STUDI PEMAKAIAN ALGORITMA RSA DALAM PROSES ENKRIPSI dan APLIKASINYA

Prasetyo Andy Wicaksono – 13505030

Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if15030@students.if.itb.ac.id

Abstrak

Makalah ini membahas tentang studi dan contoh pemakaian sistem sandi RSA dalam penyandian data di jaringan. RSA adalah sistem sandi yang saat ini praktis menjadi standar *de facto* dunia di samping DES. Sandi ini adalah hasil inovasi Ron Rivest, Adi Shamir, dan Leonard Adleman di tahun 1978. Mereka kemudian mendirikan perusahaan RSA Data Security Inc, yang memiliki paten atas sandi RSA.

RSA banyak dipakai oleh banyak perangkat lunak di dunia, contohnya adalah pada program browser internet MS Internet Explorer dan Netscape. Salah satu sistem penyandian yang juga banyak dipakai adalah DES (Data Encryption Standard). Mekanisme kerja RSA cukup sederhana dan mudah mengerti, tetapi kokoh. 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*.

***Kata kunci**: RSA, plaintext, ciphertext, brute force attack, Chinese Remainder Theoerm, hash function, hash-value,* enkripsi, dekripsi

1. Pendahuluan

Sebagai media komunikasi umum, suatu jaringan sangat rawan terhadap penyadapan, pencurian, dan pemalsuan informasi. Proses pengiriman data pada suatu jaringan harus menjamin keamanan dan keutuhan, jika tidak, akan terjadi kemungkinan-kemungkinan seperti yang dijelaskan sebelumnya, sehingga data yang dikirim dapat sampai di tujuannya. 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.

Ilmu menyandi (kriptografi) sebetulnya adalah ilmu yang sudah dikenal bahkan semenjak jaman Julius Caesar (sebelum masehi). 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 1978. 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.

Untuk menyandi informasi dan untuk menerjemahkan pesan tersandi sebuah algoritma penyandian memerlukan sebuah data biner yang disebut kunci. Tanpa kunci yang cocok orang tidak bisa mendapatkan kembali pesan asli dari pesan tersandi. Pada DES digunakan kunci yang sama untuk menyandi (enkripsi) maupun untuk menterjemahan (dekripsi), sedangkan RSA menggunakan dua kunci yang berbeda. Isitilahnya, DES disebut sistem sandi simetris sementara RSA disebut sistem sandi *asimetris*. Kedua sistem ini memiliki keuntungan dan kerugiannya sendiri. Sistem sandi simetris cenderung jauh lebih cepat sehingga lebih disukai oleh sementara kalangan industri. Kejelekannya, pihak-pihak yang ingin berkomunikasi secara privat harus punya akses ke sebuah kunci DES bersama. Walaupun biasanya pihak-pihak yang terkait sudah saling percaya, skema ini memungkinkan satu pihak untuk memalsukan pernyataan dari pihak lainnya.

RSA yang menggunakan algoritma asimetrik mempunyai dua kunci yang berbeda, disebut pasangan kunci (*key pair*) untuk proses enkripsi dan dekripsi. Kunci-kunci yang ada pada pasangan kunci mempunyai hubungan secara matematis, tetapi tidak dapat dilihat secara komputasi untuk mendeduksi kunci yang satu ke pasangannya. Algoritma ini disebut kunci publik, karena kunci enkripsi dapat disebarkan. Orang-orang dapat menggunakan kunci publik ini, tapi hanya orang yang mempunyai kunci privat sajalah yang bisa mendekripsi data tersebut.

2. Mekanisme dasar kerja RSA

Tingkat keamanan algoritma penyandian RSA sangat bergantung pada ukuran kunci sandi tersebut (dalam bit), karena makin besar ukuran kunci, maka makin besar juga kemungkinan kombinasi kunci yang bisa dijebol dengan metode mengencek kombinasi satu persatu kunci atau lebih dikenal dengan istilah *brute force attack*. Jika dibuat suatu sandi RSA dengan panjang 256 bit, maka metode *brute force attack* akan menjadi tidak ekonomis dan sia-sia dimana para *hacker* pun tidak mau/sanggup untuk menjebol sandi tersebut.

