Implementasi SHA, Algoritma HAJ, dan Algoritma RSA pada BlackBerry Messanger

Andara Livia

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, bandung
e-mail: andara.livia@gmail.com

ABSTRAK

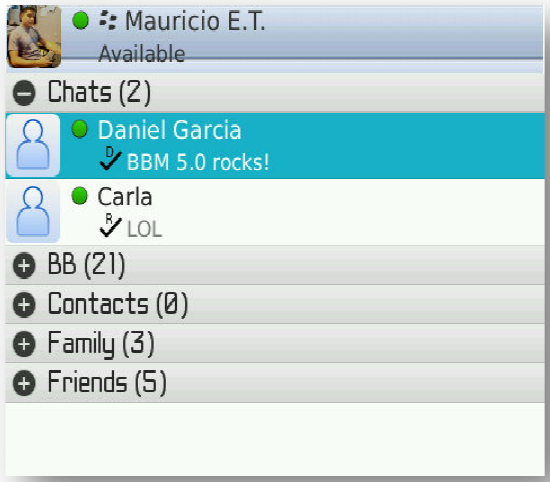
Saat ini blackberry telah menjadi lifestyle sebagian besar penduduk dunia, terutama di Indonesia. Pengguna blackberry bahkan sudah tidak mengenal usia lagi, mulai dari anak-anak, remaja, hingga orang dewasa. Tentu saja terdapat alasan dibalik maraknya gadget ini di kalangan penduduk dunia. Salah satu faktor utamanya yaitu karena banyaknya aplikasi internet yang disupport oleh gadget ini. Salah satu aplikasi yang disukai dan menjadi fitur andalan Blackberry yaitu Blackberry messanger, yaitu sejenis aplikasi instan messaging yang hanya dapat dilakukan oleh sesame pengguna blackberry.
Kenyamanan aplikasi tersebut tentu saja harus diiringi dengan keamanan dalam hal menggunakannya, yaitu keamanan dalam berkiriman pesan. Untuk itu salah satu solusi yang diajukan pada makalah ini adalah dengan menggunakan enkripsi.
Pada makalah ini akan dibahas mengenai proses enkripsi pada pesan BBM dengan menggunakan algoritma SHA-1 untuk membuat message digest pesan, kemudian menggunakan algoritma HAJ untuk mengenkripsi pesan, dan menggunakan algoritma RSA untuk mengenkripsi kunci rahasia yang digunakan untuk mengenkripsi pesan.
Kunci rahasia yang telah dienkripsi tersebut kemudian dikirimkan bersama pesan rahasia tersebut. Tentu saja proses enkripsi ini dilakukan oleh sistem BB sehingga pengguna tidak perlu mengetahui apapun mengenai proses enkripsi-dekripsi ini.

Kata kunci: BBM, SHA, HAJ, RSA.

1. PENDAHULUAN

Gambar 1Blackberry

BlackBerry adalah Perangkat Selular yang memiliki kemampuan layanan Push E-Mail, Telepon, Sms, Menjelajah Internet, dan berbagai kemampuan nirkabel lainnya. BlackBerry sudah menjadi bagian dari gaya hidup kebanyakan penduduk Indonesia saat ini. Salah satu fitur unggulan blackberry ialah BlackBerry Messenger. BlackBerry Messenger atau yang biasa disebut sebagai BBM adalah program pengirim pesan instan yang disediakan untuk para pengguna perangkat BlackBerry. Layanan Messenger ini dibuat khusus bagi pemilik BlackBerry dan dirancang khusus untuk berkomunikasi di antara pengguna. Cara menggunakan BlackBerry Messenger adalah dengan penghubung nomor PIN yang juga eksklusif dimiliki masing-masing perangkat BlackBerry.



Gambar 2 Blackberry Messanger

Karena sudah sangat umumnya penggunaan BlackBerry, BBM pun digunakan untuk berbagai macam kebutuhan. Mulai dari komunikasi antar teman, keluarga, rekan bisnis, dll. Hal ini tentu memicu berbagai pihak untuk melakukan kejahatan seperti penyadapan isi pesan BBM, seperti untuk menyadap obrolan bisnis, politik, hingga obrolan pribadi.

