Scientific Journal of Informatics
Vol. 3, No. 1, Mei 2016
p-ISSN 2407-7658 http://journal.unnes.ac.id/nju/index.php/sji e-ISSN 2460-0040

Implementasi Kriptografi Kunci Publik dengan Algoritma RSA-CRT pada Aplikasi *Instant Messaging*

Ashari Arief[1] dan Ragil Saputra[2]

[1,2]Jurusan Ilmu Komputer, FSM, Universitas Diponegoro, Semarang
[1]asharіarief@gmail.com, [2]ragil.saputra@undip.ac.id

Abstrak

Instant messaging merupakan salah satu bentuk kemajuan teknologi komunikasi yang mempermudah penyampaian informasi. Saat ini, dengan semakin banyaknya pengguna aplikasi *instant messaging* berakibat pada dampak negatif berupa penyadapan data khususnya saat terjadi komunikasi yang bersifat rahasia. Algoritma RSA merupakan salah satu satu algoritma dalam kriptografi kunci publik. Pada proses enkripsi dan dekripsi digunakan kunci yang berbeda. Proses dekripsi algoritma RSA sering terjadi kendala karena ukuran kunci dekripsi yang relatif besar dapat memperlambat proses. Untuk mempercepat proses dekripsi, algoritma RSA dapat dimodifikasi dengan algoritma CRT (*Chinese Remainder Theorem*), sering disebut dengan Algoritma RSA-CRT. Implementasi algoritma kriptografi RSA-CRT pada aplikasi *instant messaging* pada panjang bit n mulai dari 56 bit sampai 88 bit, proses dekripsi RSA-CRT dua kali lebih cepat dibandingkan proses dekripsi RSA.

Kata Kunci: Kriptografi, Kunci publik, RSA-CRT, *Instant messaging*.

1. PENDAHULUAN

Kemajuan teknologi komputer dan telekomunikasi membantu dalam menyelesaikan banyak pekerjaan dengan cepat, akurat, dan efisien. Salah satu kemajuan teknologi komunikasi yaitu menghasilkan aplikasi *instant messaging* atau pesan instan. *Instant messaging* merupakan fasilitas komunikasi *chatting* untuk para pengguna internet sehingga user dapat berkomunikasi dengan cara mengirimkan pesan berupa teks dengan user lain [1]. Namun seiring dengan kemajuan teknologi, dengan semakin banyaknya pengguna yang menggunakan aplikasi *instant messaging* terdapat dampak negatif berupa penyadapan data khususnya saat terjadi komunikasi yang bersifat rahasia dan penting sehingga aspek keamanan dalam pertukaran informasi dianggap penting.

Kriptografi adalah ilmu yang mempelajari teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan, integritas data, otentikasi entitas, dan otentikasi asal data [2]. Kriptografi bertujuan agar informasi yang bersifat rahasia dan dikirim melalui suatu jaringan, seperti LAN atau internet, tidak dapat diketahui dan dimanfaatkan oleh orang lain atau pihak yang tidak berkepentingan.
RSA merupakan algoritma kriptografi kunci publik atau sering disebut kunci asimetrik (kunci enkripsi dan kunci dekripsi berbeda) sehingga tidak membutuhkan saluran yang aman untuk distribusi kunci. RSA ditemukan oleh tiga peneliti dari MIT (*Massachussets Institute of Technology*), yaitu Ronald Linn Rivest, Adi Shamir, dan Len Adleman pada tahun 1977 [3]. Keamanan algoritma RSA terletak pada sulitnya memfaktorkan bilangan yang besar menjadi faktor-faktor prima.

