Performa Enkripsi RSA

Jessin Donnyson / 13516112
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
Jessinra@gmail.com

***Abstrak*—Salah satu algoritma enkripsi yang terkenal adalah algoritma enkripsi *RSA* yang ditemukan pada tahun 1976. Algoritma jenis ini masih banyak digunakan untuk enkripsi data penting seperti data transaksi, kartu kredit, dan informasi penting lainnya yang tidak boleh diketahui pihak yang tidak berwenang sampai sekarang (2017). Penulis akan membahas mengapa jenis enkripsi ini termasuk salah satu enkripsi yang sangat aman, menggunakan analisis kombinatorial dan teori bilangan.**

***Kata Kunci*—algoritma, enkripsi, *RSA*, kombinatorial, teori bilangan.**

I. PENDAHULUAN

Tidak bisa dipungkiri bahwa teknologi informasi berkembang sangat pesat selama satu dekade terakhir, hampir segala aspek kehidupan dipermudah dengan adanya internet yang memungkinkan pertukaran informasi dilakukan dimana saja dan kapan saja, tanpa ada batasan ruang. Kita bisa membuat suatu konten informasi dan mengirimkannya kepada rekan kita di belahan dunia lainnya dengan sekejap, dan kita juga dapat mendapatkan informasi apa saja selama kita memiliki akses menuju informasi tersebut. Dengan demikian, hampir tidak ada lagi 'kerahaasiaan' informasi di dunia internet, dan tentunya hal ini merupakan sesuatu yang berbahaya. Banyak data yang perlu disimpan di internet dengan tujuan mempermudah akses, namun tetap harus dijaga kerahasiaannya dari pihak yang tidak berwenang, dan permasalahan inilah yang memunculkan ide untuk mengenkripsi data yang dikirim, supaya selain pihak berwenang yang dituju, tidak ada yang bisa mengetahui isi sebenarnya dari data yang dikirim.

Walaupun ada banyak metode enkripsi, beberapa metode yang paling umum dipakai adalah metode *Triple DES*, *RSA*, *Blowfish*, *Twofish*, dan *AES* [1]. Dari banyak metode enkripsi yang tersedia, penulis memilih untuk membahas algoritma enkripsi *RSA*, dengan alasan algoritma *RSA* memanfaatkan teori bilangan sebagai latar belakang cara kerja enkripsinya.

II. LANDASAN TEORI (TEORI BILANGAN)

Sebelum penulis membahas mengapa algoritma enkripsi *RSA* termasuk salah satu algoritma enkripsi yang sangat aman, perlu dibahas beberapa teori mendasar terlebih dahulu, pada subbab ini akan dibahas beberapa teori bilangan yang berkaitan dengan pembahasan enkripsi *RSA*.

A. Bilangan Bulat dan Sifat Pembagian Bilangan Bulat

Bilangan bulat adalah bilangan yang tidak mempunyai pecaha desimal, misalnya 5, 1661, -14, 0, berlawanan dengan bilangan riil yang memiliki desimal, misalkan 1.1, 34.25, dsb [2, V-7].

Salah satu konsep bilangan bulat yang fundamental adalah sifat pembagian, karena sifat ini melahirkan konsep bilangan prima (akan dibahas pada subbab berikutnya), dan aritmetika modulo (akan dibahas pada subbab berikutnya). Secara intuitif, suatu bilangan bulat bukan nol (misalkan a), dapat dinyatakan sebagai hasil kali dua bilangan bulat lainnya (misalkan b dan c) [2, V-7].

$$a = bc \qquad (1)$$

Seorang matematikawan Yunani, Euchlid, menyatakan suatu teorema: "Secara umum, jika suatu hasil pembagian bilangan bulat dinyatakan dalam serangkaian operasi terhadap bilangan bulat, akan terdapat dua komponen yaitu hasil bagi, dan sisa pembagian, dengan syarat pembagi n, $\{n \mid n > 0\}$, dan sisa pembagian r, $\{r \mid 0 \leq r < n\}$" [2, V-8].

$$m = nq + r \qquad (2)$$

m = *dividend* (bilangan yang dibagi)
n = *divisor* (bilangan pembagi)
q = *quotient* (hasil bagi)
r = *remainder* (sisa bagi)

B. Pembagi Bersama Terbesar

Suatu bilangan bulat dapat memiliki faktor pembagi yang mampu membagi habis bilangan tersebut tanpa sisa ($r = 0$). Suatu bilangan bulat dapat memiliki beberapa faktor pembagi yang merupakan faktor pembagi bilangan bulat lainnya juga. Misalkan: 36, salah satu faktor pembaginya adalah 9, (36 = 9.4 + 0), dan 27, salah satu faktor pembaginya juga merupakan 9, (27 = 9.3 + 0). Hal terpenting dari konsep faktor pembagi adalah faktor pembagi terbesar, yaitu suatu bilangan yang mampu membagi habis dua bilangan lain, disebut dengan pembagi bersama terbesar (selanjutnya akan disebut PBB). Misalkan a dan b adalah dua buah bilangan bulat tidak nol. PBB dari a dan b adalah bilangan bulat terbesar d sedemikian sehingga $d \mid a$ dan $d \mid b$ [2, V-8].

