STUDI DAN ANALISIS
ALGORITMA STREAM CIPHER SOSEMANUK

Jansen – NIM : 13506028

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if16028@students.if.itb.ac.id

Abstraksi

Banyaknya algoritma enkripsi pada jaman sekarang tidak terlepas dari usaha manusia untuk membuat suatu algoritma yang *robust*, yang tidak dapat dipecahkan serta cepat dalam komputasi baik ketika enkripsi maupun dekripsi. Banyak pula algoritma yang gagal mencapai tujuan tersebut. Meski demikian, semakin banyak pula algoritma-algoritma baru yang dihasilkan yang tampak lebih menjanjikan. Salah satunya adalah algoritma stream cipher Sesomanuk.

Sosemanuk adalah sebuah algoritma *cipher* aliran yang ditemukan oleh Come Berbain, Olivier Billet, Anne Canteaut, Nicolas Courtois, Henri Gilbert, Louis Goubin, Aline Gouget, Louis Granboulan, Cédric Lauradoux, Marine Minier, Thomas Pornin dan Hervé Sibert. Algoritma ini diikut sertakan dalam lomba proyek eSTREAM pada kategori *software-oriented*. Kemudian bersama-sama dengan HC-128, Rabbit dan Salsa20/12, algoritma ini terpilih menjadi finalis eSTREAM.

Sosemanuk berasal dari bahasa *India Cree*, yaitu "ular salju". Sesuai dengan definisi tersebut, Sosemanuk menggunakan desain dasar dari algoritma *cipher* aliran Snow dan dikombinasikan dengan algoritma *cipher* blok Serpent.

Kata kunci: *cipher* aliran, *cipher* blok, Snow, Serpent, Sosemanuk, enkripsi, dekripsi, kriptografi.

1. Pendahuluan

Semakin berkembangnya jaman dan ilmu pengetahuan, semakin banyak pula para hacker baru maupun para cracker baru bermunculan. Tindakan yang mereka lakukan adalah mengambil data-data milik orang lain, mengganti atau bahkan mungkin merusakkannya.

Orang-orang kemudian merasakan kebutuhan akan sebuah mekanisme agar data-data mereka aman. Ketika dikirimkan, mereka menginginkan data tersebut tidak dapat dibaca oleh pihak yang tak bersangkutan dan ketika sampai di pihak yang bersangkutan, pesan dapat kembali dibaca seperti sedia kala.

Mekanisme tersebutlah yang dinamakan kriptografi. Kriptografi telah berkembang seiring dengan berlalunya waktu, mulai dari kriptografi yang sangat sederhana sampai kepada kriptografi modern yang sangat rumit. Namun di satu sisi, algoritma-algoritma tersebut banyak yang gagal dalam memenuhi tujuannya. Kegagalan tersebut tidak meredam semangat para ahli untuk membuat sebuah algoritma yang tangguh.

Salah satunya adalah algoritma Sosemanuk yang dalam bahasa India Cree berarti ular salju. Algoritma ini dibuat berdasarkan dua buah algoritma kriptografi: cipher aliran Snow dan algoritma cipher blok Serpent. Dengan harapan kelemahan yang satu akan tertutup oleh kekuatan yang lain, Come Berbain, Olivier Billet, Anne Canteaut, Nicolas Courtois, Henri Gilbert, Louis Goubin, Aline Gouget, Louis Granboulan, Cédric Lauradoux, Marine Minier, Thomas Pornin dan Hervé Sibert kemudian merancang algoritma ini dan mengikut sertakannya pada bulan November 2004 dalam sebuah lomba bernama eSTREAM, sebuah lomba untuk mencari algoritma-algoritma cipher aliran yang tangguh dan dapat digunakan untuk keperluan yang beragam.

Setelah melalui 3 tahap, pada tanggal 15 April 2008 Sosemanuk terpilih bersama dengan 3 algoritma lain menjadi finalis di bidang *software-oriented*.

Panjang kunci pada algoritma ini adalah pada rentang 128 sampai 256 bit. Keamanan algoritma ini adalah 128 bit, terlepas dari panjang kunci.

