Perbandingan Algoritma Fungsi Hash MD5 dengan SHA-1

Hanifah Azhar 13509016[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13509016@std.stei.itb.ac.id

Abstract*—Fungsi hash sudah dimanfaatkan dalam berbagai bidang, dari website untuk belajar masak sampai kebutuhan militer. Fungsi hash mengenkripsi pesan menjadi sebuah string dengan panjang tetap. Namun, setelah pesan dienkripsi tidak dapat didekripsi kembali untuk melihat isi pesan asli, karena hal tersebut fungsi hash sangat bermanfaat untuk penyimpanan kata sandi. Dari berbagai algoritma fungsi hash, MD5 dan SHA-1 termasuk dua algoritma yang sangat marak penggunaannya. Banyak perdebatan terjadi mengenai algoritma mana yang lebih aman karena telah ditemukan celah untuk memecahkan MD5 maupun SHA-1. Makalah ini menganalisis MD5 dan SHA-1 untuk mengetahui algoritma yang lebih unggul dari kedua algoritma fungsi hash tersebut.

***Kata kunci*—Fungsi Hash, MD5, SHA-1, pesan, digest**

I. Latar Belakang

Kriptografi sudah sangat lazim digunakan dalam kehidupan sehari-hari manusia. Hampir setiap kali internet diakses, kriptografi dimanfaatkan tanpa sepengetahuan pengguna pada umumnya. Contohnya untuk mengirim data, aplikasi-aplikasi tertentu akan mengenkripsi data yang dikirim terlebih dahulu dan ketika sampai ke penerima data didekripsi kembali. Bahkan, untuk *login* ke email membutuhkan fungsi hash untuk memeriksa kata sandi. Tentunya hal-hal tersebut sangat vital untuk digunakan karena alasan keamanan dan *privacy*.

Fungsi hash dalam kriptografi adalah fungsi hash yang berupa sebuah algoritma yang mengambil sejumlah block data dan mengembalikan *bit string* berukuran tetap. String yang dihasilkan tersebut merupakan *hash value*. Perubahan yang dilakukan pada data walaupun sangat kecil, sengaja ataupun tidak, akan menyebabkan perubahan yang sangat banyak pada hasil *hash value.* Bahkan *hash value* dapat menjadi berbeda sama sekali. Data yang di hash sering disebut pesan.

Umumnya dalam kriptografi, ketika mengenkripsi suatu pesan harus dapat di dekripsi dengan sempurna. Sedangkan, untuk fungsi hash justru tidak boleh didekripsi. Fungsi hash yang ideal adalah fungsi yang hasil *hasil hash value* tidak dapat didekripsi kembali. Hal ini dimanfaatkan untuk kerahasiaan. Ada berbagai jenis algoritma untuk melakukan *hash.* Namun, karena semakin banyak ahli kriptanalisis, banyak dari algoritma yang dulu dianggap *robust* ternyata telah ditemukan celahnya.

Algoritma yang masih dianggap *robust* hingga kini adalah MD5 dan SHA-1. MD5 adalah *Message-Digest Algorithm.* MD5 sudah sangat lazim digunakan sebagai fungsi hash kriptografi. MD5 sering dimanfaatkan untuk memastikan integritas data. Hasil yang didapat dari penggunaan MD5 adalah angka heksadesimal dengan panjang 32 character. Walaupun MD5 sudah digunakan secara umum untuk banyak aplikasi keamanan, sudah ada yang menemukan kelemahan dari MD5. Sehingga, para kriptografer sudah mulai mengusulkan penggunaan algoritma SHA-1 dibandingkan dengan MD5 (Dobbertin, 1996).

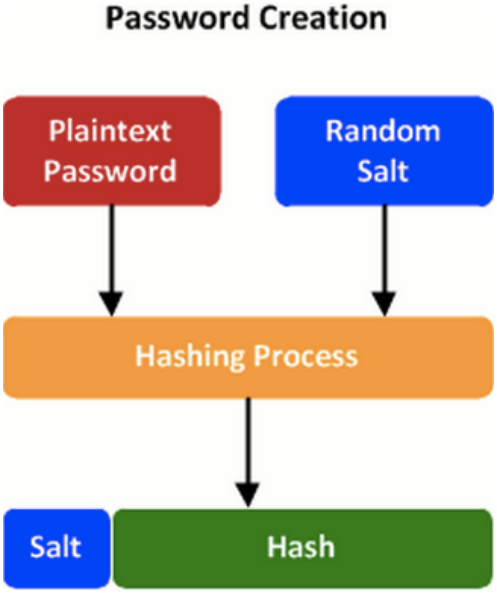
Dalam makalah ini dilakukan perbandingan terhadap algoritma fungsi hash MD5 dan algoritma fungsi hash SHA-1. Perbandingan ini dilakukan dengan tujuan menganalisis kedua algoritma dan menentukan algoritma fungsi hash yang lebih unggul.