2.1. Proses Pembuatan Kunci

Dalam membuat suatu sandi, RSA mempunyai cara kerja dalam membuat kunci publik dan kunci privat adalah sebagai berikut:

1. Pilih dua bilangan prima p dan q secara acak , $p \neq q$. Bilangan ini harus cukup besar (minimal 100 digit).
2. Hitung $N = pq$. Bilangan N disebut *parameter sekuriti*.
3. Hitung $\varphi = (p-1)(q-1)$.
4. Pilih bilangan bulat (*integer*) antara satu dan φ ($1 < e < \varphi$) yang tidak mempunyai faktor pembagi dari φ.
5. Hitung d hingga $d\,e \equiv 1 \pmod{\varphi}$.

Keterangan:

- Langkah 3 dan 4 dapat dihasilkan dengan cara algoritma Euclidean
- Langkah 4 dapat dihasilkan dengan menemukan integer x sehingga $d = (x(p-1)(q-1) + 1)/e$ menghasilkan bilangan bulat, kemudian menggunakan nilai dari $d \pmod{(p-1)(q-1)}$

Setelah melalu cara ini, maka kita akan mendapatkan kunci publik dan kunci privat. Kunci publik terdiri dari dua elemen, yaitu:

- N, merupakan modulus yang digunakan
- e, eksponen publik atau eksponen enkripsi

dan kunci privat, yang terdiri dari:

- N, merupakan modulus yang digunakan, sama seperti pada kunci publik
- d, eksponen pribadi atau eksponen deskripsi, yang harus dijaga kerahasiaanya

Nilai p dan q sebaiknya dibuang atau dijaga kerahasiaannya, karena terdapat N dimana p dan q adalah faktor pembagi dari N. Walaupun bentuk ini memperbolehkan dekripsi secara cepat dan *signing* menggunakan *Chinese Remainder Theorem* (CRT), hal ini mejadi lebih tidak aman karena bentuk ini memperbolehkan *side channel attacks*. *Side channel attacks* adalah sebuah serangan yang berdasarkan informasi yang dikumpulkan dari

implementasi fisik (atau kelemahan secara fisik) dari sebuah sistem kriptografi, dibanding dengan kelemahan teoritis dari algoritmanya sendiri. Sebagai contohnya, faktor-faktor kurun waktu dari informasi, konsumsi tenaga, bahkan suara yang ditimbulkan dapat membantu mempermudah informasi yang bisa diambil untuk menjebol sistem tersebut.

2.2. Proses Enkripsi Pesan

Misalkan pada suatu kasus si A ingin mengirim pesan *m* kepada si B. A mengubah *m* menjadi angka $n < N$, menggunakan protokol yang sebelumnya telah disepakati dan dikenal sebagai *padding scheme*. *Padding scheme* harus dibangun secara hati-hati sehingga tidak ada nilai dari *m* yang menyebabkan masalah keamanan. Contohnya, jika kita ambil contoh sederhana dari penampilan ASCII dari *m* dan menggabungkan bit-bit secara bersama-sama akan menghasilkan *n*, kemudian pessan yang berisi ASCII tunggal karakter NUL (nilai numeris 0) akan menghasilkan $n = 0$, yang akan menghasilkan *ciphertext* 0 apapun itu nilai dari *e* dan *N* yang digunakan.

Maka A mempunyai nilai *n* dan mengetahui *N* dan *e*, yang telah diumumkan oleh B. A kemudian menghitung *ciphertext c* yang terkait pada *n*:

$$c = n^e \mod N$$

Perhitungan tersebut dapat diselesaikan dengan menggunakan metode *exponentation by squaring*, yaitu sebuah algoritma yang dipakai untuk komputasi terhadap sejumlah nilai integer yang besar dengan cepat. Kemudian A mengirimkan nilai c kepada B.

