PERBANDINGAN ANALISIS SANDI LINEAR TERHADAP AES, DES, DAN AE1

Yusuf Kurniawan
Teknik Informatika, Universitas Pasundan Bandung
Email : ysfk2002@yahoo.com

ABSTRAK

AES (Advanced Encryption Standard) menjadi algoritma enkripsi standar Amerika sejak tahun 2001 untuk menggantikan DES (Data Encryption Standard) yang telah digunakan jutaan pengguna di seluruh dunia dalam beberapa dekade. AES dirancang agar dapat diimplementasikan pada berbagai platform software dan hardware dengan efisien. Sementara studi pembuatan algoritma enkripsi telah banyak mendapat perhatian di tanah air, penelitian tentang pembuktian keamanannya masih sangat sedikit. Pada makalah ini akan dibahas mengenai cara membuktikan keamanan AES menghadapi analisis sandi linear (ASL). Semua algoritma enkripsi bertaraf internasional yang baru dibuat hingga tahun terkini, selalu dianalisis dengan ASL sebagai dasarnya. Kemudian juga dibahas bagaimana analisis sandi linear bekerja terhadap DES dan AE1 (Algoritma Enkripsi 1). AE1 adalah algoritma yang pernah dibuat oleh peneliti. Makalah ini memberikan penjelasan bagaimana AE1 lebih baik dalam menghadapi ASL dibanding AES dan DES. Akan tetapi ini tidak menjamin bahwa AE1 lebih unggul dalam sifat kriptografi lainnya.

Kata kunci : *AES, DES, AE1, analisis sandi linear*

1. PENDAHULUAN

Kriptografi sebagai bagian dari keamanan komputer dan internet telah mendapat perhatian yang luas dari para peneliti di Indonesia. Namun sayangnya, sangat sedikit penelitian yang dilakukan berkaitan dengan pembuktian keamanan algoritma. Untuk membuktikan keamanan algoritma diperlukan ilmu yang disebut Analisis Sandi. Analisis sandi berusaha mendapatkan kelemahan suatu algoritma. Bila Analis sandi berhasil menemukan kelemahan algoritma yang tidak diketahui oleh pembuat algoritma, maka algoritma dinyatakan dapat dipecahkan. Artinya, meskipun suatu algoritma dinyatakan dapat dipecahkan dalam dunia akademik, bukan berarti setiap orang dapat memecahkannya dengan mudah.

Algoritma enkripsi yang menurut pembuatnya dinyatakan hanya dapat dipecahkan menggunakan brute force attack dengan mencoba 2^{128} kemungkinan kunci, bila ternyata kemudian ditemukan metode *attack* (analisis sandi) yang hanya membutuhkan kurang dari 2^{128} enkripsi, maka algoritma tersebut dinyatakan dapat dipecahkan, meskipun hanya secara teoretis.

Meskipun hanya secara teoretis, hal ini sangat penting mengingat terdapat kemungkinan bahwa di masa berikutnya, algoritma dapat dipecahkan secara realistis, mengingat tidak terdapat jaminan bahwa analisis sandi akan berhenti pada tahapan teoretis belaka.

Algoritma enkripsi perlu ditampilkan terbuka ke publik agar dalam kondisi apapun, selama kunci tetap aman, enkripsi akan tetap aman. Hal disebabkan karena ada kemungkinan orang akan mendapatkan software/hardware kita dan kemudian melakukan *reverse engineering* sehingga algoritma kita terlihat lawan. Jadi, bila seluruh pasang *plaintext/ciphertext* kita di masa lalu diketahui lawan sebanyak apapun, *cipher* kita yang sekarang harus tetap aman, selama kunci yang sedang kita pakai saat ini tidak mereka ketahui.

Demikian juga bila lawan mengetahui seluruh kunci yang pernah kita gunakan, selama mereka tidak mengetahui kunci yang kita gunakan saat ini, diharapkan data kita saat ini akan tetap aman. Artinya, kita hanya perlu menjaga kerahasiaan kunci, dan tidak perlu menjaga kerahasiaan algoritma, *plaintext/ciphertext* yang lalu yang sudah tidak bernilai lagi, dan seterusnya. Karena usaha untuk menyembunyikan banyak rahasia tentu lebih sulit dari pada hanya menjaga sedikit rahasia.

Bila seseorang diminta membuat sebuah algoritma enkripsi, sangat mungkin sekali dia akan berpikir untuk membuat algoritma enkripsi yang sangat komplek agar aman dari berbagai analisis sandi. Mungkin dia akan berpikir untuk menggunakan transformasi fourier, jaringan saraf tiruan, ataupun fungsi matematika yang sangat rumit lainnya. Hal ini mungkin benar, tapi hal ini bukan jaminan bahwa algoritmanya akan aman. Dan perlu diingat bahwa, pada umumnya, semakin rumit algoritma, semakin kurang efisien diimplementasikan pada berbagai platform. Dalam [1] dijelaskan bahwa kesulitan mendapatkan hubungan antara plaintext dan ciphertext bukan merupakan bukti ketiadaan relasi yang dapat digunakan untuk melakukan analisis sandi, khususnya analisis sandi linear dan diferensial. Karena sesuatu yang bagi seseorang sulit, mungkin bagi orang lain sangatlah mudah. Dan karena tidak ada jaminan bahwa algoritma yang rumit

pasti lebih aman daripada algoritma yang lebih sederhana, maka peneliti lebih senang menggunakan algoritma yang sederhana namun lebih terjamin keamanannya karena pembuktian keamanannya biasanya lebih mudah. Hal inilah salah satu yang mendasari dipilihnya algoritma Rijndael sebagai AES.

