Studi Perbandingan SEAL (Software-Optimized Encryption Algorithm) dengan Stream Cipher Biasa

Franciscus Borgias Dian Paskalis - 13507048

Teknik Informatika, Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganeca 10, Bandung
e-mail: dianpaskalis@yahoo.com

ABSTRAK

Stream cipher adalah cipher simetris yang beroperasi dengan transformasi bervariasi waktu pada digit plainteks individu. Dalam sebuah stream cipher, rangkaian digit plainteks, *m0m1...*, dienkripsi menjadi rangkaian digit cipherteks *c0c1...* sebagai berikut : sebuah rangkaian pseudorandom *s0s1...*, disebut kunci berjalan atau aliran kunci, dihasilkan oleh *finite state automaton* yang state awalnya ditentukan sebuah kunci rahasia. Digit aliran kunci ke i hanya bergantung pada kunci rahasia dan pada digit plainteks sebelumnya (i-1). Kemudian digit cipherteks ke i didapat dengan mengombinasikan digit plainteks ke i dengan digit aliran kunci ke i.

Stream cipher memiliki beberapa keuntungan yang membuat mereka cocok untuk beberapa aplikasi. Mereka biasanya lebih cepat dan memiliki kompleksitas *hardware* yang lebih rendah daripada block cipher. Mereka juga cocok digunakan ketika *buffering* itu terbatas karena digit-digitnya dienkripsi dan didekripsi secara individual. Selain itu, *synchronous stream cipher* tidak terpengaruh propagasi kesalahan.

SEAL (Software-Optimized Encryption Algorithm) merupakan salah satu bentuk dari stream cipher. Hal yang membedakan SEAL dari stream cipher biasa adalah SEAL beroperasi sangat cepat dan dioptimasikan untuk mesin 32-bit dan RAM yang besar. SEAL biasanya sangat cocok digunakan untuk aplikasi seperti mengenkripsi *hard drive*. SEAL berfungsi paling cepat dan dioptimasikan untuk bekerja pada software. SEAL menggunakan kunci 160 bit untuk enkripsi dan dianggap sangat aman.

Versi pertama SEAL dipublikasikan oleh Phillip Rogaway dan Don Coppersmith pada tahun 1994. Versi yang banyak digunakan sekarang, dipublikasikan pada tahun 1997, adalah versi 3.0. SEAL dilindungi dua paten untuk IBM di USA.

Kata kunci: cipher, *stream*, SEAL, algoritma, enkripsi perbandingan, kunci, cepat.

1. PENDAHULUAN

Tulisan rahasia mungkin adalah metode pertama yang digunakan secara luas untuk komunikasi secara aman melalui *channel* yang tidak aman. Teks rahasia, dikenal sebagai cipher teks, dapat dibaca oleh pembaca yang berwenang. Metode komunikasi secara aman menjadi agak lemah jika ada penyerang yang ahli dalam tulisan rahasia sehingga ia dapat mengetahui pembacaan dokumen. Munculnya komputer memberikan desainer dan kriptanalis alat baru yang kuat untuk komputasi dengan cepat. Algoritma kriptografi baru pun didesain dan serangan-serangan baru dikembangkan untuk memecahkan mereka. Kebutuhan kriptologi disebabkan adanya munculnya aplikasi komputer yang baru dan kebutuhan perlindungan informasi. Komputasi terdistribusi dan pembagian informasi bersama dalam jaringan komputer termasuk dalam aplikasi baru itu, terkadang kebutuhan alat yang menyediakan jaminan pengiriman informasi secara aman sangat dibutuhkan secara dramatis.

Kerahasiaan pesan pada zaman sekarang merupakan masalah keamanan yang penting sehingga stream cipher sangat dibutuhkan. Stream cipher adalah kelas penting dari algoritma enkripsi. Karakter individu pesan plainteks dienkripsi satu per satu menggunakan transformasi enkripsi yang bervariasi sesuai waktu. Stream cipher biasanya lebih cepat dan tidak sekompleks block cipher di sirkuit hardware. Stream cipher juga lebih cocok digunakan ketika *buffering* terbatas atau ketika karakter harus diproses secara individu begitu diterima. Karena mereka juga memiliki propagasi kesalahan yang terbatas atau malah tidak sama sekali, stream cipher juga menguntungkan pada situasi di mana kesalahan transmisi sangat mungkin terjadi.

