ISSN 1693-1629 vol.2 no.1/Mei, 2004

Jurnal Ilmiah

ilmu komputer

Publikasi Hasil Penelitian Departemen Ilmu Komputer
Institut Pertanian Bogor

Departemen Ilmu Komputer
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Pertanian Bogor

Jurnal Ilmiah **ilmu komputer**

Diterbitkan oleh: Departemen Ilmu Komputer
Fakultas Matematika dan Ilmu Pengetahuan Alam - Institut Pertanian Bogor

Vol. 2. No. 1 . Mei 2004
ISSN : 1693-1629. Tanggal 4 April 2003

Susunan Redaksi

Sekretariat Jurnal Ilmiah Ilmu Komputer :

Departemen Ilmu Komputer FMIPA IPB
Jln. Raya Pajajaran, Kampus Baranangsiang Bogor 16144
Telp/Fax : 0251-356653. E-mail : jurnal@ilkom.fmipa.ipb.ac.id
Rekening : Tabungan Taplus BNI Pajajaran Bogor.
No: 061.000322402.911 A.n : Annisa/Jurnal Ilkom

Jurnal Ilmiah Ilmu Komputer diterbitkan dua kali setahun, memuat tulisan ilmiah yang berhubungan dengan bidang Ilmu Komputer dan merupakan Media Publikasi Ilmiah di lingkungan Departemen Ilmu Komputer FMIPA-IPB.
Tulisan Ilmiah dapat berupa hasil penelitian, bahasan tentang metodologi, tulisan populer, dan tinjauan buku.

Pihak perorangan / alumni yang telah memperoleh Jurnal Ilmu Komputer mohon mengganti biaya cetak Rp50.000,- / exemplar. ditransfer melalui Tabungan Taplus ENI Pajajaran . No.Rek : 061.000322402.911. A.n : Annisa / Jurnal Ilkom.

Daftar Isi

Analisis Dan Penerapan Algoritma RIPEMD-160 Sebagai Fungsi Penyandi Dalam Proses Autentikasi Password

Hardyan Eka P.[1], Sugi Guritman[2], Agus Buono[1]

[1] *Departemen Ilmu Komputer FMIPA IPB*
[2] *Departemen Matematika FMIPA IPB*

Abstrak

Setiap pengguna komputer memiliki hak akses yang berbeda-beda dalam penggunaan sumber daya yang ada di suatu komputer. Untuk menentukan batasan hak akses ini, biasanya digunakan skema autentikasi yang umumnya menggunakan password sebagai bukti keabsahan seseorang untuk mengakses sumber daya tersebut. Namun demikian ada faktor-faktor yang bisa dimanfaatkan oleh orang yang kurang menghargai privasi orang lain untuk menggunakan sumber daya yang bukan haknya. Untuk mengatasi masalah ini perlu dibuat suatu skema autentikasi yang baik dengan tambahan mekanisme-mekanisme pendukung untuk melindungi sumber daya seorang pengguna. Salah satu mekanisme pendukung yang digunakan dan menjadi topik bahasan dalam penelitian ini adalah kriptografi. Algoritma kriptografi berfungsi sebagai fungsi penyandi dalam suatu skema autentikasi.

Dalam penelitian ini dideskripsikan dan dianalisis algoritma RIPEMD-160 baik dari segi teori, keamanan, kompleksitas algoritma dan hasil implementasi dimana sebagai pembandingnya adalah algoritma MD5 dan SHA1. Selain itu, seperti halnya algoritma RIPEMD-160, dalam penelitian ini juga dideskripsikan dan dianalisis sebuah skema autentikasi sederhana buatan Manber, 1994, dengan sedikit modifikasi untuk memudahkan implementasi skema kedalam program komputer, yang menggunakan algoritma RIPEMD-160 sebagai fungsi penyandinya.