Berdasarkan arsitektur umum algoritma cipher aliran Snow versi 2, Sosemanuk dibuat untuk memperbaiki algoritma Snow dari dua sisi, kelemahan yang mungkin muncul dan efisiensi.

2. Tipe-Tipe Algoritma Dalam Kriptografi

Algoritma-algoritma dalam kriptografi dapat dibagi menjadi dua jenis, yaitu:

1. Algoritma kunci simetri (*Symmetric-key cryptography*).
   Algoritma-algoritma yang berada pada kategori ini mempunyai kunci untuk enkripsi dan dekripsi yang sama.
2. Algoritma kunci nirsimetri (*Asymmetric-key cryptography*).
   Algoritma-algoritma dalam kategori ini mempunyai kunci untuk enkripsi (kunci publik) yang berbeda dengan kunci untuk dekripsi (kunci rahasia). Kunci publik bersifat umum/tidak rahasia.

Algoritma kunci simetri kemudian dapat pula dibagi menjadi dua jenis, yaitu:

1. *Cipher* aliran (*ctream cipher*).
   Algoritma kriptografi beroperasi pada plainteks/cipherteks dalam bentuk bit tunggal, yang dalam hal ini rangkaian bit dienkripsikan/didekripsikan bit per bit.
2. *Cipher* blok (*block cipher*).
   Algoritma kriptografi beroperasi pada plainteks/cipherteks dalam bentuk blok bit, yang dalam hal ini rangkaian bit dibagi menjadi blok-blok bit yang panjangnya sudah ditentukan sebelumnya.

Algoritma Sesomanuk, seperti Snow, beroperasi pada mode cipher aliran. Sedangkan algoritma Serpent beroperasi pada mode cipher blok.

2.1 *Cipher* Aliran

Pada cipher aliran, pemrosesan plain teks dilakukan secara bit per bit. Mode operasi enkripsi adalah sebagai berikut:

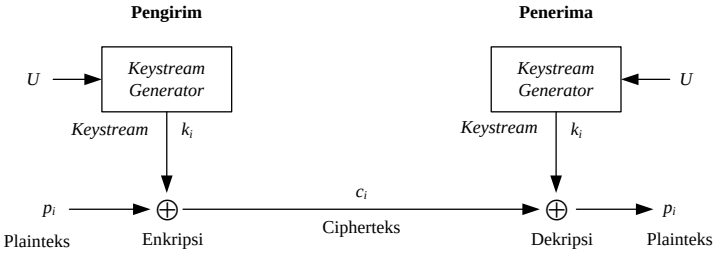
$$c_i = p_i \oplus k_i$$

Dan operasi untuk dekripsi yaitu:

$$p_i = c_i \oplus k_i$$

Notasi $\oplus$ menyatakan operasi XOR, p_i menyatakan bit dari plain teks yang sedang diproses dan c_i menyatakan cipher teks yang dihasilkan.

Dapat dilihat pada operasi di atas bahwa pada cipher aliran, bit hanya mempunyai dua buah nilai, sehingga proses enkripsi hanya menyebabkan dua keadaan pada bit tersebut: berubah atau tidak berubah. Dua keadaan tersebut ditentukan oleh kunci enkripsi yang disebut aliran-bit-kunci (*keystream*). Aliran-bit-kunci (*running key*) dibangkitkan dari sebuah pembangkit yang dinamakan pembangkit aliran-bit-kunci (*keystream generator*).

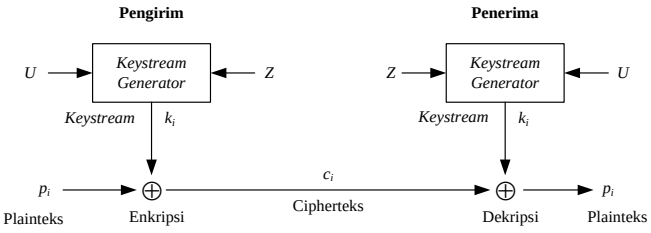


Gambar 1 *Cipher* aliran dengan pembangkit aliran-bit-kunci bergantung pada kunci U.

