Modifikasi Algoritma Caesar Cipher Menjadi SPICA-XB (Spinning Caesar dengan XOR Binary)

Rizal Panji Islami (13510066)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
rizalpanjiislami@gmail.com

***Abstrak*—Terdapat berbagai macam algoritma kriptografi yang dikenal saat ini. Diantaranya adalah algoritma Caesar Cipher. Caesar Cipher merupakan algoritma kriptografi klasik yang cukup populer di masa lalu. Algoritma ini memang mudah untuk digunakan karena tidak membutuhkan komputasi yang kompleks, namun sayangnya terdapat berbagai kelemahan dari algoritma ini. Salah satunya adalah algoritma ini akan mengenkripsi huruf plain teks yang sama dengan cipher teks yang sama. Misalkan saja jika kuncinya adalah A=Z, maka seluruh huruf A akan dienkripsi menjadi huruf Z. Hal ini menyebabkan algoritma ini dapat dengan mudah dipecahkan menggunakan analisis frekuensi.**

Algoritma kriptografi yang dianggap mangkus saat ini adalah algoritma-algoritma kriptografi yang biasanya disebut algoritma kriptografi moderen. Dalam makalah kali ini akan dilakukan penggabungan sifat antara algoritma Caesar Cipher dengan kriptografi moderen. Algoritma ini disebut algoritma Spica-XB (Spinning Caesar dengan XOR Binary). Diharapkan algoritma ini dapat memperbaiki berbagai kelemahan yang dimiliki oleh algoritma Caesar Cipher klasik.

***Kata Kunci*— Kriptografi, Caesar Cipher, Analisis Frekuensi, Algoritma**

I. PENDAHULUAN

Tujuan utama dari proses enkripsi dalam kriptografi adalah membuat pesan-pesan tertentu menjadi tidak dapat dibaca atau dikenali oleh pihak yang tidak berkepentingan. Walaupun tidak menjamin 100%, namun setidaknya proses enkripsi akan mempersulit pihak yang tidak berkepentingan untuk membaca data tersebut. Proses enkripsi ini dapat dilakukan dengan menggunakan berbagai algoritma kriptografi yang ada.

Terdapat berbagai macam algoritma kriptografi yang dikenal saat ini. Secara umum algoritma tersebut dapat dikategorikan kedalam dua kelompok, yaitu algoritma kriptografi klasik dan algoritma kriptografi moderen. Sayangnya, seluruh algoritma kriptografi klasik saat ini sudah tidak lagi dianggap aman. Hal ini dikarenakan sudah ditemukannya berbagai metode yang dapat dilakukan untuk memecahkan algoritma-algoritma kriptografi klasik.

Salah satu algoritma kriptografi klasik yang cukup populer adalah algoritma Caesar Cipher. Algoritma ini cukup mudah untuk digunakan, karena proses enkripsi dapat dilakukan hanya dengan membandingkan karakter yang akan dienkripsi dengan kuncinya. Misalkan saja, dalam proses enkripsi sebagai berikut:

Plain Teks : SELAMAT PAGI
Kunci : A=J

Berdasarkan dari kunci tersebut, dibentuk tabel sebagai panduan enkripsi sebagai berikut:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I

Tabel 1
Tabel Enkripsi Caesar Cipher dengan Kunci A=J

Dengan menggunakan panduan tabel diatas, plain teks pada contoh ini akan menghasilkan Cipher Teks seperti berikut:

Cipher Teks : BNUJVJC YJPS

Seperti yang dapat dilihat, algoritma Caesar Cipher ini memang mudah untuk digunakan, namun sebagai implikasi dari kemudahan tersebut, algoritma ini tidaklah cukup kuat dan aman. Kelemahan terbesar dari algoritma Caesar Cipher adalah huruf yang sama selalu dienkripsi menjadi cipher yang sama.

Seperti pada contoh diatas, cipher teks yang dihasilkan untuk kalimat SELAMAT PAGI adalah BNUJVJC YJPS. Dapat dilihat bahwa terdapat tiga buah huruf J dalam cipher teks tersebut. Huruf J ini merupakan cipher dari huruf A. Hal ini dapat dikatakan sebagai *back door* bagi algoritma ini, karena dimungkinkannya dilakukan proses kriptanalisis dengan menggunakan teknik Analisis Frekuensi.

