# ANALISIS ALGORITMA ORYX

Andoko Gunawan – NIM : 13503083

*Program Studi Teknik Informatika, Institut Teknologi Bandung*
*Jl. Ganesha 10, Bandung*
E-mail : if13083@students.if.itb.ac.id

## Abstrak

Algoritma ORYX adalah algoritma standard enkripsi data digital di kawasan Amerika Utara yang diperkenalkan oleh Telecommunications Industry Association (TIA) dan digunakan untuk mengenkripsi pesan digital yang dikirim melalui saluran data nirkabel pada protokol TDMA dan CDMA. Algoritma ini, dan algoritma standard enkripsi data di kawasan Eropa (A5/1) yang digunakan untuk GSM, pada dasarnya berbasis cipher aliran.

ORYX adalah sebuah cipher aliran yang berbasis *linear feedback shift registers* (LFSR) biner. Cipher ORYX menggunakan sebuah generator *keystream*. Output dari generator ini adalah sebuah urutan bytes yang nampak acak. Enkripsi dilakukan dengan melakukan operasi XOR pada *keystream* tersebut dengan data plainteks untuk menghasilkan ciphertext. Sedangkan dekripsi dilakukan dengan melakukan operasi XOR pada *keystream* dengan ciphertext.

Algoritma ORYX ini tidak terlalu kuat, banyak serangan yang dapat memecahkan algoritma ini. Namun popularitasnya menyebabkan algoritma ini memiliki peranan yang penting.

**Kata Kunci**: Kriptografi, Kriptanalisis, *ORYX*, Algoritma Kunci Simetrik, *Stream Cipher*.

### 1. Pendahuluan

Komunikasi bergerak menggunakan telepon seluler kini sangat popular. Kepopulerannya sudah dapat dibilang melebihi kepopuleran sistem komunikasi lainnya.

Permintaan terhadap system komunikasi bergerak telah meningkat drastic dalam dasawarsa terakhir. Karena komunikasi seluler menggunakan model radio broadcast, maka sangat mudah untuk menyadap sistem tersebut tanpa terdeteksi oleh system.
Untuk melindungi privasi dan mencegah terjadinya kecurangan, algoritma kriptografi dirancang untuk menyediakan lingkungan komunikasi bergerak yang aman.

Generasi pertama komunikasi bergerak menggunakan metode analog. Kebanyakan telepon selular analog tidak menggunakan metode enkripsi apapun untuk melindungi kerahasiaan data. Apabila ada peralatan analog yang menggunakan metode enkripsi, metode tersebut hanya menawarkan keamanan tingkat rendah

Sejak akhir decade-90 an, sistem komunikasi bergerak digital mulai bermunculan, seperti standard Global Systems Mobile (GSM) yang dikembangkan di Eropa dan beberapa standar Telecommunications Industry Association (TIA) yang dikembangkan di Amerika Utara. Untuk sistem – sistem digital ini, dukungan terhadap metode enkripsi modern sudah ada sehingga secara teoritis dapat dipasang sebuah sistem keamanan yang lebih canggih, dan sebuah algoritma enkripsi yang lebih baik.

Di Indonesia sendiri sudah sejak lama diadopsi standard GSM yang dikembangkan di Eropa. Jaringan GSM di Indonesia sudah sangat luas dan kuat. Banyak pemain – pemain seperti Satelindo, Indosat, Telkomsel, dan Excelkomindo yang menggunakan standard GSM ini. Namun baru – baru ini muncul pesaing baru yang dipelopori oleh telkom Fleksi yang menggunakan standard CDMA. Pesaing baru ini menawarkan banyak keunggulan, seperti tawaran internet murah dan cepat, serta banyak nilai tambah lainnya diluar layanan utama pesan suara dan SMS.

Untuk itu dalam paper ini akan dibahas mengenai bagaimanakah keamanan komunikasi data melalui saluran CDMA, yakni dengan pembahasan pada algoritma ORYX.

## 2. Tipe dan Mode Algoritma Simetri

Algoritma kriptografi (*cipher*) simetri dapat dikelompokkan menjadi dua kategori, 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.

### 2.1 Cipher Blok

Pada *cipher* blok, rangkaian bit-bit plainteks dibagi menjadi blok-blok bit dengan panjang sama [3]. Enkripsi dilakukan terhadap blok bit plainteks menggunakan bit-bit kunci (yang ukurannya sama dengan blok plainteks). Algoritma enkripsi menghasilkan blok cipherteks yang berukuran sama dengan blok plainteks. Dekripsi dilakukan dengan cara yang serupa seperti enkripsi.
Misalkan blok plainteks ($P$) yang berukuran $m$ bit dinyatakan sebagai vektor

$$P = (p_1, p_2, ..., p_m)$$

yang dalam hal ini $p_i$ adalah bit 0 atau bit 1 untuk $i = 1, 2, \ldots,$ m, dan blok cipherteks ($C$) adalah

$$C = (c_1, c_2, ..., c_m)$$

yang dalam hal ini $c_i$ adalah bit 0 atau bit 1 untuk $i = 1, 2, \ldots,$ m.

Bila plainteks dibagi menjadi $n$ buah blok, barisan blok-blok plainteks dinyatakan sebagai

$$(P_1, P_2, \ldots, P_n)$$

Untuk setiap blok plainteks $P_i$, bit-bit penyusunnya dapat dinyatakan sebagai vektor

$$P_i = (p_{i1}, p_{i2}, ..., p_{im})$$

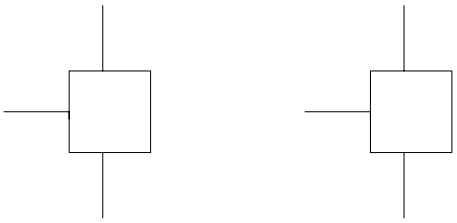
Enkripsi dengan kunci $K$ dinyatakan dengan persamaan

$$E_k(P) = C,$$

sedangkan dekripsi dengan kunci $K$ dinyatakan dengan persamaan

$$D_k(C) = P$$

Skema enkripsi dan dekripsi dengan *cipher* blok dapat dilihat pada Gambar 1.



### 2.2 Cipher Aliran

Cipher aliran adalah algoritma kriptografi yang berdasarkan algoritma kriptografi kunci simetris. Pada cipher aliran ini, digit – digit pada plainteks dienkripsi satu setiap saat, dan perubahan digit – digit itu bervariasi.

Meskipun sama – sama berbasiskan algoritma kunci simetris, cipher aliran berbeda dengan cipher blok yang beroperasi pada blok – blok digit berukuran besar yang transformasinya tetap. Namun perbedaan ini seringkali menjadi kabur, karena pada beberapa mode operasi blok cipher, sifat algoritmanya mirip dengan cipher aliran, seperti contohnya mode Cipher Feedback dari algoritma cipher blok.

Cipher aliran biasanya dapat dieksekusi lebih cepat dan hanya menggunakan perangkat keras yang sederhana dibandingkan dengan cipher blok. Namun, cipher aliran dapat terdedah pada permasalahan keamanan yang serius apabila digunakan secara tidak tepat.

### 2.3 Tipe – tipe cipher aliran

Sebuah algoritma cipher aliran menciptakan elemen – elemen yang berurutan dari sebuah *keystream*, dengan berbasis *internal state*. *State* ini diperbaharui dengan dua cara; apabila *state* berubah dengan tidak bersesuaian dengan plainteks atau cipherteks, maka cipher ini dikategorikan sebagai cipher aliran yang sifatnya *synchronous*, sedangkan sebaliknya, apabila cipher aliran mengubah *state* berdasarkan digit cipherteks sebelumnya, ia termasuk cipher aliran yang sifatnya *self-synchronising*.