Kata kunci: *kriptografi, RIPEMD-160, fungsi hash, autentikasi, skema password*

PENDAHULUAN

Latar Belakang

Saat ini hampir semua sistem operasi sudah mendukung sistem multi-pengguna dimana satu komputer dapat digunakan oleh beberapa pengguna. Penggunaan jaringan komputer yang memungkinkan komputer yang satu berkomunikasi dengan komputer yang lain, juga bukan sesuatu hal yang baru lagi. Setiap pengguna dapat menggunakan sumber daya yang ada di suatu komputer sesuai dengan hak akses yang dimilikinya. Untuk menentukan batasan hak akses ini, biasanya digunakan suatu skema yang dikenal dengan istilah **skema autentikasi** yang umumnya menggunakan password sebagai bukti keabsahan seseorang untuk mengakses sumber daya tersebut.

Namun demikian, kemajuan teknologi informasi ini, yang ditandai dengan munculnya hardware dan software dengan kemampuan yang lebih canggih, sering disalahgunakan orang-orang yang tidak bertanggung jawab untuk menggunakan sumber daya yang bukan haknya. Hal ini dimungkinkan karena informasi yang berkaitan dengan pengguna, termasuk password, dan aplikasi disimpan dalam suatu file yang dapat dilihat oleh *superuser*. Faktor lain yang memungkinkan hal ini terjadi adalah kurangnya kesadaran orang akan pentingnya suatu password yang aman sehingga password yang digunakannya mudah untuk ditebak. Oleh karena itu, untuk mengatasi masalah diatas, perlu dibuat suatu skema autentikasi yang baik dengan tambahan mekanisme-mekanisme pendukung untuk melindungi password yang digunakan pengguna.

Salah satu mekanisme pendukung yang dapat digunakan untuk mengatasi masalah penyalahgunaan hak akses tersebut dan menjadi topik utama dalam penelitian ini adalah **kriptografi**. Dengan menggunakan mekanisme pendukung ini, data atau informasi yang dimaksud akan disandikan sehingga tidak dapat dibaca atau paling tidak akan menyulitkan orang, yang tidak berhak, untuk membacanya.

Tujuan

Penelitian ini bertujuan untuk :

1. Mempelajari, memahami cara kerja, dan melakukan analisis algoritma RIPEMD-60.

Sedangkan pembandingnya adalah algoritma *MD5* dan *SHA1*.

2. Mempelajari dan memahami skema password secara umum dan mekanisme-mekanisme tambahan yang bisa diimplementasikan didalamnya.
3. Mengimplementasikan algorima *RIPEMD-160* dan mekanisme-mekanisme pendukung ke dalam suatu skema *password*, dengan harapan dihasilkan suatu sistem autentikasi *password* sederhana yang lebih aman.

Ruang Lingkup

Penelitian ini dititikberatkan pada analisis terhadap algoritma *RIPEMD-160* yang diimplementasikan sebagai fungsi penyandi pada skema *password* yang meliputi analisis teori, analisis kompleksitas algoritma, dan analisis performa dengan batasan input antara 13 sampai dengan 26 karakter. Untuk mempermudah penelitian, diasumsikan bahwa proses inisialisasi nilai vektor awal (*Initial Vector* [*IV*]) sudah dijamin keamanannya.

TINJAUAN PUSTAKA

Kriptografi

Kriptografi dapat diartikan sebagai suatu studi matematika yang berkaitan dengan aspek keamanan seperti kerahasiaan, konsistensi, autentikasi dan non-repudiasi *(Guritman, 2003)*.

Algoritma Kriptografi

Algoritma kriptografi adalah suatu fungsi yang digunakan untuk melakukan proses enkripsi dan dekripsi *(Indrajit, et al, 2002)*.

