ISSN : 1410 - 7104

Penggunaan Algorithma *AES-RIJNDAEL* Pada Sistem Enkripsi Dan Dekripsi Untuk Komunikasi Data

Implementasi of AES-Rijndael in Encrytion and Decryption System for Data Communication

M. Amarullah A.* dan Andi Suprianto**
*Maintenance Enggineer PT. Pamapersada Nusantara, Jakarta
** Dosen Program Studi Teknik Informatika,
Institut Sains dan Teknologi Nasional

Abstrak---*Data merupakan salah satu aset penting sehingga proses penyimpanan dan pendistribusian data memerlukan berbagai macam pertimbangan, terutama dalam aspek kerahasiaan dan keamanannya. Dalam penyimpanan atau pendistribusian data perlu dilakukan penyandian atau pengkodean untuk menjamin kerahasian informasi didalamnya. Ilmu kriptografi adalah ilmu yang mempelajari tentang teknik penyandian informasi yang didalamnya terdiri dari enkripsi yaitu penyandian dan dekripsi yaitu pengembalian hasil penyandian. AES Rijndael merupakan algoritma kriptografi Symmetric Key yang didesain untuk beroperasi pada blok simetris 128 bit. Rijndael menggunakan subtitusi, permutasi dan sejumlah putaran berulang. Setiap putaran menggunakan kunci internal yang berbeda. Empat proses utama pada algoritma yaitu: SubBytes, ShiftRows, MixColumn dan AddRoundKey. Hasil dari simulasi menunjukan bahwa informasi/data yang dienkripsi diubah menjadi bentuk informasi yang tidak bisa langsung dibaca. Proses enkripsinya menggunakan kunci 128 bit yang diinputkan. Untuk mengembalikan hasil penyandian kunci yang sama harus digunakan untuk mendekripsi. Penggunaan kunci yang salah tidak akan mengembalikan ke informasi semula. Proses enkripsi dekripsi juga telah dicoba untuk beberapa jenis file komputer seperti notepad, word document, sound dan video.*

Kata kunci---*Kriptografi, Rijndael , Enkripsi, Dekripsi*

Abstract---*Data is one important asset that the storage and distribution of data require different kinds of considerations, especially in the aspect of confidentiality and security. In a storage or distribution of data encryption or coding needs to be done to ensure the confidentiality of the information therein. Science of cryptography is the study of the encoding technique in which information consists of the encryption is an encryption and decryption of the return of the encoding. AES Rijndael is a symmetric key cryptographic algorithms are designed to operate on 128-bit symmetric block. Rijndael uses substitution, permutation, and a number of rounds over and over. Each round uses a different internal lock. Four main processes in the algorithm are: SubBytes, ShiftRows, MixColumn and AddRoundKey. The results of simulations show that the information / data that is encrypted is converted into a form of information that can not be directly read. The process uses a 128 bit encryption is entered. To restore the encryption key should be used to decrypt. Use of the wrong key will not return to the original information. Encryption decryption process has also been tested for several types of computer files such as notepad, word document, sound and video.*

Keywords---*Cryptographic, Rijndael , Encrypt, Dscrypt*

1. PENDAHULUAN

Data merupakan salah satu aset penting dalam kelangsungan berorganisasi kelompok manapun (perusahaan, pemerintahan, lembaga sosial dll). Proses penyimpanan dan pendistribusian data memerlukan berbagai macam pertimbangan, terutama dalam aspek kerahasiaan dan keamanannya.

Banyak kasus yang terjadi sekarang ini, data-data penting pemerintahan atau perusahaan bocor kepada pihak yang tidak berwenang. Cara penyimpanan data di media storage seperti harddisk, flashdisk, compact disk dan penyimpanan online di internet rentan terhadap pencurian jika tidak hati-hati dalam proses penyimpanan. Data yang disimpan di media Storage seperti hardisk, flasdisk dengan mudah dapat berpindah tangan ketika kecerobohan pembawa media. begitu juga dengan penyimpanan media online yang saat ini dapat dengan mudah dibobol oleh para hacker.

Dalam menjaga keamanan data terdapat sebuah metode pengamanan data yang dikenal dengan nama kriptografi. Kriptografi merupakan salah satu metode pengamanan data yang dapat digunakan untuk menjaga kerahasiaan data, keaslian data serta keaslian pengirim.

Rijndael adalah salah satu dari sekian banyak algoritma untuk keamanan data. Algoritma ini ditetapkan sebagai standard AES (Advanced Encryption Standard) yang ditetapkan oleh NIST (National Institute of Standard and Technology) pada tahun 2001.

2. TINJAUAN PUSTAKA

2.1 Definisi Kriptografi

Kriptografi adalah suatu ilmu yang mempelajari bagaimana cara mengamankan informasi. Kriptografi (cryptography) secara bahasa berasal dari bahasa Yunani yaitu: cryptos artinya secret atau rahasia dan graphein artinya writing atau tulisan. Jadi, kriptografi berarti secret writing atau tulisan rahasia. Kriptografi merupakan studi

teknik matematika yang berkaitan dengan aspek keamanan informasi seperti kerahasiaan, integritas data dan otentikasi. Teknik ini digunakan untuk mengubah data ke dalam kode-kode tertentu, dengan tujuan informasi yang disimpan atau dikirimkan tidak dapat dibaca oleh siapa pun kecuali oleh orang yang berhak.

