Algoritma RSA dalam Pengenkripsian Data

Adventus Wijaya Lumbantobing, 13505112
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
if15112@students.ifitb.ac.id

Salah satu metode pengenkripsian data dalam kriptografi menggunakan kunci publik (*public key*) dan kunci pribadi (*private key*). Proses pengenkripsian data sebelumnya dilakukan dengan sistem *chiper* menggunakan perjanjian antar kedua pihak untuk melakukan enkripsi dan dekripsi pesan. Sistem kunci publik digunakan dalam menangani proses enkripsi data dan kunci pribadi digunakan dalam proses dekripsi pesan,

Tingkat keamanan sistem tergantung pada kerahasiaan dan kesulitan publik untuk mengetahui kunci pribadi tersebut. Penggunaan *kunci publik* dalam kriptografi pertama kali menggunakan teorema yang sangat kompleks dan sangat sulit untuk dipecahkan. Metode yang lebih praktis diajukan oleh Ron Rivest, Adi Shamir, dan Leonard Adleman (RSA) dari *Massachusetts Institut of Technology (MIT)* pada tahun 1977. Tulisan ini membahas penggunaan algoritma RSA dalam kriptografi

Kata kunci: kriptografi, kunci public, kunci pribadi, algoritma RSA.

I. PENDAHULUAN

Metode penggunaan kunci publik seperti ini pertama kali ditemukan pada tahun 1973 oleh Clifford Cocks. RSA merupakan sistem enkripsi dan autentifukasi yang merupakan implementasi dari algoritma yang dikembangkan oleh Ron Rivest, Adi Shamir, dan Leonard Adleman dari Massachusetts Institut of Technology (MIT) pada tahun 1977. Nama RSA sendiri diambil dari inisial nama ketiga orang tersebut.

Algoritma bekerja dengan pemilihan dua bilangan prima dan hasil kali dari kedua bilangan tersebut. Hasil kali dari bilangan-bilangan prima tersebut digunakan dalam proses enkripsi dan juga dalam proses dekripsi yang memerlukan pengolahan dalam bilangan prima. Untuk memecahkan kode ini harus diketahui hasil faktorisasi dari hasil kali bilangan prima yang sebelumnya. Dan hal ini sangat sulit jika bilangan tersebut merupakan bilangan yang besar dan bahkan memerlukan waktu yang sangat lama.

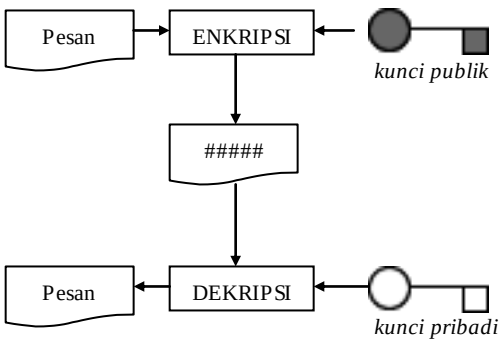
II. ALGORITMA RSA

RSA merupakan metode dalam kriptografi yang menggunakan kunci publik. RSA terdiri dari dua bagian utama yaitu :

1. *kunci publik* (untuk enkripsi)
2. *kunci pribadi* (untuk dekripsi).

Bagian *kunci publik* dapat dipublikasikan ke umum sehingga publik dapat melakukan enkripsi data sedangkan bagian *kunci pribadi* tidak dipublikasikan dan tetap dijaga kerahasiaannya untuk melakukan dekripsi data. Pesan yang dienkripsi dengan *kunci publik* tertentu hanya dapat didekripsi dengan *kunci pribadi* yang bersesuaian.

