Algoritma RSA dengan *Logistic Map* sebagai Pembangkit Bilangan Acak untuk Pembentukan Kunci

Yohanes Jhouma Parulian N / 13515053
Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung. Jl. Ganesha 10 Bandung 40132, Indonesia
13515053@std.stei.itb.ac.id

***Abstract*—Algoritma RSA merupaka algoritma kunci-publik yang paling terkenal dan paling banyak aplikasinya. Kekuatan algoritma ini berada pada sulitnya untuk mengkomputasi faktor prima pada suatu bilangan besar. Namun hanya kerana menggunakan satu pasang kunci saja untuk mengenkripsi dan dekripsi keamanan dari algoritma ini bukanlah yang terbaik. Untuk meningkatkan keaman algoritma dapat digunakan kunci dengan jumlah lebih dari satu pasang, sehingga dapat memenuhi kedua prinsip Shannon. *Logistic map* dapat digunakan untuk membangkitan bilangan-bilangan prima yang digunakna untuk membentuk pasangan-pasangan kunci untuk digunakna secara siklik untuk mengenkripsi dan dekripsi. Untuk pembangkitan bilangan prima *seed* digunakan sebagai kunci pembangkit.s**

Keywords—RSA;Logistic Map; kunci; enkripsi; dekripsi

I. PENDEHALUAN

Kriptografi sudah digunakan sejak lama untuk menjamin keamanan sebuah informasi. Secara garis besar kriptografi dibagi menjadi dua yaitu kriptografi klasik dan modern. Algoritma kriptografi klasik dapat dikenali dengan berbasis huruf alfhabet, dapat dilakukan tanpa menggunakan computer karena algoritma-algoritma ini diciptakan sebelum adanya komputer, dan merupakan algoritma kunci simetri [1]. Sedangkan algortima kriptografi modern adalah algoritma yang berkembang dengan adanya komputer, dan basis utama operasinya adalah dalam bit.

Hingga akhir dekade 1970 algoritma yang berkembang adalah algoritma kunci-simetri, dimana kunci yang digunakan untuk mengenkripsi dan mendkripsi merupakan kunci yang sama. Kesulitan dari penggunaan algoritma kunci-simetri adalah pembagian kunci kepada penerima pesan harus melalui jalur yang benar-benar aman. Karena kesulitan ini muncul ide untuk mengembangkan algoritma yang menggunakan kunci berbeda untuk enkripsi dan dekripsi atau yang disebut dengan algoritma kunci-asimetri atau algoritma kunci-publik. Salah satu algoritma kunci publik adalah algoritma RSA.

Algoritma RSA merupakan algoritma kunci-publik yang paling terkenal dan paling banyak aplikasinya [2]. Keamanan algoritma ini mengandalkan kesuliatan untuk melakukan faktorisasi bilangan prima besar. Namun bila melakukan enkripsi pada tiap bit algoritma ini masih rentan dengan serangan kripanalisis yaitu serangan analisis frekuensi, karena melakukan enkripsi pada byte yang sama akan menghasilkan cipher teks yang sama pula.

Karena alasan tersebut penulis mencoba merancang sebuah algoritma dengan dasar algoritma RSA untuk melakukan enkripsi dan dekripsi. Logistic map akan digunakan untuk membangkitkan bilangan prima untuk pembentukan kunci untuk algoritma RSA tiap bytenya.

II. LANDASAN TEORI

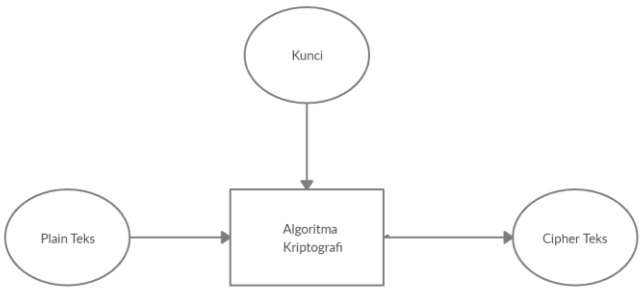
A. Kriptografi

Menurut kamus *oxford* kriptografi didefinisikan sebagai “seni menulis dan memecahkan sandi.” Namun saat ini kriptografi bukan hanya tentang memecahkan dan menulis sandi, melainkan juga melibatkan pemasitian integritas informasi, teknik untuk berbagi kunci, protokol untuk autentikasi user, pemilihan dan lelang elektronik, uang digital, dan masih banyak lagi [3].

