Penerapan Sistem Persamaan Lanjar untuk Merancang Algoritma Kriptografi Klasik

Hendra Hadhil Choiri (135 08 041)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
hendra_h2c@students.itb.ac.id

***Abstrak*— Dalam proses enkripsi-dekripsi, terdapat banyak algoritma kriptografi yang sudah diciptakan dan dikembangkan. Tiap algoritma menggunakan pendekatan tertentu yang pasti memiliki kelebihan dan kekurangan masing-masing. Melalui proposal makalah ini, penulis mengusulkan penerapan sistem persamaan lanjar (*linear equation system*) dalam perancangan suatu algoritma kriptografi. Melalui pendekatan ini, dapat diturunkan berbagai algoritma klasik untuk enkripsi dan dekripsi. Variasi algoritma yang terbentuk berbeda dalam hal penempatan plainteks dan kunci saat membentuk persamaan-persamaan lanjar. Dan tentu saja setiap algoritma enkripsi harus memiliki algoritma dekripsi yang sesuai, agar pesan yang dienkripsi dapat dikembalikan lagi. Selanjutnya akan dianalisis kelebihan dan kekurangannya, serta teknik-teknik kriptanalisis yang bisa dilakukan untuk memecahkannya. Diharapkan, sistem persamaan lanjar dapat dikembangkan untuk menciptakan suatu algoritma kriptografi yang akurat dan sulit dipecahkan sehingga pesan menjadi aman.**

***Kata Kunci*—kriptografi klasik, sistem persamaan lanjar, algoritma kriptografi.**

I. PENDAHULUAN

Kriptografi adalah ilmu dan seni untuk menjaga kerahasian pesan dengan cara menyandikannya ke dalam bentuk yang tidak dapat dimengerti lagi maknanya. Sedangkan kriptografi klasik adalah kriptografi kunci-simetris yang berbasis karakter [1] Telah banyak algoritma kriptografi yang telah diciptakan dan dikembangkan. Tiap algoritma menggunakan pendekatan tertentu yang pasti memiliki kelebihan dan kekurangan masing-masing.

Melalui makalah ini, penulis bermaksud merancang beberapa algoritma kriptografi yang menggunakan suatu konsep dalam aljabar lanjar (*linear algebra*). Yakni, dalam hal ini memanfaatkan sistem persamaan lanjar (*linear equation system*) dalam penyusunan algoritmanya. Meskipun tiap algoritma pasti punya kekurangan, diharapkan pendekatan ini dapat dikembangkan untuk menciptakan algoritma kriptografi klasik yang kokoh dan tidak mudah dipecahkan.

II. SISTEM PERSAMAAN LANJAR

Pada dasarnya, sistem persamaan lanjar adalah kumpulan persamaan – persamaan yang berbentuk:

$$\begin{aligned}
&a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\
&a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\
&\cdots \\
&\cdots \\
&a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m
\end{aligned}$$

Terlihat bahwa tiap persamaan bersifat lanjar (*linear*) terhadap $x_1, x_2, \cdots, x_n$, sehingga disebut persamaan lanjar.[2]

Selanjutnya, sistem persamaan lanjar tersebut dapat direpresentasikan menjadi suatu operasi matrik-matrik sebagai berikut:

$$\begin{bmatrix} a_{11}a_{12} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1}\ a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

Untuk selanjutnya, dalam makalah ini penulis sebut matrik-matrik di atas:

A = $\begin{bmatrix} a_{11}a_{12} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1}\ a_{m2} & \cdots & a_{mn} \end{bmatrix}$ sebagai matrik koefisien

X = $\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$ sebagai matrik variabel

B = $\begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$ sebagai matrik konstanta

III. RUMUSAN MASALAH

Dengan memanfaatkan konsep persamaan lanjar tersebut, dapat diturunkan berbagai algoritma kriptografi klasik. Secara umum, algoritma enkripsi dirancang berdasarkan bagaimana merepresentasikan plainteks dan kunci menjadi matrik-matrik koefisien dan variabel. Sedangkan cipherteks adalah matrik koefisien yang dihasilkan.

