Aplikasi Bilangan Prima dalam Pemanfaatan Algoritma RSA untuk Keamanan Dokumen Negara

Dharma Kurnia Septialoka - 13514028[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13514028@std.stei.itb.ac.id

***Abstrak*—Dokumen negara merupakan hal paling krusial yang harus dijaga kerahasiannya. Namun seringkali kita dengar atau temukan bahwa kerahasiannya bocor, entah karena diakses oleh pihak yang tidak berwenang, disebarkan, bahkan sampai dijual ke negara lain. Imbasnya, tingkat keamanan suatu negara dipertanyakan. Negara menjadi tidak aman lagi karena ada pihak luar yang telah membaca dokumen rahasia negara tersebut. Biasanya, dokumen berlabel rahasia ini bocor karena teks dokumen tersebut berbentuk *plaintext* meskipun dokumen itu telah diproteksi dengan kata sandi.. Maka, alangkah baiknya bila semua dokumen rahasia tersebut dienkripsi menjadi bentuk *ciphertext,* dan untuk mendekripsinya diperlukan kunci rahasia, dimana kunci tersebut hanya dimiliki oleh pihak yang berhak. Oleh karena itu, diperlukan suatu algoritma yang tepat untuk mengenkrispi dokumen tersebut secara aman yang sulit ditembus oleh teknologi manapun. Makalah ini membahas penerapan algoritma RSA untuk meningkatkan keamanan dokumen negara yang bersifat rahasia. Algoritma RSA ini akan terdiri dari 3 langkah utama, yakni pembangkitan kunci, enkripsi, dan dekripsi. Pada akhirnya, diharapkan dari makalah ini, pembaca dapat mengenal lebih jauh cara kerja algoritma RSA, implementasinya, serta penerapannya yang berguna bagi kehidupan kita.**

***Keywords*—algoritma RSA, enkripsi, dekripsi, dokumen**

I. PENDAHULUAN

Dokumen, berasal dari kata Bahasa Inggris yakni *document,* yang berarti sesuatu yang tertulis atau tercetak dan segala benda yang mempunyai keterangan-keterangan dipilih untuk dikumpulkan, disusun, disediakan, atau untuk disebarkan[1]. Dokumen ada yang bersifat publik, rahasia, dan sangat rahasia. Dokumen negara adalah dokumen yang sangat penting. Dokumen tersebut dapat berisi keterangan tentang suatu kejadian penting yang telah terjadi, dokumen kepemerintahan seperti perundang-undangan, dokumen tentang sejarah peradaban seperti piagam proklamasi, dokumen tentang perkembangan perekonomian negara tersebut, dokumen yang meliputi surat-surat resmi dan surat-surat negara, seperti surat perjanjian, konsesi, hibah, dan lain-lain, sampai dokumen yang bersifat sangat rahasia seperti rencana-rencana, tindakan-tindakan, dan strategi pemerintah yang akan diambil untuk melanjutkan pemerintahan, yang mana jika diketahui pihak lain atau disebarluaskan ke publik dapat memicu kontroversi, serta dokumen-dokumen rahasia lainnya yang berisi informasi penting tentang pemerintahan atau negara tersebut yang jika ada pihak atau negara lain yang membacanya dapat menjadi ancaman untuk keberlangsungan maupun stabilitas dan keamanan negara tersebut.

Kebocoran dokumen, biasanya diakibatkan karena rendahnya keamanan dalam memproteksi isi dokumen tersebut. Dokumen tersebut biasanya hanya berupa *plaintext.* Jadi, siapapun dapat mencuri dokumen tersebut untuk membaca, mengkopi, bahkan menjualnya. Apalagi jika dilakukan oleh petugas pemerintahan yang bekerja disana meskipun sebenarnya ia bukan merupakan pihak yang berhak untuk membaca isi dokumen tersebut, ia dapat dengan mudah mengaksesnya dikarenakan bekerja di tempat tersebut.

Kebocoran dokumen juga seringkali bukan hanya saat dokumen itu disimpan atau saat dikirimkan, melainkan juga saat diterima oleh pihak yang berwenang untuk menerima dokumen rahasia tersebut. Maka itu, alangkah baiknya jika dokumen tersebut selalu dienkripsi menjadi *ciphertext* dan hanya didekripsi oleh pihak yang berwenang dan memiliki akses untuk membaca dokumen rahasia tersebut. Dengan begini, kita pun dapat mengetahui pihak mana yang membocorkan atau perlu dimintai keterangan karena hanya dia-lah yang memiliki kunci rahasia untuk mendekripsi dokumen tersebut.

Jika kita melihat situs *Wikileaks*, *Wikileaks* merupakan situs dimana dokumen-dokumen rahasia dibocorkan dan disebarkan pada publik. *Wikileaks* berposisi di Swedia. Hukum Swedia dengan tegas melarang wartawan membocorkan identitas sumber, sedangkan *wikileaks* pada hal ini berposisi sebagai wartawan. Dengan melihat isi dari situs *Wikileaks* tersebut, kita tahu bahwa isi dokumen negara sangat penting dan patut dijaga kerahasiannya.

Dalam makalah ini, penulis akan membahas bagaimana algoritma RSA dimanfaatkan untuk mengubah isi dokumen rahasia yang awalnya berbentuk *plaintext* menjadi *ciphertext* dengan pembangkitan kunci RSA.

II. DASAR DARI TEORI BILANGAN DAN KRIPTOGRAFI

2.1 Teori Bilangan

Teori bilangan adalah cabang matematika yang mengkaji secara khusus bilangan bulat dan sifat-sifatnya. Bilangan bulat adalah objek diskrit. Bilangan bulat adalah bilangan yang tidak mempunyai pecahan desimal, misalnya 8, 21, 87656, -30, 0, dan sebagainya. Sifat pembagian pada bilangan bulat melahirkan konsep-konsep seperti bilangan prima dan aritmetika modulo. Satu algoritma penting yang berhubungan dengan sifat pembagian ini adalah algoritma *Euclidean*. Baik bilangan prima, aritmetika modulo, dan algoritma *Euclidean* memainkan peranan penting dalam bidang kriptografi yang akan kita bahas untuk makalah ini[2].