Untuk menentukan PBB dari dua buah bilangan bulat, dapat digunakan algoritma Euclidean (algoritma tidak dibahas lebih lanjut dalam makalah ini).

C. Aritmetika Modulo

Aritmetika modulo adalah cabang teori bilangan yang berperan penting dalam ilmu komputer, terutama pada cabang kriptografi. Suatu operasi modulo dua buah bilangan (misal: *a* mod *b*) akan menghasilkan 'sisa pembagian' dari *a* dibagi *b*. Mengadaptasi persamaan (2), jika suatu bilangan *m* dibagi dengan bilangan *n*, maka *m* mod *n* dapat dirumuskan sebagai persamaan (3) [2, V-13].

$$m = nq + r \quad \leftrightarrow \quad m \bmod n = r \qquad (3)$$

m = *dividend* (bilangan yang dibagi)
n = *divisor* (bilangan pembagi)
q = *quotient* (hasil bagi)
r = *remainder* (sisa bagi), $\{r \mid 0 \le r < n\}$

Konsep penting lainnya dari aritmetika modulo adalah konsep balikan modulo. Jika *a* dan *m* relatif prima dan $m > 1$, maka balikan (*invers*) modulo adalah *a'* [2, V-16].

$$a\,a' \equiv 1 \pmod m \qquad (4)$$

Algoritma untuk mencari balikan modulo dapat menggunakan algoritma Euclidean yang disusun ulang atau menggunakan kekongruenan lanjar (Algoritma tsb. tidak akan dijelaskan lebih lanjut dalam makalah ini).

D. Bilangan Prima

Salah satu jenis bidang dari teori bilangan yang memiliki pemanfaatan dalam ilmu komputer adalah konsep bilangan prima. Bilangan prima *p* adalah bilangan bulat positif yang lebih besar dari 1, dan hanya habis dibagi oleh 1 dan *p* itu sendiri. Apabila suatu bilangan bukan prima, maka bilangan tersebut disebut bilangan komposit [2, V-19].

Teorema penting yang menyangkut bilangan prima dinyatakan oleh teorema fundamental aritmetik: "Setiap bilangan bulat positif yang lebih besar dari 2, dapat dinyatakan sebagai hasil kali satu atau lebih bilangan prima" [2, V-19].

Hal lain yang perlu diketahui mengenai bilangan prima adalah, kemunculan bilangan prima tidak memiliki pola yang berulang, ataupun pola yang bisa diprediksi, sehingga membuktikan apakah suatu bilangan merupakan bilangan prima atau bukan merupakan suatu hal yang cukup sulit dilakukan (terutama untuk bilangan prima yang sangat besar) [4].

Salah satu cara menguji bilangan prima adalah dengan membagi bilangan prima *p* dengan semua kemungkinan bilangan prima dari 0 - $\sqrt{p}$, jika *p* habis dibagi salah satu bilangan prima tsb, maka *p* bukan prima. Algoritma ini sangat tidak mangkus jika ditinjau menggunakan kompleksitas algoritma [2, V-19].

Cara lain adalah dengan menggunakan teorema fermat, yaitu:

"Jika *p* adalah bilangan prima dan *a* adalah bilangan yang relatif prima dengan *p*, PBB (*p, a)* = 1, maka:

$$a^{p-1} \equiv 1 \pmod p \quad (5)$$

Namun teorema fermat ini juga tidak menjamin 100% dapat membuktikan bahwa suatu bilangan adalah prima atau bukan, karena terdapat beberapa bilangan komposit yang memenuhi (3), bilangan-bilangan tsb disebut prima semu (*pseudoprimes)* [2, V-20].

Menurut analisis kompleksitas algoritma, dari sekian banyak algoritma untuk mengecek apakah suatu bilangan *n* adalah bilangan prima, algoritma terbaik sampai saat ini adalah algoritma *probabilistic Miller–Rabin test*, yang memiliki kompleksitas $O((log\ n)^4)$. Jika digunakan komputer kuantum (*quantum computers*), dengan menggunakan algoritma gabungan *Shor's*, *integer factorization*, dan *Pocklington primality test*, kompleksitas hanya dapat diturunkan menjadi $O((\log n)^3\ (\log\log\ n)^{\ 2} \log\log\log n)$, hal ini menunjukan bahwa untuk membuktikan bahwa suatu bilangan adalah prima atau komposit, masih belum ditemukan algoritma yang benar-benar mangkus dan sangkil [3].

