Aplikasi Pengiriman Teks via *Email* yang Aman dengan Menggunakan Algoritma RSA-CRT

Samsul Ma'arif, Sukmawati N E

Jurusan Ilmu Komputer/ Informatika Universitas Diponegoro, Semarang
[1]sam.al.arif@gmail.com, [2]sukmawati020578@gmail.com

Abstract

The rapid advancement of technology makes human easier to do their daily activities, such as sending messages via email. On the down side, the rapid advancement of technology also makes sending a message via email becomes insecure. Therefore, a technique for securing messages is needed, so that the message would not be changed by the third party in the middle of the line, which can be done by cryptography technique. One of the cryptography algorithm which could securing texts is RSA algorithm. On the progress, the RSA algorithm still having an issue when decrypting texts, where it needs relatively more time so it has to be added with CRT (*Chinese Remainder Theorem*) algorithm to fasten the decrypting process. This study implemented RSA-CRT algorithm for sending texts via *email*. This study used *waterfall* method as model process and MatLab as programming language. Based on this study, RSA-CRT algorithm could be implemented for securing the process of sending texts via *email*. This study concludes that the elapse time of RSA-CRT algorithm is faster than RSA algoritm. The bigger the parameter value of n, affected the time of decrypting process as well as the size of texts which also got bigger.

Keywords : Cryptography, Sending Texts, RSA algorithm, CRT algorithm, RSA-CRT algorithm, *waterfall*, MatLab, *email*

Abstrak

Perkembangan teknologi yang sangat pesat membuat manusia semakin mudah dalam melakukan aktifitasnya sehari-hari contohnya pengiriman pesan via *email.* Namun dengan perkembangan teknologi juga membuat pengiriman pesan menjadi semakin tidak aman. Oleh karena itu dibutuhkan sebuah teknik untuk mengamankan pesan sehingga tidak terjadi perubahan pesan di tengah jalan oleh pihak ketiga, diantaranya dengan menggunakan kriptografi. Salah satu algoritma dalam kriptografi yang dapat berfungsi untuk mengamankan teks adalah algoritma RSA. Dalam perkembangannya proses RSA memiliki masalah saat melakukan proses dekripsinya, yaitu membutuhkan waktu yang relatif lama sehingga ditambahkan algoritma CRT (*Chinese Remainder Theorem*) untuk mempercepat waktu dekripsi. Pada penelitian ini mengimplementasikan algoritma RSA-CRT pada pengiriman teks via *email*. Penelitian ini menggunakan model proses *waterfall* yang diimplementasikan dengan menggunakan bahasa pemrograman MatLab. Berdasarkan hasil penelitian, algoritma RSA-CRT dapat diimplementasikan pada pengiriman sebuah pesan via *email*. Pada penelitian ini dapat disimpulkan bahwa kecepatan waktu algoritma RSA-CRT lebih cepat dibandingkan dengan algoritma RSA. Semakin besar parameter nilai n berdampak pada waktu proses dekripsinya begitu juga ukuran teksnya semakin besar.

Kata kunci : Kriptografi, Pengiriman Teks, algoritma RSA, algoritma CRT, algoritma RSA-CRT, *waterfall*, MatLab, *email*

1. Pendahuluan

Perkembangan teknologi yang sangat pesat dan dilengkapi dengan adanya internet membuat manusia semakin mudah dalam melakukan aktifitasnya sehari-hari. Salah satu contohnya adalah pengiriman sebuah pesan. Pengiriman sebuah pesan dengan bentuk surat dan dikirim menggunakan pos akan membutuhkan waktu yang lama. Sedangkan jika menggunakan *electronic messaging (email)*, pengiriman data, dokumen maupun gambar bisa diterima dalam hitungan detik. Namun dengan kemajuan teknologi tersebut juga membuat pengiriman pesan menjadi lebih tidak aman. Untuk meningkatkan keamanan dalam pengiriman pesan, dibutuhkan sebuah ilmu atau metode yang bisa menjaga pesan terkirim dengan aman kepada penerima yang dituju, dan tidak terjadi perubahan pesan di tengah jalan oleh pihak ketiga. Ilmu tersebut lebih dikenal dengan sebutan kriptografi.

