Penerapan *Chinese Remainder Theorem* untuk Mendekripsi *Ciphertext* pada RSA *Cryptosystem*

Vincentius Lienardo - 13518081
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13518081@std.stei.itb.ac.id

***Abstrak*—Pada era saat ini, merupakan hal yang sangat penting bahwa proses enkripsi dan dekripsi wajib dilakukan untuk mengirimkan data dari satu tempat ke tempat yang lain melalui internet untuk mencegah pengaksesan yang tidak sah (*illegal*). Penerapan *Chinese Remainder Theorem* untuk enkripsi dan dekripsi digunakan oleh banyak *crypto libraries* seperti OpenSSL, Java, dan .NET untuk mengoptimisasi algoritme sehingga lebih efisien.**

***Kata Kunci*—Kriptografi, Enkripsi, Dekripsi, RSA, *Chinese Remainder Theorem*, *Plaintext*, *Ciphertext*, *Public Key*, *Private Key*.**

I. Pendahuluan

Teknologi internet sebagai media pertukaran informasi telah dipakai oleh hampir semua orang, di mana informasi menjadi sesuatu yang sangat berharga, informasi sering menjadi target serangan oleh para *cracker*. Karena itu, keamanan suatu informasi menjadi sesuatu yang harus dijaga dengan baik. Pengamanan informasi berfungsi untuk melindungi informasi agar siapa pun yang tidak berhak tidak dapat membaca, mengubah, atau menghapus informasi tersebut. Di sinilah peranan kriptografi untuk mengatasi hal yang tidak diinginkan tersebut.

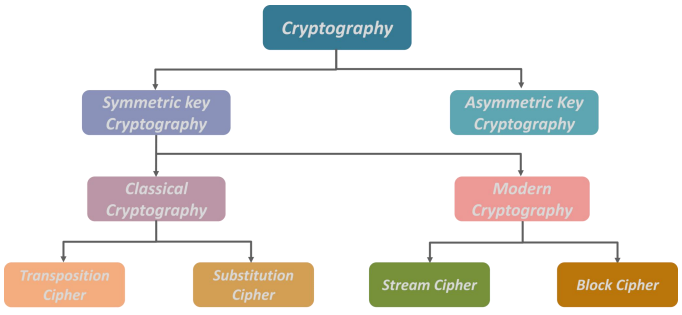
Kriptografi memegang peranan yang sangat penting dalam proses pengamanan data atau informasi. Dengan adanya kriptografi, informasi yang berupa pesan atau data yang dianggap rahasia dapat disembunyikan menggunakan teknik penyandian sehingga hanya dimengerti oleh pembuat dan penerimanya saja. Kriptografi membuat pesan yang dikirimkan dari satu tempat ke tempat lain menjadi aman dengan adanya enkripsi. Enkripsi adalah proses yang dilakukan untuk mengamankan sebuah pesan (*plaintext*) menjadi pesan yang tersembunyi (*ciphertext*). Namun, jika ada seseorang (kriptanalis) selain pemberi dan penerima yang berusaha untuk mendekripsi pesan atau data yang dikirimkan dan ternyata berhasil, maka pesan atau data yang dikirimkan sudah tidak dapat dikatakan aman lagi.

Pada awalnya, kriptografi dikembangkan dengan algoritme sandi kunci simetris, yaitu kunci yang dipakai untuk enkripsi dan dekripsi hanya satu, dalam hal ini sama. Algoritme kunci simetri yang terkenal di zaman itu adalah algoritme *Caesar Cipher*, di mana setiap huruf yang ada digeser dan menggantikan huruf lainnya. Seiring dengan perkembangan zaman dan teknologi, mulai diciptakan algoritme sandi kunci asimetris yang memiliki kunci publik (*public key*) dan kunci privat (*private key*) dan kunci yang digunakan untuk proses enkripsi dan dekripsi sudah berbeda. Data di-*encrypt* dengan menggunakan *private key* dan di-*encrypt* menggunakan *public key*. Algoritme yang sudah menggunakan prinsip algorime sandi kunci asimetris adalah algoritme Rivest–Shamir–Adenan (RSA). Namun, teknologi ada untuk terus dikembangkan, dilakukan optimisasi terus menerus, dan/atau mencari solusi terhadap suatu permasalahan dengan lebih efektif. *Chinese Remainder Theorem* dalam pengaplikasiannya di RSA membuat proses enkripsi dan dekripsi menjadi lebih cepat. Di sinilah maksud penulis untuk menghubungkan antara *Chinese Remainder Theorem* dan algoritme RSA.

II. Landasan Teori

A. *Kriptografi*

Kriptografi (*cryptography*) berasal dari bahasa Yunani yang terdiri dari dua kata, yaitu '*kryptos*' yang berarti tersembunyi dan '*graphia*' yang berarti sesuatu yang tertulis. Menurut kedua kata tersebut, dapat disebut kriptografi sebagai sesuatu yang tertulis secara tersembunyi. Di dalam kriptografi, *Plaintext* merupakan pesan yang dapat dibaca, sedangkan *ciphertext* merupakan pesan yang tidak dapat dibaca. Proses untuk mengubah *plaintext* menjadi *ciphertext* disebut enkripsi, sedangkan proses yang mengubah *ciphertext* menjadi *plaintext* disebut dekripsi. Kedua proses ini membutuhkan sejumlah informasi rahasia yang biasa disebut kunci (*key*).



