UNIVERSITAS INDONESIA

APLIKASI ALGORITMA RIJNDAEL DALAM PENGAMANAN CITRA DIGITAL

SKRIPSI

FADHILAH HANIFAH
0806452160

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

UNIVERSITAS INDONESIA

APLIKASI ALGORITMA RIJNDAEL DALAM PENGAMANAN CITRA DIGITAL

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana sains

FADHILAH HANIFAH
0806452160

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
PROGRAM STUDI SARJANA MATEMATIKA
DEPOK

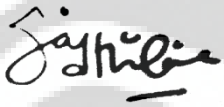
JULI 2012

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar.

Nama	: Fadhilah Hanifah
NPM	: 0806452160
Tanda Tangan	: 
Tanggal	: Juli 2012

HALAMAN PENGESAHAN

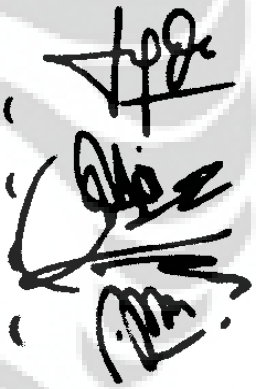
Skripsi ini diajukan oleh

Nama	:	Fadhilah Hanifah
NPM	:	0806452160
Program Studi	:	Sarjana Matematika
Judul Skripsi	:	Aplikasi Algoritma Rijndael 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	:	Alhadi Bustamam, PhD ( )
Penguji II	:	Prof. Dr. Djati Kerami ( )
Penguji III	:	Dr. Yudi Satria, M.T ( )



Ditetapkan di	:	Depok
Tanggal	:	19 Juni 2012

KATA PENGANTAR

Alhamdulillahirobbil'alamin. Segala puji dan syukur penulis panjatkan kehadirat Allah SWT karena hanya dengan kehendak dan ridho-Nya lah penulis dapat menyelesaikan pendidikan hingga sekarang dan akhirnya juga mampu menyelesaikan tugas akhir ini dengan baik.

Allah SWT tidak hanya memberikan semua rahmatnya kepada penulis, namun juga mengirimkan orang-orang hebat di kehidupan penulis. Pada kesempatan kali ini penulis hendak mengucapkan terima kasih kepada mereka yang telah banyak membantu penulis dari awal perkuliahan hingga tugas akhir ini dapat selesai dengan baik. Terima kasih kepada:

1. Bapak Dr. Yudi Satria, M.T. selaku Ketua Departemen Matematika, Ibu Rahmi Rusin, S.Si., M.ScTech. selaku Sekretaris Departemen Matematika, yang telah banyak membantu penulis menyangkut perkuliahan.
2. Bpk. Suryadi MT , M.T selaku pembimbing tugas akhir penulis yang telah banyak memberikan motivasi, waktu, saran serta ilmu yang bermanfaat selama pengerjaan tugas akhir ini.
3. Ibu Dra. Saskya Mary Soemartojo M.Si selaku pembimbing akademis penulis yang telah banyak memberikan pengarahan-pengarahan yang berguna selama masa kuliah yang telah dijalani penulis.
4. Para dewan penguji: Bapak Alhadi Bustamam, PhD., Bapak Prof. Dr. Djati Kerami., dan Bapak Dr. Yudi Satria, M.T., yang telah bersedia memberikan kritik, saran dan ilmu yang berguna bagi tugas akhir penulis.
5. Seluruh staff pengajar di Matematika UI yang telah memberikan ilmu-ilmu baru yang bermanfaat bagi penulis. Beribu terima kasih. Tak lupa seluruh karyawan di Matematika yang telah banyak membantu penulis selama melakukan kegiatan di departemen, terutama mba Santi yang telah sabar karena banyak direpotkan penulis.
6. Kedua orang tua penulis, umi Niswarti dan abi Hermansyah yang tiada henti mengalirkan doa-doa untuk penulis di setiap ibadah mereka, serta dukungan dan cinta yang tanpa bosan mereka berikan untuk penulis.

7. Adik-adik penulis, Fawwas, Rifqoh, Hazimah, Ismah, Hazis, dan Hilmy yang telah menjadi "obat" ketika penulis mengalami kejenuhan.
8. Teman seperjuangan, Anisah dan Maulia yang telah sama-sama berjuang menjadi "Kriptografi Woman", tertawa bersama, keluh kesah bersama, menyelesaikan masalah bersama, dan akhirnya lulus bersama. Juga Asri, yang akan menjadi "the next kriptografi woman". Semangat!
9. Umbu, bang Andy, Cindy dan ka Yanu yang telah banyak membantu penulis dalam hal ide-ide program dan penulisan. Terima kasih banyak.
10. Keluarga Math 08: Ade, Sita, Qiqi, Tuti, Ines, Risya, Ijut, Cindy, Numa, Arief, Adhi, Dheni, Bang Andy, Awe, Umbu, Arman, Bowo, Maimun, Ega, Nita, Citra, Luthfah, Dhea, Uni Aci, Danis, Arkies, Agy, Ko Hen, Dhewe, Mei, Mba Siwi, Vika, Nora, Janu, Resti, Ifah, Eka, Emy, Icha, MayTA, Fani, Olin, Yulial, Yulian, Agnes, Maul, Dian, Nadia, Wulan, Anisah, Uchi D, Uci L,Mas Puput, dan tak lupa Dede, Purwo, Aya, Masykur, Juni, Dini, Ze, Mela, dan Ramos. Terima kasih karena semuanya. Semua yang membuat Math 08 menjadi satu keluarga. Semuanya. Semoga kita bisa bersaudara sampai ke syurga. Amin.
11. Ade, Sita, Qiqi, Tuti, Ines, Risya, Ijut, Cindy, Numa, Arief, Adhi, Dheni, dan Bang Andy. Terima kasih atas persahabatan indah yang kita jalani selama 4 tahun. Kalian yang terbaik selama penulis berada di matematika.
12. Kakak-kakak angkatan 2006, 2007 dan juga adik-adik angkatan 2009, 2010, dan 2011. Walau kebersamaan kita hanya sebentar, namun tidak akan mudah terlupakan.
13. Mereka yang mengenalkan sepak bola, badminton, drama Asia, dan Running Man sebagai teman hiburan penulis di kala jenuh.

Dan seluruh teman serta orang-orang yang telah membantu namun tidak dapat disebutkan satu-satu. Semoga tugas akhir ini dapat bermanfaat. Maaf atas segala kekurangan, karena kekurangan hanya milik manusia dan kesempurnaan hanya milik Allah SWT. Terima Kasih.

Penulis

2012

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI
TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

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

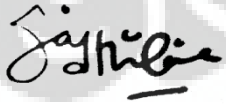
Nama	: Fadhilah Hanifah
NPM	: 0806452160
Program Studi	: Sarjana Matematika
Departemen	: Matematika
Fakultas	: Matematika dan Ilmu Pengetahuan Alam
Jenis karya	: Skripsi

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 Rijndael 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 tugas akhir 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 : 19 Juni 2012
Yang menyatakan



(Fadhilah Hanifah)

ABSTRAK

Nama : Fadhilah Hanifah
Program Studi : Matematika
Judul : Aplikasi Algoritma Rijndael dalam Pengamanan Citra Digital

Citra digital merupakan salah satu data atau informasi yang sering disalahgunakan, oleh karena itu pengamanan data citra digital menjadi hal yang penting dan mendesak. Salah satu pengamanan bisa dilakukan dengan menerapkan algoritma enkripsi Rijndael. Empat proses utama algoritma ini terdiri dari satu proses permutasi (*ShiftRows)* dan tiga proses subtitusi (*SubBytes, MixColumns,* dan *AddRoundKey)* dan juga proses penjadwalan kunci. Dalam tugas akhir ini akan dibahas tentang pengamanan data citra digital oleh algoritma Rijndael dan juga implementasi algoritma ini dalam mengamankan citra digital. Algoritma Rijndael merupakan algoritma enkripsi yang dapat diaplikasikan untuk pengamanan data berbentuk citra digital.

Kata Kunci : algoritma Rijndael, citra digital, enkripsi, dekripsi
xiv + 71 halaman ; 22 gambar, 10 tabel
Daftar Pustaka : 13 (1995 - 2011)

ABSTRACT

Name : Fadhilah Hanifah
Study Program : Mathematics
Title : Application of Rijndael Algorithm in Securing Digital Image

Digital image is a form of data or information which is often manipulated, therefore securing digital image becomes urgently important. Rijndael encryption algorithm can be used to secure it. This algorithm consists of a permutation process (ShiftRows), three substitution processes (Sub*Byte*s, MixColumns, and AddRoundKey), and also the key scheduling process. In this minithesis, the problem that will be discussed is about securing digital image with Rijndael algorithm and also the implementation of this algorithm on securing digital image. Rijndael algorithm is an encryption algorithm that can be applied for securing digital image.

Keywords : Rijndael algorithm, digital image, encryption, decryption
xiv + 71 pages ; 22 pictures, 10 tables
Bibliography : 13 (1995 - 2011)

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

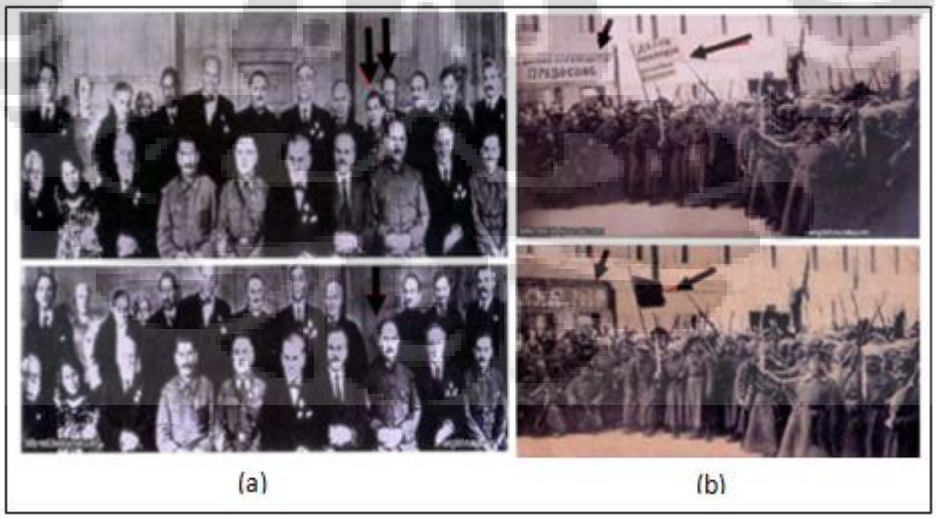
DAFTAR LAMPIRAN

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Dewasa ini, keamanan telah menjadi kebutuhan yang utama dalam kehidupan manusia disamping kebutuhan-kebutuhan lainnya. Semua orang memiliki sesuatu yang ingin kemanannya terjaga, seperti uang, surat-surat berharga, barang-barang berharga dan lain-lain. Tidak hanya itu saja yang membutuhkan keamanan, namun data atau informasi yang dimiliki, butuh kemananan yang memadai, karena data atau informasi merupakan salah satu aset penting, yang jika tidak dijaga dengan baik dapat menimbulkan kerugian.

Data atau informasi tersebut dapat berupa teks, citra/gambar, audio, dan video. Dengan adanya kemajuan teknologi informatika, data atau informasi dapat disajikan dalam bentuk digital. Akan tetapi, bentuk penyimpanan seperti ini sangat rentan aspek keamanannya. Data dapat dengan mudah diganti, dimanipulasi, dihilangkan, di*copy*, atau bahkan disalahgunakan. Salah satu data yang sering dimanipulasi dan disalahgunakan adalah citra digital.



[Sumber: Wikipedia.com]

Gambar 1. 1 Contoh Manipulasi Citra Digital
Penghilangan Gambar Dua Tokoh

(a) Perubahan Informasi pada Spanduk

Gambar 1.1 merupakan contoh manipulasi citra digital. Gambar semula dapat diubah dan dimanipulasi sehingga dapat merusak nama baik seseorang atau bahkan suatu organisasi tertentu. Hal ini tentu saja melanggar hak privasi setiap orang. Dengan demikian usaha dalam mengamankan data digital menjadi hal yang sangat penting dan sangat mendesak.

Adapun masalah keamanan dalam suatu data atau informasi dapat diatasi salah satunya dengan menerapkan kriptografi, yakni ilmu atau seni yang menggunakan matematika untuk mengamankan suatu informasi. Pengamanan ini dilakukan dengan menjalankan algoritma enkripsi (mengubah informasi awal menjadi informasi baru yang disamarkan dengan menggunakan suatu kunci) dan dekripsi (mengubah kembali menjadi informasi awal).

Berbagai algoritma enkripsi telah diciptakan oleh para ahli kriptografi, namun banyak pula upaya-upaya untuk memecahkannya dan tidak sedikit yang berhasil. Hal ini tentu saja memicu para kriptografer untuk menciptakan algoritma yang jauh lebih aman.

Hingga tahun 1990-an, algoritma enkripsi yang banyak dipakai adalah algoritma DES (*Data Encryption Standard*). Namun seiring dengan makin canggihnya teknologi dan berkembangnya dunia *cryptanalysis* (ilmu memecah sandi) maka keamanan data dengan algoritma DES yang hanya menggunakan kunci sepanjang 56 bit dianggap tidak memadai lagi. Karena itu pada tahun 2000, terpilihlah algoritma Rijndael sebagai standar algoritma kriptografi baru pengganti algoritma DES, yang akhirnya dinamakan algoritma AES (*Advanced Encryption Standard*) setelah melewati tahap seleksi pada kompetisi internasional yang diadakan NIST (*National Institute of Standard and Technology*) (Stallings,2005).

Sebagai pemenang AES, algoritma Rijndael dianggap aman dan juga efisien dalam implementasinya. Maka itu dalam tugas akhir ini akan dibahas aplikasi algoritma Rijndael dalam pengamanan data berupa citra digital atau foto dan implementasinya dengan *software* Matlab.

1.2 Perumusan Masalah dan Ruang Lingkup

Permasalahan yang menjadi bahasan dalam tugas akhir ini adalah bagaimana konsep kerja dan gambaran umum algoritma Rijndael dalam mengamankan suatu data citra digital dan bagaimana implementasinya?

Ruang lingkup dalam tugas akhir ini adalah sebagai berikut:

a. Citra digital yang akan dirahasiakan berupa citra hitam putih atau *grayscale*
b. Kunci yang akan dipakai dalam proses enkripsi-dekripsi sepanjang 128 bit
c. Program yang akan digunakan untuk mengimplementasikan algoritma Rijndael adalah Matlab.

1.3 Tujuan Penulisan

a. Menjelaskan konsep serta gambaran umum dari algoritma Rijndael
b. Mengimplementasikan algoritma Rijndael untuk mengamankan data citra digital

BAB 2
LANDASAN TEORI

Konsep dasar matematika banyak dipakai di dunia nyata, salah satunya adalah aplikasi konsep matematika dalam dunia penyandian. Dalam algoritma Rijndael, proses enkripsi dan dekripsinya memakai dasar dan teorema matematika dalam hal ini aljabar. Sebelum masuk ke dalam pembahasan algoritma Rijndael yang diimplementasikan untuk mengamankan data citra digital, akan dijelaskan konsep dasar matematika yang dipakai dalam algoritma ini terlebih dahulu, selain itu akan dijelaskan tentang definisi citra digital.

2.1 Kriptografi dan Penyandian

Kriptografi (*cryptography*) berasal dari bahasa Yunani yaitu *cryptos* yang artinya rahasia dan *graphein* yang artinya tulisan. Jadi, kriptografi adalah tulisan rahasia. Kriptografi adalah ilmu atau seni yang menggunakan matmatika untuk mengamankan suatu informasi.

Algoritma kriptografi merupakan aturan untuk *enchipering* dan *deciphering* atau fungsi matematika yang digunakan untuk melakukan proses enkripsi dan dekripsi. Algoritma kriptografi terdiri dari tiga fungsi dasar yaitu enkripsi, dekripsi, dan kunci.

Enkripsi atau penyandian adalah suatu proses pengubahan *plaintext* (informasi awal yang akan dirahasiakan) menjadi *ciphertext* (informasi yang sudah dirahasiakan) dengan sebuah algoritma tertentu dan menggunakan suatu kunci tertentu yang dipilih. Sementara dekripsi adalah proses pengembalian *ciphertext* menjadi *plaintext* oleh algoritma yang berkebalikan dengan algoritma enkripsi.

Konsep matematis yang menjadi dasar kriptografi adalah hubungan antara dua buah himpunan yang elemennya adalah *plaintext* dan himpunan lain yang elemennya adalah *ciphertext*. Fungsi yang memetakan kedua himpunan tersebut

dinamakan fungsi enkripsi dan fungsi dekripsi. Dimisalkan P menyatakan *plaintext* dan C menyatakan *ciphertext*, maka fungsi enkripsi E akan memetakan P ke C,

$$E(P) = C$$

dan fungsi dekripsi D akan memetakan C ke P,

$$D(C) = P$$

Maka, persamaan $D(E(P)) = P$ harus benar dikarenakan proses enkripsi kemudian dekripsi mengembalikan lagi pesan menjadi pesan awal. Pernyataan ini akan dijamin oleh Teorema 2.1

Definisi 2.1 (Herstein, 1995)

Fungsi $f : S \to T$ dikatakan fungsi injektif jika $\forall\, s_1, s_2 \in S\,, s_1 \neq s_2$ maka $f(s_1) \neq f(s_2)$ di T .
Fungsi $f : S \to T$ dikatakan fungsi surjektif $\leftrightarrow \forall\, t \in T, \exists\, s \in S$ sedemikian sehingga $t = f(s)$.
Fungsi $f : S \to T$ dikatakan bijektif $\leftrightarrow$ f fungsi injektif dan surjektif.

Teorema 2.1 (Rosen, 1999)

Jika $f: A \to B$ adalah fungsi bijektif, maka terdapat sebuah fungsi invers dari f yaitu $f^{-1}: B \to A$ yang bijektif pula.

Pembuktian Teorema 2.1 terdapat pada Lampiran 1.

Teorema 2.1 menunjukkan bahwa sifat bijektif dari enkripsi akan menjamin bahwa setiap *plaintext* yang disandikan akan dipetakan ke satu dan hanya satu *ciphertext* dan juga setiap *ciphertext* merupakan hasil enkripsi satu dan hanya satu *plaintext*. Jadi, sifat bijektif menjamin adanya *invers* dari proses

enkripsi yang dapat mengembalikan *ciphertext* menjadi *plaintext* yang bersesuaian.

Definisi 2.2 (Herstein, 1995)

Jika $g : S \to T$ dan $f: T \to U$, maka komposisi fungsi (dinotasikan dengan $f \circ g$) adalah fungsi $f \circ g: S \to U$ didefinisikan oleh $(f \circ g)(s) = f\big(g(s)\big)\ \forall s \in S$.

