Metode-metode untuk Menghasilkan Bilangan Prima dan Aplikasinya di Bidang Kriptografi

Jonathan Alvaro 13516023
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13516023@std.stei.itb.ac.id

***Abstract*—Salah satu jenis angka yang masih belum diketahui secara menyeluruh oleh ilmuwan saat ini adalah angka prima. Padahal, angka prima memiliki berbagai aplikasi di berbagai bidang, terutama di bidang kriptografi dimana angka prima digunakan sebagai kunci enkripsi sebagian besar data yang beredar di internet saat ini.**

***Keywords*—bilangan prima, kriptografi, RSA**

I. Pendahuluan

Pada abad ke-21 ini, internet telah menjadi pilihan pertama sebagian besar manusia untuk sarana komunikasi. Setiap harinya, lebih dari 200 milyar *e-mail* dikirim oleh manusia di seluruh penjuru dunia untuk berkomunikasi satu sama lain. Memang, banyak dari pesan yang disampaikam melalui *e-mail* ini hanyalah percakapan sehari-hari yang tidak akan membahayakan siapapun bila diketahui oleh pihak ketiga. Akan tetapi, ada pula *e-mail* yang dikirim oleh perusahaan-perusahaan besar, petinggi negara, dan orang-orang atau instansi-instansi berpengaruh lainnya yang apabila isi pesan tersebut dibaca oleh pihak yang tidak berwenang, akan menghasilkan dampak yang sangat buruk. Oleh karena itu, dibutuhkan suatu cara untuk mengamankan semua data yang dikirimkan melalui internet agar semua pihak bisa tetap menjaga privasinya.

Dikarenakan adanya resiko terjadinya kejadian seperti yang disebutkan pada paragraf sebelumnya, maka dikembangkanlah berbagai cara untuk mengirimkan data melalui internet secara aman. Dan, teknik yang paling banyak digunakan oleh masyarakat umum (tanpa sepengetahuan mereka) untuk mengamankan data mereka adalah dengan mengenkripsi data mereka, atau lebih spesifiknya dengan menggunakan *public-key cryptography*. Dengan teknik ini, data yang akan dikirim oleh pengguna internet akan dienkripsi menggunakan suatu kunci berupa angka sebelum dikirim dan akan didekripsi di tempat tujuan dengan menggunakan kunci lain, sehingga hanya pihak-pihak yang memiliki kunci dekripsi bisa melihat isi dari data yang dikirim. Sistem ini bisa diimplementasikan secara aman dikarenakan adanya angka-angka prima yang memiliki sifat khusus yaitu tidak bisa dibagi habis oleh bilangan lain kecuali angka 1 dan bilangan itu sendiri.

II. Landasan Teori

A. Bilangan Prima

Bilangan prima adalah sekelompok bilangan-bilangan yang memiliki suatu sifat khusus, yaitu suatu bilangan prima hanya bisa dibagi habis oleh angka 1 dan bilangan itu sendiri. Atau, dengan kata lain, bilangan prima adalah suatu bilangan yang hanya memiliki satu faktor prima (angka 1 tidak termasuk faktor prima). Karakteristik dari bilangan prima ini menjadi suatu hal yang sangat membantu perkembangan kriptografi di era digital karena enkripsi yang dilakukan dengan menggunakan angka prima sebagai kunci akan lebih kuat dibandingkan enkripsi dengan angka non-prima sebagai kunci. Akan tetapi, di saat yang bersamaan, sifat ini juga menjadi penghambat implementasi dari teknik-teknik enkripsi yang sudah berhasil ditemukan, karena untuk aplikasinya di dunia nyata, dibutuhkan bilangan-bilangan prima yang sangat panjang, biasanya digunakan bilangan-bilangan yang memiliki setidaknya 200 digit. Hal ini sangat menyulitkan, karena dibutuhkan waktu dan komputer dengan kemampuan komputasi yang sangat kuat untuk mencari angka-angka prima yang cukup besar untuk digunakan karena sejauh ini, ilmuwan masih belum bisa menemukan hubungan antara suatu bilangan prima dengan bilangan-bilangan prima lainnya.