Tujuan mendasar ilmu kriptografi: Kerahasiaan (confidentiality), Integritas data (integrity), Otentikasi (authentication), Ketiadaan penyangkalan (nonrepudiation)

Beberapa istilah penting yang digunakan dalam ilmu kriptografi adalah sebagai berikut:

Plaintext. Plaintext merupakan pesan asli yang belum disandikan atau informasi yang ingin dikirimkan atau dijaga keamanannya.

Ciphertext. Ciphertext merupakan pesan yang telah disandikan (dikodekan) sehingga siap untuk dikirimkan.

Enkripsi. Enkripsi merupakan proses yang dilakukan untuk menyandikan plaintext menjadi ciphertext dengan tujuan pesan tersebut tidak dapat dibaca oleh pihak yang tidak berwenang.

Dekripsi. Dekripsi merupakan proses yang dilakukan untuk memperoleh kembali plaintext dari ciphertext.

Kriptosistem. Kriptosistem merupakan sistem yang dirancang untuk mengamankan suatu sistem informasi dengan memanfaatkan kriptografi.

2.5 Algoritma Rijndael

Algoritma *Rijndael* menggunakan subtitusi, permutasi dan sejumlah putaran yang dikenakan pada tiap blok yang akan dienkripsi dan dekripsi. Untuk setiap putarannya, *Rijndael* menggunakan kunci yang berbeda. Kunci setiap putaran disebut *round key*. *Rijndael* beroperasi dalam orientasi *byte* sehingga memungkinkan untuk implementasi algoritma yang efisien ke dalam *software* dan *hardware*. Ukuran blok untuk algoritma *Rijndael* adalah 128 bit (16 *byte*).

Algoritma *Rijndael* ditetapkan sebagai AES (*Advenced Encryption Standard*) oleh NIST (*National Institute of Standard and Technology*) tahun 2001 untuk menggantikan DES (*Data Encryption Standard*) yang sudah berakhir masa penggunaannya sebagai standard enkripsi kriptografi simetri. DES dianggap sudah tidak aman lagi karena dengan perangkat keras khusus kuncinya bisa ditemukan dalam beberapa hari.

NIST kemudian mengadakan sayembara terbuka untuk membuat standard algoritma kriptografi yang baru sebagai pengganti DES. Standar tersebut nanti akan diberi nama Advanced Encryption Standart (AES).

AES mendukung panjang kunci 128 bit sampai 256 bit dengan step 32 bit. Panjang kunci dan ukuran blok dapat dipilih secara independen. Setiap blok dienkripsi dalam sejumlah putaran tertentu.

Karena AES menetapkan panjang kunci adalah 128, 192 dan 256, maka dikenal AES-128, AES-192 dan AES-256.

Tabel 1. Jumlah proses berdasarkan bit blok dan kunci

	(Nk)	(Nb)	(Nr)
AES – 128	4	4	10
AES – 192	6	4	12
AES – 256	8	4	14

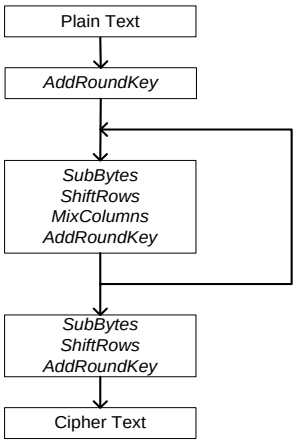
Algoritma *Rijndael* mempunyai 3 parameter:

1. *Plaintext* adalah *array* yang berukuran 16 *byte*, yang berisi data masukan
2. *Ciphertext* adalah *array* yang berukuran 16 *byte*, yang berisi hasil enkripsi.
3. *Key* adalah *array* yang berukuran 16 *byte*, yang berisi kunci *cipher* (disebut juga *cipher key*).

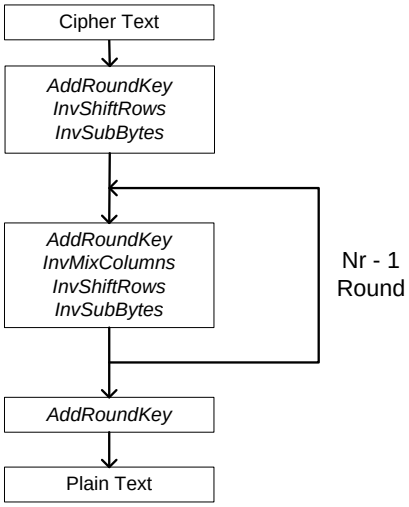
Garis besar algoritma *Rijndael* yang beroperasi pada blok 128 bit dengan kunci 128 bit adalah sebagai berikut:

1. *AddRoundKey*, melakukan XOR antara *state* awal (*plaintext*) dengan *cipher key*. Tahap ini disebut juga *initial round*.
2. Putaran sebanyak Nr-1 kali. Proses yang dilakukan pada setiap putaran adalah:
   a. *SubBytes* adalah subtitusi *byte* dengan menggunakan tabel subtitusi (*S-Box*).
   b. *ShiftRows* adalah pergeseran baris-baris *array state* secara *wrapping*.
   c. *MixColums* adalah mengacak data di masing-masing kolom *array state*.
   d. *AddRoundKey* adalah melakukan XOR antara state sekarang dengan *round key*
3. *Final round*, proses untuk putaran terakhir:
   e. *SubBytes*
   f. *ShiftRows*
   g. *AddRoundKey*

Garis besar algoritma *Rijndael:*



Gambar 1. Algoritma Enkripsi AES Rijndael



Gambar 2. Algoritma Dekripsi AES

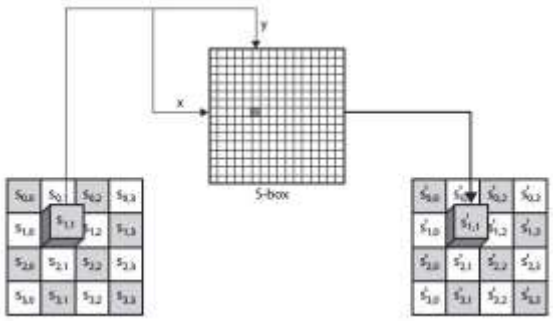
2.5.1 Transformasi *SubBytes*

Transformasi *SubBytes* memetakan setiap *byte* dari *array state* dengan menggunakan tabel subtitusi S-Box. Tidak seperti DES yang mempunyai S-Box yang berbeda pada setiap putaran, *Rijndael* hanya mempunyai satu macam tabel subtitusi.

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Gambar 3. S-Box *Rijndael*

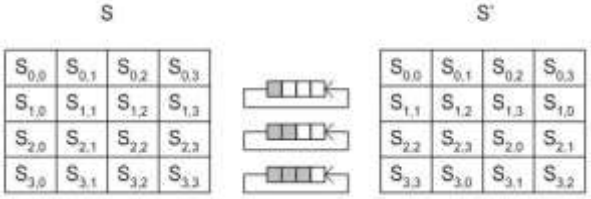
Cara pensubtitusiannya adalah sebagai berikut: jika setiap byte pada array state S[r,c] = xy, xy adalah digit heksadesimal dari nilai S[r,c], maka nilai subtitusinya, dinyatakan dengan S'[r,c], adalah elemen di dalam S-Box yang merupakan perpotongan baris x dengan kolom y.



Gambar 4. Transformasi SubByte dengan S-Box [5]

2.5.2 Transformasi *ShiftRows*

Transformasi *ShiftRows* melakukan pergeseran secara *wrapping* pada 3 baris terakhir dari array state. Jumlah pergeseran bergantung pada nilai baris r. Baris r = 1 digeser sejauh 1 byte, baris r=2 digeser sejauh 2 byte dan baris r=3 digeser sejauh 3 byte. Baris r=0 tidak digeser.



Gambar 5. Transformasi *ShiftRows*

2.5.3 Transformasi *MixColumns*

Transformasi *MixColumns* mengalikan setiap kolom dari *array state* dengan polinom a(x)mod(x^4+1). Setiap kolom diperlakukan sebagai polinom 4-suku pada GF(2^8). Polinom a(x) yang ditetapkan adalah:

$A(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$

Transformasi ini dinyatakan sebagai perkalian matrik:

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}$$

$S'(x) = a(x) \oplus s(x)$

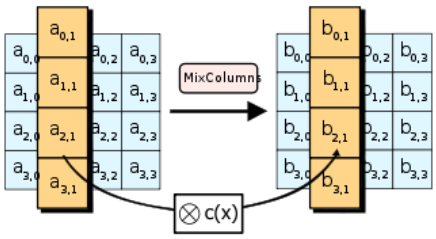
$S'_{0,c} = (\{02\} \cdot S_{0,c}) \oplus (\{03\} \cdot S_{1,c}) \oplus S_{2,c} \oplus S_{3,c}$

$S'_{1,c} = S_{0,c} \oplus (\{02\} \cdot S_{1,c}) \oplus (\{03\} \cdot S_{2,c}) \oplus S_{3,c}$

$S'_{2,c} = S_{0,c} \oplus S_{1,c} \oplus (\{02\} \cdot S_{1,c}) \oplus (\{03\} \cdot S_{3,c})$

$S'_{3,c} = (\{03\} \cdot S_{0,c}) \oplus S_{0,c} \oplus S_{1,c} \oplus (\{02\} \cdot S_{3,c})$

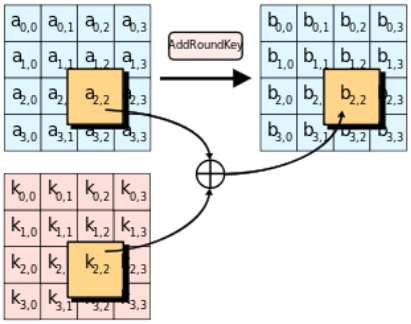
Berikut ini adalah gambaran proses *MixColumns:*



Gambar 6. Transformasi MixColumn [9]

2.5.4 Transformasi *AddRoundKey*

Transformasi ini melakukan operasi XOR terhadap sebuah *round key* dengan *array state*, dan hasilnya disimpan di *array state*.

