Kode Autentikasi Hash pada Pesan Teks Berbasis Android

Yusfrizal
Jurusan Teknik Informatika, Universitas Potensi Utama, Medan
Jl. K.L.Yos Sudarso Km.6,5 No.3-A Medan, Telp. 061-6640525
e-mail: yusfrizal@ymail.com

Abstrak

Autentikasi merupakan hal yang sangat penting dalam pengiriman informasi baik berbentuk data ataupun pesan teks. Hal ini dikarenakan dibutuhkan autentikasi terhadap keaslian dan keutuhan data pada saat pertukaran informasi. Fungsi Hash merupakan salah satu fungsi yang memberikan layanan untuk verifikasi dan autentikasi karena fungsi ini menghasilkan nilai yang unik untuk setiap masukan. Fungsi Hash disebut juga fungsi satu arah karena sangat sulit untuk mengembalikan ke input awal dengan fungsi Hash. Fungsi Hash dapat digunakan untuk menjaga keutuhan (integritas) data dengan cara membangkitkan message digest dari isi arsip (misalnya dengan menggunakan algoritma MD5). Verifikasi isi arsip dapat dilakukan secara berkala dengan membandingkan message digest dari isi arsip sekarang dengan message digest dari arsip asli. Jika terjadi perbedaan, maka disimpulkan ada modifikasi terhadap isi arsip (atau terhadap message digest yang disimpan). Tetapi bagaimanapun, Hash juga mempunyai kelemahan karena MD5 telah dianggap tidak aman penggunaannya untuk digital signature. Namun, SHA-1 masih dianggap cukup aman. Untuk lebih aman lagi, SHA-256, SHA-384 atau SHA-512 dapat digunakan. Pada aplikasi yang dirancang ini memunculkan nilai Hash yang telah digenerasi dengan berbagai fungsi Hash, meliputi MD5, SHA-1, SHA-256, SHA-384 and SHA-512. Aplikasi ini juga dapat membandingkan persamaan dua pesan teks yang diregenarasi menggunakan SHA-256.

Kata kunci: *Autentikasi, Integritas, Hash, MD5, SHA.*

Abstract

Authentication is very important in the delivery of information such as data or text massages. Because the need of authentication in the authenticity and data integrity. The Hash function is one of the functions that provide services for verification and authentication because it generates unique values for each input. The Hash function is also called one-way function because it is very difficult to input the initial input with a Hash function. Hash function can be used to keep of data integrity by awakening digest message from archives (for example by using MD5 algorithm). This verification of the archives can be done periodically by comparing digest message from the archives with the digest message from original archives. If the difference occurs, it can be concluded that modification has been occurred to the archives or digest message. In addition, Hashes also have weaknesses because MD5 cannot be used for digital signatures. However, SHA-1 is still pretty safe. For more safety, SHA-256, SHA-384 or SHA-512 can be used. In this application will show the Hash values generated from different Hash functions, including MD5, SHA-1, SHA-256, SHA-384 and SHA-512. The application can also compare the equality of two messages by using Hash values generated by SHA-256.

Keywords*: Authentication, Integrity, Hash, MD5, SHA.*

1. Pendahuluan

Masalah keamanan merupakan salah satu aspek penting dalam proses pengiriman informasi. Informasi yang dikirimkan harus dapat dijaga kerahasiaan dan keutuhannya agar tidak dapat disalahgunakan oleh pihak-pihak yang tidak berwenang yang menginginkan informasi tersebut.

Telepon seluler merupakan alat komunikasi yang sudah dipakai oleh sebagian besar orang di dunia. Telepon seluler menyediakan media komunikasi yang beragam dan salah satu di antaranya adalah media pesan teks atau sering disebut dengan SMS (*Short Message Service*). Pesan teks (SMS) merupakan suatu layanan yang memungkinkan pengguna telepon seluler untuk mengirimkan pesan singkat kepada

pengguna telepon seluler lainnya dengan cepat dan dengan biaya yang kecil. SMS bekerja pada sistem nirkabel. Layanan pengiriman pesan singkat ini sangatlah standar dan tidak jarang para pengguna telepon seluler menggunakan layanan SMS ini untuk mengirimkan suatu pesan yang penting dan rahasia, namun para pengguna layanan SMS tersebut sering kali tidak mengetahui bahwa jalur komunikasi SMS memiliki banyak sekali celah yang memungkinkan untuk terjadinya serangan pada pesan teks yang dikirim.