Dalam kamus bahasa Inggris Oxford diberikan pengertian kriptografi sebagai berikut:

"Sebuah teknik rahasia dalam penulisan, dengan karakter khusus, dengan mengguanakan huruf dan karakter di luar bentuk aslinya, atau dengan metode-metode lain yang hanya dapat dipahami oleh pihak - pihak yang memproses kunci, juga semua hal yang ditulis dengan cara seperti ini." Jadi, secara umun dapat diartikan sebagai seni menulis atau memecahkan cipher [1] . Kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan, integritas data serta otentikasi[2]. Kriptografi adalah suatu ilmu yang mempelajari bagaimana cara menjaga agar data atau pesan tetap aman saat dikirimkan, dari pengirim ke penerima tanpa mengalami gangguan dari pihak ketiga. Kriptografi memiliki banyak sekali algoritma yang digunakan untuk mengamankan pesan, salah satu algoritma kriptografi yang paling sering digunakan saat ini adalah RSA.

RSA merupakan algoritma kunci publik yang diambil dari nama penemunya yaitu Rivest-Shamir-Adleman. RSA merupakan salah satu jenis algoritma dalam sistem kriptografi asimetris dimana penggunaan kunci berbeda saat enkripsi dan dekripsi. Keamanan algoritma RSA terletak pada sulitnya memfaktorkan bilangan yang besar menjadi faktor-faktor primanya. RSA terbukti secara matematika memiliki keamanan yang tinggi akibat bersandar pada persoalan faktorisasi yang sampai saat ini belum ada algoritma faktorisasi yang efisien [3]

Namun dalam proses dekripsi, RSA membutuhkan waktu yang lama karena prosesnya melibatkan bilangan-bilangan besar sehingga untuk mempercepat dekripsi RSA digunakan metode tambahan CRT (*Chinese Remainder Theorem*). Terbukti Algoritma RSA-CRT memiliki komputasi yang lebih singkat daripada Algoritma RSA biasa, yaitu sekitar 4 kali lebih cepat [3].

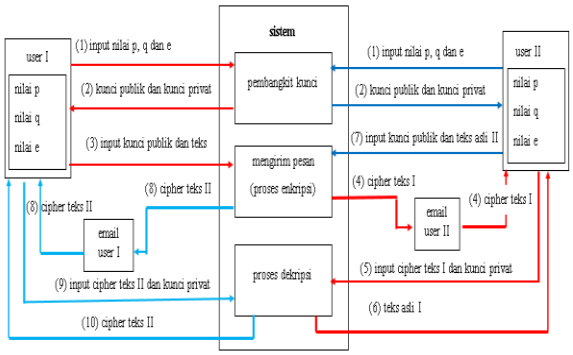
Berdasarkan penjelasan di atas, maka pada penelitian ini membuat sebuah aplikasi yang menerapkan Algoritma RSA-CRT untuk mengamankan data teks yang kemudian dikirim via *email*. Setelah itu dibandingkan apakah Algoritma RSA-CRT memiliki kecepatan komputasi yang lebih baik dibandingkan dengan Algoritma RSA biasa.

2. Metode

Aplikasi Pengiriman Teks via *Email* yang Aman dengan Menggunakan Algoritma RSA-CRT ini terbagi menjadi tiga proses yaitu proses pembangkitan kunci, enkripsi, dan dekripsi. Proses pembangkitan kunci merupakan proses mencari K_{publik} dan K_{privat} dengan *input* berupa nilai p, q, dan e. Proses pembangkitan kunci dilakukan sebelum melakukan proses enkripsi maupun proses dekripsi dilakukan oleh pengirim maupun penerima pesan sehingga tanpa proses melakukan pembangkitan kunci, proses enkripsi maupun dekripsi mustahil dilakukan. Setelah K_{publik} ditemukan selanjutnya dilakukan proses penyandian (enkripsi) dari teks asli yang menghasilkan cipher teks yang selanjutnya dikirim via *email* sehinggal ketika pesan sampai kepada penerima, teks asli yang semula dimasukkan oleh pengirim langsung berubah menjadi cipher teks. Sedangkan proses dekripsi dimulai ketika cipher teks yang telah diterima dari *email*. Setelah pesan diterima, penerima pesan yang telah memiliki K_{privat}, melakukan proses dekripsi sehingga menghasilkan teks asli.