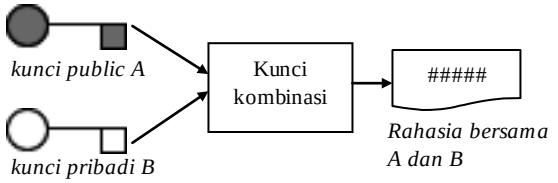
Secara garis besar, algoritma RSA melibatkan perkalian dua bilangan prima yang besar dan dengan tambahan operasi matematika lainnya menghasilkan kedua buah set kunci seperti di atas. Jika kunci sudah disebarkan bilangan prima yang pertama tidak lagi penting dan dapat diabaikan. Pembuatan kunci dilakukan dengan pemilihan bilangan acak yang besar lalu dibuat *kunci publik* (dipublikasikan untuk umum) dan *kunci pribadi* (dirahasiakan).



Gambar 1 Proses enkripsi dan dekripsidengan kunci publik dan kunci pribadi

Setelah publik mendapatkan *kunci publik* yang telah dipublikasikan maka pihak tersebut dapat melakukan enkripsi pesan kepada tujuannya. Penerima pesan akan mendekripsi pesan tersebut dengan *kunci pribadi* miliknya.

Dengan menggunakan *kunci pribadi* untuk mengenkripsi *signature* maka siapa saja dapat melakukan dekripsi dengan *kunci publik*. Dengan demikian publik dapat mengetahui siapa pengirim pesan yang ia terima. Validasi pesan tergantung pada sistem keamanan dalam *kunci pribadi*.



Gambar 2 Kombinasi kunci

Dengan menggabungkan *kunci pribadi* dengan *kunci publik* dari orang lain maka kedua pihak dapat membagi pesan rahasia yang hanya diketahui oleh kedua pihak. Proses enkripsi dan dekripsi dalam penyampaian pesan seperti di atas di gambarkan dalam tabel berikut :

Aksi	Pemilik	Kunci
Kirim pesan terenkripsi	Penerima	Kunci publik
Kirim *signature* terenkripsi	Pengirim	Kunci pribadi
Dekripsi pesan	Penerima	Kunci pribadi
Dekripsi dan enkripsi *signature*	Pengirim	Kunci publik

Berikut merupakan langkah-langkah pembuatan *kunci publik* dan *kunci pribadi* :

1. Pemilihan dua bilangan prima secara acak misalnya bilangan p dan q dan $p \neq q$.
2. Hitung $n = p\,q$. Besaran n tidak dirahasiakan.
3. Hitung $\varphi(n) = (p\text{-}1)\,(q\text{-}1)$.
4. Pilih bilangan bulat e (*integer*) antara satu dan $\varphi(n)$ ($1 < e < \varphi(n)$) yang juga merupakan relatif prima terhadap $\varphi(n)$ (e dan $\varphi(n)$ tidak memiliki factor lain selain 1). Bilangan e merupakan eksponen dari kunci publik.
5. Hitung d hingga $d\,e \equiv 1 \pmod{\varphi(n)}$. d merupakan eksponen dari kunci pribadi. Atau dengan cara lain $d\,e = 1 + \mathrm{k}\,\varphi(n)$ untuk k bilangan bulat.

Bilangan prima dapat diuji probabilitasnya menggunakan teorema *Fermat's little :*

$$a^{(n-1)} \bmod n = 1$$

Jika n adalah bilangan prima dan dilakukan pengujian dengan beberapa nilai a, kemungkinan besar n adalah bilangan prima. Terkadang hasil dari persamaan ini memang benar untuk nilai a meskipun n bukanla bilangan prima. Ada cara lain yaitu dengan *Carnichael numbers* yang benar untuk semua nilai a kecuali untuk nilai b yang merupakan faktor dari n.

Berdasarkan teorema *Fermat's Little* dan *Camichael numbers* dikembangkan pengetesan bilangan prima yaitu *Solovay-Strassen probabilistic primality test* :