2.1.1 Sifat Pembagian pada Bilangan Bulat

Misalkan a dan b bilangan bulat, $a \neq 0$.

a habis membagi b (a *divides* b) jika terdapat bilangan bulat c sedemikian sehingga b = ac.

Notasi: a | b jika b = ac, $c \in Z$ dan $a \neq 0$.

Misalkan *m* dan *n* bilangan bulat, $n > 0$. Jika *m* dibagi dengan *n* maka terdapat bilangan bulat unik *q* (*quotient*) dan *r* (*remainder*), sedemikian sehingga

$$m = nq + r \qquad (1)$$

dengan $0 \leq r < n$.

Teorema ini disebut teorema *Euclidean*. Dari teorema ini, n disebut pembagi (*divisor*), m disebut yang dibagi (*dividend*), q disebut hasil bagi (*quotient*), dan r disebut sisa (*remainder*).

2.1.2 PBB

Dua buah bilangan bulat dapat memiliki faktor pembagi yang sama. Faktor pembagi bersama yang terpenting adalah faktor pembagi persama terbesar.

Misalkan a dan b bilangan bulat tidak nol.

Pembagi bersama terbesar (PBB – *greatest common divisor* atau gcd) dari a dan b adalah bilangan bulat terbesar d sedemikian hingga d | a dan d | b.

Dalam hal ini kita nyatakan bahwa PBB(a, b) = d.

2.2 Bilangan Prima

Bilangan prima adalah bilangan bulat positif yang lebih besar dari 1 yang hanya dapat dibagi oleh bilangan 1 dan bilangan itu sendiri.

Bilangan bulat positif p (p > 1) disebut bilangan prima jika pembaginya hanya 1 dan p. Karena bilangan prima harus lebih besar dari 1, maka barisan bilangan prima dimulai dari 2, yaitu 2, 3, 5, 7, 11, ... Seluruh bilangan prima adalah bilangan ganjil, kecuali 2 yang merupakan bilangan genap.

Bilangan selain prima disebut bilangan komposit. Metode yang mangkus yang terkenal untuk menguji keprimaan suatu bilangan bulat n adalah dengan menggunakan Teorema Fermat.

2.2.1 Teorema Fermat

Jika *p* adalah bilangan prima dan *a* adalah bilangan bulat yang tidak habis dibagi dengan *p*, yaitu PBB(*a*, *p*) = 1, maka

$$a^{p-1} \equiv 1 \pmod{p} \qquad (2)$$

Namun, Teorema Fermat mengandung kekurangan sebab terdapat bilangan komposit n sedemikian sehingga $2^{n-1} \equiv 1 \pmod{n}$. Bilangan itu dinamakan prima semu (*pseudoprimes)*.

Algoritma Fermat dapat ditulis sebagai berikut:

Masukan:

n: bilangan yang diuji apakah prima, n > 3

k: parameter yang menentukan berapa kali pengujian tes apakah prima

Keluaran:

Komposit jika n adalah komposit, kalau tidak mungkin prima.

Ulang sebanyak n kali:

Ambil acak antara kisaran [2, n-2]

Jika $a^{n-1} \neq 1 \bmod n$ kembalikan 'komposit'

Jika tidak pernah mengembalikan komposit, maka kembalikan 'mungkin prima'.

Dengan menggunakan modulo eksponensial, kompleksitas algoritma ini adalah O (k × log2n × log log n × log log log n), dengan k adalah jumlah k kali pengujian angka acak a, dan n adalah bilangan yang diuji apakah prima.

Sebagai tambahan, modulo eksponensial adalah pengembangan dari algoritma *Euclidean*. Modulo eksponensial adalah jenis eksponensial yang digunakan selama proses modulo. Modulo eksponensial digunakan terutama dalam bidang kriptografi-kunci publik[3].

Operasi modulo eksponensial c adalah sebagai berikut:

$$c \equiv b^e \pmod{m} \qquad (3)$$

dengan *pseudocode* sebagai berikut:

```
function modular_pow(base, exponent, modulus)
    if modulus = 1 then return 0
    c := 1
    for e_prime = 1 to exponent
        c := (c * base) mod modulus
    return c
```

Gambar 2.2.1 *Pseudocode* modulo eksponensial

Sumber:

https://en.wikipedia.org/wiki/Modular_exponentiation/

2.2.2 Relatif Prima

Dua buah bilangan bulat a dan b dikatakan relatif prima jika PBB(a, b) = 1.

Contoh:

(i) 20 dan 3 relatif prima sebab PBB(20, 3) = 1.

(ii) 7 dan 11 relatif prima karena PBB(7, 11) = 1.

(iii) 20 dan 5 tidak relatif prima sebab PBB(20, 5)=5.

Jika a dan b relatif prima, maka terdapat bilangan bulat m dan n sedemikian sehingga

$$ma + nb = 1 \qquad (4)$$

Contoh:

Bilangan 20 dan 3 adalah relatif prima karena PBB(20, 3) = 1, atau dapat ditulis

$$2 \cdot 20 + (-13) \cdot 3 = 1 \quad (m = 2, n = -13)$$

Tetapi 20 dan 5 tidak relatif prima karena PBB(20, 5) = 5 ≠ 1 sehingga 20 dan 5 tidak dapat dinyatakan dalam m . 20 + n . 5 = 1.

2.3 Aritmetika Modulo

Aritmetika modulo (*modular arithmetic*) memainkan peranan yang penting dalam komputasi bilangan bulat, khususnya pada aplikasi kriptografi. Operator yang digunakan pada aritmetika modulo adalah mod**.** Operator mod memberikan sisa pembagian. Misalnya 23 dibagi 5 memberikan hasil = 4 dan sisa = 3, sehingga kita tulis 23 mod 5 = 3. Definisi dari operator mod dinyatakan sebagai berikut:

Misalkan a dan m bilangan bulat (m > 0). Operasi a mod m (dibaca "a modulo m") memberikan sisa jika a dibagi dengan m.

