Algoritma AES (*Advanced Encryption Standard*) dan Penggunaannya dalam Penyandian Pengompresian Data

Bernardino Madaharsa Dito Adiwidya – NIM: 135070789
Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung,
Jalan Ganesha 10, Bandung, email: if17089@students.if.itb.ac.id

Abstract – *Makalah ini membahas algoritma AES (Advanced Encryption Standard) atau Rijndael sebagai salah satu metode kriptografi. Algoritma ini diketahui sangat unggul dalam pengenkripsian dan pendekripsian data. AES digunakan dalam berbagai penyandian. Salah satunya adalah untuk penyandian sandi-lewat yang digunakan pada aplikasi pengompresian data.*

Kata Kunci*: algoritma, AES, kriptografi, penyandian, sandi-lewat, pengompresian data.*

1. PENDAHULUAN

Pada saat ini banyak orang membutuhkan komputer untuk menyelesaikan berbagai pekerjaannya. Komputer-komputer dapat digunakan untuk memenuhi kebutuhan pribadi maupun untuk kepentingan perusahaan atau organisasi tertentu dalam berbagai bidang. Bidang-bidang tersebut antara lain pemerintahan, militer, pendidikan, transportasi, perdagangan, industri, dan sebagainya.

Dengan perkembangan teknologi komputer saat ini, pertukaran informasi dari suatu pihak ke pihak lain sangatlah diperlukan. Informasi yang dipertukarkan itu biasanya tidak ingin diketahui oleh pihak-pihak lain, terutama oleh pihak yang bertentangan dengan pihak yang bertukar informasi tersebut atau pihak yang baik sengaja maupun tidak sengaja dapat memanfaatkan informasi tersebut. Jika keamanan pertukaran informasi ini tidak dapat dijaga, pihak-pihak lain tersebut dapat memanfaatkan informasi tersebut sehingga merugikan pihak-pihak yang berhak atas informasi tersebut.

Ancaman keamanan terhadap informasi tersebut dapat berupa berbagai bentuk. Bentuk ancaman tersebut dapat berupa interupsi, intersepsi, modifikasi, dan fabrikasi. Ancaman interupsi dapat mengganggu ketersediaan data. Data yang ada dapat dihapus sehingga pihak yang membutuhkan informasi tersebut tidak dapat menemukan datanya. Ancaman intersepsi merupakan ancaman terhadap kerahasiaan data. Informasi yang ada disadap dan dipergunakan oleh pihak yang tidak berhak sehingga merugikan pengguna data yang sah. Ancaman modifikasi mengakibatkan kesalahan dalam penerimaan informasi sehingga informasi yang diterima tidak sesuai dengan keinginan penerima maupun pengirimnya. Ancaman fabrikasi merupakan ancaman terhadap integritas karena informasi yang berhasil dicuri oleh pihak yang tidak berhak dipalsukan, lalu dikirimkan kepada penerima seolah-olah berasal dari pengirim yang sah.

Untuk pengatasi ancaman-ancaman tersebut, diperlukanlah suatu cara agar informasi tersebut tidak dapat diketahui oleh pihak lain. Salah satu caranya adalah dengan menggunakan kriptografi.

Kriptografi sudah dikenal sejak ribuan tahun yang lalu. Kriptografi terus-menerus dikembangkan hingga saat ini. Pengembangannya dilakukan oleh berbagai pihak dari berbagai negara. Karena banyaknya jumlah algoritma yang digunakan, diperlukanlah standar algoritma sehingga dapat dipergunakan dalam berbagai aplikasi. NIST (*National Institute of Standard and Technology*) mempublikasikan suatu algoritma pengenkripsian data baru untuk menggantikan algoritma DES (*Data Encryption Standard*) yang memiliki beberapa kelemahan. Algorima baru ini dinamakan AES (*Advanced Encryption Standard*) atau *Rijndael*. Algoritma ini diperoleh melalui kompetisi yang dilakukan pada tahun 1997. Proses seleksi ini amat ketat dan membutuhkan waktu yang cukup lama. Pada akhirnya, pada tanggal 2 Oktober 2000 terpilihlah algoritma *Rijndael* yang dibuat oleh Rijmen dan Daemen dari Belgia. Algoritma ini tepilih sebagai AES.