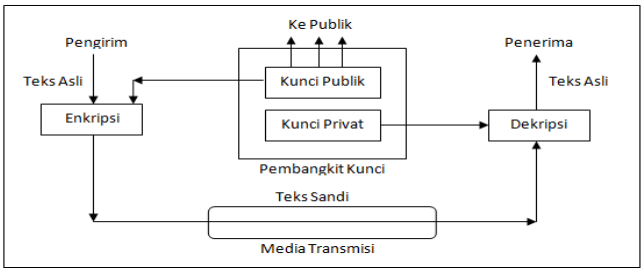
CRT (*Chinese Remainder Theorem*) merupakan suatu algoritma untuk mengurangi perhitungan aritmatika modular dengan modulus besar untuk perhitungan yang sama untuk masing-masing faktor dari modulus [4]. CRT dapat memperpendek ukuran bit eksponen dekripsi d (merupakan kunci publik RSA atau RSA-CRT) dengan cara menyembunyikan d pada sistem kongruen sehingga mempercepat waktu dekripsi serta dapat digunakan bersama algoritma RSA yang disebut RSA-CRT.

Untuk meningkatkan keamanan dari segi pengiriman pesan yang dibuat dalam saluran yang tidak aman serta modifikasi algoritma RSA dengan menggunakan teorema CRT agar dapat dibandingkan dengan algoritma RSA, perlu dibangun sebuah aplikasi instant messaging dengan mengimplementasikan algoritma kriptografi RSA-CRT.

2. METODE

2.1. Kriptografi Kunci Publik RSA

Algoritma RSA menggunakan 2 angka (e dan d) sebagai kunci publik dan kunci privat. Pada algoritma RSA e dan n diumumkan untuk umum sedangkan d dirahasiakan. Meskipun RSA dapat digunakan untuk mengenkripsi dan mendekripsi pesan, sangat lambat jika pesan tersebut panjang. Oleh karena itu, algoritma RSA berguna untuk pesan singkat. Sejak algoritma menggunakan 2 kunci untuk enkripsi dan dekripsi, algoritma RSA dianggap sebagai contoh kunci asimetrik kriptografi [5]. Desain konseptual dari algoritma RSA dapat disajikan pada Gambar 1.



Gambar 1. Konsep kriptografi kunci publik RSA

Algoritma RSA dibagi menjadi 3 langkah [6]:

1. Pembangkit Kunci
   a. Pilih 2 bilangan prima besar untuk nilai *p* dan *q*
   b. hitung nilai modulus $n = p \times q$ (1)
   c. Hitung menggunakan fungsi Euler $\phi(n) = (p-1) \times (q-1)$ (2)
   d. Pilih nilai integer *e* acak sebagai kunci publik, dengan syarat memenuhi *Greater Common Divisor* (GCD) $(e, \phi(n)) = 1$, $1 < e < \phi(n)$ (3)
   e. Hitung kunci privat d sehingga $d \times e = 1 \pmod{\phi(n)}$ (4)
2. Enkripsi
   $C = M^e \bmod n$ (5)
3. Dekripsi
   $M = C^d \bmod n$ (6)

2.2. CRT *(Chinese Remainder Theorem)*

CRT (*Chinese Remainder Theorem*) merupakan suatu algoritma untuk mengurangi perhitungan aritmatika modular dengan modulus besar untuk perhitungan yang sama untuk masing-masing faktor dari modulus [4].

Terdapat bilangan-bilangan $n_1, n_2, \dots, n_k$ adalah bilangan bulat positif di mana relatif prima pada pasangan [7]. Contohnya: FPB(n_i,n_j) = 1 di mana $i \neq j$. Lebih jauh lagi, $n = n_1 n_2, \dots, n_k$ dan $x_1, x_2, \dots, x_k$ adalah bilangan bulat. Maka sistem kongruen:

$$x \equiv x_1 \bmod n_1,$$
$$x \equiv x_2 \bmod n_2,$$
$$\dots$$
$$x \equiv x_k \bmod n_k \quad (7)$$

memiliki solusi yang simultan pada semua kongruen dan dua solusi apapun adalah saling kongruen modulo. Lebih jauh lagi terdapat tepatnya satu solusi antara 0 dan n-1. Solusi unik dari kongruen simultan memenuhi $0 \leq x \leq n$ dapat dihitung dengan rumus (8):

$$x = \left(\sum_{i=1}^{k} x_i r_i s_i \right) mod\ n$$
$$= (x_1 r_1 s_1 + x_2 r_2 s_2 + \dots + x_k r_k s_k)\ mod\ n \quad (8)$$

Dimana $r_i = \frac{n}{n_i}$ dan $s_i = r_i^{-1} \bmod n_i$ untuk $i = 1, 2, \dots, k$.