Penentuan bagus atau tidaknya suatu algoritma kriptografi ditentukan oleh beberapa aspek, diantaranya *(Menezes, et al, 1996)* :

1. *Tingkat keamanan.*
2. *Fungsionalitas.*
3. *Metode operasi.*
4. *Kecepatan (performance).*
5. *Kemudahan dalam implementasi.*

Fungsi Hash Satu Arah (One-way hash function)

Fungsi Hash satu arah adalah suatu fungsi H yang memetakan string bit dengan panjang acak ke string bit dengan panjang tetap dengan tambahan atribut *(Guritman, 2003)*:

1. Deskripsi dari fungsi H diketahui secara umum.
2. Diberikan suatu fungsi dan pesan M, maka mudah menghitung nilai *hash* dari pesan tersebut.
3. Diberikan nilai *hash* dari fungsi H, sangat sulit untuk mendapatkan suatu pesan M dimana H*(M)* = nilai *hash* yang diberikan tersebut dan jika diberikan pesan M dan fungsi H, sangat sulit untuk menemukan pesan lain *(M')* sehingga H*(M')* = H*(M)*.

Nilai Hash (Message Digest(MD))

Nilai *hash* ialah nilai yang dihasilkan dari pemrosesan suatu pesan (*message*) dengan fungsi *hash (Menezes, et al, 1996)*. Nilai *hash* juga sering disebut dengan istilah-istilah seperti *digest*, kode *hash* (*hashcode*), atau total *hash* (*hash-total*).

Kriptanalisis

Kriptanalisis merupakan istilah yang khusus berlaku dalam fungsi hash, adalah studi matematika untuk mencoba mematahkan teknik kriptografik, dan lebih umum lagi layanan keamanan informasi *(Guritman, 2003)*. Orang yang menggeluti kriptanalisis ini disebut ***kriptanalis*** *(Guritman, 2003)*.

Serangan (Attack) pada Fungsi Hash

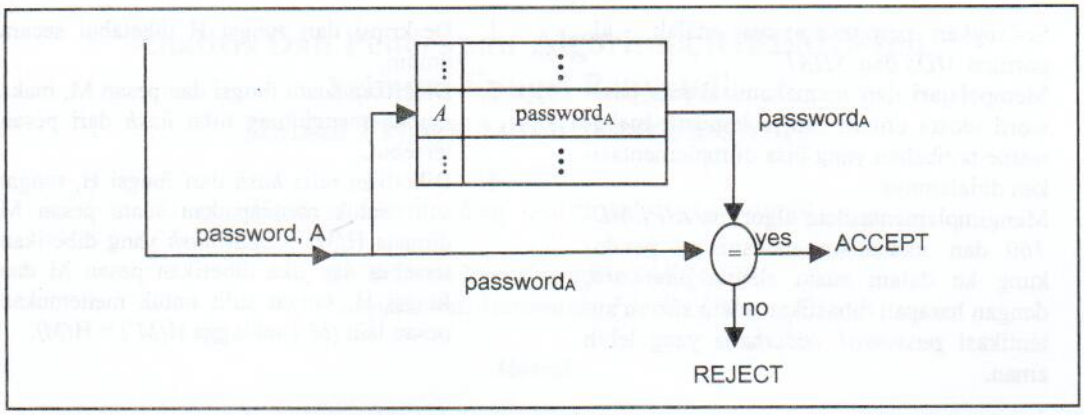
Serangan adalah usaha yang dilakukan oleh kriptanalis untuk mematahkan suatu layanan keamanan. Suatu serangan dikatakan berhasil jika dapat mengalahkan atribut keamanan yang dimiliki oleh suatu fungsi *hash*.

***Padding* Pesan**

Padding adalah proses penambahan bit pada suatu pesan asli *(yang direpresentasikan dalam bilangan biner)* sehingga panjang bit pesan asli tersebut sama dengan m*n dimana m adalah ukuran blok *(jumlah bit pada blok)* yang digunakan dalam algoritma dan n adalah jumlah bloknya.

Password

Password, diasosiasikan dengan sebuah entiti, adalah sebuah string, biasanya terdiri 6 sampai 10 karakter yang mudah diingat oleh pemiliknya dan menjadi rahasia bersama dengan sistem, yang digunakan sebagai bukti keabsahan dari identitas seseorang *(Menezes, et al, 1996)*.



