STUDI TEKNIS METODE ENKRIPSI RSA DALAM PERHITUNGANNYA

Muhammad Iqbal – 13505060

Program Studi Teknik Informatika
Institut Teknologi Bandung
Jln Ganesha 10, Bandung, Indonesia
Email: if15060@students.if.itb.ac.id

Abstrak

Makalah ini menjelaskan tentang kekuatan, kelemahan, serta perhitungan yang terjadi dalam metode enkripsi RSA.

RSA adalah salah satu metode enkripsi yang banyak dipakai saat ini. RSA ditemukan oleh Ron Rivest, Adi Shamir, dan Leonard Adleman pada tahun 1977. RSA banyak digunakan dalam berbagai macam produik teknologi, salah satunya adalah *browser* Microsoft Internet Explorer dan Netscape Navigator.

RSA memiliki kelebihan dan sukar untuk dibobol. Di samping itu pula, RSA memiliki berbagai kekurangan. Sebagai contoh, RSA membutuhkan waktu dekripsi yang lebih lama dibandingkan dengan metode lain.

Sampai saat ini satu-satunya cara untuk mendobraknya adalah dengan cara mencoba satu persatu kombinasi kunci yang mungkin atau yang biasa disebut *brute force attack*. Sehingga penentuan tingkat keamanan suatu sandi dari kemungkinan dibongkar adalah seberapa panjang dari sandi (ukuran kunci) terebut. Karena jika semakin panjang suatu kode, maka semakin banyak pula kombinasi kunci yang mungkin ada.

Sebagai contoh, Standar industri saat ini bahkan menggunakan Triple DES yang ukuran kuncinya 112 *bit*. Ini membuat usaha untuk mendobrak sandi ini dengan *brute force* menjadi 10^{16} kali lebih sulit. Untuk RSA, panjang kuncinya bisa diatur. Misalnya ukuran kunci RSA yang digunakan oleh modul sekuriti browser Netscape Anda ukurannya 48 *bit*. Ukuran ini sudah tidak aman lagi sekarang, tetapi pemerintah AS melarang ekspor produk-produk RSA yang menggunakan kunci lebih besar dari 48 *bit*.

1. Pendahuluan

Jaringan sebagai sarana umum sangat rawan terhadap pencurian, penyadapan, dan pemalsuan informasi. Maka dari itulah dibutuhkan suatu pengaman dalam pengiriman suatu paket data, apa lagi untuk data-data atau pesan-pesan yang bersifat sangat rahasia. Untuk itu salah satu cara untuk mengamankan data dari kejadian-kejadian tersbut, diperlukan penyandian terhadap data yang akan dikirim. Penyandian ini sangat penting, apalagi dalam sektor-sektor strategis seperti bisnis, perbankan, atau pemerintahan sangat memerlukan teknologi penyandian Informasi. Penyandian ini sangat penting, apalagi dalam sektor-sektor strategis seperti bisnis, perbankan, atau pemerintahan sangat memerlukan teknologi penyandian Informasi.

Ilmu sandi (kriptografi) sendiri telah ada sejak lama. Tercatat dalam sejarah bahwa Julius Ceasar (kaisar romawi) menggunakan penyandian untuk menyampaikan pesan rahasia saat perang. Ilmu ini tidak hanya mencakup teknik-teknik menyandikan informasi, tetapi juga teknik untuk membongkar sandi.

Enkripsi adalah suatu proses mengubah sebuah teks murni (*plaintext*) menjadi sebuah runtutan karakter atau data yang terlihat tidak berarti dan mempunyai urutan bit yang tidak beraturan, disebut *ciphertext.* Proses pengubahan kembali *ciphertext* menjadi *plaintext* disebut dekripsi.

Terdapat banyak algoritma penyandian di dunia ini, yang paling banyak dipakai di dunia adalah DES dan RSA. Di samping DES dan RSA, masih ada banyak sandi lain seperti MD2 (dipakai GSM), IDEA, RC2, dll. Akan tetapi, DES dan RSA adalah yang paling populer dan paling banyak dipakai. DES (*Data Encryption Standard)* adalah hasil inovasi IBM di tahun 1972 yang kemudian diangkat menjadi standar oleh dewan standar AS (ANSI).