Jika bilangan bulat $n_1, n_2, \dots, n_k$ adalah pasangan relatif prima dan $n = n_1, n_2, \dots, n_k$, maka untuk semua bilangan bulat a, b pasti akan valid di mana $a \equiv b \bmod n$ jika dan hanya jika $a \equiv b \bmod n_i$ untuk setiap $i = 1, 2, \dots, k$.

Sebagai konsekuensi dari CRT, setiap bilangan bulat positif $a < n$ dapat direpresentasikan secara unik sebagai sebuah k-tuple [$a_1, a_2, \dots, a_k$] dan sebaliknya. Di mana a_i menunjukan sisa / residu $a \bmod n_i$ untuk setiap $i = 1, 2, \dots, k$. Konversi a menjadi sistem residu didefinisikan dengan $n_1, n_2, \dots, n_k$ dilakukan secara sederhana dengan reduksi modular $a \bmod n^i$. Konversi balik dari representasi sisa menjadi "angka-angka standar" adalah lebih sulit seperti yang dibutuhkan dalam kalkulasi pada rumus (7).

Keuntungan dasar dengan menggunakan *Chinese Remainder Theorem* adalah memungkinkan untuk membagi modulo eksponensial yang besar ke dalam dua eksponensial yang jauh lebih kecil, satu di atas p dan satu di atas q. Dua modulo ini adalah faktor utama dari n yang dikenali.

2.3. Algoritma RSA dan CRT (RSA-CRT)

Sistem kriptografi RSA dapat dimodifikasi dengan menggunakan teorema CRT disebut dengan RSA-CRT. Terbukti sistem kriptografi RSA-CRT memiliki waktu komputasi yang lebih singkat daripada sistem kriptografi RSA biasa, yaitu sekitar 4 kali lebih cepat [8].

Algoritma RSA-CRT dibagi menjadi 3 langkah:

1. **Pembangkit Kunci RSA-CRT**
   Pada dasarnya RSA-CRT sama dengan RSA biasa tetapi memanfaatkan teorema CRT untuk memperpendek ukuran bit eksponen deksipsi *d* dengan cara menyembunyikan *d* pada sistem kongruen sehingga mempercepat waktu dekripsi. Berikut algoritma pembangkit kunci RSA-CRT:
   a. Bangkitkan bilangan prima besar *p* dan *q*
   b. Lihat rumus (1).
   c. Lihat rumus (2).
   d. Lihat rumus (3).
   e. Lihat rumus (4).
   f. $dP = d \bmod (p - 1)$ (9)
   g. $dQ = d \bmod (q - 1)$ (10)
   h. $qInv = q^{-1}$ pada Z_p (11)
   i. $K_{publik} = (e, n)$, $K_{privat} = (dP, dQ, qInv, p, q)$ (12)

2. **Enkripsi RSA-CRT**
   Kunci publik RSA-CRT sama dengan sistem RSA yaitu (*e*, *n*) sehingga algoritma enkripsi tidak mengalami perubahan yaitu dengan menggunakan fungsi eksponensial modular yaitu lihat pada rumus (5).

3. **Dekripsi RSA-CRT**
   Diberikan teks sandi seperti rumus (5) dan kunci privat (dP, dQ, qInv, *p*, *q*) dekripsi RSA-CRT seperti berikut:
   a. $m_1 = C^{dP} \bmod p$ (13)
   b. $m_2 = C^{dQ} \bmod q$ (14)
   c. $h = qInv.(m_1 - m_2) \bmod p$ (15)
   d. $M = m_2 + h.q$ (16)

2.4. Algoritma *Fast Modular Exponentiation*

Algoritma *fast modular exponentiation* merupakan algoritma untuk menghitung suatu *modular exponentiation* dengan cepat. Rumus *modular exponentiation* dapat dilihat pada rumus (17).