Pembahasan selanjutnya adalah sebagai berikut. Pada bab 2 akan dibahas cara kerja AES, bab 3 membahas analisis sandi linear (ASL) terhadap AES, bab 4 membahas ASL terhadap DES, bab 5 membahas AE1 dan analisis sandinya menggunakan ASL. Dan sebagai penutup adalah bab 6 yang berisi kesimpulan.

2. AES (ADVANCED ENCRYPTION STANDARD)

Pada tahun 2001, Algoritma Rijndael, karya peneliti dari universitas di Belgia ditetapkan menjadi AES. Rijndael merupakan algoritma yang dapat menerima masukan data 128 bit dan menghasilkan data 128 bit pula. Bila digunakan dengan kunci 128 bit, maka kita menyebutnya sebagai AES-128. Selain 128 bit, AES juga dapat menerima kunci 192 dan 256 bit. Penjelasan cara kerja AES secara matematis diperlukan, karena AES dibuat berdasar rumusan matematika khususnya dalam GF(2^8) Dalam tulisan ini akan dibahas AES-128 yang cara kerja enkripsinya secara formal adalah sebagai berikut [2]

$$m^{(0)} = m + k^{(0)} \quad (1)$$

$$m^{(t+1)} = \gamma \sum_{i=0}^{3}\sum_{j=0}^{3} \varphi(m_{i,j}^{(t)}) x^i y^{3i+j} + k^{(t+1)}$$

t= 0..8 (2)

$$c = m^{(10)} = \sum_{i=0}^{3}\sum_{j=0}^{3} \varphi(m_{i,j}^{(9)}) x^i y^{3i+j} + k^{(10)} \quad (3)$$

di mana kunci per ronde $k^{(t)} \in R$ dan pesan $m^{(0)}$ dienkrip untuk menghasilkan c. Simbol γ merupakan lapis linear = $(z+1)x^3 + x^2 + x + z \in R$

Kotak subtitusi φ dapat dirumuskan sebagai berikut :

$$\varphi(u)=(z^2+1)u^{254}+(z^3+1)u^{253}+ (z^7+z^6+z^5+z^4+z^3+1)u^{251} + (z^5+z^2+1)u^{247} + (z^7+z^6+z^5+z^4+z^2)\ u^{239} + u^{223} + (z^7+z^5+z^4+z^2+1)\ u^{191} + (z^7+z^3+z^2+z+1)\ u^{127}+ (z^6+z^5+z+1) \in F(u) \quad (4)$$

Pentingnya ekspresi aljabar yang komplek pada kotak-S dapat dilihat pada [3]. Sedangkan ekspansi kuncinya dapat dijelaskan sebagai berikut :

$k^{(0)} = k$

$$k_0^{(t+1)} = (\sum_{i=0}^{3} \varphi(k_{i,3}^{(t)}) x^i) x^3 + z^t + k_0^{(t)} \text{ untuk } t=0...9$$

$$k_i^{t+1} = k_{i-1}^{t+1} + k_i^{(t)} \text{ untuk } t = 0...9 \text{ dan } i = 1,2,3$$

Sedangkan dekripsi dilakukan sebagai berikut :

$$\gamma^{-1} = (z^3+z+1)x^3 + (z^3+z^2+1)x^2 + (z^3+1)x + (z^3+z^2+z) \in R$$

Bila $\varphi\psi = 1$, maka

$$c^{(0)} = c + k^{(10)}$$

$$c^{(t+1)} = \gamma^{-1}\sum_{i=0}^{3}\sum_{j=0}^{3} \psi(c_{i,j}^{(t)}) x^i y^{i+j} + \gamma^{-1} k^{(9-t)}$$

untuk t=0...8 (5)

$$c^{(10)} = \sum_{i=0}^{3}\sum_{j=0}^{3} \psi(c_{i,j}^{(9)}) x^i y^{i+j} + k^{(0)} \quad (6)$$

Penjelasan lebih lengkap tentang Rijndael dapat dilihat pada [4].

3. ANALISIS SANDI LINEAR PADA AES

Analisis sandi linear mula-mula diusulkan Matsui [5] Analisis ini berusaha mendapatkan bias dari persamaan linear antara masukan dan keluaran cipher dengan peluang sebesar mungkin.

Definisi 1. Untuk sebarang masukan/keluaran *x*,*y* dan *masking* Γx, $\Gamma y \in GF(2^m)$, peluang linear kotak-S : $GF(2^m) \rightarrow GF(2^m)$ didefinisikan sebagai berikut :

$$P_{\text{lin}}[x.\Gamma x = s(x).\Gamma s(x)] = \frac{\#\{x \in GF(2^m) \mid x.\Gamma x = s(x).\Gamma y\}}{2^m} \quad (7)$$

Bagian utama kebanyakan algoritma enkripsi adalah perulangan ronde. Struktur T ronde dapat kita nyatakan sebagai berikut :

$$E_T = \chi[k_T] \circ \chi[k_{T-1}] ... \circ \chi[k_1] \quad (8)$$

$$r_i = \chi[k_i](r_i - 1) \quad i=1...,T \quad (9)$$

di mana masukan adalah $r = r_0$ dan keluaran = r_T.