Teorema 2.2 (Herstein, 1995)

Jika $g: S \to T$ dan $f: T \to U$ fungsi yang bijektif, maka $f \circ g: S \to U$ juga fungsi yang bijektif

Teorema 2.3 (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 $(f \circ g)^{-1} = g^{-1} \circ f^{-1}$

Penjelasan mengenai Teorema 2.2 dan 2.3 terdapat pada Lampiran 2. Berdasarkan Teorema 2.2 dan 2.3, maka didapatkan Akibat 2.4 dan Akibat 2.5:

Akibat 2.4

Jika $f_i : A_{i-1} \to A_i$ fungsi bijektif $\forall i = 1,2, \dots, n,$ maka $(f_n \circ f_{n-1} \circ \dots \circ f_2 \circ f_1)$ juga bijektif.

Akibat 2.5

Jika $f_i : A_{i-1} \to A_i$ fungsi bijektif $\forall i = 1,2, \dots, n,$ maka $(f_n \circ f_{n-1} \circ \dots \circ f_2 \circ f_1)^{-1}$ juga bijektif dan $(f_n \circ f_{n-1} \circ \dots \circ f_2 \circ f_1)^{-1} = f_1^{-1} \circ f_2^{-1} \circ \dots \circ f_{n-1}^{-1} \circ f_n^{-1}$.

Kedua cipher A_{i-1} dan A_i pada Teorema 2.4 dan 2.5 disebut juga *product cipher*, yakni *cipher* yang merupakan komposisi dari n buah fungsi yang bijektif, yaitu:

$$E = E_n \circ E_{n-1} \circ \ldots \circ E_2 \circ E_1$$

Dalam suatu algoritma enkripsi, untuk meningkatkan keamanan data, dibutuhkan proses enkripsi yang dilakukan berkali-kali atau dengan melakukan beberapa iterasi. Teorema 2.4 menunjukkan bahwa proses enkripsi yang menggunakan beberapa iterasi ini merupakan fungsi yang bijektif.

Berdasarkan Teorema 2.5, suatu proses enkripsi yang menggunakan beberapa iterasi memiliki *invers* yang juga bijektif dan hasil enkripsinya dapat didekripsi secara berurutan dari proses terakhir ke proses awal hingga menghasilkan suatu *plaintext* yang bersesuaian.

$$D = E^{-1} = E_1^{-1} \circ E_2^{-1} \circ \ldots \circ E_{n-1}^{-1} \circ E_n^{-1}$$

Hasil *intermediate* dari *product cipher* yaitu hasil transformasi *plaintext* setelah dikenakan $E_1, E_2, \ldots$, sampai E_k , dengan $k = 1,2, \ldots, n-1$. Sementara, hasil yang diperoleh dari semua transformasi pada *product cipher* yang ada disebut *ciphertext*.

2.2 Konsep Matematika dalam Algoritma Rijndael

Terdapat beberapa operasi utama dalam algoritma Rijndael yaitu operasi XOR dan transformasi linier. Pada algoritma Rijndael, setiap *byte* akan direpresentasikan sebagai elemen dari *Galois Field* 2^8. Operasi-operasi pada algoritma Rijndael sebagian besar dilakukan pada level *byte* dan sebagian lagi pada level *word* atau 4 *byte* (1 *byte* = 8 bit). Sebelum membahas *Galois Field*, akan dibahas terlebih dahulu dasar-dasar operasi matematika yang digunakan pada algoritma Rijndael.

2.2.1 Operasi XOR

Pada algoritma Rijndael, terdapat beberapa transformasi yang membutuhkan operasi XOR didalamnya, berikut diberikan definisi dari operasi XOR.

Definisi 2.3 (Rosen, 1999)

p dan q adalah sebuah proposisi. Eksklusif OR dari p dan q, yang dinotasikan oleh $p \oplus q$, adalah suatu proposisi yang bernilai benar jika salah satu dari p dan q adalah benar dan bernilai salah jika keduanya bernilai benar atau salah.

Tabel 2.1 menunjukkan tabel kebenaran operasi XOR terhadap dua proposisi.

Tabel 2. 1 Tabel Kebenaran Operasi XOR

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

Pada pemrosesan suatu algoritma, unit terkecil adalah bit yang merupakan elemen dalam $\{0,1\}$. Pernyataan yang salah dinyatakan dalam bit 0 dan pernyataan yang benar dinyatakan dalam bit 1. Tabel 2.1 dapat direpresentasikan dalam bit menjadi seperti dalam Tabel 2.2.

Tabel 2. 2 Operasi XOR dalam Representasi Bit

p	q	$p \oplus q$
1	1	0
1	0	1
0	1	1
0	0	0

2.2.1 Transformasi Linier

Pada algoritma Rijndael terdapat transformasi yang menggabungkan elemen tiap kolomnya dengan menggunakan transformasi linier. Berikut diberikan definisi dari transformasi linier.

Definisi 2.4 (Anton, 1995)

Jika $F: \boldsymbol{V} \rightarrow \boldsymbol{W}$ adalah sebuah fungsi dari ruang vektor $\boldsymbol{V}$ ke dalam ruang vektor $\boldsymbol{W}$, maka F dinamakan transformasi linier jika:

$(i)\ F(\boldsymbol{u}+\boldsymbol{v}) = F(\boldsymbol{u}) + F(\boldsymbol{v})\ \forall\ \boldsymbol{u}, \boldsymbol{v} \in \boldsymbol{V}$.

$(ii)\ F(c\boldsymbol{u}) = c\ F(\boldsymbol{u})\ \forall\ \boldsymbol{u} \in \boldsymbol{V}$ dan untuk skalar c.

Dalam algoritma Rijndael, terdapat suatu transformasi yang mengalikan setiap kolom dari suatu matriks dengan suatu polinomial yang direpresentasikan dalam perkalian matriks transformasi linier. Pandang sistem persamaan linier 2.1:

$$\left.\begin{array}{l} w_1 = a_{11}v_1 + a_{12}v_2 + \cdots + a_{1n}v_n \\ w_2 = a_{21}v_1 + a_{22}v_2 + \cdots + a_{2n}v_n \\ \vdots \quad\quad \vdots \quad\quad \vdots \quad\quad\quad\quad \vdots \\ w_n = a_{n1}v_1 + a_{n2}v_2 + \cdots + a_{nn}v_n \end{array}\right\} \quad (2.1)$$

Sistem persamaan 2.1 dapat ditulis dalam bentuk:

$$\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

Akan dibuktikan bahwa $\boldsymbol{w} = A\,\boldsymbol{v}$ adalah suatu transformasi linier $F: \boldsymbol{V} \rightarrow \boldsymbol{W}$ berdasarkan Definisi 2.4 dengan $\boldsymbol{V}$ adalah basis GF (2^8) dan $\boldsymbol{W}$ adalah basis GF (2^8).

Akan ditunjukkan: $(i)\ A(\boldsymbol{u}+\boldsymbol{v}) = A(\boldsymbol{u}) + A(\boldsymbol{v})\ \forall\, \boldsymbol{u}, \boldsymbol{v} \in \boldsymbol{V}$.

$$A\,(\boldsymbol{u}+\boldsymbol{v}) = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} u_1 + v_1 \\ u_2 + v_2 \\ \vdots \\ u_n + v_n \end{bmatrix}$$

$$= \begin{bmatrix} (u_1 + v_1)a_{11} + (u_2 + v_2)a_{12} + \cdots + (u_n + v_n)a_{1n} \\ (u_1 + v_1)a_{21} + (u_2 + v_2)a_{22} + \cdots + (u_n + v_n)a_{2n} \\ \vdots \\ (u_1 + v_1)a_{n1} + (u_2 + v_2)a_{n2} + \cdots + (u_n + v_n)a_{nn} \end{bmatrix}$$

$$= \begin{bmatrix} u_1a_{11} + v_1a_{11} + u_2a_{12} + v_2a_{12} + \cdots + u_na_{1n} + v_na_{1n} \\ u_1a_{21} + v_1a_{21} + u_2a_{22} + v_2a_{22} + \cdots + u_na_{2n} + v_na_{2n} \\ \vdots \\ u_1a_{n1} + v_1a_{n1} + u_2a_{n2} + v_2a_{n2} + \cdots + u_na_{nn} + v_na_{nn} \end{bmatrix}$$

$$= \begin{bmatrix} (u_1a_{11} + u_2a_{12} + \cdots + u_na_{1n}) + (v_1a_{11} + v_2a_{12} + \cdots + v_na_{1n}) \\ (u_1a_{21} + u_2a_{22} + \cdots + u_na_{2n}) + (v_1a_{21} + v_2a_{22} + \cdots + v_na_{2n}) \\ \vdots \\ (u_1a_{n1} + u_2a_{n2} + \cdots + u_na_{nn}) + (v_1a_{n1} + v_2a_{n2} + \cdots + v_na_{nn}) \end{bmatrix}$$

$$= \begin{bmatrix} (u_1a_{11} + u_2a_{12} + \cdots + u_na_{1n}) \\ (u_1a_{21} + u_2a_{22} + \cdots + u_na_{2n}) \\ \vdots \\ (u_1a_{n1} + u_2a_{n2} + \cdots + u_na_{nn}) \end{bmatrix} + \begin{bmatrix} (v_1a_{11} + v_2a_{12} + \cdots + v_na_{1n}) \\ (v_1a_{21} + v_2a_{22} + \cdots + v_na_{2n}) \\ \vdots \\ (v_1a_{n1} + v_2a_{n2} + \cdots + v_na_{nn}) \end{bmatrix}$$

$$= \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} + \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

$A(\boldsymbol{u}+\boldsymbol{v}) = A(\boldsymbol{u}) + A(\boldsymbol{v})\ \forall\, \boldsymbol{u}, \boldsymbol{v} \in \boldsymbol{V}$. (Terbukti)

Akan ditunjukkan: $(ii)\ A(c\boldsymbol{u}) = c\ A(\boldsymbol{u})\ \forall\ \boldsymbol{u} \in \boldsymbol{V}$ dan untuk skalar c.

$$A(c\boldsymbol{u}) = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} cu_1 \\ cu_2 \\ \vdots \\ cu_n \end{bmatrix}$$

$$= \begin{bmatrix} (cu_1)a_{11} + (cu_2)a_{12} + \cdots + (cu_n)a_{1n} \\ (cu_1)a_{21} + (cu_2)a_{22} + \cdots + (cu_n)a_{2n} \\ \vdots \\ (cu_1)a_{n1} + (cu_2)a_{n2} + \cdots + (cu_n)a_{nn} \end{bmatrix}$$

$$= \begin{bmatrix} c(u_1a_{11}) + c(u_2a_{12}) + \cdots + c(u_na_{1n}) \\ c(u_1a_{21}) + c(u_2a_{22}) + \cdots + c(u_na_{2n}) \\ \vdots \\ c(u_1a_{n1}) + c(u_2a_{n2}) + \cdots + c(u_na_{nn}) \end{bmatrix}$$

$$= \begin{bmatrix} c(u_1a_{11}) + u_2a_{12} + \cdots + u_na_{1n}) \\ c(u_1a_{21}) + u_2a_{22} + \cdots + u_na_{2n}) \\ \vdots \\ c(u_1a_{n1}) + u_2a_{n2} + \cdots + u_na_{nn}) \end{bmatrix}$$

$$= c \begin{bmatrix} (u_1a_{11}) + u_2a_{12} + \cdots + u_na_{1n}) \\ (u_1a_{21}) + u_2a_{22} + \cdots + u_na_{2n}) \\ \vdots \\ (u_1a_{n1}) + u_2a_{n2} + \cdots + u_na_{nn}) \end{bmatrix}$$

$$= c \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix}$$

$A(c\boldsymbol{u}) = c\ A(\boldsymbol{u})\ \ \forall\ \boldsymbol{u} \in \boldsymbol{V}$ dan untuk skalar c. (Terbukti)

Berdasarkan Definisi 2.4, maka terbukti bahwa $\boldsymbol{w} = A\ \boldsymbol{v}$ adalah suatu transformasi linier.

2.2.2 Grup , *Ring,* dan *Field.*

Grup, *Ring*, dan *Field* merupakan struktur dasar dari aljabar abstrak dan juga merupakan dasar struktur aljabar yang diperlukan dalam berbagai definisi

dan analisis aljabar dari algoritma AES: Rijndael. Berikut akan dijelaskan definisi tentang Grup, *Ring,* dan *Field* (Stallings, 2005).

Definisi 2.6 : Himpunan tak kosong G dikatakan Grup jika G didefinisikan pada operasi biner * sedemikian sehingga:

a) $a, b \in G$ maka $a * b \in G$ (G tertutup terhadap$*$)
b) Untuk semua $a, b, c \in G$ berlaku $a * (b * c) = (a * b) * c$ (G bersifat assosiatif)
c) Terdapat $e \in G$ sedemikian sehingga untuk semua $a \in G$ berlaku $a * e = e * a = a.$ (e unik dan dinamakan elemen identitas dari G)
d) Untuk semua $a \in G$, terdapat $b \in G$ sedemikian sehingga $a * b = b * a = e.$ (elemen b sebagai a^{-1} dan dinamakan invers dari a di G)

Suatu Grup G dinamakan Grup Abelian jika G merupakan Grup dan memenuhi sifat komutatif, yaitu $a * b = b * a$

Definisi 2.7 : Himpunan tak kosong R dikatakan *Ring* jika terdapat operasi (+) dan (**.**) pada R sedemikian sehingga untuk semua $a, b, c \in R$ memenuhi:

a) R adalah Grup Abelian terhadap +
b) Untuk semua $a, b \in R$ maka $a.b \in R$ (tertutup terhadap (**.**))
c) $a.(b.c) = (a.b).c$ untuk semua $a, b, c \in R$ (assosiatif terhadap (**.**))
d) $a.(b + c) = a.b + a.c$ dan $(b + c).a = b.a + c.a$ untuk semua $a, b, c \in R$ (distributif)

Suatu *Ring* R dikatakan komutatif jika R merupakan Ring dan memenuhi sifat $a.b = b.a$ untuk semua $a, b \in R.$

Suatu *Ring* R dikatakan Integral domain yang komutatif *Ring* jika R merupakan *Ring* yang komutatif dan memenuhi:

a) Terdapat elemen $1 \in R$ sedemikian sehingga $a.1 = 1.a = a$ untuk semua $a \in R$
b) Jika $a, b \in R$ dan $a.b = 0$ maka $a = 0$ atau $b = 0$

Definisi 2.8 : Himpunan tak kosong F dikatakan *Field* jika terdapat operasi + dan **.** pada F sedemikian sehingga untuk semua $a, b, c \in F$ memenuhi:

a) F merupakan *integral domain*
b) Untuk setiap $a \in F$ kecuali 0 , terdapat elemen $a^{-1} \in F$ sedemikian sehingga $a.a^{-1} = a^{-1} = 1$ (Invers terhadap perkalian)

2.2.3 *Galois Field* 2^8

Rancangan dari algoritma AES: Rijndael didasari dari *finite fields.* Berikut akan dijelaskan mengenai *finite fields* dan operasi-operasi didalamnya yang dipakai di dalam proses algoritma Rijdael.

Definisi 2.9 (Stallings, 2005)

Jika sebuah *Field* F memiliki jumlah elemen yang berhingga, maka F disebut *Finite Fields* dan jika banyaknya jumlah elemen dalam *Field* F tak berhingga, maka F disebut *Infinite Fields.*

Definisi 2.10 (Rosen, 1999)

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

Definisi 2.10 dapat dikatakan definisi dari sisa $(a \ mod \ m)$ adalah bilangan positif r sedemikian sehingga $a = qm + r$ dan $0 \leq r < m$

Definisi 2.11 (Stallings, 2005)

Diberikan p bilangan prima, *Finite Field* order p atau GF (p) didefinisikan sebagai himpunan bilangan bulat $\{0,1, \dots, p-1\}$ dengan operasi aritmatika modulo p.

Definisi 2.12 (Stallings, 2005)

Galois Field 2^n merupakan suatu himpunan yang jumlah elemennya terhingga, yakni sebanyak 2^n buah dan memenuhi:

a) Setiap elemen dari GF (2^n) adalah polinomial dengan order $n-1$:

$$a(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1x + a_0$$

dan dapat dinyatakan dalam bilangan biner n bit yang terdiri dari $a_n, a_{n-1}, \dots, a_2, a_1, a_0$ dengan $a_i \in$ GF(2)

b) Suatu *Rings* terhadap operasi penjumlahan dan perkalian

Dalam algoritma Rijndael, semua operasi penjumlahan dan perkalian tercakup dalam ruang lingkup *Galois Field* 2^8, maka setiap elemennya adalah polinomial dengan order 7:

$$a(x) = a_7x^7 + a_6x^6 + \cdots + a_1x + a_0$$

dan dapat dinyatakan dalam bilangan biner 8 bit yang terdiri dari $a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0$. Elemen GF (2^8) juga dapat dinyatakan dalam bilangan desimal dari 0 sampai 255 atau bilangan heksadesimal dari '00' sampai 'ff'.

Contoh elemen GF (2^8) : $242 \equiv 'F2' \equiv 11110010 \equiv x^7 + x^6 + x^5 + x^4 + x$

Operasi pada GF (2^8) yang terkait dengan algoritma Rijndael yaitu operasi penjumlahan, perkalian, dan invers terhadap perkalian. Operasi penjumlahan direpresentasikan dalam GF (2^8) yang operasinya sama dengan penjumlahan pada

polinomial biasa namun masing-masing koefisiennya dimodulo 2 atau dilakukan operasi XOR.

Contoh operasi penjumlahan dalam GF (2^8) :

$(x^7 + x^3 + x + 1) + (x^6 + x^3 + x^2 + x) = (x^7 + x^6 + x^2 + 1)$

$\equiv 10001011 \oplus 01001110 = 11000101$

Operasi perkalian dalam GF(2^8) merupakan perkalian polinomial biasa dengan koefisien-koefisiennya dimodulo 2 dan polinomial tersebut akan direduksi jika dihasilkan polinomial dengan ordernya lebih besar dari 7. Proses pereduksiannya adalah dengan memodulokannya dengan suatu polinomial f(x) berorder 8 yang *irreducible.*

Definisi 2.13 (Ayres dan Mendelson, 2004)

Polinomial $p(x) \in F[x]$ ($F[x]$ menyatakan polinomial *Ring* atas *Field F*) disebut irreducible jika $p(x)$ berderajat positif dan $p(x)$ tidak dapat dinyatakan sebagai perkalian antara dua polinomial berderajat positif. Dengan kata lain jika $p(x) = a(x)b(x)$ maka $a(x)$ konstan atau $b(x)$ konstan.

$f(x)$ harus *irreducible* agar semua anggota GF(2^8) *invertible*, hal ini akan dibahas pada bagian invers terhadap perkalian. Khusus pada algoritma Rijndael, dipilih polinomial *irreducible*nya adalah: $f(x) = x^8 + x^4 + x^3 + x + 1$. (Stallings, 2005)

Contoh operasi perkalian dalam GF(2^8) :

$(x^3 + x^2) \bullet (x^6 + x^4 + x) = x^9 + x^8 + x^7 + x^6 + x^4 + x^3$

$(x^9 + x^8 + x^7 + x^6 + x^4 + x^3) mod\ (x^8 + x^4 + x^3 + x + 1)$

$= (x^7 + x^6 + x^5 + x^4 + x^2 + 1)$

Definisi 2.14 (Stallings, 2005)

Misalkan b adalah bilangan bulat tidak nol yang merupakan pembagi dari a jika $a = mb$ untuk beberapa m, dengan $a, b,$ dan m adalah bilangan bulat tidak nol. Notasi $gcd(a, b)$ dikatakan *greatest common divisors* dari a dan b. Bilangan positif c dikatakan *greatest common divisors* dari a dan b jika:

1. c adalah pembagi dari a dan b
2. Setiap pembagi dari a dan b adalah pembagi dari c

Definisi 2.15 (Stallings, 2005)

Polinomial $c(x)$ merupakan *greatest common divisors* dari $a(x)$ dan $b(x)$ jika:

1. $c(x)$ adalah pembagi dari $a(x)$ dan $b(x)$
2. Setiap pembagi dari $a(x)$ dan $b(x)$ adalah pembagi dari $c(x)$

Invers dari $a(x) mod\ f(x) \in$ GF (2^8) terhadap ● adalah $b(x)\ mod\ f(x) \in$ GF (2^8), sedemikian sehingga: $a(x) \bullet b(x)\ mod\ f(x) = 1$ atau $\gcd(b(x), f(x)) = 1$.