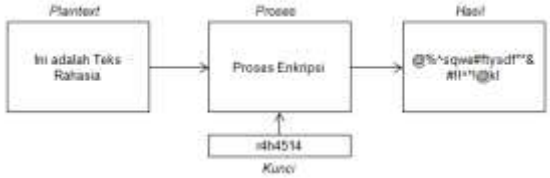


Gambar 7. Transformasi *AddRoundKey*

4. METODA

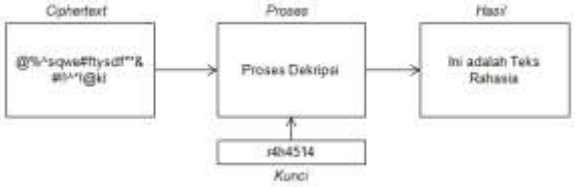
3.1 Gambaran Umum Sistem Enkripsi Dekripsi

Sistem ini dirancang untuk melakukan pengamanan data berupa teks yang akan dienkripsi menggunakan sebuah kata kunci. Proses enkripsi akan menghasilkan sebuah teks baru berupa data yang sudah diacak dan tidak dapat dikenali atau dibaca langsung.



Gambar 8. Gambaran umum sistem enkripsi

Untuk mengembalikan data yang sudah terenkripsi adalah dengan cara mendekripsi data tersebut menggunakan kunci yang sama ketika proses enkripsi. Hasil dari proses dekripsi menggunakan kunci yang benar akan mengembalikan data menjadi bentuk semula.



Gambar 9. Gambaran umum sistem dekripsi

3.2 Spesifikasi Sistem Perangkat Keras dan Lunak

3.2.1 Spesifikasi Perangkat Keras

Aplikasi ini dibangun menggunakan spesifikasi komputer sebagai berikut:

Tabel 2. Spesifikasi Perangkat Keras

Spesifikasi	Keterangan
Sistem Operasi	Windows 7 Professional
Processor	Intel Core 2 Duo @2,26 GHz
RAM	2,0 GB
Harddisk	320 GB

3.2.2 Spesifikasi Perangkat Lunak

Spesifikasi dari aplikasi yang akan direalisasikan adalah sebagai berikut:

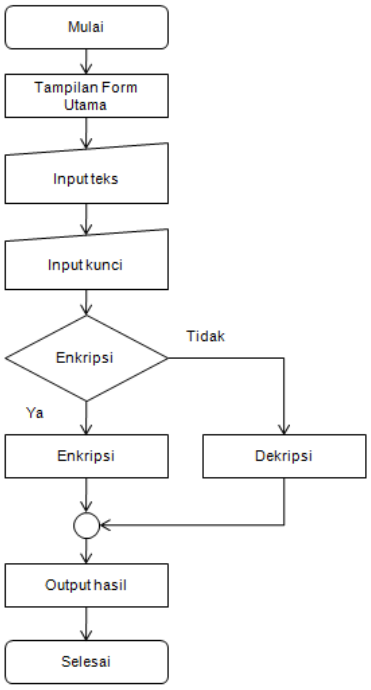
Tabel 3. Spesifikasi Aplikasi

Spesifikasi	Keterangan
Algoritma	AES Rijndael
Input data	Teks 128 bit
Input Key	Teks 128 bit
Output	Ciphertext 128 bit
Software	Visual Studio 2005 Express
Bahasa pemrograman	Visual Basic

3.3 Perancangan Sistem Enkripsi Dekripsi

3.3.1 Desain Sistem Enkripsi Dekripsi Data

Diagram alir dari aplikasi enkripsi dan dekripsi adalah sebagai berikut:



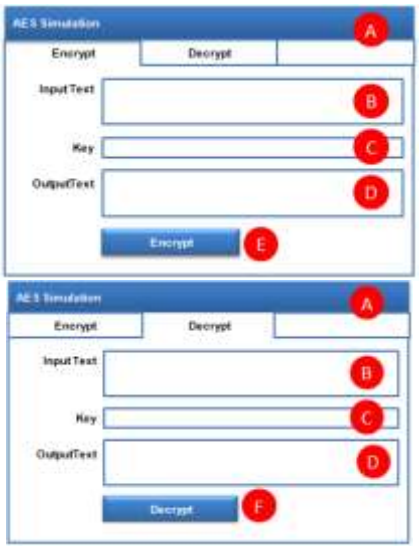
Gambar 10. Diagram Alir Sistem Enkripsi Dan Dekripsi Menggunakan Algoritma *Rijndael*

3.3.2 Perancangan Tampilan Aplikasi

Perancangan *user interface* pengguna merupakan salah satu bagian yang penting. Perancangan ini dipaparkan melalui gambar desain *form* aplikasi sistem. Perancangan *form* aplikasi ini merupakan tahapan interaksi antara manusia dengan komputer. *Form* ini dirancang agar komputer dapat berinteraksi dengan apa yang diinginkan *user* sehingga dengan *form* dapat diketahui pilihan-pilihan *input*, yang kemudian sistem akan memberikan respon berupa *output* data hasil proses enkripsi ataupun dekripsi.

