Penggunaan Algoritma Rijndael dalam Standar Enkripsi

Muhammad Abdullah Munir 13516074
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13516074@std.stei.itb.ac.id

***Abstrak*—Keamanan merupakan suatu hal yang sangat penting untuk dikembangkan pada saat ini. Hal ini disebabkan banyaknya transaksi-transaksi yang dilakukan secara online serta adanya uang digital. Untuk mengamankan data yang dikirim secara online, pengirim sering menggunakan hash atau enkripsi. Beberapa diantaranya, base64, MD5, SHA-256, RSA, dan AES. Dengan adanya pengamanan ini, tindakan pencurian data akan terminimalisir. Makalah ini berisi pembahasan mengenai algoritma yang digunakan dalam enkripsi AES, serta mode enkripsi yang terdapat dalam AES.**

***Kata kunci*—kriptografi, pegamanan data, kriptoanalisis, rijndael, block cipher.**

I. PENDAHULUAN

Pada saat ini, perkembangan teknologi informasi berkembang dengan pesat. Hal ini ditunjukkan dengan meningkatnya pengguna internet. Internet dapat menghubungkan jutaan, bahkan milyaran orang untuk saling bertukar informasi. Akan tetapi, pernahkah terpikir jika informasi yang kita kirim dibaca oleh orang lain yang tidak berwenang. Oleh karena itu, keamanan sangat diperlukan.

Keamanan informasi merupakan hal yang sangat penting di pengiriman data. Pengamanan data dapat dilakukan dengan enkripsi maupun hash. Ilmu yang mempelajari tentang cara-cara pengamanan data biasa disebut dengan kriptografi.

Kriptografi ini banyak diaplikasikan di kehidupan, bahkan sudah lama kriptografi ini diterapkan. Salah satunya *Caesar cipher*, enkripsi ini digunakan oleh Julius Caesar untuk berkomunikasi dengan tentaranya. Akan tetapi, enkripsi ini terlalu sederhana sehingga masih bisa dilakukan dekripsi secara paksa. Sedangkan kriptografi pada zaman ini diperlukan untuk mengamankan data-data penting, seperti password, PIN ATM, serta dokumen-dokumen rahasia milik negara.

Kriptografi diharuskan berkembang dengan pesat agar tidak terjadi kebocoran informasi kepada pihak yang tidak berhak untuk mendapatkan informasi. Beberapa algoritma enkripsi yang sering digunakan saat ini ialah RSA dan *Advanced Encryption Standard* (AES). AES sendiri merupakan enkripsi baru yang menggantikan *Data Encryption Standard* (DES). AES ditetapkan sebagai standar enkripsi kunci-simetris yang telah diakui oleh *National Institute of Standards and Technology* (NIST) pada tahun 2001. AES menggunakan algoritma Rijndael yang dikembangkan oleh 2 orang asal belgia, Vincent Rijmen dan Joan Daemen.

Dengan adanya berbagai enkripsi saat ini, bisa dikatakan untuk sementara data yang dikirim melalui internet akan aman. Akan tetapi tiap metode enkripsi pasti memiliki kelemahan masing-masih dan masih memungkinkan untuk dilakukannya dekripsi secara paksa. Meskipun itu terkadang membutuhkan waktu yang lama.

II. TEORI DASAR

A. Kriptografi

Kriptografi ialah ilmu yang mempelajari tentang cara-cara menjaga keamanan pesan. Dengan cara mengubah data asli menjadi data yang terlihat tidak mempunyai makna.

Dalam makalah ini akan didefinisikan, pesan asli yang masih dapat dimengerti sebagai **plainteks**. Kemudian, pesan yang telah diproses sehingga terlihat tidak memiliki makna disebut **cipherteks**. **Enkripsi** ialah proses untuk mengubah plainteks menjadi cipherteks. Sedangkan, **dekripsi** memiliki arti kebalikan dari enkripsi, yaitu mengembalikan sebuah cipher teks ke plainteks semula. Jika dimisalkan sebuah fungsi E(x) yang merupakan fungsi enkripsi. Maka E(P) akan menghasilkan cipherteks yang sesuai dengan input P (plainteks) yang akan kita sebut C (ciphertext).