RSA sendiri dibuat pada tahun 1977. RSA adalah singkatan dari nama para penemunya, yaitu Ron Rivest, Adi Shamir, dan Leonard Adleman. RSA adalah salah satu algoritma penyandian yang paling banyak mengundang kontroversi, selain DES. Sejauh ini belum seorang pun yang berhasil menemukan lubang sekuriti pada DES dan RSA, tetapi tak seorang pun juga yang berhasil memberikan pembuktian ilmiah yang memuaskan dari keamanan kedua teknik sandi ini.

2. Sejarah RSA

Algoritma RSA diperkenalkan oleh 3 peneliti dari MIT (*Massachussets Institute of Technology*), yaitu Ron Rivest, Adi Shamir, dan Leonard Adleman pada tahun 1977. RSA adalah kependekan dari Rivest Shamir Adleman yang diambil dari 3 nama penemu RSA.

MIT diberikan paten AS nomor 4405829 untuk "*Cryptographic communications system and method*" (Sistem dan metode komunikasi kriptografi) yang menggunakan algoritma RSA pada tahun 1983. Paten ini akan kadaluwarsa pada tanggal 21 September 2000. Karena sebuah makalah yang menjelaskan tentang algoritma RSA telah dipublikasikan pada bulan Agustus 1977, aplikasi paten yang telah disimpan pada Desember 1977 tidak mendapat paten dari belahan dunia manapun selain di AS karena peraturan yang terkait dan regulasi yang berlaku.

3. Kekuatan RSA

3.1 Kecepatan RSA

Suatu “Operasi RSA,” untuk enkripsi, dekripsi, tanda tangan elektronik, atau pengecekan, adalah sebuah pemangkatan modular, yang dapat dilakukan dengan berkali-kali melakukan perkalian modular.

Pada aplikasi praktiknya, sudah seharusnya menentukan sebuah eksponen publik sebagai kunci publik; Nyatanya, sebuah kelompok *user* dapat menggunakan kunci publik yang sama dengan masing-masing kunci memiliki modulonya sendiri-sendiri (Ada beberapa larangan pada faktor prima dari modulo

ketika eksponen publik ditetapkan). Hal ini membuat proses enkripsi lebih cepat daripada proses dekripsi dan proses verifikasi lebih cepat daripada Pendandatanganan. Dengan algoritma eksponensial modular yang sama, operasi kunci publik membutuhkan O(k2) langkah, operasi kunci privat membutuhkan O(k3) langkah, dan penggenerasian kunci membutuhkan O(k4) langkah, di mana k adalah jumlah bit dalam modulo (notasi O berlaku sebagai batas atas pada kecepatan bekerja asimtotik dari algoritma). Teknik "perkalian cepat", seperti metode berbasis FFT, memerlukan langkah-langkah yang lebih sedikit, walaupun pada praktiknya tidak demikian karena kekompleksan perangkat lunak dan karena ukuran bit dari sebagian kunci.

Ada banyak sekali perngkat lunak maupun perangkat keras yang megimplementasikan algoritma enkripsi RSA, dan sering kali diberitakan tentang chip yang lebih baru dan lebih cepat. Pada *Pentium* berkecepatan 90MHz, perangkat kriptografi BSAFE 3.0 dari pengaman data RSA mempunyai kecepatan untuk mengolah kunci privat sebesar 21.6Kbit per detik untuk modulo 512 bit dan 7.4 Kbit per detik untuk modulo 1024 bit. Perngkat keras dari RSA yang mempunyai kecepatan tercepat memiliki kecepatan sebesar 300 Kbit per detik untuk modulo 512 bit, yang berarti melakukan lebih dari 500 operasi kunci privat RSA setiap detiknya. Ada ruangan dalam perangkat keras tersebut untuk melakukan dua buah operasi RSA 512 bit secara paralel, kecepatannya bisa mencapai 600 Kbits per detik. Untuk kunci 970 bit, kecepatannya 185Kbit per detik. Diperkirakan kecepatan ini akan mecapai 1Mbit per detik dalam waktu kurang dari satu tahun.

Sebagai perbandingan, DES sangat jauh lebih cepat dibandingkan RSA. Dalam perangkat lunak, DES bekerja rata-rata 100 kali lebih cepat dibandingkan RSA. Pada perngkat keras, DES bekerja antara 1.000 dan 10.000 kali lebih cepat dibandingkan RSA tergantung dari implementasinya. Implementasi dari RSA mungkin akan memperkecil selisih kecepatan dalam beberapa tahun mendatang seiring berkembangnya pasar, tapi DES juga pasti akan berkembang juga.