List of Prime Numbers up to 1000

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71,
73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149,
151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227,
229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307,
311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389,
397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467,
479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571,
577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653,
659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751,
757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853,
857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947,
953, 967, 971, 977, 983, 991, 997

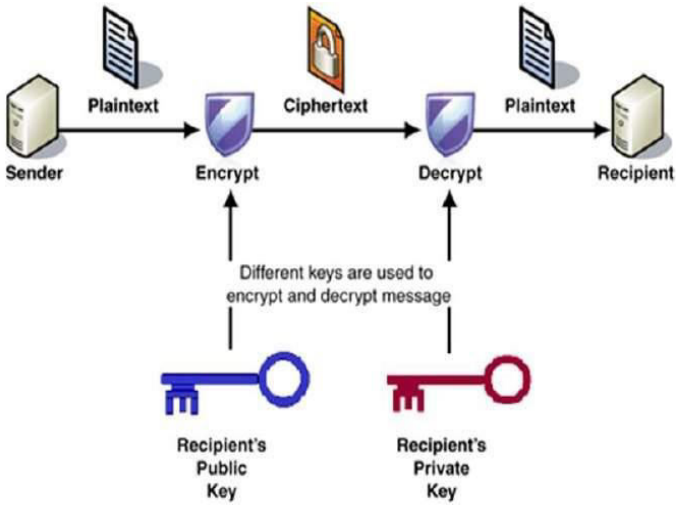
Sumber: https://www.miniwebtool.com/list-of-prime-numbers/

Gambar 1. Daftar bilangan prima yang lebih kecil dari 1000

B. *Public-key Cryptography*

Public-key Cryptography adalah suatu kategori enkripsi yang memungkinkan dua pihak untuk saling berkomunikasi secara aman tanpa harus bertemu tatap muka terlebih dahulu

untuk menentukan kunci enkripsi dan dekripsi. Sesuai namanya, pada kategori enkripsi ini, terdapat dua kunci berbeda, yaitu sebuah kunci enkripsi yang dapat diakses oleh siapapun (*public key)* dan sebuah kunci dekripsi berbeda yang hanya dimiliki oleh pihak yang akan menerima suatu data atau pesan (*private key*). Dengan adanya *public key* dan *private key*, siapapun bisa mengirim data atau pesan yang terenkripsi pada pihak lain dengan menggunakan *public key* milik pihak yang dituju, dan pesan itu akan aman sampai di tujuan tanpa bisa diketahui oleh pihak ketiga karena hanya pihak yang dituju yang memiliki *private key* yang dibutuhkan untuk mendekripsi pesan atau data tersebut.



Sumber: www.tutorialspoint.com/cryptography/public_key_encryption.html

Gambar 2. Visualisasi cara kerja *Public-key Cryptography*

Salah satu teknik enkripsi yang termasuk dalam kategori ini adalah *RSA Cipher* (Rivest-Shamir-Adleman). Kunci kesuksesan dari teknik enkripsi RSA ini adalah digunakannya bilangan prima yang sangat besar untuk menghasilkan kunci enkripsi dan dekripsi. Hal ini mengakibatkan enkripsi ini mustahil untuk ditembus dengan algoritma *brute force* bila bilangan prima yang digunakan untuk menghasilkan kunci enkripsi dan dekripsi cukup besar.

C. Teknik untuk Menghitung Bilangan Prima

Untuk saat ini, meskipun kita masih belum mengetahui secara persis hubungan antar bilangan-bilangan prima, tetapi telah ditemukan beberapa metoe untuk menghasilkan bilangan prima yang cukup besar. Akan tetapi, metode-metode ini dibatasi oleh syarat-syarat tertentu, seperti metode *Lucas-Lehmer test* yang hanya bisa mengecek bilangan prima yang merupakan bilangan Mersenne, ataupun *Miller-Rabin test* yang memiliki kemungkinan sangat kecil untuk menghasilkan bilangan bukan prima. Di saat yang bersamaan, ada pula *AKS Primality test* yang bisa menghasilkan bilangan prima tanpa syarat apapun, tetapi sayangnya *AKS Primality test* ini memiliki kompleksitas yang sangat tinggi, yaitu $O(n^6)$ yang jauh lebih tinggi ketimbang kompleksitas *Lucas-Lehmer test* yaitu $O(n^2 \log n)$ dan kompleksitas *Miller-Rabin test* yaitu $O(k\ log^3 n)$ dimana k adalah jumlah angka yang kita uji dengan metode tersebut dikarenakan *Miller-Rabin test* menguji bilangan secara acak, tidak secara deterministic seperti *Lucas-Lehmer test* dan *AKS Primality test*.