Meskipun masih baru, algoritma ini sudah dipergunakan pada berbagai aplikasi, salah satunya adalah untuk penyandian sandi-lewat (*password*). Penggunaan algoritma ini sudah sering kita lihat pada perangkat lunak untuk kompresi data. Dalam perangkat lunak tersebut, salah satu metode yang digunakan untuk mengenkripsi sandi-lewatnya adalah dengan algoritma AES.

2. KRIPTOGRAFI

2.1. Penjelasan

Kriptografi adalah ilmu sekaligus seni untuk menjaga kerahasiaan pesan dengan cara menyamarkannya menjadi bentuk tersandi yang tidak mempunyai makna [1]. Bentuk tersandi ini hanya dapat dibaca oleh pihak yang berhak membacanya. Pesan yang akan dirahasiakan sebelum disamarkan disebut plainteks, sedangkan pesan setelah disamarkan disebut chiperteks. Proses penyamaran plainteks ke chiperteks disebut enkripsi, sedangkan pengembalian chiperteks menjadi plainteks semula disebut dekripsi.

2.2. Sejarah

Kriptografi telah lama digunakan oleh tentara Sparta di Yunani sekitar tahun 400 SM. Mereka

menggunakan alat yang disebut *scytale*. Alat ini terbuat dari daun papirus yang dililitkan pada batang silinder. Pesan yang akan dikirim ditulis horizontal. Setelah ditulis, daun dilepaskan dari batang kemudian dikirimkan ke penerima. Penerima dapat membaca pesan tersebut setelah melilitkan kembali daun tersebut pada batang silinder dengan ukuran diameter yang sama. Teknik ini dikenal dengan nama transposisi chiper yang merupakan metode enkripsi tertua [1].

Pada zaman Romawi kuno, Julius Caesar juga menggunakan kriptografi untuk mengirimkan pesannya. Pesan yang ia kirimkan ditulis dengan mengganti alfabet dengan alfabet lain dengan kunci tertentu. Sang penerima tentu saja telah diberi tahu kunci tersebut. Cara menyandikannya adalah dengan mengganti semua susunan alfabet dengan alfabet yang posisinya berada setelah alfabet tersebut tergantung kunci. Sebagai contoh, Julius Caesar mengganti huruf a, b, dan c menjadi d, e, dan f [2].

Pada perang dunia kedua, Jerman menggunakan mesin untuk mengrenkripsi pesan yang dikirimkan Hitler ke tentaranya yang bernama mesin *enigma*. Jerman meyakini kode-kode enkripsi dari mesin tersebut tidak dapat dipecahkan karena memiliki sekitar 15 milyar kemungkinan untuk mendekripsikannya. Kenyataannya sekutu mampu mendekripsikannya sehingga mesin tersebut beberapa kali memgalami perubahan [2].

2.3. Algoritma Kriptografi

Berdasarkan kunci yang dipakai, algoritma kriptografi dibagi menjadi tiga macam.

2.3.1. *Hash Function*

Fungsi *hash* sering disebut sebagai fungsi satu arah (*one-way function*). Fungsi ini mengubah suatu input menjadi output, tetapi output tersebut tidak dapat dikembalikan menjadi bentuk semula. Salah satu manfaatnya adalah penggunaan sidik jari (*fingerprint*). Sidik jari digunakan sebagai identitas pengirim pesan. Fungsi lain adalah untuk kompresi dan *message digest*. Contoh algoritma fungsi ini adalah MD-5 dan SHA.

2.3.2. Asimetri

Pada algoritma ini, digunakan dua buah kunci yang berhubungan yang disebut dengan kunci umum dan kunci pribadi. Kunci umum dapat dipublikasikan sehingga pesan dapat dienkripsikan tetapi tidak dapat didekripsikan dengan kunci tersebut. Kunci pribadi hanya boleh digunakan oleh pihak yang berhak untuk mendekripsikan pesan yang terenkripsi. Algoritma yang menggunakan kunci umum dan publik ini antara lain *Digital Signature Algorithm* (DSA), *Rivest-Shamir-Adleman* (RSA), *Diffie-Hellman* (DH), dan sebegainya.

