Enkripsi dengan Menggunakan Fungsi Polinom Rekursif

Irfan Kamil (13510001)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
irfan@kamil.web.id

***Abstrak*—Makalah ini berisi tentang pembahasan enkripsi dan dekripsi dengan menggunakan fungsi polinom rekursif. Metode ini dibuat dengan tujuan membuat kunci rahasia yang bisa dikirimkan lebih fleksibel dan membuat hasil fungsi enkripsi yang seolah-olah acak.**

Faktor-faktor yang akan dibahas dalam makalah ini adalah cara kerja, proses, kompleksitas algoritma enkripsi, implementasi pada bahasa C, keteracakan fungsi enkripsi, probabilitas penebakan fungsi enkripsi, pencegahan *integer overflow*, dan keamanannya. Selain itu juga akan dibahas dimana tempat penerapan enkripsi ini.

Metode enkripsi ini sangat bergantung terhadap polinom yang digenerate. Pada suatu kasus, polinom dapat membuat keteracakan yang baik dan pada kasus polinom lain bisa membuat keteracakan yang memiliki banyak pengulangan. Apabila jumlah pengulangan kecil, fungsi akan semakin sulit untuk direduksi

***Kata Kunci*—Enkripsi, dekripsi, fungsi polinom-rekursif**

I. PENDAHULUAN

Keamanan data akhir-akhir ini menjadi sangat krusial karena data rahasia menyimpan berbagai macam informasi, misalnya informasi pribadi, transmisi pesan rahasia, dan perlindungan data yang memiliki hak cipta. Integritas data sangat diperlukan agar informasi tersebut dapat dilindungi. Apabila data rahasia jatuh ke tangan pihak yang tidak bertanggung jawab akan menjadi malapetaka bagi pemilik informasi rahasia tersebut.

Contohnya adalah kasus peretasan data pribadi Play Station Network yang ditaksir membuat kerugian milyaran dolar US. Peretasan ini menggunakan celah pada server Sony yang tidak ditambal. Setelah peretas berhasil menembus keamanan, peretas dapat mengambil semua data pribadi yang tidak dienkripsi. Kalau saja data tersebut terenkripsi, paling tidak akan menyulitkan peretas untuk mencuri data pribadi tersebut.

Selain itu, file program pada permainan juga membutuhkan enkripsi agar file tidak dapat dicuri dan dimodifikasi. Pencurian file pada permainan melanggar hak cipta pembuat permainan dan modifikasi dapat membuat *gameplay* dari permainan berubah dan tidak menutup kemungkinan data internal permainan dimodifikasi sesuka hati sehingga terjadi kelicikan pada permainan.

Salah satu algoritma enkripsi yang sudah terbukti secara matematis tidak dapat dipecahkan adalah algoritma One-time pad. Cipher dari algoritma ini sangat random dan independen antara satu elemen dengan elemen yang lain. Jadi, cara ini aman.

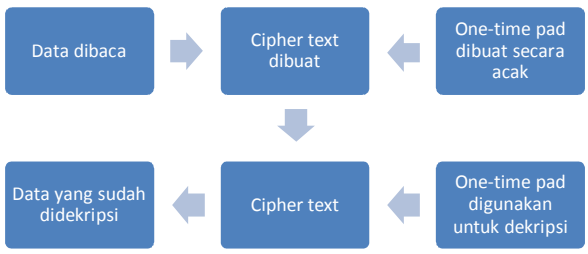
Namun, algoritma one-time pad memiliki beberapa kelemahan praktis yang besar, yaitu ukuran kunci rahasia yang simetris dengan data dan kunci rahasia harus acak. Pada pengacakan ini juga harus tidak menciptakan suatu pola pada kunci rahasia. Belum lagi apabila terjadi kesalahan intepretasi kunci rahasia akan berakibat hasil dekripsi data tidak benar.

Maka dari itu, dibutuhkan suatu metode enkripsi yang tidak dapat dipecahkan namun pengiriman kunci rahasianya lebih praktis untuk dilakukan. Penulis berusaha membuat algoritma enkripsi yang menjawab kebutuhan ini dengan cara membuat algoritma enkripsi dengan menggunakan fungsi polinom rekursif. Fungsi polinom rekursif ini dibuat seolah-olah seperti one-time pad yang digenerate secara acak.