### 2.3.1 *Synchronous stream cipher*

Pada *synchronous stream cipher*, sebuah aliran digit yang acak-semu diciptakan secara terpisah dengan plainteks (dalam proses enkripsi) atau dengan cipherteks (untuk dekripsi). Pada bentuk yang paling umum, bit – bit diciptakan secara acak-semu, kemudian *keystream* yang ada akan dioperasikan dengan plainteks menggunakan operasi XOR. Proses ini dinamakan proses *binary additive stream cipher*.

Pada *synchronous stream cipher*, pengirim dan penerima harus sejalan, sehingga proses dekripsi dapat berjalan dengan sukses. Apabila digit ditambahkan atau dihilangkan dari pesan selama transmisi, maka dikatakan bahwa sinkronisasi telah hilang. Untuk mengembalikan sinkronisasi kedalam proses, berbagai cara dapat digunakan. Ada yang menggunakan *offset* bit untuk mendapatkan proses dekripsi yang benar, dan ada pula pendekatan lain yaitu penandaan pada cipherteks dengan penanda untuk mendapatkan dekripsi yang benar.

Namun apabila gangguan yang terjadi berupa kerusakan pada bit, bukan penambahan atau kehilangan informasi, maka hanya satu digit plainteks yang akan rusak. Kerusakan pada bit tersebut tidak akan menjalar ke bit – bit berikutnya . hal ini berguna apabila terjadi tingkat kesalahan yang tinggi pada jaringa, namun kelebihan ini dapat menjadi bumerang karena cipher aliran yang bersifat *synchronous* ini menjadi terdedah pada serangan aktif.

Apabila seorang penyerang dapat mengganti sebuah digit pada cipherteks, ada kemungkinan si penyerang itu akan dapat membuat perubahan yang dapat diatur, sebab membalik bit pada cipherteks akan menyebabkan bit yang sama terbalik pula pada plainteks

### 2.3.2 *Self-synchronizing stream ciphers*

Pendekatan lain menggunakan n – digit dari cipherteks untuk menghitung *keystream*. Skema ini dikenal sebagai *Self-synchronizing stream ciphers*, *asynchronous stream ciphers*, atau *ciphertext autokey* (CTAK).

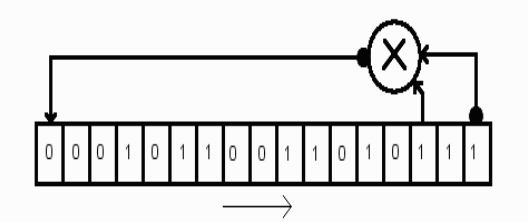
Keunggulan dari skema ini adalah penerima akan secara otomatis mensinkronkan diri dengan *keystream generator* setelah menerima cipherteks sebanyak N – digit, sehingga membuat skema ini lebih mudah direcover apabila ada digit yang ditambahkan atau dihilangkan dali aliran pesan.

Kesalahan pada satu buah bit memiliki efek terbatas hanya pada N – digit pada plainteks. Lebih sulit juga untuk melakukan serangan aktif pada skema ini, dibandingkan dengan melakukan serangan serupa pada skema sebelumnya.

Contoh algoritma yang menggunakan skema sejenis ini juga dapat dijumpai pada mode Cipher Feedback pada algoritma cipher blok.

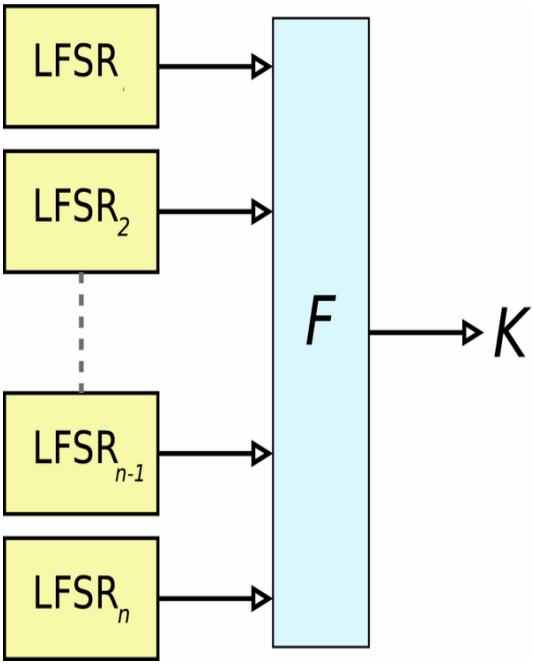
## 2.4 *Linear Feedback Shift Register*

Cipher aliran berbasis bit seringkali dibangun menggunakan *Linear Feedback Shift Register* (LFSR) karena metode ini mudah diimplementasi pada hardware dan dapat dikaji secara matematis. Penggunaan LFSR ini sendiri tidak menjamin tersedianya tingkat keamanan yang tinggi. Beberapa skema lainnya telah diusulkan untuk meningkatkan keamanan dari LFSR ini.




**Gambar 1 - Sebuah LFSR 17 - bit**

### 2.4.1 *Non – linear combining funcition*




**Gambar 2 -Penggunaan fungsi boolean dengan n - buah input**

Salah satu pendekatan yang diusulkan adalah penggunaan n – buah LFSR secara parallel, kemudian outputnya digabung menggunakan sebuah fungsi binary Boolean (F) yang menerima n – buah input.

Karena LFSR sebenarnya bersifat linear, salah satu teknik untuk menghilangkan sifat linear ini adalah dengan memasukkan output dari beberapa LFSR yang parallel kedalam fungsi Boolean yang tidak linear untuk membentuk kombinasi. Sifat dari fungsi ini sangat penting untuk memastikan keamanan dari skema hasil untuk menghindarkan kemungkinan dilakukannya serangan korelasi.

### 2.4.2 *Clock-controlled generators*

Secara normal, langkah – langkah dalam LFSR dilakukan secara regular. Pendekatan lain untuk menghilangkan sifat linear dari LFSR adalah dengan memutar LFSR secara tidak biasa. Hal ini dapat dicapai dengan mengontrol output dengan menggunakan LFSR kedua. Generator ini biasanya mengandung *stop-and-go generator*, *alternating step generator*, dan *shrinking generator*.

*Stop-and-go generator* terdiri dari dua LFSR. LFSR pertama diputar apabila keluaran dari LFSR kedua bernilai 1, namun apabila keluaran dari LFSR bernilai 0, maka LFSR pertama akan meneruskan keluaran sebelumnya. Keluaran ini kemudian dikombinasikan dengan keluaran dari LFSR ketiga yang diputar secara reguler.

*Shrinking generator* memiliki pendekatan berbeda. Generator ini menggunakan dua buah LFSR yang kedua – duanya diputar secara reguler. Apabila keluaran dari LFSR pertama bernilai 1, maka keluaran dari LFSR kedua dipakai menjadi keluaran dari generator. Apabila LFSR pertama mengeluarkan bit bernilai 0, maka keluaran dari LFSR kedua tidak dipakai dan tidak ada bit ang dikeluarkan dari generator ini.

Mekanisme ini memiliki kelemahan untuk jenis serangan *timing attack*, yang dapat dilakukan pada generator kedua. Hal ini terjadi karena kecepatan generator menghasilkan keluaran adalah variabel terhadap *state* dari generator kedua. Hal ini dapat diatasi dengan cara menambahkan buffer untuk menyangga keluaran.

### 2.4.3 *Filter generator*

Pendekatan lain untuk meningkatkan keamana pada LFSR adalah dengan melewatkan semua keadaan bit pada sebuah LFSR ke sebuah fungsi penyaring non – linear.

## 3. Komunikasi Selular

Pada zaman yang serba cepat dan menuntut kecepatan komunikasi seperti ini, diperlukan teknologi – teknologi yang memungkinkan masyarakat untuk tetap berkomunikasi meskipun sedang berada di luar rumah. Untuk itu lahirlah teknologi seluler yang pada saat ini berbasis pada dua standar, yaitu *Code Division Multiple Access / Time division Multiple Access* (CDMA / TDMA) yang merupakan standard yang ditetapkan oleh TIA di Amerika Utara dan *Global System for Mobile Communication* (GSM) yang menjadi standar di Eropa dan banyak negara di dunia.

