BAB III

METODOLOGI PENELITIAN

3.1 Analisis Algoritma Kriptografi RSA

Algoritma kriptografi RSA adalah algoritma untuk keamanan data yang ditemukan oleh Rivest, Shamir dan Adleman (RSA) pada tahun 1977-1978.

Penerapan Algoritma kriptografi pada tabel di suatu basis data merupakan salah satu cara untuk mengamankan *record* yang tersimpan di dalam basis data. Algoritma kriptografi asimetris RSA tidak mudah untuk diimplementasikan pada Oracle 10g karena algoritma RSA sendiri belum didukung secara native, sehingga pengimplentasian algoritma RSA harus *create coding* secara manual.

Menurut M. Zaki Riyanto dan Ardhi Ardhian (*Kriptografi Kunci Publik : Sandi RSA*, 2008) : Keamanan sandi RSA terletak pada sulitnya memfaktorkan bilangan yang besar.

Selain sulitnya memfaktorkan bilangan yang besar pada pembentukan kunci pada RSA, RSA menawarkan kemampuan untuk mengendalikan panjang kunci yang digunakan oleh penggunanya sehingga dapat menyesuaikan dengan kebutuhan pengguna.

3.2 Langkah Kerja Algoritma RSA

Skema algoritma kriptografi kunci publik (asimetris) RSA terdiri dari tiga proses yaitu pembentukan kunci, proses enkripsi, dan proses dekripsi.

A. Algoritma Pembentukan Kunci :

1. Menentukan 2 buah bilangan prima acak dan dirahasiakan, dimisalkan kedua bilangan prima tersebut diwakilkan oleh variabel p dan q. Dengan syarat nilai p ≠ nilai q.

   Semakin besar bilangan prima p dan q, maka kemungkinan untuk dilakukannya faktorisasi dengan metode *brute force* guna mencari nilai p dan nilai q nantinya akan semakin kecil.

2. Menentukan (RSA) modulus yang berfungsi sebagai salah satu pasangan kunci publik maupun kunci rahasia. (RSA) modulus di wakilkan dengan variabel n. Nilai n didapatkan dari perkalian dua buah bilangan prima yang ditentukan pada langkah sebelumnya, yang diwakilkan oleh variabel p dan variabel q. Secara sistematis, formula mencari nilai (RSA) modulus adalah sebagai berikut :

   $$n = p * q$$

3. Langkah selanjutnya adalah menentukan (RSA) *enciphering exponent* yang nantinya digunakan sebagai pasangan kunci publik bersama (RSA) modulus. (RSA) *enciphering exponent* diwakilkan oleh variabel e. Formula untuk menentukan nilai e sebagai berikut :

   $$gcd(e,m) = 1$$

   dengan syarat 1 < e < m dan e adalah bilangan prima.

   gcd adalah akronim dari *greatest common divisor* atau faktor persekutuan terbesar (FPB). Bila gcd atau FPB dari dua buah bilangan bernilai = 1, misalnya gcd(e,m)=1, artinya nilai e merupakan bilangan yang relatif

prima terhadap nilai m, sehingga nilai e bisa digunakan untuk menjadi pasangan kunci publik bersama (RSA) modulus.

Formula penentuan nilai (RSA) *enciphering exponent* yang disebutkan sebelumnya hanya bisa dijalankan bila nilai m telah ditentukan. Formula untuk mencari nilai m sendiri adalah :

$$m = (p-1) * (q-1)$$

4. Langkah selanjutnya adalah menentukan nilai (RSA) *deciphering exponent* yang diwakilkan oleh variabel (d), (RSA) *deciphering exponent* berfungsi sebagai pasangan (RSA) modulus dalam pembentukan kunci rahasia (n,d). Formula mencari nilai d adalah sebagai berikut :

$$d = e^{-1} \bmod m$$

Nilai dari vaiabel e dan variabel m adalah nilai yang didapat dari langkah sebelumnya dalam pembentukan (RSA) *enciphering exponent*.

Dengan syarat $1 < d < m$.