E. Kriptografi

Secara umum, kriptografi adalah ilmu yang mempelajari cara untuk menjaga kerahasiaan data / informasi, dengan cara menyamarkan data menjadi bentuk lain yang tidak memiliki makna. Dalam kriptografi, cabang matematika diskrit yang banyak dipakai adalah cabang teori bilangan, meliputi aritmetika modulo dan bilangan prima [2, V-21].

Cara kerja enkripsi adalah dengan cara merubah pesan asli (selanjutnya disebut sebagai plainteks) menjadi pesan yang tersamarkan (selanjutnya disebut sebagai chiperteks), lalu dikirim melalui media informasi. Sang penerima akan mendapat pesan dalam bentuk chiperteks, dan merubah chipperteks tsb. menjadi plainteks kembali supaya dapat dipahami, proses pengembalian chipperteks menjadi plainteks sering disebut proses dekripsi [2, V-21].

Dengan menerapkan proses enkripsi – dekripsi, pesan yang dikirim tidak akan bisa langsung diketahui makna aslinya oleh pihak yang tidak berwenang jika (entah bagaimana caranya) dapat diperoleh / 'disadap'.

Secara umum, enkripsi terbagi dua macam, yaitu enkripsi dengan algoritma simetri, dan algoritma nirsimetri. Algoritma simetri adalah algoritma yang enkripsi dan dekripsi menggunakan kunci yang sama, beberapa contoh terkenal adalah enkripsi *DES*, *Enigma machine* (mesin enkripsi Jerman pada perang dunia II). Sedangkan untuk enkripsi nirsimetri, dibutuhkan dua jenis kunci, yaitu kunci publik kunci pribadi. (analogi kunci publik adalah kotak surat yang dikunci dan hanya bisa dimasukkan surat saja, sedangkan kunci pribadi adalah kunci untuk membuka kotak surat tsb.) [2, V-24].

Pembahasan mendalam mengenai enkripsi *RSA* yang memanfaatkan bilangan prima dan aritmetika modulo akan dibahas lebih lanjut pada subbab berikutnya.

III. LANDASAN TEORI (KOMBINATORIAL)

Sebelum penulis membahas mengapa algoritma enkripsi *RSA* termasuk salah satu algoritma enkripsi yang sangat aman, perlu dibahas beberapa teori mendasar terlebih dahulu, pada subbab ini akan dibahas beberapa teori kombinatorial yang digunakan untuk menghitung jumlah kombinasi kemungkinan kunci enkripsi yang digunakan.

A. Kombinatorial

Kombinatorial adalah cabang matematika yang mempelajari pengaturan objek-objek. Tujuan yang ingin didapat dari kombinatorial adalah menghitung jumlah kemungkinan yang bisa dibentuk dari sekumpulan objek [2, VI-2].

B. Kaidah dasar

Dalam kombinatorial, dikenal dua jenis kaidah dasar untuk menghitung jumlah kemungkinan, yaitu kaidah perkalian dan penjumlahan. Misalkan percobaan 1 memiliki *p* hasil, dan percobaan 2 mempunyai *q* hasil, kaidah perkalian digunakan apabila hasil yang diinginkan adalah kemungkinan total percobaan 1 dan percobaan 2 (2 percobaan dilakukan sekaligus / simultan), total kemungkinan berasal dari hasil perkalian dari kedua hasil percobaan tsb. Sedangkan jika hasil yang diinginkan adalah total kemungkinan percobaan 1 atau 2, maka digunakan kaidah penjumlahan. (kedua percobaan berkomplemen, hanya salah 1 dari 2 yang dilakukan, atau keduanya dilakukan secara tidak simultan) [2, VI-2].

C. Permutasi

Permutasi adalah jumlah urutan berbeda dari pengaturan objek-objek. Permutasi adalah bentuk khusus aplikasi aturan perkalian. Misalkan jumlah objek adalah *n*, untuk memilih objek pertama, ada *n* kemungkinan, lalu objek ke-2 ada *n*-1 kemungkinan, objek ke-3 ada *n*-2 kemungkinan, dst. sampai tersisa 1 objek dan hanya 1 kemungkinan. Bentuk perkaliannya dapat dirumuskan menjadi persamaan (6) [2, VI-8].

$$n(n-1)(n-2)\ldots(2)(1) = n! \qquad (6)$$

Permasalahan serupa tetapi tidak sama dengan contoh diatas adalah ketika pemilihan dilakukan tidak sampai objek habis, namun hanya sebanyak *r*. Maka untuk memilih objek pertama, ada *n* kemungkinan, objek ke-2 ada *n*-1 kemungkinan, dst sampai objek ke-*r*, ada *n-r+1* kemungkinan. Bentuk perkaliannya dapat dirumuskan menjadi persamaan (7) [2, VI-9].

