terakhir terdiri dari 24 bit (3 *byte*). Tambahkan blok terakhir dengan 40 bit (5 *byte*) agar menjadi 64 bit, misalnya dengan menambahkan 4 buah *byte* 0 dan satu buah *byte* angka 5. Setelah dekripsi, hapus 5 *byte* terakhir dari blok dekripsi terakhir [MUN-06].

2.7 Nilai Korelasi

Nilai korelasi digunakan untuk mengukur tingkat korelasi atau kemiripan antara dua buah piksel yang saling bertetanggaan pada citra digital. Nilai korelasi dihitung dengan menggunakan Persamaan 2.4.

$$r = \frac{n\sum xy - \sum x \sum y}{\sqrt{[n\sum(x^2) - (\sum x)^2][n\sum(y^2) - (\sum y)^2]}} \quad (2.4)$$

Di mana :

r : Nilai korelasi

n : Jumlah pasangan piksel:

∑xy :Jumlah perkalian piksel x dengan piksel y

∑x : Jumlah data x

∑y : Jumlah data y

$\sum x^2$: Jumlah kuadrat data x

$\sum y^2$: Jumlah kuadrat data *y*

Nilai korelasi berbanding terbalik dengan kualitas pada *chipherimage*. Semakin rendah nilai korelasi yang dimiliki *chipherimage* (semakin mendekati nol) maka semakin baik kualitas yang dihasilkan dari proses enkripsi [ALI-2008].

2.8 Nilai Entropi

Suatu citra digital bisa dipandang sebagai satu sumber piksel-piksel bebas. Nilai *entropi* ini memberikan batasan minimum banyaknya bit yang diperlukan untuk mengkodekan keluaran sumber informasi tersebut. *Entropi* digunakan untuk mengetahui keragaman dari intensitas citra. Nilai *entropi* besar untuk citra dengan transisi derajat keabuan merata dan bernilai kecil jika struktur citra tidak teratur. Sehingga semakin tinggi nilai *entropi* maka *cipherimage* yang dihasilkan semakin baik, dan sebaliknya jika nilai *entropi* semakin rendah maka *cipherimage* yang dihasilkan memiliki kualitas yang semakin rendah. Indikator nilai entropi yang baik adalah apabila mendekati angka delapan. Nilai *entropi* dihitung menggunakan Persamaan 2.5 [DHA-2003].

$$He = -\sum_{k=0}^{G-1} P(k).\log2\,(P(k)) \quad \text{(bit/simbol)} \qquad (2.5)$$

Di mana :

H_e : Nilai *entropy*
G : Derajat keabu-abuan citra masukkan (dari 0-255)
P(k) : Probabilitas symbol ke-k

2.9 Citra Bitmap

bitmap adalah sebuah format citra yang digunakan untuk menyimpan citra bitmap digital. Pada citra berformat BMP (*bitmap*) yang tidak terkompresi, piksel citra disimpan dengan kedalaman warna 1, 4, 8, 16, 24, atau 32 bit per piksel. Terjemahan bebas bitmap adalah pemetaan bit. Artinya nilai intensitas piksel di dalam citra dipetakan ke sejumlah bit tertentu. Peta bit umumnya adalah 8, yang berarti setiap piksel panjangnya 8 bit. Delapan bit ini mempresentasikan nilai intensitas piksel. Dengan demikian ada sebanyak 2^8 =256 derajat keabuan, mulai dari 0 (00000000) sampai 255 (11111111).

Pada umumnya citra bitmap terdiri dari 4 blok data yaitu: *BMP header*, *Bit Information (DIB header)*, *Color Palette*, dan *Bitmap Data*. *BMP header* berisi informasi umum dari citra *bitmap* yang berada pada bagian awal file citra dan digunakan untuk mengidentifikasi citra. *Bit information* berisi informasi detail dari citra bitmap, yang akan digunakan untuk menampilkan citra pada layar. *Color palette* berisi informasi warna yang digunakan untuk indeks warna bitmap, dan *bitmap data* berisi data citra yang sebenarnya, piksel per piksel.

Model ruang warna yang digunakan pada citra *bitmap* adalah RGB (*red, green, dan blue*). Sebuah ruang warna RGB dapat diartikan sebagai semua kemungkinan warna yang dapat dibuat dari tiga warna dasar *red*, *green*, dan *blue*. RGB sering digunakan di dalam sebagian besar aplikasi komputer karena dengan ruang warna ini tidak diperlukan transformasi untuk menampilkan informasi di layar monitor. [MUN-04].

Terdapat tiga macam citra dalam format BMP, adalah sebagai berikut:

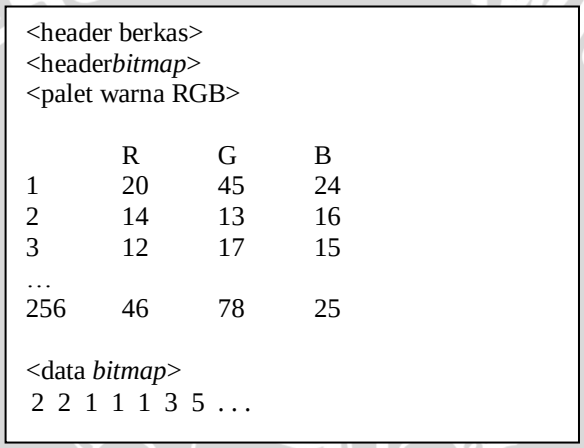
1. **Citra biner**. Citra biner hanya memiliki dua nilai keabuan 0 dan 1. Oleh kerena itu 1 bit telah cukup untuk mempresentasikan nilai piksel.
2. **Citra hitam-putih** (*grayscale*).
3. **Citra berwarna**. Citra berwarna adalah citra yang lebih umum. Warna yang terlihat didalam citra bitmap merupakan kombinasi dari tiga komponen warna, yaitu : R (Red), G (Green) dan B (Blue). Pada citra 256 warna, setiap piksel memiliki panjang 8-bit, akan tetapi komponen RGBnya disimpan dalam tabel RGB yang disebut *palet*. Berikut akan memperlihatkan panjang informasi palet untuk setiap versi *bitmap*.

Tabel 2.1 Panjang informasi *palet bitmap* berwarna

Citra *m*	warna *Palet bitmap*
Citra 16	warna 64 *byte*
Citra 256	warna 1024 *byte*
Citra 16,7 juta	warna 0 *byte*

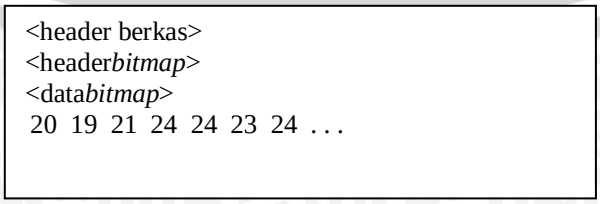
Format citra 8-*bit* dapat dilihat pada gambar 2.7. Format citra 4-*bit* (16 warna), hampir sama dengan format citra 8-*bit*. Pada citra 4-*bit* dan citra 8-*bit*, warna suatu piksel diacu dari tabel informasi palet *entry* ke-k (*k* merupakan nilai

rentang 0-15 untuk citra 16 warna dan 0-155 untuk citra 256 warna). Sebagai contoh pada gambar 2.7, piksel pertama bernilai 2, warna piksel pertama ini ditentukan oleh komponen RGB pada *palet* warna *entry* ke-2, yaitu R=14, G=13 dan B=16.piksel kedua serupa dengan piksel pertama. Piksel ketiga bernilai 1, warna ditentukan oleh komponen RGB pada *palet* warna *entry* ke-1, yaitu R=20, G=45 dan B=24. Demikian seterusnya untuk piksel-piksel lainnya. Khusus untuk citra hitam-putih 8-*bit*, komponen R,G dan B suatu piksel bernilai sama dengan data bitmap piksel tersebut. Jadi piksel dengan nilai data bitmap 129, memiliki nilai R=129, G=129 dan B=129.



Gambar 2.7 Format citra 8-bit [MUN-04].

Citra yang lebih kaya warna adalah citra 24-*bit*. Setiap piksel panjangnya 24-*bit*, karena setiap bit langsung menyatakan komponen warna merah (8-bit), komponen warna hijau (8-*bit*) dan komponen warna biru (8-*bit*). Citra 24-*bit* juga disebut citra 16 juta warna karena mampu menghasilkan 2^{24} =16.777.216 kombinasi warna. Contohnya seperti pada Gambar 2.8 berikut ini, dimana piksel pertama memiliki nilai R=20, G=19 dan B=21. Piksel kedua memiliki nilai R=24, G=23 dan B=24 dan demikian seterusnya.



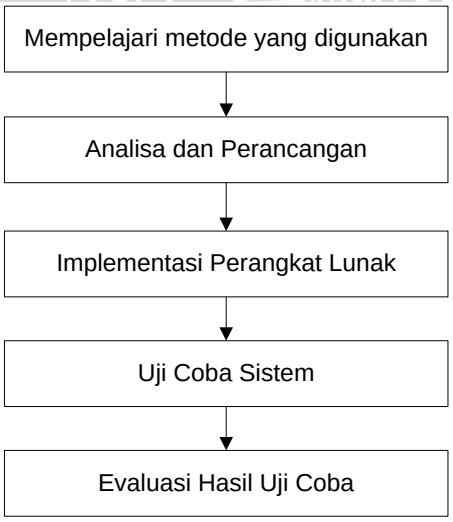
Gambar 2.8 Format citra 24-bit [MUN-04].

BAB III
METODE PENELITIAN DAN PERANCANGAN

Pada bab metode penelitian ini akan dibahas rancangan dan langkah – langkah yang dilakukan dalam pembuatan aplikasi enkripsi citra digital menggunakan algoritma *Modified Rivest Code 6*. Penelitian dilakukan dengan tahapan-tahapan sebagai berikut:

1. Mempelajari metode yang digunakan dari berbagai sumber seperti yang telah dijelaskan pada bab 2.
2. Menganalisa dan merancang perangkat lunak dengan metode yang akan digunakan.
3. Membuat perangkat lunak berdasarkan analisis dan perancangan yang dilakukan.
4. Uji coba enkripsi dan dekripsi citra digital menggunakan perangkat lunak yang telah dibuat.
5. Evaluasi hasil uji coba.

Langkah – langkah penelitian ini ditunjukkan oleh Gambar 3.1



Gambar 3.1 Tahapan – Tahapan Penelitian

3.1 Analisa Perangkat Lunak

Pada subbab ini akan dibahas mengenai deskripsi perangkat lunak dan batasan perangkat lunak.

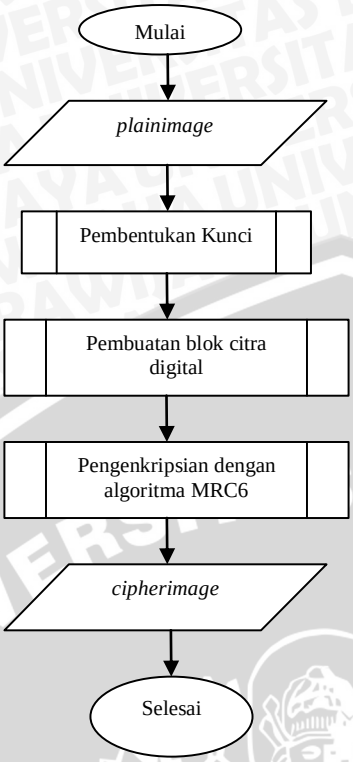
3.1.1 Deskripsi Perangkat Lunak

Aplikasi perangkat lunak yang akan dibuat merupakan implementasi teknik kriptografi yaitu algoritma *Modified Rivest Code 6* yang digunakan untuk mengenkripsi citra digital. Pada aplikasi ini terdapat dua proses utama yaitu proses pengacakan (enkripsi) dan proses penguraian kembali citra yang telah teracak (dekripsi).

Pada proses pengamanan citra digital, aplikasi akan mengubah citra digital asli (*plainimage*) menjadi citra digital yang tidak bisa dimengerti (*cipherimage*). Pembentukan kunci dilakukan dengan melakukan iterasi terhadap kunci yang telah dimasukan sehingga mendapatkan tabel kunci S. Sedangkan pada citra digital, setiap piksel citra digital diubah ke dalam bentuk biner dan dipecah menjadi blok-blok dengan panjang 512 bit. Kemudian untuk setiap blok dibagi lagi menjadi enam belas register yaitu [R1...R16] yang setiap register panjangnya 32 bit. Setelah itu proses enkripsi dilakukan menggunakan algoritma Modified Rivest Code 6. Kemudian blok-blok hasil enkripsi disusun kembali membentuk *cipherimage* yang sesuai dengan ukuran inputan citra digital. Proses – proses untuk melakukan enkripsi adalah sebagai berikut:

1. Memasukkan citra digital *bitmap* 24 bit dan kunci.
2. Pembentukan kunci dari inputan kunci.
3. Pembuatan blok-blok bit citra digital dari inputan citra digital *bitmap* 24 bit.
4. Enkripsi menggunakan algoritma *Modified Rivest Code 6*.
5. Penyusunan kembali blok-blok yang telah dienkripsi menjadi citra digital.

Langkah – langkah pengamanan citra digital ditunjukkan pada Gambar 3.2.

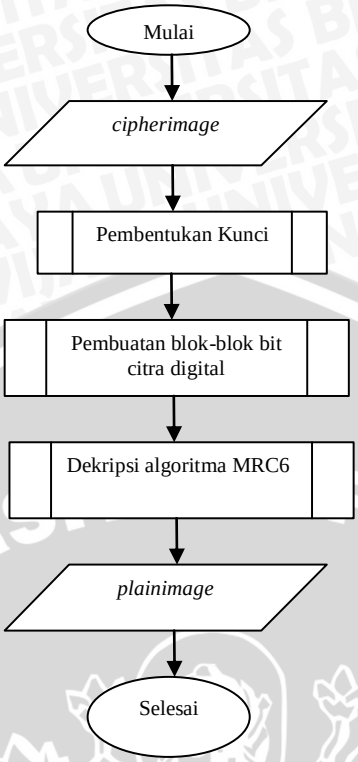


Gambar 3.2 *Flowchart* Enkripsi Citra Digital

Untuk proses penguraian data, aplikasi menerima inputan berupa *cipherimage* yang akan didekripsikan menjadi *plainimage.* Sama dengan proses enkripsi, *pixelcipherimage* terlebih dahulu akan diubah menjadi deretan biner dan dibagi menjadi beberapa blok dengan panjang tiap blok 512 bit. Blok-blok tersebut kemudian didekripsikan menggunakan algoritma *Modified Rivest Code 6*. Secara garis besar blok-blok hasil enkripsi disusun kembali membentuk *plainimage.* Proses – proses untuk proses dekripsi citra digital:

1. Memasukkan *cipherimage* enkripsi algoritma *Modified Rivest Code 6* dan kunci (*key)*.
2. Pembentukan *key* dan inputan *key*.
3. Pembuatan blok-blok bit citra digital dari inputan *cipherimage*.
4. Dekripsi dengan algoritma *Modified Rivest Code 6.*
5. Penyusunan kembali blok-blok yang sebelumnya didekripsi menjadi citra digital.

Langkah – langkah proses dekripsi citra digital ditunjukkan pada Gambar 3.3.



Gambar 3.3 *Flowchart* Dekripsi Citra Digital

3.1.2 Batasan Perangkat Lunak

Data yang akan diambil untuk diproses pada sisi perangkat lunak menggunakan citra digital berformat *bitmap* 24 bit. Untuk *key* yang digunakan menggunakan karakter ASCII.

3.2 Perancangan Perangkat Lunak

3.2.1 Proses Pembentukan *Key*

Key digunakan untuk menjaga file yang telah terenkripsi agar saat akan dikembalikan ke bentuk aslinya *key* yang dimasukan harus cocok. Sehingga inputan pada perangkat lunak ini selain citra digital adalah sejumlah karakter yang akan dibangkitkan menjadi kunci internal untuk mengisi tabel kunci S.

Langkah-langkah dalam proses pembentukan *key* adalah sebagai berikut:

1. Inputan berupa karakter sebagai *key*.
2. Proses *padding* untuk menjaga panjang *key* agar tidak kurang dari 256 bit atau 32 *byte*. Saat panjang *key* yang dimasukkan kurang dari 32 *byte*, maka

akan ditambahkan sejumlah bit 0 agar panjang *key* akhirnya menjadi 16 *byte*, 24 *byte* atau 32 *byte*. Apabila panjang *key* yang diinputkan sudah memiliki panjang 32 *byte* maka tidak perlu menambahkan lagi bit 0 pada kunci tersebut.

