BAB II
TINJAUAN PUSTAKA

2.1 Konsep Dasar Streaming

Menurut Sagita, *streaming* adalah sebuah teknologi untuk memainkan *file* video atau audio secara langsung ataupun dengan *prerecorded* dari sebuah mesin *server* (*web server*) (Sagita, 2011). Dengan kata lain, *file* video ataupun audio yang terletak pada sebuah *server* dapat secara langsung dijalankan pada komputer *client* sesaat setelah ada permintaan dari *user*, sehingga proses *download* yang memakan waktu yang lama dapat dihindari tanpa harus melakukan proses penyimpanan terlebih dahulu .

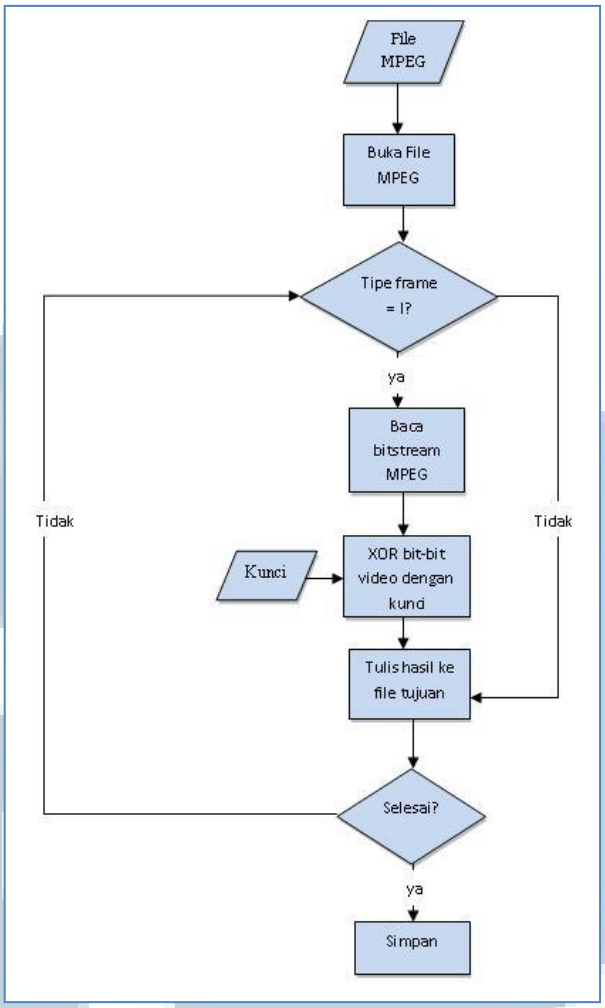
Saat *file* video atau audio di-*stream* maka akan terbentuk sebuah *buffer* di komputer *client* dan data video atau audio tersebut akan mulai di *download* ke dalam *buffer* yang telah terbentuk pada mesin *client*. Dalam waktu sepersekian detik, *buffer* telah terisi penuh dan secara otomatis *file* video atau audio akan dijalankan oleh sistem. Sistem akan membaca informasi dari *buffer* sambil tetap melakukan proses unduh *file* sehingga proses *streaming* tetap berlangsung ke mesin *client*. (Azikin, 2005).

2.2 Video Streaming

Video *streaming* merupakan suatu metode yang memanfaatkan *streaming server* untuk melakukan transmisi video digital melalui jaringan data sehingga memungkinkan video playback langsung dijalankan tanpa perlu menunggu sampai proses unduh selesai ataupun menyimpan terlebih dahulu video tersebut di komputer klien (Lestariningati, Zarman, & Perdana, 2011).

2.3 VEA

Algoritma VEA (Video Encryption Algorithm) merupakan algoritma enkripsi video yang hanya beroperasi pada sign bits dari koefisien DCT pada *frame* I dari sebuah *file* video (Daniel, 2004). Algoritma VEA akan menghasilkan sebuah kunci rahasia *k* secara *random* dalam bentuk *bitsream* dengan panjang *m* yang dapat dituliskan sebagai *k = b1b2...bm.*



Gambar 2.1 Diagram Alir Algoritma VEA untuk MPEG (Savitri, 2007)

Gambar 2.1 diatas merupakan gambaran bagaimana proses enkripsi dengan algoritma VEA. Berikut adalah proses bagaimana algoritma VEA berjalan:

1. Buka sebuah *file* video MPEG.
2. Baca *frame file* MPEG, baca tipe *frame*-nya.
3. Baca *stream* bit dari *frame* tersebut.
4. Jika *frame* dari *stream* bit bukan *frame I*, maka *stream* bit langsung ditulis ke file tujuan.
5. Jika *stream* bit tersebut merupakan *stream* bit dari *frame I*, maka *bit-bit* tersebut di-XOR-kan dengan kunci.
6. Tulis hasil enkripsi ke *file* tujuan.