Perkembangan ilmu pengetahuan dan teknologi khususnya teknologi informasi dan komunikasi membawa perubahan besar pada gaya hidup manusia. Akan tetapi kemudahan penggunaan teknologi tersebut kurang diiringi dengan kesadaran pengamanan yang memadai. Hal ini mengakibatkan ancaman terhadap keamanan data dan informasi sangat besar. Salah satu alat yang dapat digunakan untuk mengamankan data dan informasi adalah kriptografi. Menezes et al. menyatakan kriptografi adalah studi tentang teknik-teknik matematika yang berhubungan dengan aspek-aspek pengamanan informasi seperti kerahasiaan (*confidentiality*), keutuhan data (*data integrity*), autentikasi entitas (*entity authentication*), dan autentikasi asal data (*data origin authentication*). Layanan keutuhan data salah satunya dapat diperoleh dengan menggunakan fungsi Hash kriptografis (*cryptographic Hash function*). Mekanisme yang lain adalah dengan menggunakan skema tanda tangan digital (*digital signature scheme*). Layanan keutuhan data dapat mendeteksi manipulasi yang dilakukan terhadap suatu data oleh pihak yang tidak berhak. Manipulasi data mencakup penyisipan (*insertion*), penghapusan (*deletion*) dan substitusi (*substitution*) [1].

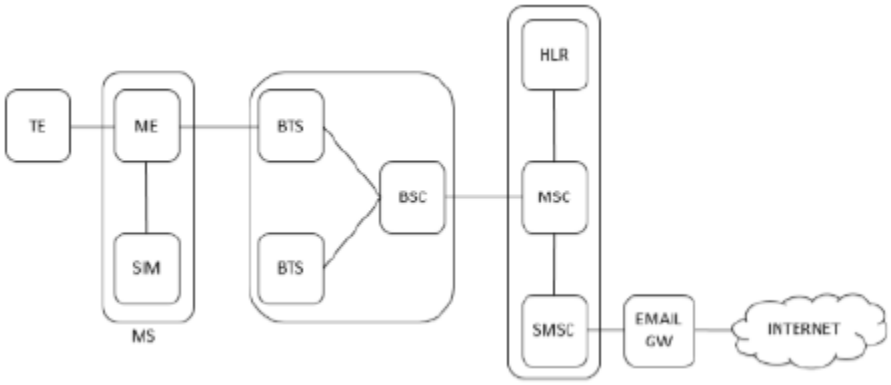
Saat ini, fungsi Hash sangat banyak digunakan untuk mengamankan data. Dengan memanfaatkan fungsi Hash ini, maka dapat dikembangkan suatu aplikasi keamanan pesan teks (SMS) yang memungkinkan pengguna untuk mengirimkan pesan singkat dengan aman. Aplikasi ini akan dibangun berbasis *mobile* pada *platform* Android.

2. Metode Penelitian

2.1. *Short Message Service* (SMS)

SMS adalah teknologi yang memungkinkan pengiriman dan penerimaan pesan antar telepon seluler. Pesan yang dikirim berupa teks yang terdiri dari kata-kata atau nomor atau kombinasi *alphanumeric*. SMS pertama kali muncul di Eropa pada tahun 1992. Setelah itu diadopsi ke teknologi nirkabel seperti CDMA dan TDMA. Salah satu kelebihan dari sarana pertukaran informasi menggunakan SMS yaitu merupakan sebuah fitur dasar pada sebuah *handphone* sehingga bisa digunakan semua *handphone* tanpa memerlukan instalasi [2].

Di bawah ini menunjukkan arsitektur dari SMS yang di integrasikan dengan jaringan GSM. Ada tiga elemen tambahan yaitu *Short Message Entity* (SME), *Short Message Service Center* (SMSC) dan email *gateway* [3].

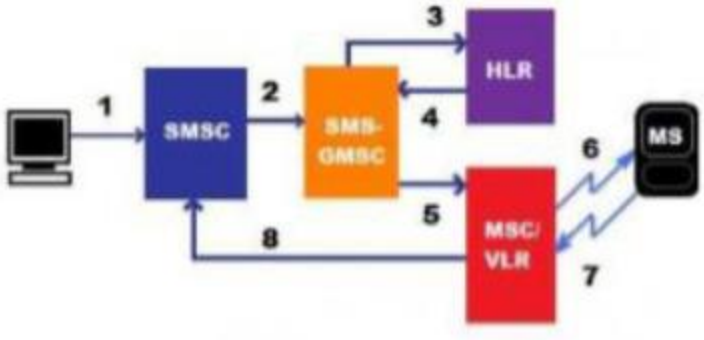


Gambar 1. Arsitektur SMS.