Notasi: a mod m = r sedemikian sehingga a = mq + r, dengan $0 \le r < m$.

Bilangan m disebut modulus atau modulo, dan hasil aritmetika modulo m terletak di dalam himpunan {0, 1, 2, …, m – 1}.

2.3.1 Kongruen

Misalkan *a* dan *b* bilangan bulat dan *m* adalah bilangan > 0, maka $a \equiv b \pmod m$ jika *m* habis membagi *a* – *b*.
Kita katakan bahwa a dan b kongruen dalam modulo m.
Notasi '≡' dibaca 'kongruen'.

2.3.2 Balikan Modulo

Jika a dan m relatif prima dan m > 1, maka balikan (invers) dari a modulo m ada.
Balikan dari a modulo m adalah bilangan bulat x sedemikian sehingga

$$xa \equiv 1 \pmod m \qquad (5)$$

Dalam notasi lainnya, $a^{-1} \pmod m = x$

Untuk mencari balikan dari a modulo m, kita harus membuat kombinasi lanjar dari a dan m sama dengan 1. Koefisien a dari kombinasi lanjar tersebut merupakan balikan dari a modulo m.

2.3.3 Kekongruenan Lanjar

Kekongruenan lanjar adalah kongruen yang berbentuk:

$$ax \equiv b \pmod m \qquad (6)$$

dengan $m > 0$, *a* dan *b* sembarang bilangan bulat, dan *x* adalah peubah bilangan bulat. Bentuk kongruen lanjar berarti menentukan nilai-nilai x yang memenuhi kekongruenan tersebut. Metode yang sederhana untuk mencari nilai-nilai x tersebut adalah dengan mengubah $ax \equiv b \pmod m$ ditulis menjadi

$$ax = b + km$$

yang dapat disusun menjadi

$$x = \frac{b+km}{a} \qquad (7)$$

dengan k adalah sembarang bilangan bulat. Cobakan untuk $k = 0, 1, 2, \ldots$ dan $k = -1, -2, \ldots$ yang menghasilkan *x* sebagai bilangan bulat. Metode lain untuk mencari solusi kekongruenan lanjar adalah dengan menggunakan balikan modulo.

2.4 Algoritma *Euclidean*

Metode yang mangkus untuk menemukan PBB dari 2 buah bilangan bulat dikenal dengan nama algoritma *Euclidean.* Algoritma *Euclidean* didasarkan pada aplikasi teorema m = nq + r secara berturut-turut sampai kita menemukan sisa pembagian bernilai nol. Secara formal, algoritma *Euclidean* dirumuskan sebagai berikut:

Misalkan m dan n adalah bilangan bulat tak negatif dengan $m \ge n$. Misalkan $r_0 = m$ dan $r_1 = n$.

Lakukan secara berturut-turut pembagian untuk memperoleh

$r_0 = r_1q_1 + r_2 \qquad 0 \le r_2 \le r_1,$
$r_1 = r_2q_2 + r_3 \qquad 0 \le r_3 \le r_2,$
.
.
.
$r_{n-2} = r_{n-1}\,q_{n-1} + r_n \qquad 0 \le r_n \le r_{n-1},$
$r_{n-1} = r_nq_n + 0$

Maka,

$PBB(m, n) = PBB(r_0, r_1) = PBB(r_1, r_2) = \ldots =$
$PBB(r_{n-2}, r_{n-1}) = PBB(r_{n-1}, r_n) = PBB(r_n, 0) = r_n.$

Jadi, PBB dari m dan n adalah sisa terakhir yang tidak nol dari runtunan pembagian tersebut.

Diberikan dua buah bilangan bulat tak-negatif m dan n (m ≥ n). Algoritma *Euclidean* berikut mencari pembagi bersama terbesar dari m dan n.

Algoritma *Euclidean*

1. Jika n = 0 maka
   m adalah PBB(m, n);
   stop.
   tetapi jika n ≠ 0, lanjutkan ke langkah 2.
2. Bagilah m dengan n dan misalkan r adalah sisanya.
3. Ganti nilai m dengan nilai n dan nilai n dengan nilai r, lalu ulang kembali ke langkah 1.

Pseudo-code algoritma *Euclidean* adalah sebagai berikut:

```
procedure Euclidean(input m, n : integer,
                    output PBB : integer)
{ Mencari PBB(m, n) dengan syarat m dan n
    bilangan tak-negatif dan m ≥ n
  Masukan: m dan n, m ≥ n dan m, n ≥ 0
  Keluaran: PBB(m, n)
}
Kamus
  r : integer

Algoritma:
  while n ≠ 0 do
    r ← m mod n
    m ← n
    n ← r
  endwhile
  { n = 0, maka PBB(m,n) = m }
  PBB ← m
```

Tabel 2.4 Algoritma *Euclidean*
Sumber: Rinaldi M/ IF2091 Struktur Diskrit

2.5 Kriptografi

Aritmetika modulo dan bilangan prima mempunyai banyak aplikasi dalam kehidupan kita. Salah satu aplikasinya adalah dalam bidang kriptografi.

Kriptografi adalah ilmu sekaligus seni untuk menjaga kerahasiaan pesan, data, atau informasi dengan cara menyamarkannya menjadi bentuk yang tidak memiliki makna, dan hanya dapat dikembalikan ke bentuk semula oleh pihak yang memiliki akses. Kriptografi digunakan untuk menyembunyikan informasi rahasia dari pihak yang tidak berhak membacanya.

2.5.1 Istilah pada Kriptografi

Plaintext atau pesan adalah data atau informasi yang dapat dibaca dan dimengerti maknanya oleh pihak manapun yang membacanya. Pesan dapat berupa teks, gambar, audio, dan video. Pesan biasanya dikirim atau disimpan dalam media penyimpanan. Jika pesan ini rahasia, biasanya hanya ditambahkan kata sandi atau *password* untuk membuka pesan tersebut.

Ciphertext adalah pesan yang telah disandikan sehingga pesan tersebut tidak memiliki makna lagi, sehingga pihak manapun yang membacanya dengan mata telanjang tidak akan mengerti makna dari pesan tersebut. *Ciphertext* ini harus dapat diubah kembali ke *plaintext* oleh pihak yang memiliki hak akses terhadap pesan tersebut.