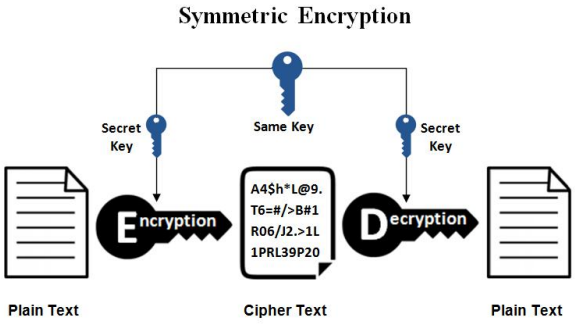
Gambar 1. Bagan Kriptografi

Sumber: https://www.edureka.co/blog/what-is-cryptography/
(diakses pada tanggal 29 November 2019)

Berdasarkan kunci yang digunakan untuk proses enkripsi dan dekripsi, kriptografi dapat dibedakan menjadi dua macam:

1. *Algoritme Sandi Kunci Simetris*

Pada algoritme sandi kunci simetris, kunci yang digunakan untuk proses enkripsi sama dengan kunci yang digunakan untuk proses dekripsi. Cara kerjanya adalah *plaintext* diubah menjadi *ciphertext* dengan sebuah *secret key,* lalu apabila ingin mendekripsi, tinggal digunakan *secret key* yang sama sehingga dapat berubah menjadi *plaintext* kembali. Pengirim dan penerima pesan atau data juga akan memiliki kunci rahasia (*secret key*) yang sama. Pengirim akan mengenkripsi pesan atau data menggunakan kunci rahasia yang sama dengan penerima yang mendekripsi pesan atau data tersebut sehingga kunci rahasia ini berperan sangat penting dalam keamanan data di algoritme sandi kunci simetris ini. Algoritme sandi kunci simetris menjadi awal mula perkembangan proses enkripsi dan dekripsi di dalam dunia kriptografi.

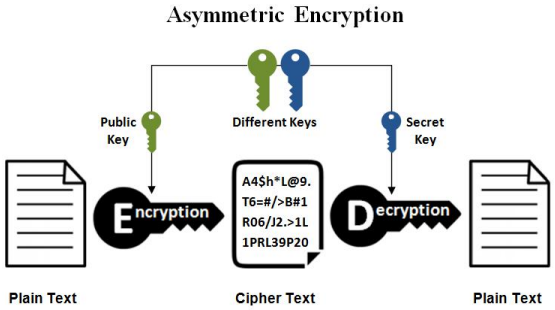


Gambar 1. Algoritme Sandi Kunci Simetris
Sumber: https://www.ssl2buy.com/wiki/symmetric-vs-asymmetric-encryption-what-are-differences
(diakses pada tanggal 29 November 2019)

Contoh algoritme yang mengimplementasikan kunci simetris adalah *Rivest Cipher*, *Blowfish*, *Twofish*, MARS, *International Data Encryption Algorithm* (IDEA), *Data Encryption Standard* (DES), *One Time Pad* (OTP), dan *Advanced Encryption Standard* (AES).

2. *Algoritme Sandi Kunci Asimetris*

Pada algoritme sandi kunci asimetris, kunci yang digunakan untuk proses enkripsi tidak sama dengan kunci yang digunakan untuk proses dekripsi. *Public key* di sini merupakan kunci yang digunakan untuk enkripsi dan dapat diketahui oleh siapa pun, sedangkan kunci untuk dekripsi hanya diketahui oleh pengirim dan penerima pesan atau data. Pada dasarnya, setiap pengguna memiliki satu pasang kunci, yaitu *public key* untuk mengenkripsi dan *private key* untuk mendekripsi. *Public key* tersebar secara luas dan pihak lain bisa melihatnya, sedangkan *private key* hanya diketahui oleh penerimanya saja. Syarat agar pesan bisa didekripsi adalah dengan menggunakan *private key* yang sesuai dengan *public key*-nya karena keduanya berhubungan secara matematis.



Gambar 2. Algoritme Sandi Kunci Asimetris
Sumber: https://www.ssl2buy.com/wiki/symmetric-vs-asymmetric-encryption-what-are-differences
(diakses pada tanggal 29 November 2019)

Contoh algoritme yang mengimplementasikan kunci asimetris adalah *Digital Signature Algorithm* (DSA), Rivest–Shamir–Adenan (RSA), Diffie–Hellman (DH), *Elliptic Curve Cryptography* (ECC), dan Kriptografi Quantum.

Kelebihan dari algoritme sandi kunci asimetris (*asymmetric key algorithms*) dibandingkan dengan algoritme sandi kunci simetris (*symmetric key algorithms*) yaitu dapat digunakan untuk mengamankan pengiriman *symmetric key*; hanya *private key* yang harus dijaga kerahasiaannya oleh pemberi dan penerima pesan atau data; dan pasangan *public key* dan *private key* tidak perlu diubah. Sedangkan kelemahan dari *asymmetric key* adalah ukuran *key* yang relatif lebih besar; enkripsi dan dekripsi membutuhkan waktu yang relatif lebih lama karena menggunakan bilangan–bilangan yang besar; dan ukuran *ciphertext* yang dienkripsi jauh lebih besar daripada *plaintext*.

B. *Aritmetika Modulo* (*Modular Arithmetic*)

1. *Sifat Pembagian pada Bilangan Bulat*

Misalkan a dan $b \in \mathbf{Z}$, $a \neq 0$. a dikatakan habis membagi b (*a divides b*) jika terdapat bilangan bulat c sedemikian sehingga $b = ac$. Notasinya adalah $a \mid b$ jika $b = ac$, $c \in \mathbf{Z}$ dan $a \neq 0$.

2. *Teorema Euclidean*

Aritmetika modulo merupakan operasi bilangan bulat yang menghasilkan sisa dari hasil bagi dua buah bilangan. Misalkan a dan m bilangan bulat, maka operasi modulo dapat dinotasikan sebagai $a \bmod m = r$, sedemikian sehingga

$$a = km + r, \qquad 0 \leq r < m, k \in \mathbf{Z}$$

m disebut sebagai modulo atau modulus dengan hasil dari operasi modulo (r) berada di *range* $\{0, 1, 2, 3, \ldots, m - 1\}$.