Untuk mengantisipasi terjadinya hal-hal yang tidak diinginkan tersebut, ada baiknya jika menggunakan sistem enkripsi pada saat proses pengiriman pesan. Hal ini beguna untuk menjamin kerahasiaan pesan yang dikirim dengan menggunakan layanan blackberry messanger tersebut.

Beberapa algoritma yang dapat digunakan antara lain algoritma SHA, algoritma HAJ, dan algoritma RSA. Masing-masing algoritma akan dijelaskan pada bab-bab berikutnya.

2. SHA-1

SHA adalah fungsi *hash* satu-arah yang dibuat oleh NIST dan digunakan bersama DSS (*Digital Signature Standard*). SHA didasarkan pada MD4 yang dibuat oleh Ronald L. Rivest dari MIT.

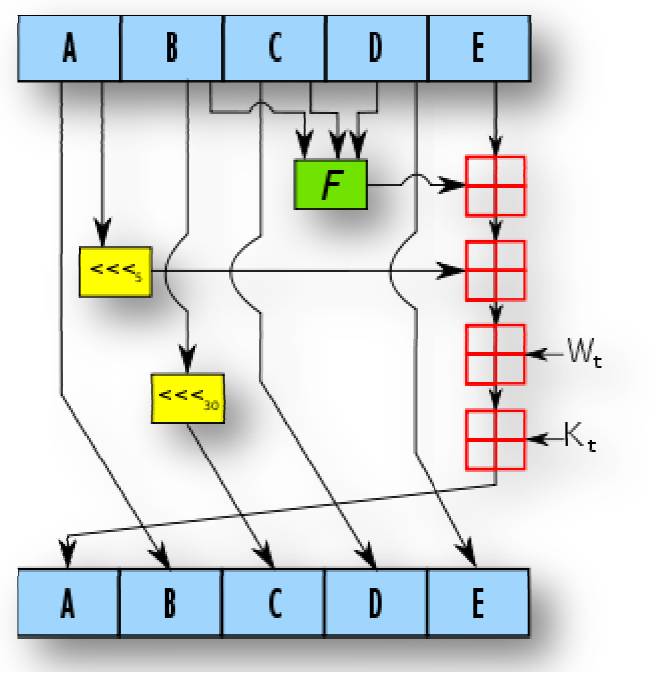
Keamanan SHA terletak pada rancanganan SHA yang membuatnya sedemikian rupa sehingga secara komputasi tidak mungkin menemukan pesan yang berkoresponden dengan *message digest* yang diberikan.

Algoritma SHA menerima masukan berupa pesan dengan ukuran maksimum 2^{64} bit (2.147.483.648 *gugabyte*) dan menghasilkan *message digest* yang panjangnya 160 bit. Lebih panjang dari *message digest* yang dihasilkan MD5.

Langkah-langkah pembuatan *message digest* secara garis besar adalah sebagai berikut:

1. Penambahan bit-bit pengganjal (*padding bits*).
   Pesan ditambah dengan sejumlah bit pengganjal sedemikian sehingga panjang pesan (dalam satuan bit) kongruen dengan 448 modulo 512.
2. Penambahan nilai panjang pesan semula.
   Pesan yang telah diberi bit-bit pengganjal selanjutnya ditambah lagi dengan 64 bit yang menyatakan panjang pesan semula. Setelah ditambah 64 bit, panjang pesan sekarang menjadi 512 bit.
3. Inisialisasi penyangga (*buffer*) MD.
   SHA membutuhkan 5 buah penyangga (*buffer*) yang masing-masing panjangnya 32 bit. Kelima penyangga ini menampung hasil antara dan hasil akhir.
   Nilai kelima penyangga tersebut adalah:
   - A = 67452301
   - B = EFCDAB89
   - C = 98BADCFE
   - D = 10325476
   - E = C3D2E1F0
4. Pengolahan pesan dalam blok berukuran 512 bit.
   Pesan dibagi menjadi L buah blok yang masing-masing panjangnya 512 bit. Setiap blok 512 bit diproses bersama dengan penyangga MD menjadi keluaran 128 bit, dimana proses ini disebut sebagai H_{SHA}. Proses H_{SHA} terdiri dari 80 buah putaran dan masing-masing putaran menggunakan bilangan penambah K_t, yaitu:
   - Putaran $0 \le t \le 19$ K_t = 5A827999
   - Putaran $20 \le t \le 39$ K_t = 6ED9EBA1
   - Putaran $40 \le t \le 59$ K_t = 8F1BBCDC
   - Putaran $60 \le t \le 79$ K_t = CA62C1D6