3.2 Keamanan RSA

Jarang orang yang mengetahui bahwa penyerangan terhadap RSA adalah memfaktor pq mnjadi p dan q. Sayangnya tidak seorangpun punya ide untuk membuktikan bagaimana faktorisasi—atau masalah realistis apapun—pada dasarnya berjalan dengan lambat. Cukup mudah bagi kita untuk menyimpulkan apa yang biasa kita sebut dengan "RSA kuat/tidak kuat"; tapi seperti apa yang dikatakan Hendrik W. Lenstra, Jr., "*Exact definitions appear to be necessary only when one wishes to prove that algorithms with certain properties do_not_exist, and theoretical computer science is notoriously lacking in such negative results.*"

Perhatikan bahwa ada 'jalan pintas' untuk membolol RSA selain memfaktorkan. Keamanan dari sistem bergantung pada pada dua buah asumsi kritikal: (1) pemfaktoran diperlukan untuk mengancurkan sistemnya, dan (2) pemfaktoran pada dasarnya 'sulit untuk dimanipulasi', atau 'pemfaktoran itu sulit' dan 'percobaan apapun yang dilakukan untuk membobol sistemnya minimal sama sulitnya dengan pemfaktoran'.

Sudah sering kali terjadi bahwa seseorang, walaupun seseorang itu adalah kriptografer profesional, membuat kesalahan dalam memperkirakan dan mengandalkan berbagai kesalahan komputasi yang sulit dimanipulasi untuk properti kriptografi yang aman. Sebagai contoh, sebuah sisitem yang dikenal sebagai '*Knapsack cipher*' berada dalam kepopuleran dibidang literatur selama bertahun-tahun sampai akhirnya didemonstrasikan bahwa bagian-bagian yang dibangkitkan secara serupa akan sangat mudah dibobol. Kemudian seluruh area riset seperti kehilangan gairah.

Pada tahun 2005, bilangan faktorisasi terbesar yang digunakan secara umum ialah sepanjang 663 bit, menggunakan metode distribusi mutakhir. Kunci RSA pada umumnya sepanjang 1024—2048 bit. Beberapa pakar meyakini bahwa kunci 1024-bit ada

kemungkinan dipecahkan pada waktu dekat (hal ini masih dalam perdebatan), tetapi tidak ada seorangpun yang berpendapat kunci 2048-bit akan pecah pada masa depan yang terprediksi. Jika *N* sepanjang 256-bit atau lebih pendek, maka kunci RSA akan dapat ditemukan dalam beberapa jam hanya dengan menggunakan PC, 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 dengan menggunakan ratusan komputer. Secara teori, perangkat keras bernama TWIRL dan penjelasan dari Shamir dan Tromer pada tahun 2003 mengundang berbagai pertanyaan akan keamanan dari kunci 1024-bit. Saat ini disarankan bahwa *N* setidaknya sepanjang 2048-bit.

Pada tahun 1993, Peter Shor, yang menerbitkan Algoritma Shor, menunjukkan bahwa sebuah komputer quantum secara prinsip dapat melakukan faktorisasi dalam waktu polinomial, mengurai RSA dan algoritma lainnya.

4. Teknis enkripsi dan dekripsi dalam RSA

Teknik enkripsi tentulah sangat dibutuhkan dalam segala macam algoritma enkripsi. Untuk mengenkripsi suatu pesan menggunakan RSA dapat menggunakan cara di bawah ini.

Misalkan pesan *M* adalah pesan yang akan dienkripsi. Pertama tentukan nilai *e* dan *n*. Representasikan pesan *M* dalam format integer dengan menggunakan representasi apa saja. Tujuannya bukan untuk mengenkripsi, tetapi untuk mendapatkan bentuk numerik dari pesan *M* untuk enkripsinya. Lalu enkripsikan pesan *M* dengan memangkatkannya dengan *e* lalu dimodulo dengan *n*. Hasilnya, *chipertext C*, adalah sisa ketika M^e dibagi *n*.

Untuk mendekirpsikannya, *chipertext C* dipangkat *d* lalu dimodulo *d*. Jika *E* adalah proses enkripsi dan *D* adalah proses dekripsi, maka:

$$C = E(M) = M^e \pmod n$$
$$M = D(C) = C^d \pmod n$$

Perhatikan bahwa enkripsi tidak menambah ukuran dari pesan; baik pesan dan *chipertext* adalah bilangan bulat antara 0 sampai *n*-1. Kunci enkripsinya adalah sepasang bilangan bulat positif (*e; n*). Serupa dengan kunci enkripsi, kunci dekripsi juga merupakan bilangan bulat positif (*d; n*).