Berangkat dari berbagai kelemahan yang terdapat dalam algoritma kriptografi klasik, saat ini sudah dikenal berbagai algoritma kriptografi baru yang biasanya disebut sebagai algoritma kriptografi moderen. Salah satu ciri

utama dari algoritma kriptografi moderen adalah proses enkripsi yang memanfaatkan sifat binary (pemrosesan dengan menggunakan bit-bit data). Hal ini dapat meningkatkan keamanan karena setidaknya hal tersebut akan menyebabkan cipher teks yang dihasilkan tidak lagi bersifat *human readable*. Selain itu algoritma kriptografi moderen juga mengenal berbagai standar lain seperti metode operasi dengan menggunakan mode operasi *Cipher Block* juga proses enkripsi yang memanfaatkan sifat operasi XOR.

Berangkat dari kedua fakta diatas (permasalahan dalam algoritma Caesar Cipher dan sifat-sifat algoritma kriptografi moderen), penulis tertarik untuk mengusulkan sebuah algoritma kriptografi baru, yang merupakan penggabungan dari algoritma Caesar Cipher dengan beberapa sifat dari algoritma kriptografi moderen. Algoritma tersebut penulis sebut dengan nama algoritma SPICA-XB (Spinning Caesar dengan XOR Binary).

II. USULAN ALGORITMA

Algoritma SPICA-XB adalah algoritma yang menggabungkan sifat dari algoritma Caesar Cipher dengan beberapa sifat dari algoritma kriptografi moderen. Idenya adalah dengan mengasumsikan algoritma Caesar Cipher layaknya memutar sebuah roda (*spinning*).

Gambar 1
Analogi Algoritma SPICA-XB
Sumber : Slide Kuliah IF3058 – Kriptografi Klasik

Berbeda dengan algoritmaCaesar Cipher biasa, algoritma SPICA-XB bekerja dengan melakukan proses substitusi dengan kunci yang berputar. Perputaran tersebut akan sangat bergantung dari kunci dasar yang ditentukan oleh pembuat pesan sebelumnya. Selain itu, jika dalam algoritma Caesar Cipher biasa proses substitusi akan dilakukan begitu saja (misalkan mengganti A dengan J), dalam algoritma SPICA-XB ini proses enkripsi akan dilakukan dengan terlebih dahulu merubah huruf-huruf yang ada kedalam bentuk binary ASCII 8 bit. Proses enkripsipun dilakukan dengan melakukan operasi penjumlahan lalu XOR antara plain teks dengan kunci.

Berikut rincian algoritma SPICA-XB secara umum:

1. Pembuat pesan menentukan pesan serta kunci yang akan digunakan, panjang kunci adalah bebas namun merupakan kombinasi huruf-huruf dalam alfabet. Kunci ini tidaklah bersifat *case sensitive*.
2. Selanjutnya setiap huruf dalam plain teks akan dipasangkan dengan huruf kunci pasangannya. Jika panjang kunci < panjang plain teks, akan dilakukan proses pengulangan kunci.
3. Setiap huruf dalam plain teks akan dirubah ke dalam bentuk *binary* ASCII berukuran 8 bit.
4. Sebagai persiapan dalam tahapan enkripsi awal, seluruh karakter dalam kunci akan dirubah menjadi nilai angka yang berhubungan. Misalkan A akan diberi nilai 0, B diberi nilai 1, dan seterusnya. Hal ini akan sangat berpengaruh dalam proses *spinning* selama proses enkripsi.
5. Dalam tahapan proses enkripsi awal, roda enkripsi akan dimulai dengan kondisi A=A. Selanjutnya proses enkripsi akan dilakukan karakter per karakter dengan sebelumnya dilakukan perputaran roda senilai dengan nilai kunci. Perputaran roda dilakukan berlawanan dengan jarum jam. Misalkan jika kunci yang saat itu aktif adalah huruf C, maka roda akan diputar sejauh tiga karakter berlawanan dengan arah jarum jam. Pasca perputaran, huruf plain teks akan memiliki pasangan kunci baru (selanjutnya akan disebut *middle key*). Karakter plain teks yang saat itu akan dienkripsi akan dilakukan operasi penjumlahan dengan *middle key* dalam bentuk *binary* ASCII 8 bit. Proses ini akan terus diulang hingga seluruh karakter terenkripsi.
6. Setelah tahapan enkripsi awal dilakukan, selanjutnya dilakukan proses enkripsi kedua, yaitu operasi XOR antara hasil enkripsi pertama dengan *binary* ASCII dari kunci awal.