Gambar 11. Desain *user interface* aplikasi

Penjelasan detail desain *user interface* aplikasi



Gambar 12. Penjelasan detail *user interface*

Tabel 4. Keterangan detail *user interface*

Kode	Keterangan
A	Header dan badan program
B	Input Teks yang akan dienkripsi atau dekripsi
C	Input kunci enkripsi
D	Output hasil enkripsi atau dekripsi
E	Tombol proses enkripsi
F	Tombol proses dekripsi

Desain *form* dirancang menggunakan *tools* Visual Studio 2005 Express Edition, dengan menggunakan bahasa pemrograman visual basic.

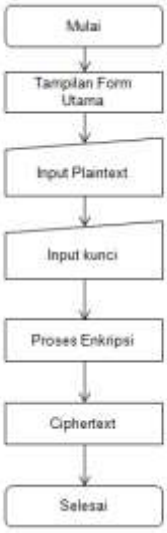
Gambar 13. Desain *user interface* menggunakan Visual Studio

4. HASIL DAN PEMBAHASAN

4.1 Pengujian Sistem

Setelah dilakukan pengerjaan keseluruhan sistem, maka perlu dilakukan pengujian serta penganalisaan terhadap alat, apakah sistem sudah berkerja dengan baik atau tidak.

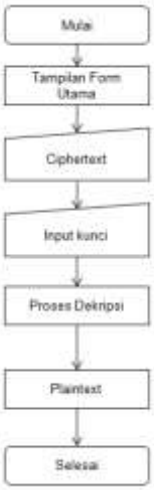
4.1.1 Skenario Pengujian Proses Enkripsi Data



Gambar 14. Diagram alir pengujian enkripsi data

Penjelasan pengujian proses enkripsi data: Masukan *input* data (teks) yang akan dienkripsi; Masukan kata kunci; Pilih proses enkripsi dengan cara menekan tombol Encrypt; Data hasil proses enkripsi (*ciphertext)* akan tampil pada kolom *output*

4.1.2 Skenario Pengujian Proses Dekripsi Data

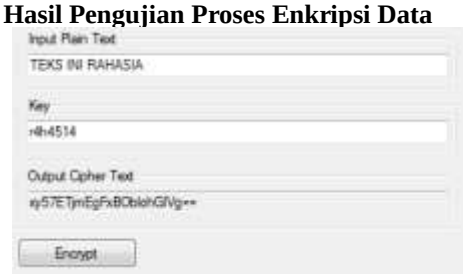


Gambar 15. Diagram alir pengujian dekripsi data

Penjelasan pengujian proses dekripsi data: Masukan *input* data hasil enkripsi (*ciphertext*) yang akan didekripsi; Masukan kata kunci yang sesuai dengan kunci enkripsi; Pilih proses dekripsi dengan cara menekan tombol *Decrypt;* Data hasil proses dekripsi (*plaintext)* akan tampil pada kolom *output*

4.2 Hasil Pengujian

4.2.1 Hasil Pengujian Proses Enkripsi Data



Gambar 16. Output ciphertext hasil enkripsi

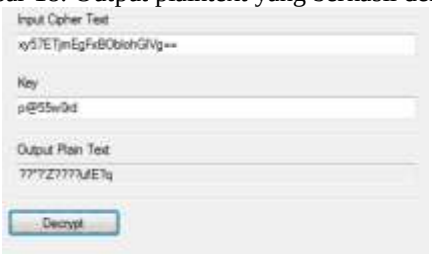
Gambar 17. Output ciphertext hasil enkripsi

Tabel 5. Hasil pengujian proses enkripsi data

Input Plaintext	Kunci	Output Ciphertext
TEKS INI RAHASIA	r4h4514	xy57ETjmEgFxB OblohGlVg==
TEKS INI RAHASIA	p@55w0rd	Jmdk91iIJrxgTW 6XKatIXg==

4.2.2 Hasil Pengujian Proses Dekripsi Data

Gambar 18. Output plaintext yang berhasil dekripsi



Gambar 19. Output plaintext yang berhasil dekripsi

4.3 Analisa Sistem Enkripsi Dekripsi Data

4.3.1 Analisa Hasil Pengujian Enkripsi

Hasil pengujian proses enkripsi data dengan *plaintext* "TEKS INI RAHASIA" berhasil dienkripsi menjadi *ciphertext*. Output *ciphertext* sangat bergantung dengan input *plaintext* dan juga kunci yang digunakan. Walaupun *plaintext* yang diinputkan sama tetapi kunci yang digunakan berbeda, maka output ciphertext hasil proses enkripsi-pun berbeda.

Proses pertama kali yang dilakukan adalah mengubah *plaintext* menjadi *block array* 16 *byte*. Setiap huruf akan dikodekan berdasarkan kode ASCII kemudian dikonversi menjadi bilangan basis 8 (hex).

Proses enkripsi yang pertama kali dilakukan adalah proses *AddRoundKey(0)*. Proses *AddRoundKey(0)* ini adalah proses operasi XOR antara *block array plaintext/State* dengan kunci *RoundKey*. Kunci *RoundKey* ini dibangkitkan sebelum proses enkripsi. Proses pembangkitan kunci *RoundKey* akan menghasilkan 11 kunci yang akan digunakan untuk proses *AddRoundKey(Round)* yang berbeda-beda untuk setiap ronde atau iterasi.

