PENERAPAN TANDA TANGAN DIGITAL UNTUK KEAMANAN TRANSAKSI SMS - BANKING

Budiono – NIM : 13503013

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl Ganesha 10, Bandung
E-mail : if13013@students.if.itb.ac.id, boedi135@students.itb.ac.id

Abstrak

Short Message Services (SMS) merupakan suatu pesan yang dapat diterima dan dikirimkan oleh ponsel melalui layanan operator seluler tertentu baik bersifat CDMA maupun GSM. SMS ini awalnya merupakan fitur yang distandarkan GSM sebagai pesan singkat biasa, namun dalam perkembangannya fitur ini justru diminati oleh masyarakat di dunia sehingga terus dikembangkan sampai saat ini.

SMS-Banking merupakan layanan perbankan yang memudahkan pelanggan untuk melakukan transaksi keuangan hanya dengan menggunakan SMS. Layanan ini merupakan layanan yang dikembangkan untuk memudahkan pelanggan bank melakukan transaksi kapanpun dan dimanapun tanpa harus berhadapan dengan *teller* bank.

Faktor keamanan menjadi sangat penting semenjak kelahiran dari produk layanan ini. Hal ini dikarenakan masyarakat sudah mulai resah dengan berbagai penipuan yang terjadi akhir-akhir ini. Selain itu juga, transaksi perbankan sering melibatkan nilai nominal yang cukup besar sehingga harus memiliki tingkat keamanan yang tinggi sehingga masyarakat yang menggunakannya tidak merasa khawatir.

Digital Signature pada transaksi SMS ini dilakukan dengan menggunakan prinsip *hashing* yang sesuai dengan *Digital Signature Standard* (DSS) yaitu menggunakan *Secure Hash Algorithm* (SHA). Dengan menggunakan *Digital Signature Algorithm* (DSA), maka proses pembentukan tanda tangan dari pesan yang akan dikirim dan proses pemeriksaan keabsahannya dapat dilakukan.

Tulisan ini akan menghasilkan suatu aplikasi J2ME yang dapat dijalankan di ponsel yang mendukung Java dan tanda tangan digital yang ada akan disertakan dalam pesan SMS yang akan dikirimkan dalam proses transaksi perbankan SMS-Banking. Selain itu, proses verifikasi transaksi lebih mudah karena tidak perlu melakukan dekripsi dari pesan SMS.

Kata Kunci : SMS-Banking, *Digital Signature Standard, Digital Signature Algorithm*, *Short Message Services, Java, Secure Hash Algorithm*.

1. Pendahuluan

Perkembangan teknologi informasi terutama dalam bidang *mobile* telah membawa perubahan kepada masyarakat untuk melakukan komunikasi antar sesamanya. Hal ini dapat dilihat melalui sarana penggunaan pesan singkat atau SMS yang mencapai angka yang cukup tinggi yaitu sekitar 122 juta SMS awal November 2006 (*sumber ww.xl.co.id*). Bahkan tidak sedikit layanan-layanan tertentu menggunakan SMS sebagai sarana transaksinya seperti perbankan yang terkenal dengan produk SMS-Banking .

SMS-Banking merupakan layanan perbankan yang memudahkan pelanggan untuk melakukan transaksi keuangan hanya dengan menggunakan SMS. Layanan ini merupakan layanan yang dikembangkan untuk memudahkan pelanggan bank melakukan transaksi kapanpun dan

dimanapun tanpa harus berhadapan dengan *teller* bank.

Faktor keamanan menjadi sangat penting semenjak kelahiran dari produk layanan ini. Hal ini dikarenakan masyarakat sudah mulai resah dengan berbagai penipuan yang terjadi akhir-akhir ini. Selain itu juga, transaksi perbankan sering melibatkan nilai nominal yang cukup besar sehingga harus memiliki tingkat keamanan yang tinggi dan masyarakat yang menggunakannya tidak merasa khawatir.

Sistem keamanan SMS-Banking saat ini masih memiliki kelemahan meskipun layanan ini lebih aman dibandingkan dengan internet-banking karena memiliki sistem keamanan yang berlapis, yaitu dari operator yang menyediakan infrastruktur serta dari jaringan perbankan sendiri [2].

Sistem keamanan SMS-Banking yang digunakan saat ini adalah dengan melakukan enskripsi pada pesan sms yang dikirimkan untuk transaksi perbankan. Dan proses ini dilakukan oleh *handphone* dengan menggunakan *Key* yang tertanam pada SIM *card* operator telepon seluler tertentu. Dan pengguna wajib memasukkan password tertentu untuk melakukan transaksi ini.

Menurut paparan Onno W. Purbo[2], menyebutkan bahwa bahaya keamanan layanan SMS-Banking ini justru datang dari aspek non teknis. Bahaya tersebut mengancam ketika ada pihak ketiga mengetahui nomor PIN dari seorang pengguna *mobile banking*. Adapun peluang itu dapat muncul dari operator atau pada orang terdekat sendiri. Ketika operator mengetahui nomor PIN pengguna ataupun *password* nasabah ini menjadi tidak aman kembali.

Kelemahan tersebut dapat ditanggulangi dengan menerapkan *digital siganture* (tanda tangan digital) pada pesan SMS yang dikirimkan. Proses ini memerlukan kunci publik dan kunci privat yang dibutuhkan. Sehingga operator perbankan dapat melakukan verifikasi SMS transaksi yang diterima dari pengguna layanan melalui kunci publik yang dimilikinya tanpa mengetahui kunci privat nasabah. Dengan solusi ini pihak ketiga sulit untuk mengetahui kunci privat yang dimiliki oleh pelanggan. Kunci privat yang dimiliki oleh pelanggan ini dihasilkan dari bilangan prima yang dihasilkan dari fungsi *random* aplikasi di *handphone* dan kunci publik dari pihak bank. Sedangkan untuk menanggulangi dari orang terdekat dapat dilakukan dengan *generating* kunci privat pada setiap periode waktu tertentu.