2.3.3. Simetri

Algoritma ini menggunakan kunci yang sama untuk mengenkripsi dan mendekripsi data. Untuk mendekripsikan data, penerima menggunakan kunci yang sama dengan kunci yang digunakan pengirim untuk mengenkripsi data. Contoh dari algoritma ini adalah *Data Encryption Standard* (DES), *International Data Encryption Algorithm* (IDEA), *Advanced Encryption Standard* (AES), dan sebagainya. Dalam makalah ini, algoritma AES akan dibahas lebih lanjut.

2.4. AES (*Advanced Encryption Standard*)

AES dipublikasikan oleh NIST (*National Institute of Standard and Technology*) pada tahun 2001 yang digunakan untuk menggantikan algoritma DES yang semakin lama semakin mudah untuk membobol kuncinya. AES diperoleh dari hasil kompetisi yang diadakan NIST pada tahun 1997. Pada tahap pertama, 15 peserta dari 21 peserta lolos ke tahap berikutnya berdasarkan penilaian tingkat keamanan, harga, algoritma, dan karakteristik implementasi. Sepuluh dari 15 peserta tersebut gugur pada tahap berikutnya karena dianggap kurang aman dan kurang efektif. Pada Agustus 1999 dipilih lima kandidat untuk seleksi akhir, yaitu Mars (IBM, Amerika Serikat), RSA (RSA corp., Amerika Serikat), Rijndael (Belgia), Serpent (Israel, Norwegia, dan Inggris), dan Twofish (Counterpane, Amerika Serikat). Pada tahap ini, NIST memberikan penilaian terhadap *general security*, implementasi *software*, ruang lingkup, implementasi *hardware*, implementasi atas serangan, enkripsi dan dekripsi, kemampuan kunci, kemampuan lain dan fleksibilitas, dan kepotensialan untuk tingkat intruksi paralel. Akhirnya, pada tanggal 2 Oktober 2000 terpilihlah algoritma *Rijndael*, yang dibuat oleh Dr. Vincent Rijment dan Dr. Joan Daemen, sebagai pemenang.

3. ALGORITMA AES (*ADVANCED ENCRYPTION STANDARD*) atau *Rijndael*

Dalam proses enkripsi input, diperlukanlah empat macam operasi yang dilakukan berulang-ulang dalam beberapa putaran dan menggunakan kunci *cipher*. Jumlah putaran yang digunakan algoritma ini ada tiga macam seperti pada tabel di bawah ini.

Tabel 1. Jumlah Putaran Pengoperasian AES [3]

Tipe	Panjang Kunci	Panjang Blok Input	Jumlah Putaran
AES-128	128 bit	128 bit	10
AES-192	192 bit	128 bit	12
AES-256	256 bit	128 bit	14

3.1. Operasi

Ada empat macam operasi yang dilakukan setiap putaran.

3.1.1. Transformasi Subtitusi *Byte*

Dalam operasi ini, setiap *byte* yang akan dienkripsi disubtitusikan dengan nilai *byte* lain dengan

menggunakan *S-box*. *S-box* dibuat dari *multiplicative inverse* dari angka yang diberikan dalam *Rijndael's finite field* yang kemudian ditransformasikan dengan *affine transformation* [4]:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Gambar 1: *Affine Transformation*

Hasilnya kemudian di-*xor* dengan 99_{10} atau $0x63_{16}$ atau 1100011_2. Operasi matriks dengan *xor* ini ekuivalen dengan persamaan:

$b'_i = b_i \oplus b_{(i+4)mod8} \oplus b_{(i+5)mod8} \oplus b_{(i+6)mod8} \oplus b_{(i+7)mod8} \oplus c_i$

dengan *b'*, *b*, dan *c* adalah array 8 bit dan nilai *c* adalah 01100011.

Proses tersebut menghasilkan masing-masing nilai dari elemen tabel *S-box* yang hasilnya sebagai berikut.

Tabel 2. *S-box*

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

Seperti yang telah diketahui sebelumnya, AES merupakan algoritma simetri, yang berarti tabel subtitusi yang dibutuhkan untuk mengenkripsi berbeda dengan untuk mendekripsi. Untuk acuan tersebut, digunakanlah tabel *S-box* inversi sebagai berikut.

