Pembangunan Tanda Tangan Digital dengan Memanfaatkan Algoritma RSA dan Modifikasi dari Fungsi SHA-1

Zakiy Firdaus Alfikri - 13508042
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
zakiy_f_a@yahoo.co.id

***Abstrak*—Tanda tangan digital adalah suatu cara yang dilakukan untuk memberikan otentikasi pada data digital seperti pesan dan dokumen elektronik. Tanda tangan digital selalu berbeda-beda antara satu isi dokumen dengan dokumen lain. Ada dua cara menandatangani pesan, yaitu pertama dengan melakukan enkripsi pesan dan kedua dengan menggunakan kombinasi fungsi hash dan kriptografi kunci-publik. Cara yang akan dilakukan pada implementasi ini adalah cara kedua dengan menggunakan fungsi hash yang telah dimodifikasi.**

Algoritma RSA adalah algoritma kunci publik yang banyak digunakan. Algoritma RSA juga bisa dimanfaatkan sebagai algoritma kunci publik yang digunakan dalam pembangunan tanda tangan digital.

Fungsi hash adalah adalah fungsi yang melakukan konversi suatu variable-sized data ke dalam suatu datum kecil yang biasanya berbentuk sebuah integer. Salah satu fungsi hash yang ada adalah SHA-1. SHA-1 merupakan salah satu fungsi hash yang cukup populer dan banyak digunakan. Akan tetapi fungsi SHA-1 masih memiliki kelemahan karena bisa terjadi collision attack, yaitu di mana dua data atau pesan yang berbeda bisa menghasilkan nilai hash yang sama. Oleh karena itu dilakukan modifikasi terhadap fungsi SHA-1 sehingga bisa menghasilkan tanda tangan digital yang lebih baik.

***Index Terms*—Tanda tangan digital, RSA, hash, SHA1, modifikasi.**

I. PENDAHULUAN

Pesan atau data merupakan hal yang penting dalam kehidupan manusia. Pertukaran pesan atau data menjadi suatu hal yang penting juga dan sering terjadi dalam kehidupan manusia. Permasalahan yang sering terjadi adalah sulit membuktikan apakah pesan atau data yang dipertukarkan itu masih dalam keadaan asli atau sudah mengalami perubahan.

Untuk pesan atau data non-digital bisa digunakan tanda tangan pembuat pesan untuk menyatakan bahwa pesan yang dikirim merupakan pesan asli dari pembuat pesan. Untuk pesan atau data digital digunakan tanda tangan digital yang menyatakan isi dari pesan yang dikirimkan.

Walaupun bernama tanda tangan digital, tanda tangan digital berbeda dengan tanda tangan biasa. Tanda tangan digital tidak selalu sama untuk pembuat pesan yang sama. Akan tetapi tanda tangan digital akan berbeda-beda untuk setiap isi pesan yang berbeda. Jadi tanda tangan digital merepresentasikan isi dari pesan bukan pembuat pesan. Penerima pesan bisa mengetahui keaslian pesan dengan mencocokkan isi pesan dengan tanda tangan digital yang menempel pada pesan yang dikirimkan.

Tanda tangan digital bisa dibuat dengan dua buah cara. Pertama dengan melakukan enkripsi pesan. Kedua dengan menggunakan kombinasi fungsi hash dan kriptografi kunci publik. Pada makalah ini akan dibahas pembangunan tanda tangan digital dengan cara yang kedua, yaitu dengan menggunakan kombinasi fungsi hash dan kriptografi kunci publik.

Fungsi hash diperlukan dalam pembangunan tanda tangan digital karena fungsi hash adalah fungsi yang bisa melakukan konversi suatu data ke dalam sebuah datum kecil yang biasanya berbentuk integer. Walaupun hasil yang diperoleh dari fungsi hash adalah sebuah datum kecil, hasil fungsi hash ini harus bisa merepresentasikan keseluruhan isi pesan. Jadi jika pesan mengalami perubahan walaupun hanya beberapa byte data, hasil fungsi hash juga akan berubah.

Salah satu fungsi hash yang bisa dipakai dalam pembangunan tanda tangan digital adalah fungsi SHA-1. SHA-1 merupakan sebuah fungsi hash yang akan menghasilkan 160-bit message digest.

SHA-1 merupakan fungsi hash yang sering digunakan di banyak implementasi aplikasi dan protokol. Walaupun merupakan salah satu fungsi hash yang banyak digunakan, SHA-1 memiliki kelemahan. Ada kemungkinan collision dalam hasil yang diperoleh dengan menggunakan SHA-1, yang artinya sebuah pesan atau data yang berbeda memiliki nilai hash yang sama jika menggunakan SHA-1.

Perlu dilakukan perbaikan dan modifikasi pada algoritma SHA-1 agar kemungkinan terjadinya collision diperkecil atau bahkan dihilangkan sama sekali sehingga pembangunan tanda tangan digital bisa menjadi lebih baik lagi. Perbaikan dan modifikasi yang dilakukan akan dibahas dan dijelaskan pada makalah ini.

II. DASAR TEORI

A. Tanda Tangan Digital

Tanda tangan digital adalah suatu bagian dari aspek keamanan dalam kriptografi yang dibuat untuk mengatasi masalah-masalah sebagai berikut.