Keterangan gambar:

a. *Terminal Equipment* (TE) merupakan perangkat yang di gunakan seperti HP yang terhubung dengan ME.
b. *Mobile Equipment* (ME) terdiri dari pemancar radio, *display* dan DSP.
c. *Subscribe Identity Module* (SIM) berfungsi untuk mengidentifikasi *user* pada jaringan.
d. *Base Transceiver Station* (BTS) terdiri dari pemancar radio untuk berkomunikasi dengan MS.

e. *Base Station Controler* (BSC) berfungsi untuk mengatur radio *resources* untuk satu atau lebih BTS.
f. *Home Location Register* (HLR) merupakan *database* yang memiliki data pelanggan tetap.
g. *Mobile Switching Center* (MSC) melaksanakan fungsi seperti registrasi, *authentication*, *update* lokasi.
h. *SMS Center* (SMSC) mengatur proses pengiriman dan penerimaan pesan dari dan atau menuju SME sesuai dengan proses *store* and *forward*.
i. *Email gateway*, sebuah *gateway* yang menghubungkan antara *email* dan SMS pada intenet.



Gambar 2. Diagram alir SMS *mobile*.

Adapun tahapan pengiriman SMS *mobile* dapat dilihat pada diagram alir berikut:
a. ME mengirim pesan ke SMSC.
b. SMSC mengirimkan pesan ke SMS-GMSC.
c. SMS-GMSC menghubungi HLR untuk informasi *routing*.
d. HLR membalas informasi *routing* ke SMS-GMSC.
e. SMS-GMSC meneruskan pesan ke MSC/VLR.
f. MS di-*paging* dan koneksi terbentuk antara MS dan *network*, sebagaimana dalam *setup* panggilan normal. (Dengan demikian posisi MS diketahui dan apakah MS boleh berada dalam *network*/proses autentikasi).
g. Jika autentikasi berhasil, MSC/VLR mengirim pesan tersebut ke MS.
h. Jika pengiriman berhasil, *delivery report* dikirim dari MSC/VLR ke SMSC. Namun jika tidak, MSC/VLR akan menginformasikan ke HLR dan *failure report* dikirim ke SMS-C. Pada kasus pengiriman yang gagal HLR dan VLR akan mendapat informasi "*Message Waiting*" yang menunjukkan ada pesan di SMSC yang menunggu untuk dikirimkan ke MS. Informasi di HLR terdiri dari *list* SMSC pengiriman pesan, sedangkan di VLR terdapat "*flag*" yang menunjukkan apakah *list* pesan dalam keadaan kosong atau tidak. Jika MS *available* dan siap menerima pesan maka, HLR akan memberi tahu SMSC [3].

2.2. Android

Sistem Operasi Android merupakan sebuah sistem operasi yang berbasis Linux untuk telepon seluler seperti telepon pintar dan komputer tablet. Sistem operasi ini bersifat *open source* (terbuka) sehingga para *programmer* dapat membuat aplikasi secara mudah untuk digunakan oleh bermacam perangkat bergerak (misalnya telepon seluler). Karena sistem operasi Android ini merupakan aplikasi *open source* maka dapat dilakukan modifikasi dan penyebaran secara bebas [4].

Android SDK adalah *tools* API *(Application Programming Interface)* yang diperlukan untuk mulai mengembangkan aplikasi pada *platform* Android menggunakan bahasa pemrograman Java. Pemrograman Android membutuhkan bagian tambahan, yaitu *plug-in* untuk berjalan dalam Eclipse yang berupa ADT (*Android Development Toolkit*) dan Android SDK *(Software Development Kit)* [5].

Pengembang memiliki beberapa pilihan ketika membuat aplikasi yang berbasis Android. Kebanyakan pengembang menggunakan Eclipse yang tersedia secara bebas untuk merancang dan mengembangkan aplikasi Android. Eclipse adalah *IDE* yang paling popular untuk pengembangan Android, karena memiliki Android *plug-in* yang tersedia untuk memfasilitasi pengembangan Android. Selain itu Eclipse juga mendapat dukungan langsung dari Google untuk menjadi *IDE* pengembangan aplikasi Android, ini terbukti dengan adanya penambahan *plugins* untuk Eclipse untuk membuat *project*

Android dimana *source software* langsung dari situs resminya Google. Tetapi hal di atas tidak menutup kemungkinan untuk menggunakan IDE yang lain seperti Netbeans untuk melakukan pengembangan Android [6].