2.3. Proses Dekripsi Pesan

B sudah menerima c dari A, dan mengetahui kunci privat yang digunakan B. B kemudian mengembalikan nilai *n* dari c dengan langkah-langkah sebagai berikut:

$$n = c^d \mod N$$

Perhitungan diatas akan menghasilkan *n*, dengan begitu B dapat mengembalikan pesan semula *m*. Prosedur dekripsi bekerja karena

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

Kemudian, karena $ed \equiv 1 \pmod{p-1}$ dan $ed \equiv 1 \pmod{q-1}$, hasil dari *Fermat's little theorem*.

$$n^{ed} \equiv n \pmod p$$

dan

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

Karena *p* dan *q* merupakan bilangan prima yang berbeda, mengaplikasikan *Chinese remainder theorem* akan menghasilkan dua macam kongruen

$$n^{ed} \equiv n \pmod{pq}$$

serta

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

2.4. Contoh Penghitungan RSA

Sekarang kita mencoba suatu contoh untuk mengenal lebih dalam sistem kerja enkripisi RSA. Misalnya kita mau mengenkripsi kata "SECRET" dengan RSA, lalu kita dekripsi kembali ke dalam *plain text*.

Karena *p* dan *q* berjumlah minimal 100 digit atau lebih, nilai *d* dan *e* bisa berjumlah sama dengan 100 digit dan nilai *N* akan berjumlah 200 digit. Untuk itu di contoh pemakaian berikut, kita akan memakai angka-angka yang kecil agar mudah dalam penghitungan. Cara pengerjaannya adalah:

1. Kita pilih $p = 3$ dan $q = 5$
2. Hitung $N = pq = 3*5 = 15$
3. Nilai *e* harus merupakan bilangan prima yang lebih besar dan relatif dekat dengan $(p-1)(q-1) = (2)(4) = 8$, sehingga kita pilih $e = 11$. Angka 11 adalah bilangan prima terdekat dan lebih besar daripada 8
4. Nilai *d* harus dipilih sehingga
   $$\frac{(ed - 1)}{(p-1)(q-1)}$$
   adalah sebuah integer. Lalu nilai
   $$(11d - 1) / [(2)(4)] = (11d - 1) / 8$$
   juga merupakan integer. Setelah melalui proses penghitungan, salah satu nilai yang mungkin adalah $d = 3$.
5. Lalu kita masukkan kata yang akan dienkripsi, "SECRET". Kita akan mengkonversi string ini ke representasi desimal menggunakan nilai karakter ASCII, yang akan menghasilkan nilai ASCII 83 69 67 82 69 84

6. Pengirim akan mengenkripsi setiap digit angka pada saat yang bersamaan menggunakan nilai kunci publik (*e*, *n*) = (11,15). Lalu setiap karakter *ciphertext* akan masuk ke persamaan
$$C_i = M_i^{11} \bmod 15.$$
Yang akan menghasilkan nilai digit masukan adalah 0x836967826984 yang akan dikirim sebagai 0x2c696d286924
7. Penerima akan mendekripsi setiap digit angka menggunakan nilai kunci privat (*d*, *n*) = (3, 15). Lalu, setiap karakter *plaintext* akan masuk persamaan
$$M_i = C_i^3 \bmod 15.$$
String masukan yang bernilai 0x2c696d286924, akan dikonversi kembail menjadi 0x836967826984, dan akhirnya angka-angka tersebut akan diubah kembali menjadi bentuk string *plaintext* yang bernilai "SECRET"

Dari contoh di atas kita dapat menangkap suatu kelemahan dari pemakaian *p* dan *q* yang bernilai kecil yaitu bisa kita lihat di digit ke-4, ke-6 dan ke-9 tidak berubah saat dienkripsi, dan nilai 2 dan 8 dienkripsi menjadi 8 dan 2, yang berarti dienkripsi menjadi kebalikannya. Tapi kesimpulan yang bisa diambil dari contoh yang sederhana ini adalah RSA dapat digunakan dalam penyandian dalam pengiriman informasi.

Kunci RSA yang mempunyai ukuran 512 dan 768 bit dianggap masih lemah dan mudah dijebol. Ukuran kunci yang dianjurkan adalah 1024 bit. Ukuran 2048 dan 3072 bit merupakan suatu ukuran yang lebih baik.

