CARA KERJA SERANGAN XSL

Muhammad Amrimirza – NIM : 13506003

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if16003@students.if.itb.ac.id

Abstrak

Pada saat ini cukup banyak cipher yang dibangun dengan memanfaatkan beberapa lapis Kotak Substitusi/*Substitution Box* (*S-Box*), yang dihubungkan dengan kunci yang saling bergantung secara linier. Dari sini keamanan yang diberikan terhadap metode kriptanalisis klasik, yang berdasarkan pada probabilitas karakterisistik, akan meningkat secara eksponensial dengan penambahan pengulangan enkripsi (N_r). Dalam tulisan ini, akan dibahas keamanan cipher ini, dengan asumsi: "*S-box* bisa didefinisikan dengan *overdefined system* dalam persamaan aljabar." Di mana asumsi ini berlaku dalam cipher *Rijndael* yang memiliki properti aljabar yang unik. Serangan *XLS*, yang merupakan singkatan dari *eXtended Sparse Linearization* memiliki sebuah parameter *P*, dimana *P* merupakan suatu konstanta. Dengan nilai *P* yang akan terus meningkat seiring banyaknya jumlah N_r. Serangan *XLS* diharapkan dapat memecahkan cipher *Rijndael*, walaupun ada kemungkinan metode *XSL* akan lebih lambat daripada *exhautive search*.

***Kata kunci**: Serangan XLS, Rijndael, AES, persamaan multivariate quadratic, Substitution Box (S-Box)*

1. Pendahuluan

Jika dihitung dari sejak tulisan ini disusun, maka bisa dikatakan kurang lebih sudah 8 tahun *NIST* (*National Institute of Standards and Technology*) menetapkan algoritma enkripsi *Rijndael* sebagai algoritma AES (*Advanced Encryption Standard*). Karena sifat algoritma *Rijndael* yang bebas digunakan secara pribadi ataupun umum, membuat algoritma enkripsi yang satu ini cukup sering diimplementasikan pada beberapa perangkat lunak, seperti: *library* pada beberapa bahasa pemrograman (seperti: C, C++, C#, Java, dan sebagainya), perangkat untuk kompresi, enkripsi, perangkat keamanan, dan sebagainya. Dalam tulisan ini, akan dibatasi pada kriptanalisis terhadap *Rijndael* sebagai pemecahan sistem persamaan *Multivariate Quadratic* (*MQ problem*).

2. *Substitution-Affine Cipher (SA cipher)*

Salah satu cara paling umum yang digunakan untuk membangun sebuah algoritma cipher, sesuai dengan paradigma *Shannon*, adalah dengan mencampurkan *confusion layer* dengan *diffusion layer*. Cipher ini disebut dengan *SA cipher*. Dan dalam tulisan ini dispesifikan juga sebuah bentuk dari *SA cipher*, yaitu *XSL cipher*, di mana serangan *XSL* akan lebih dituukan pada cipher yang memiliki tipe *XSL cipher*.

2.1. Rijndael sebagai *XSL* Cipher

Secara definisi *XSL* cipher merupakan komposisi dari langkah-langkah berikut yang dilakukan berulang sebanyak N_r kali:

1. X: pada putaran pertama (i=1) dimulai dengan melakukan XOR terhadap kunci K_{i-1}
2. S: penerapan sebuah *layer* dari *B Bijective S-Box* secara paralel, dalam *s* bits.
3. L: penerapan *linier diffusion layer*
4. X: Dilakukan XOR sekali lagi dengan Kunci K_i.

Kemudian dilakukan pengulangan hingga $i=N_r$.

Struktur Rijndael merupakan sebuah tipe khusus dari *XSL cipher*, di mana *s=8*, *B=4*Nb*. Dan pengulangan (N_r) sebanyak 10 hingga 14 kali. Data dalam *Rijndael* direpresentasikan dalam "*state*" persegi yang berisikan kolom sebanyak N_b, dengan tiap-tiap kolomnya memiliki ukuran 4 *S-Box* (4*s=32 *bits*). N_b yang dimiliki antara lain: 4, 6, dan 8, sehingga akan didapat ukuran blok N_b*32: 128, 192, dan 256 *bits*.

