IMPLEMENTASI DNA-*VIGENERE CIPHER* DAN ALGORITMA *RIJNDAEL* DENGAN ENKRIPSI PEMBANGKIT KUNCI BERGESER UNTUK CITRA *GRAYSCALE*

SKRIPSI

Oleh

Dony Romansyah
NIM 131810101050

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKAN DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2017

IMPLEMENTASI DNA-*VIGENERE CIPHER* DAN ALGORITMA *RIJNDAEL* DENGAN ENKRIPSI PEMBANGKIT KUNCI BERGESER UNTUK CITRA *GRAYSCALE*

SKRIPSI

diajukan guna memenuhi tugas akhir dan memenuhi salah satu syarat untuk
menyelesaikan Program Studi Matematika (S1)
dan mencapai gelar Sarjana Sains

Oleh

Dony Romansyah
NIM 131810101050

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKAN DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2017

PERSEMBAHAN

Bismillahirrohmanirrohim

Dengan rahmat Allah SWT yang Maha Pengasih lagi Maha Penyayang, sholawat serta salam terhadap baginda Rasulullah Muhammad SAW. Saya persembahkan skripsi ini sebagai wujud syukur dan rasa terima kasih untuk:

1. Orang tua saya, Bapak Mulyono dan Ibu Saniti, yang selalu memberikan segala bentuk dukungan, motivasi, dan do'anya untuk kemudahan, kelancaran, dan kesuksesan saya.
2. Seluruh Keluarga besar yang juga memberikan segenap dukungan dan doanya untuk saya.
3. Teman-teman ATLAS 13 dan Smokers yang sudah berjuang bersama sampai akhir.
4. Almamater tercinta Jurusan Matematika FMIPA Universitas Jember
5. Serta, beberapa pihak yang tidak bisa saya sebutkan satu persatu, yang telah memberikan dukungannya.

MOTTO

"*The story of our lives is exactly how we wrote it*". [1]

"*Manusia berusaha, Tuhan yang menentukan*". [2]

[1] Kang Gary
[2] Midorima Shintarou

PERNYATAAN

Saya yang bertanda tangan dibawah ini:

Nama: Dony Romansyah

NIM : 131810101050

menyatakan dengan sesungguhnya bahwa skripsi yang berjudul "Implementasi DNA-*Vigenere Cipher* dan Algoritma *Rijndael* dengan Enkripsi-Pembangkit Kunci Bergeser untuk Citra *Grayscale*" adalah benar-benar hasil karya sendiri, kecuali kutipan yang telah disebutkan sumbernya, belum pernah diajukan di institusi manapun, dan bukan karya jiplakan. Saya bertanggung jawab atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenarnya, tanpa ada tekanan dan paksaan dari pihak manapun serta bersedia mendapat sanksi akademik jika ternyata di kemudian hari pernyataan ini tidak benar.

Jember, November 2017

Yang menyatakan,

Dony Romansyah

NIM 131810101050

SKRIPSI

IMPLEMENTASI DNA-*VIGENERE CIPHER* DAN ALGORITMA *RIJNDAEL* DENGAN ENKRIPSI PEMBANGKIT KUNCI BERGESER UNTUK CITRA *GRAYSCALE*

Oleh

Dony Romansyah
NIM 131810101050

Pembimbing:

Dosen Pembimbing Utama : Kusbudiono, S.Si., M.Si.

Dosen Pembimbing Anggota : Ahmad Kamsyakawuni, S.Si., M.Kom.

PENGESAHAN

Skripsi berjudul "Implementasi DNA-*Vigenere Cipher* dan Algoritma *Rijndael* dengan Enkripsi-Pembangkit Kunci Bergeser untuk Citra *Grayscale*" telah diuji dan disahkan pada:

hari, tanggal :

tempat : Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Tim Penguji:

Ketua,	Anggota I,
Kusbudiono, S.Si., M.Si.	Ahmad Kamsyakawuni, S.Si., M.Kom.
NIP. 197704302005011001	NIP. 197211291998021001
Anggota II,	Anggota III,
Prof. Drs. Kusno, DEA., PhD.	M. Ziaul Arif, S.Si., M.Sc.
NIP. 196101081986021001	NIP. 198501112008121002

Mengesahkan
Dekan,

Drs. Sujito, Ph.D.
NIP. 196102041987111001

RINGKASAN

Implementasi DNA-*Vigenere Cipher* dan Algoritma *Rijndael* dengan Enkripsi-Pembangkit Kunci Bergeser untuk Citra *Grayscale*; Dony Romansyah, 131810101050; 2017; 67 Halaman; Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Perkembangan era globalisasi yang sangat pesat memberikan dampak yang signifikan terhadap perkembangan teknologi dan informasi saat ini. Perubahan yang terjadi tidak hanya berdampak postif namun juga ada yang berdampak negatif. Salah satunya adalah penyalahgunaan data privasi seseorang oleh pihak yang tidak berwenang. Perlindungan data dan informasi diperlukan untuk menjamin data privasi seseorang aman dan tidak disalahgunakan oleh orang lain.

Kriptografi adalah ilmu untuk menjaga atau mengamankan suatu informasi dengan cara mengacak atau menyembunyikan suatu informasi agar sulit untuk dianalisa. Algoritma yang digunakan dalam penelitian ini adalah algoritma DNA-Vigenere Cipher dan algoritma Rijndael. Pada prosesnya, algoritma DNA-Vigenere Cipher menggabungkan algoritma Kriptografi DNA dan Vigenere Cipher. DNA-Vigenere Cipher menggunakan barisan *deoxyribonuleic acid* (DNA), yaitu ACTG (*adenine*, *cytosine*, *thymine*, *guanine*), serta konsep pemetaan atau urutan pergeseran yang berbeda pada barisan DNA sebagai dasar perhitungan komputasinya. Namun, sifat berulang dari *key*-nya merupakan kelemahan dari DNA-*Vigenere Cipher*. Key yang sangat pendek akan memudahkan kriptanalis dalam memecahkan algoritmanya. Untuk mengatasi kelemahan dari algoritma ini, dilakukan modifikasi kunci agar kunci yang dihasilkan lebih beragam dengan algoritma Enkripsi-Pembangkit Kunci Bergeser, kemudian dilakukan enkripsi lebih lanjut dengan algoritma Rijndael. Algoritma Rijndael merupakan Algoritma modern pengganti algoritma DES , dan merupakan algoritma yang banyak dipilih untuk mengamankan suatu data/pesan karena efisien dan pengamanannya yang kuat.

Data yang digunakan dalam penelitian ini adalah *plain image* berupa citra *grayscale* dan data *text* sebagai kunci dari semua algoritmanya. Citra tersebut akan dienkripsi terlebih dahulu dengan DNA-*Vigenere Cipher* menggunakan kunci hasil pembangkitan dengan metode Enkripsi-Pembangkit Kunci Bergeser kemudian akan diperkuat keamannnya dengan mengenkripsi citra hasil enkripsi tadi menggunakan algoritma *Rijndael*. Hasil dari proses enkripsi akan menghasilkan sebuah citra tersandi atau *cipher image* yang sudah tidak mengandung informasi dari *plain image*-nya.

Analisis keamanan digunakan untuk mengetahui keamanan dari algoritma atau metode yang diajukan. Analisis keamanan yang digunakan yaitu analisis histogram derajat keabuan, analisis diferensial, analisis sensitivitas kunci, analisis panjang kunci dan analisis modifikasi kunci telah menunjukkan bahwa algoritma yang diajukan aman terhadap serangan statistik, memiliki histogram derajat keabuan yang merata, kunci yang variatif, sensitif, dan tidak dapat diprediksi panjang kuncinya.

PRAKATA

Puji syukur kehadirat Allah SWT atas rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "Implementasi DNA-*Vigenere Cipher* dan Algoritma *Rijndael* dengan Enkripsi-Pembangkit Kunci Bergeser untuk Citra *Grayscale*". Skripsi ini disusun untuk memenuhi salah satu syarat pada program pendidikan strata satu (S1) Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Skripsi ini disusun melalui beberapa tahap, baik dalam bentuk seminar maupun bimbingan intensif. Skripsi ini tidak akan terselesaikan tanpa adanya bantuan dari beberapa pihak, serta kerja keras dan keuletan dari diri pribadi. Oleh karena itu, dalam kesempatan ini penulis mengucapkan terima kasih atas bantuan dan bimbingan dalam penyusunan sripsi ini, terutama kepada yang terhormat:

1. Drs. Sujito, Ph.D., selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember;
2. Kusbudiono, S.Si., M.Si., selaku Ketua Jurusan Matematika dan Ilmu Pengetahuan Alam Universitas Jember, selaku Dosen Pembimbing Akademik dan selaku Dosen Pembimbing Utama, yang telah meluangkan waktu dan pikirannya dalam membimbing penulisan skripsi ini sampai terselesaikan;
3. Ahmad Kamsyakawuni, S.Si., M.Kom., selaku Dosen Pembimbing Anggota, Prof. Drs. Kusno, DEA., PhD., selaku Dosen Penguji I, M. Ziaul Arif, S.Si., M.Sc., selaku Dosen Penguji II yang juga telah meluangkan waktu dan pikiran dalam membimbing penulisan skripsi ini sampai terselesaikan;
4. Dosen dan Karyawan Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember yang telah menyalurkan ilmunya;
5. Kedua orang tua dan keluarga besar yang selalu mendukung dan memberikan seluruh bantuannya dari awal sampai terselesaikannya skripsi ini;
6. Teman-teman dan semua pihak yang juga telah membantu terselesaikannya skripsi ini.

Penulis menyadari masih banyak kekurangan dalam penulisan skripsi ini. Namun, suatu usaha tidak akan berakhir dan berhasil jika tidak dimulai. Oleh

karena itu, penulis mengharapkan kritik dan sarannya demi penyempurnaan skripsi ini. Penulis berharap semoga skripsi ini bermanfaat bagi para pembaca

Jember, November 2017

Penulis

DAFTAR ISI

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

DAFTAR LAMPIRAN

Halaman

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Perkembangan era globalisasi yang semakin canggih membuat perubahan yang signifikan diberbagai aspek kehidupan, misalnya perkembangan Ilmu Pengetahuan, Teknologi dan Informasi global. Kemajuan Ilmu Pengetahuan, Teknologi dan Informasi tidak hanya memberikan dampak positif bagi lingkungan manusia, tetapi juga memberikan banyak dampak negatif. Salah satunya yakni penyalahgunaan data privasi milik seseorang oleh pihak yang tidak berwenang. Proses pengamanan dan perlindungan suatu pesan maupun data sangat diperlukan untuk menjamin data privasi seseorang tersebut tidak disalahgunakan untuk kepentingan umum.

Kriptografi adalah ilmu untuk menjaga atau mengamankan suatu informasi dengan cara mengacak atau menyembunyikan suatu informasi agar sulit untuk dianalisa. Metode-metode dalam kriptografi terus dikembangkan untuk meningkatkan keamanan suatu informasi, yang menyebabkan teknik dan algoritma kriptografi saat ini semakin beragam. Najaftorkaman dan Kazazi (2015) menggabungkan algoritma kriptografi DNA dengan *Vigenere Cipher* untuk mengenkripsi pesan teks.

Kriptografi DNA adalah salah satu algoritma baru dalam ilmu kriptografi yang menggunakan barisan *deoxyribonuleic acid* (DNA), yaitu ACTG (*adenine*, *cytosine*, *thymine*, *guanine*), sebagai dasar perhitungan komputasinya. Sedangkan *Vigenere Cipher* adalah algoritma kriptografi klasik dan termasuk algoritma yang mudah diimplementasikan. Namun, menurut Malhotra (2012), algoritma *Vigenere Cipher* memiliki kelemahan, yaitu sifat berulang dari *key*-nya, bahkan Metode *Kasiski* dan *Friedman* sudah memecahkan *ciphertext* yang sudah dienkripsi menggunakan algoritma *Vigenere Cipher*. Sehingga metode yang dikenalkan oleh Najaftorkaman dan Kazazi ini akan mudah dipecahkan jika panjang *key*-nya yang berupa barisan DNA terlalu pendek dan diketahui oleh *Kriptanalis*.

Sasmita (2010), memperkenalkan modifikasi dari *Vigenere Cipher* dengan Enkripsi-Pembangkit Kunci Bergeser. Modifikasi tersebut ditujukan untuk mengatasi kelemahan dari bentuk *Vigenere Cipher* dengan merubah kunci pendek tersebut agar panjangnya sama dengan plainteks. Pembangkitan kunci baru tidak melakukan proses pengulangan kunci selayaknya metode *Vigenere Cipher* pada umumnya, tetapi dengan cara menggandakan kunci, menggeser pola kunci, lalu mengenkripsi terhadap kunci awal.

Rijmen dan Daemen (1999), mengajukan sebuah proposal standard algoritma kriptografi baru yang dinamakan sebagai Algoritma *Rijndael*, sebagai pengganti algoritma kriptografi lama (*Data Encryption Standard* / DES). Algoritma DES dianggap sudah tidak aman lagi karena dengan perangkat keras khusus kuncinya mampu ditemukan dalam beberapa hari. Algoritma *Rijndael* merupakan algoritma yang banyak dipilih untuk mengamankan suatu data/pesan karena efisien dan pengamanannya yang kuat. Pada tahun 2001, algoritma yang diajukan oleh Rijmen dan Daemen ini kemudian dipilih oleh *National Institute of Standards and Technologi* (NIST) sebagai pengganti algoritma lama (DES).

Pada penelitian ini, penulis ingin memanfaatkan berbagai teknik pengamanan suatu informasi yang sudah dipaparkan. Penulis akan mengimplementasikan DNA-*Vigenere Cipher* dengan algoritma *Rijndael* dan Enkripsi-Pembangkit Kunci Bergeser untuk pengamanan citra *grayscale*. Citra awal akan dienkripsi terlebih dahulu dengan DNA-*Vigenere Cipher* menggunakan kunci hasil pembangkitan dengan metode Enkripsi-Pembangkit Kunci Bergeser kemudian akan diperkuat keamannnya dengan mengenkripsi citra hasil enkripsi tadi menggunakan algoritma *Rijndael*.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, maka rumusan masalah pada penelitian ini adalah :

a. Bagaimana membangkitkan kunci baru untuk metode DNA-*Vigenere Cipher* dan Algoritma *Rijndael* dengan menggunakan metode Enkripsi-Pembangkit Kunci Bergeser.

b. Bagaimana mengenkripsi citra *grayscale* menggunakan algoritma DNA-*Vigenere Cipher* dan Algoritma *Rijndael*, dengan kunci hasil pembangkitan.
c. Bagaimana mendekripsi citra *grayscale* yang telah dienkripsi menggunakan metode yang sama.
d. Bagaimana analisis keamanan dari metode DNA-*Vigenere Cipher* dan Algoritma *Rijndael*.

