BSC

Block Slider Cipher

Introduction to a new sliding based block cipher algorithm

William Sentosa (13513026)
Teknik Informatika
Institut Teknologi Bandung
Bandung, Indonesia
williamsentosa.sentosa@gmail.com

Randi Chilyon Alfianto (13513087)
Teknik Informatika
Institut Teknologi Bandung
Bandung, Indonesia
randichil@gmail.com

***Abstract*—Melalui makalah ini, diusulkan sebuah algoritma enkripsi dan dekripsi berbasis *block cipher* baru yang bernama BSC (*Block Slider Cipher*). *Block cipher* ini menggunakan *block* sepanjang 128-bit dan kunci sepanjang 128-bit. Dalam perancangan algoritma BSC ini, digunakan operasi translasi bit, substitusi, dan XOR agar mendapatkan suatu algoritma yang sukar dipecahkan. Selain itu, algoritma ini juga memanfaatkan algoritma *Vigenere Cipher* pada kunci yang digunakan untuk menambahkan kerumitan dari algoritma ini.**

Kata kunci—block cipher;BSC; kunci internal; kunci eksternal; enkripsi; dekripsi

I. PENDAHULUAN

Pada zaman sekarang, pertukaran informasi dapat dilakukan dengan mudah berkat adanya internet. Informasi yang ditransfer melalui internet tersebut sangat beragam mulai dari informasi percakapan biasa hingga informasi yang bersifat rahasia. Informasi yang bersifat rahasia tersebut tentunya harus dijaga dan dipastikan keamanannya agar tidak digunakan oleh orang yang tidak berwenang. Oleh karena hal itu, diperlukan kriptografi untuk menjamin keamanan dari informasi tersebut.

Penggunaan kriptografi sudah sangat berkembang sekarang ini. Baik tidaknya suatu algoritma kriptografi ditentukan dari seberapa rumit suatu *plain text* dienkripsi menjadi *cipher text* dan seberapa sulit suatu *cipher text* dipecahkan menjadi *plain text*. Penggunaan algoritma kriptografi konvensional sudah sangat mudah dipecahkan dengan berbagai metode kriptanalisis. Oleh karena itu, diperlukan suatu algoritma kriptografi yang dapat memastikan keamanan dari informasi.

Block cipher merupakan algoritma kriptografi simetrik yang mengenkripsi *block plaintext* dengan jumlah bit tertentu dan menghasilkan *block ciphertext* dengan jumlah bit yang sama. Dalam proses enkripsinya, *block cipher* menggunakan beberapa fungsi matematika, diantaranya fungsi permutasi, fungsi substitusi, sehingga *confusion* dan *diffusion* terpenuhi.

Untuk mengatasi masalah keamanan tersebut, kami akan membuat suatu algoritma *block cipher* yang memiliki tingkat kerumitan yang kompleks sehingga sukar untuk dipecahkan. Algoritma yang kami buat bernama BSC (*Block Slider Cipher*). Algoritma ini akan menggunakan *block* sepanjang 128-bit dan menggunakan *block* kunci sepanjang 128-bit.

II. DASAR TEORI

A. Block Cipher

Dalam kriptografi, *block cipher* merupakan algoritma deterministik yang beroperasi pada sekelompok bit yang panjangnya telah dipastikan, disebut *blocks*, dengan transformasi yang tidak berubah yang dispesifikasikan dengan sebuah kunci simetri.*Block ciphers* adalah komponen yang penting dalam desain protokol kriptografi, dan secara luas digunakan untuk implementasi enkripsi dari data yang besar.

Algoritma *block cipher* mencakup 2 algoritma, yaitu enkripsi (E) dan dekripsi (D). Kedua algoritma ini menerima dua input: sebuah *block* dengan ukuran n-bit dan kunci dengan ukuran k-bit, dan keduanya menghasilkan output berupa *block* n-bit. Algoritma dekripsi D didefinisikan sebagai invers dari fungsi enkripsi, $D = E^{-1}$. Secara formal, *block cipher* dispesifikasikan dengan sebuah algoritma enkripsi:

$$E_K(P) := E(K,P) : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n,$$

yang menerima input kunci dengan panjang k, disebut *key size*, dan sebuah bit string P dengan panjang n, disebut panjang *block*, dan mengembalikan string C dengan panjang n-bit. P disebut *plaintext*, dan C disebut *ciphertext*. Untuk setiap K, fungsi $E_k(P)$ dibutuhkan sebagai pemetaan yang dapat diinvers pada $\{0,1\}^n$. Invers dari E didefinisikan sebagai fungsi:

$$E_K^{-1}(C) := D_K(C) = D(K,C) : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n,$$

Menggunakan kunci K dan *ciphertext* C untuk mengembalikan nilai *plaintext* P, seperti:

$$\forall K : D_K(E_K(P)) = P.$$

Terdapat 4 mode operasi *cipher block*:

1. *Electronic Code Book* (ECB)

   Setiap *block plaintext* P_i dienkripsi secara individual dan independen menjadi *block ciphertext* C_i.