### 3.1 CDMA

*Code Division Multiple Access* (CDMA) adalah sebuah bentuk *multiplexing* (bukan sebuah skema modulisasi) dan sebuah metode akses multipel yang tidak membagi saluran berdasar waktu (seperti pada TDMA) maupun berdasar frekuensi (seperti pada FDMA), namun

sebaliknya mengkodekan data dengan kode khusus yang bersesuaian dengan setiap saluran dan menggunakan properti interferensi konstruktif dari kode khusus tersebut untuk melakukan *multiplexing*.

Meski sebenarnya penggunaan CDMA tidak dikhususkan hanya untuk saluran komunikasi telepon, namun CDMA sudah telanjur dikenal sebagai sebuah sistem telepon seluler yang menggunakan skema akses multipel seperti diatas. CDMA baru saja masuk ke Indonesia dengan dipelopori oleh Telkom Fleksi, kemudian diikuti oleh Mobile 8 Fren, Esia Bakri, dan beberapa pemain lainnya

Selain digunakan untuk jaringan telepon, CDMA juga digunakan untuk beberapa sistem komunikasi terkenal, seperti *Global Positioning System* (GPS) dan sistem satelit OmniTRACS yang digunakan untuk pengiriman logistik.

CDMA cocok digunakan untuk transfer data dengan sifat yang kurang konsisten dan dimana waktu tunggu dapat diterima. Oleh karena itu, CDMA digunakan pada aplikasi Wireless LAN.

**3.2 Detail Teknik CDMA**

Secara garis besar, CDMA dapat dibagi menjadi dua, yaitu *Synchronous* CDMA dan *Asynchronous* CDMA.

**3.2.1 *Synchronous* CDMA**

CDMA sinkron, atau yang juga dikenal sebagai Code Division Multiplexing (CDM) memanfaatkan sifat dari inti matematikanya yaitu ortogonalitas.

Misalnya sinyal data direpresentasikan sebagi vektor, maka contohnya, string "1101" akan direpresentasikan sebagai vektor (1, 1, 0, 1). Untuk memudahkan maka vektor tadi akan dinamai sebagai A. Operasi yang dilakukan pada vektor adalah perkalian dot untuk mengalikan vektor, dengan menjumlahkan hasil perkalian dari komponen – komponen. Contohnya adalah

$$\mathbf{a} = (1, 0, 1, 1)$$
$$\mathbf{b} = (1, -1, -1, 0)$$

Yang akan ditulis sebagai :

$$\mathbf{a} \cdot \mathbf{b}$$

Operasinya akan berjalan sebagai berikut :

$$(1) \times (1) + (0) \times (-1) + (1) \times (-1) + (1) \times (0) = 1 + (-1) = 0$$

Pada suatu kasus khusus dimana perkalian dot dari dua vektor adalah 0, maka dikatakan bahwa vektor – vektor tersebut saling tegak lurus (ortogonal).

Perkalian produk memiliki beberapa sifat yang akan membantu dalam memahami CDM. Untuk vektor a, b, dan c :

$$\mathbf{a} \cdot (\mathbf{b} + \mathbf{c}) = \mathbf{a} \cdot \mathbf{b} + \mathbf{a} \cdot \mathbf{c}, \quad \text{and}$$

$$\mathbf{a} \cdot k\mathbf{b} = k(\mathbf{a} \cdot \mathbf{b}).$$

Dimana k adalah konstanta bukan vektor.

Akar kuadrat dari $\mathbf{a} \cdot \mathbf{a}$ adalah sebuah bilangan real yang disebut sebagai panjang vektor a, ditulis sebagai :

$$||\mathbf{a}|| = \sqrt{\mathbf{a} \cdot \mathbf{a}}.$$

Apabila vektor a dan vektor b saling tegak lurus, maka *rules* dibawah ini akan berlaku :

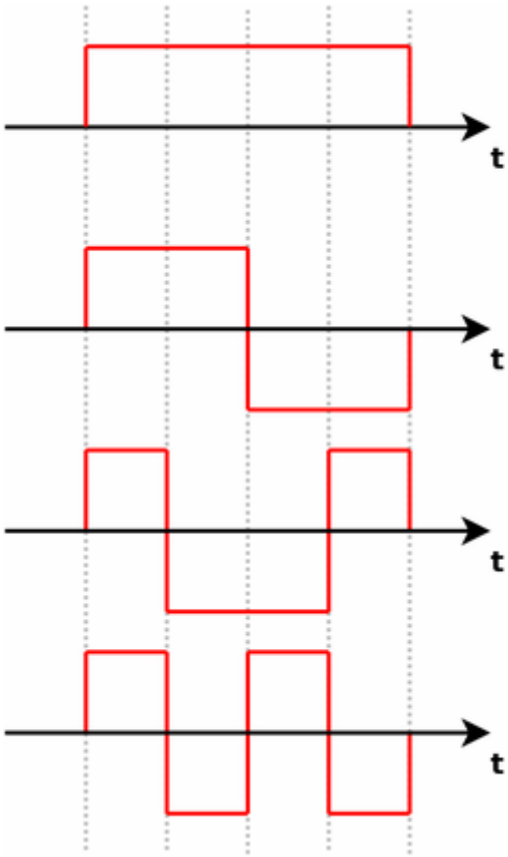
$$\mathbf{a} \cdot (\mathbf{a} + \mathbf{b}) = ||\mathbf{a}||^2 \quad \text{since} \quad \mathbf{a} \cdot \mathbf{a} + \mathbf{a} \cdot \mathbf{b} = ||a||^2 + 0,$$

$$\mathbf{a} \cdot (-\mathbf{a} + \mathbf{b}) = -||\mathbf{a}||^2 \quad \text{since} \quad -\mathbf{a} \cdot \mathbf{a} + \mathbf{a} \cdot \mathbf{b} = -||a||^2 + 0,$$

$$\mathbf{b} \cdot (\mathbf{a} + \mathbf{b}) = ||\mathbf{b}||^2 \quad \text{since} \quad \mathbf{b} \cdot \mathbf{a} + \mathbf{b} \cdot \mathbf{b} = 0 + ||b||^2,$$

$$\mathbf{b} \cdot (\mathbf{a} - \mathbf{b}) = -||\mathbf{b}||^2 \quad \text{since} \quad \mathbf{b} \cdot \mathbf{a} - \mathbf{b} \cdot \mathbf{b} = 0 - ||b||^2.$$

Dimiliki sebuah himpunan vektor yang saling tegak lurus satu sama lain (biasanya vektor – vektor ini dibangun secara khusus untuk memudahkan proses dekoding, ada kolom – kolom atau baris – baris yang berasal dari matriks Walsh yang dibangun dari fungsi Walsh).




**Gambar 3 - 4 contoh sinyal digital yang ortogonal**

Asosiasikan dengan sebuah pengirim sebuah vektor dari himpunan ini, misalnya v, yang disebut dengan kode chip. Asosiasikan sebuah digit nol dengna vektor –v. Sebagai contoh, apabila v = (1, -1), maka vektor biner (1, 0, 1, 1) akan berkoresponden dengan (1, -1, -1, 1, 1, -1, 1, -1). Untuk berikutnya, vektor ini akan disebut vektor terkirim.

Setiap pengirim memiliki vektor yang unik dan berbeda yang dipilih dari himpunan vektor diatas, namun cara pembangunan vektor terkirim adalah sama.

