UNIVERSITAS INDONESIA

APLIKASI ALGORITMA RIVEST CODE 6 DALAM PENGAMANAN CITRA DIGITAL

SKRIPSI

ANISAH MUHARINI
0806315300

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
PROGRAM STUDI SARJANA MATEMATIKA
DEPOK
JULI 2012

UNIVERSITAS INDONESIA

APLIKASI ALGORITMA RIVEST CODE 6 DALAM PENGAMANAN CITRA DIGITAL

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana sains

ANISAH MUHARINI
0806315300

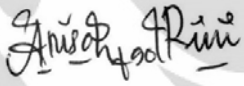
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
PROGRAM STUDI SARJANA MATEMATIKA
DEPOK
JULI 2012

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar.

Nama	: Anisah Muharini
NPM	: 0806315300
Tanda Tangan	: 
Tanggal	: Juli 2012

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh

Nama	:	Anisah Muharini
NPM	:	0806315300
Program Studi	:	Matematika
Judul Skripsi	:	Aplikasi Algoritma Rivest Code 6 dalam Pengamanan Citra Digital

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Sains pada Program Studi S1 Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Indonesia

DEWAN PENGUJI

Pembimbing	: Drs. Suryadi MT, M.T.	( )
Penguji I	: Dr. Al Haji Akbar, M.Sc	( )
Penguji II	: Dr. Yudi Satria, M.T.	( )
Penguji III	: Dr. rer. nat Hendri Murfi, M.Kom	( )

Ditetapkan di : Depok
Tanggal : 19 Juni 2012

KATA PENGANTAR

Segala puji dan syukur penulis haturkan kepada Allah SWT atas segala rahmat, petunjuk, dan karunia-Nya. Shalawat dan salam semoga senantiasa tercurah kepada pemimpin umat manusia, Nabi Muhammad SAW, beserta para sahabatnya, yang merupakan kumpulan orang - orang terbaik sepanjang masa.

Alam menggemakan siluet kehidupan, langit terus menyibakkan rahasianya, dan rasa syukur pun tak pernah terhenti akan selesainya skripsi ini. Penulisan skripsi ini disusun sebagai salah satu syarat dalam kelulusan program sarjana strata satu Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam.

Dalam penyelesaian skripsi ini, penulis menyadari akan banyaknya kesulitan yang menghadang, namun berkat bimbingan dan dorongan dari berbagai pihak, akhirnya skripsi ini dapat diselesaikan. Oleh karena itu, pada kesempatan ini, penulis ingin menyampaikan terima kasih yang sedalam-dalamnya kepada:

1. Bapak Dr. Yudi Satria, M.T. selaku Ketua Departemen Matematika, Ibu Rahmi Rusin, S.Si., M.ScTech. selaku Sekretaris Departemen Matematika, Ibu Mila Novita selaku Koordinator Kemahasiswaan, dan Ibu Dr. Dian Lestari selaku Koordinator Pendidikan yang telah membantu penulis selama penulis menjalani kuliah di Matematika UI.
2. Bapak Drs. Suryadi MT, M.T., selaku dosen pembimbing yang senantiasa memberikan waktu, tenaga, dan pikiran untuk mengarahkan penulis dengan sabar selama proses penyusunan skripsi ini hingga selesai.
3. Ibu Dr. Sri Mardiyati, M.Kom selaku Pembimbing Akademis penulis yang memberikan motivasi dan dukungan kepada penulis dalam hal apapun.
4. Bapak Dr. Al Haji Akbar, M.Sc., Bapak Dr. Yudi Satria, M.T., dan Bapak Dr. rer. nat Hendri Murfi, M.Kom selaku penguji kolokium penulis yang bersedia meluangkan waktunya, memberi kritik, saran, serta ilmu yang bermanfaat.

5. Ibu dan Bapak tercinta yang selalu memberikan do'a, perhatian, semangat dan nasehat baik secara moril, materil dan spiritual yang tiada ternilai harganya sehingga penulis bisa menyelesaikan penyusunan skripsi ini.
6. Seluruh dosen dan karyawan Departemen Matematika FMIPA UI yang tanpa mengurangi rasa hormat tidak dapat disebutkan satu per satu telah berbagi ilmu dan membimbing selama penulis mengikuti perkuliahan.
7. Muhammad Faruq Iskandar selaku belahan jiwa penulis yang senantiasa memberikan dukungan berupa doa, saran, kritik, semangat dan sangat membantu dalam proses kelancaran penyelesaian skripsi ini.
8. Kakak-kakak penulis Uda Darlis (alm), Uni, One, Ayang, Ajo Eri, dan Ajo Adi yang senantiasa mendoakan kelancaran skripsi penulis serta keponakan-keponakan penulis Salman, Farid, Ayyasy, Subhan, Aisyah, Faizah, Syahidah, Mufida, Syifa, Zidan, Nia, Syamil, Chiesa, Yasmin dan Zahid yang telah memberikan canda tawanya kepada penulis.
9. Teman-teman 2008: seperjuangan penulis Dhila, Aci, dan Maul yang telah berbagi tawa dan tangis, demi tersusunnya skripsi ini juga teman-teman yang sedia membagi ilmunya Bang Andy, Umbu, Adhi, dan Maimun serta teman yang banyak menghabiskan waktu bersama selama masa kuliah penulis sehingga penulis tegar dalam menjalani masa sulit dari awal kuliah sampai terususunnya tugas akhir ini Siwi, Ifah, Yulial, Tuti '*skirt&CD*', Fani '*pointer*', Risya, Nita, Olin, Resti, Agnes, Hindun, Icha, Uchi, Mei, Ega, Uci, May, Ines, Dhewe, Wulan, Luthfa, Dian, Citra, Nora, Icha, Numa, Janu, Kohen, Arkies, Cindy, Bowo, Vika, Dhea, Emy, Eka, Arman, Agy, Awe, Arief, Dheni, Sita, Laili, Yulian, Puput, Aya, dan Masykur.

Akhir kata, penulis berharap Allah membalas segala kebaikan semua pihak yang telah membantu dan penulis berdoa semoga apa yang telah dikerjakan ini tidak menjadi hal yang sia-sia dan dapat bermanfaat bagi pengembangan ilmu dan bagi siapapun yang membacanya.

Penulis

2012

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

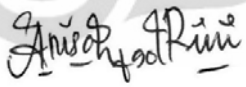
Nama	:	Anisah Muharini
NPM	:	0806315300
Program Studi	:	Sarjana Matematika
Departemen	:	Matematika
Fakultas	:	Matematika dan Ilmu Pengetahuan Alam
Jenis karya	:	Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty Free Right*) atas karya ilmiah saya yang berjudul :
Aplikasi Algoritma Rivest Code 6 dalam Pengamanan Citra Digital.

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/format-kan, mengelola dalam bentuk pangkalan data (database), merawat, dan memublikasikan saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : Juli 2012
Yang menyatakan



(Anisah Muharini)

ABSTRAK

Nama : Anisah Muharini
Program Studi : Matematika
Judul : Aplikasi Algoritma Rivest Code 6 dalam Pengamanan Citra Digital

Usaha perlindungan terhadap citra digital merupakan hal yang penting dan mendesak abad ini, karena dapat memperkecil usaha manipulasi citra digital tesebut untuk hal yang negatif. Usaha tersebut dapat dilakukan dengan penyandian(enkripsi) menggunakan algoritma Rivest Code 6 (RC6), yakni suatu algoritma *block cipher* yang menggunakan kunci simetri. Algoritma RC6 ini dikenal dengan kesederhanaannya juga variasi panjang kunci yang dapat digunakan untuk melakukan enkripsi maupun dekripsi. Oleh karena itu diharapkan sandi(*ciphertext*) dari implementasi program algoritma RC6 ini sulit dipecahkan sehingga citra digital dapat terlindungi dengan baik.

Kata Kunci : Algoritma RC6, Citra Digital, Enkripsi, Dekripsi
xii + 64 halaman : 23 gambar, 15 tabel
Daftar Pustaka : 15 (1995 - 2011)

ABSTRACT

Name : Anisah Muharini
Study Program : Mathematics
Title : Application of Rivest Code 6 Algorithm on Securing Digital Image

Efforts in securing digital image is something that important and urgent today because it can reduce risk of digital image manipulation. These efforts can be done with an encryption using Rivest Code 6 algorithm, a block cipher algorithm which use symmetric key. This Algorithm recognized by its simplicity.With RC6, user can choose the length of key that will be used for encryption and decryption. Hope that ciphertext from implementation of this algorithm can be well protected from any manipulation.

Keywords : Rivest Code 6 Algorithm, Digital Image, Encryption, Decryption.
xii + 64 pages : 23 pictures, 15 tables
Bibliography : 15 (1995 - 2011)

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi dan komunikasi saat ini memudahkan manusia untuk mengakses berbagai sumber data/informasi dari berbagai belahan dunia. Penyajian dan penyimpanan informasi atau data kini dapat disimpan dalam format digital dan memiliki beragam bentuk seperti teks, citra, audio, video, dan multimedia. Akhir-akhir ini, semakin banyak muncul berbagai *software* untuk mengedit dan memodifikasi citra digital dengan mudah, bahkan kalangan awampun bisa menggunakannya. Hasil modifikasi mereka pun tidak sedikit yang bernada negatif, baik untuk membuat fitnah, pencitraan buruk seseorang atau sekelompok orang dan berbagai maksud lainnya (Lihat Gambar 1.1).

Gambar 1.1 Contoh Manipulasi Citra Digital

Hal ini tentunya akan mengganggu para pemilik data, dan juga pihak yang ingin berkomunikasi dalam rangka bertukar informasi baik kepentingan pribadi maupun kelompok. Sehingga dapat dikatakan bahwa perlindungan orisinalitas suatu data atau informasi menjadi kebutuhan yang penting dan mendesak saat ini. Untuk itu diperlukan usaha pengamanan data tersebut supaya keaslian dan kerahasiannya bisa terjaga.

Usaha perlindungan data dapat dilakukan dengan berbagai cara, salah satunya dengan mengaplikasikan bidang kriptografi. Usaha pengamanan dalam kriptografi bisa dilakukan dengan mengenkripsi data citra digital tersebut ke dalam bentuk data lain yang tidak bisa dikenali (*ciphertext)*. Untuk mengembalikan *ciphertext* menjadi seperti data semula (*plaintext*), dapat dilakukan dengan mendekripsi *ciphertext* yakni dengan menggunakan algoritma kebalikan atau invers pada *ciphertext* tersebut.

Keamanan algoritma *Data Encryption Standard* (DES) telah berhasil ditembus oleh suatu *software* pada akhir tahun 90-an dalam beberapa jam (Tilborg, 2011). Guna mengatasi hal itu, pada tahun 2000, muncul beberapa algoritma enkripsi alternatif yang bersaing untuk menjadi standar algoritma enkripsi baru yang dikenal dengan istilah *Advanced Encryption Standard* (AES). Kandidat AES adalah algoritma Rijndael, Serpent, Twofish, RC6, dan Mars. Namun, pada akhirnya algoritma Rijndael lah yang terpilih sebagai AES. Walaupun begitu, keempat kandidat lainnya juga merupakan algoritma hebat yang tak lepas dari kelebihan dan kekurangan.

Algoritma RC6 menawarkan kesederhanaan dan fleksibilitas sehingga keamanan algoritma RC6 lebih baik dari algoritma Rijndael (Contini, 1998). Kesederhanaan algoritma ini dapat dilihat dari operasi-operasi matematika dasar yang terdapat pada algoritma RC6 ini seperti penjumlahan, pengurangan, dan XOR. Sedangkan fleksibilitas algoritma ini dapat dilihat dari 3 parameter yang siap diubah oleh pengguna. Ketiga parameter yang siap diubah tersebut adalah panjang blok yang diolah atau ukuran register w dalam bit, banyak iterasi r, dan panjang kunci b sehingga citra digital dapat diamankan lebih baik atas adanya variasi parameter yang ditawarkan. Untuk itu, algoritma yang akan dibahas dalam skripsi ini adalah algoritma Rivest Code 6 (RC6) dan direpresentasikan ke dalam perlindungan data citra digital.

1.2 Perumusan Masalah dan Ruang Lingkup Masalah

Perumusan masalah dalam tugas akhir ini adalah sebagai berikut:

a) Bagaimana konsep algoritma RC6 secara umum?

b) Bagaimana menerapkan algoritma RC6 dalam melindungi data citra digital?

c) Bagaimana membangun program aplikasi algoritma RC6 dalam implementasinya terhadap pengamanan citra digital?

Ruang Lingkup dalam tugas akhir ini adalah sebagai berikut:

a) Implementasi algoritma RC6 pada citra digital *grayscale*

b) Algoritma RC6 yang dipakai adalah algoritma RC6 dengan panjang blok yang diolah (ukuran register) $w = 32$ bit, banyak iterasi $r = 20$, dan panjang kunci $b = 16, 24,$ dan 32 *byte*

c) *Software* yang dipakai dalam implementasi algoritma RC6 adalah Matlab

1.3 Tujuan Penelitian

Tujuan dari penulisan skripsi ini adalah sebagai berikut:

a) Menjelaskan gambaran umum mengenai algoritma RC6

b) Mengimplementasikan algoritma RC6 untuk pengamanan data citra digital

BAB 2
LANDASAN TEORI

Bab ini akan membahas definisi citra digital dan konsep dasar penyandian.

2.1 Definisi Citra Digital

Citra (*image*) adalah istilah lain untuk gambar yang merupakan salah satu komponen multimedia dan memegang peranan penting sebagai bentuk informasi visual. Citra mempunyai karakteristik yang tidak dimiliki oleh data berbentuk teks yakni kaya dengan informasi. Ada sebuah peribahasa, *a picture is more than a thousand words*, artinya sebuah gambar dapat memberikan informasi lebih banyak daripada informasi tersebut disajikan dalam bentuk ribuan kata (Munir, 2004).

Citra ada dua macam, yakni citra kontinu dan citra diskrit. Citra kontinu dihasilkan dari sistem optik yang menerima sinyal analog sedangkan citra diskrit dihasilkan melalui proses digitalisasi terhadap citra kontinu.

Citra diskrit disebut juga citra digital yang dapat dinyatakan sebagai suatu fungsi dua dimensi $f(x, y)$ dengan x maupun y merupakan posisi kooordinat. Sedangkan f merupakan intensitas cahaya (*brightness*) pada (x, y). Suatu citra digital dapat dinyatakan dengan persamaan 2.1.

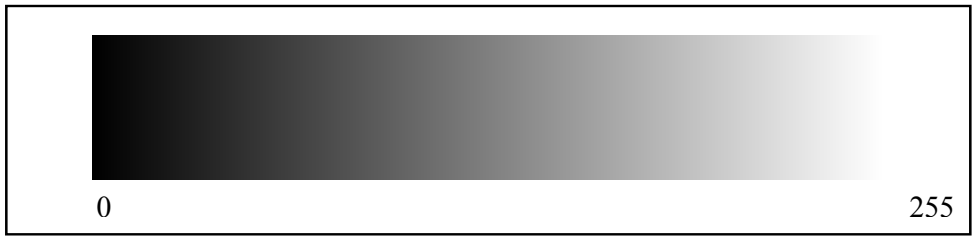
$$f(x,y) = \begin{pmatrix} f(1,1) & f(1,2) & f(1,3) & & f(1,n) \\ f(2,1) & f(2,2) & f(2,3) & \cdots & f(2,n) \\ f(3,1) & f(3,2) & f(3,3) & & f(3,n) \\ & \vdots & & \ddots & \vdots \\ f(m-1,1) & f(m-1,2) & f(m-1,3) & \cdots & f(m-1,n) \\ f(m,1) & f(m,2) & f(m,3) & & f(m,n) \end{pmatrix} \quad (2.1)$$

Dari persamaan 2.1 citra digital dapat dinyatakan sebagai matriks dengan banyak baris matriks/tinggi citra$= m$ dan banyak kolom matriks/lebar citra$= n$. Indeks baris x dan indeks kolom y menyatakan suatu koordinat titik pada citra, sedangkan $f(x, y)$ merupakan intensitas pada titik (x, y). Masing-masing elemen matriks tersebut disebut *image element*/*picture element*/*pixel*. Jadi, citra yang berukuran mxn mempunyai mn buah *pixel*.

Suatu citra disimpan dalam bentuk *file* dengan format tertentu. Format citra yang baku di lingkungan sistem operasi Microsoft Windows adalah BMP(*Bitmap*). Saat ini memang format BMP kalah popular dengan format JPG(*Join Photographic Group*) maupun GIF(*Graphics Interchange Format*). Hal ini karena format BMP tidak dimampatkan (dikompres) sehingga ukurannya relatif lebih besar daripada berkas JPG maupun GIF. Hal ini yang menyebabkan format BMP sudah jarang digunakan (Munir, 2004).

Format BMP memang tidak efisien dari segi ukuran, namun format BMP memiliki kelebihan dari segi kualitas gambar. Citra dalam format BMP lebih bagus daripada citra dalam format yang lainnya karena tidak ada informasi yang hilang akibat pemampatan (kompresi) seperti pada format JPG dan GIF. Terjemahan *bitmap* adalah pemetaan bit, artinya nilai intensitas *pixel* dalam citra dipetakan ke sejumlah bit tertentu. Oleh karena itu tugas akhir ini mengamankan citra dalam format BMP.

Salah satu citra dalam format BMP adalah citra hitam putih. Nilai intensitas cahaya f pada gambar hitam putih di titik (x, y) disebut derajat keabuan (*grey level*), sedangkan citranya disebut citra hitam-putih (*grayscale image*). Nilai derajat keabuan terletak pada skala keabuan $[0, L]$, dengan 0 menyatakan hitam dan L menyatakan putih. Citra hitam-putih yang biasa digunakan adalah citra hitam putih dengan 256 level. Citra ini direpresentasikan dalam 8 bit pada tiap *pixel*-nya, sehingga nilai derajat keabuannya ada $2^8 = 256$ level, artinya citra ini mempunyai skala keabuan $[0, 255]$. Derajat keabuan pada citra *grayscale* 256 level dapat dilihat pada Gambar 2.1.