Stream cipher sering digunakan di aplikasi yang plainteksnya muncul dengan jumlah yang tidak diketahui panjangnya – contohnya, koneksi wireless yang aman. Jika sebuah block cipher digunakan pada tipe aplikasi ini, desainer akan perlu memilih efisiensi tranmisi atau

kompleksitas implementasi karena block cipher tidak dapat bekerja secara langsung pada block yang lebih pendek dari panjang blocknya. Contohnya, jika sebuah 128-bit block cipher menerima 32-bit plainteks yang terpisah, tiga perempat data yang dikirim akan berupa *padding*. Stream cipher menghilangkan masalah *padding* ini dengan beroperasi pada unit terkecil yang dapat ditransmisikan (biasanya bytes). Contoh stream cipher adalah RC4, SEAL, A5, Oryx, dan lain-lain.

SEAL merupakan salah satu jenis stream cipher yang memproses unit atau input data, pesan atau informasi pada satu saat. Unit atau data umumnya adalah byte. Algoritma ini tidak harus menunggu sejumlah data masukan, pesan atau informasi tertentu sebelum diproses, atau menambahkan byte tambahan untuk mengenkrip.

2. STREAM CIPHER

Stream cipher adalah cipher simetrik yang beroperasi dengan tranformasi yang bervariasi dengan waktu pada digit plainteks individu.

2.1 Kunci Berjalan (*Running Key*)

Dalam sebuah stream cipher, kunci berjalan (*running key*) atau aliran kunci (*keystream*), adalah rangkaian yang digabung, digit per digit, ke rangkaian plainteks untuk mendapatkan rangkaian cipherteks. Kunci berjalan ini dihasilkan oleh sebuah *finite state automaton* yang disebut *running key generator* atau *keystream generator*.

2.2 *Karakteristik*

Untuk parameter dasarnya :
- Ukuran panjang kuncinya rahasia
- Ukuran IV : hampir setiap aplikasi stream cipher kenyataanya membutuhkan sebuah kunci yang digunakan berkali-kali, dengan IV yang berbeda
- Panjang keluaran maksimum : seberapa panjang rangkaian aliran kunci yang perlu dihasilkan?

Dari segi performansi :
- Kecepatannya mempengaruhi waktu inisialisasi, waktu inisialisasi kembali (kunci rahasia yang sama, IV baru), dan *throughput*
- Ukurannya mempengaruhi ukuran hasil implementasi dan konsumsi sumber daya (power)

Dari segi platform yang dapat diaplikasikan pada:
- Prosesor 32-bit atau 64-bit
- Prosesor 8-bit
- FPGA
- ASIC yang dibangun dengan tujuan tertentu

Stream cipher juga memiliki kelebihan seperti :
- Dapat melakukan pengubahan plainteks ke cipherteks dengan *bitwise* XOR atau dengan transformasi yang lebih kompleks pada blok plainteks/ cipherteks
- Dapat menghasilkan jutaan blok aliran kunci secara efisien tanpa melakukan putaran melalui jutaan generator *state*
- Dapat digabungkan dengan fungsionalitas lain, misalnya mekanisme integritas

2.3 *Linear Feedback Shift Register* (LFSR)

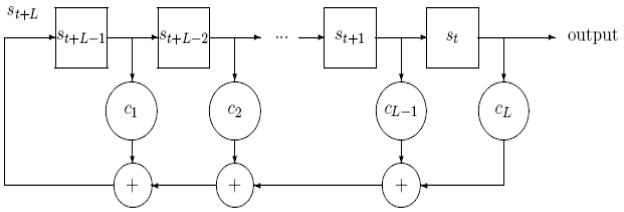
LFSR adalah komponen utama dalam banyak *running key generator* untuk aplikasi stream cipher karena mereka cocok untuk implementasi hardware dan menghasilkan rangkaian dengan property statistic yang bagus. LFSR mengacu pada suatu *feedback shift register* dengan suatu fungsi umpan balik linear.

Suatu LFSR dengan panjang L pada Fq adalah *finite state automaton* yang menghasilkan rangkaian elemen - *semi-infinite* Fq, S = (St)t ≥ 0 = S0S1…, memenuhi pengulangan linear derajat relasi L atas Fq

$$St + L = \sum_{i=1}^{L} CiSt + L - i, \forall t \geq 0 \ldots..(1)$$

L mengoefisienkan C1, …, CL adalah elemen Fq. Mereka disebut koefisien umpan balik dari LFSR.