Ada 2 pengguna (*user*) pada aplikasi ini, yaitu *user* I dan *user* II. *User* I dan *user* II memiliki nilai p, q, e yang sama, keduanya melakukan pembangkitan kunci dengan (1) menginputkan nilai p, q dan e sehingga menghasilkan (2) kunci publik dan kunci privat. Ketika *user* I berperan sebagai pengirim dan *user* II sebagai penerima pesan, *user* I (3) memasukan teks asli dan kunci publik dan mengirim pesan via *email* kepada *user* II. Proses penyandian yang mengubah teks asli menjadi cipher teks terjadi saat proses pengiriman pesan via *email* sehingga pesan yang diterima di *email user* II telah berbentuk (4) cipher teks. Setelah menerima pesan yang berisi cipher teks, *user* II melalukan proses dekripsi dengan (5) memasukkan cipher teks dan kunci privat. Setelah proses dekripsi, sistem akan menampilkan hasilnya yaitu (6) teks asli dari *user* I. Hal yang sama juga dilakukan ketika *user* II menjadi pengirim dan *user* I sebagai penerima pesan.

Arsitektur sistem dari Aplikasi Pengiriman Teks via *Email* yang Aman dengan Menggunakan Algoritma RSA-CRT dapat dilihat pada Gambar 1.

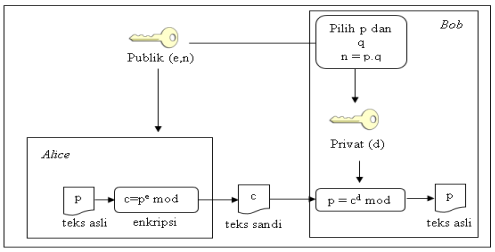


Gambar 1 Arsitektur Aplikasi Pengiriman Teks via *Email* yang Aman dengan Menggunakan Algoritma RSA-CRT

Algoritma RSA

RSA merupakan algoritma kriptografi asimetri, dimana kunci yang digunakan untuk mengenkripsi

pesan berbeda dengan kunci yang digunakan untuk mendekripsi pesan. Kunci yang digunakan untuk mengenkripsi pesan disebut dengan kunci publik, sedangkan kunci untuk mendekripsi pesan disebut dengan kunci privat. RSA adalah salah satu algoritma kriptografi yang menggunakan konsep kriptografi kunci publik. Algoritma RSA dapat dilihat pada Gambar 2.



Gambar 2. Algoritma RSA

Algoritma RSA memiliki 3 langkah yaitu pembangkit kunci RSA, enkripsi RSA, dan dekripsi RSA [4] :

1. Pembangkit Kunci
   Sebelum melakukan enkripsi maupun dekripsi dengan menggunakan algoritma RSA, terlebih dahulu dilakukan pembangkitan sepasang kunci, yaitu kunci publik dan kunci privat.
   a. Memilih 2 bilangan prima besar untuk nilai p dan q. Direkomendasikan p dan q memiliki panjang 512 bit (64 byte atau 64 nominal angka).
   b. Menghitung nilai modulus n
      $n = p \times q$.....(1)
   c. Menghitung fungsi Euler $\phi(n)$
      $\phi(n) = (p\text{-}1) \times (q\text{-}1)$…….....(2)
   d. Memilih nilai integer e acak untuk mencari kunci publik, dengan syarat memenuhi *greater common divisor* (gcd)
      $\gcd(e, \phi(n)) = 1,\ 1 < e < \phi(n)$..... (3)
      Setelah ditentukan nilai e maka $K_{publik} = (e, n)$.
   e. Menghitung kunci privat $d = e^{-1}$ sehingga
      $d \times e = 1 \pmod{\phi(n)}$.....(4)
      Setelah nilai d ditemukan, maka $K_{privat} = d$.
2. Enkripsi
   Enkripsi RSA adalah proses mengubah pesan asli (P) dengan menggunakan kunci publik (e,n) sehingga berubah menjadi cipher teks (C).
   $C = P^e \bmod n$ .....(5)
3. Dekripsi
   Dekripsi RSA adalah proses mengubah cipher teks (C) dengan menggunakan kunci privat (d) sehingga berubah menjadi pesan asli (P).
   $P = C^d \bmod n$ .....(6)

Chinese Remainder Theorem (CRT)

Chinese Remainder Theorem (CRT) merupakan problem klasik pada teori bilangan, CRT digunakan salah satunya sebagai varian algoritma RSA yang disebut algoritma RSA-CRT. CRT diformulasikan sebagai penyelesaian permasalahan kongruen dengan modulus berbeda. Secara formal CRT menyelesaikan sistem persamaan kongruen berikut ini [3].

$$x \equiv a_1 \bmod r_1,$$
$$x \equiv a_2 \bmod r_2,$$
$$\ldots$$
$$x \equiv a_k \bmod r_k \quad \ldots\ldots(7)$$

dengan nilai $a_1, \ldots, a_k$ dan $r_1, \ldots, r_k$ adalah bilangan bulat yang sudah diketahui dan x tidak diketahui. Teorema CRT menyatakan persamaan (7) memiliki solusi yang unik bila r_1, r_2 sampai r_k adalah saling prima relatif [2].

