ALGORITMA KRIPTOGRAFI

AES-RSA

Disusun Oleh :

Alfiyah Widyaningrum	(201581041)
Andre Suntoro	(201581076)
Dina Rafikah Fauziah	(201581024)
Nanda Annisa Awalia	(201581039)
Ramdhan Yogi	(201581014)

FAKULTAS ILMU KOMPUTER

JURUSAN TEKNIK INFORMATIKA

UNIVERSITAS ESA UNGGUL

JAKARTA - INDONESIA

2017/2018

ALGORITMA AES

1. Definisi Algoritma AES

AES (*Advanced Encryption Standard*) merupakan algoritma *cryptographic* yang dapat digunakan untuk mengamankan data. Algoritma AES adalah blok *chipertext* simetrik yang dapat mengenkripsi (*enchiper*) dan deskripsi (*decipher*) informasi. Enkripsi merubah data yang tidak dapat dibaca lagi disebut *ciphertext*, sebaliknya deskripsi adalah merubah *ciphertext* data menjadi bentuk semula yang kita kenal sebagai *plaintext*. **Algoritma AES menggunakan kunci kriptografi 128, 192, dan 256 bit.**

	Panjang Kunci (*Nk words*)	Ukuran Blok (*Nb words*)	Jumlah Putaran (*Nr*)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Catatan: 1 *word* = 32 bit

Gambar - Panjang Kunci Algoritma Kriptografi AES

AES adalah lanjutan dari algoritma enkripsi standar DES (*Data Encryption Standard*) yang masa berlakunya dianggap telah usai karena faktor keamanan. Kecepatan komputer yang sangat pesat dianggap sangat membahayakan DES, sehingga pada tanggal 2 Maret tahun 2001 ditetapkanlah algoritma baru Rinjadael sebagai AES. **Kriteria pemilihan AES didasarkan pada 3 kriteria utama yaitu: keamanan, harga, dan karakteristik algoritma beserta implementasinya.** Keamanan merupakan faktor penting dalam evaluasi (minimal semanan Triple DES), yang meliputi ketahanan terhadap semua analisis sandi yang telah diketahui dan diharapkan dapat menghadapi analisis sandi yang belum diketahui. Di samping itu, AES juga harus dapat digunakan secara bebas tanpa harus membayar royalti, dan juga murah untuk diimplementasikan pada *smart card* yang memiliki ukuran memori kecil. AES juga harus efisien dan cepat (minimal secepat Triple DES) dijalankan dalam berbagai mesin 8 bit hingga 64 bit, dan berbagai perangkat lunak. DES menggunakan stuktur Feistel yang memiliki kelebihan bahwa struktur enkripsi dan dekripsinya sama, meskipun menggunakan fungsi F yang tidak invertibel. Kelemahan Feistel yang utama adalah bahwa pada setiap ronde, hanya setengah data yang diolah. Sedangkan AES menggunakan struktur SPN (*Substitution Permutation Network*) yang memiliki derajat paralelisme yang lebih besar, sehingga diharapkan lebih cepat dari pada Feistel.

Kelemahan SPN pada umumnya (termasuk pada Rijndael) adalah berbedanya struktur enkripsi dan dekripsi sehingga diperlukan dua algoritma yang berbeda untuk enkripsi dan dekripsi. Dan tentu pula tingkat keamanan enkripsi dan dekripsinya menjadi berbeda. AES memiliki blok masukan dan keluaran serta kunci 128 bit. Untuk tingkat keamanan yang lebih tinggi, AES dapat menggunakan kunci 192 dan 256 bit. Setiap masukan 128 bit plaintext dimasukkan ke dalam state yang berbentuk bujursangkar berukuran 4×4 byte. State ini di-XOR dengan key dan selanjutnya diolah 10 kali dengan subtitusi-transformasi *linear-Addkey*. Dan di akhir diperoleh *ciphertext*.

Berikut ini adalah operasi Rijndael (AES) yang menggunakan 128 bit kunci:

1. Ekspansi kunci utama (dari 128 bit menjadi 1408 bit)
2. Pencampuran *subkey*

3. Ulang dari i=1 sampai i=10 Transformasi : *ByteSub* (subtitusi per byte), *ShiftRow* (Pergeseren byte perbaris), *MixColumn* (Operasi perkalian GF(2) per kolom)
4. Pencampuran *subkey* (dengan XOR)
5. Transformasi : *ByteSub* dan *ShiftRow*
6. Pencampuran *subkey*

2. Sejarah AES

Pada rahun 1997, *National Institute of Standard and Technology* (NIST) *of United States* mengeluarkan *Advanced Encryption Standard* (AES) untuk menggantikan *Data Encryption Standard* (DES). AES dibangun dengan maksud untuk mengamankan pemerintahan di berbagai bidang. Algoritma AES di desain menggunakan blok *chiper* minimal dari blok 128 bit *input* dan mendukung ukuran 3 kunci (3-*key-sizes*), yaitu kunci 128 bit, 192 bit, dan 256 bit. Pada Agustus 1998, NIST mengumumkan bahwa ada 15 proposal AES yang telah diterima dan dievaluasi, setelah mengalami proses seleksi terhadap algoritma yang masuk**, NIST mengumumkan pada tahun 1999 bahwa hanya ada 5 algoritma yang diterima, algoritma tersebut adalah:**

a. MARS
b. RC6
c. Rijndael
d. Serpent
e. Twofish

Cipher	Variations
MARS	Key size 4-39 32-bit words
RC6	Word size *w*, no. of rounds *r*, key size 0-255 bytes
Rijndael	Block length of 128, 192 or 256 bits
Serpent	Key size 0..256 bits
Twofish	Key size 0..32 bytes

