Studi dan Implementasi Algoritma Rijndael Untuk Enkripsi Halaman Web HTML

Hadyan Ghaziani Fadli – NIM : 13505005

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if15005@students.if.itb.ac.id

Abstraksi

Makalah ini membahas tentang studi dan ide implementasi Algoritma Rijndael dalam melakukan enkripsi halaman web berbasis HTML yang sebatas hanya mengenkripsi setiap kata pada halaman tersebut sehingga menjadi *ciphertext* yang tidak dapat dibaca. Algoritma yang dipakai adalah algoritma Rijndael yang merupakan algoritma AES.

Implementasi algoritma enkripsi ini bertujuan untuk mengamankan pengiriman konten dari halaman web dari server web hingga sampai di client. Proses enkripsi dilakukan pada sisi server. Server akan mengirimkan halaman web yang sudah terenkripsi disertai fungsi javascript yang tidak di enkripsi yang dapat melakukan dekripsi pada konten terenkripsi dengan cara client memasukkan kata sandi ke prompt dari fungsi javascript tersebut.

Pada Dasarnya Algoritma Rijndael merupakan salah satu finalis lomba yang diadakan oleh *National Institute of Standards and Technology* (*NIST)* yang tujuannya membuat standard algoritma kriptografi yang baru. Standard tersebut kelak diberi nama *Advanced Encryption Standard* (*AES*). Rijndael menjadi standard kriptografi yang dominan paling sedikit selama 10 tahun.

***Kata kunci**: Advanced Encryption Standard*, cipherteks, enkripsi, dekripsi, Algoritma, web HTML, prompt, server, client.

1. Pendahuluan

Dengan berkembangnya zaman, algoritma-algoritma kriptografi standar tidak lah lagi mampu merahasiakan suatu konten dengan baik. Adanya penciptaan computer menjadi alas an utamanya. Algoritma-algoritma kriptografi klasik menjadi sangat mudah untuk dipecahkan. Jika pada zaman sebelum berkembangnya komputer untuk memecahkan suatu cipherteks yang dibentuk dari algoritma kriptografi klasik dilakukan secara manual dan tentunya membutuhkan waktu yang lama, kini dengan bantuan komputer dan perangkat lunak yang diciptakan untuk memecahkan cipherteks dengan algoritma tersebut, memecahkan cipherteks tersebut sangatlah mudah, bahkan tidak sampai hitungan jam.

Karenanya berkembanglah berbagai algoritma kriptografi modern seperti *Data Encription Standard (DES)* yang menandai mulainya era kriptografi modern.

Namun dengan berkembangnya kemampuan dan kecepatan proses dari komputer, dimana sebuah komputer sudah mampu melakukan *brute force attack* dalam waktu yang cukup singkat terhadap *DES* yang hanya memiliki panjang 64 bit dengan 8 bit yang merupakan paritas, kini *DES* dirasa sudah tidak aman lagi.

Karena hal tersebut, diperlukan algoritma baru yang dapat menggantikan DES. *National Institute of Standards and Technology* (*NIST*) mengusulkan kepada Pemerintah Federal AS untuk sebuah standard kriptografi kriptografi yang baru. *NIST* mengadakan lomba membuat standard algoritma kriptografi yang baru untuk menggantikan DES. Standard tersebut kelak diberi nama *Advanced Encryption Standard* (*AES*).

Diantara hasilnya, dirumuskan sebuah algoritma *Rijndael* (dari Vincent **Rij**men da**n** Joan **Dae**men – Be**l**gia). Pada Oktober 2000, *NIST* mengumumkan untuk memilih algoritma Rijndael, kemudian pada Bulan November 2001, Algoritma Rijndael ditetapkan sebagai AES. Algoritma ini diharapkan dapat bertahan selama 10 tahun.

Tidak seperti DES, Algoritma lebih aman. Algoritma *AES* merupakan algoritma kriptografi simetrik yang beroperasi dalam mode (*block cipher*) yang memproses blok data 128-bit dengan panjang kunci 128-bit, 192-bit, atau 256-bit sehingga dikenal dengan (*AES*-128), (*AES*-192) dan (*AES*-256).

Seiring berkembangnya sistem informasi dan komunikasi, kebutuhan akan penerapan kriptografi dalam berkomunikasi kini kian berkembang. Salah satu teknologi informasi yang sangat berkembang adalah teknologi sistem informasi atau perangkat lunak berbasis web HTML. Seperti halnya teknologi informasi lainnya, teknologi web dengan HTML juga memerlukan kriptografi untuk merahasiakan data yang dikirimkan dan diterima agar tidak dapat dibajak atau dicuri oleh pihak yang tidak berhak. Berbagai protokol diciptakan seperti *https* agar pengiriman data dari server ke client dapat berjalan dengan aman.