Sifat fisik dari interferensi menyatakan bahwa apabila ada dua sinyal pada suatu titik memiliki fase yang sama, maka dua sinyal tersebut akan dijumlahkan untuk mendapatkan amplitudo sebanyak dua kali amplitudo dari masing – masing sinyal. Namun apabila fase keduanya berbeda, maka apabila amplitudo kedua sinyal tersebut berbeda, maka amplitudo yang lebih kuat akan dikurangkan dengan amplitudo yang lebih lemah.

Secara digital, sifat ini dapat dimodelkan dengan operasi penjumlahan vektor terkirim, setiap komponen vektor. Jadi, apabila ada dua pengirim, dimana keduanya mengirim secara simultan, yang pertama dengan kode chip (1, -1) dan vektor data (1, 0, 1, 1) dan yang kedua dengan kode chip (1, 1) dan vektor data (0, 0, 1, 1), maka sinyal mentah yang diterima akan menjadi penjumlahan dari vektor terkirim

(1,-1,-1,1,1,-1,1,-1)+(-1,-1,-1,-1,1,1,1,1)

=(0,-2,-2,0,2,0,2,0).

Apabila penerima menerima sinyal diatas dan ingin melihat isi data yang dikirimkan oleh pengirim dengan kode chip (1, -1), maka penerima akan memanfaatkan sifat – sifat yang menjadi dasar bagi teknik ini.

Ambil dua komponen pertama dari vektor yang diterima, yaitu (0, -2). Sekarang lakukan dot product (0, -2).(1, -1) = (0)(1)+(-2)(-1) = 2. karena hasilnya adalah positif, maka dapat disimpulkan bahwa ada satu digit yang dikirim. Apabila hasil perkalian produknya adalah negatif, maka yang dikirim adalah digit 0. lakukan hal ini pada sisa dari vektor yang diterima tersebut, dan akan muncul vektor data yang dikirim dengan menggunakan kode chip (1, -1).

Demikian pula untuk mendapatkan kode yang dikirim dengan menggunakan kode chip (1, 1).

### 3.2.2 *Asynchronous* CDMA

Contoh sebelumnya mengenai himpunan vektor ortogonal dengan menggunakan barisan Walsh mendeskripsikan bagaimana dua buah pengguna dapat diakses secara multipel bersama pada sistem yang sinkron. Contoh sekuens Walsh pada gambar 3 diatas akan mencakup sampai 4 pengguna, dan umumnya, matriks Walsh berukuran N x N dapat digunakan untuk mengakses secara multipel N pengguna. *Multiplexing* membutuhkan setiap pengguna dikoordinatkan sehingga tiap pengguna mengirimkan sekuens v masing – masing (atau komplemennya, -v) pada saat yang bersamaan. Maka, teknik ini dapat digunakan pada sambungan basis ke mobil, dimana semua transmisi yang berasal dari transmitter yang sama dapat dikoordinasi secara sempurna.

Namun sebaliknya, sambungan mobil ke basis tidak dapat dikoordinasi secara akurat, hal ini disebabkan oleh pergerakan dari device mobil, dan membutuhkan pendekatan yang berbeda. Untuk itu digunakanlah sekuens yang acak semu dan unik pada sistem CDMA Asinkron (PN, pseudo noise). Sekuens PN ini secara statistik saling bebas, dan jumlah banyak sekuens PN ini adalah Multiple Access Interference (MAI), yang didekati dengan Gaussian noise process (melalui jumlah yang cukup banyak) apabila semua pengguna diterima dengan level power yang sama, maka variansi dari MAI meningkat seiring dengan peningkatan jumlah pengguna

Semua bentuk CDMA menggunakan *spread spectrum process gain* untuk memperbolehkan penerima untuk membedakan secara parsial sinyal – sinyal yang tidak diinginkan. Sinyal dengan kode chip yang diinginkan diterima, sedangkan sinyal dengan kode chip yang berbeda (atau kode tersebar yang sama tapi dengan offset waktu yang berbeda) yang nampak sebagai gamngguan pita lebar yang dikurangi dengan *process gain*.

Karena setiap user menciptakan MAI, maka mengontrol kekuatan sinyal adalah isu yang penting dengan CDMA transmitter. Sebuah penerima CDM, TDMA, atau FDMA dapat menolak mentah – mentah sinyal kuat namun menggunakan kode, slot waktu dan frekuensi yang berbeda karena ortogonalitas sistem tersebut. Hal ini tidak berlaku pada CDMA asinkron. Penolakan sinyal yang tidak diinginkan hanya dilakukan secara sebagian. Apabila ada sebagian atau semua sinyal yang tidak diinginkan lebih kuat daripada sinyal yang diinginkan tersebut, maka sinyal tersebut akan mengalahkan sinyal yang diinginkan tersebut.

Hal ini menimbulkan suatu kebutuhan bagi sistem CDMA asinkron untuk menyeimbangkan kekuatan sinyal – sinyal pada penerima. Pada selular CDMA, stasiun basis menggunakan sebuah skema kontrol power loop tertutup cepat untuk mengontrol tiap – tiap pancaran sinyal dari mobile device secara ketat.

### 3.3 Enkripsi pada CDMA

Sistem seluler standard di Amerika Utara yang didesain oleh TIA, baik TDMA maupun CDMA, sama – sama menggunakan arsitektur keamana yang sama. Empat primitif kriptografi yang digunakan pada sistem ini dan dideskripsikan pada standar TIA adalah:

1. CAVE,
Utuk menangani protokol otentifikasi dan pembuatan kunci

2. ORYX,
Agoritma ini akan dibahas lebih mendalam pada bagian 4

3. CMEA,
Blok cipher sederhana yang digunakan untuk mengenkripsi pesan pada saluran data

4. Untuk keamanan suara, sistem TDMA menggunakan mask XOR, sedangkan sistem CDMA menggunakan teknik spektrum tersebar dengan kunci dan dikombinasi dengan mask LFSR.

## 4 Algoritma ORYX

Seperti GSM yang telah memiliki standar kriptologi mereka sendiri yang digunakan di Eropa, Telecommunications Industri Association(TIA), telah membuat standard yang digunakan di Amerika Utara.

Standar yang sekarang digunakan untuk mengenkripsi data wireless digital dikenal sebagai cipher ORYX. Seperti algoritma A5/1 yang digunakan pada jaringan GSM, cipher ORYX adalah sebuah algoritma cipher yang berbasis Linear Feedback Shift Registers (LFSR), acuan mengenai LFSR dapat dilihat pada bagian 2.4, meskipun pada bagian ini akan disinggung lagi mengenai algoritmanya, namun hanya disinggung secara singkat saja

### 4.1 Overview

Cipher ORYX adalah sebuah stream cipher yang digunakan sebagai sebuah generator *keystream.* Keluaran dari generator ini adalah sebuah aliran bytes yang bersifat acak semu. Aliran byte ini kemudian di XOR kan dengan plainteks untuk menciptakan cipherteks, dan kemudian untuk melakukan dekripsi, cipherteks diXOR kan kembali dengan aliran byte acak semu yang sama untuk mendapatkan plainteks awal.

Seperti yang diterangkan diatas, maka fokus dari algoritma ORYX ini adalah penciptaan *keystream* yang akan digunakan untuk

mengXOR kan plainteks untuk melakukan enkripsi.

**4.2 Enkripsi Algoritma ORYX**

Cipher ORYX ini terdiri dari 4 komponen utama, yang terdiri dari tiga buah 32-bit Linear Feedback Shift Register (LFSR) dan sebuah S-box yang mengandung permutasi L.

Tiga buah LFSR tersebut akan disebut dengan $LFSR_A$, $LFSR_B$, dan $LFSR_K$ . S-box diatas mengandungi permutasi L yang merupakan permutasi integer yang bernilai dari 0 – 255.