Gambar - Algoritma yang diterima oleh NIST

Algoritma-algoritma tersebut manjalani berbagai macam pengetesan. Pada bulan Oktober 2000, NIST mengumumkan bahwa **Rijndael** sebagai algoritma yang terpilih untuk standar AES yang baru. Baru pada Februari 2001 NIST mengirimkan *draft* kepada *Federal Information Processing Standards* (FIPS) untuk standar AES. Kemudian pada 26 November 2001, NIST mengumumkan produk akhir dari *Advanced Encryption Standard* (AES).

3. Metode Algoritma AES

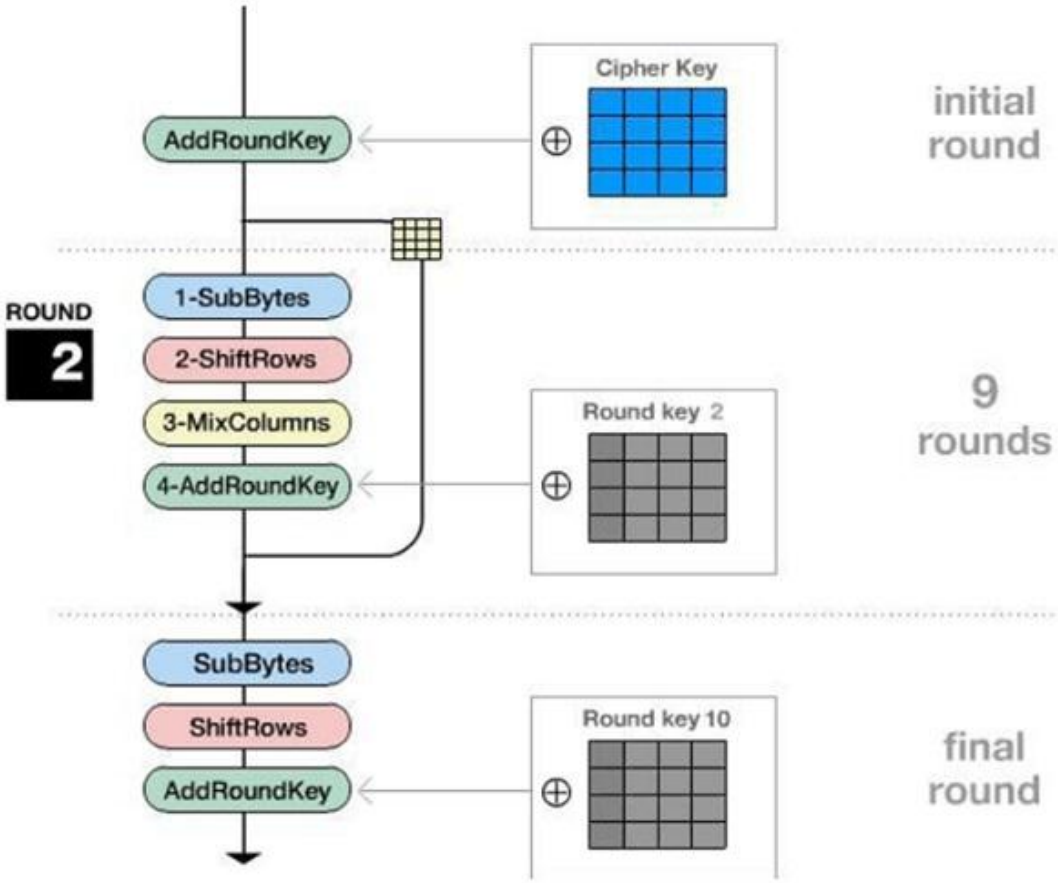
Algoritma kriptografi bernama Rijndael yang didesain oleh oleh Vincent Rijmen dan John Daemen asal Belgia keluar sebagai pemenang kontes algoritma kriptografi pengganti DES yang diadakan oleh NIST (*National Institutes of Standards and Technology*) milik pemerintah Amerika Serikat pada 26 November 2001. **Algoritma Rijndael inilah yang kemudian dikenal dengan *Advanced Encryption Standard* (AES).** Setelah mengalami beberapa proses standardisasi oleh NIST, Rijndael kemudian diadopsi menjadi standard algoritma kriptografi secara resmi pada 22 Mei 2002. Pada 2006, AES merupakan salah satu algoritma terpopuler yang digunakan dalam kriptografi kunci simetrik.