$$n(n-1)(n-2)\ldots(n-r+1) = \frac{n!}{(n-r)!} \qquad (7)$$

Jumlah susunan berbeda dari penyusunan *r* objek yang dipilih dari *n* objek disebut permutasi, dilambangkan dengan P(n, r) [2]. Sehingga, persamaan (7) dapat dibentuk menjadi persamaan umum permutasi (8) [2, VI-9].

$$P(n,r) = \frac{n!}{(n-r)!} \qquad (8)$$

D. Kombinasi

Kombinasi adalah bentuk khusus dari permutasi. Pada permutasi urutan kemunculan diperhatikan, sedangkan pada kombinasi, urutan kemunculan tidak diperhatikan. Karena urutan tidak menjadi pembeda, maka untuk banyak pemilihan *r*, maka terdapat *r*! kemunculan yang dianggap sama, sehingga persamaan (8) harus diubah menjadi (9) [2, VI-12].

$$C(n,r) = \frac{n!}{(n-r)!\,r!} \qquad (9)$$

IV. *RSA ENCRYPTION*

Algoritma enkripsi RSA adalah suatu algoritma enkripsi yang memanfaatkan teori bilangan (khususnya bilangan prima dan aritmetika modulo) dalam pembuatan kunci enkripsi-dekripsi, serta dalam enkripsi plainteks dan dekripsi chipper teks. Algoritma ini ditemukan oleh tiga peneliti dari MIT (*Massachussets Institute of Technology)*, yaitu Ron Rivest, Adi Shamir, Len Adleman, pada tahun 1976. RSA merupakan salah satu implementasi dari algoritma enkripsi yang menggunakan kunci nirsimetri, sehingga dibutuhkan satu pasang kunci berbeda (kunci enkripsi dan kunci dekripsi) untuk melakukan proses enkripsi dan dekripsi [2, V-25].

Kunci enkripsi bersifat umum, dan tidak rahasia, kunci ini digunakan semua orang yang ingin mengirimkan data yang bersifat rahasia kepada orang yang memiliki kunci dekripsi. Sebaliknya, kunci dekripsi bersifat rahasia, dan hanya bisa digunakan untuk membuka data yang dikirim dalam bentuk chipperteks [2, V-25].

Bisa disimpulkan bahwa jenis enkripsi RSA mudah untuk mengenkripsi plainteks ('kunci enkripsi' diberikan secara bebas), tetapi sulit untuk mendekripsi chipperteks apabila tidak memiliki 'kunci dekripsi'.

Pasangan kunci enkripsi dan dekripsi dibuat melalui beberapa tahap dengan algoritma pembangkitan kunci [2, V-25]:

1. Pilih dua bilangan prima berbeda misalnya a dan b. Kedua bilangan ini harus dirahasiakan.
2. Hitung $n = a \times b$. Bilangan n tidak dirahasiakan.
3. Hitung $m = (a-1) \times (b-1)$.
4. Hapus *a* dan *b* supaya mencegah ketahuan pihak lain.
4. Pilih sebuah bilangan *e* dimana bilangan ini mewakilkan kunci enkripsi (publik). Bilangan ini harus relatif prima terhadap bilangan *m* dimana PBB(e, m) = 1.
5. Hitung kunci dekripsi (pribadi) *d* yang merupakan invers *e* dalam modulo *m*, ($ed \equiv 1 \ (mod\ m)$).

Untuk proses enkripsi, pihak yang mempunyai plain teks menggunakan *e* (kunci enkripsi) dan *n* yang diberikan pihak penerima, lalu mengubah semua plainteks menjadi chipperteks dengan persamaan (10) [2, V-25].

$$C_i = p_i^e \ mod\ n \qquad (10)$$

Sedangkan untuk proses dekripsi, pihak penerima menggukanan kunci dekripsi *d* dan *n* yang sudah dibuat, lalu mengubah semua chipperteks menjadi plainteks dengan persamaan (11) [2, V-25].