Untuk memudahkan penjelasan, berikut contoh proses enkripsi yang akan dilakukan dengan kalimat “SELAMAT PAGI”:

Tahap 1 Input Plain Teks dan Kunci:
Plain teks : SELAMAT PAGI
Kunci : BERMAIN

Tahap 2 Memasangkan Setiap Karakter Plain Teks dengan Kunci yang Berhubungan:

S	E	L	A	M	A	T		P	A	G	I
B	E	R	M	A	I	N		B	E	R	M

Tahap 3 Merubah Plain Teks dalam Binary ASCII:
Plain Teks:
01010011 01000101 01001100 01000001 01001101 01000001 01010100
01010000 01000001 01000111 01001001

Tahap 4 Menentukan Nilai dari Kunci

B	E	R	M	A	I	N
1	4	16	12	0	8	13

Tahap 5 Proses Enkripsi Tahap 1
Roda Awal:

A	B	C	D	E	F	G	H	I	J	K	L	M	N
A	B	C	D	E	F	G	H	I	J	K	L	M	N

O	P	Q	R	S	T	U	V	W	X	Y	Z
O	P	Q	R	S	T	U	V	W	X	Y	Z

Untuk proses enkripsi huruf S, kunci yang berpasangan adalah huruf B. Huruf B ini memiliki nilai 1, karena itu roda akan berputar sejauh 1 karakter berlawanan dengan jarum jam (dalam bentuk tabel ini berarti pergeseran ke arah kiri). Sehingga roda akan menjadi seperti berikut: (roda ini selanjutnya akan disebut roda temp1)

A	B	C	D	E	F	G	H	I	J	K	L	M	N
B	C	D	E	F	G	H	I	J	K	L	M	N	O

O	P	Q	R	S	T	U	V	W	X	Y	Z
P	Q	R	S	T	U	V	W	X	Y	Z	A

Dengan berdasarkan pada tabel diatas, maka dalam enkripsi tahap 1 dilakukan proses enkripsi antara huruf S dengan huruf T. Proses yang dilakukan adalah seperti berikut:

a. Merubah huruf S dan huruf T yang merupakan *middle key* kedalam bentuk *binary* ASCII, sehingga memberi hasil:

   01010011 dan 01010100

b. Melakukan operasi penjumlahan antara kedua nilai ASCII tersebut, sehingga diperoleh:

   10100111

   Nilai ini merupakan nilai enkripsi pertama untuk huruf S dari kata SELAMAT.

Proses selanjutnya adalah melakukan proses enkripsi untuk huruf E. Seperti yang sudah diperoleh sebelumnya, huruf E ini akan berpasangan dengan huruf kunci E juga (dari kata BERMAIN). Karena huruf E memiliki nilai empat, maka roda akan berputar sejauh 4 karakter berlawanan dengan arah jarum jam. Perputaran dilakukan berdasarkan pada roda pada kondisi temp1, sehingga diperoleh hasil sebagai berikut:

A	B	C	D	E	F	G	H	I	J	K	L	M	N
F	G	H	I	J	K	L	M	N	O	P	Q	R	S

O	P	Q	R	S	T	U	V	W	X	Y	Z
T	U	V	W	X	Y	Z	A	B	C	D	E

Dapat dilihat pada "roda" diatas, bahwa dalam kondisi kali ini huruf E berpasangan dengan huruf J, sehingga proses enkripsi yang terjadi adalah:

01000101 + 01001010 = 10001111

Proses ini akan terus diulang untuk seluruh karakter yang terdapat di dalam plain teks (kecuali spasi).