Contoh:

Plainteks:

culik anak itu jam 11 siang

Cipherteks:

t^$gfUi89rewoFpfdWqL:p[uTcxZ

Enkripsi adalah proses menyandikan *plaintext* menjadi *ciphertext.*

Dekripsi adalah proses mengembalikan *ciphertext* menjadi *plaintext*-nya kembali.

2.5.2 Notasi Matematis

Jika *ciphertext* dilambangkan dengan C dan *plaintext* dilambangkan dengan P, maka fungsi enkripsi E memetakan P ke C.

$$E(P) = C \qquad (8)$$

Pada proses kebalikannya, fungsi dekripsi D memetakan C ke P.

$$D(C) = P \qquad (9)$$

Karena proses enkripsi kemudian dekripsi mengembalikan pesan ke pesan awal, maka kesamaan berikut harus benar.

$$D(E(P)) = P \qquad (10)$$

2.5.3 Kriptografi Modern

Algoritma kriptografi adalah fungsi matematika yang digunakan untuk enkripsi dan dekripsi. Kekuatan suatu algoritma kriptografi diukur dari banyaknya kerja yang dibutuhkan untuk memecahkan data *ciphertext* menjadi *plaintext*-nya. Kerja ini dapat kita ekuivalenkan dengan waktu. Semakin banyak usaha yang kita perlukan, maka semakin lama juga waktu yang dibutuhkan, yang mana semakin kuat algoritma kriptografinya, yang berarti semakin aman untuk menyandikan pesan.

Jika kekuatan kriptografi ditentukan dengan menjaga kerahasiaan algoritmanya, maka algoritma kriptografinya dinamakan algoritma *restricted.* Misalkan di dalam sebuah kelompok mereka sepakat menyandikan setiap pesan dengan algoritma yang sama. Misalnya, algoritmanya adalah mempertukarkan pada setiap kata karakter pertama dengan karakter ketiga, karakter kedua dengan karakter keempat, karakter kelima dengan karakter ketujuh, karakter keenam dengan karakter kedelapan, dan seterusnya. Contohnya:

Plaintext: MATEMATIKA DISKRIT

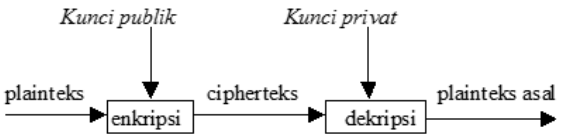
Ciphertext: TEMATIMAKA SKDITIR

Untuk mendekripsi pesan, algoritma yang sama digunakan kembali. Sayangnya, algoritma *restricted* tidak cocok lagi digunakan karena bila salah satu orang anggota kelompok keluar, maka algoritma penyandian pesan harus diubah jika tidak ingin pesan bocor ke publik. Maka, algoritma seperti ini kerahasiaannya tidak dapat diandalkan.

Kriptografi modern tidak lagi mendasarkan kekuatan pada algoritmanya. Jadi algoritmanya tidak dirahasiakan dan boleh diketahui umum. Lantas, dimana kerahasiannya? Kekuatan kriptografinya terletak pada kunci, yang berupa deretan karakter atau bilangan bulat, dijaga kerahasiannya. Hanya orang yang mengetahui kunci dapat melakukan enkripsi dan dekripsi. Analogi kunci ini mirip dengan sandi atau *password.* Bedanya, sandi digunakan untuk otorisasi akses, sedangkan kunci pada kriptografi untuk proses enkripsi dan dekripsi.

2.5.4 Algoritma pada Kriptografi Modern

Proses enkripsi dan dekripsi pada algoritma kriptografi modern adalah sebagai berikut:



Gambar 1.1 Enkripsi dan dekripsi pada algoritma kriptografi modern
Sumber: Rinaldi Munir/ IF2091/ Matematika Diskrit

Jika kunci publik = kunci privat, maka algoritma kriptografinya disebut algoritma simetri. Contoh algoritma yang sering digunakan adalah DES (*Data Encryption Standard).* Sedangkan jika kunci enkripsi dan dekripsinya berbeda, maka algoritmanya disebut algroitma nirsimetri. Contoh yang sering digunakan adalah algoritma RSA (*Rivest Shamir Adleman*).

Algoritma simetri disebut juga algoritma kunci pribadi karena kunci enkripsi dan dekripsi sama sehingga perlu dirahasiakan. Kelemahan sistem ini adalah baik pengirim maupun penerima pesan harus memiliki kunci yang sama, sehingga pengirim pesan harus mencari cara untuk memberi tahu kunci kepada penerima tanpa diketahui pihak lain.

Algoritma nirsimetri disebut juga algoritma kunci publik. Algoritma ini umum dipakai di lingkungan jaringan komputer. Algoritma ini terdiri dari 2 buah kunci yakni kunci publik untuk enkripsi dan kunci rahasia untuk dekripsi yang tentu saja harus dirahasiakan. Misal jaringan komputer menghubungkan komputer karyawan di kantor cabang dengan kantor manajer di kantor pusat. Seluruh karyawan diperintahkan bahwa kalau mereka mengirimkan laporan ke manajer di kantor pusat, mereka harus mengenkripsikan laporan tersebut dengan kunci publik milik manajer. Untuk mengembalikan data yang sudah terenkripsi ke data awal, manajer menggunakan kunci rahasia dimana hanya dia-lah yang mengetahui kunci tersebut. Selama proses transmisi *ciphertext* dari kantor cabang ke kantor pusat melalui saluran komunikasi mungkin saja data yang dikirim disadap oleh pihak ketiga, namun pihak ketiga tidak akan mendapat apa-apa karena dia tidak dapat mengembalikan *ciphertext*-nya menjadi teks semula karena tidak mengetahui kunci dekripsinya.