Setiap *user* menjadikan kunci enkripsi mereka sebagai kunci publik, dan kunci dekripsi mereka sebagai kunci privat.

Cara menentukan kode enkripsi dan dekripsi adalah sebagai berikut:

Pertama tertukan *n* sebagai hasil kali dari dua buah bilangan prima *p* dan *q*.

$$n = p \cdot q$$

bilangan-bilangan prima ini adalah bilangan acak yang sangat besar. Walaupun kita akan membuat *n* sebagai publik, *p* dan *q* tidak boleh diberitahukan karena adalah jawaban dari faktor dari *n*. Hal ini juga menyembunikan jalan untuk mendapatkan *d* dari *e*.

Kemudian tetukanlah bilangan bulat *d* yang prima terhadap $(p-1) \cdot (q-1)$ yang berarti:

$$FPB(d,(p-1)\cdot(q-1)) = 1$$

FPB=Faktor Persekutuan Terbesar

Setelah ini kita mendapatkan

$$e \cdot d \equiv 1 \pmod{(p-1) \cdot (q-1)}$$

5. Algoritma dalam RSA

5.1 Mengenkripsi dan medekripsi secara efisien

Perhitungan $M^e \pmod n$ paling banyak membutuhkan $2 \cdot \log_2(e)$ kali perkalian dan $2 \cdot \log_2(e)$ kali pembagian menggunakan prosedur berikut:

- Biarkan $e_k e_{k-1} e_{k-2} \ldots e_1 e_0$ menjadi representasi biner dari e.
- Ubah variabel C menjadi 1
- Ulangi dua prosedur di bawah ini untuk $i=k,k-1,\ldots,1,0$
  - Ubah C menjadi sisa dari C^2 ketika dibagi n.
  - Jika $e_i = 1$, ubahlah C menjadi sisa dari $C \cdot M$ ketika dibagi n.
- Berhenti, sekarang C adalah enkripsi dari M

Prosedur ini dikenal sebagai "eksponentasi dengan pengakaran dan perkalian berulang". Prosedur ini setengah efisien daripada prosedur terbaik yang dikenal.

Faktanya proses enkripsi dan proses dekripsi adalah identik terhadap suatu implementasi. Karena itulah proses ini dapat disimpan dalam sebuah IC (*Integrated Circuit*) *chip*. Sebuah komputer berkecepatan tinggi dapat mendekripsi pesan yang berisi 200 karakter dalam beberapa detika saja. *Special-purpose-computer* dapat melakukannya jauh lebih cepat lagi.

5.2 Bagaimana mencari bilangan prima besar

Setiap *user* harus (secara privat) memilih dua buah bilangan bulat acak p dan q untuk membuat kunci enkripsi dan kunci dekripsinya sendiri. Nomor-nomor ini haruslah sangat besar sehingga tidak mudah untuk memfaktorkan $n = p \cdot q$ (ingat, yang kita publikasikan hanyalah n, bukan p maupun q). Direkomendasikan p dan q memiliki panjang 100 digit sehingga n akan memiliki panjang 200 digit.

Untuk Menemukan bilangan prima sepanjang 100 digit, bangkitkanlah suatu angka 100 digit sampai satu angka prima ditemukan. Berdasarkan teorema bilangan prima, sekitar $(\ln 10^{100})/2 = 115$ angka dites sebelum kita mendapatkan angka prima.

Untuk mengetes angka besar b direkomendasikan menggunakan algoritma "probabilitas" elegan oleh Solovay dan Strassen. Algoritma ini mengambil angka a secara acak dari distribusi seragam $\{1,2,\ldots,b-1\}$ dan mengetes apakah

$$FPB(a,b) = 1 \text{ and } J(a,b) = a^{(b-1)/2}$$

Di mana $J(a,b)$ adalah simbol Jacobi. Jika b adlah bilangan prima, maka persamaan di atas selalu benar. Jika b adalah bilangan komposit, maka persamaan di atas mungkin salah dengan kemungkinan 1:2. Jika persamaan di atas menggunakan 100 angka acak a lalu b hampir pasti bilangan prima; hanya ada kemungkinan $1:2^{100}$ bahwa b bilangan komposit. Walaupun bilangan komposit tanpa sengaja terpakai dalam sistem kita, penerima akan mendeteksi kalau enkripsinya tidak berjalan normal. Jika b ganjil, $a \leq b$, dan FPB(a,b) = 1, simbol Jacobi J(a,b) besarnya antara {-1,1}, dan dapat dikerjakan dengan baik oleh program:

$J(a,b) = \text{if } a=1 \text{ then } 1 \text{ else}$
$\text{If } a \text{ adalah genap then } J(a/2,b)\cdot(-1)^{(b+1)(b-1)/8}$
$\text{else } J(b \pmod a,a)\cdot(-1)^{(a-1)(b-1)/4}$

Perhitungan J(a,b) dan FPB(a,b) dapat dikombinasikan dengan baik pula. Perhatikan pula bahwa algoritma ini tidak mengetes sebuah angka dengan langsung memfaktorkannya.

Untuk mendapatkan perlindungan tambahan dari algoritma pemfaktoran yang rumit, p dan q harus berbeda sedikit dalam jumlah digitnya. (p-1) dan (q-1) harus juga mempunyai banyak faktor-faktor prima, dan FPB((p-1),(q-1)) harus kecil. Seteleh ini kondisi lainnya dapat dicek dengan mudah.

Untuk menemukan sebuah bilangan prima p di mana (p-1) memiliki faktor prima yang berlimaph, bangkitkanlah bilangan prima besar u, kemudian jadikanlah p angka prima pertama dalam tahap $i \cdot u+1$, untuk i = 2,4,6,.. (proses ini tidak memakan waktu yang cukup lama). Untuk menambah tingkat keamanan (u-1) harus memiliki faktor prima yang banyak.

Sebuah komputer berkecepatan tinggi dapat menentukan dalam beberapa detika saja apakah bilangan dengan panjang 100 digit adalah bilangan prima atau bukan bilangan prima dan dapat menemukan bilangan prima pertama kira-kira antara satu sampai dua menit setelah diperintahkan.

Cara lain untuk menemukan bilangan prima besar adalah dengan mengambil sebuah bilangan dari faktorisasi yang telah diketahui. Tambahan satu dan ceklah apakah prima atau tidak. Jika bilangan prima p ditemukan, maka adalah mungkin membuktikan bahwa itu adalah bilangan prima dengan faktorisasi (p-1).

5.3 Bagaimana memilih *d*

Mencari angka d yang relatif prima terhadap $\phi(n)$ adalah hal yang sangat mudah. Sebagai contoh, bilangan prima manapun yang lebih besar daripada max(p,q) dapat digunakan. Sangatlah penting untuk memilih d dari himpunan yang cukup besar agar kriptanalis tidak dapat menemukannya dengan pencarian langsung.

5.4 Bagaimana menghitung *e* dari *d* dan $\phi(n)$

Untuk menghitung e, gunakan variasi dari algoritma Euclid di bawah ini untuk menghitung faktor persekutuan terbesar dari $\phi(n)$ dan n. Hitung FPB($\phi(n)$,d) dengan cara memproses beberapa seri $x_o,x_1,x_2,..$ di mana

$$x_0 \equiv \phi(n),$$
$$x_1 = d,$$
$$x_{i+1} \equiv x_{i-1} \pmod{x_i}$$

sampai $x_k = 0$ ditemukan. Kemudian FPB(x_0,x1) = x_{k-1}. Hitunglah, untuk setiap nilai x_i, nilai a_i, dan b_i sedemikian sehingga $x_i = a_i \cdot x_0 + b_i \cdot x_i$. Jika $x_{k-1} = 1$, maka b_{k-1} adalah invers dari perkalian $x_1 \pmod{x_0}$. Karena k akan selalu lebih kecil daripada 2 $\log_2(n)$, perhitungan ini akan sangat cepat.

Jika e mengahsilkan nilai yang lebih kecil daripada $\log_2(n)$, ulangi lagi dan pilih nilai d yang berbeda. Hal ini memberi menjamin bahwa setiap pesan yang dienkripsi (kecuali untuk $M = 1$ dan $M = 0$) akan mengalami "*wrap-around*" (modulo tereduksi n).

6. Aplikasi RSA dalam kehidupan sehari-hari

sebuah data. Untuk meyakinkan penerima surat elektronik yang ditandatangani, diperlukan pembuktian bahwa surat elektronik tersebut memang berasal dari si pengirim.