2. Tinjauan Pustaka Penunjang

2.1 Short Message Service

Short Message Service (SMS) merupakan sebuah layanan yang banyak diaplikasikan pada sistem komunikasi kasi tanpa kabel, memungkinkan dialkukannya pengiriman pesan dalam bentuk *alphanumeric* antara terminal pelanggan atau antara terminal pelanggan dengan sistem eksternal seperti email, *paging*, *voice mail*, dan lain-lain [4]. Isu SMS pertama kali muncul di belahan Eropa pada sekitar tahun 1991 bersama dengan sebuah teknologi komunikasi *wireless* yang saat ini cukup banyak penggunanya, yaitu *Global System for Mobile Communication* (GSM). Dipercaya bahwa pesan pertama yang dikirimkan menggunakan SMS dilakukan pada bulan Desember 1992, dikirimkan dari sebuah Personal Computer (PC) ke telepon *mobile* (bergerak) dalam jaringan GSM milik Vodafone Inggris. Perkembangannya kemudian merambah ke benua Amerika, dipelopori oleh beberapa operator komunikasi bergerak berbasis digital seperti BellSouth Mobility, PrimeCo, Nextel, dan beberapa operator lain. Teknologi digital yang digunakan bervariasi dari yang berbasis GSM, *Time Division Multiple Access* (TDMA), hingga *Code Division Multiply Access* (CDMA).

Tidak diragukan lagi SMS sangat sukses di pasaran, di tempat kelahirannya sendiri, yaitu Eropa, trafik SMS mencapai lebih dari 3 miliar per bulan meskipun tanpa ada program marketing yang proaktif dari operator seluler dan vendor pembuat perangkat komunikasi bergerak. Kesuksesan SMS dianggap sebagai kesuksesan yang tidak disengaja dan cukup mengejutkan bagi pihak-pihak yang terjun dalam industri telekomunikasi bergerak karena beberapa pihak yang berkompeten sebelumnya memprediksi bahwa SMS tidak akan laku karena penggunanya cukup sulit dan materi untuk marketingnya sulit ditentukan.

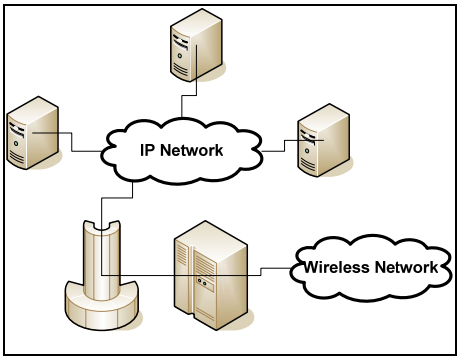
SMS menjadi fenomena tersendiri, dalam waktu yang cukup singkat pertumbuhannya sangat tinggi tanpa ada penurunan tarif yang berarti, bahkan dapat dikatakan tarifnya mengambil posisi *steady state*. Biasanya, bahkan dalam kasus layanan telepon bergerak, tarif akan turun seiring dengan meningkatnya pengguna. Fakta lainnya adalah fasilitas SMS dalam telepon bergerak ternyata punya andil cukup besar dalam menarik kaum muda masuk dalam pasar telepon bergerak.

Dalam sistem SMS, mekanisme utama yang dilakukan dalam sistem adalah melakukan pengiriman pesan singkat dari satu terminal pelanggan ke terminal yang lain. Hal ini dapat dilakukan berkat adanya sebuah entitas dalam sistem SMS yang bernama *Short Message Service Center* (SMSC), disebut juga dengan *Message Center* (MC). SMSC merupakan sebuah perangkat yang melakukan tugas *store and forward* trafik SMS. Di dalamnya termasuk penentuan atau pencarian rute tujuan akhir dari pesan SMS. Sebuah SMSC biasanya didesain untuk dapat menangani SMS dari berbagai sumber seperti *Voice Mail System* (VMS), *Web-based messaging*, *Email Integration*, External Short Message Entity (ESME), dan lain-lain. Dalam interkoneksi dengan entitas dalam jaringan komunikasi *wireless* seperti *Home Location Register* (HLR) dan *Mobile Switching Center* (MSC), SMSC biasanya selalu menggunakan *Signal Transfer Point* (STP).

Layanan SMS merupakan sebuah layanan yang bersifat *nonreal time* dimana sebuah pesan SMS dapat di-*submit* ke suatu tujuan, tidak peduli apakah tujuan tersebut aktif atau tidak. Bila dideteksi bahwa tujuan tidak aktif, maka sistem akan menunda pengiriman ke tujuan hingga tujuan aktif kembali. Pada dasarnya sistem SMS akan menjamin *delivery* dari suatu pesan SMS hingga sampai ke tujuan. Kegagalan pengiriman yang bersifat sementara seperti tujuan tidak aktif akan selalu teridentifikasi sehingga pengiriman ulang pesan SMS akan selalu dilakukan kecuali bila diberlakukan aturan bahwa pesan SMS yang telah melampui batas waktu tertentu harus dihapus dan dinyatakan gagal terkirim.

Karakteristik utama SMS adalah SMS merupakan sebuah sistem pengiriman data yang berisfat *out-of-band* dengan *bandwith* kecil. Dengan karakteristik ini, pengiriman suatu *burst data* yang pendek dapat dilakukan dengan effisiensi yang sangat tinggi. Pada awalnya SMS diciptakan untuk menggantikan layanan *paging* dengan menyediakan layanan serupa yang bersifat *two-way messaging* ditambah dengan *notification service*, khususnya untuk *voice mail*. Pada perkembangan selanjutnya, muncul jenis-jenis layanan lain seperti email, fax, dan *integration*, *interactive banking*, *information service*, dan integrasi dengan aplikasi berbasis internet. Selain itu juga berkembang layanan data *wireless* seperti SIM *download for activation*, debet, *profile editing*, dan lain-lain yang kemudian mendorong timbulnya layanan-layanan seperti *web-based messaging*, *gaming*, dan *chatting*.

Layanan SMS dibangun dari berbagai entitas yang saling terkait dan mempunyai fungsi dan tugas masing-masing. Tidak ada satupun dalam sistem SMS yang dapat bekerja secara parsial. Entitas dalam jaringan SMS ini disebut juga elemen jaringan SMS. Secara umum arsitektur sistem SMS, khususnya untuk sistem yang diintegrasikan dengan jaringan *wireless*, adalah sebagai berikut :



Gambar 1. Arsitektur dasar jaringan SMS [4]

2.2 SMS Banking

SMS-Baking merupakan suatu layanan bank yang memudahkan pelanggan untuk melakukan transaksi perbankan hanya dengan menggunakan perangkat seluler mereka. Transaksi tersebut dilakukan melalui SMS yang dikirimkan secara langsung ke nomor tujuan bank, atau dapat juga terimplementasi dalam *sim card* telepon seluler pelanggan. Aplikasi yang tertanam

pada sim card telepon seluler ini menyimpan beberapa informasi mengenai transaksi yang bisa dilakukan dengan menggunakan tarif SMS.