3. *Pembagi Bersama Terbesar* (*PBB*)

Misalkan a dan $b \in \mathbf{Z}$. $a, b \neq 0$. Pembagi bersama terbesar (PBB – *greatest common divisor* atau GCD) dari a dan b adalah bilangan bulat terbesar d sedemikian sehingga $d \mid a$ dan $d \mid b$. Dalam hal ini, dapat dinyatakan PBB(a, b) = d.

4. *Kongruen*

Misalkan a dan $b \in \mathbf{Z}^+$, a dan b kongruen dalam modulo m jika dan hanya jika $m \mid (a - b)$. Apabila diubah dalam bentuk notasi, maka akan menjadi:

$$a \equiv b \pmod{m} \Leftrightarrow m \mid (a - b)$$

5. *Modulo Inverse*
   Jika a dan m relatif prima (dalam hal ini GCD(a, m) = 1) dan $m > 1$, maka *inverse* dari a (mod m) ada. Balikan dari a (mod m) adalah x, dengan $x \in \mathbf{Z}$ sedemikian sehingga $xa \equiv 1 \pmod{m}$, dalam notasi lainnya adalah $a^{-1} \pmod{m} = x$.

6. *Kekongruenan Lanjar*
   Bentuk $ax \equiv b \pmod{m}$ untuk $m \in \mathbf{Z}^+$ dengan $a, b, x \in \mathbf{Z}$ dapat diubah menjadi:

   $$ax = b + km \Rightarrow x = (b + km)/a$$

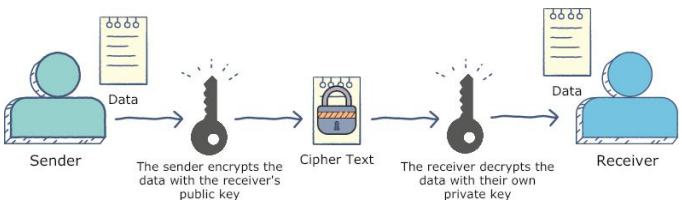
   Persamaan ini dapat digunakan untuk mencari bilangan bulat apa saja yang merupakan solusi dari persamaan yang diberikan atau mencari bilangan apa saja yang kongruen dengan bilangan yang dimaksud.

7. *Teorema Fermat* (*Fermat's Little Theorem*)
   Jika p adalah bilangan prima dan $a \in \mathbf{Z}$ yang tidak habis dibagi dengan p, yaitu GCD(a, p) = 1, maka

   $$a^{p-1} \equiv 1 \pmod{p}$$

C. *Algoritme RSA*

Algoritme RSA merupakan algoritme kriptografi berbasis kunci publik (asimetris). Nama algoritme RSA berasal dari penemu-penemunya, yang bernama Ron Rivest, Adi Shamir, dan Len Adleman pada tahun 1976. Algoritme RSA merupakan pengembangan dari Teori Bilangan terutama pada konsep bilangan prima dan aritmetika modulo. Algoritme RSA ini mempunyai kelebihan yaitu pada tingkat kesulitannya memfaktorkan bilangan bulat **Z** yang sangat besar menjadi dua buah bilangan prima (faktor–faktor primanya). Dengan melakukan pemfaktoran ini, apabila berhasil, akan memperoleh kunci privat (*private key*). Selama ini, pemfaktoran sebuah bilangan bulat **Z** yang sangat besar menjadi faktor–faktor primanya merupakan hal yang sangat sulit, dibutuhkan algoritme yang mangkus untuk memecahkannya. Oleh karena itu, keamanan algoritme RSA cukup baik. Algoritme RSA merupakan kunci publik yang penggunaannya sangat luas dan digunakan oleh banyak orang karena keamanannya cukup terjamin. Pohon faktor merupakan suatu metode yang digunakan dalam melakukan pemfaktoran. Bilangan yang akan difaktorkan pun harus semakin besar agar pemfaktoran semakin sulit untuk dilakukan, hal ini menjadi penentu dalam kekuatan algoritme RSA.



Gambar 3. Visualisasi Algoritme RSA
Sumber: https://www.educative.io/edpresso/what-is-the-rsa-algorithm
(diakses pada tanggal 30 November 2019)

Penerapan algoritme RSA dibagi menjadi tiga proses:

1. *Pembangkitan Kunci* (*Key Generation*)
   Langkah–langkah dalam melakukan pembangkitan kunci:
   1. Pilih dua bilangan prima sembarang, misalnya a dan b (nilai dari a dan b merupakan rahasia).
   2. Hitung nilai $n = ab$ (nilai dari n tidak rahasia).
   3. Hitung $m = (a - 1)(b - 1)$. Setelah didapatkan nilai m, a dan b dapat dihapus untuk meminimalisir hal–hal yang tidak diinginkan).
   4. Pilih sebuah bilangan $e \in \mathbf{Z}$ yang relatif prima terhadap m sebagai *public key*. Syarat dari relatif prima adalah GCD(e, m) = 1.
   5. Hitung *private key* atau *secret key*, misalkan d, dengan menggunakan sifat kekongruenan $ed \equiv 1 \pmod{m}$ atau bisa juga dengan $d \equiv e^{-1} \bmod m$.