Sebuah LFSR dengan panjang L pada Fq memiliki bentuk sebagai berikut :



Gambar 1. Diagram proses LFSR

Register yang berisi L *delay cell*, disebut *stage*, masing-masing berisi sebuah elemen Fq. Isi dari L *stage*, St, …, St+L-1, membentuk *state* LFSR. L *stage* awalnya terisi dengan L elemen, S0, …, SL-1, yang bisa terpilih dalam Fq; mereka membentuk *state* awal *register*.

Shift register dikendalikan oleh *clock* eksternal. Pada setiap unit waktu, tiapdigit digeser satu *stage* ke kanan. Isi dari *stage* terkanan St adalah hasil keluaran. Isi yang baru pada *stage* terkiri adalah bit umpan balik, St+L. Bit tersebut didapat dari kombinasi linier isi *register stage*, di mana koefisien kombinasi linier diberikan oleh koefisien umpan balik LFSR :

$$St + L = \sum_{i=1}^{L} CiSt + L - i \ldots\ldots(2)$$

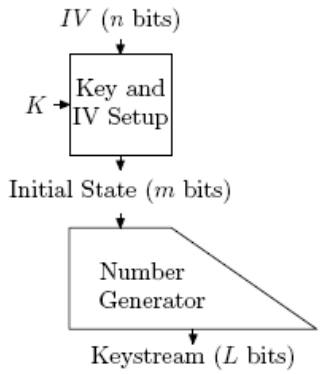
Karena itu, LFSR mengimplementasikan perulangan linier derajat relasi L :

$$St + L = \sum_{i=1}^{L} CiSt + L - i, \forall t \geq 0 \ldots\ldots(1)$$

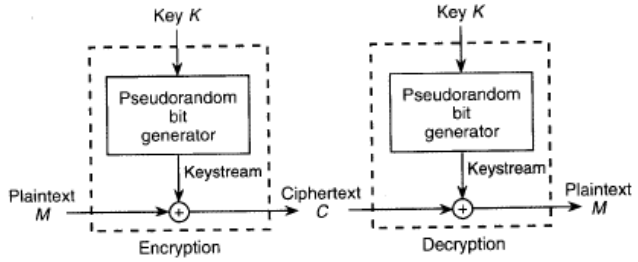
2.4 *Pseudorandom Number Generator* (PRNG)

Pseudorandom Number Generator (PRNG) adalah algoritma yang menghasilkan rangkaian angka yang tidak sebenarnya acak. PRNG juga biasanya dipakai dalam stream cipher dan biasanya dipakai saat diaplikasikan pada software.

Keluaran dari PRNG hanya mendekati beberapa ciri angka acak. Meskipun angka acak yang sebenarnya dipercaya dapat dihasilkan menggunakan *hardware number generator*, angka *pseudorandom* sangat penting dalam simulasi dan merupakan pusat dalam praktek dan teori kriptografi. Analisis matematika yang hati-hati diperlukan untuk meyakinkan bahwa PRNG menghasilkan angka yang cukup acak untuk memenuhi kebutuhan penggunaannya.



Gambar 2. Diagram proses pembentukan keystream dari PRNG dengan IV



Gambar 3. Diagram proses stream cipher yang menggunakan PNRG pada umumnya

2.5 Masalah yang Ada Pada Stream Cipher

"*Sebuah stream cipher pada umumnya adalah random number generator (RNG) berkunci yang keluarannya tergabung dengan data plainteks, biasanya menggunakan exclusive-OR sehingga menghasilkan cipherteks. Sayangnya, kriptanalisis menganggap bahwa sejumlah besar plainteks dan cipherteks yang bersangkutan (known-plaintext) dapat diakses oleh lawan. Di bawah kondisi-kondisi ini, exclusive-OR tidak memiliki kekuatan sama sekali dan dapat segera diketahui rangkaian RNG-nya. Kemudian dengan mengetahui desain RNG dan keluarannya, lawan dapat memiliki kesempatan yang baik untuk merekonstruksi internal state RNG dan hal itu berarti memecahkan cipher."* (Ritter)

Permasalahan ini diarahkan lebih jauh oleh Bruce Schneier, "Secara klasik, stream cipher didasarkan pada teori matematik yang sangat baik. Teori ini dapat digunakan untuk mencari ciri yang baik untuk cipher tetapi juga dapat digunakan untuk menemukan serangan baru terhadap cipher tersebut" (Schneier, 1996, p. 420). Permasalahan lain dengan stream cipher adalah sulit untuk diimplementasikan pada *software* tetapi pada sisi lain mereka sangat mudah diimplementasikan pada *hardware*.