Keamanan system cipher aliran bergantung sepenuhnya kepada pembangkit aliran-bit-kunci.

Jika pembangkit mengeluarkan aliran-bit-kunci yang seluruhnya nol, maka cipherteks sama dengan plainteks, dan proses enkripsi menjadi tidak ada artinya. Jikapembangkit mengeluarkan aliran-bit-kunci dengan pola berulang, maka algoritma enkripsinya menjadi sama seperti enkripsi dengan XOR sederhana yang memiliki tingkat keamanan yang tidak berarti. Jika pembangkit mengeluarkan aliran-bit-kunci yang benar-benar acak, maka algoritma enkripsinya sama dengan *one-time pad* dengan tingkat keamanan yang sempurna. Keadaan inilah yang hendak dicapai para pembuat algoritma cipher aliran.

Supaya tidak terjadi pola yang berulang, beberapa pembangkit aliran-bit-kunci menggunakan umpan (*seed*), disimbolkan dengan *Z*, atau kadang-kadang disebut juga vector inisialisasi (IV) agar diperoleh kondisi awal yang berbeda pada setiap lelaran.

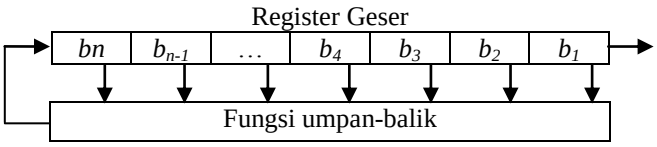


Gambar 2 *Cipher* aliran dengan pembangkit aliran-bit-kunci bergantung pada kunci U dan umpan Z.

2.2 *Feedback Shift Register* (*FSR*)

Dalam menghasilkan aliran-bit-kunci yang acak, seringkali pembangkit aliran-bit-kunci menggunakan FSR. FSR terdiri dari dua bagian:

1. Register geser
   yaitu barisan bit-bit ($b_n b_{n-1} \ldots b_3 b_2 b_1$) yang panjangnya *n* (disebut juga register geser *n*-bit).
2. Fungsi umpan-balik
   yaitu fungsi yang menerima masukan dari register geser dan mengembalikan nilai fungsi ke register geser.



Gambar 3 Bagian-bagian FSR

3. Cipher Aliran Snow

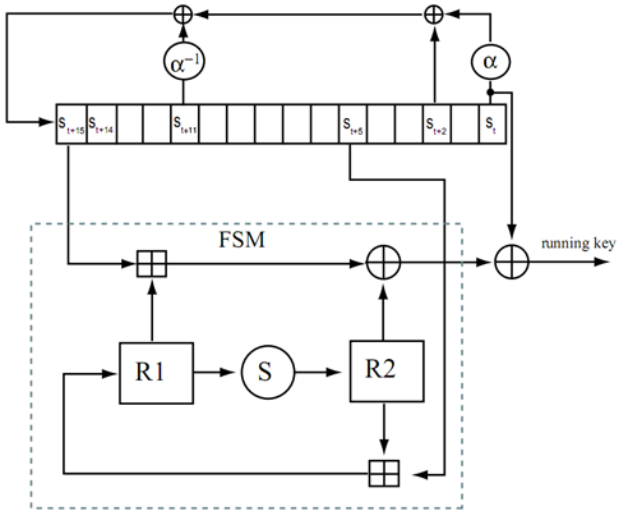
Algoritma yang akan dibahas adalah algoritma Snow versi 2. Versi pertama tidak akan dibahas karena sudah ada kriptanalisisyang berhasil menyerang Snow versi pertama tersebut. Dalam Snow, terdapat dua bagian: LFSR dan FSM (*Finite State Machine*).