Definisi 2 (MLP) [6]. Bila x,y,k adalah vektor masukan, keluaran dan kunci, maka *Maximum Linear Probability* (MLP) untuk Enkripsi E_T, dengan T ronde didefinisikan sebagai berikut :

$$MLP(E_T) \equiv \max_{\Gamma x, \Gamma y \neq 0, k} \sum_{\Gamma x_1, \Gamma x_2, ..., \Gamma x_{T-1}} \prod_{i=1}^{T} LP^{\rho_i[k_i]}(\Gamma x_i \rightarrow \Gamma x_{i-1}) \quad (10)$$

Jika MLP cukup kecil, maka dijamin tidak akan ada kunci yang lemah menghadapi ASL. Karena MLP sulit dihitung, maka digunakanlah cara perhitungan lain yaitu dihitung rata-rata MLP nya, yang disebut sebagai MALHP.

Definisi 3 (MALHP). ***Maximum Average Linear Hull Probability*** (***MALHP***) didefinisikan sebagai berikut :

$$MALHP(E_T) \equiv \max_{\Gamma x,\Gamma y\neq 0,k} rata_k^2 \sum_{\Gamma x_1,\Gamma x_2,\ldots,\Gamma x_{T-1}} \prod_{i=1}^{T} LP^{\rho_i[k_i]}(\Gamma x_i \rightarrow \Gamma x_{i-1}) \quad (11)$$

Tetapi ternyata, perhitungan MALHP juga masih sulit karena terdapat penjumlahan pada bagian tengah *block cipher*. Karena itu maka digunakan metode perhitungan lain yaitu menggunakan pendekatan jalur tunggal MLCP.

Definisi 4 (MLCP). ***Maximum Linear Characteristic Probability*** (***MLCP***) didefinisikan sebagai berikut :

$$MLCP(E_T) \equiv \max_{\Gamma x,\Gamma y\neq 0,k} rata_k^2 \sum_{\Gamma x_i} \prod_{i=1}^{T} LP^{\rho_i[k_i]}(\Gamma x_i \rightarrow \Gamma x_{i-1}) \quad (12)$$

Untuk melakukan analisis sandi linear pada sebuah *cipher*, kita harus memulainya dari bagian yang tidak linear , yaitu kotak subtitusi. Bila seluruh komponen cipher dapat dinyatakan dalam persamaan linear, maka cipher tersebut akan mudah sekali dipecahkan. Kotak subtitusi pada AES menggunakan fungsi inversi x^{-1} pada $GF(2^8)$. Fungsi ini memiliki sifat sangat tidak linear dan memiliki uniform δ yang baik yaitu 4. Menurut Nyberg[7], ketidaklinearan fungsi inversi juga hampir maksimum, yaitu 2^{n-1}-$2^{n/2}$, sedangkan nilai maksimum yang mungkin adalah 2^{n-1}-$2^{n/2-1}$ [8].

AES menggunakan metode *wide trail strategy* [9] yang diusulkan Daemen dalam disertasinya. Menurut strategi ini, setiap komponen cipher hendaknya memiliki sifat baik yang mandiri, sehingga bila komponen tersebut ditukar dengan komponen lainnya, sifat cipher secara keseluruhan tidak berubah. Mengingat jalur ASL yang dapat berbeda-beda, maka diinginkan agar setiap komponen cipher memiliki sifat yang diinginkan tanpa bergantung kepada komponen lainnya. Jadi, menurut strategi ini, sebaiknya setiap cipher memiliki komponen tidak linear yang memiliki fungsi *confusion*, komponen linear yang berfungsi sebagai *diffusion*, dan pencampuran kunci agar setiap bit keluaran bergantung pada setiap bit kunci.

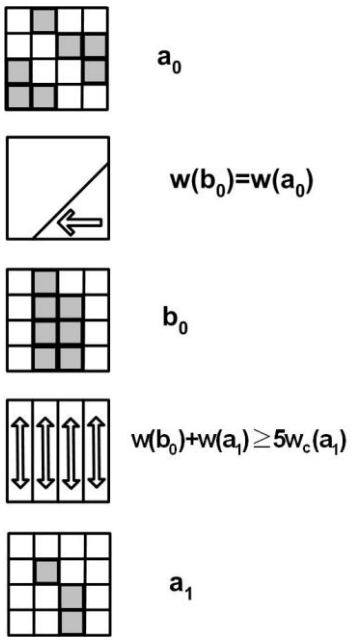
Seperti sudah diketahui, komponen tidak linear AES menggunakan fungsi inversi pada $GF(2^8)$. Sedangkan fungsi linearnya menggunakan konsep *branch* (jumlah cabang), yang artinya adalah bahwa setiap satu kotak-S yang aktif di sebuah ronde, fungsi linear yang menghubungkan antar ronde harus dapat membuat sebanyak mungkin kotak-S di ronde berikutnya menjadi aktif.

Definisi 5. Jumlah cabang B dari transformasi linear MixColumn didefinisikan sebagai berikut :

$$B = \min_{x\neq 0}(w_H(x)+w_H(P(x))) \quad (13)$$

di mana $w_H(x)$ menyatakan bobot byte Hamming dari x, dan P adalah transformasi linear.