Kali ini akan dibuat sebuah cara sederhana untuk melakukan enkripsi pada pengiriman konten dari halaman web HTML pada server dan mendekripsinya di sisi klien dengan kriptografi yang menerapkan algorimta Rijndael ini.

Algoritma Rijndael ini dapat diterapkan pada enkripsi teks, salah satunya pada pengiriman halaman web HTML. Hal ini dilakukan karena terkadang isi dari suatu halaman web hanya boleh dibaca seseorang, dan serangan-serangan di LAN seperti *Man In The Middle Attack* sering kali dilakukan di dalam jaringan. Sehingga dengan mengenkripsi halaman web tersebut, jika seseorang tidak memiliki kunci, maka dia tidak akan dapat membaca halaman web tersebut. Ini merupakan aplikasi sederhana dari algoritma Rijindael untuk mengamankan pengiriman halaman web yang cocok digunakan untuk website sederhana yang memerlukan pengamanan terhadap pihak yang tidak berhubungan dengan hanya memanfaatkan biaya yang sangat murah.

2.1 Algoritma Simetri dan Asimetri

Algoritma kunci simetri adalah sebuah kelas dari algoritma kriptografi yang menggunakan kunci kriptografi yang memiliki relasi trivial , atau bahkan biasanya identik antara yang digunakan untuk enkripsi maupun dekripsi.

Algoritma kunci simentri terbagi menjadi *stream cipher* dan *block cipher*.

- **Algoritma stream cipher :**
  Algoritma ini melakukan enkripsi terhadap bit dari *plainteks* dalam satu waktu atau bisa dibilang bit per bit seperti aliran
- **Algoritma block cipher :**
  Algoritma ini melakukan enkripsi dengan cara membagi plainteks kedalam blok-blok bit dengan ukuran tertentu dan mengenkripsinya dalam satu kesatuan. Karena algoritma Rijndael merupakan algoritma *block cipher*, maka selanjutnya hanya akan dibahas *block cipher.*

2.2 Block Cipher

2.2.1 Gambaran Umum Block Cipher

Pada *block cipher*, bit-bit plainteks dibagi menjadi blok-blok bit dengan panjang yang telah ditentukan dan tentunya panjangnya sama untuk setiap blok. Proses enkripsi dan dekripsi dilakukan secara bersamaan ke setiap blok dan menghasilkan blok cipherteks pada proses enkripse sedangkan pada proses dekripsi tentunya harus mengembalikan cipherteks tersebut menjadi blok plainteks. Ukuran blok cipherteks dan plainteks adalah sama.

Anggap E_k adalah fungsi enkripsi, P adalah plainteks dan C merupakan cipherteks sedangkan D_k adalah fungsi dekripsi, maka dapat dilihat prosesnya adalah sebagai berikut :

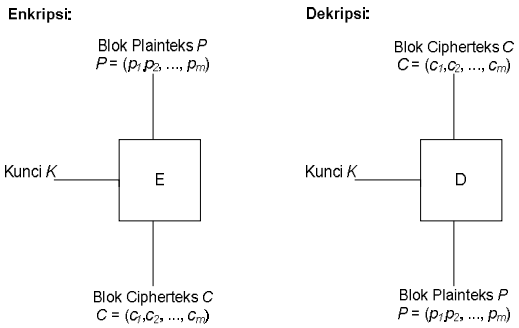
Proses enkripsi dengan kunci *K* dinyatakan sebagai berikut :

$$E_k(P) = C$$

Proses dekripsi dengan kunci *K* dinyatakan sebagai berikut :

$$D_k(C) = P$$

Skema enkripsi dan dekripsi dengan *block cipher* dapat dilihat sebagi berikut



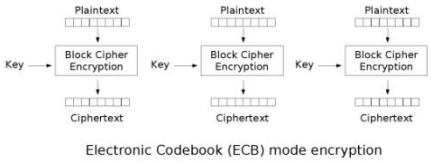
Gambar 1.
skema enkripsi dan dekripsi pada block cipher