Gambar 2.1 Derajat Keabuan

2.2 Konsep Dasar Penyandian

Kriptografi adalah seni dan ilmu dalam menggunakan matematika untuk mengamankan informasi. Pengamanannya bisa dilakukan dengan mengenkripsi data tersebut ke dalam bentuk data lain yang tidak bisa dikenali (*ciphertext)*. Untuk mengembalikan *ciphertext* menjadi data semula (*plaintext*), dapat dilakukan dengan mendekripsi *ciphertext* menggunakan algoritma kebalikan pada *ciphertext* tersebut.

Untuk melakukan enkripsi atau penyandian diperlukan suatu fungsi yang memetakan *plaintext* ke *ciphertext*. Lalu untuk melakukan dekripsi dibutuhkan fungsi invers yang memetakan *ciphertext* kembali menjadi *plaintext*. Persamaan 2.2 dan 2.3 merupakan ilustrasi pemetaan yang dilakukan oleh fungsi enkripsi E dan fungsi invers enkripsi E (fungsi dekripsi $D=E^{-1}$) antara himpunan *plaintext* P dan himpunan *ciphertext* C.

$$E: P \rightarrow C \qquad (2.2)$$

$$D: C \rightarrow P \qquad (2.3)$$

Fungsi yang digunakan haruslah fungsi yang bijektif karena akan terjamin adanya fungsi invers (lihat Definisi 2.2) sehingga *ciphertext* dapat dikembalikan menjadi *plaintext*. Berikut definisi dan teorema mengenai fungsi bijektif:

Definisi 2.1: (Rosen, 2007)

Fungsi $f: A \rightarrow B$ disebut 1-1 atau injektif $\leftrightarrow f(a) = f(b)$ maka $a = b$, untuk setiap a dan b domain dari f.

Fungsi $f: A \rightarrow B$ disebut *onto* atau surjektif $\leftrightarrow \forall\, b \in B, \exists a \in A \ni b = f(a)$

Fungsi $f: A \rightarrow B$ disebut bijektif jika f surjektif dan injektif

Definisi 2.2: (Rosen, 2007)

Misalkan f adalah fungsi bijektif dari himpunan A ke himpunan B. Fungsi invers f adalah fungsi yang memetakan $b \in B$ ke elemen tunggal $a \in A \ni f(a) =$

b. Fungsi invers dari f dinotasikan dengan f^{-1}. Oleh karena itu, $f^{-1}(b) = a$ saat $f(a) = b$.

Fungsi bijektif disebut *invertible* karena fungsi invers-nya dapat didefinisikan. Suatu fungsi dikatakan tidak *invertible* jika fungsi tersebut bukan fungsi bijektif, karena fungsi invers dari fungsi tersebut tidak ada (Rosen, 2007).

Teorema 2.1: (Herstein, 1995)

Jika $f: X \to Y$ adalah fungsi bijektif, maka ∃ sebuah fungsi invers

$f^{-1}: Y \to X$ yang juga bijektif

Bukti mengenai Teorema 2.1 dapat dilihat pada Lampiran 1.

Teorema 2.1 mengindikasikan bahwa enkripsi yang merupakan fungsi bijektif mempunyai fungsi invers yang juga bijektif yaitu dekripsi. Selanjutnya, enkripsi dan dekripsi ini bisa merupakan komposisi dari beberapa fungsi. Definisi komposisi fungsi dapat dilihat pada Definisi 2.3.

Definisi 2.3: (Herstein, 1995)

Jika $g: S \to T$ dan $f: T \to U$, maka komposisinya $(f \circ g)$adalah pemetaan

$f \circ g: S \to U$, didefinisikan dengan $(f \circ g)(s) = f\big(g(s)\big) \forall s \in S$

Komposisi dari beberapa fungsi bijektif juga merupakan fungsi bijektif, sesuai dengan Lemma 2.1 dan Teorema 2.2.

Lemma 2.1: (Herstein, 1995)

Jika $g: S \to T$ dan $f: T \to U$ keduanya merupakan fungsi bijektif maka komposisinya $f \circ g: S \to U$ juga bijektif

Bukti mengenai Lemma 2.1 dapat dilihat pada Lampiran 2.

Teorema 2.2 (Herstein, 1995)

Jika $g: S \to T$ dan $f: T \to U$ fungsi yang bijektif, maka $(f \circ g)^{-1}$ juga fungsi yang bijektif dan $(f \circ g)^{-1} = g^{-1} \circ f^{-1}$

Bukti mengenai Teorema 2.2 dapat dilihat pada Lampiran 3.

Teorema 2.2 menjamin bahwa suatu penyandian yang dilakukan berkali-kali memiliki invers yang merupakan fungsi bijektif pula dan sandi tersebut dapat dikembalikan dengan mendekripsi secara berurutan dari akhir ke awal.

Untuk melakukan enkripsi dengan fungsi bijektif seperti pada persamaan 2.2, dibutuhkan suatu kunci k elemen dari himpunan K, yaitu himpunan karakter yang elemen-elemennya berupa angka, huruf, dan simbol. Maka, fungsi enkripsi E dengan kunci k didefinisikan pada persamaan 2.4.

$Ek : P \rightarrow C$ dengan $k \in K$ (2.4)

Algoritma RC6 merupakan algoritma simetris, yaitu algoritma penyandian yang menggunakan kunci enkripsi sama dengan kunci dekripsi. Sehingga untuk melakukan dekripsi dengan fungsi dekripsi D seperti pada persamaan 2.3 juga digunakan kunci yang sama dengan kunci k pada proses enkripsi, sehingga fungsi dekripsi D sebagaimana pada persamaan 2.5.

$Dk : C \rightarrow P$ dengan $k \in K$ (2.5)

Varian algoritma RC6 sebelumnya, yaitu algoritma RC4 merupakan algoritma yang mengolah data per bit atau *byte* disebut *stream cipher*. Sedangkan algoritma RC6 merupakan algoritma yang mengolah data per blok (barisan dari bit bit yang panjangnya tetap, biasanya 64 bit atau lebih) disebut *block cipher*. Dalam hal ini, RC6 menyandikan blok data sebesar 128 bit sebagai syarat minimum AES.

2.3 Operasi Matematika

Operasi matematika yang akan dijelaskan dalam hal ini adalah mengenai operasi logika, operasi bit, operasi aritmatika modular, dan operasi aritmatika biner.

2.3.1 Operasi Logika

Suatu proposisi adalah suatu pernyataan (*statement*) yang memiliki nilai kebenaran *true* (benar, T) atau *false* (salah, F) tetapi tidak keduanya pada saat dinyatakannya. Dari proposisi-proposisi yang ada dapat dibentuk proposisi baru dengan menggunakan penghubung atau operator logika. Operator logika yang dimaksud dalam hal ini adalah konjungsi ($\wedge$), disjungsi ($\vee$), eksklusif or ($\oplus$), implikasi ($\rightarrow$), dan biimplikasi ($\leftrightarrow$), masing-masing didefinisikan sebagai berikut:

Definisi 2.4: (Rosen, 2007)

Jika p dan q adalah proposisi maka "p dan q" atau $p \wedge q$ adalah sebuah proposisi pula, yang disebut sebagai konjungsi dari p dan q. Nilai kebenaran dari $p \wedge q$ adalah benar pada saat p dan q keduanya bernilai benar dan salah bila salah satu atau kedua-duanya dari p dan q bernilai salah

Berikut disajikan tabel nilai kebenaran untuk suatu proposisi p dan proposisi q dengan operator konjungsi pada Tabel 2.2.

Tabel 2.1 Nilai Kebenaran Operator Konjungsi ($\wedge$)

p	q	$p \wedge q$
Benar	Benar	Benar
Benar	Salah	Salah
Salah	Benar	Salah
Salah	Salah	Salah

Definisi 2.5: (Rosen, 2007)

Jika p dan q adalah proposisi maka "p atau q" atau $p \vee q$ adalah sebuah proposisi pula, yang disebut sebagai disjungsi dari p dan q. Nilai kebenaran dari $p \vee q$ adalah salah pada saat p dan q keduanya bernilai salah dan benar bila salah satu atau keduanya dari p dan q bernilai benar

Berikut disajikan tabel nilai kebenaran untuk suatu proposisi p dan proposisi q dengan operator disjungsi pada Tabel 2.3.

Tabel 2.2 Nilai Kebenaran Operator Disjungsi (∨)

p	q	$p \vee q$
Benar	Benar	Benar
Benar	Salah	Benar
Salah	Benar	Benar
Salah	Salah	Salah

Definisi 2.6: (Rosen, 2007)

Jika p dan q adalah proposisi maka eksklusif or dari p dan q adalah sebuah proposisi pula. Nilai kebenaran dari $p \oplus q$ adalah benar pada saat p dan q memiliki nilai kebenaran yang berbeda, dan salah bila p dan q memiliki nilai kebenaran yang sama

Berikut disajikan tabel nilai kebenaran untuk suatu proposisi p dan proposisi q dengan operator eksklusif or pada Tabel 2.4.

Tabel 2.3 Nilai Kebenaran Operator Eksklusif Or (⊕)

p	q	$p \oplus q$
Benar	Benar	Salah
Benar	Salah	Benar
Salah	Benar	Benar
Salah	Salah	Salah

Definisi 2.7: (Rosen, 2007)

Jika p dan q adalah proposisi maka implikasi $p \rightarrow q$ adalah sebuah proposisi pula. Nilai kebenaran dari $p \rightarrow q$ adalah salah hanya pada saat p bernilai benar dan q bernilai salah, selainnya $p \rightarrow q$ akan bernilai benar

Berikut disajikan tabel nilai kebenaran untuk suatu proposisi p dan proposisi q dengan operator implikasi pada Tabel 2.5.

Tabel 2.4 Nilai Kebenaran Operator Implikasi (→)

p	q	$p \rightarrow q$
Benar	Benar	Benar
Benar	Salah	Salah
Salah	Benar	Benar
Salah	Salah	Benar

Definisi 2.8: (Rosen, 2007)

Jika p dan q adalah proposisi maka bikomposisi $p \leftrightarrow q$ juga sebuah proposisi. Nilai kebenaran dari $p \leftrightarrow q$ adalah benar pada saat p dan q memiliki nilai kebenaran yang sama, dan salah bila p dan q memiliki nilai kebenaran yang berbeda.

Berikut disajikan tabel nilai kebenaran untuk suatu proposisi p dan proposisi q dengan operator biimplikasi pada Tabel 2.6.

Tabel 2.5 Nilai Kebenaran Operator Biimplikasi (↔)

p	q	$p \leftrightarrow q$
Benar	Benar	Benar
Benar	Salah	Salah
Salah	Benar	Salah
Salah	Salah	Benar

2.3.2 Operasi Bit

Komputer merepresentasikan informasi dalam bit-bit. Suatu bit memiliki 2 kemungkinan nilai, yaitu 0 dan 1. Kata "bit" berasal dari "*binary digit*", karena 0 dan 1 adalah digit-digit yang digunakan dalam merepresentasikan bilangan biner. John Tukey, seorang *statistician* memperkenalkan istilah ini pada tahun 1946. Suatu bit dapat digunakan untuk merepresentasikan nilai kebenaran, karena terdapat dua nilai kebenaran yaitu *true* dan *false*. Gunakan bit 1 untuk *true* dan bit 0 untuk *false*. Variabel Boolean juga direpresentasikan dalam bit (Rosen, 2007).

Operasi bit serupa dengan operasi logika, ganti *true* dengan bit 1 dan *false* dengan bit 0. Tabel nilai kebenaran untuk operator ∧, ∨, dan ⊕ dapat dilihat pada Tabel 2.7 untuk operasi bit. Gunakan notasi AND, OR, dan XOR untuk operator ∧, ∨, dan ⊕.

Tabel 2.6 Nilai Kebenaran Operator $\boldsymbol{AND}$, $\boldsymbol{OR}$, dan $\boldsymbol{XOR}$ (Rosen, 2007)

AND	0	1
0	0	0
1	0	1

OR	0	1
0	0	1
1	1	1

XOR	0	1
0	0	1
1	1	0

Contoh: $01101001\ AND\ 00000011 = 00000001$

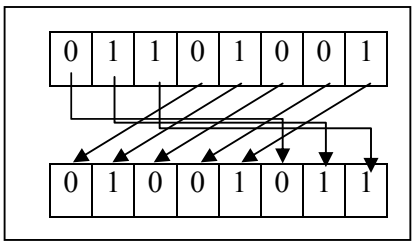
$01101001\ OR\ \ 00000011 = 01101011$

$01101001\ XOR\ 00000011 = 01101010$

Operator bit lainnya adalah operator *shift*. Operator *shift* terbagi dua, yaitu operator *left shift* (<<<) dan operator *right shift* (>>>). Operator *shift* melakukan operasi pergeseran deretan bit secara siklik. Operator *left shift* melakukan operasi pergeseran deretan bit secara siklik ke kiri sedangkan operator *right shift* melakukan operasi pergeseran deretan bit secara siklik ke kanan.

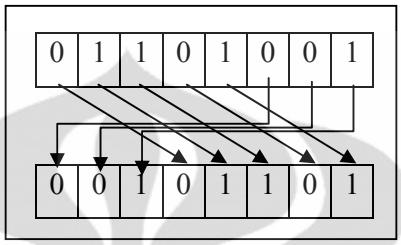
Berikut diberikan beberapa contoh perhitungan operasi *left shift* dan *right shift*.

- $01101001 <<< 00000011 = 01001011$, karena 00000011 adalah 3 dalam bilangan desimal, artinya pergeseran deretan bit 01101001 secara siklik ke kiri sebesar 3 menjadi 01001011 (lihat Gambar 2.2)



Gambar 2.2 Ilustrasi Operasi *Left Shift*

- $01101001 >>> 00000011 = 00101101$, karena 00000011 adalah 3 dalam bilangan desimal, artinya pergeseran deretan bit 01101001 secara siklik dilakukan ke kanan sebesar 3 menjadi 00101101 (lihat Gambar 2.3)



Gambar 2.3 Ilustrasi Operasi *Right Shift*

2.3.3 Operasi Aritmatika Modular

Operator $(mod\ m)$ merupakan *unary operator* yang memetakan bilangan bulat ke dalam himpunan bilangan bulat $\{0, 1, 2, \ldots, m-1\}$. Definisi mengenai operator $(mod\ m)$, dapat dilihat pada Definisi 2.9.

Definisi 2.9: (Rosen, 2007)

Misalkan a adalah sebuah bilangan bulat dan m adalah sebuah bilangan bulat positif. Notasi $a\ mod\ m$ merupakan sisa dari pembagian a oleh m.

Dengan perkataan lain $a\ mod\ m = r$, dengan r adalah bilangan bulat memenuhi $a = qm + r\ , 0 \leq r < m$. Berdasarkan Definisi 2.9, berikut diberikan beberapa contoh perhitungan dengan menggunakan *unary operator* modulo.

- $17\ \ mod\ \ 5\ = 2$
- $-133\ mod\ \ 9\ \ = 2$
- $2001\ \ mod\ 101\ = 82$

Definisi 2.10: (Rosen, 2007)

Jika a dan b adalah dua bilangan bulat, dan m adalah sebuah bilangan bulat positif, maka a dikatakan kongruen modulo m dengan b jika $(a - b)$ habis dibagi oleh m, ditulis a ≡ b (mod m)

Berdasarkan Definisi 2.10, berikut diberikan beberapa contoh kongruensi modulo.

- $17 \equiv 5 \ (mod\ 6)$ karena $17 - 5 = 12$ habis dibagi 6
- $21 \equiv -9 \ (mod\ 10)$ karena $21 - (-9) = 30$ habis dibagi 10

Selanjutnya, aturan operasi aritmatika modular dapat dilihat pada Teorema 2.4.

Teorema 2.4: (Stallings, 2005):

- $((a\ mod\ m) + (b\ mod\ m))mod\ m = (a + b)mod\ m$
- $((a\ mod\ m) - (b\ mod\ m))mod\ m = (a - b)mod\ m$
- $((a\ mod\ m)\ \text{x}\ (b\ mod\ m))mod\ m = (a\ \text{x}\ b)\ mod\ m$

Bukti mengenai Teorema 2.4 dapat dilihat pada Lampiran 4.

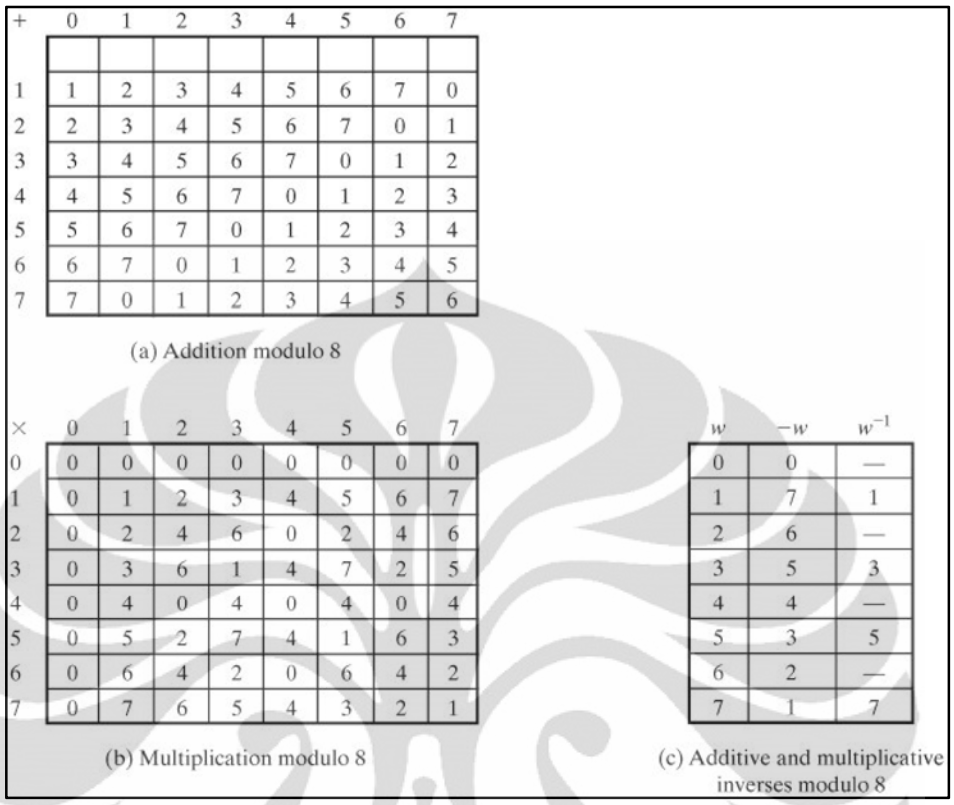
BerdasarkanTeorema 2.4, berikut diberikan beberapa c*onto*h perhitungan operasi aritmatika modular.