$a^b \bmod c$ (17)

Algoritma *fast modular exponentiation* dibagi menjadi 3 langkah [9]:

1. Ubah b pada rumus (17) menjadi *binary* dengan ketentuan pada Tabel 1.

Tabel 1. Ketentuan *Binary*

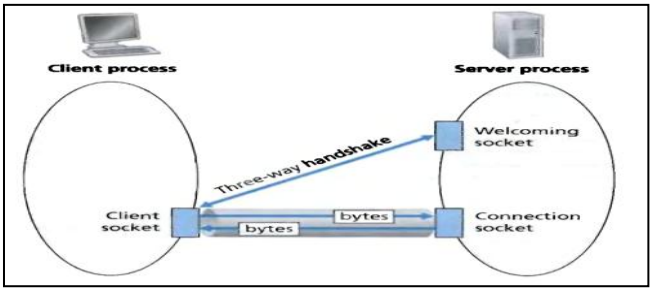
Mulai dari digit paling kanan, dengan nilai awal k = 0 dan untuk setiap digit:
a. Jika digit bernilai 1, maka ubah 1 menjadi 2 ^ k, jika tidak berinilai 1 maka abaikan.
b. Tambah 1 pada nilai k, kemudian lakukan tahapan yang sama untuk digit sebelah kirinya.

2. Pada rumus 2.5 hitung mod c dari 2 perpangkatan yang ≤ b.
3. Pada rumus 2.5 gunakan sifat perkalian modular untuk digabungkan dengan perhitungan nilai mod c.

2.4. Pemrograman Soket dengan TCP (*Transmission Control Protocol*)

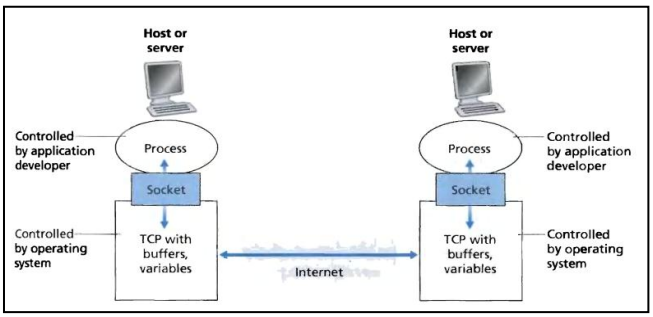
Sebagian besar aplikasi terdiri dari sepasang proses komunikasi, dengan dua proses dalam setiap pasangan mengirim pesan satu sama lain. Setiap pesan yang dikirim dari satu proses yang lain harus melalui jaringan yang mendasarinya. Sebuah proses mengirimkan pesan ke dalam, dan menerima pesan dari jaringan melalui antarmuka perangkat lunak disebut soket [10].

Aplikasi jaringan banyak terdiri dari sepasang program-program klien dan *server* program yang berada di dua sistem akhir yang berbeda. Ketika kedua program dijalankan, klien dan proses *server* dibuat, dan proses ini berkomunikasi satu sama lain dengan membaca dari dan menulis ke soket. Karena soket memainkan peran sentral dalam klien/*server* mengembangkan aplikasi klien/*server* juga disebut sebagai pemrograman soket, dapat dilihat pada Gambar 2.



Gambar 2. *Client-Socket*, *Welcoming Socket*, dan *Connection Socket*

Pemrograman soket dengan TCP adalah pemrograman soket yang berorientasi koneksi dan menyediakan *reliable* (handal) *byte-stream* yang menjamin bahwa proses *server* akan menerima (melalui koneksi soket) setiap *byte* dalam urutan yang dikirim. Selain itu proses klien tidak hanya mengirimkan *byte* tetapi juga menerima *byte* dari koneksi soket. Untuk proses komunikasi melalui soket TCP dapat dilihat pada Gambar 3.