Jika didefinisikan D(x), yang merupakan fungsi dekripsi untuk fungsi enkripsi E(x). jika kita melakukan fungsi D(C) maka pasti akan menghasilkan sebuah string P, yang merupakan plainteks awal.

Untuk tipe-tipe enkripsi yang ada dalam kriptografi terdapat 3 tipe :

1. *Symmetric encryption*

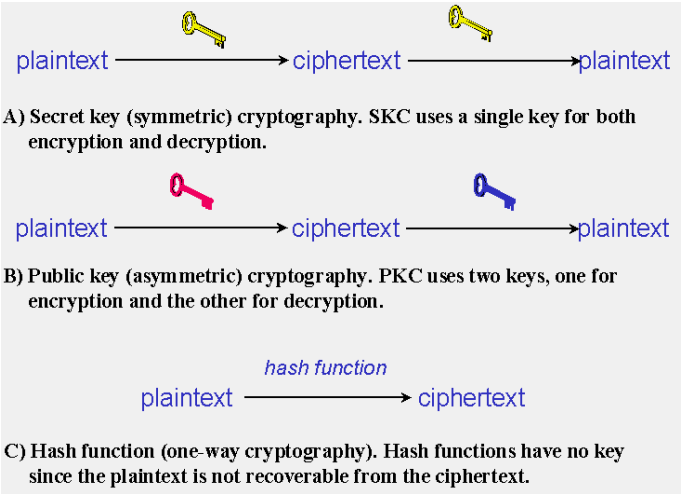
   Symmetric encryption ialah enkripsi yang hanya membutuhkan 1 kunci rahasia untuk melakukan enkripsi dan dekripsi. Jika kunci yang digunakan berbeda, maka hasil dari fungsi akan berbeda pula. Contoh dari enkripsi ini ialah AES yang akan dibahas di makalah ini.
2. *Asymmetric encryption*

   Asymmetric encryption ialah enkripsi yang

membutuhkan 1 pasang kunci yang sesuai. Contoh enkripsinya ialah RSA. RSA membutuhkan *public key* untuk mengenkripsi data dan membutuhkan *private key* untuk mendekripsi data.

3. *Hash*

Hash ialah fungsi enkripsi searah, yaitu hanya bisa menghasilkan cipherteks. Tidak ada fungsi untuk mengembalikan cipherteks ke plainteks semula. Pada fungsi *hash* masih terdapat kelemahan. Dalam fungsi hash dapat terjadi *collision*. *Collision* adalah suatu plainteks yang berbeda menghasilkan sebuah cipherteks yang sama. Contoh enkripsi yang menggunakan *hash* ialah MD5 dan SHA-256.



Gambar 2.1. Tiga jenis enkripsi dalam kriptografi. *(sumber : https://www.garykessler.net/library/crypto.html)*

B. Galois Field

Galois field atau bisa disebut *Finite field* digunakan untuk merepresentasikan bilangan byte pada algoritma Rijndael. Pada algoritma Rijndael sendiri, representasi yang digunakan ialah GF(2^8) atau yang selanjutnya disebut *Rijndael field*. Contohnya 0x53 (hex) = 0101 0011 (binary), dalam notasi *Rijndael field* akan menjadi $x^6 + x^4 + x^1 + 1$. Beberapa operasi aritmatika yang akan sering digunakan dalam algoritma Rijndael ialah :

1. Penjumlahan

Penjumlahan dilakukan dengan mengubah terlebih dahulu ke notasi *Rijndael field*, contoh :

0x53 + 0x21
= $(x^6 + x^4 + x^1 + 1) + (x^5 + 1)$
= $(x^6 + x^5 + x^4 + x^1)$
= 0x72

Jika kita perhatikan lagi operasi diatas sama dengan operasi XOR.