Misal algoritma akan diterapkan pada sistem alfabet α

karakter (misal α=26 untuk huruf Indonesia, dan α=256 untuk ASCII). Dan p_i, k_i, dan c_i berturut-turut menyatakan suatu karakter di plainteks, kunci, dan cipherteks. Berikut beberapa gambaran algoritma yang dapat dibuat: *(contoh dan penjelasan rinci akan diberikan saat makalah direalisasikan setelah disetujui*)

- ✓ **Pendekatan 1: Kunci berupa matrik koefisien (yakni *Hill cipher*)**

Diawali dari algoritma yang sudah ada, dan cukup terkenal yaitu Hill cipher yang dikembangkan oleh Lester Hill (1929).

Algoritma ini menerapkan sistem persamaan lanjar di mana digunakan *m* buah persamaan lanjar. Selanjutnya plainteks dibagi-bagi menjadi blok-blok berisi m karakter. Lalu gunakan persamaan:

$$c_1 = (k_{11}\, p_1 + k_{12} p_2 + \ldots + k_{1m}\, p_m) \bmod \alpha$$
$$c_2 = (k_{21}\, p_1 + k_{22} p_2 + \ldots + k_{2m}\, p_m) \bmod \alpha$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$
$$c_3 = (k_{31}\, p_1 + k_{32} p_2 + \ldots + k_{3m}\, p_m) \bmod \alpha$$

terlihat bahwa pada algoritma ini, kunci berupa matrik berukuran mxm (berisi elemen k_{11} hingga k_{mm}), seperti ini:

$$\begin{pmatrix} C_1 \\ C_2 \\ C_3 \end{pmatrix} = \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix}$$

Sehingga secara umum, algoritma enkripsinya dapat dinyatakan:

$$\mathbf{K}_{mxm}.\mathbf{P}_{mx1} = \mathbf{C}_{mx1}$$

dan untuk proses dekripsi, cukup diinverskan matrik K:

$$\mathbf{P}_{mx1} = \mathbf{C}_{mx1.}\, \mathbf{K}_{mxm}^{-1}$$

Algoritma ini memang cukup bagus dan aman, tetapi kurang nyaman karena harus memiliki kunci berukuran mxm. Untuk mengenkripsi suatu blok dengan karakter saja, dibutuhkan kunci sepanjang 100 karakter.

- ✓ **Pendekatan 2: Kunci sebagai faktor variabel**

Misal kuncinya adalah $k_1k_2\ldots k_m$ (panjangnya m), bagi plainteks menjadi blok-blok berisi m karakter (misal dalam suatu blok isinya $p_1p_2\ldots p_m$). Selanjutnya algoritma enkripsinya adalah sebagai berikut:

$$c_i = ((\textstyle\sum_{t=1}^{m} k_t p_t) - k_i p_i) \bmod \alpha$$

Selanjutnya untuk menemukan cara dekripsikan, dapat digunakan manipulasi aljabar seperti metode eliminasi sebagai berikut:

$$c_1 = \quad k_2p_2 + k_3p_3 + k_4p_4 + \cdots + k_mp_m$$
$$c_2 = k_1p_1 \quad + k_3p_3 + k_4p_4 + \cdots + k_mp_m$$
$$c_3 = k_1p_1 + k_2p_2 \quad + k_4p_4 + \cdots + k_mp_m$$
$$\ldots\ldots\ldots\ldots$$
$$c_m = k_1p_1 + k_2p_2 + k_3p_3 + \cdots + k_{m-1}p_{m-1}$$
$$\text{---------- +}$$
$$\sum_{t=1}^{m} c_t = m - 1(k_1p_1 + k_2p_2 + \cdots + k_mp_m)$$

Kita dapatkan persamaan baru:

. Akhirnya didapatkan algoritma dasar untuk dekripsinya:

$$p_i = \frac{1}{k_i}\left(\frac{\sum_{t=1}^{m} c_t}{m-1} - c_i\right)$$

Namun formula di atas masih harus dikoreksi, dalam hal adanya modularitas (mod α) karena adanya operasi pembagian. Untuk itu, kita lakukan koreksi mulai dari persamaan:

$$\textstyle\sum_{t=1}^{m} c_t \equiv (m-1)(k_1p_1 + k_2p_2 \ldots + k_mp_m)(mod\ \alpha)$$