Untuk mencegah penyalahgunaan dalam surat elektronik, maka tanda tangan elektronik sangat dibutuhkan. Tanda tangan elektronik inipun sebaiknya mempunyai sifat-sifat sebagai berikut:

- Tanda tangan itu asli (otentik), tidak mudah ditulis/ditiru oleh orang lain. Pesan dan tanda tangan pesan tersebut juga dapat menjadi barang bukti, sehingga penandatangan tak bisa menyangkal bahwa dulu ia tidak pernah menandatanganinya.
- Tanda tangan itu hanya sah untuk dokumen (pesan) itu saja. Tanda tangan itu tidak bisa dipindahkan dari suatu dokumen ke dokumen lainnya. Ini juga berarti bahwa jika dokumen itu diubah, maka tanda tangan digital dari pesan tersebut tidak lagi sah.
- Tanda tangan itu dapat diperiksa dengan mudah.
- Tanda tangan itu dapat diperiksa oleh pihak-pihak yang belum pernah bertemu dengan penandatangan.
- Tanda tangan itu juga sah untuk kopi dari dokumen yang sama persis.
- Sebuah tanda tangan elektronik harus bersifat *message-dependent*, juga *signer-dependent*. Maksudnya di sini adalah menyatu dengan surat elektroniknya. Jika tidak si penerima dapat memodifikasi pesan yang terkirim. Atau bisa saja si penerima melampirkan ke surat manapun, apalagi sangat tidak mungkin untuk mendeteksi "*copy - paste*" secara elektronik.

Untuk mengimplementasikan tanda tangan elektronik kepada kunci publik, sistem kriptografi harus diimplemetasikan dengan *trap-door one-way permutations*, karena algoritma dekripsi dapat diaplikasikan untuk surat yang tidak terdapat *ciphertext.*

Misalnya pada satu kasus, Doni akan mengirim sebuah surat elektronik yang dilengkapi dengan tanda tangan elektronik kepada Edi. Pertama-tama Doni mengkomputasi tanda tangan elektroniknya, S, untuk surat elektronik M, dengan menggunakan kunci D_E. Sedemikian rupa sehingga sesuai dengan persamaan

$$S = D_E (M)$$

Kemudian dia mengenkripsi S menggunakan E_D (untuk privasi), lalu mengirimkannya kepada Edi dalam bentuk E_D (S). Doni tidak perlu mengirimkan M kepada Edi, karena surat dapat dikomputasi dari S. Kemudian Edi mendekripsi *ciphertext* dengan menggunakan D_D mengubah S.Edi tahu siapa pengirim surat elektronik dari tanda tangan elektronik Doni. Diasumsikan Edi sudah tahu seperti apa tanda tangan elektronik Doni. Kemudian dia membuka (mendekripsi) paket surat yang dikirim ke Edi sama dengan prosedur enkripsi pengiriman tetapi kebalikannya:

$$M = E_E (S)$$

Setelah itu mencek kemiripan tanda tangan elektronik dari 2 surat elektronik yang ada (M,S).

Doni yang telah mengirim surat elektronik kepada Edi tidak dapat menyangkal lagi bahwa dia telah mengirimkan sebuah surat elektronik kepada Edi. Karena tidak ada orang lain yang dapat membuat $S = D_E (M)$. Dan Edi pun dapat membuktikan bahwa memang surat elektronik tersebut asli dari Doni karena tanda tangan elektroniknya asli, karena dia dapat membuktikan $E_B (S) = M$.

Atau jika diimplementasikan dalam RSA, proses enkripsi-dekripsi data saat pengecekan keaslian dari pesan yang dikirim adalah sebagai berikut. Misalkan Doni ingin mengirim pesan kepada Edi. Doni membuat sebuah *hash value* dari pesan tersebut, di pangkatkan dengan bilangan *d* dibagi *N* (seperti halnya pada deskripsi pesan), dan melampirkannya sebagai "tanda tangan" pada pesan tersebut. Saat Edi menerima pesan yang telah "ditandatangani", Edi memangkatkan "tanda tangan" tersebut dengan bilangan *e* dibagi *N* (seperti halnya pada enkripsi pesan), dan membandingkannya dengan nilai hasil dari hash value dengan hash value pada pesan tersebut. Jika kedua cocok, maka Edi dapat mengetahui bahwa pemilik dari pesan tersebut adalah Doni, dan pesan pun tidak pernah diubah sepanjang pengiriman.

Sehingga pada akhirnya Edi dapat mengecek keaslian tanda tangan elektronik dari sebuah surat elektronik, tetapi dia tidak bisa mengubah-ubah tanda tangan yang sudah ada di surat elektronik tersebut.

