Kajian Perhitungan dan Penerapan Algoritma RSA pada Proses Pengamanan Data

Sriyono[1)], Atiqah Meutia Hilda[2)]
[1,2)]Program Studi Teknik Informatika, Fakultas Teknik,
Universitas Muhammadiyah Prof. Dr. Hamka, Jakarta.
Jalan Limau II, Kebayoran Baru, Jakarta 12130. Indonesia.
Telp: +62-21-7256659, Fax: +62-21-7256659, Hp.+6281310770089
Email: ft.uhamka@yahoo.com

Abstrak

Algoritma RSA merupakan salah satu kriptografi asimetri, yang menggunakan dua kunci berbeda yaitu kunci public (public key) dan kunci pribadi (private key). Kunci publik bersifat tidak rahasia sedangkan kunci pribadi bersifat rahasia. Dalam kriptografi asimetri, dua kunci tersebut diatur sehingga memiliki hubungan dalam suatu persamaan aritmatika modulo. Dalam dunia kriptografi terdapat dua profesi, yakni kriptografer dan kriptanalis. Seorang kriptografer akan berusaha untuk menciptakan algoritma kriptografi sekompleks mungkin, sedangkan seorang kriptanalis akan berusaha untuk memecahkan algoritma tersebut agar dapat mengetahui pesan yang dirahasiakan. Kriptanalis berperan sebagai penguji dari suatu algoritma kriptografi atau menjadi penyerang untuk mengetahui informasi rahasia yang sesungguhnya tidak berhak untuk ia ketahui. Suatu serangan dapat digolongkan sebagai serangan pasif, yakni serangan yang tidak merubah isi dari pesan, atau serangan aktif, yang dapat mengubah isi dari suatu pesan. Meskipun kriptografi asimetri menggunakan kunci publik yang dapat melewati saluran tanpa pengamanan, namun bukan berarti kriptografi ini tidak bebas dari serangan. Metode timing attacks memanfaatkan proses perhitungan suatu kunci, sehingga jika terdapat suatu kebocoran selama pemrosesan, maka informasi tersebut dapat dimanfaatkan untuk mengetahui kunci pribadi yang digunakan. Pencegahan terhadap timing attacks dapat dilakukan dengan cara memodifikasi persamaan aritmatika modulo yang digunakan, sehingga menambah kompleksitas dari perhitungan. Pada makalah ini akan dibahas bagaimana algoritma RSA diterapkan dan diaplikasikan serta pencegahannya terhadap adanya timing attacks.

***Kata kunci**: penerapan, algoritma RSA, pencegahan, timing, attack*

1 PENDAHULUAN

Dari sekian banyak algoritma kriptografi yang pernah dibuat, algoritma yang paling populer adalah algoritma RSA. Algoritma RSA dibuat oleh 3 orang peneliti dari MIT (*Massachussets Institute of Technology*) pada tahun 1976, yaitu: Ron (**R**)ivest, Adi (**S**)hamir, dan Leonard (**A**)dleman. Algorima RSA mendasarkan proses enkripsi dan dekripsinya pada konsep bilangan prima dan aritmatika modulo. Kunci enkripsi maupun kunci dekripsi keduanya harus berupa bilangan bulat. Kunci enkripsi tidak dirahasiakan dan diketahui umum (sehingga dinamakan juga kunci publik), namun kunci untuk dekripsi bersifat rahasia. Kunci dekripsi dibangkitkan dari beberapa buah bilangan prima bersama-sama dengan kunci enkripsi. Untuk menemukan kunci dekripsi, suatu bilangan non prima harus difaktorkan menjadi faktor primanya. Dalam kenyataannya, memfaktorkan bilangan non prima menjadi faktor primanya bukanlah pekerjaan yang mudah. Belum ada algoritma yang secara efisien yang dapat melakukan pemfaktoran tersebut. Semakin besar bilangan non primanya maka semakin sulit pula pemfaktorannya. Semakin sulit

pemfaktorannya, semakin kuat pula algoritma RSA.