Dalam praktiknya, rumus baku untuk mencari nilai d diatas dijabarkan lebih lanjut sebagai berikut :

$d = e^{-1} \bmod m$

$e * d = 1 * (\bmod m)$

$e * d = 1 + (k * m)$

$$e * d = 1 + (k * m)$$

Variabel k adalah bilangan integer dengan nilai sembarang yang diharapkan menghasilkan d yang bernilai integer juga karena nilai d yang akan diambil jika d = integer.

5. Nilai variabel **d, p, q** dan **m** adalah nilai yang harus dirahasiakan.

   Nilai variabel **n, e** adalah nilai yang boleh dipublikasikan.

   Pasangan **(n,e)** merupakan kunci publik yang boleh dipublikasikan

   Pasangan **(n,d)** merupakan kunci rahasia yang harus dirahasiakan

B. Algoritma Enkripsi

Untuk melakukan Enkripsi menggunakan kunci publik (n,e) digunakan formula enkripsi sebagai berikut :

$C = M^e \bmod n$

Dimana :

C = *Cipher Text* / teks yang telah dienkripsi

M = *Plain Text* / Pesan asli / Teks Terang

e = *Enciphering Exponent*

n = (RSA) modulus

C. Algoritma Dekripsi

Untuk melakukan Dekripsi suatu *cipher Text* yang telah dienkrip sebelumnya, digunakan pasangan kunci rahasia (n,d), formula dekripsi yang digunakan sebagai berikut :

Rumus : $M = C^d \bmod n$

Dimana :

M = *Plain Text* / Pesan Asli / Teks Terang

C = *Cipher Text* / teks yang telah dienkrip

d = *Deciphering Exponent*

n = (RSA) modulus

3.3 Contoh Kasus Enkripsi

3.3.1 (RSA) Modulus 2 Digit

Berikut ini merupakan contoh proses enkripsi dan dekripsi menggunakan algoritma RSA dengan 2 digit (RSA) modulus :

1. Langkah awal algoritma RSA adalah menentukan kunci publik dan kunci rahasia.
   a) Menentukan nilai variabel p dan q yang merupakan bilangan prima acak dan dirahasiakan nantinya, dimana nilai p ≠ nilai q. Pada contoh kali ini dimisalkan nilai p = 7 dan q = 11.
   b) Menentukan nilai m yang akan digunakan untuk mencari nilai (RSA) *enciphering exponent*,dimana nilai p = 7 dan nilai q = 11 yang telah ditentukan di langkah sebelumnya.

      m = (p-1) * (q-1)

      m = (7-1) * (11-1)

      m = (6) * (10)

      m = 60

c) Menentukan (RSA) modulus (n) dengan menggunakan nilai p dan q yang ditentukan dilangkah sebelumnya. (RSA) modulus nantinya digunakan sebagai pasangan kunci publik dan kunci rahasia, serta digunakan dalam proses enkripsi dan dekripsi.

n = p * q

n = 7 * 11

n = 77

d) Menentukan (RSA) *enciphering exponent* (e) dengan nilai m yang ditentukan pada langkah sebelumnya.

Formula yang digunakan : gcd(e,m) = 1. Yang berarti nilai gcd atau FPB dari kedua variabel (e dan m) harus bernilai = 1.

Jika FPB dari variabel e dan m bernilai = 1 berarti bilangan pada variabel e merupakan bilangan yang relatif prima terhadap bilangan pada variabel m, sehingga bisa digunakan sebagai (RSA) *enciphering exponent* (e).

Nilai m yang didapat dari langkah sebelumnya = 60.

Tabel 3.1 menunjukkan hasil pemfaktoran angka 60.

Tabel 3.1 Faktorisasi angka 60

1 * 60	4 * 15
2 * 30	5 * 12
3 * 20	6 * 10

Nilai e yang adalah bilangan prima dengan rentang 1 < e < m.

