Teknik-Teknik Kriptanalisis Pada RSA

Felix Arya[1], Peter Paulus[2], dan Michael Ivan Widyarsa[3]

Departemen Teknik Informatika
Institut Teknologi Bandung
Jalan Ganesha 10 Bandung 40132

E-mail : if12039@students.if.itb.ac.id[1], if12040@students.if.itb.ac.id[2],
if12056@students.if.itb.ac.id[3]

Abstrak

Algoritma RSA, yang dikembangkan pada tahun 1976, merupakan algoritma kriptografi kunci publik paling populer saat ini. Algoritma ini menjadi populer oleh karena sampai saat ini, sistem kriptografi yang menggunakan algoritma ini masih sulit untuk dipecahkan. Walaupun demikian, terdapat beberapa teknik yang dapat digunakan memecahkan sistem kriptografi tersebut jika didapatkan bahan-bahan yang dibutuhkan, misalnya waktu pembangkitan kunci, dan sebagainya. Pada makalah ini kami membahas teknik-teknik yang dapat digunakan untuk melakukan kriptanalisis pada RSA. Teknik-teknik serangan ini dibagi menjadi serangan matematis, serangan pemfaktoran, serangan *timing*, dan serangan implementasi. Kesimpulan yang didapatkan adalah bahwa algoritma RSA ini masih merupakan algoritma yang sulit untuk dipecahkan, walaupun secara teori mungkin untuk dipecahkan dengan usaha yang sangat besar.

***Kata kunci**: RSA, kriptanalisis*

1. Pendahuluan

RSA adalah algoritma kriptografi kunci publik paling populer yang digunakan saat ini. Algoritma ini diciptakan pada tahun 1976 oleh Rivest, Shamir, dan Adleman. Popularitas dari algoritma ini bersumber pada tingkat keamanannya yang sangat baik, bersumber dari sulitnya pemfaktoran terhadap sebuah bilangan integer besar.

Sampai saat ini, berbagai pendekatan telah diusulkan sebagai metode kriptanalisis terhadap RSA. Menurut Statica[7)], kriptanalisis pada RSA bisa dibagi tiga, yaitu:

- Metode Faktorisasi
- Serangan *Brute-Force* dan *Timing*
- Serangan Implementasi

Selain dari tiga jenis serangan ini, masih terdapat beberapa metode lain, di antaranya serangan matematis, *rubber-hose attack*, dan lain-lain.

Di dalam makalah ini, akan dibahas secara singkat mengenai jenis-jenis kriptanalisis pada RSA, beserta dengan asumsi masing-masing dan *feasibility*-nya. Dari pembahasan ini, akan ditarik suatu kesimpulan mengenai pendekatan yang paling efektif untuk memecahkan RSA.

2. Ruang lingkup

Di dalam makalah ini akan dimuat pembahasan mengenai berbagai jenis kriptanalisis pada RSA, dibagi dalam tiga metode, yaitu faktorisasi, *brute-force and timing*, dan implementasi. Pembahasan ini

akan mencakup deskripsi singkat, asumsi masing-masing metode, dan *feasibility*.

3. Metode Kriptanalisis pada RSA

3.1 Serangan-serangan Matematis pada Algoritma RSA

- **Serangan GCD (*Greatest Common Divisor*)**

Jika terdapat dua buah pesan yang dependen linear (m_1 dan m2, $m_2 = m_1 + \Delta$) dienkripsi dengan sistem kriptografi RSA yang sama, maka dimungkinkan untuk mendapatkan kedua pesan tersebut. Dengan cara mendapatkan kedua *ciphertext* $c_1 = m_1^e \bmod n$ dan $c_2 = m_2^e \bmod n$, didefinisikan dua buah fungsi polynomial *P* dan *Q* yang didefinisikan sebagai berikut:

$$P(x) = x^2 - c_1 (\bmod n) \quad (1)$$

$$Q(x) = (x + \Delta)^2 - c_2 (\bmod n) \quad (2)$$

Karena pesan m_1 merupakan sebuah akar dari *P* dan *Q*, m_1 akan menjadi akar dari fungsi gcd(*P*,*Q*), yang kemungkinan besar merupakan polinom berderajat satu. Dengan menyelesaikan fungsi polinom ini, akan didapatkan nilai m_1 dan $m_2 = m_1 + \Delta$ [3].

