Kegunaan '*Chinese Remainder Theorem*' dalam Mempercepat dan Meningkatkan Efisiensi Peforma Sistem Kriptografi RSA

Shauma Hayyu Syakura – NIM : 13507025

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
e-mail : if17025@students.if.itb.ac.id

Abstrak

Masalah mengenai RSA sekarang mungkin sudah melebihi dua dekade. Banyak penelitian yang dilakukan untuk menyederhanakan masalah tersebut beberapa tahun belakang ini. Ada yang mencoba untuk 'menyerang' atau 'menerobos' algoritma yang katanya paling mangkus dalam penyandian tersebut, dan beberapa mencoba untuk menghindarinya. Kekuatan algoritma RSA terletak pada tingkat kesulitan dalam memfaktorkan bilangan non-prima menjadi faktor primanya[RIN] dan bilangan non-prima yang digunakan dalam enkripsi RSA saat ini umumnya adalah bilangan bulat berukuran besar (long integer). Sementara metode faktorisasi paling maju sekarang ini masih sangat lambat.

Hal itu berpengaruh pada kecepatan peforma RSA, terutama kemampuan perangkat keras sekarang terbatas. Karena dasar dari algoritma RSA adalah modulo bilangan bulat, maka salah satu solusi yang ditemukan untuk permasalahan efisiensi ini adalah penggunaan Chinese Remainder Theorem (CRT).

1. PENDAHULUAN

Banyak sekali sistem kriptografi yang populer seperti sistem enkripsi RSA, Persetujuan Kunci Diffie-Helman (DH), dan Algoritma Signatur Digital atau DSA (*Digital Signature Algorithm)*, semuanya didasari oleh modulo eksponensial bilangan bulat berukuran besar (*long integer*). Perbedaan besar antara RSA dan sistem kriptografi logaritma diskrit adalah modulo yang digunakan dalam enkripsi RSA adalah produk dari dua bilangan prima. Hal ini menyebabkan *Chinese Remainder Theorem* (CRT) bisa digunakan untuk mempercepat operasi kunci privat (*private keys*).

Dengan memanfaatkan keuntungan dari CRT, komputasi dekripsi RSA bisa dikurangi secara signifikan. Jika dua bilangan prima P dan Q dari modulo N diketahui, perhitungan eksponensasi modulo dari $M = C^D \bmod N$ menjadi modP dan modQ dengan eksponen yang lebih pendek mungkin dilakukan[6]. Karena panjang eksponennya sekitar n/2, kurang lebih 3n/4 aplikasi modulo dibutuhkan untuk satu eksponensasi modulo. Dibandingkan dengan dekripsi RSA non-CRT, RSA dengan CRT lebih cepat dengan faktornya kurang lebih 3,5[6].

2. SISTEM ENKRIPSI RSA

Sebelum kita sampai kepada penjelasan bagaimana CRT bisa mempercepat performa RSA, terlebih dahulu akan dijelaskan secara singkat mengenai dasar algoritma RSA.

Algoritma RSA (Rivest-Shamir-Adleman) diperkenalkan oleh tiga peneliti MIT (*Massachussets Institute of Technology*), yaitu Ron Rivest, Adi Shamir, dan Len Adleman, pada tahun 1976 [1]. Kerja algoritma RSA di-dasarkan oleh konsep bilangan prima dan aritmetika modulo. Baik kunci enkripsi maupun kunci deskripsi merupakan bilangan bulat (*integer*). Kunci enkripsi tidak dirahasiakan dan diketahui umum (sehingga dinamakan juga kunci publik atau *public keys*), namun kunci deskripsi dirahasiakan (dinamakan kunci privat atau *private keys*). Kunci deskripsi dibangkitkan dari beberapa buah bilangan prima bersama-sama dengan kunci enkripsi. Kekuatan kunci enkripsi terletak pada faktorisasi bilangan non-prima menjadi bilangan primanya, apalagi jika bilangan tersebut besar, dan belum ada algoritma yang efisien untuk melakukan-nya. Algoritmanya sebenarnya sangat sederhana, yaitu sebagai berikut.