Algoritma RSA merupakan salah satu kriptografi asimetri, yakni jenis kriptografi yang menggunakan dua kunci yang berbeda : kunci public (*public key*) dan kunci pribadi (*private key*). Dengan demikian, maka terdapat satu kunci, yakni kunci publik, yang dapat dikirimkan melalui saluran yang bebas, tanpa adanya suatu kemanan tertentu. Hal ini bertolak belakang dengan kriptografi simetri yang hanya menggunakan satu jenis kunci dan kunci tersebut harus terus terjaga keamanan serta kerahasiaannya. Dalam kriptografi asimetri, dua kunci tersebut diatur sedemikian sehingga memiliki hubungan dalam suatu persamaan aritmatika modulo.

Dalam dunia kriptografi terdapat dua profesi, yakni kriptografer dan kriptanalis (*cryptanalyst*). Seorang kriptografer akan berusaha untuk mencipatakan algoritma kriptografi sekompleks mungkin, sedangkan seorang kriptanalis akan berusaha untuk memecahkan algoritma tersebut agar dapat mengetahui pesan yang dirahasiakan (mendekripsi dari *ciphertext* menjadi *plaintext*). Kriptanalis dapat berperan sebagai penguji dari suatu algoritma kriptografi atau menjadi penyerang untuk mengetahui informasi rahasia yang sesungguhnya tidak berhak untuk ia ketahui. Suatu serangan dapat digolongkan sebagai serangan pasif (*passive attacks*), yakni serangan yang tidak merubah isi dari pesan, atau serangan aktif (*active attacks*), yang dapat mengubah isi dari suatu pesan. Meskipun kriptografi asimetri menggunakan kunci publik yang dapat melewati saluran tanpa pengamanan, namun bukan berarti kriptografi ini tidak bebas dari sernagan (*attack*). Metode *timing attacks* memanfaatkan proses perhitungan suatu kunci, sehingga jika terdapat suatu kebocoran selama pemrosesan, maka informasi tersebut dapat dimanfaatkan untuk mengetahui kunci pribadi yang digunakan. Pencegahan terhadap *timing attacks* dapat dilakukan dengan cara memodifikasi persamaan aritmatika modulo yang digunakan, sehingga menambah kompleksitas dari perhitungan.

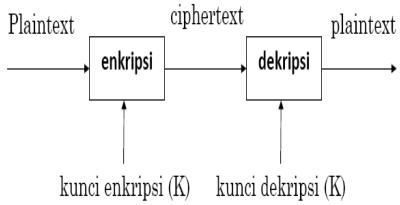
Pada makalah ini akan dibahas bagaimana algoritma RSA ini diterapkan dan diaplikasikan untuk menjamin keamanan data serta pencegahannya terhadap adanya *timing attacks*.

2 DASAR TEORI

Kriptografi adalah komputasi integer dengan Aritmatika Modulo. Operator yang digunakan pada aritmatika modulo adalah mod [1]. Operator mod memberikan sisa pembagian dari bilangan bulat. Misalkan a adalah bilangan bulat dan m adalah bilangan bulat > 0. Operasi a mod m memberikan sisa pembagian dari a dan m. Dapat dikatakan pula bahwa $a \bmod m = r$, sedemikian sehingga $a = mq + r$, dimana $0 \leq r < m$. Jika terdapat bilangan bulat a dan b sedemikian sehingga keduanya mempunyai sisa yang sama jika dibagi dengan bilangan bulat positif m, maka a dan b adalah kongruen dalam modulo m dan dilambangkan dengan $a \equiv b \pmod{m}$. Operator mod juga digunakan dalam persamaan berdasarkan teorema Fermat, yakni $a\ p-1 \equiv 1 \pmod{p}$, dengan p adalah bilangan prima dan a adalah bilangan bulat yang tidak habis dibagi dengan p. Aritmatika Modulo nantinya akan banyak dipergunakan pada pembahasan selanjutnya.