3. Keamanan RSA

Keamanan dari sistem kriptografi RSA adalah didasari oleh dua problem matematika:

- Problem dalam faktorisasi bilangan berjumlah banyak
- Problem RSA, yaitu mencari modulo akar e^{n} dari sebuah bilangan komposit *N* yang faktor-faktornya tidak diketahui

Proses dekripsi penuh dari sebuah *ciphertext* RSA dianggap sesuatu hal yang tidak mudah karena kedua problem ini diasumsikan sulit. Belum ada algoritma yang mangkus untuk menyelesaikannya.

Problem RSA didefinisikan sebagai tugas untuk mencari suatu akar modulo e^{n} (e pangkat ke n)dari bilangan komposit *N*. Mengembalikan suatu nilai *m* dimana $m^e = c \bmod \mathrm{n}$, (e, n) adalah kunci publik RSA dan *c* adalah *ciphertext* RSA.

Metode pendekatan yang diyakini dapat menyelesaikan problem RSA saat ini adalah memfaktor dari modulus *n*. Dengan kemampuan untuk mengembalikan faktor yang merupakan bilangan prima, sebuah serangan dapat menghitung eksponen rahasia dari *d* dan dari kunci publik (*e, n*), lalu mendekripsi *c* menggunakan prosedur standar. Untuk menyelesaikannya, penyerang (bisa penyadap, penguping, dll.) memfaktor nilai *n* menjadi *p* dan *q*, lalu menghitung (p-1)(q-1) yang dapat menghasilkan nilai *d* dan *e*.

Penyerangan yang paling umum pada RSA ialah pada penanganan masalah faktorisasi pada bilangan yang sangat besar. Apabila terdapat faktorisasi metode yang baru dan cepat telah dikembangkan, maka ada kemungkinan untuk membongkar RSA.

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.

3.1. Membuat RSA Sukar Dijebol

Jika nilai *N* berjumlah kecil, maka nilai faktor *p* dan *q* akan mudah diterka oleh para *hacker*.

Maka untuk membuat nilai *N* sukar untuk dijebol oleh para *hacker* kita perlu nilai faktor *p* dan *q* yang besar. Misalkan, dibandingkan kita memilih nilai 5 dan 11, lebih baik kita pilih bilangan prima yang besar, seperti 673 dan 24971, yang akan menghasilkan nilai d = 16805483 dan nilai e = 16779840.

Tetapi jika dihitung dengan suatu perangkat lunak ataupun suatu program yang kita buat yang dapat menghitung faktor-faktor dari suatu nilai. Angka-angka di atas dapat dengan mudahnya didapatkan faktor-faktornya. Sehingga hal ini menyimpulkan bahwa kita membutuhkan nilai *p* dan *q* yang sangat besar.

3.2. Ancaman yang Mungkin Menyerang RSA

Sistem pengenkripsian RSA mempunyai kemungkinan-kemungkinan kelemahan yang bisa diserang oleh para *eavesdropper* (penyadap, penguping), berikut adalah kelemahan-kelemahan dalam RSA yang sebaiknya dihindari:

- Nilai *n* terlalu kecil, sehingga mudah untuk difaktorisasi
- Jumlah nilai eksponen e^n yang terlalu kecil
- Ukuran kunci yang terlalu kecil, sehingga sandi dapat dijebol dengan *brute force attack*
- Nilai *d* terlalu kecil
- Pengunaan nilai modulus yang familiar, hal ini memudahkan para *hacker* untuk menjebol sandi yang ada

3.3. Pertimbangan Teknis dalam Enkripsi RSA

Jika kita berniat untuk mengenkripsi suatu data untuk dikirim ke suatu tujuan, ada beberapa faktor yang sebaiknya diperhatikan agar data yang kita kirim tidak mudah dijebol di tengah jalan.

