STUDI PENGGUNAAN KASUMI (A5/3) PADA MOBILE APPLICATIONS BERBASIS GSM

M. HAEKAL IZMANDA PULUNGAN
13507020

Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Jl. Ganesha No. 10, 40132
e-mail: if17020@students.if.itb.ac.id
haekal.pulungan@gmail.com

ABSTRAK

Makalah ini akan memaparkan studi mengenai penggunaan KASUMI (A5/3) sebagai *block cipher* untuk keamanan *mobile telecommunications* yang berbasis GSM (*Groupe Spécial Mobile* atau *Global System for Mobile Communications*).
Penggunaan *mobile telecommunications* adalah sesuatu yang sangat umum saat ini. Siapapun orangnya, apapun dan seberapapun kepentingannya pasti pernah memanfaatkannya. Dapat dibayangkan arus informasi yang ada. Tentunya dibutuhkan sistem pengamanan yang baik.
***Mobile communications* sempat menggunakan A5/1 sebagai sistem pengamanannya, terutama di negara-negara Eropa dan Amerika Serikat. Selanjutnya, A5/2 diperkenalkan dan digunakan di negara lainnya. Seiring dengan itu, mulailah ditemukan celah-celah pada A5/1.**
Ternyata A5/2 bahkan tidak lebih baik daripada A5/1. Akhirnya, diusulkanlah sebuah algoritma yang disebut KASUMI yang hingga saat ini digunakan.
Makalah ini terutama akan membahas mengenai KASUMI, A5/1 dan A5/2 sebagai pendahulunya, serta perbandingannya sehingga dapat dilihat di mana kelebihan yang KASUMI miliki.

Kata kunci: KASUMI, *block chiper*, *mobile telecommunications*, GSM, A5/1, A5/2.

1. PENDAHULUAN

Mobile telecommunications adalah hal yang sudah sangat umum digunakan oleh manusia saat ini. Sesuai dengan namanya, hal ini memungkinkan orang untuk menggunakan tanpa harus berdiam di satu tempat atau dapat digunakan secara *mobile*. GSM (*Groupe Spécial Mobile* atau *Global System for Mobile Communications*) adalah sistem standar *mobile telecommunications* yang paling banyak digunakan saat ini.

Dengan semakin besarnya angka penggunaan *mobile telecommunications* (khususnya GSM) dan semakin banyaknya aspek-aspek kehidupan yang "dimasuknya", maka keberadaannya menjadi semakin vital pula.

Inti dari kominukasi adalah transfer informasi. Ada informasi yang bersifat publik, namun banyak juga informasi yang sifatnya rahasia. Untuk menjaga kerahasiaan inilah dibutuhkan sistem pengamanan.

Dewasa ini, pengamanan yang dilakukan adalah dengan mengenkripsi data yang dikirimkan (terutama percakapan). Ada beberapa algoritma pengenkripsian yang telah digunakan, yang dalam perkembangannya mengalami pergantian ketika suatu algoritma telah berhasil dipatahkan.

Untuk sistem GSM, yang pertama digunakan adalah A5/1. Setelah itu, diperkenalkan A5/2 yang memiliki kemiripan dengan A5/1, akan tetapi ternyata lebih "lemah" dibandingkan dengan A5/1.

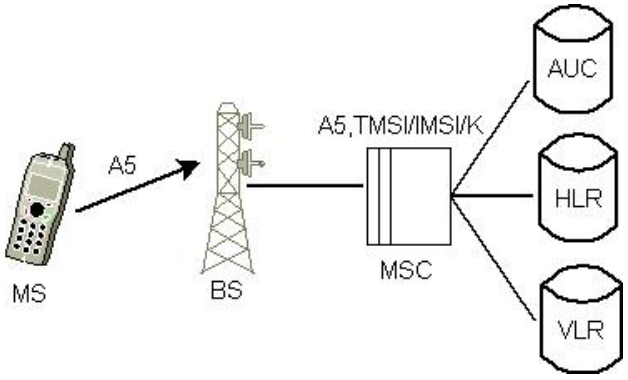
Setelah akhirnya banyak dipublikasikan serangan-serangan yang diberhasil dilakukan terhadap A5/1, dikembangkanlah algoritma baru yang akhirnya disebut KASUMI. KASUMI ini sendiri merupakan pengembangan dari MISTY1. KASUMI adalah algortima *block cipher*, sedangkan A5/1 dan A5/2 merupakan algoritma *stream cipher*.

2. GSM DAN PENGAMANANNYA

Berikut akan dipaparkan mengenai GSM dan pengamanannya, yaitu algoritma A5/1, A5/2, dan KASUMI (A5/3). MISTY1 sebagai "pendahulu" KASUMI tidak akan dibahas mendalam.