Berikut beberapa layanan yang disediakan oleh Bank untuk dapat melakukan transaksi melalui SMS :

- cek saldo
- cek kurs valuta asing
- cek tiga transaksi terakhir
- cek tagihan mitra
- pembayaran tagihan mitra
- pembayaran kartu kredit
- transfer antar rekening

layanan ini menjanjikan mobilitas yang tinggi, bisa dilakukan kapanpun dan dimanapun, bahkan saat roaming internasional sekalipun, tentu saja dengan syarat adalah sistem yang digunakan GSM dan nomor ponsel sudah terdaftar untuk roaming internasional.

2.3 Secure Hash Algorithm[1]

Fungsi *hash* merupakan sebuah fungsi yang menerima masukan *string* yang panjangnya sembarang, lalu menstransformasikannya menjadi *string* keluaran yang panjangnya tetap (*fixed*) (umumnya berukuran jauh lebih kecil daripada ukuran *string* semula).

Persamaan fungsi *hash*:

$h = H(M)$

M = pesan kuran sembarang

h = nilai *hash* (*hash value*) atau pesan-ringkas (*message-digest*)

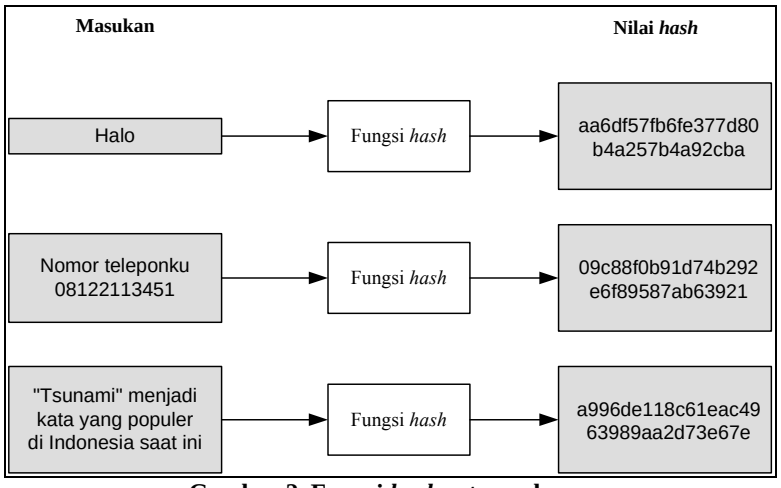
h <<<< M

- Contoh: *size*(M) = 1 MB, *size*(h) = 128 bit !!!!
- Nama lain fungsi *hash* adalah:

- *fungsi kompresi (compression function)*
- *cetak-jari (fingerprint)*
- *cryptographic checksum*
- *message integrity check (MIC)*
- *manipulation detection code (MDC)*

Sifat-sifat fungsi *hash* satu-arah adalah sebagai berikut:

1. Fungsi H dapat diterapkan pada blok data berukuran berapa saja.
2. H menghasilkan nilai (h) dengan panjang tetap (*fixed-length output*).
3. $H(x)$ mudah dihitung untuk setiap nilai x yang diberikan.
4. Untuk setiap h yang dihasilkan, tidak mungkin dikembalikan nilai x sedemikian sehingga $H(x) = h$. Itulah sebabnya fungsi H dikatakan fungsi *hash* satu-arah (*one-way hash function*).
5. Untuk setiap x yang diberikan, tidak mungkin mencari $y \neq x$ sedemikian sehingga $H(y) = H(x)$.
6. Tidak mungkin mencari pasangan x dan y sedemikian sehingga $H(x) = H(y)$.



Gambar 2. Fungsi *hash* satu arah

SHA adalah fungsi *hash* satu arah yang dibuat oleh NIST dan digunakan bersama DSS (*Digital Signature Standard*). Oleh NSA, SHA dinyatakan sebagai standard fungsi *hash* satu arah. SHA dapat dianggap sebagai kelanjutan pendahulunya, MD5, yang telah digunakan secara luas. SHA disebut aman (*secure*) karena dirancang

sedemikian rupa sehingga secara komputasi tidak mungkin menemukan pesan yang berkoresponden dengan *message digest* yang diberikan.

SHA merupakan keluarga fungsi *hash* satu-arah. Fungsi *hash* SHA yang paling umum digunakan adalah SHA-1 yangtelah diimplementasikan di dalam berbagai aplikasi dan protokol keamanan seperti TLS, SSL, PGP, S/MIME, dan Ipsec. Anggota pertama keluarga SHA adalah SHA-0 yang dipublikasikan tahun 1993. SHA-0 sering diacu sebagai SHA saja. Berikutnya pada tahun 1995 SHA-1 dipublikasikan. Empat varian lain juga telah dipublikasikan yaitu SHA-224, SHA-256, SHA-384, dan SHA-512. Keempat varian ini dianggap sebagai SHA-2.

SHA-1 menerima masukan berupa pesan dengan ukuran maksimum 2^{64} bit dan menghasilkan *message digest* yang panjangnya 160 bit, lebih panjang dari *message digest* yang dihasilkan oleh MD5 yang hanya 128 bit. Beberapa fungsi *Hash* yang ada sampai dengan saat ini dan terus dikembangkan :

Algoritma	**Ukuran *message digest* (bit)**	**Ukuran blok pesan**	**Kolisi**
MD2	128	128	Ya
MD4	128	512	Hampir
MD5	128	512	Ya
RIPEMD	128	512	Ya
RIPEMD-128/256	128/256	512	Tidak
RIPEMD-160/320	160/320	512	Tidak
SHA-0	160	512	Ya
SHA-1	160	512	Ada cacat
SHA-256/224	256/224	512	Tidak
SHA-512/384	512/384	1024	Tidak
WHIRLPOOL	512	512	Tidak

Gambar 2. Beberapa fungsi *hash*

Langkah-langkah pembuatan *message digest* dengan SHA-1 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. Ini berarti panjang pesan setelah ditambahi bit-bit pengganjal adalah 64 bit kurang dari kelipatan 512. Angka ini muncul karena SHA-1 memproses pesan dalam blok-blok yang berukuran 512. Pesan dengan panjang 448 bit pun tetap ditambah dengan bit-bit pengganjal. Jika panjang pesan 448 bit, maka pesan tersebut ditambah dengan 512 bit menjadi 960 bit. Jadi, panjang bit-bit pengganjal adalah antara 1 sam pai 512. Bit-bit pengganjal terdiri dari sebuah bit 1 diikuti dengan sisanya 0.

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 dengan 64 bit, panjang pesan sekarang menjadi kelipatan 512 bit.