Untuk membuktikan bahwa AES tahan terhadap ASL, kita harus membuktikan bahwa jumlah plaintext yang dibutuhkan untuk mendapatkan kunci AES-128 lebih besar daripada 2^{128}. Untuk kasus Rijndael, kita dapat mengamati dari alur enkripsinya, seperti diperlihatkan pada Gambar 1. Karena pencampuran kunci dan proses subtitusi tidak mempengaruhi alur byte data pada ASL, maka kedua komponen ini tidak dimasukkan pada gambar. Pola aktifitas sebelum ronde-i disebut a, sedangkan pola setelah ShiftRow pada ronde i disebut b.



Gambar 1. Ilustrasi teorema 1 dengan Q = 2

Teorema 1. Jumlah bobot byte alur dua ronde dengan jumlah kolom aktif sebanyak Q pada masukan ronde kedua memiliki batas terendah $5Q$.

Bukti : Mixcolumn menggunakan MDS (*Maximum Distance Separable*) sehingga memiliki jumlah cabang optimum yaitu 5. Hal ini menyebabkan jumlah bobot byte pada setiap kolom pada b_0 dan a_1 (Gambar 1) memiliki nilai terendah =5. Jika bobot kolom a_1 adalah Q, maka penjumlahan bobot byte pada b_0 dan a_1 akan menghasilkan paling sedikit nilai $5Q$. Karena a_0 dan b_0 memiliki bobot byte hamming yang sama, maka penjumlahan bobot a_0 dan a_1 paling sedikit juga $5Q$.

Ini berakibat bahwa untuk lintasan dua ronde, terdapat paling sedikit 5 kotak-S yang aktif.

Lemma 1. Untuk lintasan dua ronde, penjumlahan dari jumlah kolom yang aktif pada masukan dan jumlah kolom yang aktif pada keluaran, paling sedikit adalah 5. Dengan kata lain, jumlah bobot kolom a_0 dan a_2 paling sedikit sebanyak 5.

Bukti : ShiftRows menggeser byte-byte dalam kolom a_i ke kolom lainnya di b_i. Akibatnya, bobot kolom pada a_i memiliki jumlah minimal sama dengan bobot byte pada kolom b_i. Demikian pula, bobot kolom b_i terkecil sama dengan bobot byte pada kolom a_i .Pada sebuah lintasan ASL, paling sedikit, sebuah kolom pada a_1 atau b_0 akan aktif. Kolom yang aktif ini kita sebut kolom *g*. Karena Mixcolumn memiliki jumlah cabang 5, maka penjumlahan dari jumlah bobot byte kolom *g* di b_0 dan kolom *g* di a_1 paling sedikit = 5. Bobot kolom a_0 terkecil sama dengan bobot byte kolom g di b_0. Bobot kolom di b_1 yang terkecil sama dengan bobot byte pada kolom *g* di a_1. Ini mengakibatkan jumlah bobot kolom pada a_0 dan b_1 terkecil adalah 5. Dan karena bobot kolom di a_2 sama dengan di b_1, maka jumlah bobot kolom a_0 dan a_2 paling sedikit 5.

Teorema 2. Sebarang lintasan empat ronde memiliki paling sedikit 25 byte aktif.

Bukti : Dengan menggunakan teorema 1 pada dua ronde pertama (ronde 1 dan 2) serta dua ronde terakhir (3 dan 4), maka bobot byte lintasan sedikitnya adalah 5 kali jumlah bobot kolom a_1 dan a_3. Dengan Lemma 1, jumlah bobot kolom a_1 dan a_3 paling sedikit adalah 5. Dari dua fakta ini maka bobot byte terkecil untuk lintasan 4 ronde adalah 5x5 = 25.

Dari kenyataan ini, jumlah minimum kotak-S yang aktif pada AES 4 ronde dengan ASL adalah 25 buah. Peluang linear maksimum kotak-S adalah 2^{-4}, maka peluang linear maksimum Rijndael 4 ronde menjadi $pl = (2^{-4})^{25} = 2^{-100}$. Artinya, untuk memecahkan AES 4 ronde dengan ASL diperlukan $p_l^{-2} = 2^{200}$ plaintext, dengan tingkat keberhasilan sekitar 97% . Dan bila *linear hull* ikut dimasukkan, AES dengan 5 ronde diperkirakankan mampu mengatasi masalah ini.

Kesimpulannya, AES 7 ronde akan tahan menghadapi ASL berikut *linear hull* yang menggunakan serangan 2R.

4. ASL TERHADAP DES (DATA ENCRYPTION STANDARD)

DES merupakan algoritma enkripsi yang memiliki struktur Feistel sehingga struktur enkripsi dan dekripsinya sama. DES lengkap terdiri dari 16 ronde serta memiliki masukan dan keluaran 64 bit. Kunci DES hanya 56 bit sehingga dianggap tidak aman lagi. Namun DES masih digunakan hingga kini karena masalah kompatibilitas dengan perangkat lama, dan DES yang sering digunakan adalah adalah triple DES, yang menggunakan 3 kali enkripsi DES. Deskripsi lengkap tentang cara kerja DES dapat dilihat pada [10]

Seperti diketahui, ASL pertama kali diperkenalkan Matsui pada [5]. ASL lebih berhasil daripada Analisis sandi diferensial (ASD) dalam pemecahan DES karena ASD sudah diketahui para perancang DES di tahun 1970-an, sedangkan ASL belum mereka kenal, sehingga DES tidak dirancang untuk menghadapi ASL. ASL akan menganalisis kotak-S dan berusaha mencari kelinearannya. ASL membutuhkan 2^{42} *known plaintext*, sehingga lebih mudah diterapkan dibanding ASD yang merupakan *chosen text*.