II. Dasar Teori

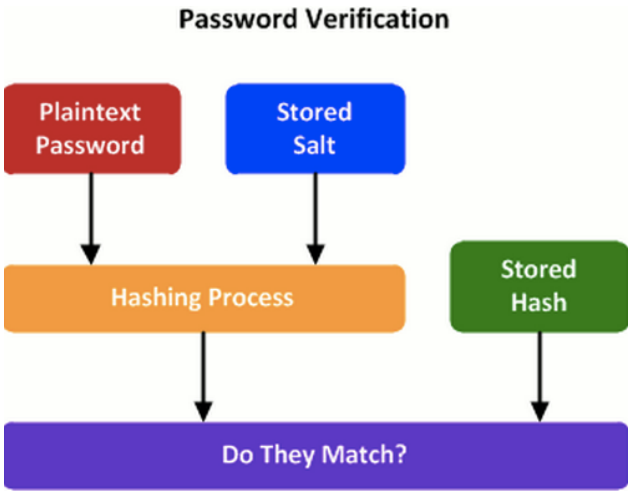
Fungsi hash dalam kriptografi adalah fungsi hash yang berupa sebuah algoritma yang mengambil sejumlah blok data dan mengembalikan *bit string* berukuran tetap. String yang dihasilkan tersebut merupakan *hash value*. Perubahan yang dilakukan pada data walaupun sangat kecil, sengaja ataupun tidak, akan menyebabkan perubahan yang sangat banyak pada hasil *hash value.* Bahkan *hash value* dapat menjadi berbeda sama sekali. Data yang di hash sering disebut pesan, *hash value* disebut *digest*.

Umumnya dalam kriptografi, ketika mengenkripsi suatu pesan harus dapat didekripsi dengan sempurna. Sedangkan, untuk fungsi hash justru tidak boleh didekripsi. Fungsi hash yang ideal adalah fungsi yang hasil *hash value* tidak dapat didekripsi kembali. Hal ini dimanfaatkan untuk kerahasiaan. Sehingga sering disebut juga *one-way encryption*.

Fungsi hash dapat digunakan untuk berbagai jenis aplikasi atau protokol. Seperti untuk fungsi cache, fungsi hash digunakan untuk set data yang besar yang disimpan pada media penyimpanan yang lambat. Namun, yang dibahas dalam makalah ini adalah kemanan fungsi hash untuk digunakan pada kata sandi. Fungsi has pada kata sandi sering memanfaatkan fungsi salt.



Gambar 1: Pembuatan *Password*
(sumber:
http://media.packetlife.net/media/blog/attachments/559/password_hashing.png)



Gambar 2: Verifikasi *Password*
(sumber:
http://media.packetlife.net/media/blog/attachments/559/password_hashing.png)

Kenyataannya adalah bahwa fungsi hash algoritma secanggih apapun yang akan ditemukan nanti, pasti selalu akan dapat dipecahkan suatu saat. Itulah dasar pemikiran yang memicu diciptakannya fungsi salt.

Fungsi salt dimanfaatkan untuk membuat *hacker* lebih kesulitan untuk memecahkan kata sandi seseorang. Fungsi salt memanfaatkan lebih dari sekedar *password* untuk dibuatkan *digest.* Umumnya yang digunakan adalah email ditambah dengan password untuk menciptakan semacam sandi rahasia yang cukup panjang agar lebih kompleks untuk dipecahkan. Namun, tidak harus hanya menggunakan email, semakin panjang sandi yang dimasukkan ke fungsi hash, semakin kompleks pula pemecahannya.