3. Inisialisasi penyangga (*buffer*) MD.

   SHA membutuhkan 5 buah penyangga (*buffer*) yang masing-masing panjangnya 32 bit. Total panjang penyangga adalah 5 x 32 bit = 160 bit. Kelima penyangga ini menampung hasil antara dan hasil akhir. Kelima penyangga tersebut diberi nama A,B,C,D, dan E. Setiap penyangga

diinisialisasi dengan nilai-nilai (dalam notasi HEX) sebagai berikut :
A = 67452301
B = EFCDAB89
C = 98BADCFE
D = 10325476
E = C3D2E1F0

4. Pengolahan pesan dalam blok berukuran 512 bit.

2.4 Protokol Tanda-tangan dengan Enskripsi[1]

Protokol ini dapat dianalogikan seperti pengiriman surat yang menggunakan amplop tertutup. Tanda tangn pada surat memberikan bukti kepemilikan, hal ini sama dengan fungsi tanda-tangan digital pada dokumen elektrinis. Sedangkan amplop memberikan perlindungan keamanan (*privacy*), hal ini sama dengan fungsi enskripsi pada dokumen. Tanda-tangan digital diberikan dengan menggunakan kunci privat pengirim dan dokumen dienskripsi dengan kunci publik penerima.

(1) Alice menandatangani dokumen atau pesan (*M*) dengan menggunakan kunci privat(*A*).

$S_A(M)$

(2) Alice mengenskripsi dokumen yang sudah ditandatangani dengan kunci publik Bob (*B*) dan mengirimkannya kepada Bob.

$E_B(S_A(M))$

(3) Bob mendekripsi chiperteks yang diterima dengan kunci privatnya.

$D_B(E_B(S_A(M))) = S_A(M))$

(4) Bob melakukan verifikasi dengan mendekripsi hasil pada langkah 3 dengan menggunakan kunci publik Alice dan sekaligus mendapatkan kembali dokumen yang belum dienskripsi.

Menandatangani dokumen sebelum mengenskripsinya adalah cara yang alamiah. Dalam kehidupan sehari-hari, kita menulis surat, menandatanganinya, dan memasukkannya ke dalam amplop. Bila alice memasukkan surat ke dalam amplop, kemudian menandatangni amplop, maka keabsahannya diragukan. Jika Bob memperlihatkan surat Alice tersebut kepada Carol, maka Carol mungkin menuduh Bob berbohong tentang isi surat tersebut.

Alice tidak harus menggunakan kunci publik / kunci privat yang sama untuk enskripsi dan tanda tangan. Alice dapat menggunakan dua pasangan kunci : sepasang kunci enskripsi dan sepasang untuk pemberian tanda tangan.

Misalkan Bob ingin mengkonfirmasi bahwa dia telah menerima dokumen dari Alice. Maka, Bob mengirimkan konfirmasi "tanda terima" kepada Alice. Protokol pengiriman pesan tanda terima adalah sebagai berikut :

(1) Alice menandatangani dokumen atau pesan (*M*) dengan menggunakan kunci privatnya, mengenskripsinya dengan kunci publik Bob dan mengirimkannya kepada Bob.

$E_B(S_A(M))$

(2) Bob mendekripsi chiperteks yang diterima dengan kunci privatnya (*B*), memverifikasi tanda tangan digitak dengan kunci publik Alice dan sekaligus mendapatkan kembali dokumen yang belum dienkripsi.

$V_A(D_B(E_B(S_A(M)))) = M$

(3) Bob menandatangani dokumen (*M*) dengan kunci privatnya, mengenkripsinya dengan kunci publik Alice, dan mengirimkannya ke Alice.

$E_A(S_B(M))$

(4) Alice mendekripsi dokumen dengan kunci privatnya dan memverifikasi tanda-tangan digital dengan kunci publik Bob.

$V_A(D_B(E_B(S_A(M)))) = M'$

Jika *M*' yang dihasilkan sama dengan dokumen yang dikirimkan oleh Alice (*M*), maka Alice tahu bahwa Bob menerima dokumennya dengan benar.

2.5 RSA[1]

Banyak algoritma kunci publik yang pernah dibuat oleh kriptografer, salah satunya adalah algortima RSA. Algoritma ini 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. Keamanan algoritma RSA ini terletak pada sulitnya memfaktorkan bilangan yang besar menjadi faktor-faktor prima. Pemfaktoran dilakukan untuk memperoleh kunci privat. Selama pemfaktoran bilangan besar menjadi faktor-faktor prima belum ditemukan algoritma yang mangkus, maka selama itu pula keamanan algoritma RSA tetap terjamin.

Algoritma RSA memiliki besaran-besaran sebagai berikut :

1. p dan q bilangan prima (rahasia)
2. $n = p \cdot q$ (tidak rahasia)
3. $\varphi(n) = (p - 1)(q - 1)$ (rahasia)
4. e (kunci enskripsi) (tidak rahasia)
5. d (kunci dekripsi) (rahasia)
6. m (plainteks) (rahasia)
7. c (cipherteks) (tidak rahasia)

Perumusan algoritma RSA

Algoritma RSA didasarkan pada teorema ueler yang menyatakan bahwa :

$$A^{\varphi(n)} \equiv 1 \pmod{n}$$

dengan syarat :

1. a harus relatif prima terhadap n
2. $\varphi(n)$ adalah fungsi yang menentukan berapa banyak dari bilangan-bilangan 1,2,3,...,n yang relatif prima terhadap n.

Berdasarkan sifat $ak \; bk \pmod{n}$ untuk k bilangan bulat ≥ 1, maka persamaan dapat ditulis menjadi :

$$A^{k\varphi(n)} \equiv 1^k \pmod{n}$$

Bila a diganti dengan m, maka persamaan menjadi :

$$m^{k\varphi(n)} \equiv 1 \pmod{n}$$

dengan sifat $ac \equiv bc \pmod{n}$, maka menjadi

$$m^{k\varphi(n)+1} \equiv 1 \pmod{n}$$

dalam hal ini m relatif prima terhadap n.
Misalkan e dan d dipilih sedemikian sehingga

$$e \cdot d \equiv 1 \pmod{n} \text{ atau } e \cdot d = k\varphi(n) + 1$$

sehingga di dapatkan persamaan :

$$(m^e)^d \equiv m \pmod{n}$$

Dan rumus enskripsi dan deskripsi didapatkan sebagi berikut :

$$E(m) = c \equiv m^e \bmod n$$
$$D(c) = m \equiv c^d \bmod n$$

Algoritma membangkitkan pasangan kunci adalah sebagia berikut :

1. Pilih dua buah bilangan prima sembarang, p dan q.
2. Hitung n = p . q
3. Hitung $\varphi(n) = (p - 1)(q - 1)$
4. Pilih kunci publik, e yang relatif prima terhadap $\varphi(n)$.
5. Bangkitkan kunci privat dengan menggunakan persamaan , yaitu
   e . d ≡ 1 (mod $\varphi(n)$).