Definisi 2.16 (Herstein, 1995)

$\gcd(a(x), c(x)) = 1$ berarti terdapat $p(x)$ dan $q(x)$ dimana $a(x)p(x) + c(x)\ q(x) = 1$. Berarti bahwa polinomial $a(x)$ tidak mereduksi polinomial $c(x)$ dan sebaliknya.

Teorema 2.6 (Herstein, 1995)

Jika $\gcd(a(x), c(x)) = 1$ maka $a(x)$ *invertible* $mod\ c(x)$

Keberadaan elemen *invers* pada anggota di GF (2^8) dijamin oleh teorema 2.6. Hal ini karena $c(x)$ dalam GF (2^8) adalah $f(x)$, yaitu polinomial yang

irreducible, maka $f(x)$ tidak memiliki faktor selain 1 dan dirinya sendiri. Jadi, untuk semua $a(x) \in$ GF (2^8) , jika $\gcd(a(x), f(x)) = 1$ maka $a(x)$ memiliki invers. Penjelasan mengenai Teorema 2.6 akan dibahas di Lampiran 3.

Semua jenis data digital atau multimedia dapat dilakukan pengamanan dengan menggunakan algoritma Rijndael. Jika sebelumnya Eko Satria dalam tugas akhirnya telah membahas pengamanan data berupa *file-file Microsoft Office* dengan menggunakan algoritma Rijndael, juga Hendry Iskandar dan Didi Surian yang telah membahas algoritma Rijndael dalam mengamankan data teks, maka tentu saja algoritma ini pun mampu mengamankan data digital lain, yaitu citra digital. Akan dibahas definisi dan pengenalan citra digital.

2.3 Citra Digital

Citra atau gambar merupakan salah satu komponen multimedia yang memiliki peranan penting sebagai bentuk informasi visual. Citra memiliki karakteristik istimewa dibanding data teks, yaitu citra kaya akan informasi. Sebuah citra dapat memberikan informasi yang lebih banyak daripada ketika informasi itu disajikan dalam bentuk teks.

Secara harfiah, citra adalah gambar pada bidang dua dimensi. Citra merupakan fungsi kontinu dari intensitas cahaya pada bidang dua dimensi. Secara matematis, fungsi intensitas cahaya pada bidang dua dimensi disimbolkan dengan $f(x, y)$, dalam hal ini (x, y) merupakan koordinat pada bidang dua dimensi dan $f(x, y)$ merupakan intensitas cahaya pada titik (x, y). (Munir, 2004)

Citra digital tersusun dari sejumlah nilai tingkat keabuan yaitu piksel pada posisi tertentu. Piksel adalah elemen terkecil dari sebuah citra digital. Piksel mempunyai dua parameter, yaitu koordinat dan warna atau intensitas. Koordinat adalah lokasi suatu piksel dari citra digital. Warna adalah intensitas cahaya yang dipantulkan oleh obyek.

Citra terbagi menjadi tiga jenis, yaitu citra biner, citra *grayscale*, dan citra warna. Citra biner merupakan citra yang hanya tersusun dari dua warna dan tiap pikselnya direpresentasikan dalam bit 0 dan 1 yang mana bit 0 artinya hitam dan bit 1 artinya putih. Citra *grayscale* merupakan citra yang intensitasnya bergerak antara 0 sampai 255 yang terdiri dari warna hitam dan putih dan juga perpaduan dari keduanya. Sedangkan citra berwarna adalah citra yang tersusun dari tiga layer yang terdiri dari intensitas merah, hijau, dan biru juga perpaduan ketiganya. Yang akan dibahas kali ini adalah citra *grayscale*.

Tingkat ketajaman / resolusi warna pada citra digital tergantung pada jumlah ”bit” yang digunakan oleh komputer untuk merepresentasikan setiap piksel tersebut. Tipe yang sering digunakan untuk merepresentasikan citra adalah ”8-bit citra”. Tingkat intensitas pada citra *grayscale* adalah 256 dan nilai dari intensitas bentuknya adalah diskrit mulai dari 0 sampai 255. 0 menyatakan warna hitam dan 255 menyatakan warna putih.

Citra yang ditangkap kamera dan telah dikuantisasi dalam bentuk nilai diskrit disebut citra digital. Pada umumnya citra digital berbentuk empat persegi panjang dan dimensi ukurannya dinyatakan sebagai tinggi x lebar.

Citra digital yang berukuran $N\ x\ M$ dinyatakan dengan matriks yang berukuran N baris dan M kolom, yaitu:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,M-1) \\ f(1,0) & f(1,1) & \dots & f(1,M-1) \\ \vdots & \vdots & \vdots & \vdots \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1,M-1) \end{bmatrix}$$

Sehingga citra yang berukuran $N\ x\ M$ memiliki $N\ x\ M$ buah piksel.

BAB 3
KONSEP ALGORITMA AES: RIJNDAEL

Algoritma Rijndael merupakan jenis algoritma kriptografi yang sifatnya simetri dan *cipher block*. Dengan demikian algoritma ini menggunakan kunci yang sama pada saat enkripsi dan dekripsi serta *input* dan *output*nya berupa blok dengan jumlah bit tertentu. Algoritma Rijndael mendukung berbagai variasi ukuran kunci yang akan digunakan. Namun algoritma Rijndael mempunyai ukuran kunci yang tetap sebesar 128, 192, dan 256 bit. Pemilihan ukuran kunci akan menentukan jumlah iterasi yang harus dilalui untuk proses enkripsi dan dekripsi.

Ukuran blok untuk algoritma Rijndael adalah 128 bit atau 16 *byte*. Jumlah iterasi dalam proses enkripsi dan dekripsi dipengaruhi oleh ukuran kunci yang akan dipakai. Misalkan N_b adalah panjang blok dibagi 32 dan N_k adalah panjang kunci dibagi 32, maka jumlah iterasinya:

$$N_r = max\,(N_k, N_b) + 6 \qquad (3.1)$$

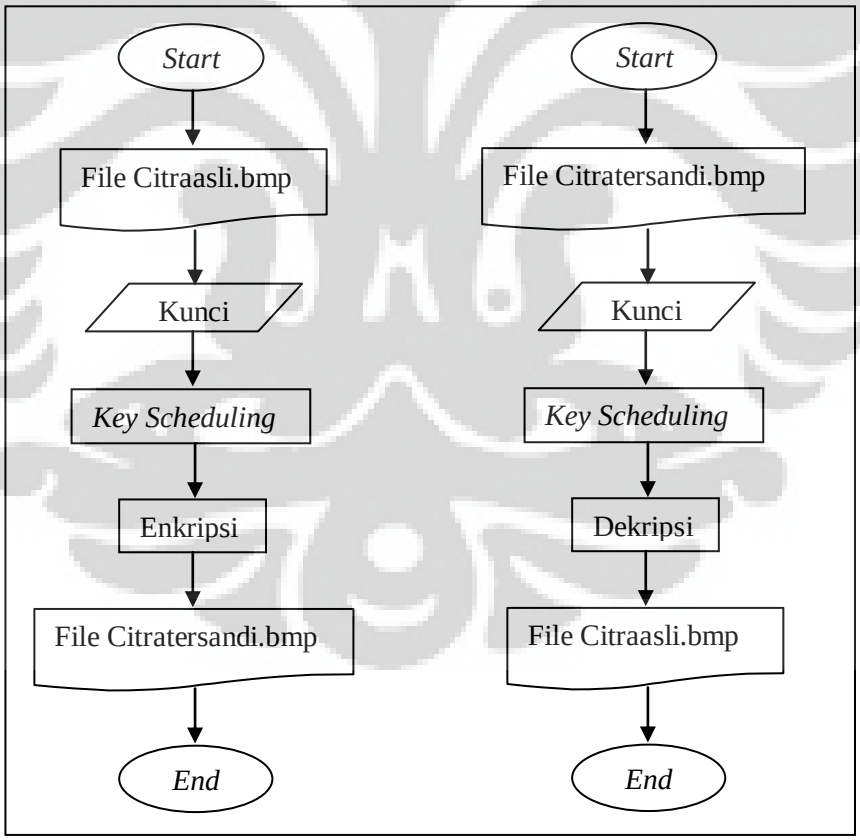
Berdasarkan persamaan 3.1, dihasilkan Tabel 3.1 yang menunjukkan perbandingan jumlah iterasi pada tiap input kunci yang berbeda. *Words* merupakan matriks 4 x 1 yang terdiri dari 4 *byte* elemen.

Tabel 3. 1 Perbandingan Jumlah Iterasi Algoritma Rijndael

Versi AES	Panjang Kunci (Nk)	Panjang Blok (Nb)	Jumlah Iterasi (Nr)
AES – 128 bit	4 *words*	4 *words*	10
AES – 192 bit	6 *words*	4 *words*	12
AES – 256 bit	8 *words*	4 *words*	14

[Sumber: Satria, 2009]

Dalam tugas akhir ini akan dibahas algoritma Rijndael menggunakan kunci sebesar 128 bit. Tabel 3.1 menunjukkan, dengan menggunakan blok cipher sebesar 128 bit dan kunci sebesar 128 bit, maka dalam proses enkripsi dan dekripsinya, algoritma Rijndael melakukan 10 iterasi. Blok-blok data masukan dan kunci dioperasikan dalam bentuk *array*. Bila data yang akan dienkripsi bukan 128 bit atau kelipatannya, maka pada akhir blok, akan ditambahkan bit yang bernilai 0 pada blok tersebut sampai menjadi 128 bit atau kelipatannya. Setiap anggota *array* sebelum menghasilkan *output ciphertext* dinamakan dengan *state*. Setiap *state* akan mengalami proses yang secara garis besar terdiri dari empat tahap yaitu, *AddRoundKey*, *SubBytes*, *ShiftRows*, dan *MixColumns*. Semua tahap akan diulang pada 9 iterasi awal, sedangkan tahap *MixColumns* tidak akan dilakukan pada iterasi terakhir (iterasi ke 10). Langkah-langkah algoritma Rijndael meliputi *key schedule* dan enkripsi/dekripsi terhadap blok-blok *plaintext / ciphertext*, sebagaimana tampak pada Gambar 3.1.



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

3.1. *Key Schedule*

Tahap *key schedule* bertujuan membangun 10 sub-kunci yang akan digunakan pada tiap iterasi tahap enkripsi dan dekripsi. Di awal akan dimasukkan suatu kunci yang disebut *cipher key*, yang seterusnya akan dilakukan ekspansi terhadap *cipher key* tersebut.

Terlebih dahulu ditetapkan panjang kunci adalah 128 bit, jika kunci yang digunakan kurang dari 128 bit, maka dilakukan penambahan bit-bit "0" pada akhir kunci, sehingga panjangnya mencapai 128 bit yang diinginkan. Jika kunci yang digunakan lebih dari 128 bit, maka akan diambil 128 bit pertama sebagai *cipher key*.

Pada *cipher key* ini dilakukan partisi blok 128 bit dari kunci menjadi 16 nilai masing-masing dua karakter heksadesimal dan direpresentasikan ke dalam matriks 4x4, disebut matriks kunci. Kolom pertama, kedua, ketiga dan, keempat dari matriks kunci tersebut merepresentasikan *word* ke-0, *word* ke-1, *word* ke-2, dan *word* ke-3 yang masing-masing berukuran 4 *byte* atau 32 bit.

Pada kasus ini akan dibangkitkan 40 *word* dengan menggunakan *cipher key* sebagai *input* kunci, yang selanjutnya akan dipakai pada tiap iterasi pada proses enkripsi. Untuk menghasilkan *word* ke-4 atau kolom pertama pada sub-kunci pertama (W_i), maka prosesnya terdiri dari beberapa operasi yang berurutan yaitu:

a. Operasi *RotWord* atau Rotasi pada *Word* yaitu operasi perputaran 8 bit pada 32 bit W_{i-1} dengan cara pergeseran kolom secara siklis ke atas. Contoh: Input *word* (a_0, a_1, a_2, a_3) menjadi (a_1, a_2, a_3, a_0)

b. Operasi *SubWord* atau Substitusi pada *Word*, yaitu operasi substitusi 8 bit pada hasil *RotWord* dengan nilai dari S-Box. S-Box pada proses ini adalah S-Box yang sama yang akan dipakai pada transformasi *SubByte* pada proses enkripsi.

c. Proses XOR hasil *SubWord* dengan suatu nilai konstan R-Con yang bersesuaian tiap round.

$$Rcon\,[\,j] = (RC[\,j], 0, 0, 0)$$

dengan $RC[1] = 01$

$$RC[\,j] = 2 * RC[\,j-1]$$

Dengan sifat operasi * merupakan perkalian yang terdefinisikan pada GF(2^8). Akan dihasilkan R-Con yang bersesuaian yaitu:

$$\begin{bmatrix} 01 & 02 & 04 & 08 & 10 & 20 & 40 & 80 & 1B & 36 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 \end{bmatrix}$$

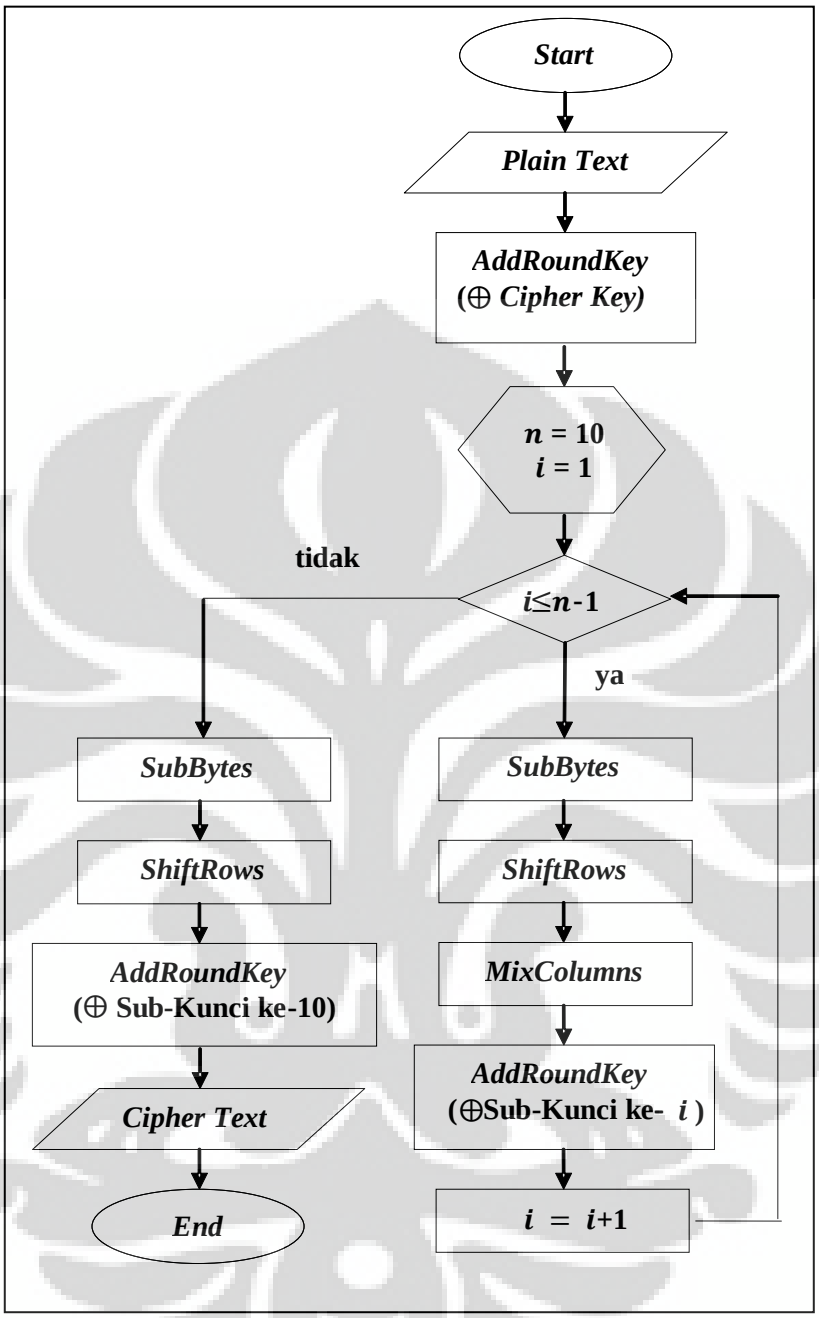
(Satria, 2009)

d. Proses XOR antara *word* hasil proses sebelumnya dengan W_{i-4}

Untuk menghasilkan *word* ke-5 (W_{i+1}) cukup dilakukan proses XOR antara W_i dengan W_{i-3}. Proses yang sama untuk mendapatkan *word* ke-6 dan ke-7 dari sub-kunci pertama. Ulangi semua proses diatas untuk mendapatkan sub-kunci selanjutnya.

3.2. Proses Enkripsi

Proses enkripsi Rijndael diawali dengan proses *AddRoundKey* diikuti sembilan iterasi dengan struktur yang tersusun atas empat proses yaitu *SubBytes, ShiftRows, MixColumns, dan AddRoundKey*. Akhir proses enkripsi yaitu iterasi kesepuluh yang tersusun atas tiga proses terurut *SubBytes, ShiftRows*, dan *AddRoundKey* yang keseluruhan proses tersebut diiringi proses *key schedule* bagi setiap iterasi. Seluruh fungsi operasi (penjumlahan dan perkalian) yang tercakup dalam AES merupakan operasi-operasi yang didefinisikan dalam ruang lingkup GF(2^8) dengan polinomial *irreducible* pembangkit $f(x) = x^8 + x^4 + x^3 + x + 1$.



Gambar 3. 2 *Flow Chart* Proses Enkripsi Algoritma Rijndael

Sebelum memasuki proses enkripsi, sebagaimana pada Gambar 3.2, dilakukan partisi setiap blok *plaintext* 128 bit menjadi masing-masing delapan bit dan direpresentasikan dalam dua karakter heksadesimal pada matriks 4x4 yang disebut matriks *input*.