7. Baca *frame* selanjutnya, kembali ke langkah nomor 2 sampai *End-of-File.*

VEA diciptakan dengan tujuan untuk melakukan pengamanan informasi yang berupa gambar video berformat *Moving Picture Experts Group* (MPEG) tetapi sebenarnya algoritma sistem kriptografi ini juga dapat diterapkan pada gambar berformat lainnya (Barmawi, Syakrani, Faren, & Budianto, 2009). Kriptografi ini ditujukan untuk menjamin agar informasi yang dikirimkan tidak dapat dibuka oleh pihak lain yang tidak berhak membaca. Gagasan dasar algoritma ini adalah melakukan proses pengamanan informasi dengan memanfaatkan kunci rahasia. Secara umum, algoritma sistem kriptografi VEA sangat sederhana yaitu meng-XOR-kan kunci rahasia dengan setiap koefisien *Discrete Cosine Transform* (DCT) yang terdapat pada MPEG.

Gambar 2.2 Contoh Hasil Enkripsi VEA (Ramsky, 2009)

VEA merupakan sebuah algoritma enkripsi video yang berbasis pada *stream cipher*. Adapun secara lebih rinci dapat dijelaskan sebagai berikut. Misalnya, sebuah informasi (*plain-text*) berupa video S yang dapat direpresentasikan dalam Rumus 1.1. (Barmawi, Syakrani, Faren, & Budianto, 2009)

$$S=s_1s_2s_3.....s_m \quad \text{.....} \quad \text{Rumus 1.1}$$

s_i adalah seluruh koefisien *Alternating Current* (AC)(informasi) dan *Direct Current* (DC) (nilai rata-rata *brightness*) yang akan diamankan (dienkripsi) menggunakan kunci K yang dapat direpresentasikan pada Rumus 1.2. (Barmawi, Syakrani, Faren, & Budianto, 2009).

$$K=k_1k_2k_3.....k_m \quad \text{.....} \quad \text{Rumus 2.2}$$

k_i adalah *bit* ke-i dari kunci rahasia yang akan digunakan. Hasil enkripsinya $E_k(S)$ dapat dilihat pada Rumus 1.3. (Barmawi, Syakrani, Faren, & Budianto, 2009).

$$E_k(S)= (k_1 \oplus s_1) (k_2 \oplus s_2) (k_3 \oplus s_3) ...(k_m \oplus s_m) \quad \text{....} \quad \text{Rumus 3.3}$$

2.4 Kriptografi

Kriptografi (*crypthography*) berasal dari Bahasa Yunani: "*cryptos*" artinya "*secret*"(rahasia), sedangkan "*graphein*" artina "*writing*"(tulisan). Jadi, kriptografi berarti "*secret writing*" (tulisan rahasia).

Kriptografi adalah ilmu sekaligus seni untuk menjaga kerahasiaan pesan dengan cara menyamarkannya menjadi bentuk tersandi yang tidak mempunyai makna (Munir, Matematika Diskrit, 2006). 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 disebut dekripsi.

2.5 Algoritma Kriptografi

Berdasarkan kunci yang digunakan , algoritma kriptografi terbagi menjadi tiga macam (Adiwidya, 2009).

2.5.1 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. Sidik jari digunakan sebagai identitas pengirim pesan. Fungsi lain adalah untuk kompresi dan *message digest*. Contoh algoritma fungsi ini adalah MD-5 dan SHA (Adiwidya, 2009).

2.5.2 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 yangterenkripsi. Algoritma yang menggunakan kunci umum dan publik ini antara lain *Digital Signature Algorithm* (DSA), *Rivest-Shamir-Adleman* (RSA), *Diffie-Hellman (*DH), dan sebagainya (Adiwidya, 2009).

2.5.3 Simetri

Algoritma ini menggunakan kunci yang sama untuk mengenkripsi dan dekripsi 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. Teori mengenai algoritma AES akan dibahas lebih lanjut (Adiwidya, 2009).

2.6 Algoritma Advanced Encryption Standard (AES)

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 (Adiwidya, 2009).