   Perlu diperhatikan $ed \equiv 1 \pmod{m}$ ekivalen dengan $ed = 1 + km$ dengan k merupakan faktor pengali m, sehingga d dapat dihitung dengan:

   $$d = (1 + km)/e$$

2. *Enkripsi*
   Langkah–langkah dalam melakukan enkripsi:
   1. Ubahlah pesan menjadi blok-blok *plaintext* $p_1, p_2, p_3, \ldots, p_n$. Perhatikan nilai p harus terletak dalam {0, 1, 2, 3, … , n - 1} sebagai syarat bahwa perhitungan berada di dalam *range* himpunan yang telah disebutkan di atas.
   2. Hitung blok *ciphertext*, misalkan c, dengan p sebagai blok *plaintext*, melalui persamaan sebagai berikut:

      $$c_i = p_i^e \bmod n$$

3. *Dekripsi*
   Persamaan yang digunakan dalam proses dekripsi adalah

   $$p_i = c_i^d \bmod n$$

   dengan d adalah *private key*.

Berikut ini adalah representasi variabel–variabel RSA dalam bentuk tabel:

No.	Variabel	Rahasia
1.	a dan b yang merupakan bilangan prima sembarang	Ya
2.	$n = ab$	Tidak
3.	$m = (a - 1)(b - 1)$	Ya
4.	e (*encryption key* atau *public key*)	Tidak
5.	d (*decryption key* atau *private key*)	Ya
6.	p (*plaintext*)	Ya
7.	c (*ciphertext*)	Tidak

Tabel 1. Tabel Representasi Variabel RSA

Keamanan algoritme RSA ditentukan oleh pemfaktoran bilangan nonprima menjadi faktor-faktor primanya. Jika n dapat difaktorkan menjadi a dan b, maka m dapat ditentukan. Algoritme RSA disarankan memberikan nilai a dan b yang panjang karakternya lebih dari seratus digit sehingga hasil kali $n = ab$ merupakan bilangan dengan lebih dari dua ratus digit. Rivest mengatakan untuk mencari faktor bilangan dua ratus

digit dibutuhkan waktu komputasi selama 4 miliar tahun.

D. *Chinese Remainder Theorem*

Chinese Remainder Theorem pertama kali lahir berdasarkan *statement* yang dikemukakan oleh Sunzi di buku ketiganya: "Tentukan sebuah bilangan bulat yang bila dibagi dengan 5 menyisakan 3, bila dibagi 7 menyisakan 5, dan bila dibagi 11 menyisakan 7", persoalan ini merupakan contoh permasalahan yang kemudian dikenal secara luas sebagai *Chinese Remainder Theorem*.

Misalkan $m_1, m_2, m_3, \ldots, m_n \in \mathbf{Z}^+$ sedemikian sehingga GCD(m_i, m_j) = 1 untuk $i \neq j$. Maka sistem kongruen lanjar

$$x \equiv a_k \pmod{m_k}$$

mempunyai sebuah solusi unik dalam modulo $m = m_1 \cdot m_2 \cdot \ldots \cdot m_n$.

Penyelesaian *statement* dari Sunzi dengen menggunakan *Chinese Remainder Theorem* adalah sebagai berikut:

$x \equiv 3 \pmod 5 \rightarrow x = 3 + 5k_1$ (i)

Masukkan (i) ke dalam kongruen kedua menjadi:

$3 + 5k_1 \equiv 5 \pmod 7 \rightarrow k_1 \equiv 6 \pmod 7$ atau $k_1 = 6 + 7k_2$ (ii)

Masukkan (ii) ke dalam (i):

$x = 3 + 5k_1 = 3 + 5(6 + 7k_2) = 33 + 35k_2$ (iii)

Masukkan (iii) ke dalam kongruen ketiga menjadi:

$33 + 35k_2 \equiv 7 \pmod{11} \rightarrow k_2 \equiv 9 \pmod{11}$ atau $k_2 = 9 + 11k_3$

Masukkan k_2 ke dalam (iii) menghasilkan:

$x = 33 + 35(9 + 11k_3) = 348 + 385k_3$ atau $x \equiv 348 \pmod{385}$

$x \equiv 348 \pmod{385}$ merupakan solusinya, perhatikan 348 adalah bilangan bulat positif terkecil yang merupakan solusi dari sistem kekongruenan di atas. Perhatikan 348 mod 5 = 3, 348 mod 7 = 5, dan 348 mod 11 = 7; dan $385 = 5 \cdot 7 \cdot 11$. Perhatikan x merupakan himpunan tak hingga solusi $\mathbf{Z}^+$ yang apabila dibagi dengan 385 bersisa sama dengan 348.

Chinese Remainder Theorem mempunyai banyak kegunaan dalam pengaplikasiannya di dunia nyata, salah satu di antaranya adalah mempercepat dan mengoptimisasi algoritme.

III. IMPLEMENTASI RSA–CRT

Chinese Remainder Theorem dalam implementasinya mendekripsi RSA jauh lebih cepat daripada RSA standar yang masih menggunakan metode pemangkatan modular (*modular exponentation*). Dengan menggunakan *Chinese Remainder Theorem*, metode dalam pembangkitan kunci dan dekripsi berbeda dengan RSA standar, hal ini karena *secret exponent d* tidak dapat dibuat pendek, apabila $d < N^{0.292}$, dengan N menyatakan hasil kali antara a dan b (dengan a dan b merupakan bilangan prima sembarang), maka sistem RSA akan hancur secara total [6].