AES ini merupakan algoritma *block cipher* dengan menggunakan sistem permutasi dan substitusi (P-*Box* dan S-*Box*) bukan dengan jaringan Feistel sebagaimana *block cipher* pada umumnya. **Jenis AES terbagi menjadi 3, yaitu:**

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

Pengelompokkan jenis AES ini adalah berdasarkan panjang kunci yang digunakan. Angka-angka di belakang kata AES menggambarkan panjang kunci yang digunakan pada tiap-tiap AES. Selain itu, hal yang membedakan dari masing-masing AES ini adalah banyaknya *round* yang dipakai. **AES-128 menggunakan 10 round, AES-192 sebanyak 12 *round*, dan AES-256 sebanyak 14 *round*.**

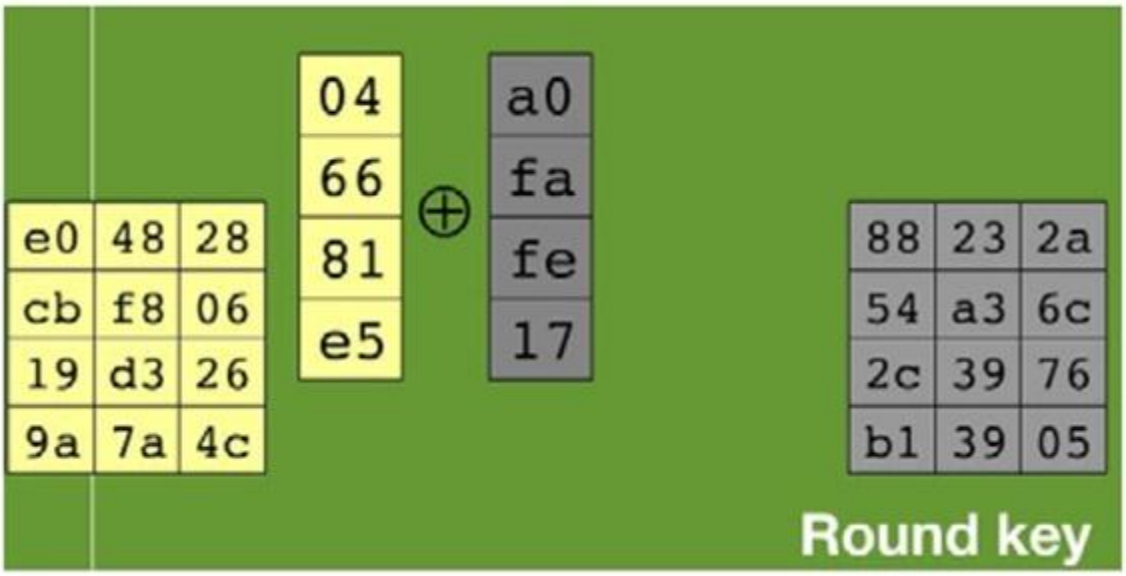
AES memiliki ukuran *block* yang tetap sepanjang 128 bit dan ukuran kunci sepanjang 128, 192, atau 256 bit. Tidak seperti Rijndael yang *block* dan kuncinya dapat berukuran kelipatan 32 bit dengan ukuran minimum 128 bit dan maksimum 256 bit. Berdasarkan ukuran *block* yang tetap, AES bekerja pada matriks berukuran 4x4 di mana tiap-tiap sel matriks terdiri atas 1 byte (8 bit). Sedangkan Rijndael sendiri dapat mempunyai ukuran matriks yang lebih dari itu dengan menambahkan kolom sebanyak yang diperlukan. Blok *chiper* tersebut dalam pembahasan ini akan diasumsikan sebagai sebuah kotak. Setiap *plainteks* akan dikonversikan terlebih dahulu ke dalam blok-blok tersebut dalam bentuk *heksadesimal*. Barulah kemudian blok itu akan diproses dengan metode yang akan dijelaskan. Secara umum metode yang digunakan dalam pemrosesan enkripsi dalam algoritma ini dapat dilihat melalui Gambar 1.



Gambar 1. Diagram AES

3.1 *Add Round Key*

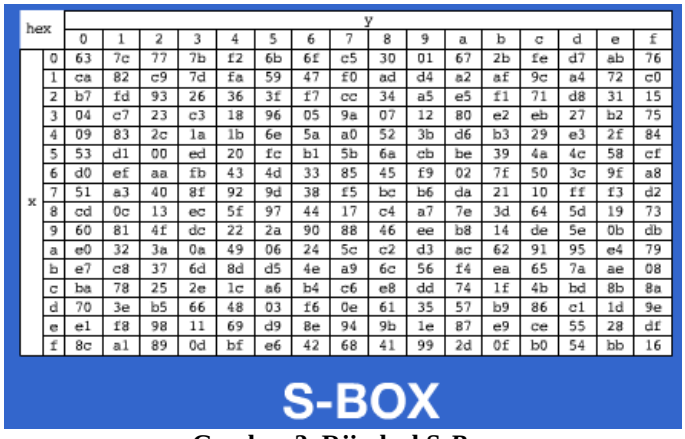
Add Round Key merupakan transformasi yang melakukan operasi XOR terhadap sebuah *round key* dengan *array state* dan hasilnya disimpan di *array state*.