Dan algoritma enkripsi dari *Rijndael* adalah:

1. X: melakukan XOR terhadap kunci K_{i-1}
2. S: akan ada $B = Nb*4$ *S-Box* dengan ukuran $s=8$ *bits*.
3. L: Kita akan memiliki permutasi *Byte* (yang disebut *ShiftRow*), diikuti transformasi linier:
   $GF(256)^4 \rightarrow GF(256)^4$
   Yang disebut *MixColumn* dan dilakukan secara paralel untuk tiap kolom N_b.
4. X: Dilakukan XOR sekali lagi dengan Kunci K_i.

Kemudian dilakukan pengulangan hingga $i=N_r$.

Dalam bentuk tidak di-*expand*, kunci akan memiliki panjang:

$Hk = N_k*32$ bits

(dengan $N_k = 4, 6,$ atau 8)

dan jika dikembangkan akan menjadi:

$E_k = (N_r+1)*s*B = (N_r+1)*N_b*32$ *bits*

Kunci dinotasikan dalam *XSL cipher* dengan variabel K_{ij}, dengan $i=0..N_r$ dan $j=1..s*B$. *Session Key* yang dimiliki ada sejumlah $Nr+1$, di mana K_0 yang pertama dan K_{Nr} yang terakhir. Jumlah bit kunci sebelum diekspansi adalah H_k, sedangkan jumlah kunci setelah diekspansi adalah E_k, serta jumlah kunci yang saling bebas linier adalah L_k.

Disebut X_{ij} untuk bit ke-*j* dari input fungsi ke-*i* dari XSL cipher diambil setelah XOR oleh session key. Dan Y_{ij} untuk masukan ke-*j* dari bagian linear fungsi ke-*i* dari *XSL cipher* diambil setelah aplikasi dari *S-Box* dengan *s* yang berkorespondensi dengan X_{ij}.

Hal ini juga mirip pada Z_{ij}, di mana output bit ke-*j* dari fungsi (sebelum XOR dengan session key berikutnya). Sehingga kita notasikan plaintext dengan Z_0 dan ciphertext dengan X_{Nr+1}, di mana hal ini adalah constrain, bukan variabel.

Dengan notasi ini didapat:

$$X_{i+1\ j} = Z_{i\ j} \oplus K_{i\ j},$$

Untuk semua $i = 0..N_r$.

3. Serangan XSL Pertama

Dimulai dari persamaan untuk tiap *S-Box* di mana terdapat *r* persamaan dan *t term*, kita akan menuliskan himpunan persamaan kuadrat yang benar-benar mendefinisikan kunci dari cipher tersebut. Kemudian dikalikan dengan beberapa monomial yang dipilih. Di mana sebaiknya digunakan produk monomial yang sudah pernah mucul dalam persamaan lainnya. Jika r >= t, kita akan memiliki persamaan sebanyak *term* yang muncul dalam persamaan-persamaan ini. Di mana sistem persamaannya dapat dipecahkan dengan menambahkan variabel baru pada tiap term dan diselesaikan secara linier (metode ini disebut *linierization*)

3.1. Kondisi agar Serangan XSL berhasil atau disebut juga "*T' Method*"

1. Dengan sekali *eliminasi gaussian* kita akan membawa sistem menjadi bentuk, yang dalam istilah *T'* disebut, kombinasi linier.
2. Kita lakukan pra-komputasi yang sama sebanyak 2 kali. Contoh: pendefinisian *T'* untk x_1 terpisah dengan x_2.
3. Dalam kedua sistem, akan ada subsistem persamaan *C* yang hanya akan mengandung term dari *T'*.
4. Dalam kedua subsistem, kita kalikan tiap persamaan dengan x_1, kemudian x_2. Lalu kita substitusikan dengan ekspresi dari poin 1 untuk mendapatkan persamaan lain yang hanya mengandung term dari *T'*, tetapi untuk variabel yang berbeda. Persamaan ini diharapkan bersifat baru dan berbeda.
5. Kemudian, jika *Free* >= *C* + *T* - *T'*, kita bisa menambahkan jumlah persamaannya.
6. Dari sini diharapkan jumlah persamaan baru akan meningkat secara eksponensial.
7. Jika sistem awalnya memiliki sebuah solusi yang unik, maka diharapkan pada saat akhir akan didapat *Free* = *T*.
8. Untuk tiap persamaan yang hanya mengandung term dari *T'*, biaya untuk mengomputasikan tambahan persamaan turunan kurang lebih T'^2. Karena akan ada persamaan *T'* yang hilang, diharapkan akan dilakukan kurang lebih operasi sebanyak T'^3. Di mana bisa dikurangi menjadi T'^w dan akan lebih kecil dari T^w.
9. Jika keseluruhan serangan gagal, lakukan lagi dengan pasangan variabel selain x_1 dan x_2, atau gunakan 3 variabel (dan 3 sistem).