   Setiap putaran menggunakan operasi dasar yang sama. Operasi dasar tersebut, yaitu:



Gambar 3 Operasi Dasar pada SHA-1

Operasi dasar yang diperlihatkan pada gambar di atas dapat dituliskan dalam persamaan:

$a, b, c, d, e \leftarrow (CLS_5(a) + f_t(b, c, d) + e, W_t + K_t), a, CLS_{30}(b), c, d$

a, b, c, d, e = lima buah penyangga 32 bit (A, B, C, D, E)

t = putaran, $0 \leq t \leq 79$

f_t = fungsi logika

CLS_s = *circular shift left* sebanyak s bit

W_t = word 32 bit yang diturunkan dari blok 512 bit yang sedang diproses

K_t = konstanta penambah

$+$ = operasi penjumlahan modulo 2^{32}

Fungsi f_t adalah fungsi logika yang melakukan operasi *bitwise*. Operasi *bitwise* yang dilakukan adalah sebagia berikut:

Putaran	***$f_t(b, c, d)$***
0 … 19	$(b \wedge c) \vee (\sim b \wedge d)$
20 … 39	b ⊕ c ⊕ d
40 … 59	$(b \wedge c) \vee (b \wedge d) \vee (c \wedge d)$
60 … 79	b ⊕ c ⊕ d

Nilai W_t sampai W_{16} berasal dari 16 *word* pada blok yang sedang diproses. Sedangkan nilai W_t berikutnya didapatkan dari persamaan

$$W_t = W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3}$$

Setelah putaran ke-79, a, b, c, d, dan e ditambahkan ke A, B, C, D, dan E dan selanjutnya algoritma memproses data berikutnya. Keluaran akhir dari program SHA adalah hasil penyambungan bit-bit di A, B, C, D, dan E.

2.1 SHA-1 Pseudocode

Pseudocode untuk SHA-1 adalah sebagai berikut.

```
Inisialisasi Variable:
h0 = 0x67452301
h1 = 0xEFCDAB89
h2 = 0x98BADCFE
h3 = 0x10325476
h4 = 0xC3D2E1F0

Pre-processing:
append the bit '1' to the message
append 0 ≤ k < 512 bits '0', so that
the resulting message length (in
bits)
   is congruent to 448 ≡ −64 (mod
512)
append length of message (before pre-
processing), in bits, as 64-bit big-
endian integer

Proses pesan dalam pecahan 512-bit:
break message into 512-bit chunks
for each chunk
    break chunk into sixteen 32-bit
big-endian words w[i], 0 ≤ i ≤ 15
```

```
    for i from 16 to 79
        w[i] = (w[i-3] xor w[i-8] xor w[i-
14] xor w[i-16]) leftrotate 1

    Inisialisasi nilai hash:
    a = h0
    b = h1
    c = h2
    d = h3
    e = h4

    Main loop:
    for i from 0 to 79
        if 0 ≤ i ≤ 19 then
            f = (b and c) or ((not b) and d)
            k = 0x5A827999
        else if 20 ≤ i ≤ 39
            f = b xor c xor d
            k = 0x6ED9EBA1
        else if 40 ≤ i ≤ 59
            f = (b and c) or (b and d) or (c
and d)
            k = 0x8F1BBCDC
        else if 60 ≤ i ≤ 79
            f = b xor c xor d
            k = 0xCA62C1D6

        temp = (a leftrotate 5) + f + e + k
+ w[i]
        e = d
        d = c
        c = b leftrotate 30
        b = a
        a = temp

    h0 = h0 + a
    h1 = h1 + b
    h2 = h2 + c
    h3 = h3 + d
    h4 = h4 + e

digest = hash = h0 append h1 append h2
append h3 append h4
```

3. HAJ

HAJ merupakan algoritma block cipher yang dikembangakan oleh mahasiswa Teknik Informatika ITB angaktan 2007, yaitu Hanugrha Abidianto (13507008), Andara Livia (13507054), dan Juliana Amytianty (13507068). Awalnya algoritma ini dikembangakan untuk memenuhi tugas besar pertama pelajaran Kriptografi (IF3058). Namun pada saat tugas besar tersebut, algoritma ini belum berhasil diimplementasikan.