2.3. *Hash Message Authentication Code*

Hash Message Authentication Code pada pesan teks adalah suatu metode *authentication* untuk pesan (HMAC) atau naskah menggunakan *secure Hashing* dan kunci rahasia. Jika *k* adalah kunci rahasia dan *m* adalah pesan atau naskah, maka rumus yang digunakan adalah [7]:

$$HMAC(k,m) = H((k \oplus p_0) \pm H(k \oplus p_i) \pm m)) \quad (1)$$

Dimana *H* adalah fungsi *secure Hashing* (contohnya MD5 atau SHA-1), ± adalah operasi penyambungan (*concatenation*), p_0 dan p_i masing-masing merupakan *padding* sebesar blok yang digunakan *H*. *Padding* p_0 berisi *byte* 0x5c (*hexadecimal*) yang diulang untuk memenuhi blok, dan *padding* p_i berisi *byte* 0x36 (*hexadecimal*) yang diulang untuk memenuhi blok. Jika kunci *k* lebih kecil dari blok, maka *k* di-*padding* dengan *0* sampai memenuhi blok. Jika *k* lebih besar dari blok, maka *k* dipotong belakangnya hingga besarnya sama dengan blok. Metode HMAC digunakan karena metode yang menggunakan rumus berikut [7]:

$$MAC(k,m) = H(k \pm m) \quad (2)$$

Mempunyai kelemahan yaitu kelemahan terhadap *length extension attack*. Tergantung dari fungsi *H*, seseorang dapat menambahkan sesuatu (misalnya m_a) ke *m*:

$$m_0 = m \pm ma \quad (3)$$

Dan tanpa mengetahui *k* dapat membuat:

$$MAC(k;m_0) \quad (4)$$

Metode HMAC menggunakan MD5 dinamakan HMAC-MD5. Demikian juga, metode HMAC menggunakan SHA-1 dinamakan HMAC-SHA-1. Dua contoh algoritma *secure Hashing* adalah MD5 dan SHA-1. MD5 telah dianggap tidak aman penggunaannya untuk *digital signature*. SHA-1, meskipun memiliki kelemahan, masih dianggap cukup aman. Untuk lebih aman lagi, SHA-256, SHA-384 atau SHA-512 dapat digunakan.

3. Hasil dan Pembahasan

Fungsi Hash yang akan diimplementasikan pada aplikasi ini yaitu MD5, SHA-1, SHA-256, SHA-384 dan SHA-512. Aplikasi ini juga membandingkan persamaan antara dua pesan dengan menggunakan fungsi *Hash* yang digenerasi menggunakan SHA-256.

3.1. MD5

Secara garis besar, algoritma fungsi *Hash* MD5 terdiri dari langkah-langkah sebagai berikut [6]:

1. *Pre-processing* pesan dengan penambahan *bit* pengganjal (*padding bits*) dan nilai panjang pesan.
2. Inisialisasi nilai *digest* (disebut penyangga/*buffer*)
3. Pemrosesan tiap blok pesan 512 *bit* untuk mentransformasikan nilai *digest*.

Algoritma *secure Hashing* MD5 dirancang oleh Ron Rivest dan penggunaannya sangat populer di kalangan komunitas *open source* sebagai *checksum* untuk *file* yang dapat di-*download*. MD5 juga kerap digunakan untuk menyimpan *password* dan juga digunakan dalam *digital signature* dan *certificate*. Besarnya blok untuk MD5 adalah 512 *bit* sedangkan *digest size* adalah 128 *bit*. Karena *word size* ditentukan sebesar 32 *bit*, satu blok terdiri dari 16 *word* sedangkan *digest* terdiri dari 4 *word*. Indeks untuk bit dimulai dari 0. *Preprocessing* dimulai dengan *padding* sebagai berikut:

1. *Bit* dengan nilai 1 ditambahkan setelah akhir naskah.