2. Perkalian

Sama seperti operasi penjumlahan, ubah dulu ke notasi *Rijndael field*. Kemudian kalikan seperti perkalian polinom. Jika hasilnya lebih besar dari 0xFF, maka harus di mod dengan m(x). m(x) untuk GF(2^8) ialah, m(x) = $(x^8 + x^4 + x^3 + x + 1)$ atau ekuivalen dengan 0x11b (hex). Untuk mengalikan dalam notasi ini cukup sulit dan tidak mudah untuk dipahami. Contoh :

0xb6 * 0x53
= $(x^7 + x^5 + x^4 + x^2 + x) * (x^6 + x^4 + x + 1)$
= $(x^{13} + x^{11} + x^{10} + x^8 + x^7) + (x^{11} + x^9 + x^8 + x^6 + x^5) + (x^8 + x^6 + x^5 + x^3 + x^2) + (x^7 + x^5 + x^4 + x^2 + x)$
= $(x^{13} + x^{10} + x^9 + x^8 + x^5 + x^4 + x^3 + x)$ …. (1)
m(x) * (x^5) = $(x^{13} + x^9 + x^8 + x^6 + x^5)$ ….(2)
m(x) * (x^2) = $(x^{10} + x^6 + x^5 + x^3 + x^2)$ ….(3)

Operasikan persamaan (1), (2), dan (3) akan didapatkan:

= $(x^5 + x^4 + x^2 + x)$ = 0x36

Atau untuk lebih mudahnya, dapat menggunakan *script* python.

```
# produce log and alog tables, needed for multiplying in the
# field GF(2^m) (generator = 3)
alog = [1]
for i in xrange(255):
    j = (alog[-1] << 1) ^ alog[-1]
    if j & 0x100 != 0:
        j ^= 0x11B
    alog.append(j)

log = [0] * 256
for i in xrange(1, 255):
    log[alog[i]] = i

# multiply two elements of GF(2^m)
def mul(a, b):
    if a == 0 or b == 0:
        return 0
    return alog[(log[a & 0xFF] + log[b & 0xFF]) % 255]
```

Gambar 2.2. Potongan kode untuk perkalian *galois field* *(sumber : https://gist.github.com/jeetsukumaran/1291836)*

3. Perkalian dengan konstanta k

Untuk mengalikan dengan konstanta k, kita bisa memecah k ke bentuk yang lebih sederhana atau hanya mengandung 1 bit saja. Misal 0x15 bisa dipecah menjadi (0x10 + 0x4 + 0x1). Dan untuk menghitung perkalian akan digunakan fungsi xtime.

```
def xtime(a):
    if (a & 0x80):
        return (((a << 1) ^ 0x1B) & 0xFF)
    else:
        (a << 1)
```

Gambar 2.3. Potongan kode xtime (*sumber : https://github.com/bozhu/AES-Python/blob/master/aes.py*)

Fungsi xtime mengembalikan hasil perkalian dengan 2. Untuk perkalian lebih dari 0x02 bisa menggunakan nested. Misal untuk dikali dengan 0x08 bisa ditulis xtime(xtime(xtime(x))) atau nested sebanyak n dari 2^n = pengali. Contoh :

0x57 * 0x13
= 0x57 * (0x10 ^ 0x02 ^ 0x01)
= (0x57 * 0x10) ^ (0x57 * 0x02) ^ (0x57 * 0x01)

= 0x07 ^ 0xAE ^ 0x57
= 0xFE

Cara ini dapat digunakan untuk mempermudah perkalian dibandingkan menggunakan cara sebelumnya.

C. *Rijndael S-Box*

Rijndael S-Box ialah tabel yang akan digunakan untuk proses subtitusi pada algoritma Rijndael.

	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. Rijndael S-Box
(sumber : https://www.ibm.com/developerworks/library/se-power8-in-core-cryptography/index.html)

Contoh penggunaan table ini ialah misal 0x9a akan disubtitusi ke tabel diatas akan diubah menjadi 0xB8.

III. RIJNDAEL

A. *Sejarah Rijndael*

Rijndael ialah sebuah algoritma enkripsi standar yang digunakan pada masa ini. Rijndael dikembangkan oleh 2 orang kriptografer yang berasal dari Belgia, yaitu Joan Daemen dan Vincent Rijmen.