Tahap 6 Proses Enkripsi Tahap 2

Proses enkripsi tahap kedua yang dilakukan adalah dengan melakukan proses XOR antara hasil enkripsi pada enkripsi pertama, dengan kunci awal dalam bentuk *binary* ASCII. Misalkan untuk enkripsi kata "SELAMAT PAGI" dengan menggunakan kunci "BERMAIN" akan dilakukan seperti berikut:

a. Dalam proses enkripsi yang sebelumnya telah diperoleh hasil (sebagai contoh proses enkripsi hanya akan dilakukan untuk huruf S dan E saja demi kesederhanaan) seperti berikut:

   10100111 10001111

b. Karena kunci yang digunakan adalah kata "BERMAIN" dan dalam tahap sebelumnya sudah pernah ditentukan bahwa huruf S dan E berpadanan dengan huruf B dan E, maka hasil enkripsi diatas akan dilakukan opersi XOR dengan huruf B dan E sehingga diperoleh hasil sebagai berikut:

   10100111 10001111
   XOR
   01000010 01000101
   =
   11100101 11001010

Sehingga secara keseluruhan, kalimat SELAMAT PAGI dengan kunci BERMAIN menggunakan algoritma SPICA-XB, maka akan diperoleh hasil seperti berikut:

1110010111001010101001010011000101111100
0000000010000000001001010011000101111100
0

Ket. Hasil enkripsi diatas sudah dihilangkan seluruh spasinya.

Proses dekripsi bagi algoritma SPICA-XB dapat dilakukan dengan membalik proses diatas.

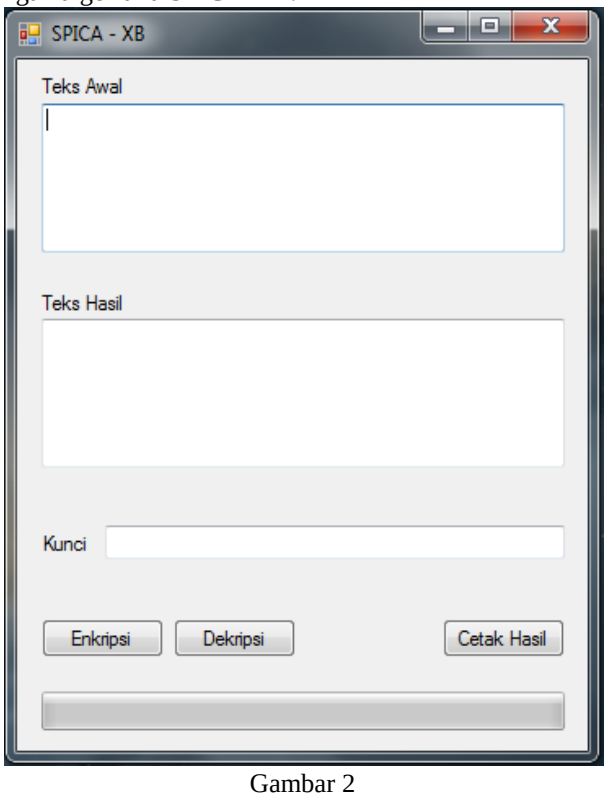
III. Implementasi Algoritma Pada Program

Untuk menguji keberhasilan algoritma, dalam makalah kali ini penulis mencoba untuk mengimplementasikan algoritma SPICA-XB kedalam sebuah program untuk melakukan enkripsi teks. Dalam implementasi kali ini digunakan bahasa pemrograman C#. Secara umum algoritma enkripsi untuk program ini adalah seperti berikut:

```
Procedure Enkripsi()
{
        Daftarkata : array of char;
        Kunci : array of char;
        Padanankata : array of char 2 dimensi;

        While (i=1;i<=daftarkata.length();i++)
        {
```