- Otentikasi
- Keaslian pesan
- Anti-penyangkalan

Seperti pada pesan atau data dalam bentuk cetak, tanda tangan digunakan sebagai otentikasi pesan atau data tersebut, pesan cetak yang ditandatangani juga menjamin keaslian pesan dan tidak bisa disangkal. Walaupun begitu tanda tangan pada pesan atau data cetak tidak bisa menjamin seutuhnya.

Tanda tangan cetak merupakan representasi dari pembuat pesan sedangkan tanda tangan digital adalah representasi dari isi pesan itu sendiri. Jadi tanda tangan cetak akan sama untuk pembuat pesan yang sama sedangkan tanda tangan digital akan berbeda walaupun pembuat pesan adalah orang yang sama, tanda tangan digital bergantung pada isi pesan bukan pada pembuat pesan.

Ada dua buah cara dalam menandatangani suatu pesan secara digital. Yang pertama adalah dengan menggunakan enkripsi pesan. Dan yang kedua adalah dengan menggunakan kombinasi fungsi hash dan kriptografi kunci publik.

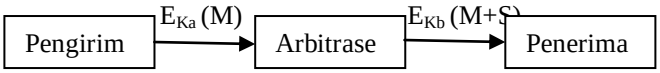
B. Penandatanganan Digital dengan Cara Mengenkripsi Pesan

Penandatanganan dengan cara mengenkripsi pesan bisa dilakukan dengan dua cara. Dua cara tersebut adalah penandatanganan dengan menggunakan kriptografi simetri dan penandatanganan dengan menggunakan kriptografi kunci publik.

Penandatanganan dengan menggunakan kriptografi simetri memberikan solusi untuk otentikasi pengirim dan keaslian pesan karena kunci simetri hanya diketahui oleh pengirim dan penerima. Akan tetapi cara ini tidak menyediakan mekanisme untuk anti-penyangkalan.

Agar dapat mengatasi masalah penyangkalan maka diperlukan pihak ketiga yang dipercaya baik oleh pengirim maupun penerima yang disebut dengan penengah atau arbitrase. Arbitrase akan memberikan kunci rahasia Ka kepada pengirim dan kunci rahasia Kb kepada penerima.

Pengiriman pesan akan dilakukan dengan cara sebagai berikut. Pengirim akan mengenkripsi pesan M dengan menggunakan kunci Ka lalu mengirimkan hasilnya kepada arbitrase. Arbitrase akan mendekripsinya dengan kunci Ka dan menambahkan pesan tambahan yang merupakan pernyataan bahwa pesan tersebut adalah dari pengirim. Arbitrase lalu mengenkripsinya dengan menggunakan kunci Kb dan mengirimkannya pada penerima. Penerima akan mendekripsi pesan tersebut dengan kunci Kb yang ia miliki. Penerima akan mendapatkan pesan M dan S dari hasil dekripsi pesan yang dikirimkan oleh arbitrase tersebut.



Gambar 1. Skema pengiriman pesan dengan penandatangan digital menggunakan kriptografi simetri.

Penandatanganan dengan menggunakan pengenkripsian juga bisa dilakukan dengan menggunakan kriptografi kunci publik. Proses penandatanganan digital dengan menggunakan kriptografi kunci publik adalah sebagai berikut.

Pesan akan dienkripsi dengan menggunakan kunci privat oleh pengirim. Setelah pesan yang sudah dienkripsi tersebut diterima oleh penerima, maka penerima akan mendekripsi pesan tersebut dengan menggunakan kunci publik pengirim.

Proses menandatantangani pesan (oleh pengirim):

$$S = E_{SK}(M)$$

Proses membuktikan otentikasi pesan (oleh penerima):

$$M = D_{PK}(S)$$

Keterangan:

SK = *secret key* = kunci privat pengirim
PK = *public key* = kunci publik pengirim
E = fungsi enkripsi *D* = fungsi dekripsi
M = pesan semula
S = *signature* = hasil enkripsi pesan

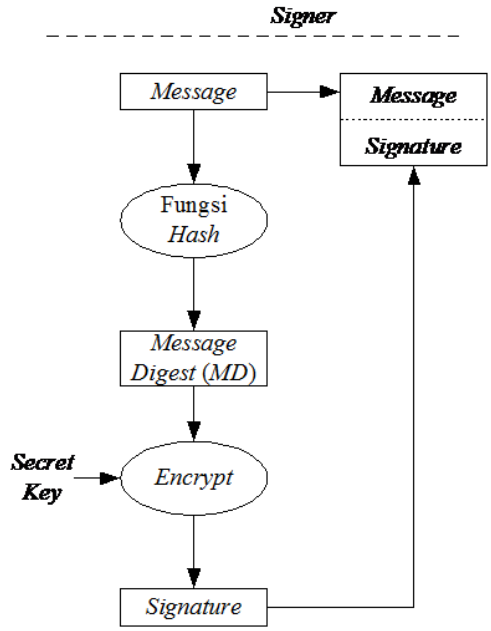
Dengan cara yang telah dijelaskan tersebut maka kerahasiaan pesan akan terjamin dan otentikasi pesan pun tercapai. Pada cara ini tidak diperlukan adanya arbitrase seperti pada penggunaan kunci simetri.

C. Penandatanganan Digital dengan Cara Menggunakan Kriptografi Kunci Publik dan Fungsi Hash

Penandatanganan pesan dengan cara mengenkripsi memberikan dua tujuan yang berbeda yaitu kerahasiaan pesan dan juga otentikasi pesan. Akan tetapi pada banyak kasus seringkali kerahasiaan pesan tidak dibutuhkan. Pada kasus-kasus tersebut hanya otentikasi pesan yang dibutuhkan, tidak untuk kerahasiaan pesannya. Untuk masalah seperti ini penandatanganan dengan menggunakan kriptografi kunci publik dan fungsi hash akan diperlukan.