Tabel 3. *S-box* Inversi

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	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

Sebagai contoh, input yang akan dienkripsikan adalah
95 95 08 19
4f 6b 5c 6e
c8 89 80 26
fc 75 4e 6c

Dengan menggunakan *S-box*, hasil dari operasi ini adalah
2a 2a 30 d4
84 7f 4a 9f
e8 a7 cd f7
b0 9d 2f 50

Jika hasil tersebut ingin dikembalikan ke nilai semula sebelum operasi, nilai-nilainya dapat disubtitusikan dengan menggunakan tabel *S-box* inversi.

Operasi transformasi subtitusi *byte* pada proses enkripsi dan dekripsi tidak dilakukan pada putaran pertama. Operasi ini hanya dilakukan pada putaran kedua hingga terakhir.

3.1.2. Transformasi Pergeseran Baris

Pada operasi ini, *byte-byte* pada setiap baris digeser secara memutar dengan pergeseran yang berbeda dari tiap-tiap baris. Setiap baris digeser dengan aturan tertentu untuk jenis panjang blok yang berbeda.

Baris pertama blok untuk semua jenis panjang blok (128, 196, dan 256 bit) tidak digeser. Baris kedua untuk semua jenis panjang blok digeser 1 ke kiri. Pergeseran baris ketiga dan keempat untuk panjang blok 128 dan 196 bit berbeda dengan 256 bit. Pada panjang blok 128 dan 196 bit, baris ketiga digeser ke kiri sebanyak dua kali dan baris keempat digeser ke kiri sebanyak tiga kali. Pada panjang blok 256 bit, baris ketiga digeser ke kiri sebanyak tiga kali dan baris keempat digeser ke kiri sebanyak empat kali [5]. Untuk lebih jelasnya, proses tersebut dapat dilihat sebagai berikut.

S

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

→

S'

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

Gambar 2: Operasi pada Blok 128 bit

S

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$	$S_{0,4}$	$S_{0,5}$	$S_{0,6}$	$S_{0,7}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$	$S_{1,4}$	$S_{1,5}$	$S_{1,6}$	$S_{1,7}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$	$S_{2,4}$	$S_{2,5}$	$S_{2,6}$	$S_{2,7}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$	$S_{3,4}$	$S_{3,5}$	$S_{3,6}$	$S_{3,7}$

↓

S'

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$	$S_{0,4}$	$S_{0,5}$	$S_{0,6}$	$S_{0,7}$
$S_{1,1}$	$S_{1,2}$	$S_{1,3}$	$S_{1,4}$	$S_{1,5}$	$S_{1,6}$	$S_{1,7}$	$S_{1,0}$
$S_{2,3}$	$S_{2,4}$	$S_{2,5}$	$S_{2,6}$	$S_{2,7}$	$S_{2,0}$	$S_{2,1}$	$S_{2,2}$
$S_{3,4}$	$S_{3,5}$	$S_{3,6}$	$S_{3,7}$	$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

Gambar 3: Operasi pada Blok 256 bit

Sebagai contoh, hasil operasi ini terhadap input yang nilainya adalah output dari hasil operasi subtitusi *byte* sebelumnya adalah sebagai berikut
2a 2a 30 d4
7f 4a 9f 84
cd f7 e8 a7
50 b0 9d 2f

3.1.3. Transformasi Percampuran Kolom

Transformasi ini mengoperasikan blok pada masing-masing kolomnya. Setiap kolom diperlakukan sebagai *four-term polynomial* dengan cara Galois Field (GF) (2^8) dan dimodulokan dengan x^4+1 dengan polinom tetap $a(x)$ [3], yaitu

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

Hal ini dapat dituliskan sebagai perkalian matriks sebagai berikut.

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

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

dengan c adalah letak kolom, sehingga hasilnya

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

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

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

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

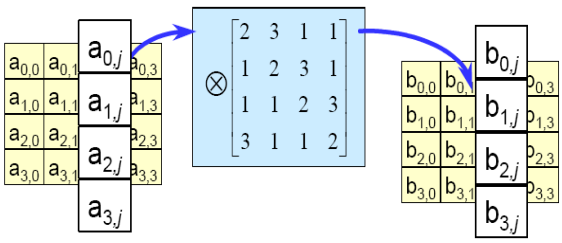
Jika hasil perkalian memiliki lebih dari 8 bit, bit yang lebih tidak begitu saja dibuang. Hasil tersebut di-*xor* dengan 100011011_2 [5]. Sebagai contoh, perkalian 11001010 dengan 11 dengan GF(2^8) akan berlangsung sebagai berikut.