3.2.1. Transformasi *AddRoundKey*

Transformasi ini melakukan proses XOR antara tabel *state* pada *plaintext* dengan 128 bit kunci yang sudah dibangkitkan sebelumnya. Proses *AddRoundKey* diberikan pada Gambar 3.3:

$$\begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \oplus \begin{bmatrix} k_{0,0} & k_{0,1} & k_{0,2} & k_{0,3} \\ k_{1,0} & k_{1,1} & k_{1,2} & k_{1,3} \\ k_{2,0} & k_{2,1} & k_{2,2} & k_{2,3} \\ k_{3,0} & k_{3,1} & k_{3,2} & k_{3,3} \end{bmatrix} = \begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \end{bmatrix}$$

[Sumber: Stallings, 2005]

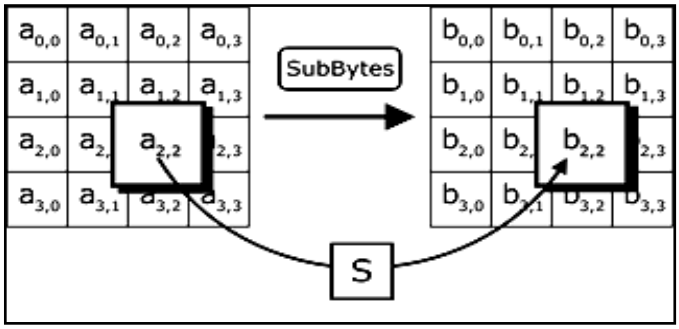
Gambar 3. 3 Proses *AddRoundKey*

Invers atau kebalikan dari transformasi *AddRoundKey* adalah proses XOR antara *state* 128 bit pada *ciphertext* dengan 128 bit *round-key* yang dibangkitkan sebelumnya, dengan menggunakan kunci tiap iterasi yang berkebalikan dari proses enkripsi. *Invers* dari transformasi ini digunakan pada proses dekripsi.

3.2.2. Transformasi *SubBytes*

Transformasi *SubBytes* memetakan setiap *array state* dengan menggunakan tabel substitusi S-Box. Sebuah tabel S-Box terdiri dari 16x16 baris dan kolom dengan masing-masing berukuran 1 *byte*. Rijndael memiliki satu buah S-Box yang akan dipakai pada setiap iterasi.

Cara substitusinya adalah sebagai berikut: Setiap *byte* pada tabel *state* $a(r, c) = xy$ dengan xy adalah digit heksadesimal dari $a(r, c)$, nilai substitusinya adalah elemen dalam S-Box yang merupakan perpotongan baris ke x dengan kolom ke y dan menghasilkan nilai substitusi baru yaitu $b(r, c)$. Ilustrasi proses *SubByte* dapat dilihat pada Gambar 3.4:



[Sumber: Stallings, 2005]

Gambar 3. 4 Proses *SubByte*

S-Box yang akan dipakai dalam algoritma Rijndael untuk proses transformasi *SubByte* tampak pada Tabel 3.2.

Tabel 3. 2 S-Box Algoritma Rijndael

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

[Sumber: Stallings, 2005]

Tujuan dari transformasi ini adalah untuk menghasilkan *confusion*, yaitu mengaburkan hubungan antara *plaintext* dan *ciphertext*.

Invers dari transformasi *SubByte* adalah substitusi yang menggunakan tabel *invers* S-Box. Tabel 3.3 menunjukkan *invers* S-Box yang dipakai.

Tabel 3. 3 *Invers* S-Box Algoritma Rijndael

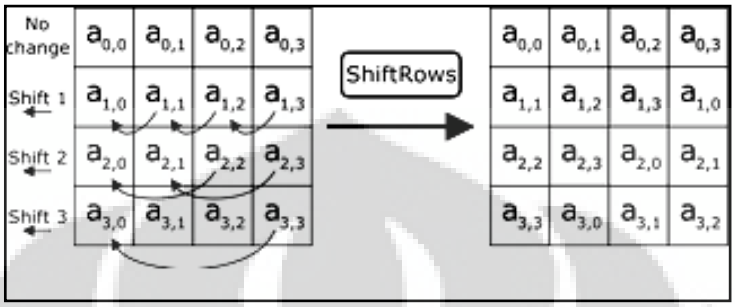
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	AC	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

[Sumber: Stallings, 2005]

3.2.3. Transformasi *ShiftRows*

Transformasi *ShiftRows* akan beroperasi pada tiap baris dalam tabel *state*. Proses ini akan bekerja dengan cara memutar elemen matriks hasil proses transformasi Sub*Byte*, pada 3 baris terakhir (baris 1, 2, dan 3) ke kiri dengan

jumlah perputaran yang berbeda-beda. Baris 1 akan diputar sebanyak 1 kali, baris 2 akan diputar sebanyak 2 kali, dan baris 3 akan diputar sebanyak 3 kali. Sedangkan baris 0 tidak akan diputar. Proses *ShiftRows* diperlihatkan pada Gambar 3.5:



[Sumber: Stallings, 2005]

Gambar 3. 5 Proses *ShiftRows*

Tujuan dari transformasi ini adalah untuk menghasilkan *diffusion*, yaitu penyebaran pengaruh setiap bit *plaintext* dan kunci terhadap *ciphertext* yang dihasilkan.

Invers dari transformasi *ShiftRows* juga memutar *byte-byte* pada 3 baris terakhir dengan jumlah putaran yang sama hanya saja dengan arah kebalikannya yaitu ke kanan.

3.2.4. Transformasi *MixColumns*

Transformasi *MixColumns* akan beroperasi pada setiap kolom pada table *state* dengan menggabungkan 4 *byte* dari setiap kolom table *state* dengan menggunakan transformasi linier. Transformasi ini mengalikan setiap kolom dari tabel *state* dengan polinom $a(x)\ mod\ (x^4 + 1)$. Setiap kolom diperlakukan sebagai polinom 4 suku pada GF (2^8). Sementara itu, polinom $a(x)$ yang ditetapkan yaitu $a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}\,x + \{02\}$ dengan tiap konstantanya merupakan bilangan heksadesimal.

Transformasi ini juga dapat dipandang sebagai perkalian matriks transformasi linier yang digambarkan pada Gambar 3.6.

$$\begin{bmatrix} b_{0,c} \\ b_{1,c} \\ b_{2,c} \\ b_{3,c} \end{bmatrix} = \begin{bmatrix} \{02\} & \{03\} & \{01\} & \{01\} \\ \{01\} & \{02\} & \{03\} & \{01\} \\ \{01\} & \{01\} & \{02\} & \{03\} \\ \{03\} & \{01\} & \{01\} & \{02\} \end{bmatrix} \begin{bmatrix} a_{0,c} \\ a_{1,c} \\ a_{2,c} \\ a_{3,c} \end{bmatrix}$$

$$b_{0,c} = (2 \bullet a_{0,c}) \oplus (3 \bullet a_{1,c}) \oplus (a_{2,c}) \oplus (a_{3,c}))$$

$$b_{1,c} = (a_{0,c}) \oplus (2 \bullet a_{1,c}) \oplus (3 \bullet a_{2,c}) \oplus (a_{3,c}))$$

$$b_{2,c} = (a_{0,c}) \oplus (a_{1,c}) \oplus (2 \bullet a_{2,c}) \oplus (3 \bullet a_{3,c}))$$

$$b_{3,c} = (3 \bullet a_{0,c}) \oplus (a_{1,c}) \oplus (a_{2,c}) \oplus (2 \bullet a_{3,c}))$$

[Sumber: Stallings, 2005]

Gambar 3. 6 Proses *MixColumns*

Operasi penjumlahan ($\oplus$) dan perkalian ($\bullet$) pada Gambar 3.6 merupakan operasi yang terdefinisikan dalam ruang lingkup GF (2^8).

Jika transformasi *ShiftRows* bertujuan menyebarkan pengaruh setiap bit plaintext dan kunci terhadap ciphertext yang dihasilkan pada arah baris, maka transformasi *MixColumns* menyebarkan pada arah kolom.

Invers dari transformasi *MixColumns* adalah mengalikan setiap kolom dari tabel *state* pada 16 *byte ciphertext* dengan polinom $b(x)\ mod\ (x^4 + 1)$. Setiap kolom diperlakukan sebagai polinom 4 suku pada GF (2^8). Sementara itu, polinom $b(x)$ yang ditetapkan yaitu b(x)= {0B} x^3 + {0D} x^2 + {09} x + {0E} dengan tiap konstantanya merupakan bilangan heksadesimal.

Invers MixColumns ini juga dapat dipandang sebagai perkalian matriks transformasi linier yang digambarkan pada Gambar 3.7.

$$\begin{bmatrix} a_{0,c} \\ a_{1,c} \\ a_{2,c} \\ a_{3,c} \end{bmatrix} = \begin{bmatrix} \{0E\} & \{0B\} & \{0D\} & \{09\} \\ \{09\} & \{0E\} & \{0B\} & \{0D\} \\ \{0D\} & \{09\} & \{0E\} & \{0B\} \\ \{0B\} & \{0D\} & \{09\} & \{0E\} \end{bmatrix} \begin{bmatrix} b_{0,c} \\ b_{1,c} \\ b_{2,c} \\ b_{3,c} \end{bmatrix}$$

$$a_{0,c} = (14 \bullet b_{0,c}) \oplus (11 \bullet b_{1,c}) \oplus (13 \bullet b_{2,c}) \oplus (9 \bullet b_{3,c}))$$

$$a_{1,c} = (9 \bullet b_{0,c}) \oplus (14 \bullet b_{1,c}) \oplus (11 \bullet b_{2,c}) \oplus (13 \bullet b_{3,c}))$$

$$a_{2,c} = (13 \bullet b_{0,c}) \oplus (9 \bullet b_{1,c}) \oplus (14 \bullet b_{2,c}) \oplus (11 \bullet b_{3,c}))$$

$$a_{3,c} = (11 \bullet b_{0,c}) \oplus (13 \bullet b_{1,c}) \oplus (9 \bullet b_{2,c}) \oplus (14 \bullet b_{3,c}))$$

[Sumber: Stallings, 2005]

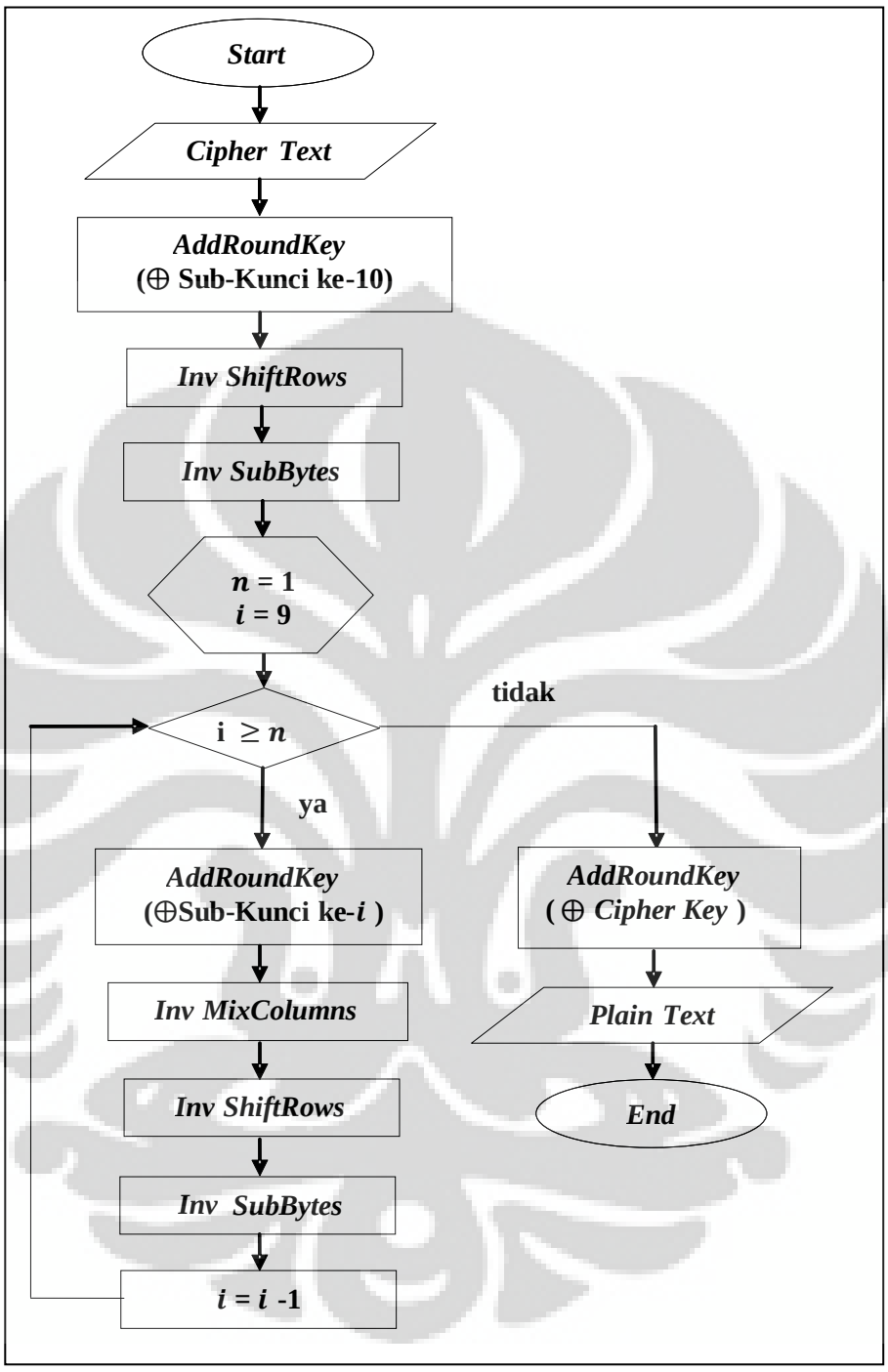
Gambar 3. 7 Proses *Invers MixColumns*

Operasi penjumlahan (⊕) dan perkalian (●) pada Gambar 3.7 merupakan operasi yang terdefinisikan dalam ruang lingkup GF (2^8).

3.3. Proses Dekripsi

Struktur proses dekripsi Rijndael secara umum sama dengan proses enkripsi, tetapi pada proses dekripsi Rijndael memiliki urutan proses transformasi penyusun tiap iterasi yang berbeda. Tidak hanya itu, transformasi yang digunakan pun merupakan transformasi kebalikan atau *invers* dari proses transformasi penyusun setiap iterasi pada proses enkripsi.

Meskipun proses pembentukan *key schedule* pada proses dekripsi dan enkripsi identik, akan tetapi proses penjadwalan penggunaan kunci pada setiap iterasi pada dekripsi berkebalikan dengan proses enkripsi. Penjadwalan kunci pada proses dekripsi pada tiap iterasi dimulai dari word ke-43 sampai word ke-0 atau dimulai dari sub-kunci ke 10 sampai *cipher key*. Gambar 3.8 menunjukkan *flow chart* dari proses dekripsi secara umum.



Gambar 3. 8 *Flow Chart* Proses Dekripsi Algoritma Rijndael

BAB 4
IMPLEMENTASI ALGORITMA RIJNDAEL

Dalam penelitian kali ini, algoritma Rijndael akan diimplementasikan menggunakan *software* Matlab untuk data berupa citra digital atau foto. *Input* awal berupa *file* citra dan kunci sepanjang 128 bit. Jika kunci yang dimasukkan tidak mencapai 128 bit, maka akan ditambahkan elemen 0 pada akhir kunci hingga mencapai 128 bit yang diperlukan. Sementara jika kunci yang dimasukkan lebih dari 128 bit, maka kunci tidak akan diterima, sehingga akan diminta *input* kunci yang baru.

4.1 *Key Schedule*

Seperti yang dijelaskan dalam sub bab 3.1, proses pembentukan kunci terdiri dari empat tahap yaitu *RotWord*, *SubWord*, proses XOR dengan nilai R-Con, dan proses XOR dengan *word* sebelumnya. Pertama-tama dimasukkan suatu *input* kunci. Proses dilakukan dalam lingkungan heksadesimal, dan beberapa proses membutuhkan konversi ke biner, yaitu pada proses XOR.

Input: "enkripsi citra"

Setiap input yang dimasukkan akan diubah menjadi bilangan ASCII. Maka, input "enkripsi citra" akan diubah menjadi:

101 110 107 114 105 112 115 105 32 99 105 116 114 97 0 0

Kemudian, 16 bit bilangan ASCII akan direpresentasikan menjadi matriks 4 x 4 dan diubah menjadi heksadesimal, sehingga akan dihasilkan *cipher key* yaitu:

Cipher Key: $\begin{bmatrix} 65 & 69 & 20 & 72 \\ 6E & 70 & 63 & 61 \\ 6B & 73 & 69 & 00 \\ 72 & 69 & 74 & 00 \end{bmatrix}$

Tahap *RotWord*

Untuk menghasilkan kolom pertama pada sub-kunci pertama (W_i), maka yang akan dioperasikan pertama adalah kolom ke empat dari *cipher key* atau W_{i-1}. Prosesnya hanya menggeser kolom secara siklis ke atas 1 kali.

Hasil Tahap *RotWord* = $\begin{bmatrix}72\\61\\00\\00\end{bmatrix} \rightarrow \begin{bmatrix}61\\00\\00\\72\end{bmatrix}$

Tahap *SubWord*

Word hasil proses *RotWord* akan disubstitusi dengan nilai dalam tabel S-Box.

Hasil Tahap *SubWord*: $\begin{bmatrix}61\\00\\00\\72\end{bmatrix} \rightarrow S-Box \rightarrow \begin{bmatrix}EF\\63\\63\\40\end{bmatrix}$

Tahap terakhir untuk mendapatkan kolom ke W_i adalah proses XOR yang dilakukan antara hasil *SubWord* dengan nilai RCon yang bersesuaian, lalu proses XOR dengan kolom W_{i-4}.

Prosesnya adalah dengan mengubah bilangan heksadesimal pada matriks sebelumnya menjadi bilangan biner, kemudian dilakukan proses XOR.

$$\begin{bmatrix}EF\\63\\63\\40\end{bmatrix} \oplus \begin{bmatrix}01\\00\\00\\00\end{bmatrix} \oplus \begin{bmatrix}65\\6E\\6B\\72\end{bmatrix} = \begin{bmatrix}8B\\0D\\08\\32\end{bmatrix} \; atau$$

$$\begin{bmatrix}11101111\\01100011\\01100011\\01000000\end{bmatrix} \oplus \begin{bmatrix}00000001\\00000000\\00000000\\00000000\end{bmatrix} \oplus \begin{bmatrix}01100101\\01101110\\01101011\\01110010\end{bmatrix} = \begin{bmatrix}10001011\\00001101\\00001000\\00110010\end{bmatrix}$$

Jadi, kolom pertama sub-kunci pertama (W_i) = $\begin{bmatrix} 8B \\ 0D \\ 08 \\ 32 \end{bmatrix}$

Untuk mendapatkan kolom kedua pada sub-kunci pertama (W_{i+1}), cukup dengan melakukan operasi XOR antara W_i dengan kolom W_{i-3}. Cara yang sama untuk kolom ketiga dan keempat.