II. ALGORITMA ONE-TIME PAD

One-Time Pad adalah enkripsi yang telah terbukti secara matematis tidak dapat dipecahkan. Cara enkripsinya, setiap komponen dari data masukkan dijumlahkan secara modular dengan kunci yang dibuat secara acak dengan ukuran yang sama dengan data masukkan. Apabila kuncinya terjaga rahasia dan pengulangannya terjadi sesedikit mungkin, hasil enkripsinya tidak mungkin dikembalikan tanpa diketahui kunci yang benarnya.[1]

Cara dekripsinya, setiap komponen yang telah dienkripsi dikurangkan secara modular dengan kunci rahasia yang diketahui. Hanya saja, apabila terjadi kesalahan pada kunci yang diterima, hasil dekripsinya akan kacau.



Gambar 1 Cara kerja One-time pad

Berikut ini adalah contoh penggunaan One Time Pad.

Data :

```
AF 7C 55 26 3B 89 FF A2
```

Kunci Rahasia :

```
59 0A FE DA 52 37 2E 45
```

(Data + Kunci) mod 256 :

```
08 86 53 00 8D C0 2D E7
```

Jadi, cipher textnya adalah 08 86 53 00 8D C0 2D E7.

Data cipher text :

```
08 86 53 00 8D C0 2D E7
```

Kunci Rahasia :

```
59 0A FE DA 52 37 2E 45
```

(Data - Kunci) mod 256 :

```
AF 7C 55 26 3B 89 FF A2
```

Jadi, hasil dekripsinya adalah AF 7C 55 26 3B 89 FF A2

Algoritma enkripsi ini memiliki kelemahan besar dalam hal praktisnya[1], yaitu pengiriman kunci rahasia yang simetris dengan jumlah datanya. Maka dari itu, diperlukan suatu metode enkripsi yang tidak dapat dipecahkan namun pengiriman kunci rahasianya lebih visibel. Maka dari itu, perlu suatu cara agar ukuran kunci rahasia kecil namun tetap tidak dapat dipecahkan.

III. Algoritma Enkripsi dengan Menggunakan Fungsi Polinom Rekursif

Algoritma ini adalah algoritma yang menggunakan kunci rahasia yang hanya diketahui oleh pengirim dan penerima. Kunci rahasia dalam hal ini adalah fungsi polinom yang digunakan pada enkripsi dan dekripsi. Kunci rahasia dibuat secara random.

Berikut ini adalah langkah-langkah enkripsi :

1. Kunci rahasia digenerate
2. Hasil fungsi polinom rekursif ke-n dibuat
3. Cipher text ke-n dibuat dengan cara menjumlahkan data ke-n dengan fungsi polinom rekursif dari kunci rahasia ke-n.
4. Kembali ke langkah ke-2 hingga seluruh data terenkripsi.

Berikut ini adalah langkah-langkah dekripsi :

1. Kunci rahasia didapatkan
2. Hasil fungsi polinom rekursif ke-n dibuat
3. Data ke-n dibuat dengan cara mengurangkan cipher text ke-n dengan fungsi polinom rekursif dari kunci rahasia ke-n
4. Kembali ke langkah ke-2 hingga seluruh data terdekripsi.

Algoritma ini adalah algoritma yang menggunakan kunci rahasia yang hanya diketahui oleh pengirim dan penerima. Kunci rahasia dalam hal ini adalah fungsi polinom yang digunakan pada enkripsi dan dekripsi. Kunci rahasia dibuat secara random.

$$P(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_j x^j \quad (1)$$

Fungsi diatas adalah suatu fungsi polinom berderajat j. Fungsi tersebut akan digunakan untuk generator fungsi F pada persamaan (2).

$$F(n) = (P_1(F(n-1)) + P_2(F(n-2)) + \cdots + P_i(F(n-i) + \varphi(n \bmod 256)) mod 256 \quad (2)$$