Gambar 1. *Bentuk awal skema password pada saat pertama kali dibuat*

Skema Password

Skema *password* adalah sekumpulan algoritma, yang saling bekerja sama, untuk proses autentikasi *password.*

Dibawah ini adalah beberapa teknik yang bisa digunakan dalam mendisain sebuah skema *password* yang baik *(Menezes, et al, 1996) :*

1. Menyimpan *password* dalam database
2. Mengenkripsi *password* yang disimpan dalam database.
3. Mengontrol pemilihan *password*
4. Memperlambat pemetaan *password* dengan memperbanyak jumlah iterasi.
5. *Password salt*
6. Penggunaan frase sebagai *password*

Autentikasi Entitas (Identifikasi)

Identifikasi adalah sebuah proses, merupakan implementasi dari sebuah skema *password,* dimana sebuah entiti diyakinkan, dengan bukti yang kuat, akan identitas entiti lainnya yang terlibat dalam suatu protokol *(Guritman, 2003).*

Endian Kecil(Little-Endian) Vs Endian Besar (Big-Endian)

Istilah endian merujuk pada suatu metode atau cara penyimpanan suatu elemen, dengan ukuran yang identik, dalam memori. Ada dua metode pengkonversian yang digunakan untuk penyimpanan byte dalam memori, yaitu endian kecil dan endian besar.

Dalam metode *endian* kecil, *byte* dengan nilai terkecil (*least significant byte*) dari nilai aslinya disimpan dalam memori dengan alamat terkecil pada selang alamat memori yang digunakan untuk menyimpan nilai asli tersebut. Kemudian diikuti dengan *byte* berikutnya sampai byte dengan nilai terbesar (*most significant byte*) tersimpan pada alamat memori terbesar dari selang tersebut. Sedangkan metode endian besar sebaliknya.

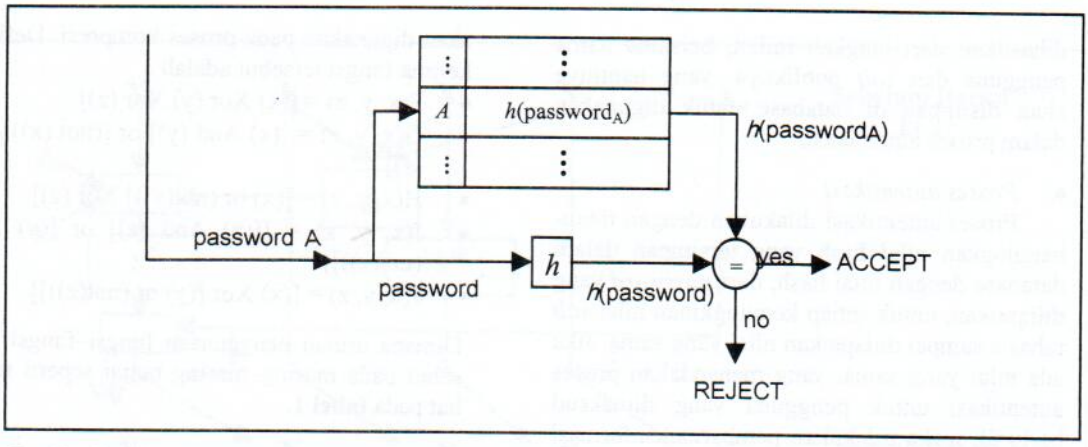
DESKRIPSI

Skema Password

Pada awalnya skema *password* hanya merupakan skema sederhana yang menyimpan pasangan ID-pengguna dan *password*nya dalam suatu file teks tanpa menggunakan mekanisme pendukung seperti dapat dilihat pada **gambar 1.** Hal ini sangat berbahaya karena jika kriptanalis berhasil mendapatkan otoritas sebagai *superuser,* maka dia dapat melihat dan dapat langsung menggunakan pasangan ID-pengguna dan *password* yang ada didalam file tersebut untuk kemudian login kedalam sistem dengan seakan-akan kriptanalis tersebut merupakan pengguna yang sah.