3. SEAL

SEAL (Software-Optimized Encryption Algorithm) merupakan salah satu bentuk dari stream cipher. Hal yang membedakan SEAL dari stream cipher biasa adalah SEAL beroperasi sangat cepat dan dioptimasikan untuk mesin 32-bit dan RAM yang besar. SEAL, sesuai namanya, dioptimasikan untuk bekerja pada software.

3.1 Latar Belakang

Enkripsi harus dapat sering dilakukan pada jumlah data yang besar, kebutuhan minimum yang kadang didukung adanya hardware kriptografi yang sesuai. Sayangnya, hardware kriptografi sering tidak bisa didaptkan dan keamanan data menjadi dikorbankan karena biaya software kriptografi dinilai terlalu besar.

Biaya komputasi software kriptografi adalah fungsi dari algoritma dan kualitas implementasi yang ada. Dengan mengesampingkan implementasi, sebuah algoritma kriptografi yang dirancang untuk berjalan dengan baik di hardware tidak akan berjalan pada software sebaik algoritma yamg dirancang untuk eksekusi software. Apa yang banyak diperlukan sekarang adalah rancangan metode enkripsi yang dioptimasikan untuk software yang baik untuk komputer yang digunakan untuk tujuan umum. Untuk masalah seperti di inilah maka SEAL dirancang.

3.2 Karakteristik

Karena SEAL merupakan salah satu bentuk stream cipher maka tentunya banyak karakteristik stream cipher juga dimiliki oleh SEAL. Selain karakteristik stream cipher yang telah disebutkan, ada juga karakteristik lainnya sebagai berikut :

Kebutuhan RAM – pengaplikasian SEAL membutuhkan mesin yang memiliki RAM besar agar dapat berjalan baik. Hal ini disebabkan oleh algoritma dan inisialisasi table yang digunakan.

Pemrosesan kunci awal – pada tipikal aplikasi yang memerlukan enkripsi dengan cepat diperlukan adanya pengaturan kunci yang digunakan pada suatu proses kriptografi. Kunci ini disusun pada pengaturan sesi menggunakan table yang dibuat menggunakan *Secure Hash Algorithm*. Biasanya penyusunan ini memerlukan waktu beberapa milidetik. SEAL menggunakan kunci tersebut. Dapat disimpulkan bahwa SEAL tidak cocok digunakan untuk aplikasi yang memerlukan banyak rentetan pengaturan kunci.

Fungsi Pseudorandom yang Menambahkan Panjang – Variabel Keluaran dan Panjang Kunci – SEAL adalah tipe kriptografi objek yang disebut *pseudorandom function family* (PRF). PRF dapat digunakan untuk membuat stream cipher yang baik. Dalam sebuah stream cipher enkripsi pesan tidak hanya bergantung pada kunci dan pesan tetapi juga posisi pesan dalam aliran data.

Platform Sasaran – semua prosesor 32-bit modern.

Cipher Berdasar Tabel – cipher menjadi lebih cepat dan mudah didesain. Selain itu, kita juga dapat memperoleh difusi yang sangat cepat.

3.3 Algoritma Umum

Algoritma yang dipakai untuk SEAL didesain untuk berjalan dengan cepat di software khususnya di platform prosesor 32-bit.