$15\ mod\ 8 = 7$ dan $11\ mod\ 8 = 3$

$(15\ mod\ 8\ -\ 11\ mod\ 8)\ mod\ 8\ =\ (7 - 3)\ mod\ 8\ =\ 4\ mod\ 8\ =\ 4$
$(15\ mod\ 8\ -\ 11\ mod\ 8)\ mod\ 8\ =\ (15 - 11)\ mod\ 8\ =\ 4\ mod\ 8\ =\ 4$

$((15\ mod\ 8)\ \text{x}\ (11\ mod\ 8))\ mod\ 8\ =\ (7\text{x}3)\ mod\ 8\ =\ 21\ mod\ 8\ =\ 5$
$((15\ mod\ 8)\ \text{x}\ (11\ mod\ 8))\ mod\ 8\ =\ (15\text{x}11)\ mod\ 8\ =\ 165\ mod\ 8\ =\ 5$

Operasi aritmatika modulo 8 dapat dilihat pada Gambar 2.4. Gambar 2.4 (a) menunjukkan operasi penjumlahan modulo 8, Gambar 2.4 (b) menunjukkan operasi perkalian modulo 8. Gambar 2.4 (c) mengindikasikan invers penjumlahan modulo 8, karena $0 + 0 = 0$, $1 + 7 = 0$, $2 + 6 = 0$, $3 + 5 = 0$, $4 + 4 = 0$, $5 + 3 = 0$, $6 + 2 = 0$,$7 + 1 = 0$ maka invers penjumlahan dari 0 adalah 0,invers penjumlahan dari 1 adalah 7 dan seterusnya.



+	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0
2	2	3	4	5	6	7	0	1
3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

(a) Addition modulo 8

×	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1

(b) Multiplication modulo 8

w	$-w$	w^{-1}
0	0	—
1	7	1
2	6	—
3	5	3
4	4	—
5	3	5
6	2	—
7	1	7

(c) Additive and multiplicative inverses modulo 8

Gambar 2.4 Operasi Aritmatika Modulo 8 (Stallings, 2005)

2.3.4 Operasi Aritmatika Biner

Bilangan yang biasa digunakan manusia adalah bilangan decimal (basis 10) dengan setiap digitnya adalah dari himpunan $\{0, 1, 2, \ldots, 9\}$. Dalam basis 10, setiap posisi dari bilangan desimal dikalikan10 dengan pangkat yang terus naik jika perpindahan dilakukan dari kanan ke kiri. Penomoran posisi dari kanan ke kiri dimulai dari 0.

Contoh $4096 = 4 \times 10^3 + 0 \times 10^2 + 9 \times 10^1 + 6 \times 10^0$

Secara umum, jika n digit dari suatu bilangan desimal adalah

$d[n-1]d[n-2]d[n-3]\ldots d[2]d[1]d[0]$, maka bilangan yang direpresentasikan oleh desimal (basis 10) tersebut adalah

$$d[n-1] \times 10^{n-1} + d[n-2] \times 10^{n-2} + d[n-3] \times 10^{n-3} + \ldots + d[2] \times 10^2 + d[1] \times 10^1 + d[0] \times 10^0$$

Dalam biner, basisnya adalah 2, maka digit-digitnya adalah 0 dan 1 serta perkalian tiap posisi oleh 2^i (i adalah posisi digit). Digit-digit ini kelak disebut bit. Contoh bilangan biner 1101 adalah $1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 8 + 4 + 1 = 13$.

Penjumlahan biner dinotasikan dengan "+" memiliki aturan dasar $0 + 0 = 0$, $0 + 1 = 1$, $1 + 0 = 1$, dan $1 + 1 = 0$ (simpan 1). Pengurangan biner dinotasikan dengan "–" memiliki aturan dasar $0 - 0 = 0$, $1 - 0 = 1$, $1 - 1 = 0$, dan $0 - 1 = 1$ (pinjam 1). Perkalian biner dinotasikan dengan "x" dapat dilakukan dengan cara yang sama dengan perkalian bilangan desimal yaitu $0x0 = 0$, $1x0 = 0$, $0x1 = 0$, dan $1x1 = 1$.

Banyaknya digit bilangan biner biasanya adalah kelipatan 8. Barisan 8 bit suatu bilangan biner disebut *byte*. Setiap register pada RC6 memiliki panjang bit yang tetap. Ukurannya sesuai dengan ukuran register w yang telah ditentukan yaitu 32 bit. Jadi saat melakukan operasi aritmatika, ada hasil yang dibuang. Berikut adalah contoh operasi penjumlahan dengan n=4 bit.

1101
0101 +
0010 (*carry register* dibuang)

Jadi contoh ini menunjukkan $13 + 5 = 2$. Memang terlihat aneh, tapi ini disebut aritmatika modular dengan $modulo\ m = 2^n = 2^4 = 16$, sehingga $(13 + 5)\ mod\ 16 = 2$.

Oleh karena itu, bekerja dengan n-bit suatu bilangan biner hanyalah melakukan aritmatika modular $m = 2^n$. Artinya nilai terbesar yang bisa direpresentasikan adalah dalam n bit adalah $2^n - 1$. Tiap register yang dioperasikan pada algoritma RC6 berisi 32 bit sehingga nilai terbesar registernya adalah $2^n - 1 = 4294967295$.

Kombinasi dari operasi penjumlahan dan perkalian pada aritmatika dapat membentuk fungsi kuadrat. Umumnya, fungsi kuadrat berbentuk $f(x) = ax^2 + bx + c$ dengan $a, b, c \in R$ dan $a > 0$. Sedangkan fungsi kuadrat modular adalah

dengan menambahkan operator modulo sehingga fungsi kuadrat menjadi berbentuk $f(x) = ax^2 + bx + c \; mod \; m$ dengan $a, b, c \in R$ dan $a, m > 0$.

Kembali pada Definisi 2.1, fungsi 1-1 adalah fungsi yang memetakan elemen domain ke tepat satu dan hanya satu elemen kodomain. Dalam hal ini, fungsi kuadrat modular termasuk dalam fungsi injektif jika sesuai dengan Lemma 2.2.

Lemma 2.2: (Contini,1998)

Diberikan $f(x) = x(ax + b) mod \; m$ dengan a genap, b ganjil $m = 2^w$,

maka $f(x)$adalah pemetaan 1-1 dari $\{0,1\}^w$ ke $\{0,1\}^w$

Bukti: (dengan kontradiksi)

Diberikan $x_1, \; x_2 \in \{0,1\}^w$

Andaikan $x_1 \neq x_2$ maka $f(x_1) = f(x_2)$. Maka,

$$ax_1^2 + bx_1 = ax_2^2 + bx_2 \; mod \; m$$

$$ax_1^2 + bx_1 - ax_2^2 - bx_2 \; mod \; m = 0$$

$$ax_1^2 - ax_2^2 + bx_1 - bx_2 \; mod \; m = 0$$

$$a(x_1^2 - x_2^2) + b(x_1 - x_2) \; mod \; m = 0$$

$$(x_1 - x_2)(a(x_1 + x_2) + b) \; mod \; m = 0$$

Sehingga didapat $(x_1 - x_2)(ax_1 + ax_2 + b) \; mod \; m = 0$ (2.6)

Karena $(ax_1 + ax_2 + b)$ ganjil agar memenuhi persamaan 2.6 maka $(x_1 - x_2)$ merupakan perkalian dari $m = 2^w$, sehingga diperoleh $x_1 = x_2$. Hal ini kontradiksi dengan pemisalan. Oleh karena itu terbukti bahwa $f(x)$ adalah fungsi satu-satu dari $\{0,1\}^w$ke $\{0,1\}^w$.

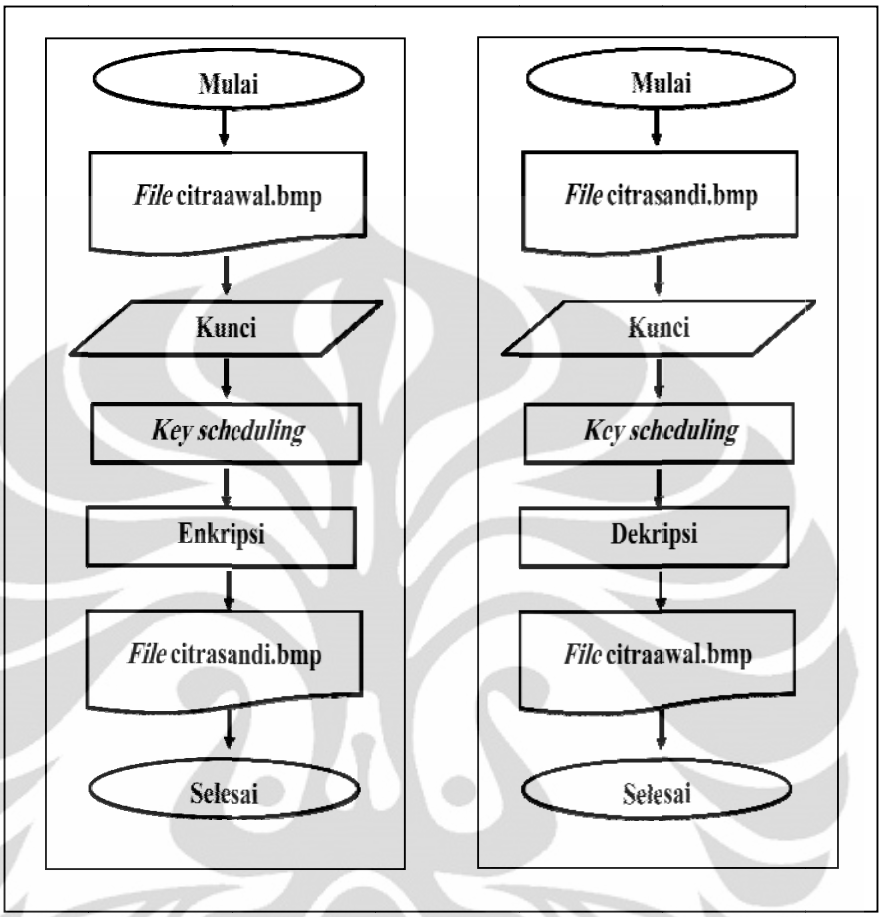
BAB 3
KONSEP ALGORITMA RC6

Algoritma RC6 merupakan algoritma yang menggunakan kunci simetris (kunci enkripsi sama dengan kunci dekripsi) didesain oleh Ronald Linn Rivest, Matt J.B Robshaw, Ray Sydney, dan Yuqin Lisa Yin dari Laboratorium RSA. RC adalah singkatan dari 'Rivest Code' sedangkan 6 berarti algoritma ini merupakan pengembangan dari algoritma sebelumnya.

Letak perbedaan RC6 dari varian-varian sebelumnya adalah adanya langkah transformasi dan perbedaan banyak register. Register diperlukan untuk penempatan blok data yang diolah pada algoritma sehingga algoritma mengolah data pada tiap register yang ada. Algoritma RC5 membagi *plaintext* ke dalam 2 register, lain halnya dengan RC6 yang membagi *plaintext* ke dalam 4 register. Sedangkan algoritma RC4 bukanlah algoritma *block cipher* (algoritma penyandian yang mengolah data per blok) seperti RC5 dan RC6 melainkan algoritma *stream cipher* (algoritma penyandian yang mengolah data per bit).

RC6 mempunyai tiga parameter, sehingga dituliskan sebagai RC6-$w/r/b$. Ketiga paramater tersebut berarti ukuran register w (32, 64 atau 128 bit), banyak iterasi r yang tidak boleh bernilai negatif, dan panjang kunci b. Saat algoritma ini masuk kandidat AES, ditetapkan nilai $w = 32$ bit, $r = 20$, dan b bervariasi antara 16, 24, dan 32 *byte*. Pemilihan $w = 32$ bit mengindikasikan bahwa operasi-operasi yang ada pada algoritma RC6 seperti penjumlahan, pengurangan, perkalian, XOR, dan *shift* dilakukan pada *operand* berupa tepat 32 bit bilangan biner.

Selanjutnya akan dijelaskan langkah-langkah penyandian dan pengembalian sandi (lihat Gambar 3.1) yang terdapat pada algoritma RC6 yaitu algoritma *key scheduling*, enkripsi, dan dekripsi.



Gambar 3. 1 Proses Enkripsi dan Dekripsi Citra Digital pada Algoritma RC6

3.1 Algoritma *Key Scheduling*

Sebelum melakukan proses enkripsi dan dekripsi, dibutuhkan suatu *key scheduling* untuk mendapatkan kunci-kunci yang akan digunakan dalam proses enkripsi dan dekripsi tersebut.

Algoritma *key scheduling* RC6 terdiri dari 3 tahap utama, yaitu:

a) Penempatan kunci yang di-*input* pengguna kedalam *array L*

b) Inisialisasi kunci yang ditempatkan dalam *array S*

c) Kombinasi *L* dan *S*

Kunci yang di-*input* pengguna $K[0..b-1]$ memiliki panjang sebesar b *byte*. Besar b bervariasi yaitu 16, 24, atau 32 *byte*. Kemudian kunci dimasukkan kedalam *array* berukuran c *word* $L[0,1,\ldots,c-1]$ dengan nilai $c = \lceil \frac{b}{4} \rceil$. Misal $b = 16$ *byte*, maka algoritma RC6 menyiapkan *array* $L[0,1,2,3]$ karena panjang kunci sebesar 16 *byte* terdiri dari 4 *word* (1 $word = 32\ bit = 4$ *byte*) untuk penempatan kunci yang dimasukkan oleh pengguna. Proses tersebut dapat dilakukan dengan menggunakan Algoritma 3.1.

Algoritma 3.1

```
{penempatan kunci pengguna}
for i=b-1 downto 0
      L[i/u)]=L[i/u<<<8]+K[i]
end
```

Tabel 3. 1 Penempatan kunci yang dimasukkan oleh pengguna pada *array* ***L***

Kunci	Bilangan ASCII	Bilangan Biner	*Array*
M	77	01001101	$L[0]$
A	65	01000001	01000101010101000100000101001101
T	84	01010100	
E	69	01000101	E T A M
M	77	01001101	$L[1]$
A	65	01000001	01001001010101000100000101001101
T	84	01010100	I T A M
I	73	01001001	
K	75	01001011	$L[2]$
A	65	01000001	01010101001000000100000101001011
[spasi]	32	00100000	U [spasi] A K
U	85	01010101	
I	73	01001001	$L[3]$
0	0	00000000	00000000000000000000000001001001
0	0	00000000	0 0 0 I
0	0	00000000	

Misal: kunci yang di-*input* pengguna sepanjang 13 *byte* dengan kata kunci "MATEMATIKA UI", maka tambahkan 3 *byte* 0 ke dalam kunci tersebut menjadi "MATEMATIKA UI000" sehingga panjang kunci menjadi 16 *byte*. Lalu, masing-masing karakter diubah ke dalam bilangan ASCII dan diubah ke dalam bilangan biner 8 bit. Kemudian, setiap 32 bitnya dimasukkan ke dalam array $L[0]$, $L[1]$, $L[2]$, dan $L[3]$ sesuai dengan aturan penempatan yang berlaku (Lihat Tabel 3.1).

Penempatan kunci pengguna ke dalam *array* $L[0,1,2,3]$ adalah dengan menempatkan *byte* pertama dari kunci ke *byte* paling kanan (*least significant*) dari $L[0]$, lalu *byte* berikutnya di tempatkan di sisi kiri dari *byte* sebelumnya pada $L[0]$ sampai dengan $L[0]$ terisi sebanyak 4 *byte*. *Byte* kelima dari kunci diletakkan pada *byte* paling kanan(*least significant*) $L[1]$, lalu *byte* berikutnya ditempatkan di sisi kiri *byte* sebelumnya pada $L[1]$ sampai dengan $L[1]$ terisi sebanyak 4 *byte*. Begitu pula penempatan *byte* pada $L[2]$ dan $L[3]$ sehingga *byte* terakhir pada kunci yang di-*input* pengguna terdapat pada *byte* paling kiri (*most significant*) $L[3]$. Jika kunci yang dimasukkan kurang dari 16 *byte*, maka tambahkan bit-bit 0 sehingga panjang kunci menjadi 16 *byte* seperti pada c*onto*h (lihat Tabel 3.1).

Kemudian inisialisasi kunci S menggunakan *magic constant* P_w dan Q_w sesuai Algoritma 3.2. Proses untuk membangun kunci inisialisasi tersebut didapat dari dua buah *magic constant* P_w dan Q_w. Beberapa nilai *magic constant* dalam heksadesimal pada beberapa ukuran register (w) disajikan dalam Tabel 3.2.

