Rancangan Algoritma Kriptografi Simetri Dengan Menggunakan Derivasi Algoritma Klasik Subtitusi

Sila Wiyanti Putri [1)]
1) Program Studi Teknik Informatika ITB, Bandung 40132, email: silawp@gmail.com

Abstract *– Makalah ini membahas mengenai rancangan algoritma kriptografi kunci simetri dengan menggunakan derivasi algoritma kriptografi klasik, yaitu algoritma subtitusi abjad-tunggal dan algoritma* Caesar Cipher*. Algoritma ini diharapkan dapat menambah tingkat keamanan dari algoritma kriptografi klasik, yang sangat rentan terhadap* exhaustive key search, *pendekatan analisa frekuensi dan metode Kasiski, terutama jika pesan yang disandikan adalah pesan panjang. Selain bertujuan untuk meningkatkan keamanan, algoritma ini juga dirancang sedemikian sehingga faktor kesederhanaan dari algoritma subtitusi klasik tetap terjaga, sehingga praktis dan mudah diaplikasikan.*

Kata Kunci*: algoritma klasik, subtitusi, Caesar Cipher, subtitusi abjad-tunggal.*

1. PENDAHULUAN

Algoritma kriptografi klasik seperti subtitusi dan transposisi merupakan algoritma yang sangat sederhana. Kesederhanaan algoritma-algoritma tersebut sangat menguntungkan karena membuat proses enkripsi dan dekripsi pesan menjadi praktis dan tidak rumit. Tetapi di lain pihak, tingkat keamanan dari pesan tersandi yang dihasilkannya tidak terlalu baik. Hal ini terutama disebabkan oleh banyak dan mudahnya pendekatan-pendekatan yang dapat digunakan untuk mendekripsi pesan tersandi tersebut.

Untuk subtitusi dengan menggunakan algoritma *Caesar Cipher* misalnya, dapat mudah dapat dipecahkan dengan menggunakan *exhaustive key search* karena jumlah kuncinya sangat sedikit, yaitu hanya terdapat dua puluh enam buah kunci. Begitu juga dengan subtitusi abjad-tunggal, dengan pendekatan yang sama "hanya" dibutuhkan paling banyak 26! kali percobaan, jauh lebih sedikit dibandingkan percobaan untuk mendekripsi pesan tersandi yang dihasilkan oleh algoritma kriptografi yang lebih modern. Dekripsi dengan pendekatan analisis frekuensi juga dapat menguak pesan dengan sangat mudah, karena dengan subtitusi abjad-tunggal, setiap huruf hanya akan dipetakan menjadi satu huruf (baik lainnya maupun dirinya sendiri), sehingga sejalan dengan konsep analisis frekuensi.

Dalam perkembangannya, dengan tujuan meningkatkan tingkat keamanan sekaligus tetap mempertahankan kesederhanaan metode, dipakailah algoritma *cipher* abjad-majemuk. Tetapi metode inipun, terutama yang menggunakan subtitusi periodik pada periode tertentu, masih tetap mudah dipecahkan dengan pendekatan sederhana, seperti teknik analisis frekuensi dan metode pengulangan Kasiski. Singkat kata, metode subtitusi yang digunakan oleh algoritma kriptografi kalsik, baik *Caesar Cipher,* abjad-tunggal maupun abjad-majemuk, sangatlah mudah untuk dipecahkan terutama jika *plaintext* memiliki jumlah karakter tinggi, karena pada *plaintext* panjang sampel perulangan akan semakin besar kemungkinannya untuk ditemui, dan frekuensi kemunculan akan menjadi lebih akurat.

2. PEMBAHASAN

Algoritma yang akan diperkenalkan dapat dikatakan merupakan suatu derivasi dari algoritma klasik subtitusi. Hal ini dikarenakan algoritma ini memodifikasi dan mengkombinasikan dua buah algoritma klasik subtitusi, yaitu algoritma subtitusi abjad-tunggal dan algoritma *Caesar Cipher*.

Algoritma subtitusi abjad-tunggal dipilih karena memiliki derajat nilai acak (*random scale*) yang dimilikinya cukup tinggi. Pada algoritma subtitusi abjad-tunggal, setiap karakter dipetakan menjadi karakter lain atau dirinya sendiri tanpa memiliki formulasi, pola, atau perhitungan tertentu. Terdapat 26! kombinasi kunci yang mungkin digunakan dalam algoritma subtitusi abjad-tunggal.

Sedangkan algoritma *Caesar Cipher* dipilih karena merupakan algoritma subtitusi klasok yang paling sederhana. Algoritma *Caesar Cipher* melakukan penyulihan setiap karakter dengan karakter lain yang berjarak *n* dari dirinya dalam susunan abjad. Dengan demikian, algoritma *Caesar Cipher* hanya membutuhkan sebuah bilangan sebagai repsentasi dari jarak pergeseran tersebut sebagai kunci sehingga algoritma ini sangatlah efisien dari segi ukuran kunci yang dibutuhkan.