Sebelum adanya komputer kriptografi dikerjakan dengan menggunakan kertas dan pensil. Sehingga kriptografi klasik umumnya berkutat pada huruf alfabet dan juga operasi yang digunakan juga operasi-operasi yang sederhana seperti rotasi, substitusi, dan permutasi. Setelah adanya komputer operasi kriptografi bergeser yang awalnya berdasarkan alfabet kini berdasarkan operasi bit dan juga perhitungan yang digunakan menjadi lebih rumit.

Secara garis besar algoritma kriptografi adalah mengubah suatu plain teks menjadi sebuah cipher teks.

Gambar 1 Diagram Proses Kriptografi



B. Prinsip Shannon

Sebuah cipherteks dari algoritma kriptografi belum tentu pasti aman. Serangan terhadap sebuah algoritma kriptografi disebut sebagai kripanalisis, yaitu metode untuk memecahkan cipher teks menjadi plain teks tanpa mengetahui kunci. Untuk menghindari serangan ini Shannon merumuskan dua buah prinsip yang harus dimiliki sebuah algoritma kriptografi yaitu *confusion* dan *diffusion*.

1) Confusion

Pada dasarnya prinsip *confusion* ini mengharuskan sebuah algoritma kriptografi untuk mampu menyembunyikan hubungan antara kunci dan cipher teks. Prinsip ini mempersulit untuk menemukan kunci dengan informasi cipher teks. Selain itu prinsip ini juga membuat perubahan kecil pada kunci berakibat perubahan besar pada cipher teks dari plain teks yang sama.

2) Diffusion

Prinsip ini menyatakan algoritma kriptografi harus mampu menyembunyikan hubungan antara plain teks dan cipher teks. Dengan kata lain bila ada sedikit perubahan pada plain teks maka cipher teks akan berubah sepenuhnya.

C. Algoritma RSA

Algoritma RSA merupakan salah satu algoritma kriptografi kunci-asimetri, yaitu kunci yang digunakan untuk mengenkripsi dan dekripsi berbeda. Kunci yang digunakan untuk mengenkripsi adalah kunci publik secara bebas diketahui, sedangkan kunci yang digunakan untuk mendekripsi adalah kunci private, dimana hanya orang yang berhak menerima informasi yang memegangnya. Keamanan dari algoritma ini mengandalkan sulitnya untuk melakukan faktorisasi prima pada bilangan yang besar.

Properti yang dimiliki algoritma ini adalah sebagai berikut.

- *p* dan *q* (rahasia) dua buah bilangan prima, pada prakteknya bilangan prima yang digunakan adalah bilangan prima yang sangat besar
- $n = p*q$ (tidak rahasia)
- $phi(n) = (p-1)*(q-1)$ (rahasia) yaitu bilangan totient yang menggambarkan jumlah bilangan relatif prima terhadap n.
- *e* sebagai kunci enkripsi (tidak rahasia), dengan syarat $GCD(e,phi(n)) = 1$
- *d* sebagai kunci dekripsi (rahasia), yang dapat dihitung dengan $d = e^{-1} \bmod (phi(n))$
- *m* merupakan plain teks (rahasia)
- *c* cipher teks (tidak rahasia)

Proses enkripsi algoritma RSA adalah sebagai berikut

$$c = E(m) = m^e \bmod (n)$$

Sedangakan proses dekripsinya adalah sebagai berikut

$$m = D(c) = c^d \bmod (n)$$

D. Logistic Map

Logistic map merupakan sistem pemetaan polinomial dengan derajat 2, kehebatan hal ini adalah dapat memunculkan sifata yang kompleks dan *chaotic* dengan persamaan yang sederhana, peta ini dipopulerkan oleh jurnal yang ditulis oleh Robert May [1]. Persamaan matematis dari logistic map ini adalah sebagai berikut:

$$X_{n+1} = \mu X_n(1-X_n)$$

Dimana terdapat satu parameter untuk persamaan ini yaitu μ. Nilai μ harus berada diantara nilai 0 dan 4, dan semakin tinggi nilainya semaki *chaotic* juga sifat dari *logistic map*. Sehingga untuk mendapatkan nilai acak maksimum μ yang digunakan adalah 4.0.

Gambar 2. Diagram bifurcation logistic map

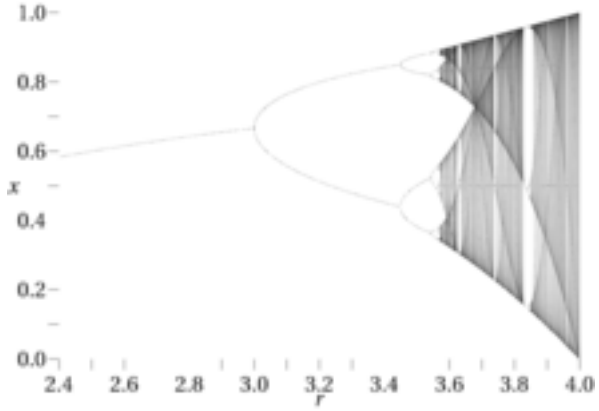


Diagram diatas merupakan diagram bifurication dari logistic map, diagram tersebut menggambarkan hubungan parameter μ dengan nilai *x*. Dapat dilihat pada diagram dengan nilai μ makin tinggi sifat logistic map semakin *chaotic*.