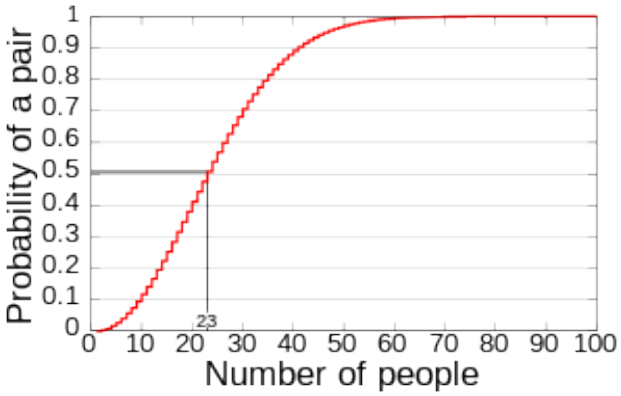
Walaupun terdapat banyak jenis algoritma fungsi hash, makalah ini hanya akan membahas mengenai algoritma MD5 dan SHA-1. Berikut ini adalah penjelasan mengenai kedua algoritma fungsi hash:

a. Algoritma MD5

MD5 adalah salah satu dari rangkaian algoritma yang diciptakan oleh seorang Professor di MIT bernama Ronald Rivest (Rivest, 1992). Beliau menciptakan MD5 karena algoritma sebelumnya telah dianggap tidak aman, yaitu MD4.

Den Boer dan Bosselaers menemunakan bentrokan pada MD5. Yaitu dua pesan yang berbeda yang menhasilkan *digest* yang identik pada tahun 1993. Pada tahun 1996, Dobbertin mengumumkan fungsi *compression* bentrokan untuk MD5. Hal ini tidak menyebabkan MD5 jadi dikatakan tidak aman. Namun, cukup untuk membuat para pakar kriptografi untuk mengusulkan pengganti, yaitu SHA-1 [2]. Ukuran hash (128 bit) cukup kecil untuk dilakukan *bitrhday attack.*

Birthday attack adalah serangan yang ditujukan pada MD5 untuk mendapatkan bentrokan sebuah *digest*. Serangan ini berasal dari teori matematik yang ditemukan bahwa dari kumpulan orang-orang yang dipilih secara acak, ada kemungkinan bahwa sepasang dari orang-orang tersebut memiliki tanggal ulang tahun yang sama. Teori tersebut memiliki probalbilitas 100% jika ada sejumlah 367 orang, 99% dengan 57 orang, dan 50% dengan 23 orang. Teori inilah dasar dari *birthday attack* yang akhirnya digunakan untuk memecahkan MD5.



Gambar 3: *Birthday Attack*

(sumber:
http://crppit.epfl.ch/documentation/Hash_Function/WiKi/Birthday_problem_fichiers/)

Proses iterasi untuk fungsi Hash:

Umumnya fungsi hash diiterasi dengan melakukan *compression* pada fungsi **X = f(Z).** Fungsi tersebut melakukan *compression* pada **l**-bit pesan pada blok Z ek **s**-bit *hash value* **X** untuk **l>s**.

Pada MD5, l adalah 512 dan s adalah 128. Metode iterasi yang digunakan disebut *MerkleDamgard meta-method* [2].

Berikut ini menjelaskan fungsi *compression* untuk MD5:

1. Bagi M_i menjadi blok-blok 32-bit untuk setiap blok 512-bit. $M_i = (m_0, m_1, ..., m_{15})$.
2. Algoritma *compression* algorithm untuk M_i terdiri dari 4 tahap. Setiap tahapnya terdiri dari 16 fungsi. 4 tahap berturut-turut adalah sebagai berikut:

$$a = b + ((a+\varphi_i(b, c, d) + w_i + t_i \ll s_i)),$$

$$d = a + ((d+\varphi_{i+1}(a, b, c) + w_{i+1} + t_{i+1}) \ll s_{i+1}),$$

$$c = d + ((c+\varphi_{i+2}(d, a, b) + w_{i+2} + t_{i+2}) \ll s_{i+2}),$$

$$b = c + ((b+\varphi_{i+3}(c, d, a) + w_i + _3 + t_{i+3}) \ll s_{i+3}),$$

operasi (+) memliki arti ADD modulo 2^{32} . t_{i+j}
dan s_{i+j} (j = 0, 1, 2, 3) adalah konstanta *yang step-dependent*. w*i+j* adlah pesan.$\ll s_{i+j}$ adalah *circularly leftshift* dari posisi bit s_{i+i}.