Berikut ini adalah skema yang dipakai pembangunan tanda tangan digital dengan menggunakan kriptografi

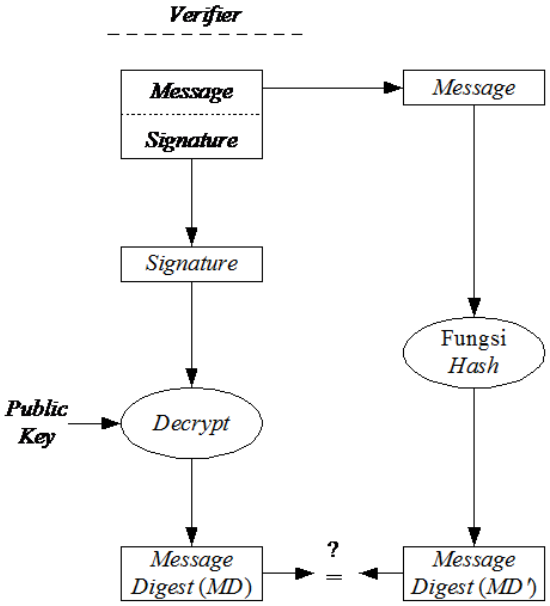
kunci publik dan fungsi hash.



Gambar 2. Skema pembangunan tanda tangan digital dengan menggunakan algoritma kriptografi kunci publik dan fungsi hash.

Bisa dilihat pada skema, pada saat pembangunan tanda tangan digital pesan yang akan dikirim akan dihash dengan menggunakan fungsi hash. Message digest yang dihasilkan akan dienkripsi dengan menggunakan kunci privat dari pengirim. Hasil enkripsi inilah yang disebut dengan tanda tangan digital. Selanjutnya tanda tangan digital akan ditambahkan dalam pesan yang akan dikirim, lalu dikrimkan pada penerima.

Berikut ini adalah skema yang dilakukan penerima pesan untuk memverifikasi keotentikan pesan yang sudah diterima.



Gambar 3. Skema verifikasi pesan oleh pengguna.

Dalam skema tersebut bisa dilihat, penerima akan menerima pesan yang sudah ditambahi dengan tanda tangan digital yang dibuat oleh pengirim. Pesan dan tanda tangan digital akan dipisahkan oleh penerima. Tanda tangan digital akan didekripsi dengan menggunakan kunci publik yang dimiliki penerima sehingga akan menghasilkan message digest dari pesan yang dikirim. Pesan yang telah dipidahkan akan dihash dengan menggunakan fungsi hash sehingga akan menghasilkan message digest dari pesan yang diterima. Selanjutnya kedua message digest yang dihasilkan akan dibandingkan nilainya. Jika nilai dari kedua message digest sama maka pesan yang diterima merupakan pesan otentik yang dikirim oleh pengirim. Jika nilai dari kedua message digest berbeda maka berarti sudah ada perubahan pada pesan yang diterima.

Dua algoritma kriptografi yang sering digunakan dalam pembangunan tanda tangan digital adalah algoritma RSA dan ElGamal. Selain itu juga terdapat algoritma lain yang dikhususkan untuk pembangunan tanda tangan digital. Algoritma tersebut adalah Digital Signature Algorithm atau DSA yang merupakan standar untuk Digital Signature Standard (DSS). Pada DSA, algoritma pembangunan tanda tangan dan verifikasi berbeda, tidak seperti RSA yang identik dalam proses pembangunan tanda tangan digital dan verifikasinya karena keidentikan algoritma enkripsi dan dekripsi pada RSA.

D. Algoritma RSA

Algoritma RSA adalah algoritma kunci publik yang banyak digunakan dan paling banyak aplikasinya. Keamanan algoritma RSA terletak pada sulitnya memfaktorkan bilangan yang besar menjadi faktor-faktor prima.

Algoritma RSA melakukan perhitungan terhadap properti-properti pada algoritma RSA. Properti-properti tersebut adalah sebagai berikut.

- p dan q bilangan prima (rahasia)
- n = p . q (tidak rahasia)
- $\phi(n) = (p-1)(q-1)$ (rahasia)
- e; PBB(e, $\phi(n)$)=1 (tidak rahasia)
- d; $d \equiv e^{-1} \bmod (\phi(n))$ (rahasia)
- m (plain teks) (rahasia)
- c (cipher teks) (tidak rahasia)

Penurunan rumus RSA mengunakan prinsip teorema Euler $a^{\phi(n)} \equiv 1 \pmod{n}$. Syarat-syarat yang harus dipenuhi dalam penggunaan teorema Euler dalam penurunan rumus RSA adalah sebagai berikut.

- a harus relatif prima terhadap n
- $\phi(n)$ = Toitent Euler = fungsi yang menentukan

berapa banyak dari bilangan-bilangan 1, 2, 3, …, *n* yang relatif prima terhadap *n*

Untuk penurunan rumus RSA, dilakukan sebagai berikut. Jika e dan d dipilih sedimikian rupa sehingga:

$$e \cdot d \equiv 1 \pmod{\phi(n)}$$

Maka,

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

$$\downarrow$$

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

Dari penurunan di atas didapatkan rumus untuk melakukan enkripsi dan dekripsi dengan menggunakan algoritma RSA.