A. *RSA–CRT Key Generation*

Untuk membangkitkan *key* dalam algoritme RSA–CRT, berikut merupakan langkah–langkahnya:

1. Misalkan p dan q adalah dua bilangan prima yang sangat besar (dengan kedua bilangan memiliki nilai yang hampir sama) sedemikian sehingga GCD(p - 1, q - 1) = 2.
2. Hitunglah $N = pq$.
3. Tentukan dua buah bilangan bulat d_p dan d_q sedemikian sehingga GCD(d_p, p - 1) = 1, GCD(d_q, q - 1) = 1, dan $d_p \equiv d_q \pmod 2$.
4. Tentukan sebuah bilangan d sedemikian sehingga $d \equiv d_p \pmod{p - 1}$ dan $d \equiv d_q \pmod{q - 1}$.
5. Hitunglah $e = d^{-1} \pmod{\varphi(N)}$

Kunci publiknya adalah <N, e> dan kunci rahasianya adalah <p, q, d_p, d_q>. Dapat dilihat bahwa GCD(d_p, p - 1) = 1 dan GCD(d, p - 1) = 1, maka GCD(d, p - 1) = 1 = GCD(d, q - 1). Langkah tersebut dilanjutkan dengan menentukan GCD dari d dengan $\varphi(N)$, didapat bahwa GCD(d, $\varphi(N)$) = 1. Perhitungan *Chinese Remainder Theorem* belum bisa dilakukan saat ini karena hasil dari p - 1 dan q - 1 merupakan bilangan genap (seharusnya di dalam aturan RSA, p - 1 dan q - 1 harus prima). Perhatikan GCD((p - 1)/2, (q - 1)/2) pasti bernilai 1. d, d_p, dan d_q merupakan bilangan ganjil karena GCD(d, p - 1) = 1, GCD(d_p , p - 1) = 1 dan GCD(d_q , q - 1) = 1. Karena d, d_p,dan d_q bernilai ganjil, maka tentu d - 1, d_p - 1, dan d_q - 1 bernilai genap.

Didapat empat buah persamaan yang berguna untuk menemukan solusi *d:*

$d \equiv d_p \pmod{p - 1}$ (i)
$d \equiv d_q \pmod{q - 1}$ (ii)
$(d - 1) \equiv (d_p - 1) \pmod{p - 1}$ (iii)
$(d - 1) \equiv (d_q - 1) \pmod{q - 1}$ (iv)

Dengan menggunakan *cancellation law* dan menarik faktor 2 keluar, kita akan mendapatkan

$x = d' \equiv ((d - 1)/2) \equiv ((d_p - 1)/2) \bmod ((p - 1)/2)$ (v)
$x = d' \equiv ((d - 1)/2) \equiv ((d_q - 1)/2) \bmod ((q - 1)/2)$ (vi)

Kita baru dapat menerapkan *Chinese Remainder Theorem* untuk persamaan (v) dan (vi). Dengan menggunakan *Chinese Remainder Theorem*, kita menemukan nilai $d = (2d') + 1$ dengan d' bernilai sama dengan x.

B. *RSA–CRT Decryption*

Perlu diperhatikan enkripsi RSA–CRT sama dengan enkripsi RSA standar, maka dari itu tidak perlu diimplementasikan karena penulis sudah mengimplementasikannya di bab sebelumnya. Sekarang kita beralih ke dekripsi RSA–CRT.

Misalkan M adalah sebuah *plaintext* dan C adalah sebuah *ciphertext*. Jika C tidak habis dibagi oleh p dan $d_p \equiv d \pmod{p - 1}$, maka $C^{dp} \equiv C^d \pmod p$.

Dengan itu, dapat dibuat dua buah persamaan secara *general:*

$$M_p = C^{dp} \pmod p = C^d \pmod p$$
$$M_q = C^{dq} \pmod q = C^d \pmod q$$

Dengan menggunakan *Chinese Remainder Theorem*, dapat ditentukan nilai dari M sebagai berikut:

$$M = M_p \pmod p = C^d \pmod p$$
$$M = M_q = C^{dq} \pmod q = C^d \pmod q$$

Contoh: Misalkan $p = 23$, $q = 29$, tentukan nilai dari M_p dan M_q!

Kita akan menentukan nilai M_p dan M_q dari informasi–informasi yang telah diketahui. Untuk $p = 23$ dan $q = 29$, GCD(p - 1, q - 1) = 2.

Untuk mencari nilai N, kalikan p dengan q didapat $N = pq = 667$. Langkah selanjutnya adalah mencari $\varphi(N) = (p - 1)(q - 1) = 616$. Misalkan $d_p = 7$ dan $d_q = 5$ sehingga GCD(d_p, p - 1) = 1 = GCD(d_q, q - 1).

Kita akan mencari nilai d sedemikian sehingga:

$$d \equiv 7 \pmod{22} \quad \text{(i)}$$
$$d \equiv 5 \pmod{28} \quad \text{(ii)}$$

Kita tidak dapat menerapkan *Chinese Remainder Theorem* secara langsung karena GCD(22, 28) ≠ 1. Kita harus menggunakan teknik untuk mengubah bentuk dari sistem kekongruenannya tanpa mengubah makna dari kekongruenan tersebut, teknik yang digunakan adalah *cancellation law*.

Dengan mengurangi *left-handed side* dan *right-handed side* sebesar 1 dari sistem kekongruenan, didapat:

$$(d - 1) \equiv (7 - 1) \pmod{22} \quad \text{(iii)}$$
$$(d - 1) \equiv (5 - 1) \pmod{28} \quad \text{(iv)}$$

Dengan menggunakan *cancellation law*:

$$((d - 1)/2) \equiv ((7 - 1)/2) \pmod{(22/2)} \quad \text{(v)}$$
$$\rightarrow x = d' = ((d - 1)/2) \equiv 3 \pmod{11} \quad \text{(vi)}$$

$$((d - 1)/2) \equiv ((5 - 1)/2) \pmod{(28/2)} \quad \text{(vii)}$$
$$\rightarrow x = d' = ((d - 1)/2) \equiv 2 \pmod{14} \quad \text{(viii)}$$

Pecahkan dengan menggunakan *Chinese Remainder Theorem:*

$$M = 11 \cdot 14 = 154,\ M_1 = 154/11 = 14,\ M_2 = 154/14 = 11$$
$$14 \cdot N_1 \equiv 1 \pmod{11} \rightarrow N_1 = 4$$
$$11 \cdot N_2 \equiv 1 \pmod{14} \rightarrow N_2 = 9$$