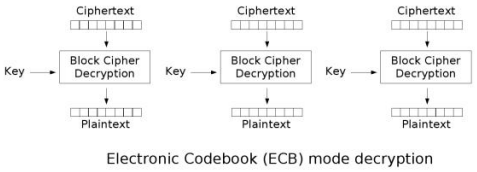
2.2.2 Mode Operasi *Block Cipher*

Dalam kriptografi, blok chipper berjalan di blok dengan ukuran yang fix biasanya 64 atau 128 Bits. Karena pesan bisa terdiri dari panjang yang tidak tentu dan karena mengenkripsi sebuah plainteks dengan kunci yang sama akan menghasilkan output yang sama, maka beberapa mode operasi telah dibuat.
Model yang paling awal dalam literature seperti ECB, CBC, OFB dan CFB menyediakan hanya keintegrasian pesan atau kerahasiaan, tetapi tidak melakukan keduanya.
Semua mode kecuali ECB memerlukan *initialization vector*, sebuah dummy block untuk memulai proses untuk sebuah blok yang sesungguhnya dan juga menyediakan pengacakan untuk proses. *IV* tidak perlu dirahasiakan.

2.2.2.1 *Electronic Code Book* (*ECB*)

Mode enkripsi yang paling sederhana adalah **electronic codebook** (ECB). Pesan dibagi ke beberapa blok dan setiap blok dienkripsi secara terpisah. Kerugian dari metode ini adalah plainteks yang identik akan dienkripsikan menjadi cipherteks yang identik.

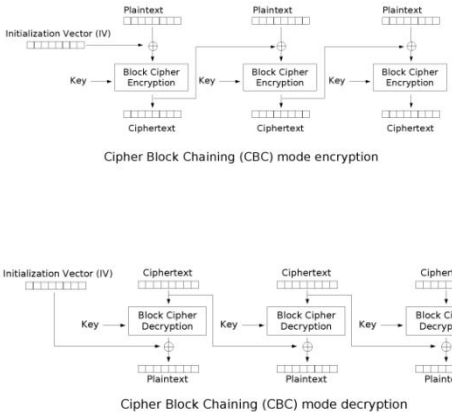




Gambar 2. skema EBC

2.2.2.2 *Cipher-block chaining (CBC)*

Mode CBC pertama diluncurkan oleh IBM pada tahun 1976. Di sini, tiap blok plainteks di XOR dengan cipherteks sebelumnya sebelum dienkripsi. Disini tiap cipherteks tergantung pada semua blok plainteks. tentu saja *Initial Vector* digunakan untuk proses blok pertama.



Gambar 3. skema CBC

Jika blok pertama mendapat index 1, maka rumus enkripsinya adalah :

$$C_i = E_K(P_i \oplus C_{i-1}), C_0 = IV$$

Rumus dekripsinya :

$$P_i = D_K(C_i) \oplus C_{i-1}, C_0 = IV$$

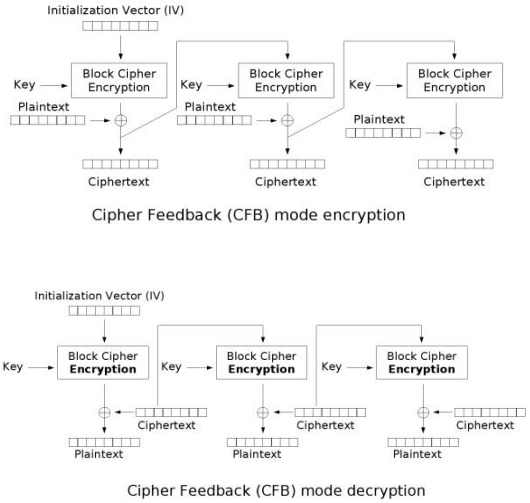
2.2.2.3 *Cipher-Feedback* (*CFB*)

Mode **cipher feedback** (CFB), mirip dengan CBC , membuat cipher block menjadi *close self-synchronizing steam cipher*. Operasinya sangat mirip dan proses dekripsinya juga sangan mirip dengan CBC.

$$C_i = E_K(C_{i-1}) \oplus P_i$$

$$P_i = E_K(C_{i-1}) \oplus C_i$$

$$C_0 = \text{IV}$$



Gambar 4. skema CFB

Cara termudah dalam menggunakan CFB dapat dilihat diatas, yang tidak lagi *self-synchronizing* disbanding mode cipher lainnya. Jika seluruh ukuran blok cipherteks hilang, CBC dan CFB akan melakukan sinkronisasi tetapi jika kehilangan 1 byte saja, maka tidak dapat lagi didekripsi. Untuk mensikronsasi kehilangan byte tersebut maka setiap byte harus dienkripsi disaat yang bersamaan. CFB dapat digunakan demgam ,menglombinasikan dengan shift register sebagai input dari blok cipher.