Jika kedua algoritma klasik tersebut dimodifikasi dan dikombinasikan dengan baik, diharapkan akan didapatkan sebuah algoritma subtitusi yang lebih aman namun tetap sederhana dalam hal pengaplikasiannya.

Untuk selanjutnya, akan dijelaskan tahapan-tahapan dalam melakukan proses enkripsi dan dekripsi dengan menggunakan rancangan algoritma ini.

Langkah pertama dari algoritma ini adalah membangkitkan sebuah kunci yang serupa dengan kunci algoritma subtitusi abjad-tunggal, yaitu berupa pemetaan satu-satu setiap karakter menjadi karakter lain atau dirinya sendiri secara acak. Dari pemetaan tersebut, dibangkitkan sebuah pemetaan lainnya dengan cara melakukan pergeseran sebanyak *C* karakter dari pemetaan sebelumnya (seperti pada algoritma subtitusi *Caesar Cipher*). Hal ini dilakukan hingga akhirnya terdapat *N* buah pemetaan. Pemetaan-pemetaan ini kemudian diberi indeks berupa bilangan (dimulai dari angka satu) sesuai dengan urutan pembangkitannya. Untuk lebih jelasnya mengenai tahapan ini dapat dilihat pada ilustrasi di bawah. Pada ilustrasi ini, digunakan $N = 3$, $C = 4$.

$N_1 =$ $N_2 =$ $N_3 =$

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

Gambar 1: Pemetaan yang dibangkitkan dengan N_1 dibangkitkan secara acak, $N = 3$, dan $C = 4$

Pemetaan-pemetaan tersebut, bersama dengan nilai *N* dan *C*, selanjutnya akan bertindak sebagai kunci dalam proses enkripsi deksipsi. Pemetaan digunakan secara berurutan berdasarkan nomor indeksnya untuk setiap karakter yang akan dienkripsi maupun didekripsi.

Sebagai contoh, akan dilakukan enkripsi kalimat "SEBELUM LEBARAN BERMAAF MAAFAN" dengan menggunakan pemetaan-pemetaan yang sudah dibangkitkan pada ilustrasi Gambar 1 ($N = 3$, $C = 4$). Proses enkripsi dapat dilihat pada ilustrasi di bawah.

Tabel 1. Proses enkripsi dengan menggunakan pemetaan berurut (Gambar 1)

Plaintext	**N_x**	**Ciphertext**
S	1	G
E	2	P
B	3	U
E	1	W
L	2	M
U	3	L
M	1	C
L	2	M
E	3	C
B	1	I
A	2	W
R	3	O
A	1	S
N	2	Z
B	3	U
E	1	W
R	2	Q
M	3	V
A	1	S
A	2	W
F	3	J
M	1	C
A	2	W
A	3	P
F	1	Y
A	2	W
N	3	Q

Dari proses enkripsi yang dilakukan di atas, didapatkan *ciphertext* berupa *string* "GPUWMLC MCIWOSZ UWQVSWJ CWPYWQ". Jika spasi dihilangkan, maka hasilnya akan menjadi "GPUWMLCMCIWOSZUWQVSWJ CWPYWQ".

Untuk melakukan dekripsi dari *ciphertext* yang dihasilkan oleh algoritma ini, informasi yang harus diketahui oleh pihak yang akan melakukan dekripsi supaya proses dekripsi dapat berjalan dengan baik adalah:

1. N_1 , yaitu set pemetaan pertama yang merupakan hasil dari pembangkitan secara acak
2. N , yaitu jumlah pemetaan yang digunakan dalam proses enkripsi.

3. C , yaitu banyaknya pergeseran yang dilakukan untuk membangkitkan pemetaan N_{x+1} dari pemetaan N_x.

Sehingga yang berlaku sebagai kunci pada algoritma kunci-simetri ini adalah (N_1, N, C).

Jika seluruh kunci yang diperlukan diketahui, pihak yang akan melakukan dekripsi harus membangkitkan pemetaan-pemetaan sesuai dengan informasi yang tertuang dalam kunci tersebut. Pembangkitan pemetaan dilakukan dengan cara yang sama dengan pembangkitan saat melakukan enkripsi, yaitu dengan cara melakukan pergeseran sejauh C sebanyak N kali (lihat Gambar 1).