2.6 Algoritma RSA

Algoritma RSA diperkenalkan oleh tiga peneliti dari MIT (*Massachussets Institute of Technology*), yaitu Ron Rivest, Adi Shamir, dan Len Adleman, pada tahun 1976. RSA mendasarkan proses enkripsi dan deskripsinya pada konsep bilangan prima dan aritmetika modulo. Baik kunci enkripsi maupun kunci deskripsi keduanya berupa bilangan bulat. Kunci enkripsi tidak dirahasiakan dan diketahui umum (sehingga dinamakan juga kunci publik), namun kunci untuk deskripsi bersifat rahasia. Kunci deskripsi dibangkitkan dari beberapa buah bilangan prima bersama-sama dengan kunci enkripsi. Untuk menemukan kunci deskripsi, orang yang ingin menemukannya harus memfaktorkan suatu bilangan non prima menjadi faktor primanya. Kenyataannya, memfaktorkan bilangan non prima menjadi faktor primanya bukanlah pekerjaan yang mudah. Belum ada algoritma yang efisien yang ditemukan untuk pemfaktoran itu. Semakin besar bilangan non primanya tentu semakin sulit pula pemfaktorannya. Semakin sulit pemfaktorannya, semakin kuat pula algoritma RSA. Algoritma RSA sebenarnya sederhana sekali. Secara ringkas, algoritma RSA adalah sebagai berikut:

a. Pembangkitan pasangan kunci

1. Pilih dua bilangan prima, a dan b (rahasia)
2. Hitung n = a b. Besaran n tidak perlu dirahasiakan.
3. Hitung m = (a – 1)(b – 1). Sekali m telah dihitung, a dan b dapat dihapus untuk mencegah diketahui oleh pihak lain.
4. Pilih sebuah bilangan bulat untuk kunci publik, sebut namanya e, yang relatif prima terhadap m.
5. Hitung kunci dekripsi, d, melalui e x d ≡ 1 (mod m).

b. Enkripsi

1. Nyatakan pesan menjadi blok-blok plainteks: p_1, p_2, p_3, … (harus dipenuhi persyaratan bahwa nilai p_i harus terletak dalam himpunan nilai 0, 1, 2, …, n – 1 untuk menjamin hasil perhitungan tidak berada diluar himpunan)
2. Hitung blok cipherteks c_i untuk blok plainteks p_i dengan persamaan

$$c_i = p_i^e \bmod n \qquad (11)$$

yang dalam hal ini, e adalah kunci publik.

c. Dekripsi

Proses dekripsi dilakukan dengan menggunakan persamaan

$$p_i = c_i^d \bmod n, \qquad (12)$$

yang dalam hal ini, d adalah kunci privat.

III. DOKUMEN RAHASIA DAN PERANAN *WIKILEAKS*

3.1 Dokumen Rahasia

Dokumen negara, ada yang bersifat rahasia ataupun publik. Untuk yang berlabel rahasia, tidak boleh diketahui atau diakses oleh pihak yang tidak berhak. Namun, penyebaran informasi rahasia ini sering terjadi.

Seperti kita ketahui, dokumen pemecatan Prabowo beredar dan sempat menimbulkan kekacauan[4]. Dokumen rahasia itu berisi tentang hasil sidang Dewan Kehormatan Perwira terhadap Letnan Jenderal Prabowo Subianto. Kementerian Pertahanan mengatakan kasus ini merupakan ranah Badan Intelijen Negara karena dokumen itu masuk dalam rahasia negara. Tetapi, Kepala BIN malah meminta Markas Besar Tentara Nasional Indonesia untuk melakukan evaluasi internal terkait kasus tersebut. Mereka saling melempar tanggung jawab. Dokumen ini dengan mudah bocor, dikarenakan isi dokumen ini *plaintext* sehingga siapapun dapat membacanya asal memiliki dokumen tersebut.

Dokumen negara biasanya bocor bukan hanya karena penyadapan, namun sangat mungkin dilakukan oleh orang dalam, biasanya pegawai pemerintah yang sebenarnya tidak mempunyai hak akses ke dokumen tersebut.

Pernah terjadi kasus di Banyuwangi dimana PNS di lingkungan pemerintah membocorkan dokumen SPJ pada satuan kerja daerah[5]. Demikian pula pernah terjadi kasus hilangnya dokumen negara yang dibawa Pemerintah Indonesia ketika melakukan perjanjian jual-beli senjata di Seoul, Korea[6]. Pembocoran ini dapat terjadi karena dokumen negara dikirim atau disimpan dalam bentuk *plaintext*. Apabila dokumen tersebut berupa *ciphertext*, maka kekhawatiran tentang bocornya rahasia negara tidak perlu terjadi.

3.2 *Wikileaks*

Situs independen *Wikileaks* tidak menyadap kawat diplomatik suatu negara, melainkan menerima file dari siapa saja orang di dunia ini yang mengirimkan dokumen pada *Wikileaks.* Kebocoran dokumen seperti ini dapat memicu kekacauan, kesalah-pahaman, dan bahkan ketegangan politik.

Markas *Wikileaks* berada di pegunungan dekat Stockholm, ibu kota Swedia. *Wikileaks* bertindak sebagai wartawan dan pengirim dokumen bertindak sebagai sumber sehingga seperti kita tahu Hukum Swedia melarang

untuk membocorkan identitas sumber, jika dibocorkan maka dapat dituntut karena telah melakukan tindak kriminal. Saat pengiriman dokumen, *Wikileaks* memanfaatkan teknologi SSL untuk mengenkripsi dan mengamankan sambungan. *Wikileaks* juga menyediakan situs menggunakan jaringan TOR agar pengguna dapat menyembunyikan alamat IP asalnya. *Wikileaks* juga melakukan penghapusan metadata dokumen[7].

Terkait dengan Indonesia, *WikiLeaks* menengarai telah mengantongi lebih dari 3.000 dokumen rahasia atau laporan diplomatik Amerika Serikat yang dikirim ke dan dari Kedutaan Besar Amerika Serikat di Jakarta dan konsulat jenderal di Surabaya.