Untuk menemukan nilai x dapat menggunakan prosedur berikut :

1. Menghitung $R = r_1 \times r_2 \times r_k$……....(8)
2. Mengitung $R_1 = R / r_1$, $R_2 = R / r_2$,…. $R_k = R / r_k$
   ……....(9)
3. Mencari invers perkalian $R_1^{-1}, R_2^{-1}, \ldots R_k^{-1}$ dengan menggunakan modulus $r_1, r_2, \ldots r_k$.. Notasi invers perkalian R_k adalah y_k.
   $R_k \times y_k \equiv 1 \pmod{r_k}$……(10)
4. Mencari nilai x dengan menghitung.
   $x = (a_1 \times R_1 \times y_1 + \ldots + a_k \times R_k \times y_k) \bmod R$..(11)

RSA Dengan CRT (RSA-CRT)

Algoritma RSA dapat dimodifikasi dengan menggunakan teorema CRT disebut dengan Algoritma RSA-CRT. Terbukti Algoritma RSA-CRT memiliki waktu komputasi yang lebih singkat daripada Algoritma RSA biasa, yaitu sekitar empat kali lebih cepat [3].

Pada dasarnya Algoritma RSA-CRT sama dengan Algoritma RSA biasa namun memanfaatkan teorema CRT untuk memperpendek ukuran bit eksponen deksipsi d dengan cara menyembunyikan d pada sistem kongruen sehingga mempercepat waktu dekripsi.

Algoritma RSA-CRT dibagi menjadi 3 langkah :

1. Pembangkit Kunci RSA-CRT
   Sesuai dengan Algoritma RSA, Algoritma RSA-CRT juga dilakukan pembangkitan sepasang kunci, yaitu kunci publik dan kunci privat sebelum melakukan enkripsi maupun dekripsi. Langkah-langkah pembangkit kunci RSA-CRT adalah sebagai berikut :

a. Memilih 2 bilangan prima yang besar untuk nilai nilai *p* dan *q*. Direkomendasikan *p* dan *q* memiliki panjang 512 bit (64 byte atau 64 digit angka).
b. Menghitung nilai modulus *n* menggunakan persamaan (1).
c. Menghitung fungsi Euler $\phi(n)$ menggunakan persamaan (2).
d. Memilih nilai integer *e* acak untuk mencari kunci publik, dengan syarat memenuhi *greater common divisor* (gcd) menggunakan persamaan (3). $K_{publik} = (e, n)$.
e. Menghitung nilai d dengan menggunakan persamaan (4).
f. Menghitung nilai dP, dQ, qInv dengan persamaan :
   $dP = d \bmod (p - 1)$……...(12)
   $dQ = d \bmod (q - 1)$……...(13)
   $qInv = q^{-1}$ pada Z_p ……...(14)

Setelah nilai dP, dQ, qInv ditemukan maka
$K_{privat} = (dP, dQ, qInv, p, q)$ ……...(15)

2. Enkripsi RSA-CRT
   Kunci publik Algoritma RSA-CRT sama dengan Algoritma RSA yaitu (*e*, *n*) sehingga enkripsi tidak mengalami perubahan yaitu dengan menggunakan fungsi eksponensial modular sesuai dengan persamaan (5).

3. Dekripsi RSA-CRT
   Perhitungan proses dekripsi RSA-CRT dengan $C = P^e \bmod n$ dan kunci privat (dP, dQ, qInv, *p*, *q*) adalah sebagai berikut :
   a. $m_1 = C^{dP} \bmod p$ .....(16)
   b. $m_2 = C^{dQ} \bmod q$.....(17)
   c. $h = qInv.(m_1 - m_2) \bmod p$.....(18)
   d. $P = m_2 + h.q$ .....(19)

Berikut ada proses yang terjadi pada Aplikasi Pengiriman Teks via *Email* yang Aman dengan Menggunakan Algoritma RSA-CRT :