Tabel 3. 2 *Magic Constant* (Wisnu, 2010)

w	P_w	Q_w
16	b7e1	9e37
32	b7e15163	9e3779b9
64	b7e151628aed2a6b	9e3779b97f4a7c15

Algoritma 3.2

```
{Inisialisasi kunci S[i}]
S[0]=Pw
for  i=1 to 2r+3
      S[i]=S[i-1]+ Qw
end
```

Selanjutnya adalah proses kombinasi kunci yang di-*input* pengguna L dengan kunci inisialisasi S dilakukan berdasarkan algoritma *key scheduling* RC6 yang dapat dilihat pada Algoritma 3.3 menghasilkan 44 kunci ronde yang ditempatkan pada array $S[0,1,\dots,43]$.

Algoritma 3.3

```
{Key Scheduling RC6}
Input       :     array L[0,1,2,…,c-1]
Output      :     array S[0,1,2,…,2r+4]
Prosedur    :
S[0]=Pw
for i=1 to 2r+3
      S[i]=S[i-1]+ Qw
end

A=B=i=j=0
V=3*max(c,2r+4)
for s=1 to v
      S[i]=(S[i]+A+B)<<<3
      A=S[i]
      L[j]=(L[j]+A+B)<<<(A+B)
      B=L[j]
      i=i+1 mod 2r+4
      j=j+1 mod c
end
```

Proses kombinasi kunci pengguna L dengan hasil kunci inisialisasi S menjadi kunci-kunci ronde ditampung dalam *array* $S[0,1,2,\dots,2r+3]$ dengan panjang masing-masing kunci ronde 32 bit. Kunci insialisasi $S[i]$ ditambahkan dengan nilai A dan B. Pada langkah awal, nilai A dan B adalah nol, lalu hasilnya digeser sebesar 3 bit ke kiri, kemudian nilai $S[i]$ yang baru disimpan di A. $L[j]$ ditambahkan dengan nilai A dan B, lalu hasilnya digeser ke kiri sebesar 5 bit dari nilai penjumlahan A dan B. Langkah ini memperlihatkan kunci pengguna telah dikombinasikan dengan kunci inisialisasi. Iterasi dilakukan sebanyak tiga kali nilai maksimum$\{c, 2r+4\}$.

3.2 Algoritma Enkripsi

Algoritma RC6 merupakan algoritma *block cipher*. Blok data yang diolah sebesar 128 bit dikelompokkan menjadi 4 buah blok. Masing-masing blok data

berukuran 32 bit tersebut kemudian disimpan dalam register A, B, C, dan D lalu dioperasikan dengan kunci-kunci berukuran 32 bit pula yang telah diperoleh dari algoritma *key scheduling* RC6 (subbab 3.1) .

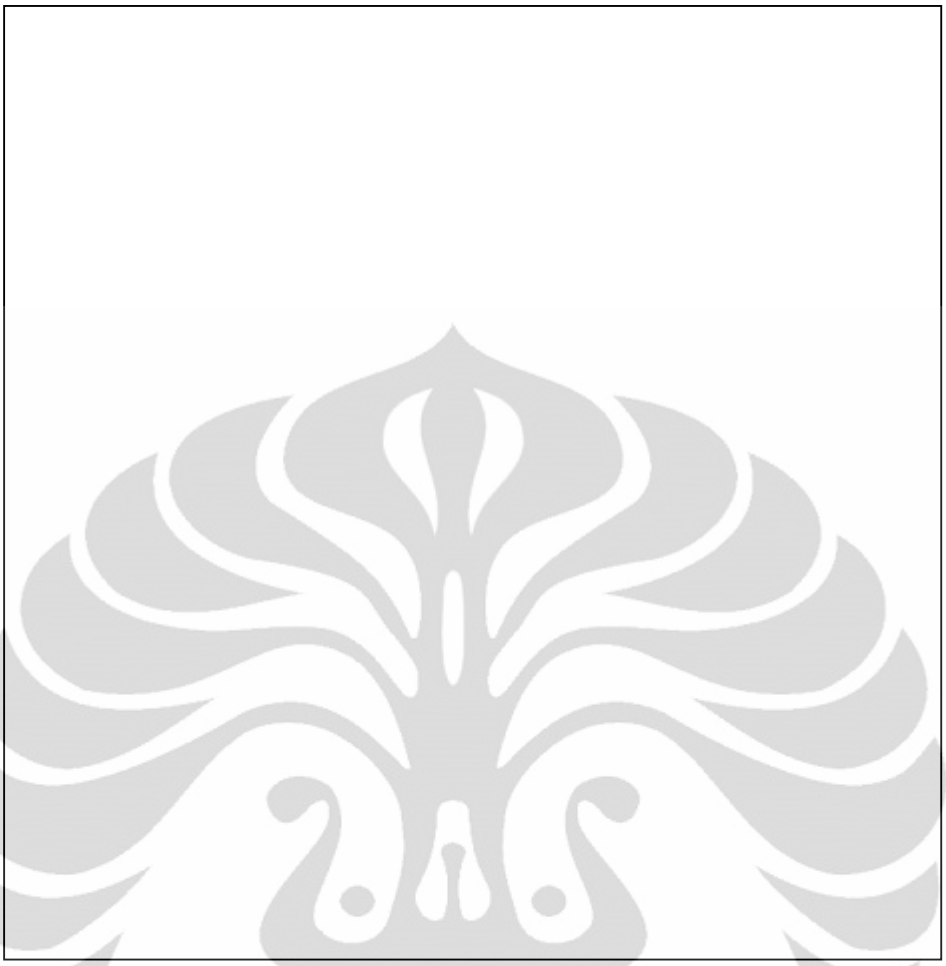
Penempatan *plaintext* tiap blok data 16 *byte* ke dalam 4 register A,B,C, dan D serupa dengan penempatan kunci pengguna yang dimasukkan ke dalam *array* $L[0,1,2,3]$. Aturan penempatannya adalah dengan meletakkan *byte* pertama dari blok data ke *byte* paling kanan(*least significant*) dari register A, lalu *byte* berikutnya ditempatkan di sisi kiri *byte* sebelumnya pada register A sampai dengan register A terisi sebanyak 4 *byte*. Begitupula dengan penempatan *byte* pada register B, C dan D sehingga *byte* terakhir blok data terdapat pada *byte* paling kiri (*most significant*) register D.

Secara umum enkripsi dengan menggunakan algoritma RC6 berdasarkan algoritma 3.4 memiliki langkah-langkah terurut sebagai berikut:

- *Whitening* awal
- Transformasi
- *Mixing*
- *Swap* register
- *Whitening* akhir

Algoritma 3.4

```
{Enkripsi RC6}
Input       :     plaintext dalam register [A,B,C,D]
                  r jumlah iterasi
Output      :     ciphertext dalam register [A,B,C,D]
Prosedur
B = B + S[0]                                        } whitening
D = D + S[1]                                          awal
for i=1 to r
      t= (  B x ( 2B + 1 )) <<< 5                   } transformasi
      u= (  D x ( 2D + 1 ))<<< 5
      A = ((A ⊕  t) <<< u ) + S[2i]                 } mixing
      C = ((C ⊕  u) <<< t ) + S[2i+1]
      ( A, B, C, D ) = ( B, C, D, A )               } swap register
end
A = A +  S[2r+2]                                    } whitening
C = C +  S[2r+3]                                      akhir
```



Gambar 3. 2 *Flowchart* Enkripsi RC 6

Langkah pertama yang dilakukan algoritma enkripsi RC 6 adalah *whitening* awal, yakni menambahkan kunci ronde pertama dengan register B dan kunci ronde kedua dengan register D. Tujuan *whitening* awal adalah untuk untuk menyamarkan *input* sebelum iterasi pertama.

Langkah kedua adalah transformasi yang dilakukan dengan menggunakan fungsi kuadrat $f(x) = x(2x+1)(mod\ 2^{32})$ pada register B dan D diikuti pergeseran siklik ke kiri sebesar 5 bit. Fungsi $f(x) = x(2x+1)(mod\ 2^{32})$ adalah fungsi satu-satu (Lemma 2.2), sehingga penggunaan fungsi $f(x)$ pada proses transformasi ini menjamin tiap nilai x menghasilkan suatu nilai $f(x)$. Setelah pergeseran sebesar 5 bit, simpan nilai yang didapat dalam variabel t dan u untuk dapat melakukan langkah selanjutnya.

Langkah ketiga adalah *mixing*, yakni pencampuran isi keempat register yang dilakukan dengan melakukan operasi XOR terhadap register A dan C dengan

variabel t dan u yang telah dibahas sebelumnya. Lalu, hasil dari operasi XOR ini digeser ke kiri sebesar nilai variabel u dan t, barulah ditambahkan kunci-kunci ronde yang bersesuaian. Di langkah *mixing* ini terlihat bahwa keempat register yang berisi blok data saling bercampur.

Langkah keempat adalah *swap* register, yakni melakukan permutasi antar register atau mengubah urutan register dari (A, B, C, D) menjadi (B, C, D, A).

Langkah terakhir adalah *whitening* akhir, yakni menambahkan kunci ronde ke-42 dengan register A dan kunci ronde ke-43 dengan register C. *Whitening* akhir bertujuan untuk menyamarkan *output* pada iterasi terakhir.

3.3 Algoritma Dekripsi

Setelah melakukan penyandian menggunakan algoritma enkripsi RC6, pengguna butuh mengembalikan *ciphertext* menjadi *plaintext*. Untuk itu, dilakukan proses dekripsi menggunakan algoritma dekripsi RC6.

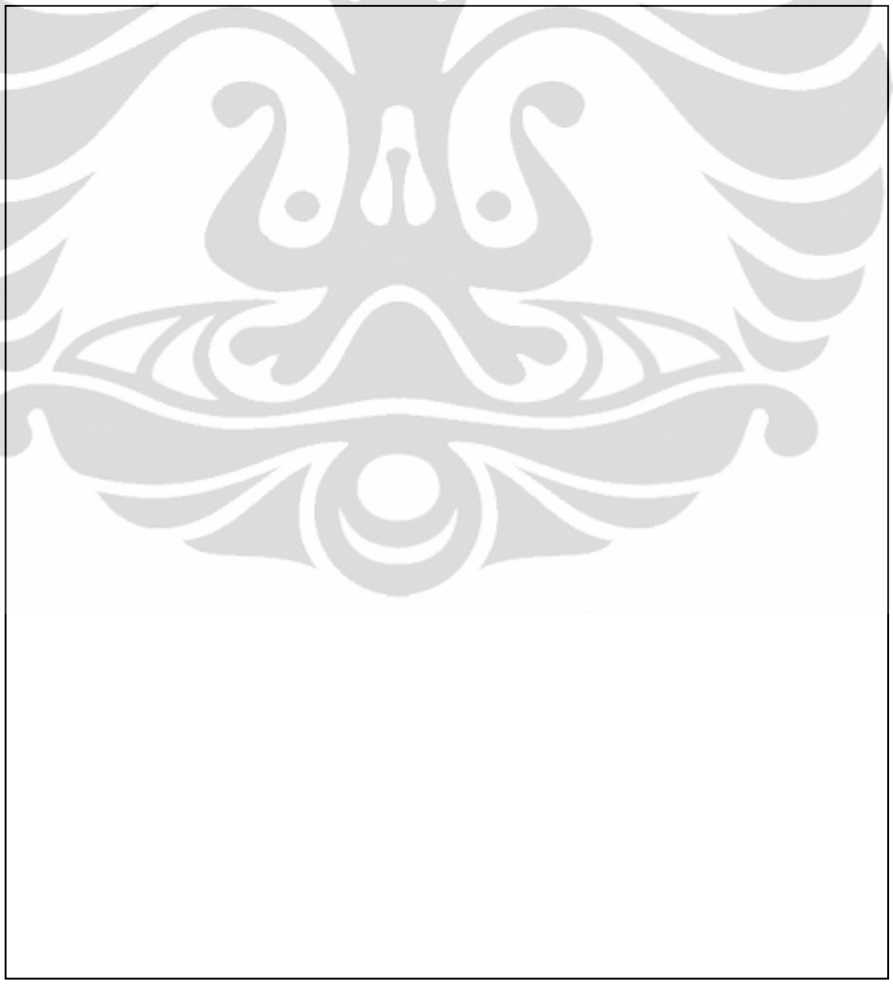
Algoritma dekripsi adalah invers atau kebalikan dari algoritma enkripsi. Operator yang ada dalam algoritma dekripsi adalah kebalikan dari operator dalam algoritma enkripsi. Blok data yang diolah sama dengan blok data pada proses enkripsi, yaitu sebesar 128 bit yang dibagi ke dalam 4 register. Bedanya, *input* dari algoritma dekripsi adalah *ciphertext* yang akan mengeluarkan *output plaintext*. Masing-masing blok data *ciphertext* berukuran 32 bit kemudian juga disimpan dalam register A, B, C, dan D lalu dioperasikan dengan kunci-kunci berukuran 32 bit pula yang telah dibangun pada algoritma *key scheduling* RC6.

Kunci dan banyaknya iterasi yang digunakan pada proses dekripsi sama dengan yang digunakan pada proses enkripsi. Langkah-langkah yang dilakukan pada proses dekripsi sama dengan proses enkripsi, hanya saja urutannya dibalik bagaikan mereka ulang kejadian yang dialami oleh blok data *plaintext* yang telah menjadi *ciphertext* (Lihat Gambar 3.3). Jadi, langkah pertama pada proses dekripsi adalah langkah terakhir pada proses enkripsi.

Langkah-langkah tersebut adalah *whitening* akhir, *swap* register, transformasi, *mixing*, dan *whitening* awal yang dapat dilihat pada Algoritma 3.5.

Algoritma 3.5

```
{dekripsi RC6}
Input       :     ciphertext dalam register [A,B,C,D]
                  r jumlah iterasi
Output      :     plaintext dalam register [A,B,C,D]
Prosedur
C = C – S[2r+3]                                        } whitening
A = A – S[2r+2]                                          akhir
for i=r down to 1
      (A, B, C, D) = (D, A, B, C)                      } swap register
      u = (D x (2D + 1)) <<<5                          } transformasi
      t = (B x (2B + 1)) <<<5
      C = ((C – S[2i+1]) >>> t) ⊕ u                    } mixing
      A = ((A – S[2i])   >>> u) ⊕ t
end
D = D – S[1]                                           } whitening awal
B = B – S[0]
```



Langkah pertama algoritma dekripsi RC6 adalah w*hitening* akhir, yakni mengembalikan proses *whitening* akhir pada proses enkripsi dengan mengurangi register A dan C dengan kunci yang bersesuaian (sama dengan yang dipakai pada saat proses *whitening* akhir enkripsi).

Langkah kedua adalah s*wap* register, yakni mengembalikan proses s*wap* register yang dilakukan pada saat enkripsi dari (B, C, D, A) kembali menjadi (A, B, C, D).

Langkah ketiga adalah transformasi sama halnya dengan langkah transformasi pada proses enkripsi, dilakukan pada register B dan D serta hasilnya disimpan pada variabel yang sama pula, yaitu t dan u.

Langkah keempat adalah m*ixing*, yakni mengurangi isi register C dan A dengan kunci ronde yang bersesuaian, lalu masing-masing digeser ke kanan sebesar 5 bit terakhir nilai variabel t dan u, kemudian dilakukan operasi XOR dengan nilai variabel u dan t.

Langkah terakhir adalah *whitening* awal, yakni mengembalikan proses *whitening* awal pada enkripsi dengan mengurangi register B dan D dengan kunci yang bersesuaian (sama dengan yang dipakai saat proses *whitening* awal enkripsi).

BAB 4
IMPLEMENTASI ALGORITMA RC6 PADA CITRA DIGITAL

Bab ini akan menjelaskan proses pengamanan data berupa citra digital menggunakan algoritma RC6. Dimulai dari *key scheduling*, proses penyandian citra digital tersebut (enkripsi), dan pengembalian data citra digital yang telah tersandi menjadi data citra digital semula (dekripsi).

4.1 *Key Scheduling*

Sesuai dengan Algoritma *key scheduling* pada subbab 3.1, langkah pertama dalam melakukan *key scheduling* adalah penempatan kunci pengguna (Tabel 3.2). Selanjutnya inisialisasi kunci berdasarkan Algoritma 3.2 menggunakan P_{32} dan Q_{32} sehingga didapat kunci inisialisasi dalam biner yang ditempatkan dalam *array S* seperti pada Tabel 4.1.

Tabel 4. 1 Kunci Inisialisasi

i	$S[i]$
0	10110111111000010101000101100011
1	01010110000110001100101100011100
2	11110100010100000100010011010101
3	10010010100001111011111010001110
4	00110000101111110011100001000111
5	11001110111101101011001000000000
6	01101101001011100010101110111001
7	00001011011001011010010101110010
8	10101001100111010001111100101011
9	01000111110101001001100011100100
10	11100110000011000001001010011101
11	10000100010000111000110001010110
12	00100010011110110000011000001111
13	11000000101100100111111111001000
14	01011110111010011111100110000001
15	11111101001000010111001100111010

Tabel 4.1 Kunci Inisialisasi

i	$S[i]$
16	10011011010110001110110011110011
17	00111001100100000110011010101100
18	11010111110001111110000001100101
19	01110101111111110101101000011110
20	00010100001101101101001111010111
21	10110010011011100100110110010000
22	01010000101001011100011101001001
23	11101110110111010100000100000010
24	10001101000101001011101010111011
25	00101011010011000011010001110100
26	11001001100000111010111000101101
27	01100111101110110010011111100110
28	00000101111100101010000110011111
29	10100100001010100001101101011000
30	01000010011000011001010100010001
31	11100000100110010000111011001010
32	01111110110100001000100010000011
33	00011101000010000000001000111100
34	10111011001111110111101111110101
35	01011001011101101111010110101110
36	11110111101011100110111101100111
37	10010101111001011110100100100000
38	00110100000111010110001011011001
39	11010010010101001101110010010010
40	01110000100011000101011001001011
41	00001110110000111101000000000100
42	10101100111110110100100110111101
43	01001011001100101100001101110110

Setelah mendapatkan kunci-kunci inisialisasi yang ditempatkan dalam *array S* (Tabel 4.1), algoritma RC6 mengkombinasikan hasil inisialisasi kunci *S* dengan kunci pengguna *L*. Setelah melakukan kombinasi sesuai dengan algoritma *key scheduling* RC6, didapatkan 44 buah kunci ronde ditempatkan pada *array S* (lihat Tabel 4.2) yang akan digunakan dalam algoritma enkripsi maupun dekripsi RC6.