Tiga dokumen telah dirilis, antara lain mengungkapkan Program Pelatihan dan Pendidikan Militer Internasional bagi Indonesia pascatragedi Santa Cruz yang disebut-sebut melibatkan TNI/Kopassus, serta intervensi Amerika Serikat dalam proses referendum Timor Timur pada 1999 yang bermuara pada lepasnya wilayah itu dari Indonesia[8].

Dengan ini, dapat kita simpulkan bahwa dokumen negara yang diserahkan sumber kepada *Wikileaks* adalah berupa *plaintext*.

IV. PENERAPAN ALGORITMA RSA DALAM MENGENKRIPSI DOKUMEN RAHASIA NEGARA

4.1 Penggunaan pada Dokumen Negara

Seperti pada pembahasan di Bab III, dokumen negara bisa bocor karena dokumen tersebut berbentuk *plaintext*. Kebocoran dokumen ini dapat ditanggulangi, jika seluruh dokumen dienkripsi ke bentuk *ciphertext*. Semua pegawai boleh tahu kunci publiknya, namun hanya pejabat yang berwenang yang mengetahui kunci rahasia untuk dekripsinya.

Berikut akan dijelaskan tentang konsep dari algoritma RSA. Pada algoritma RSA terdapat 3 langkah utama yaitu *key generation* (pembangkitan kunci), enkripsi, dan dekripsi. Kunci pada RSA mencakup dua buah kunci, yaitu *public key* dan *private key*. *Public key* digunakan untuk melakukan enkripsi, dan dapat diketahui oleh orang lain. Sedangkan *private key* tetap dirahasiakan dan digunakan untuk melakukan dekripsi.

Untuk algoritma RSA sendiri sudah dibahas pada Bab II, akan diberikan contoh disini agar lebih mudah dipahami.

Misalnya dipilih 2 bilangan prima, a = 47 dan b = 71. Maka kita hitung n = a x b = 3337 dan m = (a – 1) x (b –1) = 3220. Pilih kunci publik e yang relatif prima dengan m misal e = 79 karena PBB (79, 3220) = 1. Nilai e dan n dapat dipublikasikan ke umum untuk enkripsi.

Selanjutnya akan dihitung kunci dekripsi *d* dengan kekongruenan:

$$e \times d \equiv 1 \pmod{m} \qquad (13)$$

Maka didapat persamaan:

$$d = \frac{1+(k\ x\ 3220)}{79} \qquad (14)$$

Dengan mencoba nilai-nilai k = 1, 2, 3, …, diperoleh nilai *d* yang bulat adalah 1019. Ini adalah kunci dekripsi.

Misalkan plainteks *P* = HARI INI atau dalam desimal ASCII: 7265827332737873.

Pecah *P* menjadi blok yang lebih kecil (misal 3 digit):

$p_1 = 726$ $p_4 = 273$
$p_2 = 582$ $p_5 = 787$
$p_3 = 733$ $p_6 = 003$

Enkripsi setiap blok:

$c_1 = 726^{79} \bmod 3337 = 215$
$c_2 = 582^{79} \bmod 3337 = 776$
dst untuk sisa blok lainnya

Keluaran: chiperteks *C* = 215 776 1743 933 1731 158.

Dekripsi (menggunakan kunci privat d = 1019)

$p_1 = 215^{1019} \bmod 3337 = 726$
$p_2 = 776^{1019} \bmod 3337 = 582$
dst untuk sisi blok lainnya

Keluaran: plainteks *P* = 7265827332737873 yang dalam ASCII karakternya adalah HARI INI.

Bilangan prima p dan q yang biasa dipilih untuk pembangkitan kunci RSA biasanya berkisar antara 512 bit sampai 1024 bit.

Cara untuk memilih bilangan prima p dan q yang besar untuk pembangkitan kunci RSA adalah sebagai berikut:

1. Pilih bilangan ganjil sebesar 512 bit secara acak, misal p
2. Cek apakah p prima, jika ya, gunakan p. ini diprediksikan berhasil setelah menguji sebanyak log(p) / 2 atau sekitar 177 percobaan
3. Jika p bukan prima, coba untuk p = p + 2, lalu ulangi ke langkah 2

Ini merupakan cara yang digunakan oleh OpenSSL.

Secara resmi, cara membangkitkan kunci pada RSA sudah dijelaskan pada *Federal Information Processing Standard* pada FIPS pub 186-4 pada kategori *Computer Security* dan subkategori *Cryptography*, pada *section* 5.1 dan *appendix* B.3.1[9]. FIPS 186-4 merupakan publikasi yang mendeskripsikan secara lengkap bagaimana penentuan bilangan prima untuk aplikasi kriptografi. FIPS menggunakan dasar *Miller-Rabin* tetapi dengan penambahan minor didalamnya seperti *nlen* (panjang modulus n pada bit), dan sebagainya.

Untuk Bahasa C sendiri, sudah ada *function* yang mengembalikan bilangan prima selanjutnya dari masukan bilangan yang kita input. GMP, atau *GNU Multi Precision Arithmetic Library*, adalah *library* gratis yang disediakan untuk masukan angka acak yang biasanya dioperasikan dalam operasi bilangan bulat, rasional, dan real. GMP ditujukan untuk aplikasi kriptografi dan tujuan penelitian[10]. Untuk fungsi teori bilangan yang berhubungan dengan bilangan prima, dapat digunakan fungsi berikut:

1. Menentukan apakah n prima

int mpz_probab_prime_p (const mpz_t n, int reps)

Fungsi ini akan mengecek apakah n prima. Jika n pasti prima, dikembalikan angka 2. Jika n mungkin prima, maka dikembalikan 1. Mengembalikan 0 jika n pasti bukan prima atau komposit. Fungsi ini menggunakan percobaan pembagian yang berulang-ulang beserta Test Uji Prima *Miller-Rabin* didalamnya.

2. Mengembalikan bilangan prima yang lebih besar terdekat dari op

void mpz_nextprime (mpz_t rop, const mpz_t op)

Prosedur ini akan menentukan nilai rop sebagai pengembalian bilangan prima terdekat yang lebih besar dari op. Fungsi ini menggunakan algoritma probabilitas untuk menentukan bilangan-bilangan prima.