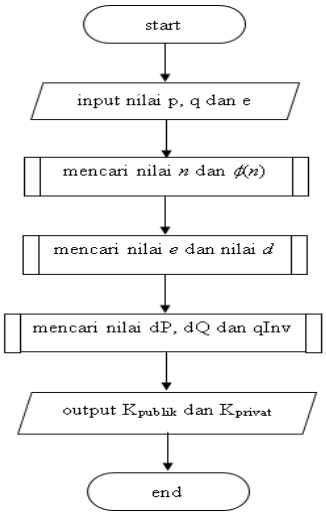
1. Pembangkit kunci
   Proses pembangkitan kunci RSA-CRT berfungsi untuk mencari kunci publik dan kunci privat. Proses ini dimulai dengan *input* berupa bilangan prima p dan q. Lalu mencari nilai n dan $\phi(n)$. Selanjutnya mencari nilai e dan nilai d. Setelah diperoleh nilai *n*, $\phi(n)$, *e* dan *d*, selanjutnya dilakukan proses menghitung nilai dP, dQ, qInv. Alur proses membangkitkan kunci public dan kunci privat algoritma RSA-CRT dapat dilihat pada Gambar 3.

   Proses pembangkit kunci terdiri dari 3 sub proses antara lain :

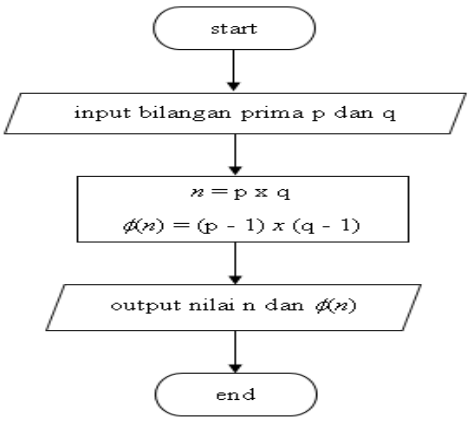
   a. Mencari nilai *n* dan $\phi(n)$
      Proses ini memiliki *input* berupa nilai p dan q yang menghasilkan *output* berupa nilai *n* dan $\phi(n)$. Alur proses mencari nilai *n* dan $\phi(n)$ dapat dilihat pada Gambar 4.
   b. Mencari nilai *e* dan nilai *d*
      Proses ini memiliki *input* berupa nilai e dan $\phi(n)$ yang menghasilkan *output* berupa nilai *e* dan nilai *d*. Alur proses mencari nilai *e* dan nilai *d* dapat dilihat pada Gambar 5.
   c. Mencari nilai dP, dQ dan qInv
      Proses ini memiliki *input* berupa nilai p, q dan d yang menghasilkan *output* berupa nilai dP, dQ dan qInv. Alur proses mencari nilai dP, dQ dan qInv dapat dilihat pada Gambar 6.

2. Mengirim pesan via email
   Proses mengirim pesan memiliki *input* berupa data *email* teks asli dan K_{publik} (e, n). Setelah itu diproses sehingga menghasilkan notifikasi menandakan apakah pesan berhasil dikirim atau gagal dikirim. Alur proses mengirim pesan dapat dilihat pada gambar 7. Terdapat sub proses pada proses mengirim pesan via *email* yakni proses enkripsi pesan. Proses enkripsi dengan *input* teks asli dan K_{publik} (e, n). Pertama pada enkripsi adalah mengubah teks asli menjadi angka dalam ASCII. Setelah itu diproses dengan rumus enkripsi RSA-CRT sehingga menghasilkan *output* berupa cipher teks. Alur proses enkripsi dapat dilihat pada Gambar 8.

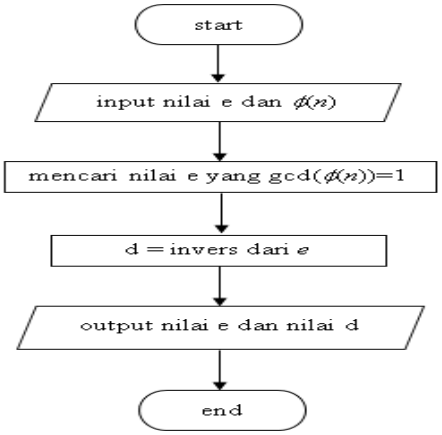
3. Dekripsi
   Setelah cipher teks diterima dalam *email* selanjutnya dilakukan dekripsi dengan *input* berupa cipher teks dan kunci privat (dP, dQ, qInv, p dan q) dan *output* berupa teks asli. Alur proses dekripsi dapat dilihat pada Gambar 9.