Setelah itu, tentukan indeks pemetaan yang akan digunakan oleh tiap karakter dalam *ciphertext*. Hal ini dilakukan dengan cara menuliskan nomor indeks pemetaan dari 1 sampai N secara berurutan dan berulang-ulang dari awal sampai akhir *ciphertext*. Kemudian untuk setiap karakter pada *ciphertext*, lakukan pemetaan terbalik untuk mengetahui karakter yang berkenaan dengan karakter *ciphertext* tersebut pada pemetaan yang memiliki nomor indeks yang sesuai (lihat Tabel 1).

Setelah setiap karakter yang didapatkan dari pemetaan terbalik *ciphertext* dirangkai, maka akan didapatkan kembali *plaintext* yaitu "SEBELUMLEBARAN BERMAAFMAAFAN". Langkah terakhir adalah menambahkan spasi sehingga *plaintext* memiliki makna dan lebih mudah terbaca. Pada akhirnya, *plaintext* yang didapatkan akan menjadi "SEBELUM LEBARAN BERMAAF MAAFAN".

3. ANALISIS HASIL

Dari pembahasan algoritma dan proses enkripsi dekripsi yang dijabarkan pada bagian sebelumnya, dapat dilakukan analisis dari berbagai segi. Dari segi keamanan, algoritma ini dapat dikatakan lebih aman dari algoritma-algoritma klasik subtitusi yang sudah umum dikenal, seperti algoritma subtitusi abjad-tunggal dan algoritma *Caesar Cipher*. Hal ini terutama disebabkan karena kunci yang digunakan lebih dari satu, dan memiliki nilai acak.

Algoritma subtitusi periodik sebenarnya juga memiliki lebih dari satu "kunci", karena setiap karakter pada *plaintext* dipetakan dengan cara yang berbeda-beda, yaitu dengan melakukan pergeseran sebanyak indeks karakter pada kunci yang bersesuaian dengannya. Tetapi hal ini sangat rentan karena masing-masing dari pemetaan tersebut memiliki urutan yang sama, yaitu urutan abjad standar yang dimulai dari karakter A dan diakhiri dengan karakter Z, sehingga sangat mudah dipecahkan dengan menggunakan metode Kasiski untuk menentukan panjang kunci, diikuti dengan teknik analisis frekuensi untuk menentukan kuncinya.

Dengan algoritma yang diperkenalkan di bagian sebelumnya, dapat dilihat bahwa pada *ciphertext* tidak terdapat perulangan pada bagian yang seharusnya berulang, yaitu kata "MAAF". Kata "MAAF" yang pertama dipetakan menjadi "VSWJ", sedangkan yang kedua menjadi "CWPY". Perulangan pada *ciphertext* justru terdapat pada string "WQ", padahal "WQ" pertama merupakan bentuk tersandi dari "ER", sedangkan yang kedua adalah bentuk tersandi dari "AN". Karena perulangan yang terjadi pada *ciphertext* tidak bersesuaian dengan perulangan pada *plaintext*, maka metode Kasiski akan cukup sulit diterapkan untuk mengetahui banyaknya pemetaan yang digunakan. Hal ini diharapkan akan mempersulit penyerang dalam mendekripsi *plaintext*.

Andaikata terjadi kasus dimana perulangan yang terjadi cukup sesuai sehingga metode Kasiski dapat diterapkan, algoritma inipun masih lebih aman jika dibandingkan dengan algoritma subtitusi periodik, karena pada algoritma subtitusi periodik, penyerang cukup menggunakan teknik analisis frekuensi terhadap satu huruf untuk mendapatkan nilai pergeseran yang digunakan, setelah itu, huruf-huruf lainnya akan dapat diketahui. Sedangkan pada algoritma ini, karena susunan huruf yang digunakan adalah susunan acak, maka penyerang harus melakukan teknik analisis frekuensi untuk setiap huruf yang ada. Itupun dengan asumsi jumlah pemetaan yang didapatnya dengan metode Kasiski benar, jika tidak, proses dekripsi akan menjadi sangat rumit.

Nilai keamanan dari algoritma ini juga dapat ditingkatkan dengan metode pendistribusian kuncinya. Karena algoritma ini memerlukan ketiga elemen N_1 , N, C untuk melakukan dekripsi, maka akan lebih aman jika ketiga elemen kunci tersebut didistribusikan secara terpisah. Dengan mendistribusikannya secara terpisah, jika sebagian dari kunci terungkap oleh pihak yang tidak berkepentingan, maka pihak tersebut tetap tidak akan memiliki cukup informasi untuk melakukan dekripsi terdapat *ciphertext*.

Hal ini juga dapat digunakan bila menginginkan suatu pesan hanya dapat diungkap hanya jika setiap pihak yang berkepentingan hadir. Contoh sederhananya misalnya pada kasus pembagian warisan dimana surat wasiat yang menerangkan detail pembagian harta dienkripsi dengan algoritma ini. Jika ketiga elemen kunci dibagikan secara terpisah kepada tiga orang anggota keluarga, maka surat wasiat tersebut tidak akan dapat didekripsi dan dibacakan sebelum ketiga anggota keluarga tersebut hadir, atau menyetujui untuk memberikan elemen kunci yang menjadi bagiannya.