Gambar 3. Proses komunikasi melalui soket TCP

3. HASIL DAN PEMBAHASAN

3.1. Mengolah *Server* dan *Client*

Aplikasi *instant messaging* menggunakan algoritma kriptografi RSA-CRT dengan pemrograman soket berbasis TCP, sehingga dibuatnya *class* TcpListener dengan *port* 8888 dan *class* TcpClient (untuk *server*) seperti kode berikut:

```
Dim serverSocket As New TcpListener(8888)
'mendeklarasikan variabel serverSocket sebagai TcpListener (kelas yg berfungsi
'mendengarkan koneksi dari tcp klien) dengan port 8888
Dim clientSocket As TcpClient
'mendeklarasikan variabel clientSocket sebagai TcpClient (kelas yg menyediakan
'koneksi client untuk layanan jaringan TCP)
```

class TcpClient (untuk *client*) seperti kode berikut:

```
Dim clientSocket As New System.Net.Sockets.TcpClient
```

serta dibuatnya inisialisasi *port* 8888 untuk melakukan hubungan (pada sisi *client*) seperti kode berikut:

```
clientSocket.Connect(ip.Text, 8888)
```

Pada sisi *client* terdapat masukan IP (*localhost*) dan nama (Andi) untuk pengguna pertama, pengguna kedua dengan masukan IP (*localhost*) dan nama (Budi), seperti pada Gambar 4 untuk sisi *client* dan Gambar 5 untuk sisi *server*.

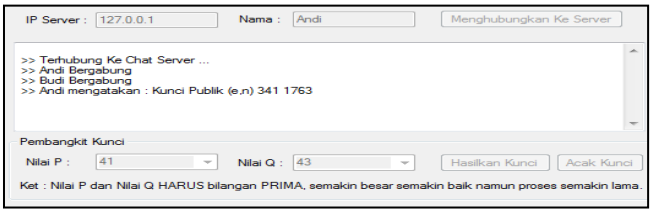
Gambar 4. Tampilan sisi *client*

Gambar 5. Tampilan sisi *server*

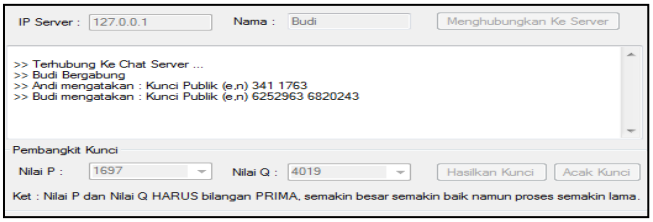
3.2. Mengolah Kunci

Pada alur proses mengolah kunci, Andi membangkitkan kunci dengan nilai P = 41 dan nilai Q = 43, seperti pada gambar 6 sedangkan Budi menggunakan kunci acak seperti pada gambar 7. Gambar 8 merupakan tampilan *server* setelah menerima dua kunci publik dari 2 pengguna.

Pada Gambar 6, Andi memiliki kunci publik e = 341 dan n = 1763 dengan waktu pengolahan kunci selama 0 milidetik. Pada Gambar 7, Budi membangkitkan kunci secara acak, dengan kunci publik e = 6252963 dan n = 6820243 dengan waktu pengolahan kunci selama 1088 milidetik, serta memiliki maksimum karakter yang digunakan untuk mengirimkan pesan yaitu 1.350 karakter dikarenakan nilai n milik Andi kurang dari 1.000.000.



Gambar 6. Andi membangkitkan kunci dengan nilai p = 41, q = 43 (*client*)



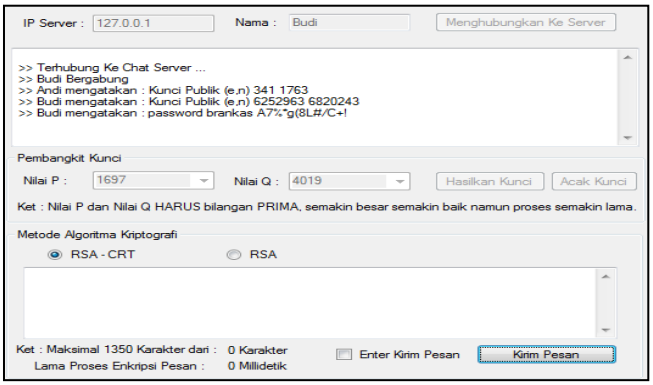
Gambar 7. Budi membangkitkan kunci acak (*client*)