Tabel 4. 2 Kunci Ronde

i	$S[i]$
0	10111111000010101000101100011101
1	11010011101111001011110000110010
2	10100010000111101111000111101100
3	01000100000010110010001101111010
4	00000101111011010100000010110111
5	01101011100100110110101100000001
6	00000100100011001001101100111100
7	11010001100001001101011001110111
8	11010000011101010010011111100110
9	00110100001101010110001010101101
10	00011101111101100010101010110110
11	01010100111110101110011110100101
12	00001000011001110000111101101010
13	00010000100110001011001111100101
14	01111100101010101100010110011111
15	11011110000111101000000000111111
16	00101000110100100110100000110110
17	01101011001100100011110001111011
18	01100010010110010111001100011001
19	00010110000011011000100001001010
20	11100111111100000001010000111001
21	01010010110111100111111100100000
22	11101100111100101101010000101110
23	11001111100011010001001111000010
24	01100011111001001001110010101110
25	01101010000111100000110001010001
26	01100011101001000110101110001110
27	01100011100011001110001101001010
28	01111001100110010101101010000100
29	10011110000010001001100000101101
30	00111100010010111100001011011010
31	11101011101110010011100001101001
32	10001011110101000010101001110010
33	11010110011001110110101101101100
34	01010101001000111010110010011111
35	01111101110111111000100000111000
36	11010111110101000000110010110101
37	10111011101111111101000100101100
38	01100101001110111111001001110010

Tabel 4.2 Kunci Ronde

i	$S[i]$
39	01111100111101000010001001111110
40	01101001010011010110111001001110
41	00000100011011010010010000100011
42	01100100001101000001110101101010
43	01010101000101101010110001011001

4.2 Enkripsi Citra Digital

Untuk dapat mengenkripsi suatu citra digital, representasikan citra digital tersebut dalam bentuk matriks seperti pada persamaan (2.1) sesuai dengan ukuran *pixel* citra yang bersangkutan. Berikut diberikan contoh untuk memperjelas proses enkripsi suatu citra digital.

Misal: Citra yang akan dienkripsi berukuran 16x17 *pixel* (Gambar 4.1)

Gambar 4. 1 *Plaintext*

Kemudian, ubah representasi citra dalam matriks ukuran 16 x 17.

$$\begin{bmatrix}
157 & 172 & 85 & 99 & 127 & 128 & 138 & 123 & 136 & 126 & 122 & 161 & 157 & 148 & 109 & 136 & 54 \\
169 & 157 & 76 & 106 & 124 & 132 & 136 & 172 & 200 & 123 & 114 & 148 & 147 & 140 & 224 & 50 & 47 \\
85 & 153 & 85 & 99 & 121 & 116 & 132 & 145 & 217 & 218 & 81 & 153 & 120 & 146 & 48 & 52 & 160 \\
91 & 150 & 74 & 102 & 172 & 122 & 116 & 166 & 191 & 203 & 205 & 153 & 218 & 225 & 51 & 140 & 159 \\
100 & 150 & 76 & 96 & 169 & 102 & 146 & 173 & 156 & 187 & 176 & 201 & 204 & 62 & 51 & 160 & 157 \\
103 & 157 & 70 & 98 & 205 & 132 & 133 & 101 & 111 & 192 & 187 & 203 & 120 & 45 & 139 & 159 & 153 \\
104 & 151 & 70 & 102 & 95 & 129 & 55 & 87 & 128 & 164 & 206 & 62 & 52 & 46 & 148 & 157 & 152 \\
101 & 150 & 77 & 99 & 168 & 52 & 53 & 114 & 124 & 87 & 193 & 98 & 66 & 138 & 157 & 155 & 134 \\
97 & 147 & 85 & 141 & 96 & 88 & 50 & 78 & 150 & 142 & 190 & 83 & 46 & 158 & 151 & 131 & 189 \\
83 & 142 & 69 & 68 & 70 & 119 & 185 & 103 & 154 & 121 & 230 & 40 & 41 & 153 & 146 & 193 & 208 \\
63 & 153 & 70 & 67 & 78 & 180 & 44 & 83 & 145 & 103 & 112 & 70 & 130 & 146 & 140 & 207 & 215 \\
53 & 145 & 60 & 49 & 109 & 51 & 51 & 60 & 126 & 166 & 63 & 60 & 163 & 148 & 126 & 214 & 129 \\
148 & 129 & 145 & 96 & 51 & 53 & 60 & 57 & 133 & 148 & 188 & 207 & 150 & 136 & 111 & 70 & 95 \\
47 & 121 & 71 & 40 & 51 & 86 & 64 & 128 & 145 & 144 & 171 & 211 & 145 & 134 & 210 & 114 & 86 \\
91 & 101 & 99 & 51 & 56 & 127 & 55 & 99 & 140 & 137 & 156 & 196 & 108 & 45 & 117 & 80 & 59 \\
49 & 109 & 61 & 56 & 79 & 69 & 114 & 130 & 140 & 144 & 152 & 174 & 138 & 126 & 86 & 119 & 104
\end{bmatrix}$$

Untuk melakukan enkripsi citra digital dengan algoritma RC6, ambil *plaintext* per *16 byte* blok data per kolomnya. Jika pada *block cipher* terakhir kurang dari 16 byte, maka tambahkan *dummy* 0 sehingga *block cipher* menjadi 16 *byte*. Berikut langkah demi langkah algoritma RC6 dalam enkripsi 16 *byte* pertama dari citra pada Gambar 4.1:

a) Tempatkan 16 *byte* pertama *plaintext*

157 169 85 91 100 103 104 101 97 83 63 53 148 47 91 49

ke dalam 4 buah register A, B, C, dan D (Lihat Tabel 4.3)

Tabel 4. 3 Penempatan Blok Data *Plaintext* pada keempat Register

Blok data	Bilangan Biner	Register
157	10011101	A 01011011101010101101010011001110 1
169	10101001	
85	01010101	
91	01011011	
100	01100100	B 01100101011010000110011101100100
103	01100111	
104	01101000	
101	01100101	
97	01100001	C 00110101001111110101001101100001
83	01010011	
63	00111111	
53	00110101	
148	10010100	D 00110001010110110010111110010100
47	00101111	
91	01011011	
49	00110001	

b) Lakukan langkah *whitening* awal:

$B = B + S[0]$

$$\begin{aligned} & 01100101011010000110011101100100 \\ & \underline{10111111000010101000101100011101} \ + \\ = \ & 00100100011100101111001010000001 \end{aligned}$$

$D = D + S[1]$

$$\begin{aligned} & 00110001010110110010111110010100 \\ & \underline{11010011101111001011110000110010} \ + \\ = \ & 00000101000101111110101111000110 \end{aligned}$$

Kemudian, lakukan langkah transformasi, *mixing*, dan *swap register* sebanyak $r = 20$ iterasi. Berikut adalah langkah transformasi, *mixing*, dan *swap register* pada iterasi pertama.

$t = B\text{x}(2B + 1) <<< 5$

$$\begin{aligned} & 00100100011100101111001010000001 \\ & \underline{01001000111001011110010100000011} \ \text{x} \\ = \ & 01110101101010110011110010000011 <<< 5 \\ = \ & 10110101011001111001000001101110 \end{aligned}$$

$u = D\text{x}(2D + 1) <<< 5$

$$\begin{aligned} & 00000101000101111110101111000110 \\ & \underline{00001010001011111101011110001101} \ \text{x} \\ = \ & 11110001101110001010110011101010 <<< 5 \\ = \ & 00110111000101011001110101011110 \end{aligned}$$

$A = (A \oplus t) <<< u + S[2]$

$$\begin{aligned} & 01011011010101011010100110011101 \\ & \underline{11110001101110001010110011101010} \oplus \\ = \ & 10101010111011010000010101110111 <<< 11110 + S[2] \\ = \ & 11101111110111001101001101010001 \end{aligned}$$

$C = (C \oplus u) <<< t + S[3]$

$= 11101100010011111001111011110111$

$B = 11101111110111001101001101010001$

$C = 00100100011100101111001010000001$

$D = 11101100010011111001111011110111$

$A = 00000101000101111110101111000110$

Lakukan langkah transformasi, *mixing*, dan *swap register* sebanyak $r = 20$ iterasi sehingga keempat register menjadi:

$A = 01000100100011100010111010111010$

$B = 10110001011101011001100000111100$

$C = 00110101100100010110010010011001$

$D = 01011100101100111101101100101110$

c) Lakukan langkah *whitening* akhir sehingga register A dan C menjadi:

$A = A + S[42]$

$01000100100011100010111010111010$

$\underline{01100100001101000001110101101010}$ +

$= 10101000110000100100110000100100$

$C = C + S[43]$

$00110101100100010110010010011001$

$\underline{01010101000101101010110001011001}$ +

$= 10001010101010000001000011110010$

d) Sehingga 16 *byte* pertama*Ciphertext* menjadi:

36 76 194 168 96 152 117 177 242 16 168 138 46 219 179 92

Proses enkripsi ini terus berlangsung hingga semua blok data *plaintext* tersandikan. Setiap 16 *byte* data yang telah tersandikan kembali dimasukkan dalam matriks 16 x 17 per kolomnya:

$$\begin{bmatrix}
36 & 70 & 134 & 92 & 1 & 125 & 165 & 159 & 96 & 18 & 121 & 7 & 49 & 77 & 133 & 56 & 123 \\
76 & 27 & 241 & 241 & 138 & 3 & 84 & 226 & 102 & 253 & 237 & 206 & 48 & 52 & 107 & 113 & 72 \\
194 & 125 & 134 & 226 & 128 & 152 & 4 & 17 & 93 & 116 & 142 & 204 & 49 & 84 & 13 & 65 & 228 \\
168 & 138 & 197 & 116 & 78 & 135 & 167 & 152 & 252 & 80 & 173 & 30 & 227 & 172 & 0 & 95 & 21 \\
96 & 24 & 238 & 26 & 117 & 182 & 164 & 52 & 131 & 165 & 130 & 203 & 74 & 213 & 87 & 110 & 33 \\
152 & 65 & 203 & 140 & 142 & 70 & 145 & 160 & 168 & 15 & 115 & 14 & 69 & 13 & 65 & 249 & 87 \\
117 & 226 & 234 & 239 & 120 & 39 & 0 & 62 & 28 & 251 & 162 & 23 & 152 & 249 & 98 & 40 & 203 \\
177 & 19 & 30 & 212 & 175 & 9 & 94 & 127 & 87 & 197 & 21 & 179 & 34 & 155 & 212 & 38 & 251 \\
242 & 45 & 113 & 194 & 89 & 135 & 161 & 149 & 65 & 85 & 174 & 17 & 86 & 24 & 202 & 77 & 163 \\
16 & 239 & 133 & 14 & 67 & 5 & 146 & 77 & 236 & 242 & 216 & 87 & 41 & 197 & 206 & 177 & 139 \\
168 & 130 & 69 & 161 & 227 & 64 & 67 & 38 & 27 & 154 & 231 & 226 & 235 & 121 & 221 & 218 & 48 \\
138 & 76 & 62 & 92 & 125 & 123 & 62 & 65 & 90 & 81 & 245 & 250 & 153 & 86 & 242 & 68 & 63 \\
46 & 50 & 219 & 166 & 3 & 226 & 190 & 159 & 136 & 151 & 106 & 138 & 2 & 82 & 152 & 149 & 16 \\
219 & 206 & 78 & 185 & 97 & 229 & 26 & 247 & 106 & 97 & 164 & 207 & 158 & 252 & 150 & 167 & 9 \\
179 & 37 & 72 & 146 & 128 & 109 & 112 & 230 & 58 & 98 & 149 & 88 & 85 & 2 & 192 & 33 & 100 \\
92 & 77 & 12 & 58 & 206 & 92 & 109 & 184 & 187 & 26 & 146 & 121 & 187 & 89 & 70 & 164 & 189
\end{bmatrix}$$

Setelah itu, representasi matriks citra digital menjadi seperti pada Gambar 4.2.

Gambar 4. 2 *Ciphertext*

4.3 Dekripsi Citra Digital

Untuk mendekripsi *ciphertext* citra, ubah representasi citra pada Gambar 4.2 menjadi matriks ukuran 16 x17 (sama dengan ukuran *plaintext*). Tempatkan tiap 16 *byte ciphertext* ke dalam 4 buah register A,B,C, dan D sesuai aturan. Lalu, lakukan langkah *whitening* akhir, *swap* register, transformasi, *mixing*, dan *whitening* awal sesuai Algoritma 3.5. Proses dekripsi terus berlangsung hingga semua blok data *ciphertext* dikembalikan menjadi blok data *plaintext*. Setelah itu, ubah kembali matriks 16 x 17 tersebut menjadi *plaintext* dalam bentuk citra sesuai Gambar 4.1.

4.4 Analisa Hasil Pengujian

Berikut akan dianalisa mengenai hasil pengujian terhadap algoritma RC6 (*running time*) dalam mengamankan citra digital. Analisa dilakukan dengan uji

coba algoritma terhadap variasi panjang kunci, variasi karakter yang digunakan pada kunci, dan variasi ukuran citra yang digunakan sebagai data uji. Variasi citra yang dipilih adalah 30 citra dengan variasi ukuran *pixel* yaitu citra ukuran 25x25 *pixel*, 50x50 *pixel*, 75x75 *pixel*, 100x100 *pixel*, 125x125 *pixel*, 150x150 *pixel*, 175x175 *pixel*, 200x200 *pixel*, 225x225 *pixel*, 250x250 *pixel*, 275x275 *pixel*, 300x300 *pixel*, 325x325 *pixel*, 350x350 *pixel*, 375x375 *pixel*, 400x400 *pixel*, 425x25 *pixel*, 450x50 *pixel*, 475x475 *pixel*, dan 500x500 *pixel*. Variasi panjang kunci b yang dipilih adalah 16 *byte*, 24 *byte*, dan 32 *byte*, serta variasi karakter yang digunakan pada kunci adalah berupa angka saja, huruf saja, simbol saja, dan kombinasi dari ketiganya.

Variasi *input* kunci pada $b = 16$ *byte*:

- angka (1234567890123456)
- huruf (abcdefghijklmnop)
- simbol (!@#$%^&*()!@#$%^)
- kombinasi (12345abcde!@#$%^).

Variasi *input* kunci pada $b = 24$ *byte*:

- angka (123456789012345678901234)
- huruf (abcdefghijklmnopqrstuvwx)
- simbol (!@#$%^&*()! @#$%^&*()!@#$)
- kombinasi (12345678abcdefgh!@#$%^&*)

Variasi *input* kunci pada $b = 32$ *byte*:

- angka (12345678901234561234567890123456)
- huruf(abcdefghijklmnop abcdefghijklmnop)
- simbol (!@#$%^&*()!@#$%^! @#$%^&*()!@#$%^)
- kombinasi (12345abcde!@#$%^12345abcde!@#$%^)

Spesifikasi komputer yang digunakan untuk menjalankan program RC6 untuk analisa performanya adalah:

1. Sistem Operasi: Microsoft Windows XP Proffesional 32-bit
2. BIOS: Phoenix BIOS 4.0 Release 6.1
3. Memori RAM 1912 MB
4. Prosesor: Pentium(R) Dual-Core CPU, T4200 @ 2.00 GHz (2 CPUs)

Hasil *running time* dari implementasi program untuk setiap data uji yang digunakan dapat dilihat pada Tabel 4.4 dan Tabel 4.5.