11001010
11
---------- *
11001010
11001010
---------- xor
101011110
100011011
---------- xor
1000101

Nilai 1000101 merupakan hasil dari perkalian tersebut. Misalnya, jika dalam transfomasi ini input yang dipakai adalah hasil dari operasi pergeseran baris sebelumnya, hasil yang diperoleh adalah sebagai berikut.

48 cd af ac
c8 0c ab 1a
24 5e d8 74
6c b8 06 fa

Transformasi ini dapat diilustrasikan sebagai berikut [6].



Gambar 4: Ilustrasi Transformasi Percampuran Kolom

Operasi transformasi ini tidak digunakan dalam putaran terakhir, baik untuk enkripsi maupun dekripsi.

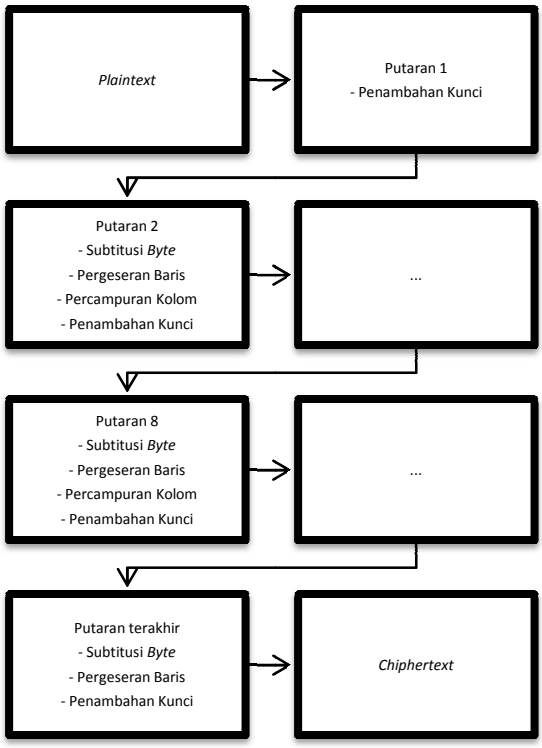
3.1.4. Transformasi Penambahan Kunci

Dalam operasi transformasi ini, digunakanlah upakunci untuk masing-masing putaran yang berasal dari kunci utama dengan menggunakan jadwal kunci Rijndael (*Rijndael's key schedule*) yang ukuran upakunci tersebut sama dengan ukuran blok yang akan diproses. Upakunci tersebut kemudian di-*xor* dengan blok input sehingga diperoleh hasilnya [7].

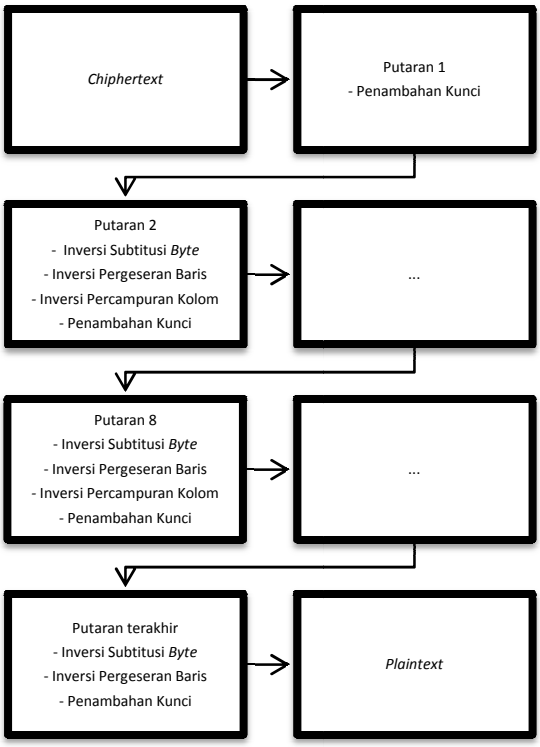
Sebagai contoh, jika inputnya adalah
a3 c5 08 08
78 a4 ff d3
00 ff 36 36
28 5f 01 02
dan diperoleh upakunci
36 8a c0 f4
ed cf 76 a6
08 a3 b6 78
31 31 27 6e
Maka, hasilnya adalah
a6 34 1a 00
24 dd f1 0e
62 a8 73 cf
48 b9 5d 61

3.2. Putaran

Seperti yang telah diketahui sebelumnya pada Tabel 1, jumlah putaran pengoperasian blok input untuk setiap macam panjang blok berbeda-beda. Akan tetapi, jumlah putaran untuk proses enkripsi dan dekripsi tetap sama. Proses enkripsi dan dekripsi dapat digambarkan sebagai berikut [2].