1. Dalam pembuatan kunci, sebaiknya memilih nilai *p* dan *q* yang jumlahnya tidak saling berdekatan dan besar, karena jika nilai *N* kecil, faktor dari *N* akan sangat mudah didapat. Seseorang seharusnya tidak melakukan metode pencarian bilangan prima yang hanya akan memberikan informasi penting tentang bilangan prima tersebut kepada penyerang. Biasanya, pembangkit bilangan acak yang baik akan memulai nilai bilangan yang digunakan. Harap diingat, bahwa kebutuhan disini ialah acak dan tidak-terduga. Berikut ini mungkin tidak memenuhi kriteria, sebuah bilangan mungkin dapat dipilah dari proses acak (misal, tidak dari pola apapun), tetapi jika bilangan itu mudah untuk ditebak atau diduga (atau mirip dengan bilangan yang mudah ditebak), maka metode tersebut akan kehilangan kemampuan keamanannya. Misalnya, tabel bilangan acak yang diterbitkan oleh Rand Corp pada tahun 1950-an mungkin memang benar-benar teracak, tetapi dikarenakan diterbitkan secara umum, hal ini akan mempermudah para penyerang dalam mendapatkan bilangan tersebut. Jika penyerang dapat menebak separuh dari digit *p* atau *q*, para penyerang dapat dengan cepat menghitung separuh yang lainnya.

2. RSA memiliki kecepatan yang lebih lambat dibanding DES dan algoritma simetrik lainnya. Contohnya si A membuat sandi dari sebuah pesan menggunakan algoritma simetrik, membuat sandi kunci simetrik menggunakan RSA dan juga mengirimkan pesan yang dienkripsi secara simetrik kepada B.

   Prosedur ini menambah permasalahan akan keamanan. Singkatnya, sangatlah penting untuk menggunakan pembangkit bilangan acak yang kuat untuk kunci simetrik yang digunakan, karena *eavesdropper* dapat melakukan *bypass* terhadap RSA dengan menebak kunci simetrik yang digunakan.

3. Sebagaimana halnya *cipher*, bagaimana *public key* RSA didistribusi menjadi hal penting dalam keamanan. Distribusi kunci harus aman dari *man-in-the-middle attack* (*penghadang-ditengah-jalan*). Anggap *eavesdropper* dengan suatu cara mampu memberikan kunci arbitari kepada A dan membuat A percaya bahwa kunci tersebut milik B. Anggap *eavesdropper* dapatmenghadang sepenuhnya transmisi antara A dan B. *Eavesdropper* mengirim A kunci publik milik B, dimana A percaya bahwa kunci publik tersebut milik B. *Eavesdropper* dapat menghadap seluruh *ciphertext* yang dikirim oleh A, melakukan dekripsi dengan kunci rahasia milik *Eavesdropper* sendiri, menyimpan salinan dari pesan tersebut, melakukan enkripsi menggunakan kunci publik milik B, dan mengirimkan *ciphertext* yang baru kepada B. Secara prinsip, baik A atau B tidak menyadari kehadiran *eavesdropper*

diantara transmisi mereka. Pengamanan terhadap serangan semacam ini yaitu menggunakan sertifikat digital atau kompnen lain dari infrastuktur kunci publik

4. Paul Kocher, kriptograf berkebangsaan Amerika Serikat, pemimpin perusahaan Cryptoghrapy Research Inc., menjelaskan sebuah serangan baru yang cerdas pada RSA di tahun 1995: jika penyerang, *eavesdropper*, mengetahui perangkat keras yang dimiliki oleh B secara terperinci dan mampu untuk mengukur waktu yang dibutuhkan untuk melakukan dekripsi untuk beberapa *ciphertext*, *eavesdropper* dapat menyimpulkan kunci dekripsi *d* secara cepat. Penyerangan ini dapat juga diaplikasikan pada skema "tanda tangan" RSA. Salah satu cara untuk mencegah penyerangan ini yaitu dengan memastikan bahwa operasi dekripsi menggunakan waktu yang konstan untuk setiap *ciphertext* yang diproses. Cara yang lainnya, yaitu dengan menggunakan properti multipikatif dari RSA. Sebagai ganti dari menghitung $c^d \bmod N$, B pertama-tama memilih nilai bilangan acak *r* dan menghitung $(r^e c)^d \bmod N$. Hasil dari penghitungan tersebut ialah $rm \bmod N$ kemudian efek dari *r* dapat dihilangkan dengan perkalian dengan inversenya. Nilai baru dari *r* dipilih pada tiap *ciphertext*. Dengan teknik ini, dikenal sebagai *message blinding* (pembutaan pesan), waktu yang diperlukan untuk proses dekripsi tidak lagi berhubungan dengan nilai dari *ciphertext* sehingga penyerangan waktu akan gagal.