Metode-metode untuk menghasilkan bilangan prima yang disebutkan pada paragraf sebelumnya bisa dikelompokkan menjadi satu kategori yaitu *primality test*. Secara umum, metode-metode dalam kategori ini terspesialisasi untuk menghasilkan bilangan-bilangan prima yang sangat besar nilainya. Akan tetapi, untuk menghasilkan bilangan prima yang tidak terlalu besar dan dalam jumlah besar, biasanya digunakan metode yang lebih mudah yang dinamakan *Sieve of Erastothenes*. Metode ini jauh lebih efisien dan lebih mudah diimplementasikan dibandingkan *primality testing*. Sayangnya, algoritma metode ini menjadi tidak efisien bila kita ingin mencari bilangan prima yang cukup besar untuk bisa mengenkripsi data secara aman.

III. *METODE-METODE UNTUK MENCARI BILANGAN PRIMA*

A. *Lucas-Lehmer Test*

Lucas-Lehmer test adalah salah satu metode *primality testing* yang diciptakan Édouard Lucas pada tahun 1878 dan dikembangkan oleh Derrick Henry Lehmer pada tahun 1930-an. Metode ini termasuk metode yang deterministik, karena bilangan-bilangan yang akan diuji menggunakan metode ini tidak dipilih secara acak, melainkan dihasilkan menurut suatu algoritma tertentu.

Seperti yang telah disebutkan pada bab II, metode ini memiliki kompleksitas yang cukup rendah, yaitu $O(n^2 \log n)$ namun terbatas hanya untuk menguji bilangan prima yang merupakan bilangan Mersenne. Bilangan Mersenne adalah bilangan-bilangan yang memenuhi persamaan berikut:

$$M_n = 2^n - 1$$

dimana M_n adalah bilangan prima Mersenne dan n adalah suatu bilangan prima bukan 2. Setelah itu, untuk menguji apakah suatu bilangan M_n adalah prima untuk nilai n tertentu, bilangan tersebut harus memenuhi syarat lain, yaitu residu *Lucas-Lehmer*-nya harus bernilai 0, atau jika dituliskan dalam persamaan matematika adalah sebagai berikut:

$$s_{n-2} \equiv 0 \ (mod\ M_n)$$

dengan nilai s_{n-2} diperoleh dari fungsi berikut:

$$s_n = \begin{cases} 4 & n = 0; \\ (s_{n-1})^2 - 2, & n > 0. \end{cases}$$

Namun, pada praktiknya, fungsi di atas sedikit dimodifikasi menjadi:

$$s_n = \begin{cases} 4 & n = 0; \\ (s_{n-1})^2 - 2, & n > 0. \end{cases}$$

Berdasarkan algoritma di atas, metode ini memiliki kompleksitas $O(n^2 \log n)$ yang sekilas tidaklah terlalu besar

dan cukup efisien. Namun, kenyataannya, kompleksitas algoritma ini tumbuh dengan cukup cepat dikarenakan batasan bahwa bilangan-bilangan yang bisa diuji dengan metode ini hanya sebatas bilangan Mersenne saja. Dimana, nilai bilangan-bilangan Mersenne itu sendiri berkembang secara eksponensial ($O(2^n)$). Oleh karena itu, kalkulasi residu *Lucas-Lehmer* menjadi cukup sulit karena nilai s_n dan M_n yang berekembang dengan sangat cepat. Bahkan, dalam praktiknya, sejauh ini hanya ada 49 buah bilangan prima yang berhasil ditemukan dengan metode ini. Dan, dari 49 buah bilangan prima tersebut, hanya 34 buah bilangan prima pertama yang bisa dihasilkan oleh satu superkomputer. Sedangkan, 15 buah bilangan prima sisanya dihasilkan oleh hasil gabungan kemampuan komputasi setidaknya 1 juta komputer di seluruh dunia selama lebih dari 15 tahun.