Untuk proses *AddRoundKey(0)* atau *AddRoundKey* yang pertama, *RoundKey* yang digunakan adalah *RoundKey(0)* atau *RoundKey* yang pertama. *RoundKey* yang pertama adalah kunci asli enkripsi yang masukan untuk proses enkripsi data.

```
 54 20 20 41
 45 49 52 53
 4B 4E 41 49
 53 49 48 41
                AddRoundKey(0)
     (+)        XOR With RoundKey(0)

 72 35 00 00
 34 31 00 00
 68 34 00 00
 34 00 00 00

--------------- State After AddRoundKey(0)
 26 15 20 41
 71 78 52 53
 23 7A 41 49
 67 49 48 41
```

Setiap elemen *block array plaintext/state* diproses opersi XOR dengan elemen *block* kunci dengan indeks yang sama. Proses yang terjadi untuk elemen Plaintext(1,1) XOR Kunci (1,1):

```
54 XOR 72 = 0101 0100 XOR 0111 0010 = 0010 0110 = 26
```

Setelah proses *AddRoundKey(0)* selesai selanjutnya adalah proses *SubBytes*. Setiap elemen *block array state* hasil proses sebelumnya disubtitusi/ditukar dengan elemen dari *array* S-Box sesuai dengan alamat yang ditunjuk oleh nilai elemen *block array*.

```
--------------- State After AddRoundKey(0)
 26 15 20 41
 71 78 52 53
 23 7A 41 49
 67 49 48 41
                SubBytes S-Box
--------------- State After SubBytes S-Box
 F7 59 B7 83
 A3 BC 00 ED
 26 DA 83 3B
 85 3B 52 83
```

Elemen *block array state* (1,1) dengan nilai 26 disubtitusi oleh S-Box(2,6) yang mempunyai nilai F7. *State(1,2)* yang bernilai 15 disubtitusi oleh S-Box(1,5) yang mempunyai nilai 59. Begitu seterusnya sampai setiap elemen tersubtitusi dengan *array* S-Box sesuai dengan nilai dari setiap elemen.

Proses selanjutnya adalah proses *ShiftRows*. Pada tahapan ini akan dilakukan pergeseran byte kearah kiri secara berputar untuk setiap baris. Jumlah pergeseran byte bergantung pada indeks baris dari *State*. Untuk *State* baris pertama tidak dilakukan pergeseran, baris kedua dilakukan pergeseran dan perputaran byte ke kiri sejauh 1 byte, baris ketiga dan keempat pergeseran dan perputaran sejauh 2 byte dan 3 byte.

```
--------------- State After SubBytes S-Box
  F7  59 B7 83
 [A3] BC 00 ED
 [26] DA 83 3B
 [85] 3B 52 83

                ShiftRows
--------------- State After ShiftRows
 F7  59  B7  83
 BC  00  ED [A3]
 83  3B [26] DA
 83 [85] 3B  52
```

Proses selanjutnya adalah proses *MixColumns*. Setiap kolom *State* akan dikalikan dengan *array* yang diberikan oleh sistem kemudian ditambah operasi XOR untuk setiap hasil perkalian setiap elemen yang kemudian membentuk kolom *State* baru.

```
--------------- State After ShiftRows
 [F7] 59 B7 83
 [BC] 00 ED A3
 [83] 3B 26 DA
 [83] 85 3B 52

--------------- Array Galois Field x Column(0) State
 [2] [3] [1] [1]      [F7]
 [1] [2] [3] [1]  x   [BC]
 [1] [1] [2] [3]      [83]
 [3] [1] [1] [2]      [83]
```

Hasil untuk keseluruhan proses *MixColumns* untuk iterasi yang pertama:

```
                State After MixColumns
 2A 0C 44 6B
 89 91 27 F9
 C8 BB 5B 79
 20 C1 7F 43
```

Proses selanjutnya adalah *AddRoundKey(1)*. Dan kemudian berlanjut semua proses dilakukan sampai dengan semua iterasi selesai.

Setelah melalui keseluruhan iterasi, diperoleh *block array state* terakhir dan hasil konversi menjadi *ciphertext* adalah sebagai berikut:

```
--------------- State After AddRoundKey(10)
 C7 38 71 A2
 2E E6 04 11
 7B 12 E6 A5
 11 01 E5 56

Ciphertext  : "xy57ETjmEgFxBOblohGlVg=="
```

Untuk proses simulasi enkripsi, pengkodean output *ciphertext* hasil proses enkripsi dikonversi menjadi teks berbasis 64 *encoding.* Hal ini dilakukan karena tidak semua elemen *state* bisa dikonversi dalam kode ASCII (ASCII terbatas sampai 7F).