1.3 Tujuan Penelitian

Adapun tujuan dari penelitian ini adalah :

a. Membangkitkan kunci baru untuk metode DNA-*Vigenere Cipher* dan Algoritma *Rijndael* dengan menggunakan metode Enkripsi-Pembangkit Kunci Bergeser.
b. Mengenkripsi citra *grayscale* menggunakan algoritma DNA-*Vigenere Cipher* dan Algoritma *Rijndael*, dengan kunci hasil pembangkitan.
c. Mendekripsi citra *grayscale* yang telah dienkripsi menggunakan metode yang sama.
d. Menganalisis keamanan dari metode DNA-*Vigenere Cipher* dan Algoritma *Rijndael*.

1.4 Manfaat Penelitian

Adapun manfaat yang diharapkan pada penelitian ini adalah :

a. Mampu mengenkripsi citra *grayscale* menggunakan metode yang diajukan.
b. Mampu mendekripsi citra hasil enkripsi menggunakan metode yang diajukan.
c. Mampu menganalisis keamanan dari metode DNA-*Vigenere Cipher* dan Algoritma *Rijndael*.

BAB 2. TINJAUAN PUSTAKA

2.1 Kriptografi

Kriptografi berasal dari dua kata dari bahasa Yunani, yaitu *cryptós* (rahasia) dan *gráphein* (tulisan). Oleh sebab itu, dapat diartikan bahwa kriptografi mempelajari tentang tulisan rahasia atau tulisan tersembunyi. Secara garis besar, ilmu kriptografi mempelajari teknik untuk menyembunyikan, melindungi dan mengamankan suatu informasi dengan cara membuat suatu bentuk baru (sandi) sehingga sulit untuk dipahami dan dibaca oleh indera visual seseorang (Munir, 2002).

Awal mula perkembangan ilmu kriptografi sebatas dipahami sebagai ilmu tentang penyandian/penyembunyian pesan (Sadikin, 2012). Seiring perkembangan jaman, kriptografi tidak hanya digunakan sebagai penyembunyian pesan *privacy* seseorang saja, tetapi juga dipakai sebagai media keamanan informasi seperti *Confidelity* (kerahasiaan), *Data integrity* (keutuhan data), *Autentication* (otentik), dan *Non-repudiation* (anti-penyangkalan).

Beberapa istilah penting dalam kriptografi adalah *plaintext*, *ciphertext*, enkripsi, dekripsi, kunci (*key*), dan algoritma. *Plaintext* merupakan informasi awal yang bisa dibaca. *Ciphertext* merupakan informasi hasil pesan *plaintext* yang sudah disandikan. Enkripsi adalah teknik untuk menjadikan data *plaintext* agar tidak dapat dibaca. Dekripsi adalah teknik untuk mengembalikan *ciphertext* menjadi *plaintext* kembali. Kunci (*key*) berfungsi untuk mengatur dan menjalankan suatu algoritma. Sedangkan, algoritma adalah suatu metode untuk melalukan proses enkripsi dan dekripsi tersebut (Prayudi, 2005).

Istilah-istilah tersebut dijalankan secara sistematis dengan diagram aliran seperti berikut:

Gambar 2.1 Diagram aliran proses enkripsi dan dekripsi

2.2 Citra

Citra merupakan gambar pada bidang dua dimensi (dwimatra) dan merupakan suatu fungsi *continue* dari intensitas cahaya pada bidang dua dimensi tersebut. Citra digolongkan menjadi dua bentuk yaitu citra diam dan citra bergerak. Citra diam merupakan citra tunggal yang tidak bergerak, sedangkan citra bergerak merupakan kumpulan citra diam yang ditampilkan secara beruntun sehingga tampak bergerak oleh pandangan visual (Munir, 2002).

Pengolahan citra dapat dilakukan dengan sistem komputerisasi. Agar dapat mengolah citra dengan sistem komputer digital maka citra yang harus direpresentasikan secara numerik dalam nilai-nilai *diskrit*. Proses representasi tersebut disebut dengan proses *digitalisasi* dan citra hasil dari proses tersebut di istilahkan sebagai citra digital. Citra digital memiliki ukuran yang direpresentasikan NxM (tinggi x lebar atau lebar x panjang).

Citra digital yang merupakan suatu fungsi matematis dapat direpresentasikan sebagai fungsi berikut,

$$f(x,y) = z, \qquad dimana \begin{cases} x, & 0 \le x \le M \\ y, & 0 \le y \le N \\ z, & 0 \le z \le L \end{cases}$$

f(*x*,*y*) merupakan intensitas cahaya pada titik (*x*,*y*). Intensitas cahaya ini sering disebut sebagai derajat keabuan citra (*gray level*). sedangkan *z* adalah nilai derajat keabuannya. Pergeseran nilai derajat keabuan yaitu dari hitam ke putih $[0, L]$. Secara matematis untuk citra digital berukuran NxM juga dapat direpresentasikan dalam bentuk matriks sebagai berikut:

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

banyaknya elemen matriks tersebut berjumlah NxM (*NM* buah), dimana banyaknya intensitas pada satu citra tersebut diistilahkan sebagai *image element*, *picture elemen*, *pixel*, atau *pel* (Dulimarta, 1997).

2.2.1 Citra *Grayscale*

Citra *grayscale* termasuk dalam jenis citra digital dengan intensitas warna yang bergerak antara hitam sampai putih, atau gelap sampai terang. Setiap pixel

dari citra grayscale hanya memiliki satu nilai kanal saja, yang berarti nilai dari RGB nya adalah sama (*Red = Green = Blue*). Pada citra *grayscale*, warna hitam menunjukkan intensitas terlemah dan warna putih menunjukkan intensitas terkuat. Variasi warna diantara hitam dan putih yang sangat banyak menyebabkan warna terlihat abu-abu (*grayscale*).

Gambar 2.2 Citra Lena (*Grayscale*)
(Sumber: cosy.sbg.ac.at)

Derajat keabuan / kedalaman yang dimiliki oleh citra *grayscale* bervarisi, ada citra *grayscale* 4 bit, 8 bit, dsb. tapi pada umumnya yang sering digunakan adalah citra *grayscale* 8 bit. Citra *grayscale* dengan kedalaman 8 bit memiliki derajat keabuan antara 0 – 255. Dimana, 0 menunjukkan warna hitam/gelap dan 255 menunjukkan warna putih/terang. Nilai maksimum 255 didapat dari perhitungan derajat keabuan citra digital, yaitu $2^n - 1$ ($2^8 - 1 = 255$), dengan *n* adalah kedalaman citra digital tersebut.

2.3 ASCII (*American Standart Code for Information Interchange*)

Kode ASCII (*American Standart Code for Information Interchange*) merupakan suatu kumpulan karakter, baik huruf maupun simbol seperti *Hex* dan *Unicode* sesuai dengan standar yang sudah ditentukan. Kode ASCII bersifat universal dan digunakan oleh komputer untuk menunjukkan teks. Kode ASCII masih dibagi lagi menjadi beberapa bagian, seperti ASCII *Control Characters*, ASCII *Printable Character*, dan *The Extended* ASCII *Codes*. Ketiga macam kode ASCII tersebut dapat dilihat pada table-tabel berikut:

Tabel 2.1 Kode ASCII *Control Character* (Character code 0-31)

Dec	***Hex***	***Bin***	***Symbol***	***Dec***	***Hex***	***Bin***	***Symbol***
0	0	0000 0000	NUL	16	10	0001 0000	DLE
1	1	0000 0001	SOH	17	11	0001 0001	DC1
2	2	0000 0010	STX	18	12	0001 0010	DC2
3	3	0000 0011	ETX	19	13	0001 0011	DC3
4	4	0000 0100	EOT	20	14	0001 0100	DC4
5	5	0000 0101	ENQ	21	15	0001 0101	NAK
6	6	0000 0110	ACK	22	16	0001 0110	SYN
7	7	0000 0111	BEL	23	17	0001 0111	ETB
8	8	0000 1000	BS	24	18	0001 1000	CAN
9	9	0000 1001	HT	25	19	0001 1001	EM
10	A	0000 1010	LF	26	1A	0001 1010	SUB
11	B	0000 1011	VT	27	1B	0001 1011	ESC
12	C	0000 1100	FF	28	1C	0001 1100	FS
13	D	0000 1101	CR	29	1D	0001 1101	GS
14	E	0000 1110	SO	30	1E	0001 1110	RS
15	F	0000 1111	SI	31	1F	0001 1111	US

Tabel 2.2 Kode ASCII *Printable Character* (Character code 32-127)

Dec	***Hex***	***Bin***	***Symbol***	***Dec***	***Hex***	***Bin***	***Symbol***
32	20	0010 0000	Spasi	80	50	0101 0000	P
33	21	0010 0001	!	81	51	0101 0001	Q
34	22	0010 0010	”	82	52	0101 0010	R
35	23	0010 0011	#	83	53	0101 0011	S
36	24	0010 0100	$	84	54	0101 0100	T
37	25	0010 0101	%	85	55	0101 0101	U
38	26	0010 0110	&	86	56	0101 0110	V
39	27	0010 0111	’	87	57	0101 0111	W
40	28	0010 1000	(	88	58	0101 1000	X
41	29	0010 1001	)	89	59	0101 1001	Y
42	2A	0010 1010	*	90	5A	0101 1010	Z
43	2B	0010 1011	+	91	5B	0101 1011	[
44	2C	0010 1100	,	92	5C	0101 1100	\
45	2D	0010 1101	-	93	5D	0101 1101	]
46	2E	0010 1110	.	94	5E	0101 1110	^
47	2F	0010 1111	/	95	5F	0101 1111	_
48	30	0011 0000	0	96	60	0110 0000	~
49	31	0011 0001	1	97	61	0110 0001	a
50	32	0011 0010	2	98	62	0110 0010	b
51	33	0011 0011	3	99	63	0110 0011	c
52	34	0011 0100	4	100	64	0110 0100	d
53	35	0011 0101	5	101	65	0110 0101	e
54	36	0011 0110	6	102	66	0110 0110	f

Dec	*Hex*	*Bin*	*Symbol*	*Dec*	*Hex*	*Bin*	*symbol*
55	37	0011 0111	7	103	67	0110 0111	g
56	38	0011 1000	8	104	68	0110 1000	h
57	39	0011 1001	9	105	69	0110 1001	i
58	3A	0011 1010	:	106	6A	0110 1010	j
59	3B	0011 1011	;	107	6B	0110 1011	k
60	3C	0011 1100	<	108	6C	0110 1100	l
61	3D	0011 1101	=	109	6D	0110 1101	m
62	3E	0011 1110	>	110	6E	0110 1110	n
63	3F	0011 1111	?	111	6F	0110 1111	o
64	40	0100 0000	@	112	70	0111 0000	p
65	41	0100 0001	A	113	71	0111 0001	q
66	42	0100 0010	B	114	72	0111 0010	r
67	43	0100 0011	C	115	73	0111 0011	s
68	44	0100 0100	D	116	74	0111 0100	t
69	45	0100 0101	E	117	75	0111 0101	u
70	46	0100 0110	F	118	76	0111 0110	v
71	47	0100 0111	G	119	77	0111 0111	w
72	48	0100 1000	H	120	78	0111 1000	x
73	49	0100 1001	I	121	79	0111 1001	y
74	4A	0100 1010	J	122	7A	0111 1010	z
75	4B	0100 1011	K	123	7B	0111 1011	{
76	4C	0100 1100	L	124	7C	0111 1100	|
77	4D	0100 1101	M	125	7D	0111 1101	}
78	4E	0100 1110	N	126	7E	0111 1110	~
79	4F	0100 1111	O	127	7F	0111 1111	Delete

Tabel 2.3 *The Extended* ASCII *Codes* (Character code 128-255)

Dec	*Hex*	*Bin*	*Symbol*	*Dec*	*Hex*	*Bin*	*Symbol*
128	80	1000 0000	€	192	C0	1100 0000	À
129	81	1000 0001		193	C1	1100 0001	Á
130	82	1000 0010	‚	194	C2	1100 0010	Â
131	83	1000 0011	*ƒ*	195	C3	1100 0011	Ã
132	84	1000 0100	„	196	C4	1100 0100	Ä
133	85	1000 0101	…	197	C5	1100 0101	Å
134	86	1000 0110	†	198	C6	1100 0110	Æ
135	87	1000 0111	‡	199	C7	1100 0111	Ç
136	88	1000 1000	ˆ	200	C8	1100 1000	È
137	89	1000 1001	‰	201	C9	1100 1001	É
138	8A	1000 1010	Š	202	CA	1100 1010	Ê
139	8B	1000 1011	‹	203	CB	1100 1011	Ë
140	8C	1000 1100	Œ	204	CC	1100 1100	Ì
141	8D	1000 1101		205	CD	1100 1101	Í
142	8E	1000 1110	Ž	206	CE	1100 1110	Î
143	8F	1000 1111		207	CF	1100 1111	Ï