Algoritma 1 – generasi kunci untuk enkripsi kunci publik pada RSA :

1) Pilih dua buah bilangan prima, yaitu a dan b, keduanya dirahasiakan.
2) Hitung $n = a \times b$, besaran n tidak dirahasiakan
3) Hitung $m = (a - 1) \times (b - 1)$. Sekali m telah dihitung, a dan b dapat dihapus
4) Pilih sebuah bilangan bulat untuk kunci pu-blik, namanya e, yang relatif prima terhadap m.
5) Bangkitkan kunci deskripsi, d, dengan kekongruenan $ed \equiv 1 \pmod{m}$. Lakukan enkripsi terhadap isi pesan dengan persamaan $c_i = p_i^e \bmod n$, p_i adalah blok plainteks, c_i adalah chiperteks, dan e adalah kunci enkripsi (kunci publik). Harus dipenuhi bahwa nilai p_i harus terletak dalam himpunan nilai 0,1,2,..., $n - 1$ untuk menjamin hasil perhitungan tidak berada di luar himpunan.

6) Proses deskripsi dilakukan dengan menggunakan persamaan $p_i = c_i^d \bmod n$, yang dalam hal ini d adalah kunci deskripsi.

Plainteks *m* bisa dienkripsikan dengan Algoritma 2:

Algoritma 2 : Enkripsi untuk Kunci Publik RSA

```
DESKRIPSI : /**mengenkripsikan
sebuah pesan m dengan kunci publik
e**/
Input : m, e, n
Output : c = m^e mod n
```

Cipherteks c bisa didekripsikan dengan Algoritma 3:

Algoritma 3 : Dekripsi untuk Kunci Publik RSA

```
DESKRIPSI : /**mengenkripsikan
sebuah chiperteks c dengan kunci
privat d**/
Input : c, d, n
Output : m = c^d mod n
```

Spesifikasi ini disebut dengan *multi-prime RSA* atau RSA banyak-prima, dimana modulusnya bisa memiliki lebih dari dua buah faktor prima. Keuntungan dari *multi-prime RSA* adalah membutuhkan kemampuan komputasi yang lebih rendah untuk dekripsi dan primitif-primitif signatur. Peforma yang lebih baik bisa didapatkan dengan menggunakan satu prosesor, namun lebih baik lagi bila multi-prosesor digunakan karena eksponensiasi modulo dapat dikerjakan secara pararel.

Teknik lain dimana eksponen yang dirahasiakan [5] *d* memiliki notasi biner $(d_{i-1}, d_{i-2}, ..., d_1, d_0)_2$, dan $d_{i-1} = 1$ menyatakan bit yang paling signifikan. Eksponensiasi modular dilakukan secara bit per bit dengan menggunakan perkalian modular secara berulang-ulang. Algoritmanya dinamakan dengan metode Pengkuadratan dan Perkalian (*Square & Multiply method*).

Algoritma 4 : Algoritma Pengkuadratan dan Perkalian untuk Eksponensiasi.

```
Input : c, n, d = (d_{i-1}, d_{i-2}, ...,
d_1, d_0)_2
Output : s = c^d mod n
Let s = c. If d = 0 then return (1).
For j = i -2 to 0 do
s = s^2 mod n
if d_i == 1 then
Let s = (s.c) mod n.
end if
end for
return (s)
```

Kedua teknik ini akan digunakan dalam makalah ini untuk mengevaluasi peforma berbagai variasi (*variants*)[5] dari RSA dan membandingkan dengan RSA berbasis CRT.