- **Serangan *Common Modulus***

Serangan ini dapat dilakukan jika diketahui dua buah *ciphertext* hasil enkripsi sebuah pesan *m* yang sama dengan dua buah kunci enkripsi yang berbeda, yaitu $c_1 = m^{e_1} \bmod n$ dan $c_2 = m^{e_2} \bmod n$. Selain itu, nilai *n* yang digunakan dalam enkripsi harus sama, dan kedua kunci enkripsi e_1 dan e_2 harus relatif prima. Karena faktor pembagi terbesar e_1 dan e_2 = 1, maka dengan menggunakan algoritma *euclidean*, terdapat dua buah bilangan $r, s \in Z$ sedemikian sehingga:

$$re_1 + se_2 = 1 \quad (3)$$

Dengan demikian *m* dapat dihitung dengan cara sebagai berikut:

$$m = m^{re_1 + se_2} = c_1^r c_2^s \bmod n \quad [3] \quad (4)$$

3.2 Serangan Faktorisasi

Penyerangan ini bertujuan untuk memfaktorkan nilai *n* menjadi dua buah faktor primanya yaitu *p* dan *q*. Jika *p* dan *q* berhasil difaktorkan, fungsi euler $\varphi(n) = (p-1)(q-1)$ akan dapat dikomputasi dengan mudah, dan kemudian kunci privat $d = e^{-1} \bmod \varphi(n)$ dapat segera dihitung [8].

Pemfaktoran ini pada dasarnya sangat mungkin untuk dilakukan, dan jika berhasil, maka skema enkripsi RSA yang telah diaplikasikan akan runtuh. Akan tetapi penerapan teknik ini tidaklah mudah, oleh karena algoritma RSA ini memanfaatkan *integer factorization problem*, dimana memfaktorkan sebuah bilangan menjadi faktor-faktor primanya merupakan hal yang sulit (kompleksitasnya dalam orde non-polinomial) [5]. Selain itu, *p* dan *q* yang dipilih biasanya merupakan bilangan yang sangat besar, sehingga proses faktorisasi akan memakan waktu yang sangat lama.

Pada saat ini, algoritma pemfaktoran tercepat adalah *General Number Field Sieve*. Kompleksitas algoritma ini masih dalam orde eksponensial, yaitu $\exp((c + o(1))n^{1/3} \log^{2/3} n)$ dimana n merupakan jumlah bit pada *integer* yang difaktorkan, dan c < 2 [7]. Hal ini membuat RSA merupakan algoritma kriptografi yang aman, selama nilai *n* yang digunakan sangat

besar dan belum ditemukan algoritma pemfaktoran dalam orde polinomial.

Jenis serangan ini dapat dilakukan hanya dengan bermodalkan nilai *n* dan nilai *w* (yang memang tidak dirahasiakan). Serangan ini cocok untuk dilakukan jika waktu pemfaktoran bukan merupakan masalah krusial dalam usaha mencari kunci dekripsi.

Salah satu cara untuk mengatasi serangan faktorisasi ini adalah untuk membuat panjang faktor prima *p* dan *q* tidak seimbang. Dengan memilih nilai *p* yang jauh lebih kecil daripada *q*, dan nilai plainteks *T* masih berada dalam jangkauan $0 \leq T < p$, serangan menggunakan algoritma faktorisasi dapat "ditipu" dengan membuat seakan-akan terdapat dua faktor prima yang panjangnya hampir sama [8].

3.3 Serangan *Brute-Force* dan *Timing*

Metode *Brute-Force* adalah metode yang kriptanalisis yang paling pasti dalam mendapatkan hasil. Dengan mencoba setiap nilai kunci yang mungkin, seorang krriptanalis bisa mendekripsi ciphertext seperti apapun. Masalah utama dari metode ini adalah waktu komputasi yang sangat lama. Pencarian nilai kunci dengan *exhaustive search* adalah mustail dilakukan dengan teknologi saat ini. Walaupun demikian, beberapa cara untuk memperkecil ruang pencarian telah diusulkan, di antaranya adalah dengan menggunakan metode *Timing*.