Tabel 4. 4 Waktu Enkripsi (detik) pada Citra dengan Variasi *Input* Kunci, Panjang Kunci $\boldsymbol{b}$, dan Ukuran Citra

	Variasi kunci *input*	Variasi panjang kunci b		
		$b = 16$	$b = 24$	$b = 32$
25 x 25 *pixel*	Angka	2,308	2,1376	2,1435
	Huruf	2,8052	2,3373	2,6919
	Simbol	2,45	2,2165	2,2998
	Kombinasi	2,197	2,2279	2,201
50 x 50 *pixel*	Angka	8,6106	8,5915	8,6508
	Huruf	8,9677	8,5103	8,7761
	Simbol	8,9179	8,6135	8,5714
	Kombinasi	8,5361	8,6497	8,6497
75 x 75 *pixel*	Angka	19,5	19,4681	19,9401
	Huruf	19,7914	19,653	19,9338
	Simbol	19,7306	19,992	19,9502
	Kombinasi	19,6978	19,9524	19,7386
100x100 *pixel*	Angka	36,9526	35,2297	38,1111
	Huruf	36,4766	36,1464	36,331
	Simbol	38,3155	37,6163	39,3083
	Kombinasi	37,2464	36,9528	35,2668

Tabel 4.4 Waktu Enkripsi (detik) pada Citra dengan Variasi *Input* Kunci, Panjang Kunci $\boldsymbol{b}$, dan Ukuran Citra

	Variasi kunci *input*	Variasi panjang kunci b		
		$b = 16$	$b = 24$	$b = 32$
125x125 *pixel*	Angka	57,4692	58,9734	56,9773
	Huruf	59,9444	56,8077	56,4271
	Simbol	56,8628	55,4523	55,4601
	Kombinasi	55,9039	56,2887	56,0552
150x150 *pixel*	Angka	74,7944	74,6301	74,1712
	Huruf	74,1868	75,1308	74,5686
	Simbol	74,2412	74,8716	74,6282
	Kombinasi	74,2431	74,0896	74,5575
175x175 *pixel*	Angka	102,6714	102,7968	102,3765
	Huruf	102,3068	102,2299	102,3059
	Simbol	102,2364	102,9832	102,9832
	Kombinasi	102,244	103,1933	102,3844
200x200 *pixel*	Angka	134,0776	133,7281	134,1127
	Huruf	133,8762	133,2255	133,9054
	Simbol	133,6836	133,1791	133,5769
	Kombinasi	132,3971	132,9558	133,9178
225x225 *pixel*	Angka	169,6141	169,5771	169,7193
	Huruf	169,1596	169,3002	169,6784
	Simbol	170,0479	169,3089	169,9783
	Kombinasi	170,0863	170,0133	170,0176
250x250 *pixel*	Angka	207,3373	211,7862	212,2831
	Huruf	211,801	211,1477	212,5611
	Simbol	210,5441	210,817	211,4314
	Kombinasi	210,1214	211,2812	211,4438
275x275 *pixel*	Angka	256,8411	256,8631	257,315
	Huruf	255,7756	257,3952	256,1949
	Simbol	255,1226	255,3372	256,8351
	Kombinasi	254,5617	255,8478	256,8478

Tabel 4.4 Waktu Enkripsi (detik) pada Citra dengan Variasi *Input* Kunci, Panjang Kunci $\boldsymbol{b}$, dan Ukuran Citra

	Variasi kunci *input*	Variasi panjang kunci b		
		$b = 16$	$b = 24$	$b = 32$
300x300 *pixel*	Angka	314,1099	315,4361	315,1159
	Huruf	314,7566	315,7081	315,7682
	Simbol	314,4305	315,4055	315,9181
	Kombinasi	313,0688	314,9091	315,2158
325x325 *pixel*	Angka	356,5739	357,3324	357,5701
	Huruf	357,3851	357,1436	358,4236
	Simbol	358,1007	357,1994	358,985
	Kombinasi	358,2744	357,8315	357,7763
350x350 *pixel*	Angka	416,3441	416,75	416,9667
	Huruf	416,4954	415,1696	415,2155
	Simbol	416,6828	418,1021	416,7635
	Kombinasi	416,3866	416,8915	418,4889
375x375 *pixel*	Angka	478,0098	478,8132	479,0526
	Huruf	479,3278	476,1696	478,5696
	Simbol	478,9407	479,5064	477,8614
	Kombinasi	478,9727	479,0108	479,5166
400x400 *pixel*	Angka	545,1051	546,6631	547,7769
	Huruf	545,9356	545,7098	546,0811
	Simbol	546,2709	546,0099	546,3517
	Kombinasi	546,0135	546,4305	546,6764
425x425 *pixel*	Angka	620,075	619,2107	619,7581
	Huruf	619,1427	619,6649	619,9931
	Simbol	619,2213	620,1318	620,8506
	Kombinasi	619,7928	619,6952	620,0177
450x450 *pixel*	Angka	696,0003	695,1916	698,2133
	Huruf	696,2144	696,3877	698,5679
	Simbol	696,1258	696,3688	697,6681
	Kombinasi	696,4471	697,2876	697,3045

Tabel 4.4 Waktu Enkripsi (detik) pada Citra dengan Variasi *Input* Kunci, Panjang Kunci $\boldsymbol{b}$, dan Ukuran Citra

	Variasi kunci *input*	Variasi panjang kunci b		
		$b = 16$	$b = 24$	$b = 32$
475x475 pixel	Angka	779,2787	780,0018	780,8613
	Huruf	779,6935	779,9029	779,9171
	Simbol	780,8801	779,4933	780,0033
	Kombinasi	779,4719	779,7618	779,9125
500x500 pixel	Angka	869,9612	871,9775	877,0186
	Huruf	870,6645	873,7533	877,1949
	Simbol	870,1059	872,1443	873,6537
	Kombinasi	871,8317	873,2981	873,5523
525x525 pixel	Angka	968,4569	963,203	963,1666
	Huruf	965,847	963,7618	963,74
	Simbol	963,6309	963,1679	963,3072
	Kombinasi	962,3365	963,8634	963,4168
550x550 pixel	Angka	1061,5684	1063,9332	1063,6128
	Huruf	1060,7789	1063,7478	1063,79
	Simbol	1072,6245	1063,9599	1063,7531
	Kombinasi	1063,8962	1063,7311	1063,8007
575x575 pixel	Angka	1164,653	1171,8206	1175,4771
	Huruf	1177,1858	1170,6199	1171,5802
	Simbol	1174,7156	1175,2605	1175,9319
	Kombinasi	1175,0549	1174,3861	1175,5363
600x600 pixel	Angka	1286,1532	1286,7383	1286,93
	Huruf	1285,5349	1286,6116	1286,7002
	Simbol	1284,3042	1285,0945	1286,4861
	Kombinasi	1285,1685	1286,75	1286,3858
625x625 pixel	Angka	1402,9079	1400,2851	1401,4055
	Huruf	1400,8596	1401,3559	1402,5231
	Simbol	1399,9753	1402,7148	1402,8306
	Kombinasi	1400,7538	1401,5664	1402,5771

Tabel 4.4 Waktu Enkripsi (detik) pada Citra dengan Variasi *Input* Kunci, Panjang Kunci $\boldsymbol{b}$, dan Ukuran Citra

	Variasi kunci *input*	Variasi panjang kunci b		
		$b = 16$	$b = 24$	$b = 32$
650x650 pixel	Angka	1497,6506	1500,8577	1501,3209
	Huruf	1594,7286	1511,8562	1502,8353
	Simbol	1500,8405	1501,4558	1501,4308
	Kombinasi	1512,6103	1502,26	1500,593
675x675 pixel	Angka	1668,3207	1661,7513	1661,9773
	Huruf	1660,5948	1662,4966	1662,5831
	Simbol	1653,4607	1662,2509	1664,3053
	Kombinasi	1662,0951	1663,7834	1662,8492
700x700 pixel	Angka	1733,2157	1736,2693	1737,7018
	Huruf	1746,7905	1737,4318	1739,6365
	Simbol	1736,1951	1736,5824	1737,4682
	Kombinasi	1739,0064	1736,2707	1736,5451
725x725 pixel	Angka	1930,9955	1931,2414	1930,9371
	Huruf	1931,1216	1931,5608	1932,6285
	Simbol	1930,3455	1931,3637	1931,8778
	Kombinasi	1930,6109	1932,1685	1932,3921
750x750 pixel	Angka	2090,8692	2091,2141	2095,87538
	Huruf	2096,9059	2093,4388	2094,5643
	Simbol	2087,8253	2090,3731	2091,1712
	Kombinasi	2091,4505	2091,7765	2092,8093

Tabel 4. 5 Waktu Dekripsi (detik) pada Citra dengan Variasi *Input* Kunci, Panjang Kunci ***b***, dan Ukuran Citra

	Variasi kunci *input*	Variasi panjang kunci b		
		$b = 16$	$b = 24$	$b = 32$
25 x 25 *pixel*	Angka	1,7254	1,8942	1,706
	Huruf	2,1116	1,8305	2,1343
	Simbol	1,904	1,7561	2,144
	Kombinasi	1,775	1,8345	1,9638
50x50 *pixel*	Angka	6,9437	6,8663	7,9397
	Huruf	6,8994	6,9908	6,826
	Simbol	6,8191	6,8484	6,8759
	Kombinasi	6,7169	6,8351	6,9518
75x75 *pixel*	Angka	15,681	16,4048	15,3843
	Huruf	15,8311	15,6115	15,6869
	Simbol	16,1142	15,9926	15,5749
	Kombinasi	16,277	15,7342	16,0584
100x100 *pixel*	Angka	28,466	27,9607	28,6216
	Huruf	29,3288	28,7411	29,3097
	Simbol	29,133	28,6982	28,8853
	Kombinasi	28,5768	28,3197	28,3913
125x125 *pixel*	Angka	46,882	49,1359	45,1376
	Huruf	49,4866	46,979	46,077
	Simbol	45,2427	45,3038	43,6791
	Kombinasi	46,4605	46,6563	46,0175
150x150 *pixel*	Angka	59,9774	60,0116	59,5245
	Huruf	59,5829	60,0395	59,9679
	Simbol	59,7502	60,0858	59,8183
	Kombinasi	59,7686	60,0633	59,7767
175x175 *pixel*	Angka	82,3545	82,0505	82,9877
	Huruf	82,0031	81,9058	82,0624
	Simbol	82,1302	82,1205	82,7069
	Kombinasi	81,8013	82,2347	82,3378

Tabel 4.5 Waktu Dekripsi (detik) pada Citra dengan Variasi *Input* Kunci, Panjang Kunci b, dan Ukuran Citra

	Variasi kunci *input*	Variasi panjang kunci b		
		$b = 16$	$b = 24$	$b = 32$
200x200 *pixel*	Angka	107,1433	107,3568	107,0065
	Huruf	107,3445	107,0614	107,7972
	Simbol	107,2754	107,2498	107,6167
	Kombinasi	107,4413	106,6054	107,7724
225x225 *pixel*	Angka	137,0035	137,4217	136,3122
	Huruf	137,4186	136,3208	136,9874
	Simbol	136,0071	136,1794	136,6847
	Kombinasi	136,3485	136,7391	136,447
250x250 *pixel*	Angka	166,2359	167,1251	166,1935
	Huruf	169,9339	169,6132	167,2132
	Simbol	168,0445	168,8251	169,8722
	Kombinasi	168,6409	168,4358	168,1065
275x275 *pixel*	Angka	208,7461	205,3749	206,515
	Huruf	207,1198	203,5228	206,9337
	Simbol	206,0894	204,3933	207,6546
	Kombinasi	205,5912	206,1068	205,1399
300x300 *pixel*	Angka	262,5241	261,1552	261,2476
	Huruf	250,9751	263,7413	263,8447
	Simbol	249,9447	269,8607	259,5238
	Kombinasi	251,2812	260,2221	250,9136
325x325 *pixel*	Angka	288,3227	289,355	287,8678
	Huruf	289,2837	287,4728	288,8102
	Simbol	291,6215	286,0899	288,3591
	Kombinasi	290,4246	291,9411	290,1426
350x350 *pixel*	Angka	334,2995	334,3902	334,5584
	Huruf	334,1211	333,5873	335,7125
	Simbol	336,5516	335,4766	334,8531
	Kombinasi	335,7419	334,1964	334,1096

Tabel 4.5 Waktu Dekripsi (detik) pada Citra dengan Variasi *Input* Kunci, Panjang Kunci b, dan Ukuran Citra

	Variasi kunci *input*	Variasi panjang kunci b		
		$b = 16$	$b = 24$	$b = 32$
375 x 75 *pixel*	Angka	385,7673	385,5565	386,2141
	Huruf	386,1147	385,4213	386,5838
	Simbol	386,7318	396,3758	385,5428
	Kombinasi	386,1215	385,4632	386,2685
400x400 *pixel*	Angka	440,3417	441,1502	441,8507
	Huruf	439,4071	438,1966	439,4316
	Simbol	442,7536	440,0911	439,9665
	Kombinasi	442,1446	439,6524	441,1609
425x455 *pixel*	Angka	499,3332	499,0511	499,6166
	Huruf	498,1272	497,7584	497,244
	Simbol	499,5859	499,5121	497,5802
	Kombinasi	499,1761	498,6322	498,5317
450x450 *pixel*	Angka	566,7408	563,7453	565,1381
	Huruf	565,1943	565,8133	563,8114
	Simbol	566,2511	566,9205	564,525
	Kombinasi	564,95	566,4157	566,3105
475x475 *pixel*	Angka	629,0034	629,7961	630,1551
	Huruf	630,5918	631,6878	631,9423
	Simbol	630,2556	630,0145	629,6188
	Kombinasi	630,1564	631,2395	630,4054
500x500 *pixel*	Angka	701,5795	703,4724	703,60
	Huruf	700,3945	702,9929	702,2711
	Simbol	700,1448	703,0713	703,1543
	Kombinasi	701,5303	701,8146	702,9209
525x525 *pixel*	Angka	784,6559	785,145	785,7833
	Huruf	783,9201	784,2676	785,9685
	Simbol	783,7048	784,6394	786,0474
	Kombinasi	783,0532	785,8678	785,3162

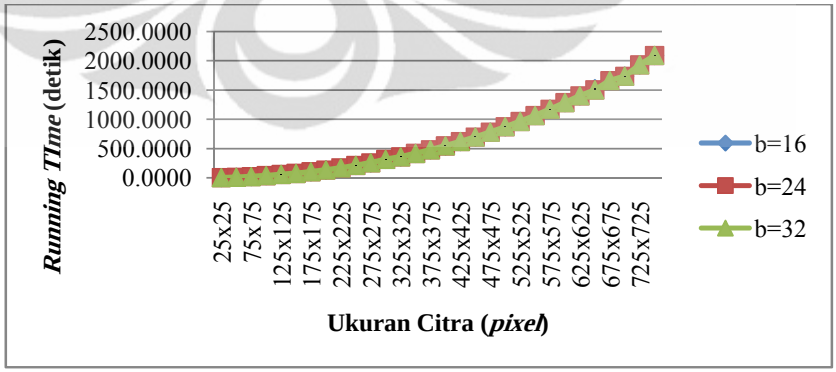
Tabel 4.5 Waktu Dekripsi (detik) pada Citra dengan Variasi *Input* Kunci, Panjang Kunci b, dan Ukuran Citra

	Variasi kunci *input*	Variasi panjang kunci b		
		$b = 16$	$b = 24$	$b = 32$
550x 550 *pixel*	Angka	861,766	862,5306	863,1145
	Huruf	859,2106	863,8286	864,3906
	Simbol	868,5048	863,2157	863,7094
	Kombinasi	863,97	864,0443	866,2512
575x575 *pixel*	Angka	945,5854	946,4611	946,6272
	Huruf	945,4979	946,2059	945,1749
	Simbol	946,271	946,3865	946,0592
	Kombinasi	945,7005	945,7385	946,8301
600x600 *pixel*	Angka	1049,2653	1045,7798	1046,2106
	Huruf	1045,8308	1045,9413	1045,7447
	Simbol	1040,7605	1045,1005	1045,3594
	Kombinasi	1045,6419	1046,8241	1046,7055
625x625 *pixel*	Angka	1140,7258	1145,0317	1142,5545
	Huruf	1140,6602	1140,8738	1140,8301
	Simbol	1140,3195	1141,1944	1141,4126
	Kombinasi	1140,0947	1140,5264	1143,2923
650x650 *pixel*	Angka	1235,3813	1235,7485	1236,6881
	Huruf	1235,1244	1235,5101	1236,9014
	Simbol	1240,0975	1235,95	1236,7752
	Kombinasi	1236,8029	1236,7288	1235,9906
675x675 *pixel*	Angka	1356,5088	1351,1742	1351,8309
	Huruf	1350,2704	1351,5304	1351,7742
	Simbol	1348,3858	1350,9552	1351,6835
	Kombinasi	1351,6944	1351,6833	1351,935
700x700 *pixel*	Angka	1490,0347	1483,3382	1483,9678
	Huruf	1483,414	1483,1649	1483,7312
	Simbol	1484,2185	1483,0985	1483,4764
	Kombinasi	1483,6937	1483,4221	1483,8596

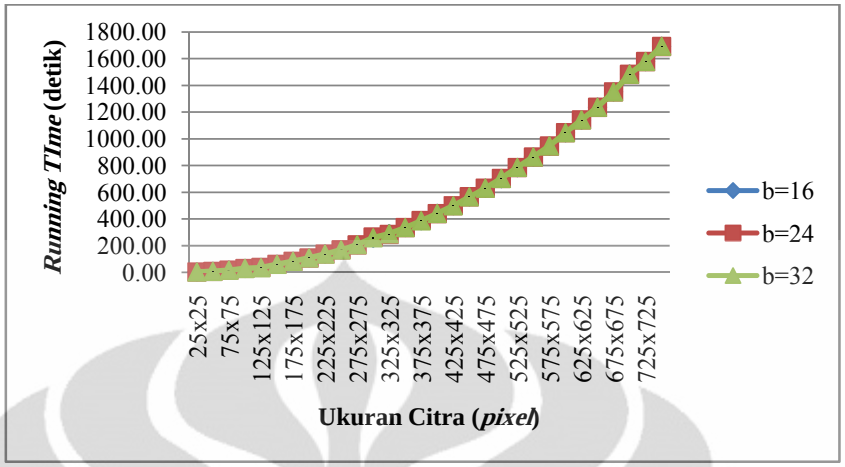
Tabel 4.5 Waktu Dekripsi (detik) pada Citra dengan Variasi *Input* Kunci, Panjang Kunci b, dan Ukuran Citra

	Variasi kunci *input*	Variasi panjang kunci b		
		$b = 16$	$b = 24$	$b = 32$
725 x725 pixel	Angka	1578,8462	1578,9721	1579,736
	Huruf	1578,7092	1579,0435	1579,8203
	Simbol	1577,6331	1579,4746	1579,5241
	Kombinasi	1578,1853	1578,2892	1579,3715
750x750 pixel	Angka	1691,3057	1692,47	1692,8265
	Huruf	1697,5878	1691,63	1692,3701
	Simbol	1691,9258	1692,32	1692,5463
	Kombinasi	1690,4173	1690,78	1692,9387

Berdasarkan Tabel 4.4 dan Tabel 4.5 didapat bahwa *running time* dengan variasi ukuran citra memiliki pengaruh yang cukup berarti. Semakin besar ukuran citra, maka semakin besar pula *running time* yang dihasilkan. Untuk lebih jelasnya, perbandingan hasil *running time* enkripsi algoritma RC6 pada variasi panjang kunci dapat dilihat pada Gambar 4.3 dan perbandingan hasil *running time* dekripsi algoritma RC6 pada variasi panjang kunci dapat dilihat pada Gambar 4.4.