b. Algoritma SHA-1

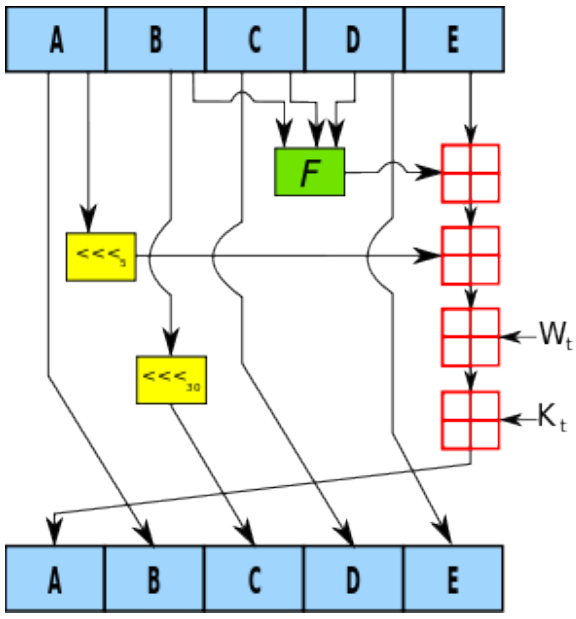
SHA-1 adalah salah satu dari rangkaian algoritma yang diciptakan *United States National Security Agency*. SHA adalah singkatan dari *Secure Hash Algorithm*. Dari semua jenis SHA yang ada, SHA-1 adalah yang paling umum digunakan. SHA-1 sudah digunakan dalam berbagai macam aplikasi dan protokol [8]. Pada tahun 2005, kriptanalis menemukan beberapa serangan yang mengatakan bahwa SHA-1 kemungkinan tidak cukup aman untuk terus digunakan.

SHA terdiri dari :

- SHA-0
- SHA-1
- SHA-2
- SHA-3

Dan juga SHA-256, SHA-384, dan SHA-512.

SHA-1 menghasilkan *digest* sebesar 160-bit. Asal-usul SHA-1 adalah dari prinsip-prinsip yang mirip dengan yang digunakan oleh Ronald L. Rivest untuk algoritma MD4 dan MD5.



Gambar 4. Fungsi kompresi pada SHA-1

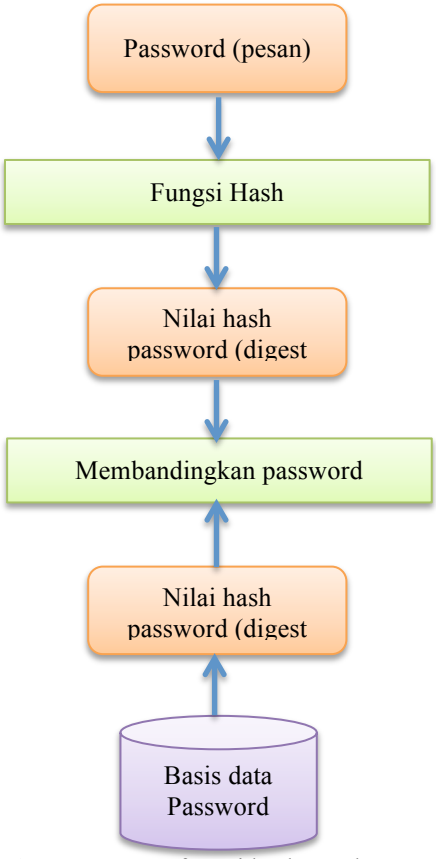
(sumber: http://crppit.epfl.ch/documentation/Hash_Function/Documentation/)

A, B, C, D, E adalah pesan 32-bit. *F* adalah fungsi nonlinear yang berbeda-beda. $<<<_n$ menandakan perubahan bit sesuai *n*. W_t adalah *expanded message word of round t*. K_t adalah *round constant of t.*

III. FUNGSI HASH

Bentrokan pada fungsi hash sangat tidak diinginkan. Fungsi hash yang memiliki banyak bentrokan daat dikatakan adalah fungsi hash yang lemah. Bentrokan tidak dapat dihindarkan ketika anggota set terlalu besar, menurut teori *the pigeonhole principle* [7]. Dampak bentrokan bergantung pada aplikasi. Fungsi hash dirancangan sedemikian rupa agar memaksimalkan probabilitas bentrokan untuk data yang berbeda namun mirip.

Gambaran umum proses yang dilakukan pada fungsi hash terdapat pada Bagan 1.