3.2. Inti dari Serangan XSL Pertama

Jika A merupakan sebuah S-Box dari XSL-cipher, disebut "active S-Box", untuk S-Box A ini dapat kita tulis persamaan r:

$$0 = \sum \alpha_{ijk} X_{i\ j} Y_{i\ k} + \sum \beta_{ij} X_{i\ j} + \sum \gamma_{ij} Y_{i\ j} + \delta.$$

Di mana jumlah monomial yang muncul pada fungsi ini kecil, hanya t (dalam betuk $X_{ij}Y_{ik}$). Kita akan mengalikan persamaan ini dengan salah satu t monomial yang ada pada S-Box lainnya, disebut "passive S-Box". Jika S merupakan total S-Box dalam serangan. Karena akan digunakan serangan yang

mengacuhkan key schedule dari cipher, kita anggap eksekusi Nr+1 dan S akan sama dengan B*N_r*(N_r+1).
Parameter penting dari serangan ini adalah P ϵ IN. Dalam serangan akan dikalikan tiap persamaan dari "active S-Box" dengan semua kemungkinan term untuk tiap subset dari (P-1) dari "passie S-Box" lainnya.
Jumlah persamaan yang akan dihasilkan oleh metode ini adalah:

$$R \approx r * S * t^{P-1} * \binom{S-1}{P-1}$$

Dan jumlah term dalam persamaan ini sekitar:

$$T \approx t^P * \binom{S}{P}$$

3.3. Penghapusan Linear Dependency

Akan sangat mungkin jika persamaan-persamaan yang kita dapatkan dari penjelasan di atas tidak bebas linear. Pertama, jika kita asumsikan *P*=2, dan $Eq_1...Eq_r$ dan $Eq'_1..\ Eq'_r$ merupakan persamaan yang ada untuk 2 S-Box *A* dan *A'*. $T_1..T_t$ merupakan *term* yang muncul di *Eqi*. Daripada menuliskan produk: $T_1Eq'_1,...,TtEq'_1$ kita bisa menuliskan: $T_1Eq'_1,...,T_{t-r}Eq'_1$ dan kemudian dilengkapi dengan $Eq_1Eq'_1,...,\ Eq_rEq'_1$. Tapi jika kita terapkan transformasi ini pada semua persamaan yang sudah kita tulis sebelumnya akan terlihat $Eq_iEq'_j$ muncul 2 kali. Dari contoh ini terlihat bahwa untuk tiap *P*, kita harus menghasilkan persamaan dengan cara berikut:

1. Pada satu sisi kita membatasi perkalian persamaan "*active*" hanya dengan salah satu monomial $T_1..T_{t-r}$ untuk beberapa "*passive S-Box*" dari sistem.
2. Dan di sisi lainnya kita juga tambahkan persamaan yang mengandung beberapa "*active S-Box*". Sehingga jumlah persamaan pada bagian pertama XSL kuran lebih:

$$R \approx \sum_{i=1..P} \binom{S}{i} r^i * \binom{S-i}{P-i} (t-r)^{P-i} = \binom{S}{P} \left(t^P - (t-r)^P\right)$$

Dan seperti sebelumnya jumlah *term* dari persamaan ini adalah:

$$T \approx t^P * \binom{S}{P}$$

3.4. Persamaan pada Diffusion Layer

Kita masih belum memiliki sistem yang memiliki solusi tunggal dan unik dan kita butuh beberapa persamaan tambahan. Kita akan membangun persamaan ini dengan suatu cara sehingga mereka bisa dikalikan dengan banyak *term*, dan persamaan tersebut masih tertulis dalam monomial *T* yang sama.
Akan kita eliminasi semua variabel kunci dan tambahkan persamaan, sehingga terbentuk

$$X_{i\ j} \oplus \sum \alpha_j Y_{i-1\ j} =$$
$$X'_{i\ j} \oplus \sum \alpha_j Y'_{i-1\ j} =$$
$$X''_{i\ j} \oplus \sum \alpha_j Y''_{i-1\ j} = \dots$$