Dan untuk mencari nilai dari $k_1p_1 + \cdots + k_mp_m$, kita tidak boleh langsung membagi $\sum_{t=1}^{m-1} c_t$ dengan m-1, karena pembagian bukan termasuk operasi dalam persamaan kongruensi. Sehingga yang bisa dilakukan yaitu mengalikan kedua ruas dengan $(m-1)^{-1} mod\ \alpha$ sehingga akan memenuhi

$$(m-1)^{-1}.m - 1(k_1p_1 + k_2p_2 + \cdots + k_mp_m)(mod\ \alpha)$$
$$\equiv 1.(k_1p_1 + k_2p_2 + \cdots + k_mp_m)(mod\ \alpha)$$
$$\equiv (k_1p_1 + k_2p_2 + \cdots + k_mp_m)(mod\ \alpha)$$

Jadi telah kita dapatkan persamaan

$$k_1p_1 + k_2p_2 \ldots k_mp_m \equiv (m-1)^{-1} \sum_{t=1}^{m} c_t\ (mod\ \alpha)$$

Setelah mendapat persamaan di atas, dapat dengan mudah kita cari nilai p_i, yakni dengan mengurangkan persamaan di atas dengan c_i, akan diperoleh:

$$(m-1)^{-1}\left(\sum_{t=1}^{m} c_t\right) - c_i \equiv k_ip_i(mod\ \alpha)$$

Jadi telah kita temukan:

$$p_i = k_i^{-1}((m-1)^{-1}(\sum_{t=1}^{m} c_t) - c_i)\ mod\ \alpha$$

Dilihat dari proses enkripsi, terlihat bahwa algoritma ini cukup baik dan sulit dipecahkan karena dalam satu blok, tiap karakter saling mempengaruhi.

Namun, masih terdapat masalah dalam menerapkan algoritma ini, yakni adanya invers. Karena a^{-1} mod m ada jika dan hanya jika a dan m relatif prima (GCD (a,m)=1).

Sehingga pada kasus di atas haruslah GCD(k_i,α)=GCD(m-1,α)=1. Untuk itu, perlu dibuat

batasan dalam penerapan algoritma tersebut, berikut 3 alternatif yang mungkin:

- Memilih k_i dan m-1 yang relatif prima terhadap α. Untuk memilih m tidak ada masalah karena panjang blok memang tetap. Namun masalah terjadi dalam memilih k_i. Karena kunci ditentukan oleh pihak yang akan berkirim pesan. Akan sangat tidak nyaman jika harus diberi aturan 'karakter-karakter pada kunci harus relatif prima dengan jumlah alfabet'. Untuk itu cara ini tidak disarankan.
- Menggunakan alfabet yang banyak karakternya merupakan bilangan prima. Karena setiap bilangan asli yang kurang dari bilangan prima p, pasti saling prima dengan p. Masalahnya, tidak setiap himpunan karakter banyak karakternya adalah bilangan prima. Misalnya saja abjad A-Z ada 26 karakter, dan ASCII ada 256 karakter. Namun, hal ini dapat diselesaikan dengan menambah/mengurangi sejumlah karakter sehingga banyak karakternya berupa bilangan prima. Berhubung ini algoritma ini diterapkan untuk algoritma klasik, maka manipulasi alphabet bisa dilakukan dengan mudah. Namun tetap saja, meskipun banyak karakter alfabetnya berupa bilangan prima, dekripsi tidak bisa dilakukan jika kunci mengandung karakter bernilai nol, karena $GCD(0,\alpha)\neq 1$.
- Alternatif selanjutnya adalah menyesuaikan karakter kunci. Yakni, jika k_i ternyata relatif prima dengan α, ganti k_i dengan karakter lain yang relatif prima dengan α.Bisa dengan karakter terdekatnya, atau ganti dengan suatu karakter khusus.

  Dengan begitu, tidak perlu memberikan batasan terhadap kunci dan alphabet. Namun tetap saja disarankan memilih alphabet yang banyak karakternya bukan kelipatan angka-angka kecil seperti 2 atau 3, karena akan terdapat banyak karakter kunci yang tidak sesuai. Pada makalah ini penulis sertakan implementasi dari kriptografi yang menggunakan metode ini.