3. Melakukan duplikasi kunci yang masih berupa *array of bytes* ke dalam *array of word* L[0...*c*-1].

Di mana:

u=*w*/8 à *w* = ukuran kata dalam satuan bit

c=*b*/*u* à *b* = ukuran kunci dalam satuan *byte*

4. Inisialisasi, sehingga tabel S berisi pola bit *pseudo-random* yang tetap dengan algoritma:

$S[0] = P_w$

for *i* = 1 **to** (8*r*+15) **do**

$S[i] = S[i-1] + C_q$

5. Menggabungkan S dan L dalam 3 langkah, dengan algoritma:

$i = 0$

$j = 0$

$A = 0$

$B = 0$

for $3 \max(8(t+1))$ **times do**

$S[i] = (S[i] + A + B) \lll 3$

$A = S[i]$

$L[j] = (L[j] + A + B) \lll (A+B)$

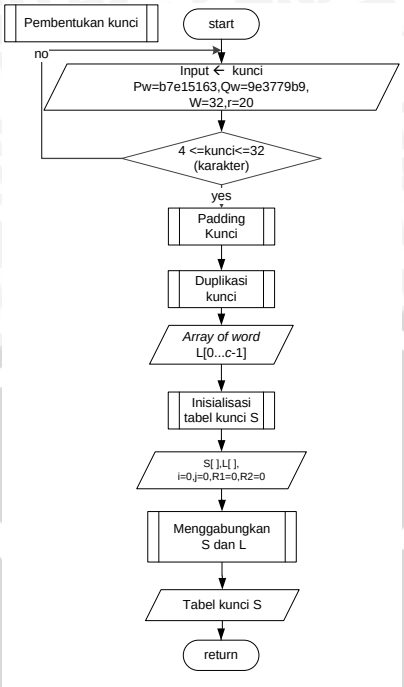
$B = L[j]$

$i = (i+1) \bmod (t)$

$j = (j+1) \bmod c$

6. Menyusun tabel *key* S.

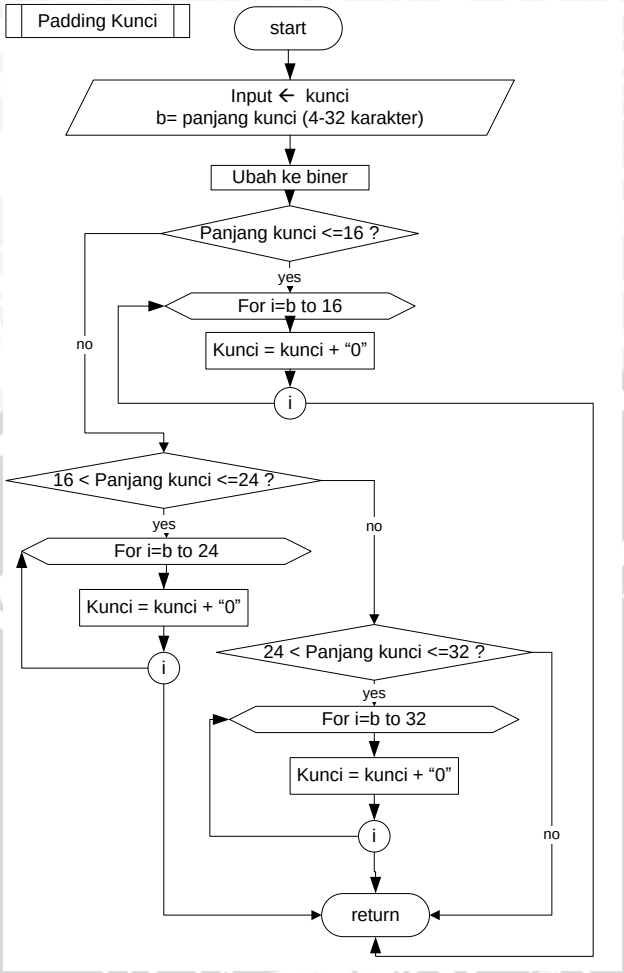
Diagram alir untuk pembentukan *key* ditunjukkan pada Gambar 3.4.



Gambar 3.4 *Flowchart* Pembentukan *key*

3.2.1.1 Sub-Proses Key Padding

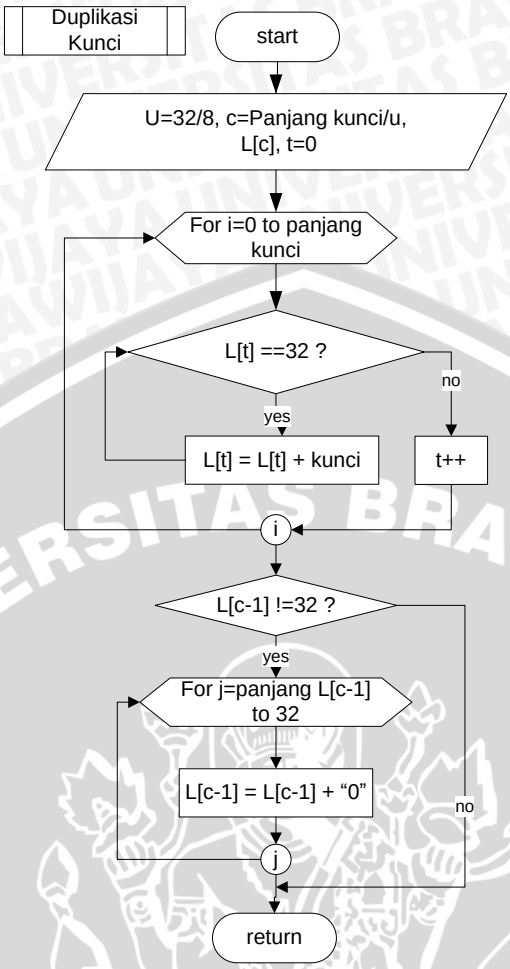
Diagram alir tahapan proses *key padding* berfungsi untuk menjaga *key* memiliki panjang tidak lebih dari 256 bit seperti ditunjukan Gambar 3.5.



Gambar 3.5 *Flowchart* Padding *key*

3.2.1.2 Sub-Proses Duplikasi Key

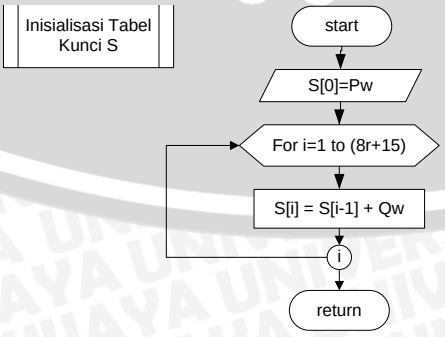
Diagram alir tahapan proses duplikasi *key* yang berfungsi untuk mengubah *array of bytes* ke dalam *array of word* L[0...*c*-1], ditunjukan pada Gambar 3.6.



Gambar 3.6 *Flowchart* Duplikasi *Key*

3.2.1.3 Sub-Proses Inisialisasi Tabel *key* S

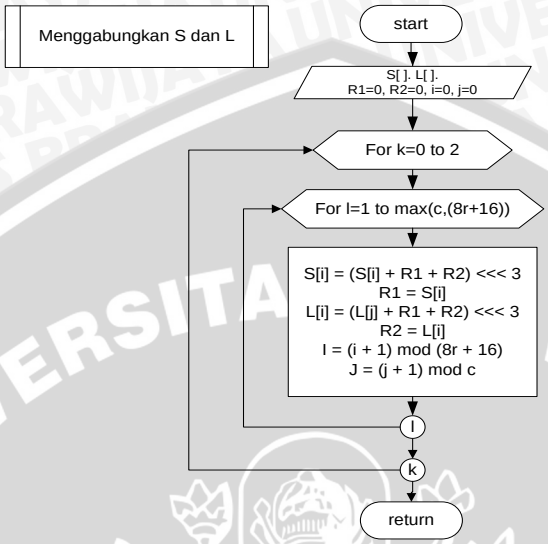
Diagram alir tahapan proses inisialisasi tabel s berfungsi untuk membentuk tabel *key* S, ditunjukan pada Gambar 3.7.



Gambar 3.7 *Flowchart* Inisialisasi Tabel *Key* S

3.2.1.4 Sub-Proses *Merging* S dan L

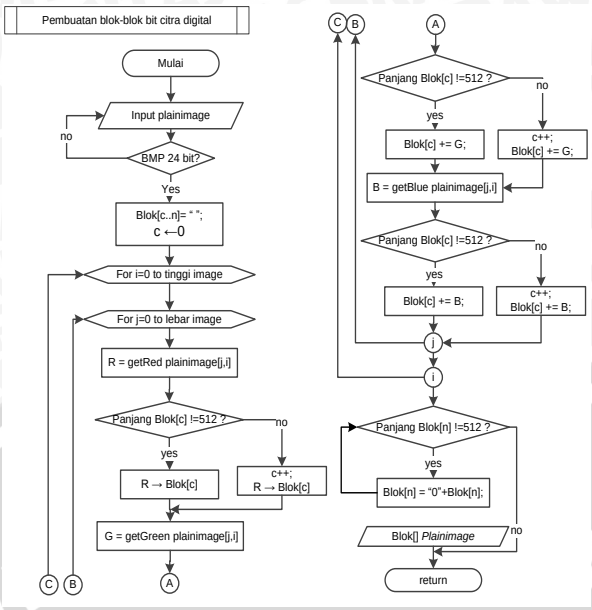
Diagram alir tahapan penggabungan tabel *key* S dengan array L untuk membentuk tabel *key* S yang baru, ditunjukan pada Gambar 3.8.



Gambar 3.8 *Flowchart* Penggabungan S dan L

3.2.2 Proses Pembentukan Blok Bit Citra

Proses awal pembuatan blok-blok bit dari *plainimage* adalah melakukan pengecekan terhadap citra apakah telah berformat *bitmap* (*.bmp) dengan *bit-depth* 24 bit. Setiap piksel dari citra digital akan diambil sampel komposisi warnanya (RGB) untuk ditentukan nilainya. Setelah ditentukan nilainya, dikonversi ke dalam bentuk biner dan disusun menjadi blok-blok sepanjang 512 bit. Apabila panjang blok yang terakhir kurang dari 512 bit maka dilakukan proses padding, yaitu menambahkan bit nol hingga blok memiliki panjang 512 bit. Algoritma enkripsi Modified Rivest Code 6 akan digunakan untuk melakukan enkripsi terhadap blok-blok tersebut. Diagram alir tahapan pembuatan blok-blok bit *plainimage* ditunjukan pada Gambar 3.9.



Gambar 3.9 *Flowchart* Pembuatan Blok-Blok Bit *Plainimage*

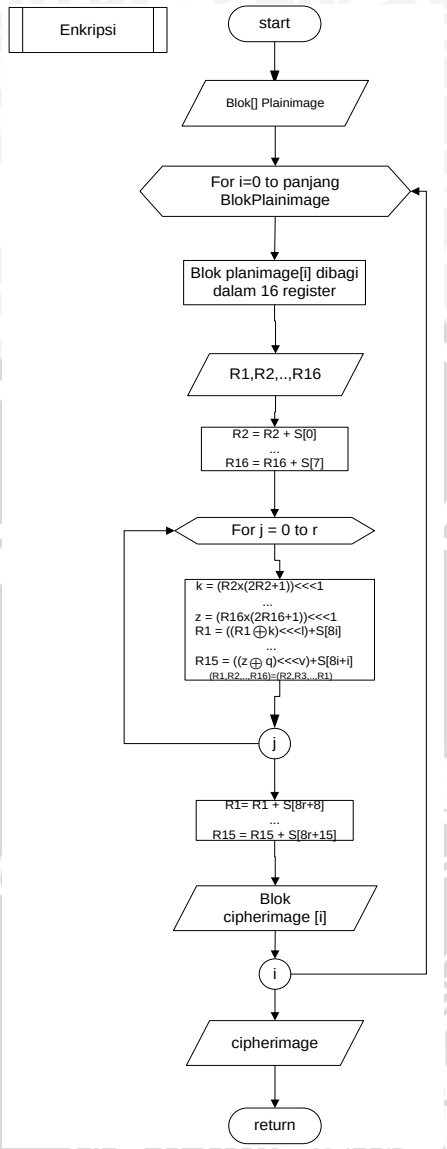
3.2.3 Proses Enkripsi Menggunakan Algoritma *Modified Rivest Code 6*

Tujuan pengenkripsian blok-blok bit *plainimage* adalah untuk mendapatkan *cipherimage.* Proses pengenkripsian memiliki tahapan:

1. Inputan berupa blok-blok bit *plainimage* yang setiap blok berukuran 512 bit didapatkan dari proses pembentukan blok-blok bit *plainimage*.
2. Setiap blok dibagi ke dalam 16 register Ri yang masing-masing berupa w-bit word, [i=1,2,3,...,16] . Sehingga setiap register memiliki ukuran sebesar 32 bit.
3. Tabel *key* S yang didapat dari proses pembentukan *key*, digunakan untuk menyamarkan iterasi yang pertama dan yang terakhir dari proses enkripsi.
4. Pada setiap iterasi menggunakan 8 buah *subkey* (k,l,m,n,t,u,v,z). Untuk mendapat nilai k - z, setiap putaran mengikuti aturan sebagai berikut, nilai R_2,$R_{4,...,}R_{16}$ dimasukkan ke dalam fungsi *f*, yang didefinisikan sebagai $f(x)=x*(2x+1)$, kemudian diputar ke kiri sejauh 5 bit. Perputaran 5 bit digunakan karena fungsi *f(x)* adalah fungsi *one-to-one* modulo logaritma basis 2 *w*. Karena *w* adalah 32 bit, maka lg(w)=5.
5. Kemudian Nilai k - z di XOR dengan $R_{1,}$ $R_{3,}$ ...,R_{15} kemudian diputar ke kiri sejauh berturut-turut berurutan l,k,n,m,u,t,z,v bit dan dijumlahkan dengan sub kunci S[8i],S[8i+1]...S[8i+7] akan menghasilkan nilai $R_{1,}R_{3,}...,R_{15}$.

6. Enam belas register dari setiap blok kemudian dipertukarkan dengan mengikuti aturan nilai R_1 menggantikan nilai R_2, nilai R_2 menggantikan nilai R_3, demikian seterusnya sehingga dapat digambarkan $(R_1,R_2,R_3,..,R_{16})=(R_2,R_3,R_4,...,R_1)$.
7. Demikian iterasi pada langkah 4 - 6 berlangsung hingga r putaran.
8. Setelah putaran ke-r selesai, dilakukan proses *whitening* akhir dimana nilai R_1 dijumlahkan dengan S[8r+8], R_3 dijumlahkan dengan S[8r+9] seterusnya hingga R_{15} dijumlahkan dengan S[8r+15].
9. Setelah proses *whitening* akhir, nilai R_1 hingga R_{16} yang berupa blok-blok bit disusun kembali menjadi citra.
10. Hasil akhir dari proses enkripsi adalah berupa citra digital yang telah disembunyikan informasinya (*cipherimage)*.

Diagram alir untuk proses enkripsi ditunjukan Gambar 3.10.

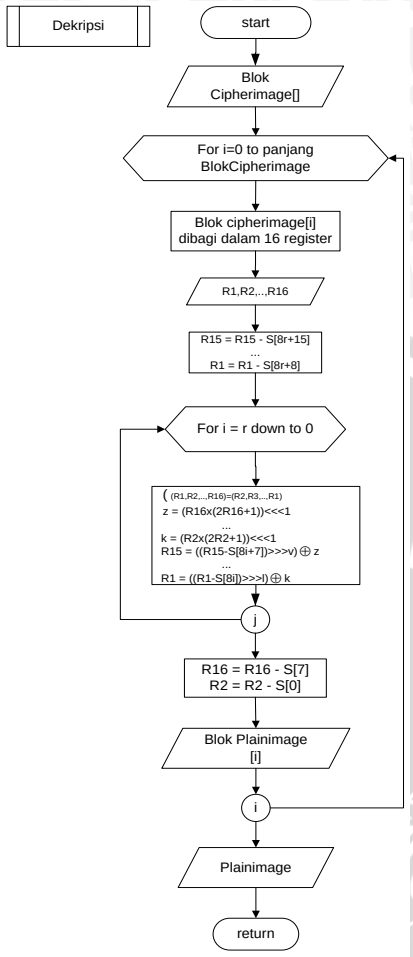


Gambar 3.10 *Flowchart* Enkripsi *Modified Rivest Code 6*

3.2.4 Proses Dekripsi Menggunakan Algoritma *Modified Rivest Code 6*

Proses dekripsi merupakan kebalikan dari proses enkripsi. Untuk tiap-tiap rotasi, operasi yang digunakan adalah operasi pengurangan, terbalik dengan pada proses enkripsi. Urutan sub kunci yang digunakan juga terbalik.

Diagram alir proses dekripsi ditunjukkan pada gambar 3.11.

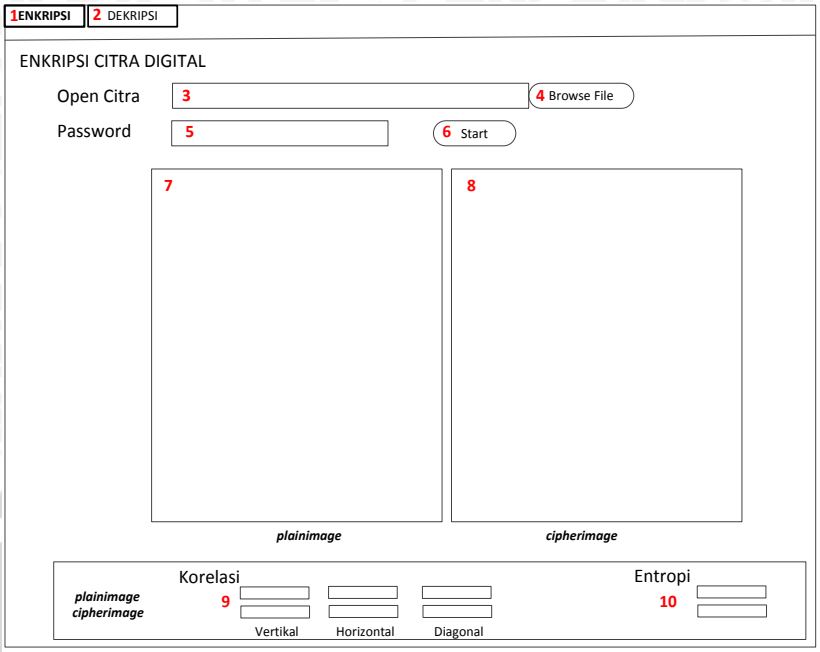


Gambar 3.11 *Flowchart* Dekripsi *Modified Rivest Code 6*

3.3 Perancangan *Interface*

3.3.1 *Interface* Menu Enkripsi

Rencana rancangan awal antar muka (*interface)* yang akan digunakan dalam perangkat lunak dapat dilihat pada gambar 3.12.



Gambar 3.12 Rancangan *Interface* Perangkat lunak

Keterangan gambar:

1. *Button* menu enkripsi.
2. *Button* menu dekripsi.
3. *Field path* berkas citra yang dipilih.
4. *Button* untuk meramban berkas citra dari media simpan.
5. *Field* untuk input *password*.
6. *Button* proses enkripsi.
7. Area citra yang akan dienkripsi
8. Area untuk memuat citra hasil enkripsi
9. *Field* Nilai korelasi
10. *Field* Nilai *entropi*

3.4 Perhitungan Manual

Sampel yang diambil dan digunakan untuk simulasi penghitungan adalah input berupa citra *bitmap* 24 bit berukuran 4x4 *pixel*, dengan *password* 8 karakter, yaitu "modified". Parameter yang digunakan dalam perhitungan manual algoritma *Modified Rivest Code 6* antara lain w=32, r=5, dan b=8.

3.4.1 Perhitungan Proses Pembentukan *Key*

Kunci internal dibangkitkan berdasarkan *password* yang dimasukan untuk mengisi tabel *key* S. Untuk mencari ukuran tabel *key* S yang telah dijelaskan pada bab 2.6.3 digunakan persamaan $t=8(r+16)$ dimana pada contoh perhitungan manual ini dimisalkan nilai r adalah 1 putaran maka didapatkan ukuran tabel kunci S adalah $t=56$.