Untuk mengatasi masalah yang ada pada bentuk awalnya, maka para pembuat skema password mencoba menerapkan mekanisme-mekanisme pendukung sehingga dihasilkan skema *password* yang lebih aman. Ada skema yang menggunakan database sehingga bisa memanfaatkan fungsi-fungsi yang terintegrasi di dalamnya, ada juga yang memanfaatkan algoritma kriptografi untuk mengenkripsi *password* yang akan disimpan di database dan lain-lain.

Saat ini skema *password* yang umum digunakan adalah seperti **gambar 2.**Pada skema ini informasi pengguna disimpan dalam database sehingga fungsi terintegrasi yang ada di database dapat dimanfaatkan untuk membantu melindungi dan mempermudah manajemen data yang tersimpan didalamnya. Selain itu dalam skema ini *password* yang tersimpan di database merupakan *password* yang sudah di-



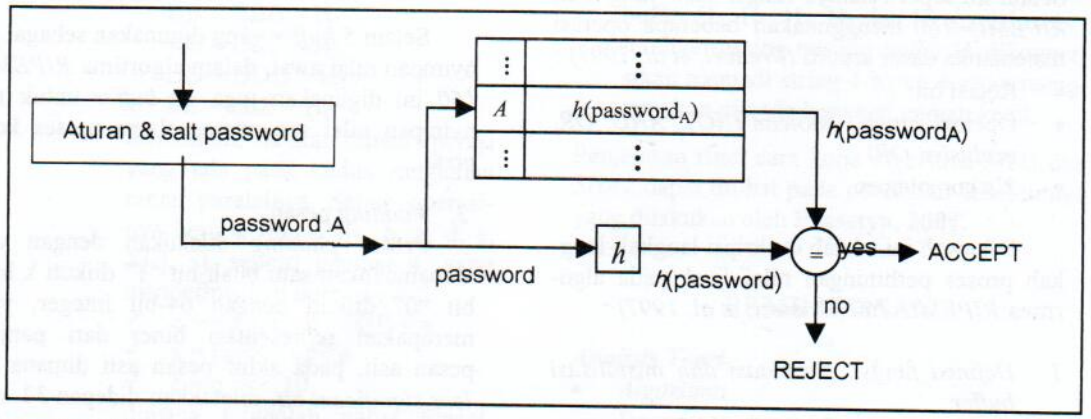
***Gambar 2**. Skema password yang umum digunakan saat ini*

hash sehingga akan mempersulit atau paling tidak memperlambat penggunaan data jika ada orang yang berhasil mendapatkan file tersebut. Namun demikian, banyaknya pengguna yang kurang sadar akan pentingnya *password* yang baik membuat skema yang ada saat ini menjadi kurang aman. Banyak pengguna yang menggunakan namanya, tanggal lahirnya atau hal yang berkaitan dengan informasi pengguna sehingga kriptanalis dengan mudah dapat menebak *password* yang digunakan. Dalam penelitian ini, dicoba diimplementasikan kedalam suatu program sederhana teori skema *passowrd* buatan **Manber, 1994** yang menggunakan dua buah salt *password,* untuk mempersulit dan memperlama waktu yang diperlukan dalam menebak *password,* dengan sedikit modifikasi untuk mempermudah pemrograman. Selain itu, dalam skema ini juga diimplementasikan aturan pemilihan *password* untuk menjamin bahwa *password* yang digunakan pengguna termasuk dalam kriteria *password* yang baik. Skema ini, yang dapat dilihat pada **gambar 3**, secara garis besar terbagi menjadi dua proses utama, yaitu proses daftar dan proses autentikasi *(login).* Dibawah ini adalah deskripsi dari masing-masing proses tersebut :

- *Proses daftar*

  Proses input diawali melakukan pengecekan terhadap ID-pengguna dan *password* yang diinputkan agar memenuhi syarat yang telah ditentukan.