$$2^{74,207,281} - 1$$

Bilangan prima Mersenne terbesar yang diketahui

B. *Miller-Rabin Test*

Miller-Rabin test adalah metode untuk mengecek bilangan prima yang termasuk kategori metode probabilistik. Metode ini lebih efisien bila dibandingkan dengan metode *Lucas-Lehmer* karena metode ini tidak memiliki batasan-batasan tertentu dan bisa digunakan untuk menguji apakah suatu bilangan prima atau tidak untuk semua bilangan bulat positif.

Cara kerja metode ini didasarkan pada karakteristik sekelompok bilangan yang dinamakan *strong pseudoprime*. Sebuah bilangan $n = d * 2^s + 1$ dikatakan adalah sebuah *strong pseudoprime* terhadap suatu basis a apabila bilangan tersebut memenuhi salah satu dari syarat berikut:

$$a^d \equiv 1 \, mod \, n$$

atau

$$a^{d*2^r} \equiv -1 \, mod \, n, \; 0 \leq r \leq s-1.$$

Kelompok bilangan ini didapat dari penurunan definisi kelompok bilangan lain yaitu *Fermat pseudoprime*.

Jadi, semua bilangan n yang memenuhi salah satu dari 2 persamaan di atas terhadap suatu basis a adalah sebuah bilangan *strong pseudoprime* terhadap a. Dan, untuk sebuah bilangan n prima, maka n akan lolos uji tersebut untuk semua basis a.

Sayangnya, metode ini juga memiliki kelemahannya sendiri, yaitu adanya kemungkinan bahwa bilangan yang lolos uji *strong pseudoprime* di atas adalah bilangan bukan prima dikarenakan sangat sulit untuk melakukan uji tersesbut terhadap semua nilai basis yang mungkin bila nilai n cukup besar. Tetapi, sejauh ini berhasil dipastikan bahwa metode ini dipastikan benar untuk semua nilai $n < 10^{16}$. Sayangnya, belum ditemukan satupun bukti bahwa ada bilangan bukan prima yang salah diidentifikasi oleh metode ini sebagai bilangan prima. Dan, sekalipun bilangan seperti itu ada, maka ekspektasi kemunculan bilangan tersebut sangatlah kecil, jauh lebih kecil dibandingkan kemungkinan kesalahan perangkat keras komputer yang menjalankan algoritma untuk metode ini.

Pada praktiknya, metode ini biasanya digunakan bersamaan dengan beberapa metode lain untuk meningkatkan efisiensinya. Salah satu metode pengecekan bilangan prima lain yang sering digunakan bersamaan dengan metode *Miller-Rabin* adalah *trial division*. Pada dasarnya, *trial division* menguji apakah suatu bilangan adalah prima atau bukan dengan membagi bilangan tersebut terhadap bilangan-bilangan lain yang telah diketahui adalah merupakan bilangan prima. *Trial division* biasanya dilakukan dengan menggunakan bilangan-bilangan prima kecil sebagai pembagi untuk mengurangi kemungkinan suatu bilangan yang akan diuji dengan metode *Miller-Rabin* adalah bilangan komposit.

Selain itu, implementasi metode ini di dunia nyata secara umum juga tidak menggunakan banyak basis melainkan hanya satu basis saja yang dipilih secara acak dimana

$$a \in \{1, \ldots, n-1\}$$

Secara umum, akurasi dari pengujian terhadap satu basis saja dianggap sudah cukup akurat karena bila diinginkan pengujian yang menyeluruh, akan dibutuhkan waktu yang sangat panjang dan metode ini akan menjadi kurang efisien untuk penggunaan oleh masyarakat umum. Pemilihan basis yang dilakukan secara acak inilah yang membuat metode ini dikategorikan sebagai metode probabilistic karena ada kemungkinan sangat kecil bahwa salah satu basis yang tidak dipilih bisa membagi habis bilangan yang dianggap prima.