Kriptografi adalah seni dan ilmu dalam menuliskan pesan rahasia, artinya suatu informasi diubah sedemikian sehingga menjadi tidak dapat dimengerti oleh orang yang tidak diinginkan untuk mengetahui informasi tersebut. Namun, perubahan informasi tersebut harus dapat dikembalikan seperti semula (*reversible*) agar dapat dibaca oleh orang yang berhak[3]. Penyandiaan pesan tersebut selanjutnya dinamakan *cipher* atau *cryptosystem*. *Chiper* adalah sepasang fungsi yang tidak dapat dibalik, yakni $k\ f$ (*enciphering function*) dan k (*deciphering function*). Fungsi $k\ f$ memetakan elemen x dalam himpunan S menjadi elemen $k\ f\ (x)$ dalam himpunan T, sehingga mencari pemetaan balikan (*inverse*) menjadi sangat sulit tanpa mengetahui k'. Elemen dari S disebut sebagai *plaintext* dan elemen dari T disebut sebagai *ciphertext*. Fungsi $k\ g$ adalah balikan (*inverse*) dari $k\ f\ .\ k$' yang disebut juga sebagai *deciphering key* (kunci dekripsi). Jika $k = k$', atau k' sangatlah gampang untuk dihitung dengan memanfaatkan nilai k, maka kriptografi yang

dipakai disebut dengan *symmetric cryptography* (kriptografi simetri) dan kunci dari kriptografi ini disebut *secret key* (kunci rahasia). Namun, jika *k'* sangat sulit untuk diketahui, walaupun dengan mengetahui *k*, maka kriptografi yang dipakai disebut dengan *asymmetric cryptography* (kriptografi asimetri) dan *k* disebut sebagai *public key* (kunci publik) dan *k'* disebut dengan *private key* (kunci pribadi). Proses enkripsi dan dekripsi seperti ditunjukkan pada Gambar 1[4



Gambar 1.1 Proses enkripsi dan dekripsi menggunakan kunci publik dan kunci pribadi[6]

2.1 Kriptografi Simetri

Kriptografi simetri adalah metode enkripsi dimana pengirim dan penerima pesan memiliki kunci yang sama, atau dalam beberapa kasus kedua kunci berbeda namun mempunya relasi dengan perhitungan yang mudah. Studi modern terfokuskan pada *block cipher* dan *stream cipher* serta aplikasinya. *Block cipher* adalah aplikasi modern dari *Alberti's polyphabetic cipher*. *Block cipher* menerima masukan berupa blok *plaintext* dan sebuah kunci dan kemudian menghasilkan keluaran blok *ciphertext* dengan ukuran yang sama. Dikarenakan pesan yang dikirim hampir selalu lebih panjang dari *single block* (blok tunggal), maka diperlukan metode penggabungan beberapa blok. *Data Encryption Standard* (DES) dan *Advanced Encryption Standard* (AES) adalah contoh *block ciphers* yang dijadikan standar kriptografi oleh pemerintahan Amerika Serikat. Walaupun AES telah diresmikan sebagai standar kriptografi terbaru, namun DES, khususnya varian *triple-DES*, masih banyak digunakan sebagai enkripsi ATM, keamanan surat elektronik (*e-mail*), dan *secure remote access*. *Stream cipher* adalah lawan dari *block cipher*, yakni menciptakan arus kunci yang panjang dan sembarang (*arbitrarily long stream of key*) yang dikombinasikan dengan *plaintext* bit-per-bit (*bit by bit*) dan karakter-per-karakter (*character by character*). Pada *stream cipher*, arus keluaran dibangkitkan berdasarkan keadaan internal (*internal state*) yang berubah-ubah seiring dengan jalannya *cipher*. Perubahaan tersebut diatur oleh kunci dan dibeberapa *stream cipher* diatur pula oleh *plaintext cipher*. *RC4* adalah contoh dari *stream cipher*. [4]