3. CHINESE REMAINDER THEOREM (CRT)

Kompleksitas dari dekripsi RSA $M = C^D \bmod N$ bergantung secara langsung pada ukuran *D* dan *N*. Eksponen dekripsi *D* menspesifikasikan bilangan perkalian modulo yang dibutuhkan untuk melakukan eksponensiasi, dan modulo *N* menetukan ukuran dari hasil sementaranya. Cara untuk mengurangi ukuran *D* dan *N* adalah dengan menggunakan keuntungan dari *Chinese Remainder Theorem (CRT)* dan *Fermat's Little Theorem*.

3.1 Latar Belakang

Berikut ini, beberapa fakta dasar dan kesimpulan dari CRT akan dijelaskan. Latar belakang ini sangat penting untung efisiensi dekripsi RSA.

Teorema 1 (Chinese Remainder Theorem)
Misalkan, $n_1, n_2, ..., n_k$ adalah bilangan bulat positif sedemikian sehingga PBB $(n_i, n_j) = 1$ untuk $i \neq j$. Lebih jauh lagi, misalkan $n = n_1, n_2, ..., n_k$, dan misalkan $x_1, x_2, ..., x_k$ adalah bilangan bulat, maka sistem kekongruenan lanjar

$$x \equiv x_1 \bmod n_1$$
$$x \equiv x_2 \bmod n_2$$
$$\vdots$$
$$x \equiv x_k \bmod n_k$$

memiliki solusi x untuk semua kekongrunenan dan setiap dua solusi kongruen dengan modulo n yang lain. Maka hanya ada tepat satu solusi x antara 0 *dan* $n - 1$.

Bukti konstruktif dari *Chinese Remainder Theorem* adalah sebagai berikut. Solusi unik *x* dari semua kekongruenan memenuhi $0 \leq x < n$ dapat dihitung sebagai

$$x = \left(\sum_{i=1}^{k} x_i r_i s_i \right) \bmod n \qquad (1)$$
$$= (x_1 r_1 s_1 + x_2 r_2 s_2 + \ldots + x_k r_k s_k) \bmod n$$

Dimana dan $s_i = r_{i-1} \bmod n_i$ untuk $i = 1, 2, ..., k$. Metode ini disebut algoritma Gauss [6].

Corollary 1.1 *Jika bilangan bulat $n_1, n_2, ..., n_k$ adalah sepasang relatif prima dan $n = n_1n_2...n_k$, maka untuk semua bilangan bulat a,b selalu benar bila $a \equiv b \bmod n$ jika dan hanya jika $a \equiv b \bmod n_i$ untuk setiap $i = 1, 2, ..., k$.*

Sebagai konsekuensi dari CRT, setiap bilangan bulat positif $a < n$ bisa secara unik direpresentasikan sebagai sebuah k-*tuple* $[a_1, a_2, ..., a_k]$ dan sebaliknya, diamana a1 adalah sisa dari a mod ni, untuk setiap $i = 1, 2, ..., k$. Sistem konversi dari a ke sisanya didefinisikan ole*h* $n_1, n_2, ..., n_k$ mudah dilakukan dengan reduksi modulo $a \bmod n_i$. Konversi balik dari sisa ke representasi "notasi standar" sulit dilakukan kerenan membutuhkan kalkulasi yang dinyatakan pada persamaan (1).

Teorema 2 (Fermat's Little Theorem)
Misalkan p bilangan prima, setiap bilangan bulat a yang tidak bisa dibagi dengan p memenuhi $a^{p-1} \equiv a^{-1} \bmod p$.[6]

Fermat's Little Theorem sengat berguna untuk menghitung inverse multiplikatif dari sebuah bilangan bulat a karena $a^{p-1} \equiv a^{-1} \bmod p$.

Collolary 2.1 *Jika sebuah bilangan bulat a tidak bisa dibagi dengan p dan jika $n \equiv m \bmod p - 1$, maka $a^n \equiv a^m \bmod p$.*

Collolary (2.1) menyatakan bila mengerjakan modulo dari sebuah bilangan prima p, bilangan eksponennya bisa dikurangi sebesar mod$(p - 1)$. Hal ini menyebabkan dekripsi RSA bisa dijalankan dengan eksponen yang lebih rendah.