Fungsi *Linear Feedback* yang digunakan untuk $LFSR_K$ adalah sebagai berikut :

$$P_K: \; x_{n+1} = x_n^{32} + x_n^{28} + x_n^{19} + x_n^{18} + x_n^{16} + x_n^{14} + x_n^{11} + x_n^{10} + x_n^{9} + x_n^{6} + x_n^{5} + x_n + 1$$

$LFSR_A$ menggunakan dua buah fungsi *Linear Feedback*, yang berisikan :

$$P_{A1}: \; x_{n+1} = x_n^{32} + x_n^{26} + x_n^{23} + x_n^{22} + x_n^{16} + x_n^{12} + x_n^{11} + x_n^{10} + x_n^{8} + x_n^{7} + x_n^{5} + x_n^{4} + x_n^{2} + x_n + 1$$

Dan

$$P_{A2}: \; x_{n+1} = x_n^{32} + x_n^{27} + x_n^{26} + x_n^{25} + x_n^{24} + x_n^{23} + x_n^{22} + x_n^{17} + x_n^{13} + x_n^{11} + x_n^{10} + x_n^{9} + x_n^{8} + x_n^{7} + x_n^{2} + x_n + 1$$

Dan $LFSR_B$ menggunakan fungsi sebagai berikut :

$$P_B: \; x_{n+1} = x_n^{32} + x_n^{28} + x_n^{19} + x_n^{18} + x_n^{16} + x_n^{14} + x_n^{11} + x_n^{10} + x_n^{9} + x_n^{6} + x_n^{5} + x_n + 1$$

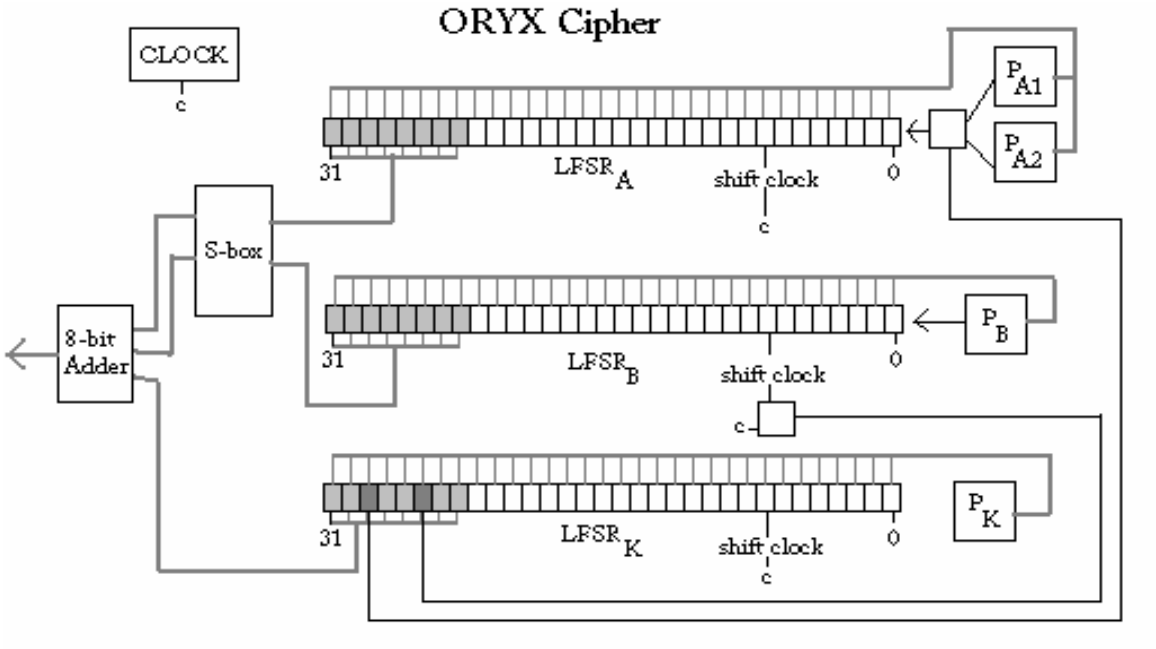
Pemutasi L tidak berubah selama pemanggilan. Permutasi tersebut diproduksi oleh algoritma yang diketahui yang diinisiasi dengan sebuah nilain yang dikirim pada waktu *setup* pemanggilan.

Setiap byte dari *keystream* dibuat dengan mengikuti algoritma berikut ini:

1. $LFSR_K$ dilewati sekali dengan fungsi $\mathbf{P_K}$
2. $LFSR_A$ dilewati sekali menggunakan salah satu fungsi dari $\mathbf{P_{A1}}$ atau $\mathbf{P_{A2}}$. Fungsi mana yang dipilih untuk digunakan berdasar dari salah satu 8 bit atas dari $LFSR_K$
3. Byte tinggi dari llll ditambahkan ke byte tinggi dari lll setelah dipermutasi dengan L dan byte tinggi dari llll setelh dipermutasi dengan L modulus 256 untuk membuat sebuah byte pada keystream. Rumus untuk membuat *keystream* pada ORYX cipher yang umumnya dipakai adalah sebagai berikut

$$\text{Keystream} = (\text{High8}_K + L[\text{High8}_A] + L[\text{High8}_B]) \bmod 256$$

Untuk ilustrasi bagaimana proses enkripsi dengan algoritma ORYX, dapat dilihat gambar dibawah ini :




**4.3 Dekripsi Algoritma ORYX**

Proses dekripsi pada algoritma oryx adalah dengan cara menXOR kan byte – byte pada cipherteks sesuai dengan *keystream* yang digenerate dengan cara yang sama dengan cara untuk membuat *keystream* pada proses enkripsi ORYX.

**4.4 Prosedur Serangan Pada Algoritma ORYX**

Byte tinggi pada $LFSR_K$ yang menentukan penjadwalan kunci pada $LFSR_A$ dan $LFSR_B$ dapat bervariasi sesuai dengan implementasi masing – masing. Meskipun berbeda pada tiap implementasi tetapi byte tinggi tersebut tidak berubah selama pembuatan *keystream* dan diketahui oleh penyerang. Karena itu, karena algoritma pembuatan *keystream* tersebut diketahui oleh penyerang dan yang tidak diketahui oleh penyerang hanyalah isi awal dari tiga buah 32-bit *Linear Feedback Shift Register* ini.

Karena hal diatas, ORYX dapat dianggap memiliki 96 – bit ruang kunci. Karena memiliki 96 – bit ruang kunci, maka ada $\mathbf{2^{96}}$ kemungkinan nilai awal dari LFSR – LFSR tersebut. Banyaknya kemungkinan ini membuat upaya untuk menebak initial state dari $LFSR_K$, $LFSR_A$ , dan $LFSR_B$ menjadi tidak mungkin, demikian pula upaya untuk melakukan brute force pada initial state sistem.

Namun apabila status awal generator dapat dibagi - bagi menjadi bagian yang lebih kecil, dan dapat dilakukan penebakan pada bagian dari status awal generator, sebab ruang kuncinya sdah mengecil, maka apabila tebakan kecil tersebut benar, dapat dilakukan serangan pada *keystream generator* tersebut. Setelah pengecekan apakah tebakan yang kita lakukan benar, maka dilakukan pendekatan divide and conquer.

Sifat dari Algoritma ORYX yang menyebabkan serangan ini dapat dilakukan secara efektif adalah karena dua level $LFSR_K$ yang isinya mengontrol seleksi dari *feedback* polinom untuk $LFSR_A$ dan jumlah $LFSR_B$ dilalui kedua – duanya berada pada 8 stage tinggi dari $LFSR_K$ . Karena byte *keystream* dibentuk dari isi dari 8

stage tinggi dari ketiga status LFSR, maka ruang kunci dapat dibagi, dan serangan akan difokuskan pada 24 bit ini.

Cara lain bagi penyerang untuk mendapatkan bagian dari keystream adalah dengan serangan pasangan plainteks dan cipherteks.