2.1 GSM

GSM adalah jaringan selular, yang berarti bahwa perangkat *mobile telecommunications* terkoneksi dengan jaringan tersebut dengan pencarian *cells* (sel) secara langsung di area sekitarnya.



Gambar 1. Stuktur jaringan pada GSM

Berikut adalah keterangan dari masing-masing bagian pada **Gambar 1**.

- **MS** atau ***mobile station***, adalah perangkat *mobile* yang pengguna gunakan untuk masuk ke dalam jaringan.
- **BS** atau ***base station***, merupakan seksi yang bertanggung jawab dalam menangani lalu lintas (*traffic*) dan pensinyalan (*signalling*) antara *mobile station* dengan *network switching subsystem* (NSS).
- **MSC** atau ***mobile switching centre***, merupakan bagian dari NSS yang bertanggung jawab terhadap routing dari setiap layanan GSM (*voice call*, SMS, dll).
- **AUC** atau ***authentication centre***, merupakan bagian dari NSS yang berfungsi untuk mengotentifikasi setiap kartu SIM (*subscriber identity modul*) yang masuk ke dalam jaringan.
- **HLR** atau ***home location register***, merupakan bagian dari NSS yang berperan sebagai basis data pusat (*central database*) yang memuat keterangan-keterangan (details) dari setiap mobile phone yang telah diberi kewenangan untuk masuk ke dalam jaringan.
- **VLR** atau ***visitor local register***, merupakan bagian dari NSS yang sebagai basis data sementara (*temporary database*) untuk pelanggan pada suatu daerah tertentu, di mana pelanggan tersebut melakukan *roaming*.

A5 pada **Gambar 1** menunjukkan penggunaan algoritma pengamanan, dalam hal ini A5/1, A5/2, atau A5/3.

2.2 A5/1

Berikut adalah gambaran umum dari algortima ini[2].

Masukan Kunci *K* (rahasia) berukuran 64-bit, IV *Fn* berukuran 22-bit

Keluaran Dua blok *keystream* berukuran 114-bit yang digunakan untuk mengenkripsi data

Enkripsi *Bitwise* XOR

Secara singkat, berikut adalah cara pengenkripsian data oleh A5/1[3].

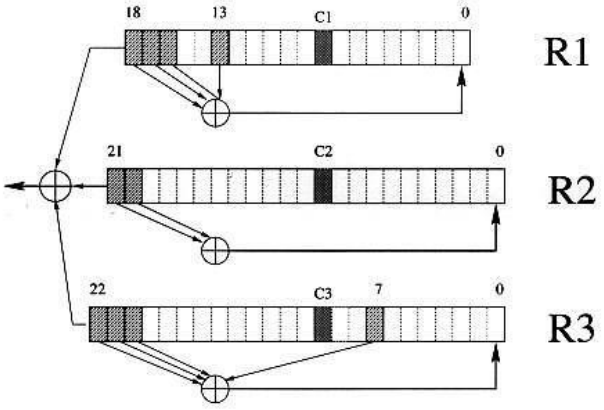
Sebuah percakapan (*conversation*), sebagai contoh antara A dan B, dikirim melalui *frame-frame* secara sekuensial setiap 4,6 milidetik. Setiap *frame* memuat 114 bit yang merepresentasikan komunikasi A ke B secara digital dan 114 bit yang merepresentasikan komunikasi B ke A secara digital. Setiap percakapan dienkripsi oleh *K*, yang berbeda untuk setiap sesi. Untuk setiap *frame*, *K* bercampur dengan *Fn*, dan menghasilkan 228 bit yang terlihat acak. Bit-bit ini kemudian di-XOR dengan dua bagian yang merupakan 114+114 bit *plaintext*, yang kemudian menghasilkan 114+114 bit *ciphertext*.

A5/1 dibangun dari tiga buah LFSR (*linear feedback shift register*) dengan panjang masing-masing 19 bit, 22 bit, dan 23 bit (selanjutnya, secara berurutan, akan disebut *R1*, *R2*, dan *R3*). Bit yang terletak pada paling kanan pada setiap *register* dilabeli dengan indeks 0 (nol). Pada masing-masing *register* ditandai *tap-tap* untuk bit-bit tertentu, dengan ketentuan sebagai berikut.

R1 = 13, 16, 17, 18
R2 = 20, 21
R3 = 7, 20, 21, 22

Ketika sebuah *register* mendapat sinyal *clock* (*clocked*), tap-tap dari register tersebut di-XOR (sekaligus), dan hasilnya dimuat pada bit-paling-kanan-nya.