Matsui mendapatkan pendekatan persamaan linear DES :

$$P_L[15] \oplus P_H[7,18,24,29] \oplus C_L[7,18,24] = K_1[22] \oplus K_3[22] \oplus K_4[44] \oplus K_5[22] \oplus K_7[22] \oplus K_8[44] \oplus K_9[22] \oplus K_{11}[22] \oplus K_{12}[44] \oplus K_{13}[22] \quad (14)$$

Pendekatan persamaan linear DES yang lain adalah :

$$P_L[7,18,24] \oplus C_L[15] \oplus C_H[7,18,24,29] = K_2[22] \oplus K_3[44] \oplus K_4[22] \oplus K_6[22] \oplus K_7[44] \oplus K_8[22] \oplus K_{10}[22] \oplus K_{11}[44] \oplus K_{12}[22] \oplus K_{14}[22] \quad (15)$$

Persamaan (14) menyatakan bahwa bila bit no 15 dari setengah blok kanan plaintext di-XOR-kan dengan bit nomor 7,18,24,29 dari setengah blok kiri plaintext kemudian di-XOR-kan dengan bit bernomor 7,18,24 dari ciphertext blok kiri akan memiliki nilai "0" atau "1" (bergantung nilai hasil XOR bit-bit kuncinya).

Persamaan (15) dipenuhi dengan peluang p = ½ -1,19 x 2^{-21}. Dari sini didapatkan bias peluang sebesar 1,19 x 2^{-21}. Dan menurut Matsui, diperlukan ε^{-2} plaintext untuk mendapatkan kunci dengan tingkat kebenaran mencapai 97,7%. Artinya diperlukan $(1{,}19 \times 2^{-21})^{-2} \approx 2^{42}$ plaintext untuk menerka kunci dengan tingkat keyakinan 97,7%.

Mula-mula kita cari kotak-S DES yang memiliki bias linear sebesar mungkin, dan kemudian kita telusuri lintasan bit-bit ini baik ke depan (ke arah plaintext) maupun ke arah ciphertext, melintasi ronde demi ronde hingga akhirnya ke-14 ronde DES dapat dilibatkan. Dengan bias 14 ronde ini, kita dapat menerka bit-bit kunci pada ronde 16. Untuk setiap kotak-S, dapat kita hitung jumlah munculnya (NS) pola berikut :

$$NSa(\alpha,\beta) \xrightarrow{\text{def}} \#\{x \mid 0 \le x \le p, (\bigoplus_{s=0}^{y-1}(x[s] \bullet \alpha[s])) = (\bigoplus_{t=0}^{z-1}(s_a(x)[t] \bullet \beta[t]))\} \quad (16)$$

di mana α dan β adalah masking nilai, sedangkan *x* adalah masukan kotak-S, dan $s_a(x)$ adalah keluaran kotak-S, dan simbol • menyatakan operasi AND. Bila

ini dikembangkan, akan diperoleh persamaan (14) dan (15).

5. ALGORITMA ENKRIPSI 1

AE1 (Algoritma Enkripsi 1) adalah algoritma enkripsi yang peneliti rancang berdasar algoritma keluarga AES. Algoritma AE1 memiliki struktur yang serupa dengan AES, yaitu SPN. Namun AE1 dirancang agar memiliki struktur yang *involutional*, yaitu struktur enkripsi dan dekripsinya sama, tidak seperti AES. Struktur AE1 menyerupai *block cipher* ARIA asal korea [11].

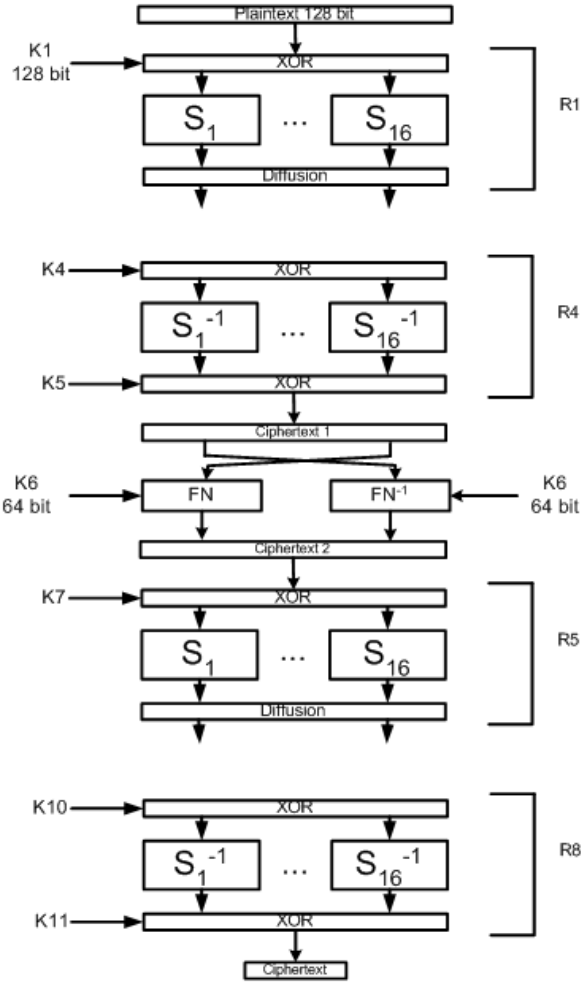
Keistimewaan yang diharapkan dari AE1 adalah kebal terhadap berbagai macam analisis sandi seperti ASD, ASL, Square Attack, Slide Attack, related key attack, Boomerang attack, interpolation attack dan impossible attack. Namun AE1 juga memiliki kekurangan, yaitu kurang cepat dalam implementasi di perangkat lunak pada prosesor 32 bit bila dibandingkan AES. Hal ini disebabkan karena AE1 menggunakan matrik MDS(*Maximum Distance Separable*) yang berukuran besar yaitu 16 x 16. MDS ini menjadi salah satu faktor terpenting keamanan AE1, namun juga memberikan pengaruh melambatnya algoritma.