Hasil dari algoritma diatas :

- Kunci publik adalah pasangan (e , n)
- Kunci privat adalah pasangan (d, n)

Algoritma enskripsi :

1. Ambil kunci publik penerima pesan, e, dan modulus m.
2. Nyatakan plainteks m menjadi blok-blok $m1$, $m2$, ..., sedemikian sehingga tiap blok mempresentasikan nilai di dalam selang $[0, n - 1]$
3. Setiap blok $m1$ dideskripsikan menjadi blok ci dengan rumus $ci = m_i^e \bmod n$.

Deskripsi :
1. Setiap blok chiperteks ci dideskripsi kembali menjadi blok mi dengan rumus $mi = c_i^d \bmod n$

Keamanan algoritma RSA didasarkan pada sulitnya memfaktorkan bilangan besar menjadi faktor-faktor primanya.

Selama 300 tahun para matematikawan mencoba mencari pemfaktoran bilangan

yang besar namun tidak banyak membuahkan hasil. Semua bukti yang diketahui menunjukkan bahwa upaya pemfaktoran itu luar biasa ulit. Belum ditemukan algoritma pemfaktoran bilangan besar dalam waktu polinomial, tetapi juga tidak dapat dibuktikan algortima tersebut ada. Fakta inilah yang membuat algoritma *RSA* dianggap aman. Penemu algoritma RA bahkan menyarankan nilai *p* dan *q* panjangnnya lebih dari 100 angka. Dengan demikian hasil kali n = p x q akan berurukuran lebih dari 200 angka.

Menurut Rivest dan kawan-kawan, usaha untuk mencari faktor prima dari bilangan 200 angka mebutuhkan waktu komputasi selama 4 milyar tahun, sedangkan untuk bilangan 500 angka membutuhkan waktu 10^{25} tahun.

Secara umum dapat disimpulkan bahwa RSA hanya aman jika n cukup besar. Jika panjang *n* hanya 256 bit atau kurang, ia dapat difaktorkan dalam beberapa jam saja dengan sebuah komputer *PC* dan program yang tersedia secara bebas. Jika panjang n 512 bit atau kurang, ia dapat difaktorkan dengan beberapa ratus komputer.

Algoritma RSA lebih lambat dari algoritma kriptografi kunci simetri seperti DES dan AES. Dalam praktek, pesan dienskripsi dengan kunci rahasia dengan menggunakan salah satu algoritma kriptografi kunci simetri, sedangkan RSA digunakan untuk mengenskripsi kunci rahasia. Pesan dan kunci rahasia yang masing-masing sudah dienskripsi dikirim bersama-sama. Penerima pesan mula-mula mendekripsi kunci rahasia dengan kunci privatnya, lalu menggunakan kunci rahasia tersebut untuk mendekripsi pesan.

Karena pengirim dan penerima harus berbagai kunci publik, maka distribusi kunci publik dapat mengalami serangan *man-in-the-middle attack.* Misalkan Alice dan Bob mengirim kunci publiknya masing-masing melalui saluran komunikasi. Orang-di-tengah, misalkan carol, memutus komunikasi antara Bob dan Alice lalu ia berpura-pura sebagai salah satu pihak (Alice atau Bob). Carol (yang menyamar sebagai Alice) mengirimkan kunci publiknya kepada Bob (Bob percaya itu adalah kunci publik Alice) , dan Carol (yang menyamar sebagai Bob) mengirimkan kunci publiknya kepada Alice (Alice percaya itu adalah kunci dari Bob). Selanjutnya, Carol mendekripsi pesa dari Bob dengan kunci privatnya, menyimpan salinannya, lalu mengenkripsi pesan tersebut dengan kunci publik Alice, dan mengirim chiperteks tersebut kepada Alice. Alice dan Bob tidak dapat mendeteksi keberadaan carol.

2.6 Java

Bahasa pemrograman Java pertama lahir dari The Green Project, yang berjalan selama 18 bulan, dari awal tahun 1991 hingga musim panas 1992. Proyek tersebut belum menggunakan versi yang dinamakan Oak. Proyek ini dimotori oleh Patrick Naughton, Mike Sheridan, James Gosling dan Bill Joy, beserta sembilan pemrogram lainnya dari Sun Microsystems. Salah satu hasil proyek ini adalah maskot Duke yang dibuat oleh Joe Palrang.

Pertemuan proyek berlangsung di sebuah gedung perkantoran Sand Hill Road di Menlo Park. Sekitar musim panas 1992 proyek ini ditutup dengan menghasilkan sebuah program Java Oak pertama, yang ditujukan sebagai pengendali sebuah peralatan dengan teknologi layar sentuh (*touch screen*), seperti pada PDA sekarang ini. Teknologi baru ini dinamai "*7" (Star Seven).

Setelah era Star Seven selesai, sebuah anak perusahaan TV kabel tertarik ditambah beberapa orang dari proyek The Green Project. Mereka memusatkan kegiatannya pada sebuah ruangan kantor di 100 Hamilton Avenue, Palo Alto.

Perusahaan baru ini bertambah maju: jumlah karyawan meningkat dalam waktu singkat dari 13 menjadi 70 orang. Pada rentang waktu ini juga ditetapkan pemakaian Internet sebagai medium yang menjembatani kerja dan ide di antara mereka. Pada awal tahun 1990-an, Internet masih merupakan rintisan, yang dipakai hanya di kalangan akademisi dan militer.

Mereka menjadikan perambah (*browser*) Mosaic sebagai landasan awal untuk membuat perambah Java pertama yang dinamai Web Runner, terinsipirasi dari film 1980-an, Blade Runner. Pada perkembangan

rilis pertama, Web Runner berganti nama menjadi Hot Java.

Pada sekitar bulan Maret 1995, untuk pertama kali kode sumber Java versi 1.0a2 dibuka. Kesuksesan mereka diikuti dengan untuk pemeritaan pertama kali pada surat kabar *San Jose Mercury News* pada tanggal 23 Mei 1995.

Sayang terjadi perpecahan di antara mereka suatu hari pada pukul 04.00 di sebuah ruangan hotel Sheraton Palace. Tiga dari pimpinan utama proyek, Eric Schmidt dan George Paolini dari Sun Microsystems bersama Marc Andreessen, membentuk Netscape.