Selain itu, syarat algoritma RSA yang lain adalah p-1 dan q-1 dianjurkan agar tidak mudah untuk difaktorkan, serta p dan q bukan bilangan yang berdekatan.

Cara lain yang mudah untuk menghasilkan angka prima yang besar adalah dengan memilih angka acak dengan panjang digitnya sebesar angka prima yang kita inginkan, lalu uji dengan Teori Fermat dengan memilih basis 2 agar paling optimal untuk kecepatan pengujian, lalu kemudian menerapkan sejumlah *Miller-Rabin* tes (tergantung pada panjang dan tingkat *error* yang diperbolehkan seperti 2^{-100}) untuk mendapatkan angka yang sangat mungkin merupakan bilangan prima.

RSA tidak menggunakan angka prima yang sudah diketahui. RSA memilih angka baru yang sangat besar lalu menggunakan suatu algoritma untuk mencari bilangan terdekat yang mungkin saja prima. Banyak *pseudocode* algoritma yang bertebaran di internet untuk menguji apakah bilangan tersebut merupakan prima, ataupun mengembalikan angka prima selanjutnya dari input yang kita masukkan mulai *dari AKS test, APR test, Fermat, Lucas, Solovay-Strassen, Miller-Rabin*, dan lain-lain.

Dokumen negara biasanya merupakan *plaintext* yang berupa huruf, maka dapat kita gunakan kode ASCII untuk mengkonversi huruf ke angka. Namun dengan algoritma RSA, ukuran dokumen dalam *ciphertext* bisa membengkak jauh lebih besar dari *plaintext*-nya, sehingga membutuhkan memori yang sangat besar. Tetapi, secara logika memori bukanlah masalah jika dibandingkan dengan kerahasiaan suatu negara.

Tindakan bawahan yang menjual dokumen negara ke pihak luar, atau tindakan pencurian dokumen, dapat diminimalisir dan ditingkatkan keamanannya apabila dokumen tersebut selalu dalam bentuk *ciphertext* dan hanya didekripsi ketika dibaca, dan akan dienkripsi lagi ketika disimpan. Dengan demikian, keamanan dokumen tersebut dapat terjamin.

4.2 Kekuatan dan Keamanan RSA

Kekuatan algoritma RSA terletak pada tingkat kesulitan dalam memfaktorkan bilangan non-prima menjadi faktor primanya, yang dalam hal ini n = a x b. Sekali n berhasil difaktorkan menjadi a dan b, maka m = (a – 1) x (b – 1) dapat dihitung. Selanjutnya, karena kunci enkripsi e diumumkan (tidak rahasia), maka kunci dekripsi d dapat dihitung dari persamaan $e \times d \equiv 1 \pmod m$. Ini berarti proses dekripsi dapat dilakukan oleh orang yang tidak berhak.

Mungkin sebagian dari kita akan berpikir bahwa jika sudah ada suatu *library* yang menyimpan daftar bilangan prima pada suatu bit tertentu, dapat dengan mudah tentunya menjebol keamanan dari pembangkitan kunci RSA. Tapi perlu diketahui, bahwa untuk 1024 bit saja, terdapat sekitar 2.8×10^{147} bilangan prima. Dari 4096 bit, terdapat sekitar 7×10^{163} bilangan prima yang dapat digunakan untuk pembangkitan kunci RSA. Jadi dari kombinasi kedua itu saja, kita memiliki 4.9×10^{1227} kemungkinan pasangan bilangan prima. Tentu ini jumlah yang sungguh besar. Jadi jika hanya kita coba-coba, sangat tidak mungkin untuk memilih p dan q secara tepat. Jika kita telah mencoba triliunan kali, bahkan *septillion* kali untuk memasangkan p dan q agar sesuai dengan kunci RSA yang ingin kita tembus, peluang kita benar tak lebih dari probabilitas 10^{-123}[11].

Penemu algoritma RSA menyarankan nilai a dan b panjangnya lebih dari 100 digit. Dengan demikian hasil kali n = a x b akan berukuran lebih dari 200 digit. Bayangkan seberapa besar usaha kerja yang diperlukan untuk memfaktorkan bilangan bulat 200 digit menjadi faktor primanya. Usaha untuk mencari faktor bilangan 200 digit membutuhkan waktu komputasi selama 4 milyar tahun dengan asumsi bahwa algoritma pemfaktoran yang digunakan adalah algoritma yang tercepat saat ini dan komputer yang digunakan mempunyai kecepatan 1 milidetik. Selagi belum ditemukan algoritma yang mangkus untuk memfaktorkan bilangan bulat menjadi faktor primanya, algoritma RSA masih direkomendasikan untuk penyandian pesan.

4.3 Penyerangan yang Mungkin terhadap RSA

Meski RSA cenderung aman bukan berarti tidak bisa dilakukan *attack* terhadap enkripsinya, apalagi didukung perkembangan *hardware* yang semakin cepat dan berkembang. Untungnya sampai saat ini, memecahkan enkripsi RSA yang memakai 129 digit bilangan bulat dipecahkan dengan menggunakan kumpulan 87 unit komputer *dual processor*, 30 unit komputer empat *processor*, 100Mhx SPARCstation. Berikut diterangkan segala kemungkinan *attack* terhadap RSA[12]:

1. Mencari kunci pribadi berdasarkan kunci publik.

Dilakukan dengan melakukan pemfaktoran n dari kunci publik. Kemudian dari n didapatkan p dan q, bersama dengan e maka didapatkan d. Masalah disini adalah memfaktorkan bilangan n. Apabila n cukup besar, penyerangan seperti ini semakin sulit.

2. Menebak sebagian dari kata atau *single message attack*

Dilakukan dengan mengenkripsi kata tersebut menggunakan kunci publik dan membandingkan hasilnya dengan data asli yang terenkripsi. Meskipun cara ini sangat mungkin, ini dapat ditangkal dengan menyusupi bit-bit random dalam pesan.