Langkah-langkah pembentukan *key*:

1. Misalkan *password* yang digunakan adalah "modified" maka didapatkan panjang 8 karakter untuk nilai b.
2. Ubah *password* "modified" ke dalam bentuk biner, sehingga didapatkan hasil:

   modified= 1101101 1101111 1100100 1101001 1100110 1101001 1100101 1100100
3. *Password* "modified" dalam *array of bytes* yang berarti tiap *array* berukuran 1 *byte* atau 8 bit, adalah:

   L[1] = 1101101
   L[2] = 1101111
   L[3] = 1100100
   L[4] = 1101001
   L[5] = 1100110
   L[6] = 1101001
   L[7] = 1100101
   L[8] = 1100100

Kemudian *Password* "modified" diduplikasi dari *array of bytes* ke *array of word* L[0…*c*-1]:

Panjang kunci *b* = 8 *byte*

u = *w* / 8 → *w* = 32

= 32 / 8 = 4

c = *b* / *u* → *b* = 8

= *8 / 4 = 2*

L[0…*c*-1] = L[0...2-1]

= L[0... 1], yaitu L[0] dan L[1]

L[0] = L[1...4] dari *array of bytes*

= 1101101 1101111 1100100 1101001

L[1] = L[5...8] dari *array of bytes*

= 1100110 1101001 1100101 1100100

4. Inisialisasi tabel kunci S sehingga berisi pola bit *pseudo-random* yang tetap dengan *r* = 1 dan digunakan 2 *magic constants* yaitu P_{32} dan Q_{32}, karena ukuran *w* (*word*) dalam *Modified Rivest Code 6* adalah 32 maka nilai konstanta adalah:

P_{32} = b7e15163

= 10110111111000010101000101100011

Q_{32} = 9e3779b9

= 10011110001101110111100110111001

Algoritma tabel *key* S sehingga berisi pola bit *pseudo-random* adalah:

$S[0] = P_w$

= 10110111111000010101000101100011

for *i* = 1 **to** (8*r*+15) **do**

= 1 **to** 23 **do**

$S[i] = S[i-1] + Q_{32}$

S[0]	=	P_{32}	= 10110111111000010101000101100011
		Q_{32}	= 10011110001101110111100110111001 +
S[1]	=	S[0]+ Q_{32}	= 10101011000011000110010110001100
		Q_{32}	= 10011110001101110111100110111001 +
S[2]	=	S[1]+ Q_{32}	= 11111010001010000010001001101010
		Q_{32}	= 10011110001101110111100110111001 +
S[3]	=	S[2]+ Q_{32}	= 10100100101000011110111110100011 10
		Q_{32}	= 10011110001101110111100110111001 +
S[4]	=	S[3]+ Q_{32}	= 11001100001011111100111000010001 11
		Q_{32}	= 10011110001101110111100110111001 +
S[5]	=	S[4]+ Q_{32}	= 11110011101111011010110010000000 00
		Q_{32}	= 10011110001101110111100110111001 +
S[6]	=	S[5]+ Q_{32}	= 10001101101001011100010101110111001
S[6]	=	S[5]+ Q_{32}	= 10001101101001011100010101110111001
		Q_{32}	= 10011110001101110111100110111001 +
S[7]	=	S[6]+ Q_{32}	= 10100001011011001011010010101110010
		Q_{32}	= 10011110001101110111100110111001 +
S[8]	=	S[7]+ Q_{32}	= 10110101001100111010001111100101011
		Q_{32}	= 10011110001101110111100110111001 +
S[9]	=	S[8]+ Q_{32}	= 11001000111110101001001100011100100
		Q_{32}	= 10011110001101110111100110111001 +
...		.....	....
S[23]	=	S[22]+ Q_{32}	=11101110111011011101010000010000001 0
		Q_{32}	= 10011110001101110111100110111001
S[24]	=	S[23]+ Q_{32}	=11111000110100010100101110101011101 1

5. Menggabungkan S dan L dalam 3 langkah, algoritma yang digunakan:

   R1 = R2 = i = j = 0

   V = 3 * max (*c*,(8*r*+16)) = max (2,24) = 24

   3 langkah masing-masing 24 putaran.

   $S[i] = (S[i] + R1 + R2) < < 3$

   $R1 = S[i]$

   $L[i] = (L[j] + R1 + R2) < < 3$

   $R2 = L[i]$

   $i = (i+1) \bmod 8$

   $j = (j+1) \bmod 8$

Langkah 1

1

S[*i*] = S[0]= 10110111111000010101000101100011
R1= 0
R2= 0 +
S[0]+R1+R2= 10110111111000010101000101100011
S[0]= (S[0]+R1+R2)<<<3 = 10111111000010101000101100011101
A= S[0]

L[*j*] = L[0]= 01100101011011100110101101110010
A= 10111111000010101000101100011101
B= 0 +
L[0]+R1+R2= 00100100011110001111011010001111
L[0]= (L[0]+R1+R2)<<<3 = 00100011110001111011010001111001
B= L[0]
i= (0+1) mod 24 = 1
j= (0+1) mod 2 = 1

2

S[*i*] = S[1]= 01010110000110001100101100011100
R1= 10111111000010101000101100011101
R2= 00100011110001111011010001111001 +
S[1]+R1+R2= 00111000111010110000101010110010
S[1]= (S[1]+R1+R2)<<<3 = 11000111010110000101010110010001
R1= S[1]

L[*j*] = L[1]= 01101001011100000111001101101001
R1= 11000111010110000101010110010001
R2= 00100011110001111011010001111001 +
L[1]+R1+R2= 01010100100100000111110101110011
L[1]= (L[1]+R1+R2)<<<3 = 10100100100000111110101110011010
R2= L[1]
I= (1+1) mod 24 = 2
J= (1+1) mod 2 = 0

....

24

S[*i*] = S[23]= 11000000101100100111111111001000
R1= 11000011111001100100101001111000
R2= 11100110100110100001000111101111 +
S[23]+R1+R2= 01101011001100101101110000101111
S[23] = (S[23]+A+B)<<<3 = 01011001100101101110000101111011
R1 = S[23]

L[*j*] = L[1]= 10100100100000111110101110011010
R1= 01011001100101101110000101111011
R2= 11100110100110100001000111101111 +
L[1]+R1+R2= 11100100101101001101111100000100
L[13] = (L[1]+R1+R2)<<<3 = 00100101101001101111100000100111
R2 = L[23]
I =(23+1) mod 24 = 0
J = (1+1) mod 2 = 0

Langkah 2

1

S[*i*] = S[0] = 10111111000010101000101100011101

	R1	= 01011001100101101110000101111011	
	R2	= 00100101101001101111100000100111	+
	S[0]+R1+R2	= 00111110010010000110010010111111	
S[0]	= (S[0]+R1+R2)<<<3	= 11110010010000110010010111111001	
R1	= S[0]		
	L[*j*] = L[0]	= 00100011110001111011010001111001	
	R1	= 11110010010000110010010111111001	
	R2	= 00100101101001101111100000100111	+
	L[0]+R1+R2	= 00111011101100011101001010011001	
L[0]	= (L[0]+R1+R2)<<<3	= 11011101100011101001010011001001	
B	= L[0]		
I	= (0+1) mod 24 = 1		
J	= (0+1) mod 2 = 1		
2	S[*i*] = S[1]	= 11000111010110000101010110010001	
	R1	= 11110010010000110010010111111001	
	R2	= 11011101100011101001010011001001	+
	S[1]+R1+R2	= 10010111001010100001000001010011	
S[1]	= (S[1]+R1+R2)<<<3	= 10111001010100001000001010011100	
R1	= S[1]		
	L[*j*] = L[1]	= 10100100100000111110101110011010	
	R1	= 10111001010100001000001010011100	
	R2	= 11011101100011101001010011001001	+
	L[1]+R1+R2	= 00111011011000110000001011111111	
L[1]	= (L[1]+R1+R2)<<<3	= 11011011000110000001011111111001	
R2	= L[1]		
I	= (1+1) mod 24 = 2		
J	= (1+1) mod 2 = 0		
.....	.....	.....	.....
24	S[*i*] = S[23]	= 01011001100101101110000101111011	
	R1	= 11110101101111011100010000000111	
	R2	= 10010001001001011100100111001111	+
	S[23]+R1+R2	= 11100000011110100110111101010001	
S[23]	= (S[23]+R1+R2)<<<3	= 00000011110100110111101010001111	
R1	= S[23]		
	L[*j*] = L[1]	= 11011011000110000001011111111001	
	R1	= 00000011110100110111101010001111	
	R2	= 10010001001001011100100111001111	+
	L[1]+R1+R2	= 01110000000100010101110001010111	
L[13]	= (L[1]+R1+R2)<<<3	= 10000000100010101110001010111011	
R2	= L[23]		
i	=(23+1) mod 24 = 0		
j	= (1+1) mod 2 = 0		

Langkah 3

1	S[*i*] = S[0]	= 11110010010000110010010111111001	
	R1	= 00000011110100110111101010001111	
	R2	= 10000000100010101110001010111011	+

	S[0]+R1+R2	= 01110110101000011000001101000011	
S[0]	= (S[0]+R1+R2)<<<3	= 10110101000011000001101000011011	
R1	= S[0]		
	L[*j*] = L[0]	= 11011101100011101001010011001001	
	R1	= 10110101000011000001101000011011	
	R2	= 10000000100010101110001010111011	+
	L[0]+R1+R2	= 00010011001001011001000110011111	
L[0]	= (L[0]+R1+R2)<<<3	= 10011001001011001000110011111000	
R2	= L[0]		
i	= (0+1) mod 24 = 1		
j	= (0+1) mod 2 = 1		
2	S[*i*] = S[1]	= 10111001010100001000001010011100	
	R1	= 10110101000011000001101000011011	
	R2	= 10011001001011001000110011111000	+
	S[1]+R1+R2	= 00000111100010010010100110101111	
S[1]	= (S[1]+R1+R2)<<<3	= 00111100010010010100110101111000	
R1	= S[1]		
	L[*j*] = L[1]	= 11011011000110000001011111111001	
	R1	= 00111100010010010100110101111000	
	R2	= 10011001001011001000110011111000	+
	L[1]+R1+R2	= 10110000100011011111001001101001	
L[1]	= (L[1]+R1+R2)<<<3	= 10000100011011111001001101001101	
R2	= L[1]		
i	= (1+1) mod 24 = 2		
j	= (1+1) mod 2 = 0		
...	.....	.....	
24	S[*i*] = S[23]	= 11010101110110001011101101001010	
	R1	= 11101001000010010000011111101110	
	R2	= 01110100000000110000101011111010	+
	S[10]+R1+R2	= 00110010111001001100111000110010	
S[23]	= (S[23]+R1+R2)<<<3	= 10010111001001100111000110010000	
R1	= S[23]		
	L[*j*] = L[0]	= 10011001001011001000110011111000	
	R1	= 10010111001001100111000110010001	
	R2	= 01110100000000110000101011111010	+
	L[0]+R1+R2	= 10100100010101100000100110000011	
L[23]	= (L[0]+R1+R2)<<<3	= 00100010101100000100110000011101	
R2	= L[23]		
i	=(23+1) mod 24 = 0		
j	= (0+1) mod 2 = 1		

6. Menyusun tabel kunci S dalam bentuk *array* S. Ditunjukkan pada tabel 3.1

Tabel 3.1 Tabel Kunci S

S[0]	= 10110101000011000001101000011011
S[1]	= 00111100010010010100110101111000
S[2]	= 01111000111110011010110001001011
S[3]	= 11110101101101110101010000110101
S[4]	= 01001000100011011110001010101110
S[5]	= 01100100101000110001011111100111
S[6]	= 01011000110000101000001111010110
S[7]	= 01101000011001110111100100110011
S[8]	= 10101110001000111111011011111000
S[9]	= 11101001000010010000011111101110
..	.....
S[23]	= 10010111001001100111000110010001

3.4.2. Perhitungan Proses Pembentukan Blok Bit Citra

Piksel citra yang digunakan sebagai model perhitungan adalah citra BMP 24 bit berukuran 7x3 piksel dan kunci yang digunakan adalah "*modified*".
Nilai warna RGB yang dimiliki oleh citra uji dapat diketahui pada setiap pikselnya. Nilai RGB yang didapat kemudian dikonversi menjadi bentuk biner. Hasil konversi nilai RGB setiap piksel pada citra uji ke dalam bentuk biner yang ditunjukan pada Tabel 3.2.

Tabel 3.2 Nilai RBG dan Biner Citra Uji

Piksel	R	G	B	Biner RGB
Pixel ke - 1	14	13	12	10010011-10001011-10000000
Pixel ke - 2	7	9	8	10011100-10100011-10001010
Pixel ke - 3	15	16	13	10101001-10111000-10001011
Pixel ke - 4	6	3	8	10101110-10111101-10000010
Pixel ke - 5	16	18	13	10101101-10110011-01110000
.....	.....	.....	.....	.....
Pixel ke - 21	152	180	133	10011000-10110100-10000101

Nilai RGB akan dimasukkan dalam blok 512 bit untuk memulai proses enkripsi dan dekripsi.

Plainimage =

Blok

→ 10010011100010111000000010011100
10100011100010101010100110111000
10001011101011101011110110000010
10101101101100110111000010111001

11000001100111001011000011000101
10110101101110001100001110101111
01100000011101110111011110100110
10110010101000001101000011011010

10101111110010011100100110101000
10000100100010110111111010011101
10100010011110111001100010110100
10000101101011001010110110001110

10011100101000111000101010101001
11000001100111001011000011000101
11001001101010001000010010001011
01111011100110001011010010000101

3.4.3 Perhitungan Proses Enkripsi Algoritma *Modified Rivest Code 6*

Pada perhitungan proses enkripsi inputannya berasal dari proses pembentukan kunci dan pembentukan blok bit *plainimage.* Langkah-langkah perhitungan proses enkripsi adalah sebagai berikut:

1. Pada contoh perhitungan manual, masukan berupa blok-blok bit *plainimage* yang ukurannya telah dibuat menjadi 512 bit.
2. Blok tersebut dibagi dalam 16 register, yaitu R1, R2,...,R16. Sehingga setiap register memiliki ukuran sebesar 32 bit.
3. Tabel kunci S yang didapat dari proses pembentukan kunci akan digunakan untuk proses *whitening* awal. Proses *whitening* awal dilakukan dengan menjumlahkan register R2 dengan S[0], register R4 dengan S[1] seterusnya hingga register R16 dengan S[7].

	R2	= 10100011100010101010100110111000	
	S[0]	= 10110101000011000001101000011011	+
R2 = R2 + S[0]		= 01011000100101101100001111010011	
.. ..	.....	.....	
	R16	= 01111011100110001011010010000101	
	S[7]	= 01101000011001110111100100110011	+
R16 = R16 + S[7]		= 11100100000000000010110110111000	

4. Setelah Proses *whitening* awal dilakukan putaran sebanyak 1 putaran (r =1) pada algoritma berikut:

   k = (R2*(2R2+1))<<<1

   ↓

   z=(R16*(2R16+1))<<<1

   R1 = ((R⊕ k)<<<l)+S[8i]

   ↓

   R15 = ((R15 ⊕ z)<<<v)+S[8i+7]

   (R1,R2,...,R15,R16)=(R2,R3,...R16,R1)}

Putaran 1

	R2	= 01011000100101101100001111010011	
	2R2 + 1	= 10110001001011011000011110100111	x
	R2*(2R2+1)	= 10011010101101010000001110100101	
K	= (R2x(2R2+1))<<<1	= 01010110101000000111010010110011	
	↓	↓	
	R16	= 01111011100110001011010010000101	
	2R16+ 1	= 11110111001100010110100100001011	x
	R16*(2R16+1)	= 11010010000011110100111010110111	
Z	= (R16x(2R16+1))<<<1	= 11010010000011110100111010110111	
	R1	= 10010011100010111000000010011100	
	k	= 01010110101000000111010010110011	⊕
	R1⊕k	= 11000101001010111111010000101111	
	(R1⊕k)<<<l	= 10100101011111101000010111111000	
	S[8*i*] = S[8]	= 10101110001000111111011011111000	+
R1	= ((R1⊕k)<<<l)+S[8]	= 10101001110100010011111001110000	
	↓	↓	
	R15	= 11001001101010001000010010001011	
	z	= 00110000110010010001111001101001	⊕
	R15⊕l	= 11111001011000011001101011100010	
	(R15⊕z)<<<v	= 01101100111101000111110101110111	
	S[8*i*+1] = S[9]	= 11101001000010010000011111101110	+
R15	= ((R15⊕z)<<<v)+S[9]	= 10101010111111101100001010110010 1	
	R1 =R2	= 01011000100101101100001111010011	
	↓	↓	
	R15 = R1	= 10101001110100010011111001110000	

5. Setelah selesai, dilakukan proses *whitening* akhir dimana nilai R1 dijumlahkan dengan S[8r+8] = S[16], seterusnya pada register ganjil hingga nilai R15 dijumlahkan dengan S[8r+15] = S[23].

6. Ulangi langkah 2 sampai 5 untuk setiap blok yang dimiliki oleh *plainimage*. Apabila blok terakhir kurang dari 512 bit maka dilakukan proses *padding*, yaitu dengan menambahkan bit 0 pada bagian kanan sampai blok sepanjang 512 bit. Sehingga blok-blok yang telah dienkripsi menggunakan algoritma *Modified Rivest Code 6* nantinya dapat disatukan kembali menjadi blok=blok bit *cipherimage*.

7. Blok-blok bit *cipherimage* hasil enkripsi dengan algoritma *Modified Rivest Code 6* dikembalikan ke bentuk citra dengan format BMP 24 bit.

Tabel 3.3 perubahan citra hasil enkripsi

Pikse l	***Plainimage***			***Cipherimage***		
	Nilai RGB		**Biner RGB**	**Nilai RGB**		**Biner RGB**
BLOK [0]						
ke-1	R	147	10010011	R	128	10000000
	G	139	10001011	G	185	10111001
	B	128	10000000	B	125	01111101
ke-2	R	156	10011100	R	169	10101001
	G	163	10100011	G	18	00010010
	B	138	10001010	B	83	01010011
ke-3	R	169	10101001	R	190	10111110
	G	184	10111000	G	190	10111110
	B	139	10001011	B	210	11010010
ke-4	R	174	10101110	R	140	10001100
	G	189	10111101	G	220	11011100
	B	130	10000010	B	110	01101110
ke-5	R	173	10101101	R	229	11100101
	G	179	10110011	G	140	10001100
	B	112	01110000	B	153	10011001
ke-6	R	185	10111001	R	32	00100000

Untuk proses dekripsi merupakan kebalikan dari proses enkripsi dengan urutan *subkey* yang juga terbalik. Urutan proses dekripsi yaitu

1. Masukan berupa blok-blok bit *cipherimage* yang ukurannya telah dibuat menjadi 512 bit.
2. Setiap blok dibagi dalam 16 register, sehingga setiap register memiliki ukuran sebesar 32 bit. Blok yang telah dienkripsi akan dikembalikan ke blok aslinya melalui proses dekripsi.