$$
\begin{array}{l}
\textbf{procedure } \mathsf{Initialize}(n, \ell,\ A, B, C, D,\ n_1, n_2, n_3, n_4) \\
\\
A \leftarrow n \oplus R[4\ell]; \\
B \leftarrow (n \ggg 8) \oplus R[4\ell + 1]; \\
C \leftarrow (n \ggg 16) \oplus R[4\ell + 2]; \\
D \leftarrow (n \ggg 24) \oplus R[4\ell + 3]; \\
\\
\textbf{for } j \leftarrow 1 \textbf{ to } 2 \textbf{ do} \\
\quad P \leftarrow A \ \&\ \text{0x7fc};\ B \leftarrow B + T[P/4];\ A \leftarrow A \ggg 9; \\
\quad P \leftarrow B \ \&\ \text{0x7fc};\ C \leftarrow C + T[P/4];\ B \leftarrow B \ggg 9; \\
\quad P \leftarrow C \ \&\ \text{0x7fc};\ D \leftarrow D + T[P/4];\ C \leftarrow C \ggg 9; \\
\quad P \leftarrow D \ \&\ \text{0x7fc};\ A \leftarrow A + T[P/4];\ D \leftarrow D \ggg 9; \\
\\
(n_1,\ n_2,\ n_3,\ n_4) \leftarrow (D,\ B,\ A,\ C); \\
\\
P \leftarrow A \ \&\ \text{0x7fc};\ B \leftarrow B + T[P/4];\ A \leftarrow A \ggg 9; \\
P \leftarrow B \ \&\ \text{0x7fc};\ C \leftarrow C + T[P/4];\ B \leftarrow B \ggg 9; \\
P \leftarrow C \ \&\ \text{0x7fc};\ D \leftarrow D + T[P/4];\ C \leftarrow C \ggg 9; \\
P \leftarrow D \ \&\ \text{0x7fc};\ A \leftarrow A + T[P/4];\ D \leftarrow D \ggg 9;
\end{array}
$$

Gambar 4. Algoritma inisialisasi A, B, C, D

$$
\begin{array}{l}
\textbf{function } \mathsf{SEAL}(a, n, L) \\
y = \lambda; \\
\textbf{for } \ell \leftarrow 0 \textbf{ to } \infty \textbf{ do} \\
\quad \mathsf{Initialize}(n, \ell,\ A, B, C, D,\ n_1, n_2, n_3, n_4); \\
\quad \textbf{for } i \leftarrow 1 \textbf{ to } 64 \textbf{ do} \\
\qquad P \leftarrow A \ \&\ \text{0x7fc};\quad B \leftarrow B + T[P/4];\ A \leftarrow A \ggg 9;\ B \leftarrow B \oplus A; \\
\qquad Q \leftarrow B \ \&\ \text{0x7fc};\quad C \leftarrow C \oplus T[Q/4];\ B \leftarrow B \ggg 9;\ C \leftarrow C + B; \\
\qquad P \leftarrow (P + C) \ \&\ \text{0x7fc};\ D \leftarrow D + T[P/4];\ C \leftarrow C \ggg 9;\ D \leftarrow D \oplus C; \\
\qquad Q \leftarrow (Q + D) \ \&\ \text{0x7fc};\ A \leftarrow A \oplus T[Q/4];\ D \leftarrow D \ggg 9;\ A \leftarrow A + D; \\
\qquad P \leftarrow (P + A) \ \&\ \text{0x7fc};\ B \leftarrow B \oplus T[P/4];\ A \leftarrow A \ggg 9; \\
\qquad Q \leftarrow (Q + B) \ \&\ \text{0x7fc};\ C \leftarrow C + T[Q/4];\ B \leftarrow B \ggg 9; \\
\qquad P \leftarrow (P + C) \ \&\ \text{0x7fc};\ D \leftarrow D \oplus T[P/4];\ C \leftarrow C \ggg 9; \\
\qquad Q \leftarrow (Q + D) \ \&\ \text{0x7fc};\ A \leftarrow A + T[Q/4];\ D \leftarrow D \ggg 9; \\
\qquad y \leftarrow y \ \|\ B + S[4i-4] \ \|\ C \oplus S[4i-3] \ \|\ D + S[4i-2] \ \|\ A \oplus S[4 \\
\qquad \textbf{if } |y| \geq L \textbf{ then return } (y_0 y_1 \ldots y_{L-1}); \\
\qquad \textbf{if } \mathit{odd}(i) \textbf{ then } (A,\ B,\ C,\ D) \leftarrow (A + n_1,\ B + n_2,\ C \oplus n_1,\ D \oplus n_2) \\
\qquad\qquad \textbf{else } (A,\ B,\ C,\ D) \leftarrow (A + n_3,\ B + n_4,\ C \oplus n_3,\ D \oplus n_4);
\end{array}
$$

Gambar 5. Algoritma proses enkripsi SEAL

Notasi :