C. *Sieve of Erastothenes*

Metode ketiga yang akan dibahas adalah *Sieve of Erastothenes* yang, berbeda dengan metode lain yang telah dibahas sejauh ini, diciptakan sekitar tahun 240 SM. Metode ini jauh lebih sederhana dan lebih mudah diimplementasikan disbanding metode-metode sebelumnya. Untuk bilangan prima yang tidak terlalu besar (lebih kecil dari 10,000,000), metode ini adalah salah satu metode yang paling efisien, baik secara waktu maupun memori yang dibutuhkan untuk mencari bialngan prima. Sayangnya, untuk bilangan-bilangan prima yang besar, metode ini menjadi tidak praktis.

Cara kerja metode ini sangatlah mudah. Pertama-tama, tentukan sebuah bilangan n sebagai batas atas bilangan-bilangan prima yang ingin kita cari. Lalu, buatlah daftar bilangan x yang memenuhi $1 < x \leq n$. Untuk mengecek bilangan mana sajakah yang prima, hapus semua bilangan yang merupakan kelipatan dari bilangan-bilangan prima yang lebih kecil dari $\sqrt{n}$. Berikut ini adalah contoh penggunaan *Sieve of Erastothenes* untuk menghasilkan semua bilangan prima ≤ 20.

Langkah 1:

Tentukan nilai n dan buat daftar bilangan yang lebih kecil atau sama dengan n selain 1.

Ambil $n = 20$,

Buat daftar bilangan P sebagai berikut:

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

Langkah 2:

Untuk setiap bilangan prima yang diketahui, hapus semua kelipatannya selain bilangan itu sendiri.

Hapus semua bilangan kelipatan 2 dari P:

2 3 5 7 9 11 13 15 17 19

Hapus semua bilangan kelipatan 3 dari P:

2 3 5 7 11 13 17 19

Karena $5^2 = 25 > n = 20$, maka tidak perlu dilakukan penghapusan bilangan kelipatan 5 dan semua bilangan yang tersisa adalah bilangan prima. (Untuk implementasi dari algoritma ini silahkan lihat apendiks A).

D. *Optimasi-optimasi untuk Meningkatkan Efisiensi Algoritma Pencarian Bilangan Prima*

Dalam aplikasi di dunia nyata, bahkan metode-metode yang disebutkan pada subbab-subbab sebelumnya masih secara umum dikategorikan kurang efisien karena metode-metode tersebut hanya bisa menghasilkna bilangan prima yang cukup besar untuk digunakan di bidang kriptografi apabila komputasi yang diperlukan dilakukan dengan menggunakan ratusan atau bahkan ribuan CPU yang bekerja secara bersamaan dalam jangka waktu yang tidak pendek. Oleh karena itu, dibutuhkan optimasi-optimasi agar metode-metode tersebut bisa diimplementasikan sedemikian rupa sehingga menghemat waktu dan biaya yang dibutuhkan untuk mendapat hasil.

Bila diamati, sebagian besar metode-metode di atas banyak menggunakan operasi perkalian dan modulus untuk menghitung apakah suatu bilangan prima atau tidak. Oleh karena itu, diciptakanlah berbagai algoritma yang bisa meningkatkan efisiensi dari operasi-operasi tersebut. Salah satu algoritma yang dikenal dan digunakan secara luas adalah algoritma Karatsuba yang merupakan optimisasi untuk operasi perkalian. Algoritma ini bisa mengurangi kompleksitas operasi perkalian dari $O(n^2)$ menjadi $O(n^{1.465})$. Selain itu, ada pula algoritma *Schönhage–Strassen* yang bahkan lebih efisien lagi dengan kompleksitas $O(n \log n \log n)$. Pengurangan kompleksitas tersebut sekilas memang nampaknya tidak terlalu berarti. Namun, untuk operasi-operasi pada pengujian bilangan prima yang melibatkan bilangan-bilangan yang terdiri atas ratusan digit, peningkatan efisiensi tersebut akan terlihat dengan jelas dan menjadi cukup berarti.