✓ **Pendekatan 3: Sebuah karakter kunci mengenkripsi satu blok.**

Apabila pendekatan 2 sulit diimplementasikan karena tidak bisa memanipulasi alphabet (misalnya ASCII yang harus 256 karakter, pengurangan ataupun penambahan karakter menyebabkan kerusakan file), dapat diterapkan metode yang ini. Yakni, plainteks dipartisi menjadi blok-blok, lalu blok ke-i (misal berisi m karakter) dienkripsi dengan karakter kunci ke-i. Lalu gunakan suatu persamaan lanjar untuk mengenkripsi sebuah blok dengan sebuah karakter, misalnya:

$$c_i = k_i + (\sum_{t=1}^{m} p_t) - p_t \bmod \alpha$$

dan untuk menemukan algoritma dekripsinya, gunakan cara seperti di pendekatan 2, jumlahkan semua karakter cipherteks dalam satu blok, didapat:

$$\sum_{t=1}^{m} c_t \equiv (m-1)(p_1 + p_2 \ldots + p_m) + mk_i (mod\ \alpha)$$

Yang setara dengan:

$$p_1 + p_2 \ldots + p_m \equiv (m-1)^{-1}(\sum_{t=1}^{m} c_t - mk_i)\ (mod\ \alpha)$$

Selanjutnya dengan mengurangkan bentuk di atas dengan $(c_i\text{-}k_i)$, diperoleh karakter plainteks:

$$\left(\sum_{t=1}^{m} p_t\right) - (c_i - k_i)$$
$$= (\textstyle\sum_{t=1}^{m} p_t) - (k_i + (\sum_{t=1}^{m} p_t) - p_t - k_i) = \mathbf{p_i}$$

Jadi dituliskan kembali, formula dekripsinya adalah:

$$p_i = ((m-1)^{-1}(\sum_{t=1}^{m} c_t) - mk_i) - c_i + k_i mod\ \alpha$$

Dan seperti pendekatan 2, pada metode ini harus tetap berlaku bahwa $GCD(m\text{-}1,\alpha)=1$, tetapi karakter kunci boleh bebas, karena satu karakter kunci mengenkripsi satu blok.

Namun metode ini tidak cocok untuk plainteks yang kecil karena kunci menjadi kurang terpakai. Untuk itu metode ini dapat dipadukan dengan Vigenere cipher agar lebih kuat.

✓ **Pendekatan 4: Karakter kunci sebagai panjang partisi blok**

Misal kuncinya adalah $k_1k_2\ldots k_m$. Anggap karakter-karakter kunci sebagai bilangan bulat. Selanjutnya partisi plainteks menjadi blok-blok, dengan tiap blok berisi k_i karakter, dan kunci diulang.

Misal kunci = halo (7,0,11,14), plainteks akan dipartisi menjadi:

$(p_1p_2p_3p_4p_5p_6p_7)()(p_8p_9p_{10}p_{11}p_{12}p_{13}p_{14}p_{15}p_{16}p_{17}p_{18})(p_{19}..)$

Lalu, tiap blok didekripsi dengan suatu algoritma enkripsi, baik dengan algoritma klasik lain, maupun dengan pendekatan sistem persamaan lanjar.

✓ **Masih banyak pendekatan yang dapat dirancang dengan memanfaatkan SPL.**

Dan untuk menciptakan algoritma yang bagus, gunakan sistem enkripsi m-gram. Yakni, dibagi menjadi blok-blok. Dan pada blok yang terdiri dari m karakter, gunakan formula enkripsi

$$c_i = f(p_1,p_2,p_3,\ldots,p_m, k_1,k_2,k_3,\ldots,k_m)$$

IV. IMPLEMENTASI

Dari pendekatan-pendekatan di bab 3, penulis telah mengimplementasikan salah satunya, yaitu pendekatan 2.

Karena sepertinya memang menggunakan metode yang paling kuat dan sulit dipecahkan, dan akan dijelaskan lebih lanjut di bagian analisis.

Untuk enkripsi, digunakan fungi yang paling sederhana, yaitu:

$$c_i = ((\sum_{t=1}^{m} k_t p_t) - k_i p_i) \bmod \alpha$$

Dengan kata lain, c_i adalah total dari perkalian karakter plainteks kali karakter kunci selain dirinya dalam satu blok.