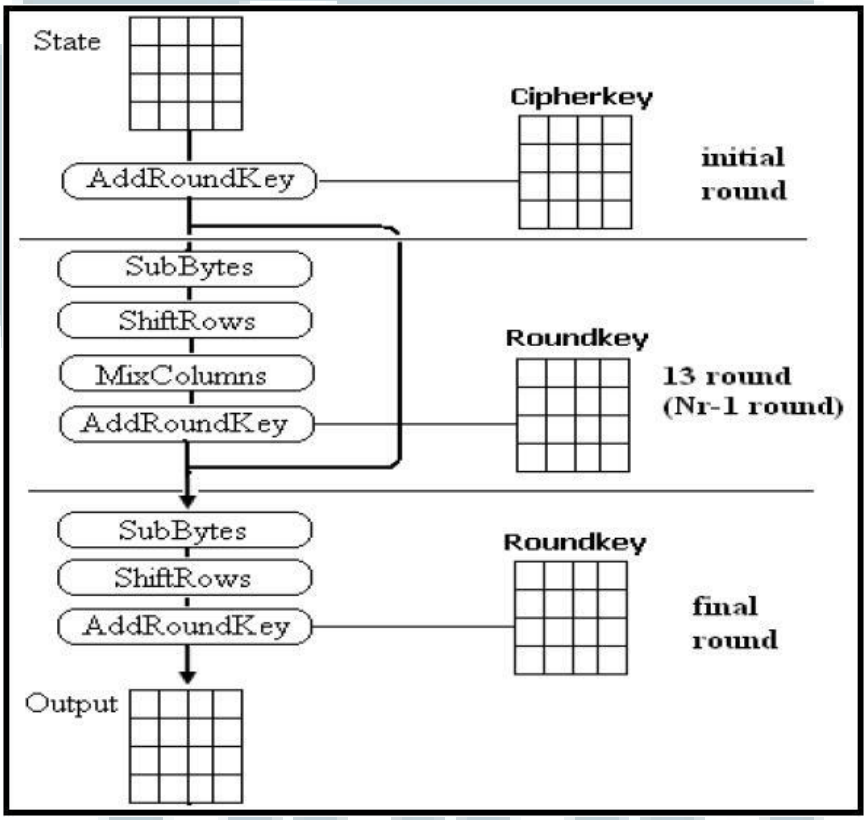
AES menetapkan panjang kuncinya 128,192, dan 256 bit. Karena itu, maka dikenal AES-128, AES-192, dan AES-256. Table 2.1 merangkum perbedaan ketiga versi AES tersebut.

Tabel 2.1 Jumlah Putaran Pengoperasian AES (Technology, 2012)

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

2.6.1 Proses Enkripsi AES

Menurut Rinaldi Munir (Munir, Kriptografi, 2006),algoritma AES menggunakan subtitusi dan permutasi, dan sejumla putaran (cipher berulang), dimana setiap putaran menggunakan kunci yang berbeda (kunci setiap putaran disebut *round key*).



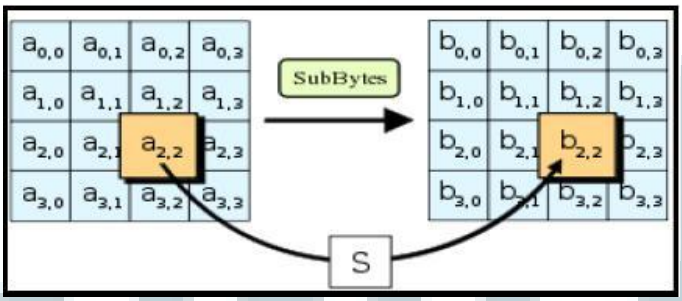
Gambar 2.3 Diagram Proses Enkripsi AES (Munir, Kriptografi, 2006)

Garis besar Algoritma AES yang beroperasi pada blok 128 bit dengan kunci 128 bit (Gambar2.3) adalah sebagai berikut(di luar proses pembangkitan *round key*) (Munir, Kriptografi, 2006):

1. *AddRoundKey* : melakukan *XOR* antara *state* awal (plainteks) dengan *cipher key*. Tahap ini juga disebut *initial round*. Putaran sebanyak *Nr* – 1 kali. Proses yang dilakukan pada setiap putaran.
2. *SubBytes* : subtitusi *byte* dengan menggunakan tabel subtitusi (S-Box). Tabel 2.2 merupakan tabel S-Box *SubBytes*. Untuk setiap *byte* pada *array state*, misalkan S[r,c] = xy yang dalam hal ini xy adalah digit heksadesimal dari nilai S[r,c], maka nilai subtitusinya yang dinyatakan dengan S'[r,c] adalah elemen di dalama *S-Box* yang merupakan perpotongan baris x dengan kolom y. Gambar 2.4 merupakan gambar proses transformasi *SubBytes.*