IV. APLIKASI BILANGAN PRIMA

A. Penjelasan Singkat Enkripsi

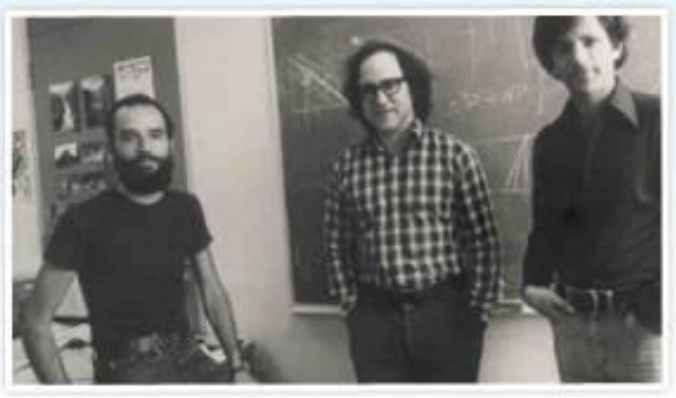
Dalam bidang informatika, enkripsi adalah proses dimana kalimat atau tipe data lainnya diubah dari bentuk yang bisa dibaca menjadi bentuk terenkripsi yang hanya bisa dibaca oleh pihak-pihak yang memiliki kunci dekripsinya. Enkripsi digunakan secara luas di internet utuk melindungi data pengguna yang dikirim browser kepada sebuah server, seperti nama, tempat tinggal, PIN kartu kredit, dan data-data lain yang tidak seharusnya dimiliki oleh pihak lain. Selain itu, enkripsi juga banyak digunakan oleh organisasi-organisasi besar seperti pemerintah dan perusahaan besar untuk mengamankan data-data sensitif yang disimpan dalam bentuk digital.

Secara umum, enkripsi dapat dikategorikan menjadi dua golongan, yaitu enkripsi simetrik dan enkripsi asimetrik. Pada enkripsi simetrik, kunci yang digunakan untuk enkripsi maupun dekripsi sama persis sehingga orang yang melakukan enkripsi juga bisa melakukan dekripsi dan sebaliknya. Sedangkan, pada enkripsi asimetrik kunci enkripsi dan dekripsi berbeda sehingga pihak yang bisa melakukan enkripsi belum tentu bisa melakukan dekripsi dan pihak yang bisa melakukan dekripsi belum tentu. bisa melakukan enkripsi.

Pada dasarnya, algoritma-algoritma enkripsi simetrik jauh lebih cepat dibandingkan enkripsi asimetrik dikarenakan kunci yang digunakan untuk enkripsi dan dekripsi sama persis. Tetapi, hal ini menimbulkan masalah keamanan karena pada saat kedua pihak mensepakati kunci yang akan digunakan, ada kemungkinan adanya pihak lain yang bisa menyadap dan mengetahui kunci tersebut sehingga pesan mereka bisa dibaca oleh pihak lain. Sebaliknya, enkripsi asimetrik jauh lebih aman dibandingkan enkripsi simetrik karena kedua belah pihak yang akan menggunakan enkripsi tidak perlu mensepakati kunci yang sama sehingga tidak ada kemungkinan adanya pihak lain yang menyadap dan mengetahui kunci tersebut. Sayangnya, enkripsi asimetrik yang jauh lebih aman ini juga jauh lebih lambat jika dibandingkan dengan enkripsi asimetrik dikarenakan kunci untuk enkripsi dan dekripsi berbeda sehingga komputer perlu melakukan perhitungan lebih untuk mengenkripsi maupun mendekripsi suatu pesan. Sehingga, pada praktiknya di dunia nyata, biasanya pesan atau data dienkripsi dengan menggunakan enkripsi simetrik dan kunci enkripsi simetrik tersebut dienkripsi dengan enkripsi asimetrik sebelum dikirimkan bersamaan dengan data yang dienkripsi. Dengan begitu, sebagian besar data dapat dienkripsi dan didekripsi dengan cepat menggunakan enkripsi simetrik sedangkan kunci yang enkripsi simetrik terjaga dengan aman karena terenkripsi dengan enkripsi asimetrik.