Untuk memperhatikan bagaimana algoritma ini bekerja, perhatikan contoh kasus berikut:

Pada sistem alfabet 27 karakter (a-z dan spasi), plainteks "**kriptografi**" akan dienkripsi dengan kunci "**kunci**". Karena panjang kunci adalah 5 karakter, maka plainteks dipartisi menjadi blok-blok berisi 5 karakter, yakni "kript", "ograf", dan "i….". Berikut proses enkripsinya pada blok pertama ("**kript**"):

(k.k+r.u+i.n+p.c+t.i) mod 27
= (10.10+17.20+8.13+15.2+19.8) mod 27
= (100+340+104+30+152) mod 27
= (19+16+23+3+17) mod 27 = 78 mod 27 = **24**

Selanjutnya setelah menemukan persamaan di atas, dapat dengan mudah diperoleh cipherteksnya, yakni:

C_1 = 24 – 19 mod 27 = 5 (f)
C_2 = 24 – 16 mod 27 = 8 (i)
C_3 = 24 – 23 mod 27 = 1 (b)
C_4 = 24 – 3 mod 27 = 21 (v)
C_5 = 24 – 17 mod 27 = 7 (h)\

Sehingga kita peroleh:
"kripto"→(enkrip "kunci")→"fibvh"

Selanjutnya kita dekripsi kembali dengan formula yang sudah diturunkan:

$$p_i = {k_i}^{-1}((m-1)^{-1}(\sum_{t=1}^{m} c_t) - c_i) \; mod \; \alpha$$

Kita cari dulu penjumlahan cipherteks, yakni:

f+i+b+v+h mod 27 = 5+8+1+21+7 mod 27 = 15

Selain itu dapat ditemukan bahwa:

$(m\text{-}1)^{-1}$ mod 27 = 4^{-1} mod 27 = 7
(karena 4x7 = 28 dan 28 mod 27=1)
Yang berarti
$(m-1)^{-1}(\sum_{t=1}^{m} c_t) \; mod \; \alpha = 7.15 \; mod \; 27 = 24$

Serta invers karakter-karakter pada kunci:
10^{-1} mod 27 = 19
20^{-1} mod 27 = 23
13^{-1} mod 27 = 25
2^{-1} mod 27 = 14
8^{-1} mod 27 = 17

Selajutnya, dapat langsung didapatkan plainteksnya, seperti ini:

P1 = 19(24-5) mod 27 = 10 (k)
P2 = 23(24-8) mod 27 = 17 (r)
P3 = 25(24-1) mod 27 = 8 (i)
P4 = 14(24-21) mod 27 = 15 (p)
P5 = 17(24-7) mod 27 = 19 (t)

Terlihat bahwa proses dekripsi menghasilkan plainteks yang sebelumnya dienkripsi

"fibvh"→(dekrip "kunci")→ "kripto"

Jadi dapat dipastikan bahwa fungsi enkripsi dan dekripsi tersebut sudah valid dan siap digunakan.

Dalam algoritma yang penulis rancang, didefinisikan:

- Agar banyak karakter merupakan bilangan prima, alfabet terdiri dari huruf-huruf, angka, dan tanda baca yang merupakan karakter ASCII 32-122 sehingga α=91. Berikut karakter-karakter tersebut:

0		16	0	32	@	48	P	64	`	80	p
1	!	17	1	33	A	49	Q	65	a	81	q
2	"	18	2	34	B	50	R	66	b	82	r
3	#	19	3	35	C	51	S	67	c	83	s
4	$	20	4	36	D	52	T	68	d	84	t
5	%	21	5	37	E	53	U	69	e	85	u
6	&	22	6	38	F	54	V	70	f	86	v
7	'	23	7	39	G	55	W	71	g	87	w
8	(	24	8	40	H	56	X	72	h	88	x
9	)	25	9	41	I	57	Y	73	i	89	y
10	*	26	:	42	J	58	Z	74	j	90	z
11	+	27	;	43	K	59	[	75	k	91	{
12	,	28	<	44	L	60	\	76	l		
13	-	29	=	45	M	61	]	77	m		
14	.	30	>	46	N	62	^	78	n		
15	/	31	?	47	O	63	_	79	o		

- Untuk α=91 = 7x13, terdapat beberapa bilangan kurang dari 91 yang tidak relatif prima, yaitu 0, 7, dan 13. Untuk itu, jika kunci mengandung karakter-karakter bernilai 0 (spasi), 7 ('), atau 13 (-) perlu diganti dulu. Dalam hal ini penulis definisikan:
  0 →1, 7→8, 13→14.
- Pada blok terakhir, bisa ditambahkan padding untuk memenuhi blok. Dalam implementasi ini, penulis tambahkan tanda-tanda titik hingga blok penuh. Meskipun plainteks seragam, berisi titik semua, cipherteks akan tetap terlihat acak karena fungsi enkripsinya melibatkan banyak variabel, meliputi semua karakter plainteks dan kunci dalam satu blok.