Satu hal yang menjadi titik lemah dari algoritma ini adalah ukuran kuncinya. Dibutuhkan dua puluh enam karakter untuk merepresentasikan susunan acak yang

digunakan sebagai N_1 , minimal satu karakter untuk N, dan minimal satu karakter untuk C. Sehingga total ukuran kunci minimal adalah dua puluh delapan karakter. Ukuran ini lebih besar dari ukuran kunci yang dibutuhkan oleh algoritma *Cipher Caesar* (minimal satu karakter), algoritma subtitusi periodik (minimal satu karakter, walaupun sebaiknya menggunakan kunci dengan jumlah karakter lebih banyak), bahkan algoritma subtitusi abjad-tunggal (dua puluh enam karakter, untuk merepresentasikan susunan acak yang digunakan sebagai pemetaan).

Faktor kunci tersebut membuat algoritma ini kurang baik performansinya jika digunakan untuk mengenkripsi pesan yang singkat, yaitu yang jumlah karakternya lebih kecil dari jumlah karakter minimal kunci, dua puluh delapan karakter. Jika pesan yang ingin dienkripsi singkat, dan ukuran kunci tidak menjadi masalah, maka sebaiknya pesan tersebut dienkripsi dengan menggunakan algoritma *one-time pad* yang merupakan *unbreakable cipher* dengan panjang kunci sama dengan *plaintext* dan pemetaan yang digunakan dibangkitkan secara acak.

Untuk *plaintext* yang cukup panjang, algoritma ini cukup efektif untuk digunakan, karena walaupun ukuran kuncinya lebih besar daripada algoritma-algoritma subtitusi, namun tingkat keamanan dari pesan tersandi yang dihasilkan juga lebih tinggi dari algoritma-algoritma klasik subtitusi.

4. KESIMPULAN

Algoritma kriptografi klasik yang berdasarkan pada prinsip subtitusi, seperti algoritma *Caesar Cipher* dan algoritma subtitusi abjad tunggal sangatlah baik jika dipandang dari segi kesederhanaan dan kemudahan dalam mengaplikasikannya, namun memiliki tingkat keamanan yang rendah karena mudah dipecahkan dengan pendekatan-pendekatan seperti metode Kasiski dan teknik analisis frekuensi.

Algoritma yang diperkenalkan pada makalah ini merupakan algoritma kriptografi kunci-simetri yang dihasilkan dengan melakukan modifikasi dan kombinasi dari dua algoritma kriptografi klasik subtitusi, yaitu algoritma *Caesar Cipher* dan algoritma subtitusi abjad-tunggal.

Dari segi keamanan, algoritma ini dapat dikatakan lebih aman dari algoritma-algoritma klasik subtitusi. Hal ini dikarenakan perulangan yang terjadi pada *ciphertext* yang dihasilkan oleh algoritma ini kecil kemungkinannya bersesuaian dengan perulangan pada *plaintext*. *String* yang berulang pada *plaintext* dapat tidak terlihat berulang pada *ciphertext*, sedangkan *string* yang muncul berulang pada *ciphertext* mungkin saja sebenarnya bukan perulangan pada *plaintext*.

Untuk melakukan enkripsi dan dekripsi menggunakan algoritma ini, dibutuhkantiga komponen terpisah sebagai kunci, yaitu. N_1 sebagai set pemetaan pertama hasil dari pembangkitan secara acak, N ,jumlah pemetaan yang digunakan dalam proses enkripsi, dan C , yaitu banyaknya pergeseran yang dilakukan untuk membangkitkan pemetaan N_{x+1} dari pemetaan N_x. Ketiga elemen ini dapat didistribusikan secara terpisah untuk meningkatkan tingkat keamanan, karena proses dekripsi akan menjadi rumit jika tidak memiliki ketiga elemen kunci tersebut.

Ukuran kunci algoritma ini, yaitu dua puluh delapan karakter, lebih besar dari ukuran kunci yang dibutuhkan oleh algoritma *Cipher Caesar*, algoritma subtitusi periodik, dan algoritma subtitusi abjad-tunggal. Hal ini menjadikan algoritma ini tidak sesuai untuk enkrisi pesan singkat yang ukurannya lebih kecil dari dua puluh delapan karakter. Karena itu algoritma ini diperuntukkan untuk enkripsi pesan yang panjang (setidaknya lebih besar dari panjang kunci).

DAFTAR PUSTAKA

Munir, Rinaldi, “Kriptografi”, Institut Teknologi Bandung, 2006.