1. Pertama pemilihan nilai a dengan nilai antara 1 sampai n-1. Tes ini untuk melihat apakah b relatif prima terhadap n, jika tidak maka sudah jelas n bukan bilangan prima.
2. Sebuah fungsi untuk b dan n dilambangkan $J(b,n)$ disebut simbol Jacobi. Persamaannya adalah sebagai berikut:

$$J(b,n) = b^{((n-1)/2)}$$

3. Nilai $J(b,n)$ selalu bernilai 1 atau -1.
4. Jika b adalah bilangan genap dan n bilangan ganjil, maka

$$J(b,n) = J(b/2,n)((-1)^{(n \times n-1)/8)})$$

Jika sebaliknya, maka persamaannya menjadi:

$$J(b,n) = J(n \bmod b/2,b)((-1)^{((b-1) \times (n-1)/4)})$$

Pemilihan bilangan e yang umum adalah :

$$e = 2^{16} + 1 = 65537.$$

Beberapa aplikasi menggunakan nilai e yang kecil misalnya 3, 5, 35 dengan tujuan agar proses enkripsi dapat berlangsung lebih cepat namun hal ini tentu mengandung risiko yang lebih besar.

Kunci publik terdiri atas :

1. modulus n
2. bilangan eksponen publik e (sering juga disebut eksponen enkripsi).

Kunci pribadi terdiri atas:

1. modulus n
2. bilangan eksponen d (sering juga disebut eksponen dekripsi), yang harus dijaga kerahasiaannya.

Agar lebih efisien bentuk dari *kunci pribadi* dapat dituliskan:

1. p dan q, bilangan prima.
2. *d mod (p-1)* dan *d mod (q-1)* (ditulis dengan *dmp1* dan *dmq1*).
3. *(1/q) mod p* (ditulis dengan *iqmp*).

Dalam bentuk seperti ini proses dekripsi dan signing lebih cepat namun kurang aman karena adanya *side channel attacks*. Jika $y = x^e \bmod n$ dan sistem akan mendekripsikan persamaan tersebut. Sistem akan mengomputasi $y^d \pmod p$ dan $y^d \pmod q$ yang menghasilkan nilai z. Maka nilai $\gcd(z\text{-}x,n)$ akan menghasilkan nilai p dan q. Oleh karena itu semua bagian dari *kunci pribadi* harus dijaga kerahasiaannya.

Nilai p dan q sangat sensitif karena kedua bilangan tersebut merupakan faktorial dari n, dan mengakibatkan perhitungan dari d dapat menghasilkan e. Jika p dan q tidak disimpan dalam bentuk CRT dari *kunci pribadi*, maka p dan q telah terhapus bersama nilai-nilai lain dari proses pembangkitan kunci.

III. PROSES ENKRIPSI DAN DEKRIPSI

1. Proses Enkripsi Pesan

Pertama-tama pengirim mempublikasikan *kunci publik* (*q* dan *e*) kepada publik sedangkan *kunci pribadi* tetap dijaga rahasianya.

Misalkan publik akan megirim pesan *p*, maka *p* diubah terlebih dahulu menjadi angka *q* (*q* < *n*) dengan menggunakan protokol yang telah disepakati sebelumnya (dikenal dengan *padding scheme*).

Dengan demikian, publik akan memiliki *q* dan mengetahui nilai *n* dan *e*. Lalu publik akan menghitung *chipertext c* yang bersesuaian dengan *q* dengan persamaan :

$$c = q^e \bmod n$$

Perhitungan tersebut dapat diselesaikan dengan cepat menggunakan metode *exponentiation by squaring*. Kemudian publik mengirim *c* kepada pengirim.

2. Proses Dekripsi Pesan

Pengirim akan mengembalikan nilai *q* dari *c* dengan menggunakan *kunci pribadi d* dengan prosedur sebagai berikut :

$$q = c^d \bmod n$$

Langkah-langkah dekripsi tersebut sebagai berikut :

$$c \equiv (q^e)^d \equiv q^{ed} \pmod n$$

Karena

$$ed \equiv 1 \pmod{a - 1} \text{ dan } ed \equiv 1 \pmod{b - 1}$$

menurut teorema Fermat's little :

$$q^{ed} \equiv q \pmod a \text{ dan } q^{ed} \equiv q \pmod b.$$

Karena *a* dan *b* bilangan prima, maka dengan menerapkan teorema *Chinese Remainder* pada kedua persamaan ini diperoleh :

$$q^{ed} \equiv q \pmod{ab}$$

Dengan demikian,

$$c^d \equiv q \pmod n$$

atau

$$q = c^d \bmod n$$

Jika nilai q diperoleh maka pengirim dapat megubahnya menjadi pesan yang sebenarnya yaitu *p*.