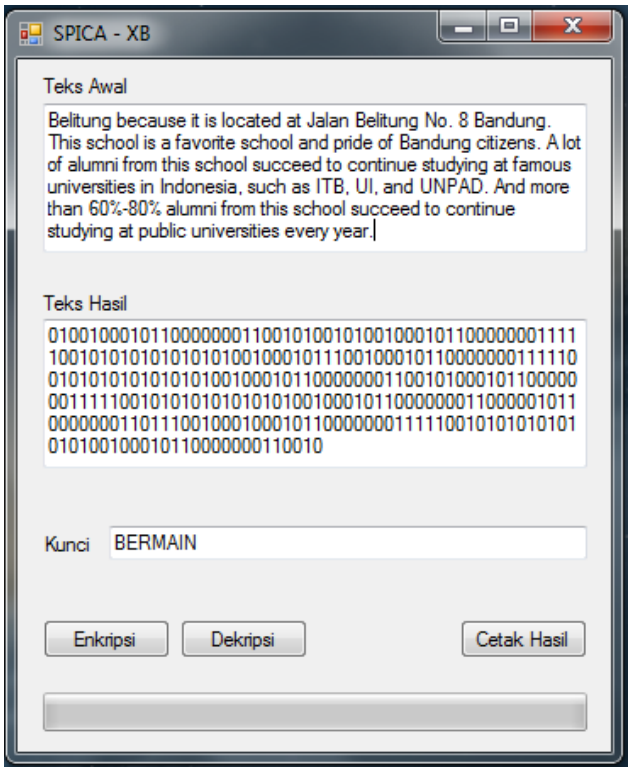
```
                PadanankanKata(Daftarkata,Kunci);
                //Procedure untuk memadankan kata
                dengan kunci dan menyimpannya dalam
                struktur data Padanankata
        }

        While (i=1;i<=daftarkata.length();i++)
        {
                PutarRoda(kunci);
                //Memutar roda sejauh kunci
                OperasiJumlah();
                //Melakukan operasi penjumlahan
        }

        While (i=1;i<=daftarkata.lenth();i++)
        {
                OperasiXOR();
                //Melakukan operasi XOR sebagai
                langkah enkrpsi akhir
        }
}
```

Berikut adalah *screenshot* hasil implementasi program dengan algoritma SPICA-XB:



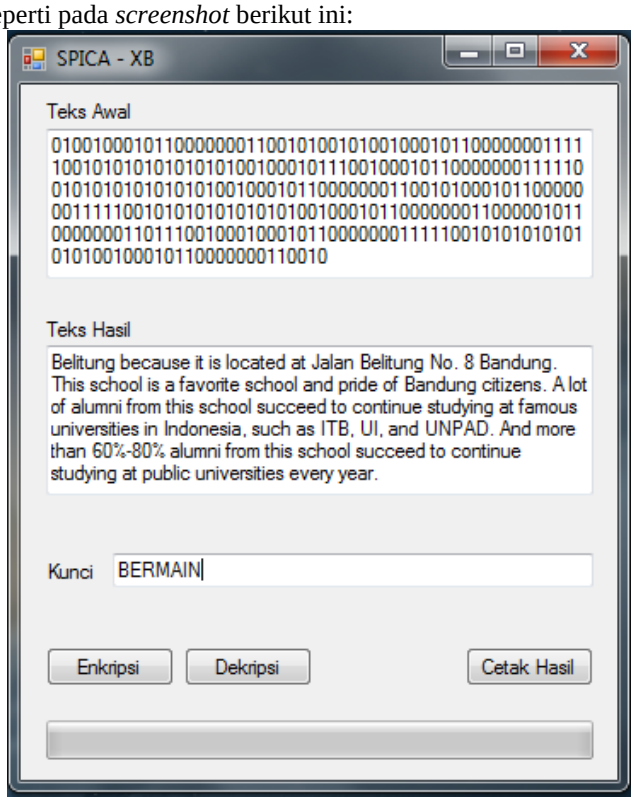
Gambar 2
Tampilan Program

Untuk menguji coba program kali ini dilakukan input sebuah teks lalu dicoba untuk dilakukan proses enkripsi, sehingga diperoleh hasil seperti berikut:



Gambar 3
Hasil Enkripsi Program

Proses dekripsipun berhasil dilakukan dengan baik, seperti pada *screenshot* berikut ini:



Gambar 4
Hasil Dekripsi Program

Berdasarkan pada hasil uji coba diatas, telah terbukti

bahwa algoritma SPICA-XB dapat digunakan sebagai algoritma kriptografi alternatif.