Gambar 3.1. Joan Daemen (kiri) dan Vincent Rijmen (kanan)
(sumber : https://geek.hellyer.kiwi/plugins/end-to-end-encryption/)

Pada tahun 1997, National Institute of Standard and Technology (NIST) mengadakan sebuah kompetisi untuk membuat algoritma enkripsi, *Advanced Encryption Standard* (AES). Kompetisi ini diadakan untuk menggantikan standar enkripsi yang sebelumnya, *Data Encryption Standard* (DES) yang telah ditemukan celah-celah untuk dilakukannya kriptoanalisis. Algoritma AES menggunakan *block cipher* dengan ukuran minimal blok 128 bit, serta mendukung 3 ukuran kunci, yaitu 128 bit, 192 bit, dan 256 bit.

Pada bulan Agustus 1998, NIST mengumumkan ada 15 rancangan untuk AES yang telah diterima dan dievaluasi. Akan tetapi, pada tahun 1999 NIST mengumumkan hanya ada 5 algoritma yang lolos dari seleksi yang dilakukan. Lima algoritma itu adalah :

1. MARS
2. RC6
3. Rijndael
4. Serpent
5. Twofish

Pada bulan oktober 2000, NIST mengumumkan algoritma pemenang yang akan digunakan sebagai standar enkripsi untuk digunakan secara global. Algoritma tersebut ialah algoritma Rijndael.

B. *Algoritma Rijndael*

Algoritma Rijndael atau yang sekarang lebih dikenal sebagai AES, ialah algoritma *block cipher* yang menggunakan sistem subtitusi (S-Box). AES tergolong dalam *symmetric encryption* dikarenakan hanya memerlukan satu key untuk melakukan enkripsi maupun dekripsi. AES terbagi menjadi 3 berdasarkan panjang kuncinya, yaitu :

1. AES-128
2. AES-192
3. AES-256

Angka dibelakang kata AES menunjukkan panjang kunci yang digunakan untuk enkripsi. Sebelum penjelasan mengenai proses enkripsi, akan didefinisikan ***state*** sebagai hasil antara dari proses cipher, ***round*** ialah banyak pengulangan, serta ***cipher key*** ialah sebuah kunci dengan panjang yang dapat dibagi 32.

Banyaknya round ditentukan berdasarkan tabel di bawah.

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

Table 1: Number of rounds (Nr) as a function of the block and key length.

Tabel 3.1. Menghitung banyak *round*
(sumber : https://csrc.nist.gov/csrc/media/projects/cryptographic-standards-and-guidelines/documents/aes-development/rijndael-ammended.pdf)

Untuk mengisi tabel state digunakan fungsi KeyExpansion, dengan input *cipher key* dan output *round key*.

```
KeyExpansion(byte Key[4*Nk] word W[Nb*(Nr+1)])
{
      for(i = 0; i < Nk; i++)
            W[i] = (Key[4*i],Key[4*i+1],Key[4*i+2],Key[4*i+3]);

      for(i = Nk; i < Nb * (Nr + 1); i++)
      {
            temp = W[i - 1];
            if (i % Nk == 0)
                  temp = SubByte(RotByte(temp)) ^ Rcon[i / Nk];
            W[i] = W[i - Nk] ^ temp;
      }
}
```

Gambar 3.2. Fungsi KeyExpansion
(sumber : https://csrc.nist.gov/csrc/media/projects/cryptographic-standards-and-guidelines/documents/aes-development/rijndael-ammended.pdf)

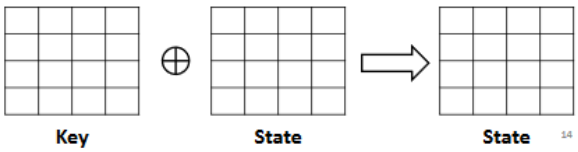
Keterangan :
Nb = *state column*
Nk = *cipher key column*
Nr = *number of rounds*