Dec	*Hex*	*Bin*	*Symbol*	*Dec*	*Hex*	*Bin*	*Symbol*
144	90	1001 0000		208	D0	1101 0000	Đ
145	91	1001 0001	‘	209	D1	1101 0001	Ñ
146	92	1001 0010	’	210	D2	1101 0010	Ò
147	93	1001 0011	“	211	D3	1101 0011	Ó
148	94	1001 0100	”	212	D4	1101 0100	Ô
149	95	1001 0101	•	213	D5	1101 0101	Õ
150	96	1001 0110	–	214	D6	1101 0110	Ö
151	97	1001 0111	—	215	D7	1101 0111	×
152	98	1001 1000	˜	216	D8	1101 1000	Ø
153	99	1001 1001	™	217	D9	1101 1001	Ù
154	9A	1001 1010	š	218	DA	1101 1010	Ú
155	9B	1001 1011	›	219	DB	1101 1011	Û
156	9C	1001 1100	œ	220	DC	1101 1100	Ü
157	9D	1001 1101		221	DD	1101 1101	Ý
158	9E	1001 1110	ž	222	DE	1101 1110	Þ
159	9F	1001 1111	Ÿ	223	DF	1101 1111	ß
160	A0	1010 0000		224	E0	1110 0000	à
161	A1	1010 0001	¡	225	E1	1110 0001	á
162	A2	1010 0010	¢	226	E2	1110 0010	â
163	A3	1010 0011	£	227	E3	1110 0011	ã
164	A4	1010 0100	¤	228	E4	1110 0100	ä
165	A5	1010 0101	¥	229	E5	1110 0101	å
166	A6	1010 0110	¦	230	E6	1110 0110	æ
167	A7	1010 0111	§	231	E7	1110 0111	ç
168	A8	1010 1000	¨	232	E8	1110 1000	è
169	A9	1010 1001	©	233	E9	1110 1001	é
170	AA	1010 1010	ª	234	EA	1110 1010	ê
171	AB	1010 1011	«	235	EB	1110 1011	ë
172	AC	1010 1100	¬	236	EC	1110 1100	ì
173	AD	1010 1101		237	ED	1110 1101	í
174	AE	1010 1110	®	238	EE	1110 1110	î
175	AF	1010 1111	¯	239	EF	1110 1111	ï
176	B0	1011 0000	°	240	F0	1111 0000	ð
177	B1	1011 0001	±	241	F1	1111 0001	ñ
178	B2	1011 0010	²	242	F2	1111 0010	ò
179	B3	1011 0011	³	243	F3	1111 0011	ó
180	B4	1011 0100	´	244	F4	1111 0100	ô
181	B5	1011 0101	µ	245	F5	1111 0101	õ
182	B6	1011 0110	¶	246	F6	1111 0110	ö
183	B7	1011 0111	·	247	F7	1111 0111	÷
184	B8	1011 1000	¸	248	F8	1111 1000	ø
185	B9	1011 1001	¹	249	F9	1111 1001	ù
186	BA	1011 1010	º	250	FA	1111 1010	ú
187	BB	1011 1011	»	251	FB	1111 1011	û
188	BC	1011 1100	¼	252	FC	1111 1100	ü

Dec	*Hex*	*Bin*	*Symbol*	*Dec*	*Hex*	*Bin*	*Symbol*
189	BD	1011 1101	½	253	FD	1111 1101	ý
190	BE	1011 1110	¾	254	FE	1111 1110	þ
191	BF	1011 1111	¿	255	FF	1111 1111	ÿ

2.4 Sistem Basis Bilangan

Sistem basis bilangan merupakan suatu bilangan yang mewakili besaran dari suatu item fisik yang memiliki basis. Sistem basis bilangan yang sering digunakan adalah bilangan desimal, bilangan biner, bilangan oktal dan bilangan hexadesimal. Bilangan desimal merupakan suatu sistem bilangan yang memiliki 10 basis, yakni 0, 1, 2, 3, 4, 5, 6, 7, 8, dan 9. Bilangan biner merupakan sistem bilangan yang memiliki 2 basis, yakni 0 dan 1. Bilangan oktal merupakan sistem bilangan yang memiliki 8 basis, yakni 0, 1, 2, 3, 4, 5, 6, dan 7. Sedangkan bilangan hexadesimal merupakan sitem bilangan yang memiliki 16 basis, yakni 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, dan F (Feoh, 2011).

Suatu sistem basis bilangan dapat dikonversikan kedalam sistem basis bilangan yang lainya. Beberapa teknik untuk mengkonversi suatu sistem basis bilangan menjadi bentuk sistem basis bilangan yang lain adalah sebagai berikut:

a. Konversi bilangan desimal ke bilangan biner

   Proses konversi bilangan desimal menjadi bentuk bilangan biner yaitu dengan membagi bilangan desimal dengan 2 secara terus menerus sampai tersisa 0. Nilai dari pembagian terakhir dan sisa-sisa dari setiap pembagian tersebut merupakan nilai dari bilangan biner yang diinginkan.

   Contoh:

   $21_{(10)} = 10101_{(2)}$

   21 / 2 = 10 sisa 1

   10 / 2 = 5 sisa 0

   5 / 2 = 2 sisa 1

   2 / 2 = 1 sisa 0

   Maka hasil dari konversi $21_{(10)}$ adalah $10101_{(2)}$

b. Konversi bilangan desimal ke bilangan hexadesimal

Proses konversi bilangan desimal menjadi bentuk bilangan hexadesimal yaitu dengan membagi bilangan desimal dengan 16 secara terus menerus sampai tersisa 0. Nilai dari setiap pembagian dan nilai dari sisa pembagian terakhir tersebut merupakan nilai dari bilangan hexadesimal yang diinginkan.

Contoh:

$173_{(10)} = AD_{(16)}$

173 / 16 = 10 (*A* dalam hexadesimal) sisa 13

13 / 16 = 0 sisa 13 (*D* dalam hexadesimal)

Maka hasil dari konversi $173_{(10)}$ adalah $AD_{(16)}$

c. Konversi bilangan biner ke bilangan desimal

Proses konversi bilangan biner menjadi bentuk bilangan desimal yaitu dengan memisah setiap digit kemudian mengalikan setiap digitnya dengan 2^n. dimana, *n* merupakan posisi dari basis. Seluruh nilai dari proses-proses tersebut kemudian dijumlahkan untuk mendapatkan nilai dari bilangan desimal yang diinginkan.

Contoh:

$10110_{(2)} = 22_{(10)}$

$(1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) = 16 + 0 + 4 + 2 + 0 = 22$

Maka hasil dari konversi $10110_{(2)}$ adalah $22_{(10)}$

d. Konversi bilangan hexadesimal ke bilangan desimal

Proses konversi bilangan hexadesimal menjadi bentuk bilangan desimal yaitu dengan memisah setiap digit lalu merubah digit yang masih berupa A – F menjadi desimal (10 – 15), kemudian mengalikan setiap digitnya dengan 16^n. dimana, *n* merupakan posisi dari basis. Seluruh nilai dari proses-proses tersebut kemudian dijumlahkan untuk mendapatkan nilai dari bilangan desimal yang diinginkan.

Contoh:

$ABC_{(16)} = 2748_{(10)}$

$C \times 16^0 = 12 \times 1 = 12$

$B \times 16^1 = 11 \times 16 = 176$

A x 16^2 = 10 x 256 = 2560

2560 + 176 + 12 = 2748

Maka hasil dari konversi $ABC_{(16)}$ adalah $2748_{(10)}$

2.5 Kriptografi DNA

Kriptografi DNA merupakan ilmu kriptografi yang tergolong baru dan sederhana. Ilmu ini terbentuk dari susunan *deoxyribonuleic acid* (DNA) sebagai algoritmanya. Pada tahun 2015, Song dan Qiao menjelaskan dalam artikelnya mengenai beberapa bentuk dari algoritma DNA. Struktur DNA pada dasarnya terdapat 4 buah basa nitrogen, yaitu *Adenine*, *Cytosine*, *Guanine*, dan *Thymine* (ACGT), dengan A dan T, G dan C merupakan pasangan komplemen. Seperti halnya bilangan biner, yakni 1 dan 0 yang juga merupakan pasangan komplemen, serta 00, 11, 10, 01 juga termasuk pasangan komplemen. Hal tersebut mendasari penelitian tentang pengkodean data biner menjadi bentuk data DNA.

Song dan Qiao (2015), menjelaskan jika basa 00, 11, 10, dan 01 dikodekan menjadi bentuk basa A, C, T, dan G, maka akan mendapatkan 4! = 24 macam bentuk pengkodean yang ditunjukkan dalam tabel berikut:

Tabel 2.4 Bentuk pengkodean DNA
(Sumber: Song dan Qiao, 2015)

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
A	00	00	01	01	10	10	11	11
T	11	11	10	10	01	01	00	00
G	01	10	00	11	00	11	01	10
C	10	01	11	00	11	00	10	01

Tabel 2.4 menunjukkan 24 bentuk pengkodean dari algoritma DNA, yang di bagi menjadi 8 kombinasi pola pengkodean. Bilangan $11001001_{(2)}$ = $201_{(10)}$ jika dikodekan menjadi bentuk DNA dengan kombinasi 8 (kolom 8) adalah ATGC. Hasil dari pengkodean tersebut akan membuat suatu penafsiran yang berbeda saat hendak dikembalikan ke bentuk bilangan biner ketika seseorang salah memakai bentuk kombinasi. Misalnya, dengan menggunakan kombinasi 7 (kolom 7) maka

hasil dari pengkodean ATGC yang seharusnya bernilai $201_{(10)}$ menjadi bernilai $11000110_{(2)} = 198_{(10)}$.

2.6 Algoritma DNA-*Vigenere Cipher*

Vigenere Cipher adalah algoritma klasik yang tergolong dalam metode substitusi abjad-majemuk dan dikembangkan setelah algoritma *Caesar Cipher* (algoritma klasik pertama). Dasar dari algoritma *Vigenere Cipher* menggunakan 26 karakter *English alphabet* (A-Z) sebagai media enkripsi dan dekripsinya. *Vigenere Cipher* membutuhkan *key* yang panjangnya harus sama dengan panjang *plaintext*-nya. Ketika panjang *key* kurang dari panjang *plaintext* maka akan dilakukan pembangkitan kunci baru, yaitu dengan mengulang *key* awal sampai panjangnya sama dengan panjang *plaintext*. Proses enkripsi dan dekripsi dari *Vigenere Cipher* menggunakan pemetaan atau urutan pergeseran yang berbeda dari tabel yang disebut sebagai *Tabula Recta*, *Vigenere Table*, atau *Vigenere Square* (Malhotra, 2012).

Tabel 2.5 *Vigenere Table*
(Sumber: Malhotra, 2012)

	A	***B***	***C***	***D***	***E***	***F***	***G***	***H***	***I***	***J***	***K***	***L***	***M***	***N***	***O***	***P***	***Q***	***R***	***S***	***T***	***U***	***V***	***W***	***X***	***Y***	***Z***
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Tabel 2.5 tersebut menginformasikan hasil karakter dari pemetaan dua karakter. Semua baris pertama dan kolom pertama menyatakan karakter *plaintext* dan karakter kunci (*key*). Sedangkan yang lain nya merupakan karakter *ciphertext* yang dihasilkan.

Contoh:

Plaintext : ABC

Key : DEF

(A,D) → **D**, (B,E) → **F**, (C,F) → **H**

Maka, *ciphertext* yang dihasilkan adalah **DFH**

Kazazi dan Najaftorkaman (2015), memperkenalkan algoritma modifikasi dari *Vigenere Cipher* dan kriptografi DNA. Proses enkripsi dan dekripsi pada modifikasi ini sebagaian besar menggunakan langkah-langkah dari *Vigenere Cipher*. Pembeda dari modifikasi ini yaitu bentuk *plaintext*, *key*, dan *ciphertext* yang ada harus di ubah menjadi bentuk basa nitrogen DNA (ACGT). Bentuk dari tabel *vigenere* pun harus dimodifikasi menjadi bentuk basa nitrogen DNA, sehingga menghasilkan bentuk seperti berikut:

Tabel 2.6 DNA-*Vigenere* Table
(Sumber: Najaftorkaman dan Kazazi, 2015)

	A	**T**	**C**	**G**
A	A	T	C	G
T	T	C	G	A
C	C	G	A	T
G	G	A	T	C

Proses enkripsi dan dekripsi dari metode ini ditunjukkan pada contoh berikut:

a. Proses Enkripsi DNA-*Vigenere Cipher*

Plaintext : ACGC

Key : TCAG

Ciphertext : TAGT

b. Proses Dekripsi DNA-*Vigenere Cipher*

Ciphertext : TAGT

Key : TCAG

Ciphertext : ACGC

Sama halnya dengan *Vigenere cipher*, pada modifikasi ini jika panjang *key* kurang dari panjang plain text maka akan dilakukan pengulangan kunci sampai panjangnya sama dengan panjang *plaintext*.

2.7 Enkripsi-Pembangkit Kunci Bergeser

Metode Enkripsi-Pembangkit Kunci Bergeser adalah suatu metode mengenkripsi kunci dengan tujuan untuk mengamankan kunci dari analisis frekuensi. Metode ini mampu diterapkan untuk memodifikasi kunci yang memiliki panjang lebih pendek daripada panjang *plaintext* agar memiliki panjang yang sama. Proses pembangkitan kunci dari metode ini yaitu dengan menggandakan kunci, menggeser pola kunci, lalu mengenkripsi dengan metode kriptografi yang sesuai. Sasmita (2010), memperkenalkan modifikasi *Vigenere Cipher* dengan Enkripsi-Pembangkit Kunci Bergeser. Metode modifikasi ini melakukan proses enkripsi tambahan pada kunci terlebih dahulu sebelum proses enkripsi terhadap *plaintext*. Kunci awal di enkripsi dengan suatu kunci baru yang merupakan pergeseran dari kunci awal sebanyak *n* karakter menggunakan *vigenere table*. Hasil dari enkripsi tersebut kemudian di gabungkan dengan kunci awal. Proses selanjutnya yaitu mengenkripsi hasil enkripsi sebelumnya dengan proses yang sama dan pergeseran karakter yang sama juga. Langkah tersebut diulangi sampai menghasilkan panjang kunci yang sama dengan panjang *plaintext*.