B. Enkripsi RSA

Enkripsi RSA pertama kali diciptakan pada tahun 1973 oleh GCHQ (*Government Communications Headquarters*), sebuah organisasi pemerintah Inggris. Akan tetapi, sistem ini baru diketahui public pada tahun 1977 ketika tiga orang peneliti di Amerika menciptakan sistem yang sama persis dan mematenkannya. Sistem enkripsi ini merupakan sistem enkripsi kunci publik (asimetrik) pertama. Pada sistem ini, ada dua buah kunci yang digunakan, yaitu kunci publik yang berfungsi untuk mengenkripsi data dan bisa diakses semua orang, dan kunci privat yang digunakan untuk dekripsi dan hanya dimiliki oleh pihak yang dituju. Pasangan kunci publik dan kunci privat ini unik untuk setiap orang, sehingga tidak akan ada dua buah kunciprivat maupun kunci public yang bernilai sama.



Sumber: https://rsaalgorithm.wordpress.com

Gambar 3. Tiga orang matematikawan yang menciptakan enkripsi RSA pada tahun 1977

Cara kerja sistem enkripsi RSA dapat dibagi menjadi dua tahap, yaitu tahap membuat kunci dan tahap kedua yaitu tahap mengenkripsi data. Deskripsi mengenai kedua tahap tersebut ada di dua subbab berikutnya.

C. Tahap Pembuatan Kunci RSA

Pada tahap ini, akan digunakan 2 buah bilangan prima yang sangat besar (setidaknya terdiri atas 512 digit) untuk menghasilkan kunci privat dan kunci public. Berikut ini langkah-langkah pembuatan kunci tersebut:

Langkah 1
Tentukan dua buah angka prima p dan q untuk membuat kunci privat dan public.

Langkah 2
Tentukan suatu nilai n yang akan digunakan sebagai modulus dengan persamaan berikut:

$$n = pq$$

Langkah 3
Kalkulasi *totient* ($\phi(n)$) dimana:

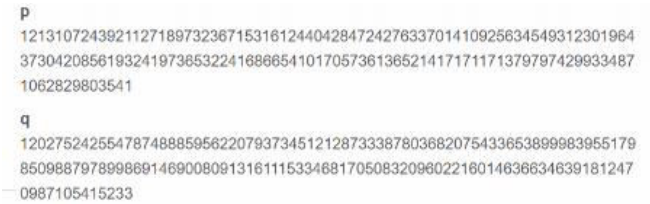
$$\phi(n) = (p-1) * (q-1)$$

Langkah 4
Tentukan nilai kunci publik e dimana $65537 \leq e \leq \phi(n)$ dan e harus koprima (tidak memiliki factor prima yang sama kecuali 1) terhadap $\phi(n)$.

(Nilai 65537 dipilih tidak secara acak. Dengan nilai tersebut, kunci privat akan cukup sulit untuk ditebak namun nilai kunci public tidak terlalu besar sehingga enkripsi data tidak memakan waktu terlalu lama).

Langkah 5
Lalu, terakhir tetntukan kunci privat d yang memenuhi:

$$e * d \equiv 1 \ (mod\ \phi(n))$$



Sumber: http://doctrina.org/How-RSA-Works-With-Examples.html

Gambar 4. Contoh pasangan bilangan prima p dan q (155 digit) yang bisa digunakan untuk menghasilkan pasangan kunci RSA

D. Tahap Enkripsi Data RSA

Dengan kunci publik dan privat yang diperoleh sesuai dengan algoritma pada subbab sebelumnya, digunakan suatu algoritma lain untuk mengenkripsi data yang akan dikirim m sebagai berikut:

$$F(m,e) = m^e mod\ n = c$$

dimana c adalah data setelah dienkripsi. Dan, untuk mendekripsi data tersebut di tujuan, digunakan fungsi yang serupa dengan menggunakan kunci privat, yaitu:

$$F(c,d) = c^d\ mod\ n = m$$

Meskipun terlihat sederhana, namun kedua fungsi matematis di atas hampir tidak mungkin ditembus hanya dengan mengetahui kunci publik e tanpa mengetahui bilangan prima p dan q yang digunakan untuk menciptakan kunci privat karena fungsi tersebut berupa aritmatika modulus yang sangat sulit untuk ditembus secara paksa.