Setelah nama pengguna dan *password* memenuhi syarat, kemudian dihitung nilai *hash*-nya. Perhitungan ini dimulai dengan membangkitkan dua buah bilangan yang berfungsi sebagai *salt* publik (*public salt*), didapat dari waktu sistem yang merupakan waktu pengguna *login,* dan *salt* rahasia (*secret salt*) yang merupakan hasil dari pembangkitan nilai acak. Setelah nilai kedua *salt* ini didapat, keduanya ditambahkan ke bagian akhir *password* yang diinputkan pengguna dan kemudian dihitung nilai *hash*-nya dengan menggunakan fungsi penyandi. Nilai yang



***Gambar 3.** Skema password dalam penelitian ini*

dihasilkan dari langkah inilah, bersama nama pengguna dan *salt* publiknya, yang nantinya akan disimpan di database untuk digunakan dalam proses autentikasi.

- *Proses autentikasi*

Proses autentikasi dilakukan dengan membandingkan nilai hash yang tersimpan dalam database dengan nilai hash, dari *password* yang diinputkan, untuk setiap kemungkinan nilai *salt* rahasia sampai didapatkan nilai yang sama. Jika ada nilai yang sama, yang menandakan proses autentikasi untuk pengguna yang dimaksud berhasil, maka dilakukan pembaruan informasi yang tersimpan di dalam database yang berkaitan dengan pengguna tersebut. Proses pembaruan informasi ini hampir sama dengan proses daftar, hanya saja nilai hash dan salt publiknya bukan disimpan sebagai *record* baru di dalam database, melainkan digunakan sebagai informasi terbaru untuk pengguna yang bersangkutan. Sedangkan jika tidak ditemukan nilai yang sama untuk semua kemungkinan nilai *salt* rahasia, maka proses autentikasi akan ditolak.

Algoritma RIPEMD-160

Algoritme RIPEMD-160 [*RACE Integrity Primitives Evaluation Message Digest–160*], dirancang oleh Bart Preneel, Antoon Bosselaers dan Hans Dobbertin pada tahun 1996, merupakan fungsi *hash* kriptografi yang cepat yang dibuat untuk diimplementasikan pada software yang dijalankan pada mesin berarsitektur 32-bit.

Dalam proses pemetaan string dengan panjang acak ke string dengan panjang tetap, salah satu varian *MD4* ini menggunakan dua rantai fungsi komputasi yang berbeda dan saling bebas yang dirangkaikan secara paralel. Selain itu seperti halnya fungsi *hash* yang lain, *RIPEMD–160* menggunakan beberapa operasi matematika dasar seperti *(Preneel, et al, 1997)* :

- Rotasi bit
- Operasi *bitwise boolean (NOT, AND, OR, exclusive-OR)*
- 2's komplemen

Dibawah ini adalah deskripsi langkah-langkah proses perhitungan nilai *hash* pada algoritma *RIPEMD-160 (Preneel, et al, 1997) :*

1. *Definisi fungsi, permutasi dan inisialisasi buffer*

Sebelum proses perhitungan, terlebih dahulu didefinikan fungsi-fungsi dan permutasi yang akan digunakan pada proses kompresi. Definisi kelima fungsi tersebut adalah :

- F(x, y, z) = [(x) Xor (y) Xor (z)]
- G(x, y, z) = [(x) And (y)] or [(not (x)) And (z)]
- H(x, y, z) = [(x) or (not(y))] Xor (z)]
- I(x, y, z) = [[(x) And (z)] or [(y) And (not(z))]]
- J(x, y, z) = [(x) Xor [(y) or (not(z))]]

Dimana urutan penggunaan fungsi–fungsi tersebut pada masing–masing rantai seperti terlihat pada **tabel 1.**