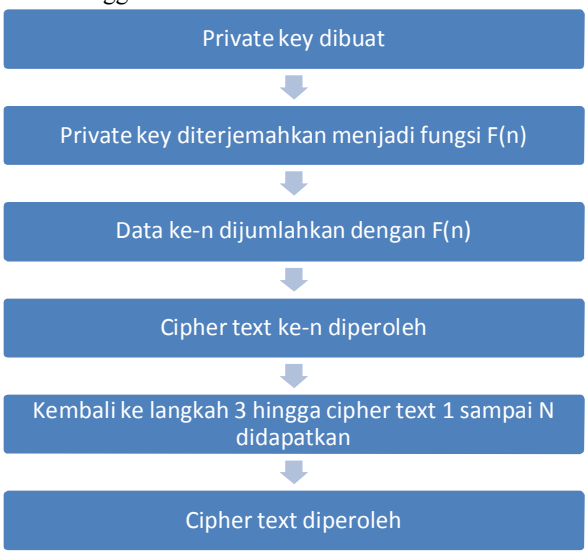
Dengan F adalah fungsi polinom rekursif ke-n, P_1 hingga P_i adalah fungsi polinom yang digenerate, dan φ adalah fungsi polinom untuk n. Yang harus digenerate adalah i+1 buah fungsi polinom dan i buah nilai F untuk basis fungsi rekursif. Hasil generate tersebut lalu diterjemahkan menjadi kunci rahasia. Ukuran kunci rahasia tersebut tergantung dengan i dan kebesaran fungsi polinomnya.

IV. Proses Enkripsi dan Dekripsi

A. Proses Enkripsi

Proses enkripsi diawali dengan pembuatan kunci rahasia. Kunci rahasia ini diterjemahkan ke i+1 buah fungsi polinom dan i buah basis fungsi. Fungsi polinom terdiri dari beberapa konstanta sejumlah derajat polinom yang diinginkan.

Setelah itu, hasil fungsi dari ke-1 sampai ke-N dihitung. Lalu, cipher text dihasilkan dengan cara menjumlahkan data ke-n dengan hasil fungsi ke-n untuk data ke-1 hingga ke-N.

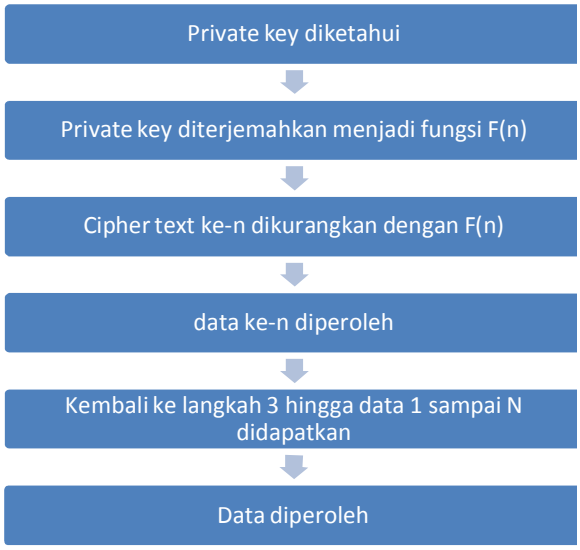


Gambar 2 Cara kerja enkripsi

B. Proses Dekripsi

Proses dekripsi diawali dengan pengambilan kunci rahasia. Kunci rahasia ini diterjemahkan ke i+1 buah fungsi polinom dan i buah basis fungsi. Fungsi polinom terdiri dari beberapa konstanta sejumlah derajat polinom yang diinginkan.

Setelah itu, hasil fungsi dari ke-1 sampai ke-N dihitung. Lalu, data hasil dekripsi dihasilkan dengan cara menjumlahkan data ke-n dengan hasil fungsi ke-n untuk cipher text ke-1 hingga ke-N.



Gambar 3 Cara kerja dekripsi

C. Contoh Enkripsi

Misalkan dibuat kunci rahasia sebagai berikut:

$i = 2$

j = 5

$P_1(x) = x + 2x^2 + 3x^3 + 4x^4 + 5x^5$

$P_2(x) = x + x^2 + x^3 - x^4 + x^5$

$\varphi(x) = 77 + x + x^2 + x^3 - x^4 + x^5$

$F(0) = 24$

$F(-1) = 176$ (3)

Untuk mengenkripsi data teks irfankamil yang representasi bytenya adalah 69 72 66 61 6E 6B 61 6D 69 6C. Dari key di atas, dibuatlah hasil F dari 1 sampai 10.

$F(1) = (P_1(24) + P_2(176) + \varphi(1))mod\ 256 = 149$

$F(2) = (P_1(149) + P_2(24) + \varphi(2))mod\ 256 = 146$

$F(3) = (P_1(146) + P_2(149) + \varphi(3))mod\ 256 = 10$

$F(4) = (P_1(10) + P_2(146) + \varphi(4))mod\ 256 = 171$

$F(5) = (P_1(171) + P_2(10) + \varphi(5))mod\ 256 = 196$

$F(6) = (P_1(196) + P_2(171) + \varphi(6))mod\ 256 = 204$

$F(7) = (P_1(204) + P_2(196) + \varphi(7))mod\ 256 = 217$

$F(8) = (P_1(217) + P_2(204) + \varphi(8))mod\ 256 = 238$

$F(9) = (P_1(238) + P_2(217) + \varphi(9))mod\ 256 = 54$

$F(10) = (P_1(54) + P_2(238) + \varphi(10))mod\ 256 = 31$

sehingga diperoleh representasi bit hasil fungsi F dari 1 sampai 10 adalah 95 92 0A AB C4 CC D9 EE 36 1F.