Kolom kedua sub-kunci pertama (W_{i+1}) = $\begin{bmatrix} 8B \\ 0D \\ 08 \\ 32 \end{bmatrix} \oplus \begin{bmatrix} 69 \\ 70 \\ 73 \\ 69 \end{bmatrix} = \begin{bmatrix} E2 \\ 7D \\ 7B \\ 5B \end{bmatrix}$

Kolom ketiga sub-kunci pertama (W_{i+2}) = $\begin{bmatrix} E2 \\ 7D \\ 7B \\ 5B \end{bmatrix} \oplus \begin{bmatrix} 20 \\ 63 \\ 69 \\ 74 \end{bmatrix} = \begin{bmatrix} C2 \\ 1E \\ 12 \\ 2F \end{bmatrix}$

Kolom empat sub-kunci pertama (W_{i+3}) = $\begin{bmatrix} C2 \\ 1E \\ 12 \\ 2F \end{bmatrix} \oplus \begin{bmatrix} 72 \\ 61 \\ 00 \\ 00 \end{bmatrix} = \begin{bmatrix} B0 \\ 7F \\ 12 \\ 2F \end{bmatrix}$

Sehingga akan didapatkan sub-kunci pertama yaitu: $\begin{bmatrix} 8B & E2 & C2 & B0 \\ 0D & 7D & 1E & 7F \\ 08 & 7B & 12 & 12 \\ 32 & 5B & 2F & 2F \end{bmatrix}$

Proses-proses diatas akan diulangi sampai 10 iterasi hingga menghasilkan 10 sub-kunci yang akan dipakai dalam proses enkripsi, yaitu:

Sub-kunci pertama: $\begin{bmatrix} 8B & E2 & C2 & B0 \\ 0D & 7D & 1E & 7F \\ 08 & 7B & 12 & 12 \\ 32 & 5B & 2F & 2F \end{bmatrix}$

Sub-kunci kedua: $\begin{bmatrix} 5B & B9 & 7B & CB \\ C4 & B9 & A7 & D8 \\ 1D & 66 & 74 & 66 \\ D5 & 8E & A1 & 8E \end{bmatrix}$

Sub-kunci ketiga: $\begin{bmatrix} 3E & 87 & FC & 37 \\ F7 & 4E & E9 & 31 \\ 04 & 62 & 16 & 70 \\ CA & 44 & E5 & 6B \end{bmatrix}$

Sub-kunci keempat: $\begin{bmatrix} F1 & 76 & 8A & BD \\ A6 & E8 & 01 & 30 \\ 7B & 19 & 0F & 7F \\ 50 & 14 & F1 & 9A \end{bmatrix}$

Sub-kunci kelima: $\begin{bmatrix} E5 & 93 & 19 & A4 \\ 74 & 9C & 9D & AD \\ C3 & DA & D5 & AA \\ 2A & 3E & CF & 55 \end{bmatrix}$

Sub-kunci keenam: $\begin{bmatrix} 50 & C3 & DA & 7E \\ D8 & 44 & D9 & 74 \\ 3F & E5 & 30 & 9A \\ 63 & 5D & 92 & C7 \end{bmatrix}$

Sub-kunci ketujuh: $\begin{bmatrix} 82 & 41 & 9B & E5 \\ 60 & 24 & FD & 89 \\ F9 & 1C & 2C & B6 \\ 90 & CD & 5F & 98 \end{bmatrix}$

Sub-kunci kedelapan: $\begin{bmatrix} A5 & E4 & 7F & 9A \\ 2E & 0A & F7 & 7E \\ BF & A3 & 8F & 39 \\ 49 & 84 & DB & 43 \end{bmatrix}$

Sub-kunci kesembilan: $\begin{bmatrix} 4D & A9 & D6 & 4C \\ 3C & 36 & C1 & BF \\ A5 & 06 & 89 & B0 \\ F1 & 75 & AE & ED \end{bmatrix}$

Sub-kunci kesepuluh: $\begin{bmatrix} 73 & DA & 0C & 40 \\ DB & ED & 2C & 93 \\ F0 & F6 & 7F & CF \\ D8 & AD & 03 & EE \end{bmatrix}$

4.2 Proses Enkripsi

Seperti yang sudah dijelaskan dalam sub bab 3.2, proses enkripsi Rijndael terdiri dari 4 transformasi yaitu: *AddRoundKey, SubBytes, ShiftRows, dan MixColumns*. *Input* berupa citra digital atau citra *grayscale* dan direpresentasikan dalam bentuk matriks sesuai dengan ukuran piksel citra tersebut. Algoritma Rijndael beroperasi pada mode blok *cipher*, sehingga matriks citra akan dipartisi menjadi beberapa blok matriks *input* sebesar 128 bit dan elemennya

direpresentasikan dalam heksadesimal. Cara pengambilan bloknya adalah dengan menyusun seluruh elemen matriks *plaintext* menjadi *array* dan membaginya menjadi beberapa partisi 128 bit. Jika pada akhir partisi, elemennya tidak mencapai 128 bit, maka akan ditambahkan dummy berupa elemen 0 sampai berjumlah 128 bit. Berikut akan dijelaskan implementasi algoritma Rijndael pada citra *grayscale* dengan ukuran piksel 15 x 15.

Input:

[sumber: digital.cs.usu.edu]

Gambar 4. 1 Citra Awal Ukuran 15 x 15 yang Diperbesar

Plaintext:

$$\begin{bmatrix}
173 & 99 & 110 & 120 & 129 & 136 & 126 & 130 & 131 & 106 & 165 & 152 & 205 & 123 & 93 \\
161 & 159 & 102 & 119 & 129 & 127 & 177 & 176 & 126 & 106 & 150 & 147 & 145 & 45 & 50 \\
85 & 155 & 103 & 116 & 110 & 118 & 153 & 170 & 211 & 98 & 138 & 145 & 151 & 48 & 76 \\
54 & 159 & 96 & 111 & 109 & 133 & 175 & 195 & 201 & 223 & 147 & 205 & 49 & 54 & 163 \\
174 & 166 & 104 & 217 & 109 & 118 & 145 & 163 & 188 & 156 & 198 & 167 & 56 & 156 & 161 \\
146 & 169 & 98 & 184 & 92 & 136 & 120 & 91 & 115 & 195 & 93 & 204 & 57 & 162 & 155 \\
76 & 169 & 97 & 113 & 116 & 96 & 55 & 164 & 180 & 206 & 155 & 154 & 127 & 159 & 155 \\
63 & 172 & 99 & 149 & 136 & 38 & 141 & 59 & 128 & 138 & 45 & 49 & 162 & 155 & 147 \\
173 & 181 & 109 & 118 & 80 & 45 & 81 & 177 & 142 & 155 & 82 & 46 & 153 & 141 & 193 \\
122 & 175 & 111 & 66 & 36 & 175 & 102 & 150 & 126 & 147 & 62 & 149 & 147 & 196 & 210 \\
64 & 182 & 99 & 139 & 205 & 46 & 67 & 136 & 125 & 48 & 69 & 154 & 146 & 214 & 210 \\
80 & 176 & 43 & 152 & 47 & 41 & 50 & 131 & 157 & 52 & 76 & 114 & 141 & 209 & 66 \\
171 & 181 & 48 & 53 & 146 & 35 & 84 & 135 & 154 & 190 & 42 & 150 & 127 & 180 & 90 \\
96 & 178 & 54 & 55 & 45 & 47 & 125 & 144 & 138 & 169 & 215 & 96 & 133 & 88 & 86 \\
54 & 187 & 47 & 105 & 122 & 40 & 131 & 139 & 144 & 160 & 208 & 100 & 113 & 91 & 96
\end{bmatrix}$$

Plaintext yang semula berupa matriks dengan ukuran 15 x 15, di bentuk menjadi *array* ukuran 1 x 225 dengan diurutkan berdasarkan kolom, sehingga akan menjadi:

173 161 85 54 174 146 76 63 173 122 64 80 171 96 54 99 159 155
159 166 169 169 172 181 175 182 176 181 178 187 110 102 103 96
104 98 97 99 109 111 99 43 48 54 47 120 119 116 111 217 184
113 149 118 66 139 152 53 55 105 129 129 110 109 109 92 116 136
80 36 205 47 146 45 122 136 127 118 133 118 136 96 38 45 175
46 41 35 47 40 126 177 153 175 145 120 55 141 81 102 67 50

84 125 131 130 176 170 195 163 91 164 59 177 150 136 131 135 144
139 131 126 211 201 188 115 180 128 142 126 125 157 154 138 144
106 106 98 223 156 196 206 138 155 147 48 52 190 169 160 165 150
138 147 198 93 155 45 82 62 69 76 42 215 208 152 147 145 205
167 204 54 49 46 149 154 114 150 96 100 205 145 151 49 56 57
127 162 153 147 146 141 127 133 113 123 45 48 54 156 162 159 155
141 196 214 209 180 88 91 93 50 76 163 161 155 155 147 193 210
210 66 90 86 96

Setelah diurutkan menjadi *array*, akan dipartisi menjadi beberapa blok dengan panjang 128 bit atau 16 *byte* dengan cara pengambilan adalah 4 *byte* pertama menjadi kolom pertama, 4 *byte* kedua menjadi kolom kedua dan seterusnya. Matriks input pertama merupakan 4 kolom pertama yang direpresentasikan dalam heksadesimal.

Blok 16 *byte* pertama: 173 161 85 54 174 146 76 63 173 122 64 80 171 96 54 99

Matriks *input plaintext* 16 *byte* pertama: $\begin{bmatrix} 173 & 174 & 173 & 171 \\ 161 & 146 & 122 & 96 \\ 85 & 76 & 64 & 54 \\ 54 & 63 & 80 & 99 \end{bmatrix}$

Dalam heksadesimal: $\begin{bmatrix} AD & AE & AD & AB \\ A1 & 92 & 7A & 60 \\ 55 & 4C & 40 & 36 \\ 36 & 3F & 50 & 63 \end{bmatrix}$

Cipher key: $\begin{bmatrix} 65 & 69 & 20 & 72 \\ 6E & 70 & 63 & 61 \\ 6B & 73 & 69 & 00 \\ 72 & 69 & 74 & 00 \end{bmatrix}$

Transformasi *AddRoundKey*

Dilakukan proses XOR antara matriks *input* dan *cipher key*. Pertama dengan mengubah bilangan heksadesimal menjadi bilangan biner, lalu dilakukan proses XOR.

$$\begin{bmatrix} AD \\ A1 \\ 55 \\ 36 \end{bmatrix} \oplus \begin{bmatrix} 65 \\ 6E \\ 6B \\ 72 \end{bmatrix} = \begin{bmatrix} C8 \\ CF \\ 3E \\ 44 \end{bmatrix} \text{ atau}$$

$$\begin{bmatrix} 10101101 \\ 10100001 \\ 01010101 \\ 00110110 \end{bmatrix} \oplus \begin{bmatrix} 01100101 \\ 01101110 \\ 01101011 \\ 01110010 \end{bmatrix} = \begin{bmatrix} 11001000 \\ 11001111 \\ 00111110 \\ 01000100 \end{bmatrix}$$

$$\begin{bmatrix} AE \\ 92 \\ 4C \\ 3E \end{bmatrix} \oplus \begin{bmatrix} 69 \\ 70 \\ 73 \\ 69 \end{bmatrix} = \begin{bmatrix} C7 \\ E2 \\ 3F \\ 56 \end{bmatrix} \text{ atau}$$

$$\begin{bmatrix} 10101110 \\ 10010010 \\ 01001100 \\ 00111110 \end{bmatrix} \oplus \begin{bmatrix} 01101001 \\ 01110000 \\ 01110011 \\ 01101001 \end{bmatrix} = \begin{bmatrix} 11000111 \\ 11100010 \\ 00111111 \\ 01010110 \end{bmatrix}$$

$$\begin{bmatrix} AD \\ 7A \\ 40 \\ 50 \end{bmatrix} \oplus \begin{bmatrix} 20 \\ 63 \\ 69 \\ 74 \end{bmatrix} = \begin{bmatrix} 8D \\ 19 \\ 29 \\ 24 \end{bmatrix} \text{ atau}$$

$$\begin{bmatrix} 10101101 \\ 01111010 \\ 01000000 \\ 01010000 \end{bmatrix} \oplus \begin{bmatrix} 00100000 \\ 01100011 \\ 01101001 \\ 01110100 \end{bmatrix} = \begin{bmatrix} 10001101 \\ 00011001 \\ 00101001 \\ 00100100 \end{bmatrix}$$

$$\begin{bmatrix} AB \\ 60 \\ 36 \\ 63 \end{bmatrix} \oplus \begin{bmatrix} 72 \\ 61 \\ 00 \\ 00 \end{bmatrix} = \begin{bmatrix} D9 \\ 01 \\ 36 \\ 63 \end{bmatrix} \text{ atau}$$

$$\begin{bmatrix} 10101011 \\ 01100000 \\ 00110110 \\ 01100011 \end{bmatrix} \oplus \begin{bmatrix} 01110010 \\ 01100001 \\ 00000000 \\ 00000000 \end{bmatrix} = \begin{bmatrix} 11011001 \\ 00000001 \\ 00110110 \\ 01100011 \end{bmatrix}$$

Hasil transformasi *AddRoundKey* = $\begin{bmatrix} C8 & C7 & 8D & D9 \\ CF & E2 & 19 & 01 \\ 3E & 3F & 29 & 36 \\ 44 & 56 & 24 & 63 \end{bmatrix}$

Setelah melakukan transformasi *AddRoundKey* yang pertama, proses enkripsi memasuki iterasi pertama dari 10 kali iterasi dengan 9 iterasi pertama

terdiri dari transformasi *SubBytes, ShiftRows, MixColumns, dan AddRoundKey*. Pada iterasi terakhir, dilakukan semua transformasi kecuali *MixColumns*.

Transformasi *SubBytes*

$$\begin{bmatrix} C8 & C7 & 8D & D9 \\ CF & E2 & 19 & 01 \\ 3E & 3F & 29 & 36 \\ 44 & 56 & 24 & 63 \end{bmatrix} \rightarrow S-Box \rightarrow \begin{bmatrix} E8 & C6 & 5D & 35 \\ 8A & 98 & D4 & 7C \\ B2 & 75 & A5 & 05 \\ 1B & B1 & 36 & FB \end{bmatrix}$$

Transformasi *ShiftRows*

$$\begin{bmatrix} E8 & C6 & 5D & 35 \\ 8A & 98 & D4 & 7C \\ B2 & 75 & A5 & 05 \\ 1B & B1 & 36 & FB \end{bmatrix} \rightarrow \begin{bmatrix} E8 & C6 & 5D & 35 \\ 98 & D4 & 7C & 8A \\ A5 & 05 & B2 & 75 \\ FB & 1B & B1 & 36 \end{bmatrix}$$

Transformasi *MixColumns*

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} E8 \\ 98 \\ A5 \\ FB \end{bmatrix} = \begin{bmatrix} 26 \\ CC \\ 37 \\ F3 \end{bmatrix}$$

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} C6 \\ D4 \\ 05 \\ 1B \end{bmatrix} = \begin{bmatrix} EE \\ 61 \\ 35 \\ B6 \end{bmatrix}$$

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} 5D \\ 7C \\ B2 \\ B1 \end{bmatrix} = \begin{bmatrix} 3D \\ D9 \\ 96 \\ 50 \end{bmatrix}$$

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} 35 \\ 8A \\ 75 \\ 36 \end{bmatrix} = \begin{bmatrix} AC \\ 93 \\ 0F \\ CC \end{bmatrix}$$

Hasil Transformasi MixColumns = $\begin{bmatrix} 26 & EE & 3D & AC \\ CC & 61 & D9 & 93 \\ 37 & 35 & 96 & 0F \\ F3 & B6 & 50 & CC \end{bmatrix}$

Berikut adalah ilustrasi perhitungan proses *MixColumns* dimana semua proses penjumlahan dan perkalian tercakup dalam ruang lingkup *Galois Field* 2^8:

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} E8 \\ 98 \\ A5 \\ FB \end{bmatrix} = \begin{bmatrix} b_{1,1} \\ b_{2,1} \\ b_{3,1} \\ b_{4,1} \end{bmatrix}$$

$b_{1,1} = 02 \bullet E8 \oplus 03 \bullet 98 \oplus 01 \bullet A5 \oplus 01 \bullet FB$

$b_{1,1} = (00000010) \bullet (11101000) \oplus (00000011) \bullet (10011000) \oplus (10100101)$

$\oplus (11111011)$

$b_{1,1} = [(x)(x^7 + x^6 + x^5 + x^3)] \; mod \; (x^8 + x^4 + x^3 + x + 1)$

$+[(x+1)(x^7 + x^4 + x^3)] \; mod \; (x^8 + x^4 + x^3 + x + 1)$

$+(x^7 + x^5 + x^2 + 1) \; mod \; (x^8 + x^4 + x^3 + x + 1)$

$+(x^7 + x^6 + x^5 + x^4 + x^3 + x + 1) \; mod \; (x^8 + x^4 + x^3 + x + 1)$

$b_{1,1} = (x^8 + x^7 + x^6 + x^4) mod \; (x^8 + x^4 + x^3 + x + 1)$

$+(x^8 + x^7 + x^5 + x^3) \; mod \; (x^8 + x^4 + x^3 + x + 1)$

$+(x^7 + x^5 + x^2 + 1) \; mod \; (x^8 + x^4 + x^3 + x + 1)$

$+(x^7 + x^6 + x^5 + x^4 + x^3 + x + 1) \; mod \; (x^8 + x^4 + x^3 + x + 1)$

$b_{1,1} = (x^7 + x^6 + x^3 + x + 1) + (x^7 + x^5 + x^4 + x + 1)$

$+ (x^7 + x^5 + x^2 + 1) + (x^7 + x^6 + x^5 + x^4 + x^3 + x + 1)$

$b_{1,1} = x^5 + x^2 + x$

$b_{1,1} = (00100110)$

$b_{1,1} = 26$

Pada transformasi *MixColumns*, terjadi sedikit perubahan pada *tools* "gf" di Matlab. Algoritma Rijndael menggunakan polinomial $x^8+x^4+x^3+x+1$, sedangkan Matlab menggunakan polinomial $x^8+x^4+x^3+x^2+1$, sehingga Matlab mengeluarkan *warning* ketika program dijalankan. Agar Matlab tidak mengeluarkan *warning*, maka dilakukan sedikit perubahan dengan cara menghapus baris ke 95 pada *tools* "gf" :

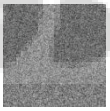
```
if(~isprimitive(double(prim_poly)))

        warning('comm:gf:primPolyPrim','PRIM_POLY must be a
        primitive polynomial.');

end
```

Setelah transformasi *MixColumns*, terdapat transformasi *AddRoundKey* dengan jalan yang sama seperti yang dijelaskan sebelumnya namun proses XOR nya dengan sub-kunci yang bersesuaian tiap iterasi. Setelah melewati 10 iterasi, akan didapatkan matriks *output ciphertext* 16 *byte* pertama yaitu:

$$\begin{bmatrix} 2F & 00 & 68 & F1 \\ 3D & 18 & 1F & 2D \\ 37 & 11 & 87 & DA \\ E6 & D9 & 3B & 25 \end{bmatrix} atau \begin{bmatrix} 47 & 0 & 104 & 241 \\ 61 & 24 & 31 & 45 \\ 55 & 17 & 135 & 218 \\ 230 & 217 & 59 & 37 \end{bmatrix}$$

Setelah dilakukan seluruh proses enkripsi pada seluruh blok *plaintext*, maka akan didapatkan *ciphertext* yang selanjutnya akan dihasilkan citra yang sudah dirahasiakan.