2. Deretan *bit* dengan nilai 0 ditambahkan setelah itu sehingga besar dari naskah mencapai nilai 448 (*mod* 512) (sedikitnya 0 dan sebanyaknya 511 *bit* dengan nilai 0 ditambahkan sehingga tersisa 64 *bit* untuk diisi agar besar naskah menjadi kelipatan 512).
3. 64 *bit* yang tersisa diisi dengan besar naskah asli dalam *bit*. Jika besar naskah asli lebih dari 264 *bit* maka hanya 64 *lower order bit* yang dimasukkan. *Lower order word* untuk besar naskah asli dimasukkan sebelum *high order word*.

Setelah *padding*, naskah terdiri dari *n word M[0 ... n - 1]* dimana *n* adalah kelipatan 16. Langkah berikutnya dalam *preprocessing* adalah menyiapkan *MD buffer* sebesar 4 *word*:

(A,B,C,D).

Dimana *A* merupakan *lower order word*. *Buffer* diberi nilai awal sebagai berikut (nilai dalam *hexadecimal* dimulai dengan *lower order byte*).

A	:	01	23	45	67
B	:	89	*ab*	*cd*	*ef*
C	:	*fe*	*dc*	*ba*	98
D	:	76	54	32	10

Proses *Hashing* dilakukan per blok, dengan setiap blok melalui 4 putaran. Proses *Hashing* menggunakan 4 fungsi *F*, *G*, *H*, dan *I* yang masing-masing mempunyai *input* 3 *word* dan *output* 1 *word*:

F(X, Y, Z)	=	$(X \wedge Y) \vee (\neg X \wedge Z)$
G(X, Y, Z)	=	$(X \wedge Z) \vee (Y \wedge \neg Z)$
H(X, Y, Z)	=	$X \oplus Y \oplus Z$
I(X, Y,Z)	=	$Y \oplus (X \vee \neg Z)$

Dimana $\wedge$ adalah *bitwise and*, $\vee$ adalah *bitwise or*, $\oplus$ adalah *bitwise exclusive or*, dan $\neg$ adalah *bitwise not* (*one's complement*). Selain keempat fungsi di atas, proses *Hashing* juga menggunakan tabel dengan 64 *word*, T[1] sampai dengan T[64].

Secara garis besar, algoritma untuk *Hashing* untuk satu blok adalah sebagai berikut (menggunakan *array* 16 *word* X[0] sampai dengan X[15] yang dapat menyimpan satu blok:

1. *Copy* satu blok (*word* M[16*i*] sampai dengan M[16*i*+15]) ke X[0] sampai dengan X[15].
2. Simpan A,B,C,D dalam A′,B′,C′,D′.
3. Lakukan putaran 1 pada A,B,C,D.
4. Lakukan putaran 2 pada A,B,C,D.
5. Lakukan putaran 3 pada A,B,C,D.
6. Lakukan putaran 4 pada A,B,C,D.
7. Tambahkan nilai simpanan pada A,B,C,D:
   A → $(A + A') \bmod 2^{32}$
   B → $(B + B') \bmod 2^{32}$
   C → $(C + C') \bmod 2^{32}$
   D → $(D + D') \bmod 2^{32}$

Algoritma di atas diulang hingga semua blok dalam *M* terproses (dimulai dengan $i = 0$ sampai dengan $i = n/16 - 1$). Setelah semua blok diproses, maka hasil akhir A,B,C,D menjadi MD5 *digest* dari naskah asli. Urutan *byte* untuk *digest* dimulai dengan *lower order byte* dari *A* dan diakhiri oleh *higher order byte* dari *D*.

3.2. SHA

Algoritma *secure Hashing* SHA dirancang oleh *National Security Agency* (NSA) dan dijadikan standard FIPS. Ada 4 varian SHA dalam standar FIPS-180-2 dengan parameter yang berbeda. Keamanan algoritma didasarkan pada fakta bahwa *birthday attack* pada *digest* sebesar *n bit* menghasilkan *collision* dengan faktor kerja sekitar $2^{n/2}$. Kita hanya akan membahas SHA-1 di sini karena SHA-256, SHA-384 dan SHA-512 algoritmanya mirip dengan SHA-1.

Tabel 1. Empat varian SHA.