3.1 Algoritma Snow

.Operasi-operasi dalam Snow terdiri dari:

1. Inisialisasi kunci, untuk mengisi tabel LFSR, register R1 dan register R2.
2. *Update* LFSR
   $s_{t+16} = \alpha^{-1} s_{t+11} \oplus s_{t+12} \oplus \alpha s_t$
3. *Update* FSM
   - $R1_{t+1} = s_{t+5} \boxplus R2_t$
   - $R2_{t+1} = S(R1_t)$
4. Output FSM
   $F_t = (s_{t+15} \boxplus R1_t) \oplus R2_t$
5. Output Keystream
   $z_t = F_t \oplus s_t$

dengan $\boxplus$ menyatakan penjumlahan modulus 2^{32}, $\oplus$ menyatakan operasi XOR dan *S* menyatakan S-box (akan dijelaskan selanjutnya).



Gambar 4 Skema Algoritma Snow

3.2 S-Box

S-Box, *S*(*w*), adalah seperti fungsi putaran pada Rijndael. Anggap $w = (w_0, w_1, w_2, w_3)$.

Dalam transformasi *MixColumn* Rijndael, setiap 4 byte dianggap sebagai polinomial dalam *y* yang didefinisikan oleh polinomial $x^8 + x^4 + x^2 + x + 1$. Setiap bilangan dapat direpresentasikan dalam polinomial berderajat 3. Perkalian polinom tersebut dapat dianggap sebagai perkalian matriks:

$$\begin{pmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{pmatrix} = \begin{pmatrix} x & x+1 & 1 & 1 \\ 1 & x & x+1 & 1 \\ 1 & 1 & x & x+1 \\ x+1 & 1 & 1 & x \end{pmatrix} \begin{pmatrix} S_R[w_0] \\ S_R[w_1] \\ S_R[w_2] \\ S_R[w_3] \end{pmatrix}$$

dengan $r = (r_0, r_1, r_2, r_3)$ menyatakan hasil dari S-box, S_R menyatakan S-box Rijndael.

3.3 Inisialisasi Kunci

Snow menerima dua buah input parameter: kunci rahasia dan *IV* (*initialization value*) yang berukuran 128 bit. *IV* tersebut kemudian dibagi menjadi 4 bagian (IV_3, IV_2, IV_1, IV_0).

Dalam 128 bit, anggap K = (k_3, k_2, k_1, k_0) yang merupakan kunci rahasia, pengisian LFSR:

$$s_{15} = k_3 \oplus IV_0, \quad s_{14} = k_2, \quad s_{13} = k_1, \quad s_{12} = k_0 \oplus IV_1,$$
$$s_{11} = k_3 \oplus \mathbf{1}, \quad s_{10} = k_2 \oplus \mathbf{1} \oplus IV_2, \quad s_9 = k_1 \oplus \mathbf{1} \oplus IV_3, \quad s_8 = k_0 \oplus \mathbf{1},$$

$$s_7 = k_3, \quad s_6 = k_2, \quad s_5 = k_1, \quad s_4 = k_0,$$
$$s_3 = k_3 \oplus \mathbf{1}, \quad s_2 = k_2 \oplus \mathbf{1}, \quad s_1 = k_1 \oplus \mathbf{1}, \quad s_0 = k_0 \oplus \mathbf{1},$$

di mana **1** menyatakan vektor 32 bit (1, 1, …, 1).

Dalam 256 bit, anggap K = (k_7, k_6, k_5, k_4, k_3, k_2, k_1, k_0) yang merupakan kunci rahasia, pengisian LFSR:

$$s_{15} = k_7 \oplus IV_0, \quad s_{14} = k_6, \quad s_{13} = k_5, \quad s_{12} = k_4 \oplus IV_1$$
$$s_{11} = k_3, \quad s_{10} = k_2 \oplus IV_2, \quad s_9 = k_1 \oplus IV_3, \quad s_8 = k_0,$$
$$s_7 = k_7 \oplus \mathbf{1}, \quad s_6 = k_6 \oplus \mathbf{1} \quad \ldots, \quad s_0 = k_0 \oplus \mathbf{1}.$$

Kemudian R1 dan R2 diisi dengan 0 dan cipher dijalankan 32 kali tanpa menghasilkan keluaran apapun. Baru setelah 32 kali, keluaran dihasilkan.