Dalam kriptografi asimetri, kunci publik dapat secara bebas disebarluaskan, sedangkan kunci pribadi harus senantiasa dijaga kerahasiaa nnya. Kunci publik digunakan untuk enkripsi, sedangkan kunci pribadi digunakan untuk dekripsi. Diffie dan Hellman membuktikan bahwa kriptografi asimetri adalah mungkin dengan menerapkan protokol pertukaran kunci Diffie-Hellman. [4]

3 PEMBAHASAN

3.1 Perhitungan dan PenerapanAlgoritma RSA

Keamanan algoritma RSA terletak pada sulitnya memfaktorkan bilangan yang besar menjadi faktor-faktor prima. Pemfaktoran dilakukan untuk memperoleh kunci pribadi. Selama pemfaktoran bilangan besar menjadi faktor-faktor prima belum ditemukan algoritma yang mangkus, maka selama itu pula keamanan algoritma RSA tetap terjamin. Besaran-besaran yang digunakan pada algoritma RSA: [5]

1. p dan q bilangan prima (rahasia)
2. $r = p \cdot q$ (tidak rahasia)
3. $\phi(r) = (p-1)(q-1)$ (rahasia)
4. PK (kunci enkripsi) (tidak rahasia)
5. SK (kunci dekripsi) (rahasia)
6. X (plainteks) (rahasia)

7. Y (cipherteks) (tidak rahasia)

Algoritma RSA didasarkan pada teorema Euler yang menyatakan bahwa

$$a^{\phi(r)} \equiv 1 \pmod{r} \quad \text{.....(1)}$$

dengan ketentuan,

1. a harus relatif prima terhadap r
2. $\phi(r) = r(1 - 1/p_1)(1 - 1/p_2) \ldots (1 - 1/p_n)$, yang dalam hal ini $p_1, p_2, \ldots, p_n$ adalah faktor prima dari r.

$\phi(r)$ adalah fungsi yang menentukan berapa banyak dari bilangan-bilangan 1, 2, 3, …, r yang relatif prima terhadap r. Berdasarkan sifat $a^m \equiv b^m \pmod{r}$ untuk m bilangan bulat ≥ 1, maka persamaan (1) dapat ditulis menjadi

$$a^{m\phi(r)} \equiv 1^m \pmod{r}$$

atau

$$a^{m\phi(r)} \equiv 1 \pmod{r} \quad \text{.....(2)}$$

Bila a diganti dengan X, maka persamaan (2) menjadi

$$X^{m\phi(r)} \equiv 1 \pmod{r} \quad \text{.....(3)}$$

Berdasarkan sifat $ac \equiv bc \pmod{r}$, maka bila persamaan (3) dikali dengan X menjadi:

$$X^{m\phi(r)+1} \equiv X \pmod{r} \quad \text{.....(4)}$$

yang dalam hal ini X relatif prima terhadap r.

Misalkan SK dan PK dipilih sedemikian sehingga

$$SK \cdot PK \equiv 1 \pmod{\phi(r)} \quad \text{.....(5)}$$

atau

$$SK \cdot PK = m\phi(r) + 1 \quad (6)$$

Sulihkan (6) ke dalam persamaan (4) menjadi:

$$X^{SK \cdot PK} \equiv X \pmod{r} \quad \text{.....(7)}$$

Persamaan (7) dapat ditulis kembali menjadi

$$(X^{PK})^{SK} \equiv X \pmod{r} \quad \text{.....(8)}$$

yang artinya, perpangkatan X dengan PK diikuti dengan perpangkatan dengan SK menghasilkan kembali X semula. Berdasarkan persamaan (8), maka enkripsi dan dekripsi dirumuskan sebagai berikut:

$$E_{PK}(X) = Y \equiv X^{PK} \bmod r \quad \text{.....(9)}$$
$$D_{SK}(Y) = X \equiv Y^{SK} \bmod r \quad \text{.....(10)}$$