Bagan 1: Penggunaan fungsi hash untuk *Login e-mail*

IV. ANALISIS

Implementasi dalam makalah ini menggunakan lingkungan pengembangan sebagai berikut:

Operating System	Mac OS X Lion
Bahasa Pemrogaman	Java
RAM	2 GB

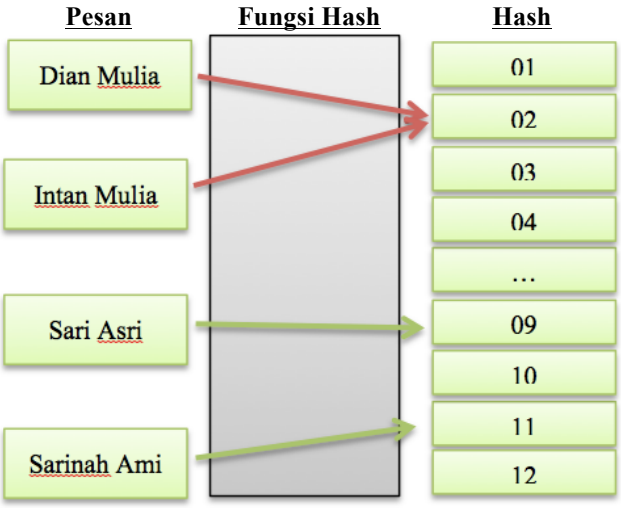
Tabel 1. Lingkungan pengembangan

Implementasi yang dilakukan adalah untuk mengetahui bagaimana perubahan *hash value* jika pesan berubah. Berikut adalah beberapa contoh yang dicoba:

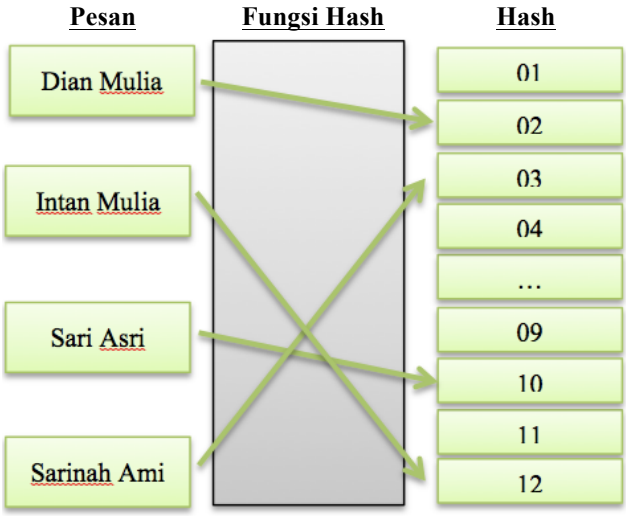
MD5 Hash Values
hanifah : c0d467ce4c04bf343c15254c1dcb67b1
Hanifah : 46c04e1be7e2dba58b37df2e99c1f7e2
hanifih : 2a5dd0715868a26e88685897943cba3c

SHA-1 Hash Values
hanifah : 19fc27f4c222b6f6738b3691b644d9c025afe90f
Hanifah : 174d3f141c23b8338f448bd2fd442f89a4196c5c
hanifih : 74811cc5861ffd117d085fcf7f24b2a1a50c3a31

Pada dua contoh diatas, algoritma MD5 dan SHA-1 menunjukkan perubahan yang signifikan ketika salah satu huruf pada pesan diubah menjadi huruf besar dan juga ketika salah satu huruf dalam pesan diubah menjadi huruf lain. Sehingga untuk kasus sederhana kedua jenis algoritma terlihat tidak mungkin terjadi bentrokan.



Contoh 1 : Fungsi hash yang buruk



Contoh 2 : Fungsi hash yang baik

Dari hasil analisis terhadap fungsi hash. Dibuat 2 contoh di atas, dapat dilihat bahwa fungsi hash yang baik adalah seperti Contoh 2. Pada Contoh 2, perbedaan sedikit tetap menyebabkan perubahan yang signifikan dan tidak ada pesan yang dapat menyebabkan nilai hash yang identik. Sedangkan, fungsi hash yang buruk adalah ketika pesan berubah sedikit, nilai hash pun berubah hanya sedikit. Hal ini dapat membuat isi pesan asli menjadi mudah ditebak. Selain itu, jika ditemukan ada pesan berbeda yang dapat menghasilkan nilai hash yang sama, algoritma yang digunakan perlu diwaspadai pengunaannya.

Ukuran *digest* yang dihasilkan MD5 lebih pendek ketimbang *digest* yang dihasilkan SHA-1. *Digest* MD5 juga cukup pendek untuk dipecahkan oleh serangan *birthday attack* dalam waktu realtif singkat [2].