Gambar 2. Ilustrasi *Add Round Key*

3.2 *Sub Bytes*

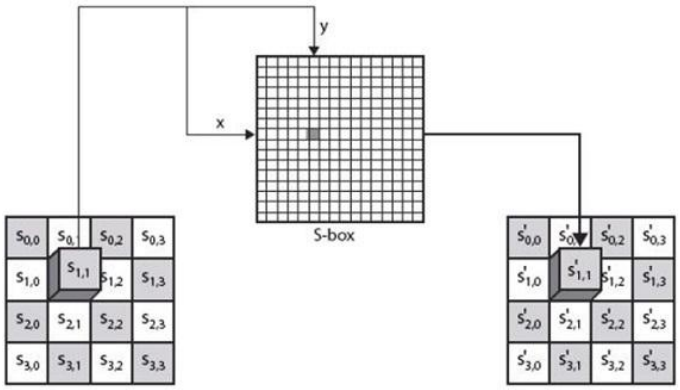
Prinsip dari *Sub Bytes* adalah menukar isi matriks/tabel yang ada dengan matriks/tabel lain yang disebut dengan **Rijndael S-*Box***. Dibawah ini adalah contoh *Sub Bytes* dan Rijndael S-*Box*.



hex	y															
x	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

S-BOX

Gambar 3. Rjindael S-*Box*



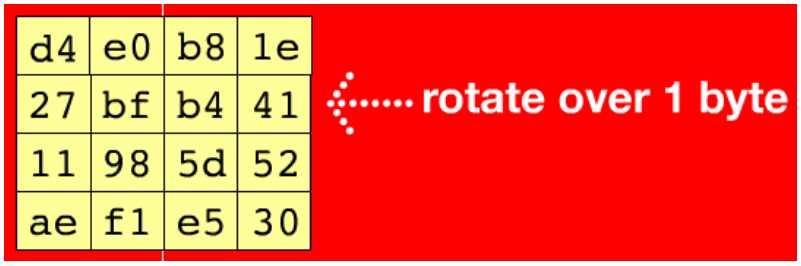
Gambar 4. Ilustrasi *Sub Bytes*

Keterangan: Gambar 4 adalah contoh dari Rijndael S-*Box*, di sana terdapat nomor kolom dan nomor baris. Seperti yang telah disebutkan sebelumnya, tiap isi kotak dari blok chiper berisi informasi dalam bentuk *heksadesimal* yang terdiri dari dua digit, bisa angka-angka, angka-huruf, ataupun huruf-angka yang semuanya tercantum dalam Rijndael S-*Box*. Langkahnya adalah mengambil salah satu isi kotak matriks, mencocokkannya dengan digit kiri sebagai baris dan digit kanan sebagai kolom. Kemudian dengan mengetahui kolom dan baris, kita dapat mengambil sebuah isi tabel dari Rijndael S-*Box*. Langkah terakhir adalah mengubah keseluruhan blok *chiper* menjadi blok yang baru yang isinya adalah hasil penukaran semua isi blok dengan isi langkah yang disebutkan sebelumnya.

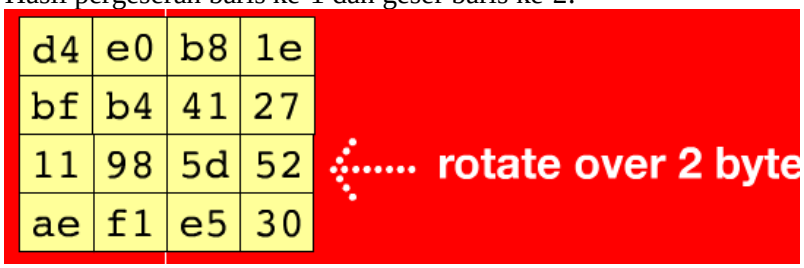
3.3 *Shift Rows*

Shift Rows seperti namanya adalah sebuah proses yang melakukan *shift* atau pergeseran pada setiap elemen blok/tabel yang dilakukan per barisnya. Yaitu baris pertama tidak dilakukan pergeseran, baris kedua dilakukan pergeseran 1 *byte*, baris ketiga dilakukan pergeseran 2 *byte*, dan baris keempat dilakukan pergeseran 3 *byte*. Pergeseran tersebut terlihat dalam sebuah blok adalah sebuah pergeseran tiap elemen ke kiri tergantung berapa byte tergesernya, tiap pergeseran 1 *byte* berarti bergeser ke kiri sebanyak satu kali. Ilustrasi dari tahap ini diperlihatkan oleh **Gambar 5. Ilustrasi Shift Rows** di bawah ini.

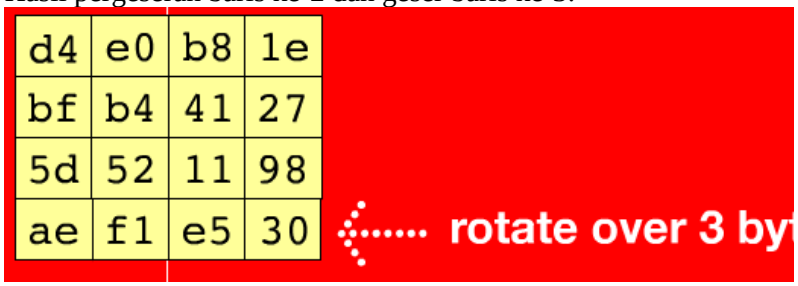
Geser baris ke-1:



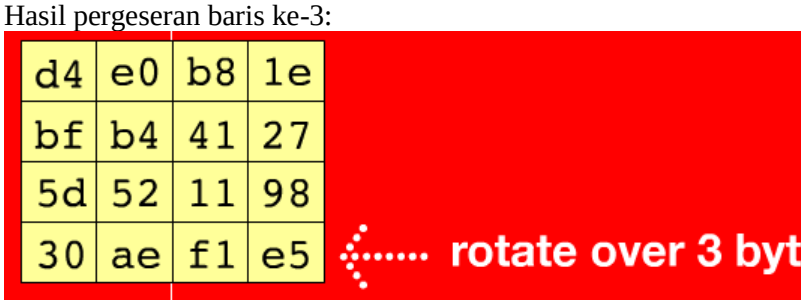
Hasil pergeseran baris ke-1 dan geser baris ke-2:



Hasil pergeseran baris ke-2 dan geser baris ke-3:



Hasil pergeseran baris ke-3:



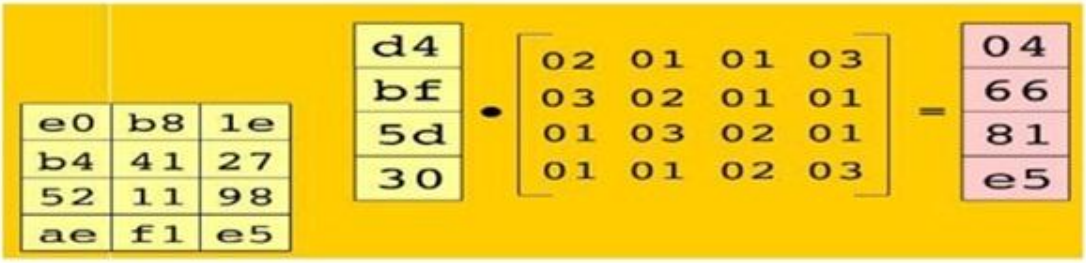
Seperti yang terlihat pada Gambar 5, tahap *Shift Row* sama sekali tidaklah rumit, karena ini adalah proses standar yang hanya berupa pergeseran. Langkah terakhir adalah *Mix Column*.

3.4 *Mix Column*

Yang terjadi saat *Mix Column* adalah mengalikan tiap elemen dari blok *chiper* dengan matriks yang ditunjukkan oleh Gambar 6. Tabel sudah ditentukan dan siap pakai. Pengalian dilakukan seperti perkalian matriks biasa yaitu menggunakan *dot product* lalu perkalian keduanya dimasukkan ke dalam sebuah blok chiper baru. Ilustrasi dalam Gambar 7 akan menjelaskan mengenai bagaimana perkalian ini seharusnya dilakukan. Dengan begitu seluruh rangkaian proses yang terjadi pada AES telah dijelaskan dan selanjutnya adalah menerangkan mengenai penggunaan tiap-tiap proses tersebut.