Tentukan nilai d' dan d, dengan memasukkan informasi–informasi yang telah diketahui:

$$d' = x = ((d_p - 1)/2) \cdot M_1 \cdot N_1 + ((d_q - 1)/2) \cdot M_2 \cdot N_2$$
$$= 3 \cdot 14 \cdot 4 + 2 \cdot 11 \cdot 9 = 366$$

$$d = 2d' + 1 = 2 \cdot 366 + 1 = 733$$

Setelah itu, cari nilai e sedemikian sehingga:

$$ed \equiv 1 \pmod{\varphi(N)} \rightarrow 733e \equiv 1 \pmod{616} \rightarrow e = 437$$

Misalkan sebuah *plaintext* $M = 7$, maka:

$$C = M^e \pmod N = 7^{437} \pmod{667} = 517$$

Tentukan nilai M_p dan M_q:

$$M_p = C^d \pmod p = 517^{733} \pmod{23} = 7$$
$$M_q = C^d \pmod q = 517^{733} \pmod{29} = 7$$

Dengan menggunakan *Chinese Remainder Theorem* untuk pembuktian:

$$M = 23 \cdot 29 = 667,\ M_1 = 667/23 = 29,\ M_2 = 667/29 = 23$$
$$29 \cdot N_1 \equiv 1 \pmod{23} \rightarrow N_1 = 4$$
$$23 \cdot N_2 \equiv 1 \pmod{29} \rightarrow N_2 = 24$$

Variabel N_1 dan N_2 digunakan untuk perhitungan x yang pada akhirnya membuktikan kebenaran untuk solusi yang telah ditemukan.

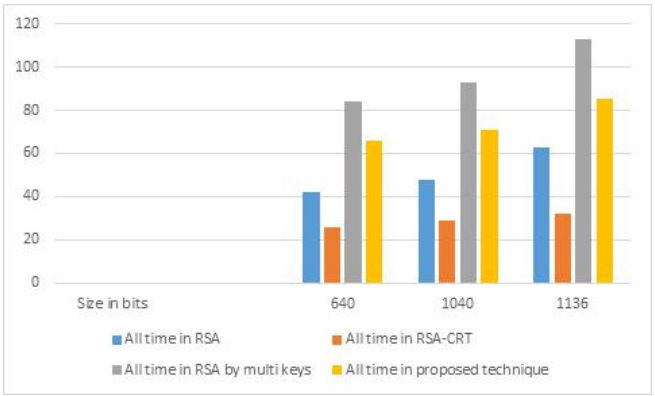
Tentukan apakah x bernilai sama dengan M, jika sama, perhitungan berhasil dilakukan dan merupakan solusinya:

$$x = (M_p \cdot M_1 \cdot N_1 + M_q \cdot M_2 \cdot N_2) \pmod M$$
$$= (7 \cdot 29 \cdot 4 + 7 \cdot 23 \cdot 24) \pmod{667}$$
$$= 4676 \pmod{667} = 7$$

Didapatkan $x = M = M_p = M_q = 7$, solusi sudah ditemukan dan terbukti kebenarannya.

IV. Hasil Pengamatan

Menurut pengamatan yang dilakukan, RSA–CRT merupakan algoritme yang cukup mangkus dibanding RSA standar. Hal ini tentu dapat dilihat dari grafik–grafik dan tabel yang sudah penulis sematkan di bawah.



Gambar 4. Grafik Perbandingan RSA dengan RSA–CRT [2]

Dapat dilihat algoritme RSA–CRT lebih unggul dibanding metode–metode yang lainnya, dari grafik diperoleh kecepatan eksekusi algoritme RSA–CRT < 40 ms walaupun *bits*-nya semakin besar, sedangkan RSA standar memiliki peningkatan yang lumayan ketika dinaikkan dari 1040 *bits* menjadi 1136 *bits*. RSA–CRT beroperasi sampai empat kali lebih cepat di kasus terbaiknya (*best case*) karena untuk ukuran hasil kali N dan d bisa di-*reduced.*

Size in bits	All time encryption and decryption of RSA in MS	All time encryption and decryption of RSA-CRT in MS
640	42	26
1040	48	29
1136	63	32

Tabel 2. Tabel Perbandingan RSA dengan RSA–CRT [2]

Untuk lebih jelasnya, tabel di atas merupakan representasi data yang berada di grafik perbandingan RSA dengan RSA–CRT. Tapi perlu diperhatikan algoritme enkripsi RSA dan

RSA–CRT tidak memiliki perbedaan sehingga difokuskan ke bagian dekripsinya saja. Dengan menggunakan RSA–CRT, komputasi dibagi menjadi dua bagian dan ukuran enkripsinya setengah dari ukuran RSA standar, hal ini membuat waktu komputasi relatif lebih sedikit dibandingkan dengan RSA standar.

Untuk kasus lain, misalkan *key* berukuran 1024 *bits.* Nilai dari *private key* dan *public key* yang didapatkan merupakan bilangan acak. Sedangkan nilai *e* dipilih 3 karena memenuhi syarat GCD(e, $\varphi(N)$) = 1 yang menandakan bahwa *e* relatif prima terhadap $\varphi(N)$. Dengan menggunakan metode *cancellation law* yang telah dijelaskan di bab sebelumnya, akan didapatkan nilai d_p dan d_q. Nilai dari d_p dan d_q sepanjang 155 desimal. Dengan ini, kita akan melihat performa RSA standar dibandingkan dengan RSA–CRT.