Enkripsi: $E_e(m) = c \equiv m^e \bmod n$

Dekripsi: $D_d(c) = m \equiv c^d \bmod n$

Untuk pembangkitkan sepasang kunci yang dibutuhkan untuk melakukan enkripsi dan dekripsi dapat dilakukan dengan langkah-langkah yang akan dijabarkan berikut ini.

1. Pilih dua bilangan prima, *p* dan *q* (rahasia)
2. Hitung $n = pq$.
3. Hitung $\phi(n) = (p-1)(q-1)$.
4. Pilih sebuah bilangan bulat *e* untuk kunci publik, sebut, *e* relatif prima terhadap $\phi(n)$.
5. Hitung kunci dekripsi, *d*, dengan persamaaan $ed \equiv 1$ (**mod** $\phi(n)$) atau $d \equiv e^{-1} \bmod (\phi(n))$

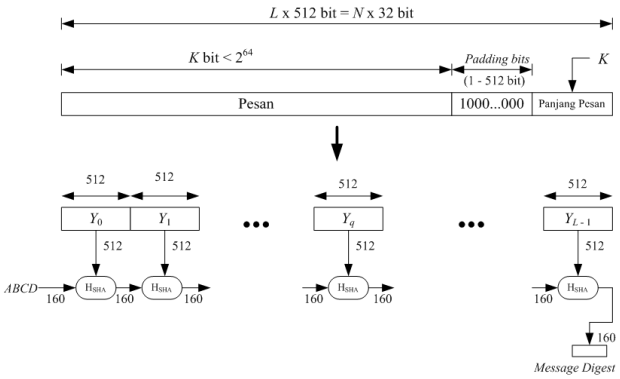
Jika sudah ada kunci privat dan kunci publik yang dihasilkan serta dengan menggunakan rumus RSA yang telah dijelaskan sebelumnya maka dapat dilakukan pengenkripsian pesan. Pesan yang dienkripsi dibagi terlebih dahulu menjadi blok-blok kecil. Proses enkripsi dan dekripsi akan dilakukan perblok-blok pesan.

D. Fungsi SHA-1

SHA merupakan standar fungsi hash satu arah. Salah satu jenis SHA yang ada adalah SHA-1. SHA-1 akan menerima masukan berupa pesan dengan ukuran maksimum 2^{64} bit dan menghasilkan message digest yang mempunyai panjang sebesar 160 bit, lebih panjang dari message digest yang dihasilkan oleh MD5.

Langkah-langkah yang dilakukan dalam pembuatan message digest dengan menggunakan SHA-1 adalah sebagai berikut. Yang pertama adalah penambahan bit-bit pengganjal atau padding bits. Lalu penambahan nilai panjang pesan semula. Lalu inisialisasi penyangga atau buffer message digest. Berikutnya adalah pengolahan pesan dalam blok berukuran 512 bit.

Berikut ini adalah skema pembuatan message digest dengan meggunakan fungsi SHA-1 sesuai langkah-langkah yang sudah dijelaskan.

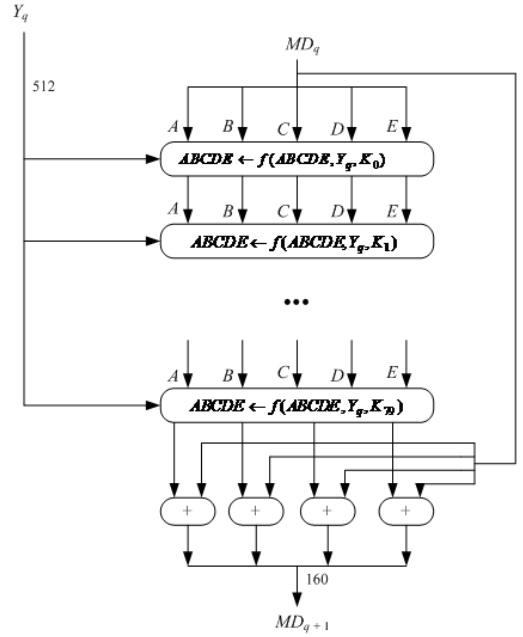


Gambar 4. Skema pembuatan message digest dengan menggunakan SHA-1

SHA-1 membutuhkan 5 buah penyangga atau buffer yang masing-masing panjangnya 32 bit. Sehingga total panjang penyangga adalah 160 bit. Kelima penyangga ini diberi nama A, B, C, D, dan E. Masing-masing penyangga tersebut diinisialisasi dengan nilai-nilai sebagai berikut.

A = 67452301
B = EFCDAB89
C = 98BADCFE
D = 10325476
E = C3D2E1F0

Dalam SHA-1 pesan dibagi menjadi blok-blok kecil berukuran 512 bit untuk kemudian diproses dalam fungsi SHA-1 ini. Berikut ini adalah skema dari pengolahan blok 512 bit di dalam proses SHA-1 yang dinamakan proses H_{SHA}.