Berberapa serangan bentrokan terhadap SHA-1 sudah ada. Namun, tidak praktis karena kekuatan CPU yang dibutuhkan diperkirakan memakan biaya kurang lebih $

3 juta untuk setiap hashnya [6]. Penyerangan dikatakan tidak praktis karena langkah yang dibutuhkan untuk menyerang SHA-1 terlalu panjgan untuk setiap hashnya. Berbeda dengan MD5 yang langkah untuk penyerangannya relatif lebih sedikit.

V. KESIMPULAN & USULAN

MD5 sudah terlalu banyak bentrokan yang ditemukan. Selain itu, sudah ditemukan berbagai metode penyerangan yang perlu dilakukan untuk memecahkan kode pada MD5 dan pelaksanaanya dapat dilakukan dengan biaya yang rendah. Sehingga, MD5 dapat dikatakan sudah tidak aman untuk digunakan untuk aplikasi-aplikasi yang membutuhkan keamanan.

Keunggulan MD5 jika dibandingkan dengan SHA-1 adalah bahwa panjang string *digest* yang dihasilkan lebih pendek. Sehingga, untuk aplikasi-aplikasi yang tidak dimanfaatkan untuk menyembunyikan data, seperti pembuatan cache, MD5 lebih baik untuk digunakan. String yang lebih pendek mempercepat akses dan juga membutuhkan tempat penyimpanan yang lebih sedikit.

SHA-1 tergolong lebih aman daripada MD5 karena serangan-serangan yang dilakukan pada SHA-1 membutuhkan biaya yang sangat besar. Hal ini adalah sebuah keunggulan untuk SHA-1 karena tidak banyak *hacker* yang memiliki *processor* yang cukup kuat untuk melakukan *birthday attack* pada *digest* SHA-1. Sedangkan, serangan *birthday attack* untuk MD5 dapat memanfaatkan *processor* yang sudah lazim digunakan pada komputer sehari-hari.

Salah satu alasan SHA-1 lebih *robust* jika dibandingkan dengan MD5 adalah karena *digest* yang dihasilkan SHA-1 lebih panjang. Sehingga, serangan yang memanfaatkan iterasi membutuhkan waktu lebih lama jika menggunakan *processor* dengan kecepatan yang sama untuk menyerang kedua algoritma.

Walaupun SHA-1 lebih *robust* jika dibandingkan dengan MD5, tidak berarti SHA-1 sepenuhnya aman dari bentrokan ataupun serangan. Ada begitu banyak algoritma fungsi hash yang dapat dimanfaatkan. Bahkan sudah terbukti bahwa SHA-256 lebih robust jika dibandingkan dengan SHA-1. Namun, tidak dilakukan penelitian dan analisis lebih dalam pada algoritma SHA-256 dalam makalah ini.

Hasil yang didapat pada makalah ini adalah SHA-1 masih aman digunakan untuk pengamanan kata sandi yang akan dipergunakan untuk hal yang tidak terlalu kritis. Namun, untuk penggunaan yang kritis seperti misalnya pertahanan keamanan data sebuah negara seperti fungsi militer, sebaiknya tidak memanfaatkan fungsi hash SHA-1 karena memang sudah diketahui metode penyerangan yang perlu dilakukan untuk memecahkan kode walaupun membutuhkan CPU yang sangat canggih.

REFERENCES

[1] Dobbertin, Hans. 1996. "The Status of MD5 After a Recent Attack". CryptoBytes 2, Vol.2.

[2] Xiaoyun, Wang., Hongbo, Yu. (2005). "How to Break MD5 and Other Hash Functions". EUROCYPT. ISBN 3-540-25910-4.

[3] H. Dobbertin. 1996. The status of MD5 after a recent attack, CryptoBytes 2 (2), ftp://ftp.rsasecurity.com/pub/cryptobytes/crypto2n2.pdf.

[4] Biham, E., & Chen, R. 2004. Near collision for SHA-0, Advances in Cryptology, Crypto'04. LNCS 3152, pp. 290-305

[5] I. B. Damgard. 1990. A design principle for hash functions, Advances in Cryptology. *Crypto'89 Proceedings*.

[6] Gilbery, H., & Handschuch, H. 2003. Security Analysis of SHA-256 and Sisters. *Selected Areas in Cryptography*. Pp 175–193.

[7] Wang, X., Feng, D,. Lai, X,. Yu, H. 2004. Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD, Cryptology ePrint Archive Report 2004/199.

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, 20 Mei 2013

Hanifah Azhar 13509016