Nama Oak, diambil dari pohon oak yang tumbuh di depan jendela ruangan kerja "bapak java", James Gosling. Nama Oak ini tidak dipakai untuk versi release Java karena sebuah perangkat lunak sudah terdaftar dengan merek dagang tersebut, sehingga diambil nama penggantinya menjadi "Java". Nama ini diambil dari kopi murni yang digiling langsung dari biji (kopi tubruk) kesukaan Gosling.

Java adalah teknologi dan bahasa pemrograman yang berjalan pada multiflatform sesuai dengan semboyannya yaitu "*Write Once, Run Anywhere*". Pada site official Java dari Sun yaitu http://java.sun.com bisa ditemui tiga pembagian paket Java yaitu :

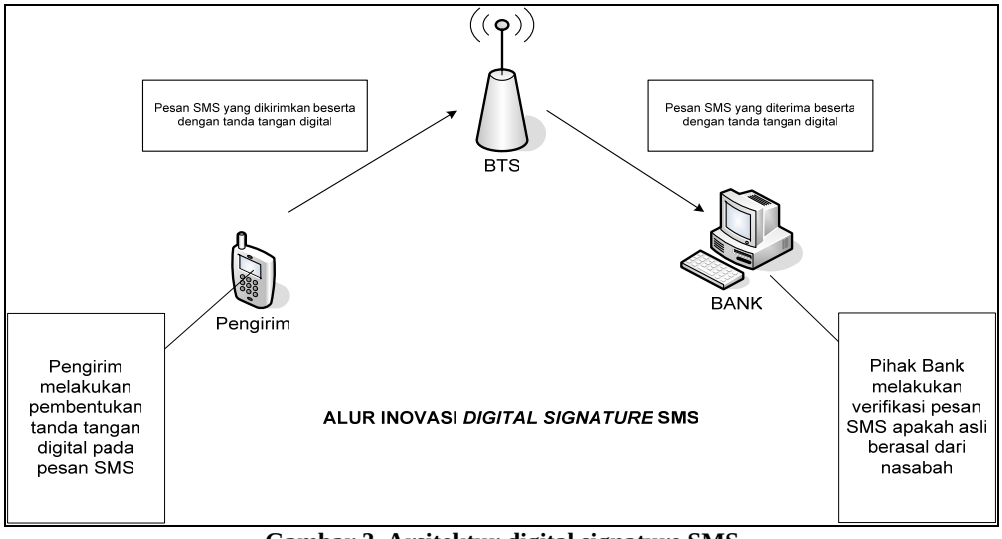
- Java 2 Enterprise Edition (J2EE).
- Java 2 Standart Editon (J2SE).
- Java 2 Micro Edition (J2ME).

Penjelasan paling simple atas pembagian tersebut berdasarkan atas perangkat keras yang
digunakan.

a. Paket J2EE digunakan pada perangkat keras yang mempunyai spesifikasi dan memory yang besar seperti pada komputer server.
b. Paket J2SE digunakan pada perangkat keras seperti komputer desktop.
c. Paket J2ME digunakan pada perangkat yang memiliki memory kecil seperti ponsel, pager atau PDA.

Paparan singkat di atas adalah penjelasan singkat mengenai Java dan sedikit gambaran dimana paket J2ME digunakan. Sebenarnya masih panjang penjelasan tentang Java dan paket J2ME, tapi tidak dibahas pada tulisan ini, mungkin bisa menjadi pekerjaan rumah buat para pembaca yang tertarik akan Java atau J2ME.

3. Arsitektur Tanda Tangan Digital SMS-Banking



Gambar 3. Arsitektur digital signature SMS

Digital Signature SMS merupakan tanda tangan digital pada pesan SMS yang merupakan nilai kriptografis dengan menggunakan prinsip-prinsip pada bidang

ilmu kriptografi. Tanda tangan digital ini selalu berbeda-beda antara satu isi pesan SMS dengan pesan SMS lain. Berikut contoh tanda tangan digital pada *file* teks :

```
TRANSFER 1000000000 121029021212
-----BEGIN PGP SIGNATURE-----

iQA/AwUAQnibsbPbxejK4Bb3EQJXvQCg8zN6UL0xnwBTPR5
FfWNt4uxh3AEAn2NC/G2VTUrLpcSyo2l/S/D/+rUl=pZeh

-----END PGP SIGNATURE-----
```

Gambar 4. Contoh tanda tangan digital SMS

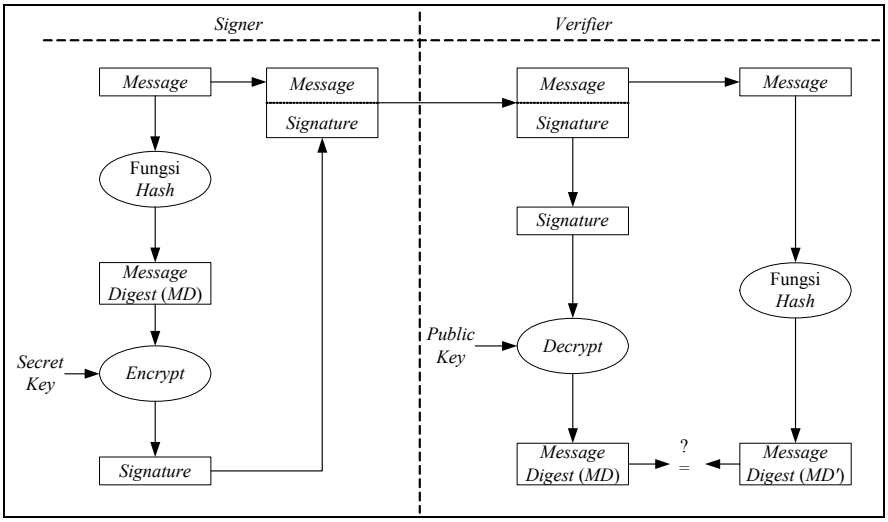
Standar yang dilakukan untuk melakukan algoritma tanda tangan digital ini adalah *Digital Signature Standard* (DSS). DSS terdiri dari dua komponen :

1. Algoritma tanda tangan digital yang disebut sebagai *Digital Signature Algorithm* (DSA).
2. Fungsi *hash* (pemetaan secara matematik) standar yang disebut *Secure Hash Algorithm* (SHA).

DSA termasuk dalam algoritma kriptografi kunci publik (kunci yang diketahui oleh penerima pesan SMS) dengan dua fungsi utama :

a. Pembentukan tanda tangan digital yaitu dengan menggunakan kunci rahasia (kunci yang hanya diketahui oleh pengirim pesan)
b. Pemeriksaan keabsahan tanda tangan dengan menggunakan kunci publik.