4.3.2 Analisa Hasil Pengujian Dekripsi

Tabel 6. Hasil proses dekripsi data

Input Ciphertext	Kunci	Output Plaintext
xy57ETjmEgFxBOblohGlVg==	r4h4514	TEKS INI RAHASIA
Jmdk91iIJrxgTW6XKatIXg==	p@55w0rd	TEKS INI RAHASIA
xy57ETjmEgFxBOblohGlVg==	p@55w0rd	??*?'Z????ufE?q
Jmdk91iIJrxgTW6XKatIXg==	r4h4514	?W+????????5X*i

Hasil pengujian proses dekripsi data seperti ditunjukan oleh Tabel 6.. Output *plaintext* hasil dekripsi sangat bergantung dengan kunci yang digunakan. Ketika *ciphertext* yang diinputkan menggunakan kunci yang berbeda dengan kunci yang digunakan ketika proses enkripsi, maka tidak akan didekripsi.

Sebelum memasuki tahapan proses dekripsi yang pertama yaitu proses *AddRoundKey(10)* terlebih dahulu *ciphertext* dikonversi dari teks yang berbasis 64 *encoding* menjadi *block array* 16 byte. Kunci dekripsi juga diubah menjadi *block array key* 16 byte. Karena kunci yang digunakan untuk proses dekripsi sama dengan kunci yang digunakan untuk proses enkripsi, maka *RoundKey* yang dibangkitkanpun sama dengan *RoundKey* ketika proses enkripsi.

```
Ciphertext  : "xy57ETjmEgFxBOblohGlVg=="

                Cipher

 C7 38 71 A2
 2E E6 04 11
 7B 12 E6 A5
 11 01 E5 56
```

Tahapan yang pertama adalah proses *AddRoundKey(10)*. Karena proses dekripsi ini adalah kebalikan dari proses enkripsi, maka iterasi/putaran proses-nya adalah urutan terbalik dari 10 ke 0 putaran. Proses *AddRoundKey* dekripsi pada dasarnya sama seperti *AddRoundKey* pada proses enrkripsi, yang berbeda hanya urutan pemanggilan *RoundKey*.

```
 C7 38 71 A2
 2E E6 04 11
 7B 12 E6 A5
 11 01 E5 56

                AddRoundKey(10)
     (+)        XOR With RoundKey(10)

 45 51 EA C7
 D1 5A CF D7
 8D C1 14 24
 41 4A 2B 98

--------------- State After AddRoundKey(10)
 82 69 9B 65
 FF BC CB C6
 F6 D3 F2 81
 50 4B CE CE
```

Proses selanjutnya adalah proses *InvShitRows.* Proses *InvShiftRows* pada dasarnya sama dengan proses *ShiftRows* pada proses enkripsi. Perbedaannya adalah putaran 1 byte berlawanan arah dengan putaran pada saat enkripsi, yaitu berputar 1 byte ke kanan.

```
--------------- State After AddRoundKey(10)
 82 69 9B 65
 FF BC CB C6
 F6 D3 F2 81
 50 4B CE CE
                InvShiftRows
--------------- State After InvShiftRows
 82 69 9B 65
 C6 FF BC CB
 F2 81 F6 D3
 4B CE CE 50
```

Proses berikutnya adalah proses *InvSubBytes*. Proses *InvSubBytes* pada dasarnya sama dengan proses *SubBytes* pada proses enkripsi. Perbedaannya adalah *array* yang digunakan adalah InvS-Box. *Array* InvS-Box bisa dilihat dilampiran proses detail pembangkitan kunci.

```
--------------- State After InvShiftRows
 82 69 9B 65
 C6 FF BC CB
 F2 81 F6 D3
 4B CE CE 50
                InvSubBytes S-Box
```

```
---------- State After InvSubBytes S-Box
 11 E4 E8 BC
 C7 7D 78 59
 04 91 D6 A9
 CC EC EC 6C
```

Proses berikutnya dilanjutkan pada proses *AddRoundKey(9)* yang secara konsep sama dengan *AddRoundKey* sebelumnya namun penggunaan kunci *RoundKey*-nya saja yang berbeda.

Tahapan dekripsi selanjutnya adalah proses *InvMixColumns*. Proses ini pada dasarnya sama dengan proses *MixColumns* pada proses enkripsi.

```
---------- State After AddRoundKey(9)
 CF F0 53 91
 12 F6 ED 41
 E4 DD 03 99
 55 E7 8D DF

----------  Inv  Array  Galois  Field  x  Column(0)
State
 [E]  [B]  [D]  [9]      [CF]
 [9]  [E]  [B]  [D]   x  [12]
 [D]  [9]  [E]  [B]      [E4]
 [B]  [D]  [9]  [E]      [55]
```

Setelah melalui keseluruhan iterasi, diperoleh *block array state* terakhir dan hasil konversi menjadi *Plaintext* adalah sebagai berikut:

```
---------- State After AddRoundKey(0)
 54 20 20 41
 45 49 52 53
 4B 4E 41 49
 53 49 48 41

Plaintext  : "TEKS INI RAHASIA"
```

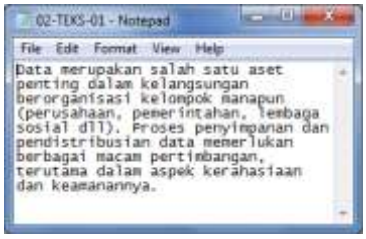
4.4 Pengujian Program Menggunakan Input File

Setelah melakukan pengujian dan analisa proses enkripsi dekripsi menggunakan teks, pada bagian ini ditampilkan beberapa hasil pengujian menggunakan input berupa file.