### 4.5 Algoritma Serangan Pada Algoritma ORYX

Tandai delapa bit tinggi dari tiga buah LFSR pada saat byte ke I dari keystream digenerate oleh *High*8*A*(*i*), *High*8*B*(*i*) dan *High*8*K*(*i*). Isi awal adalah *High*8*A*(0), *High*8*B*(0) dan *High*8*K*(0), dah semua register dilalui sebelum byte pertama dari keystream, yang ditandai dengan Z(1). Untuk membuat byte keystream ke Z(i+1) pada waktu i+1, LFSR*K* dilalui sekali, kemudian LFSR*B* dilalui antara sekali atau dua kali. Isi dari *High*8*A*(*1*), *High*8*B*(*1*) dan *High*8*K*(*1*) kemudian dikombinsikan untuk membentuk byte keystream ke Z(i+1). Oleh karena itu, maka tidak perlu penebakan terhadap 24 bit isi dari shift register. Apabila isi dari *High*8*A*(*1*), *High*8*B*(*1*) dapat ditebak, maka dapat digunakan byte pertama dari keystream Z(1) dan fungsi kombinasi untuk menghitung isi dari *High*8*K*(*1*) yang bersesuaian.

Hal itu menandakan bahwa serangan dapat saja hanya dilakukan untuk 16 – bit subkey saja, isi dari High8A(1) dan High8B(1). Untuk menebak 16 – bit subkey diatas, dapat digunakan Z(1) untuk menghitung isi High8K(1) yang bersesuaian. Setelah perhitungan ini, serangan akan berlangsung secara iteratif setelah terbentuk tebakan – tebakan terhadap High8A(i), High8B(i) dan High8K(i) yang konsisten terhadap keystream yang diketahui. Pada tiap iterasi, sebuah himpunan prediksi untuk keystream yang berikutnya dibentuk, dan tebakan dievaluasi dengan memperbandingkan byte keystream yang sudah diketahui dengan nilai yang telah diprediksi tersebut.

Pada iterasi ke i, fakta bahwa setelah melewati tiga LFSR untuk memproduksi output byte yang berikutnya, maka *High*8*K*(*i+1*), *High*8*A*(*i+1*) secara efektif digeser bitnya dengan jumlah pergeseran belum diketahui, dan berdasar pada High8K(i+1), High8B(i+1) memiliki satu atau dua bit yang belm diketahui, untuk setiap byte keystream keluaran.

Dari hal diatas, dapat dicoba sebanyak 12 kombinasi dari bit – bit input yang baru ini, dan byte keluaran dari setiap kasus pada kombinasi tersebut. Paling banyak ada 12 hasil byte keluaran yang berbeda yang akan sesuai dengan tebakan terhadap High8A(i), High8B(1) , dan High8K(i). kemudian keystream byte Z(i+1) dibandingkan dengan byte keluaran yang diprediksi.

Apabila Z(i+1) sama dengan byte keluaran yang diprediksi dari 12 kombinasi diats, maka ada sebuah himpunan nilai yang eksis untuk High8K(i+1), High8A(i+1), dan High8B(i+1). Nilai ini kemudian akan digunakan untuk iterasi serangan berikutnya.

Terkadang, ketika kurang dari 12 keluaran berbeda dan byte keystream sama dengan lebih dari satu kombinasi dari bit input yang baru, maka perlu dipertimbangkan ada lebih dari satu kemungkinan nilai – nilai untuk bit – bit tinggi dari tiga register LFSR tersebut. Hal ini berarti jalur tebakan konsisten yang terbentuk sudah bercabang. Pada kondisi bercabang ini, maka digunakanlah metode depth-first search.

Apabila byte keystream tidak sama dengan byte keluaran yang diprediksi, maka tebakan isi dari High8A(i) dan High8B(i) tidak benar, kemudian dari sini akan dilakukan *backtrack* sampai ke percabangan terakhir dan jalur yang lain akan dicoba. Apabila tidak ada jalur yang benar pada percabangan – percabangan tersebut, maka tebakan awal isi dari High8A(i) dan High8B(i) tidak benar, dan dilakukan lagi tebakan 16 – bit awal dan lakukan kembali algoritma diatas.

Ketika ditemukan sebuah jalur yang cukup dalam yang berisikan tebakan yang benar, maka dapat diasumsikan bahwa isi awal dari bit tinggi tiga register LFSR yang ditebak di depan adalah benar. Ini berarti isi dari tiga LFSR pada state awal saat byte pertama keystream diproduksi. Untuk 24 tebakan isi 3 register LFSR yang berurutan, untuk langkah ke 2 sampai langkah ke 25, tiap himpunan nilai High8K(i) dan High8A(i) menentukan bit pada state LFSRK dan LFSRA. High8B(i) memberikan satu atau dua bit pada state LFSRB. Setelah 32-bit state LFSR telah direkonstruksi, pada saat pertama kali byte keystream dibuat, kemudian status LFSR dapat ditrack kebelakang unutk merecover state awal dari tiga LFSR tersebut, yang adalah kunci rahasia dari algoritma ORYX ini.

Seluruh kunci dapat direcover dengan menggunakan minimum 25 byte keystream yang diketahui, dan tebakan sebanyak kurang dari 2^16 tebakan. Kemudian status awal yang telah didapatkan tadi dibuat sebagai kandidat keystream dan apabila kandidat keystream tersebut sama dengan keystream yang telah diketahui, maka serangan ini sukses, sedangkan apabila tidak, maka algoritma untuk mencari kunci rahasia ORYX ini harus diulang dari awal, dengan tebakan 16 – bit yang berbeda.

Pada prakteknya, dibutuhkan lebih dari 25 byte keystream yang diketahui untuk menghilangkan ambiguitas pada beberapa bit terakhir pada LFSR. Oleh karena itu, diperlukan penelusuran pada beberapa jalur percabangan yang salah untuk meyakinkan bahwa telah didapat kunci yang benar.

**4.6 Efek dari Banyaknya Keystream yang Diketeahui**

Panjang *keystream* minimum yang dibutuhkan untuk melakukan serangan ini adalah 25 byte. Satu byte untuk mendapatkan

The minimum length of keystream required for this attack to be successful is 25. satu bit untuk mendapatkan nilai dari $High8_K(1)$, delapan bit yang diketahui pada setiap 32 bit status awal LFSR, dan kemudian satu byte untuk merecover setiap 24 bit pada tiap status awal ketiga LFSR tersebut. Semakin banyak keystream yang tersedia, semakin banyak kemungkinan sukses yang ada. Namun apabila jumlah panjang *keystream* yang diketahui kurang dari 25 byte, serangan ini masih dapat dilakukan, dan teknik exhaustive search dilakukan untuk mencari isi dari stage terakhir.

**4.7 Serangan *Ciphertext Only***

Serangan pada algoritma ORYX ini juga dapat dilakukan dengan metode serangan pada cipherteks. Namun hal ini harus dibarengi dengan data statistik plainteks untuk meningkatkan kemungkinan sukses serangan. Apabila statistik yang dimiliki cukup bagus, maka serangan jenis ini dapat dilakukan dengan panjang cipherteks hanya beberapa ribu bytes.

Untuk melakukan serangan pada cipherteks, dapat dimulai dengan mengidentifikasi kemungkinan kata berisi tujuh karakter yang mungkin muncul pada plainteks. Contoh kata yang mungkin adalah “login: ”, dan banyak kata lain. Kemudian string yang mungkin tersebut kemudian digeser – geser posisinya pada cipherteks untuk menemukan padanannya dengan pesan teks yang benar.

Apabila plainteks yang mungkin tadi diletakkan secara benar, maka kemudian didapatkan potongan keystream dengan panjang sama dengan panjang plainteks tersebut. Kemudian cara known – plainteks diatas dapat diterapkan pada potongan keystream ini.