Metode *Timing* ini memanfaatkan lamanya waktu komputasi dalam proses dekripsi sebuah cipher RSA. Dalam sebuah proses dekripsi RSA, pengguna akan melakukan proses $m = c^d \bmod n$ dimana nilai *n* adalah publik dan *cipher c* bisa didapat dengan melakukan penyadapan pada saluran komunikasi. Untuk mendapatkan statistik waktu yang diperlukan, kriptanalis harus bisa mendapatkan waktu komputasi pengguna untuk sejumlah nilai *c* dengan kunci privat tetap. Apabila dalam setiap operasi dekripsi pengguna menggunakan kunci privat *d* yang berbeda, maka metode *timing* ini tidak bisa dilakukan. Walaupun demikian komputasi kunci privat yang berbeda untuk setiap proses dekripsi sangat mahal dan tidak banyak pihak yang sanggup melakukannya tanpa terlalu mengganggu performansi. Selain itu, dalam metode ini kriptanalis harus mengetahui desain dan arsitektur dari sistem sasaran.

Dengan menyadap protokol komunikasi dari sasaran, kriptanalis bisa mendapatkan statistik yang diperlukan dari waktu respon sistem sasaran terhadap setiap pesan. Pemanfaatan statistik ini dalam mendapatkan kunci privat bergantung pada fungsi eksponensiasi modular yang digunakan oleh sistem sasaran. Untuk beberapa fungsi eksponensiasi modular, kriptanalis bsia mendapatkan nilai-nilai bit eksponen dan menghitung kunci privat. Contoh penggunaan tehnik *timing* ini bisa dilihat di [4].

Menurut Kocher, bila tidak terjadi banyak kesalahan pada penentuan statistik waktu dan desain sistem sasaran diketahui dengan jelas, metode *timing* ini bisa mendapatkan kunci publik dengan cukup cepat. Pada percobaan dengan RSAREF, dalam keadaan optimal hanya perlu dilakukan empat kali evaluasi untuk setiap operasi yang dilakukan oleh sistem sasaran. Jumlah operasi ini bergantung pada jenis fungsi eksponensiasi modular yang digunakan pada sistem.

Walaupun demikian, metode *Timing* ini memiliki kelemahan pada penentuan statistik waktu dan penentuan bilangan eksponen. Kesalahan pada penentuan bilangan eksponen ini akan memperlama waktu komputasi. Selain itu, penentuan statistik waktu juga bisa dipengaruhi oleh faktor-faktor luar seperti *load* yang berubah pada

sistem sasaran atau fluktuasi *load* pada saluran komunikasi. Selain itu, sistem sasaran bisa melakukan *blinding* seperti yang digunakan oleh RSA Securities[6] untuk mencegah waktu komputasinya terbaca.

3.4 *Implementation Attack*

Implementation attack, atau disebut juga protocol attack adalah penyerangan terhadap sebuah kriptosistem yang mengeksploitasi desain dan penggunaan yang buruk terhadap kriptosistem tersebut. Implementation attack tidak melakukan kriptanalisis terhadap algoritma dari RSA[8].

Salah satu contoh implementation attack terhadap RSA adalah penyerangan terhadap sebuah varian dari RSA yang disebut *RSA for Paranoids* [6]. Misalkan e dan n adalah kunci public Bob. Prosedur penyerangan adalah sebagai berikut. Alice mengenkripsi message m, dimana m > p, menjadi ciphertext dengan kunci publik Bob. Bob kemudian mendekripsi ciphertext tersebut menjadi m'. Karena m > p maka m' ≠ m. Jika Alice bisa mendapatkan m', p dapat diperoleh dengan menghitung PBB(m – m', n). Satu hal yang paling penting dan merupakan syarat yang harus dipenuhi agar implementation attack ini dapat dijalankan adalah Alice harus mendapatkan hasil dekripsi dari ciphertext yang dikirimnya [3].