III. ALGORITMA

Algoritma yang dibuat terdiri dari dua buah komponen yaitu, komponen RSA dan komponen *Logistic map*. Komponen RSA merupakan algoritma yang digunakan untuk mengenkripsi dan mendekripsi suatu pesan. Sedangkan komponen *logistic map* merupakan komponen yang digunakan untuk membangkitkan bilangan acak, yang nantinya digunakaan untuk membangkitkan kunci yang dapat digunakan untuk algoritma RSA.

Program akan menerima sebuah file untuk dienkripsi. Untuk menenkripsi file tersebut akan digunakan byte-byte dari file sebagai masukan untuk fungsi enkripsi RSA. Akan digunakan *k* buah kunci yang akan digunakan secara siklik, dimana *k* adalah salah satu parameter yang akan digunakan untuk membangkitkan kunci. Begitu pula dengan dekripsi yang akan digunakan adalah *k* buah kunci yang juga dibangkitkan dari *logistic map*.

Untuk membangkitkan kunci yang digunakan algoritma RSA, akan digunakan bilangan yang dibangkitkan oleh algoritma *logistic map*. Setiap bilangan acak yang dibangkitkan akan dipetakan menjadi sebuah bilangan prima yang nantinya bilangan prima (*p* dan *q*) tersebut akan digunakan untuk menjadi pembangkit kunci yang akan digunakan. Dalam algoritma ini kunci yang dibangkitkan bukan hanya satu melainkan ada *n* buah kunci, dimana *k* akan menjadi masukan untuk program pembangkitan kunci.

Proses pembentukan kunci untuk RSA akan menghasilkan kunci-publik dan kunci-privat. Bila pada RSA standard kunci yang public yang dihasilkan sepasang bilangan *(e,n)*, pada algoritma ini kunci yang dihasilkan adalah *k* pasang bilangan tersebut. Sama seperti kunci-publik, kunci-privat juga akan merupakan *k* buah pasangan bilangan *(d,n).* Selanjutnya kunci-kunci tersebut disimpan dalam file eksternal *public.pub* untuk kunci-publik dan *private.pri* untuk kunci-privat.

Untuk melakukan enkripsi dan dekripsi, program akan menerima masukan berupa file biner dan kunci, *public.pub* untuk enkripsi dan *private.pri* untuk dekripsi. Proses dekripsi dan enkripsi sesuai dengan algoritma RSA yang dijelaskan sebelumnya, hanya saja penggunaan kunci akan dilakukan secara siklik.

Berikut adalah fungsi dalam pseudocode yang digunakan untuk membangkitkan *k* bilangan acak dengan algoritma *logistic map*.

```
Input nu
Input x0
Input k
Initiate empty array X

for i in range(k):
    X[i] = nu*x0*(1-x0)
    x0 = X[i]

output X
```

Berikut adalah pseudocode yang digunakan untuk membangkitkan kunci-publik dan kunci-privat.

```
Input k
Input prime_list
Initate empty array pri, pub

X = generate_random_number(2*k)
i = 1
While i < 2*k:
   pri[i], pub[i] = generate_key(X[i], X[i+1])

save pri to private.pri
save pub to public.pub
```

Dan berikut adalah pseudocode algoritma yang digunakan untuk mengenkripsi dan mendekripsi suatu file

```
Input file
Input key
Input k
Initiate empty array cipher
i = 1
for byte in file:
   add to cipher encryptbyte(byte,key[i mod k])
   i = i +1

output cipher
```

Untuk melakukan dekripsi hal yang perlu dilakukan adalah mengubah fungsi enkripsi menjadi dekripsi.

IV. EKSPERIMEN DAN ANALISIS

Eksperimen dengan algoritma ini akan dilakukan dengan mengimplementasikan algoritma pada bahasa pemrograman Python. Percobaan akan dilakukan dengna membandingkan RSA standard dengan algoritma yang coba dikembangkan. Dengan hal-hal yang coba diubah adalah parameter seperti jumlah kunci dan *seed* yang digunakan untuk mengenerasi bilangan acak.

Percobaan pertama dilakukan dengan mencoba mengenkripsi sebuah file berekstensi txt yang berisi:

```
Ini merupakan pesan rahasia.
```

Kemudian algoritma juga akan dicoba pada sebuah file txt dengan ukuran 767 KB.