IV. UJI COBA KEAMANAN

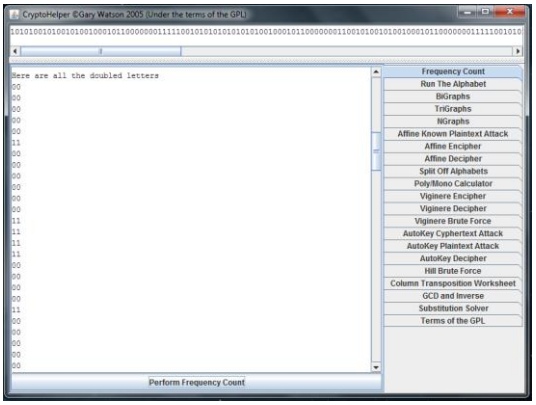
Berangkat dari permasalahan yang penulis angkat mengenai algoritma Caesar Cipher Klasik, pasca implementasi program penulis mencoba untuk melakukan beberapa uji serta kriptanalisis untuk algoritma SPICA-XB. Berikut adalah beberapa uji dan kriptanalisis yang dilakukan:

1. Kriptanalisis Dengan Analisis Frekuensi

   Seperti yang sudah dibahas sebelumnya, algoritma Caesar Cipher klasik dapat dipecahkan dengan mudah menggunakan algoritma kriptografi klasik, karena itu uji coba pertama yang penulis lakukan adalah memastikan bahwa algoritma SPICA-XB tidak dapat dipecahkan dengan analisis frekuensi.

   Dalam uji coba kali ini digunakan tools program bernama CryptoHelper. Berikut langkah-langkah yang dilakukan beserta analisa:

   a. Memasukkan *cipher* teks kedalam program CryptoHelper dan melakukan analisis frekuensi.



Gambar 5
Uji Coba Dengan Analisis Frekuensi

   b. Memeriksa frekuensi kemunculan angka dalam teks.

   CryptoHelper dalam analisis kali ini tidak dapat melakukan analisis frekuensi untuk teks hasil enkripsi yang dimasukan. Hal ini dikarenakan CryptoHelper hanya melakukan analisis frekuensi untuk huruf dan bukan angka. Hal ini dapat menunjukkan bahwa analisis frekuensi tidak mungkin dilakukan untuk *cipher* teks hasil enkripsi dengan algoritma SPICA-XB. Selain itu, seandainya dicoba untuk dilakukan analisa terhadap huruf-huruf yang ada, hanya akan terdapat 2 macam huruf, yaitu huruf 1 dan 0, sehingga data tersebut tidak akan memberikan arti apa-apa jika dilakukan analisis frekuensi.

2. Kriptanalisis Dengan Analisa Bit

   Proses kriptanalisis kedua yang penulis coba lakukan adalah dengan menganalisa bit-bit dari *cipher* teks hasil enkripsi. Metode yang penulis lakukan adalah sebagai berikut:

   a. Dilakukan pengelompokan biner ke dalam blok-blok dimana setiap blok mengandung 8 bit data. Misalkan untuk *cipher* teks berikut:

      10101001010010101001010000

      Ket. Setiap bit dalam blok yang sama diberi warna yang sama.

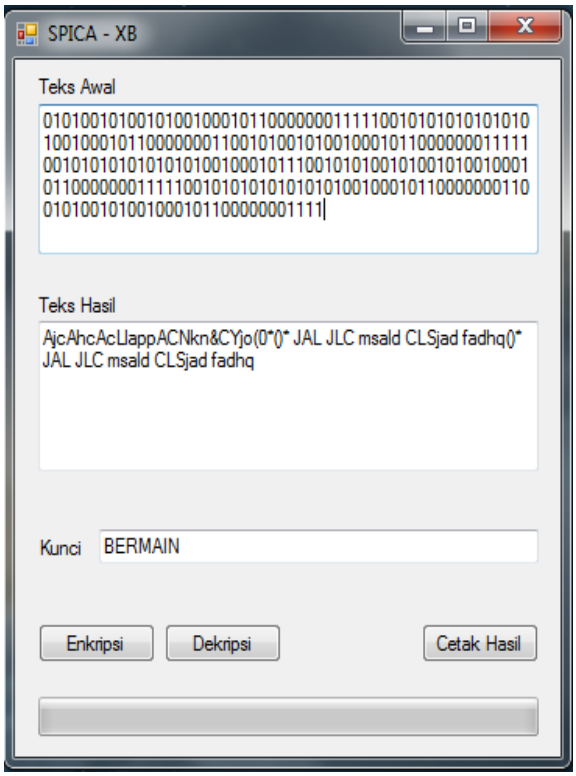
   b. Selanjutnya dilakukan pemeriksaan frekuensi kemunculan suatu blok dalam suatu teks. Misalkan untuk blok yang pertama yaitu 10101001 ternyata memiliki frekuensi kemunculan 2 kali, sementara blok-blok lainnya tidak terdapat pengulangan. (Hal ini cenderung disebabkan karena teks yang digunakan untuk uji coba kali ini adalah sebuah teks yang pendek).