IV. IMPLEMENTASI

RSA dapat diimplementasikan secara hardware dan software, dimana standar implementasi menggunakan PKCS#1. *Kunci publik e* harus terlindungi secara baik sehingga pemilihan 3,17 dan 65537 sering digunakan. Rekomendasi PKCS#1 : 3 (011_2),17 (10001_2) atau 65537 (1000000000000001_2) karena ketiga bilangan tersebut memiliki bit 1 sebanyak 2 buah, sehingga diperlukan hanya 17 perkalian untuk operasi eksponensiasi, akibatnya operasi dekripsi menjadi lebih cepat.

Metoda penghitungan biner untuk $q^e \bmod n$, integer *q*,*e* dan *n* menggunakan algoritma modular exponential, yaitu eksponensial yang disusun dari operasi perkalian modular dan pangkat 2 modular. Berikut adalah algoritma tersebut :

1. *a* :=1;
2. *b* :=*q*;
3. for *i*=0 to *h*-2
    a. if *ei*=1 then *a*:=*a.b* mod *n*
    b. *b* :=*b.b* mod *n*
4. if *eh*-1=1 then *a*:=*a.b* mod *n*
5. return a

Tiap bit dari *e* dis*can* dari bit 0 LSB hingga h-1 MSB, hal ini dilakukan dalam loop, jika $e_i = 1$ maka dilakukan perkalian modular *a.b* mod *n* dan kemudian diteruskan dengan *b.b mod n*. Algoritma ini membuat algoritma model *Fermat F4 ($2^{16}+1$)* atau 65537, melakukan 17 perkalian , yaitu 15 kali menghitung *b.b mod n* dan 1 kali *a.b mod n* dalam loop 3 dan kemudian 1 kali *a.b mod n* pada langkah 4.

Padding Schemes

Padding scheme adalah protocol yang telah disepakati bersama untukmengubah data atau pesan ke dalam bentuk numerik.

Padding Scheme harus dibuat dengan hati-hati untuk mencegah timbulnya masalah dalam pesan yang telah diubah dalam angka (nilai *q*).

RSA yang tidak menggunakan *padding scheme* akan mengalami beberapa permasalahan misalnya :

1. Jika kita ambil contoh sederhana dari penampilan ASCII dari *p* dan menggabungkan bit-bit secara bersamaan akan menghasilkan *n*, kemudian pesan yang berisi ASCII tunggal karakter NUL (nilai numeris 0) akan menghasilkan *q* = 0, yang akan menghasilkan *ciphertext* 0 untuk setiap nilai *e* dan *n* yang digunakan.
   Sama halnya dengan karakter ASCII tunggal SOH (nilai numeris 1) akan selalu menghasilkan *chiphertext* 1.

2. Untuk sistem yang menggunakan nilai *e* yang kecil (misalnya 3) seluruh karakter tunggal ASCII pada pesan akan disandikan menggunakan skema yang tidak aman. Hal ini disebabkan nilai terbesar *q* adalah nilai 255, dan 255^3 menghasilkan nilai yang lebih kecil dari modulus yang seharusnya.

Maka dengan pola dasar dari *ciphertext* dalam kondosi tersebut proses dekripsi akan menjadi lebih sederhana tanpa perlu menggunakan modulus *n*.

Validasi Pesan

Misalkan A menggunakan *kunci publik* milik B untuk mengirim pesan kepada B. Dalam pesan tersebut A dapat mengidentitaskan dirinya sebagai A namun permasalahannya B tidak dapat mengidentifikasi bahwa pesan tersebut berasal dari A. Hal ini dikarenakan publik memiliki *kunci publik* B dan siapa saja bisa mengirimkan pesan yang telah dienkripsi pada B. Jadi, untuk menjaga orisinalitas pesan (identitas pengirim) RSA juga dapat diterapkan untuk menvalidasi pesan yang diterima.