3.2 Dekripsi RSA menggunakan CRT

Karena P dan Q adalah bilangan prima, maka setiap pesan $M < N = PQ$ secara unik direpresentasikan dengan tuple $[M_P,M_Q]$, dimana $M_P = M \bmod P$ dan $M_Q = M \bmod Q$. Sehingga, mungkin untuk mendapatkan M dengan menghitung M_P, M_Q, dan memasukkannya ke persamaan (1), dari pada perhitungan biasa dengan $M = C^D \bmod N$. Dengan menggunakan *corollaly* (2.1), ukuran eksponen bisa diturunkan:

$$\begin{aligned} M_P &= M \bmod P = (C^D \bmod N) \bmod P \\ &= C^D \bmod P \quad (\text{karena } N = PQ) \\ &= C^{D \bmod (P-1)} \bmod P \\ &= C^{D_P} \bmod P \quad \text{saat } D_P = D \bmod (P-1) \end{aligned} \tag{2}$$

Selanjutnya, bisa kita lihat bahwa chiperteks C bisa direduksi menjadi modulo P sebelum menghitung M_P, sehingga panjang dari semua operand bisa dikurangi menjadi setengahnya. Dari $C_P = C \bmod P$ dan $C_Q = C \bmod Q$, juga $D_P = D \bmod (P - 1)$ dan $D_Q = D \bmod (Q - 1)$, kita mendapatkan persamaan berikut untuk M_P dan M_Q:

$$M_P = {C_P}^{D_P} \bmod P \text{ dan } M_Q = {C_Q}^{D_Q} \bmod Q \tag{3}$$

Rekombinasi dari M_P dan M_Q untuk mendapatkan M bisa dilakukan dengan menggunakan persamaan (1). Untuk kasus spesial seperti $k = 2$, $n_1 = P$, $n_2 = Q$, dan $n = N = PQ$, kita mendapatkana $r_1 = N/P = Q$ dan $r_2 = N/Q = P$. Selain itu persamaan (1) bisa di sederhanakan dengan teori kecil Fermat :

$$\begin{aligned} M &= \left(M_P Q \left(Q^{-1} \bmod P\right) + M_Q P \left(P^{-1} \bmod Q\right)\right) \bmod N \\ &= \left(M_P Q \left(Q^{P-2} \bmod P\right) + M_Q P \left(P^{Q-2} \bmod Q\right)\right) \bmod N \\ &= \left(M_P \left(Q^{P-1} \bmod N\right) + M_Q \left(P^{Q-1} \bmod N\right)\right) \bmod N \end{aligned} \tag{4}$$

Persamaan terakhir pada persamaan (4) didapatkan dari fakta bahwa $a\ (b \bmod c) = (ab) \bmod (ac)$ untuk semua bilangan bulat tak negatif a, b, c. Perlu diingat bahwa koefisien $Q^{P-1} \bmod N$ dan $P^{Q-1} \bmod N$ adalah konstanta dan bisa diprekomputasi, sehingga perhitungan untuk rekombinasi dari M_P dan M_Q bisa dikurangi menjadi dua perkalian, yang satu pertambahan dan yang lainnya pengurangan modulo M.