4. Cipher Blok Serpent

Berbeda dengan Snow yang merupakan cipher aliran, Serpent merupakan cipher blok.

4.1 Algoritma Serpent

Serpent beroperasi pada 128 bit dan memiliki putaran sebanyak 32 buah. Kunci pada Serpent dapat bervariasi, namun biasanya dibulatkan menjadi 128, 192, atau 256 bit dengan cara melakukan padding "1" dan diikuti dengan "0" sampai 256 bit.

Terdapat *Initial Permutation* (*IP*) yang dioperasikan pada plain teks *P* untuk menghasilkan B_0 yang kemudian dioperasikan pada putaran pertama menghasilkan B1 dan seterusnya sampai menghasilkan B_{32}. B_{32} tersebut dioperasikan dengan *Final Permutation* (*FP*) untuk menghasilkan cipher teks *C*.

$B_0 = IP(P)$
$B_{i+1} = R_i(B_i)$
$C = FP(B_{32})$

di mana

$R_i(X) = L(S_i(X \oplus K_i))$; $i = 0, .., 30$
$R_i(X) = S_i(X \oplus K_i) \oplus K_{32}$; $i = 31$

S menyatakan operasi S-box $S_{i \bmod 8}$ sebanyak 32 kali secara paralel dan L menyatakan transformasi linier.

4.2 S-Box

S-box Serpent dapat dibuat dengan pseudocode:

```
index := 0
repeat
  currentsbox := index modulo 32;
  for i:=0 to 15 do
    j := sbox[(currentsbox+1) modulo
          32][serpent[i]];
    swapentries(sbox[currentsbox][i],
                sbox[currentsbox][j]);
  if sbox[currentsbox][.] has the desired
    properties, save it;
  index := index + 1;
until 8 S-boxes have been generated
```

4. Cipher aliran Sosemanuk

Sesuai yang telah dijelaskan sebelumnya, Sosemanuk memiliki panjang kunci antara 128 sampai 256 bit. Panjang kunci apapun akan menghasilkan keamanan sebesar 128 bit.

Adapun beberapa alasan mengapa pemilihan cipher blok jatuh pada Serpent:

- Serpent telah diteliti selama proses seleksi AES dan keamanannya sudah terjamin.
- Serpent tidak membutuhkan tabel data statik, dan selanjutnya hanya menambahkan sedikit simpanan data atau bahkan tidak sama sekali.
- Fungsi putaran pada Serpent dioptimasi untuk operasi data 32 bit, jelas sama dengan data pada Sosemanuk.
- Sosemanuk membutuhkan cipher blok untuk penjadwalan kunci dan injeksi *IV*.

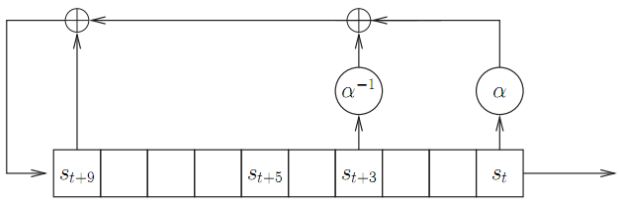
4.1 Algoritma Sosemanuk

Pada Sosemanuk, terdapat dua primitif:

1. Serpent1
   Serpent1 adalah satu putaran Serpent, tanpa penambahan kunci dan transformasi linier. Serpent1 merupakan operasi Serpent yang menggunakan hanya S_2 dari delapan S-box, S_0 sampai S_7 yang terdefinisi.
2. Serpent24
   Serpent24 adalah Serpent yang memiliki putaran hanya sebanyak 24, di mana Serpent yang asli memiliki putaran sebanyak 32. Putaran terakhir Serpent24 merupakan putaran lengkap dengan transformasi linier dan sebuah XOR dengan subkunci ke-25.
   $R_{23}(X) = L(\mathrm{S23}(X \oplus \mathrm{K23})) \oplus K_{24}$

4.1.1 LFSR

LFSR harus berjumlah kecil supaya ukuran baik kode maupun penyimpanan menjadi lebih baik sehingga dipillih LFSR dengan panjang 10.