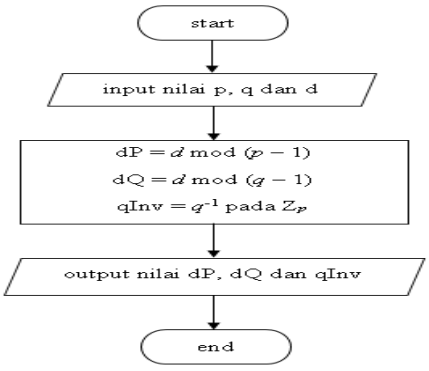
Gambar 3 Alur Proses Pembangkit Kunci



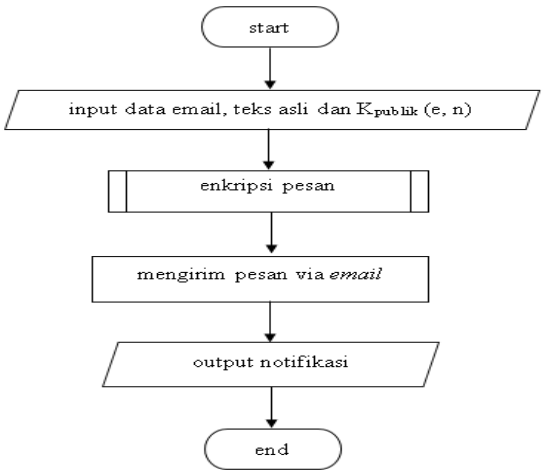
Gambar 4 Alur Proses Mencari Nilai *n* dan $\phi(n)$



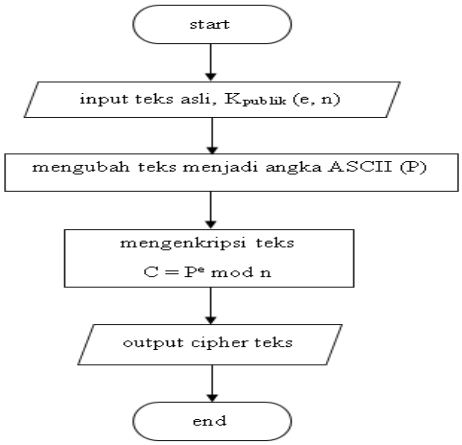
Gambar 5 Alur Proses Mencari Nilai *e* dan Nilai *d*



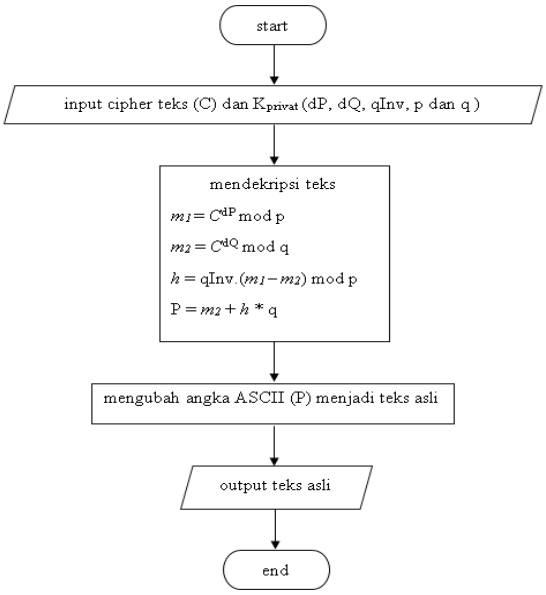
Gambar 6 Alur Proses Mencari Nilai dP, dQ dan qInv



Gambar 7 Alur Proses Mengirim Pesan via *Email*



Gambar 8 Alur Proses Enkripsi Pesan



Gambar 9 Alur Proses Dekripsi Pesan

3. Hasil dan Analisis

Contoh studi kasus, POLRI ingin mengirim pesan kepada POLSEK Semarang isinya "Akan dilakukan penggrebekan terhadap transaksi narkoba di daerah Simpang Lima Semarang pada pukul 23.00 WIB". Agar pesan POLRI terkirim dengan aman sampai kepada POLSEK Semarang, POLRI melakukan pengamanan pesan menggunakan algoritma RSA-CRT sebelum pesan tersebut dikirim via *email*. Berdasarkan konsep algoritma RSA-CRT, POLRI harus membuat kunci publik yang diberikan ke POLSEK Semarang serta kunci privat yang akan digunakan untuk dekripsi pesan. Terdapat 3 langkah untuk menggunakan algoritma RSA-CRT :

Melakukan Pembangkit Kunci

Pembangkitan kunci dilakukan sebelum melakukan enkripsi maupun dekripsi pesan. Langkah-langkahnya sebagai berikut :

a. Memilih 2 bilangan prima untuk nilai *p* adalah 29 sedangkan untuk nilai *q* adalah 23.

b. Menghitung nilai modulus *n* menggunakan persamaan (1).
$n = 29 * 23 =$ **667**

c. Menghitung fungsi Euler $\phi(n)$ menggunakan persamaan (2).
$\phi(n) = (p - 1) \; x \; (q - 1)$
$= (29 - 1) \times (23 - 1)$
$=$ **616**

d. Memilih nilai integer *e* acak, dengan syarat memenuhi *greater common divisor* (gcd) menggunakan persamaan (3). Nilai e yang dipilih adalah **81**.