Ketika mengasumsikan eksponen $D_P = D \bmod (P - 1)$ dan $D_Q = D \bmod (Q - 1)$, juga konstanta-konstanta yang diperlukan untuk rekombinasi $R_P = P^{Q-1} \bmod N$ dan $R_Q = P^{Q-1} \bmod N$ sudah di prekomputasi, dekripsi RSA berbasis CRT bisa dilakukan menurut langkah-lang-kah berikut ini[6] :

1) Hitung $C_P = C \bmod P$ dan $C_Q = C \bmod Q$
2) Hitung eksponensiasi $M_P = {C_P}^{D_P} \bmod P$ dan $M_Q = {C_Q}^{D_Q} \bmod Q$
3) Hitung koefisien $S_P = M_P R_P \bmod N$ dan $S_Q = M_Q R_Q \bmod N$
4) Hitung $M = S_P + S_Q$. Jika $M \geq N$ lalu hitung $M = M - N$

Sangat jelas dapat dilihat bahwa reduksi inisial dari cipherteks C (langkah 1) dan rekombinasi CRT (langkah 3 dan 4) tidak menyebabkan biaya komputasi yang sangat signifikan dibandingkan dengan perhitungan pada langkah 2. Dua eksponensiasi (langkah 2) bisa dikomputasikan secara independen dari satu sama lain dan pararel. Dibandingkan dengan dekripsi RSA non-CRT, dekripsi yang dilakukan pada perangkat keras n bit bisa 4 kali lebih cepat jika perangkat keras $n/2$ bit digunakan. Kecepatan yang dramatis ini disebabkan oleh pengurangan ukuran bilangan sebesar 50% dari keduanya, yaitu eksponen dan modulusnya.

4. HASIL

Berikut ini akan diberikan hasil dari pengujian peforma RSA tanpa CRT dan RSA dengan CRT, serta dengan varian-varian RSA yang lain, antara lain: Mprime RSA (*MultiPrime RSA*), Mpower RSA (*MultiPower RSA*), *Rebalanced RSA*, Rprime RSA,

dan Batch RSA. Pengujian dilakukan dengan sebuah CPU AMD Athlon; Sistem Operasi Windows XP dan Linux, dengan RAM sebesar 256 MB, dan menggunakan bahasa C dengan GNU MP [9] (*library* GMP). Untuk grafik dan tabel dibuat dengan Microsoft excel.

4.1 Pembandingan Kecepatan

Yang disarankan dalam pengujian ini adalah, kita diusahakan untuk tidak menganalisa algoritma kriptografi dengan panjang yang tetap; tetapi lebih ke mengevaluasi kecepatan dan kebutuhan memori bergantung dari panjang kunci, sehingga hasil yang kita dapatkan tidak akan 'ketinggalan zaman' jika panjang kunci yang direkomendasikan menjadi semakin besar di masa depan. Dibawah ini dutunjukkan percepatan dalam setiap implementasi RSA yang berbeda-beda. Yang pertama adalah RSA sederhana dan yang kedua adalah RSA dengan CRT.

Table 1: With CRT and without CRT

Speedup	Key Length		
	768	1024	2048
RSA without CRT	1.0	1.0	1.0
RSA with CRT	3.24	3.32	3.47

Lalu pembandingan dengan varian lain:

Table 2: Major RSA variants

Speedup	Key Length		
Variant	768	1024	2048
Mprime	1.95	1.89	1.97
Mpower	2.49	2.54	2.79
Rebalanced	2.52	3.02	5.98
Rprime	3.00	3.88	7.83
Batch	2.47	2.78	3.42

Percepatan untuk proses Dekripsi saat b = 4 (jumlah pesan), k = 3 (jumlah bilangan prima), dan s = 160

Untuk moduli 768-bit varian yang menunjukkan performa yang lebih baik adalah RCT-RSA, tapi untuk moduli 1024 dan 2048 bit, Rprime RSA-lah yang menunjukkan peforma terbaik[5]. Kita dapat melihat bahwa varian R-Prime dan Rebalanced RSA percepatannya meningkat secara signifikan untuk moduli yang lebiih besar. Ini terjadi mengingat besar s yang tetap dan sama dengan 160 bit (ingat bahwa s adalah besar eksponen yang digunakan dalam algoritma dekripsi), sedangkan eksponen ini meningkat untuk semua varian yang lain. Di bawah ini adalah pembandingan percepatan ditunjukkan dalam sebuah grafik, yang pertama untuk 768 bits, selanjutnya untuk 1024 bits, dan selanjutnya lagi untuk 2048 bits.