Contoh:

Plain text = NAMASAYADONY (panjang 12 karakter)

Key = KRIP

Arah geser = Kanan,

Jumlah geser = 1.

a. Kunci 1 = KRIP

   Kunci 2 = PKRI

   Hasil enkripsi kunci (*K-Enkripsi*) = ZBZX

   Hasil penggabungan =KRIPZBZX (panjang 8 karakter)

b. Kunci 1 = ZBZX

Kunci 2 = XZBZ

Hasil enkripsi kunci (K-Enkripsi) = WAAW

Hasil penggabungan =KRIPZBZXWAAW (panjang 12 karakter)

Kunci (*key*) yang baru adalah KRIPZBZXWAAW

2.8 Algoritma Rijndael

Algoritma Rijndael atau pada umumnya sering disebut Algoritma *Advanced Encryption Standard* (AES) merupakan algoritma terobosan baru menggantikan algoritma *Data Encryption Standard* (DES) yang sudah dianggap tidak aman lagi untuk digunakan sebagai media pengamanan suatu pesan. Algoritma Rijndael adalah jenis algoritma yang bersifat simetri dan *cipher block*. Proses pengamanan suatu pesan dilakukan dengan menggunakan kunci yang sama, baik saat proses enkripsi maupun dekripsi. Ukuran kunci pada algoritma ini sebesar 128 bit, 192 bit, dan 256 bit yang dikenal dengan AES-128, AES-192, dan AES 256. Jika ukuran kunci tidak mencapai 128 bit, 192 bit, atau 256 bit maka akan dilakukan penambahan nilai 0 sampai ukuran kunci terpenuhi. Ukuran kunci yang dipilih akan berdampak pada jumlah iterasi yang akan dijalankan saat proses pengamanan pesan. Semakin panjang ukuran kunci maka iterasi juga akan semakin banyak. Panjang kunci maupun panjang *block* dinyatakan dalam bentuk matriks yang disebut dengan *words*, dimana 1 *words* merupakan matriks 4 x 1 yang memiliki 4 *byte* elemen.

Tabel 2.7 Banyaknya iterasi Algoritma Rijndael
(Sumber: Rijmen dan Daemen, 1999)

AES-bit	Panjang *Key* (Nk)	Panjang *Block* (Nb)	Banyak Iterasi (Nr)
128 bit	4 *words*	4 *words*	10
192 bit	6 *words*	4 *words*	12
256 bit	8 *words*	4 *words*	14

Patel dan Padate (2015), menjelaskan tahapan proses enkripsi dan dekripsi algoritma *Rijndael* pada pesan berupa citra dengan menggunakan kunci AES-128 bit. Hal pertama sebelum melakukan proses enkripsi dan dekripsi adalah membangi *plaintext*/citra menjadi partisi matriks 128 bit kemudian

direpresentasikan dalam hexadesimal. Dilanjutkan dengan membangun 10 sub kunci yang akan dipakai pada setiap iterasi. Kunci yang pertama kali di *input* oleh *user* disebut sebagai *cipherkey*. *Cipherkey* dari AES-128 tersebut dibagi menjadi 4 partisi 32 bit dan direpresentasikan ke bentuk hexadesimal dalam matriks 4x4. Representasi ini akan menghasilkan 4 *words* yakni *words*-0 (W_0), *words*-1 (W_1), *words*-2 (W_2), dan *words*-3 (W_3). Selanjutnya akan dibangkitkan 10 sub-*cipherkey* baru dengan menggunakan nilai *cipherkey* sebagai *plaintext*. Proses untuk menghasilkan *words*-4 ($W_{i=4}$) yaitu dengan melakukan tahapan berikut:

a. Operasi *RotWord*

   Operasi ini melakukan perputaran 8 bit pada 32 bit W_{i-1}, dengan menggeser kolom secara siklis ke atas satu kali. *Words* (k_0,k_1,k_2,k_3) menjadi *words* (k_1,k_2,k_3,k_0).

b. Operasi *SubWord*

   Hasil *RotWord* kemudian disubstitusi dengan nilai dari tabel S-*Box*. S-*Box AES Algoritm table* direpresentasikan dalam tabel dberikut:

Tabel 2.8 S-*Box AES Algoritm*
(Sumber: Petel dan Padate, 2015)

		Y															
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**a**	**b**	**c**	**d**	**e**	**f**
x	**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

c. Operasi XOR

Hasil operasi *SubWord* kemudian di XOR dengan nilai konstan R-*Con* yang bersesuaian tiap round. Matriks dan formula dari R-*Con* adalah sebagai berikut:

$$RCon\,[i] = (RC\,[i], 0, 0, 0), \tag{2.1}$$

dimana,

$$RC\,[1] = 01_{16} \tag{2.2}$$

$$RC\,[i] = 2 * RC\,[i - 1] \tag{2.3}$$

dengan sifat operasi * merupakan perkalian yang didefinisakan pada GF(2^8). Pada GF(2^8), setiap elemennya merupakan polinomial dengan order 7 (n – 1). nilai dari maksimal dari RC adalah $255_{(10)}$, dan jika didapatkan nilai RC yang lebih dari $255_{(10)}$ saat proses perkalian, maka hasilnya akan dimodulo dengan polinomial *irreducible*. Polinomial *irreducible* pada algoritma *Rijndael*, yaitu:

$$f(x) = x^8 + x^4 + x^3 + x + 1 \tag{2.4}$$

polinomial tersebut merupakan representasi matematis dari bentuk biner $100011011_{(2)}$. Nilai biner tersebut setara dengan $283_{(10)}$. Sedangkan, nilai R*Con* (dalam hexadesimal) yang didapat yaitu,

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

(Rijmen dan Daemen, 1999)

d. Operasi XOR dengan W_{i-4}

Operasi ini merupakan operasi terakhir untuk mendapatkan W_i. Pada tahap ini, hasil dari proses sebelumnya di XOR dengan W_{i-4}, dan nilai dari operasi ini adalah nilai dari W_i.

Proses selanjutnya, untuk mendapatkan nilai dari W_{i+1} dan seterusnya, yaitu cukup dengan melakukan operasi XOR antara W_i dengan W_{i-3}. Lakukan langkah ini sampai mendapatkan seluruh nilai untuk sub-*cipherkey*.

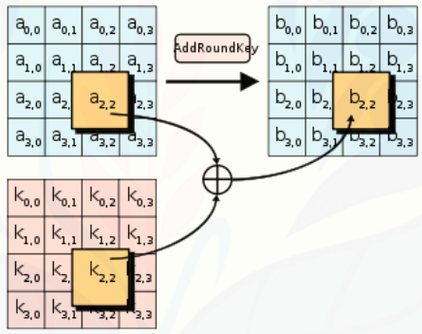
Begitu seluruh sub-*cipherkey* sudah didapatkan, maka akan dilanjutkan dengan proses enkripsi ataupun proses dekripsi:

a. Proses Enkripsi

Proses enkripsi dilakukan dengan proses *AddRoundKey* dan dilanjutkan dengan melakukan empat tahapan, yaitu *SubBytes*, *ShiftRows*, *MixColumns*, dan *AddRoundKey*. Empat tahapan tersebut diulang sampai iterasi terakhir. Hanya saja, sedikit perubahan pada iterasi terakhir yaitu tidak dilakukannya operasi *MixColumns*.

1) Transformasi *AddRoundKey* (proses enkripsi)

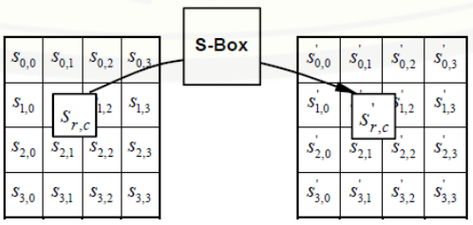
Tahapan ini melakukan proses XOR antara 128 bit partisi plaintext pertama dengan 128 bit *cipherkey* dan sub-*cipherkey* yang sudah dibangkitkan sebelumnya.



Gambar 2.3 Transformasi *AddRoundKey*
(Sumber: slideplayer.com)

2) Transformasi *SubBytes*

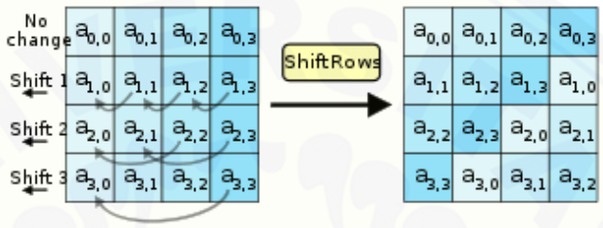
Hasil *AddRoundKey* kemudian disubstitusi menggunakan tabel S-*Box* Algoritma *Rijndael* (Tabel 2.8). Proses ini dilakukan untuk mengaburkan (*confusion*) hubungan antara *plaintext* dan *ciphertext*.



Gambar 2.4 Transformasi *SubBytes*
(Sumber: slideplayer.com)

3) Tranformasi *ShiftRows*

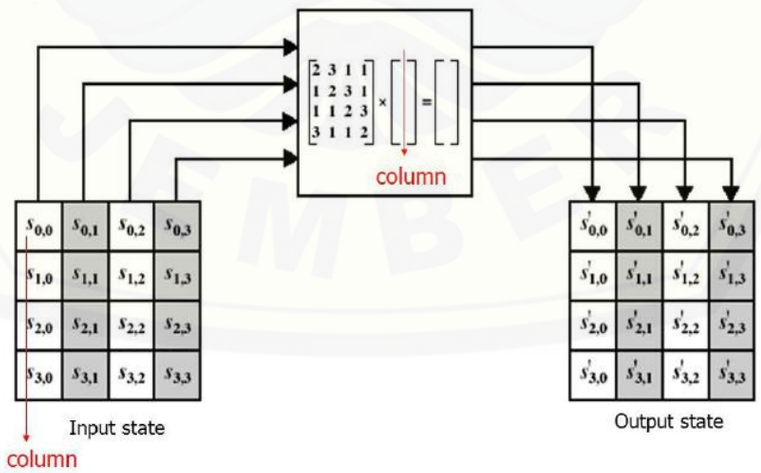
Hasil *SubBytes* kemudian di transformasi menggunakan *ShiftRows*, yaitu memutar elemen matriks pada 3 baris terakhir ke kiri dengan jumlah perputaran yang berbeda. Baris 0 tidak diputar, kemudian baris 1 diputar sebanyak 1 kali, baris 2 diputar 2 kali, dan baris 3 diputar 3 kali. Proses ini dilakukan untuk menyebarkan (*diffusion*) pengaruh setiap bit *plaintext* dan *cipherkey* terhadap *ciphertext* pada arah baris.



Gambar 2.5 Transformasi *ShiftRows*
(Sumber: slideplayer.com)

4) Transformasi *MixColumns*

Hasil *ShiftRows* kemudian di transformasi dengan *MixColumns*, yaitu dengan menggabungkan 4 *byte* dari setiap kolom dengan transformasi linier. Proses ini dilakukan untuk menyebarkan (*diffusion*) pengaruh setiap bit *plaintext* dan *cipherkey* terhadap *ciphertext* pada arah kolom. Bentuk dari operasi *MixColumns* dapat direpresentasikan dalam gambar berikut:



Gambar 2.6 Transformasi *MixColumns*
(Sumber: slideplayer.com)

Gambar 2.6 direpresentasikan dengan formula dibawah ini:

$$\begin{bmatrix} s'_{0,c} = (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus (s_{2,c}) \oplus (s_{3,c}) \\ s'_{1,c} = (s_{0,c}) \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus (s_{3,c}) \\ s'_{2,c} = (s_{0,c}) \oplus (s_{1,c}) \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c}) \\ s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus (s_{1,c}) \oplus (s_{2,c}) \oplus (\{02\} \bullet s_{3,c}) \end{bmatrix} \quad (2.5)$$

Operasi perkalian (•) dan penjumlahan (⊕) merupakan operasi dalam GF(2^8). Operasi penjumlahan pada GF(2^8) tersebut merupakan operasi XOR, atau operasi penjumlahan polinomial biasa dengan setiap koefisiennya dimodulo 2. Sedangkan operasi perkalian pada GF(2^8) tesebut merupakan operasi perkalian polinomial biasa dengan setiap koefisiennya dimodulo 2 dan akan direduksi dengan cara dimodulo dengan polinomial *irreducible* (persamaan 2.4) ketika hasil polinomialnya beroder lebih besar dari 7. Contoh perhitungan untuk mendapatkan $s'_{0,0}$ dengan adalah sebagai berikut:

Misal nilai, $\begin{bmatrix} s_{0,0} \\ s_{1,0} \\ s_{2,0} \\ s_{3,0} \end{bmatrix} = \begin{bmatrix} D8 \\ 33 \\ 64 \\ 86 \end{bmatrix}$, maka *MixColumns* = $\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} D8 \\ 33 \\ 64 \\ 86 \end{bmatrix} = \begin{bmatrix} s'_{0,0} \\ s'_{1,0} \\ s'_{2,0} \\ s'_{3,0} \end{bmatrix}$

$s'_{0,0} = 02 \bullet D8 \oplus 03 \bullet 33 \oplus 01 \bullet 64 \oplus 01 \bullet 86$ (dalam hexadesimal)

$s'_{0,0} = (00000010 \bullet 11011000) \oplus (00000011 \bullet 00110011) \oplus$
$(00000001 \bullet 01100100) \oplus (00000001 \bullet 10000110)$
(konversi dalam biner)

$s'_{0,0} = [(x)(x^7 + x^6 + x^4 + x^3)] \; mod \; (x^8 + x^4 + x^3 + x + 1)$
$+ [(x + 1)(x^5 + x^4 + x + 1)] \; mod \; (x^8 + x^4 + x^3 + x + 1)$
$+ [(1)(x^6 + x^5 + x^2)] \; mod \; (x^8 + x^4 + x^3 + x + 1)$
$+ [(1)(x^7 + x^2 + x)] \; mod \; (x^8 + x^4 + x^3 + x + 1)$
(konversi dalam polinomial)

$s'_{0,0} = [(x^8 + x^7 + x^5 + x^4) \; mod \; 2] \; mod \; (x^8 + x^4 + x^3 + x + 1)$
$+ [(x^6 + 2x^5 + x^4 + x^2 + 2x + 1) \; mod \; 2] \; mod \; (x^8 + x^4 + x^3 + x + 1)$
$+ [(x^6 + x^5 + x^2) \; mod \; 2] \; mod \; (x^8 + x^4 + x^3 + x + 1)$
$+ [(x^7 + x^2 + x) \; mod \; 2] \; mod \; (x^8 + x^4 + x^3 + x + 1)$

Hasil perkalian tersebut di modulo 2, karena termasuk operasi dalam GF(2^8)

$$s'_{0,0} = [(x^8 + x^7 + x^5 + x^4)] \; mod \; (x^8 + x^4 + x^3 + x + 1)$$
$$+ [(x^6 + x^4 + x^2 + 1)] \; mod \; (x^8 + x^4 + x^3 + x + 1)$$
$$+ [(x^6 + x^5 + x^2)] \; mod \; (x^8 + x^4 + x^3 + x + 1)$$
$$+ [(x^7 + x^2 + x)] \; mod \; (x^8 + x^4 + x^3 + x + 1)$$
$$s'_{0,0} = (x^7 + x^5 - x^3 - x - 1) \; mod \; 2 + (x^6 + x^4 + x^2 + 1) \; mod \; 2$$
$$+ (x^6 + x^5 + x^2) \; mod \; 2 + \; (x^7 + x^2 + x) \; mod \; 2$$