Kita sebut string 32-bit “word” dan string 8-bit “byte”. String kosong ditulis sebagai λ . Angka hexadecimal ditulis dengan awalan “0x” dan kemudian menggunakan symbol “a”-“f” untuk merepresentasikan angka decimal 10-15. Dengan y >>> t kita menuliskan pergeseran sirkuler y ke kanan sebanyak t bit. Dengan cara yang sama, y <<< t berarti pergeseran sirkuler y ke kiri sebanyak t bit. Simbol “&”, “V”, dan “⊕” menandakan *bitwise* AND, OR, dan XOR; dengan A kita menandakan komplemen A. A +B berarti jumlah, mengabaikan *carry*, *unsigned integer* A dan B; ini adalah jumlah dari mod 2^{32} angka A dan B. “||” menyatakan operator konkatenasi. *Odd*(.) menghasilkan true jika dan hanya jika argument adalah angka ganjil.

4. ANALISIS

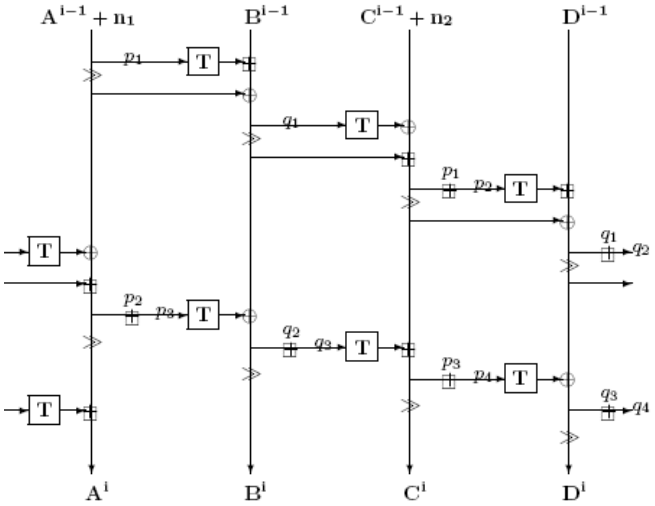
Stream cipher meskipun cepat ternyata mudah dianalisis dan dipecahkan. Hal ini disebabkan ketergantungannya pada operasi XOR maupun LFSR dan PRNG yang hanya diproses dengan operasi feedback untuk kunci berikutnya. Selain itu, banyak stream cipher meskipun efisien di aplikasi *hardware* ternyata tidak efisien pada implementasi *software*.

Pada SEAL, metode yang dipakai untuk menanggulangi permasalahan *software* akan banyaknya aliran cipher adalah dengan memproses tiga tabel turunan kunci terlebih dahulu. Prosedur pemrosesan awal ini berasal dari *Secure Hash Algorithm* (SHA). SHA dikembagkan oleh NIST dan NSA, dan membutuhkan pesan sekurangnya sepanjang 2^{64} bit dan menghasilkan 160-bit pesan. Tabel-tabel ini membentuk s-box yang merupakan sebuah 2 kilobyte tabel yang berisi 9 x 32 bit (Schneier, 1996, p.399). SEAL juga memerlukan empat register 32-bit (A, B,C, D) yang diubah-ubah pada delapan putaran iterasinya. Pada tiap putaran, bit-bit dari sebuah register digunakan untuk pencarian s-box. Nilai s-box ditambahkan atau di XOR kan dengan isi dari register yang berbeda. Register yang awal kemudian digeser. Proses ini dilakukan untuk 8 putaran, kemudian nilai register (A, B, C, D) ditambahkan ke aliran kunci.

Untuk memperumit enkripsi, ternyata nilai register pada algoritma SEAL ditambahkan atau di XOR dengan nilai dari salah satu s-box tabel (Schneier, 1996, p. 399). Kunci pertimbangan untuk SEAL adalah agar aplikasi *software* dapat berjalan dengan cepat, tetapi sayangnya membutuhkan waktu yang lama untuk memproses terlebih dahulu dan juga memerlukan memori yang besar untuk menyimpannya. Kekuatan enkripsi SEAL memang kuat, tetapi bisa menjadi tidak jelas karena adanya kriptanalisis yang dikenakan ke algoritma (Schneier, 1996, p. 400).

5. SERANGAN PADA SEAL

Berikut ini adalah serangan yang dapat dilakukan pada SEAL yang versinya disederhanakan.