Tabel 1. *Urutan fungsi round pada fungsi kompresi algoritma RIPEMD-160*

Line	Round				
	1	2	3	4	5
Left	f_1	f_2	f_3	f_4	f_5
right	f_5	f_4	f_3	f_2	f_1

Sedangkan permutasi yang digunakan untuk menentukan urutan *word* seperti terlihat pada **tabel 2.**

Tabel 2. *Urutan pengacakan word*

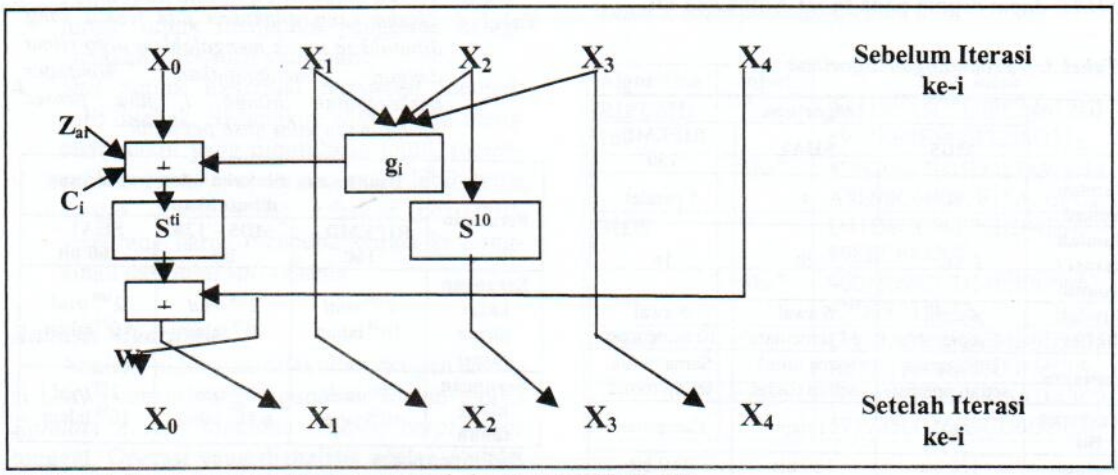
Line	Round				
	1	2	3	4	5
Left	Id	ρ	ρ^2	ρ^3	ρ^4
right	π	$\rho\pi$	$\rho^2\pi$	$\rho^3\pi$	$\rho^4\pi$

Dimana π = [(9 * Id) + 5] mod 16. Kemudian disiapkan juga suatu vektor awal, yaitu *buffer* (H_0, H_1, H_2, H_3, H_4), yang masing-masing nilainya dalam notasi heksa-desimal adalah:
H_0 : 67452301_X; H_1 : $EFCDAB89_X$
H_2 : $98BADCFE_X$; H_3 : 10325476_X
H_4 : $C3D2E1F0_X$

Selain 5 *buffer* yang digunakan sebagai penyimpan nilai awal, dalam algortima *RIPEMD-160* ini digunakan juga 10 *buffer* untuk menyimpan nilai sementara selama proses kompresi.

2. *Padding* pesan

Proses *padding* dilakukan dengan cara menambahkan satu buah bit "1" diikuti k buah bit "0" diikuti dengan 64-bit integer, yang merupakan representasi biner dari panjang pesan asli, pada akhir pesan asli dimana 32-*leas significant bit* diletakkan didepan 32-*most significant bit* sehingga panjangnya merupakan kelipatan 512.