5.1. Deskripsi AE1

AE1 merupakan algoritma yang memiliki masukan keluaran 128 bit, dan kunci 128 bit. AE1 memiliki 6 ronde regular (ronde 1,2,3,5,6,7) dan ronde khusus (ronde 4,8 dan fungsi FN). Setiap ronde regular AE1 dapat dinyatakan sebagai $\chi = \psi \circ \sigma \circ \kappa \circ x$ di mana x adalah masukan 128 bit yang diXORkan dengan subkey κ dan hasilnya dimasukkan ke dalam 16 buah kotak subtitusi σ yang sama dan kemudian dilalukan ke lapis difusi, yaitu matrik MDS ψ yang berukuran 16x16. Kotak-S pada ronde genap merupakan inversi dari kotak-S pada ronde ganjil serta MDS yang merupakan matrik involutional digunakan untuk membuat sistem menjadi involutional secara total. Struktur AE1 dapat dilihat pada Gambar 2.

MDS AE1 diilhami cipher Khazad [12] yang menggunakan matrik hadamard untuk mempermudah mendapatkan matrik yang memenuhi kriteria MDS dan sekaligus *involution*. Meskipun Khazad juga *involution*, AE1 tidak mengikuti struktur Khazad karena kuatir sifat kotak-S *involution* yang dimiliki khazad[13]. Matrik Hadamard memiliki sifat $m_{i,j} = di_{\oplus}j$ di mana m adalah matrik berukuran n x n dan d adalah vektor kode MDS. AE1 menggunakan d = {14, 1, 4, 5, 8, 11, 16, 11, 17, 13, 10, 12, 15, 3, 6, 7}. Matrik ini peneliti peroleh secara semi acak menggunakan lemma 2.

Lemma 2. Sebuah kode (n,k,d) yang dibentuk dari generator G=[I A], di mana A adalah matrik k x (n-k), adalah MDS, jika dan hanya jika setiap submatrik bujursangkar A(yang tersusun dari sebarang baris i dan kolom i untuk i = 1,2,...min{k,n-k}) merupakan matrik nonsingular. Beberapa metode untuk mendapatkan kode MDS dapat dilihat pada [14].

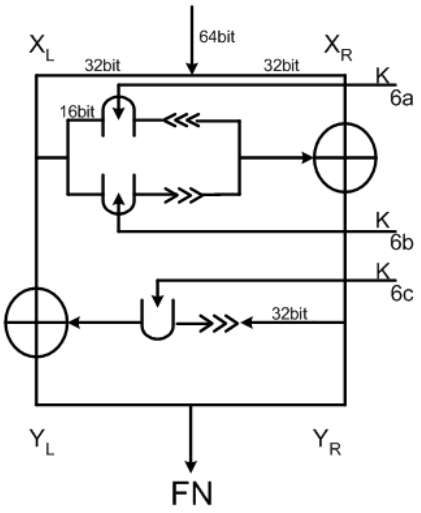


Gambar 2. Algoritma AE1

Untuk membuat *involutional*, diperlukan pengolahan subkey pada dekripsi sebagai berikut. Kunci pertama pada dekripsi adalah kunci terakhir pada enkripsi atau K_1 dekripsi (K_{1d}) = K_{11} enkripsi (K_{11e}). Kemudian $K_{2d} = \psi \circ K_{10e}$, $K_{3d} = \psi \circ K_{9e}$ dan seterusnya. Jadi perbedaan antara subkey enkripsi dengan dekripsi terletak pada urutan dan proses difusi beberapa subkey dekripsi.

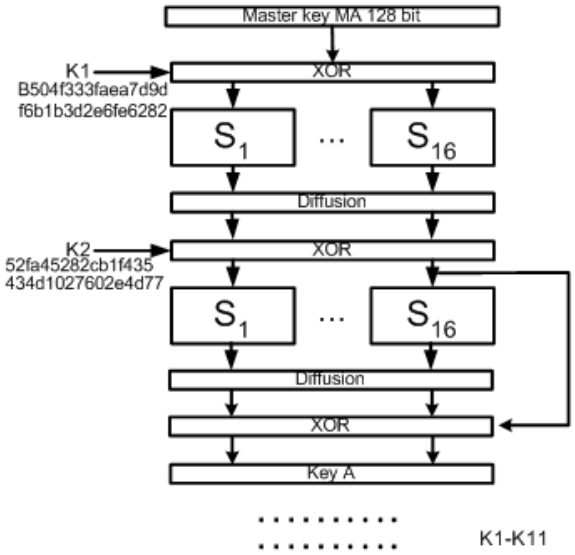
Fungsi FN (Gambar 3)dibuat mengikuti fungsi FL pada Camellia [15]. Struktur FN dan inversi FN dibuat sedemikian rupa agar cipher keseluruhan tetap *involutional*. Lambang □ merupakan OR, □ adalah AND, dan <<< adalah rotasi ke kiri 1 bit.