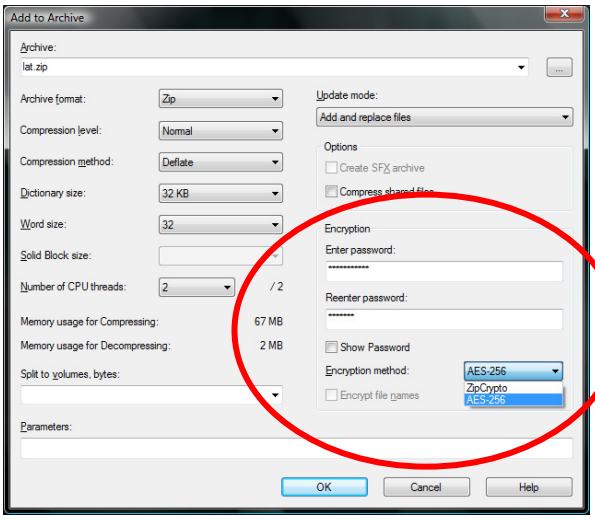
Gambar 5: Diagram Proses Enkripsi



Gambar 6: Diagram Proses Dekripsi

4. PENGGUNAAN AES DALAM ENKRIPSI DAN DEKRIPSI SANDI-LEWAT *FILE ZIP*

Dalam pembuatan sandi-lewat pada *file* yang sudah terkompresi, algoritma penyandian bukan digunakan untuk kunci sandi-lewatnya saja tetapi untuk *file* tersebut. Pada menu ketika suatu *folder* atau *file* yang telah disiapkan akan dikompresi, ada pilihan untuk memberikan sandi-lewat atau tidak, yang tampak pada gambar di bawah ini dengan mengguanakan aplikasi *7-zip*.



Gambar 7: Menu Pemilihan Penggunaan Sandi-lewat

Pada gambar 7 terdapat pilihan untuk mengenkripsi *file* dengan menggunakan sandi-lewat. Pada menu tersebut terdapat juga pemilihan metode enkripsi. Pada aplikasi *7-zip* pada gambar tersebut, pilihan metode enkripsi yang dapat digunakan adalah ZipCrypto dan AES-256. Metode enkripsi ZipCrypto lebih lemah dan tua daripada metode AES sehingga dianjurkan untuk menggunakan metode AES [8].

Dalam proses enkripsi, *file* yang telah dikompresi dienkripsi dengan sandi-lewat. Tentu saja ukuran hasil enkripsi tidak berbeda jauh dengan hasil kompresi sebelum dienkripsi. Hasil enkripsi pada *file zip* bertambah sedikit karena dalam format *file* enkripsi terdapat empat macam isi, yaitu nilai *salt*, nilai verifikasi sandi-lewat, kode autentikasi, dan data yang dienkripsi [9].

4.1. Nilai *Salt*

Nilai *salt* adalah barisan *byte* yang acak atau semu-acak yang dikombinasikan dengan sandi-lewat enkripsi untuk membuat data enkripsi dan kode autentikasi. Nilai ini dibuat oleh aplikasi pengenkripsi dan disimpan tanpa terenkripsi dengan *file* data.

Penyandian yang baik tentu saja menggunakan nilai *salt* yang berbeda meskipun sandi-lewatnya sama. Hal ini diperlukan karena jika sandi-lewat dan nilai *salt* pada kedua *file* sama, dapat terjadi kebocoran informasi akibat adanya pola hubungan antara sandi-lewat dengan nilai *salt*. Seseorang yang sudah mengetahui isi dari suatu *file* yang berisi sandi-lewat dan nilai *salt* yang sama dengan *file* lain dapat mengetahui juga sebagian atau seluruh *file* lain tersebut karena ia dapat mencocokkan polanya. Oleh karena itu, diperlukanlah usaha untuk membuat agar nilai *salt* dapat berbeda meskipun sandi-lewatnya sama [9].