5. Pada tahun 1998, Daniel Bleichenbacher menjelaskan penggunaan penyerangan *ciphertext* adaptif, terhadap pesan yang terenkripsi menggunakan RSA dan menggunakan PKCS #1 v1 *padding scheme*. Dikarenakan kecacatan pada skema PKCS #1, Bleichenbacher mampu untuk melakukan serangkaian serangan terhadap implementasi RSA pada protokol Secure Socket Layer, dan secara potensial mengungkap kunci-kunci yang digunakan. Sebagai hasilnya, para pengguna kriptografi menganjurkan untuk menggunakan *padding scheme* yang relatif terbukti aman seperti *Optimal Asymmetric Encryption Padding*, dan Laboratorium RSA telah merilis versi terbaru dari PKCS #1 yang tidak lemah terdapat serangan ini.

6 Banyak prosesor yang memakai sebuah *branch predictor*, *branch predicto* adalah*r* sebuah arsitektur komputer yang merupakan bagian dari prosesor yang memutuskan apakah sebuah percabangan kondisional dari sebuah instruksi dari program sebaiknya diambil atau tidak. Biasanya prosesor ini juga mengimplementasikan *Simulteneous Multithreading* (SMT). *Branch Prediction Analysis* menyerang menggunakan suatu proses memata-matai untuk mencari kunci privat yang dipakai oleh prosesor ini.

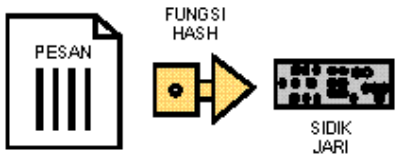
   Penyerangan seperti ini membutuhkan proses memata-matai untuk menjalankan mesin yang sama dengan mesin yang digunakan dalam memroses kunci privat tetapi proses ini tidak memerlukan kelebihan apapun dari sistem yang diserang.

3.4. Pemakaian Fungsi *Hash* Satu Arah untuk Menjamin Keamanan Data

Pada suatu ilustrasi diberikan misalkan Andi ingin mengirimkan surat bukti pembayaran kepada Budi sebesar 1 juta rupiah, namun entah bagaimana caranya Joni dapat membobol sandi dan menambahkan satu angka nol di belakang nominal 1.000.000, yang akan mejadikan nominal tersebut menjadi 10 juta rupiah. Di mata Budi yang menjadi penerima surat, pesan tersebut haruslah utuh dan tidak diubah-ubah oleh siapapun, termasuk Joni, Budi dan Andi, juga dari gangguan pada transmisi sekalipun. Hal ini dapat dilakukan dengan menggunakan fungsi *hash* satu arah (*one-way hash function*), yang terkadang disebut juga sebagai sidik jari (*fingerprint*), *message integrity check, atau manipulation detection code.*

Saat Andi ingin mengirimkan pesannya, dia harus membuat sidik jari dari pesan yang akan dikirim untuk Bobi. Pesan yang akan di-*hash* disebut *pre-image*, sedangkan keluarannya yang memiliki nilai tetap, disebut *hash-value* (nilai *hash*). Kemudian melalui saluran komunikasi yang aman, dia mengirimkan sidik jarinya kepada Bobi. Setelah Bobi menerima pesan Andi (tidak peduli melalui jalur komunikasi manapun) Bobi kemudian juga membuat sidik jari dari pesan yang telah diterimanya dari Andi. Kemudian Bobi membandingkan sidik jari yang dibuatnya dari Andi. Jika kedua sidik jari itu identik, maka Bobi dapat yakin bahwa pesan itu utuh dan tidak diubah-ubah sejak dibuatkan sidik jari yang diterima Bobi. Jika pesan pembayaran 1

juta rupiah diubah menjadi 10 juta rupiah, maka akan menghasilkan nilai *hash* yang berbeda.



Fungsi *hash* untuk membuat sidik jari tersebut dapat diketahui oleh siapapun, tidak terkecuali, sehingga siapapun dapat memeriksa keutuhan dokumen atau suatu pesan tertentu. Tidak ada algoritma rahasia dan umumnya tidak ada pula kunci rahasia.