Gambar 4. 3 Grafik Perbandingan *Running Time* (dalam detik) Algoritma Enkripsi RC6 pada Variasi Panjang Kunci dan Ukuran Citra



Gambar 4. 4 Grafik Perbandingan *Running Time* (dalam detik) Algoritma Dekripsi RC6 pada Variasi Panjang Kunci dan Ukuran Citra

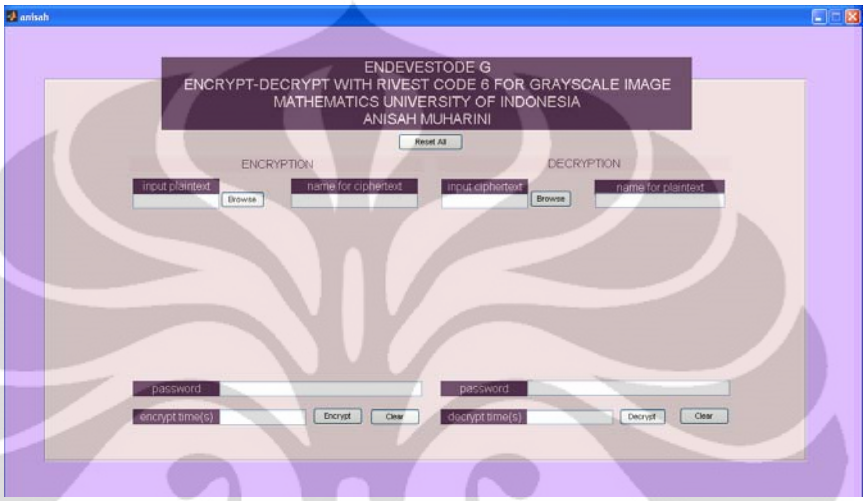
Dari segi variasi panjang kunci dan karakter pada kunci tidak memiliki pengaruh yang berarti terhadap *running time* yang dihasilkan untuk masing-masing ukuran citra yang menjadi data uji. Hal tersebut berarti untuk kemanan lebih baik, pengguna dapat memasukkan kunci dengan panjang maksimum $b = 32$ *byte* tanpa pengaruh *significant* terhadap *running time* algoritma RC6 (lihat Tabel 4.6). Tabel 4.6 memperlihatkan probabilitas *exhaustive search* terhadap kunci pengguna berupa angka saja, huruf saja, simbol saja, dan kombinasi ketiganya. Tabel 4.6 mengindikasikan bahwa sebaiknya pengguna menggunakan kunci kombinasi dari angka, huruf, dan simbol serta panjang kunci 32 *byte* untuk kemanan yang lebih baik, selain itu tak ada pengaruh *significant* terhadap *running time* algoritma.

Tabel 4. 6 Probabilitas Serangan *Input* Kunci

Probabilitas		Angka saja	Huruf saja	Simbol saja	Kombinasi
Banyak Domain		10	52	32	94
Panjang kunci b dalam *byte*	$b = 16$	$1/(10^{16})$	$1/(52^{16})$	$1/(32^{16})$	$1/(94^{16})$
	$b = 24$	$1/(10^{24})$	$1/(52^{24})$	$1/(32^{24})$	$1/(94^{24})$
	$b = 32$	$1/(10^{32})$	$1/(52^{32})$	$1/(32^{32})$	$1/(94^{32})$

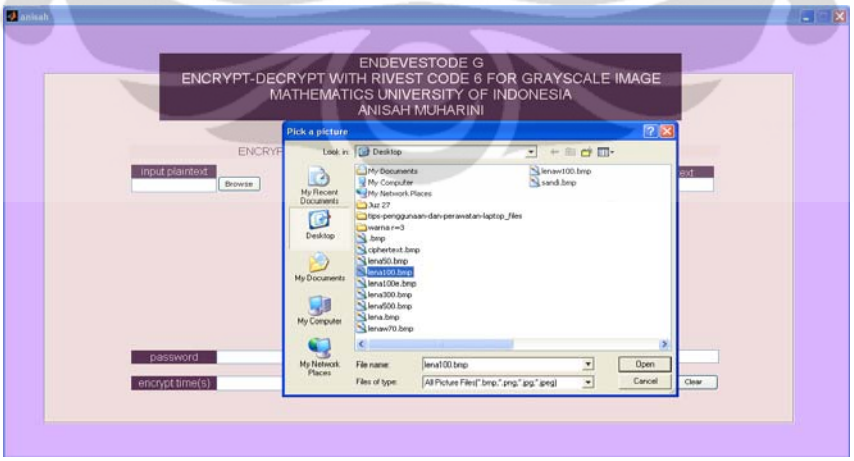
4.4 Pembangunan Program Aplikasi Algoritma RC6

Modifikasi Algoritma RC6 terhadap pengamanan citra digital diimplementasikan pada *software* matlab dengan nama "ENDEVESTODE G" dan tampilan seperti pada Gambar 4.5.



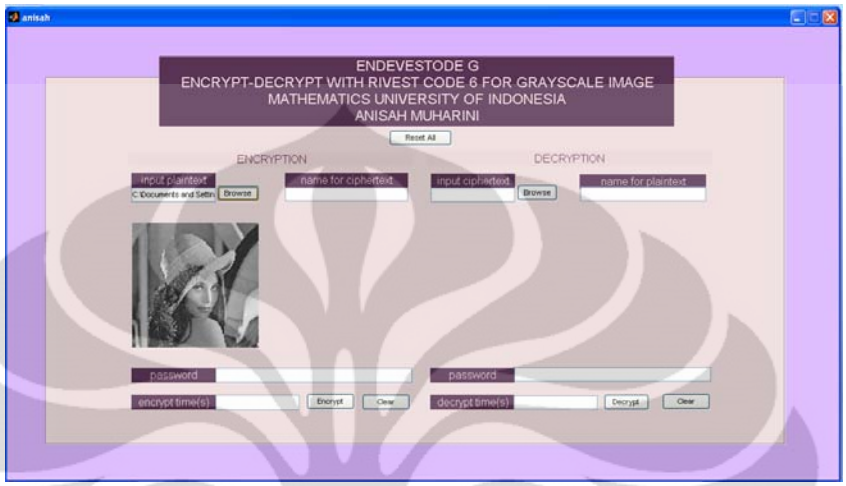
Gambar 4. 5 Tampilan Program Enkripsi-Dekripsi Citra Digital dengan Algoritma RC6

Untuk melakukan enkripsi citra digital dengan program ENDEVESTODE G, langkah pertama yang harus dilakukan adalah pilih citra yang akan dienkripsi dengan klik tombol *browse* pada bagian enkripsi (lihat Gambar 4.6).



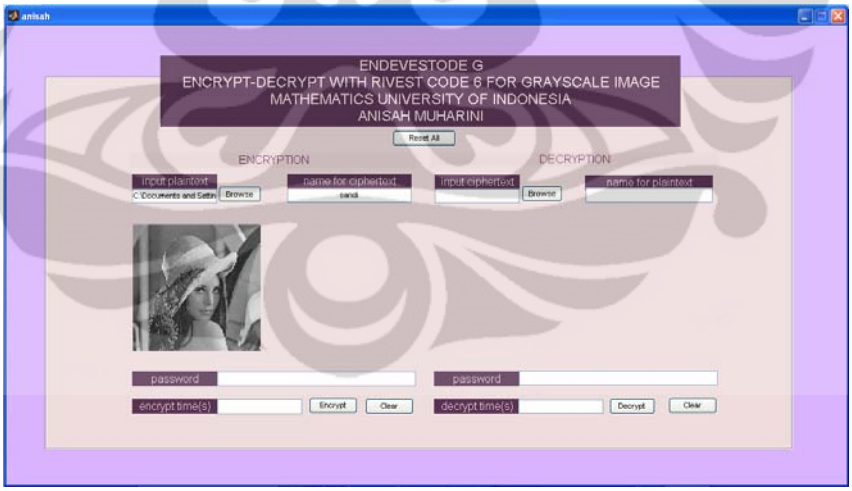
Gambar 4. 6 Tampilan Jendela *Pick a Picture* Enkripsi

Setelah memilih citra yang akan dienkripsi, klik tombol *open* pada jendela *pick a picture* sehingga citra yang akan dienkripsi ditampilkan oleh program seperti pada Gambar 4.7.



Gambar 4. 7 Tampilan *Plaintext* yang akan dienkripsi pada Program

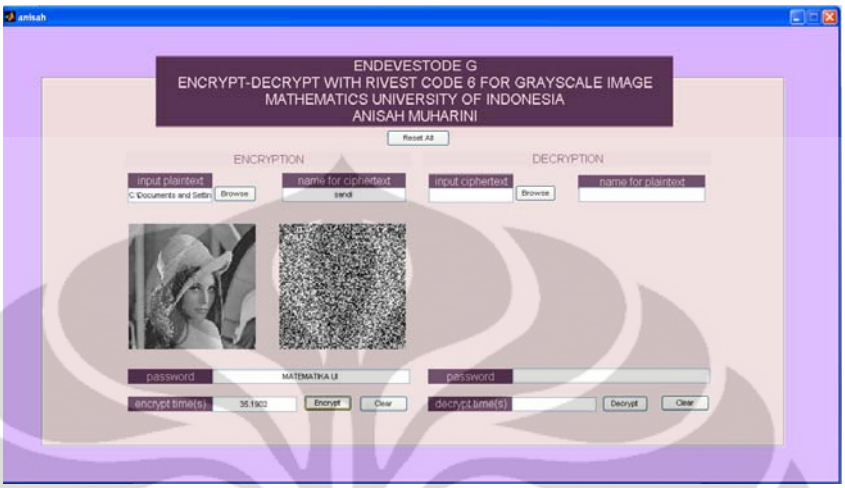
Nama *file* dari *ciphertext* , misalnya "sandi"yang akan dihasilkan dari enkripsi ini, diketik pada kotak *output of ciphertext* (lihat Gambar 4.8)



Gambar 4. 8 Pengetikan Nama *File* dari *Ciphertext* yang akan didapat dari proses Enkripsi

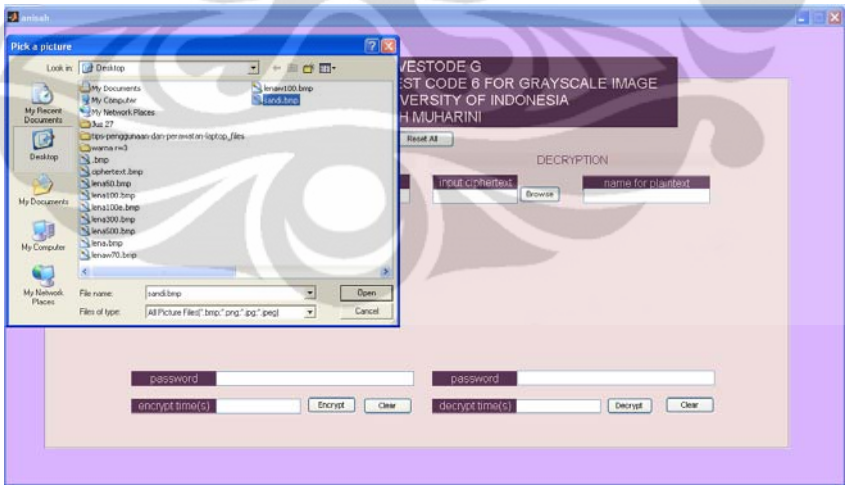
Kemudian, masukkan kunci, misalnya "MATEMATIKA UI" dalam kotak *password*, dan mulai lakukan proses enkripsi dengan klik tombol *Encrypt*. Waktu dalam melakukan proses enkripsi akan muncul pada kotak *encrypt time* saat

enkripsi berhasil dilakukan terhadap *plaintext*. *Ciphertext* akan muncul pada program seperti pada Gambar 4.9.



Gambar 4. 9 Tampilan Program sesaat setelah Enkripsi

Sama halnya dengan enkripsi, langkah pertama yang harus dilakukan untuk dekripsi citra digital pada program ENDEVESTODE G adalah dengan memilih citra yang akan didekripsi dengan klik tombol *browse* pada bagian dekripsi. Misal, pilih citra yang sebelumnya telah dienkripsi dan diberi nama "sandi" (lihat Gambar 4.10).

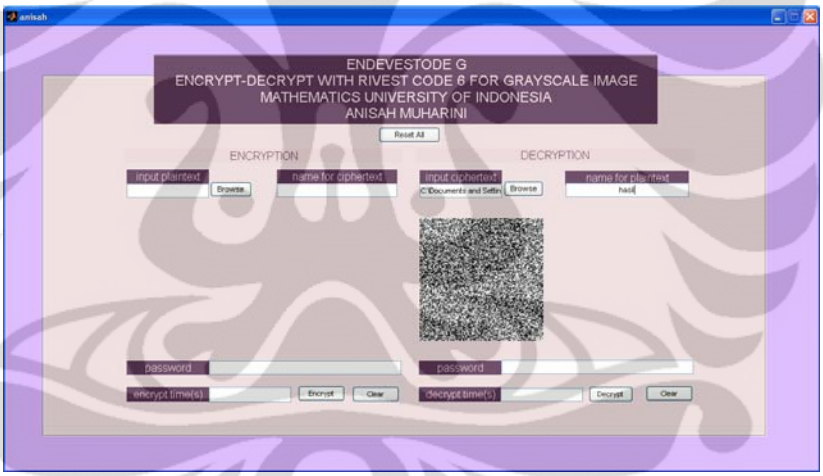


Gambar 4. 10 Tampilan Jendela *Pick a picture* Dekripsi

Setelah memilih citra yang akan didekripsi, klik tombol *open* pada jendela *pick a picture* sehingga citra yang akan didekripsi terlihat seperti pada Gambar 4.11.

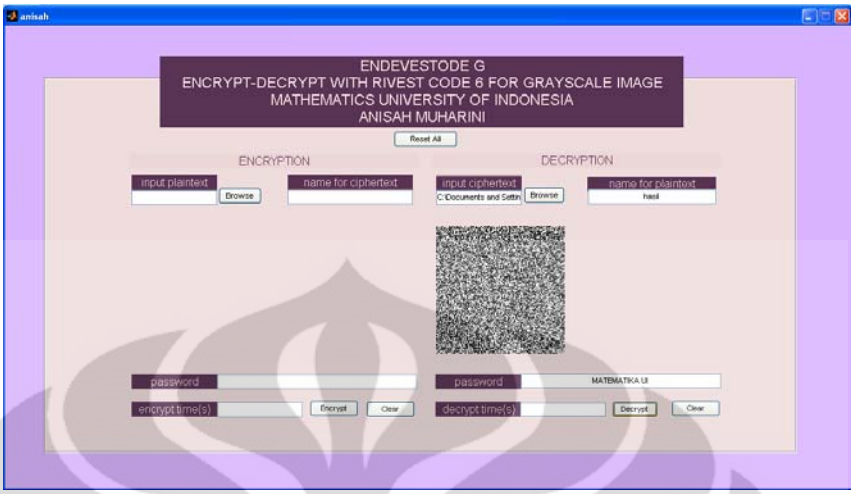
Gambar 4. 11 Tampilan *Ciphertext* yang akan dienkripsi

Nama dari *plaintext* , misalnya "hasil"yang akan dihasilkan dari dekripsi ini, diketik pada kotak *output of plaintext* (lihat Gambar 4.12).



Gambar 4. 12 Pengetikan Nama *File* dari *Plaintext* yang akan didapat dari Dekripsi

Kemudian, masukkan kunci dekripsi sama dengan kunci enkripsi, yaitu"MATEMATIKA UI" dalam kotak *password*, dan mulai lakukan proses enkripsi dengan menekan tombol *Decrypt* (lihat Gambar 4.13)

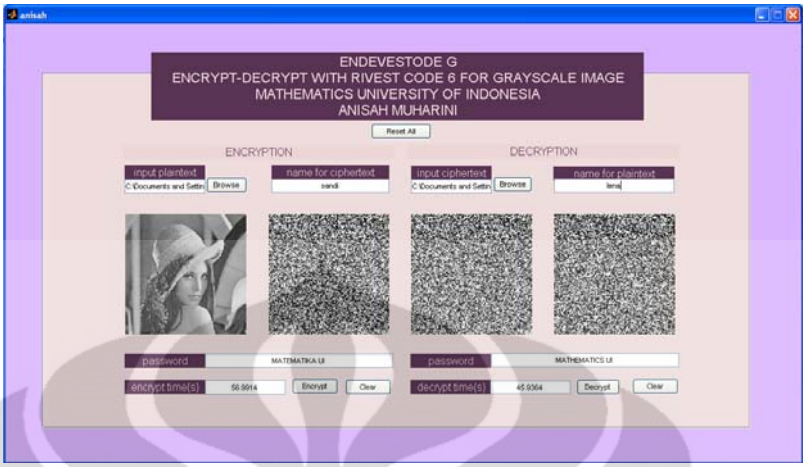


Gambar 4. 13 Tampilan program saat siap melakukan dekripsi

Waktu dalam melakukan proses dekripsi akan muncul pada kotak *decrypt time* saat dekripsi berhasil dilakukan terhadap *ciphertext. Ciphertext* akan muncul pada program seperti pada Gambar 4.14.