Algoritma	Naskah (bit)	Blok (bit)	Word (bit)	Keamanan (bit)
SHA-1	$<2^{64}$	512	160	80
SHA-256	$<2^{64}$	512	256	128
SHA-384	$<2^{128}$	1024	384	192
SHA-512	$<2^{128}$	1024	512	256

SHA-1 menggunakan fungsi f_t, dimana $0 \le t < 79$, dengan *input* 3 *word* masing-masing sebesar 32 *bit* dan menghasilkan *output* 1 *word*:

$$ft = \begin{cases} Ch(x,y,z) = (x \wedge y) \oplus (\neg x \wedge z) & 0 \le t \le 19 \\ Parity(x,y,z) = x \oplus y \oplus z & 20 \le t \le 39 \\ Maj(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) & 40 \le t \le 59 \\ Parity(x,y,z) = x \oplus y \oplus z & 60 \le t \le 79 \end{cases}$$

SHA-1 menggunakan konstan k_t sebagai berikut (dengan nilai dalam *hexadecimal*):

$$kt = \begin{cases} 5a827999 & 0 \le t \le 19 \\ 6ed9eba1 & 20 \le t \le 39 \\ 8f1bbcdc & 40 \le t \le 59 \\ ca62c1d6 & 60 \le t \le 79 \end{cases}$$

Selain enam fungsi logika yang digunakan, SHA-256 juga menggunakan nilai konstanta K [0..63]. Berikut nilai konstanta K[0..63] [8]:

Tabel 2. Nilai K[0..63]

428a2f98	71374491	b5c0fbcf	e9b5dba5
3956c25b	59f111f1	923f82a4	ab1c5ed5
d807aa98	12835b01	243185be	550c7dc3
72be5d74	80deb1fe	9bdc06a7	c19bf174
e49b69c1	efbe4786	0fc19dc6	240ca1cc
2de92c6f	4a7484aa	5cb0a9dc	76f988da
983e5152	a831c66d	b00327c8	bf597fc7
c6e00bf3	d5a79147	06ca6351	14292967
27b70a85	2e1b2138	4d2c6dfc	53380d13
650a7354	766a0abb	81c2c92e	92722c85
a2bfe8a1	a81a664b	c24b8b70	c76c51a3
d192e819	d6990624	f40e3585	106aa070
19a4c116	1e376c08	2748774c	34b0bcb5
391c0cb3	4ed8aa4a	5b9cca4f	682e6ff3
748f82ee	78a5636f	84c87814	8cc70208
90befffa	a4506ceb	bef9a3f7	c67178f2

Seperti halnya dengan MD5, SHA-1 terdiri dari dua tahap yaitu *preprocessing* dan *Hashing*. *Preprocessing* dimulai dengan *padding* yang prosesnya persis sama dengan MD5, yaitu setelah akhir naskah, 1 *bit* dengan nilai 1 ditambahkan, disusul oleh *bit* dengan nilai 0 sebanyak 0 sampai dengan 511 tergantung panjang naskah, dan diakhiri dengan 64 *bit* yang merepresentasikan besar naskah asli. Setelah *padding*, naskah terdiri dari *n word* M[0 … n - 1] dimana *n* adalah kelipatan 16. Langkah berikutnya dalam *preprocessing* adalah menyiapkan SHA-1 *buffer* sebesar 5 *word*:

$$(H_0^{(0)}, H_1^{(0)}, H_2^{(0)}, H_3^{(0)}, H_4^{(0)})$$

Buffer diberi nilai awal sebagai berikut (nilai dalam *hexadecimal*):

$$H_0^{(0)} \rightarrow 67452301$$

$H_1^{(0)} \rightarrow efcdab89$
$H_2^{(0)} \rightarrow 98badcfe$
$H_3^{(0)} \rightarrow 10325476$
$H_4^{(0)} \rightarrow c3d2e1f0$

Setelah *buffer* diberi nilai awal, tahap kedua yaitu *Hashing* dilakukan terhadap setiap blok ($M^{(1)}, M^{(2)}, \ldots, M^{(n)}$) sebagai berikut (i = 1, 2, … , n):