3.1 Cipher Block (*Block Cipher)*

Pada algoritma block cipher, algoritma kriptografi beroperasi pada plainteks/cipherteks dalam bentuk blok bit, yang dalam hal ini rangkaian bit dibagi menjadi blok-blok bit yang panjangnya sudah ditentukan sebelumnya. Misalnya panjang blok adalah 64 bit, maka itu berarti algoritma enkripsi memperlakukan 8 karakter setiap kali penyandian.
Enkripsi dilakukan terhadap blok bit plainteks menggunakan bit-bit kunci berukuran sama dengan ukuran blok plainteks. Algoritma enkripsi menghasilkan blok cipherteks yang berukuran sama dengan plainteks.
Mode Operasi Cipher Blok
Plainteks dibagi menjadi beberapa blok dengan panjang tetap. Beberapa mode operasi dapat diterapkan untuk melakukan enkripsi terhadap keseluruhan blok plainteks. Empat mode operasi yang lazim digunakan pada sistem cipher blok, yaitu:
Electronic Code Book
Cipher Block Chaining
Cipher Feedback
Output Feedback

3.1.1 Electronic Code Book (ECB)

Pada mode ini, setiap blok plainteks Pi dienkripsi secara individual dan independen menjadi blok cipherteks Ci. Secara matematis, enkripsi dengan mode ECB dinyatakan sebagai:

$$C_i = E_K(P_i)$$

dan didekripsi sebagai:

$$P_i = D_K(C_i)$$

yang dalam hal ini, Pi dan Ci masing-masing blok plainteks dan blok cipherteks ke-i.

3.1.2 Cipher Block Chaining (CBC)

Mode ini menerapkan mekanisme umpan-balik pada sebuah blok. Hasil enkripsi blok sebelumnya di-umpan-balikkan ke dalam enkripsi blok saat ini. Caranya, blok plainteks saat ini di-XOR-kan terlebih dahulu dengan blok cipherteks hasil enkripsi sebelumnya, selanjutnya hasil enkripsi tersebut akan digunakan untuk fungsi enkripsi selanjutnya. Dengan mode ini, setiap blok cipherteks bergantung pada blok plainteksnya dan juga pada seluruh blok plainteks sebelumnya.
Dekripsi dilakukan dengan memasukkan blok cipherteks saat ini ke fungsi dekripsi, kemudian melakukan XOR terhadap hasilnya dengan blok cipherteks sebelumnya. Dalam hal ini, blok cipherteks sebelumnya berfungsi sebagai umpan-maju pada akhir proses dekripsi.
Secara matematis, enkripsi dengan mode ECB dinyatakan sebagai:

$$C_i = E_K(P_i \oplus C_{i-1})$$

dan didekripsi sebagai:

$$P_i = D_K(C_i) \oplus C_{i-1}$$

Sebagai awal, C0 = IV (initialization vector). IV dapat diberikan oleh pengguna atau dibangkitkan secara acak oleh program. Jadi, untuk menghasilkan blok cipherteks pertama (C1), IV digunakan untuk menggantikan blok cipherteks sebelumnya, C0.

3.1.3 Cipher Feedback (CFB)

Jika mode CBC diterapkan pada aplikasi komunikasi data, maka enkripsi tidak dapat dilakukan bila blok plainteks yang diterima belum lengkap. Pada mode CFB, data dienkripsikan dalam unit yang lebih kecil dari ukuran blok. Unit yang dienkripsikan dapat berupa bit per bit, 2 bit, 3 bit, sampai n-bit.

3.1.4 Output Feedback (OFB)

Mode OFB mirip dengan mode CFB, namun n-bit dari hasil enkripsi terhadap antrian disalin menjadi elemen posisi paling kanan di antrian.