Untuk lebih jelasnya, perhatikan **Gambar 2**.



Gambar 2. A5/1 *stream cipher*

Berikut adalah proses diperolehnya bit acak dari *K* dan *Fn*.

- Ketiga *register* dikosongkan, dan di-*clocked* untuk 64 putaran. Selama tahap ini, setiap bit dari *K* (dari *lsb* ke *msb*) di-XOR secara paralel dengan *lsb* dari ketiga *register*.
- Ketiga *register* di-*clocked* kembali untuk 22 putaran tambahan. Selama tahap ini, setiap bit dari *Fn* (dari *lsb* ke *msb*) di-XOR secara paralel dengan *lsb* dari ketiga *register*. Setelah tahap ini, hasil yang diperoleh disebut *initial state* dari *frame*.
- Ketiga *register* di-*clocked* untuk 100 putaran dengan *stop/go control* tanpa menghasilkan sebuah keluaran.
- Ketiga *register* di-*clocked* kembali untuk 228 putaran dengan *stop/go control* untuk menghasilkan sebuah keluaran berukuran 228 bit. Pada setiap putaran, sebuah keluaran dihasilkan sebagai XOR dari *msb* ketiga *register*.

2.2 A5/2

A5/2 dibangun dari kerangka kerja yang mirip dengan A5/1. Jika A5/1 menggunakan tiga *register*, maka A5/2 menggunakan empat *register*.

Berikut adalah penjelasan mengenai struktur inernal dan cara kerjanya[4].

A5/2 menggunakan 4 LFSR (selanjutnya disebut *R1*, *R2*, *R3*, dan *R4*) dengan panjang masing-masing (secara berurutan) 19 bit, 22 bit, 23 bit, dan 17 bit. Setiap *register* memiliki *tap-tap* dan fungsi *feedback*. Polinomial yang *irreducible* masing-masing (secara berurutan) adalah , , , dan .

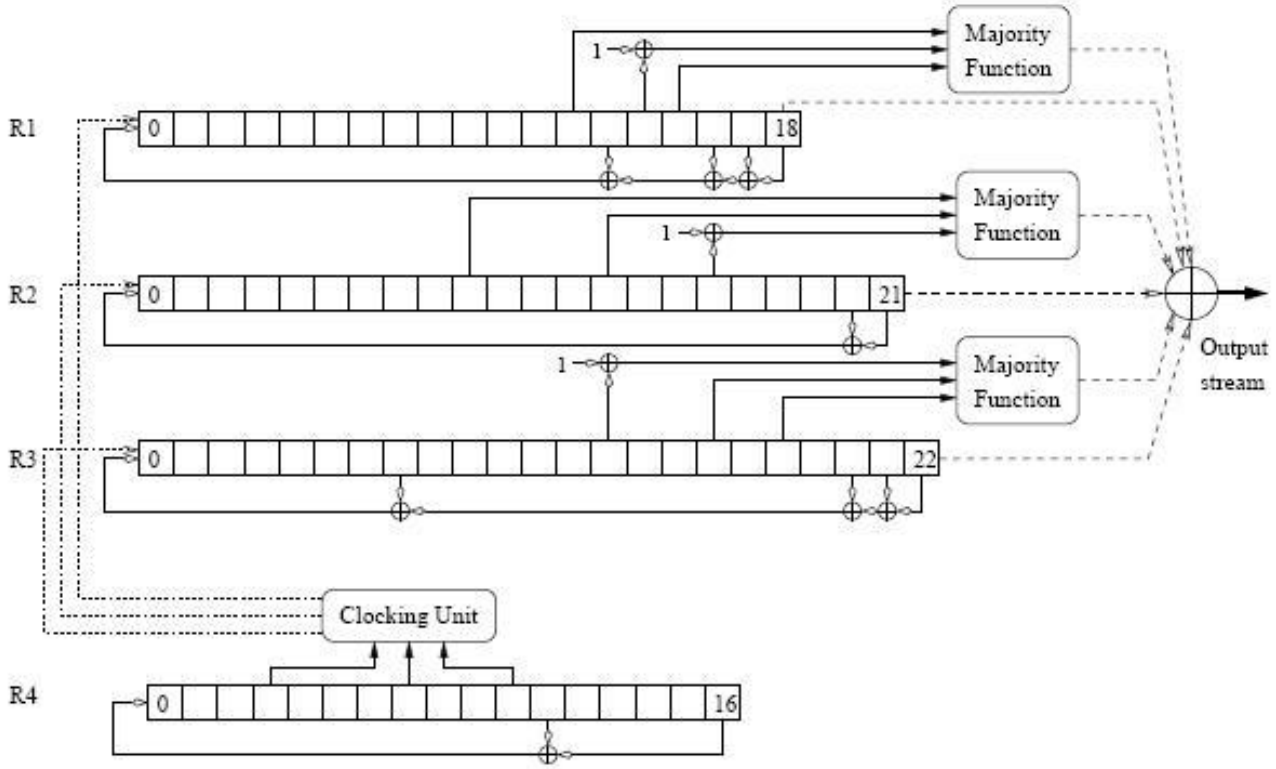
Berikut adalah cara memperoleh *initial state* dari *frame*.