Enkripsi: $C_i = E_k(P_i)$

Dekripsi: $P_i = D_k(C_i)$

2. *Cipher Block Chaining* (CBC)

   Setiap *ciphertext* bergantung tidak hanya pada *block plaintext*-nya tetapi juga pada seluruh *block plaintext* sebelumnya. Hasil enkripsi *block* sebelumnya di-umpan-balikkan ke dalam *block* enkripsi yang sekarang.

   Enkripsi *block* pertama memerlukan *block* semu (C_0) yang disebut IV (*initialization vector*). IV dapat diberikan oleh pengguna atau dibangkitkan secara acak oleh program. Pada dekripsi, *block plaintext* diperoleh dengan cara meng-XOR-kan IV dengan hasil dekripsi terhadap *block ciphertext* pertama.

3. *Cipher Feedback* (CFB)

   CFB mengatasi kelemahan pada mode CBC jika diterapkan pada komunikasi data (ukuran *block* yang belum lengkap). Data dienkripsikan dalam unit yang lebih kecil daripada ukuran *block*. Unit yang dienkripsikan dapat berupa bit per bit, 2-bit, 3-bit, dan seterusnya. Bila unit yang dienkripsikan satu karakter setiap kalinya, maka mode CFB-nya disebut CFB 8-bit. CFB n-bit mengenkripsi plainteks sebanyak n-bit setiap kalinya, n<=m (m = ukuran *block*). Dengan kata lain, CFB mengenkripsi *cipher block* seperti pada *stream cipher*. Mode CFB membutuhkan sebuah antrian yang berukuran sama dengan ukuran *block* masukan.

4. *Output Feedback* (OFB)

   Mode OFB mirip dengan mode CFB, kecuali n-bit hasil dari enkripsi terhadap antrian disalin menjadi elemen posisi paling kanan pada antrian. Dekripsi dilakukan sebagai kebalikan dari enkripsi.

B. *Shannon's Confusion and Diffusion*

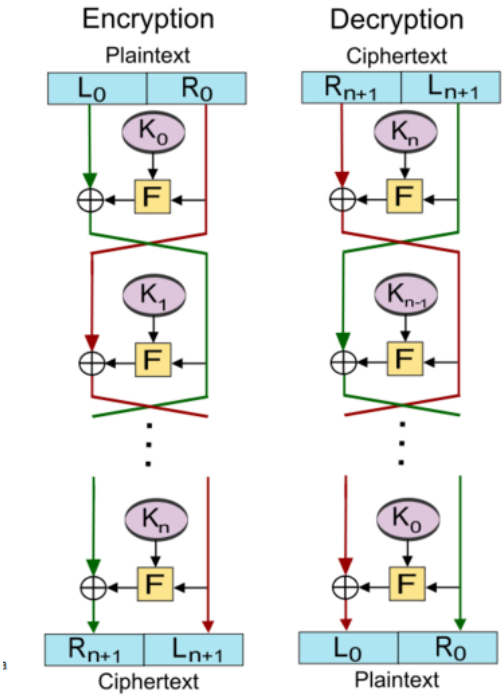
Dalam kriptografi, *confusion* dan *diffusion* merupakan dua sifat dari operasi dalam *cipher* yang aman yang diidentifikasikan oleh Claude Shannon pada 1945 dalam laporannya "A Mathematical Theory of Cryptography".

Confusion merupakan prinsip untuk menyembunyikan hubungan antara *plaintext, ciphertext,* dank unci sehingga informasi statistik sulit untuk dianalisis. *Confusion* berarti setiap bit dari *ciphertext* harus berdasarkan pada beberapa bagian dari kunci.

Diffusion merupakan prinsip untuk menyebarkan pengaruh perubahan 1 bit dalam *plaintext* atau kunci ke sebanyak mungkin *ciphertext*. *Diffusion* berarti apabila kita mengganti sebuah bit pada *plaintext*, maka satu bit dari 2 bit pada *ciphertext* harus berubah, atau sebaliknya, apabila kita mengubah 1 bit dari *ciphertext*, maka setengah dari 1 bit *plaintext* harus berubah.