   c. Dengan membandingkan blok yang memiliki kemunculan 2 kali (yaitu blok 10101001), penulis mencoba untuk menganalisa apakah blok tersebut merepresentasikan karakter asli yang sama. Namun ternyata kedua blok tersebut merupakan representasi karakter yang berbeda, yang pertama merupakan hasil enkripsi dari karakter E dan yang kedua adalah hasil enkripsi untuk karakter G.

   Dari hasil analisis, penulis menarik kesimpulan bahwa proses enkripsi dengan SPICA-XB akan memberikan hasil *cipher* teks yang tidak berpola. Meskipun kemungkinan variasi blok yang dihasilkan hanyalah 2^8 variasi, namun setiap blok tersebut adalah representasi yang berbeda dari setiap karakter *plain* teksnya. Hal ini menyebabkan algoritma ini tidak akan dapat dipecahkan dengan menggunakan metode Kasiski seperti layaknya pada algoritma Vigenere Cipher.

3. Uji Coba Dengan Merubah *Cipher* Teks

   Selanjutnya penulis mencoba untuk menguji apakah yang akan terjadi jika *cipher* teks yang merupakan hasil enkripsi dirubah dan didekripsi ulang menggunakan kunci yang sama. Dari hasil uji coba tersebut penulis memperoleh hasil sebagai berikut:



Gambar 6
Hasil Dekripsi Dengan *Modified Cipher* Teks

Dari hasil uji coba diatas, penulis menarik kesimpulan bahwa algoritma SPICA-XB dapat memberi proteksi kemanan yang cukup baik dikarenakan perubahan pada *cipher* teks yang ada, termasuk memasukan hanya sepenggal dari *cipher* teks akan memberikan hasil yang tidak akurat bahkan tidak bermakna.

V. KESIMPULAN

Berdasarkan hasil implementasi dan analisa yang telah dilakukan, penulis menarik kesimpulan bahwa algoritma SPICA-XB yang penulis kembangkan dalam tulisan kali ini dapat berfungsi dengan baik sebagai sebuah algoritma kriptografi alternatif.

Berdasarkan berbagai uji coba yang dilakukan, terbukti bahwa algoritma ini memiliki kekuatan yang lebih baik daripada algoritma Caesar Cipher klasik yang merupakan ide awal dari algoritma SPICA-XB.

Meskipun demikian, sebenarnya perlu dilakukan kriptanalisis lebih lanjut untuk mengetahui seberapa dalam kekuatan algoritma ini, terutama jika dilakukan pemecahan dengan metode *brute force.*

Kecepatan dari proses enkripsi untuk algoritma ini dapat dikatakan cukup baik. Penulis mencoba untuk melakukan enkripsi sebuah teks yang cukup panjang dan proses tersebut berhasil dilakukan dengan baik. Namun tentu saja hal ini belumlah bersifat kuantitatif dan masih dibutuhkan analisa lebih teliti lagi.

REFERENSI

[1] Sadikin, Rifki. 2012. Kriptografi Untuk Keamanan Jaringan. Yogyakarta : Penerbit Andi.
[2] Rasheed, Faraz. 2006. C# School. Programmers Heaven.
[3] Knuth, Donald E. 1985. The Art of Computer Programming. Stanford University : Addison-Wesley Publishing Company

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 25 Maret 2013

Rizal Panji Islami
13510066