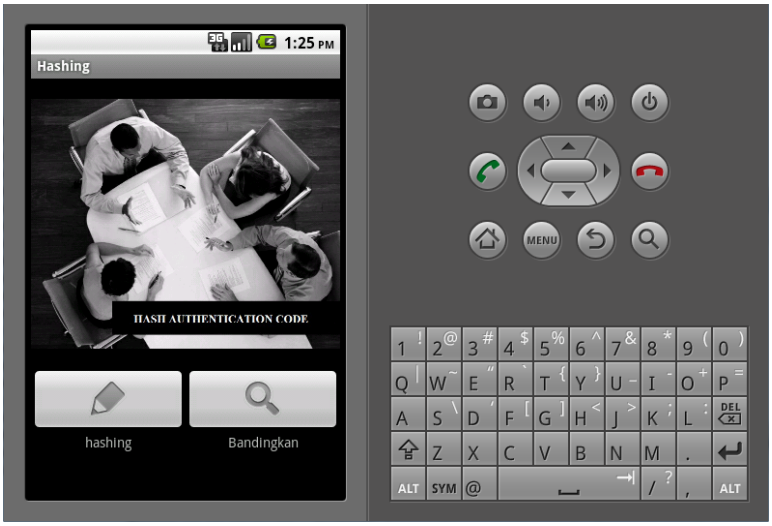
1. Siapkan *message schedule* {Wt}:
$$Wt \leftarrow \begin{cases} Mt^{(i)} & 0 \le t \le 15 \\ (Wt-3 \oplus Wt-8 \oplus Wt-14 \oplus Wt-16 <<< 1 & 16 \le t \le 79 \end{cases}$$
2. Berikan nilai awal untuk variabel *a*, *b*, *c*, *d*, dan *e*:
   a → $H_1^{(i-1)}$
   b → $H_2^{(i-1)}$
   c → $H_3^{(i-1)}$
   d → $H_4^{(i-1)}$
   e → $H_5^{(i-1)}$
3. Untuk t = 0, 1, 2, … , 79:
   T → (a <<< 5) + f_t(b, c, d) + e + Kt +Wt (mod 2^{32})
   e → d
   d → c
   c → b <<< 30
   b → a
   a → T
4. Lakukan kalkulasi *Hash value* tahap *i*:
   $H_0^i \rightarrow a + H_0^i$
   $H_1^i \rightarrow b + H_1^i$
   $H_2^i \rightarrow c + H_2^i$
   $H_3^i \rightarrow d + H_3^i$
   $H_4^i \rightarrow e + H_4^i$
5. Setelah *Hashing* dilakukan pada semua blok ($M^{(1)}, M^{(2)}, \ldots, M^{(n)}$), kita dapatkan *digest* sebagai berikut:
   ($H_0^{(n)}; H_1^{(n)}; H_2^{(n)}; H_3^{(n)}; H_4^{(n)}$)

Beberapa ahli kriptografi telah mencoba mencari kelemahan SHA-1. Xiaoyun Wang dan koleganya berhasil memperkecil ruang pencarian untuk *collision* SHA-1 dari 280 operasi SHA-1 (yang merupakan “kekuatan" teoretis SHA-1 jika SHA-1 tidak memiliki kelemahan, berdasarkan *birthday attack*) menjadi 2^{69} operasi. Walaupun demikian, penggunaan SHA-1 masih dianggap cukup aman, dan jika ingin lebih aman lagi, maka SHA-256, SHA-384 atau SHA-512 dapat digunakan.

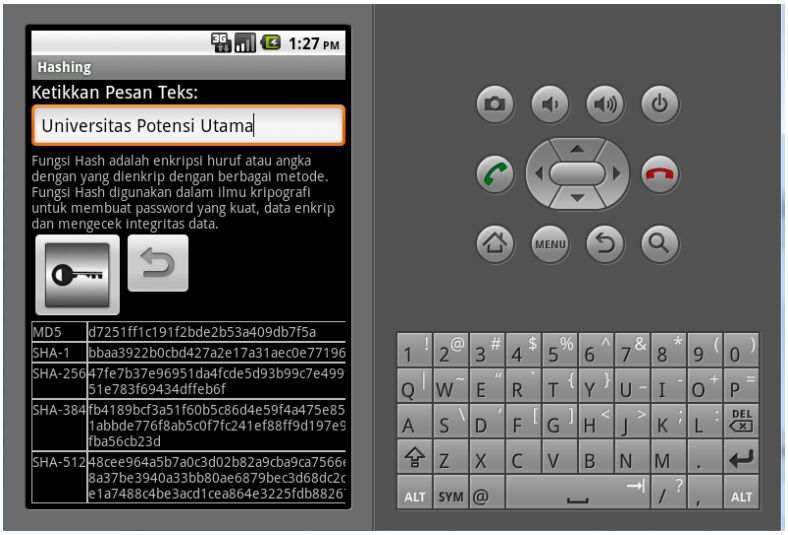
3.3. Pengujian

Pada aplikasi ini, pengguna dapat mengetikkan pesan teks pada kotak pesan, kemudian menekan tombol fungsi. Maka aplikasi akan memunculkan nilai Hash yang telah digenerasi dengan berbagai fungsi Hash, meliputi MD5, SHA-1, SHA-256, SHA-384 dan SHA-512. Aplikasi ini juga dapat membandingkan persamaan dua pesan teks yang diregenerasi menggunakan SHA-256.