Dimisalkan nilai e yang diambil = 7, sehingga deret faktor dari kedua bilangan bisa dilihat pada tabel 3.2 :

Tabel 3.2 FPB Angka 60 dan 7

Bilangan	**Faktor Pembagi**
60	**1**, 2 , 3 , 4, 5, 6, 10, 12, 15, 20, 30, 60
7	**1**, 7

Tabel 3.2 menunjukkan bahwa FPB dari kedua bilangan =1 yang berarti memenuhi persamaan : gcd(e, m) =1 sehingga e = 7 bisa digunakan untuk menjadi (RSA) *enciphering exponent*.

e) Menentukan (RSA) *decryption exponent* (d) dengan nilai e=7 dan m=60 yang didapat dari langkah sebelumnya.

e * d = 1 mod (m)

e * d = 1+ (k.m)

7*d = 1+(k.60)

d = (1+k.60) / 7

k = bilangan integer sembarang. Perhitungan selesai ketika didapat variabel d yang bernilai integer. Tabel 3.3 menunjukkan percobaan perhitungan nilai d dengan k sembarang hingga didapatkan d yang bernilai integer.

Tabel 3.3 Perhitungan nilai d, modulus 2 digit

e	k	M	d
7	1	60	8,714286
7	2	60	17,28571
7	3	60	25,85714
7	4	60	34,42857
7	5	60	43

Ketika k=5 didapatkan hasil d yang bernilai integer sehingga perhitungan dihentikan dan nilai d=43.

f) Dari langkah-langkah sebelumnya, sekarang didapatkan pasangan kunci RSA sebagai berikut :

kunci publik (n, e) = (77, 7)

kunci rahasia (n, d) = (77, 43)

2. Dimisalkan *record* yang akan dienkrip menggunakan algoritma RSA adalah rangkaian karakter (*plain text*) berikut : "GOESTIAN" maka langkah selanjutnya adalah sebagai berikut :

a) Untuk melakukan proses enkripsi, *plain text* dikonversi terlebih dahulu ke format bilangan. Dimisalkan proses konversi dilakukan berdasarkan tabel ASCII desimal, maka hasilnya ditunjukkan pada tabel 3.4.

Tabel 3.4 Konversi ke ASCII, modulus 2 digit

Char	ASCII Dec
G	71
O	79
E	69
S	83
T	84
I	73
A	65
N	78

Sehingga jika hasil konversi *plain text* ke ASCII desimal pada tabel 3.4 dijadikan suatu deret, maka didapatkan deret dari *PlainText* ASCII desimal :

(M) = **7179698384736578**.

b) Agar bisa dienkrip, deret *plain text* dijadikan blok-blok yang jumlah digit setiap bloknya kurang dari jumlah digit (RSA) modulus. Karena nilai n = 77 (2 digit) maka jumlah digit untuk setiap blok = 1 digit.

c) Jumlah digit pada M = 16, dan untuk setiap bloknya hanya memuat 1 digit, maka akan terdapat 16 buah blok yang berarti proses enkripsi terjadi 16 kali.

Proses Enkripsi menggunakan formula : $C = M^e \bmod n$, sehingga didapat hasil seperti pada tabel 3.5.

Tabel 3.5 Perhitungan enkripsi, modulus 2 digit

blok ke	M	e	n	Cipher
1	7	7	77	28
2	1	7	77	1
3	7	7	77	28
4	9	7	77	37
5	6	7	77	41

Tabel 3.6 Perhitungan enkripsi, modulus 2 digit (lanjutan)

blok ke	M	e	n	Cipher
6	9	7	77	37
7	8	7	77	57
8	3	7	77	31
9	8	7	77	57
10	4	7	77	60
11	7	7	77	28
12	3	7	77	31
13	6	7	77	41
14	5	7	77	47
15	7	7	77	28
16	8	7	77	57

Panjang digit maksimal untuk setiap cipher text (C) adalah sepanjang digit dari (RSA) modulus (n). Panjang digit n =2digit (n=77), sehingga panjang maksimal C = 2 digit. Jika terdapat nilai C yang kurang dari digit maksimal, maka ditambahkan angka nol (0) didepan nilai tersebut untuk menjaga konsistensi panjang setiap blok sehingga tidak terjadi kesalahan pemenggalan blok pada saat akan dilakukan dekripsi. Contohnya pada tabel 3.5 blok ke-2 nilai C=1 , maka saat dijadikan deret Cipher menjadi C=01.