FN lebih komplek dari FL untuk meningkatkan ketahanan AE1 terhadap *truncated differential attack* yang mengeksploitasi cipher berorientasi byte. FN juga berfungsi untuk menghancurkan lintasan *linear hull* yang mungkin muncul, serta lintasan *impossible differential attack*.



Gambar 3. Fungsi FN

FN tidak diletakkan di bagian awal cipher karena difusinya yang lambat sehingga dikuatirkan tidak akan banyak bermanfaat. Bila diletakkan di tengah, maka kemungkinan sangat kecil bahwa analis sandi dapat memberi masukan FN sesuai yang diinginkannya, mengingat difusi yang boleh dikatakan maksimal pada ronde-ronde sebelumnya.



Gambar 4. Ekspansi kunci AE1

Ekspansi kunci diperlihatkan pada Gambar 4 Ekspansi kunci ini menggunakan struktur yang serupa dengan struktur enkripsi untuk penghematan ruang bila diimplementasikan pada perangkat keras. Konstanta c_1 (pecahan $2^{-0,5}$ yang dinyatakan dalam hexadesimal) dan c_2 ($5^{-0,7}$) digunakan sebagai kunci enkripsi dengan plaintext Masterkey untuk menghasilkan kunci A. Dan dari kunci A inilah diturunkan subkey K_1 - K_{11} dengan cara merotasi 3 bit ke kiri untuk ronde ganjil dan rotasi 5 bit ke kiri untuk ronde genap. Dengan struktur tersebut diharapkan bahwa apabila analis sandi berhasil mendapatkan subkey, K_{11}, misalnya, dia tetap tidak akan dapat menurunkan masterkey-nya. Berbeda dengan DES, karena sangat sederhananya proses ekspansi kunci, bila subkey berhasil diketahui, maka masterkey-nya akan mudah ditemukan. Dengan proses difusi dan *confusion* yang kuat pada ekspansi kunci diharapkan bahwa *related-key attack* akan sangat dipersulit.

5.2. ASL Terhadap AE1

Karena sejak awal, AE1 telah dirancang untuk dapat menahan ASL, maka AE1 dapat dibuktikan akan sanggup menghadapi ASL. ASL sangat dipengaruhi oleh jumlah kotak subtitusi yang aktif. Semakin besar jumlah kotak subtitusi yang aktif, semakin kecil pula peluang ASL akan berhasil. Pada kasus *block cipher* SPN sederhana, di mana digunakan permutasi bit untuk lapis difusinya, dengan mudah kita mendapatkan jalur diferensial dan linear yang menggunakan sesedikit mungkin kotak-S, sehingga ASL dengan mudah pula menembusnya.

Definisi 6. Kotak-S aktif linear adalah kotak-S dengan masking keluaran tidak nol. Dan bila kotak-S yang digunakan bijektif, maka definisi kotak-S aktif linear dapat dikatakan juga sebagai kotak-S aktif dengan masking masukan tidak nol [6].

Kode linear [n,k,d] pada $GF(2^m)$ merupakan subruang dari ruang vektor $GF(2^m)^n$ berdimensi-k , di mana jarak hamming antara dua vektor yang berbedanya paling sedikit d. Dalam kasus AE1, m adalah jumlah bit masukan atau keluaran kotak-S (=8 bit), dan n adalah jumlah kotak-S pada ronde berikutnya (16 buah). Jarak Hamming yang dimaksud dalam aplikasi *block cipher* adalah jumlah komponen tidak yang sering dinyatakan dengan $w_h(a)$. Untuk aplikasi *block cipher*, digunakanlah kode $[2n,n,n+1]$ yang artinya adalah panjang kodenya $2n$, berdimensi n dan $w_{h,\min}(a)$ adalah $d=n+1$. Kode linear yang memenuhi $d = n - k + 1$ disebut kode yang memiliki jarak pemisahan maksimal (*Maximal Distance Separable).* Untuk kasus AE1, $d = 2n - n + 1 = n + 1$.

Teorema 3. Dalam 4 ronde pertama AE1, $LP_{min} = 2^{136}$

Bukti : Karena MDS yang digunakan memiliki jarak minimal = 17, maka jumlah kotak-S yang aktif baik pada lintasan diferensial maupun lintasan linear pada dua ronde yang berturutan minimal adalah 17. Jika pada ronde pertama terdapat sebanyak x kotak-S aktif, maka pada ronde kedua akan terdapat $(17-x)$ kotak-S yang aktif, dan pada ronde ketiga sebanyak x, dan pada ronde keempat sebanyak $(17-x)$. Sehingga dalam 4 ronde, minimal terdapat 34 kotak-S aktif. Karena itu, untuk 4 ronde, *Linear Probability* $LP_{min} = (2^{-4})^{34} = 2^{-136}$

Dari teorema tersebut dapat disimpulkan bahwa bila lintasan 4 ronde AE1 dapat digunakan untuk memecahkan 6 ronde AE1 (2R-*attack*), maka diperlukan 2^{272} plaintext yang diketahui untuk memecahkan AE1 menggunakan ASL dengan tingkat keyakinan 97,7%. Sedangkan diketahui bahwa hanya terdapat 2^{128} plaintext yang mungkin ada. Artinya, AE1 terbukti mampu menghadapi ASL. Dan bila *linear hull* ikut diperhitungkan, maka fungsi FN akan menghancurkan lintasannya, itupun bila dalam 4 ronde memiliki peluang yang lebih besar dari pada 2^{-128}. Sebagai perbandingan, DES dapat dipecahkan dengan 2^{43} plaintext, sedangkan AES 6 ronde membutuhkan 2^{200} plaintext, bila digunakan karakteristik linear 4 ronde.