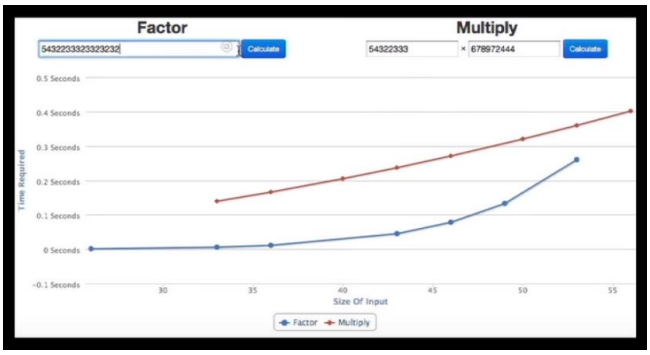
$$P_i = c_i^d \ mod\ n \qquad (11)$$

B. Kekuatan secara umum

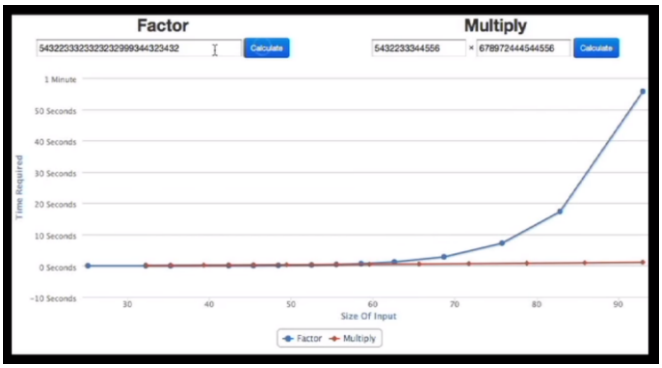
Secara sederhana, kemampuan algoritma RSA menghindari *bruteforce decryption* bergantung pada fakta bahwa perbandingan waktu yang diperlukan untuk mengalikan 2 bilangan berukuran sangat besar (katakanlah masing-masing

100 digit), sangat sedikit dibandingkan dengan waktu yang diperlukan untuk mencari 2 faktor prima dari hasil perkalian kedua bilangan tsb [5] .

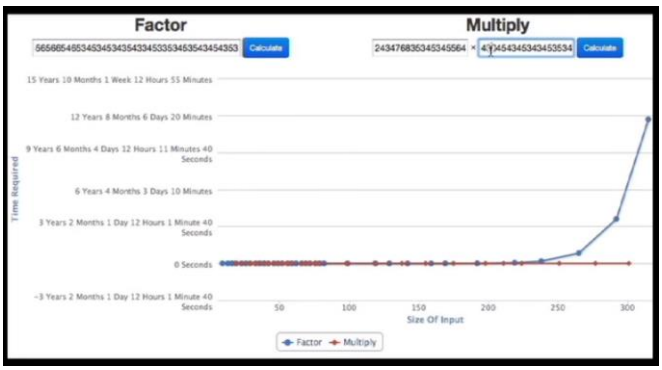
Perbandingan laju waktu rata-rata yang dibutuhkan untuk memproses pemfaktoran prima (kurva biru) dan perkalian dua bilangan (kurva merah) dinyatakan dalam beberapa gambar berikut:



Gambar 1. Perbandingan Laju Waktu Pemfaktoran dan Perkalian untuk Rentang Bilangan 0 – 50 digit.
Sumber (https://youtu.be/wXB-V_Keiu8?t=471)
Diakses: 2 Desember 2017



Gambar 2. Perbandingan Laju Waktu Pemfaktoran dan Perkalian untuk Rentang Bilangan 0 – 90 digit.
Sumber (https://youtu.be/wXB-V_Keiu8?t=476)
Diakses: 2 Desember 2017



Gambar 3. Perbandingan Laju Waktu Pemfaktoran dan Perkalian untuk Rentang Bilangan 0 – 300 digit.
Sumber (https://youtu.be/wXB-V_Keiu8?t=481)
Diakses: 2 Desember 2017

Dari perbandingan gambar 1, 2 dan 3, terlihat bahwa pertambahan waktu yang diperlukan untuk pemfaktoran prima bersifat eksponensial, sedangkan untuk perkalian 2 bilangan bersifat relatif konstan (peningkatan yang relatif tidak signifikan).

Terlihat dari grafik pemfaktoran untuk bilangan berdigit 400 (RSA-1024) memerlukan waktu rata-rata 633 tahun untuk memecahkan faktorisasi prima. Namun karena sumber tidak mencatat spesifikasi *processor* yang digunakan dalam percobaan, maka yang bisa disimpulkan dari grafik hanyalah 'perilaku' pertumbuhan dari kedua grafik.

C. Jenis Enkripsi RSA

Enkripsi RSA secara umum dapat dikategorikan menurut panjang bilangan hasil perkalian dua buah bilangan prima. Berdasarkan referensi [6], RSA diklasifikasikan menjadi beberapa jenis yaitu:

Tabel 1. Klasifikasi Enkripsi RSA

	Panjang Bilangan	Record *attack*	Kegunaan
RSA 516	155 digit	6 bulan	Sudah hampir tidak digunakan, bersifat edukasi.
RSA 768	231 digit	4 tahun (2009)	(dummy experiment)
RSA 1024	308 digit	-	- Perusahaan, - web page, - media social, - game company (pembelian barang game dengan *real-cash*) (skala menegah)
RSA 2048	617 digit	-	- Perusahaan internasional, - web page, - media sosial - *financial / trading*, - *internet banking*, - e-*commerce* (skala besar) Contoh : google, facebook, twitter, amazon, ebay
RSA 3072	924 digit	-	- *classified document* - *digital signature*, - data transaksi kartu kredit bank internasional - enkripsi pesan tingkat negara ,dll.