3. Proses *whitening* awal dimana nilai R15 dikurangkan dengan S[8r+15], seterusnya hingga nilai R1 dikurangkan dengan S[8r+8]. Nilai S diperoleh dari tabel kunci S yang memiliki r =1 putaran.
4. Setelah Proses *whitening* awal dilakukan putaran sebanyak 1 putaran (r = 1) seperti pada algoritma langkah 4 enkripsi hanya dibalik pada penentuan nilai register R1-R15

Tabel 3.4 Perubahan Citra hasil Dekripsi

Pikse l	*Cipherimage*			Citra Hasil Dekripsi		
	Nilai RGB		Biner RGB	Nilai RGB		Biner RGB
BLOK [0]						
ke-1	R	128	10000000	R	147	10010011
	G	185	10111001	G	139	10001011
	B	125	01111101	B	128	10000000
ke-2	R	169	10101001	R	156	10011100
	G	18	00010010	G	163	10100011
	B	83	01010011	B	138	10001010
ke-3	R	190	10111110	R	169	10101001
	G	190	10111110	G	184	10111000
	B	210	11010010	B	139	10001011
ke-4	R	140	10001100	R	174	10101110
	G	220	11011100	G	189	10111101
	B	110	01101110	B	130	10000010
ke-5	R	229	11100101	R	173	10101101
	G	140	10001100	G	179	10110011
	B	153	10011001	B	112	01110000
ke-6	R	32	00100000	R	185	10111001

3.5 Perancangan Uji Coba

Perancangan pengujian perangkat lunak enkripsi citra digital menggunakan algoritma *Modified Rivest Code 6* ini dimaksudkan agar dapat mengetahui kinerja dari perangkat lunak. Selain itu, sebagai bahan untuk mengevaluasi hasil dari implementasi analisa dan perancangan perangkat lunak. Bahan uji coba adalah citra dengan *format bitmap* 24 bit dengan berbagai ukuran.

3.5.1. Pengujian Fungsionalitas Perangkat Lunak

Pada pengujian ini dilakukan untuk menguji fungsi perangkat lunak dalam mengamankan sebuah citra digital. Uji coba dilakukan baik saat proses enkripsi, yaitu terhadap citra hasil enkripsi apakah telah dapat menyembunyikan informasi/ keaslian dari citra sebelumnya. Dan saat proses dekripsi uji coba dilakukan

terhadap keakuratan *password* yang dimasukan. Apabila *password* benar seharusnya proses dekripsi akan berhasil dilakukan sehingga citra yang semula tidak memiliki informasi (*cipherimage*) dapat kembali seperti citra aslinya. Tetapi apabila *password* salah seharusnya bentuk asli citra akan tetap terlundungi dan masih menghasilkan citra yang berupa cipherimage. Rancangan pengujian keamanan perangkat lunak ditunjukkan pada tabel 3.5.

Tabel 3.5 Pengujian Fungsionalitas Perangkat Lunak

no	Citra input	Nama Proses	Citra Output	Hasil
1		enkripsi		
		dekripsi (*password* benar)		
		dekripsi (*password* salah)		

3.5.2 Pengujian Hasil Proses Enkripsi

Pengujian terhadap hasil proses enkripsi yang dilakukan antara lain dengan memasukan kunci dengan panjang yang berbeda-beda pada setiap percobaan. Kemudian menghitung nilai korelasi dan *entropy* dari citra asli dan setiap percobaan yang didapatkan. Nilai korelasi yang rendah dan nilai *entropy* yang tinggi terhadap citra aslinya menunjukan kinerja perangkat lunak baik dalam menghasilkan *cipherimage*.

Rancangan pengujian nilai korelasi dan *entropy* ditunjukkan pada tabel 3.6 dan 3.7.

Tabel 3.6 Percobaan Ukuran Kunci

Percobaan	Panjang Kunci	Kunci
I	8 bit	modified
II	24 bit	Algoritmakriptografimrc6

Tabel 3.7 Pengujian Nilai Korelasi dan *Entropy*

Nama File				
Pengujian		Citra Asli	Percobaan	
			I	II
Korelasi	Horizontal			
	Vertikal			
	Diagonal			
	Rata-rata			
Entropy				

Keterangan:

- Citra Asli : citra asli yang belum dienkripsi.
- Percobaan I : citra yang telah dienkripsi dengan *password* "modified".
- Percobaan II : citra yang telah dienkripsi dengan *password* "algoritmakriptografirc6".

3.5.3 Pengujian Hasil Proses Dekripsi

Pengujian hasil proses dekripsi dilakukan untuk menguji ketahanan *cipherimage* yang dihasilkan. Uji ketahanan meliputi manipulasi citra yang umum dilakukan seperti rotasi, pemotongan citra, pengubahan ukuran citra, penambahan efek khusus terhadap *cipherimage*. Tabel pengujian ketahanan *cipherimage* dapat dilihat pada Tabel 3.8.

Tabel 3.8 Pengujian Ketahanan *Cipherimage*

***Image* Asli**		***Cipherimage***		
Manipulasi		**Citra Manipulasi**	**Citra Dekripsi**	**Hasil**
Rotasi	90° cw			
	90° ccw			
	180°			
Croping	Horizontal			
	Vertikal			
Flip	Horizontal			
	Vertikal			
Penambahan Efek	*Noise*			

Keterangan:

- *Plainimage* : Berisikan citra asli yang belum dienkripsi.
- *Cipherimage* : Berisikan citra hasil enkripsi
- Citra Manipulasi : Berisikan *cipherimage* yang telah di manipulasi sesuai dengan jenis manipulasi.
- Citra Dekripsi : Berisikan citra hasil dekripsi dari citra manipulasi.
- Hasil : Keterangan pengujian ketahanan *cipherimage* berhasil atau gagal.

BAB IV
IMPLEMENTASI

Bab ini berisi tentang implementasi sistem. Implementasi sistem merupakan penerapan dari rancangan sistem yang sudah dibahas pada bab sebelumnya.

4.1 Lingkungan Implementasi

Lingkungan implementasi yang dijelaskan pada subbab ini merupakan implementasi dari perangkat keras dan perangkat lunak yang digunakan di dalam perancangan enkripsi algoritma MRC6.

4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan saat proses pengembangan dan aplikasi Enkripsi MRC6 adalah :

1. Prosesor Intel® Atom™ CPU N475 @ 1.8GHz
2. Memory 1,024 GB RAM
3. Harddisk 320 GB

4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam pengembangan program enkripsi dan uji coba enkripsi citra menggunakan algoritma MRC6 adalah:

1. Sistem operasi Windows 7 Ultimate 32-bit.
2. Bahasa Pemrograman Java dengan menggunakan IDE Netbeans 6.8. dengan JDK 1.6.

4.2 Implementasi Program

Berdasarkan analisa dan perancangan proses pada Bab III, maka pada bab berikutnya akan dijelaskan implementasi dalam pembuatan aplikasi. Bab selanjutnya adalah mengenai aplikasi perangkat lunak berikut proses-proses enkripsi dan dekripsi citra digital dengan algoritma MRC6.

BAB V
PENGUJIAN DAN ANALISIS

Sebelum dilakukan pengujian, akan diberikan implementasi program yang meliputi proses pembangkitan kunci, pembentukan blok bit citra, proses enkripsi-dekripsi dan penghitungan entropi-korelasi. Selanjutnya mulai dilakukan pengujian pada citra dan analisis hasil.

5.1 Implementasi Program

5.1.1 Proses Pembangkitan Kunci

Sebelum melakukan proses enkripsi atau dekripsi, perlu dilakukan pembangkitan kunci (key generation) terlebih dahulu. Pengguna diharuskan memasukkan kata kunci (*password)* minimal 4 karakter dan maksimal 32 karakter. Proses *padding* yang dilakukan mengacu pada perancangan proses pembentukan kunci pada subbab 4.2.1 agar ukuran kunci dari pengguna disesuaikan sedemikian hingga menjadi 16, 24, atau 32 byte. Implementasi proses *padding* kunci dapat dilihat pada tabel kodesumber 5.1

```
1   public void padding() {
2                   //padding terhadap panjang kunci
3               if(key.length()<=16){
4                       for (int i = key.length(); i < 16; i++) {
5                               key = key + "0";
6                   }
7               }else if((key.length()> 16)&&(key.length()<= 24)){
8                       for (int i = key.length(); i < 24; i++) {
9                               key = key + "0";
10                  }
11              }
12               else if((key.length()> 24)&&(key.length()<= 32)){
13                      for (int i = key.length(); i < 32; i++) {
14                              key = key + "0";
15                  }
16              }
17          }
```

Kodesumber 5.1 *Padding* Kunci

Sejumlah karakter untuk kunci (key) yang didapatkan dari hasil padding akan dibangkitkan menjadi kunci internal untuk mengisi tabel

kunci S. Untuk mendapatkan tabel kunci S dilakukan antara lain duplikasi kunci, inisialisasi S, dan penggabungan (joining) L dengan S. Serangkaian proses tersebut dapat dilihat pada tabel kodesumber 5.2-5.4

```
1  public void duplikasi() {
2                padding();
3                c = key.length() / 4;
4             if (key.length() % 4 != 0) {
5                c++;
6             }
7             L = new String[c];
8             for (int i = 0; i < L.length; i++) {
9                L[i] = "";
10            }
11            int t = 0;
12            for (int i = 0; i < key.length(); i++) {
13               // get biner setiap char kunci
14               String biner = Integer.toBinaryString((int)
15 key.charAt(i));
16               if (biner.length() < 8) {
17            for (int k = biner.length(); k < 8; k++) {
18                        biner = "0" + biner;
19                  }
20               }
21            System.out.println(key.charAt(i) + " : " + biner);
22               if (L[t].length() == 32) {
23                  t++;
24               }
25               L[t] = L[t] + biner;
26            }
27            // padding apabila L < 32
28            if (L[L.length - 1].length() != 32) {
29 for (int i = L[L.length - 1].length(); i < 32; i++) {
30               L[L.length - 1] = L[L.length - 1] + "0";
31               }
32            }
33            for (int i = 0; i < L.length; i++) {
34               System.out.println("L" + i + " : " + L[i]);
35            }
36       }
```

Kodesumber 5.2 Duplikasi Kunci

```
1  public void inisialisasi() {
2
3                S = new String[8 * r + 16];
4                Lj = new String[8 * r + 16];
5                Lk = new String[8 * r + 16];
6             S[0] = P;
7             for (int i = 1; i < 8 * r + 16; i++) {
8                S[i] = addBiner(S[i - 1], Q);
9             }
10       }
```

Kodesumber 5.3 Inisialisasi Tabel S

```
1   public void joinLS() {
2               int max = 1;
3                   int i= 0;
4               int j = 0;
5               String R1 = "00000000000000000000000000000000";
6               String R2 = "00000000000000000000000000000000";
7
8               for (int l = 0; l < 3; l++) {
9                                   for (int k = 0; k <
10  Math.max(c,8*r+16); k++) {
11                      S[i] = addBiner(addBiner(S[i], R1), R2);
12                            S[i] = Geser3(S[i]);
13                            R1 = S[i];
14                      Lk[j] = addBiner(addBiner(L[j], R1), R2);
15                                    if(i==j){
16                                    L[j]= Geser3(Lk[j]);
17                                    }else{
18                                    Lj[i] =Geser3(Lk[j]);
19                                    }
20                                    R2 = Geser3(Lk[j]);
21                              i = (i + 1) % (8 * r + 16);
22                              j = (j + 1) % c;
23                      }
24              }
25          }
```

Kodesumber 5.4 Penggabungan L dan S

5.1.2 Proses Pembentukan Blok Bit Citra

Pembentukan blok bit citra merupakan langkah yang diperlukan di samping pembentukan kunci. Pada saat proses pembentukan ini langkah awal adalah melakukan pemuatan berkas citra (ditunjukkan pada kodesumber 5.5). Pada proses pembentukan blok citra, akan dilakukan pengecekan apakah berkas citra telah memiliki format yang sesuai.

```
1  Private void
2  OpenMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
3          int returnVal =
4  fileChooser.showOpenDialog(OpenMenuItem);
5          if (returnVal == JFileChooser.APPROVE_OPTION) {
6              berkas = fileChooser.getSelectedFile();
7              try {
8                  BufferedImage image = ImageIO.read(berkas);
9                  fname = "";
10                 floc = "";
11                 ext = "";
12                 int mid = berkas.getName().lastIndexOf(".");
13                 int loc = berkas.getPath().lastIndexOf("\\");
14                 floc = berkas.getPath().substring(0, loc);
15                 System.out.println(floc);
16                 fname = berkas.getName().substring(0, mid);
17                 ext = berkas.getName().substring(mid + 1,
18 berkas.getName().length());
19                 System.out.println(ext + " - " +
20 image.getType());
21                 // mengecek apakah file BMP 24 bit
22                 if (!ext.equalsIgnoreCase("bmp")) {
23         JOptionPane.showMessageDialog(null, "File bukan BMP");
24                 } else {
25                 if (image.getType() != 5) {
26   JOptionPane.showMessageDialog(null, "Filebukan BMP 24 bit");
27                     }else{
28                         //menampilkan image ke frame
29                         ImageFrame.setTitle(berkas.getName());
30                         imageTrue = true;
31                         imageInput = image;
32                         outputPanel.setVisible(false);
33                         new Thread(new thread1()).start();
34 //Start the thread
35                         statusMessageLabel.setText("Memuat
36 Citra");
37
38          displayimage(berkas);
39          statusMessageLabel.setText("Citra Berhasil Dimuat");
40                     }
41                 }
42             }catch (Exception e) {
43                 JOptionPane.showMessageDialog(null, "File
44 Tidak bisa diakses");
45             }
46         } else {
47             System.out.println("Batal membuka file ");
48         }
49 }
```

Kodesumber 5.5 *Function* untuk Memuat Citra

Proses selanjutnya adalah membagi citra menjadi blok-blok bit dan memasukkan nilai biner warna ke dalam indeks tiap-tiap blok yang panjangnya dibatasi 512 bit.

```
1   public void createBlok(BufferedImage image) {
2               //mendapatkan tinggi dan lebar image
3               t = image.getHeight();
4               l = image.getWidth();
5               //mendapatkan jumlah blok yang akan dibentuk
6
7               int nBlok = ((t * l) * 24);
8               if (nBlok % 512 == 0) {
9                   nBlok = nBlok / 512;
10              } else {
11                  nBlok = (nBlok / 512) + 1;
12              }
13
14              System.out.println("Dimensi = " + l + " x " + t);
15              System.out.println("jumlah blok = " + nBlok);
16              //membuat array blok sejumlah nBlok
17              Blok_input = new String[nBlok];
18              //inisialisasi blok input
19              for (int j = 0; j < Blok_input.length; j++) {
20                  Blok_input[j] = "";
21              }
22              int red, green, blue;
23              //insialisasi indeks blok dengan 0
24              int b = 0;
25              //mendapatkan pixel dari image dengan ukuran (t x
26  l)
27              for (int i = 0; i < t; i++) {
28                  for (int j = 0; j < l; j++) {
29                      int clr = image.getRGB(j, i);
30                      red = (clr & 0x00ff0000) >> 16;
31                      green = (clr & 0x0000ff00) >> 8;
32                      blue = clr & 0x000000ff;
33                      //konversi  nilai  red  dari  int  menjadi
34  string biner
35                      String R = Integer.toBinaryString(red);
36                      //tambahkan bit 0 apabila kurang dari 8
37                      if (R.length() < 8) {
38                          for (int k = R.length(); k < 8; k++) {
39                              R = "0" + R;
40                          }
41                      }
42                      //memasukan string biner R ke dalam blok
43  indeks b yang dibatasi 128/512 panjangnya
44                      //if (Blok_input[b].length() < 128)
45                        if (Blok_input[b].length() < 512) {
46                          Blok_input[b] += R; //merah
47                      } else {
48                          Blok_input[b + 1] += R;
49                          b++;
50                      }
51                      //konversi  nilai  green  dari  int  menjadi
52  string biner
53                      String G = Integer.toBinaryString(green);
54                      if (G.length() < 8) {
55                          for (int k = G.length(); k < 8; k++) {
56                              G = "0" + G;
57                          }
58                      }
59
```

```
                    if (Blok_input[b].length() < 512) {
                        Blok_input[b] += G; //hijau
                    } else {
                        Blok_input[b + 1] += G;
                        b++;
                    }
                    //konversi nilai blue dari int menjadi
string biner
                    String B = Integer.toBinaryString(blue);
                    if (B.length() < 8) {
                        for (int k = B.length(); k < 8; k++) {
                            B = "0" + B;
                        }
                    }
                    //memasukan string biner B ke dalam blok
indeks b
                    if (Blok_input[b].length() < 512) {
                        Blok_input[b] += B; //biru
                    } else {
                        Blok_input[b + 1] += B;
                        b++;
                    }
                }
            }
            //memasukan string biner 0 ke dalam blok indeks b
agar panjangnya 512
            for (int j = 0; j < Blok_input.length; j++) {
                if (Blok_input[j].length() < 512) {
                    for (int i = Blok_input[j].length(); i <
512; i++) {
                        Blok_input[j] = Blok_input[j] + "0";
                    }
                }
            }
```