Setelah itu, dihitung hasil cipher text dengan menjumlahkan data dengan hasil fungsi dalam modulus 256 sehingga diperoleh FE 04 70 0C 32 37 3A 5B 9F 8B.

D. Contoh Dekripsi

Setelah itu, dihitung hasil cipher text dengan menjumlahkan data dengan hasil fungsi dalam modulus 256 sehingga diperoleh 95 92 0A AB C4 CC D9 EE 36 1F.

Misalkan digunakan F yang sama dengan sistem persamaan (3) dan cipher text yang akan didekripsi adalah FE 04 70 0C 32 37 3A 5B 9F 8B. Data hasil dekripsinya adalah hasil pengurangan cipher text dengan fungsi F dari 1 sampai 10 dalam modular 256. Data tersebut adalah 69 72 66 61 6E 6B 61 6D 69 6C. Sehingga, apabila data tersebut diterjemahkan menjadi string adalah irfankamil.

V. Analisis dan Pembahasan

A. Pembuatan Fungsi dari Kunci rahasia

Pembuatan fungsi enkripsi tergantung dengan nilai *i* dan derajat polinom yang diinginkan. Semakin besar nilai *i* dan derajat polinom, semakin sulit untuk ditebak bentuk fungsinya.

Koefisien pada polinom bisa angka berapapun. Begitu juga dengan basis fungsi. Namun, karena akan dihitung dalam modulo 256, koefisien polinom direpresentasikan dalam signed 8 bit dan koefisien basis direpresentasikan dalam unsigned 8 bit. Sehingga, setiap koefisien dan basis memiliki 256 kemungkinan.

Jadi, ukuran dari kunci rahasia enkripsi ini adalah

$$U(i,j) = 2 + (j+1)(i+1) + i = (2j + ij + 2i + 3)\ Byte \quad (4)$$

Dengan *i* adalah jumlah fungsi polinom dengan rekurens dan j adalah derajat masing-masing polinom. Koefisien 2 adalah masing-masing 1 byte untuk nilai *i* dan *j*.

B. Pseudocode Program Enkripsi

Dari penjabaran spesifikasi di atas, berikut ini adalah pseudocode aplikasi untuk mengenkripsi masukan.

```
Input(i)
Input(j)
For a = 1 to i do
    Read_polynom(P[a])
Read_polynom(Ph)
Read_base(Prev)
Input(Data)
N = Length(Data)
For a = 1 to N do
    F = 0
    For b = 1 to i do
        F = F + Pol(P[b],Prev[b])
    F = F + Pol(Ph,a mod 256)
    F = F mod 256
    For b = i downto 2 do
        Prev[b] = Prev[b+1]
    Prev[1] = F
    Cipher[a] = (Data[a] + F)mod 256
Output(Cipher)
```

C. Kompleksitas Algoritma Enkripsi

Dari contoh pseudocode di atas, proses khas yang dilakukan adalah menghitung nilai F dan Cipher. Jadi

$O(f(n)) = n$ (5)

karena pengulangan dari baris 9 sampai line 18 tersebut bersifat linier. Jadi, waktu eksekusi program berbanding lurus dengan jumlah data.

Selain itu, perhitungan polinom dilakukan dengan proses penjumlahan dan perkalian. Kompleksitas waktu asimptotik dari perhitungan polinom adalah

$O(g(j)) = j$ (6)

diperoleh dari perhitungan

$P(x) = a_0 + x(a_1 + x(a_2 + x(\ldots + (a_j)\ldots)$ (7)

Kompleksitas waktu asimptotik dari penjumlahan perhitungan polinom adalah

$O(h(i)) = i$ (8)

karena jumlah penjumlahan fungsi F berbanding lurus dengan jumlah polinom yang ada.