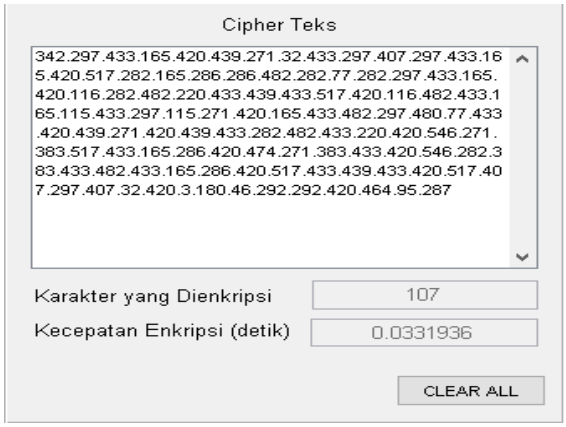
e. Menghitung nilai d dengan menggunakan persamaan (4).
$d = 81^{-1}$ terhadap Z_{616}. Maka nilai $d = 905$.

f. Menghitung nilai dP, dQ, qInv dengan persamaan :
$dP = d \bmod (p - 1) = 905 \bmod (29 - 1) =$ **9**
$dQ = d \bmod (q - 1) = 905 \bmod (23 - 1) =$ **3**
$qInv = q^{-1}$ pada $Z_p = 23^{-1}$ pada Z_{29}. Maka nilai qInv = 24.

Enkripsi

Rumus melakukan enkripsi RSA-CRT adalah $C = P^e \bmod n$. Namun untuk mendapatkan nilai P dari teks "Akan dilakukan penggrebekan terhadap transaksi narkoba di daerah Simpang Lima Semarang pukul 23.00 WIB". Teks tersebut dilakukan perubahan kedalam bentuk nilai ASCII per karakter sebelum dilakukan proses enkripsi.

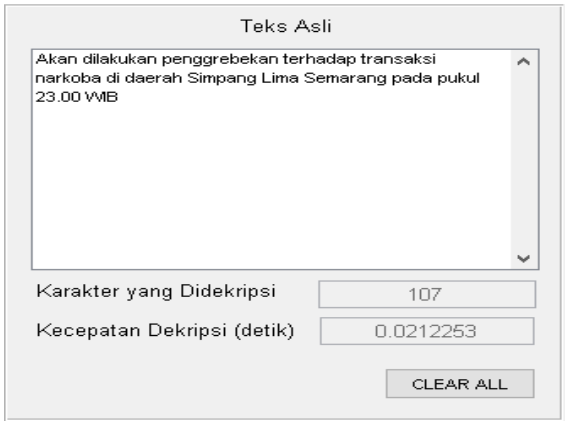
Setelah dirubah menjadi ASCII dilakukan proses enkripsi dengan rumus $C = P^e \bmod n$ dan Kunci Publik (81,667). Hasilnya dapat dilihat pada Gambar 10.



Gambar 10 Tampilan Hasil Enkripsi Pesan Aplikasi Pengiriman Teks via *Email* Menggunakan Algoritma RSA-CRT

Dekripsi

Dekripsi pesan adalah mengembalikan teks cipher menjadi teks asli dengan K_{privat} = (9, 3, 24, 29, 23). Namun sebelum dilakukan dekripsi, dilakukan terlebih dahulu proses pemisahan string ".". Setelah terpisah maka diubah menjadi ASCII teks. Selanjutnya setelah menjadi nilai ASCII lalu dilakukan perubahan ASCII menjadi karakter asli. Hasilnya dapat dilihat pada Gambar 11.



Gambar 11 Tampilan Hasil Dekripsi Pesan Aplikasi Pengiriman Teks via *Email* Menggunakan Algoritma RSA-CRT

Pengujian algoritma RSA-CRT dilakukan dengan melakukan perbandingan proses dekripsi antara algoritma RSA-CRT dengan algoritma RSA, apakah algoritma RSA-CRT memiliki kecepatan proses dekripsi yang lebih baik dibandingkan dengan algoritma RSA dengan menggunakan metode *fast modular exponentiation* [5]. Pengujiannya dilakukan

dengan mengenkripsi dan mendekripsi karakter *dummy* "undip".