02	01	01	03
03	02	01	01
01	03	02	01
01	01	02	03

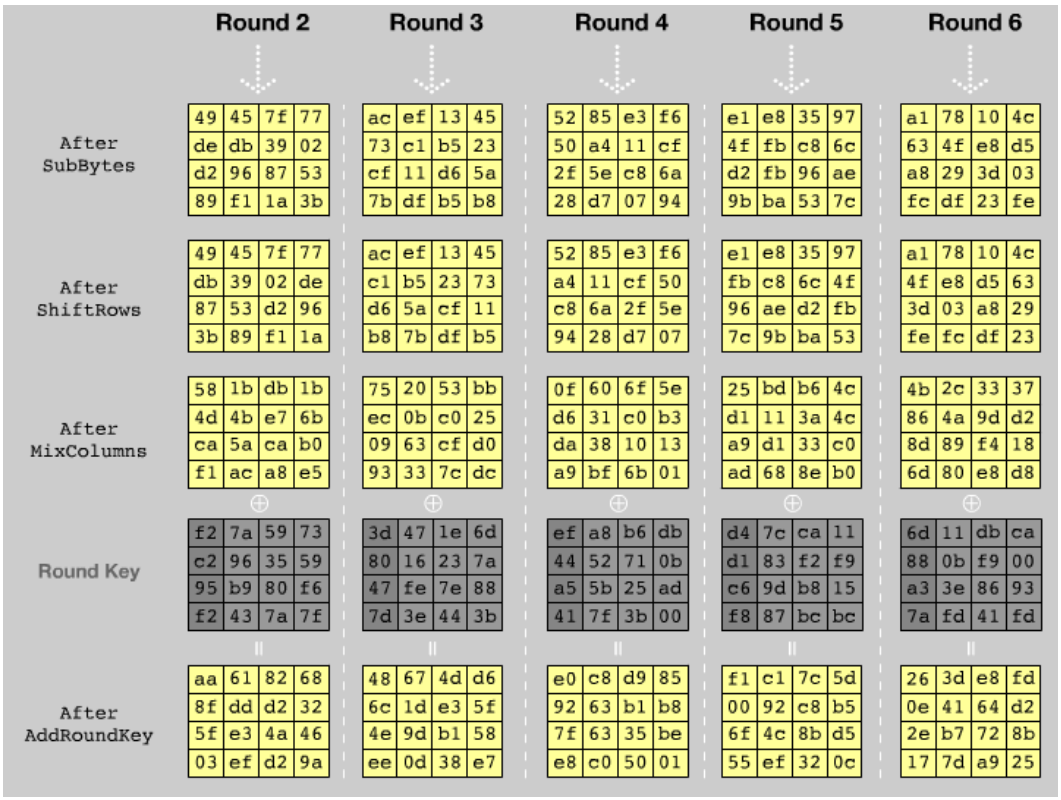
Gambar 6. Tabel untuk *Mix Column*



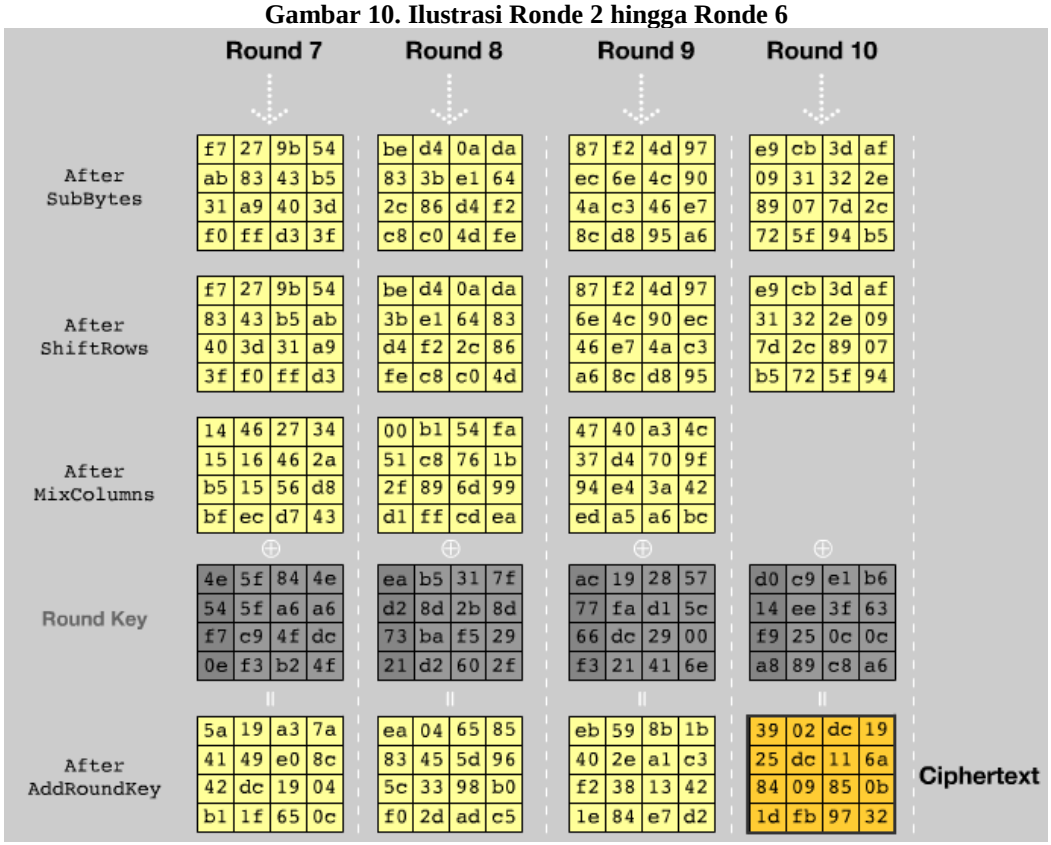
Gambar 7. Ilustrasi *Mix Column*

4. Diagram Alir AES

Kembali melihat diagram yang ditunjukkan oleh Gambar 1. Seperti yang terlihat semua proses yang telah dijelaskan sebelumnya terdapat pada diagram tersebut. Yang artinya adalah mulai dari ronde dua, dilakukan pengulangan terus menerus dengan rangkaian proses *Sub Bytes, Shift Rows, Mix Columns*, dan *Add Round Key*, setelah itu hasil dari ronde tersebut akan digunakan pada ronde berikutnya dengan metode yang sama. Namun pada ronde kesepuluh, proses *Mix Columns* tidak dilakukan, dengan kata lain urutan proses yang dilakukan adalah *Sub Bytes, Shift Rows*, dan *Add Round Key*, hasil dari *Add Round Key* inilah yang dijadikan sebagai *chiperteks* dari AES. Lebih jelasnya bisa dilihat dengan Gambar 8 dan 9 yang akan menerangkan mengenai kasus tersebut.



Gambar 10. Ilustrasi Ronde 2 hingga Ronde 6



Gambar 11. Ilustrasi Ronde 7 hingga Ronde 10

Dengan mengetahui semua proses yang ada pada AES, maka kita dapat menggunakannya dalam contoh kasus yang muncul di kehidupan sehari-hari.

5. Implementasi *Advanced Encryption Standard* (AES)

AES atau algoritma Rijndael sebagai salah satu algoritma yang penting tentu memiliki berbagai kegunaan yang sudah diaplikasikan atau diimplementasikan di kehidupan sehari-hari yang tentu saja membutuhkan suatu perlindungan atau penyembunyian informasi di dalam prosesnya. **Salah satu contoh penggunaan AES adalah pada kompresi 7-Zip.** Salah satu proses di dalam 7-Zip adalah mengenkripsi isi dari data dengan menggunakan metode AES-256. Yang kuncinya dihasilkan melalui fungsi *Hash*. Perpaduan ini membuat suatu informasi yang terlindungi dan tidak mudah rusak terutama oleh virus yang merupakan salah satu musuh besar dalam dunia komputer dan informasi karena sifatnya adalah merusak sebuah data.