Sebelum melakukan enkripsi perlu dibangkitkan dulu pasangan kunci sebanyak *k* berikut adalah waktu pembangkitan *k* buah kunci

k	**Waktu**
5	0.00498
10	0.00495
100	0.02991

Tabel 1 Waktu pembangkitan kunci

Dapat dilihat bahwa waktu pembangkitan kunci memiliki kompleksitas yang lanjar. Pembangkitan kunci tidak memerlukan waktu yang lama.

Berikut adalah table perbandingan waktu enkripsi RSA standard dengan algoritma yang dikembangkan dengan ukuran kunci yang berbeda-beda, untuk mengenkripsi file pertama.

Algoritma	**Waktu**
RSA Enkripsi	0.07533
RSA Dekripsi	0.00145
(k=5) Enkripsi	0.00103
(k=5) Dekripsi	0.00143

(k=10) Enkripsi	0.0009973
(k=10) Dekripsi	0.0009975
(k=100) Enkripsi	0.000955
(k=100) Dekripsi	0.001993

Berikut adalah table yang menunjukan waktu eksekusi program pada file kedua dengan ukuran 767 KB.

K	**Waktu Enkripsi**	**Waktu Dekripsi**
5	7.4127	4.6859
10	4.9000	5.1754
100	5.5561	5.5865

Dapat dilihat bahwa jumlah kunci yang digunakan tidak menambah waktu komputasi yang diperlukan untuk mengkomputasi cipher teks maupun plain teks. Dapat disimpulkan kompleksitas komputasi RSA standard dan juga algoritma yang dikembangkan sama, hanya saja yan berbeda adalah kompleksitas mengenerasi kunci, karena pada RSA standard kunci sudah ditentukan sebelumnya pada algoritma ini yang menjadi kunci adalah *seed* yang digunakan untuk mengenerasi bilangan prima.

Percobaan berikutnya yang dilakukan adalah mengubah seed penggenerasi bilangan prima untuk melihat kemampuan *confusion* dari algoritma ini. Berikut adalah hasil enkripsi file pertama dengan parameter $k = 5$ dan seed yang digunakan adalah 1.

```
97070254
120774126
454212705
1101594606
453165916
2667795
…
```

Dan berikut adalah enkripsi file yang sama dengan mengubah seed menjadi 2.

```
71817777
28794097
353141559
1164778879
3718740182
38088896
…
```

Dapat dilihat bahwa berubahan sedikit pada key algoritma ini yaitu *seed* untuk membangkitkan bilangan random, terjadi perubahan seluruhnya pada cipher teks dari plain teks yang sama. Dapat dikatakan algoritma ini dapat memenuhi salah satu prinsip Shannon yaitu prinsip *confusion*. Sayangnya karena dasar dari algoritma ini adalah algoritma RSA, tingakat *diffusion* dari algoritma ini mirip dengan RSA, namun karena penggunaan kunci yang berbeda pada tiap bytenya maka hasil enkripsi tidak dapat diserang dengan menggunakan analisis frekuensi, karena byte yang sama belum tentu menghasilkan cipher teks yang sama.

V. KESIMPULAN

Penggunaan *logistic map* dan algoritma RSA untuk membangkitkan bilangan prima acak dan juga mengenkripsi sebuah file, dapat dikatakan algoritma yang memenuhi dua prinsip Shannon. Dimana prinsip *confusion* terjadi karena perubahan sedikit nilai key mengubah keseluruhan cipher teks dari plain teks yang sama. Prinsip *diffusion* dicapai dikarenakan penggunaan kunci yang berbeda-beda untuk satu buah file sehingga byte yang sama belum tentu menjadi cipher teks yang sama, karena hal ini analisis frekuensi tidak dapat dilakukan untuk memecahkan cipherteks ini.

Sayang algoritma ini memiliki kekurangan yaitu memori key yang lebih besar daripada key yang dibutuhkan RSA standard, selain itu pula karena penggunaan kunci yang siklik masih dapat dianalisa dengan menggunakan metode-metode yang lebih muktahir. Meski begitu algoritma ini masih dapat dikatakan lebih aman daripada algoritma RSA standard yang mengkripsi byte-byte sebuah file dengan kunci yang sama.

REFERENSI

[1] R. Munir, "Algoritma RSA Bahan Kuliah IF4020 Kriptografi," Bandung, 2020.

[2] R. Munir, "Kriptografi Klasik (Bagian 1)," Bandung, 2020.

[3] J. Katz and Y. Lindell, Introduction to Modern Crypthography, CRC Press, 2014.

[4] R. May, "Simple mathematical models with very complicated dynamics," *Nature*, vol. 261, no. 5560, pp. 459-467, 1976.