Menurut referensi [8] dan [9], usaha pembobolan RSA 1024 sudah dapat dilakukan, namun dengan catatan, upaya pembobolan tidak melalui proses pencarian kunci, namun dengan serangan hardware / menggunakan library *Libgcrypt* yang memang mempunyai 'bug', dan dieksploitasi untuk melakukan pemulihan kunci dekripsi secara parsial sedikit demi sedikit (paper original disertakan sebagai referensi) [10]. Berita baiknya, bug pada *Libgcrypt* sudah diupdate dan bug tersebut diperbaiki pada versi 1.7.8. [8]

C. Jumlah Kombinasi

Salah satu kekuatan algoritma RSA bergantung pada sulitnya mencari faktorisasi prima dari sebuah bilangan yang sangat besar. Namun, seandainya katakanlah seseorang berhasil membuat sebuah catatan data semua bilangan prima untuk memecahkan algoritma RSA, peluangnya pun akan tetap sangat kecil. Hal itu akan dibuktikan selanjutnya.

Berhubung bilangan prima tidak memiliki sekuens dalam kemunculan di deret bilangan, jumlah kemunculan bilangan prima tidak dapat dinyatakan secara pasti tanpa menghitung manual. Namun, dengan menggunakan pendekatan fungsi, jumlah suatu bilangan prima dari 1 sampai *n*, dapat dinyatakan sebagai persamaan (12) dan disederhakan menjadi (13) [4]. Penulis selanjutknya hanya akan menggunakan persamaan (13) untuk menghitung kombinatorial.

$$\lim_{x\to\infty} \frac{\pi(x)}{(\frac{x}{\ln(x)})} = 1 \qquad (12)$$

$$p \approx \frac{x}{\ln(x)} \qquad (13)$$

$\pi(x)$ = jumlah bilangan prima sebenarnya
x = banyak bilangan (0 sampai x)
p = banyak bilangan prima (dengan pendekatan)

Berdasarkan (12), maka jumlah setiap bilangan prima untuk setiap jenis enkripsi RSA dapat diaproksimasikan, sebagai contoh: untuk RSA 516 terdapat bilangan dengan panjang 155 digit, maka jumlah bilangan prima diantar 1 – 10^{155} adalah sebanding dengan $\frac{10^{155}}{155\ln(10)}$ atau setara dengan 2.820094 x 10^{151} bilangan. (perhitungan untuk jenis lain tidak akan ditulis secara eksplisit).

.

Tabel 2. Jumlah Bilangan Prima yang Berpotensi untuk beberapa Jenis RSA

	Panjang Bilangan	Aproksimasi Jumlah Prima
RSA 516	155 digit	$\approx 2.820094 \times 10^{152}$
RSA 1024	308 digit	$\approx 1.414640 \times 10^{305}$
RSA 2048	617 digit	$\approx 7.050235 \times 10^{613}$
RSA 3072	924 digit	$\approx 4.659812 \times 10^{920}$

Dengan mengetahui jumlah bilangan prima yang mungkin tersedia, dapat diperikirakan jumlah kombinasi dua bilangan prima yang diperlukan dengan menggunakan (9).

Sebagai contoh: untuk RSA 516, terdapat ± 2.820094 x 10^{151} bilangan prima diantara 0 – 10^{154}. Dari sekian banyak bilangan prima, hanya dipilih dua bilangan prima untuk menjadi faktor prima dari bilangan modulus RSA 516. Dengan demikian, digunakan $C(n, 2)$, dengan *n* adalah jumlah bilangan prima. Karena untuk bilangan sangat besar *n*, *n-1* sangat tidak signifikan (berbeda 1 angka dari 152 angka) maka bisa dikatakan $n \approx n\text{-}1$ (untuk mempermudah kalkulasi). Sehingga untuk menghitung kombinasi dari RSA, dapat digunakan persamaan (14).

$$C(n, 2) = \frac{n \times (n-1) \times (n-2)!}{r\,!(n-2)!} \approx \frac{n^2}{2} \qquad (14)$$

Tabel 3. Hasil Perhitungan Jumlah Kombinasi yang Mungkin untuk beberapa Jenis RSA

	Aproksimasi Jumlah Prima	Aproksimasi Jumlah Kombinasi Prima
RSA 516	$\approx 2.820094 \times 10^{152}$	$\approx 3.976465 \times 10^{304}$
RSA 1024	$\approx 1.414640 \times 10^{305}$	$\approx 1.000603 \times 10^{610}$
RSA 2048	$\approx 7.050235 \times 10^{613}$	$\approx 2.485291 \times 10^{1226}$
RSA 3072	$\approx 4.659812 \times 10^{920}$	$\approx 1.085692 \times 10^{1840}$

Berdasarkan tabel, terlihat jumlah kombinasi 2 bilangan prima yang dapat dipilih mempunyai orde pangkat 302, 608, 1225 dan 1839. Hal ini membuktikan jelas bukan hal yang mudah mencoba memecahkan enkripsi RSA dengan metode *bruteforce*.