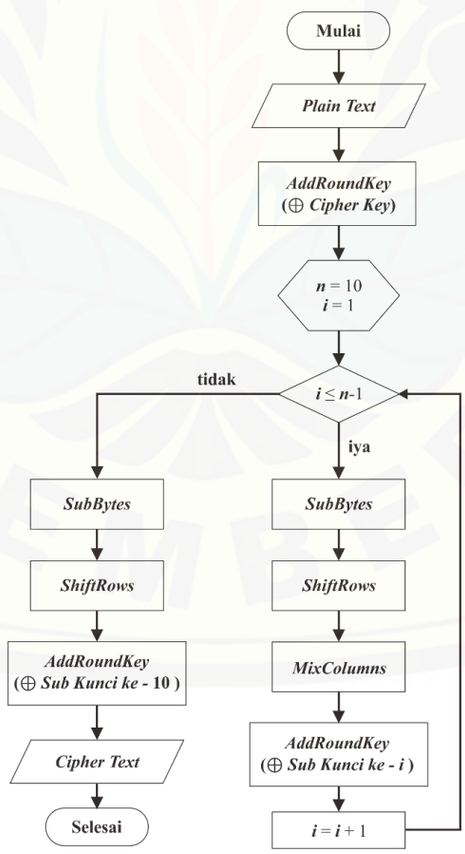
Hasil modulo diatas harus di modulo 2, karena termasuk operasi dalam GF(2^8)

$$s'_{0,0} = (x^7 + x^5 + x^3 + x + 1) + (x^6 + x^4 + x^2 + 1) + \; (x^6 + x^5 + x^2)$$
$$+ (x^7 + x^2 + x)$$
$$s'_{0,0} = (2x^7 + 2x^6 + 2x^5 + x^4 + x^3 + 3x^2 + 2x + 2) \; mod \; 2$$

Hasil penjumlahan diatas di modulo 2, karena termasuk operasi dalam GF(2^8)

$$s'_{0,0} = (x^4 + x^3 + x^2) = 00011100 = \mathbf{1C}$$

Berikut adalah *Flowchart* enkripsi dari Algoritma Rijndael,



Gambar 2.7 *Flowchart* enkripsi Algoritma Rijndael

b. Proses Dekripsi

Proses dekripsi dilakukan dengan proses yang sedikit berbeda daripada proses enkripsi. Pada proses awal yaitu dengan melakukan transformasi *AddRoundKey*, *Inv ShiftRows*, lalu *Inv SubBytes* dan dilanjutkan dengan melakukan empat tahapan, yaitu *AddRoundKey*, *Inv MixColumns*, *Inv ShiftRows*, dan *Inv SubBytes*. Empat tahapan tersebut diulang, dan pada iterasi terakhir hanya melakukan operasi *AddRoundKey* saja.

1) *AddRoundKey* (proses dekripsi)

Tahapan ini melakukan proses XOR antara 128 bit partisi *ciphertext* pertama dengan 128 bit *cipherkey* dan sub-*cipherkey* yang sudah dibangkitkan sebelumnya. Perbedaannya adalah kunci yang digunakan pada iterasi pertama adalah sub-*cipherkey* terakhir (berkebalikan).

2) *Inv SubBytes*

Inv SubBytes menggunakan tabel invers S-Box sebagai proses transformasi. Tabel transformasi *Inv SubBytes* adalah sebagai berikut:

Tabel 2.9 Transformasi *Inv SubBytes* (*Inv* S-*Box AES Algoritm*)
(Sumber: Petel dan Padate, 2015)

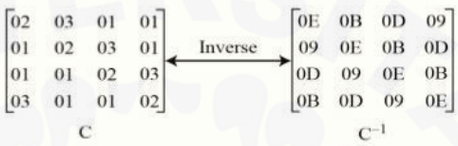
		y															
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**a**	**b**	**c**	**d**	**e**	**f**
x	**0**	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	Ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	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	1c	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

3) *Inv ShiftRows*

Inv ShiftRows memutar elemen matriks pada 3 baris terakhir ke kanan dengan jumlah perputaran yang sama dengan proses *ShiftRows*.

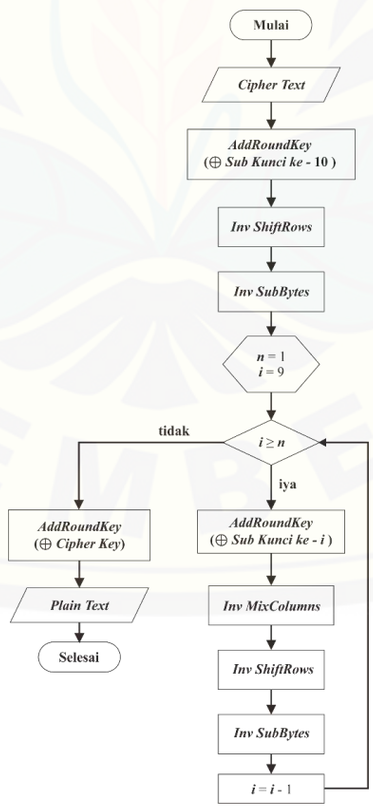
4) *Inv MixColumns*

Hampir sama dengan proses *MixColumns*, hanya saja bentuk transformasi liniernya yang berbeda. Bentuk *Inv MixColumns* direpresentasikan dalam gambar berikut:



Gambar 2.8 Transformasi *Inv MixColumns*
(Sumber: slideplayer.com)

Berikut adalah *Flowchart* dekripsi dari Algoritma Rijndael,



Gambar 2.9 *Flowchart* dekripsi Algoritma Rijndael

2.9 Analisis Keamanan

Proses Analisis kemanan suatu metode atau algoritma enkripsi sangat penting untuk dilakukan untuk mengetahui seberapa aman ketika algoritma enkripsi itu digunakan. Beberapa analisis keamanan suatu algoritma enkripsi adalah sebegai berikut:

a. Analisis Histogram Derajat Keabuan

Analisis histrogram digunakan untuk mengetahui penyebaran nilai pixel pada suatu *cipher image*. Algoritma enkripsi yang baik akan menghasilkan bentuk histogram dengan penyebaran *pixel* yang merata atau memiliki distribusi (relatif) *uniform*. Distribusi yang (relatif) *uniform* pada *cipher image* menunjukkan bahwa algoritma enkripsi memiliki keamanan yang baik.

$$P(x) = \frac{1}{n} \tag{2.6}$$

Persamaan (2.6) digunakan untuk mencari peluang kemunculan dari setiap nilai pixel pada *cipher image*. dengan, P(*x*) merupakan peluang terjadinya *x*, *x* adalah nilai *pixel* pada *cipher image* (*x* = 0 sampai 255), sedangkan *n* adalah frekuensi kemunculan *pixel x*. Nilai setiap P(*x*) yang (relatif) sama menunjukkan suatu *cipher image* tersebut berdistribusi (relatif) *uniform* (Munir, 2012).

b. Analisis Diferensial

Analisis Diferensial digunakan untuk menentukan perbedaan dari dua citra. Langkahnya yaitu dengan menghitung nilai dari *number of pixels change rate* (*NPCR*). NPCR dengan nilai lebih besar dari 90% akan menyulitakan kriptanalisis dalam mencari hubungan statistik antara citra asli dengan citra tersandi (Dharmaadi, dkk., 2013).

$$NPCR = \frac{\sum_{i,j} D(i,j)}{W \times H} \times 100\% \tag{2.7}$$

dengan W dan H merepresentasikan lebar citra dan tinggi citra, dan bentuk dari $D(i,j)$ dapat ditentukan seperti berikut:

$$D(i,j) = \begin{cases} 0, & C(i,j) = C'(i,j) \\ 1, & C(i,j) \neq C'(i,j) \end{cases}$$

dengan $C(i,j)$ dan $C'(i,j)$ merepresentasikan nilai derajat keabuan dari baris *i*, dan kolom *j* dari citra *C* dan *C'* (Atani, dkk., 2013).

c. Analisis Sensitivitas Kunci

Analisis sensitivitas kunci dapat dilakukan dengan dasar dari tahapan berikut,

1) Ketika kunci yang digunakan untuk mengenkripsi citra tersebut sedikit berbeda maka akan menghasilkan *cipher image* yang sangat berbeda.
2) Jika ada perbedaan kunci antara proses enkripsi dan dekripsi maka tidak akan memperoleh *plain image* yang diinginkan.

Song dan Qiao (2015).

d. Analisis Panjang Kunci

Analisis panjang kunci merupakan salah satu poin penting dalam mengetahui keamanan dari algoritma enkripsi yang digunakan. Kelemahan utama dari *Vigenere cipher* adalah pola perulangan dari kuncinya ketika kunci yang dipakai terlalu pendek. Pengujian *Friedman* mengimplementasikan konsep peluang matematika untuk menentukan panjang kunci yang digunakan. Persamaan yang digunakan untuk menganalisa panjang kunci adalah sebagai berikut:

$$Key\ length = \left|\frac{K_p - K_r}{K_0 - K_r}\right| \tag{2.8}$$

dengan,

K_p = Peluang dua *ciphertext* yang dipilih adalah sama (Nilai = 0,067, dalam Bahasa Inggris)

K_r = Peluang terambilnya dua karakter yang sama dari alfabet. Nilainya adalah 1/4 untuk kriptografi DNA (A,C,T,G = 4 karakter)

K_0 = Tingkat kebetulan (*Index of Coincidence*):

$$K_0 = \frac{\sum_{i=1}^{c}(f_i \times (f_i - 1))}{N(N-1)} \tag{2.9}$$

dengan f_i adalah frekuensi kemunculan alfabet, *N* adalah panjang *ciphertext*, dan *c* adalah banyaknya alfabet (Najaftorkaman dan Kazazi, 2015).

BAB 3. METODE PENELITIAN

3.1 Data Penelitian

Data yang digunakan dalam penelitian ini adalah data berupa citra *grayscale* sebagai *plain image*, dan data *text* sebagai kunci dari semua algoritma. Data text tersebut merupakan karakter dari ASCII *Control Characters*, ASCII *Printable Character*, dan *The Extended* ASCII *Codes*. Gambar 3.1 merupakan citra *grayscale* yang digunakan sebagai *plain image*. Dimensi dari citra adalah 256x256

Gambar 3.1 Cameraman.tif
(Sumber : MATLAB *library*)

3.2 Langkah-langkah Penelitian

Langkah-langkah penelitian yang dilakukan dalam penelitian ini adalah sebagai berikut:

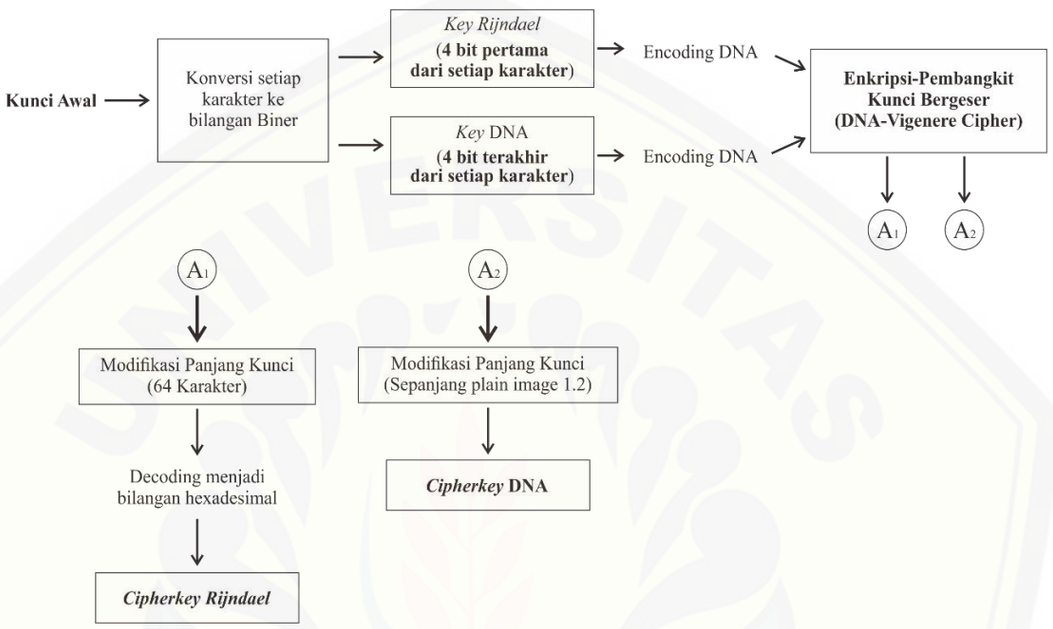
a. Studi Literatur

   Tahap ini dilakukan sebagai pembelajaran mengenai teori-teori yang dipakai sebagai acuan penelitian. Teori yang dipakai dalam penelitian ini adalah teknik kriptografi DNA-*Vigenere Cipher*, Enkripsi-Pembangkit Kunci Bergeser, dan algoritma *Rijndael* untuk mengenkripsi dan mendekripsi pesan berupa citra *grayscale*.

b. Analisa Data

1. Proses Pembangkitan Kunci

Proses pembangkitan kunci dilakukan sesuai dengan alur dari Gambar 3.2,



Gambar 3.2 Proses pembangkitan kunci

a) Konversi kunci awal menjadi bentuk biner

Nilai setiap karakter dari kunci awal akan dikonversi menjadi bentuk biner. Lalu akan diambil 4 bit pertama dari masing-masing karakter sebagai *Key Rijndael*, dan 4 bit terakhir dari masing-masing karakter sebagai *Key* DNA.

b) Encoding kunci baru menjadi bentuk basa nitrogen DNA

Key Rinjdael dan *key* DNA akan diencoding menjadi bentuk basa nitrogen DNA dengan menggunakan Tabel 2.4 dan dengan memakai salah satu kombinasi saja.

c) Enkripsi-Pembangkit Kunci Bergeser untuk *key Rijndael* dengan DNA-*Vigenere Cipher*