Jadi, kompleksitas waktu asimptotik dari program enkripsi adalah

$O(f(n).g(j).h(i)) = nji$ (9)

D. Implementasi pada Bahasa C

Berikut ini adalah pseudocode pada bagian 4B yang diterjemahkan ke bahasa C. Kode ini khusus untuk *i*=2 dan *j*=5.

```
int main() {
	int p1[5];
	int p2[5];
	int p3[5];
	char d[10];
	int de[10];
	int i;
	int a1,a2,c;
	for (i = 0; i < 5; i++)
		scanf("%d",&p1[i]);
	for (i = 0; i < 5; i++)
		scanf("%d",&p2[i]);
	for (i = 0; i < 5; i++)
		scanf("%d",&p3[i]);
	scanf("%d",&c);
	scanf("%d %d",a1,a2);
	scanf("%s",d);
	for (i = 0; i < 10; i++) {
		//tampilkan byte data test
		printf("%.2X ",d[i]);
	}
	printf("\n");
	int k[10];
	for (i = 0; i < 10; i++) {
		k[i] = (pol(p1,a1) +
		pol(p2,a2) + pol(p3,i) +
		c) % 256;
		a2 = a1;
		a1 = k[i];
		//tampilkan fungsi F
		printf("%.2X ",k[i]);
	}
	printf("\n");
	for (i = 0; i < 10; i++) {
		de[i] = xp(d[i],k[i]);
		//tampilkan hasil enkripsi
		printf("%.2X ",de[i]);
	}
	printf("\n");
	for (i = 0; i < 10; i++) {
		//tampilkan hasil dekripsi
		printf("%.2X ",xp(de[i],
		-k[i]));
	}
	return 0;
}
```

E. Keteracakan Nilai F

Keteracakan dari nilai F(n) diperlukan agar *reverse engineering* untuk mencari fungsi F semakin sulit untuk dilakukan. Dari hasil tes penulis dengan nilai F(n) untuk n dari 1 hingga 100 untuk beberapa fungsi polinom tertentu dengan nilai *i*=2 dan *j*=5,dan basis 24 dengan 176 didapatlah hasil berikut.

No	$P_1(a_0..a_5)$	$P_2(a_0..a_5)$	$\varphi(a_0..a_5)$	Pengulangan
1	0 1 2 3 4 5	0 1 1 1 -1 1	77 1 1 1 -1 1	3
2	0 -4 5 1 0 3	0 6 1 1 1 2	77 -1 7 2 3 5	5
3	0 10 -9 8 -7 6	0 -5 4 -3 2 -1	5 0 -1 2 -3 4	2
4	0 4 5 1 2 3	0 4 5 1 2 3	0 4 5 1 2 3	6
5	0 1 5 1 2 1	0 8 3 4 1 2	1 9 1 9 1 9	16
6	0 0 0 0 0 1	0 0 0 0 0 2	0 0 0 0 0 3	3

Tabel 1 Hasil pengulangan fungsi

Pengulangan yang dimaksud di atas adalah banyaknya nilai F(n) yang sama dengan F(n) dengan n yang lain dengan n dari 1 hingga 100. Dari tes diatas, tidak ditemukan pengulangan 2 byte.

Dari tabel diatas, pengulangan nilai F(n) terjadi dalam jumlah tertentu, tergantung fungsi polinomnya. Fungsi polinom no 3 adalah kasus terbaik dari percobaan tersebut karena hanya terjadi maksimal 2 kali pengulangan byte yang sama. Sedangkan, kasus terburuk dari percobaan di atas adalah pada fungsi polinom no 5 karena terjadi pengulangan 16 kali byte yang sama. Sehingga, rotasi fungsi F(n) dapat terjadi dengan periode yang kecil sehingga enkripsi dari fungsi ini lebih mudah direduksi daripada fungsi yang lain.

F. Probabilitas Penebakan Fungsi F

Kemungkinan fungsi F tergantung dari nilai i, j, dan ukuran integer. Secara umum, jumlah kemungkinan nilai F yang mungkin adalah

$N = (256)^{(j)(i+1)+i+1}$(10)

Nilai (j)(i+1) diperoleh dari jumlah kemungkinan permutasi berulang dari koefisien polinom tanpa konstanta dan i adalah jumlah nilai basis. Konstanta polinom berpengaruh saling respektif untuk setiap polinom dan karena akhirnya akan dimodulus dengan 256, maka jumlah kemungkinan yang dibuat dari konsanta semua persamaan adalah 256.