- Seluruh LFSR di-set ke 0 (*R1* = *R2* = *R3* = *R4* = 0)
- for *i* := 0 to 63 do
  - *clock* seluruh LFSR.
  -
  -
  -
  -
- for *i* := 0 to 21 do
  - *clock* seluruh LFSR.
  -
  -
  -
  -

Setelah proses inisialisasi di atas, *key-stream* diperoleh dengan:

- nilai bit *R1*[15], *R2*[16], *R3*[18], dan *R4*[10] dijadikan 1.
- jalankan A5/2 sebanyak 99 putaran dan abaikan keluarannya
- jalankan A5/2 sebanyak 228 putaran dan gunakan keluarannya sebagai *key-stream*.

Berikut adalah struktur internal dari A5/2.



Gambar 3. Struktur internal A5/2

2.3 A5/3

KASUMI beroperasi pada 64-bit masukan **I** menggunakan kunci **K** 128-bit untuk menghasilkan keluaran 64-bit. Penjelasan selengkapnya akan dijelaskan di bawah ini.

Pertama, masukan I dibagi menjadi dua bagian L_0 dan R_0 masing-masing 32-bit, dimana

$$I = L_0 \| R_0$$

Kemudian, untuk setiap integer i dengan $1 \leq i \leq 8$, didefinisikan:

$$R_i = L_{i-1},\ L_i = R_{i-1} \oplus f_i(L_{i-1}, RK_i)$$

Operasi ini akan menghasilkan keluaran $(L_8 \| R_8)$ pada akhir iterasi kedelapan.

KASUMI terdiri dari beberapa komponen.

Fungsi ***fi( )*** mengambil masukan 32-bit ***I*** menghasilkan keluaran 32-bit ***O*** dikendalikan oleh round key ***RKi***, dimana *round key* tersebut terdiri dari *subkey triplet* dari (***KLi***, ***KOi***, ***KIi***). Fungsi ***fi( )*** itu sendiri terbentuk dari dua subfungsi; ***FL*** and ***FO*** yang berasosiasi dengan subkunci ***KLi*** (untuk ***FL***) dan subkey ***KOi*** dan ***KIi*** (untuk ***FO***).

Fungsi ***fi( )*** memiliki dua bentuk yang berbeda tergantung kepada iterasi yang sedang berjalan, apakah iterasi ganjil atau genap. Untuk iterasi 1,3,5 and 7 didefinisikan:

$$fi(I,RKi) = FO(\ FL(\ I,\ KLi),\ KOi,\ KIi\ )$$

sedangkan untuk iterasi 2,4,6 and 8 didefinisikan:

$$fi(I,Ki) = FL(\ FO(\ I,\ KOi,\ KIi\ ),\ KLi\ )$$

Ilustrasi dari fungsi f_i ditunjukkan pada Gambar 3.

Selanjutnya adalah fungsi *FL*.

Masukan dari fungsi *FL* terdiri dari 32-bit data masukan ***I*** dan 32bit subkey ***KLi***. Subkey ***Kli*** dibagi menjadi dua bagian, ***KLi,1*** and ***KLi,2*** dimana:

$$KLi = KLi,1 \| KLi,2$$

Data masukan ***I*** juga dibagi dua, masing-masing 16-bit, ***L*** and ***R*** dimana:

$$I = L \| R$$

Didefinisikan:

$$R' = R \oplus ROL(L \cap KL_{i,1})$$

$$L' = L \oplus ROL(R' \cap KL_{i,2})$$

Hasil keluaran dari fungsi ini yaitu sebesar 32-bit yang nilainya adalah:

$I = L' \| R'$

Selanjutnya adalah fungsi *FO*.

Masukan dari fungsi ini terdiri dari 32-bit data masukan I dan dua set dari subkunci, 48-bit subkunci KO_i and 48-bit subkunci KI_i.