File	Decryption of RSA (nano second)	Decryption of RSA-CRT (nano second)
1	103,534,963	29,710,330
2	102,998,756	29,400,697
3	103,437,564	29,477,594
4	103,300,180	29,538,084
5	103,515,483	29,730,836

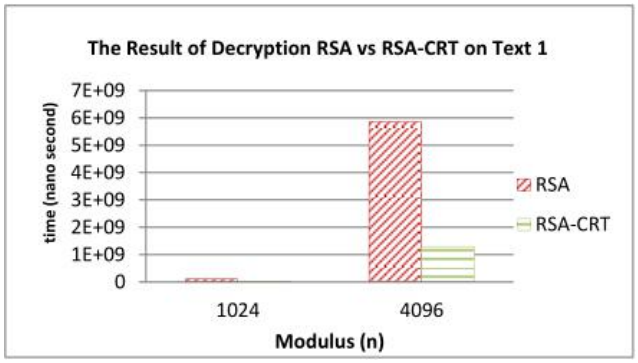
Tabel 3. Tabel Perbandingan RSA dengan RSA–CRT dalam 1024 *bits* [4]

Di setiap kasus, kurang lebih RSA–CRT tiga kali lebih cepat daripada RSA standar. RSA–CRT menempuh waktu selama ±29.71 ms sedangkan RSA standar menempuh waktu selama ±103.53 ms.

File	Decryption of RSA (nano second)	Decryption of RSA-CRT (nano second)
1	5,853,890,548	1,304,281,228
2	5,842,122,477	1,149,450,813
3	5,906,614,132	1,171,997,568
4	5,944,532,789	1,311,686,767
5	5,945,090,537	1,168,936,106

Tabel 4. Tabel Perbandingan RSA dengan RSA–CRT dalam 4096 *bits* [4]

Untuk 4096 *bits*, dekripsi dengan menggunakan RSA–CRT memakan waktu ± 1s sedangkan dekripsi menggunakan RSA standar memakan waktu ± 6s, hal ini tentu sangat jauh berbeda, RSA–CRT lebih cepat dan teroptimisasi enam kali lebih cepat daripada RSA standar.



Gambar 5. Grafik Perbandingan RSA dengan RSA–CRT [4]

Berdasarkan gambar 5, RSA–CRT sangat efektif untuk mengoptimisasi proses dekripsi pesan walaupun nilai yang digunakan dalam modulus *N* berbeda dengan RSA standar. Pada 1024 *bits*, waktu yang dibutuhkan oleh RSA standar dan RSA–CRT tidak jauh berbeda, yaitu mendekati 0. Sedangkan jika *bits* dinaikkan menjadi 4096 *bits* di mana merupakan empat kalinya yang sebelumnya, waktu eksekusi (dalam ns) jauh berbeda, RSA–CRT enam kali lebih cepat dibandingkan RSA biasa dalam hal mendekripsi pesan dalam kasus ini.

Key length	En&De Time	RSA Std	RSA-CRT
128	En-Time	0 ms	0 ms
	De-time	0 ms	0 ms
256	En-Time	0 ms	3.75 ms
	De-time	31 ms	3.75 ms
512	En-Time	0 ms	3 ms
	De-time	18.8 ms	6.2 ms
1024	En-Time	34.2 ms	37.8 ms
	De-time	138.2 ms	37.4 ms
2048	En-Time	225 ms	281.4 ms
	De-time	672 ms	227.8 ms

Tabel 5. Tabel Perbandingan RSA dengan RSA–CRT dengan berbagai *key length* [5]

Terdapat perbandingan waktu untuk melakukan enkripsi dan dekripsi pada RSA standar dan RSA–CRT, namun karena kita lebih memfokuskan untuk dekripsi, maka dapat dilihat di baris kedua setiap kolom *key length*, di sini dicoba beberapa variasi *key length*, yaitu 128 *bits*, 256 *bits*, 512 *bits*, 1024 *bits*, dan 2048 *bits*. Untuk setiap *key length* mulai dari 256 *bits* sampai 2048 *bits*, waktu yang dibutuhkan RSA–CRT untuk mendekripsi selalu lebih sedikit relatif terhadap RSA standar. Untuk 256 *bits*, RSA–CRT mendekripsi dalam waktu 3.75 ms, sedangkan RSA standar mendekripsi dalam waktu 31 ms. Untuk 512 *bits*, RSA–CRT mendekripsi dalam waktu 6.2 ms, sedangkan RSA standar mendekripsi dalam waktu 18.8 ms.Untuk 1024 *bits*, RSA–CRT mendekripsi dalam waktu 37.4 ms, sedangkan RSA standar mendekripsi dalam waktu 138.2 ms. Dan untuk 2048 *bits*, RSA–CRT mendekripsi dalam waktu 227.8 ms, sedangkan RSA standar mendekripsi dalam waktu 672 ms. Dapat dilihat untuk setiap *key length* kecuali 128 *bits*, RSA–CRT lebih cepat minimal tiga kali lebih sedikit waktu untuk mendekripsi dibandingkan dengan RSA standar. Hal ini tentu sangat berguna dalam mengoptimisasi data dan membuatnya lebih efektif lagi.

V. Analisis Dan Pembahasan

Dari berbagai teori dasar dan implementasi yang telah dilakukan, RSA standar menggunakan *modular exponentation* yang dilakukan dengan bertahap perkalian modular dan pastinya memakan waktu komputasi yang lama. Maka dari itu, digunakan *Chinese Remainder Theorem* yang dapat mereduksi waktu komputasi untuk mendekripsi pesan atau data yang dikirimkan oleh pengirim (*sender*). Berikut ini analisis mengapa RSA–CRT jauh lebih ter-*optimized* daripada RSA standar.

Perhatikan untuk persamaan berikut:

$$p_i = c_i^d \bmod n$$

Untuk mendapatkan p_i yang berupa *plaintext* pada RSA standar, dibutuhkan *modular exponentation* dari blok *ciphertext* dan ditentukan sisanya apabila dibagi dengan n, ingat n adalah hasil perkalian antara a dan b, di mana a dan b merupakan bilangan prima acak dan direkomendasikan sebesar mungkin agar proses mendekripsi semakin susah dilakukan oleh kriptanalis selain *receiver* yang tidak memiliki hak untuk melihat informasi atau pesan yang bersifat rahasia tersebut (hanya *sender* dan *receiver* yang seharusnya tahu).