Cipherkey Rijndael akan dibangkitkan dengan mengenkripsi *key Rijndael* hasil encoding menggunakan metode enkripsi-pembangkit

kunci bergeser untuk DNA-*Vigenere Cipher* berdasarkan Tabel 2.6. Proses enkripsi terus diulang sampai panjang *cipherkey Rijndael* sama dengan panjang *key Rijndael* yang ditambah dengan 64.

d) Modifikasi panjang *cipherkey Rijndael*

Kunci yang diambil sebagai *cipherkey Rijndael* dimulai dari karakter kunci hasil enkripsi pertama sampai terakhir pada proses enkripsi-pembangkit kunci bergeser pada proses *c*, sehingga akan didapatkan panjang *cipherkey Rijndael* yaitu 64 karakter basa nitrogen DNA.

e) Decoding menjadi bentuk biner kembali

64 karakter basa nitrogen DNA pada *cipherkey Rijndael* akan didecoding menjadi bentuk bilangan desimal berdasarkan Tabel 2.4 dan dengan kombinasi yang sama seperti saat proses encoding.

f) Konversi *cipherkey Rijndael* dari biner ke bilangan hexadesimal

Cipherkey Rijndael yang masih berupa bilangan biner kemudian akan di konversi menjadi bilangan hexadesimal.

g) Enkripsi-Pembangkit Kunci Bergeser untuk *key* DNA dengan DNA-*Vigenere Cipher*

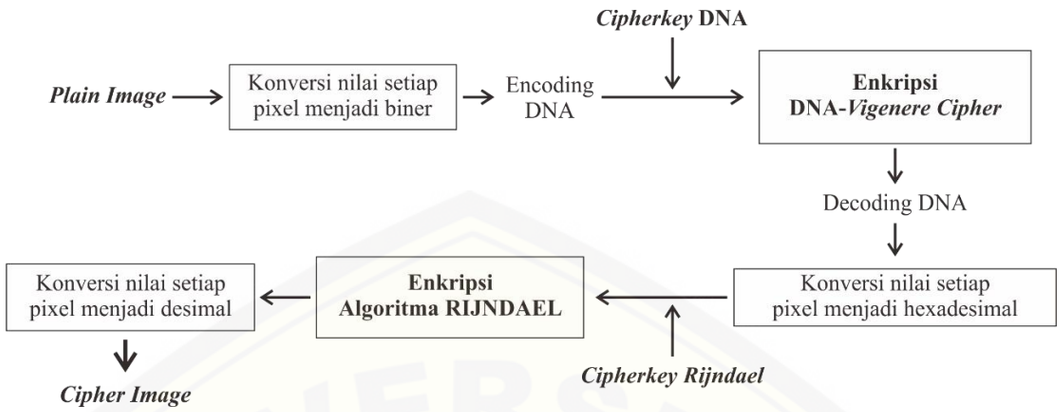
Cipherkey DNA akan dibangkitkan dengan mengenkripsi *key* DNA dengan metode enkripsi-pembangkit kunci bergeser untuk DNA-*Vigenere Cipher* berdasarkan Tabel 2.6. Proses enkripsi terus diulang sampai panjang *cipherkey* DNA sama dengan panjang *plain image* yang ditambah dengan panjang *key* DNA.

h) Modifikasi panjang *cipherkey* DNA

Kunci yang diambil sebagai *cipherkey* DNA dimulai dari karakter kunci hasil enkripsi pertama sampai terakhir pada proses enkripsi-pembangkit kunci bergeser pada proses g, sehingga akan didapatkan panjang *cipherkey* DNA yang sama dengan panjang *plain image*.

2. Proses Enkripsi

Proses enkripsi *plain image* dilakukan sesuai dengan alur dari Gambar 3.3,



Gambar 3.3 Proses enkripsi *plain image*

a) Konversi nilai setiap pixel *plain image* menjadi bentuk biner

   Nilai setiap pixel dari *plain image* akan dikonversi menjadi bentuk biner.

b) Encoding *plain image* menjadi bentuk basa nitrogen DNA

   Nilai setiap pixel dari *plain image* akan dikonversi lagi menjadi bentuk basa nitrogen DNA dengan menggunakan Tabel 2.4 dan dengan memakai salah satu kombinasi saja. Panjang *plain image* pada tahap ini yaitu sebanyak jumlah karakter basa nitrogen DNA yang ada dalam *plain image* tersebut.

c) Enkripsi DNA-*Vigenere Cipher*

   Plain image akan dienkripsi dengan algoritma DNA-*Vigenere Cipher* menggunakan *cipherkey* DNA berdasarkan Tabel 2.6. Hasil proses enkripsi akan mendapatkan suatu *plain image* baru yang berbeda dengan *plain image* sebelumnya.

d) Decoding *plain image* hasil enkripsi menjadi bilangan biner

   Melakukan proses decoding setiap pixel pada *plain image* hasil enkripsi yang masih berupa rangkaian basa nitrogen DNA menjadi bilangan biner terlebih dahulu berdasarkan Tabel 2.4 dan dengan kombinasi yang sama seperti saat proses encoding.

e) Konversi *plain image* hasil decoding menjadi bentuk hexadesimal

Nilai setiap pixel dari *plain image* terenkripsi yang sudah berbentuk biner kemudian akan dikonversi menjadi bentuk hexadesimal.

f) Enkripsi dengan Algoritma Rijndael

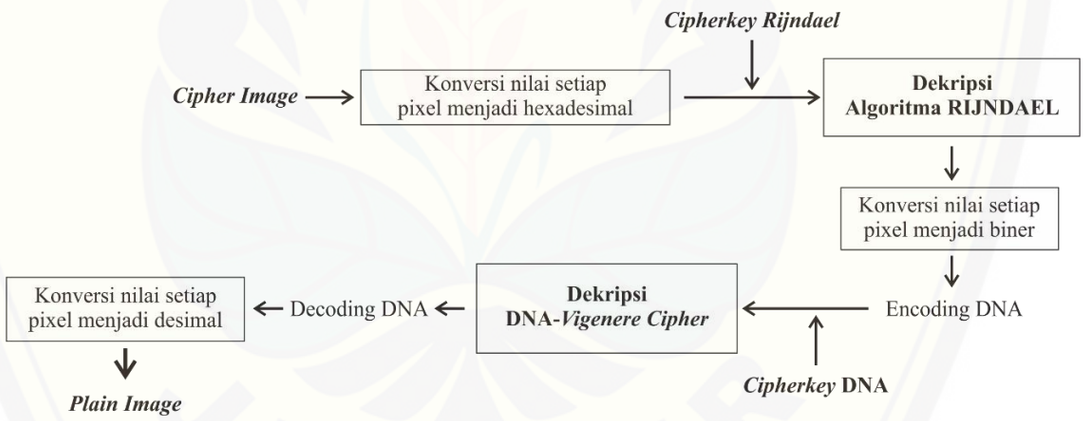
Plain image tersebut akan dienkripsi dengan algoritma *Rijndael* menggunakan *cipherkey Rijndael*. Hasil proses enkripsi akan mendapatkan suatu *cipher image* akhir.

g) Konversi *cipher image* menjadi bentuk desimal

Nilai setiap pixel dari *cipher image* akan dikonversi menjadi bentuk desimal kembali.

3. Proses Dekripsi

Proses dekripsi cipher image akhir dilakukan sesuai dengan alur dari Gambar 3.4,



Gambar 3.4 Proses dekripsi *cipher image*

a) Konversi *cipher image* menjadi bentuk hexadesimal

Nilai setiap pixel dari *cipher image* akan dikonversi menjadi bentuk hexadesimal.

b) Dekripsi dengan Algoritma Rijndael

Cipher image akan di dekripsi dengan algoritma *Rijndael* menggunakan *cipherkey Rijndael*. Hasil proses dekripsi akan mendapatkan *cipher image* baru yang berbeda dengan *cipher image* awal.

c) Konversi *cipher image* baru menjadi bentuk biner

Nilai setiap pixel dari *cipher image* hasil dekripsi akan dikonversi menjadi bentuk biner, sehingga akan didapatkan *cipher image* dengan pixel-pixelnya berbentuk bilangan biner.

d) Encoding *cipher image* menjadi bentuk basa nitrogen DNA

Cipher image akan diencoding lagi menjadi bentuk basa nitrogen DNA dengan menggunakan Tabel 2.4 dan dengan menggunakan kombinasi yang sama saat proses encoding pada tahap enkripsi. Panjang *cipher image* hasil encoding yaitu sebanyak jumlah karakter basa nitrogen DNA dalam *cipher image*.

e) Dekripsi dengan DNA-*Vigenere Cipher*

Cipher image hasil encoding akan di dekripsi dengan algoritma DNA-*Vigenere Cipher* menggunakan *cipherkey* DNA berdasarkan Tabel 2.6. Hasil proses dekripsi akan mendapatkan *plain image*.

f) Decoding *plain image* menjadi bilangan biner

Setiap pixel-pixel pada *plain image* yang masih berupa rangkaian basa nitrogen DNA didecoding kembali berdasarkan Tabel 2.4 dan dengan menggunakan kombinasi yang sama seperti pada saat proses encoding.

g) Konversi *plain image* menjadi bilangan desimal

Setiap pixel-pixel pada *plain image* yang masih berupa bilangan biner di konversi menjadi bilangan desimal. Sehingga akan didapatkan *plain image* awal.

c. Perancangan Program

Perancangan program dilakukan dengan menggunakan *software* Matlab 2015b. Desain program menggunakan aplikasi GUI (*Graphic User Interface*) yang ada dalam Matlab untuk merancang form *layout* dan desain interaksi program agar lebih menarik.

d. Pembuatan Program

Pembuatan program didasarkan pada konsep enkripsi dan dekripsi algoritma *Rinjdael*, konsep enkripsi dan dekripsi DNA-*Vigenere Cipher*, serta konsep Enkripsi-Pembangkit kunci bergeser.

e. Analisis Hasil

Tahap pengujian dan menganalisa program yang telah dibuat agar sesuai dengan konsep teori yang digunakan dan agar sesuai dengan hasil yang diinginkan.

f. Kesimpulan

Mengambil kesimpulan dari penelitian yang telah dilakukan, yaitu dengan menganalisis proses dan hasil enkripsi dan dekripsi suatu *plain*/*cipher image* menggunakan metode yang di ajukan.

Flowchart dari langkah-langkah penelitian adalah sebagai berikut:

Gambar 3.5 *Flowchart* Penelitian

BAB 5. PENUTUP

5.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan dengan berdasar pada data nilai derajat keabuan 16 *pixel* awal pada baris pertama dari data derajat keabuan dalam Lampiran A atau data penelitian cameraman.tif, maka didapatkan beberapa kesimpulan sebagai berikut,

a. Metode Enkripsi-Pembangkit Kunci Bergeser mampu digunakan untuk membangkitkan dua kunci baru dari satu buah kunci *input*. Dua kunci baru tersebut berbeda satu sama lain, dan lebih variatif daripada kunci *input*, dan dapat digunakan sebagai kunci Algoritma DNA-*Vigenere Cipher* dan Algoritma *Rijndael*.
b. Proses enkripsi citra *grayscale* mampu memberikan keamanan sebab citra tersandi (*cipher image*) yang dihasilkan tidak mengandung informasi dari citra awal.
c. Proses dekripsi dapat berjalan dengan baik, dibuktikan dengan *cipher image* yang mampu menjadi *plain image* kembali tanpa ada informasi yang hilang.
d. Analisis keamanan menunjukkan bahwa metode yang digunakan oleh penulis dengan mengimplementasikan Algoritma DNA-*Vigenere Cipher* dan Algoritma *Rijndael* dengan kunci hasil Enkripsi-Pembangkit Kunci Bergeser merupakan metode yang aman dalam penyandian suatu informasi, dibuktikan dengan histogram *cipher image* yang merata atau berdistribusi (relatif) *uniform*, nilai NPCR $\geq$ 90%, memiliki kunci yang variatif, sensitif dan tidak dapat diprediksi panjang kuncinya.

5.2 Saran

Saran yang dapat diberikan untuk penelitian selanjutnya yaitu memodifikasi algoritma DNA-Vigenere Cipher dengan Algoritma Rijndael menjadi satu Algoritma, sehingga tidak perlu melakukan dua kali tahap enkripsi maupun dekripsi.

DAFTAR PUSTAKA

Atani, R. E., R. M. Rad, dan A. Attar. 2013. A new fast and simple image encryption algorithm using scan patterns and xor. *International Journal of Signal Processing, Image Processing and Pattern Recognition*. Vol.6(5): 275-290.

Dharmaadi, I. P. A., A. M. Barmawi., dan G. Bayu. 2013. *Enkripsi Gambar Parsial dengan Kombinasi Metode Stream Cipher RC4 dan Chaotic Function*. Bandung: Institut Teknologi Telkom.

Dulimarta, H. S. 1997. *Diktat Kuliah Pengolahan Citra*. Bandung: Jurusan Teknik Informatika ITB.

Feoh, G. 2011. *Sistem Bilangan Dan Konversi Bilangan*. Bali: Sekolah Tinggi Ilmu Komputer.

Malhotra, V., M. K. I. Rahmani., dan N. Wadhwa. 2012. Alpha-qwerty cipher: an extended vigenère cipher. *Advanced Computing: An International Journal* (*ACIJ*). Vol.3(3).

Munir, R. 2002. *Diktat kuliah Pengolahan Citra, Edisi Kedua*. Bandung: Departemen Teknik Informatika ITB.

Munir, R. 2012. *Analisis Keamanan Algoritma Enkripsi Citra Digital Menggunakan Kombinasi Dua Chaos Map Dan Penerapan Teknik Selektif*. Bandung: Sekolah Teknik Elektro dan Informatika (STEI)

Najaftorkaman M., dan N. S. Kazazi. 2015. A method to encrypt information with dna-based cryptography. *International Journal of Cyber-Security and Digital Forensics* (*IJCSDF*). Vol. 4(3). 417-426.

Patel, A., dan R. Padate. 2015. Image encryption and decryption using aes algorithm. *International Journal of Electronics and Communication Engineering & Technology* (*IJECET*). Vol. 6(1). 23-29.