Ukuran nilai *salt* tergantung panjangnya kunci enkripsi sebagai berikut.

Tabel 4. Ukuran Nilai *Salt*

Ukuran kunci	Ukuran Nilai *Salt*
128 bit	8 byte
192 bit	12 byte
256 bit	16 byte

4.2. Nilai Verifikasi Sandi-lewat

Nilai verifikasi ini berukuran dua byte. Nilai ini berasal dari proses enkripsi dan dekripsi dari sandi-lewat. Dalam pengenkripsian, nilai ini diperoleh dari sandi-lewat enkripsi dan disimpan dengan *file* yang telah dienkripsi dengan tanpa dienkripsi. Sebelum pendekripsian, nilai ini diperoleh dari sandi-lewat dekripsi dan dibandingkan dengan nilai yang tersimpan dengan *file* tadi. Ada satu dari $2^{(2x8)}$=65536 kemungkinan nilai verifikasi hasil proses sebelum pendekripsian sama dengan nilai verifikasi hasil pengenkripsian, yang tersimpan bersama *file*, sama. Meskipun sama, bukan berarti sandi-lewat yang digunakan untuk dekripsi adalah sandi-lewat yang benar [9].

4.3. Kode Autentikasi

Kode autentikasi diperoleh dari hasil enkripsi dan disimpan dalam *file* hasil enkripsi. Kode ini berukuran 10 byte. Kode ini melakukan pengecekan bahwa isi dari *file* tidak pernah diubah atau diinterferensi secara sengaja sejak pertama kali dienkripsi [9].

4.4. *File* data yang dienkripsi

Pengenkripsian *file* hanya dilakukan pada isi dari *file* tersebut saja. Proses ini dilakukan setelah pengompresian. Data *file* dienkripsi dengan metode AES dalam mode "CTR" sehingga ukuran data sebelum dienkripsi sama dengan hasil setelah dienkripsi.

5. KESIMPULAN

Algoritma AES (*Advanced Encryption Standard*) atau *Rijndael* merupakan algoritma simetri yang sangat cocok dipakai untuk berbagai keperluan yang berkaitan dengan kriptografi saat ini. Algoritma ini banyak dipakai untuk keperluan penyandian. Salah satunya adalah untuk penyandian sandi-lewat untuk *file* kompresi. Algoritma ini tentu saja sangat sesuai dengan berbagai aplikasi kompresi, yang pada pembahasan di atas menggunakan kompresi *zip*, sehingga dapat digunakan di berbagai merk aplikasi kompresi.

DAFTAR REFERENSI

[1] Rinaldi Munir, *Matematika Diskrit*, Prodi Teknik Informatika ITB, 2006

[2] Dony Ariyus, *Kriptografi, Keamanan Data dan Komunikasi*, Graha Ilmu, 2006

[3] Specification for the Advanced Encryption Standard (AES),
URL: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
Tanggal akses: 31Desember 2008 22:06

[4] Rijndael S-box,
URL: http://en.wikipedia.org/wiki/Rijndael_S-box
Tanggal akses: 31 Desember 2008 21:58

[5] The Advanced Encryption Standard (Rijndael),
URL: http://www.quadibloc.com/crypto/co040401.htm
Tanggal akses: 31 Desember 2008 21:54

[6] Rijndael,
URL: http://csrc.nist.gov/archive/aes/rijndael/misc/nissc2.pdf
Tanggal akses: 31 Desember 2008 21:58

[7] Advanced Encryption Standard,
URL: http://en.wikipedia.org/wiki/Advanced_Encryption_Standard
Tanggal akses: 31 Desember 2008 22:04

[8] Guidance on Encryption of Email and Email Attachments,
URL: http://www.ucl.ac.uk/cert/EmailEncryption.html
Tanggal akses: 3 Januari 2009 16:26

[9] AES Encryption Information: Encryption Specification AE-1 and AE-2,
URL: http://www.winzip.com/aes_info.htm
Tanggal akses: 31 Desember 2008 22:19