Kodesumber 5.6 Pembuatan Blok-blok Bit Citra

5.1.3 Proses Enkripsi

Setelah dilakukan pembentukan terhadap tabel kunci S dan juga blok-blok citra. Langkah selanjutnya adalah melakukan proses enkripsi. Tabel kunci S didapatkan dari hasil pembentukan kunci sedangkan blok *plainimage* berasal dari proses pembentukan blok-blok bit citra. Proses enkripsi blok *plainimage* dapat dilihat pada kodesumber 5.7.

```
1   public class Enkripsi {
2           String R1 = "";
3           String R2 = "";
4           String R3 = "";
5           String R4 = "";
6           String R5 = "";
7           String R6 = "";
8           String R7 = "";
9           String R8 = "";
10          String R9 = "";
11          String R10 = "";
12          String R11 = "";
13          String R12 = "";
14          String R13 = "";
15          String R14 = "";
16          String R15 = "";
17          String R16 = "";
18          //String kunci = "enkripsi";
19          int r;
20          public static String blok[];
21          public static String sAkhir[];
22          public String bloknew[];
23          int Y;
24          //String key1;
25          public Enkripsi(int r,String sAkhir[],String blok[]){
26                      this.r=r;
27                      this.sAkhir = sAkhir;
28                      this.blok=blok;
29                      //enkrip();
30                     System.out.println("");
31          }
```

Kodesumber 5.7 Enkripsi Blok Plainimage

Implementasi enkripsi dengan algoritma Modified Rivest Code 6 dilakukan dengan cara membagi tiap-tiap blok ke dalam 16 buah *register*, yaitu R1, R2, R3, ... ,R16 dan dilakukan *whitening* awal, putaran (round) sebanyak r, dan *whitening* akhir sehingga mendapatkan bloknew[] yang berisi blok hasil enkripsi.

```
1   public String[] enkrip(){
2                       String[] bloknew=new String[blok.length];
3
4               //ENKRIPSI semua blok
5               for (int c = 0; c < blok.length; c++) {
6
7                   //padding apabila blok  kurang dari 512 bit
8                   if (blok[c].length() < 512) {
9                       for (int j = blok[c].length(); j < 512;
10  j++) {
11                          blok[c] += "0";
12                      }
13                  }
14
15                      R1 = blok[c].substring(0, 32);
```

```
                    R2 = blok[c].substring(32, 64);
                    R3 = blok[c].substring(64, 96);
                    R4 = blok[c].substring(96, 128);
                    R5 = blok[c].substring(128, 160);
                    R6 = blok[c].substring(160, 192);
                    R7 = blok[c].substring(192, 224);
                    R8 = blok[c].substring(224, 256);
                    R9 = blok[c].substring(256, 288);
                    R10 = blok[c].substring(288, 320);
                    R11 = blok[c].substring(320, 352);
                    R12 = blok[c].substring(352, 384);
                    R13 = blok[c].substring(384, 416);
                    R14 = blok[c].substring(416, 448);
                    R15 = blok[c].substring(448, 480);
                    R16 = blok[c].substring(480, 512);

                    R2= addBiner(R2, sAkhir[0]);
                    R4= addBiner(R4, sAkhir[1]);
                    R6= addBiner(R6, sAkhir[2]);
                    R8= addBiner(R8, sAkhir[3]);
                    R10= addBiner(R10, sAkhir[4]);
                    R12= addBiner(R12, sAkhir[5]);
                    R14= addBiner(R14, sAkhir[6]);
                    R16= addBiner(R16, sAkhir[7]);

                    String k = "";
                    String l = "";
                    String m = "";
                    String n = "";
                    String t = "";
                    String u = "";
                    String v = "";
                    String z = "";
                    String R1x = "";


                    //putaran sebanyak r
                    for(int i=1; i<r+1; i++){


                        k =
Geser5(crossBiner(R2,addBiner(addBiner(R2,
R2),"00000000000000000000000000000001")));
                        l =
Geser5(crossBiner(R4,addBiner(addBiner(R4,
R4),"00000000000000000000000000000001")));
                        m =
Geser5(crossBiner(R6,addBiner(addBiner(R6,
R6),"00000000000000000000000000000001")));
                        n =
Geser5(crossBiner(R8,addBiner(addBiner(R8,
R8),"00000000000000000000000000000001")));
                        t =
Geser5(crossBiner(R10,addBiner(addBiner(R10,
R10),"00000000000000000000000000000001")));
                        u =
Geser5(crossBiner(R12,addBiner(addBiner(R12,
R12),"00000000000000000000000000000001")));
                        v =
```

```
Geser5(crossBiner(R14,addBiner(addBiner(R14,
R14),"00000000000000000000000000000001")));
                                z =
Geser5(crossBiner(R16,addBiner(addBiner(R16,
R16),"00000000000000000000000000000001")));

                                R1 = addBiner(Geser5(XOR(R1,
k)),sAkhir[8*i]);
                                R3 = addBiner(Geser5(XOR(R3,
l)),sAkhir[8*i+1]);
                                R5 = addBiner(Geser5(XOR(R5,
m)),sAkhir[8*i+2]);
                                R7 = addBiner(Geser5(XOR(R7,
n)),sAkhir[8*i+3]);
                                R9 = addBiner(Geser5(XOR(R9,
t)),sAkhir[8*i+4]);
                                R11 = addBiner(Geser5(XOR(R11,
u)),sAkhir[8*i+5]);
                                R13 = addBiner(Geser5(XOR(R13,
v)),sAkhir[8*i+6]);
                                R15 = addBiner(Geser5(XOR(R5,
z)),sAkhir[8*i+7]);


                                R1x=R1;
                                R1 = R2;
                                R2 = R3;
                                R3 = R4;
                                R4 = R5;
                                R5 = R6;
                                R6 = R7;
                                R7 = R8;
                                R8 = R9;
                                R9 = R10;
                                R10 = R11;
                                R11 = R12;
                                R12 = R13;
                                R13 = R14;
                                R14 = R15;
                                R15 = R16;
                                R16 = R1x;



                        R1= addBiner (R1, sAkhir[8*r+8]);
                        R3= addBiner (R3, sAkhir[8*r+9]);
                        R5= addBiner (R5, sAkhir[8*r+10]);
                        R7= addBiner (R7, sAkhir[8*r+11]);
                        R9= addBiner (R9, sAkhir[8*r+12]);
                        R11= addBiner (R11, sAkhir[8*r+13]);
                        R13= addBiner (R13, sAkhir[8*r+14]);
                        R15= addBiner (R15, sAkhir[8*r+15]);


                        bloknew[c] = R1 + R2 + R3 + R4 + R5 + R6
+ R7 + R8 + R9 + R10 + R11 + R12 + R13 + R14 + R15 + R16;
                    }
```

```
135             return bloknew;
136         }
137
```

Kodesumber 5.8 Proses Enkripsi Modified Rivest Code 6

5.1.4 Proses Dekripsi

Sama halnya dengan proses enkripsi, pada proses ini mengambil masukan dari proses pembentukan kunci dan blok-blok citra dari *chiperimage*. Implementasi dekripsi blok *cipherimage* dapat dilihat pada kodesumber 5.9.

```
1   public class Dekripsi {
2
3       String R1 = "";
4       String R2 = "";
5       String R3 = "";
6       String R4 = "";
7       String R5 = "";
8       String R6 = "";
9       String R7 = "";
10      String R8 = "";
11      String R9 = "";
12      String R10 = "";
13      String R11 = "";
14      String R12 = "";
15      String R13 = "";
16      String R14 = "";
17      String R15 = "";
18      String R16 = "";
19
20          int r;
21          public static String blok[];
22          public static String sAkhir[];
23          public String bloknew[];
24          int Y;
25          public Dekripsi(int r,String sAkhir[],String blok[]){
26                      this.r=r;
27                      this.sAkhir = sAkhir;
28                      this.blok=blok;
29                      System.out.println("");
30          }
```

Kodesumber 5.9 Dekripsi *Plainimage*

Untuk proses penghitungan dekripsi, langkah pengkodeannya merupakan kebalikan dari proses penghitungan enkripsi.

```
1   public String[] dekrip(){
2               String[] bloknew=new String[blok.length];
3               //DESKRIPSI  semua blok
4               for (int c = 0; c < blok.length; c++) {
5                   //padding apabila blok  kurang dari 512 bit
6                   if (blok[c].length() < 512) {
```

```
7                       for (int j = blok[c].length(); j < 512;
8   j++) {
9                           blok[c] += "0";
10                      }
11                  }
12
13                  R1 = blok[c] .substring(0, 32);
14                  R2 = blok[c] .substring(32, 64);
15                  R3 = blok[c] .substring(64, 96);
16                  R4 = blok[c] .substring(96, 128);
17                  R5 = blok[c] .substring(128, 160);
18                  R6 = blok[c] .substring(160, 192);
19                  R7 = blok[c] .substring(192, 224);
20                  R8 = blok[c] .substring(224, 256);
21                  R9 = blok[c] .substring(256, 288);
22                  R10 = blok[c] .substring(288, 320);
23                  R11 = blok[c] .substring(320, 352);
24                  R12 = blok[c] .substring(352, 384);
25                  R13 = blok[c] .substring(384, 416);
26                  R14 = blok[c] .substring(416, 448);
27                  R15 = blok[c] .substring(448, 480);
28                  R16 = blok[c] .substring(480, 512);
29
30
31                  R15= minusBiner(R15, sAkhir[8*r+15]);
32                  R13= minusBiner(R13, sAkhir[8*r+14]);
33                  R11= minusBiner(R11, sAkhir[8*r+13]);
34                  R9= minusBiner(R9, sAkhir[8*r+12]);
35                  R7= minusBiner(R7, sAkhir[8*r+11]);
36                  R5= minusBiner(R5, sAkhir[8*r+10]);
37                  R3= minusBiner(R3, sAkhir[8*r+9]);
38                  R1= minusBiner(R1, sAkhir[8*r+8]);
39
40                      String k = "";
41                      String l = "";
42                      String m = "";
43                      String n = "";
44                      String t = "";
45                      String u = "";
46                      String v = "";
47                      String z = "";
48                      String R1x = "";
49                      String R2x = "";
50                      String R3x = "";
51                      String R4x = "";
52                      String R5x = "";
53                      String R6x = "";
54                      String R7x = "";
55                      String R8x = "";
56                      String R9x = "";
57                      String R10x = "";
58                      String R11x = "";
59                      String R12x = "";
60                      String R13x = "";
61                      String R14x = "";
62                      String R15x = "";
63
64                      //putaran sebanyak r
65                      for(int i=r; i>0; i--){
```

```
66
67                          R1x=R1;
68                          R2x=R2;
69                          R3x=R3;
70                          R4x=R4;
71                          R5x=R5;
72                          R6x=R6;
73                          R7x=R7;
74                          R8x=R8;
75                          R9x=R9;
76                          R10x=R10;
77                          R11x=R11;
78                          R12x=R12;
79                          R13x=R13;
80                          R14x=R14;
81                          R15x=R15;
82                          R1=R16;
83                          R2=R1x;
84                          R3=R2x;
85                          R4=R3x;
86                          R5=R4x;
87                          R6=R5x;
88                          R7=R6x;
89                          R8=R7x;
90                          R9=R8x;
91                          R10=R9x;
92                          R11=R10x;
93                          R12=R11x;
94                          R13=R12x;
95                          R14=R13x;
96                          R15=R14x;
97                          R16=R15x;
98
99                          z =
100 gKiri(crossBiner(R16,addBiner(addBiner(R16,
101 R16),"00000000000000000000000000000001")));
102                         v =
103 gKiri(crossBiner(R14,addBiner(addBiner(R14,
104 R14),"00000000000000000000000000000001")));
105                         u =
106 gKiri(crossBiner(R12,addBiner(addBiner(R12,
107 R12),"00000000000000000000000000000001")));
108                         t =
109 gKiri(crossBiner(R10,addBiner(addBiner(R10,
110 R10),"00000000000000000000000000000001")));
111                         n =
112 gKiri(crossBiner(R8,addBiner(addBiner(R8,
113 R8),"00000000000000000000000000000001")));
114                         m =
115 gKiri(crossBiner(R6,addBiner(addBiner(R6,
116 R6),"00000000000000000000000000000001")));
117                         l =
118 gKiri(crossBiner(R4,addBiner(addBiner(R4,
119 R4),"00000000000000000000000000000001")));
120                         k =
121 gKiri(crossBiner(R2,addBiner(addBiner(R2,
122 R2),"00000000000000000000000000000001")));
123                         R15 =
124 XOR(gKanan(minusBiner(R15,sAkhir[8*i+7])),z);
```

```
                        R13 =
XOR(gKanan(minusBiner(R13,sAkhir[8*i+6])),v);
                        R11 =
XOR(gKanan(minusBiner(R11,sAkhir[8*i+5])),u);
                        R9 =
XOR(gKanan(minusBiner(R9,sAkhir[8*i+4])),t);
                        R7 =
XOR(gKanan(minusBiner(R7,sAkhir[8*i+3])),n);
                        R5 =
XOR(gKanan(minusBiner(R5,sAkhir[8*i+2])),m);
                        R3 =
XOR(gKanan(minusBiner(R3,sAkhir[8*i+1])),l);
                        R1 =
XOR(gKanan(minusBiner(R1,sAkhir[8*i])),k);
                    }

                    R16= minusBiner(R16, sAkhir[7]);
                    R14= minusBiner(R14, sAkhir[6]);
                    R12= minusBiner(R12, sAkhir[5]);
                    R10= minusBiner(R10, sAkhir[4]);
                    R8= minusBiner(R8, sAkhir[3]);
                    R6= minusBiner(R6, sAkhir[2]);
                    R4= minusBiner(R4, sAkhir[1]);
                    R2= minusBiner(R2, sAkhir[0]);
            //ganti blok dengan string hasil dari enkripsi

                    bloknew[c] = bloknew[c] = R1 + R2 + R3 +
R4 + R5 + R6 + R7 + R8 + R9 + R10 + R11 + R12 + R13 + R14 +
R15 + R16;
            }
            return bloknew;
        }
```

Kodesumber 5.10 Proses Dekripsi MRC6

5.1.5 Proses Perhitungan Korelasi dan *Entropy*

Nilai korelasi adalah sebuah variabel yang digunakan untuk mengetahui derajat tingkat kesamaan dari dua buah piksel yang saling berdempetan pada suatu citra.

```
public void hitKorelasi(){
            //hitung jumlah data
            int n = (l*t);
            int[] Xr= new int[n];
            int[] Xg= new int[n];
            int[] Xb= new int[n];
                //array untuk menampung RGB tetangga
            int[] Yr_h = new int[n];
            int[] Yg_h = new int[n];
            int[] Yb_h = new int[n];
            int[] Yr_v = new int[n];
            int[] Yg_v = new int[n];
            int[] Yb_v = new int[n];
            int[] Yr_d = new int[n];
            int[] Yg_d = new int[n];
            int[] Yb_d = new int[n];
```

```
17
18  //          Random randomGenerator_T = new Random();
19  //          Random randomGenerator_L = new Random();
20              int k = 0;
21              for (int i = 0; i < t; i++) {
22                    for (int j = 0; j < l; j++) {
23                          int a,b;
24                          //PIXEL AWAL
25                          Xr[k] = pxR[j][i];
26                          Xg[k] = pxG[j][i];
27                          Xb[k] = pxB[j][i];
28
29                          //TETANGGA HORIZONTAL
30                          if(j+1 == l){
31                                a = j-1;
32                          }else{
33                                a = j+1;
34                          }
35                          Yr_h[k] = pxR[a][i];
36                          Yg_h[k] = pxG[a][i];
37                          Yb_h[k] = pxB[a][i];
38                          //TETANGGA VERTIKAL
39                          if(i+1 == t){
40                                b = i-1;
41                          }else{
42                                b = i+1;
43                          }
44                          Yr_v[k] = pxR[j][b];
45                          Yg_v[k] = pxG[j][b];
46                          Yb_v[k] = pxB[j][b];
47
48                          //TETANGGA DIAGONAL_1
49                          Yr_d[k] = pxR[a][b];
50                          Yg_d[k] = pxG[a][b];
51                          Yb_d[k] = pxB[a][b];
52                                //System.out.print("Xr=
53  ["+j+"]["+i+"]"+"Yr_h= ["+a+"]["+b+"]\n");
54                          k++;
55                    }
56              }
57
58              rh  = (Rxy(Xr, Yr_h)+Rxy(Xg, Yg_h)+Rxy(Xb,
59  Yb_h))/3;
60              rv  = (Rxy(Xr, Yr_v)+Rxy(Xg, Yg_v)+Rxy(Xb,
61  Yb_v))/3;
62              rd  = (Rxy(Xr, Yr_d)+Rxy(Xg, Yg_d)+Rxy(Xb,
63  Yb_d))/3;
64              r   = (rh+rv+rd)/3;
65
66              NumberFormat formatter = new
67  DecimalFormat("#0.00000000");
68              System.out.println("Korelasi Pixel Horizontal :
69  "+formatter.format(rh));
70              System.out.println("Korelasi Pixel Vertikal   :
71  "+formatter.format(rv));
72              System.out.println("Korelasi Pixel Diagonal   :
73  "+formatter.format(rd));
74              System.out.println("Rata-Rata Korelasi        :
75  "+formatter.format(r));
```

```
76
77              //System.out.println("rg : "+Rxy(Xg, Yg_h));
78              //System.out.println("rb : "+Rxy(Xb, Yb_h));
79          }
80
```

Kodesumber 5.11 Penghitungan Nilai Korelasi

Nilai entropi digunakan untuk mengukur variasi intensitas warna dari sebuah citra. Proses perhitungan nilai *entropy* dapat dilihat pada kodesumber 5.12:

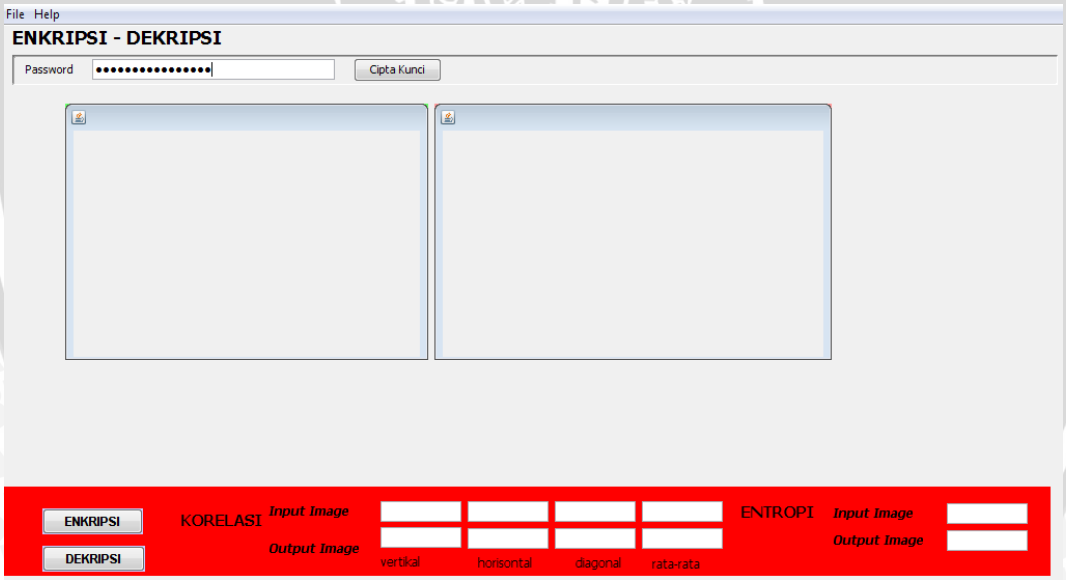
```
1   public entropi_korelasi(BufferedImage cipgerimage) throws
2   IOException {
3               this.CipherImg = cipgerimage;
4
5               t = CipherImg.getHeight();
6               l = CipherImg.getWidth();
7               pxR= new int[l][t];
8               pxG= new int[l][t];
9               pxB= new int[l][t];
10                  //array untuk menampung jumlah peluang RGB
11              PR = new double[256];
12              PG = new double[256];
13              PB = new double[256];
14
15              for (int i = 0; i < PR.length; i++) {
16                    PR[i] = PG[i] = PB[i] = 0;
17              }
18              for (int i = 0; i < t; i++) {
19                    for (int j = 0; j < l; j++) {
20                          int clr = CipherImg.getRGB(j, i);
21                          R = (clr & 0x00ff0000) >> 16;
22                          G = (clr & 0x0000ff00) >> 8;
23                          B =  clr & 0x000000ff;
24
25                          pxR[j][i] = R;
26                          pxG[j][i] = G;
27                          pxB[j][i] = B;
28
29                          PR[R] += 1;
30                          PG[G] += 1;
31                          PB[B] += 1;
32                                  //System.out.print("PxR=
33  "+pxR[j][i]);
34                                  //System.out.print("PxG=
35  "+pxG[j][i]);
36                                  //System.out.print("PxB=
37  "+pxB[j][i]);
38
39                    }
40              }
41              //cari peluang tiap pixel
42              for (int i = 0; i < PR.length; i++) {
43                    PR[i] = PR[i]/(t*l);
44                    PG[i] = PG[i]/(t*l);
45                    PB[i] = PB[i]/(t*l);
46              }
```

```
47              //hit Entropy
48              pR=pG=pB = 0;
49              for (int i = 0; i < PR.length; i++) {
50                  if(PR[i]!=0){
51                      pR = pR +(PR[i]*log2(PR[i]));
52                  }
53                  if(PG[i]!=0){
54                      pG = pG +(PG[i]*log2(PG[i]));
55                  }
56                  if(PB[i]!=0){
57                      pB = pB +(PB[i]*log2(PB[i]));
58                  }
59              }
60              H = -(pR+pG+pB)/3;
61              hitKorelasi();
62          }
63
```

Kodesumber 5.12 Penghitungan Nilai Entropy

5.2 Implementasi *Interface*

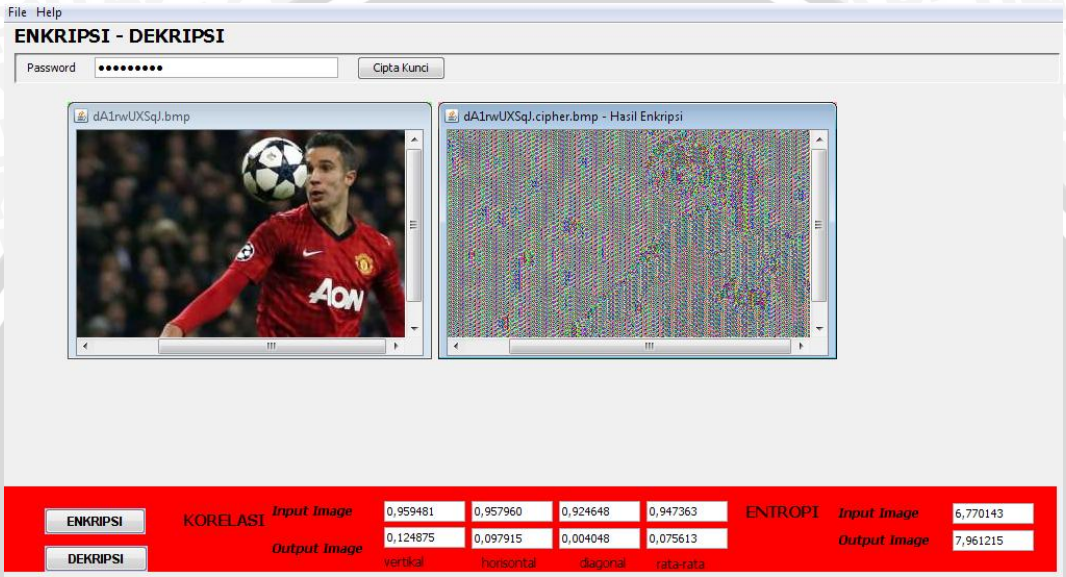
Implementasi antar muka pada aplikasi enkripsi citra menggunakan algoritma Modified Rivest Code 6 dapat dilihat pada gambar 5.1.



Gambar 5.1 Interface

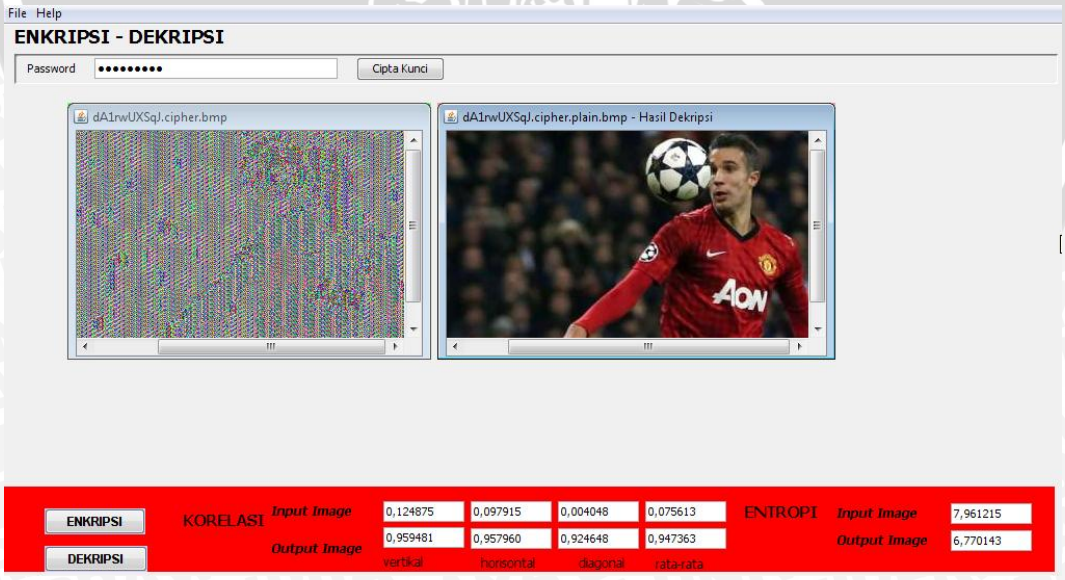
Sebelum pengguna melakukan proses enkripsi, langkah pertama yang harus dilakukan adalah membuka file citra terlebih dahulu. File citra dapat dibuka dengan cara mengklik menu bar File → open image. Langkah kedua adalah memasukkan kunci sebagai pengamanan citra. Untuk melakukan langkah kedua ini, pengguna dapat mengklik tombol 'Cipta Kunci' setelah sebelumnya terlebih dahulu teks yang akan digunakan sebagai kunci pada kolom password.

Setelah semuanya selesai dilakukan, maka pengguna dapat mengklik tombol ‘Enkripsi’ yang terletak di bagian bawah interface untuk mulai melakukan proses enkripsi. File citra hasil enkripsi merupakan file cipher (*cipher image)*. Untuk membuka file cipher tersebut, pengguna dapat membuka file cipher dengan cara yang sama seperti membuka file citra untuk pertama kali lalu mengklik tombol ‘dekripsi’. Interface proses enkripsi dapat dilihat pada gambar 5.2.



Gambar 5.2 Proses Enkripsi

Interface proses dekripsi dapat dilihat pada gambar 6.3

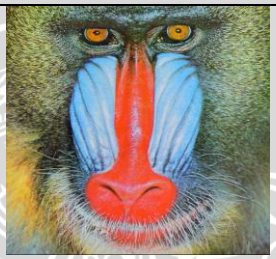


Gambar 5.3 Proses Dekripsi

5.3 Implementasi dan Pembahasan Uji Coba

Uji coba terhadap citra merupakan pengujian terhadap keberhasilan proses pengenkripsian dan pendekripsian citra. Sesuai dengan batasan masalah, citra yang digunakan adalah citra berformat *bitmap* (.bmp) dengan kedalaman warna 24-bit. Tabel 5.1 memperlihatkan daftar citra yang digunakan di dalam pengujian.

Tabel 5.1 Daftar Citra Uji

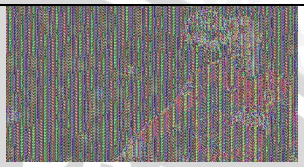
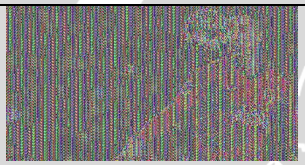
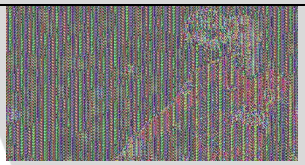
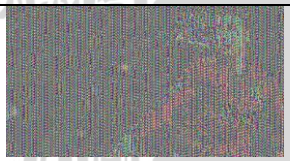
No	Nama File	Citra	Resolusi
1.	dA1rwUXSqJ.bmp		400 x 220 piksel
2.	BaboonRGB.bmp		250x250 dan 512x512
3.	Boy.bmp		250x250 dan 512x512
4.	PeppersRGB.bmp		250x250 dan 512x512

5.4 Hasil dan Pembahasan Uji Fungsionalitas

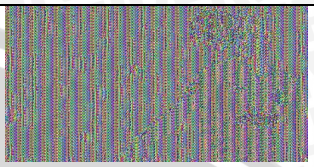
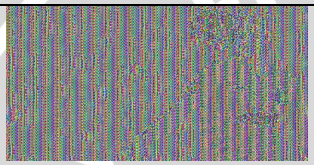
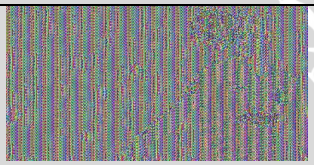
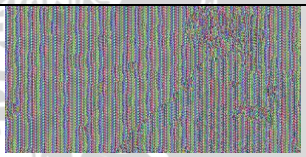
Pengujian yang dilakukan adalah terkait dengan fungsi keamanan citra yang disediakan oleh aplikasi. Citra yang telah diamankan dan telah berwujud *cipherimage* akan diuji dengan cara didekripsi menggunakan subkey yang salah (berbeda dengan subkey pada saat pengenkripsian). Apabila tidak berhasil

membuka (dekripsi) menjadi citra semula, berarti citra telah dapat diamankan dengan sempurna.

Tabel 5.2 Hasil Pengujian Fungsionalitas Proses Enkripsi-Dekripsi

Pengujian Fungsionalitas 1	
Proses Enkripsi (password: **encrypts)**	
Plainimage	*Cipherimage*
	
Hasil : Berhasil, citra yang dihasilkan tidak memiliki makna	
Proses Dekripsi (password: **encrypts**)	
Cipherimage	*Decrypted Cipherimage*
	
Hasil : Berhasil, menghasilkan citra yang asli	
Proses Dekripsi (password: **dekripsi)**	
Cipherimage	*Decrypted Cipherimage*
	
Hasil : Berhasil, citra yang didekripsi menggunakan password yang salah tetap tidak memiliki makna. Citra tetap aman.	

Tabel 5.3. Hasil Pengujian 2 Fungsionalitas Proses Enkripsi-Dekripsi

<table>
<tr><td colspan="2">Pengujian Fungsionalitas 2</td></tr>
<tr><td colspan="2">Proses Enkripsi (password: enkripsidenganmenggunakanalgmrc6)</td></tr>
<tr><td>Plainimage</td><td>Cipherimage</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td colspan="2">Hasil : Berhasil, citra yang dihasilkan tidak memiliki makna</td></tr>
<tr><td colspan="2">Proses Dekripsi (password: enkripsidenganmenggunakanalgmrc6)</td></tr>
<tr><td>Cipherimage</td><td>Decrypted Cipherimage</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td colspan="2">Hasil : Berhasil, menghasilkan citra yang asli</td></tr>
<tr><td colspan="2">Proses Dekripsi (password: enkripsidenganmenggunakanmrc6)</td></tr>
<tr><td>Cipherimage</td><td>Decrypted Cipherimage</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td colspan="2">Hasil : Berhasil, citra yang didekripsi menggunakan password yang salah tetap tidak memiliki makna. Citra tetap aman.</td></tr>
</table>

Kedua tabel di atas, yakni Tabel 5.2 dan Tabel 5.3 membandingkan fungsionalitas perangkat lunak dalam mengamankan citra. Dari informasi pada kedua tabel di atas, untuk pemberian *cipherimage* dengan password yang sama dengan password enkripsi pada saat akan mendekripsi citra, maka citra akan terdekripsi menjadi seperti keadaan semula sebelum dienkripsi. Pada saat *cipherimage* diberikan password yang berbeda dengan password awal pada saat citra dienkripsi, maka citra tidak akan terdekripsi sehingga keamanan citra terjamin.

Untuk dapat mengembalikan suatu citra dari bentuk *cipherimage* ke bentuk citra aslinya diperlukan kata kunci yang sama persis dengan kata kunci yang digunakan untuk proses enkripsi. Apabila berbeda satu karakter saja, maka citra sudah tidak akan dapat didekripsi dengan sempurna. Di dalam proses dekripsi *cipherimage*, adakalanya citra tetap tidak kembali ke wujud aslinya namun masih bisa dilihat struktur yang membentuk dasar wujud yang ditampilkan dalam suatu citra aslinya. Ini karena algoritma enkripsi hanya mengacak bit warna tanpa mengubah/ menulis ulang letak piksel pada citra asli.

5.5 Hasil dan Pembahasan Uji *Cipherimage*

Pengujian ini memeriksa hubungan antara panjang password (*password length)* dengan tingkat keberhasilan penghilangan makna gambar yang dienkripsi yang diukur dari nilai korelasi dan *entropy*. Kriteria pengujian diberikan oleh Tabel 5.4.

Tabel 5.4 Kriteria Pengujian

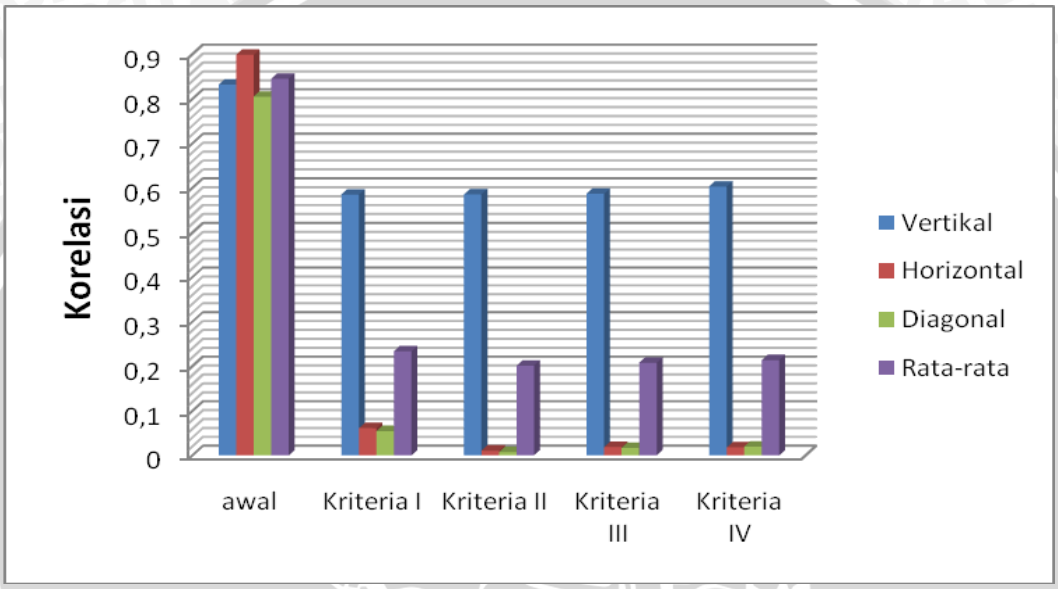
No	Panjang Karakter	Key
1.	8 byte	abcdefgh
2.	16 byte	abcdefghijklmnop
3.	24 byte	abcdefghijklmnopqrstuvwx
4.	32 byte	abcdefghijklmnopqrstuvwxyz123456

Nilai korelasi bertujuan untuk mengukur korelasi antara dua buah piksel yang saling bersinggungan. Jika nilai korelasi memiliki nilai yang hampir sama ataupun sama persis, itu artinya kedua piksel yang dibandingkan masih memiliki ketetanggaan yang relatif dekat. Jika nilai korelasi semakin mendekati nol itu artinya antara *cipherimage* dan citra asli benar-benar memiliki perbedaan. Piksel-piksel yang bersinggungan tidak memiliki kaitan yang signifikan sehingga disimpulkan bahwa pengacauan (enkripsi) citra berhasil. Hasil pengujian ditunjukkan pada Tabel 5.5 – 5.10.