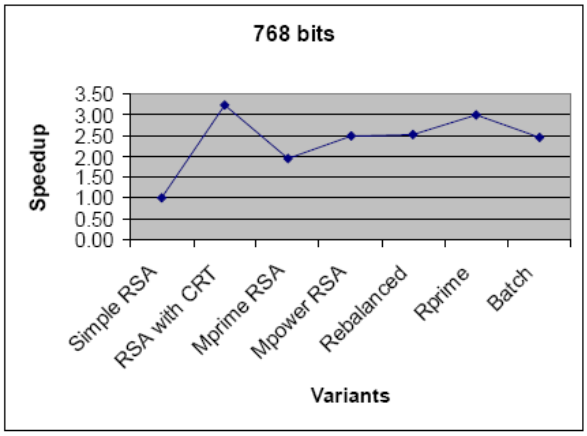

Fig. 1 Comparison of RSA variants using 768 bits

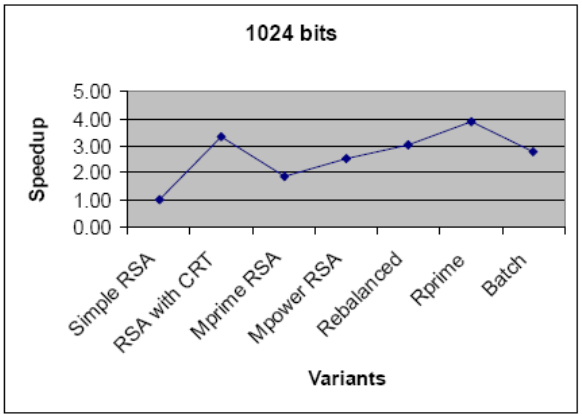

Fig. 2 Comparison of RSA variants using 1024 bits

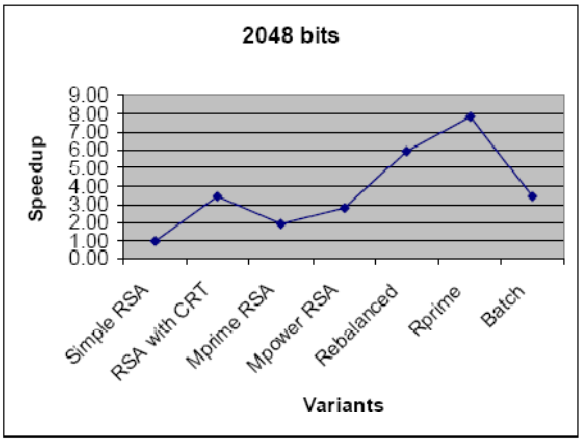

Fig. 3 Comparison of RSA variants using 2048 bits

Dapat dilihat secara lebih jelas RSA dengan CRT percepatannya paling tinggi saat panjang kunci sebesar 768 bit, namun untuk jumlah bit yang lebih besar, RSA dengan CRT kurang signifkan jika dibandingkan dengan varian Rprime, Rebalanced, dan Batch.

4.2 Pembandingan Memori

Akan ditunjukkan kebutuhan memori untuk implementasi RSA yang berbeda-beda. Di bawah ini ditunjukkan hasil implementasi RSA biasa dengan RSA berbasis CRT.