Data masukan sebesar 32-bit tersebut dibagi menjadi dua bagian , L_0 and R_0, dengan:

$I = L_0 \| R_0$

Subkunci sebesar 48-bit tersebut dibagi menjadi tiga subkunci sebesar 16-bit, dimana:

$KO_i = KO_{i,1} \| KO_{i,2} \| KO_{i,3}$
$KI_i = KI_{i,1} \| KI_{i,2} \| KI_{i,3}$

Kemudian, untuk setiap integer j dengan $1 \leq j \leq 3$, didefinisikan:

$$R_j = FI(L_{j-1} \oplus KO_{i,j}, KI_{i,j}) \oplus R_{j-1}$$
$$L_j = R_{j-1}$$

Fungsi tersebut akan menghasilkan keluaran ($L3 \| R3$) sebesar 32-bit.

Fungsi *FI* mengambil masukan 16-bit data masukan I dan 16-bit subkey $KI_{i,j}$. Masukan I dibagi menjadi dua bagian yang tidak sama panjang, 9-bit untuk bagian kiri L_0 and a 7-bit bagian kanan R_0 dimana:

$I = L_0 \| R_0$

Sama dengan data input, key $KI_{i,j}$ dibagi menjadi komponen 7-bit $KI_{i,j,1}$ dan komponen 9-bit $KI_{i,j,2}$ dimana:

$KI_{i,j} = KI_{i,j,1} \| KI_{i,j,2}$

Fungsi *FI* menggunakan dua S-Box, $S7$ yang memetakan masukan 7-bit menjadi keluaran 7-bit, dan $S9$ yang memetakan masukan 9-bit menjadi keluaran 9-bit. Hal ini kan dijelaskan lebih rinci pada bagian selanjutnya. Fungsi ini juag menggunakan dua fungsi tambahan $ZE(\,)$ dan $TR(\,)$. Kedua fungsi tambahan tersebut didefinisikan sebagai berikut:

$ZE(x)$:
Mengambil masukan 7-bit dari x mengubahnya menjadi 9-bit dengan menambahkan dua 0 bit ke bagian MSB.

$TR(x)$: Mengambil masukan 9-bit dari x mengubahnya menjadi 7-bit dengan menghilangkan dua 0 bit dari bagian MSB.

Selanjutnya didefinisikan operasi sebagai berikut ini:

$$L_1 = R_0$$
$$R_1 = S9[L_0] \oplus ZE(R_0)$$
$$L_2 = R_1 \oplus KI_{i,j,2}$$
$$R_2 = S7[L_1] \oplus TR(R_1) \oplus KI_{i,j,1}$$
$$L_3 = R_2$$
$$R_3 = S9[L_2] \oplus ZE(R_2)$$
$$L_4 = S7[L3] \oplus TR(R_3)$$
$$R_4 = R_3$$

Hasil dari fungsi ini yaitu ($L_4 \| R_4$)sebesar 16-bit.

KASUMI memiliki dua S-box yang telah dirancang untuk dapat diimplementasikan dengan mudah baik dalam logika kombinasi maupun dalam *look-up table*.

Masukan x terdiri dari tujuh atau sembilan bit dengan nomor bt yang sama pada keluaran y.

$x = x8 \| x7 \| x6 \| x5 \| x4 \| x3 \| x2 \| x1 \| x0$
$y = y8 \| y7 \| y6 \| y5 \| y4 \| y3 \| y2 \| y1 \| y0$

dimana bit $x8, y8$ dan $x7, y7$ hanya digunakan pada **S9**, dan bit $x0$ dan $y0$ adalah LSB.

***S7*- S-Box**
Gerbang Logika:

```
y0 = x1x3 ⊕ x4 ⊕ x0x1x4 ⊕ x5

⊕ x2x5 ⊕ x3x4x5 ⊕ x6 ⊕ x0x6

⊕ x1x6 ⊕ x3x6 ⊕ x2x4x6 ⊕ x1x5x6
```