Sedangkan pada *Chinese Remainder Theorem* yang diaplikasikan pada RSA menggunakan bantuan Fermat's Little Theorem, berikut ini merupakan persamaan lanjutan yang menentukan kecepatan optimisasi:

$$\begin{aligned} M' \pmod p &= M_q \pmod p + (M_p - M_q) \pmod p \\ M' \pmod q &= M_q \quad \text{(persamaan } \textit{trivial}\text{)} \\ M' \pmod p &= M_q \pmod p + (M_p - M_q) \pmod p \\ &= M_p \pmod p \\ &= M_p \end{aligned}$$

dengan M adalah *plaintext*, p dan q adalah bilangan prima acak.

Dapat disimpulkan bahwa nilai dari M' sama dengan M. Hal ini yang membuat perhitungan nilai M lebih cepat daripada menggunakan metode RSA standar karena tidak perlu dilakukan reduksi modulo N secara sekuensial. RSA–CRT pada $n/2$ *bit hardware* empat kali lebih cepat daripada non–CRT (RSA standar) untuk n *bit hardware* sehingga ini mengurangi panjang *bit* yang digunakan untuk melakukan dekripsi menjadi setengah kali dari RSA standar. Algoritme RSA–CRT melakukan operasi secara paralel yang menggunakan $n/2$ *bit* sebagai faktor pengali modular.

VI. Simpulan Dan Saran

Dari hasil uraian di atas, dapat diambil simpulan:

1. Algoritme RSA merupakan algoritme sandi kunci asimetris (kunci publik) yang menggunakan *modular exponentation*. Proses *modular exponentation* disimpulkan memakan waktu relatif lebih lama.
2. *Chinese Remainder Theorem* memiliki banyak aplikasi di dunia *computer science*, salah satu contohnya adalah untuk modifikasi algoritme RSA.
3. Algoritme RSA standar dengan RSA–CRT memiliki kesamaan di proses enkripsi (*encryption*) dan berbeda di proses pembangkitan kunci (*key generation*) dan proses dekripsi (*decryption*).
4. Algoritme RSA–CRT lebih mangkus dan unggul dalam proses dekripsi dibandingkan dengan algoritme RSA standar.
5. Waktu komputasi dapat direduksi secara signifikan dengan mengimplementasikan *Chinese Remainder Theorem* pada RSA.

Saran dari penulis adalah semoga ke depannya ditemukan algoritme–algoritme yang lebih mangkus dan efisien serta lebih menjamin keamanan privasi data bagi para penggunanya. Di zaman sekarang, mulai diterapkan *Advanced Encryption Standard* (AES) dan *Elliptic Curve Cryptography* (ECC), namun penulis yakin bahwa teknologi ada untuk selalu berkembang menjadi lebih baik lagi dan belajar dari kelemahan–kelemahan yang ada.

VII. Ucapan Terima Kasih Dan Penutup

Puji syukur kepada Tuhan karena atas berkat-Nya, *paper* berjudul "Penerapan *Chinese Remainder Theorem* untuk Mendekripsi *Ciphertext* pada RSA *Cryptosystem*" dapat selesai tepat waktu. Saya mengucapkan terima kasih kepada orang tua saya atas dukungannya selama ini. Saya juga mengucapkan terima kasih kepada Ibu Fariska Zakhralativa Ruskanda, Bapak Rinaldi Munir, dan Bapak Judhi Santoso sebagai dosen pengajar IF2120 Matematika Diskrit atas bimbingannya selama satu semester ini sehingga *paper* ini dapat selesai.

Referensi

[1] Munir, Rinaldi. 2006. *Diktat Kuliah IF2120 Matematika Diskrit* (*Edisi Keempat*). Bandung: Institut Teknologi Bandung.

[2] Abdeldaym, Rasha Samir, Hatem Mohamed Abd Elkader, dan Reda Hussein. 2019. *Modified RSA Algorithm Using Two Public Key and Chinese Remainder Theorem.* I.J. of Electronics and Information Engineering, Vol. 10, No. 1. DOI: 10.6636/IJEIE.201903 10(1).06.

[3] Rai, Aarushi dan Shitanshu Jain. 2017. *Encryption and Decryption through RSA Cryptosystem using Two Public Keys and Chinese Remainder Theorem*. International Journal of Computer Applications (0975 – 8887).

[4] Wulansari, Desi, Much. Aziz Muslim, dan Endang Sugiharti. 2016. *Implementation of RSA Algorithm with Chinese Remainder Theorem for Modulus N 1024 Bit and 4096 Bit*. IJCSS.

[5] Noureldien dan Wafa M. Mustafa. 2014. *Performance Evaluation of RSA-CRT Rebalanced Variants.* International Journal of Advance Foundation and Research in Computer (IJAFRC). ISSN 2348 - 4853.

[6] Shinde, G. N., dan H.S. Fadewar. 2008. *Faster RSA Algorithm for Decryption Using Chinese Remainder Theorem.* ICCES, vol.5, no.4, pp.255-261.

[7] Grobler, T. L., dan W. T. Penzhorn. 2004. *Fast Decryption Methods for the RSA Cryptosystem.* IEEE Africon. 7th Africon Conference in Africa.

[8] Mantri, Ankur dkk. 2016. *Analytical Comparison of RSA with Chinese Remainder Theorem*. Journal of Independent Studies and Research – Computing Volume 14 Issue 1.

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 1 Desember 2019

Vincentius Lienardo, 13518081