Hal yang serupa digunakan pada WinZip sebagai salah satu perangkat lunak yang digunakan untuk melakukan kompresi. Tapi prinsip kompresi pun tidak sama dengan prinsip enkripsi. Karena kompresi adalah mengecilkan ukuran suatu data, biasanya digunakan kode Huffman dalam melakukan hal tersebut. Contoh penggunaan lain adalah pada perangkat lunak *DiskCryptor* yang kegunaannya adalah mengenkripsi keseluruhan isi disk/partisi pada sebuah komputer. Metode enkripsi yang ditawarkan adalah menggunakan AES-256, Twofish, atau Serpent.

6. Kesimpulan

Melindungi data dari serangan merupakan hal yang sulit. Salah satu cara untuk mengamankan data dari serangan adalah dengan menggunakan enkripsi. Salah satunya menggunakan metode enkripsi AES yang sudah dijabarkan dalam makalah ini. Dirancang untuk menggantikan DES (*launching* akhir 2001), menggunakan *variable length block chipper*, *key length*: 128-bit, 192-bit, 256-bit, dapat diterapkan untuk smart card. Algoritma Rijndael yang ditetapkan sebagai AES memiliki karakteristik yang istimewa yang menjadikannya mendapat status tersebut. Dalam hal ini pula maka algoritma ini perlulah untuk dipelajari karena penggunaannya di kehidupan sehari-hari sudah sangatlah banyak dan hal ini akan berguna dalam pengembangan dari teknologi kriptografi agar dapat menemukan terobosan-terobosan baru. Tujuan utama dari kriptografi adalah melindungi sebuah informasi, begitu pula dengan AES yang dengan serangkaian tahap atau ronde yang dilakukan dengan menggunakan kunci simetris. Penggunaan AES pun bukan hanya digunakan dalam hal yang sederhana melainkan perannya sangatlah krusial dalam sebuah perangkat lunak ataupun dalam hal lain dimana AES tersebut digunakan.

7. Daftar Pustaka

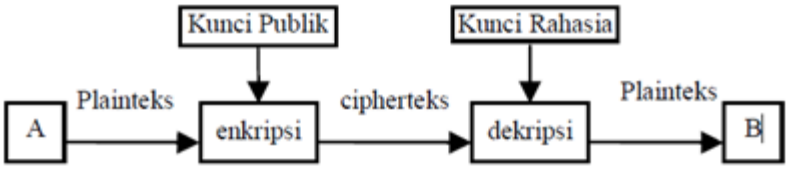
- Enkripsi Algoritma AES (*Advanced Encryption Standard*) http://kriptografijaringan.blogspot.co.id/2016/03/enkripsi-algoritma-aes-advanced.html

ALGORITMA RSA

1. Definisi Algoritma RSA

Sandi RSA merupakan algoritma kriptografi kunci publik (asimetris). Ditemukan pertama kali pada tahun 1977 oleh R. Rivest, A. Shamir, dan L. Adleman. Nama RSA sendiri diambil dari ketiga penemunya tersebut.

Sebagai algoritma kunci publik, RSA mempunyai dua kunci, yaitu kunci publik dan kunci rahasia. Kunci publik boleh diketahui oleh siapa saja, dan digunakan untuk proses enkripsi. Sedangkan kunci rahasia hanya pihak-pihak tertentu saja yang boleh mengetahuinya, dan digunakan untuk proses dekripsi. Keamanan sandi RSA terletak pada sulitnya memfaktorkan bilangan yang besar. Sampai saat ini RSA masih dipercaya dan digunakan secara luas di internet.



Gambar – Skema Algoritma Kunci Publik

Sandi RSA terdiri dari tiga proses, yaitu proses pembentukan kunci, proses enkripsi dan proses deskripsi. Sebelumnya diberikan terlebih dahulu beberapa konsep perhitungan matematis yang digunakan RSA.

2. Pembahasan

2.1 *Public Key – Cryptography*

Konsep fundamental dari *Public Key – Cryptography* ditemukan oleh Whitfield Diffie dan Martin Hellman, dan secara terpisah oleh Ralph Merkle.

Sedangkan konsep dasar *Public Key – Cryptography* terletak pada pemahaman bahwa *keys* selalu berpasangan: ***encryption key*** dan ***decryption key*****.** Juga perlu diingat bahwa sebuah *key* tidak dapat digenerate dari *key* lainnya. Pemahaman *encryption* dan *decryption key* sering disebut sebagai *public* dan *private key*. Seseorang harus memberikan public *key*-nya agar pihak lain dapat meng-*encrypt* sebuah pesan. *Decryption* hanya terjadi jika seseorang mempunyai *private key*.

2.2 *Scenario*

Bagian ini menjelaskan skenario bagaimana *public-key cryptosystem* bekerja. Kami akan menggunakan partisipan klasik Alice dan Bob sebagai orang-orang yang melakukan pertukaran informasi.

a. Alice dan Bob setuju untuk menggunakan *public-key cryptosystem*.
b. Bob mengirimkan *public key*-nya kepada Alice.
c. Alice meng-*encrypt* pesan yang dibuatkan dengan menggunakan *public key* milik Bob dan mengirimkan pesan yang sudah di-*encrypt* kepada Bob.
d. Bob meng-*encrypt* pesan dari Alice menggunakan *private key* miliknya.