Tabel 5.5 Hasil pengujian *cipherimage* BaboonRGB.bmp 512 x 512

Uji-1		BaboonRGB.bmp	Kriteria			
			I	II	III	IV
Korelasi	Vertikal	0,832241	0,584756	0,585610	0,586861	0,603167
	Horizontal	0,898592	0,061059	0,010604	0,018678	0,017987
	Diagonal	0,804692	0,054502	0,007422	0,016810	0,019820
	Rata-rata	0,845175	0,233439	0,201212	0,207450	0,213658
Entropy		7,644440	7,996967	7,995093	7,992061	7,997190

Berikut adalah grafik korelasi dari nilai korelasi yang ditunjukkan pada Tabel 5.5



Gambar 5.4 Grafik Korelasi Citra BaboonRGB.bmp 512 x 512

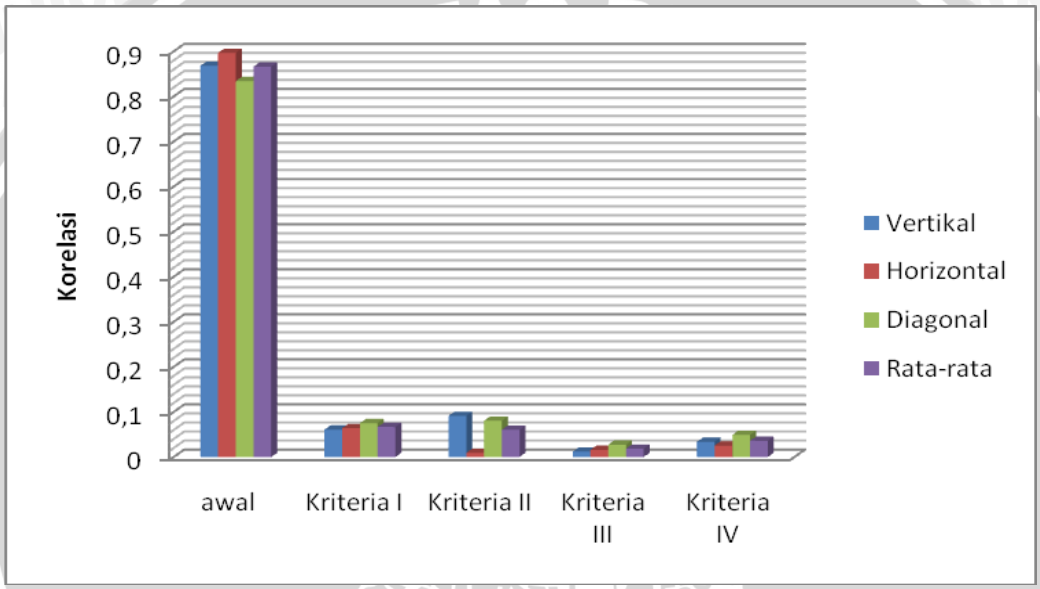
Berdasarkan data yang tercantum oleh grafik pada gambar 5.4, nilai korelasi antara *cipherimage* dalam pengujian kriteria I hingga IV mengalami penurunan yang signifikan terhadap citra aslinya, terutama pada piksel horizontal dan diagonal.. Korelasi citra relatif menunjukkan arah semakin turun seiring urutan kriteria pengujian. Hal ini menunjukkan bahwa panjang kunci yang diberikan berpengaruh terhadap korelasi citra.

Uji *cipherimage* BaboonRGB.bmp dengan resolusi 250 x 250 piksel didapatkan hasil berikut:

Tabel 5.6 Hasil pengujian *cipherimage* BaboonRGB.bmp 250 x 250

Uji-2		BaboonRGB.bmp	Kriteria			
			I	II	III	IV
Korelasi	Vertikal	0,867950	0,060815	0,091329	0,011744	0,033631
	Horizontal	0,896561	0,064010	0,009157	0,015937	0,025228
	Diagonal	0,833974	0,075179	0,080454	0,027333	0,049040
	Rata-rata	0,866162	0,066668	0,060313	0,018338	0,035966
Entropy		7,559831	7,994379	7,991768	7,988608	7,994532

Berikut adalah grafik korelasi dari nilai korelasi yang ditunjukkan pada Tabel 5.6



Gambar 5.5 Grafik Korelasi Citra BaboonRGB.bmp 250 x 250

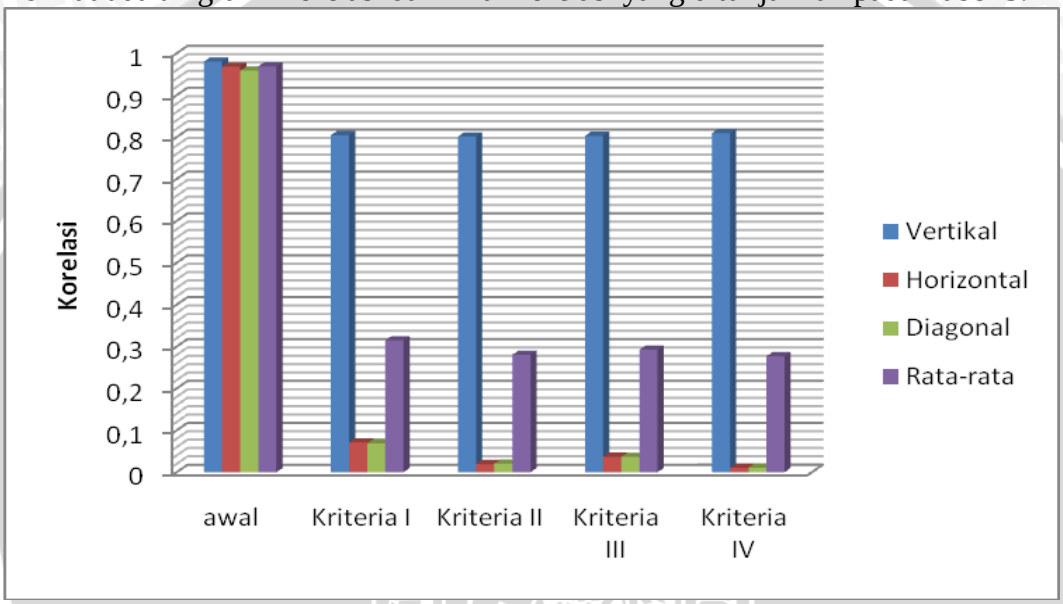
Berdasarkan grafik yang ditunjukkan pada gambar 5.5, maka dapat dilihat penurunan nilai korelasi pada kriteria I hingga IV. Nilai korelasi piksel yang bertetangga pada tiap-tiap kriteria mengalami penurunan yang serentak antara vertikal, horisontal dan diagonal. Semakin panjang kunci, maka dapat dilihat rata-rata nilai semakin rendah.

Uji *cipherimage* Boy.bmp dengan resolusi 512 x 512 piksel didapatkan hasil berikut:

Tabel 5.7 Hasil pengujian *cipherimage* Boy.bmp 512 x 512

Uji-3		Boy.bmp	Kriteria			
			I	II	III	IV
Korelasi	Vertikal	0,980263	0,804455	0,801222	0,802912	0,809383
	Horizontal	0,967875	0,070812	0,019426	0,037028	0,010311
	Diagonal	0,958580	0,069122	0,020382	0,036724	0,010767
	Rata-rata	0,968906	0,314797	0,280343	0,292221	0,276820
Entropy		7,488955	7,991356	7,989442	7,981721	7,993907

Berikut adalah grafik korelasi dari nilai korelasi yang ditunjukkan pada Tabel 5.7



Gambar 5.6 Grafik Korelasi Citra Boy.bmp 512 x 512

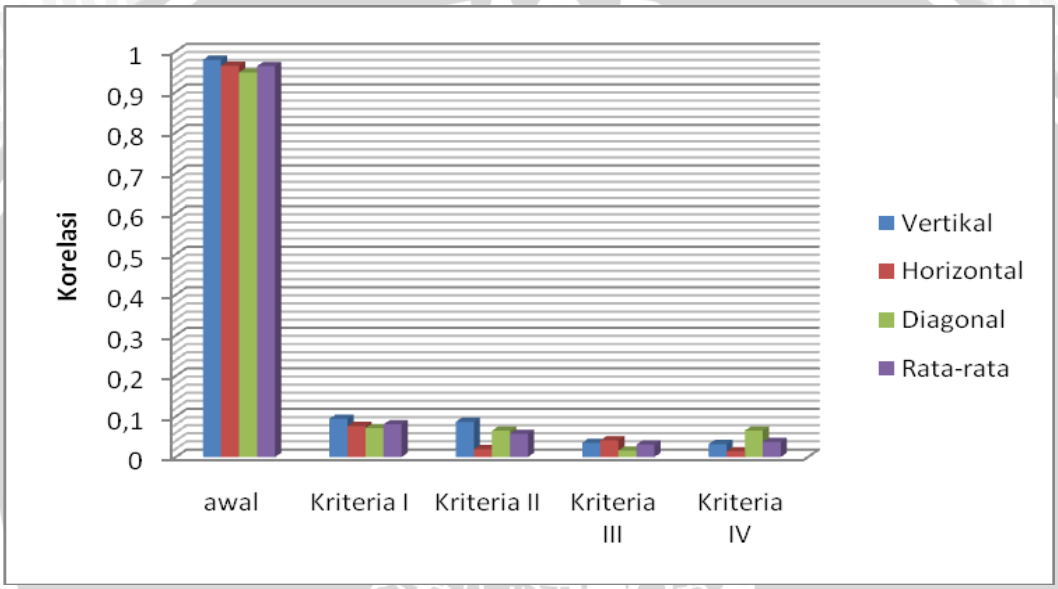
Berdasarkan grafik yang ditunjukkan pada gambar 5.6, nilai korelasi citra dalam kritera pengujian I hingga IV terlihat mengalami penurunan apabila dibandingkan dengan citra aslinya. Pada semua kriteria pengujian nilai korelasi piksel bertetangga vertikal mengalami relatif sedikit penurunan dibanding citra asli, namun nilai horisontal dan diagonal mengalami penurunan yang signifikan. Semakin panjang kunci, maka nilai korelasi akan semakin turun. Pada Kriteria IV gambar Boy.bmp yang diuji ini, nilai korelasi diagonal, horisontal dan rata-rata menempati tempat paling rendah di antara semua pengujian.

Uji *cipherimage* Boy.bmp dengan resolusi 250 x 250 piksel didapatkan hasil berikut:

Tabel 5.8 Hasil pengujian *cipherimage* Boy.bmp 250 x 250

Uji-4		Boy.bmp	Kriteria			
			I	II	III	IV
Korelasi	Vertikal	0,979190	0,094023	0,086606	0,034852	0,031794
	Horizontal	0,964180	0,076327	0,019135	0,040787	0,013869
	Diagonal	0,947788	0,071198	0,064724	0,015789	0,064970
	Rata-rata	0,963719	0,080516	0,056822	0,030476	0,036878
Entropy		7,475765	7,990513	7,989090	7,979325	7,993578

Berikut adalah grafik korelasi dari nilai korelasi yang ditunjukkan pada Tabel 5.8



Gambar 5.7 Grafik Korelasi Citra Boy.bmp 250 x 250

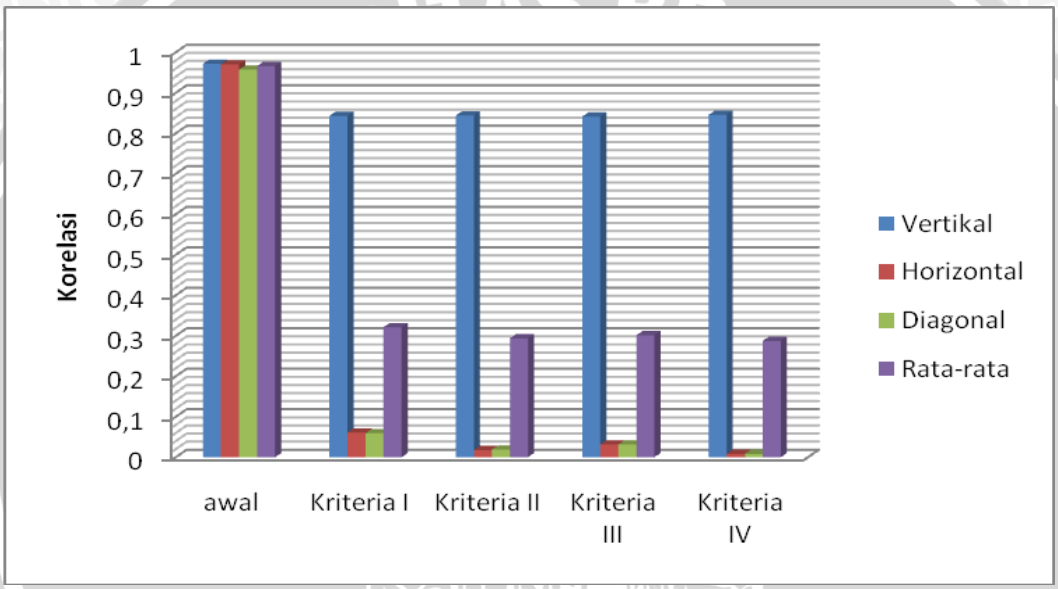
Berdasarkan grafik yang ditunjukkan pada gambar 5.7, nilai korelasi citra dalam kritera pengujian I hingga IV terlihat mengalami penurunan apabila dibandingkan dengan citra aslinya. Pada kriteria I digunakan kata kunci 8 byte yang menghasilkan nilai korelasi menjauhi citra awal. Pada kriteria II, digunakan kata kunci 16 byte, menghasilkan nilai horisontal dan diagonal yang lebih kecil lagi daripada kriteria I. Kriteria III menggunakan kata kunci dengan panjang 24 byte, menghasilkan nilai vertikal dan diagonal yang lebih rendah daripada kriteria II. Kriteria IV menggunakan kata kunci 32 byte, menghasilkan nilai vertikal dan horisontal yang lebih rendah daripada kriteria III. Semakin panjang kunci yang digunakan, maka nilai korelasi yang dihasilkan akan semakin rendah pula.

Uji *cipherimage* PeppersRGB.bmp dengan resolusi 512 x 512 piksel didapatkan hasil berikut:

Tabel 5.9 Hasil pengujian *cipherimage* PeppersRGB.bmp 512 x 512

Uji-5		PeppersRGB.bmp	Kriteria			
			I	II	III	IV
Korelasi	Vertikal	0,971542	0,842830	0,843835	0,841177	0,845400
	Horizontal	0,969960	0,060795	0,017671	0,031396	0,008129
	Diagonal	0,957243	0,059448	0,018892	0,031405	0,008208
	Rata-rata	0,966249	0,321024	0,293466	0,301326	0,287246
Entropy		7,297795	7,985890	7,986248	7,975508	7,987914

Berikut adalah grafik korelasi dari nilai korelasi yang ditunjukkan pada Tabel 5.9



Gambar 5.8 Grafik Korelasi Citra PeppersRGB.bmp 512 x 512

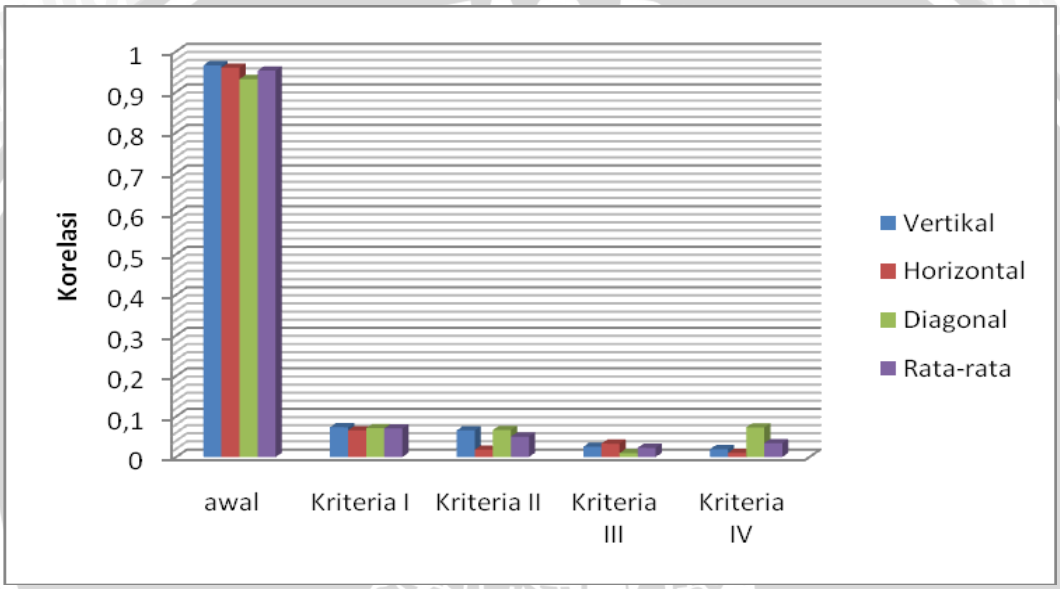
Berdasarkan grafik yang ditunjukkan pada gambar 5.8, dapat diamati penurunan piksel bertetangga vertikal dari kriteria I hingga IV jika dibandingkan dengan citra awal. Nilai korelasi piksel bertetangga horisontal dan diagonal pada keempat kriteria menunjukkan penurunan yang jauh signifikan terhadap piksel horisontal dan diagonal citra awal. Nilai rata-rata terendah didapat pada kriteria IV, penurunan paling signifikan yang dapat diamati pada kriteria IV adalah pada nilai korelasi ketetanggaan horisontal dan diagonal. Hal ini menunjukkan bahwa semakin panjang kunci yang digunakan, nilai korelasi semakin rendah.

Uji *cipherimage* PeppersRGB.bmp dengan resolusi 250 x 250 piksel didapatkan hasil berikut:

Tabel 5.10 Hasil pengujian *cipherimage* PeppersRGB.bmp 250 x 250

Uji-6		PeppersRGB.bmp	Kriteria			
			I	II	III	IV
Korelasi	Vertikal	0,965726	0,073277	0,065674	0,025299	0,019244
	Horizontal	0,959832	0,065782	0,018148	0,032832	0,010101
	Diagonal	0,931624	0,071331	0,066442	0,009465	0,072952
	Rata-rata	0,952394	0,070130	0,050088	0,022532	0,034099
Entropy		7,324691	7,987817	7,986958	7,978741	7,989328

Berikut adalah grafik korelasi dari nilai korelasi yang ditunjukkan pada Tabel 5.10



Gambar 5.9 Grafik Korelasi Citra PeppersRGB.bmp 250 x 250

Berdasarkan grafik yang ditunjukkan pada gambar 5.9, dapat diamati penurunan piksel bertetangga vertikal dari kriteria I hingga IV jika dibandingkan dengan citra awal. Nilai korelasi pada seluruh piksel bertetangga pada keempat kriteria menunjukkan penurunan yang jauh signifikan terhadap seluruh piksel bertetangga citra awal. Nilai rata-rata terendah didapat pada kriteria III, penurunan paling signifikan yang dapat diamati pada kriteria III adalah pada nilai korelasi ketetanggaan diagonal. Nilai korelasi ketetanggaan horisontal dan vertikal terendah adalah pada kriteria pengujian IV.