Selain menggunakan karakter *dummy*, pengujian juga dilakukan dengan mengubah parameter nilai n yang memiliki panjang 32, 40 dan 48 bit. Pengujian Algoritma RSA-CRT dapat dilihat pada Tabel 1.

Tabel 1 Pengujian Algoritma RSA-CRT

Bit n	Nilai p	Nilai q	Nilai e	Waktu Olah Kunci (ms)	Waktu Dekripsi RSA (ms)	Waktu Dekripsi RSA-CRT (ms)	Perbandingan Waktu Dekripsi RSA dg RSA-CRT
32	71	43	757	5.419	121.945	40.143	3 kali
40	479	37	7571	4.763	116.959	43.465	3 kali
48	313	251	50347	5.134	108.865	40.935	3 kali

Berdasarkan Tabel 1 menunjukkan bahwa algoritma RSA-CRT ternyata hampir tiga kali lebih cepat dibandingkan dengan algoritma RSA biasa. Hasil pengujian yang dilakukan pada Tabel 1 tersebut menunjukkan bahwa penelitian ini sesuai dengan penelitian-penelitian sebelumnya yang menyatakan bahwa algoritma RSA-CRT lebih cepat dibandingkan dengan algoritma RSA biasa.

Pengujian juga dilakukan untuk mengetahui ukuran teks sebelum dienkripsi dan hasil setelah dilakukan enkripsi dan dekripsi. Teks tersebut disimpan dalam bentuk .txt agar lebih mudah mengetahui ukuran dari teks asli maupun cipher teks yang diuji dengan menguji nilai n yang memiliki panjang 32, 40 dan 48 bit. Pengujian ukuran teks dapat dilihat pada Tabel 2.

Tabel 2 Pengujian Ukuran Teks

Bit n	Nilai p	Nilai q	Nilai e	Nilai n	Ukuran Teks Asli (byte)	Ukuran Teks Hasil Enkripsi (byte)	Ukura Teks Hasil Dekripsi (byte)
32	71	43	757	3053	5	20	5
40	379	67	25393	25393	5	26	5
48	541	479	159419	259139	5	31	5

Hasil pengujian dari Tabel 2 menunjukkan bahwa ukuran teks asli setelah diubah menjadi cipher teks memiliki ukuran yang lebih besar, tetapi setelah dilakukan dekripsi ukuran teks berubah menjadi seperti semula. Ukuran teks memiliki ukuran bervariatif, semakin besar nilai n maka semakin besar pula ukuran teks dikarenakan setelah proses enkripsi, satu karakter yang semula hanya 1 byte bisa berubah ukuran hingga enam kali lipat.

4. Kesimpulan

Berdasarkan hasil pengujian dan analisis yang telah dilakukan terhadap beberapa sampel data, maka dapat diambil kesimpulan sebagai berikut :

1. Implementasi algoritma RSA-CRT pada Aplikasi Pengiriman Teks via *Email* yang Aman dengan Menggunakan Algoritma RSA-CRT pada bahasa pemrogaman MatLab berhasil dibangun dan sesuai dengan perhitungan manual.
2. Dekripsi dari Aplikasi Pengiriman Teks via *Email* yang Aman dengan Menggunakan Algoritma RSA-CRT memiliki kecepatan rata – rata hampir tiga kali lebih cepat dibandingkan dengan menggunakan algoritma RSA biasa sehingga sesuai dengan dasar teori yang menyatakan bahwa algoritma RSA-CRT memiliki waktu komputasi dekripsi yang lebih cepat dibandingkan dengan algoritma RSA biasa.
3. Semakin besar panjang string nilai n maka akan membutuhkan waktu yang semakin lama pula dalam proses dekripsi Algoritma RSA-CRT. Selain itu nilai n yang semakin besar juga membuat ukuran teks semakin besar.

5. Daftar Pustaka

[1]. Talbot, J. & Welsh, D., 2006. *Complexity and Cryptography*. New York: Cambrigde University Press.

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

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

[4]. Rivest, R. R., Shamir, A. & Adleman, L., 1983. A method for obtaining digital signatures and public-key cryptosystems. s.l.:Commun.

[5]. Arief, A., 2016. *Implementasi Algoritma Kriptografi RSA - CRT pada Aplikasi Instant Messaging.* Semarang: Universitas Diponegoro.