Proses pembuatan tanda tangan digital dapat dilihat dari gambar sebagi berikut :



Gambar 5. Proses pembentukan tanda tangan

Setiap pesan SMS yang akan dikirim akan diubah menjadi *Message Digest* (MD) dengan menggunakan fungsi *hash*. Setelah itu, akan dilakukan proses enskripsi untuk membentuk tanda tangan digital dengan menggunakan kunci *private* yang hanya diketahui oleh pengirim. Tanda tangan digital yangterbentuk tersebut akan dikirimkan beserta dengan pesan SMS.

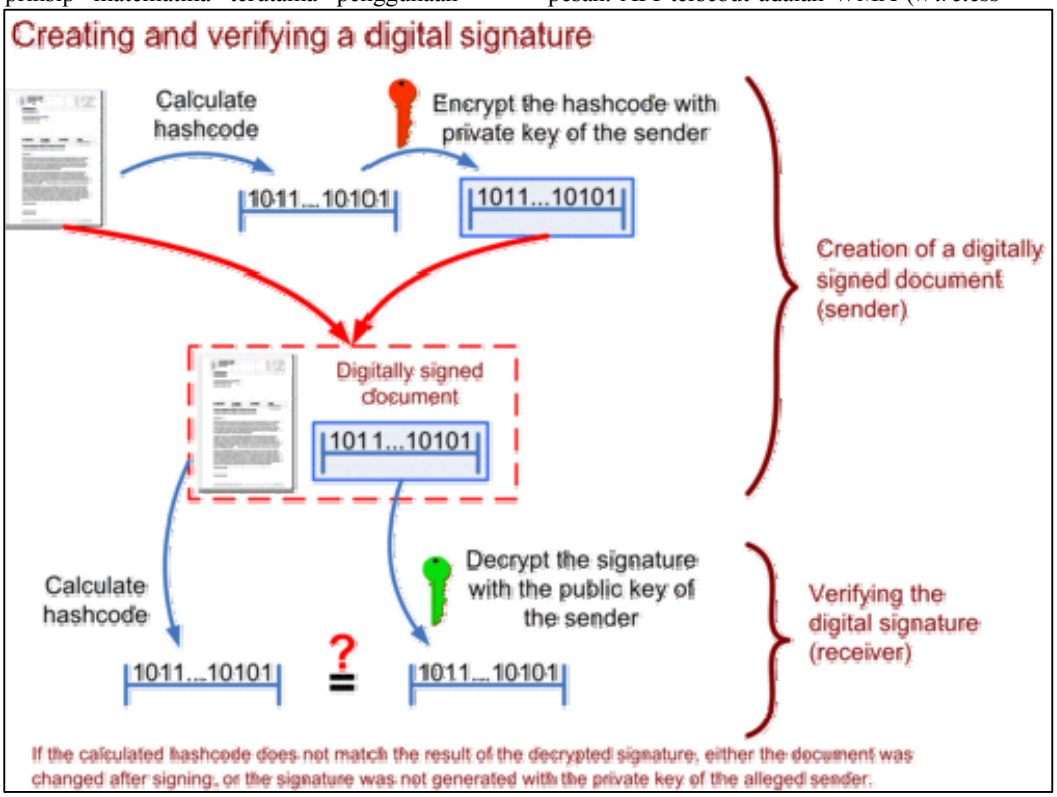
Fungsi *hash* yang digunakan merupakan suatu fungsi pemetaan yang menggunakan prinsip matematika terutama penggunaan modulo dan perpangkatan suatu bilangan. Selain itu juga digunakan prinsip pengoperasian *bit* per *bit* [3] .

Pesan SMS yang diterima diverifikasi dengan menggambil bagian tanda tangan digitalnya lalu melalukan dekripsi tanda tangan tersebut dengan kunci publik yang diketahui oleh penerima SMS. Tanda tangan yang didekrip tersebut akan menghasil MD yang kemudian dicocokan dengan hasil fungsi *hash* dari pesan SMS. Jika *Message Digest* (MD) menunjukkan kesamaan maka pesan SMS itu memang dikirimkan dari pemilik nomor ponsel yang sebenarnya.

Untuk melakukan proses pembentukan tanda tangan digital pada ponsel, akan digunakan teknologi JAVA J2ME yang mampu melakukan pengoperasian bit-bit pesan SMS secara sederhana dan tidak membuat ponsel menjadi lebih lambat dalam melakukan pengiriman pesan.

4. Implementasi Tanda Tangan Digital SMS-Baking dengan teknologi mobile J2ME

Pada J2ME disediakan API yang dapat melakukan pengiriman dan penerimaan pesan. API tersebut adalah WMA (*Wireless Messaging APIs*).



Gambar 6. Proses bit tanda tangan digital

Untuk menggunakannya, implementasi ini menggunakan perangkat lunak (*tools* penunjang) yaitu Ktoolbar yang disediakan oleh WTK (*Wireless Tool Kit*). Untuk menggunakannya, langkah-langkah yang harus dilakukan adalah :

1. Setelah aplikasi dibuka, klik *icon new project*.
2. Ketikkan nama *project* dan nama kelas *Midlet* yang akan diimplementasikan.
3. Setelah itu buka *folder home* WTK , lalu *folder apps*.
4. Pilih *folder* yang sesuai dengan nama *project* yang telah dibuat.
5. Buka *folder src* dan buat kelas (*file* java) yang mengimplementasikan Midlet dan prosedur SMS dari APIs WMA.
6. Setelah semua kelas telah dibuat, *build* aplikasi dengan menekan *icon* BUILD.
7. Jika hasil *build* menunjukkan bahwa kelas-kelas yang telah dibuat tidak mengalami *error*, maka dapat

dijalankan dengan menekan tombol *Run*, maka aplikasi akan berjalan diatas emulator yang disediakan.

8. Jika ingin melakukan *deployment* ke dalam telepon seluler yang menyediakan JVM, maka dapat dibuat *package* nya terlebih dahulu dalam bentuk JAR dengan menekan tombol menu *Project -> Package -> Create Package*.

Berikut hasil implementasi SMS-Banking J2ME dalam simulator WTK (*Wireless Tool Kit*):

1. Aplikasi akan meminta masukan nomor tujuan BANK. Nomor tujuan yang dimasukkan merupakan nomor tujuan yang memang sudah disediakan oleh pihak Bank untuk dapat melakukan transaksi SMS-Banking secara langsung. Dalam implementasi yang lebih baik nantinya, nomor masukkan in ilebih baik diimplementasikan secara langsung dari aplikasi, sehingga pengguna tidak perlu memasukkan nomor tujuan secara manual.