⊕ x4x5x6

y1 = x0x1 ⊕ x0x4 ⊕ x2x4 ⊕ x5

⊕ x1x2x5 ⊕ x0x3x5 ⊕ x6 ⊕ x0x2x6

⊕ x3x6 ⊕ x4x5x6 ⊕ 1

y2 = x0 ⊕ x0x3 ⊕ x2x3⊕ x1x2x4

⊕ x0x3x4 ⊕ x1x5 ⊕ x0x2x5 ⊕ x0x6

⊕ x0x1x6 ⊕ x2x6 ⊕ x4x6⊕ 1

y3 = x1 ⊕ x0x1x2 ⊕ x1x4 ⊕ x3x4

⊕ x0x5 ⊕ x0x1x5 ⊕ x2x3x5

⊕ x1x4x5 ⊕ x2x6 ⊕ x1x3x6

y4 = x0x2 ⊕ x3 ⊕ x1x3 ⊕ x1x4

⊕ x0x1x4 ⊕ x2x3x4 ⊕ x0x5 ⊕ x1x3x5 ⊕

x0x4x5 ⊕ x1x6 ⊕ x3x6

⊕ x0x3x6 ⊕ x5x6 ⊕ 1

y5 = x2 ⊕ x0x2 ⊕ x0x3 ⊕ x1x2x3

⊕ x0x2x4 ⊕ x0x5 ⊕ x2x5 ⊕ x4x5

⊕ x1x6 ⊕ x1x2x6 ⊕ x0x3x6

⊕ x3x4x6 ⊕ x2x5x6 ⊕ 1

y6 = x1x2 ⊕ x0x1x3⊕x0x4 ⊕ x1x5

⊕ x3x5 ⊕ x6 ⊕ x0x1x6 ⊕ x2x3x6

⊕ x1x4x6 ⊕ x0x5x6

Tabel Desimal
54, 50, 62, 56, 22, 34, 94, 96,
38, 6, 63, 93, 2, 18, 123, 33,

55, 113, 39,114, 21, 67, 65, 12,
47, 73, 46, 27, 25, 111, 124,
81, 53, 9, 121, 79, 52, 60, 58,
48, 101, 127, 40, 120, 104, 70,
71, 43, 20, 122, 72, 61, 23,
109, 13, 100, 77, 1, 16, 7, 82,
10, 105, 98, 117, 116, 76, 11,
89, 106, 0, 125, 118, 99, 86,
69, 30, 57, 126, 87, 112, 51,
17, 5, 95, 14, 90, 84, 91, 8,
35, 103, 32, 97, 28, 66, 102,
31, 26, 45, 75, 4, 85, 92, 37,
74, 80, 49, 68, 29, 115, 44, 64,
107, 108, 24, 110, 83, 36, 78,
42, 19, 15, 41, 88, 119, 59, 3

Contoh:
Jika kita memiliki nilai masukan sebesar 38, maka dengan menggunakan tabel desimal S7[38], maka keluaran yang akan dihasilkan adalah 58.
Sedangkan untuk gerbang logika, kita jabarkan:
$38 = 0100110_2$, dimana x6 = 0, x5 = 1, x4 = 0, x3 = 0, x2 = 1, x1 = 1, x0 = 0, yang jika dimasukkan ke dalam rumus gerbang logika S7 akan menjadi sebagai berikut:

y0 = 0 ⊕ 0 ⊕ 0 ⊕ 1 ⊕ 1 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0

⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 = 0

y1 = 0 ⊕ 0 ⊕ 0 ⊕ 1 ⊕ 1 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0

⊕ 0 ⊕ 1 = 1

y2 = 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 1 ⊕ 0 ⊕ 0 ⊕ 0

⊕ 0 ⊕ 0 ⊕ 1 = 0

y3 = 1 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0

⊕ 0 = 1

y4 = 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0

⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 1 = 1

y5 = 1 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 1

⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 1 = 1

y6 = 1 ⊕ 0 ⊕ 0 ⊕ 1 ⊕ 0 ⊕ 0 ⊕ 0

⊕ 0 ⊕ 0 ⊕ 0 = 0

Maka, $y = 0111010_2 = 58$.