3. Algoritma *Rijndael / AES*

3.1 Gambaran Umum Algoritma *Rijndael / AES*

Seperti yang sudah dijelaskan sedikit pada pendahuluan , AES merupakan algoritma yang capat baik di sisi perangkat lunak maupun perangkat keras dan relatif mudah untuk diimplementasikan serta hanya memerlukan sedikit memory.
Tidak seperti predesesornya DES, AES tidak menggunakan Feistel network.
AES memiliki ukuran blok yang tetap yaitu 128, 192 dan 256. Tidak seperti DES, AES bekerja pada byte.

AES spesifik sebagai sejumlah repetisi dari *transformation round* yang mengkonversi input *plainteks* menjadi *cipherteks*. Tiap *round* terdiri dari sejumlah tahapan proses termasuk satu yang bergantung pada kunci enkripsi.
Untuk melakukan dekripsi, urutan *round* dibalik sehingga cipherteks dapat kembali menjadi plainteks.

3.2 Garis Besar Algoritma Rijndael

Garis besar Algoritma *Rijndael* adalah sebagai berikut (di luar proses pembangkitan *round key*):

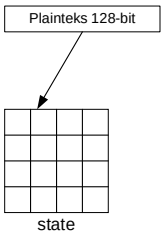
1. *AddRoundKey*: melakukan *XOR* antara *state* awal (plainteks) dengan *cipher key*. Tahap ini disebut juga *initial round*.
2. Putaran sebanyak *Nr* – 1 kali. Proses yang dilakukan pada setiap putaran adalah:
    a. *SubBytes*: substitusi *byte* dengan menggunakan tabel substitusi (*S-box*).
    b.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

3.2.1 Ronde awal beserta penjelasan State, Kunci Cipher dan jumlah ronde

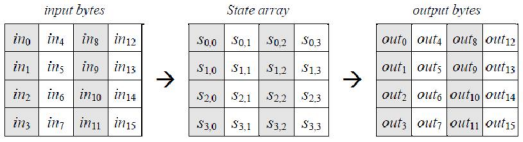
Algoritma *Rijndael* mempunyai 3 parameter sebagai berikut (untuk AES-128) :

1. **plainteks :** *array* yang berukuran 16 *byte*, yang berisi data masukan.
2. **cipherteks :** *array* yang berukuran 16 *byte*, yang berisi hasil enkripsi.
3. **key :** *array* yang berukuran 16 *byte*, yang berisi kunci ciphering (disebut juga *cipher key*).

Jika dilihat, 16 byte sama dengan 128 bit (1byte = 8 bit).
Selama kalkulasi plainteks menjadi cipherteks, status sekarang dari data disimpan di dalam *array of bytes* dua dimensi, yang disebut state, yang berukuran NROWS × NCOLS. Untuk blok data 128-bit, ukuran state adalah 4 × 4.
Elemen *array* state diacu sebagai $S[r,c]$, $0 \le r < 4$ dan $0 \le c < Nb$ (*Nb* adalah panjang blok dibagi 32. Pada *AES*-128, *Nb* = 128/32 = 4).



Pada awal enkripsi, 16-*byte* data masukan, in_0, in_1, …, in_{15} disalin ke dalam *array* state :



Kemudian Kunci cipher digambarkan sebagai suatu persegi dengan ketentuan panjang blok kunci adalah panjang kunci dibagi 32 yang akan dilambangkan sebagai Nk. Untuk kunci dengan panjang 128 bit bisa dilihat seperti gambar dibawah yaitu dengan panjang blok 4, sedangkan untuk panjang kunci 192 bit dan 256 bit masing-masing sepanjang 6 dan 8.