Prayudi. 2005. Studi Analisis Algoritma Rivest Code 6 (RC6) Dalam Enkripsi/Dekripsi Data. *Seminar Nasional Aplikasi Teknologi Informasi* (*SNATI*) 2005. 15 Juni 2005.

Rijmen, V., dan J. Daemen. 1999. AES Proposal: Rijndael. http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf. [Diakses pada 1 Maret 2017]

Sadikin, R. 2012. *Kriptografi untuk Keamanan Jaringan dan Implementasinya dalam Bahasa Java*. Yogyakarta: Penerbit Andi.

Sasmita, A. B. 2010. *Modifikasi Vigenere Cipher dengan Enkripsi-Pembangkit Kunci Bergeser*. Bandung: PS Teknik Informatika ITB

Song, C., dan Y. Qiao. 2015. A Novel Image Encryption Algorithm Based on DNA Encoding and Spatiotemporal Chaos. www.mdpi.com/journal/entropy. [Diakses pada 1 Maret 2017].

LAMPIRAN

LAMPIRAN A. Matriks derajat keabuan dari Gambar 4.1 (a)

Ukuran matriks : 256x256; nilai pixel dalam desimal

	1	2	3	4	5	6	7	8	9	10	11	12	...	256
1	**156**	**159**	**158**	**155**	**158**	**156**	**159**	**158**	**157**	**158**	**158**	**159**	**…**	152
2	160	154	157	158	157	159	158	158	158	160	155	156	...	153
3	156	159	158	155	158	156	159	158	157	158	158	159	...	152
4	160	154	157	158	157	159	158	158	158	160	155	156	...	153
5	156	153	155	159	159	155	156	155	155	157	155	154	...	151
6	155	155	155	157	156	159	152	158	156	158	152	153	...	153
7	156	153	157	156	153	155	154	155	157	156	155	156	...	149
8	159	159	156	158	156	159	157	161	162	157	157	159	...	147
9	158	155	158	154	156	160	162	155	159	161	156	161	...	147
10	155	154	157	158	160	160	159	160	158	161	160	160	...	153
11	154	157	157	157	156	155	159	154	159	158	161	158	...	155
12	152	150	155	154	152	156	157	156	157	154	157	159	...	148
13	157	153	156	155	157	160	160	157	159	159	160	161	...	151
14	151	154	157	156	156	158	158	156	157	159	158	156	...	148
15	156	157	157	160	159	159	156	158	159	162	161	160	...	150
16	157	158	159	157	157	154	153	158	159	155	160	159	...	156
17	154	154	156	157	158	159	157	160	158	158	156	157	...	152
18	151	153	157	152	156	156	155	156	157	157	155	157	...	152
19	153	155	154	153	156	155	153	155	153	155	154	156	...	155
20	152	154	152	156	159	154	156	155	161	157	157	161	...	151
21	154	157	155	156	157	154	158	158	158	158	158	162	...	155
22	155	153	155	155	159	160	159	161	158	159	160	161	...	150
23	151	151	153	155	153	156	155	155	157	156	157	156	...	156
24	150	151	155	154	155	154	156	152	158	157	158	159	...	157
25	153	154	151	155	154	153	155	157	158	157	157	157	...	153
26	154	154	155	156	155	156	155	156	158	154	159	161	...	155
27	162	157	155	154	156	155	156	157	155	161	157	161	...	151
28	158	155	156	157	160	157	157	162	157	160	158	163	...	154
29	161	157	158	157	159	156	156	157	160	159	162	159	...	153
30	158	159	163	157	158	155	163	159	158	158	162	162	...	156
31	154	154	156	156	159	155	156	159	157	159	159	157	...	148
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮
256	121	126	130	162	102	160	172	133	150	164	147	116	⋯	113

LAMPIRAN B. Encoding DNA potongan derajat keabuan *cameraman.tif*

Kombinasi : kolom ke-1 (00=A, 11=T, 01=G, 10=C)

	1	2	3	4	5	6	7	...	256
1	**CGTA**	**CGTT**	**CGTC**	**CGCT**	**CGTC**	**CGTA**	**CGTT**	**…**	CGCA
2	CCAA	CGCC	CGTG	CGTC	CGTG	CGTT	CGTC	...	CGCG
3	CGTA	CGTT	CGTC	CGCT	CGTC	CGTA	CGTT	...	CGCA
4	CCAA	CGCC	CGTG	CGTC	CGTG	CGTT	CGTC	...	CGCG
5	CGTA	CGCG	CGCT	CGTT	CGTT	CGCT	CGTA	...	CGGT
6	CGCT	CGCT	CGCT	CGTG	CGTA	CGTT	CGCA	...	CGCG
7	CGTA	CGCG	CGTG	CGTA	CGCG	CGCT	CGCC	...	CGGG
8	CGTT	CGTT	CGTA	CGTC	CGTA	CGTT	CGTG	...	CGAT
9	CGTC	CGCT	CGTC	CGCC	CGTA	CCAA	CCAC	...	CGAT
10	CGCT	CGCC	CGTG	CGTC	CCAA	CCAA	CGTT	...	CGCG
11	CGCC	CGTG	CGTG	CGTG	CGTA	CGCT	CGTT	...	CGCT
12	CGCA	CGGC	CGCT	CGCC	CGCA	CGTA	CGTG	...	CGGA
13	CGTG	CGCG	CGTA	CGCT	CGTG	CCAA	CCAA	...	CGGT
14	CGGT	CGCC	CGTG	CGTA	CGTA	CGTC	CGTC	...	CGGA
15	CGTA	CGTG	CGTG	CCAA	CGTT	CGTT	CGTA	...	CGGC
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮
256	GTCG	GTTC	CAAC	CCAC	GCGC	CCAA	CCTA	...	GTAG

LAMPIRAN C. Hasil Pembangkitan kunci *Rijndael* bentuk DNA (*key Rijndael*)

Kunci awal : kripto

Kunci *Rijndael* awal : GCGT GCGT GTGC (Arah pergeseran = Kanan; Jumlah pergeseran = 5)

Key Rijndael

ATAATTTAAATTATATCTCAATCCATTGATGATGGAACACATATAG

AGTCGCGTCTCGTGCACGCCAAACCT

LAMPIRAN D. Hasil Pembangkitan kunci DNA (*key* DNA)

Kunci awal : kripto

Kunci DNA awal : CTAC CGAA GATT (Arah pergeseran = Kanan; Jumlah pergeseran = 5)

Key DNA (*cipherkey* DNA = blok warna kuning)

CAAGGTTATCATCTCGAGTAATTCCTGACTCCGTAGAAAATGGTG
GTTTGGTCGGTGAAACCGTCACAACGGCCTATCAGTAGTTGTGCA
CATTTTTACAGTTTAGTGCATTCCTGACCATAGGCATATAATTTGG
GTTGACACATTGTCCCGTTGCACAACGGCCGTTCAGTAGTCCTAC
ACATTTTTGTGGTTTAGTGTACACCTGACCACATTCATATAATTAG
GGTTGACACTTGGTCCCGTTGATTGACGGCCGTCACATAGTCCTA
TCAGTTTTTGTGCACAAGTGTACAGTTTACCACATTCCTGTAATTA
GGCATAACACTTGGGTTGGTTGATTGTCCCCCGTCACAACGGCCT
ATCAGTAGTTGTGCACATTTTTACAGTTTAGTGCATTCCTGACCAT
AGGCATATAATTTGGGTTGACACATTGTCCCGTTGCACAACGGCC
GTTCAGTAGTCCTACACATTTTTGTGGTTTAGTGTACACCTGACCA
CATTCATATAATTAGGGTTGACACTTGGTCCCGTTGATTGACGGCC
GTCACATAGTCCTATCAGTTTTTGTGCACAAGTGTACAGTTTACCA
CATTCCTGTAATTAGGCATAACACTTGGGTTGGTTGATTGTCCCCC
GTCACAACGGCCTATCAGTAGTTGTGCACATTTTTACAGTTTAGTG
CATTCCTGACCATAGGCATATAATTTGGGTTGACACATTGTCCCGT
TGCACAACGGCCGTTCAGTAGTCCTACACATTTTTGTGGTTTAGTG
TACACCTGACCACATTCATATAATTAGGGTTGACACTTGGTCCCGT
TGATTGACGGCCGTCACATAGTCCTATCAGTTTTTGTGCACAAGTG
TACAGTTTACCACATTCCTGTAATTAGGCATAACACTTGGGTTGGT
TGATTGTCCCCCGTCACAACGGCCTATCAGTAGTTGTGCACATTTT
TACAGTTTAGTGCATTCCTGACCATAGGCATATAATTTGGGTTGAC
ACATTGTCCCGTTGCACAACGGCCGTTCAGTAGTCCTACACATTTT
TGTGGTTTAGTGTACACCTGACCACATTCATATAATTAGGGTTGAC
ACTTGGTCCCGTTGATTGACGGCCGTCACATAGTCCTATCAGTTTT
TGTGCACAAGTGTACAGTTTACCACATTCCTGTAATTAGGCATAAC
ACTTGGGTTGGTTGATTGTCCCCCGTCACAACGGCCTATCAGTAG
TTGTGCACATTTTTACAGTTTAGTGCATTCCTGACCATAGGCATAT
AATTTGGGTTGACACATTGTCCCGTTGCACAACGGCCGTTCAGTA
GTCCTACACATTTTTGTGGTTTAGTGTACACCTGACCACATTCATA
TAATTAGGGTTGACACTTGGTCCCGTTGATTGACGGCCGTCACAT
AGTCCTATCAGTTTTTGTGCACAAGTGTACAGTTTACCACATTCCT
GTAATTAGGCATAACACTTGGGTTGGTTGATTGTCCCCCGTCACAA
CGGCCTATCAGTAGTTGTGCACATTTTTACAGTTTAGTGCATTCCT
GACCATAGGCATATAATTTGGGTTGACACATTGTCCCGTTGCACAA
CGGCCGTTCA……GTGCACAAGTGTACAGTTTACCACATTCCTGTA
ATTAGGCATAACACTTGGGTTGGTTGATTGTCCCCCGTCACAACGG

LAMPIRAN E. Decoding *key Rijndael*

Kombinasi : kolom ke-1 (00=A, 11=T, 01=G, 10=C)

Key Rijndael (DNA) :

ATAA TTTA AATT ATAT CTCA ATCC ATTG ATGA TGGA ACAC ATAT AGAG TCGC GTCT CGTG CACG

Key Rijndael (Biner) :

00110000 11111100 00001111 00110011 10111000 00111010 00111101 00110100 11010100 00100010 00110011 00010001 11100110 01111011 10011101 10001001

Key Rijndael (Hexadesimal) :

30 FC 0F 33 B8 3A 3D 34 D4 22 33 11 E6 7B 9D 89 (*Cipherkey* Rijndael)

Cipherkey Rijndael (Matriks) :

$$\begin{bmatrix} 30 & B8 & D4 & E6 \\ FC & 3A & 22 & 7B \\ 0F & 3D & 33 & 9D \\ 33 & 34 & 11 & 89 \end{bmatrix}$$

LAMPIRAN F. Hasil enkripsi plain image dengan *cipherkey* DNA

Cipher image DNA

AGTGTACTGTTGAAAACCCCCACCAAATAAGATATCCGTCGCAGTCCC
GTGTATGTTCAAATAGACCTCAGCAATCGAAGGCCTGTTGACTGGGTC
GCCGTGTCCTTCACATAAACCACCGCGCACAGGCGAGGCGTGATCAGT
CGAGGATATGAAACTGCACCAACTGACTGCCGAAATACTTGGGGGTG
GTGGACTATGCAGAAAACAACACGAACGGGCCCGCTCTGGACACTGG
TAGAAATGGCCGCCCAACAAATACATGCAAAGTAGATTGCTTGAAGAT
CTCATTGTCCTCAACGTGATAAAGTCGTGTGTGTATTTCGGCGCCGTCA
GTCTTCCGGGGTTAGCGATTTCGCTAGATGGGCGTCGGAAAATTCGCG
AGTAATCAACAACATACAGGGCGTTGTTGATATGTCGGCTTTCGACGC
CCCAAGGATTCGCATAGCGCTAGCCGGCTGTCGAGGAATATGCGAGAC
GTCGACAGCGTACAGGGCGAAATAGATGCGTCGGTTCTCACCGCAGCC
TGGTATCGCCCAACGCGAGCAATTCGTCAAGGGGGGTGCGGGAATATT
GCAAAGTTCTCAACGTGATAAAGTAGTGAGTGCATAACGGTGCGTTCT
TTCAGCCTAGGTCAGGAATCTCGCGAGTCGGAGGTGGGTCTATCGGCT
GGCGGTCTACATCATTTAACGCAGAGCAGATTAGCTGGCGCTGCCCAT
TCGGCGAGCTGGCACATCACAAACCGAGTGCCCAAGAACAAGGGTGT
GGTGGACTATGCAGAAACCATCACAGACACGCGTGCCTTGGCCAGAG
GTGGATATGACCGGGCACAAATTACTGGCCGAAACGAGGGGTCGTAG
AGGTCTAGGCTATACGCACCACTGAATTGCAAGGAAACTGCAATGAA
ATACATGTGCAGAGTAGATTGAGATCTTCATCAGTCAGGAGGCCGAGA
TGAGCGGGATAGGCCAAAGCTATGACTAAGGTAAGGTGGCCGATCTA
CGCTCACAGCGTTGTAGGAGAACAGGATTGGTTCGGAGATCGAGGTG
ACGGCCTTACGCTTGGGATATCGCTGTAGCGACTAGGTAGCCCTACAC
AATGAATGACATGTTCGTCCAGAATACACCCGCCTAATTGAGGGGAAG
TGCACTACTTAGCTGTGTACACTACAAAAGGGCGCGCCCACAACAAGG
GTCTGGCTGGCCGGTGAAAAATAACACCATTAAGCGATTGCTGGTGGA
CATGGGCGCAGTGGTCTGTACGAAAGTCAAGGCACACGAGCTTGGATT
GCACATGCCTGGGCCTCTAATCGCGGTAGGAGGACGTCAAGAAATTAG
CTGGATTTCAACAACGTTGAGAGCGTTATAAATAGGTGGGTCGTCGGC
GGAGCCCGGCGTCAGCCATCGCTAGCCATTGGTGCAGAGGGACGCGT
GAATATAACACGGTAGTCGACGAAATAAAGGGGTGGGTTCATGACGG
AGCCTTCACTCGGCCGCGGACAGTAATAGCGTGAGGTGGGAGTTCGAC
GATAGGCGAGTTATCGGCAACATACAGGTCGAAGTCCATGGGTCTGCC
ATCAACGCCCCGCGGTGTCGGATTTCGCGAGTAGGGAGTTTAGTTATT
GGCAAGTCATCAACATTGTTAAGTCCGTAATAAATAGGTAGGTCCTGC
ACGACGCGTGGTCTGCTCCTCCAAAAGGTACGTGCACAACCGATAGGC
CGATTACGGCTCTGCGGTGGGCAATACGGAAGCGCCCGCTAACGTCAC
GGGTATGGTTGGACGGCGAATAATTACAACATGAAATGACGGCTTGTG
GACTTGGGTGCATTGGGCTACACCTAAGTCATCGCTCACGGGCCTGGT
……TATTGAATGATGCCGCACGTACGCTGAAGGCCGCGATTTATAGGC
ATCCCTTCCATCCACCCTCGAGAGGGTTTCCCTCCACCTACGCCATGGA