***S9*- S-Box**

y0 = x0x2 ⊕ x3 ⊕ x2x5 ⊕ x5x6 ⊕ x0x7 ⊕

x1x7 ⊕ x2x7 ⊕ x4x8 ⊕ x5x8 ⊕ x7x8 ⊕ 1

y1 = x1 ⊕ x0x1 ⊕ x2x3 ⊕ x0x4 ⊕ x1x4 ⊕

x0x5 ⊕ x3x5 ⊕ x6 ⊕ x1x7 ⊕ x2x7 ⊕ x5x8 ⊕

1

y2 = x1 ⊕ x0x3 ⊕ x3x4 ⊕ x0x5 ⊕ x2x6 ⊕

x3x6 ⊕ x5x6 ⊕ x4x7 ⊕ x5x7 ⊕ x6x7 ⊕ x8 ⊕

x0x8 ⊕ 1

y3 = x0 ⊕ x1x2 ⊕ x0x3 ⊕ x2x4 ⊕ x5 ⊕ x0x6

⊕ x1x6 ⊕ x4x7 ⊕ x0x8 ⊕ x1x8 ⊕ x7x8

y4 = x0x1 ⊕ x1x3 ⊕ x4 ⊕ x0x5 ⊕ x3x6 ⊕

x0x7 ⊕ x6x7 ⊕ x1x8 ⊕ x2x8 ⊕ x3x8

y5 = x2 ⊕ x1x4 ⊕ x4x5 ⊕ x0x6 ⊕ x1x6 ⊕

x3x7 ⊕ x4x7 ⊕ x6x7 ⊕ x5x8 ⊕ x6x8 ⊕ x7x8

⊕ 1

y6 = x0 ⊕ x2x3 ⊕ x1x5 ⊕ x2x5 ⊕ x4x5 ⊕

x3x6 ⊕ x4x6 ⊕ x5x6 ⊕ x7 ⊕ x1x8 ⊕ x3x8 ⊕

x5x8 ⊕ x7x8

y7 = x0x1 ⊕ x0x2 ⊕ x1x2 ⊕ x3 ⊕ x0x3 ⊕

x2x3 ⊕ x4x5 ⊕ x2x6 ⊕ x3x6 ⊕ x2x7 ⊕ x5x7

⊕ x8 ⊕ 1

y8 = x0x1 ⊕ x2 ⊕ x1x2 ⊕ x3x4 ⊕ x1x5 ⊕

x2x5 ⊕ x1x6 ⊕ x4x6 ⊕ x7 ⊕ x2x8 ⊕ x3x8

Tabel Desimal:

167, 239, 161, 379, 391, 334, 9,
338, 38, 226, 48, 358, 452, 385,
90, 397, 183, 253, 147, 331,
415, 340, 51, 362, 306, 500,
262, 82, 216, 159, 356, 177,
175, 241, 489, 37, 206, 17, 0,
333, 44, 254, 378, 58, 143, 220,
81, 400, 95, 3, 315, 245, 54,
235, 218, 405, 472, 264, 172,
494, 371, 290, 399, 76, 165,
197,395, 121, 257, 480, 423,
212, 240, 28, 462, 176, 406,
507, 288, 223, 501, 407, 249,
265, 89, 186, 221, 428, 164,
74, 440, 196, 458, 421, 350,
163, 232, 158, 134, 354, 13,
250, 491, 142, 191, 69, 193,
425, 152, 227, 366, 135, 344,
300, 276, 242, 437, 320, 113,
278, 11, 243, 87, 317, 36, 93,
496, 27, 487, 446, 482, 41, 68,
156, 457, 131, 326, 403, 339,
20, 39, 115, 442, 124, 475, 384,
508, 53, 112, 170, 479, 151,
126, 169, 73, 268, 279, 321,
168, 364, 363, 292, 46, 499,
393, 327, 324, 24, 456, 267,
157, 460, 488, 426, 309, 229,
439, 506, 208, 271, 349, 401,
434, 236, 16, 209, 359, 52, 56,
120, 199, 277, 465, 416, 252,
287, 246, 6, 83, 305, 420, 345,
153, 502, 65, 61, 244, 282, 173,
222, 418, 67, 386, 368, 261,
101, 476, 291, 195, 430, 49, 79,
166, 330, 280, 383, 373, 128,
382, 408, 155, 495, 367, 388,
274, 107, 459, 417, 62,454, 132,
225, 203, 316, 234, 14, 301, 91,
503, 286, 424, 211, 347, 307,
140,374, 35, 103, 125, 427, 19,
214, 453, 146, 498, 314, 444,
230,256, 329, 198, 285, 50, 116,
78, 410, 10, 205, 510, 171, 231,
45, 139, 467, 29, 86, 505, 32,
72, 26, 342, 150, 313, 490, 431,
238, 411, 325, 149, 473, 40,
119, 174, 355, 185, 233, 389,
71, 448, 273, 372, 55, 110, 178,
322, 12, 469, 392, 369, 190, 1,
109, 375, 137, 181, 88, 75, 308,
260, 484, 98, 272, 370, 275,
412, 111, 336, 318, 4, 504, 492,
259, 304, 77, 337, 435, 21, 357,
303, 332, 483, 18, 47, 85, 25,
497, 474, 289, 100, 269, 296,
478, 270, 106, 31, 104, 433, 84,
414, 486, 394, 96, 99, 154, 511,
148, 413, 361, 409, 255, 162,
215, 302, 201, 266, 351, 343,
144, 441, 365, 108, 298, 251,
34, 182, 509, 138, 210, 335,
133, 311, 352, 328, 141, 396,
346, 123, 319, 450, 281, 429,
228, 443, 481, 92, 404, 485,
422, 248, 297, 23, 213, 130,
466, 22, 217, 283, 70, 294, 360,
419, 127, 312, 377, 7, 468, 194,
2, 117, 295, 463, 258, 224, 447,
247, 187, 80,398, 284, 353, 105,
390, 299, 471, 470, 184, 57,
200, 348, 63, 204, 188, 33, 451,
97, 30, 310, 219, 94, 160, 129,
493, 64, 179, 263, 102, 189,
207, 114, 402, 438, 477, 387,
122, 192, 42, 381, 5, 145, 118,
180, 449, 293, 323, 136, 380,
43, 66, 60, 455, 341, 445, 202,
432, 8, 237, 15, 376, 436, 464,
59, 461

Contoh:
Jika kita memiliki nilai masukan sebesar 138, maka dengan menggunakan tabel desimal S9[138], maka keluaran yang akan dihasilkan adalah 339.