Kunci publik Andi digunakan untuk proses enkripsi ketika Budi mengirimkan pesan, begitu juga sebaliknya kunci publik Budi digunakan untuk proses enkripsi ketika Andi mengirimkan pesan. Proses ini dapat dilihat pada Gambar 8.

Gambar 8. *Server* menerima kunci publik (*server*)

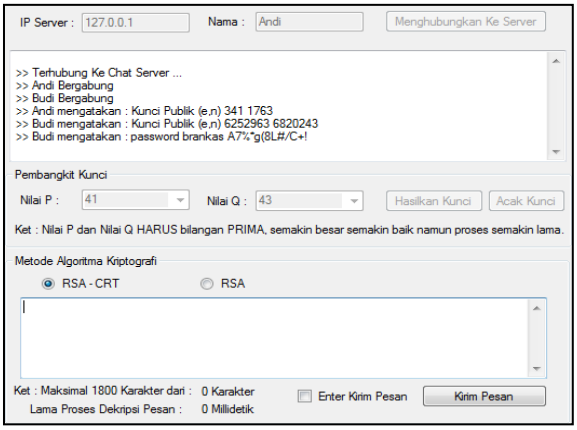
3.3. Mengolah Pesan

Pada alur proses mengolah pesan, Budi mengirimkan pesan "*password* brankas A7%*g(8L#/C+!" seperti pada Gambar 9 dengan menggunakan kunci publik Andi (e = 341, n = 1763) serta waktu proses enkripsi 0 milidetik. Pada Gambar 10, *server* menerima pesan Budi yang kemudian akan disebarkan *server* ke Andi. Andi menerima pesan Budi dengan menggunakan metode dekripsi RSA-CRT. Aplikasi mendekripsikan pesan Budi menggunakan kunci privat Andi (dP = 21, dQ = 17, qInv = 21, p = 41, q = 43) dengan waktu proses dekripsi 0 milidetik seperti pada gambar 14. Andi membalas pesan budi dengan pesan "terima kasih" seperti pada Gambar 11 dengan menggunakan kunci publik Budi (e = 6252963, n = 6820243) serta waktu proses enkripsi 0 milidetik.



Gambar 9. Budi mengirim pesan (*client*)

Gambar 10. *Server* menerima pesan Budi (*server*)



Gambar 11. Andi menerima pesan (*client*)

4. Pengujian

Pengujian algoritma kriptografi RSA-CRT pada aplikasi *instant messaging* yang utama adalah membandingkan kecepatan dekripsi antara algoritma kriptografi RSA dengan algoritma kriptografi RSA-CRT. Waktu yang digunakan untuk melakukan proses dekripsi sebagai perbandingan antar kedua algoritma kriptografi.

Pengujian dilakukan dengan menggunakan 1800 karakter *dummy*, jumlah bit n yang digunakan mulai dari 56 bit sampai 88 bit, dikarenakan pesan yang digunakan untuk melakukan pengujian yaitu 1.800 karakter sehingga mempunyai syarat nilai n harus lebih besar atau sama dengan 1.000.000. Hasil pengujian dapat dilihat pada Tabel 2.

Tabel 2. Pengujian algoritma kriptografi RSA-CRT

Bit N	Nilai P	Nilai Q	Nilai N	Nilai E	Olah Kunci (ms)	Dekripsi RSA-CRT (ms)	Dekripsi RSA (ms)
56	8.669	541	4.689.929	2.804.371	1.248	32	50
64	8.311	5.861	48.710.771	24.330.337	8.670	30	38
72	31.511	14.489	456.562.879	360.447.427	70.611	47	70
80	80.803	59.747	4.827.736.841	1.703.007.661	146.809	37	84
88	333.997	149.099	49.798.618.703	36.157.708.349	15.493.691	30	100

Berdasarkan hasil pengujian algoritma kriptografi RSA-CRT yang dapat dilihat pada Tabel 2 dapat diberi kesimpulan dengan rata-rata kecepatan yang dapat dilihat pada Tabel 3.