Apabila setiap jalur tebakan tidak ada yang benar pada N=7 byte dari teks yang diketahui, maka tampaknya peletakan plainteks yang mungkin diatas salah. Apabila sebaliknya, dapat disimpulkan bahwa telah ditemukan padanan yang valid. Ini tampaknya terlalu optimistis, namun kemudian akan ditunjukkan bahwa kemungkinan error sebenarnya cukup kecil.

Dengan prosedur ini, padanan yang salah seharusnya cukup jarang, sebab jalur – jalur yang salah pada tiap tebakan seharusnya cukup dangkal. Setelah menganalisa byte pertama, ada 216 kemungkinan byte tinggi dari tiap register LFSR.

Setelah menganalisis byte kedua, maka kemungkinan terjadinya kesalahan yang tidak terdeteksi adalah hanya 0,0459. setelah byte ketiga, juga ada 0,0459 kemungkinan kesalahan. Ini berarti hanya $2^{16}$ x $(0{,}0459)^6$ atau 0,00061 peluang kesalahan dapat terjadi setelah pengecekan byte ketujuh.

Oleh karena itu, dengan lebih dari seribu byte cipherteks, dapat diharapkan bahwa tidak mungkin ada kesalahan pemadanan untuk kata yang mungkin sepanjang minimal tujuh karakter. Penggunaan kata yang lebih panjang akan lebih meminimalisasi kemungkinan terjadinya kesalahan.

Untuk mengecek posisi cipherteks untuk padanan yang mungkin dibutuhkan 216 percobaan, dengan panjang cipherteks yang kurang dari seribu byte, maka total usaha komputasi yang digunakan untuk mengetes kata yang mungkin kuranga dari 226, dan pengecekan terhadap kamus kata – kata yang mungkin menjadi mudah.

Sepanjang dimiliki panjang cipherteks dan dapat mengidentifikasi himpunan kata – kata yang mungkin, seharusnya mudah untuk menemukan dua atau tiga padanan dan kemudian merecover seluruh kunci algoritma ORYX ini. Dengan kata lain, serangan cipherteks pada algoritma ORYX memiliki tingkat kompleksitas yang rendah, ketika beberapa pengetahuan dari statistik plainteks diketahui.

Komputasi yang dibutuhkan tidak terlalu banyak dan jumlah cipherteks yang dibutuhkan juga tidak terlalu panjang. Sehingga serangan – serangan sejenis ini tampak sangat mungkin.

**5. Analisis Terhadap Keamanan ORYX**

Algoritma ORYX adalah algoritma stream cipher yang cukup sederhana. Untuk melakukan serangan known plainteks pada algoritma ini tidak dibutuhkan komputasi yang berat.

Hal ini dapat dilihat dari jumlah percobaan serangan dan jumlah *keystream* yang hanya berjumlah sekitar 65000 percobaan dan 25 byte *keystream.* Jumlah yang relatif kecil untuk ukuran komputasi sekarang ini menyebabkan algoritma ini mudah terdedah kepada serangan.

Yang lebih mengkhawatirkan lagi adalah dimungkinkannya serangan berjenis Cipherteks only attack yang hanya membutuhkan data statistik dari plainteks dan hanya membutuhkan pencocokan kata yang mungkin dengan clear teks, kemudian algoritma penyerangan seperti *known plainteks attack* dapat dilakukan.

**6. Algoritma Pembanding : A5/1 pada GSM**

Algoritma A5/1 adalah algoritma berbasis cipher aliran yang digunakan untuk voice privacy di komunikasi wireless pada GSM.

**6.1. Sejarah**

Algoritma A5/1 digunakan di Eropa dan Amerika Serikat. Cipher yang lebih lemah, A5/2, digunakan pada Negara – Negara yang tidak begitu membutuhkan algoritma yang kua. Kedua cipher ini, dikembangkan pada 1989, pada awalnya dirahasiakan, namun desain algoritma ini bocor pada tahun 1994 dan direverse engineered pada tahun 1999 oleh Marc Briceno dari GSM telepon.

**6.2 Dekripsi Algoritma**

Percakapan telepon pada GSM dikirim sebagai urutan frames satu frame dikirim tiap 4.6 milidetik, dan 228 bit panjangnya, 114 bit dari tiap arah. A5/1 didesain untuk memproduksi 228 bit keystream yang kemudian diXORkan dengan frame. A5/1 diinisialisasi dengan menggunakan kunci 64 bit dengan 22 bit nomor frame yang diketahui. Pada implementasi GSM, 10 dari bit kunci diisi 0, sehingga panjang kunci efektif adalah 54 bits.

A5/1 berbasis kombinasi dari 3 Linear Feedback Shift Register dengan clockin yang tidak regular. Tiga shift register ini dispesifikasikan sebagai berikut :

| LFSR number | Length in bits | Characteristic polynomial | Clocking bit | Tapped bits |
|---|---|---|---|---|
| 1 | 19 | $x^{19} + x^5 + x^2 + x + 1$ | 8 | 13, 16, 17, 18 |
| 2 | 22 | $x^{22} + x + 1$ | 10 | 20, 21 |
| 3 | 23 | $x^{23} + x^{15} + x^2 + x + 1$ | 10 | 7, 20, 21, 22 |

Bit – bit ini diindeks dengan LSB sebagai 0 (catatan : LSB biasanya adalah bit – bit yang berada pada tengah kanan suatu urutan bit).

Register – register LFSR tersebut diputar menggunakan generator stop and go, menggunakan aturan mayoritas. Tiap register diasosiasikan dengan bit yang diputar. Pada tiap putaran, pemutaran bit oleh ketiga register LFSR tersebut akan diperiksa, dan bit mayoritas ditentukan. Register diputar apabila bit yang diputar sesuai dengan bit mayoritas. Maka setiap langkah dua atau tiga register diputar, dan tiap register melangkah dengan probabilitas ¾.

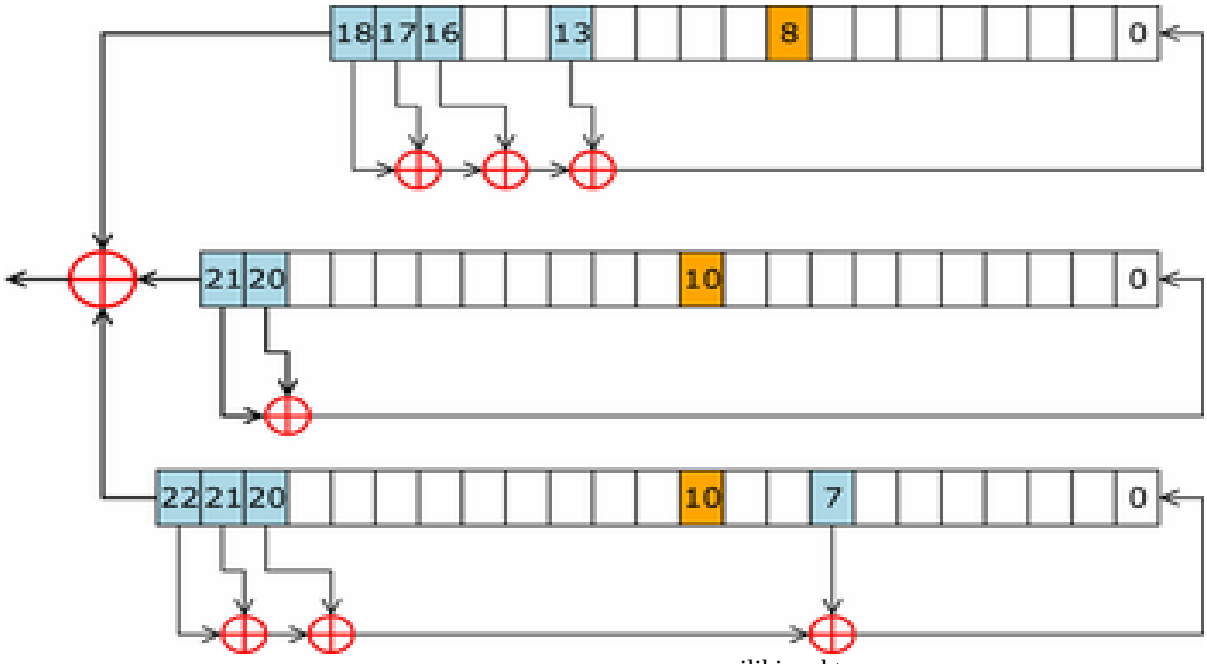
Pada awalnya, register – register diset ke 0, kemudian untuk 64 putaran, kunci rahasia 64 bit diaduk sesuai dengan skema berikut ini

Pada putaran $0 \leq i < 64$, bit ke I ditambahkan ke Least Significant Bit (LSB) tiap register LFSR menggunakan XOR.