Output:



Gambar 4. 2 Citra Hasil Enkripsi Ukuran 15 x 15 yang Diperbesar

4.3 Proses Dekripsi

Seperti yang diterangkan dalam sub bab 3.3 , urutan proses dekripsi Rijndael merupakan kebalikan dari proses enkripsinya dan menggunakan *invers* dari tiap transformasi enkripsi. Iterasi pertama dilakukan semua transformasi kecuali *invers MixColumns.*

Matriks *input ciphertext* 16 *byte* pertama: $\begin{bmatrix} 2F & 00 & 68 & F1 \\ 3D & 18 & 1F & 2D \\ 37 & 11 & 87 & DA \\ E6 & D9 & 3B & 25 \end{bmatrix}$

Transformasi *AddRoundKey*

Proses XOR antara matriks *input* dengan sub-kunci ke-10

$$\begin{bmatrix} 2F \\ 3D \\ 37 \\ E6 \end{bmatrix} \oplus \begin{bmatrix} 73 \\ DB \\ F0 \\ D8 \end{bmatrix} = \begin{bmatrix} 5C \\ E6 \\ C7 \\ 3E \end{bmatrix}$$

$$\begin{bmatrix} 00 \\ 18 \\ 11 \\ D9 \end{bmatrix} \oplus \begin{bmatrix} DA \\ ED \\ F6 \\ AD \end{bmatrix} = \begin{bmatrix} DA \\ F5 \\ E7 \\ 74 \end{bmatrix}$$

$$\begin{bmatrix} 68 \\ 1F \\ 87 \\ 3B \end{bmatrix} \oplus \begin{bmatrix} 0C \\ 2C \\ 7F \\ 03 \end{bmatrix} = \begin{bmatrix} 64 \\ 33 \\ F8 \\ 38 \end{bmatrix}$$

$$\begin{bmatrix} F1 \\ 2D \\ DA \\ 25 \end{bmatrix} \oplus \begin{bmatrix} 40 \\ 93 \\ CF \\ EE \end{bmatrix} = \begin{bmatrix} B1 \\ BE \\ 15 \\ CB \end{bmatrix}$$

Hasil transformasi *AddRoundKey* = $\begin{bmatrix} 5C & DA & 64 & B1 \\ E6 & F5 & 33 & BE \\ C7 & E7 & F8 & 15 \\ 3E & 74 & 38 & CB \end{bmatrix}$

Transformasi *Invers ShiftRows*

$$\begin{bmatrix} 5C & DA & 64 & B1 \\ E6 & F5 & 33 & BE \\ C7 & E7 & F8 & 15 \\ 3E & 74 & 38 & CB \end{bmatrix} \rightarrow \begin{bmatrix} 5C & DA & 64 & B1 \\ BE & E6 & F5 & 33 \\ F8 & 15 & C7 & E7 \\ 74 & 38 & CB & 3E \end{bmatrix}$$

Transformasi *Invers SubBytes*

$$\begin{bmatrix} 5C & DA & 64 & B1 \\ BE & E6 & F5 & 33 \\ F8 & 15 & C7 & E7 \\ 74 & 38 & CB & 3E \end{bmatrix} \rightarrow Invers\ S-Box \rightarrow \begin{bmatrix} A7 & 7A & 8C & 56 \\ 5A & F5 & 77 & 66 \\ E1 & 2F & 31 & B0 \\ CA & 76 & 59 & D1 \end{bmatrix}$$

Setelah itu masuk ke iterasi ke-2 sampai ke-10 yang terdiri dari semua transformasi.

Transformasi *Invers MixColumns*

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} 26 \\ CC \\ 37 \\ F3 \end{bmatrix} = \begin{bmatrix} E8 \\ 98 \\ A5 \\ FB \end{bmatrix}$$

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} EE \\ 61 \\ 35 \\ B6 \end{bmatrix} = \begin{bmatrix} C6 \\ D4 \\ 05 \\ 1B \end{bmatrix}$$

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} 3D \\ D9 \\ 96 \\ 50 \end{bmatrix} = \begin{bmatrix} 5D \\ 7C \\ B2 \\ 1B \end{bmatrix}$$

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} AC \\ 93 \\ 0F \\ CC \end{bmatrix} = \begin{bmatrix} 35 \\ 8A \\ 75 \\ 36 \end{bmatrix}$$

Hasil Transformasi *Invers MixColumns* = $\begin{bmatrix} E8 & C6 & 5D & 35 \\ 98 & D4 & 7C & 8A \\ A5 & 05 & B2 & 75 \\ FB & 1B & B1 & 36 \end{bmatrix}$

Setelah dilakukan semua iterasi, maka akan dihasilkan matriks *output* *plaintext* 16 *byte* pertama yaitu:

$$\begin{bmatrix} AD & AE & AD & AB \\ A1 & 92 & 7A & 60 \\ 55 & 4C & 40 & 36 \\ 36 & 3F & 50 & 63 \end{bmatrix}$$

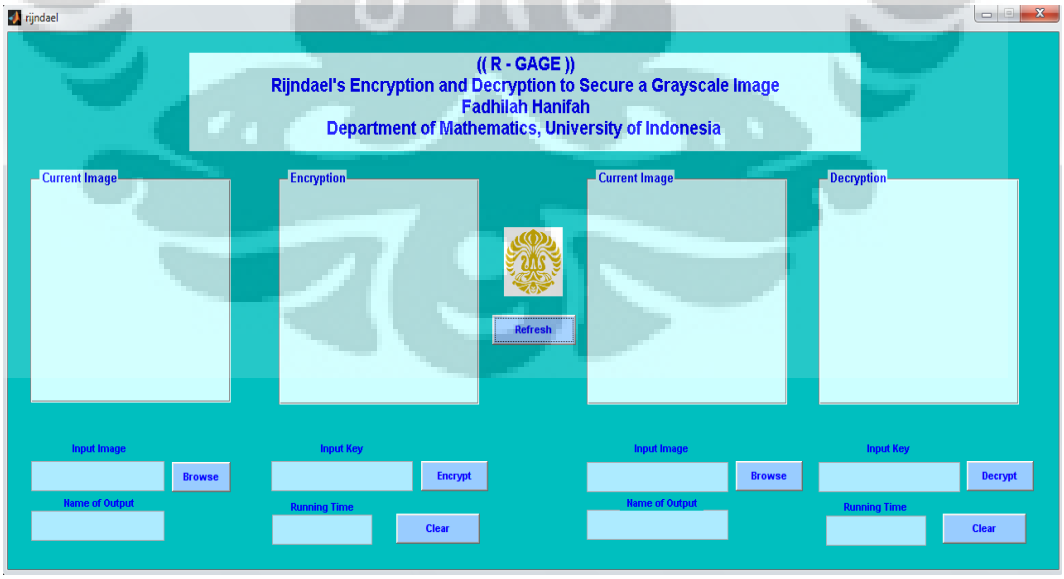
Setelah dilakukan proses dekripsi pada semua blok partisi, maka dihasilkan suatu *plaintext* yang akan menghasilkan citra awal kembali.

Output:

Gambar 4. 3 Citra Hasil Dekripsi Ukuran 15 x 15 yang Diperbesar

4.4 Pembangunan Program Aplikasi Algoritma Rijndael

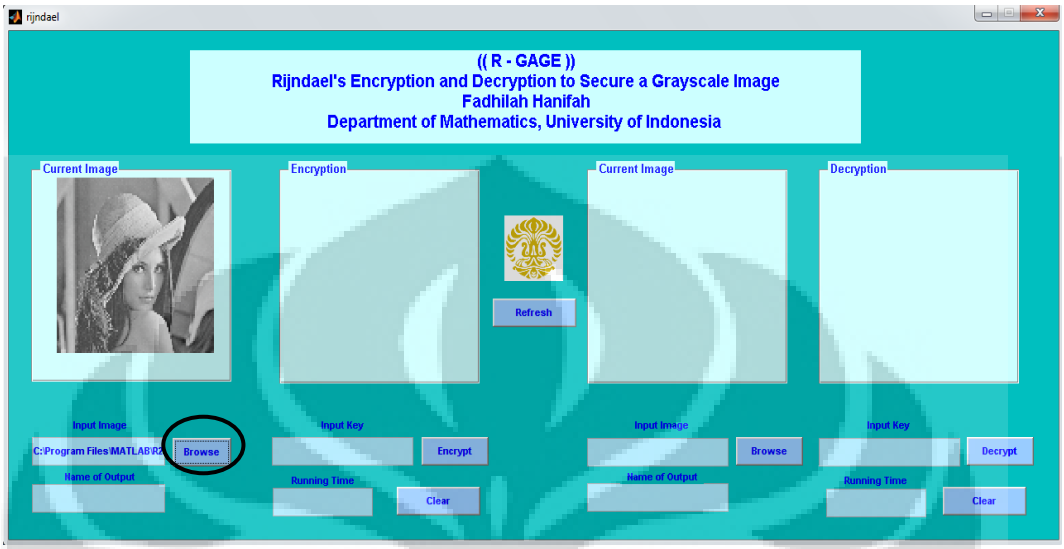
Proses implementasi algoritma Rijndael dilakukan dengan menggunakan *software* Matlab. Gambar 4.4 berupa tampilan utama program algoritma Rijndael yang terdiri dari dua proses utama yaitu enkripsi dan dekripsi.



Gambar 4. 4 Tampilan Program Utama

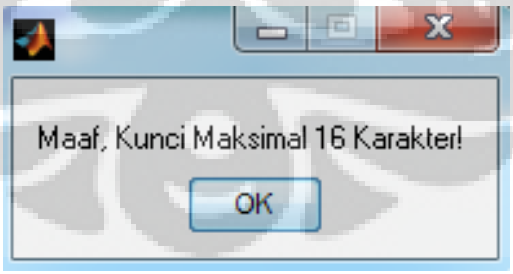
Untuk proses enkripsi, pertama-tama dimasukkan suatu citra *grayscale*, dengan memilih tombol "Browse", yang kemudian citra tersebut akan ditampilkan dalam kotak "Current Image"

.



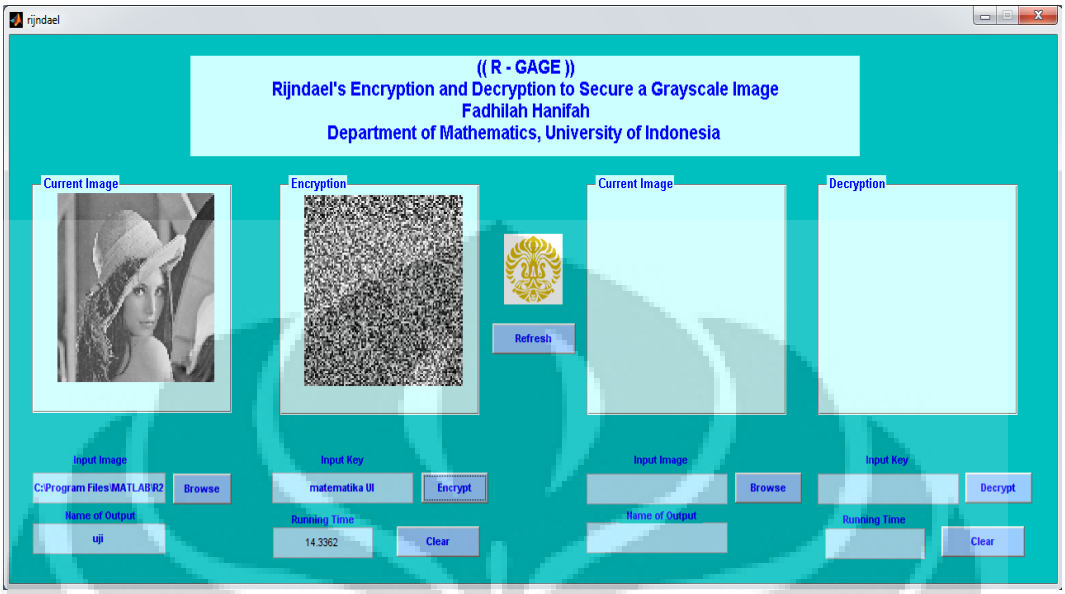
Gambar 4. 5 Tampilan Setelah *Input* Citra

Setelah itu, dimasukkan nama *output* yang diinginkan, lalu dimasukkan kunci yang diinginkan dengan panjang kunci maksimal adalah 16 karakter atau 128 bit. Jika kunci yang dimasukkan lebih dari 16 karakter, maka akan keluar *message box*.



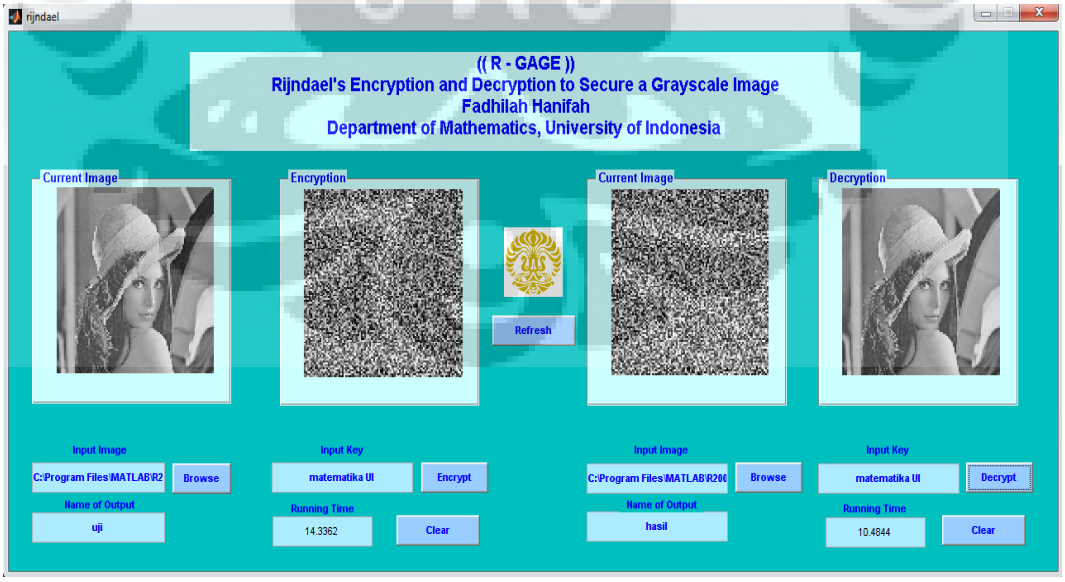
Gambar 4. 6 *Message Box*

Untuk proses enkripsi cukup pilih tombol “Encrypt”. Hasil proses enkripsi akan ditampilkan dalam kotak “Encryption”.



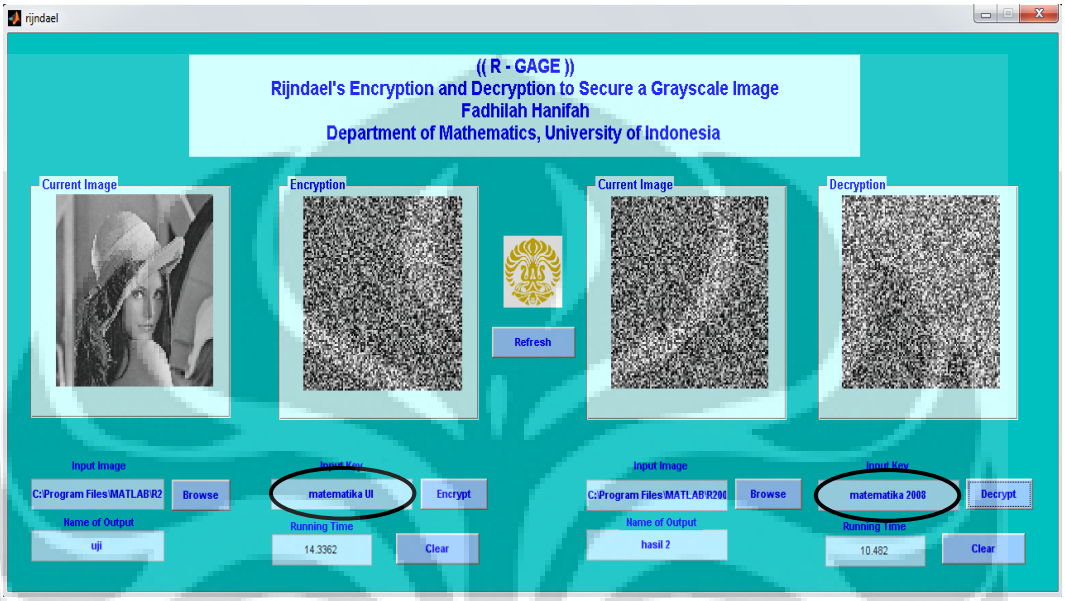
Gambar 4. 7 Hasil Proses Enkripsi

Untuk proses dekripsi dimasukkan *input* citra yang sebelumnya sudah dirahasiakan, lalu pilih tombol “Decrypt” dan hasil dekripsi akan ditampilkan di kotak “Decryption”. Setiap dilakukan proses enkripsi dan dekripsi maka akan ditampilkan *running time* setiap proses.



Gambar 4. 8 Hasil Proses Enkripsi dan Proses Dekripsi

Gambar 4.8 menunjukkan hasil dekripsi sama dengan citra awal. Kunci yang dimasukkan untuk enkripsi harus sama dengan kunci untuk dekripsi karena algoritma Rijndael merupakan algoritma yang simetris. Jika kunci yang dimasukkan berbeda maka proses dekripsi akan menghasilkan citra yang berbeda dengan citra awal. Hal ini akan ditunjukkan oleh Gambar 4.9.



Gambar 4. 9 Hasil Dekripsi dengan Kunci yang Berbeda

4.5 Uji Coba dan Analisis Hasil

Dalam implementasi algoritma Rijndael dalam mengamankan data citra digital, dilakukan beberapa uji coba untuk beberapa variasi panjang kunci input, jenis kunci input dan variasi ukuran piksel dari citra digital yang diamankan untuk mendapatkan analisis implementasi terhadap *running time* yang dihasilkan.

Adapun spesifikasi dari komputer yang digunakan selama pengerjaan tugas akhir adalah:

1. Prosesor: Intel(R) Pentium(R), P6200 @ 2,13 GHz (2 CPUs), ~2,1 GHz
2. Memori RAM 1024 MB
3. BIOS: InsydeH2O Version 29CN33WW (V2.10)

4. Sistem Operasi: Windows 7 Proffesional 32-bit (6.1, Build 7600)

Variasi Kunci *Input*: 1. Kunci sepanjang 128 bit:

a. Berupa angka saja: 1234567890123456 (K1)

b. Berupa huruf saja: enkripsidekripsi (K2)

c. Berupa simbol saja: (,/;'[\=?#$%*&^} (K3)

d. Berupa kombinasi a,b,dan c:

kunci12345+|&$!? (K4)

2. Kunci yang kurang dari 128 bit:

a. Berupa angka saja:123456 (KK1)

b. Berupa huruf saja: enkrip (KK2)

c. Berupa simbol saja: @#$%)> (KK3)

d. Berupa kombinasi a,b,dan c: ku12&~ (KK4)

Tabel 4.1 dan Tabel 4.2 menunjukkan tabel perbandingan *running time* proses enkripsi dari beberapa uji coba variasi ukuran piksel citra dan kunci input dalam satuan detik.