C. *Jaringan Feistel*

Jaringan feistel merupakan struktur simetris yang digunakan dalam konstruksi dari *block cipher*, yang dinamakan dari fisikawan Jerman dan kriptografer Horst Feistel menjadi pelopor riset ketika bekerja untuk IBM(USA). Sebuah proporsi besar dari *block cipher* menggunakan skema, termasuk Data Encryption Standard (DES). Struktur feistel memiliki keuntungan di mana operasi enkripsi dan dekripsi sangat mirip, bahkan identik dalam beberapa kasus, dan hanya membutuhkan kunci yang dibalik. Karena itu, ukuran dari kode atau sirkuit yang perlu diimplementasi seperti *cipher* hampir setengahnya.



Gambar 1. Jaringan Feistel (sumber: https://upload.wikimedia.org/wikipedia/commons/f/fa/Feistel_cipher_diagram_en.svg)

III. RANCANGAN *BLOCK CIPHER*

Block yang akan kami gunakan adalah sepanjang 128-bit. Panjang kunci eksternal yang kami digunakan adalah 16 karakter (128-bit). Panjang plaintext akan dibagi menjadi block-block yang terdiri dari 128-bit. Banyaknya putaran yang dilakukan pada Jaringan Feistel adalah 8 putaran.

Plain text akan kami bagi menjadi blok-blok berukuran 128 bit. Setelah itu, kami akan melakukan proses enkripsi. String *ciphertext* akan di-*encode* dengan menggunakan *base64 encoder* sebelum diproses. Kunci masukan pengguna juga kami enkripsi terlebih dahulu sebelum kami gunakan. Algoritma untuk mengenkripsi kunci adalah *vigenere chiper* 26 bit. Setelah kunci terenksripsi, kunci tersebut akan berubah menjadi **kunci internal**.

A. Pembangkitan Kunci Internal

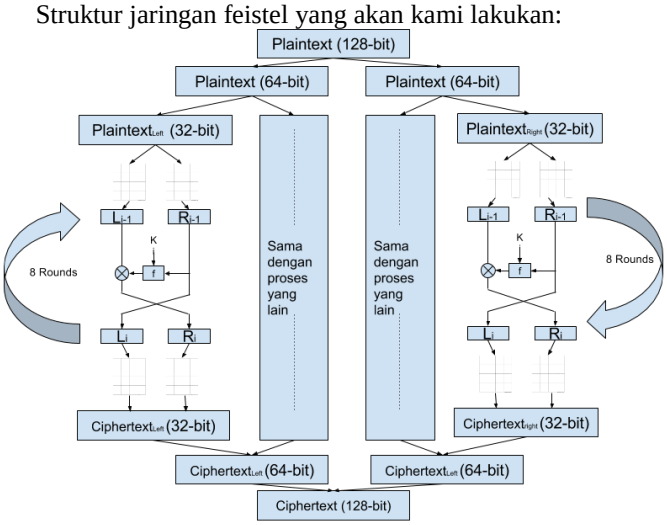
Berikut merupakan cara untuk membangkitkan kunci internal pada algoritma BSC:

1. Ubah bentuk kunci masukan pengguna (dalam representasi karakter) menjadi bit.
   a. Bila kunci eksternal itu kurang dari 128 bit, lakukan duplikasi pada kunci sehingga panjang kunci menjadi 128 bit.
   b. Bila kunci eksternal itu lebih dari 128 bit, potong kunci tersebut menjadi 128 bit.
2. Lakukan *looping* pada bit kunci tersebut tersebut dan untuk setiap bit yang di-mod dengan 4 = 0, ambil bit tersebut dan simpan di suatu array.
3. Dari hasil di atas, maka akan didapatkan 32-bit dan satukan menjadi 4-byte.
4. Lakukan vigenere cipher pada kunci eksternal (dalam karakter) dan gunakan kunci 4 byte yang didapatkan dari proses nomor 3.
   Contoh :
   Kunci : 101**0**101**1**000**1**000**0**101**0**101**0**…….101**0**
   Kunci vigenere : 01100011...0
   Kunci vigenere dalam byte : 99 20 30 123
5. Hasil dari enkripsi kunci eksternal tersebut akan digunakan sebagai kunci internal.
6. Dalam hal ini, kunci internal terdiri dari 128-bit.
7. Bagi kunci internal menjadi block-block yang terdiri dari 16-bit (akan didapatkan 8 block).
8. Block kunci di atas akan digunakan sebagai kunci pada jaringan feistel.