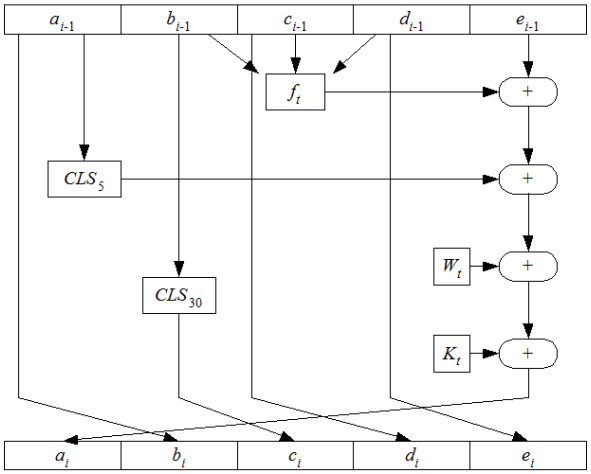
Gambar 5. Skema pengolahan blok 512 bit dalam fungsi SHA-1, proses H_{SHA}.

Proses H_{SHA} terdiri dari 80 kali putaran dan masing-masing putaran menggunakan bilangan penambah K_t.

Nilai K_t ditentukan sebagai berikut.

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

Berikut ini adalah skema detail operasi dasar yang terjadi pada setiap putaran proses H_{SHA}.



Gambar 6. Skema operasi dasar pada setiap putaran proses H_{SHA}.

f_t pada operasi di atas merupakan fungsi logika yang bmengikuti aturan sebagai berikut untu setiap putaran prosesnya.

Putaran	$f(b, c, d)$
0 .. 19	$(b \wedge c) \vee (\neg b \wedge d)$
20 .. 39	$b \oplus c \oplus d$
40 .. 59	$(b \wedge c) \vee (b \wedge d) \vee (c \wedge d)$
60 .. 79	$b \oplus c \oplus d$

Lalu nilai W_1 sampai W_{16} berasal dari 16 word pada blok yang sedang diproses, sedangkan nilai W_t berikutnya didapat dari persamaan berikut.

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

Fungsi SHA-1 merupakan fungsi SHA yang performansinya bagus. Akan tetapi SHA-1 Akan tetapi masih bisa dilakukan kriptanalisis pada fungsi SHA-1 ini. Ditemukan kasus yang menyebabkan kolisi pada SHA-1 yang telah direduksi menjadi 53 putaran. Dan bahkan ditemukan juga kolisi pada versi penuh SHA-1 yang mengalami putaran sebanyak 80 kali. Serangan ini membutuhkan sekitar 2^{69} operasi.

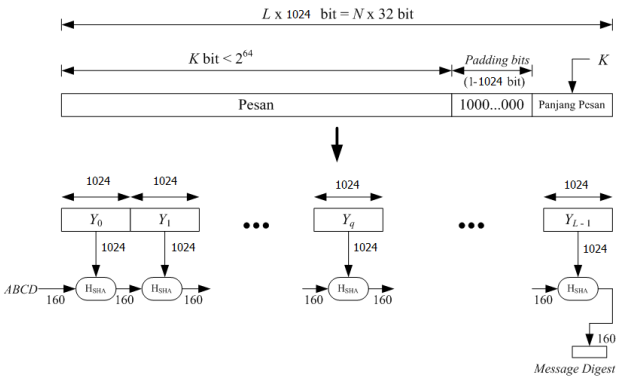
III. RANCANGAN MODIFIKASI SHA-1

Modifikasi SHA-1 yang akan dirancang dan diimplementasikan adalah modifikasi SHA-1 menjadi sebuah fungsi hash yang lebih sederhana sehingga hasil implementasi pada aplikasi yang menghasilkan tanda tangan digital menjadi lebih ringan dan cepat.

Modifikasi yang dilakukan akan dijelaskan menurut langkah-langkah pembuatan message digest dengan menggunakan SHA-1 agar terlihat perubahannya. Modifikasi untuk tiap langkahnya akan dijabarkan di bawah ini.

Langkah-langkah yang dilakukan dalam pembuatan message digest dengan menggunakan SHA-1 standar adalah sebagai berikut. Yang pertama adalah penambahan bit-bit pengganjal atau padding bits. Lalu penambahan nilai panjang pesan semula. Lalu inisialisasi penyangga atau buffer message digest. Berikutnya adalah pengolahan pesan dalam blok berukuran 512 bit. Pada modifikasi SHA-1 yang dirancang untuk pembangunan tanda tangan digital dalam makalah ini pada dasarnya mengikuti langkah-langkah yang sama pada SHA-1 standar. Perbedaan ada pada detail tiap langkahnya. Selain itu panjang blok bukan 512 bit melainkan 1024 bit. Hal ini bertujuan untuk mengurangi jumlah proses yang akan memproses setiap blok data.

Berikut ini adalah skema pembuatan message digest dengan meggunakan modifikasi dari fungsi SHA-1 sesuai langkah-langkah yang sudah dijelaskan.



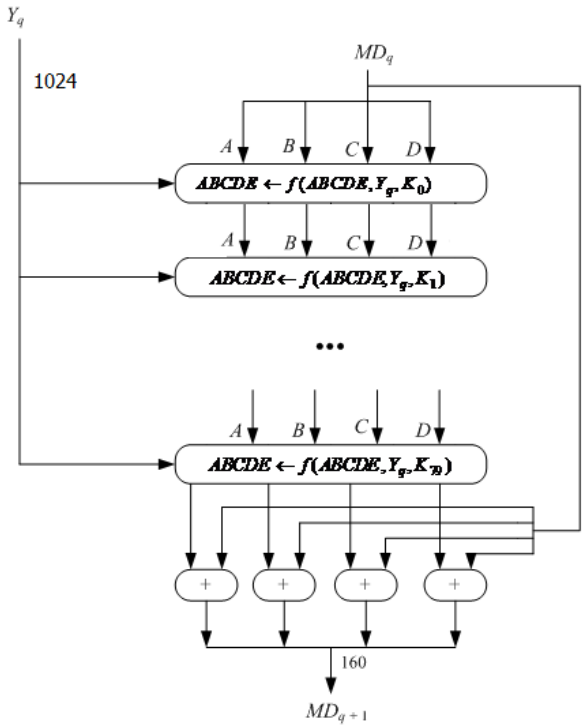
Gambar 7. Skema pembuatan message digest dengan menggunakan modifikasi dari SHA-1