Setelah segala kebutuhan sudah didefiniskan, algoritma sudah siap diimplementasikan. Berikut adalah pseudocode dari algoritma ini dengan bentuk perhitungan yang paling mudah:

```
VARIABEL GLOBAL
P[0..n-1] adalah plainteks, nilainya dianggap integer
K[0..m-1] adalah kunci, nilainya dianggap integer
C[0..n-1] adalah cipherteks, nilainya dianggap integer
```

```
KONVERSI_KUNCI
for(i=0 to m-1)
  if(P[i]=0) then P[i]=1
  if(P[i]=7) then P[i]=8
  if(P[i]=13) then P[i]=14
endfor
```

```
ALGORITMA ENKRIPSI
Variabel Lokal
n_blok = round_up(n/m)
B[0..n_blok-1][0..m-1] ← partisi_blok(P,n)
Sigma_KP // ΣKP

tambahi_titik_di_akhir(B[n_blok-1])

for(a=0 to n_blok-1)
  sigma_KP←0
  for(t=0 to m-1)
    sigma_KP+=K[t]*P[a*m+t] mod 27;
  endfor
  for(t=0 to m-1)
    C[a*m+t]←sigma_KP – K[t]*P[a*m+t] mod 27;
  endfor
endfor
```

```
ALGORITMA DEKRIPSI
Variabel Lokal
n_blok = round_up(n/m)
B[0..n_blok-1][0..m-1] ← partisi_blok(C,n)
sigma_C // ΣC
mc //invers(m-1)* ΣC

/** daftar invers **/
In_M = invers(m-1)
In_K[0..m-1]

for(i=0 to m-1)
  In_K[i]= invers(K[i])
Endfor

for(a=0 to n_blok-1)
  sigma_C=0
  for(t=0 to m-1)
    sigma_C+=C[a*m+t] mod 27
  endfor
  mc = sigma_C*In_M mod 27
  for(t=0 to m-1)
    P[a*m+t]=In_K[t]*(mc-C[t])
  endfor
endfor
```

V. ANALISIS

Untuk menerapkan sistem persamaan lanjar ini ke dalam algoritma kriptografi, berikut hal-hal yang harus diperhatikan:

- ✓ Banyak elemen dalam alphabet harus terbatas dan diketahui (yakni α)
- ✓ Harus dapat dipastikan bahwa bilangan yang diinvers harus relatif prima dengan jumlah karakter dalam alphabet
- ✓ Dengan algoritma ini, kunci tidak boleh kosong. Sehingga jika kunci kosong harus diberikan penanganan khusus (misal dengan kunci default).

Dari implementasi di atas, terdapat beberapa analisis yang dilakukan penulis.

Dalam hal kemangkusan algoritma, terlihat pada algoritma enkripsi dan dekripsi, di awal dilakukan mn kali operasi assignment saat partisi. Karena keduanya melakukan operasi yang sama, dapat diabaikan dulu.

Pada algoritma enkripsi, setiap blok terjadi 2m operasi perkalian, 2m operasi penjumlahan/pengurangan, dan 2m+1 operasi assignment. Jika panjang plainteksnya adalah m, maka akan terdapat n_blok buah blok, di mana n_blok adalah m/n dibulatkan ke atas. Sehingga banyak operasi yaitu

n_blok(2m [*] + 2m [+/-] + (2m+1) [←])

Sedangkan pada algoritma dekripsi, pada tiap blok terdapat 2m+1 operasi penjumlahan/pengurangan, m+1 operasi perkalian, serta 2m+2 assignment, sehingga total operasi yaitu:

n_blok((m+1) [*] + (2m+1) [+/-] + (2m+2) [←])

Dari perbandingan banyak operasi tersebut, apakah dapat disimpulkan bahwa algoritma dekripsi justru lebih mangkus?

Tidak juga..

Karena pada algoritma dekripsi, di awal harus didefinisikan dulu invers kongruensi dari beberapa bilangan, yakni m-1 dan k_i, yang berarti dibutuhkan m+1 operasi invers. Padahal jika dijabarkan, mencari invers kongruensi tidaklah mudah. Dengan brute-force saja dibutuhkan α kali perkalian, dan dalam hal ini ada 27 perkalian untuk dicari tahu mana yang menghasilkan sisa 1 jika dibagi 27.