Total lima pengujian yang dilakukan menunjukkan hasil penurunan nilai korelasi citra setelah dilakukan enkripsi. Variabel yang ditentukan adalah nilai ketetanggaan piksel vertikal, horisontal dan diagonal dari citra. Semua citra berukuran 512 x 512 piksel menunjukkan penurunan korelasi piksel vertikal

terhadap citra aslinya yang tidak terlalu signifikan. Berbeda dengan piksel horisontal dan diagonal yang memiliki penurunan relatif lebih signifikan terhadap nilai korelasi diagonal dan horisontal yang sama pada citra aslinya.

Semua citra berukuran 250 x 250 piksel menunjukkan penurunan korelasi yang rata pada semua arah ketetanggaan. Hal ini karena pada resolusi citra yang lebih rendah, diperlukan lebih sedikit blok yang perlu diproses. Secara umum, aplikasi dapat menghasilkan pengamanan yang diingingkan ditandai dengan nilai korelasi *cipherimage* yang mengalami penurunan terhadap citra awal masing-masing. Hal ini menunjukkan perbedaan antar piksel saling bertetangga sehingga relasi antara piksel citra *cipherimage* dan citra asli semakin jauh. Semakin mendekati nol nilai korelasi *cipherimage*, maka semakin baik proses pengenkripsian dalam mengaburkan jejak informasi citra.

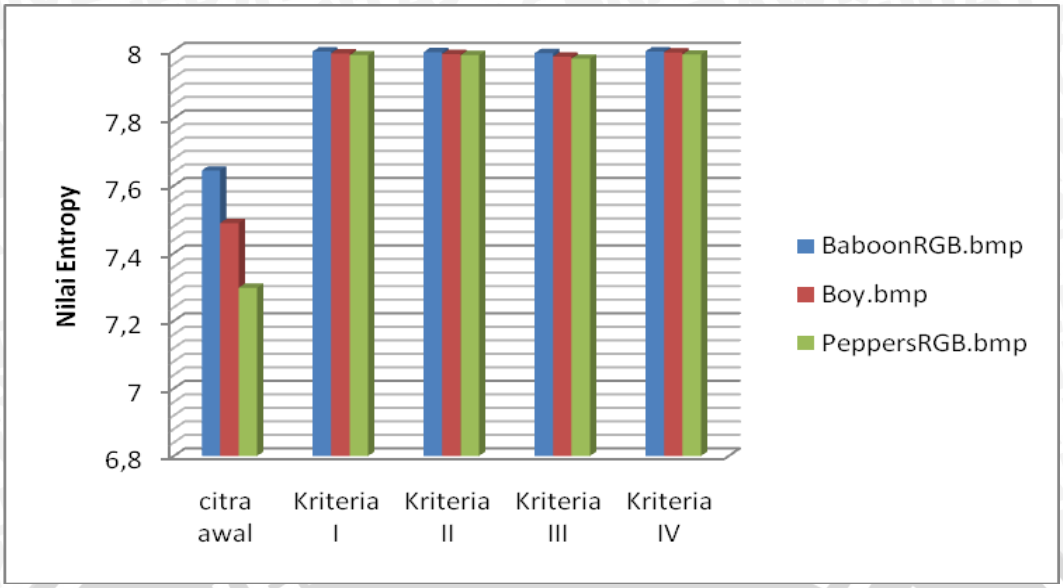
Dari seluruh hasil pengujian didapatkan bahwa semakin panjang kunci yang digunakan maka berpengaruh kepada seberapa besar perubahan nilai korelasi pada *cipherimage*. Semakin besar *array of word* yang digunakan, maka keragaman tabel kunci yang digunakan akan semakin meningkat yang menyebabkan makin besarnya variasi pengacakan blok-blok citra. Faktor lain yang menjadi indikator tingkat keberhasilan penyamaran citra adalah entropi. Semakin tinggi nilai entropi maka semakin tinggi probabilitas variasi piksel pembentuk citra yang artinya semkain besar pula perbedaan piksel antara citra awal dengan *cipherimage*.

Pengujian yang akan dilakukan adalah pengujian nilai entropi dari tabel citra yang terdapat pada tabel 5.5 – 5.9 sesuai citra berukuran 512 x 512, sebagai berikut:

Tabel 5.11 Entropy citra 512 x 512

Image 512x512		Nilai Awal	Kriteria Pengujian			
			I	II	III	
Entropy	BaboonRGB.bmp	7,644440	7,996967	7,995093	7,992061	7,997190
	Boy.bmp	7,488955	7,991356	7,989442	7,981721	7,993907
	PeppersRGB.bmp	7,297795	7,985890	7,986248	7,975508	7,987914

Berikut adalah grafik nilai *entropy* yang ditunjukkan pada Tabel 5.11:



Gambar 5.10 Grafik *Entropy* Citra Uji 512x512

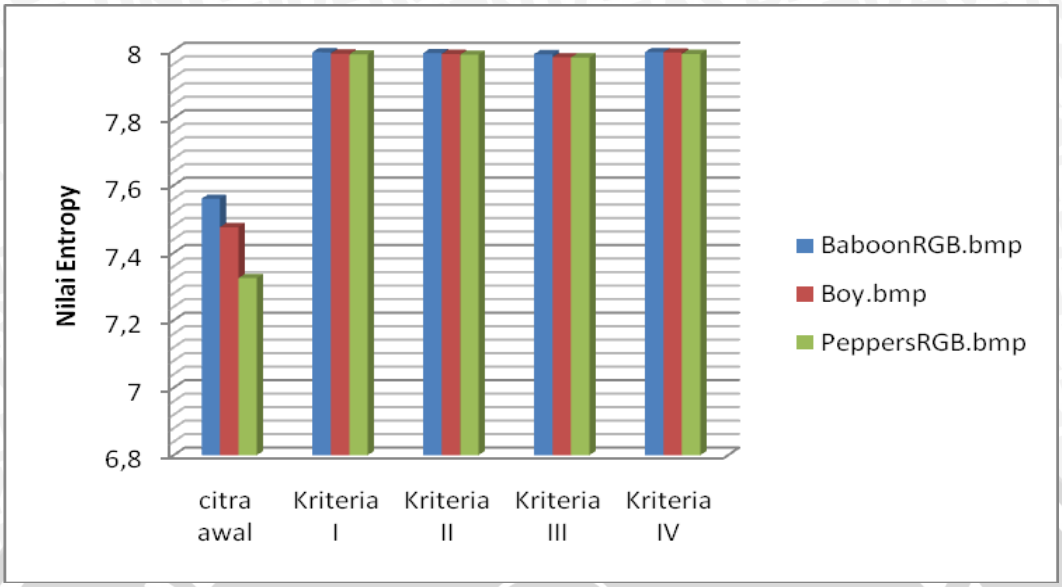
Perubahan nilai *entropy* dapat diamati pada grafik gambar 5.10. Kriteria pengujian I hingga IV menunjukkan perubahan entropi di mana nilai entropi pada keempat pengujian mengalami peningkatan secara merata jika dibandingkan dengan entropi citra awal.

Berdasarkan nilai *entropy* yang didapatkan pada Tabel 5.6 - 5.10, tabel nilai *entropy* pada citra uji yang berukuran 250x250 adalah sebagai berikut:

Tabel 5.12 Entropy citra 250 x 250

Image 250x250		Nilai Awal	Kriteria Pengujian			
			I	II	III	
Entropy	BaboonRGB.bmp	7,559831	7,994379	7,991768	7,988608	7,994532
	Boy.bmp	7,475765	7,990513	7,989090	7,979325	7,993578
	PeppersRGB.bmp	7,324691	7,987817	7,986958	7,978741	7,989328

Berikut adalah grafik nilai *entropy* yang ditunjukkan pada Tabel 5.12

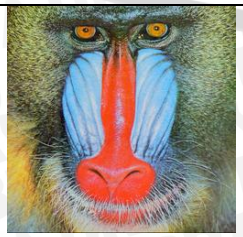
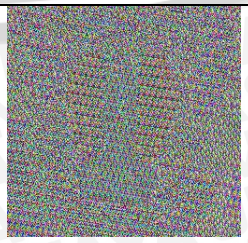


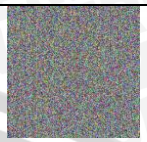
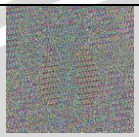
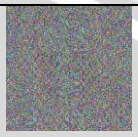
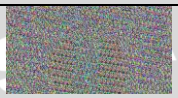
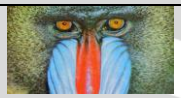
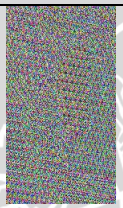
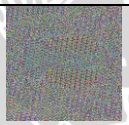
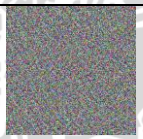
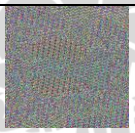
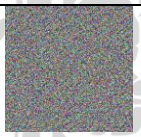
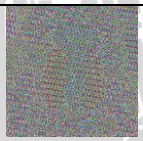
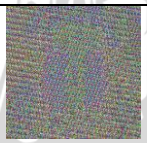
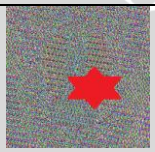
Gambar 5.11 Grafik *Entropy* Citra Uji 250x250

Perubahan nilai *entropy* dapat diamati pada grafik gambar 6.11. Kriteria pengujian I hingga IV menunjukkan perubahan entropi di mana nilai entropi pada keempat pengujian mengalami peningkatan secara merata jika dibandingkan dengan entropi citra awal. Hal ini sesuai dengan grafik pengukuran *entropy* pada citra 512 x 512 piksel. Semakin panjang jumlah word yang diberikan sebagai kunci, maka nilai *entropy cipherimage* akan semakin besar pada citra beresolusi apapun.

5.6 Hasil dan Pembahasan Uji Ketahanan *Cipherimage*

Dalam uji ketahan *cipherimage* ini, citra yang akan digunakan adalah citra BaboonRGB.bmp berukuran 250 x 250 piksel. Kunci yang digunakan adalah kunci dengan panjang 24 byte (digitalimageencryptionss). Sebelum dilakukan pengujian, *cipherimage* dilakukan proses manipulasi. Hasil pengujian adalah sebagai berikut:

Image asli (BaboonRGB.bmp)	***Cipherimage* BaboonRGB.bmp asli**
	

No.	Manipulasi *Cipherimage*	*Cipherimage* Termanipulasi	Hasil Dekripsi	Deskripsi
1.	Flip Horizontal			Gagal mendekripsi
2.	Flip Vertikal			Gagal mendekripsi
3.	Crop Horizontal			Berhasil mendekripsi sebagian
4.	Crop Vertikal			Gagal mendekripsi
5.	Rotate 90° cw			Gagal mendekripsi
6.	Rotate 90° ccw			Gagal mendekripsi
7.	Rotate 180°			Gagal mendekripsi
8.	Add noise			Berhasil mendekripsi bagian yang tidak terkena noise

Dari ketujuh pengujian yang dilakukan pada citra, hanya dua pengujian yang menghasilkan citra yang mampu terdekripsi. Lima pengujian yaitu *flip horizontal, flip vertical, crop vertical, rotate 90° clockwise, rotate 90° counterclockwise* dan *rotate 180°* menyebabkan kegagalan dalam dekripsi citra.

Manipulasi *croptop* dan *add noise* mampu mengembalikan keadaan sebagian citra terenkripsi.

Hal ini menunjukkan bahwa pendekripsian citra tergantung kepada posisi bit-bit blok awalnya. Saat susunan bloknya berbeda seperti saat dilakukan *flip horizontal* (lokasi bit terbalik secara simetris horizontal), *flip vertical* (lokasi bit terbalik simetris vertikal), *crop vertical* (crop sisi vertikal), *Rotate 90° cw* (lokasi bit ditukar rotasi 90°searah jarum jam), *Rotate 90° ccw* (lokasi bit ditukar rotasi 90° berlawanan jarum jam) dan *Rotate 180°* (setengah putaran/rotasi 180°; lokasi bit ditukar dengan lokasi bit yang saling bertolak belakang), *cipherimage* tidak dapat terdekripsi.

Namun pada pengujian dengan *cropping* dan penambahan *noise*, posisi blok citra pada dasarnya adalah tetap. Pada *cropping*, yang di*crop* adalah bagian atas citra sehingga blok-blok pada bagian yang hilang (sisi setengah bagian bawah citra) tidak akan mengganggu bagian blok yang masih dapat didekripsi. Pada pemberian *noise*, hanya bagian yang tertutup objek asing yang tidak terdekripsi. Sisa blok yang tetap sesuai kondisi *cipherimage* awalnya masih dapat didekripsi dengan baik.

BAB VI
PENUTUP

6.1. Kesimpulan

Kesimpulan dari penelitian ini adalah:

1. Algoritma *Modified Rivest Code 6* merupakan algoritma yang dapat digunakan untuk melakukan proses enkripsi dan dekripsi. Algoritma ini dapat diimplementasikan dalam pengamanan citra. Algoritma ini bekerja melalui pembentukan tabel kunci S berupa w-bit *word* yang berasal dari kata yang dipilih *user* sebagai pengaman citra. Langkah selanjutnya adalah menyusun ulang blok-blok citra dari citra awal menjadi blok yang baru yang tidak memiliki korelasi dengan blok-blok pada citra awal.
2. Pada saat proses enkripsi nilai korelasi antar *piksel* yang bertetanggaan pada citra digital mengalami rata-rata penurunan 0,769357 dan nilai *entropy* mengalami rata-rata penambahan sebesar 0,523834 untuk 6 kali pengujian. Penurunan pada nilai korelasi untuk setiap pengujian semakin mengarah ke nilai nol, sedangkan perbedaan nilai entropy pada *cipherimage* diketahui mendekati angka delapan. Hal ini menunjukkan bahwa pengenkripsian telah menghasilkan *cipherimage* yang baik dan rata-rata semakin panjang kunci yang digunakan, maka nilai korelasi yang didapatkan akan semakin rendah dan *cipherimage* yang dihasilkan memiliki tingkat pengacakan tinggi (nilai entropy semakin tinggi).
3. Susunan blok pada citra berpengaruh terhadap keberhasilan pendekripsian dan berpengaruh pula terhadap tingkat keamanan citra. Apabila dilakukan manipulasi citra yang mengubah susunan blok-blok penyusunnya, meskipun diberikan kata kunci yang sesuai dengan yang diberikan saat akan mengenkripsi, tetapi citra tidak akan dapat didekripsi secara keseluruhan. Apabila dilakukan manipulasi terhadap sebagian wilayah citra tanpa mengubah susunan blok-blok aslinya, maka citra masih akan dapat terdekripsi.

6.2. Saran

Saran untuk mengembangkan penelitian ini pada kemudian hari adalah:

1. Peneliti hanya menguji metode yang digunakan dari segi enkripsi-dekripsi. Untuk peningkatan keamanan, dapat dilakukan penambahan variasi pengamanan dengan melakukan penyisipan citra pada media citra atau audio dengan lebih dahulu mengimplementasikan dasar enkripsi menggunakan algoritma *Modified Rivest Code 6* yang telah diteliti.
2. Algoritma *MRC6* memanfaatkan 16 register yang memungkinkan *throughput* lebih besar dengan jumlah rotasi yang diberikan seminimal mungkin, sehingga secara teori menghasilkan efisiensi enkripsi yang lebih besar dan cepat. Untuk pembuktian efisiensi algoritma *Modified Rivest Code 6* dibandingkan dengan algoritma kriptografi lain, peneliti selanjutnya dapat membandingkan waktu pengenkripsian yang dibutuhkan terhadap satu file yang sama menggunakan beberapa macam algoritma lainnya.
3. Dapat dilakukan alternatif penambahan penghitungan nilai PSNR (*peak signal to noise ratio)* untuk mengetahui tingkat keberhasilan penyamaran *ciphermimage* dibandingkan dengan *plainimage*.

DAFTAR PUSTAKA

[ALI-08] Ali Bani, Younes and Jantan, Aman. 2008. *Image Encryption Using Block-Based Transformation Algorithm*. IAENG International Journal of Computer Science. Universiti Sains. Malaysia.

[BOE-97] Boehm, B. W. and Bose, P. 1997. A Collaborative Spiral Software Process Model Based on Theory W, Proceedings of the 1994 International Conference on Software Process, IEEE Computer Society, Washington.

[BUU-05] Bu'ulolo, Roland L. 2006. *Perbandingan Algoritma Block Cipher RC5 dan RC6*. ITB. Bandung.

[DAE-99] Daemen, Joan. 1999. *Advanced Encryption Standard*. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf. Tanggal akses: 28 Maret 2012

[ELF-05] El Fishawy, N, et al. 2005. *A Modification of RC6 Cipher Algorithm for Data Security (MRC6)*. ieeexplore.ieee.org/iel5/9457/30014/01374428.pdf?arnumber=1374428. Tanggal akses: 01 Juni 2012.

[DHA-03] Dhani, Pratikaningtyas, dkk. 2003. Makalah Tugas Akhir : Klasifikasi Motif Batik Menggunakan Metode Transformasi Paket Wavelet. Universitas Diponegoro. Semarang.

[MUN-04] Munir, R. 2004. Kuliah IF5054 Kriptografi :*Tipe dan Mode Algoritma Simetri*. Institut Teknologi Bandung. Bandung.

[MUN-06] Munir, R. 2006. *Kriptografi*. Penerbit Informatika. Bandung.

[RAG-01] Ragab, A.H.M. 2001. *Enhancements and implementation of RC6^{TM} block cipher for data security*. ieeexplore.ieee.org/iel5/7544/20542/00949566.pdf. Tanggal Akses: 02 januari 2013

[RAH-99] Raharjo. 1999. *Keamanan Sistem Informasi Berbasis Internet*. PT Insan komunika/Infonesia. Bandung.

[RIV-98] Rivest L, Ronald., Robshaw, M.J.B., Sidney, R., Yin, Y.L. 1998. *The RC6 Block Cipher*. RCA Laboratories. New York

[ROB-99] Robertus, Leonard. 1999. Perancangan Perangkat Lunak Untuk Siswa SMK. Penerbit Erlangga. Jakarta.

[RUD-07] Rudianto. 2007. *Analisis Keamanan Algoritma Kriptografi RC6*. Institut Teknologi Bandung. Bandung.

[WAH-03] Wahana. 2003. *Memahami Model Enkripsi dan Security Data*. Andi Offset. Yogyakarta.