Merujuk pada tabel 3.5, deret cipher text yang dihasilkan adalah sebagai berikut **C = 28012837413757315760283141472857,** deret inilah yang nantinya akan disimpan ke dalam basis data atau bisa juga dikonversikan kembali kedalam bentuk alphabet atau lainnya.

3. Jika suatu deret *cipher text* yang telah disimpan di basis data akan dipanggil, proses dekripsi dilakukan dengan panjang setiap blok sama dengan panjang maksimal *cipher text* yang dihasilkan saat proses enkripsi

sebelumnya. Panjang maksimal *cipher text* pada saat enkripsi adalah 2 digit per blok, maka deret cipher text akan dipecah menjadi 2 digit untuk setiap blok sehingga akan menghasilkan perhitungan yang ditunjukkan pada tabel 3.6 berikut :

Tabel 3.7 Perhitungan dekripsi, modulus 2 digit

blok ke	C	d	n	M
1	28	43	77	7
2	1	43	77	1
3	28	43	77	7
4	37	43	77	9
5	41	43	77	6
6	37	43	77	9
7	57	43	77	8
8	31	43	77	3
9	57	43	77	8
10	60	43	77	4
11	28	43	77	7
12	31	43	77	3
13	41	43	77	6
14	47	43	77	5
15	28	43	77	7
16	57	43	77	8

Sehingga jika nilai M pada tabel 3.6 dijadikan suatu deret kembali akan menghasilkan nilai *plaintext* ASCII **M = 7179698384736578**. Langkah terakhir adalah mengembalikan deret M menjadi *plain text* yang bisa dibaca secara langsung, dengan mengonversi *plaintext ASCII* menggunakan tabel ASCII. Hasil konversi *plaintext ASCII* menggunakan tabel ASCII desimal ditunjukkan pada tabel 3.7.

Tabel 3.8 Konversi ke Char, modulus 2 digit

ASCII Dec	Char
71	G
79	O
69	E
83	S
84	T
73	I
65	A
78	N

Jika hasil konversi pada tabel 3.7 dijadikan deret maka akan didapatkan nilai *plaintext* **M = "GOESTIAN"** yang berarti sama dengan nilai *plaintext* saat sebelum dienkrip.

3.3.2 (RSA) modulus 3 digit

Contoh kali ini menggunakan (RSA) modulus yang berjumlah 3 digit, langkah yang dilakukan sama dengan contoh pengerjaan algoritma RSA menggunakan (RSA) modulus 2 digit sebelumnya, berikut contoh pengerjaannya :

1. Pembentukan Kunci
    a) Menentukan 2 bilangan prima sembarang dan acak yang diwakilkan oleh variabel p dan variabel q.

        Dimisalkan p = 13 dan q=17

    b) Menghitung (RSA) modulus (n) dengan formula n = p * q, Dimana nilai p=13 dan nilai q= 17

        n = p * q

        n = 13 * 17

        n = 221

c) Menghitung nilai m yang akan digunakan untuk mencari (RSA) *enciphering exponent* (e)

m = (p-1) * (q-1)

m = (13-1) * (17-1)

m = (12) * (16)

m = 192

d) Menghitung nilai e dengan formula : gcd(e, m) = 1 , dengan syarat e = bilangan prima dan 1< e < m.

gcd(e, 192) = 1

dimisalkan e = 5, maka formula gcd(5,192) = 1 bernilai *true*.

e) Menghitung nilai (RSA) *deciphering exponent* (d) dengan menggunakan formula e*d = 1 mod (m), dimana nilai e dan m didapatkan dari langkah sebelumnya e=5 dan m = 192.

e * d = 1 mod (m)

d = 1 + (k*m) / e

d = 1 + (k*192) / 5

dengan nilai k = integer sembarang, maka dimisalkan nilai d yang akan diambil adalah d yang bernilai integer. Nilai d yang diambil kali ini adalah d=77

f) Dari langkah diatas, nilai n, e dan d telah ditemukan yang berarti juga pasangan kunci telah terbentuk.