SHA-1 membutuhkan 5 buah penyangga atau buffer yang masing-masing panjangnya 32 bit. Kelima penyangga ini diberi nama A, B, C, D, dan E. Pada modifikasi SHA-1 yang dilakukan juga digunakan penyangga tapi hanya sebanyak 4 buffer yang masing-masing panjangnya 32 bit. Keempat penyangga ini diberi nama A, B, C, dan D.

Masing-masing penyangga tersebut diinisialisasi dengan nilai-nilai sebagai berikut.

A = 67452301
B = CDAB8998
C = DCFE1032
D = 76C3D2E1

Seperti sudah dijelaskan sebelumnya, dalam modifikasi SHA-1 ini pesan dibagi menjadi blok-blok kecil berukuran 1024 bit untuk kemudian diproses dalam fungsi modifikasi SHA-1 ini. Berikut ini adalah skema dari pengolahan blok 1024 bit di dalam proses SHA-1 yang dinamakan proses H_{SHA}.



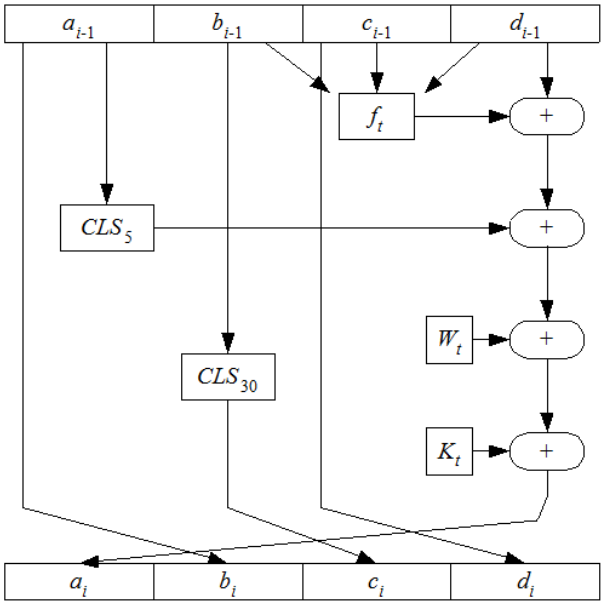
Gambar 8. Skema pengolahan blok 1024 bit dalam modifikasi fungsi SHA-1, proses H_{SHA}.

Pada fungsi SHA-1 standar proses H_{SHA} terdiri dari 80 kali putaran. Pada fungsi modifikasi SHA-1 proses H_{SHA} hanya terdiri dari 60 kali putaran. Hal ini untuk mengurangi jumlah proses dalam pembangkitan message digest pada fungsi modifikasi SHA-1.

Pada masing-masing putaran digunakan bilangan penambah K_t. Nilai K_t ditentukan sebagai berikut.

Putaran $0 \le t \le 19$ K_t = 5A827999
Putaran $20 \le t \le 39$ K_t = D9EBA18F
Putaran $40 \le t \le 59$ K_t = BCDCCA62

Berikut ini adalah skema detail operasi dasar yang terjadi pada setiap putaran proses H_{SHA} pada modifikasi fungsi SHA-1 yang dibuat.



Gambar 9. Skema operasi dasar pada setiap putaran proses H_{SHA} pada fungsi modifikasi SHA-1.

f_t pada operasi di atas merupakan fungsi logika yang bmengikuti aturan sebagai berikut untu setiap putaran prosesnya.

Putaran	$f(b, c, d)$
0 .. 19	$(b \wedge c) \vee (\neg b \wedge d)$
20 .. 39	$b \oplus c \oplus d$
40 .. 59	$(b \wedge c) \vee (b \wedge d) \vee (c \wedge d)$
60 .. 79	$b \oplus c \oplus d$

Lalu nilai W_1 sampai W_{16} berasal dari 16 word pada blok yang sedang diproses, sedangkan nilai W_t berikutnya didapat dari persamaan berikut.

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

Modifikasi dilakukan dengan menggunakan langkah-langkah yang sudah dijelaskan di atas. Langkah-langkah pada modifikasi SHA-1tidak berbeda jauh dari SHA-1 standar, akan tetapi terdapat perubahan-perubahan dalam detail-detail proses tiap langkahnya.