$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$
$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$
$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$
$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$

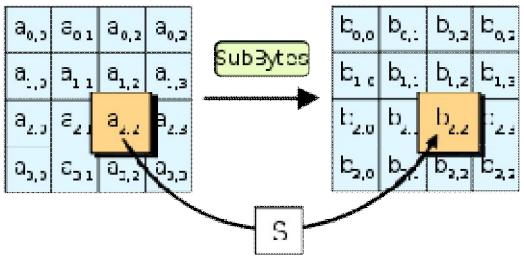
Jumlah ronde (Nr) bergantung pada Nk dan Nb

Nr	Nb = 4	Nb = 6	Nb = 8
Nk = 4	10	12	14
Nk = 6	12	12	14
Nk = 8	14	14	14

3.2.2 Ronde Transformasi

Ronde ini terdiri dari 4 tahap yaitu :

3.2.2.1 Transformasi SubBytes



Gambar 5. skema transformasi subbytes

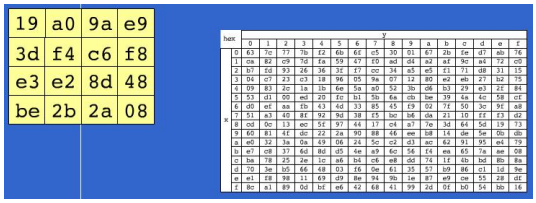
$$s'(x) = a(x) \otimes s(x)$$

Dalam transformasi SubBytes, tiap byte dalam state digantikan oleh entry yang ada di tabel S-Box berikut.

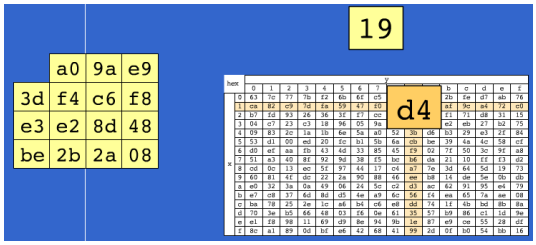
hex	y 0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x 0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	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

Gambar 6. S-Box

Caranya adalah dengan memetakan tiap byte dengan S-Box contohnya :



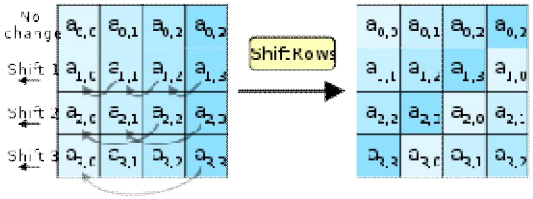
Dipetakan dengan cara :



Gambar 7. cara pemetaan dengan S-Box

Misalkan pada kolom $a_{0,0}$ nilainya adalah 19, kemudian nilai tersebut dipecah kedalam 2 variabel x dan y. nilai x adalah nilai pada digit pertama dari kiri dalam hal ini x = 1 dan y merupakan digit ke 2 dalam hal ini y = 9. Nilai x = 1 dan y = 9 pada S-Box dipetakan menjadi d4.

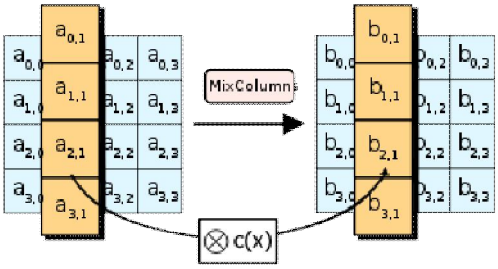
3.2.2.2 Transformasi ShiftRows



Gambar 8. skema transformasi ShiftRows

Transformasi *ShiftRows* dilakukan dengan melakukan pergeseran secara siklik pada 3 baris terakhir dari *array state*. Jumlah pergeseran bergantung pada urutan 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.

3.2.2.3 Transformasi *MixColumns*



Gambar 9. skema 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).c(x) yang ditetapkan adalah:

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

c(x) = ‘03’ x^3 + ‘01’ x^2 + ‘01’ x + ‘02’

invers dari MixColumn sama dengan MixColumn. Setiap kolom ditransformasi dengan mengalikannya dengan sebuah perkalian polynomial d(x)

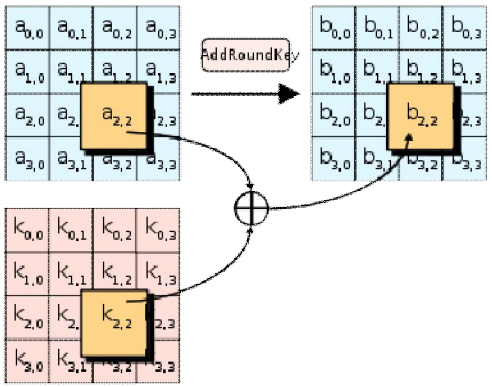
(‘03’ x^3 + ‘01’ x^2 + ‘01’ x + ‘02’) ⊗ d(x) = ‘01’

Yang dijelaskan dengan :

d(x) = ‘0B’ x3 + ‘0D’ x2 + ‘09’ x + ‘0E’

3.2.2.4 Transformasi Add Round Key

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



4. 1 Implementasi Pada Halaman Web HTML

Cara yang digunakan untuk melakukan enkripsi ini adalah dengan melakukan skenario berikut. Skenario yang digunakan adalah :

1. user akan mendaftarkan passwordnya secara aman di suatu website yang akan menjadi tujuan enkripsi tadi, dengan asumsi tidak ada pencurian password disini. Dalam pendaftaran tadi diberikan 3 variabel yang diperlukan yaitu username login , password login, password enkripsi.