Sistem enkripsi RSA dapat diaplikasikan dalam pengiriman data, apalagi data-data yang bersifat rahasia dan memerlukan autentikasi, seperti dalam mengirimkan nomor kartu kredit atau nomor rekening pada saat transaksi di internet. Sebuah sistem pengecekan *online* sebaiknya mengikuti sistem tanda tangan elektronik serperti di atas. Dan jika semua hal di internet yang membutuhkan autentikasi pengirim data menggunakan metode enkripsi, maka pengguna komputer akan dengan mudahnya membuat suatu transaksi di internet. Seperti yang telah terjadi sekarang, semuanya sudah *online*. Untuk membayarnya pun cukup memasukkan nomor kartu kredit dan beberapa nomor pengaman dari kartu kredit tersebut.

Pada sistem RSA, *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.

Pada saat ini RSA pada dunia nyata dipakai di segala bidang. RSA dewasa ini dipakai di berbagai macam produk, industri dan standar di seluruh dunia. RSA ditemukan di berbagai perangkat lunak komersil dan direncanakan agar penggunaan lebih ditingkatkan. RSA dibangun ke dalam sistem operasi yang dipakai pada saat ini oleh Microsoft, Apple, Sun, dan Novell. Pada skala perangkat keras, RSA dapat digunakan pada secure telephone (telepon yang dilengkapi piranti keamanan), kartu jaringan Ethernet, dan pada kartu cerdas (smart cards). RSA juga digabungkan ke

dalam internet protokol berskala besar untuk digunakan pada komunikasi internet yang aman, seperti SSL, S-HTTP, SEPP, S/MIME, S/WAN, STT and PCT. RSA juga digunakan di banyak institusi, termasuk cabang-cabang dari pemerintah Amerika Serikat, perusahaan-perusahaan besar, laboratorium besar dan universitas

7. Keuntungan dan kerugian dalam pengaplikasian RSA

Seperti pada algoritma enkripsi lainnya, RSA mempunyai beberapa keuntungan dan kerugian dalam pemakaiannya. Berikut ini adalah keuntungan dan kerugiannya.

Keuntungan utama dari RSA adalah sistem RSA yang merupakan sistem kunci publik ini dapat menyediakan sebuah metode untuk tanda tangan digital atau tanda tangan elektronik. Autentikasi melalui kunci rahasia memerlukan pembagian dari beberapa rahasia dan terkadang juga memerlukan rasa kepercayaan terhadap pihak ketiga, yang hasilnya, pengirim dapat menolak pesan autentikasi sebelumnya dengan cara membuktikan bahwa rahasia yang dibagikan bagaimanapun caranya disetujui oleh pihak lain yang berbagi rahasia tersebut. Sebagai contoh, sistem autentikasi kunci rahasia Cerberus melibatkan sebuah basis data pusat yang menyimpan salinan dari kunci-kunci rahasia dari semua *user*. Sebuah serangan ke basis data tersebut dapat menyebabkan data tersebut tersalin banyak. Autentikasi kunci publik, di lain sisi, mencegah kejadian penolakan seperti ini. Setiap *user* mempunyai tanggung jawab sendiri untuk menjaga kunci privatnya masing-masing. Jenis autentikasi yang seperti ini sering disebut *non-repudation*.

Salah satu keuntungan utama lainnya dari RSA yang merupakan kriptografi kunci publik adalah menambah keamanan dan kenyamanan. Kunci privat tidak pernah diperlukan untuk dikirim atau diberi tahu ke orang lain. Pada sebuah sistem kunci rahasia, secara terang-terangan kunci rahasia ini harus dikirim (bisa secara manual atau melalui sebuah saluran komunikasi), dan akan terjadi suatu kemungkinan dimana penyerang dapat mencari tahu kunci rahasia tersebut saat proses pengiriman.

Kekurangan dari pemakaian kriptografi kunci publik, dalam hal ini RSA, adalah dalam masalah kecepatan. Banyak metode enkripsi kunci rahasia yang populer yang memiliki kecepatan enkripsi-dekripsi yang lebih cepat dibandingkan dengan metode enkripsi kunci publik yang ada sekarang. Namun kriptografi kunci publik dapat digunakan dengan kriptografi kunci rahasia untuk mendapatkan metode enkripsi yang terbaik di dunia. Untuk enkripsi, solusi terbaik adalah dengan cara mengkombinasi sistem kunci publik dan sistem kunci rahasia untk mendapatkan kedua keuntungan yang dimiliki oleh kedua metode enkripsi ini, keuntungan keamanan dari segi sistem kunci publik, dan keuntungan kecepatan dari segi sistem kunci rahasia. Sistem kunci publik dapat digunakan untuk mengenkripsi sebuat kunci rahasia, yang bisa digunakan untuk mengenkripsi file atau pesan yang berukuran besar sekalipun.