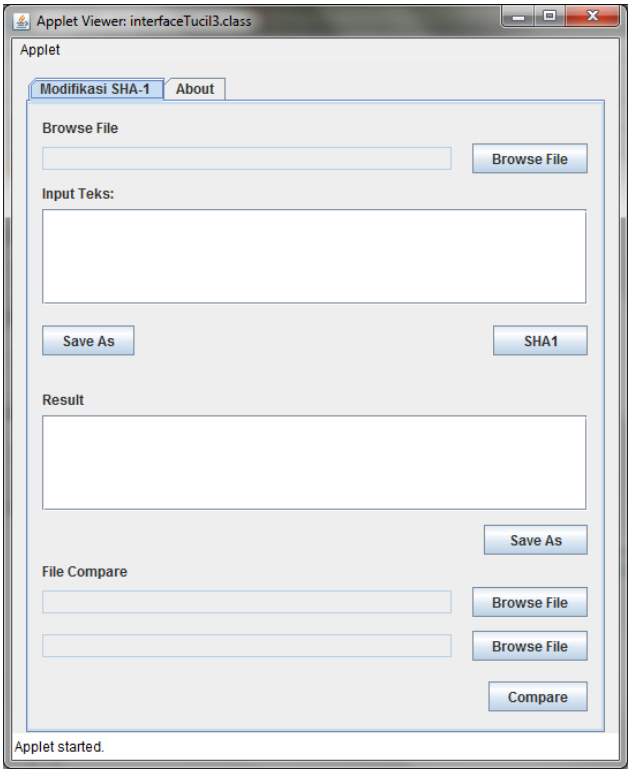
IV. IMPLEMENTASI

Modifikasi dari fungsi SHA-1 yang sudah dirancang akan diimplementasikan dengan menggunakan bahasa pemrograman Java dengan menggunakan NetBeans IDE 6.9.1 dalam lingkungan Windows 7.

Implementasi akan mengikuti langkah-langkah yang sudah dijelaskan sebelumnya untuk modifikasi fungsi SHA-1. Berikut ini adalah fungsi utama dari program hasil implementasi modifikasi dari fungsi SHA-1.

```
  private void getHash(byte[] input) {

     byte[] message = preprocess_input(input);

     System.arraycopy(h, 0, digest, 0,
digest.length);

     int chunkCount = message.length / 64;

     for (int i = 0; i < chunkCount; i++)  {

        for (int j = 0; j < 16; j++)  {
             w[j] = arrayOfByteToInt(message, (i
* 64) + (j * 4));
        }
        for (int j = 16; j <= 59; j++) {
             w[j] = leftrotate(w[j-3] ^ w[j-8] ^
w[j-14] ^ w[j-16], 1);
        }
        int[] abcde = new int[4];
        System.arraycopy(h, 0, abcde, 0,
abcde.length);

        for (int j = 0; j <= 19; j++) {
             int f = (abcde[1] & abcde[2]) |
(~abcde[1] & abcde[3]);
             int temp = leftrotate(abcde[0], 4)
+ f + abcde[3] + w[j] + 0x5a827999;
             abcde[3] = abcde[2];
             abcde[2] = leftrotate(abcde[1],
30);
             abcde[1] = abcde[0];
             abcde[0] = temp;
         }
         for (int j = 20; j <= 39; j++) {
             int f = abcde[1] ^ abcde[2] ^
abcde[3];
             int temp = leftrotate(abcde[0], 4)
+ f + abcde[3] + w[j] + 0xd9eba18f;
             abcde[3] = abcde[2];
             abcde[2] = leftrotate(abcde[1],
30);
             abcde[1] = abcde[0];
             abcde[0] = temp;
         }
         for (int j = 40; j <= 59; j++) {
             int f = (abcde[1] & abcde[2]) |
(abcde[1] & abcde[3]) | (abcde[2] & abcde[3]);
             int temp = leftrotate(abcde[0], 4)
+ f + abcde[3] + w[j] + 0xbcdcca62;
             abcde[3] = abcde[2];
             abcde[2] = leftrotate(abcde[1],
30);
             abcde[1] = abcde[0];
             abcde[0] = temp;
         }

         for (int j = 0; j < abcde.length; j++){
             digest[j] = digest[j] + abcde[j];
         }

         for (int n = 0; n < w.length; n++) {
             w[n] = 0;
         }
      }
   }
```

Program di atas diimplementasikan dengan menggunakan interface walaupun bisa juga dijalankan di konsol. Berikut ini adalah tampilan dari program implementasi dari modifikasi fungsi SHA-1.



Gambar 10. Interface program implementasi modifikasi fungsi SHA-1.

Berikut ini adalah hasil dari pembuatan message digest dari file txt yang berisi data sebagai berikut.

```
nama saya zakiy
```

Hasil message digest yang didapat dari implementasi modifikasi fungsi SHA-1 dari file txt tersebut adalah sebagai berikut.

```
9A7621FFA86CEF9E5C9E85812B47118D00000000
```

Untuk tanda tangan digital masih memakai tambahan program lain, yaitu program untuk mengenkripsi maupun mendekripsi data dengan menggunakan algoritma RSA. Berikut ini adalah hasil tanda tangan digital yang diperoleh dari message digest yang dihasilkan. Enkripsi dilakukan dengan nilai n=1537 dan nilai k=47

```
0934 1216 0193 0086 0751 1303 0671 1127 1419 0436
0848 0021 0934 0284 0022 0791 1277 0374 1383 0570
0250 0007 0006 0871 0006 0871 0340
```

V. ANALISIS DAN PERBANDINGAN

Berikut ini adalah hasil perbandingan dari tanda tangan digital yang dihasilkan dengan menggunakan implementasi dari modifikasi SHA-1 dengan tanda tangan digital yang dihasilkan dengan menggunakan SHA-1 standar.

Akan dibandingkan hasil tanda tangan digital untuk file txt yang berisi data sebagai berikut.

```
nama saya zakiy
```

Untuk hasil tanda tangan digital dari hasil implementasi modifikasi dari fungsi SHA-1 didapatkan hasil sebagai berikut.