Sesuai dengan ukuran bloknya, AES bekerja pada blok berukuran 4x4 atau 128 bit. Untuk proses enkripsi pada awalnya plainteks akan diubah ke representasi hexadesimal. Operasi yang dilakukan dalam proses enkripsi pada AES ada 4, yaitu :

1. *Add Round Key*
2. *Sub Bytes*
3. *Shift Rows*
4. *Mix Columns*

Seluruh proses diatas diulang sebanyak jumlah round. Kecuali pada round terakhir tidak dilakukan *Mix Columns.* untuk lebih jelasnya, yang dilakukan tiap proses diatas ialah :

1. *Add Round Key*
   Pada proses ini dilakukan XOR antara *state* dengan *round key* yang telah dibuat berdasarkan *cipher key*. Hasil dari operasi ini disimpan untuk digunakan sebagai state yang baru
   - **AddRoundKey(State, Key)**:



Gambar 3.3. Proses *Add Round Key*
(sumber : https://www.lri.fr/~fmartignon/documenti/systemesecurite/5-AES.pdf)

2. *Sub Bytes*
   Yang dilakukan pada proses ini ialah menukar tiap byte yang ada dalam blok *state* awal dengan yang ada pada Rijndael S-Box sehingga terbentuk sebuah blok baru yang akan digunakan sebagai *state*. Contoh :

A					B			
53	45	53	54	==>	20	6E	ED	20
45	4E	4B	52		6E	2F	B3	00
49	50	53	49		3B	53	ED	3B
4b	41	54	41		B3	83	20	83

Tabel 3.2. Contoh proses *Sub-Bytes*

Keterangan :
B[0][0] = RijndaelS-Box[A[0][0]]
= RijndaelS-Box[53] = 20
Dst.

3. *Shift Rows*
   Seperti namanya, pada proses ini dilakukan *shift* atau penggeseran elemen tiap barisnya. Pada baris pertama tidak dilakukan penggeseran. Pada baris kedua dilakukan satu kali penggeseran. Pada baris ketiga dilakukan dua kali penggeseran. Terakhir, pada baris keempat dilakukan tiga kali penggeseran. Penggeseran ini dilakukan ke arah kiri. Contoh :

A					B			
20	6E	ED	20	==>	20	6E	ED	20
6E	2F	B3	00		2F	B3	00	6E
3B	53	ED	3B		ED	3B	3B	53
B3	83	20	83		83	B3	83	20

Tabel 3.3. Contoh proses *Shift-Rows*

4. *Mix Columns*
   Pada proses ini dilakukan pengalian blok dengan sebuah

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

matriks.

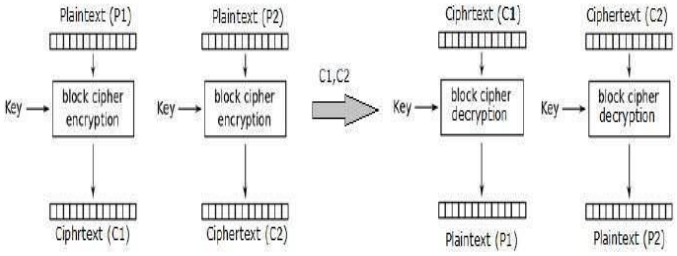
Gambar 3.4. Matriks *Mix-Column*
(sumber : https://csrc.nist.gov/csrc/media/projects/cryptographic-standards-and-guidelines/documents/aes-development/rijndael-ammended.pdf)

Perkalian yang digunakan pada operasi ini ialah perkalian dengan menggunakan Galois Field. Setelah dilakukan perkalian, akan terbentuk sebuah blok baru yang digunakan untuk enkripsi selanjutnya.