Untuk keperluan perbandingan, dibuat sebuah tabel jumlah kombinasi prima relatif suatu RSA terhadap RSA lainnya. Isi dari tabel menggambarkan kekuatan RSA yang ditunjuk baris / kekuatan RSA yang ditunjuk kolom. (Pembuatan grafik tidak menunjang untuk data yang sangat besar perbedaanya dan diskrit).

Tabel 4. Jumlah Kombinasi Prima Relatif

	RSA 516	RSA 1024	RSA 2048	RSA 3072
RSA 516	1	(-)	(-)	(-)
RSA 1024	2.516×10^{306}	1	(-)	(-)
RSA 2048	6.250×10^{921}	2.484×10^{617}	1	(-)
RSA 3072	2.730×10^{1535}	1.085×10^{1230}	4.368×10^{612}	1

Misalkan penulis ingin mencoba menghitung berapa lama yang dibutuhkan untuk mencoba *bruteforce* algoritma RSA. Dalam perhitungan waktu. Beberapa asumsi yang dipakai adalah:

1. Semua bilangan sudah diketahui prima / komposit tanpa perlu melakukan verivikasi ulang.
2. Processor yang digunakan adalah Intel i7 octa-core (3.7GHz) secara maksimal. $\approx 3.178 \times 10^{10}$ proses/ detik.
3. Processor *super-computer* yang digunakan adalah processor termutakhir saat ini dengan *clockrate* 1 x 10^9 GHz secara maksimal. $\approx 1.073 \times 10^{18}$ proses /detik.

Tabel 5. Perkiraan Waktu Maksimum yang Dibutuhkan untuk Melakukan *Bruteforce Attack* Terhadap RSA 516

	i7 octa-core	*supercomputer*
RSA 516	3,9676 $\times 10^{284}$ tahun	1,1751 $\times 10^{277}$ tahun

Dari tabel, jelas terlihat sangat tidak mungkin melakukan *bruteforce attack*, sehingga serangan yang dicoba berikutnya adalah dengan mengeliminasi kombinasi dan menggunakan algoritma yang lebih cepat lagi. Diantaranya dengan cara:

1. Iterasi kombinasi dapat dioptimalisasi lagi sehingga hanya di cek perkalian yang menghasilkan digit angka hampir serupa dengan modulo yang digunakan dalam enkripsi (pasangan prima dengan digit *i* dan digit (*j-i)* dimana *j + i* adalah jumlah digit *n*. (kombinasi yang mungkin diminimalisir sampai $\approx \frac{jumlah\ prima}{2}$.
2. Validasi pasangan hanya dilihat dari beberapa digit terakhir hasil perkalian angka prima, sehingga mempercepat jauh proses validasi.
3. Pencarian dilakukan dengan algoritma yang memusat pada jawaban sehingga tidak semua kombinasi perlu dicoba.

Sayangnya, belum ada yang dapat memastikan berapa kompleksitas minimal yang bisa dicapai dengan menerapkan segala macam algoritma. Sehingga penulis tidak dapat memastikan estimasi waktu yang cukup akurat untuk melakukan *attack* terhadap enkripsi RSA.

Namun dengan menggunakan data bahwa RSA 516 dapat dibobol dalam 6 bulan, dan menggunakan estimasi dari Rivest dkk, "Untuk memfaktorkan bilangan 200 digit diperlukan waktu komputasi 4 milyar tahun dengan komputer berkecepatan 1milidetik [2, V-27]", serta fakta RSA 768 telah dapat dipecahkan dalam waktu 4 tahun. Penulis mencoba mengkalkulasi berapa kombinasi yang diperlukan sebelum RSA 516 terpecahkan. Penulis ingin mencoba membuat regresi linear antara jumlah digit dan lama pemecahan sebagai berikut:

$$T(X) \approx 0{,}046\,X - 6{,}638 \qquad (15)$$

T(X) = waktu yang dibutuhkan (tahun)
X = jumlah digit modulo enkripsi

Berdasarkan hasil regresi diatas, maka didapatkan estimasi waktu yang diperlukan untuk memecahkan RSA secara optimal dan tercepat adalah sebagai berikut:

Tabel 6. Perikraan Waktu Minimum yang Dibutuhkan untuk Melakukan *Organized Attack* Terhadap RSA

	Waktu yang dibutuhkan
RSA 516	0.5 Tahun
RSA 1024	7.5 Tahun
RSA 2048	22 Tahun
RSA 3072	36 Tahun

Perlu diketahui bahwa pertumbuhan kombinasi yang diperlukan bersifat eksponensial, sedangkan regresi yang dibuat adalah regresi linear, sehingga hasil yang didapat dengan regresi sangat kecil dibandingkan hasil sebenarnya yang mungkin ada. Juga perlu diingat perkembangan *processor* terus berkembang dengan sangat cepat, sehingga data pada tabel diatas hanya bisa menjadi patokan sangat kasar dari estimasi waktu yang diperlukan untuk membobol RSA.