Sebuah contoh lain dari implementation attack adalah chosen text attack [1]. Dalam serangan ini penyerang akan mengenkripsi pesan M dengan kunci enkripsi orang yang akan diserang menjadi C. Kemudian penyerang akan memilih bilangan acak X dan akan membangkitkan C' berdasarkan rumus sebagai berikut

$$C' = C \times X^E \pmod N \qquad (5)$$

Orang yang diserang kemudian akan mendekripsi C' menjadi $(C')^D$. Jika penyerang bisa mendapatkan $(C')^D$, maka penyerang dapat mendapatkan kunci dekripsi D dengan rumus sebagai berikut.

$$(C')^D = C^D \times X = M \times X \pmod N \qquad (6)$$

Seperti contoh sebelumnya, chosen text attack ini memiliki syarat penyerang mendapatkan hasil dekripsi dari pesan yang ia kirimkan.

4. Kesimpulan

Dari serangan-serangan yang telah dibahas, bisa disimpulkan bahwa serangan faktorisasi adalah jenis serangan yang menggunakan paling sedikit asumsi dan kondisi awal. Serangan faktorisasi ini bisa dilakukan tanpa harus mengetahui desain sistem dan tidak perlu memiliki cipherteks. Walaupun demikian, karena belum adanya algoritma pemfaktoran yang mampu melakukan pemfaktoran bilangan integer besar secara cepat, cara serangan ini akan memakan waktu yang lama, sehingga tidak *feasible* untuk digunakan.

Metode-metode matematis, seperti GCD dan *common modulus* dapat memecahkan RSA, tapi hanya dengan kondisi-kondisi yang sudah ditentukan, dan memerlukan waktu yang cukup lama. Selain itu, penggunaannya terbatas pada pesan-pesan yang memenuhi kondisi bersangkutan.

Metode *timing* dan *brute-force* juga dapat menghitung kunci enkripsi RSA apabila statistik waktu komputasi dari komputer sasaran bisa didapat. Statistik waktu ini bisa didapat dengan penyadapan pada saluran komunikasi, tetapi masih terdapat maslaah pada akurasi dari penentuan statistik ini. Hal ini karena *load* dari mesin sasaran dan *load* dari saluran komunikasi bisa mempengaruhi

akurasi penentuan statistik waktu, dan pada akhirnya akan mengakibatkan pada tebakan yang salah untuk bilangan eksponen sehingga waktu komputasi akan lama.

Dari seluruh metode serangan yang telah disajikan, semuanya masih belum ada yang sanggup memecahkan RSA dengan waktu dan usaha yang cukup *feasible*. Apabila sistem sasaran memilih bilangan integer besar secara hati-hati dan menjaga keamanan komunikasi, algoritma RSA masih sangat sulit untuk ditembus, kecuali dengan metode *rubber-hose attack.*

Daftar Pustaka

[1] G. Davida, *Chosen signature cryptoanalyses of the RSA (MIT) public key cryptosystem*, Technical Report TR-CS-82-2, Department of Electrical Engineering and Computer Science, University of Wisconsin, Milwaukee, WI, 1982.

[2] H. Gilbert, D. Gupta, A. Odlyzko and J.J. Quisquater, *Attacks on Shamir's 'RSA for Paranoids'*, xxx, 1998.

[3] M. Joye dan J.J. Quisquater, Cryptanalysis Of RSA-Type Cryptosystems: A Visit, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 38, pp. 21-31, American Mathematical Society, 1998.

[4] P.C. Kocher. *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems,* Cryptography Research, Inc, 1999.

[5] A.Menezez, *et al*. *Handbook of Applied Cryptography*, CRC Press, 1996.

[6] RSA Security Laboratory, Inc. www.rsasecurity.com/rsalabs, diakses tanggal 2 Januari 2006 pukul 22:03.

[7] A.Shamir, *RSA For Paranoids*. RSA Laboratories Volume 1, Number 3 - Autumn, 1995.

[8] R.Statica, *New Approach To Cryptanalysis of RSA*, New Jersey Institute of Technology, 2003.