Pasangan kunci publik (n,e) = (221, 5)

Pasangan kunci rahasia (n,d) = (221,77)

2. Dimisalkan terdapat himpunan karakter : “GOESTIAN” yang akan dienkrip menggunakan kunci yang telah dibentuk pada langkah sebelumnya, maka terlebih dahulu karakter tersebut dikonversi kedalam bentuk numerik, proses konversi bisa menggunakan teknik tersendiri dari pengguna atau menggunakan tabel ASCII seperti contoh berikut :

Tabel 3.9 Konversi ke ASCII, modulus 3 digit

Char	ASCII Dec
G	71
O	79
E	69
S	83
T	84
I	73
A	65
N	78

Dari tabel 3.8, hasil konversi setiap baris dijadikan satu deret, akan menghasilkan *plain text* ASCII (**M**) dari karakter yang akan dienkrip sebagai berikut : **M = 7179698384736578**

3. Untuk menjalankan proses enkripsi, digunakan kunci publik yang telah dibentuk sebelumnya yaitu kunci publik (221,5), dengan formula C = M^e mod n. Namun sebelum melakukan perhitungan, terlebih dahulu dilakukan pemecahan deret *plaintext* ASCII menjadi blok yang panjang digit setiap bloknya kurang dari panjang digit n. Pada contoh kali ini, (RSA) modulus yang digunakan adalah = 221 (3 digit), jadi untuk setiap blok dibatasi maksimal 2 digit per blok.

Hasil perhitungan enkripsi dengn formula $C = M^e \bmod n$ sebagai berikut :

Tabel 3.10 Perhitungan Enkripsi, modulus 3 digit

blok ke-	M	e	n	C
1	71	5	221	158
2	79	5	221	27
3	69	5	221	205
4	83	5	221	70
5	84	5	221	67
6	73	5	221	99
7	65	5	221	182
8	78	5	221	91

Dari hasil nilai C pada tabel 3.9, dilakukan penyamaan panjang digit setiap blok dengan menambahkan 0 sebanyak kekurangan panjang digit blok tersebut, kemudian jadikan suatu deret, maka akan dihasilkan deret *cipher text* : **C = 158027205070067099182091**

4. Untuk membuktikan pasangan kunci publik dan kunci rahasia berjalan sempurna, dicoba untuk mendekrip *cipher text* yang telah dienkrip sebelumnya. Proses dekripsi menggunakan kunci rahasia (221, 77), dengan formula $M = C^d \bmod n$. Hasil perhitungan untuk proses dekripsi ditunjukkan pada tabel 3.10.

Tabel 3.11 Perhitungan Dekripsi, modulus 3 digit

blok ke-	C	d	n	M
1	158	77	221	71
2	27	77	221	79
3	205	77	221	69
4	70	77	221	83
5	67	77	221	84
6	99	77	221	73
7	182	77	221	65
8	91	77	221	78

Jika hasil perhitungan dekripsi (nilai M) dijadikan deret akan terbentuk kembali deret *plaintext* ASCII yang sama antara deret sebelum dienkrip dan setelah didekrip yaitu :

M = 7179698384736578

5. Langkah terakhir adalah mengubah deret *plaintext* ASCII menjadi *plaintext* menggunakan tabel ASCII sebagai pedoman konversi.

Tabel 3.12 Konversi ke Char, modulus 3 digit

ASCII Dec	Char
71	G
79	O
69	E
83	S
84	T
73	I
65	A
78	N

Dari uraian sebelumnya, dapat disimpulkan algoritma RSA seharusnya bisa diterapkan pada Oracle 10g karena hanya dibutuhkan algoritma perhitungan, pembagianan blok dan penderetan berdasarkan panjang modulus saja. Semakin panjang pasangan kunci akan membuat ciphertext semakin berbeda dengan plaintext.