V. KESIMPULAN

Berdasarkan analisis kombinatorial terhadap algoritma RSA, dapat disimpulkan bahwa enkripsi RSA masih sangat sulit untuk dipecahkan, terlihat dari jumlah kombinasi bilangan prima, kesulitan mencari faktor prima, dan pembuktian suatu bilangan adalah prima. Ketiga hal tersebut menjamin apabila data dienkripsi dengan metode RSA (2048 atau 3072), data akan sangat aman, setidaknya sampai RSA 1024 dapat dipecahkan (beberapa tahun kedepan).

Aspek pendukung lain dari RSA adalah untuk mengganti kunci enkripsi, usaha yang diperlukan jauh lebih mudah dibandingkan usaha untuk menebak kunci dekripsi. Bahkan kunci enkripsi dan modulo RHS 1024 hanya sebesar ± 1Kb saat dikirim kepada pengguna. Bukan suatu masalah untuk mengganti kunci secara periodik (misal setiap 1 bulan).

Untuk RSA 1024+, bentuk serangan yang dilakukan kemungkinan tidak lagi berusaha mencari kunci, namun lebih menggunakan serangan hardware / serangan virus dll, hal ini dikarenakan hampir mustahil mencari pemfaktoran dari bilangan yang sangat besar (300+ digit) menggunakan teknologi dan algoritma termutahkhir sampai saat ini.

.

VII. UCAPAN TERIMA KASIH

Saya mengucapkan terima kasih kepada Tuhan Yesus Kristus, yang atas berkat dan kasih karunia-Nya yang melimpah, saya dapat menyelesaikan makalah ini dengan baik.

Saya mengucapkan terima kasih kepada Bpk.Rinaldi Munir selaku dosen mata kuliah IF-2120 Matematika Diskrit yang telah memberikan bimbingan dan dasar pengetahuan yang cukup kepada saya untuk menyusun makalah ini.

Saya juga mengucapkan terima kasih kepada keluarga, khususnya orang tua saya, dan juga semua orang yang terlibat secara tidak langsung, yang senantiasa memberikan dukungan dalam doa, semangat, nasihat, dll.

REFERENCES

[1] https://www.storagecraft.com/blog/5-common-encryption-algorithms/
Diakses : 1 Desember 2017

[2] Munir, Rinaldi. “Matematika Diskrit ed 4” (2006)

[3] Gary L. Miller "Riemann's Hypothesis and Tests for Primality". (1976).

[4] The prime number theorem | Journey into cryptography | Computer Science | Khan Academy
https://www.youtube.com/watch?v=7jzCJJIc59E
Diakses : 1 Desember 2017

[5] Public Key Cryptography: RSA Encryption Algorithm
https://www.youtube.com/watch?v=wXB-V_Keiu8
Diakses : 1 Desember 2017

[6] Johnson, J.; Kaliski, B. "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1". (March 2016).

[7] Encryption and HUGE numbers – Numberphile
https://www.youtube.com/watch?v=M7kEpw1tn50
Diakses : 1 Desember 2017

[8] https://thehackernews.com/2017/07/gnupg-libgcrypt-rsa-encryption.html
Diakses : 1 Desember 2017

[9] https://www.engadget.com/2010/03/09/1024-bit-rsa-encryption-cracked-by-carefully-starving-cpu-of-ele/
Diakses : 1 Desember 2017

[10] Daniel J. Bernstein, Joachim Breitner, Daniel Genkin, Leon Groot Bruinderink, Nadia Heninger, Christine van Vredendaal, Tanja Lange and Yuval Yarom, “Sliding right into disaster: Left-to-right sliding windows leak”, (July 2017)
https://eprint.iacr.org/2017/627.pdf
Diakses : 2 Desember 2017

[11] https://security.stackexchange.com/questions/37907/how-long-is-a-2048-bit-rsa-key
Diakses : 2 Desember 2017

[12] https://security.stackexchange.com/questions/4518/how-to-estimate-the-time-needed-to-crack-rsa-encryption
Diakses : 2 Desember 2017

[13] https://crypto.stackexchange.com/questions/752/how-long-does-it-take-to-crack-des-and-aes
Diakses : 2 Desember 2017

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, 2 Desember 2017

Ttd (scan atau foto ttd)

Jessin 13516112