B. Jaringan Feistel

Blok-blok *plaintext* berukuran 128-bit tersebut akan diproses melalui jaringan feistel. Pada jaringan feistel ini, blok akan kita pecah menjadi blok lebih kecil yang berukuran 16-bit. Setelah itu, dengan menggunakan kunci yang telah dibangkitkan, kita bisa memproses blok-blok tersebut.

Struktur jaringan feistel yang akan kami lakukan:



Berikut merupakan teknik pemrosesan blok-blok *plaintext*:

1. Blok plaintext akan dipecah menjadi blok 16 bit sebanyak 8 blok. Blok tersebut dipasangkan dua-dua dengan blok di sebelahnya.
2. Setelah itu, blok plaintext akan melalui proses jaringan feitsel yang diputar selama 8 kali dengan **fungsi f**.
3. Kunci yang digunakan pada saat pemrosesan adalah **kunci internal** yang panjangnya 128 bit. Kunci tersebut akan dibagi menjadi sub key yang berukuran 16 bit. Untuk setiap pemutaran, sub key akan digunakan, sehingga untuk 8 kali putaran, diperlukan 8 sub key.
4. Setelah seluruh blok plaintext dienkripsi, blok digabung menjadi *cipher text* yang utuh.

C. Fungsi f

Berikut merupakan algoritma dari fungsi f yang kami lakukan:

1. Proses baris pertama pada blok plaintext, lihatlah bit pertama dari kunci.
   a. Apabila bit kunci tersebut **bernilai nol**, geser **baris tersebut ke kiri** dan untuk setiap elemen dalam baris tersebut, lakukan **operasi xor dengan bit kunci** tersebut.
   b. Apabila bit kunci tersebut **bernilai satu**, **geser baris tersebut ke kanan** dan untuk setiap elemen dalam baris tersebut, lakukan **operasi xor dengan bit kunci** tersebut.

   Contoh : baris pertama dari blok 4x4 adalah 1001, sedangkan kuncinya adalah 1011111001101110. Karena bit pertamanya 1 maka lakukan pergeseran blok ke kanan sehingga baris tersebut berubah menjadi 0011. Setelah itu lakukan xor seluruh elemen tersebut dengan 1 sehingga hasilnya adalah 1100.
2. Proses baris selanjutnya pada blok plaintext. Kunci yang dilihat adalah kunci ke-n sesuai dengan iterasi proses. Langkah-langkah proses sama seperti langkah
   Contoh : baris ke dua diproses dengan menggunakan bit kedua dari kunci, baris ke tiga diproses dengan menggunakan bit ketiga dari kunci, dst.
3. Bila seluruh baris telah diproses, proses lah kolom pertama pada blok tersebut. Bit kunci yang digunakan adalah bit kunci terkecil yang belum pernah digunakan, yaitu bit ke lima.
   a. Apabila bit kunci tersebut **bernilai nol**, geser **kolom tersebut ke atas** dan untuk setiap elemen dalam baris tersebut, lakukan **operasi xor dengan bit kunci** tersebut.
   b. Apabila bit kunci tersebut **bernilai satu**, **geser baris tersebut ke bawah** dan untuk setiap elemen dalam baris tersebut, lakukan **operasi xor dengan bit kunci** tersebut.

   Contoh :
   Blok text :
   1011

0100
0011
1110
Kunci :1011111001101110
Bit kunci ke-5 adalah 1 sehingga kolom pertama digeser ke atas. Hasilnya adalah
0011
0100
1011
1110
Elemen kolom tersebut akan di xor dengan 1 sehingga hasil akhir pemrosesan ini adalah
1011
1100
0011
0110

4. Ulangi proses tersebut sampai semua kolom terproses. Bit kunci yang digunakan adalah bit ke enam sampai ke delapan.
5. Setelah itu proses ulang blok tersebut dengan sisa kunci yang ada. Ulangi langkah-langkah diatas dimulai dari pemrosesan baris dan kolom sehingga masing2 baris dan kolom diproses sebanyak dua kali.

IV. Eksperimen dan Pembahasan Hasil

Kami melakukan percobaan algoritma BSC dengan berbagai metode pengoperasian. Percobaan dilakukan dengan mengoperasikan alogritma BSC dengan berbagai metode pengoperasian. Untuk setiap percobaan, dicatat waktu pengoperasiannya dan hasil enkripsi teks.

Berikut merupakan hasil pengorperasian BSC oleh mode ECB, CBC, CFB.