Karena $SK \cdot PK = PK \cdot SK$, maka enkripsi diikuti dengan dekripsi ekivalen dengan dekripsi diikuti enkripsi:

$$E_{SK}(D_{SK}(X)) = D_{SK}(E_{PK}(X)) \equiv X^{PK} \bmod r \quad \text{.....(11)}$$

Oleh karena $X^{PK} \bmod r \equiv (X + mr)^{PK} \bmod r$ untuk sembarang bilangan bulat m, maka tiap plainteks X, $X + r$, $X + 2r$, …, menghasilkan cipherteks yang sama. Dengan kata lain, transformasinya dari banyak ke satu. Agar transformasinya satu-ke-satu, maka X harus dibatasi dalam himpunan $\{0, 1, 2, \ldots, r - 1\}$ sehingga enkripsi dan dekripsi tetap benar seperti pada persamaan (9) dan (10).

Prosedur Membuat Pasangan Kunci

1. Pilih dua buah bilangan prima sembarang, p dan q.

2. Hitung $r = p \cdot q$. Sebaiknya $p \neq q$, sebab jika $p = q$ maka $r = p^2$ sehingga p dapat diperoleh dengan menarik akar pangkat dua dari r.

3. Hitung $\phi(r) = (p - 1)(q - 1)$.

4. Pilih kunci publik, PK, yang relatif prima terhadap $\phi(r)$.

5. Bangkitkan kunci rahasia dengan menggunakan persamaan (5), yaitu $SK \cdot PK \equiv 1 \pmod{\phi(r)}$.

Perhatikan bahwa $SK \cdot PK \equiv 1 \pmod{\phi(r)}$ ekivalen dengan $SK \cdot PK = 1 + m\phi(r)$, sehingga SK dapat dihitung dengan

$$SK = \frac{1 + m\phi(r)}{PK} \quad \text{.....(12)}$$

Akan terdapat bilangan bulat m yang menyebabkan memberikan bilangan bulat SK. PK dan SK dapat dipertukarkan urutan pembangkit- annya. Jika langkah 4 diganti dengan "Pilih kunci rahasia, SK, yang …", maka pada langkah 5 kita menghitung kunci publik dengan rumus yang sama.

Contoh 1. Misalkan p = 47 dan q = 71 (keduanya prima). Selanjutnya, hitung nilai

$$r = p \cdot q = 3337$$

dan

$$\phi(r) = (p-1)(q-1) = 3220$$

Pilih kunci publik SK = 79, karena 79 relatif prima dengan 3220. PK dan r dapat dipublikasikan ke umum.

Selanjutnya akan dihitung kunci dekripsi SK seperti yang dituliskan pada langkah instruksi 5 dengan menggunakan persamaan (12),

$$SK = \frac{1 + (m \times 3220)}{79}$$

Dengan mencoba nilai-nilai m = 1, 2, 3, …, diperoleh nilai SK yang bulat adalah 1019. Ini adalah kunci dekripsi yang harus dirahasiakan.

Proses Enkripsi

- Plainteks disusun menjadi blok-blok x_1, x_2, …, sedemikian sehingga setiap blok merepresentasikan nilai di dalam rentang 0 sampai $r - 1$.
- Setiap blok x_i dienkripsi menjadi blok y_i dengan rumus

$$y_i = x_i^{PK} \bmod r$$

Proses Dekripsi

- Setiap blok cipherteks y_i didekripsi kembali menjadi blok x_i dengan rumus

$$x_i = y_i^{SK} \bmod r$$

Contoh 2. Misalkan plainteks yang akan dienkripsikan adalah

X = HARI INI

atau dalam sistem desimal (pengkodean ASCII) adalah

7265827332737873

Pecah X menjadi blok yang lebih kecil, misalnya X dipecah menjadi enam blok yang berukuran 3 digit:

$x_1 = 726$ $x_4 = 273$
$x_2 = 582$ $x_5 = 787$
$x_3 = 733$ $x_6 = 003$

Nilai-nilai x_i ini masih terletak di dalam rentang 0 sampai 3337 – 1 (agar transformasi menjadi satu-ke-satu).