6. KESIMPULAN

Dari pembahasan di atas, kita dapat menyimpulkan hal-hal berikut :

- AES-128 memiliki ketahanan yang besar untuk menghadapi ASL, karena analisis sandi tersebut hanya mampu memecahkan AES hingga 6 ronde, sedangkan AES-128 memiliki 10 ronde. Bandingkan dengan DES lengkap yang dapat dipecahkan ASL dengan 2^{43} plaintext, di mana DES memiliki masukan 64 bit.
- Ketahanan cipher terhadap sebuah analisis sandi tidak otomatis menyebabkannya tahan terhadap analisis sandi lainnya. Ini terlihat dari ketahanan Rijndael terhadap ASL setelah 4 ronde, namun dengan Square attack, 4 Ronde Rijndael dapat dipecahkan dengan mudah. Sebaliknya, DES yang lemah menghadapi ASD dan ASL, ternyata memiliki ketahanan yang besar terhadap square attack.
- AE1 yang memiliki ketahanan yang lebih besar dibanding AES maupun DES terhadap ASL untuk jumlah ronde yang sama, namun lebih lambat dalam hal kecepatan eksekusi. Untuk memecahkan DES 16 ronde lengkap dengan serangan 2 ronde, diperlukan 2^{43} plaintext, untuk AES 6 ronde diperlukan 2^{200} plaintext, dan untuk AE1 6 ronde diperlukan 2^{272} plaintext yang diketahui. Sedangkan untuk ronde lengkap, AE1 mebutuhkan 2^{408} plaintext dengan 2*R-attack*, sedangkan AES membutuhkan 2^{400} plaintext.
- Pembuktian keamanan algoritma kriptografi lebih sulit dari pada pembuatan algoritmanya, sehingga penulis menyarankan agar penelitian mengenai analisis sandi ditingkatkan.

7. DAFTAR PUSTAKA

[1] Nippon Telegraph and Telephone Corporation, *Supporting Document on E2*, pp 5, 1998

[2] Joachim Rosenthal, A Polynomial Description of the Rijndael Advanced Encryption Standard. *Journal of Algebra and Its Applications*, 2 (2):pp.223-236, 2003

[3] T. Jakobsen and L.R. Knudsen The interpolation attack on block ciphers, E.Biham ed., *LNCS 1267, Springer-Verlag*, Berlin, 1997

[4] J.Daemen and V. Rijmen, AES Proposal : Rijndael, *AES submission*, http: //www.nist.gov /aes, 1999

[5] M. Matsui, Linear cryptanalysis method for DES cipher, *Advances in Cryptology, Proceedings Eurocrypt* '93, *LNCS 765*, T. Helleseth, Ed., Springer-Verlag, 1994, pp. 386-397

[6] K. Ohkuma, H. Shimizu, F. Sano, and S. Kawamura. Security Assessment of Hierocrypt and Rijndael against the Differential and Linear Cryptanalysis. *In Proceedings of the 2nd NESSIE workshop*, 2001

[7] Kaisa Nyberg (1994), Differentially uniform mapping for Cryptography, *Advances in Cryptology, Proc. Eurocrypt'93, LNCS 765* T. Helleseth, Ed., Springer- Verlag, pp 439-444

[8] Yusuf Kurniawan et al. The Design Of Block-Cipher Resistant to Cryptanalysis. *International Conference on Applied Mathematics (ICAM05), ITB*. August, 2005

[9] J. Daemen and V. Rijmen. *The Design of Rijndael : AES- The Advanced Encryption Standard*. Springer Verlag, 2002.

[10] Man Young Rhee, *Cryptography and Secure Communications*, Mc Graw-Hill, 1994.

[11] Kwon et al *New Block Cipher ARIA*. National Security Research Institute, Specification of ARIA, 2003.

[12] P. S. L. M. Barreto and V. Rijmen, *The Khazad legacy-level block cipher*.Primitive submitted to NESSIE ,Sept. 2000.

[13] A. Biryukov, Analysis of involutional ciphers : Khazad and Anubis. in *Proceedings of Fast Software Encryption* (FSE' 03) in T.Johansson,ed., Lecture Notes in Computer Science, Springer-Verlag, 2003.

[14] A. M.Youssef, S.and S. E. Tavares, On the design of linear transformations for substitution permutation encryption networks, *Workshop on Selected Areas of Cryptography, SAC'97*, Workshop record, 1997,pp.40–48.

[15] K. Aoki et al, *Camellia: A 128 Bit Block Cipher Suitable for Multiple Platforms*. NTT and Mitsubishi Electric Corporation, 2000.