V. KESIMPULAN

Seperti yang telah dijabarkan pada bab-bab sebelumnya, sangat dibutuhkan bilangan prima yang sangat besar untuk memastikan privasi semua orang yang menggunakan internet secara rutin setiap harinya. Selain itu, bilangan-bilangan prima tersebut juga dibutuhkan dalam jumlah besar dikarenakan setiap pasangan kunci privat dan publik haruslah unik untuk memastikan keamanan enkripsi. Sehingga, diperlukan pasangan bilangan prima yang unik pula sehingga tidak menghasilkan duplikat untuk kunci privat maupun kunci publik. Namun, seiring dengan meningkatnya kemampuan komputasi komputer yang tersedia kepada masyarakat umum, semakin membesar pula nilai bilangan prima yang dibutuhkan untuk memastikan data terenkripsi dengan baik dan tidak bisa dilihat oleh pihak yang tidak diinginkan. Oleh karena itu, sangatlah dibutuhkan metode-metode baru yang lebih efisien dari metode yang sudah ada untuk menghasilkan bilangan-bilangan prima yang besar. Selain itu, masalah ini juga bisa diatasi dengan diciptakannya algoritma-algoritma baru yang akan meningkatkan efisiensi algoritma-algoritma di atas, seperti algoritma Karatsuba yang meningkatkan efisiensi operasi perkalian.

Atau, ada juga jalan lain yang bisa ditempuh, yaitu dengan menciptakan skema pertukaran kunci baru ataupun algoritma enkripsi baru yang tidak terlalu bergantung pada bilangan prima yang hingga saat ini masih tergolong hal yang belum dipahami secara menyeluruh oleh matematikawan di seluruh dunia.

VI. APPENDIX

A. *Implementasi Sieve of Erastothenes*

```
def sieve(n):
  l = [2]
  l += list(range(3, n, 2))
  for i in range(1, len(l)):
    if l[i] != -1:
      for j in range(i + 1, len(l)):
        if (l[j] % l[i]) == 0:
          print(l[j], l[i])
          l[j] = -1

  for x in l:
    if x != -1:
      print(x)
```

Gambar 5. Implementasi *Sieve of Erastothenes* dalam bahasa Python

VII. UCAPAN TERIMA KASIH

Terima kasih penulis sampaikan kepada Tuhan Yang Maha Esa karena tanpa Nya makalah ini tidak akan bisa selesai tepat waktu. Selain itu, terima kasih juga penulis sampaikan pada orang tua yang telah membesarkan dan membimbing penulis hingga saat ini. Terima kasih banyak kepada Dra. Harlili M. Sc. yang telah mengajarkan materi matematika diskrit pada penulis sehingga penulis memilki pengetahuan yang dibutuhkan untuk menulis makalh ini. Terakhir, terima kasih penulis sampaikan pada pihak-pihak lain yng mendukung selesainya pengerjaan makalah ini.

REFERENSI

[1] S. Barry, http://doctrina.org/How-RSA-Works-With-Examples.html diakses pada tanggal 2 Desember 2017.
[2] http://mathworld.wolfram.com/RSAEncryption.html diakses pada tanggal 2 Desember 2017.
[3] http://blog.wolfram.com/2016/09/30/mersenne-primes-and-the-lucas-lehmer-test/ diakses pada tanggal 2 Desember 2017.
[4] https://www.mersenne.org diakses pada tanggal 2 Desember 2017.
[5] http://mathworld.wolfram.com/Rabin-MillerStrongPseudoprimeTest.html diakses pada tanggal 2 Desember 2017.
[6] http://mathworld.wolfram.com/StrongPseudoprime.html diakses pada tanggal 2 Desember 2017.
[7] S. Singh, *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*. New York, NY: Doubleday, 1999, pp. 273-305

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, 3 Desember 2017

Jonathan Alvaro - 13516023