3. *General Number Field Sieve* (GNFS)

Ini adalah cara *attack* yang paling dikenal. Dilakukan dengan memfaktorkan n ke bilangan prima p dan q. Tetapi waktu yang dibutuhkan untuk RSA dengan besar kunci 1024 bit sekitar 3×10^{11} MIPS-year (MIPS-year adalah komputasi 1 juta instruksi per detik selama setahun).

Meskipun kecepatan masih bisa ditingkatkan dengan *hardware* yang lebih baik, apalagi dengan semakin

meningkatnya daya komputasi komputer dari waktu ke waktu, maka harus diakui semakin memperbesar kemungkinan memecahkan ekripsi RSA, namun sampai saat ini RSA masih dirasa sangat aman untuk digunakan pada kriptografi karena digit bilangan bulat yang dipakai masih dapat diperbesar dan disesuaikan.

V. KESIMPULAN

Tentu dalam mengaplikasikan algoritma RSA ini, dibutuhkan pemahaman yang cukup mengenai teori bilangan, seperti pembagi bilangan terbesar, kombinasi lanjar, relatif prima, aritmetika modulo, kekongruenan, invers modulo, dan lain-lain. Dengan pemahaman secara penuh, kita dapat menemukan banyak aplikasi lain dari teori bilangan yang tidak ditemukan sebelumnya. Dengan teori bilangan, banyak hal yang kita rasa sulit untuk dipecahkan menjadi lebih mudah diselesaikan.

Hikmah penting dari kasus kebocoran dokumen rahasia negara adalah perlunya meninjau dan merevisi serta memperketat sistem informasi intelijen, termasuk menata ulang dan meningkatkan standardisasi pengiriman, penyimpanan, dan dokumentasi data intelijen. Dalam hal ini, penulis menyarankan agar semua dokumen rahasia dienkripsi dalam bentuk *ciphertext* dengan memanfaatkan algoritma RSA dengan kunci RSA yang digunakan minimal 1024 bit.. Dapat kita lihat dari penjelasan tadi bahwa RSA masih sangat sulit ditembus dan masih menjadi algoritma yang paling direkomendasikan untuk penyandian. Kedepannya, penulis mengharapkan pemerintah dapat merealisasikannya sehingga kasus-kasus kebocoran yang sudah terjadi tidak terjadi lagi dan Badan Intelijen Negara bersama Kementerian Pertahanan serta Kementerian Teknologi Informasi bersama-sama mengembangkan pemanfaatan RSA ini sehingga dokumen negara akan benar-benar dapat aman penyimpanannya, dapat dipercaya kerahasiaannya, dan dapat dipertanggung-jawabkan hanya oleh pihak yang berhak.

VI. UCAPAN TERIMA KASIH

Penulis pertama-tama ingin mengucapkan terima kasih kepada orang tua penulis, karena sampai detik ini ayah dan ibu selalu membimbing, mendidik, dan mendukung penulis, baik secara moril maupun non-moril. Penulis juga mengucapkan terima kasih yang sebesar-besarnya kepada Pak Rinaldi Munir dan Bu Harlili atas pengajarannya selama ini pada mata kuliah Matematika Diskrit. Penulis mendapat begitu banyak manfaat setelah mempelajari mata kuliah ini, baik secara alur pemikiran, cara penulis memandang masalah, menyelesaikan masalah, dan lainnya. Terakhir, penulis juga ingin bersyukur dan mengucapkan terima kasih kepada keluarga, teman, lingkungan, dosen, dan seluruh elemen semesta yang tak lepas saling berhubungan dan mendukung sehingga penulis menjadi seperti diri penulis yang sekarang.

REFERENSI

[1] Pengertian Dokumen dan Dokumentasi. http://inamayladin.blogspot.co.id/2013/11/pengertian-dokumen-dokumentasi.html. Diakses pada 8 Desember 2015

[2] Munir, Rinaldi. Matematika Diskrit. Bandung: Informatika Bandung. 2001.

[3] Igor L. Markov, Mehdi Saeedi, "Constant-Optimized Quantum Circuits for Modular Multiplication and Exponentiation", Quantum Information and Computation, Vol. 12, No. 5&6, pp. 0361-0394, 2012.http://arxiv.org/abs/1202.6614

[4] Dokumen Pemecatan Prabowo Bocor, Kemenhan: Tanya BIN. http://politik.news.viva.co.id/news/read/511580-dokumen-pemecatan-prabowo-bocor--kemenhan--tanya-bin. Diakses pada 8 Desember 2015.

[5] Bocorkan Rahasia Negara, Pemkab Banyuwangi Peringatkan PNS. http://adf.ly/25959/banner/http://www.suarasurabaya.net/v06/kelanakota/?id=03d8cf62f619871ad6b080e0c1f11009201197375. Diakses pada 6 Desember 2015.

[6] Kebodohan Pejabat Indonesia Ancam Keamanan Negara, Dokumen Kok Bisa Hilang! http://rimanews.com/read/20110220/17361/kebodohan-pejabat-indonesia-ancam-keamanan-negara-dokumen-kok-bisa-hilang Diakses pada 7 Desember 2015.

[7] Penerapan Algoritma RSA untuk Dokumen Negara. http://ahimsadenhasafrizal.wordpress.com/2012/05/13/penerapan-algoritma-rsa-untuk-keamanan-dokuen-negara/. Diakses pada 8 Desember 2015.

[8] Membedah Kasus Wikileaks. http://nasional.kompas.com/read/2010/12/14/03333177/Membedah.Kasus.WikiLeaks. Diakses pada 8 Desember 2015.

[9] FIPS PUB 186-4. 2013. *Federal Information Processing Standard Publication.* Gaithersburg, MD. National Institute of Standards and Technology.

[10] Number Theoric Function. https://gmplib.org/manual/Number-Theoretic-Functions.html. Diakses pada 8 Desember 2015.

[11] How are Primes Generated for RSA? http://crypto.stackexchange.com/questions/1970/how-are-primes-generated-for-rsa. Diakses pada 7 Desember 2015.

[12] Algoritma RSA. http://stanlysk.blogspot.co.id/2012/05/algoritma-rsa.html. Diakses pada 8 Desember 2015.

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, 8 Desember 2015

Dharma Kurnia Septialoka - 13514028