Gambar 4. 14 Tampilan Program sesaat setelah Dekripsi

Jika kunci yang dimasukkan pada saat dekripsi berbeda dengan pada saat enkripsi, maka *ciphertext* tidak bisa kembali menjadi *plaintext*. Tampilan program saat kunci enkripsi berbeda dengan kunci dekripsi ditampilkan pada Gambar 4.15. Kunci pada saat enkripsi data adalah "MATEMATIKA UI" sedangkan kunci pada dekripsi adalah "MATHEMATICS UI".Terlihat bahwa *ciphertext* tidak berhasil dikembalikan menjadi *plaintext*.



Gambar 4. 15 Tampilan Program saat Kunci Dekripsi tidak sama dengan Kunci Enkripsi

BAB 5
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Pembahasan pada skripsi ini terkait aplikasi algoritma RC6 dalam pengamanan citra digital dan berdasarkan hasil uji coba yang telah dilakukan didapat beberapa kesimpulan, yaitu:

1. Algoritma RC6 berhasil diaplikasikan untuk pengamanan data dalam bentuk citra digital.
2. Panjang kunci pengguna tidak banyak mempengaruhi *running time* dari algoritma RC6. Untuk itu, pengguna dapat menggunakan kunci dengan panjang maksimum $b = 32$ *byte* demi keamanan yang lebih baik.
3. Variasi kunci pengguna berupa angka, huruf, maupun simbol tidak banyak mempengaruhi *running time* algoritma RC6. Untuk itu pengguna dapat menggunakan kunci dengan kombinasi dari simbol, angka, dan huruf untuk meningkatkan keamanan.

5.2 Saran

Saran yang perlu diperhatikan terkait penelitian selanjutnya adalah

1. Skripsi ini menggunakan algoritma RC6 untuk pengamanan data citra digital, untuk lebih lanjut dapat gunakan algoritma RC6 ini untuk pengamanan data bentuk lainnya, seperti audio atau video.
2. Skripsi ini mengimplementasikan algoritma RC6 pada *software* matlab, untuk lebih lanjut dapat ambil *software* lain dalam pengimplementasian algoritma RC6 ini.

DAFTAR PUSTAKA

Adiyanto, Suhud. (2003). Skripsi: *Algoritma Enkripsi Serpent pada Mode Cipher Feedback*. Depok: Universitas Indonesia

Bishop, Matt. (2003). *Computer Security.* Pearson Education

Contini, S, Rivest, R.L, Robshaw, M.J.B, and Y.L. Yin. (1998). Jurnal: *The Security of the RC6™ Block Cipher*. Dipetik 3 Mei 2012 dari *home page* RSA. Website: http: //www.rsalabs.com/rc6/

Herstein, I.N. (1995). *Abstract Algebra.* New Jersey: Prentice-Hall.

James, H dan Michael, B. (2012). *Concrete Computing*. Version 1.39 2012-05-30

Made, I Dwi Saputra Jaya. (2002). Skripsi: *Algoritma Enkripsi:RC6.* Depok: Universitas Indonesia

Munir, Rinaldi. (2004). *Pengolahan Citra Digital dengan Pendekatan Algoritma.* Bandung: Informatika.

Purnomo, M dan Muntasa, A. (2010). *Konsep Pengolahan Citra Digital dan Ekstraksi Fitur.*Yogyakarta: Graha Ilmu.

Robshaw. (2001). Jurnal: *RC6 and AES.* Dipetik 3 Mei 2012 dari *home page* RSA. Website:http: //www.rsalabs.com/rc6/

Prayudi, Y. dan Halik, I. (2005). Jurnal: *Studi dan Analisis Algoritma Rivest Code 6 dalam Enkripsi/Dekripsi Data*. SNATI:Yogyakarta

Rivest, R., Robshaw, M., Sidney, R., dan Yin Y.L.(1998). Jurnal: *The RC6 Cipher*. Dipetik 3 Mei 2012 dari *home page* RSA. Website: http: //www.rsalabs.com/rc6/

Rosen, Kenneth. (2007). *Discrete Mathematics and Its Applications.* New York: Mc Graw Hill.

Stallings, William. (2005). *Cryptography and Network Security Principles and Practices.* New Jersey: Prentice Hall

Tilberg, van Henk., Jajodia, Sushil. (2011). *Encyclopedia of Cryptography and Security.* Second Ed. New York: Springer Science

Wisnu, Rangga. (2010). Skripsi: *Implementasi Algoritma RC6 untuk Enkripsi SMS pada Telepon Selular*. Bandung: Program Studi Teknik Informatika - Institut Teknologi Bandung.

LAMPIRAN

Lampiran 1 Pembuktian Teorema 2.1 (Herstein, 1995)

Diketahui $f: X \rightarrow Y$ adalah fungsi bijektif, maka berdasarkan Definisi 2.1, maka f 1-1 dan *onto*.

- f 1-1 berlaku jika dan hanya jika $f(x_1) = f(x_2)$ maka $x_1 = x_2$, $\forall\, x_1, x_2 \in X$
- f *onto* berlaku $\forall\, y \in Y,\ \exists x \in X\ \ni y = f(x)$

$f: X \rightarrow Y$ bijektif, maka $f^{-1} \circ f = id_Y$ dan $f \circ f^{-1} = id_X$ dengan id_X dan id_Y adalah fungsi identitas dari X dan Y (Herstein, 1995).

Jika $y \in Y$ maka $(f \circ f^{-1})(y) = f(f^{-1}(y) = f(x) = y$ atau dengan kata lain, $(f \circ f^{-1}(y)) = y, \forall\, y \in Y$ dan jika $x \in X$ maka $(f \circ f^{-1})(x) = f(f^{-1}(x) = f(y) = x$ dengan kata lain $(f \circ f^{-1}(x)) = x, \forall\, x \in X$

Selanjutnya akan dibuktikan $f^{-1}: Y \rightarrow X$ adalah fungsi 1-1:

Ambil $y_1, y_2 \in Y$ dengan $f^{-1}(y_1) = f^{-1}(y_2)$

$$f\big(f^{-1}(y_1)\big) = f\big(f^{-1}(y_2)\big)$$

$$y_1 = y_2$$

sehingga terbukti $f^{-1}: Y \rightarrow X$ merupakan fungsi 1-1

Selanjutnya akan dibuktikan $f^{-1}: Y \rightarrow X$ adalah fungsi *onto*:

Ambil $x \in X$, maka $f(x) \in Y$.

$Misalkan\ y = f(x)\ maka\ f^{-1}(y) = f^{-1}\big(f(x)\big) = x$

sehingga terbukti $f^{-1}: Y \rightarrow X$ merupakan fungsi *onto*

Karena $f^{-1}: Y \rightarrow X$ merupakan fungsi 1-1 dan *onto* maka terbukti $f^{-1}: Y \rightarrow X$ adalah fungsi bijektif.

Lampiran 2 Pembuktian Lemmma 2.1 (Herstein, 1995)

Diketahui $g: S \to T$ dan $f: T \to U$ adalah fungsi bijektif, maka berdasarkan Definisi 2.1 f dan g merupakan fungsi 1-1 dan *onto*.

Selanjutnya akan dibuktikan $f \circ g: S \to U$ adalah fungsi 1-1:

Ambil $s_1, s_2 \in S$, dengan $(f \circ g)(s_1) = (f \circ g)(s_2)$

$$f\big(g(s_1)\big) = f\big(g(s_2)\big) \quad \text{(definisi fungsi komposisi)}$$

$$g(s_1) = g(s_2) \qquad (f \text{ 1-1})$$

$$s_1 = s_2 \qquad (g \text{ 1-1})$$

sehingga terbukti$f \circ g: S \to U$ merupakan fungsi 1-1

Selanjutnya akan dibuktikan $f \circ g: S \to U$ adalah fungsi *onto*:

g surjektif maka $\forall t \in T \ \exists s \in S$ sedemikian sehingga $g(s) = t$

f surjektif maka $\forall u \in U \ \exists t \in T$ sedemikian sehingga $f(t) = u$

Ambil $u_1 \in U$, karena f maka $\exists t_1 \in T$ sedemikian sehingga $f(t_1) = u_1$

Ambil $t_1 \in T$, karena g surjektif maka $\exists s_1 \in S$ sedemikian sehingga $g(s_1) = t_1$

Maka, $f\big(g(s_1)\big) = u_1$

$$f \circ g(s_1) = u_1$$

$\therefore \ \forall u_1 \in U \ \exists s_1 \in S$ sedemikian sehingga $f \circ g(s_1) = u_1,$ sehingga $f \circ g: S \to U$ merupakan fungsi yang surjektif.

Karena$f \circ g: S \to U$ merupakan fungsi 1-1 dan *onto* maka terbukti $f \circ g: S \to U$ adalah fungsi bijektif.

Lampiran 3 Pembuktian Teorema 2.2 (Herstein, 1995)

Jika $g: S \to T$ dan $f: T \to U$ fungsi yang bijektif, maka $(f \circ g)^{-1}$ juga fungsi yang

bijektif dan $(f \circ g)^{-1} = g^{-1} \circ f^{-1}$

Bukti:

Dengan menggunakan Teorema 2.2 yaitu $f \circ g: S \to U$ fungsi yang bijektif dan Teorema 2.1 yaitu $f^{-1} : U \to T$ fungsi yang bijektif, maka $(f \circ g)^{-1}: U \to S$ juga bijektif.

Akan dibuktikan: $(f \circ g)^{-1} = g^{-1} \circ f^{-1}$

$f: T \to U$ fungsi bijektif, maka menurut Teorema 2.1 , $f^{-1}: U \to T$ juga bijektif.

$g: S \to T$ fungsi bijektif, maka menurut Teorema 2.1 , $g^{-1}: T \to S$ juga bijektif

$(f \circ g) \circ (f \circ g)^{-1} = id_U$ (id_U adalah fungsi identitas di U)

$f^{-1} \circ (f \circ g) \circ (f \circ g)^{-1} = f^{-1} \circ id_U$

$g \circ (f \circ g)^{-1} = f^{-1}$

$(f \circ g)^{-1} = g^{-1} \circ f^{-1}$

∴ Terbukti $(f \circ g)^{-1} = g^{-1} \circ f^{-1}$

Lampiran 4 Pembuktian Teorema 2.4 (Stallings, 2005)

Misalkan $a \bmod n = r_a$ dan $b \bmod n = r_b$

Maka $a = r_a + jn$ dan $b = r_b + kn$ dengan $j, k \in Z$

$(a + b) \bmod n = (r_a + jn + r_b + kn) \bmod n$

$= (r_a + r_b + (j + k)n) \bmod n$

$= (r_a + r_b) \bmod n$

$= (a \bmod n + b \bmod n) \bmod n$

$(a - b) \bmod n = (r_a + jn + (r_b - kn)) \bmod n$

$$= (r_a - r_b + (j-k)n)\ mod n$$

$$= (r_a - r_b)\ mod\ n$$

$$= (a\ mod\ n - \ b\ mod\ n)\ mod\ n$$

$$(axb) mod\ n = ((r_a + jn)(r_b + kn))\ mod n$$

$$= (r_a r_b + r_a kn + r_b jn + jnkn)\ mod n$$

$$= (r_a r_b + n(r_a k + r_b j + jkn))\ mod n$$

$$= (r_a r_b)\ mod n$$

$$= ((a\ mod\ n\ )\text{x}(\ b\ mod\ n))\ mod\ n$$

Lampiran 5 Beberapa potongan *source code* program

```
{pengambilan blok data}

Input:foto
foto1=imread(foto);
[x1 y1]=size(foto1);
a1=x1*y1;
foto2=reshape(foto1,1,a1);
foto3=double(foto2);
a2=ceil(a1/16);
a3=a2*16;
foto4=zeros(1,a3);
for i=1:a1
    foto4(1,i)=foto3(1,i);
end

a4=1;
for i=1:a2
    data(i,1:16)=foto4(1,a4:a4+15);
    a4=a4+16;
end


{key sheduling}

Input:a=kunci, r
[x,y]=size(a);
```

```
%--penempatan kunci pengguna pada array L--
j=1;
for i=1:c
      N(i,:)=fliplr(a(j:j+3));
      j=j+4;
end


%--inisialisasi kunci pada array S--
S0=zeros(2*r+4,1);
S0(1)=hex2dec('B7E15163');
for i=2:44
S0(i)=mod(S0(i-1)+hex2dec('9E3779B9'),2^32);
end

%--kombinasikan L dan S--

A = 0;
B = 0;
i = 1;
j = 1;
for k = 1:3*(max(c,2*r+4))
      S1(i)   = mod(S0(i) + A(i) + B(i),2^32);
      S2(i,:) = dec2bin(S1(i),32);
      S3(i,:) = circshift((S2(i,:)),[0 -3]);
      S4(i,:) = bin2dec(S3(i,:));

      L2=dec2bin(N(j,:),8);
      L3=reshape(L2',1,32);
      L4=bin2dec(L3);
      LL(j,:)=mod(L4 + A(i) + B(i),2^32);
      LLL=dec2bin(LL(j,:),32);

      ges=mod(A(i)+B(i),2^32);
      ges2=dec2bin(ges,32);
      ges3=ges2(28:32);
      ges4=bin2dec(ges3);

      L=circshift(LLL,[0 -ges4]);
      L5(j)=bin2dec(L);

      A(i+1) = S4(i);
      B(i+1) = L5(j);
      i = mod(( i ),2*r+4)+1;
      j = mod(( j ),c)+1;
end


{Enkripsi}

for i=1:a2
    %--whitening awal--

    B8=dec2bin(B,8);
    B9=reshape(B8',1,32);
```

```
B2 = dec2bin(mod(bin2dec(B9) + S4(1),2^32),32);

kk7=1;
for kk8=1:8:25
    g(1,kk7)=bin2dec(B2(1,kk8:kk8+7));
    kk7=kk7+1;
end
B=g;

D8=dec2bin(D,8);
D9=reshape(D8',1,32);

D2 = dec2bin(mod(bin2dec(D9) + S4(2),2^32),32);

kk9=1;
for kk10=1:8:25
    gg(1,kk9)=bin2dec(D2(1,kk10:kk10+7));
    kk9=kk9+1;
end
D=gg;

for j = 1:r
    Bi=dec2bin(B,8);
    Bii=reshape(Bi',1,32);
    Bee=bin2dec(Bii);

    p = dec2bin(mod(Bee*(2*Bee+1),2^32),32);
    s = circshift(p,[0 -5]);

    kk1=1;
    for kk=1:8:25
        q(1,kk1)=bin2dec(s(1,kk:kk+7));
        kk1=kk1+1;
    end
    t=q;

    Di=dec2bin(D,8);
    Dii=reshape(Di',1,32);
    Dee=bin2dec(Dii);

    r2= dec2bin(mod(Dee*(2*Dee+1),2^32),32);
    x = circshift(r2,[0 -5]);

    kk2=1;
    for kkk=1:8:25
        qq(1,kk2)=bin2dec(x(1,kkk:kkk+7));
        kk2=kk2+1;
    end
    u=qq;

    tt=dec2bin(t,8);        uu=dec2bin(u,8);
    t3=reshape(tt',1,32);   u3=reshape(uu',1,32);
    t4=t3(28:32);           u4=u3(28:32);
    t5=bin2dec(t4);         u5=bin2dec(u4);

    AA=dec2bin(A,8);        CC=dec2bin(C,8);
```

```
ttt=mod(AA + tt,2);     uuu=mod(CC+uu,2);

e1=zeros(4,1);
for ii=1:4
    for jjj=1:8
        e1(ii)=e1(ii)+ttt(ii,jjj)*2^(8-jjj);
    end
end

e2=zeros(4,1);
for iii=1:4
    for jjjj=1:8
        e2(iii)=e2(iii)+uuu(iii,jjjj)*2^(8-jjjj);
    end
end

A=reshape(e1,1,4);      C=reshape(e2,1,4);

d=dec2bin(A,8);
dd=reshape(d',1,32);

A=circshift(dd,[0 -t5]);

kk3=1;
for kkkk=1:8:25
    q2(1,kk3)=bin2dec(A(1,kkkk:kkkk+7));
    kk3=kk3+1;
end
AAAAA=q2;

e=dec2bin(C,8);
ee=reshape(e',1,32);

C=circshift(ee,[0 -u5]);

kk4=1;
for k5=1:8:25
    qq2(1,kk4)=bin2dec(C(1,k5:k5+7));
    kk4=kk4+1;
end
CCCCC=qq2;

A6=dec2bin(AAAAA,8);
AA6=reshape(A6',1,32);
AA7=bin2dec(AA6);
C6=dec2bin(CCCCC,8);
CC6=reshape(C6',1,32);
CC7=bin2dec(CC6);

A=dec2bin(mod(AA7+S4(2*j+1),2^32),32);
C=dec2bin(mod(CC7+S4(2*j+2),2^32),32);

kk7to=1;
for kk8to=1:8:25
    gto(1,kk7to)=bin2dec(A(1,kk8to:kk8to+7));
```

```
            kk7to=kk7to+1;
        end
        A=gto;

        kk9to=1;
        for kk10to=1:8:25
            ggto(1,kk9to)=bin2dec(C(1,kk10to:kk10to+7));
            kk9to=kk9to+1;
        end
        C=ggto;

        %--swap register--
        B=A;
        C=B;
        D=C;
        A=D;
    end

    %--whitening akhir--
    A7=dec2bin(A,8);
    A=reshape(A7',1,32);
    C7=dec2bin(C,8);
    C=reshape(C7',1,32);

    A2 = dec2bin(mod(bin2dec(A) + S4(2*r+3),2^32),32);

    kk70=1;
    for kk80=1:8:25
        g3(1,kk70)=bin2dec(A2(1,kk80:kk80+7));
kk70=kk70+1;
    end

    C2 = dec2bin(mod(bin2dec(C) + S4(2*r+4),2^32),32);

    kk90=1;
    for kk100=1:8:25
        gg3(1,kk90)=bin2dec(C2(1,kk100:kk100+7));
        kk90=kk90+1;
    end
end
```