Gambar 5 LFSR pada Sosemanuk

Sebagai fungsi geser, LFSR pada Sosemanuk memiliki rumus:

$$s_{t+9} = s_{t+9} \oplus \alpha^{-1} s_{t+3} \oplus \alpha s_t$$

4.1.2 FSM

FSM adalah komponen 64 bit terdiri dari dua register berukuran masing-masing 32 bit R1 dan R2.

$R1_t = (R2_{t-1} + \text{mux}(\text{lsb}(R1_{t-1}), s_{t+1}, s_{t+1} \oplus s_{t+8})) \bmod 2^{32}$
$R2_t = \text{Trans}(R1_{t-1})$
$f_t = (s_{t+9} + R1_t \bmod 2^{32}) \oplus R2_t$
di mana lsb(x) adalah bit *least significant* dari x, mux(c, x, y) adalah x apabila $c = 0$ atau y apabila c = 1, Trans(z) = (M x z mod 2^{23}) $_{<<<7}$, M adalah konstanta 0x54655307 (heksadesimal dari sepuluh digit awal π dan <<< menyatakan putaran bit pada nilai 32 bit.

4.1.3 Alur Algoritma Sesomanuk

Sama seperti Serpent yang membutuhkan 32 putaran awal tanpa menghasilkan keluaran apapun, Sesomanuk hanya membutuhkan 4 putaran awal.

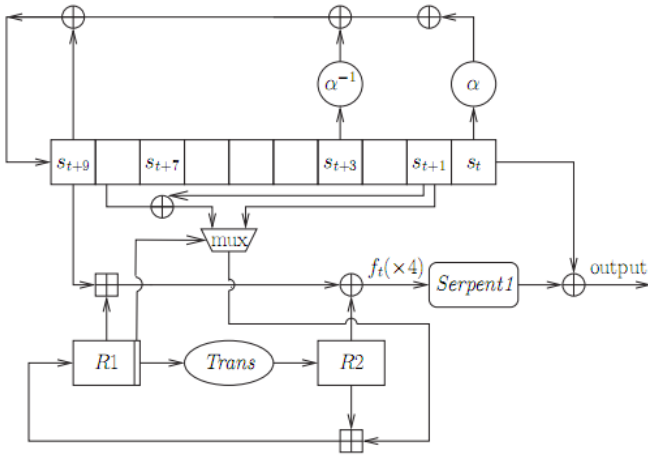
Tahapan tersebut adalah

- LFSR memiliki nilai awal s_1 sampai s_9, tidak ada nilai s_0. FSM awal memiliki nilai $R1_0$ dan $R2_0$.
- Pada langkah pertama, $R1$, $R2$ dan f_1 dihitung dari $R1_0$, $R2_0$, s_2, s_9 dan s_{10} (s_{10} di sini menyatakan nilai yang akan dimasukkan ke dalam register geser).
- Langkah pertama menghasilkan nilai yang disimpan s_1 dan f_1.
- Dalam langkah pertama tersebut, umpan balik s_{11} dihasilkan dari s_{10}, s_4 dan s_1, dan nilai-nilai LFSR di-*update*, menghasilkan nilai s_2 sampai s_{11}.
- Empat nilai keluaran pertama, z_1, z_2, z_3 dan z_4 dihasilkan dari operasi Serpent1 dengan (f_4, f_3, f_2, f_1) yang keluarannya dikombinasikan dengan XOR (s_4, s_3, s_2, s_1).