LAMPIRAN G. Matriks derajat keabuan dari *cipher image* DNA

Ukuran matriks : 256x256; nilai pixel dalam hexadesimal

	1	2	3	4	5	6	7	8	9	10	11	12	⋯	256
1	**1D**	CB	7D	00	AA	8A	03	04	CE	9E	61	EA	**…**	10
2	**85**	3D	7E	51	39	17	49	6B	C9	BD	53	39	...	5E
3	**18**	BD	43	C4	CB	89	75	31	15	B3	3A	7C	...	65
4	**DF**	A7	55	3B	B5	1A	1A	5D	53	DD	B9	41	...	3C
5	**14**	BE	37	7C	52	64	14	46	53	C8	A4	15	...	56
6	**5C**	C4	B9	42	76	C8	85	B3	33	16	5F	35	...	44
7	**54**	C4	AA	1B	36	B3	5F	17	5C	41	C4	BE	...	BA
8	**32**	17	53	3A	7D	4A	6A	29	BF	55	C8	B7	...	1F
9	**5D**	4C	CB	B8	89	4A	DF	13	B0	04	7D	64	...	1B
10	**A4**	52	C8	B6	2F	0D	5E	65	1A	83	55	DD	...	11
11	**BF**	30	7E	50	38	13	4A	67	CA	AB	69	3B	...	5C
12	**14**	B4	4C	C7	C5	89	77	33	15	BF	39	7C	...	61
13	**C8**	A6	54	34	B5	2F	2C	5C	50	C8	82	56	...	3A
14	**1F**	BF	39	7D	53	6B	16	4B	55	CA	AB	1B	...	57
15	**51**	CA	BF	55	75	C8	89	B3	32	2A	64	0C	...	41
16	**55**	C9	A8	18	3A	B2	5E	1A	5E	4C	DD	BD	...	B1
17	3D	12	53	39	7F	4A	6A	28	AB	56	CB	B5	...	14
18	56	4E	CA	BA	89	76	C4	14	B2	30	78	50	...	10
19	A6	53	C5	B1	1B	34	58	5C	15	B9	4F	C9	...	13
20	BD	3D	79	53	3B	12	4B	64	DC	AA	55	0E	...	58
21	16	BF	4C	C9	CA	87	74	31	16	B3	3A	43	...	64
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮
256	A4	4B	4A	0F	4E	2F	28	44	5B	D1	B1	29	⋯	D4

LAMPIRAN H. Hasil pembangkitan 10-sub kunci dari *cipherkey* Rijndael

Cipherkey Rijndael			
30	**B8**	**D4**	**E6**
FC	**3A**	**22**	**7B**
0F	**3D**	**33**	**9D**
33	**34**	**11**	**89**

Sub-kunci 1			
10	A8	7C	9A
A2	98	BA	C1
A8	95	A6	3B
BD	89	98	11

Sub-kunci 2			
6A	C2	BE	24
40	D8	62	A3
2A	BF	19	22
05	8C	14	05

Sub-kunci 3			
64	A6	18	3C
D3	0B	69	CA
41	FE	E7	C5
33	BF	AB	AE

Sub-kunci 4			
18	BE	A6	9A
75	7E	17	DD
A5	5B	BC	79
D8	67	CC	62

Sub-kunci 5			
C9	77	D1	4B
C3	BD	AA	77
F	54	E8	91
60	07	CB	A9

Sub-kunci 6			
1C	6B	BA	F1
42	FF	55	22
DC	88	60	F1
D3	D4	1F	B6

Sub-kunci 7			
CF	A4	1E	EF
E3	1C	49	6B
92	1A	7A	8B
72	A6	B9	0F

Sub-kunci 8			
30	94	8A	65
DE	C2	8B	E0
E4	FE	84	0F
AD	0B	B2	BD

Sub-kunci 9			
CA	5E	D4	B1
A8	6A	E1	01
9E	60	E4	EB
E0	EB	59	E4

Sub-kunci 10			
80	DE	0A	BB
41	2B	CA	CB
F7	97	73	98
28	C3	9A	7E

LAMPIRAN I. Matriks derajat keabuan dari *cipher image* Rijndael

Ukuran matriks : 256x256; nilai pixel dalam desimal

	1	2	3	4	5	6	7	8	9	10	11	12	...	256
1	**246**	128	76	27	165	246	191	248	28	187	179	129	…	248
2	**180**	110	197	66	175	154	103	231	123	45	132	131	...	13
3	**180**	98	124	108	15	82	16	19	138	90	179	26	...	225
4	**234**	246	59	186	18	84	26	79	57	41	52	44	...	78
5	**97**	227	26	169	51	168	251	240	88	222	253	43	...	60
6	**62**	142	81	236	224	171	53	51	221	229	124	208	...	50
7	**48**	214	95	208	129	101	241	55	211	186	7	65	...	228
8	**231**	95	158	80	250	58	181	248	251	24	110	170	...	112
9	**193**	158	145	15	10	94	4	123	95	13	194	182	...	98
10	**196**	249	80	13	214	66	51	226	41	225	125	149	...	231
11	**24**	184	28	155	167	40	178	3	186	10	55	242	...	106
12	**184**	96	203	128	99	120	252	180	106	246	170	10	...	25
13	**61**	129	76	49	190	1	129	197	200	211	202	197	...	179
14	**246**	167	21	149	247	108	22	190	48	219	241	140	...	125
15	**209**	28	245	151	97	80	125	126	74	58	199	150	...	223
16	**86**	193	209	9	141	47	22	230	16	236	128	209	...	134
17	145	240	231	109	25	119	242	71	156	208	100	66	...	11
18	246	89	168	164	191	159	255	222	90	34	99	200	...	26
19	253	110	244	219	23	150	162	71	220	182	196	191	...	17
20	95	70	142	139	53	25	232	10	116	237	231	127	...	163
21	244	238	251	18	208	181	21	121	143	31	70	228	...	84
22	129	49	103	7	234	144	42	94	35	208	255	173	...	50
23	76	227	219	251	179	134	40	59	78	212	15	165	...	25
24	179	41	41	231	41	139	184	117	27	13	43	106	...	80
25	162	169	209	178	207	25	104	150	225	182	3	67	...	221
26	200	43	168	56	214	165	20	191	14	166	191	235	...	126
27	225	150	114	78	207	117	215	204	28	118	54	251	...	0
28	137	159	255	98	181	46	78	85	226	99	136	121	...	144
29	71	223	5	240	146	170	203	186	179	17	217	188	...	57
30	251	133	97	168	228	18	51	95	98	75	138	74	...	228
31	193	177	119	154	150	51	119	10	180	157	68	128	...	27
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮
256	185	92	221	49	229	81	48	34	68	90	220	132	...	243

LAMPIRAN J. Nilai *P*(x) untuk nilai *x* pada *cipher image* Gambar 4.1(b)

x	P(x)	x	P(x)	x	P(x)	x	P(x)	x	P(x)	x	P(x)	x	P(x)	x	P(x)
0	0.004	**32**	0.004	**64**	0.004	**96**	0.004	**128**	0.004	**160**	0.004	**192**	0.004	**224**	0.004
1	0.004	**33**	0.004	**65**	0.004	**97**	0.004	**129**	0.004	**161**	0.004	**193**	0.003	**225**	0.004
2	0.004	**34**	0.004	**66**	0.004	**98**	0.004	**130**	0.004	**162**	0.004	**194**	0.004	**226**	0.004
3	0.004	**35**	0.004	**67**	0.004	**99**	0.004	**131**	0.004	**163**	0.004	**195**	0.004	**227**	0.004
4	0.004	**36**	0.004	**68**	0.004	**100**	0.004	**132**	0.004	**164**	0.004	**196**	0.004	**228**	0.003
5	0.004	**37**	0.004	**69**	0.004	**101**	0.004	**133**	0.004	**165**	0.004	**197**	0.004	**229**	0.004
6	0.004	**38**	0.004	**70**	0.004	**102**	0.004	**134**	0.004	**166**	0.004	**198**	0.004	**230**	0.004
7	0.004	**39**	0.004	**71**	0.004	**103**	0.004	**135**	0.004	**167**	0.004	**199**	0.004	**231**	0.004
8	0.004	**40**	0.004	**72**	0.004	**104**	0.004	**136**	0.004	**168**	0.004	**200**	0.004	**232**	0.004
9	0.004	**41**	0.004	**73**	0.004	**105**	0.004	**137**	0.004	**169**	0.004	**201**	0.004	**233**	0.004
10	0.004	**42**	0.004	**74**	0.004	**106**	0.004	**138**	0.004	**170**	0.004	**202**	0.004	**234**	0.004
11	0.004	**43**	0.004	**75**	0.004	**107**	0.004	**139**	0.004	**171**	0.004	**203**	0.003	**235**	0.004
12	0.004	**44**	0.004	**76**	0.004	**108**	0.004	**140**	0.004	**172**	0.004	**204**	0.004	**236**	0.004
13	0.004	**45**	0.004	**77**	0.004	**109**	0.004	**141**	0.004	**173**	0.004	**205**	0.004	**237**	0.004
14	0.004	**46**	0.005	**78**	0.003	**110**	0.004	**142**	0.004	**174**	0.004	**206**	0.004	**238**	0.004
15	0.004	**47**	0.004	**79**	0.004	**111**	0.004	**143**	0.005	**175**	0.004	**207**	0.004	**239**	0.004
16	0.004	**48**	0.004	**80**	0.004	**112**	0.004	**144**	0.004	**176**	0.004	**208**	0.004	**240**	0.004
17	0.004	**49**	0.004	**81**	0.004	**113**	0.004	**145**	0.004	**177**	0.004	**209**	0.004	**241**	0.004
18	0.004	**50**	0.004	**82**	0.003	**114**	0.004	**146**	0.004	**178**	0.004	**210**	0.004	**242**	0.004
19	0.004	**51**	0.004	**83**	0.004	**115**	0.004	**147**	0.003	**179**	0.003	**211**	0.004	**243**	0.004
20	0.005	**52**	0.004	**84**	0.004	**116**	0.004	**148**	0.004	**180**	0.004	**212**	0.004	**244**	0.004
21	0.004	**53**	0.004	**85**	0.004	**117**	0.004	**149**	0.004	**181**	0.004	**213**	0.004	**245**	0.004
22	0.005	**54**	0.004	**86**	0.004	**118**	0.004	**150**	0.004	**182**	0.004	**214**	0.004	**246**	0.004
23	0.004	**55**	0.004	**87**	0.004	**119**	0.004	**151**	0.004	**183**	0.004	**215**	0.004	**247**	0.004
24	0.004	**56**	0.004	**88**	0.004	**120**	0.004	**152**	0.004	**184**	0.004	**216**	0.004	**248**	0.004
25	0.004	**57**	0.004	**89**	0.004	**121**	0.004	**153**	0.004	**185**	0.004	**217**	0.004	**249**	0.004
26	0.003	**58**	0.004	**90**	0.003	**122**	0.004	**154**	0.004	**186**	0.004	**218**	0.004	**250**	0.004
27	0.004	**59**	0.004	**91**	0.004	**123**	0.004	**155**	0.004	**187**	0.003	**219**	0.004	**251**	0.004
28	0.004	**60**	0.004	**92**	0.004	**124**	0.004	**156**	0.004	**188**	0.004	**220**	0.004	**252**	0.004
29	0.004	**61**	0.004	**93**	0.004	**125**	0.004	**157**	0.004	**189**	0.004	**221**	0.004	**253**	0.004
30	0.004	**62**	0.004	**94**	0.004	**126**	0.004	**158**	0.004	**190**	0.004	**222**	0.004	**254**	0.004
31	0.004	**63**	0.004	**95**	0.004	**127**	0.004	**159**	0.004	**191**	0.004	**223**	0.004	**255**	0.004

LAMPIRAN K. Skrip Program Distribusi (relatif) *Uniform*

distribusiuniform.m

```
a=imread(''encryption_cameraman.tif '); %citra / cipher image
jumlah=imhist(a); %jumlah kemunculan setiap nilai pixel
b=length(jumlah);

for i=1:b
    pixel(i)=i-1; %nilai pixel 0-255
    nilai(i)=1/jumlah(i); %peluang kemunculan setiap pixel
    bulat(i)=round(nilai(i),3); %pembulatan nilai peluang
end

c=unique(bulat); %nilai-nilai yang muncul
d=histc(bulat,c); %frekuensi dari nilai-nilai yang muncul

for j=1:length(d)
    e(j)=(d(j)/256)*100; %presentase peluang kemunculan
end
```

LAMPIRAN L. Skrip Program Analisis Diferensial

analisisdiferensial.m

```
a=imread('cameraman 256x256.tif');        %Plain Image
b=imread('encryption_cameraman.tif');     %Cipher Image DNAES
[c d]=size(a);

for j=1:c
        for k=1:d
            if a(j,k)==b(j,k);
                beda(j,k)=0;
            else
                beda(j,k)=1;
            end
        end
end

hasil=(c*d)-sum(sum(sum(beda)));
imwrite(mat2gray(beda,[0 1]),'AD.tif');
npcr=(sum(sum(sum(beda)))/(c*d))*100;

%NB = ganti nilai 'b=imread('... .tif');' dengan cipher image lain
```