Dan menurut penulis, ini adalah seni dari pembalikan sistem persamaan lanjar menggunakan operator kongruensi.

Dari analisis tersebut, dapat bahwa algoritma dekripsi menjadi lebih rumit daripada enkripsi, sehingga keamanan menjadi lebih terjamin. Dan ini akan mampu membuat kriptanalis mengalami kesulitan.

Sifat enkripsi m-gram pada metode ini juga memberikan kesulitan lain bagi para kriptanalis. Yakni, setiap perubahan 1 karakter saja di suatu blok berisi m karakter, akan memberikan hasil enkripsi yang jauh berbeda karena memang semua karakter saling berpengaruh dalam fungsi enkripsi. Namun karakter pada suatu blok tidak mempengaruhi blok lain, sehingga jika satu blok ‘rusak’, blok lain masih dapat ‘diselamatkan’.

Selanjutnya keamanan dalam hal pemilihan kunci.

Berhubung karakter-karakter kunci perlu disesuaikan dulu agar relatif prima dengan α, maka bisa terjadi tumbukan (*collide*). Yakni beberapa kunci menghasilkan hasil enkripsi yang sama. Pada kasus di atas misalnya kunci "^-^ V" akan sama dengan "^.^ V" ataupun "^.^!V" karena memang pada prosesnya, '-'(13) akan dikonversi menjadi '.'(14), serta spasi(0) dikonversi menjadi '!'(1).

Namun hal ini tidak menjadi masalah dalam kriptografi klasik, karena peluang mendapatkan kunci-kunci tersebut secara acak tetap saja kecil. Seperti halnya menemukan keberuntungan menemukan kunci yang tepat. Masalah seperti ini juga sering terjadi pada beberapa algoritma lain, misalnya vigenere cipher yang 26 karakter. Bahkan kriptografi dengan konsep hashing pasti terjadi tumbukan juga, karena berapapun panjang plainteksnya, akan diubah menjadi string dengan panjang tertentu. Sesuai dengan prinsip sarang merpati (PHP: Pigeon Hole Principle):

> *"Jika ada n+1 merpati dan n sarang dan kita harus memasukkan semua merpati ke sarang, maka pasti terdapat sarang yang berisi minimal 2 merpati."*

VI. KESIMPULAN

Secara umum, penggunaan sistem persamaan lanjar dapat menghasilkan enkripsi data yang sulit dipecahkan, tetapi tidak mangkus dalam proses enkripsinya. Berikut kesimpulan yang dapat diambil dari analisis di atas:

- ✓ Secara umum, untuk menggunakan teknik membagi plainteks menjadi blok-blok berisi m karakter, lalu gunakan fungsi enkripsi

$$c_i = f(p_1,p_2,p_3,\ldots p_m, k_1,k_2,k_3,\ldots,k_m)$$

- ✓ Proses enkripsi dan dekripsi berbasis SPL sebaiknya dilakukan dengan komputasi karena memang ada banyak perhitungan, serta kesalahan hitung dapat berakibat fatal.
- ✓ Dalam pengembangannya, sistem persamaan lanjar dapat juga diterapkan dalam hashing maupun algoritma kunci public, karena sifatnya yang enkripsinya mudah tetapi dekripsinya rumit dan beresiko salah hitung.
- ✓ Sistem persamaan lanjar kurang cocok untuk kriptografi modern, karena adanya batasan bahwa banyak karakter alfabet dan nilai karakter kunci harus saling prima. Serta dibutuhkan padding di blok terakhir. Berarti jika pesan yang dienkripsi bukan teks, ada kemungkinan plainteks setelah didekripsi sedikit berbeda dengan plainteks awal.

DAFTAR PUSTAKA

[1] Munir, Rinaldi, Ir.,M.T. 2007. *Diktat Kuliah IF-5054 Kriptografi*. Bandung : Informatika ITB

[2] H. Wiryanto, Leo. *Diktat Kulah Matematika Teknik*. Bandung : Matematika ITB

[3] URL http://blog.uin-malang.ac.id/muchad/2010/06/03/penerapan-matriks-dalam-kriptografi-ilmu-pembacaan-sandi/ diakses tanggal 1 Maret 2011 jam 23.07

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, 19 Maret 2011

Hendra Hadhil Choiri
(135 08 041)