Table 3: With CRT and without CRT

	Total Memory
RSA without CRT	4n or 4096 bits
RSA with CRT	8n or 8192 bits

Dilakukan dengan = 1024 bit

Di bawah ini adalah kebutuhan memori untuk berbagai varian RSA lainnya, ditampilkan dalam bentuk tabel dan grafik:

Table 4: Major RSA variants

	Total Memory
Mprime $(p\ r\ q)$	26n/3 bits=8875 bits
Mpower (p^2q)	25n/3 bits=8534 bits
Rebalanced (k=160)	7n+2k bits =7200 bits
Batch (b=4)	67n/2+784 bits=35088 bits

Here n=1024 bits

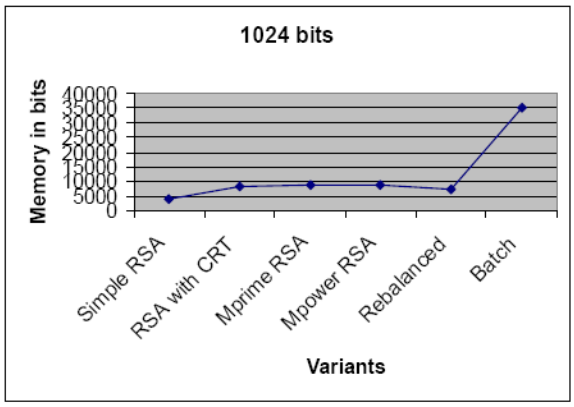

Fig. 4 Comparison of memory of RSA variants using 1024 bits

Dapat dilihat semakin cepat waktu dekripsi semakin besar memori yang dibutuhkan. Hal ini dapat dijelaskan karena operasi yang dikerjakan oleh RSA-RSA dengan percepatan yang lebih tinggi lebih banyak dari RSA-RSA dengan percepatan yang lebih rendah.

Namun RSA ber-CRT, beserta Mprime RSA dan Rebalanced RSA masih menempati urutan rendah dibandingkan dengan Batch RSA.

5. ANALISIS DAN KESIMPULAN

Kesimpulan yang dapat diperoleh dari makalah ini adalah:

1) Parameter keamanan dalam algoritma sekarang adalah panjang kunci. Namun semakin besar bit yang digunakan, semakin besar waktu komputasi dan memori yang dibutuhkan. Dan saat ini, panjang kunci standar untuk sistem kripto RSA adalah lebih dari 512 bit.
2) Salah satu solusi dari permasalahan untuk mempercepat waktu dekripsi RSA adalah dengan menggunakan *Chinese Remainder Theorem* (CRT).
3) CRT dapat mempercepat komputasi dekripsi RSA karena perhitungan eksponen modulo pada dekripsi bisa dipecah menjadi dua dan dikerjakan secara pararel.
4) RSA berbasis CRT dapat mempercepat waktu komputasi sekitar 3,5 kali dari standar kecepatan.
5) RSA berbasis CRT mempercepat waktu dekripsi secara signifikan dibanding RSA yang lain untuk panjang kunci kurang dari sama dengan 768 bit. Namun lebih dari 768 bit percepatan CRT tidak terlalu signifikan dibanding varian RSA yang lain.
6) RSA berbasis CRT membutuhkan memori yang lebih besar dari CRT biasa.

DAFTAR PUSTAKA

[1] Munir, Rinaldi. 2004. Diktat Kuliah Matematika Diskrit. Departemen Teknik Informatika. Institut Teknologi Bandung.

[2] http://www.math.cornell.edu/~mec/2003-2004/cryptography/RSA/RSA.html
Tanggal Akses 2 Januari 2008
Pukul 17.23

[3] www.math.unc.edu/Faculty/petersen/Coding/cr2.pdf
Tanggal Akses 2 Januari 2008
Pukul 17.40

[4] http://www.scribd.com/doc/209760/Twenty-Years-Of-Attacks-On-The-RSA-Cryptosystem (CRT)
Tanggal Akses 2 Januari 2008
Pukul 18.11

[5] http://www.acsac.org/2000/papers/48.pdf
Tanggal Akses 3 Januari 2008
Pukul 21.49

[6] http://paper.ijcsns.org/07_book/200807/20080702.pdf
Tanggal Akses 3 Januari 2008
Pukul 22.03