Blok-blok plainteks dienkripsikan sebagai berikut.

$726^{79} \bmod 3337 = 215 = y_1$
$582^{79} \bmod 3337 = 776 = y_2$
$733^{79} \bmod 3337 = 1743 = y_3$
$273^{79} \bmod 3337 = 933 = y_4$
$787^{79} \bmod 3337 = 1731 = y_5$
$003^{79} \bmod 3337 = 158 = y_6$

Jadi, cipherteks yang dihasilkan adalah

Y = 215 776 1743 933 1731 158.

Dekripsi dilakukan dengan menggunakan kunci rahasia

SK = 1019

Blok-blok cipherteks didekripsikan sebagai berikut:

$215^{1019} \bmod 3337 = 726 = x_1$
$776^{1019} \bmod 3337 = 582 = x_2$
$1743^{1019} \bmod 3337 = 733 = x_3$, dst.nya

Blok plainteks yang lain dikembalikan dengan cara yang serupa. Akhirnya kita memperoleh kembali plainteks semula

P = 7265827332737873

yang dalam karakter ASCII adalah

P = HARI INI.

Kekuatan dan Keamanan RSA

Keamanan algoritma *RSA* terletak pada tingkat kesulitan dalam memfaktorkan bilangan non prima menjadi faktor primanya, yang dalam hal ini $r = p \times q$.

Sekali r berhasil difaktorkan menjadi p dan q, maka $\phi(r) = (p - 1)(q - 1)$ dapat dihitung. Selanjutnya, karena kunci enkrispi *PK* diumumkan (tidak rahasia), maka kunci dekripsi *SK* dapat dihitung dari persamaan $PK \cdot SK \equiv 1 \pmod{\phi(r)}$.

Penemu algoritma *RSA* menyarankan nilai p dan q panjangnya lebih dari 100 digit. Dengan demikian hasil kali $r = p \times q$ akan berukuran lebih dari 200 digit. Menurut Rivest dan kawan-kawan, uasaha untuk mencari faktor bilangan 200 digit membutuhkan waktu komputasi selama 4 milyar tahun! (dengan asumsi bahwa algoritma pemfaktoran yang digunakan adalah algoritma yang tercepat saat ini dan komputer yang dipakai mempunyai kecepatan 1 milidetik).

Untunglah algoritma yang paling mangkus untuk memfaktorkan bilangan yang besar belum ditemukan. Inilah yang membuat algoritma *RSA* tetap dipakai hingga saat ini. Selagi belum ditemukan algoritma yang mangkus untuk memfaktorkan bilangan bulat menjadi faktor primanya, maka algoritma *RSA* tetap direkomendasikan untuk menyandikan pesan.

3.2 Serangan Terhadap Kriptografi

Serangan terhadap kriptografi berarti usaha untuk menemukan kunci atau *plaintext* dari *ciphertext*. Untuk lebih jelasnya, pembahasan mengenai serangan terhadap kriptografi dimulai dari penjelasan tentang kriptanalisis (*cryptanalysis*).