3.2 Algoritma HAJ

Karena algoritma ini menggunakan jaringan feistel, maka untuk proses enkripsi dan dekripsi menggunakan algoritma yang sama. Putaran pada jaringan feistel dilakukan sebanyak 16 kali.
Pertama-tama plainteks yang telah dibagi menjadi blok berukuran 64bit dimasukkan ke dalam jaringan feistel. Di dalam jaringan feistel, plainteks tersebut dipermutasi dengan menggunakan teknik Mongean Shuffle. Permutasi yang dilakukan adalah sebanyak 4 kali. Setelah itu, blok hasil permutasi tersebut dipecah menjadi dua bagian, bagian kiri (s1) dan bagian kanan (s2), masing-masing berukuran 32 bit. S1 dan s2 ini yang akan diputar sebanyak 16 kali. Perputaran dilakukan dengan memasukkan s2 dan kunci internal ke dalam fungsi Feistel. Kemudian keluaran dari fungsi tersebut di-XOR dengan s1. Nilai s2 dimasukkan ke dalam s1 dan hasil XOR dimasukkan ke dalam s2. Setelah prosedur tersebut diulang sebanyak 16 kali, s1 dan s2 disatukan kembali. Blok hasil penggabungan tersebut kemudian dipermutasi kembali dengan menggunakan fungsi Mongean Shuffle sebanyak 3 kali. Hasil permutasi tersebut lah yang menjadi chipertext.

4. RSA

RSA termasuk ke dalam keluarga algoritma kriptografi kunci public. Algoritma kunci public atau disebut juga algoritma kunci nirsimetri (*asymmetric cryptosystem*) sendiri adalah algoritma yang memungkinkan pengguna berkomunikasi secara aman tanap perlu berbagi kunci rahasia. Pada algoritma kunci public, kunci untuk enkripsi diumumkan kepada public sehingga dapat diketahui oleh siapapun. Sementara kunci untuk dekripsi hanya diketahui oleh penerima pesan (dirahasiakan). Siapapun dapat mengirim pesan yang dienkripsi dengan kunci public tersebut, tetapi hanya penerima pesan yang dapat mendekripsi pesan karena hanya ia yang mengetahui kunci privatnya sendiri.
Algoritma RSA dibuat oleh 2 orang peneliti dari MIT pada tahun 1976, yaitu Ron Rivest, Adi Shamir, dan Leonard Adleman. Keamanan pada RSA terletak pada sulitnya memfaktorkan bilangan yang besar menjadi faktor-faktor prima. Pemfaktoran dilakukan untuk memperoleh kunci privat.

4.1 Algoritma Membangkitkan Pasangan Kunci

1. Pilih dua bilangan prima $p \neq q$ secara acak dan terpisah untuk tiap-tiap p dan q.
2. Hitung n = p q. n hasil perkalian dari p dikalikan dengan q.
3. Hitung $\varphi = (p-1)(q-1)$.
4. Pilih bilangan bulat (integer) e antara satu dan φ ($1 < e < \varphi$) yang relatif prima terhadap φ.
5. Bangkitkan kunci privat d dimana $d.e \equiv 1 \pmod{\varphi}$.

- Bilangan prima dapat diuji probabilitasnya menggunakan Fermat's little theorem:

  a^(n-1) mod n = 1

  jika n adalah bilangan prima, diuji dengan beberapa nilai a menghasilkan kemungkinan yang tinggi bahwa n ialah bilangan prima. Carmichael numbers (angka-angka Carmichael) dapat melalui pengujian dari seluruh a, tetapi hal ini sangatlah langka.
- Langkah 3 dan 4 dapat dihasilkan dengan algoritma extended Euclidean; lihat juga aritmetika modular.
- 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 (mod (p-1)(q-1)).
  Hasil dari algoritma di atas adalah:
  - Kunci public adalah pasangan (e, n).
  - Kunci privat adalah pasangan (d, n).
- m adalah plainteks yang akan dienkripsi. Bersifat rahasia.
- c adalah cipherteks yang dihasilkan dari proses enkripsi, bersifat tidak rahasia.
- Pada public key terdiri atas:
  - n, modulus yang digunakan. Didapatkan dari p x q.
  - e, eksponen publik (sering juga disebut eksponen enkripsi).
- Pada private key terdiri atas:
  - n, modulus yang digunakan, digunakan pula pada public key.
  - d, eksponen pribadi (sering juga disebut eksponen dekripsi), yang harus dijaga kerahasiaannya.