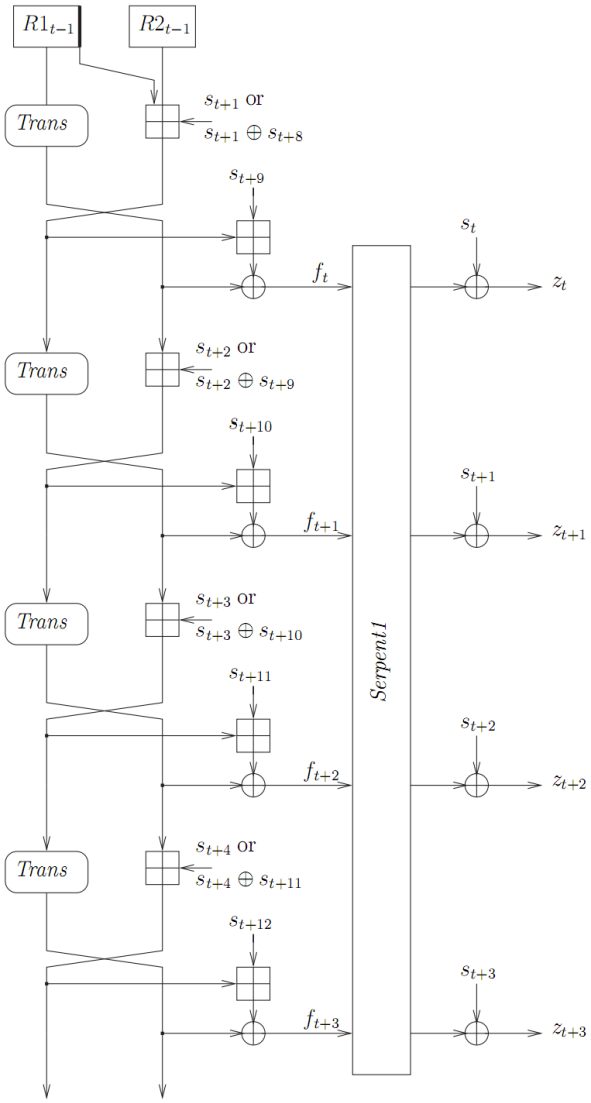
Adapun inisialisasi pada Sosemanuk terdiri dari dua langkah:

1. penjadwalan kunci, pemrosesan hanya dengan menggunakan kunci rahasia.
   Pemrosesan ini berkorespondensi dengan penjadwalan kunci pada Serpent24.
2. injeksi *IV*, pemrosesan dengan menggunakan keluaran dari penjadwalan kunci dan *IV*; digunakan untuk menginisialisasi state internal dari Sesomanuk.
   IV bernilai 128 bit dan digunakan sebagai masukan pada cipher blok Serpent24. Serpent24 memiliki putaran sebanyak 24 dan keluaran dari putaran ke-12, ke-18 dan ke-24 yang dipakai.
   - (Y_3^{12}, Y_2^{12}, Y_1^{12}, Y_0^{12}): keluaran dari putaran ke-12
   - (Y_3^{18}, Y_2^{18}, Y_1^{18}, Y_0^{18}): keluaran dari putaran ke-18
   - (Y_3^{24}, Y_2^{24}, Y_1^{24}, Y_0^{24}): keluaran dari putaran ke-24

   Nilai-nilai tersebut kemudian dipakai untuk mengisikan state internal Sesomanuk:
   $(s_7, s_8, s_9, s_{10}) = (Y_3^{12}, Y_2^{12}, Y_1^{12}, Y_0^{12})$
   $(s_5, s_6) = (Y_1^{18}, Y_3^{18})$
   $(s_1, s_2, s_3, s_4) = (Y_3^{24}, Y_2^{24}, Y_1^{24}, Y_0^{24})$
   $R1_0 = Y_0^{18}$
   $R2_0 = Y_2^{18}$



Gambar 5 Skema Algoritma Sosemanuk



Gambar 6 Putaran pada Sosemanuk yang menghasilkan 4 keluaran pertama

5. Perbandingan

Jelas bahwa pada Sosemanuk, adanya fungsi mux dan lsb serta Trans mengurangi celah potensial pada Snow 2.0.