Jaminan keamanan dari sidik jari berasal dari pernyataan bahwa hampir tidak ada dua *pre-image* yang memiliki *hash-value* yang sama. Hal inilah yang disebut dengan sifat *collision free* dari suatu fungsi *hash* yang baik. Selain itu sangat sulit untuk membuat suatu *pre-image* jika hanya diketahui *hash-value*-nya saja.

Contoh algoritma fungsi *hash* satu arah adalah MD-5 dan SHA. *Message authentication code* (MAC) adalah salah satu variasi dari fungsi *hash* satu arah, hanya saja selain *pre-image,* sebuah kunci rahasia juga menjadi input bagi fungsi MAC.

4. Contoh Nyata Pemakaian RSA

Salah satu contoh nyata pemakaian sistem penyandian RSA dalam kehidupan sehari-hari adalah dalam pemakaian *signature* atau tanda tangan digital dalam surat elektronik dam untuk autentikasi 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, Andi ingin mengirim sebuah surat elektronik yang dilengkapi dengan tanda tangan elektronik kepada Bobi. Pertama-tama Andi mengkomputasi tanda tangan elektroniknya, S, untuk surat elektronik M, dengan menggunakan kunci D_B. Sedemikian rupa sehingga sesuai dengan persamaan

$$S = D_B (M)$$

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

$$M = E_B (S)$$

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

Andi yang telah mengirim surat elektronik kepada Bobi tidak dapat menyangkal lagi bahwa dia telah mengirimkan sebuah surat elektronik kepada Bobi. Karena tidak ada orang lain yang dapat membuat $S = D_B (M)$. Dan Bobi pun dapat membuktikan bahwa memang surat elektronik tersebut asli dari Andi 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 Andi ingin mengirim pesan kepada Bobi. Andi 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 Bobi menerima pesan yang telah "ditandatangani", Bobi 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 Bobi dapat mengetahui bahwa pemilik dari pesan tersebut adalah Andi, dan pesan pun tidak pernah diubah sepanjang pengiriman.

Sehingga pada akhirnya Bobi 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.

5. Keuntungan dan Kerugian Memakai RSA

Dalam pemakaiannya, RSA tentu mempunyai beberapa keuntungan dan kerugian. Pada bagian ini kita akan membahas hal-hal ini.

Keuntungan utama 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 kemingkinan dimana penyerang dapat mencari tahu kunci rahasia tersebut saat proses pengiriman.

Keuntungan utama lainnya 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. Sebagai hasilnya, pengirim dapat menolak pesan autentikasi sebelumnya dengan cara membuktikan bahwa rahasia yang dibagikan bagaimanapun caranya disetujui oleh pihak lain yang berbagi rahasia tersebut. Contohnya, sistem autentikasi kunci rahasia Cerberus melibatkan sebuah basis data pusat yang menyimpan salinan dari kunci-kunci rahasai 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*.

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.

6. Pemakaian RSA Pada Saat Ini

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. Kesimpulan

Kesimpulan yang dapat diambil dari studi pemakaian RSA dalam penyandian adalah:

1. RSA merupakan metode penyandian yang masih kokoh untuk mengatasi masalah keamanan dalam pengiriman data pada suatu jaringan pada media elektronik
2. Dari segi teknis penghitungan, sistem RSA mempunyai cara enkripsi yang mudah, tetapi jika sudah dienkripsi, data yang terenkripsi sulit untuk dibobol jika hanya mempunyai kunci publiknya saja
3. 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
4. Dalam proses pembuatan kunci publik dan kunci privat, terdapat beberapa faktor yang menjadi pertimbangan, yaitu ukuran dari kunci, penentuan nilai p dan q agar sulit untuk dibobol, dan kemungkinan-kemungkinan kelemahan yang dapat diketahui saat data selesai dienkripsi
5. Pada 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

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

[2] Kessler, Gary C. 1998. *An Overview of Cryptography*. Auerbach.

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

[4] Wikipedia. 2006. http://wikipedia.org, waktu akses: 28 Desember 2006, 17.00

[5] http://www.muppetlabs.com/~breadbox/txt/rsa.html#17, waktu akses: 28 Desember 2006, 17.00

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