1. Mode Pengoperasian ECB

Text	william sentosa dan randi chilyon alfianto
Key	Ifitb
Encripted Text	hjgmle+mtuSot+85ptWn/3em/S4qhKJT+W/bpa2JKekArQ==
Durasi	3 ms

2. Mode Pengoperasian CBC

Text	william sentosa dan randi chilyon alfianto
Key	Ifitb
Encripted Text	hjhliVW2GuhN98C5Ll2nDydWpSYzkhUbWG5FZHYHUsLPzQ==
Durasi	4 ms

3. Mode Pengoperasian CFB

Text	william sentosa dan randi chilyon alfianto
Key	Ifitb
Encripted Text	winjsU0Dvf4WRdP2jNWKjoL3dt9eUwDzY/Zc4p70R+0QHQ==
Durasi	16 ms

V. Analisis Keamanan

Tingkat keamanan dari suatu algoritma *block cipher* dapat ditinjau berdasarkan kompleksitas dari algoritma itu sendiri. Hal tersebut dapat dilihat dari metode dari algoritma tersebut. Algoritma ini mengacu pada prinsip *confusion* dan *diffusion* dari Shannon. Selain itu seluruh metode pengoperasian dilakukan dalam operasi bit sehingga penerkaan kunci akan sulit untuk dilakukan. Algoritma ini juga menggunakan kombinasi antara jaringan feistel dan cipher berulang sehingga akan memberikan efek *diffusion* dan menambah kerumitan dari alogiritma ini.

Selain dari algoritma pengoperasian, panjang kunci juga akan mempengaruhi tingkat kekuatan dari algoritma BSC, terutama untuk menangani serangan dengan *exhaustive search*. Semakin panjang kunci, akan dibutuhkan waktu yang lebih lama untuk memecahkan algoritma tersebut. Karena BSC menggunakan kunci sepanjang 128-bit, memecahkan algoritma BSC dengan metode *exhaustive search* akan membutuhkan percobaan sebanyak 2^{128} kali.

Dengan asumsi bahwa dalam satu detik, jumlah percobaan yang dapat dilakukan adalah 10^6, maka mencoba semua kombinasi yang ada diperlukan waktu 5.4×10^{24} tahun. Ini membuktikan bahwa algoritma BSC cukup tahan dari serangan *exhaustive search*.

VI. Kesimpulan dan Saran

Algoritma *Block Slider Cipher* (BSC) dapat melakukan enkripsi terhadap suatu informasi yang diberikan serta dapat mengembalikan informasi tersebut ke bentuk semula dengan melakukan dekripsi. Algoritma ini dapat mengenksripsi suatu informasi bila informasi tersebut dapat diubah dalam bentuk byte. Algoritma ini juga menerapkan prinsip *confusion* dan *diffusion* dari Shannon dan juga menerapkan jaringan feistel. Dengan menggunakan prinsip tersebut, algoritma ini menjadi semakin rumit dan sukar untuk dipecahkan.

Acknowledgment

Pertama-tama kami ingin berterima kasih kepada Tuhan karena melalui anugrah-Nya saya bisa menyelesaikan makalah ini. Saya juga sangat berterima kasih kepada dosen saya Dr. Ir. Rinaldi Munir, karena telah memberikan inspirasi bagi saya. Saya juga berterima kasih kepada Institut Teknologi Bandung, tempat dimana saya menuntut ilmu ketika saya menyelesaikan makalah ini.

REFERENSI

[1] Cusick, Thomas W. & Stanica, Pantelimon (2009). *Cryptographic Boolean functions and applications.* Academic Press. pp. 158–159. ISBN 9780123748904.

[2] Menezes, Alfred J.; van Oorschot, Paul C.; Vanstone, Scott A. (1996). "Chapter 7: Block Ciphers". *Handbook of Applied Cryptography*. CRC Press. ISBN 0-8493-8523-7.

[3] *Bellare, Mihir; Rogaway, Phillip (11 May 2005), Introduction to Modern Cryptography(Lecture notes)*, chapter 3.

[4] Stallings, William (2014). *Cryptography and Network Security* (6th ed.). Upper Saddle River, N.J.: Prentic Hall. pp. 67–68. ISBN 978-0133354690.

[5] http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2015-2016/Metode-BPCS.pdf, diakses pada 23 Maret 2016.

[6] Cryptography Engineering: Design Principles and Practical Applications. Ferguson, N., Schneier, B. and Kohno, T. Indianapolis: Wiley Publishing, Inc. 2010. pp. 63, 64. ISBN 978-0-470-47424-2.