Gambar 6. Generator SEAL yang kedua (iterasi ke i)

– ⊕ adalah fungsi XOR
– □ adalah jumlah (mod 2^{32})
– >> rotasi kanan sebanyak 9 bit
–*p*1 sampai *p*4 dan *q*1 sampai *q*4 adalah masukan tabel T yang didapat dari 9 bit 2 sampai 11 dari nilai A, B, C, dan D

Dengan mengganti semua penambahan mod 2^{32} pada gambar 5 dengan XOR, serangan bisa dibagi menjadi 4 langkah.

Anggap D^{i-1}, C^i, dan D^i berhubungan. Kemudian anggap mereka adalah kata masukan. Nyatakan tabel keluaran pada gambar 5 dengan : T1 = T [p2], T2 = T [q3] dan T3 = T [p4] sehingga bisa didapatkan :

$$(D^{i-1}+T1)\oplus(C^i<<9+T2) = (D^i<<18)\oplus(T3<<9) \ldots(3)$$

Langkah 1 :

Turunkan set nilai tabel T yang tidak terurut sampai sebuah konstan 32-bit yang tidak diketahui. Persamaan 3 di atas merepresentasikan awal penurunan ini. Ganti + dengan ⊕ dan D^{i-1}, C^i D^i dengan $X_4 = Y_4^i$, $Y_3 = Y_3^i$, dan $Y_4 = Y_4^i$ didapat :

$$Y_4\oplus Y_3>>9\oplus X_4>>18 = T_3>>9\oplus(T1\oplus T2)>>18\oplus\Delta^i>>9 \ldots.(4)$$

Dengan konstan Δ^i bergantung pada tabel S. T1 dan T2 adalah 2 dari 512 nilai tabel T.

Langkah 2 :

Anggap persamaan di bawah dibuat dengan cara yang sama ke persamaan 4 dari hubungan B^{i-1}dan kata keluaran :

$Y_4 >> 9 \oplus Y_2 \oplus Y_1 >> 9 \oplus X_2 >> 18 \oplus T_3 >> 18 = (T'_1 \oplus T'_2) >> 18 \oplus (T'_3 \oplus T'_4) >> 9 \oplus (S_4^i >> 9 \oplus S_2^{i-1} >> 18 \oplus S_2^i \oplus S_1^i >> 9)$ ….(5)

Di mana

$\alpha^i = S_4^i >> 9 \oplus S_2^{i-1} >> 18 \oplus S_2^i \oplus S_1^i >> 9 \oplus \Delta^i >> 18$ ….(6)

Untuk tiap contoh, kita dapat mengetahui $T_3 \oplus \Delta^i$ dengan mencari kombinasi (T_1, T_2, T_3) yang tepat pada persamaan 4 secara *exhaustive*.

Langkah 3:

Anggap persamaan 7 dihasilkan dari hubungan A^{i-1} dan kata keluaran :

$X_1 >> 18 \oplus Y_1 \oplus Y_4 \oplus T'_2 >> 9 \oplus T'_4 \oplus T_3 >> 9 = n1 >> 18 \oplus S_1^{i-1} 1 >> 18 \oplus S_4^i \oplus S_1^i$….(7)

Kita dapat mengetahui $T'_2 \oplus \Delta^i$ dan $T'_4 \oplus \Delta^i$ dengan mencari kombinasi (T'_1, T'_2, T'_3, T'_4) yang tepat menggunakan nilai α^i dari persamaan 6.

Kemudian kita perlu tahu tabel $T \oplus \Delta^i$, tabel $T \oplus \Delta^j$, α^i dan α^j. Kita kumpulkan contoh :

– $n_1 >> 18 \oplus \beta^i$ di mana $\beta^i = S_1^{i-1} >> 18 \oplus S_4^i \oplus S_1^i \oplus \Delta^i$

– $n_1 >> 18 \oplus \beta^j \oplus \Delta^j >> 9 \oplus \Delta^i >> 9$ dengan nilai T_3 dari tabel $T \oplus \Delta^i$ dan nilai T'^2 dan T'^4 sampai tabel $T \oplus \Delta^j$

XOR semua contoh putaran I dengan semua contoh putaran j. Ketika semua contoh nilai putaran i dan j, bandingkan keduanya dan buat nilai $n_1 >> 18 \oplus \beta^i$ yang benar.