3.2.1 Kriptanalisis

Tujuan dari kriptanalisis adalah untuk mengetahui kelemahan dari sebuah skema kriptografi. Kriptanalisis dapat diterapkan oleh seseorang yang bermaksud mengetahui informasi rahasia atau bias juga diterapkan oleh perancang suatu sistem keamanan. Terdapat beberapa variasi dalam serangan kriptanalisis dan dapat diklasifikasikan dalam beberapa cara. Dalam serangan *ciphertext-only*, kriptanalisis mempunyai akses hanya kepada *ciphertext* (kriptosistem modern yang bagus pada umumnya kebal terhadap serangan ini). Pada serangan *known-plaintext*, kriptanalis mempunyai akses kepada *ciphertext* dan *plaintext* yang berkorespondensi. Pada serangan *chosen-plaintext*, kriptanalis memilih sebuah *plaintext* dan mempelajari *ciphertext* yang berkorespondensi. Terakhir, pada serangan *chosen-ciphertext*, kriptanalis memilih sebuah *ciphertext* dan mempelajari *plaintext* yang berkorespondensi. Kriptanalisis dari *cipher* kunci simetri pada umumnya melibatkan serangan terhadap *block cipher* atau *stream cipher* yang lebih mangkus dari serangan manapun terhadap *cipher* sempurna (*the perfect cipher*). Sebagai contoh, sebuah serangan *brute force* terhadap DES memerlukan satu buah *plaintext* yang diketahui dan 255 dekripsi, mencoba hamper kira-kira setengah dari kunci yang mungkin. Namun, ini bukanlah suatu kepastian, karena sebuah serangan kriptanalisis lanjar (*linear cryptanalysis attacks*) terhadap DES memerlukan 243 *plaintext* yang diketahui dan 243 operasi DES. Kriptografi asimetri (atau biasa disebut algoritma kunci publik) pada dasarnya memiliki kekuatan pada kesulitan dalam perhitungan variasi masalah. Masalah yang paling terkenal adalah masalah faktorisasi integer (pada RSA) dan masalah logaritma diskret (*discrete logarithm problems*). Kebanyakan kripanalisis kunci publik memusatkan pada algoritma numerik untuk menyelesaikan masalah perhitungan. Sebagai contoh, algoritma terbaik saat ini untuk menyelesaikan logaritma diskret *elliptic curvebased* memakan waktu yang lebih banyak daripada algoritma terbaik untuk faktorisasi, setidaknya dengan bobot masalah yang sama. Maka dari itu, untuk memperoleh kekuatan dari serangan yang sama, teknik enkripsi dengan faktorisasi harus menggunakan kunci yang lebih besar dibandingkan teknik dengan kurva eliptis.

3.2.2 Jenis-jenis Serangan

Suatu serangan dapat digolongkan menjadi dua jenis serangan berdasarkan keterlibatan penyerang (kriptanalis) pada komunikasi, yakni serangan pasif dan serangan aktif[4]. Serangan pasif berarti seorang penyerang tidak berinteraksi dengan kelompok yang sedang berkomunikasi dan hanya memanfaatkan data yang diperoleh (seperti *ciphertext*). Ciri-ciri dari serangan pasif adalah penyerang tidak terlibat dalam komunikasi antara pengirim dan penerima, serta penyerang hanya melakukan penyadapan (*eavesdropping*) untuk memperoleh data sebanyak-banyaknya. Penyadapan dapat berbentuk *wiretapping, electromagnetic eavesdropping,* dan *acoustic eavesdropping*. Serangan aktif berarti penyerang tidak hanya mengintervensi komunikasi, tetapi juga ikut mempengaruhi sistem. Penyerang dapat saja menghapus sebagian *ciphertext*, mengubah *ciphertext*, menyisipkan potongan *ciphertext* palsu, membalas pesan lama, atau mengubah informasi yang tersimpan. Salah satu contoh serangan aktif adalah *man-in-the-middle attack* (MITM). MITM adalah serangan dimana penyerang mampu untuk membaca, menambah dan mengubah sesuai kehendak pesan antara dua kelompok tanpa sepengetahuan kelompok tersebut. MITM biasanya dilakukan terhadap kriptografi asimetri dan biasanya juga pada protokol Diffi-Hellman. Untuk lebih jelasnya kita misalkan Si-A dan Si-B sebagai dua orang yang akan berkomunikasi, dan Si-C berkehendak untuk menyadap komunikasi tersebut serta mengirimkan pesan yang salah kepada Si-B. Untuk memulai komunikasi, pertama-tama Si-A meminta Si-B untuk mengirimkan kunci publik miliknya. Ketika Si-C dapat menyadap kunci publik yang dikirim, maka MITM akan dapat segera dimulai. Si-C dapat dengan mudah mengirimkan Si-A kunci publik yang cocok dengan kunci pribadi milik Si-A. Si-A yang mempercayai bahwa kunci publik yang ia terima berasal dari Si-B, mulai mengenkripsi kunci yang sebenarnya dikirim oleh Si-C dan kemudian mengirimkan kembali ke Si-B pesan yang telah dienkripsi. Si-C lalu kembali menyadap komunikasi dan mendekripsi pesan tersebut, menyimpan salinan pesan, kemudian mengenkripsi pesan yang telah diubah menggunakan kunci publik yang dikirimkan oleh Si-B dan kemudian mengirimkannya kepada Si-B. Ketika Si-B menerima *ciphertext*, maka ia akan percaya bahwa pesan tersebut berasal dari Si-A. Untuk menjaga keamanan dari serangan MITM maka dapat menggunakan beberapa teknik sebagai berikut :