Tabel 2.2 S-Box *SubBytes* (Technology, 2012)

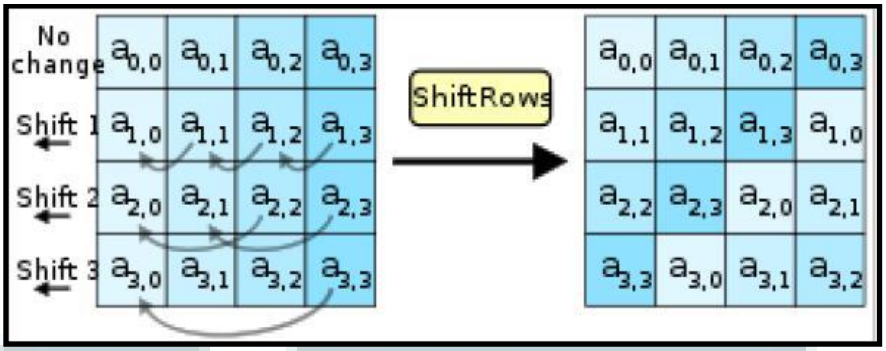
x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed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



Gambar 2.4 Transformasi *SubBytes* (Munir, Kriptografi, 2006)

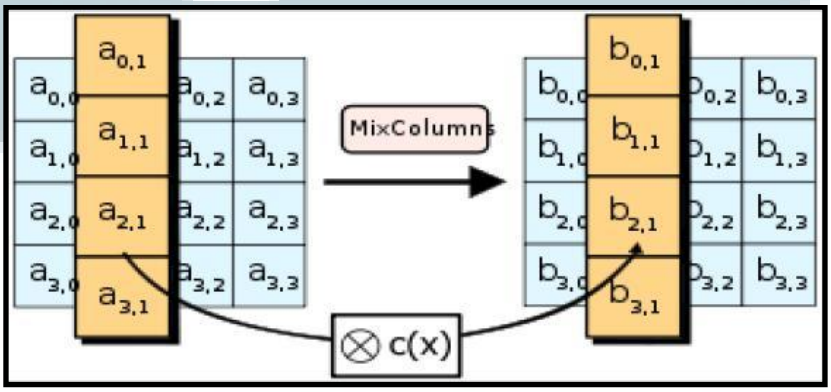
3. *ShiftRows* : pergeseran baris-baris *array state* secara *wrapping* pada 3 baris terakhir dari *array state*, dimana pada proses ini *bit* paling kiri akan dipindahkan menjadi *bit* paling kanan (rotasi *bit*). Jumlah pergeseran

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 tida digeser.



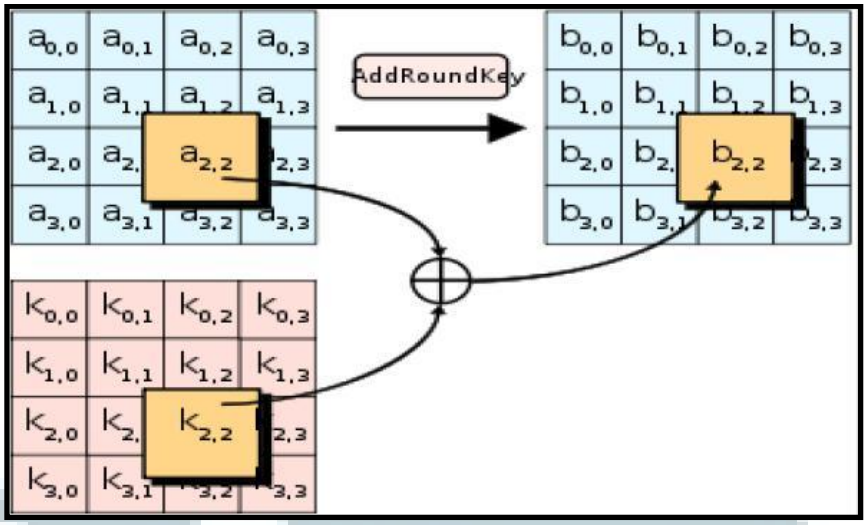
Gambar 2.5 Transformasi *ShiftRows* (Munir, Kriptografi, 2006)

4. *MixColumns* : mengacak data di masing-masing kolom *array state* (Gambar 2.6). Dalam proses MixColumn terdapat beberapa perkalian, yaitu *Matrix Multiplication* dan *Galis Field Multiplication.*



Gambar 2.6 Transformasi *MixColumn* (Munir, Kriptografi, 2006)

5. *AddRoundKey* : melakukan *XOR* antara *state* sekarang dengan *round key*.