Sedangkan untuk gerbang logika, kita jabarkan:

138 = 0100010102, dimana x8 = 0, x7 = 1, x6= 0, x5 = 0, x4 = 0, x3 = 1, x2 = 0, x1 = 1, x0 =0, yang jika dimasukkan ke dalam rumus gerbang logika S9 akan menjadi sebagai berikut:

$$y0 = 0 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 1 \oplus 0 \oplus 0 \oplus 0$$

$$\oplus 0 \oplus 1 = 1$$

$$y1 = 1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1$$

$$\oplus 0 \oplus 0 \oplus 1 = 1$$

y2 = 1 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0

⊕ 0 ⊕ 0 ⊕ 0 ⊕ 1 = 0

y3 = 0 ⊕ 0 ⊕ 0 ⊕0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0

⊕ 0 ⊕ 0 = 0

y4 = 0 ⊕ 1 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0

⊕ 0 = 1

y5 = 0 ⊕0 ⊕0 ⊕0 ⊕0 ⊕1 ⊕0 ⊕0 ⊕0 ⊕0

⊕0 ⊕1 = 0

y6 = 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 1

⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 = 1

y7 = 0 ⊕ 0 ⊕ 0 ⊕ 1 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0

⊕ 0 ⊕ 0 ⊕ 0 ⊕ 1 = 0

y8 = 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 1

⊕ 0 ⊕ 0 = 1

Maka, y = 1010100112 = 339.

Penjadwalan Kunci
KASUMI memiliki kunci ***K*** sebesar 128-bit. Untuk setiap iterasi, KASUMI menggunakan kunci 128-bit yang diturunkan dari ***K***.
Kunci K pertama dibagi menjadi 16-bit sebanyak 8 buah K1, .., K8 dimana:

K = K1 || K2 || K3 ||…|| K8

Array dari subkey kedua, ***Kj'*** diturunkan dari ***Kj*** dengan cara:

Untuk setiap integer ***j*** dengan $1 \leq j \leq 8$, maka:

Kj' = Kj ⊕ Cj

3. ANALISIS DAN KESIMPULAN

Hal termudah yang menjadi dasar analisis adalah kategori dari masing-masing algoritma. A5/1 dan A5/2 merupakan *stream cipher*, sedangkan A5/3 merupakan *block cipher*.

Terhadap A5/1 dan A5/2 telah diuji coba dengan serangan-serangan semacam *known-plaintext attack*, *flip-bit attack*, dan lain-lain.

Pada 2000, Alex Biryukov, Adi Shamir, dan David Wagner menunjukkan bahwa A5/1 dapat di-kriptanalisis pada *real-time* dengan *time-memory tradeoff attack*.

Pada tahun yang sama, dengan *known-plaintext attack*, Eli Biham dan Orr Dunkelman mempublikasikan keberhasilannya melakukan serangan terhadap A5/1.

A5/2 bahkan disebut-sebut lebih lemah daripada A5/1.

R4 yang digunakan malah menambah celah untuk diserang.

Di lain pihak, A5/3 ternyata telah ditemukan kelemahannya. Pada Januari 2010, algoritma ini berhasil diserang dengan *sandwich attack* dan *basic related-key boomerang attack*.

Dapat disimpulkan bahwa algoritma yang kita sebut modern pun ternyata masih memiliki celah untuk diserang. Dengan semakin vitalnya peran *mobile telecommunication*, jawaban terhadap tantangan untuk pengamannya harus ditemukan secepatnya.

REFERENSI

[1] Donnie, “Studi Algoritma KASUMI (A5/3) Block Cipher”, 2006

[2] Dunkelman, Orr, et all, "A Practical-Time Attack on the A5/3 Cryptosystem Used in Third Generation GSM Telephony", 2010

[3] Wallen, Johan, "Design Principles of the KASUMI Block Cipher", Helsinki University of Technology

[4] Kasper, Emilia, "On Hardware-Assisted Cryptanalysis of A5/1", 2005