4.2 Algoritma Enkripsi/Dekripsi

- **Enkripsi**
  1. Ambil kunci public penerima pesan *e*, dan modulus *n*.
  2. Nyatakan plainteks m menjadi blok-blok m_1, m_2, m_3, …., sedemikian sehingga setiap blok merepresentasikan nilai di dalam selang [0, n-1].
  3. Setiap blok m_i dienkripsi menjadi blok c_i dengan rumus $c_i = m_i^e \bmod n$.
  4. Persamaan enkripsi adalah sebagai berikut:
     $$E_e(m) = c \equiv m^e \bmod n$$

- **Dekripsi**
  1. Setiap blok cipherteks c_i didekripsi kembali menjadi blok $m_i = c_i^d \bmod n$.
  2. Bila terdapat cipherteks c, n, yang telah desepakati sebelumnya,
     $$Dd(c) = m \equiv cd \bmod n$$

Berikut ini merupakan contoh dari enkripsi RSA dan dekripsinya. Parameter yang digunakan disini berupa bilangan kecil.
Pertama-tamat tentukan

Public key yang digunakan adalah (e,n). Private key yang digunakan adalah d. Fungsi pada enkripsi ialah:

$$E(m) = me \bmod n = m17 \bmod 3233$$

dimana m adalah plaintext.
Fungsi dekripsi ialah:

$$D(c) = cd \bmod n = c2753 \bmod 3233$$

dimana c adalah ciphertext.

Untuk melakukan enkripsi plaintext bernilai "123", perhitungan yang dilakukan

$$E(123) = 12317 \bmod 3233 = 855$$

Untuk melakukan dekripsi ciphertext bernilai "855" perhitungan yang dilakukan

$$D(855) = 8552753 \bmod 3233 = 123$$

4.3 Keamanan RSA

Keamanan algoritma RSA didasarkan pada sulitnya memfaktorkan bilangan besar menjadi faktor-faktor primanya.
Masalah pemfaktoran: faktorkan *n* menjadi dua faktor primanya, *p* dan *q*, sedemikian hingga n = p.q. Sekali n berhasil difaktorkan menjadi p dan q, maka $\varphi = (p-1)(q-1)$ dapat dihitung. Karena kunci enkripsi e diumumkan (tidak rahasia), maka kunci dekripsi d dapat dihitung dari persamaan $e.d \equiv 1 \pmod{\varphi}$.

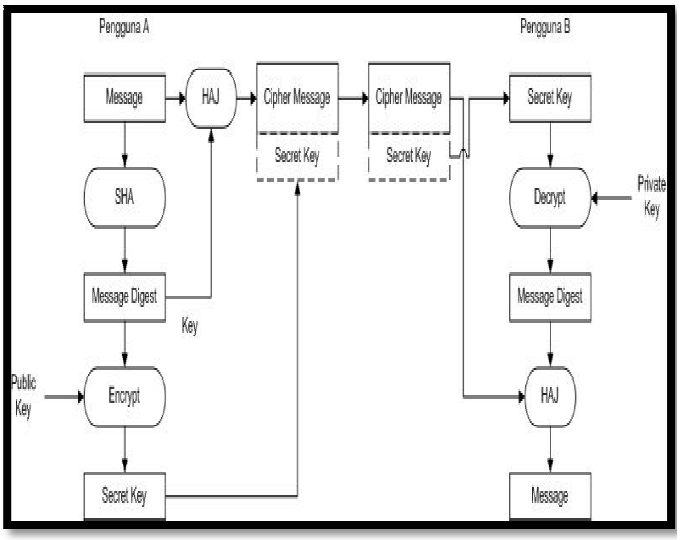
Sampai saat ini semua bukti yang diketahui menunjukkan bahwa upaya pemfaktoran bilangan besar dalam waktu polynomial, tetapi juga tidak dapat dibuktikan algoritma tersebut ada. Fakta inilah yang membuat algoritma RSA dianggap aman. [enemu lagoritma RSA menyarankan nilai p dan q panjangnya lebih dari 100 angka. Dengan demikian hasil kali n = p x q akan berukuran lebih dari 200 angka. Menurut pengembang algoritma ini, usaha untuk mencari faktor prima dari bilangan 200 angka membutuhkan waktu komputasi sebanyak 4 milyar tahun, sedangkan untuk bilangan 500 angka membutuhkan wakut 10^{25} tahun.
Secara umum dapat disimpulkan bahwa RSA hanya aman jika n cukup besar. Pada tahun 1994, RSA sudah dapat dipecahkan dalam waktu 8 bulan.
Pada thaun 1993, Peter Shor menerbitkan Algoritma Shor, menunjukkan bahwa sebuah komputer quantum secara prinsip dapat melakukan faktorisasi dalam waktu polinomial, mengurai RSA dan algoritma lainnya. Bagaimanapun juga, masih terdapat pedebatan dalam pembangunan komputer quantum secara prinsip.