Kemudian tiap register akan diputar.

22 bit dari nomor frame ditambahkan ke 22 putaran dengan cara yang sama. Kemudian seluruh system diputar menggunakan mekanisme pemutaran mayoritas normal untuk 100 putaran, dengan keluaran yang dibuang. Setelah hal ini selesai, cipher telah siap untuk memproduksi 228 bit keystream keluaran.

Ilustrasi algoritma A5/1 :




### 6.3 Masalah Keamanan

Algoritma A5/1 ini lemah terhadap beberapa serangan. Beberapa serangan membutuhkan *preprocessing stage* yang cukup mahal, dan setelah itu cipherteks dapat diserang dalam hitungan menit atau detik. Sampai sekarang, kelemahan utama adalah serangan pasif menggunakan *known plaintext attack*.

Pada tahun 2003, kelemahan lain yang lebih serius ditemukan dapat mengeksploitasi serangan cipherteks saja. Atau oleh serangan aktif. Pada tahun 2006, Elad Barkan, Eli Biham dan Nathan Keller mendemonstrasikan serangan terhadap A5/1, A5/3 dan bahkan GPRS yang membolehkan si penyerang menyadap pembicaraan telepon dan mendekripsinya pada saat real time atau pada saat yang lainnya.

#### *6.3.1 Known Plainteks Attack*

Pada tahun 1997, Golic memperlihatkan sebuah serangan yang berbasis himpunan penyelesaian dari persamaan linear yang memiliki kompleksitas waktu $\mathbf{2^{40.16}}$ (unit yang digunakan berdasar pada solusi persamaan linear yang dibutuhkan).

Penyelesaian himpunan dari persamaan linear yang memiliki waktu .

Pada tahun 2000, Alex Biryukov, Adi Shamir dan David Wagner menunjukkan bahwa A5/1 dapat dikriptanalis secara real time menggunakan *time memory tradeoff attack*. Sebuah tradeoff memperbolehkan seorang penyerang untuk merekonstruksi kunci dalam beberapa menit dari 2 detik percakapan (plainteks yang diketahui), namun si penyerang harus melalukan terlebih dahulu preprocessing stage yang membutuhkan data 300 GB dan $2^{48}$ waktu komputas. Beberapa tradeoff dari preprocessing, data requirement, kompleksitas memori dan waktu serangan dimungkinkan.

Pada tahun yang sama, Eli Biham dan Orr Dunkelman juga mempublikasikan serangan pada A5/1 dengan kompleksitas usaha $2^{39.91}$ A5/1 clockings apabila diketahui $2^{20.8}$ bits plainteks. Serangan ini membutuhkan penyimpanan data sebanyak 32 GB setelah $2^{38}$ langkah prekomputasi.

Ekdahl dan Johannson(2003) mempublikasikan sebuah serangan pada prosedur inisialisasi yang dapat memecahkan algoritma A5/1 dalam beberapa menit dengan menggunakan data plainteks percakapan selama dua sampai lima menit. Serangan ini tidak membutuhkan preprocessing stage. Pada tahun 2004, Maximov dkk meningkatkan hasil ini dengan sebuah serangan yang membutuhkan komputasi kurang dari satu menit dan beberapa detik percakapan yang diketahui.

### 6.3.2 Serangan A5/1 pada GSM

Pada tahun 2003, Barkan dkk mempublikasikan beberapa serangan pada enkripsi GSM. Serangan aktif adalah yang pertama. Telepon GSM dapat dipaksa menggunakan algoritma A5/2 yang lebih lemah secara singkat, algoritma A5/2 dapat dipecahkan dengan sangat mudah, dan telepon menggunakan kunci yang sama untuk algoritma A5/1 yang lebih kuat. Serangna kedua pada A5/1 adalah sebuah *cipherteks only time memory tradeoff attack* yang membutuhkan prekomputasi yang intensif.

Pada tahun 2006, Elad Barkan, Eli Biham, Nathan Keller mempublikasikan serangan kepada A5/x cipher, yang diklaim dapat menggunakan metode cipherteks only secara efektif. Cara yang digunakan mirip dengan cara diatas.

### 6.3.4 Analisis Singkat Keamanan GSM

Dengan banyaknya kelemahan pada algoritma A5/1, dan waktu serangan yang sekarang dapat dilakukan dibawah 10 menit, algoritma A5/1 yang terdapat pada GSM ini benar – benar memiliki masalah keamanan yang serius.

Hal ini diperburuk dengan singkatnya juga waktu yang dibutuhkan untuk melakukan serangan *ciphertext only attack* yang dapat dilakukan dibawah 10 menit dengan beberapa detik waktu rekaman percakapan.

## 7. Kesimpulan

Kesimpulan yang dapat diambil dari studi perbandingan algoritma ORYX dengan algoritma A5/1 yang merupakan dapat dianggap sebanding

1. Algoritma pada CDMA, ORYX adalah algoritma yang berbasis cipher aliran. Begitu pula untuk algoritma pada GSM, algoritma A5/1
2. Algoritma – algoritma yang digunakan pada dua standard komunikasi digital saat ini, tingkat keamanannya ternyata masih sangat rendah
3. Tingkat keamanan yang rendah ini dapat dibuktikan dengan mudahnya algoritma ini diserang dengan *ciphertext only attack*. Hal ini menjadi krusial karena ternyata serangan ini dapat dilakukan dengan sederhana pada dua algoritma tersebut
4. Karena populernya protokol GSM dan CDMA pada dunia pertelekomunikasian sekarang ini, ditambah dengan meningkatnya lalulintas komunikasi wireless yang memanfaatkan layanan kedua protokol ini, maka seharusnya tingkat keamanan protokol – protokol ini seharusnya diperbaiki.

## DAFTAR PUSTAKA

1. E. Dawson and L. Nielsen. Automated cryptanalysis of XOR plaintext strings. *Cryptologia*, volume XX Number 2, pages 165{181. April 1996.

2. B. Goldburg, E. Dawson and S. Sridharan. The automated cryptanalysis of analog speech scramblers. *Advances in Cryptology - EUROCRYPT'91*, volume 547 of *Lecture Notes in Computer Science*, pages 422{430. Springer-Verlag, 1991.

3. M. Briceno, I. Goldberg and D. Wagner. GSM cloning. 20 April, 1998. http://www.isaac.cs.berkeley.edu/isaac/gsm.htm

4. J. Dj. Goli_c. Cryptanalysis of alleged A5 stream cipher. *Advances in Cryptology* -

*EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 239{
255. Springer-Verlag, 1997.

5. H. Schildt. *C the Complete Reference* Osborne McGraw-Hill, Berkeley, CA, 1990.

6. TIA TR45.0.A, *Common Cryptographic Algorithms* June 1995, Rev B.

7. D. Wagner, B. Schneier and J. Kelsey. Cryptanalysis of the cellular message encryption algorithm. *Advances in Cryptology - CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 526{537. Springer-Verlag, 1997.

8. D. Wagner, B. Schneier and J.Kelsey. *Cryptanalysis of ORYX*. unpublished manuscript, 4 May 1997.