0934 1216 0193 0086 0751 1303 0671 1127 1419 0436 0848 0021 0934 0284 0022 0791 1277 0374 1383 0570 0250 0007 0006 0871 0006 0871 0340

Untuk hasil tanda tangan digital yang menggunakan fungsi SHA-1 standar dihasilkan hasil tanda tangan sebagai berikut.

0022 0524 1383 1122 0022 1321 0140 1177 1383 0595 1312 0524 0083 0001 1460 1394 1383 1178 0671 0368 0671 0583 0340 0007 0354 1232 0871

Pada kasus di atas kedua proses sama-sama membutuhkan waktu 1 milidetik. Untuk melihat perbedaan waktu yang dibutuhkan untuk mendapatkan hasil tanda tangan digital maka akan dilakukan pengetesan lagi dengan menggunakan suatu file bmp yang berukuran 148 kilobyte.

Berikut ini adalah tanda tangan digital yang dihasilkan oleh program hasil implementasi modifikasi SHA-1.

0250 0583 1328 0485 0653 0368 1328 1013 1277 0142 1328 0368 1312 0151 0663 0410 1054 0791 0751 1157 0653 0387 0006 0871 0006 0871 0340

Lalu berikut ini adalah tanda tangan digital yang dihasilkan oleh program yang menggunakan fungsi SHA-1 standar.

0022 0870 0354 0151 1445 0621 0357 1054 0193 1075 0663 1250 1312 1502 0340 0509 0934 0102 0821 0385 1445 0699 0247 1372 0193 0728 0056

Pada kasus di atas terdapat perbedaan waktu pada proses yang dilakukan. Pada proses yang menggunakan hasil modifikasi SHA-1 diperlukan waktu 11 milidetik. Sedangkan pada proses yang menggunakan fungsi SHA-1 standar diperlukan waktu 14 milidetik. Bisa dilihat bahwa modifikasi dari fungsi SHA-1 yang dirancang dalam makalah ini lebih cepat dan ringan dibandingkan dengan fungsi SHA-1 standar.

VI. KESIMPULAN

Tanda tangan digital bisa dijadikan bukti dari otentikasi dan keaslian pesan ataupin data. Tanda tangan digital bisa dibangun dengan menggunakan banyak cara. Salah satu cara yag baik dalam membangun tanda tangan digital adalah dengan menggunakan algoritma kriptografi RSA sebagai algoritma kriptografi kunci publik dan fungsi SHA-1.

Untuk lebih meningkatkan performansi dari pembangunan tanda tangan digital bisa dilakukan modifikasi pada fungsi SHA-1. Seperti apa yang sudah dijelaskan pada makalah ini proses pembangunan tanda tangan digital dengan menggunakan implementasi dari modifikasi fungsi SHA-1 yang sudah dirancang dalam makalah ini lebih cepat dibandingkan dengan menggunakan implementasi dari fungsi SHA-1 standar.

Walaupun begitu masih ada masalah dalam modifikasi fungsi SHA-1 ini, yaitu masalah kolisi yang juga dialami oleh fungsi SHA-1 standar. Masalahnya bukan berarti ada kolisi pada modifikasi fungsi SHA-1 ini akan tetapi ada kemungkinan terjadi kolisi. Kemungkinan terjadi atau tidaknya kolisi pada modifikasi fungsi SHA-1 ini belum bisa diketahui dengan pasti. Perlu beberapa penelitian lagi untuk mengetahui hal tersebut.

Tapi dengan adanya modifikasi fungsi SHA-1 ini membuktikan bahwa proses pembuatan tanda tangan digital bisa ditingkatkan lagi performansinya maupun dari segi kecepatannya.

VII. ACKNOWLEDGEMENT

Atas tersusunnya makalah ini, sebagai pembuat, saya mengucapkan terima kasih kepada:

1. Bapak Rinaldi Munir selaku dosen mata kuliah Kriptografi dalam membimbing dan memberikan masukan kepada kami dalam perancangan dan penyusunan makalah ini.
2. Seluruh anggota Ganesha Coding Dormitory (GCD) dan teman-teman semua yang telah memberikan bantuan dan semangat dalam penyusunan makalah ini.

REFERENSI

[1] Menezes, A. 1996. *Handbook of Applied Cryptography*. CRC Press.

[2] http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf, diakses tanggal 27 April 2011

[3] http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf, diakses tanggal 27 April 2011

[4] http://eprint.iacr.org/2005/010, diakses tanggal 8 Mei 2011

[5] http://www.schneier.com/blog/archives/2005/02/sha_1broken.html, diakses tanggal 8 Mei 2011

[6] http://www.infosec.sdu.edu.cn/uploadfile/papers/Finding%20Collisions%20in%20the%20Full%20SHA-1.pdf, diakses tanggal 8 Mei 2011

[7] http://www.rsa.com/rsalabs/node.asp?id=2927, diakses tanggal 8 Mei 2011

[8] http://www.secureworks.com/research/blog/index.php/2009/6/3/sha-1-collision-attacks-now-252/, diakses tanggal 8 Mei 2011

[9] http://csrc.nist.gov/groups/ST/hash/documents/Wang_SHA1-New-Result.pdf, diakses tanggal 8 Mei 2011

[10] Munir, Rinaldi. *Slide kuliah IF3058 Kriptografi*. STEI-ITB.

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, 9 Mei 2011

Zakiy Firdaus Alfikri