Kita memiliki persamaan berwujud $N_r*(N_r+1*(sB))$. Tiap persamaan, yang dsebut "*active equation*", akan dikalikan dengan hasil dari term pada beberapa (*P*-1) "*passive S-Box*". Di sini kita perlu mengeluarkan term untuk beberapa *S-Box* tetangga (yang memiliki variabel yang sama dengan *active equation*), walaupun beberapa dari term tersebbut masih bisa dimasukkan dan tidak akan menambahkan term baru pada *T*. Jumlah dari persamaan baru tersebut:

$$R' \approx N_r * (N_r+1) * (sB) * t^{P-1} * \binom{S}{P-1} =$$
$$S * s * t^{P-1} * \binom{S}{P-1}$$

Dan masih mungkin kita hanya perlu menghasilkan sebagian dari persamaan, dan sisanya harus bebas linear:

$$R' \approx S * s * (t-r)^{P-1} * \binom{S}{P-1}$$

3.5. Kompleksitas yang diharapkan serangan XSL

Tujuan dari serangan ini untuk mebdapat
$T - R - R' > T'I$. Ini mengakibatkan:

$$\frac{S}{P}\binom{S-1}{P-1}(t-r)^P -$$
$$S * s * (t-r)^{P-1} * \binom{S}{P-1} <$$
$$t' t^{P-1} \binom{S-1}{P-1}$$
$$\frac{S}{P}(t-r)^P <$$
$$\frac{S^2}{S-P+1} * s * (t-r)^{P-1} + t' t^{P-1}$$

Kita akan mengasumsikan $P << S$ (*S* biasanya cukup besar pada *S* sekitar BN_r^2) sehingga:
$S - P + 1$ kurang lebih mendekati *S*.

$$\frac{S}{P}\left(1-\frac{r}{t}\right)^P < S\frac{s}{t}+\frac{t'}{t}$$

$$\left(1-\frac{r}{t}\right)^P < \frac{Ps}{t}+\frac{Pt'}{St}$$

Terlihat kondisi ini selalu bisa dipenuhi dan dengan *P*, sisi kanan akan menigkat. Jika kita anggap:

$$\left(1-\frac{r}{t}\right)^{\frac{t}{r}} \approx 1/e$$

Kita akan mendapat aproksimasi:

$$e^{-P\frac{r}{t}} < \frac{Ps}{t}+\frac{Pt'}{St}$$

$$P > \frac{t}{r}\left(-ln\left(\frac{Ps}{t}+\frac{Pt'}{St}\right)\right) \qquad (\#)$$

Ketika r=0 kita akan katakan *P* bernilai tak-hingga dala serangan XSL (atau dengan kata lain serangannya tidak bekerja). Jika T^w merupakan kompleksitas dari reduksi *Gaussian* maka kompleksitas serangan XSL sekitar:

$$WF = T^\omega \approx t^{\omega P}\binom{S}{P}^\omega \approx (tS)^{\omega P} \approx$$

$$(t\cdot B\cdot N_r^2)^{\omega P} \approx (t/s\cdot Bs\cdot N_r^2)^{\omega P} \approx$$

$$(t/s)^{\omega P}\cdot(B\cdot \acute{s}\cdot \grave{N}_r^2)^{\omega P} \approx$$

$$(t/s)^{\omega P}\cdot(\text{Block size})^{\omega P}\cdot(\text{Number of rounds})^{2\omega P}$$

Sekarang kita terapkan estimasi (#). Akan mudah ketika nilai $\left(-ln\left(\frac{Ps}{t}+\frac{Pt'}{St}\right)\right)$ terikat pada konstanta yang tidak bergantung ada ukuran block dan jumlah pengulangan cipher. Dan dalam praktik, kita akan mendapatkan nilai dari $\left(-ln\left(\frac{Ps}{t}+\frac{Pt'}{St}\right)\right)$ mendekati 1. Sehingga akan sangat menarik untuk mengevaluasi kompleksitas tambahan dari serangan XSL terhadap bock-cipher.