Berikut ini adalah beberapa tampilan pada aplikasi ini.



Gambar 3. Halaman utama.



Gambar 4. Menu *hashing*.

Gambar 5. Menu bandingkan.

4. Kesimpulan

Dari hasil pengujian sistem yang telah dilakukan, maka dapat diambil beberapa kesimpulan diantaranya sebagai berikut:

a. Algoritma MD5 menerima masukan berupa pesan dengan ukuran sembarang dan menghasilkan *message digest* yang panjangnya 128 bit.
b. Fungsi *Hash* dapat digunakan untuk menjaga keutuhan (integritas) data. Caranya, bangkitkan *message digest* dari isi arsip (misalnya dengan menggunakan algoritma MD5). *Message digest* dapat digabung ke dalam arsip atau disimpan di dalam arsip. Verifikasi isi arsip dapat dilakukan secara berkala dengan membandingkan *message digest* dari isi arsip sekarang dengan *message digest* dari arsip asli. Jika terjadi perbedaan, maka disimpulkan ada modifikasi terhadap isi arsip (atau terhadap *message digest* yang disimpan).
c. MD5 telah dianggap tidak aman penggunaannya untuk *digital signature*. SHA-1, meskipun memiliki kelemahan, masih dianggap cukup aman. Untuk lebih aman lagi, SHA-256, SHA-384 atau SHA-512 dapat digunakan.
d. Aplikasi yang dirancang dalam penelitian ini telah mampu melakukan proses verifikasi keutuhan *file* dengan menggunakan algoritma MD5.
e. Algoritma MD5 dapat digunakan untuk memverifikasi keutuhan dari suatu *file* dengan cara menghitung nilai MD5 dari *file* tersebut dan membandingkannya dengan nilai MD5 yang diperoleh dari si pengirim.

5. Saran

Beberapa hal yang perlu dikembangkan dari penelitian ini adalah :

a. Diharapkan pengembangan aplikasi yang langsung diterapkan pada pengiriman SMS.
b. Diharapkan pengembangan aplikasi yang dikombinasikan dengan algoritma-algoritma keamanan yang lain.

Daftar Pustaka

[1] Susila Windarta, 2013, Fungsi Hash Berbasis Teori Graf: Sebuah Survei, Seminar Nasional Sistem Informasi Indonesia, Skripsi, Program Studi Ilmu Komputer, Institut Teknologi Sepuluh Nopember (ITS), Surabaya.
[2] Imam Prayogo Pujiono, 2015, Implementasi Algoritma AES dan Modifikasi Vigenere untuk Pengamanan Pesan SMS Dengan Nomor Pengirim dan Penerima Sebagai Kunci Tambahan, Jurnal Algoritma, Vol. 12 No. 1, 2302-7339.
[3] Ari Dwi Ananto, 2014, Desain dan Implementasi Aplikasi SMS (Short Message Service) pada Android Menggunakan Algoritma AES, Universitas Telkom, Bandung.
[4] I Komang Setia Buana, 2015, Aplikasi Komik Untuk Toilet Training Berbasis Android, CSRID, Jurnal Sistem dan Informatika Vol. 10, No. 1, STMIK STIKOM, Bali.
[5] Dedy Hermanto, 2015, Pengontrolan Gerak Mobile Robot Menggunakan Sensor Accelerometer pada Perangkat Bergerak Android, CSRID Journal, Vol.7 No.1, AMIK Multi Data, Palembang.
[6] elib.unikom.ac.id/download.php?id=256870.
[7] Sentot Kromodimoeljo, 2009, Teori dan Aplikasi Kriptografi, SPK IT Consulting, Bandung.
[8] Steven Andrew, 2012, Pembangkit Bilangan Acak Berbasis Fungsi Hash, Makalah IF3058 Kriptografi – Sem. II, Institut Teknologi Bandung, Bandung.