Metode yang digunakan sama seperti halnya pada enkripsi dan dekripsi pesan. Untuk menyertakan identitas maka pengirim pesan menggunakan metode dekripsi. Sebaliknya untuk mengidentifikasi pesan yang dikirim penerima pesan menggunakan metode dekripsi untuk menerjemahkan pengirim pesan tersebut.

Berdasarkan ilustrasi di atas A membuat sebuah *hash value* dari pesan tersebut. Bilangan tersebut dipangkatkan dengan bilangan *d* dimodulus *n* (seperti halnya pada deskripsi pesan) dan melampirkannya sebagai identitas pengirim pada pesan tersebut.

$$s = h(k)^d \pmod{n}$$

Saat B menerima pesan yang telah memiliki identitas pengirim (*signature*), B memangkatkan *signature* tersebut dengan bilangan *e* dimodulus dengan *n* (seperti halnya pada enkripsi pesan) dan membandingkannya dengan nilai hasil dari *hash value* dengan *hash value* pada pesan tersebut.

$$h(k) = s^e \pmod{n}$$

Jika kedua cocok *hash value*, maka B dapat mengetahui bahwa pemilik dari pesan tersebut adalah A. Pesan tidak mengalai perubahan sepanjang pengiriman.

Padding scheme merupakan hal yang esensial untuk mengamankan pengesahan pesan seperti halnya pada enkripsi pesan, oleh karena itu kunci yang sama tidak digunakan pada proses enkripsi dan pengesahan.

IV. KEAMANAAN

Pengamanan dengan metode RSA didasarkan pada dua permasalahan matematika seacra umum yaitu :

1. Penanganan masalah faktorisasi pada bilangan yang sangat besar.
2. Permasalahan RSA. Yang dimaksud permasalahan RSA adalah penghitungan nilai *n* yang sesuai dari persamaan $q^e = c \bmod n$, dengan (*e*, *n*) adalah *kunci publik* RSA dan *c ciphertext* dari RSA. Saat ini solusi yang paling baik untuk memecahkan algoritma RSA adalah dengan memfaktorkan modulus *n*. Dengan kemampuan faktorisasi bilangan prima maka ada kemungkinan untuk melakukan menghitung bilangan eksponen *d* dari kunci publik (*e*, *n*), lalu mendekripsi *c* dengan menggunakan prosedur yang dipaparkan di atas. Untuk melakukan ini harus dilakukan faktorisasi bilangan prima *n* menjadi *p* dan *q*, lalu menghitung (*p*-1)(*q*-1) yang mengakibatkan munculnya bilangan *d* dari *e*.

Penemu algoritma RSA menyarankan nilai *p* dan *q* panjangnya lebih dari 100 digit. Dengan demikian hasil kali n = *p* x *q* akan berukuran lebih dari 200 digit. Bayangkan betapa besar usaha yang diperlukan untuk memfaktorkan bilangan 200 digit menjadi faktor primanya. Menurut Rivest dan kawan-kawan usaha mencari bilangan 200 digit membutuhkan komputasi selama 4 milyar tahun (dengan asumsi bahwa algoritma pemfaktoran yang digunakan adalah algoritma yang tercepat saat ini dan komputer yang dipakai mempunyai kecepatan 1milidetik).

Berdasarkan ilustrasi sebelumnya meskipun ada pihak tertentu yang mendapatkan *kunci publik n* dan *e*, dan *ciphertext c*. Bagaimanapun juga, pihak tesebut tidak mampu untuk secara langsung memperoleh *d* yang dijaga kerahasiannya. Cara paling efektif yang untuk memperoleh *n* dari *c* ialah dengan melakukan faktorisasi *n* kedalam *p* dan *q*, dengan tujuan untuk menghitung (*p*-1)(*q*-1) yang dapat menghasilkan *d* dari *e*.