$$WF \approx (t/s)^{\omega\lceil\frac{t}{r}\rceil+o(1)}\cdot(B\cdot s\cdot N_r^2)^{\omega\lceil\frac{t}{r}\rceil+o(1)} \approx$$

$$\Gamma^\omega\cdot\left((\text{Block size})\cdot(\text{Number of rounds})^2\right)^{\omega\lceil\frac{t}{r}\rceil}$$

$$WF = \Gamma^\omega\cdot(\text{Block size})^{\mathcal{O}(\frac{t}{r})}(\text{Number of rounds})^{\mathcal{O}(\frac{t}{r})}$$

Ini bentuk polinomial dalam ukuran blok dan jumlah pengulangan cipher. Nilai konstanta bergantung pada nilai Γ yang hanya bergantung pada parameter *S-Box* yang digunakan dalam cipher. Untuk suatu cipher, nilai konstanta dari Γ^w akan tetap (tapi biasanya sangat besar).

3.6. Kompleksitas sebenarnya dari serangan XSL

Melalui simulasi, diyakini serangan *XSL* akan bekerja untuk beberapa kasus. Pada penurunana di atas diasumsikan semua persamaan di $R+R'$ bebas linear dan ini erimplikasi serangan *XSL* pada sebagian nilai *P* yang tetap serangannya akan berhasil untuk pengulangan cipher dalam jumlah berapapun. (Walaupun dengan penambahan jumlah pengulagan *P* hanya akan bertambah secara perlahan).
Jika *P* bernilai konstan, untuk sejumlah *S-Box* yang memiliki banyak persamaan *overdefined*, serangan *XSL* akan berbentuk *polynomial* dalam jumlah pengulangan cipher. Walau nilai *P* bertambah perlahan dan *XSL* bersifat *subeksponensial*, serangan *XSL* merupakan suatu terobosan, karena berbeda dengan kriptanalisis klasik yang kompleksitasnya berkembang secara eksponensial seiring dengan peningkatan jumlah pengulangan cipher.

4. Akibat Serangan XSL

4.1. Seberapa Realistik Serangan XLS

Walaupun serangan *XSL* akan bekerja pada beberapa *P*, diperhitungkan nilai minimum *P* sehingga

$$\frac{R+R'+R''}{T-T'} \geq 1$$

Perubahan kecil pada *P* akan menimbulkan penigkatan drastis pada kompleksitas.

4.2. Konsekuensi Design lock Cipher

Terdapat 2 pendekatan dasar dalam perancangan *block cipher*, yaitu perancangan dengan sedikit pengulangan cipher tetapi sangat kopleks (contoh: *DFC*) dan bentuk perancangan di mana cipher banyak diulang tetapi dengan algoritma yang relatif sederhana (contoh: *Serpent*).
Menurut pencetus serangan *XSL*, dengan menggunakan banyak *layer S-Box* yang sederhana, dapat mengakibatkan mudahnya penyerangan yang bersifat perlahan perkembangan kompleksitasnya terhadap perulangan cipher.

5. Kesimpulan

Dalam tulisan ini, dapat dilihat *property* unik dari cipher *Rijndael*, yaitu cipher *Rijndael* bisa dipaparkan dalam bentuk persamaan *sparse quadratic* dan *overdefined*. Berdasarkan *Eurocrypt'00*, suatu sistem akan lebih mudah dipecahkan jika sistem tersebut berupa persamaan overdefined.
Untuk menghindari serangan *XSL* membongkar cipher *Rijndael*, maka sebaiknya sebagian dari *S-Box* tidak dipaparkan dalam bentuk persamaan *overdefined multivariate*.

Walaupun serangan XSL masih bersifat teori dan belum dipraktikan pada contoh ciphertext besar, serangan XSL merupakan bentuk ancaman terhadap beberapa algoritma *block cipher* seperti *Rijndael*.

DAFTAR PUSTAKA

[1] T. Courtois, Nicolas & Pieprzyk, Josef (2002). Cryptanalysis of Block Cipher with Overdefined Systems of Equations. *http://eprint.iacr.org/2002/044.pdf*