IV. MODE OPERASI BLOCK CIPHER

Berdasarkan penjelasan di atas, AES menggunakan sistem *block cipher* untuk melakukan proses enkripsi. Plainteks yang akan dienkripsi akan dibagi sesuai dengan ukuran blok yang digunakan oleh algoritma enkripsi. Jika pada AES, ukuran bloknya 128 bit. Jika plainteks tidak habis dibagi oleh ukuran blok, maka sisanya akan di ini dengan *padding*. Dalam menggunakan *block cipher*, harus ditentukan mode apa yang akan digunakan. Beberapa mode dari *block cipher* yaitu :

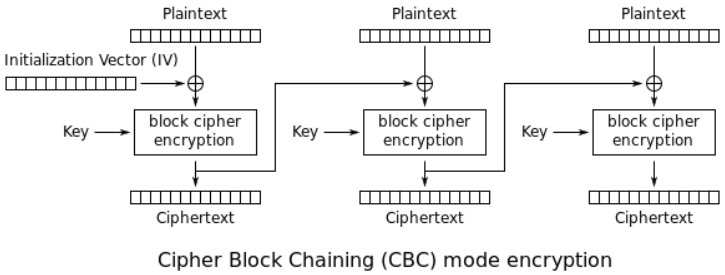
1. *Electronic Codebook Mode* (ECB)
   Mode ini ialah mode yang paling mudah dalam proses enkripsi. Sebuah blok plainteks akan dienkripsi menjadi sebuah blok cipherteks. Karena suatu blok plainteks selalu menghasilkan blok cipherteks yang tetap, maka dapat dibuat buku yang berisi pasangan plainteks dengan cipherteks. Akan tetapi, karena blok berukuran 128 bit maka terdapat banyak sekali kombinasi untuk disimpan.



Gambar 4.1. *Electronic Codebook* (ECB)
(sumber : https://www.tutorialspoint.com/cryptography/block_cipher_modes_of_operation.htm)

Pada mode ini, cipherteks akan lebih mudah untuk di dekripsi secara paksa. Misalnya jika ada kriptoanalis yang memiliki pasangan plainteks dan cipherteks, orang tersebut dapat mulai membuat kode buku tanpa harus mengetahui key yang digunakan untuk mengenkripsi plainteks tersebut dan dapat mengenkripsi data-data penting tertentu yang dikirim menggunakan key yang cocok dengan pasangan plainteks dan cipherteks tersebut.

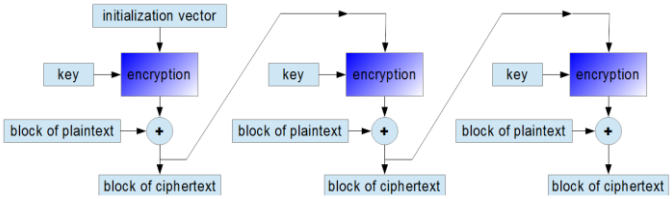
2. *Cipher Block Chaining* (CBC)
Pada mode CBC, plainteks yang akan di enkripsi akan diXOR terlebih dahulu dengan hasil enkripsi dari blok sebelumnya atau dengan *Initialization Vector* jika pada blok pertama. Sehingga hasil dari enkripsi yang memiliki key yang sama dapat menghasilkan cipherkey yang berbeda dikarenakan hasil enkripsi juga dipengaruhi oleh blok sebelumnya. Pada mode ini diperlukan *Initialization Vector* (IV) yang digunakan sebagai inisialisasi variabel. IV digunakan agar pesan



yang dihasilkan menjadi unik.
Gambar 4.2. *Cipher Block Chaining* (CBC)
(sumber : https://www.tutorialspoint.com/cryptography/block_cipher_modes_of_operation.htm)

Jika ditulis dalam rumus matematika, mode CBC sebagai berikut.
Untuk enkripsi :
$C_i = Ek(P_i \wedge C_{i-1})$
$C_0 = IV$
Untuk dekripsi :
$P_i = Dk(C_i) \wedge C_{i-1}$
$C_0 = IV$

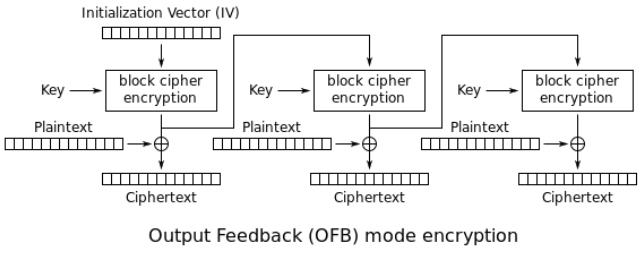
3. *Cipher Feedback* (CFB)
Mode ini hampir sama dengan mode CBC, akan tetapi jika di CFB blok cipherteks yang sebelumnya dienkripsi terlebih dahulu baru diXOR dengan plainteks. Untuk menggunakan mode CFB juga diperlukan *Initialization Vector*. IV tidak perlu untuk dirahasiakan.