Tabel 4.3 dan Tabel 4.4 menunjukkan tabel perbandingan *running time* proses dekripsi dari beberapa uji coba variasi ukuran piksel citra dan kunci input dalam satuan detik.

Tabel 4. 1 Perbandingan *Running Time* Proses Enkripsi Variasi Ukuran Piksel Citra dan Variasi Kunci *Input* dengan Panjang 128 Bit (dalam detik)

Ukuran Piksel Citra	Jenis Kunci dengan panjang 128 bit			
	K1	K2	K3	K4
25x25	0.439	0.459	0.448	0.435
50x50	1.519	1.523	1.542	1.565
75x75	3.329	3.492	3.319	3.338
100x100	5.885	5.798	5.827	5.816
125x125	9.393	9.434	9.392	9.438
150x150	13.782	13.774	13.731	12.835
175x175	18.803	19.268	19.333	18,961
200x200	25.094	25.149	25.408	24.823
225x225	32.507	32.730	31.892	32.294
250x250	41.355	41.362	40.897	41.458
275x275	55.465	54.156	54.242	53.448
300x300	68.657	69.235	68.541	69.682
325x325	86.293	86.076	85.592	84.923
350x350	106.839	107.392	106.938	107.760
375x375	132.790	133.206	131.951	133.129
400x400	163.257	164.431	164.117	164.302
425x425	201.101	201.084	202.345	201.421
450x450	242.569	243.005	242.743	241.992
475x475	293.459	295.603	292.741	295.034
500x500	356.819	353.492	355.024	352.743
525x525	428.679	429.034	430.387	429.562
550x550	501.022	499.354	500.134	496.573
575x575	598.290	595.183	596.932	595.831
600x600	683.920	683.451	680.923	682.502
625x625	789.229	793.032	785.924	789.026
650x650	933.619	934.932	936.032	934.573
675x675	1140.932	1146.291	1145.378	1146.395
700x700	1462.934	1467.735	1465.892	1462.991
725x725	1658.933	1666.589	1649.823	1656.936
750x750	1883.923	1884.562	1885.672	1883.121

Tabel 4. 2 Perbandingan *Running Time* Proses Enkripsi Variasi Ukuran Piksel Citra dan Variasi Kunci *Input* dengan Panjang Kurang Dari 128 Bit (dalam detik)

Ukuran Piksel Citra	Jenis Kunci dengan panjang kurang dari 128 bit			
	KK1	KK2	KK3	KK4
25x25	0.498	0.432	0.398	0.442
50x50	1.558	1.593	1.533	1.642
75x75	3.393	3.291	3.331	3.341
100x100	5.865	5.823	5.829	5.833
125x125	9.397	9.429	9.327	9.402
150x150	13.612	13.651	12.985	13.865
175x175	19.284	19.323	18.799	18.879
200x200	24.933	25.255	24.738	24.971
225x225	31.926	32.349	31.845	32.211
250x250	41.162	40.899	41.218	41.122
275x275	54.237	52.982	55.342	55.327
300x300	68.535	68.678	67.988	69.734
325x325	86.457	84.934	85.423	86.347
350x350	108.212	107.569	108.032	107.271
375x375	134.032	133.542	133.532	133.693
400x400	162.623	162.618	163.659	164.055
425x425	202.032	201.431	202.094	202.077
450x450	241.848	242.427	242.531	241.832
475x475	292.470	293.492	295.085	292.451
500x500	354.721	356.862	355.318	355.032
525x525	431.283	428.902	430.226	430.654
550x550	501.684	499.932	501.248	500.030
575x575	596.921	595.893	600.021	596.764
600x600	682.034	681.367	684.023	682.349
625x625	795.625	799.019	793.902	802.343
650x650	933.872	935.414	933.981	934.459
675x675	1144.762	1142.364	1144.578	1146.832
700x700	1462.228	1463.894	1469.021	1464.762
725x725	1665.212	1660.319	1657.384	1660.022
750x750	1889.023	1884.793	1881.932	1883.782

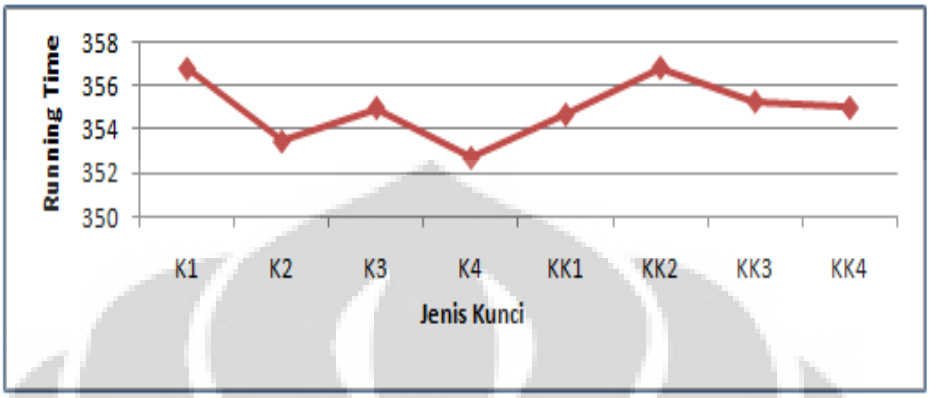
Tabel 4. 3 Perbandingan *Running Time* Proses Dekripsi Variasi Ukuran Piksel Citra dan Variasi Kunci *Input* dengan Panjang 128 Bit (dalam detik)

Ukuran Piksel Citra	Jenis Kunci dengan panjang 128 bit			
	K1	K2	K3	K4
25x25	0.365	0.376	0.354	0.328
50x50	1.197	1.154	1.153	1.171
75x75	2.679	2.702	2.638	2.686
100x100	4.6941	4.629	4.626	4.623
125x125	7.493	7.587	7.442	7.429
150x150	10.841	10.834	11.194	11.535
175x175	15.099	16.620	15.991	15.805
200x200	20.428	21,022	21.127	20.331
225x225	26.265	26.402	26.823	25.924
250x250	32.823	33.442	33.832	33.982
275x275	44.029	44.271	43.557	43.992
300x300	57.294	57.214	58.102	58.333
325x325	75.942	76.324	77.453	75.984
350x350	94.200	93.471	94.455	92.909
375x375	116.162	116.143	115.342	116,229
400x400	145.214	144.982	144.857	145.385
425x425	178.007	177.336	177.392	178.007
450x450	220.125	218.743	218.932	219.523
475x475	266.983	267.771	267.384	264.673
500x500	322.609	325.893	321.932	323.456
525x525	387.413	385.945	387.023	386.832
550x550	464.274	464.502	463.945	462.282
575x575	564.590	563.094	564.530	565,552
600x600	640.927	642.378	640.892	641.242
625x625	749.770	748.992	751.034	748.921
650x650	870.229	869.994	871.238	870.387
675x675	1012.395	1010.792	1012.347	1011.229
700x700	1357.934	1353.319	1354.845	1356.934
725x725	1549.997	1553.978	1550.873	1552.034
750x750	1723.349	1721.325	1723.456	1724.945

Tabel 4. 4 Perbandingan *Running Time* Proses Dekripsi Variasi Ukuran Piksel Citra dan Variasi Kunci *Input* dengan Panjang Kurang Dari 128 Bit (dalam detik)

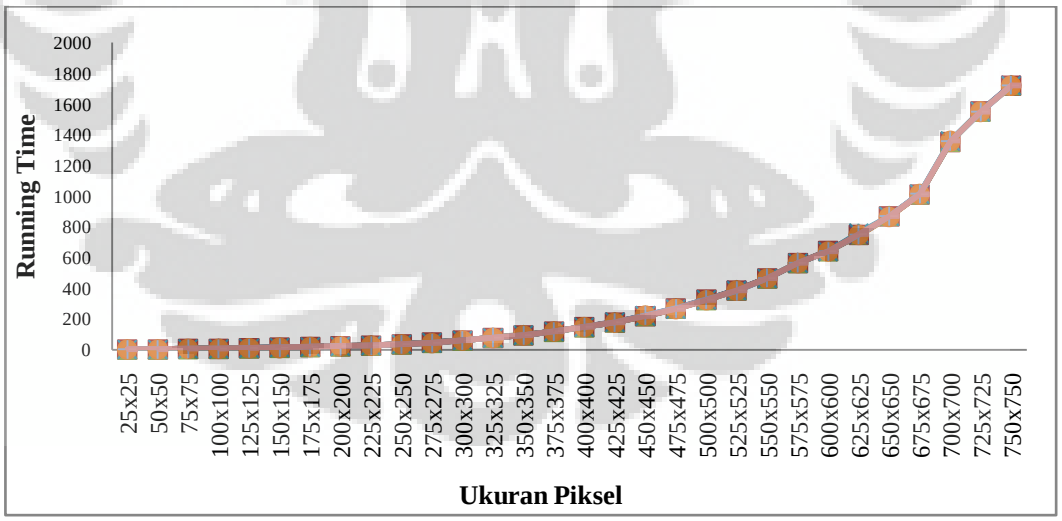
Ukuran Piksel Citra	Jenis Kunci dengan panjang kurang dari 128 bit			
	KK1	KK2	KK3	KK4
25x25	0.382	0.342	0.359	0.362
50x50	1.244	1.132	1.173	1.158
75x75	2.602	2.646	2.703	2.603
100x100	4.634	4.702	4.695	4.726
125x125	7.438	7.562	7.482	7.434
150x150	10.930	10.670	11.049	11.037
175x175	15.442	16.033	15.722	15.923
200x200	21.211	21.021	20.923	20.445
225x225	25.991	25.732	26.346	26.327
250x250	33.245	34.238	34.222	32.992
275x275	44.223	42.987	42.933	43.286
300x300	58.022	56.985	57.546	57.287
325x325	75.929	76.336	77.144	77.028
350x350	94.236	93.329	92.911	94.201
375x375	116.331	115.656	115.872	116.772
400x400	146.394	146.311	144.921	145.892
425x425	177.208	176.991	176.917	176.890
450x450	219.426	219.732	221.025	219.378
475x475	263.982	264.783	266.893	264.561
500x500	324.672	322.769	326.984	323.389
525x525	383.459	385.692	382.069	384.593
550x550	465.048	462.983	466.023	464.374
575x575	564.792	563.782	565.902	564.630
600x600	640.432	639.153	638.992	641.024
625x625	759.876	745.893	748.932	752.038
650x650	868.932	865.941	870.023	869.456
675x675	1014.578	1012.238	1015.825	1010.233
700x700	1360.381	1358.932	1358.972	1360.032
725x725	1551.235	1550.756	1549.893	1550.347
750x750	1720.216	1719.781	1723.851	1721.653

Gambar 4.10 merupakan grafik perbandingan *running time* proses enkripsi pada citra dengan ukuran piksel 500 x 500 dan dengan delapan buah jenis kunci yang di uji coba.



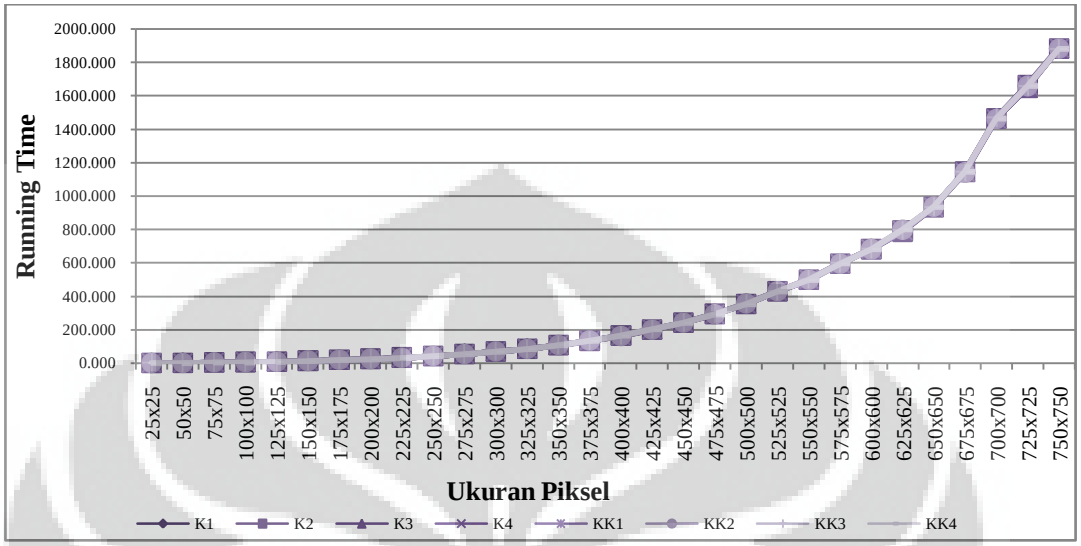
Gambar 4. 10 Grafik Perbandingan *Running Time* Proses Enkripsi Variasi Kunci *Input* pada Citra dengan Ukuran Piksel 500 x 500 (dalam detik)

Gambar 4.11 merupakan grafik perbandingan *running time* proses enkripsi pada jenis kunci berukuran 128 bit dan dengan jenis kunci angka saja di semua ukuran piksel yang di uji coba.

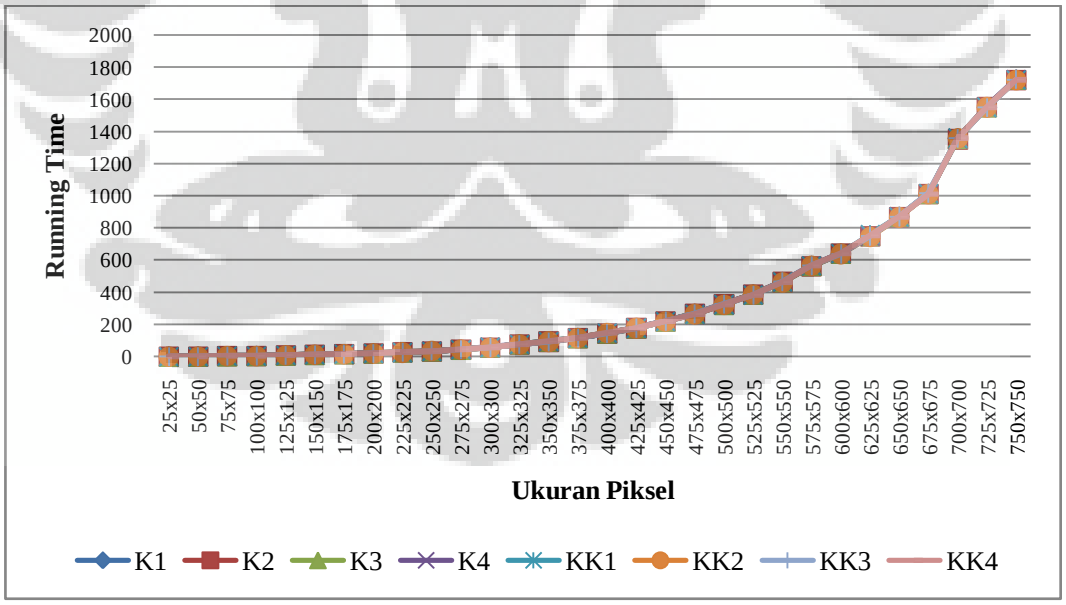


Gambar 4. 11 Grafik Perbandingan Running Time Proses Enkripsi Variasi Ukuran Piksel dan Jenis Kunci *Input* Angka Saja Berukuran 128 Bit (dalam detik)

Gambar 4.12 dan Gambar 4.13 menunjukkan grafik perbandingan running time proses enkripsi dan proses dekripsi variasi ukuran piksel dan variasi jenis kunci *input*.



Gambar 4. 12 Grafik Perbandingan Proses Enkripsi pada Variasi Ukuran Piksel dan Variasi Jenis Kunci *Input* (dalam detik)



Gambar 4. 13 Grafik Perbandingan Proses Dekripsi pada Variasi Ukuran Piksel dan Variasi Jenis Kunci *Input* (dalam detik)

Tabel 4.1, Tabel 4.2, Tabel 4.3, Tabel 4.4, Gambar 4.11, Gambar 4.12, dan Gambar 4.13 menunjukkan bahwa terdapat perbedaan yang cukup signifikan untuk *running time* dengan ukuran piksel yang berbeda. Makin besar ukuran piksel, maka makin besar pula *running time* yang dibutuhkan.

Dalam kasus variasi panjang kunci, ternyata tidak dihasilkan perbedaan yang cukup signifikan antara kunci *input* dengan ukuran 128 bit dan kunci *input* dengan ukuran kurang dari 128 bit. Ini dikarenakan, dalam implementasi yang dilakukan dalam tugas akhir ini, semua kunci *input* yang dimasukkan akan diolah sepanjang 128 bit. Jika kunci *input* yang masuk kurang dari 128 bit, maka akan ditambahkan elemen 00, sehingga yang akan diolah tetap sepanjang 128 bit, sehingga membutuhkan running time yang tidak jauh berbeda.

Dalam kasus variasi jenis kunci *input* pun, tidak ditemukan perbedaan *running time* yang cukup signifikan, karena semua angka, kunci, simbol, dan kombinasi ketiganya, akan diubah semua menjadi bilangan ASCII, sehingga tidak terjadi perbedaan yang besar dari segi *running time* ketika menggunakan variasi jenis kunci.

Jenis kunci angka saja, huruf saja, simbol saja, dan kombinasi serta variasi panjang kunci input tidak menghasilkan perbedaan yang cukup signifikan tersaji pada Gambar 4.10.

Berdasarkan alasan tersebut, akan lebih baik jika input kunci untuk mengamankan data citra menggunakan kunci sepanjang 128 bit atau 16 karakter dan jenis kunci kombinasi antara huruf, angka, dan simbol.

Jika terjadi serangan dengan menggunakan teknik *brute force*, maka input kunci sepanjang 128 bit dan jenis kunci kombinasi akan lebih aman, dikarenakan probabilitas kunci akan terbongkar lebih kecil dibandingkan jenis kunci yang lain. Hal ini akan ditunjukkan dalam Tabel 4.5.

Tabel 4. 5 Probabilitas Terbongkarnya Kunci Terhadap Serangan *Brute Force*

	Angka Saja	Simbol Saja	Huruf Saja	Kombinasi
Domain	10	32	52	94
Probabilitas Terbongkar	$1/10^{16} = 10^{-16}$	$1/ 32^{16} =$ $8,27 \times 10^{-25}$	$1/ 52^{16} =$ $3,49 \times 10^{-28}$	$1/ 94^{16} =$ $2,69 \times 10^{-32}$

BAB 5
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan pembahasan dalam tugas akhir ini terkait aplikasi algoritma Rijndael dalam pengamanan citra digital dan berdasarkan hasil uji coba yang telah dilakukan, didapat beberapa kesimpulan sebagai berikut:

1. Algoritma Rijndael berhasil diaplikasikan untuk mengamankan data citra digital dan dapat diimplementasikan di basis GF (2^8) dengan polinomial $f(x) = x^8 + x^4 + x^3 + x + 1$ dalam *software* Matlab.