4.4 Kecepatan

RSA memiliki kecepatan yang lebih lambat dibandingkan dengan DES dan algoritma simetrik lainnya. Pada praktek pesan dienkripsi dengan kunci rahasia dengan menggunakan salah satu algoritma kriptografi kunci-simetri, sedangkan RSA digunakan untuk mengenkripsi kunci rahasia. Pesan dan kunci rahasia yang masing-masing sudah dienkripsi dikirim bersama-sama. Penerima pesan mula-mula mendekripsi kunci rahasia dengan kunci privatnya, lalu menggunakan kunci rahasia tersebut untuk mendekripsi pesan.

5. IMPLEMENTASI

Untuk menggabungkan ketiga algoritma di atas sehingga dapat digunakan dalam proses pengenkripsian pesan Blackberry Messanger, perhatikan gambar di bawah ini.



Gambar 4 Proses pengiriman pesan dengan menggunakan SHA, HAJ, dan RSA

Pertama-tama pesan yang akan dikirimkan oleh pengguna A diolah dengan menggunakan SHA-1 untuk mendapatkan message digest pesan tersebut. Kemudian pesan akan dienkripsi dengan menggunakan algoritma HAJ dengan message digest yang telah dibuat sebelumnya digunakan sebagai kunci rahasia.
Sebelum dikirimkan message digest yang digunakan sebagai kunci rahasia dienkripsi etrlebih dahulu dengan menggunakan kunci publik yang dimiliki pengguna A. Setelah itu message digest yang telah dienkripsi ditempelkan pada pesan yang telah dienkripsi dengan menggunakan algoritma HAJ. Kemudian pesan beserta kunci rahasia tersebut dikirimkan kepada pengguna B. sesampainya di pengguna B, kunci rahasia didekripsi dengan menggunakan kunci privat yang dimiliki pengguna B. Proses dekripsi tersebut akan menghasilkan message digest yang kemudian akan digunakan sebagai kunci untuk mendekripsi pesan rahasia dengan menggunakan algoritma HAJ. Hasil dekripsi tersebut akan menghasilkan pesan seperti sedia kala yang dapat dibaca oleh pengguna B.

6. KESIMPULAN

Algoritma SHA-1 digunakan untuk membuat message digest. SHA-1 menerima masukan berupa pesan dengan panjang maksimum 2^{64} bit dan menghasilkan message digest berukuran 160 bit. Pesan pada BBM dibuat message digestnya dengan menggunakan SHA-1.
Algoritma HAJ merupakan cipher block yang bekerja seperti layaknya DES. HAJ menerima pesan dalam bentuk blok-blok berukuran 64 bit dan menghasilkan cipherteks yang merupakan gabungan dari blok-blok hasil enkripsi. Setiap blok hasil enkripsi juga berukuran 64 bit. Pada penerapannya saat mengenkripsi pesan pada BBM, pesan dienkripsi dengan menggunakan algoritma HAJ dengan menggunakan message digest sebagai kunci rahasianya.
Kunci rahasia tersebut dienkripsi dengan menggunakan RSA yang merupakan algoritma kunci public.
Pengembang telah membuat prototype untuk ketiga algoritma ini untuk mempraktekan proses enkripsi pada BBM. Pada prototype tersebut, p dan q pada algoritma RSA menggunakan bilangan berukuran minimal 4 digit untuk proses keamanan, namun minimal hanya 4 digit karena keterbatasan kemampuan komputer pengguna.
Sayangnya pengembang belum mencoba mengintegrasikan prototype tersebut ke Blackberry karena tidak adanya objek percobaan.

REFERENSI

[1] http://en.wikipedia.org/wiki/SHA-1

[2] http://id.wikipedia.org/wiki/RSA

[3] http://na.blackberry.com/eng/services/blackberrymessenger/

[4] http://id.wikipedia.org/wiki/BlackBerry_Messenger

[5]http://www.blackberryforums.com/general-blackberry-discussion/121914-pin-blackberry-messenger.html

[6]http://supportforums.blackberry.com/t5/MDS-Runtime-Development/C-and-VS2008/m-p/203997

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 17 Mei 2010

Andara Livia (13507054)