Tabel 3. Pengujian kecepatan algoritma kriptografi RSA-CRT

No.	Bit N	Waktu Dekripsi RSA-CRT (ms)	Waktu Dekripsi RSA (ms)	Kecepatan (RSA / RSA-CRT)
1	56	32	50	1,6 kali lebih cepat
2	64	30	38	1,3 kali lebih cepat
3	72	47	70	1,5 kali lebih cepat
4	80	37	84	2,3 kali lebih cepat
5	88	30	100	3,3 kali lebih cepat
			Rata - rata	2 kali lebih cepat

Dari pengujian kecepatan algoritma kriptografi RSA-CRT yang dapat dilihat pada Tabel 3, dapat diberi kesimpulan semakin besar jumlah bit n maka kemungkinan besar kecepatan waktu dekripsi RSA-CRT lebih cepat dan dari 5 pengujian yang dilakukan mulai bit n = 56 bit sampai bit n = 88 bit dapat disimpulkan kecepatan yang diperolah rata-rata yaitu dua kali lebih cepat ketika menggunakan dekripsi RSA-CRT dibandingkan menggunakan dekripsi RSA.

4. SIMPULAN

Implementasi algoritma kriptografi kunci publik dengan algoritma RSA-CRT pada aplikasi *instant messaging*, proses dekripsi menggunakan algoritma RSA-CRT untuk 1.800 karakter dengan bit n dari 56 bit sampai 88 bit memiliki kecapatan rata-rata dua kali lebih cepat dibandingkan menggunakan algoritma RSA. Semakin besar panjang *string*, nilai n kemungkinan besar semakin cepat waktu dekripsi menggunakan RSA-CRT.

5. REFERENSI

[1] Zuliarso, E. & Februariyanti, H., 2013. Pemanfaatan Instant Messaging untuk Aplikasi Layanan Akademik. *Jurnal Teknologi Informasi DINAMIK.* Vol. 18(2): 112-121.

[2] Menezes, A. J., Oorschot, P. C. v. & Vanstone, S. A., 1996. *Handbook of Applied Cryptography.* 1st penyunt. Boca Raton: CRC Press.

[3] Rivest, R. L., Shamir, A. & Adleman, L., 1978. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM.* Vol. 21(2): 120-126.

[4] Tilborg, H. C. A. v., 2005. *Ecnyclopedia of Cryptography and Security.* New York: SpringerScience+Business Media.

[5] Ashioba, N. C. & Yoro, R. E., 2014. RSA Cryptosystem using Object-Oriented Modeling Technique. *International Journal of Information and Communication Technology Research.* Vol. 4(2): 57-61.

[6] Garg, V. & Arunachalam, V., 2011. Architectural Analysis of RSA Cryptosystem on FPGA. *International Journal of Computer Applications*. Vol. 26(8): 30-34.

[7] Pohan, R. Y., 2007. *Algoritma RSA dengan Chinese Remainder Theorem dan Hensel Lifting.* Bandung, Jurusan Teknik Informatika ITB.

[8] Sadikin, R., 2012. *Kriptografi untuk Keamanan Jaringan dan Implementasinya dalam Bahasa Java.* Yogyakarta: Penerbit ANDI.

[9] Academy, K., 2015. *Fast Modular Exponentiation.* (Online), (https://www.khanacademy.org/computing/computer-science/cryptography/modarithmetic/a/fast-modular-exponentiation, diakses 01 November 2015)

[10] Kurose, J. F. & Ross, K. W., 2010. *Computer Networking: A Top-Down Approach.* 5th penyunt. Boston: Pearson Education.