2. Berdasarkan uji coba yang dilakukan dalam tugas akhir ini, didapatkan bahwa ukuran panjang kunci (128 bit dan kurang dari 128 bit) dan jenis kunci input (angka saja, huruf saja, simbol saja dan kombinasi ketiganya) tidak terlalu mempengaruhi *running time* dari proses enkripsi dan dekripsi algoritma Rijndael. Maka itu untuk lebih meningkatkan keamanan citra digital, lebih baik digunakan jenis kunci kombinasi dari angka, huruf dan simbol, dan dengan ukuran kunci yang maksimal (dalam hal ini adalah 128 bit)

5.1 Saran

Saran yang perlu diperhatikan untuk penelitian lebih lanjut adalah:

1. Di dalam tugas akhir ini data yang dirahasiakan berupa citra digital *grayscale*, untuk penelitian selanjutnya, bisa digunakan citra digital berwarna dalam implementasinya agar mendapat hasil yang lebih bervariasi untuk pengamanan citra.

2. Untuk penelitian selanjutnya, dapat mengaplikasikan algoritma Rijndael dalam mengamankan data selain citra digital, seperti audio, video, ataupun multimedia.

3. Ukuran kunci yang digunakan dalam implementasi algoritma rijndael di dalam tugas akhir ini menggunakan ukuran 128 bit, untuk penelitian selanjutnya bisa digunakan kunci dengan ukuran beragam yakni 128 bit, 192 bit, dan 256 bit.

DAFTAR PUSTAKA

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

Anton, Howard.,dan Rorres,Chris. (2005). *Aljabar Linier Elementer (9th ed.)*. New Jersey: John Wiley & Sons, Inc

Ayres, Frank, Jr., dan Mendelson, E. (2004). *Schaum's Outlines: Kalkulus (4th ed.)*. Jakarta: Erlangga

Herstein, I. N. (1995). *Abstract Algebra (3rd ed.)*. New Jersey: Prentice-Hall.

Iskandar, H. (2001). Tugas Akhir: *Algoritma Enkripsi Advances Encryption Standard (AES): Rijndael.* Depok: Universitas Indonesia.

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

Purnomo, Mauridhi, Hery., dan Muntasa, Arif. (2010) *Pengolahan Citra Digital dan Ekstraksi Fitur.* Yogyakarta: Graha Ilmu

Rosen,Kenneth,H. (1999). *Discrete Mathematics and Its Applications (4th ed.).* NewYork: McGraw-Hill

Satria, E. (2009). Tugas Akhir: *Studi Algoritma Rijndael dalam Sistem Keamanan Data.* Medan: Universitas Sumatera Utara.

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

Surian, D. (1996). Algoritma Kriptografi AES Rijndael. *Jurnal Teknik Elektro* , 97-101.

Tilborg, Henk C.A, van., dan Jajodia, Sushil. (2011). *Encyclopedia of Cryptography and Security*. New York: Springer

Widodo, Tutur. (2010). Skripsi: *Lapangan Berhingga.* Surakarta: Universitas Negeri Solo

LAMPIRAN

Lampiran 1 Pembuktian Teorema 2.1

Jika $f: A \to B$ adalah fungsi bijektif, maka terdapat sebuah fungsi invers dari f yaitu $f^{-1}: B \to A$ yang bijektif pula.

Bukti:

$f: A \to B$ bijektif, maka terdapat $f^{-1}: B \to A$ sedemikian sehinga $f^{-1} \circ f = id_A$ dan $f \circ f^{-1} = id_B$ dengan id_A adalah fungsi identitas di A dan id_B adalah Fungsi identitas di B. Maka,

$$f \circ f^{-1}(y) = y = id_B$$
$$f^{-1} \circ f(x) = x = id_A$$

Akan dibuktikan: $f^{-1}: B \to A$ merupakan fungsi injektif

Ambil $x, y \in B$ dimana $f^{-1}(x) = f^{-1}(y)$

$$f\big(f^{-1}(x)\big) = f\big(f^{-1}(y)\big) \quad \text{(karena } f \text{ fungsi)}$$
$$x = y$$

∴ Untuk sembarang $x, y \in B$ dimana jika $f^{-1}(x) = f^{-1}(y)$ maka $x = y$, sehingga $f^{-1}: B \to A$ merupakan fungsi injektif.

Akan dibuktikan: $f^{-1}: B \to A$ merupakan fungsi surjektif

Ambil $a \in A$, maka $f(a) \in B$.

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

∴ Untuk sembarang $a \in A, f(a) \in B$, sedemikian sehingga $f^{-1}\big(f(a)\big) = a$, maka terbukti $f^{-1}: B \to A$ merupakan fungsi surjektif.

∴ Karena $f^{-1}: B \to A$ merupakan fungsi injektif dan surjektif, maka $f^{-1}: B \to A$ merupakan fungsi bijektif.

Lampiran 2 Pembuktian Teorema 2.2 dan 2.3

Teorema 2.2:

Jika $g: S \to T$ dan $f: T \to U$ pemetaan yang bijektif, maka $f \circ g: S \to U$ juga pemetaan yang bijektif

Bukti:

Akan dibuktikan: $g: S \to T$ dan $f: T \to U$ fungsi injektif $\to f \circ g: S \to U$ merupakan pemetaan yang injektif.

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

$$f\big(g(s_1)\big) = f\big(g(s_2)\big)$$

$g(s_1) = g(s_2)$ (karena f fungsi injektif)

$s_1 = s_2$ (karena g fungsi injektif)

$\therefore$ Untuk sembarang $s_1, s_2 \in S$, dimana $(f \circ g)(s_1) = s_1$ dan $(f \circ g)(s_2) = s_2$, sehingga terbukti $f \circ g: S \to U$ merupakan pemetaan yang injektif.

Akan dibuktikan: Jika $g: S \to T$ dan $f: T \to U$ fungsi surjektif maka $f \circ g$: $S \to U$ merupakan fungsi yang surjektif.

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.

$\therefore$ Karena $f \circ g : S \to U$ merupakan fungsi yang injektif dan surjektif maka terbukti $f \circ g : S \to U$ merupakan pemetaan yang bijektif.

Teorema 2.3:

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)

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

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

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

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

Akibat 2.4:

Jika $f_i: A_{i-1} \to A_i$ pemetaan bijektif $\forall i = 1,2,\dots,n,$ maka $(f_n \circ f_{n-1} \circ \dots \circ f_2 \circ f1$ juga pemetaan bijektif.

Bukti:

Berdasarkan Teorema 2.3, maka $(f_n \circ f_{n-1})$ bijektif.

$(f_{n-1} \circ f_{n-2})$ bijektif.

$\vdots$

$(f_2 \circ f_1)$ bijektif.

Maka $(f_n \circ f_{n-1}) \circ (f_{n-1} \circ f_{n-2})$ bijektif

$(f_n \circ f_{n-1}) \circ (f_{n-1} \circ f_{n-2}) \circ \dots \circ (f_2 \circ f_1)$ bijektif

$\therefore$ Terbukti $(f_n \circ f_{n-1} \circ f_{n-1} \circ f_{n-2} \circ \dots \circ f_2 \circ f_1)$ bijektif

Lampiran 3 Pembuktian Teorema 2.6

Teorema 2.6:

Jika gcd $(a(x), c(x)) = 1$ maka $a(x)$ *invertible* $mod\ c(x)$

Bukti:

Jika gcd $(a(x), c(x)) = 1$ maka terdapat $p(x)$ dan $q(x)$dimana,

$a(x)p(x) + c(x)q(x) = 1$

$a(x)p(x) = -q(x)c(x) + 1$

Ambil $p_0(x) = p(x)$dan $q_0(x) = -q(x)$, maka

$a(x)p_0(x) = q_0(x)c(x) + 1$

$a(x)p_0(x) = 1\ mod\ c(x)$

Maka a *invertible* $mod\ c(x)$

∴Terbukti jika gcd $(a(x), c(x)) = 1$ maka $a(x)$ *invertible* $mod\ c(x)$.

Lampiran 4 Beberapa Bagian *Pseudo Code Key Schedule* Algoritma Rijndael

```
r=input('masukan key yang anda inginkan(maksimal 16
karakter)=','s'); %cipher key
[x,y]=size(r);
if y<=16
desimal=zeros(1,16);    %dummy
a=0;
for i=y:-1:1
    desimal(1,i)=double(r(y-a));
    a=a+1;
end
k=reshape(desimal,4,4);
kunc(:,:,1)=k;       %cipher key = kunci 1

for w=1:10
    a=k(:,1);
    b=[k(14);k(15);k(16);k(13)]; %rootword
    rcon=[1 2 4 8 16 32 64 128 27 54;0 0 0 0 0 0 0 0 0 0;0 0 0 0 0
0 0 0 0 0;0 0 0 0 0 0 0 0 0 0];
    c=rcon(:,w); %r-con

    b1=reshape(b,1,4); %sub byte
    b1=double(b1);

    sbox= {Hint: masukkan nilai-nilai S-box dalam bentuk matriks}
    wee=hex2dec(sbox);
    box=reshape(wee,16,16);
    b2=zeros(1,4);
    for i=1:4
        baris=floor(b1(i)/16)+1;
        kolom=mod(b1(i),16)+1;
        b2(i)=box(baris,kolom);
    end

    satu=dec2bin(a,8);
    dua=dec2bin(b2,8);
    tiga=dec2bin(c,8);

    k1=zeros(4,8); %xor

    for i=1:4
        for j=1:8
            k1(i,j)=mod(satu(i,j)+dua(i,j)+tiga(i,j),2);
        end
    end

    d=k(:,2);
    empat=dec2bin(d,8);

    k2=zeros(4,8);

    for i=1:4
```

```
    for j=1:8
        k2(i,j)=mod(k1(i,j)+empat(i,j),2);
    end
end

e=k(:,3);
lima=dec2bin(e,8);

k3=zeros(4,8);

for i=1:4
    for j=1:8
        k3(i,j)=mod(k2(i,j)+lima(i,j),2);
    end
end

f=k(:,4);
enam=dec2bin(f,8);

k4=zeros(4,8);

for i=1:4
    for j=1:8
        k4(i,j)=mod(k3(i,j)+enam(i,j),2);
    end
end

s=zeros(4,1); %ubah ke desimal
for i=1:4
    for j=1:8
        s(i)=s(i)+k1(i,j)*2^(8-j);
    end
end

t=zeros(4,1);
for i=1:4
    for j=1:8
        t(i)=t(i)+k2(i,j)*2^(8-j);
    end
end

u=zeros(4,1);
for i=1:4
    for j=1:8
        u(i)=u(i)+k3(i,j)*2^(8-j);
    end
end

v=zeros(4,1);
for i=1:4
    for j=1:8
        v(i)=v(i)+k4(i,j)*2^(8-j);
    end
end

kunc(:,:,w+1)=[s t u v];
k=kunc(:,:,w+1);
```

```
end
else
    kunc=0;
    fprintf('Input salah, password tidak boleh lebih dari 16
karakter.\n');
end
end
```

Lampiran 5 . *Pseudo Code* Pengambilan Blok dari Matriks Citra Digital dalam Algoritma Rijndael

```
foto=input('masukan nama foto yang ingin dienkripsi= ','s');
kunc=key();
if kunc==0
    fprintf('Maaf');
else
    tic;

     %% Mengatur dummy dan matriks blok plaintext
    foto1=imread(foto);
    I=foto1;
    for z=1:3
    foto1=I(:,:,z);
    [x1 y1]=size(foto1);
    a1=x1*y1;
    foto2=reshape(foto1,1,a1);
    foto3=double(foto2);
    a2=ceil(a1/16);                    %banyak blok yang dibuat
    a3=a2*16;
    foto4=zeros(1,a3);                %Tambahkan dummy di belakang
    for i=1:a1
        foto4(1,i)=foto3(1,i);
    end
    a4=1;
 %dibagi blok-blok sepanjang 128 bit->Jadi ada 16 kolom
    for i=1:a2
        foto5(i,1:16)=foto4(1,a4:a4+15);
        a4=a4+16;
    end
```

Lampiran 6 Beberapa Bagian *Pseudo Code* Proses Enkripsi Algoritma Rijndael

```
for p=1:a2
    f3=reshape(foto5(p,:),4,4);

  %Transformasi Add Round Key di awal putaran

    b1=dec2bin(f3( :,1),8);
    b2=dec2bin(f3(:,2),8);
    b3=dec2bin(f3(:,3),8);
    b4=dec2bin(f3(:,4),8);

    kunc1=dec2bin(kunc(:,1,1),8);  %cipher key
    kunc2=dec2bin(kunc(:,2,1),8);
    kunc3=dec2bin(kunc(:,3,1),8);
    kunc4=dec2bin(kunc(:,4,1),8);

    c1=zeros(4,8); %xor

    for i=1:4
        for j=1:8
            c1(i,j)=mod(b1(i,j)+kunc1(i,j),2);
        end
    end

    c2=zeros(4,8); %xor

    for i=1:4
        for j=1:8
            c2(i,j)=mod(b2(i,j)+kunc2(i,j),2);
        end
    end
    c3=zeros(4,8); %xor

    for i=1:4
        for j=1:8
            c3(i,j)=mod(b3(i,j)+kunc3(i,j),2);
        end
    end

    c4=zeros(4,8); %xor

    for i=1:4
        for j=1:8
            c4(i,j)=mod(b4(i,j)+kunc4(i,j),2);
        end
    end

    e1=zeros(4,1); %ubah ke desimal
    for i=1:4
        for j=1:8
```

```
        e1(i)=e1(i)+c1(i,j)*2^(8-j);
    end
end

e2=zeros(4,1); %ubah ke desimal
for i=1:4
    for j=1:8
        e2(i)=e2(i)+c2(i,j)*2^(8-j);
    end
end
e3=zeros(4,1); %ubah ke desimal
for i=1:4
    for j=1:8
        e3(i)=e3(i)+c3(i,j)*2^(8-j);
    end
end
e4=zeros(4,1); %ubah ke desimal
for i=1:4
    for j=1:8
        e4(i)=e4(i)+c4(i,j)*2^(8-j);
    end
end

ARK=[e1 e2 e3 e4];
f4=reshape(ARK,1,16);
w=1;
for w=1:10   %10 iterasi

    sbox= {Hint: Masukkan nilai-nilai S-Box}
    wee=hex2dec(sbox);
    box=reshape(wee,16,16);

    %Transformasi Sub byte

    sub byte=zeros(1,16);
    for i=1:16
        baris=floor(f4(i)/16)+1;
        kolom=mod(f4(i),16)+1;
        sub byte(i)=box(baris,kolom);
    end
    sub byte=reshape(sub byte,4,4);

    %Transformasi Shiftrow

    shiftrow(1,:)=sub byte(1,:);
    shiftrow(2,:)=circshift(sub byte(2,:),[0 -1]);
    shiftrow(3,:)=circshift(sub byte(3,:),[0 -2]);
    shiftrow(4,:)=circshift(sub byte(4,:),[0 -3]);

    if w~=10

        %Transformasi Mix Columns

        matriks=[2 3 1 1;1 2 3 1;1 1 2 3;3 1 1 2];
        r1=gf(matriks,8,283)*gf(shiftrow,8,283);
```

```
            r2=r1.x;
            MixColumns=double(r2)
        else
            MixColumns=shiftrow;
        end

        %Selanjutnya lakukan transformasi Add Round Key pada
tiap iterasi (Sama seperti Add Round Key sebelumnya)
```

Lampiran 7 Beberapa Bagian *Pseudo Code* Proses Dekripsi Algoritma Rijndael

```
%Transformasi Add Round Key

b1=dec2bin(f3(:,1),8);
b2=dec2bin(f3(:,2),8);
b3=dec2bin(f3(:,3),8);
b4=dec2bin(f3(:,4),8);

kunc1=dec2bin(kunc(:,1,11),8);
kunc2=dec2bin(kunc(:,2,11),8);
kunc3=dec2bin(kunc(:,3,11),8);
kunc4=dec2bin(kunc(:,4,11),8);

c1=zeros(4,8); %xor

for i=1:4
    for j=1:8
        c1(i,j)=mod(b1(i,j)+kunc1(i,j),2);
    end
end

c2=zeros(4,8); %xor

for i=1:4
    for j=1:8
        c2(i,j)=mod(b2(i,j)+kunc2(i,j),2);
    end
end
c3=zeros(4,8); %xor

for i=1:4
    for j=1:8
        c3(i,j)=mod(b3(i,j)+kunc3(i,j),2);
    end
end

c4=zeros(4,8); %xor
```

```
for i=1:4
    for j=1:8
        c4(i,j)=mod(b4(i,j)+kunc4(i,j),2);
    end
end

e1=zeros(4,1); %ubah ke desimal
for i=1:4
    for j=1:8
        e1(i)=e1(i)+c1(i,j)*2^(8-j);
    end
end

e2=zeros(4,1); %ubah ke desimal
for i=1:4
    for j=1:8
        e2(i)=e2(i)+c2(i,j)*2^(8-j);
    end
end
e3=zeros(4,1); %ubah ke desimal
for i=1:4
    for j=1:8
        e3(i)=e3(i)+c3(i,j)*2^(8-j);
    end
end
e4=zeros(4,1); %ubah ke desimal
for i=1:4
    for j=1:8
        e4(i)=e4(i)+c4(i,j)*2^(8-j);
    end
end

ARK=[e1 e2 e3 e4];

%Transformasi Inv Shiftrow

shiftrow(1,:)=ARK(1,:);
shiftrow(2,:)=circshift(ARK(2,:),[0 1]);
shiftrow(3,:)=circshift(ARK(3,:),[0 2]);
shiftrow(4,:)=circshift(ARK(4,:),[0 3]);

shift=reshape(shiftrow,1,16);
invsbox= {hint: Masukkan nilai-nilai invers s-box}
wee=hex2dec(invsbox); %sub byte
box=reshape(wee,16,16);

%Transformasi Inv SubBytes

sub byte=zeros(1,16);
for i=1:16
    baris=floor(shift(i)/16)+1;
    kolom=mod(shift(i),16)+1;
    sub byte(i)=box(baris,kolom);
end
```

```
            subbyte=reshape(subbyte,4,4);

            %Iterasi ke 9-1

            for w=9:-1:1

                %Lakukan Transformasi Add Round Key di awal
iterasi (sama seperti transformasi Add Round Key sebelumnya)

            ARK=[e1 e2 e3 e4];

                % Transformasi Inv Mix Columns

            matriks=[14 11 13 9;9 14 11 13;13 9 14 11;11 13 9 14];
            r1=gf(matriks,8,283)*gf(ARK,8,283);
            r2=r1.x;
            MixColumns=double(r2);

                % Transformasi Inv Shiftrow

                shiftrow(1,:)=MixColumns(1,:);
                shiftrow(2,:)=circshift(MixColumns(2,:),[0 1]);
                shiftrow(3,:)=circshift(MixColumns(3,:),[0 2]);
                shiftrow(4,:)=circshift(MixColumns(4,:),[0 3]);

                %Transformasi Inv Subbyte

                subbyte2=zeros(1,16);
                for i=1:16
                    baris=floor(shiftrow(i)/16)+1;
                    kolom=mod(shiftrow(i),16)+1;
                    subbyte2(i)=box(baris,kolom);
                end
                subbyte2=reshape(subbyte2,4,4);
                subbyte=subbyte2;
            end

            %Lakukan Transformasi Add Round Key terakhir (sama
seperti transformasi Add Round Key sebelumnya)
```