Kriptografi kunci publik dapat menjadi lemah terhadap pemalsuan identitas *user*, bagaimana pun juga, walupun jika kunci privat dari pemakai tidak tersedia. Sebuah serangan yang sukses pada sebuah otoritas sertifikasi akan memperbolehkan lawan untuk menyelinap siapapun yang lawan pilih dengan cara memilih sertifikat kunci publik dari sebuah otoritas yang memilikinya untuk menggabungkan kunci tersebut ke nama user yang lain.

Di beberapa situasi, kriptografi kunci publik tidak perlu dan kriptografi rahasia saja sudah cukup. Situasi ini termasuk lingkungan dimana persetujuan kunci rahasia mempunyai sebuah tempat penting, sebagai contoh oleh beberapa *user* yang melakukan pertemuan dengan sifat pribadi. Hal lain yang mempengaruhi adalah lingkungan dimana sebuah otoritas mengetahui dan mengatur semua kunci, misalkan sebuah sistem perbankan tertutup. Karena otoritas sudah mengetahui kunci setiap orang, tidak terdapat keuntungan berarti untuk menggunakan sesuatu yang bersifat publik dan yang lainnya pribadi. Dan juga kriptografi kunci publik seringkali tidak diperlukan di sebuah

lingkungan yang hanya terdapat satu *user*. Sebagai contoh, jika kita ingin mentimpan file-file secara terenkripsi, kita bisa melakukannya dengan menggunakan algoritma enkripsi kunci rahasia yang menggunakan, bisa dikatakan, sandi lewat pribadi kita. Pada pemakaian biasanya, kriptografi kunci publik sangat tepat untuk digunakan pada suatu lingkungan dengan banyak *user* yang bersifat terbuka.

Kriptografi kunci publik tidak diciptakan untuk menggantikan kriptografi kunci privat, tetapi lebih untuk menutupi kekurangan yang ada, untuk membuat suatu sistem enkripsi menjadi aman. Kegunaan awal dari teknik kunci publik ini adalah untuk penukaran kunci pengaman di dalam suatu sistem kunci rahasia. Ini masih menjadi salah satu dari kegunaan utamanya. Kriptografi kunci rahasia tetap menjadi sangat penting dan menjadi sebuah bahan pembelajaran dan riset yang sedang diadakan.

8. Kesimpulan

Kesimpulan yang dapat diambil dari makalah ini adalah:

- RSA adalah salah satu metode/algoritma enkripsi yang banyak digunakan dalam berbagai macam perangkat.
- RSA merupakan salah satu algoritma enkripsi terkuat saat ini, dan lebih baik dibandingkan DSA dan SED.
- Algoritma RSA berisi pemfaktoran yang amat rumit sehingga tidak mudah dibobol
- RSA memiliki dua buah kunci, yaitu kunci privat dan kunci publik. Kunci publik adalah kunci yang dipublikasikan, sedangkan kunci privat adalah kunci yang tidak boleh diberitahukan kepada siapapun.
- Dalam kehidupan sehari-hari, aplikasi sisten RSA dapat ditemukan pada sistem autentikasi data dan pembuatan tanda tangan digital pada komputer, pada tingkat perangkat keras, RSA banyak digunakan pada telepon yang mempunyai sistem pengaman dari penyadapan, kartu jaringan ethernet, dan pada kartu cerdas. RSA juga dimasukkan ke dalam protokol internet yang mempunyai sistem pengaman, seperti S-HTTP, S/MIME dan lain-lain

Daftar Pustaka

Menezes A.; van Oorschot P.; and Vanstone S., 1996. *Handbook of Applied Cryptography*. CRC Press

http://www.geocities.com/amwibowo/resource/komparasi/komparasi.html, waktu akses: 31 Desember 2006, 21:47

http://www.muppetlabs.com/~breadbox/txt/rsa.html#17, waktu akses: 30 Desember 2006, 15.00

http://www.wikipedia.org, waktu akses: 30 Desember 2006, 15.00