Implementasi dengan meminta masukkan ini dilakukan dengan sangat sederhana, karena hanya menggunakan kelas *textfield* yang sudah disediakan oleh Java untuk aplikasi yang akan dijalankan dalam telepon seluler yang mempunyai keterbatasan *storage* dan memory. Berikut potongan beberapa kode program untuk mengimplementasikan masukkan dari pengguna.

```
display= Display.getDisplay(this);

mainForm.append(new TextField("Input n",
null, 10, TextField.PASSWORD));
mainForm.append(new      TextField("Input
kunci-privat",          null,          10,
TextField.PASSWORD));

mainForm.addCommand(CMD_EXIT);
mainForm.addCommand(CMD_OK);
```

Gambar 7. Kode implementasi masukan

Gambar 8. Tampilan input nomor tujuan

2. Aplikasi akan meminta masukan pesan transaksi yang ingin dilakukan oleh nasabah. Pesan yang diketikkan merupakan pesan yang memang disediakan oleh pihak Bank untuk dapat melakukan transaksi perbankan secara *mobile*. Untuk implementasi yang lebih baik, pesan yang diminta sudah digenerate dalam sebuah pilihan menu transaksi sehingga pengguna tidak perlu menuliskan jenis transaksi yang dilakukan. Misalnya pengguna dapat memilih menu transfer tabungan ke rekenening tertentu.

Gambar 9. Tampilan input pesan transaksi

Gambar 10. Tampilan input pasangan kunci

3. Aplikasi akan meminta masukan pasangan kunci privat yang dimiliki oleh nasabah. Masukkan kunci merupakan kunci yang dimiliki oleh nasabah dan memang sepasang dengan kunci publik yang dimiliki oleh pihak Bank. Lebih baik lagi jika masukkan kunci ini dapat disimpan ke dalam telepon seluler pengguna sehingga pengguna tidak perlu mengingatnya, mungkin pengguna hanya perlu mengingat nilai bilangan kunci privat, tidak perlu nilai bilangan n.

4. Aplikasi memastikan kembali bahwa pesan yang akan dikirim dan tanda tangan benar. Pesan beserta tanda tangan yang dihasilkan merupakan kesatuan paket transaksi yang nantinya akan dienskripsi kembali menjadi sebuah paket seperti halnya satu surat dalam suatu amplop.

Gambar 11. Hasil tanda tangan digial SMS

Setelah pesan beserta tanda tangan terbentuk maka akan dilakukan proses enskripsi sehingga ketika akan dikirimkan sudah dalam keadaan satu paket pesan transaksi. Berikut potongan kode untuk melakukan pengiriman SMS dan RSA dengan menggunakan J2ME.

```
String aliceSignature =
obj.doRSA(

aliceEncodedDigest,
aliceKeys.d,aliceKeys.n,
blockSize

);
```

Gambar 12. Kode implementasi pembentukan tanda tangan

```
String        address        =
destinationAddressBox.getString
();

if(!SMSSend.isValidPhoneNumber(
address))
{

      errorMessageAlert.setStri
      ng("Invalid          phone
      number");

      display.setCurrent(errorM
      essageAlert,
      destinationAddressBox);
      return;
}

String statusMessage = "Sending
message to " + address + "...";

sendingMessageAlert.setString(s
tatusMessage);

sender.promptAndSend("sms://"  +
address);
```

Gambar 13. Kode implementasi pengiriman SMS

5. Kesimpulan

Dari analisa yang didapatkan dari hasil implementasi, dapat disimpulkan bahwa tanda tangan digital terhadap sebuah pesan dapat dilakuakan terhadap pesan SMS melalui perangkat seluler. Untuk itu hal ini sangat memungkinkan untuk dapat diterapkan dalam proses transaksi SMS-Banking dimana sistem keamanan SMS-Banking saat ini masih mempunyai kekurangan dalam keamanan di level non teknis.

Selain itu, dapat juga disimpulkan bahwa perangkat seluler mampu melakukan fungsi SHA dan RSA untuk membentuk tanda tangan digital dengan memanfaatkan APIs yang tersedia.

Secara garis besar tulisan ini telah mampu membuktikan penerapan *digital signature* untuk keamanan transaksi SMS-Banking dengan menggunakan teknik fungsi SHA dan RSA yang cukup sederhana namun

mampu menampung bilangan pasangan kunci yang cukup besar.

```
class RSA
{
  private BigInteger n, d, e;

  public Rsa(int bitlen)
  {
    SecureRandom r = new SecureRandom();
    BigInteger p = new BigInteger(bitlen / 2, 100, r);
    BigInteger q = new BigInteger(bitlen / 2, 100, r);
    n = p.multiply(q);
    BigInteger m = (p.subtract(BigInteger.ONE))
                   .multiply(q.subtract(BigInteger.ONE));
    e = new BigInteger("3");
    while(m.gcd(e).intValue() > 1) e = e.add(new BigInteger("2"));
    d = e.modInverse(m);
  }
  public BigInteger encrypt(BigInteger message)
  {
    return message.modPow(e, n);
  }
  public BigInteger decrypt(BigInteger message)
  {
    return message.modPow(d, n);
  }
}
```

Gambar 14. Kode implementasi kelas RSA

Referensi

[1] Munir, Ir. Rinaldi, " Kriptografi ", 2006.

[2] Yudi Somantri, Miftahuljannah Saleh, Ratna Ariyanti, "M-Banking, Aman !, www.trendigital.com/31082003/Artikel/ , 2006.

[3] R.L. Rivest, A. Shamir, L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Massachusetts Institute of Technology, Cambridge, 1994.

[4] R. Romzi Imron, "Membuat Sendiri SMS Gateway Berbasis Protokol SMPP", ANDI, Yogyakarta, 2004.

[5] G. Baldwin Ricahrd , "Digital Signature using Message Digest with Java ", www.developer.com, 2006.

[6] Lei Yu, "Generating Digital Signature on Mobile Device", Zhejiang University, 2002.

[7] Public Key Cryptography Standards(PKCS), No.1, RSA Encryption standard, http://www.rsasecurity.com/rsalabs/.

[8] National Institute for Standards and Technology. Digital Signature Standard (DSS). *Technical Report 169*, August 30, 1991.

[9] N. Asokan, G. Tsudik, M. Waidner. Server-supported signatures. *Journal of Computer Security*, Volume 5, Issue 1, pages 91–108, January 1997.