- Menggunakan otentifikasi mutual
- Menggunakan sandi lewat (*password*)
- Menggunakan pengenal suara (*voice recognition*)

3.3 *Timing Attacks*

Sistem kripto sering kali menghabiskan periode waktu yang berbeda dalam mengolah setiap masukan yang berbeda. Hal tersebut antara lain disebabkan oleh optimasi kemampuan untuk mengenyampingkan beberapa operasi yang tidak perlu, kecepatan *cache*, instruksi *processor* dalam waktu yang tak tentu, dan masalah masalah lainnya[2]. Karakteristik kemampuan bergantung pada kunci enkripsi yang dipakai dan data masukan (*plaintext* dan *ciphertext*). Data atau kunci dapat saja sewaktu-waktu bocor dari saluran waktu (*time channel*) pada saat proses berlangsung, namun pada umumnya data yang bocor hanya berisi sebagian kecil informasi dari sistem kripto, seperti bobot *Hamming* dari sebuah kunci. *Timing attacks* dapat memanfaatkan kebocoran tersebut sehingga mendapatkan keseluruhan informasi tentang kunci pribadi.

4 SIMPULAN

Berdasarkan hasil analisis dan perhitungan algoritma RSA dapat disimpulkan bahwa:

- Keamanan algoritma *RSA* terletak pada tingkat kesulitan dalam memfaktorkan bilangan non prima menjadi faktor primanya, yang dalam hal ini $r = p \times q$.
- Semakin tinggi angka yang digunakan maka akan semakin sulit pula pesan/sandi dapat ditebak oleh pihak ketiga.
- Untuk menjaga keamanan dari serangan MITM (*man-in-the-middle attack*) maka dapat menggunakan otentifikasi mutual,

menggunakan sandi lewat (*password*) dan menggunakan pengenal suara (*voice recognition*)

DAFTAR KEPUSTAKAAN

[1] *http://en.wikipedia.org/wiki /Cryptography,"Cryptography",* Tanggal akses: 2 Januari 2011

[2] Kocher, Paul C. (2005), "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other System. Cryptography ", Research Inc, San Fransisco, USA

[3] Bishop, David. (2003). "Introduction to Cryptography with JavaTM Applets", Jones and Barlett Publisher, Massachusetts, USA

[4] Munir, Rinaldi. (2004). "Kriptografi", Departemen Teknik Informatika, Institut Teknologi Bandung

[5] Nurhayati OD, (2007), "Keamanan Multimedia", Jurusan Sistem Komputer,Universitas Diponegoro, Semarang