Selain itu, Sosemanuk memperbaiki algoritma Snow dalam dua bidang. Pertama, Sosemanuk membuang susunan struktur yang memungkinkan untuk menjadi kelemahan, walaupun Snow dengan 128 bit mampu menahan semua serangan yang diketahui. Kedua, efisiensi Sosemanuk ditingkatkan dengan mengurangi ukuran internal state.

Sosemanuk juga memiliki kelebihan bahwa Sosemanuk hanya memerlukan jumlah data statik yang lebih sedikit; sehingga dalam beberapa arsitektur, Sosemanuk memiliki kinerja yang lebih baik daripada Snow 2.0. Kelebihan lainnya adalah bahwa prosedur inisialisasi kunci pada Sosemanuk adalah berdasarkan pada algoritma terkenal, cipher blok Serpent dalam versi yang dikurangi, sehingga menjadikan inisialisasi pada Sosemanuk menjadi lebih efisien dan aman.

6. Serangan dan Kripanalisis Sesomanuk

Beberapa serangan dan kriptanalisis pada Sesomanuk adalah:

- T. Tsunoo et al. (2006)
  Menggunakan kriptanalisis tipe Guess-And-Determine. Kompleksitas yang dihasilkan 2^{224}
- TMD-Tradeoff
  Pre-komputasinya adalah 2^{192}, waktu yang dibutuhkan 2^{192} juga dan memori dan data yang dibutuhkan masing-masing 2^{200} bit.
- Correlation Attak
  Fungsinya adalah menemukan state internal dari Sesomanuk. Kompleksitasnya adalah < 2^{150} dengan waktu yang dibutuhkan $2^{147.12}$, memori yang dibutuhkan 2^{147} bit dan data 2^{145} bit.

7. Kesimpulan

Kesimpulan yang dapat diambil dari makalah ini adalah:

1. Walaupun algoritma kunci simetri sudah tergolong lama, namun berbagai algoritma terus bermunculan dan dengan harapan mencapai *one-time-pad*, algoritma-algoritma tersebut mencoba untuk membuat komputasi yang lebih rumit.
2. Sesomanuk, sebagai finalis untuk lomba yang bersangkutan, merupakan algoritma cipher aliran yang tangguh, memiliki komputasi dan efisiensi yang lebih besar dari predesesornya, Snow 2.0.
3. Kesalahan 1 bit pada algoritma cipher aliran Sesomanuk hanya berpengaruh kepada bit yang bersangkutan.
4. Keamanan Sesomanuk adalah sebesar 128 bit, tidak tergantung dari panjang kunci.

DAFTAR PUSTAKA

[1] Munir, Rinaldi. (2004). Bahan Kuliah IF5054 Kriptografi. Departemen Teknik Informatika, Institut Teknologi Bandung.

[2] e-Crypt, h*ttp://www.ecrypt.eu.org/stream/sosemanuk.html*. Tanggal akses: 2 April 2009 pukul 13:00.

[3] Jung-Keun Lee, Dong Hoon Lee, and Sangwoo Park. (2008). Cryptanalysis of Sosemanuk and SNOW 2.0 Using Linear Masks. *http://www.deakin.edu.au/scitech/eit/asiacrypt2008/Cryptanalysis%20of%20Sosemanuk%20and%20SNOW%202.0%20Using%20Linear%20Masks.ppt*. Tanggal akses: 2 April 2009 pukul 13:00.

[4] Serpent, *http://www.cl.cam.ac.uk/~rja14/serpent.html*. Tanggal akses: 2 April 2009 pukul 13:00.

[5] Serpent: A Proposal for the Advanced Encryption Standard. *http://www.certainkey.com/resources/article/serpent.pdf*. Tanggal akses: 2 April 2009 pukul 13:00.

[6] A First Report on the Stream Cipher SNOW, *https://www.cosic.esat.kuleuven.be/nessie/reports/phase1/sagwp3-026_2.pdf*. Tanggal akses: 2 April 2009 pukul 13:00.

[7] Snow. h*ttp://www.it.lth.se/cryptology/snow*. Tanggal akses: 2 April 2009 pukul 13:00.