Langkah 4:

Pada langkah ini akhirnya kita turunkan masukan dan keluaran tabel T dari persamaan berikut :

$$p_1 = ((X_1 \oplus n_1 \oplus S_1^{i-1}) \& 0x7fc)/4 \quad \ldots\ldots(8)$$

Pada persamaan ini p_1 adalah masukan tabel T. Kita telah melihat tiga langkah pertama untuk menurunkan nilai T_1 dari contoh masukan dan keluaran SEAL.

Akhirnya kita menurunkan beberapa nilai :

$T [p \oplus \delta^i] \oplus \Delta^i$

dengan $\delta^i = ((S_1^{i-1} \oplus \beta^i << 18) \& 0x7fc)/4$.

Mungkin kitadapat melanjutkan untuk memecahkan SEAL versi sederhana dengan menemukan nilai (n_1, n_2, n_3, n_4) dan mencoba memecahkan generator sebelumnya lalu menemukan tabel R, tetapi hal itu tidak akan dibahas di sini. Dapat dikatakan bahwa serangan yang dibahas di atas dapat diadaptasi ke aplikasi SEAL yang sebenarnya, tetapi belum diketahui apakah kompleksitasnya masih dalam batas yang wajar untuk diusahakan dipecahkan.

6. PERBANDINGAN LAINNYA

- Tabel yang digunakan di SEAL sebanyak tiga buah dan dibangkitkan di awal, sedangkan stream cipher hanya menggunakan kurang dari tiga tabel.
- SEAL memang lebih cepat daripada stream cipher biasa, tetapi hal itu memiliki konsekuensi pada pemakaian memori untuk tabel dan algoritma SEAL yang jauh lebih besar dibanding stream cipher biasa.
- Algoritma SEAL lebih rumit dan sangat sulit untuk dipecahkan dibanding stream cipher yang umumnya sederhana.

7. KESIMPULAN

- Stream cipher dapat mengeksekusi dengan lebih cepat dibanding block cipher.
- Stream cipher menggunakan LFSR (*Linerar Feedback Shift Register*) untuk *hardware* dan PRNG (*Pseudorandom Number Generator*) untuk *software*.
- SEAL (*Software-Optimized Encryption Algorithm*) yang merupakan salah satu bentuk stream cipher merupakan salah satu bentuk stream cipher yang tercepat dan dianggap aman.
- SEAL dioptimasikan untuk bekerja pada *software* pada prosesor 32-bit.
- SEAL dapat lebih cepat dari algoritma stream cipher biasa karena algoritmanya dan pembangkitan tabel untuk aliran kunci pada inisialisasinya. Karena hal inilah maka SEAL membutuhkan RAM yang besar.
- Karena SEAL belum pernah dianalisis sampai benar-benar selesai, sampai saat ini belum diketahui serangan-serangan yang layak dilakukan ke SEAL sehingga SEAL dianggap aman.

REFERENSI

[1] Clarke, Jason. M. 2004. “ENCRYPTION, Cryptographic Techniques and Implications”, halaman 12-14.

[2] Handschuh, Helena; Gilbert, Henri. 1997. “Fast Software Encryption”, vol. 1267 dari *Lecture Notes in Computer Science*, halaman 1-12.

[3] Jabiullah, M.Ismail; Rahman, M. Lutfar. 2006. “Stream Cipher for Message Confidentiality”. Halaman 360-361.

[4] Rogaway, Phillip; Coppersmith. 1997. “A Software-Optimized Encryption Algorithm”. Halaman 1-10.

[5] Zeng, Kencheng; Yang, Chung-Huang, et al. 1991. "Pseudorandom Bit Generators in Stream-Cipher Cryptography". Halaman 9.
[6] Pseudorandom number generator. http://en.allexperts.com/e/p/ps/pseudorandom_number_generator.htm. Dibuka tanggal 24 Maret 2010.
[7] Stream Cipher. http://www-rocq.inria.fr/who/Anne.Canteaut/encyclopedia.pdf. Dibuka pada tanggal 23 Maret 2010.
[8] Stream Ciphers. http://www.cs.bilkent.edu.tr/~selcuk/teaching/cs519/cs519.5.ppt. Dibuka tanggal 4 Maret 2010.
[9] Stream ciphers – what does industry want?. http://www.ecrypt.eu.org/stvl/sasc/slides21.pdf. Dibuka tanggal 24 Maret 2010.