Masih belum ada bukti pula bahwa melakukan faktorisasi *n* adalah satu-satunya cara untuk memperoleh *n* dari *c*, tetapi sampai saat ini belum ditemukan adanya metode yang lebih mudah.

Sampai saat ini belum ada metode yang efisien dan efektif untuk memfaktorkan bilangan-bilangan bulat yang besar dengan teknologi komputer saat ini, walapun tidak dapat dipastikan bahwa metode tersebut benar-benar tidak ada.

Pada tahun 2005, bilangan faktorisasi terbesar yang digunakan adalah sepanjang 663 bit, dengan menggunakan metode distribusi mutakhir. Kunci RSA pada umumnya sepanjang 1024-2048 bit. Walaupun demikia diyakini bahwa kunci 1024-bit akan dapat dipecahkan di masa depan dan sampai saat makalah ini dituliskan hal ini masih dikembangkan.

Jika *n* sepanjang 256-bit atau lebih pendek, *n* akan dapat difaktorisasi dalam beberapa jam pada *Personal Komputer*, dengan menggunakan perangkat lunak yang tersedia. Jika *n* sepanjang 512-bit atau lebih pendek, *n*

akan dapat difaktorisasi dalam hitungan ratusan jam seperti pada tahun 1999.

Secara teori, perangkat keras bernama TWIRL dan penjelasan dari Shamir dan Tromer pada tahun 2003 mengundang berbagai pertanyaan akan keamanan dari kunci 1024-bit. Oleh karena itu, disarankan bahwa *n* setidaknya sepanjang 2048-bit.

Pada tahun 1993, Peter Shor menerbitkan Algoritma Shor, menunjukkan bahwa sebuah komputer quantum secara prinsip dapat melakukan faktorisasi dalam waktu polinomial, mengurai RSA dan algoritma lainnya. Bagaimanapun juga, masih terdapat pedebatan dalam pembangunan komputer quantum secara prinsip.

V. KESIMPULAN

RSA merupakan sistem pengamanan yang menggunakan metode kunci publik. Komponen utama RSA terdiri atas *kunci publik* yang sifatnya bebas dan digunakan dalam prose enkripsi dan *kunci pribadi* yang sifatnya rahasia yang digunakan dalam proses dekripsi.

Saat ini algoritma RSA masih banyak digunakan dalam sistem pengamanan dalam dunia komputer. Metode RSA masih bertahan dan masih teruji keamanannya karena algoritmanya yang sulit dipecahkan dengan teknologi komputer saat ini. Namun hal ini tidak menutup kemingkinan algoritma ini dapat di pecahkan di masa mendatang. Belum ada teknik pembobolan lain yang lebih efektif daripada *brute force* attack, jadi untuk ukuran kunci yang panjang, sistem penyandian dengan RSA masih sangat baik dan sulit untuk dibobol

DAFTAR PUSTAKA

[1] Munir, Rinaldi. 2006. *Diktat Kuliah IF2153 Matematika Diskrit*. Program studi Teknik Informatika, Institut Teknologi Bandung. Bandung.

[2] Nursanto, Djoko. 2003. *Meningkatkan Kecepatan Dekripsi RSA Menggunakan Integrasi Metode Montgomery Multification Chinese Remainder Theorem*. Program studi Teknik Informatika, Institut Teknologi Bandung. Bandung.

[3] Rahardjo, Budi. 2002. *Keamanan Sistem Informasi Berbasis Internet*. PT Insan Infonesia. Bandung .

[4] Rivest, R.L. ; Shamir, A. ;and Adleman, L.. 1977. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. ACM

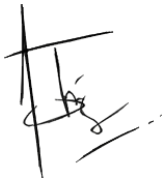
[5] http://en.wikipedia.org/wiki/RSA. Tanggal akses 15 Desember 2010 pukul 14.00 WIB.

[6] http://www.quadibloc.com/crypto/pk0502. htm. Tanggal akses 15 Desember 2010 pukul 14.20 WIB.

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, 15 Desember 2010



Adventus Wijaya Lumbantobing
13505112