Gambar 4.3. *Cipher Feedback* (CFB)
(sumber : https://www.tutorialspoint.com/cryptography/block_cipher_modes_of_operation.htm)

Jika ditulis dalam rumus matematika, mode CBC sebagai berikut.
Untuk enkripsi :
$C_i = Ek(C_{i-1}) \wedge P_i$
$C_0 = IV$
Untuk dekripsi :
$P_i = Ek(C_{i-1}) \wedge C_i$
$C_0 = IV$
Jika terjadi kesalahan 1 bit pada suatu blok, hal ini berpengaruh ke blok-blok selanjutnya pada proses enkripsi.

4. *Output Feedback* (OFB)
Mode OFB ini hampir mirip dengan mode CFB, perbedaannya terletak pada blok yang dienkripsi. Jika pada CFB, blok yang dienkripsi adalah blok cipherteks sebelumnya. Sedangkan pada OFB, blok yang dienkripsi adalah hasil output dari enkripsi sebelumnya, sebelum diXOR dengan plainteks. Pada mode ini juga diperlukan *Initialization Vector* untuk inisialisasi. Untuk lebih jelasnya lihat gambar di bawah.



Gambar 4.4. *Output Feedback* (OFB)
(sumber : https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation)

Jika ditulis dalam rumus matematika, mode OFB sebagai berikut.
$O_i = E_k(I_i)$
$I_i = O_{i-1}$
$I_0 = IV$
Untuk enkripsi :

$C_i = P_i \wedge O_i$
Untuk dekripsi :
$P_i = C_i \wedge C_{i-1}$

V. KESIMPULAN

Algoritma Rijndael yang sekarang digunakan sebagai Advanced Encryption Standard (AES) sudah bisa mengatasi permasalahan mengenai keamanan pengiriman data. Terlebih jika dikombinasikan dengan mode block cipher yang memiliki sedikit peluang untuk di *crack*, seperti mode CBC.

VII. UCAPAN TERIMAKASIH

Puji syukur kepada Allah SWT, atas izin-Nya penulis dapat menyelesaikan makalah ini. Terima kasih kepada Ibu Harlili, selaku dosen pengajar Matematika Diskrit yang telah memberikan materi selama satu semester ini sehingga penulis merasa tertarik untuk lebih mempelajari tentang kriptografi.

DAFTAR PUSTAKA

[1] Munir, Rinaldi. 2006. "Diktat Kuliah IF2120 Matematika Diskrit". Bandung: Penerbit Informatika ITB.
[2] Schneier, Bruce. 1996. "Applied Cryptography". New York: John Wiley & Sons, Inc.
[3] https://csrc.nist.gov/csrc/media/projects/cryptographic-standards-and-guidelines/documents/aes-development/rijndael-ammended.pdf
diakses pada 2 Desember 2017
[4] https://www.ibm.com/developerworks/library/se-power8-in-core-cryptography/index.html
diakses pada 2 Desember 2017
[5] https://www.lri.fr/~fmartignon/documenti/systemesecurite/5-AES.pdf
diakses pada 2 Desember 2017
[6] https://github.com/bozhu/AES-Python/blob/master/aes.py
diakses pada 2 Desember 2017
[7] https://www.garykessler.net/library/crypto.html
diakses pada 1 Desember 2017
[8] http://www.cs.utsa.edu/~wagner/laws/FFM.html
diakses pada 1 Desember 2017

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 3 Desember 2017

Muhammad Abdullah Munir
13516047