Pada pengujian program ini, yang menjadi input data adalah file komputer yaitu: file *notepad* (.txt), *word document* (.doc), *image* (.png), *sound* (.mp3) dan *video* (.flv). Untuk melihat pengujian hasil proses enkripsi dibantu dengan menggunakan program HxD yaitu aplikasi *freeware* yang berfungsi untuk membaca byte-byte dari suatu file, sehingga perbedaan file sebelum dan sesudah proses enkripsi bisa dilihat perubahannya.

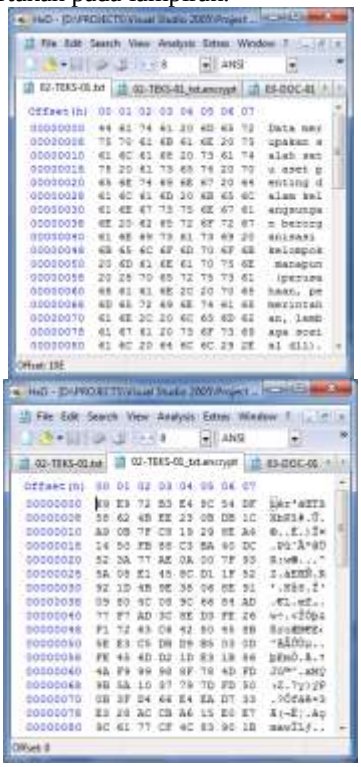
Proses enkripsi file pada dasarnya sama dengan proses enkripsi teks, file yang berukuran lebih dari 128 bit / 16 byte akan dipecah menjadi potongan-potongan 16 byte dan akan diproses setiap 16 byte untuk dienkripsi. Setelah proses enkripsi selesai akan digabung/disusun ulang sehingga menjadi satu file kembali.

Setiap file yang dienkripsi akan menghasilkan file baru dengan format ekstensi .encrypt. Untuk mengembalikan file agar bisa terbaca kembali, sama seperti halnya proses enkripsi pada teks, yaitu dilakukan proses dekripsi dengan menggunakan kunci enkripsi dekripsi yang sama. Berikut hasil proses enkripsi menggunakan input file *notepad*.



Gambar 20. File *notepad* yang akan dienkripsi

Hasil dari proses enkripsi file bisa dilihat menggunakan bantuan aplikasi HxD. Pada gambar 4.16 terlihat perbedaan antara byte-byte file *notepad* sebelum enkripsi dan setelah proses enkripsi. Proses untuk file lainnya disertakan pada lampiran.



Gambar 21. Pembacaan byte file *notepad* sebelum dan sesudah proses enkripsi

1. SIMPULAN

Proses enkripsi menggunakan algoritma AES Rijndael dengan kunci 128-bit berhasil dilakukan yang disimulasikan pada aplikasi AES Simulator.

Ciphertext hasil dari proses enkripsi sebelumnya berhasil didekripsi dengan menggunakan kunci yang sama ketika proses enkripsi

Output *Plaintext* hasil dekripsi sangat bergantung dengan kunci yang digunakan.

Ketika *ciphertext* didekripsi menggunakan kunci yang berbeda dengan kunci yang digunakan ketika proses enkripsi, maka tidak akan didekripsi / Informasi tidak akan terbuka.

Proses enkripsi dekripsi juga telah diterapkan pada berbagai jenis file komputer diantaranya file *notepad, word document, sound* dan *video*.

DAFTAR PUSTAKA

Cobb, Chey, 2004. *Cryptography For Dummies"*, John Wiley & Sons.

Delf, Hans, 2007. *Introduction to Cryptography Principles and Applications Second Edition*, Springer, Berlin.

Munir, Renaldi, 2006. *Kriptografi*, Penerbit Informatika, Bandung.

Petre, Ion. *"Cryptography and Network Security "*. www.abo.fi/~ipetre/crypto/lecture4.pdf . (diakses selama Juni – September 2011)

Satria, Eko, 2009. *Studi Algoritma Rijndael Dalam Sistem Keamanan Data. Skripsi*, USU Medan.

Surian, Didi, 2006. *Algoritma Kriptografi Aes Rijndael*, Jurnal Teknik Elektro TESLA Vol. 8 No. 2, 97 – 101 (Oktober 2006)),

Stalling, W., 2005. *Cryptography and Network Security, Principle and Practice 4thEdition*", Prentice Hall,

Stalling, W., 2004. *Data and Computer Communications 7th Edition*", Prentice Hall.

Virgan R.Y. 2004. *Aplikasi Enkripsi dan Dekripsi Menggunakan Algoritma Rijndael". Makalah:* Undip Semarang.

Xiang-Yang Li, *"Cryptography and Network Security ".* www.cs.iit.edu/~cs549/lectures/CNS-1.pdf (diakses selama Juni - September 2011)

Zabala, Enrique, *"Rijndael Cipher".* www.cryptool.org. (diakses selama Juni – September 2011)