Pembentukan nilai e dengan formula gcd(e,m)=1 , yang berarti nilai FPB antara kedua variabel e dan m harus bernilai 1 atau kedua bilangan tersebut harus relatif prima. Syarat e adalah 1 < e < m, sehingga untuk mendapatkan nilai e dimiliki kemungkinan lebih dari 1, karena kemungkinan bilangan prima dari 1 hingga m bisa lebih dari 1.

Sedangkan dalam pembentukan nilai d dengan formula akhir "**d = 1+ (k*m) / e**" dimungkinkan ditambahkan aturan lain untuk mendapatkan nilai d. Variabel k adalah nilai integer sembarang, yang jika dimasukkan kedalam formula akan menghasilkan nilai d, namun nilai d yang memenuhi syarat adalah nilai d yang integer. Sehingga dalam proses penentuan nilai d dengan menggunakan nilai k sebagai integer sembarang, dapat ditambahkan aturan misalnya dalam penentuan *range* percobaan nilai k, atau nilai d yang akan diambil adalah d yang bernilai integer yang setelah berapa kali ditemukan dan lain sebagainya.

Dan juga salah satu syarat wajib yang menjadi perhatian dalam proses enkripsi dan dekripsi adalah nilai M atau *Message* atau *Number* harus lebih kecil jika dibandingkan nilai Modulus, yang berarti untuk membuatnya berada dibatas aman (selalu lebih kecil) ditetapkan jumlah digit maksimal nilai M atau *Message* atau *Number* adalah jumlah digit nilai modulus dikurang 1.

Untuk mengubah *string* menjadi bilangan tidak diharuskan menggunakan tabel konversi tertentu sehingga pengguna dapat menerapkan tabel konversi sendiri sesuai dengan kebutuhan pengguna.

Pada akhirnya, untuk mengimplementasi algoritma kriptografi asimetris RSA dimungkinkan untuk diterapkan banyak kondisi dan batasan yang menyesuaikan dengan kebutuhan pengguna, sehingga memungkinkan dalam penerapan RSA terbentuk algoritma yang berbeda dari tiap penyusun sintak diluar algoritma RSA yang utama. Hal ini memungkinkan untuk menjadi alasan tidak dapat dibentuk *function* khusus untuk algoritma kriptografi asimetris RSA karena banyaknya sisi algoritma yang tidak mutlak.

Penerapan algoritma kriptografi pada basis data salah satunya menguntungkan dalam sisi monitoring program aplikasi. Jika implementasi algoritma kriptografi berada di dalam program aplikasi, maka tingkat kecepatan proses perhitungan yang terjadi akan bergantung salah satunya pada perangkat keras di mana program aplikasi tersebut dijalankan. Sementara itu, jika algoritma kriptografi berada pada sisi basis data, yang menjadi sorotan dalam proses pemantauan kinerja hanya terpusat pada perangkat keras dimana basis data tersebut dijalankan, bukan pada dimana masing-masing program aplikasi dijalankan.

3.4 Analisis Hardware dan Software

Untuk membangun sistem keamanan data pada Oracle 10g menggunakan algoritma RSA kali ini menggunakan perangkat keras dan perangkat lunak sebagai berikut :

3.4.1 Perangkat Keras / *Hardware*.

1. **Komputer Personal**

Spesifikasi komputer yang digunakan sebagai berikut :

a) Processor Dual Core 7850 2,8Ghz
b) RAM DDR2 2Gb PC6400
c) VGA 9800gt 512Mb DDR3
d) Harddisk 320Gb
e) LCD 18.5Inch

2. **Notebook**

Spesifikasi Notebook yang digunakan sebagai berikut :

a) Processor Dual Core T5800 2,0Ghz
b) RAM 1Gb DDR2
c) VGA Mobile Intel® GM45 Shared
d) Harddsik 320Gb
e) LCD 14,1Inch

3.4.2 Perangkat Lunak

Berikut adalah perangkat lunak / *software* yang digunakan untuk mendukung pengimplementasian algoritma RSA pada Oracle 10g :

a. Oracle 10g Enterprise Edition

b. PL/SQL Form Developer

c. Java SE Development Kit (JDK) 6.7

d. Windows XP Professional Edition

e. Windows 7 Ultimate Edition