2.3 *Mathematical Notation*

Untuk memahami algoritma RSA, seseorang harus memahami beberapa notasi matematika dasar, teori dan formula. Hal tersebut dibutuhkan untuk mendukung semua kalkulasi yang dilakukan dalam algoritma RSA.

a. Modulo (didenotasikan dengan 'x mod m' atau 'x % m' dalam beberapa bahasa komputer)

x % m = x mod m = pembagian x dengan m dan mengambil sisanya.

Contoh : 25 mod 5 = 0 karena 5 habis membagi 25
25 mod 4 = 1 karena 25 / (4 * 6) menyisakan 1
x mod m = x jika dan hanya jika x < m

b. GCD(A,B)

GCD adalah *Greatest Common Divisor*.

GCD(A,B) = D
GCD(78,32) = 2, karena tidak ada bilangan yang lebih besar dari dua yang membagi 78 dan 32.

GCD(A,B) dapat ditemukan dengan menggunakan algoritma *extended euclid*. Jika GCD(A,B) = 1 maka A and B adalah *coprime* satu sama lainnya (dengan kata lain, A dan B adalah *relatively prime*).

c. *Key Generation*

Misalkan Alice ingin Bob mengirimnya sebuah pesan melalui jalur yang aman. Alice akan memberikan *public key*-nya kepada Bob dan menyimpan *private key* untuk dirinya.

1) Pilih 2 bilangan prima besar seperti p,q dimana p tidak sama dengan q
2) Hitung M = p x q
3) Hitung phi(M) = phi(p) * phi(q)
4) Pilih sebuah integer 'e' dimana 1 < e < phi(M) dan 'e' serta phi(M) adalah *coprime*
5) Hitung 'd' integer sehingga (d * e) mod M = 1
6) (M,e) adalah public key dimana M adalah modulo dan e adalah eksponen *encryption*
7) (M,d) adalah private key dimana M adalah modulo dan d adalah eksponen *decryption*

d. *Decrypting Message*

Misalkan Alice menerima sebuah pesan ter-*encrypt,* ia akan men-*decrypt*-nya menggunakan tahapan-tahapan berikut ini:

1) Alice mempunyai *private key* dari langkah-langkah diatas (M,d)
2) N = C^d (mod M)
3) N adalah bilangan. Gunakan konversi table alphabet untuk mengubah N menjadi karakter yang direpresentasikan

3. Contoh Algoritma RSA

|----------|

Jika diketahui :

p = 3

q = 7

Penyelesaian :

n = p.q

= 3 x 7

= **21**

M = (p-1) (q-1)

= (3 -1)(7 – 1)

= **12**

e * d mod 12 = 1 (Cari bilangan prima yang jika mod M hasilnya 1)

e = 5

d = 17

Public Key = (e,n) = (5,21)

Private Key = (d,n) = (17,21)

S	I	S	J	A	R
83	73	83	74	65	82

|----------|

Proses Enkripsi dan Deskripsi

S = 83

Enkripsi	Deskripsi
C = M ^ e mod n	M = C ^ d mod n
8 ^ 5 mod 21 = 8	8 ^ 17 mod 21 = 8
3 ^ 5 mod 21 = 12	12 ^ 17 mod 21 = 3

I = 73

Enkripsi	Deskripsi
C = M ^ e mod n	M = C ^ d mod n
7 ^ 5 mod 21 = 7	7 ^ 17 mod 21 = 7
3 ^ 5 mod 21 = 12	12 ^ 17 mod 21 = 3

S = 83

Enkripsi	Deskripsi
C = M ^ e mod n	M = C ^ d mod n
8 ^ 5 mod 21 = 8	8 ^ 17 mod 21 = 8
3 ^ 5 mod 21 = 12	12 ^ 17 mod 21 = 3

J = 74

Enkripsi	Deskripsi
C = M ^ e mod n	M = C ^ d mod n
7 ^ 5 mod 21 = 7	7 ^ 17 mod 21 = 7
4 ^ 5 mod 21 = 16	16 ^ 17 mod 21 = 4

A = 64

Enkripsi	Deskripsi
C = M ^ e mod n	M = C ^ d mod n
6 ^ 5 mod 21 = 6	6 ^ 17 mod 21 = 6
4 ^ 5 mod 21 = 16	16 ^ 17 mod 21 = 4

R = 82

Enkripsi	Deskripsi
C = M ^ e mod n	M = C ^ d mod n
8 ^ 5 mod 21 = 8	8 ^ 17 mod 21 = 8
2 ^ 5 mod 21 = 11	11 17 mod 21 = 2

4. Kesimpulan

RSA merupakan contoh yang *powerful* dan cukup aman dari *Public-Key Cryptography*. Berdasarkan matematika, proses yang digunakan berdasarkan fungsi-fungsi *trap-door* satu arah. Sehingga melakukan *encryption* dengan menggunakan *public key* sangat mudah bagi semua orang, namun proses *decryption* menjadi sangat sulit.

Proses *decryption* sengaja dibuat sulit agar seseorang, walaupun menggunakan *Cray supercomputers* dan ribuan tahun, tidak dapat men-*decrypt* pesan tanpa mempunyai *private key*.

Perlu diingat juga bahwa pemilihan p*q = M haruslah sebuah bilangan yang sangat besar sehingga sulit dicari eksponen decoding-nya karena sulit melakukan pemfaktoran bilangan prima.

5. Daftar Pustaka

- Algoritma Kriptografi RSA
  http://ezine.echo.or.id/ezine12/echo12-05.txt