2. ketika user melakukan login ke website tersebut di suatu tempat, user hanya memasukkan username login dan password login saja.

3. Server web tersebut akan men-generate halaman web dengan sebelumnya mengenkripsi terlebih dahulu dengan menggunakan algoritma *Rijndael* dan tentu saja password / key yang digunakan adalah password enkripsi tadi.

4. server web juga men-generate sebuah dekriptor javascript di halaman web tersebut, dimana user tadi harus memasukkan password enkripsinya. Proses dekripsi ini aman karena tidak melalui jaringan. Setiap komunikasi selain saat login dilakukan dengan melakukan enkripsi terlebih dahulu. Untuk pengiriman data dari client ke server juga dilakukan dengan cara yang sama yaitu ketika client melakukan submit, sebuah fungsi javascript akan mengenkripsi data yang disubmit client kemudian di server akan di dekripsi

dengan password enkripsi yang telah terdaftar.

4. 2 Contoh sederhana enkripsi Halaman Web HTML

Server akan mengirim sebuah fungsi javascript disertai konten teks seperti ini :

* teks ini akan dienkripsi dan dikirim kan ke client *

Selanjutnya teks tersebut akan dienkripsi terlebih dahulu sebelum dikirimkan ,sehingga yang dikirimkan ke client adalah sebuah fungsi javascript untuk mendekripsi dengan meminta masukan password enkripsi user lewat form dan teks berikut :

* zRmvSYeHh4fjEk9KJjz++E60AqxO5jSiH9yGImxSUj8uw9VDYRBmCslhAUy1fiJYEph5YgK3+dEmjQ== *

Kemudian user akan memasukkan kunci enkripsi sebagai berikut

Akhirnya fungsi javascript di client akan mendekripsi isi dari div tersebut apapun isi teksnya, tentunya hasil dekripsi tergantung kunci yang dimasukkan, jika benar maka plainteks akan terdekripsi seperti semula.

5. Kesimpulan

Kesimpulan yang dapat dimbil dari studi dan implementasi AES pada enkripsi halaman web HTML adalah:

1. *Advanced Encryption Standard* (*AES*) / algoritma Rijndael merupakan salah satu solusi yang baik untuk melakukan pengamanan pengiriman dan penerimaan data dengan melakukan enkripsi.
2. *AES* merupakan algoritma yang efisien baik dari segi hardware maupun software. Tetapi AES memiliki keamanan yang cukup baik untuk saat ini.
3. Implementasi pada enkripsi halaman web berhasil karena tag-tag pada halaman web HTML berbasiskan teks. Konten halaman web yang bukan berbasiskan teks tidak dienkripsi, tetapi tidak akan muncul jika tidak dipanggil oleh tag HTML.
4. Implementasi ini berguna untuk web-web HTML sederhana yang memerlukan fitur kriptografi.

DAFTAR PUSTAKA

[1] Daemen, Joan, Vincent Rijmen. (2004). The *Rijndael* Specification. *http://csrc.nist.gov/encryption/AES/Rijndael/Rijndael.pdf.* Tanggal akses: 4 Desember 2004 pukul 20:00.

[2] Munir, Rinaldi. (2004). Bahan Kuliah IF5054 Kriptografi. Departemen Teknik Informatika, Institut Teknologi Bandung.

[4] *NIST*. (2004). National Institute of Standards and Technology. *http://www.nist.gov*. Tanggal akses: 1 April 2009 pukul 20:00.

[5] Advanced Encryption Standard. *http://en.wikipedia.org/wiki/Rijndael_algorithm* . Tanggal akses: 1 April 2009 pukul 20:00.

[6] Data Encryption Standard. *http://en.wikipedia.org/wiki/Data_Encryption_Standard* . Tanggal akses: 1 April 2009 pukul 20:00.

[7] Block cipher modes of operation *http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation* . Tanggal akses: 1 April 2009 pukul 20:00.