Contohnya, jika i=2 dan j=5 dengan ukuran integer 8 bit

$N = (256)^{(5)(2+1)+2+1} = 256^{18} = 2.23 \times 10^{43}$(11)

Penambahan nilai i dan j akan membuat N berubah secara eksponensial sehingga kemungkinan penebakan kombinasi fungsi F hampir tidak mungkin karena nilainya mendekati 0, walaupun *i* hanya bernilai 2 dan *j* bernilai 5.

G. Pencegahan Integer Overflow

Bahasa pemrograman memiliki kelemahan yaitu hanya bisa menyimpan nilai dalam suatu ukuran. Fungsi polinom bisa menghasilkan nilai fungsi yang sangat besar dan melebihi kapasitas bilangan bulat bahasa pemrograman. Maka dari itu, karakteristik operasi

perkalian diubah dengan sifat aritmatika modulo.

$$Kali(a,b) = (a * b) mod\ 256 = ((a\ mod\ 256) * (b\ mod\ 256)) mod\ 256 \quad (11)$$

Dengan cara diatas, hasil perantara perkalian a dan b tidak akan jauh melebihi 256.

H. Keamanan

Dari hasil perhitungan probabilitas penebakan fungsi F, kemungkinannya sangat kecil untuk dapat ditebak. Sekalipun terjadi tebakan yang benar, tidak ada suatu parameter bahwa hasil dekripsi dengan suatu fungsi F tebakan itu benar.

Suatu fungsi F(n) bisa saja digunakan berkali-kali. Namun, apabila fungsi F(n) digunakan lebih dari sekali, fungsi F(n) bisa ditebak dengan cara analisis reduksi kriptografi dari masing-masing cipher text yang didapatkan. Hanya saja, apabila data sebelum enkripsi tidak diketahui, tetap reduksi akan sulit untuk dilakukan.

Namun, apabila kunci rahasia diketahui, dekripsi menjadi sangat mudah. Jadi, kunci rahasia harus diamankan secara maksimal.

VI. IMPLEMENTASI ENKRIPSI

A. Transmisi Data

Algoritma enkripsi ini dapat diterapkan untuk transmisi data baik melalui perangkat keras, gelombang elektromagnetik, maupun dengan media online. Hanya saja, untuk transmisi data, penerima harus sudah mengetahui kunci rahasia sebelum cipher text bisa didekripsi.

B. Proteksi Arsip Digital dengan Sandi Lewat

Arsip digital dapat diproteksi dengan sandi lewat. Salah satunya dengan algoritma enkripsi ini. Sandi lewat yang bertipe larik dapat diubah dengan suatu generator fungsi menjadi fungsi F untuk enkripsi dan dekripsi.

C. Proteksi Permainan Online dari Modifikasi Program dan Reverse Engineering

Data dari permainan online adalah data yang mengandung hak cipta dari pembuatnya. Namun, apabila tidak diproteksi, bisa saja pihak yang tidak bertanggung jawab melakukan modifikasi atau bahkan reverse engineering. Keberadaan enkripsi sangat dibutuhkan untuk mencegah hal ini.

Enkripsi bisa diimplementasikan pada semua data eksternal program pada permainan. Setiap patch dilakukan, kode rahasia bisa diganti sehingga sekalipun kode rahasia pernah tertebak, namun akan menyulitkan peretas karena kode rahasia berubah dalam waktu yang singkat. Penggantian kode rahasia bisa dilakukan dengan cara data ke-n dikurangkan dengan F(n) dari kode rahasia yang lama dan ditambahkan dengan F(n) dari kode rahasia yang baru.

VII. KESIMPULAN

Dari pembahasan sebelumnya, dapat disimpulkan bahwa cara enkripsi ini aman dari reduksi analisis kriptografi. Selain itu, pengiriman kunci rahasia lebih praktis dibandingkan kunci rahasia dari metode one-time pad. Hanya saja, tingkat keamanan enkripsi ini tergantung dengan keteracakan fungsi F yang hanya terjadi pada sebagian kondisi F yang memiliki pengulangan sejarang mungkin. Selain itu, kunci rahasia harus diganti setiap telah digunakan atau dalam kurun waktu tertentu agar menyulitkan proses reduksi.

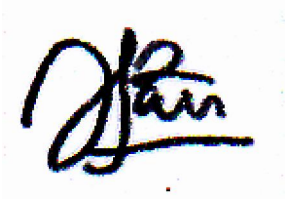
REFERENSI

[1] http://en.wikipedia.org/wiki/One-time_pad 6.15 AM 11/12/2011

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, 11 Desember 2011



Irfan Kamil | 13510001