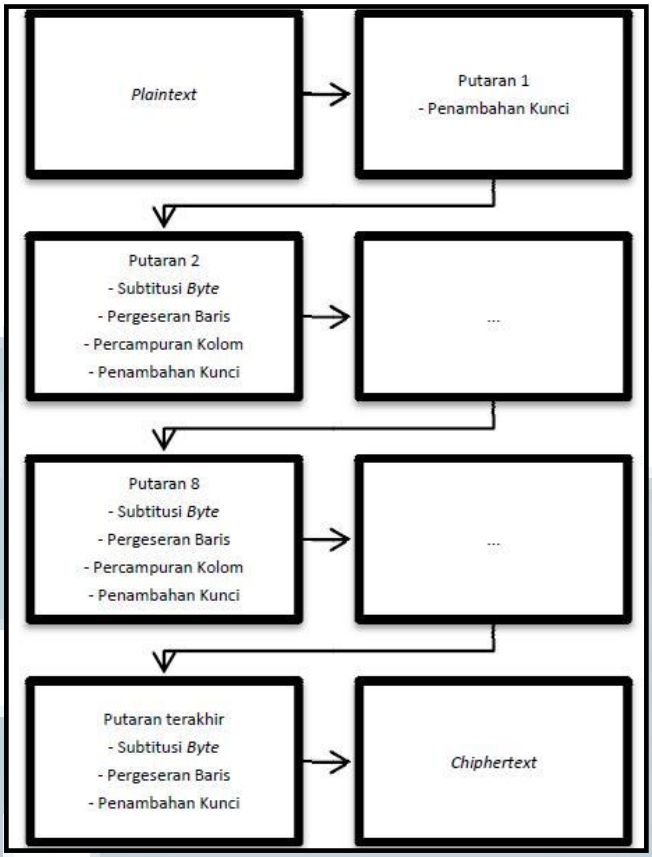


Gambar 2.7 Transformasi *AddRoundKey* (Munir, Kriptografi, 2006)

6. *Final Round* (proses untuk putaran terakhir): *SubBytes*, *ShiftRows*, dan *AddRoundKey*.

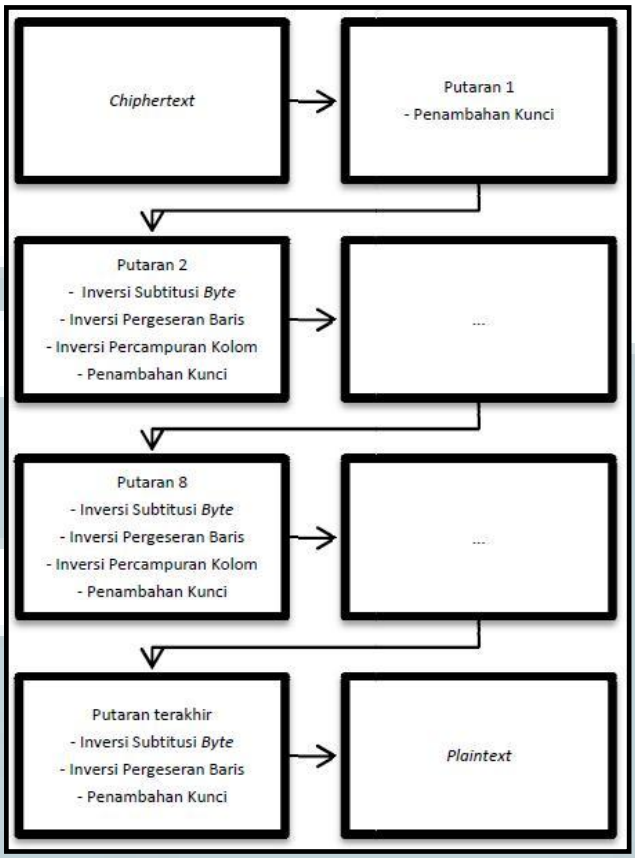
2.6.2 Putaran

Seperti yang telah diketahui sebelumnya pada Tabel 2.1, jumlah putaran pengoperasian blok input untuk setiap macam panjang blok berbeda-beda. Akan tetapi, jumlah putaran untuk proses enkripsi dan dekripsi tetap sama (Adiwidya, 2009). Proses enkripsi digambarkan pada gambar 2.8.



Gambar 2.8 Diagram Proses Enkripsi AES (Adiwidya, 2009)

Gambar 2.9 dibawah ini adalah proses dekripsi dengan algoritma AES



Gambar 2.9 Diagram Proses Dekripsi AES (Adiwidya, 2009)