Gambar 4. *Proses iterasi dalam fungsi round algoritma RIPEMD-160*

3. *Proses kompresi*

Bagian yang merupakan inti dari proses algoritma *RIPEMD-160* ini, terdiri dari dua buah rantai fungsi, masing-masing terdiri dari 5 *round*, dimana setiap *round*-nya terdiri dari 16 operasi, yang dirangkai secara paralel. Langkah-langkahnya adalah:

a. Pendefinisian fungsi F dan F', konstanta K dan K' untuk masing-masing *round*.

b. Pada setiap blok $M_1, M_2, .. M_n$ lakukan langkah-langkah sebagai berikut:

- Bagi M_i ke dalam 16 subblok 32 bit yaitu X[k], $0 \leq k \leq 15$, menggunakan metode endian kecil, dengan mengubah susunan urutan byte, dimana paling kanan dijadikan *high order* diikuti oleh byte selanjutnya sampai dengan byte paling kiri dijadikan *low order*.
- Set AA = AAA = H_0,
  BB = BBB = H_1,
  CC = CCC = H_2,
  DD = DDD = H_3,
  EE = EEE = H_4.
- Kemudian, lakukan semua operasi yang ada pada kedua rangkaian rantai paralelnya. Setiap operasinya berupa fungsi f[a, b, c, d, e, X[i], s], seperti gambar 4, yang mendefinisikan operasi :
  - ✓ a += f(b, c, d) + X[i] + k
  - ✓ a = [a <<< s] + e
  - ✓ c = c <<< 10

  dimana f adalah nama fungsi. *Buffer* a, b, c, d dan e adalah *buffer* untuk menampung nilai sementara, X[i] adalah notasi dari 16 subblok dan k adalah konstanta yang digunakan untuk masing-masing *round*. Sedangkan <<<s adalah operasi pergeseran bit ke kiri sebanyak s bit.

c. Untuk penyelesaian akhir, lakukan penjumlahan seperti berikut :

$ddd = H_1 + cc + ddd$
$H_1 = H_2 + dd + eee$
$H_2 = H_3 + ee + aaa$
$H_3 = H_4 + aa + bbb$
$H_4 = H_0 + bb + ccc$
$H_0 = ddd$

4. *Output*

Output didapat setelah semua blok M_1, $M_2, .., M_n$ selesai diproses. Nilai *hash*-nya merupakan penggabungan nilai kelima buffer H_0, H_1, H_2, H_3 dan H_4 setelah sebelumnya masing-masing *buffer* H_i dikonversikan menjadi string 4-bytes dengan menggunakan metode konversi endian kecil.

Penjelasan rinci cara kerja algoritma *MD5* dan *SHA1* dapat dilihat pada penelitian sebelumnya yang dilakukan oleh **Prasetya, 2001.**

PEMBAHASAN

Analisis Teori

- *Algoritma*

Persamaan dan perbedaan antara algoritma *RIPEMD-160* dengan algoritma *MD5* dan

SHA1 dapat dilihat pada **tabel 3** dibawah ini:

Tabel 3. *Perbandingan algoritma*

	Algoritma		
	MD5	**SHA1**	**RIPEMD 160**
Jumlah round	4	4	5 paralel
Jumlah iterasi / round	16	20	16
Jumlah Buffer	4 awal 4 sementara	5 awal 64 sementara	5 awal 10 sementara
Konstanta	Unik untuk setiap operasi	Sama untuk setiap round	Sama untuk setiap round
Pergeseran Bit	Acak	Tetap	Campuran
Output	128 bit	160 bit	160 bit
Konversi	Endian kecil	Endian besar	Endian kecil

Tabel 4. *Usaha yang dilakukan dan waktu yang dibutuhkan untuk mengalahkan algo-ritma dengan menggunakan komputer berkecepatan hitung 1 juta proses perhitungan nilai hash per detik*

Serangan	**Usaha yang dilakukan dan waktu yang dibutuhkan**		
	RIPEMD – 160	**MD5 – 128 bit**	**SHA1 – 160 bit**
Serangan kasar (*brute force*)	2^{160} *trial* 10^{34} tahun	2^{128} *trial* 10^{25} tahun	2^{160} *trial* 10^{34} tahun
Serangan ulang tahun	2^{80} *trial* 10^{10} tahun	2^{64} *trial* 6.10^{5} tahun	2^{80} *trial* 10^{10} tahun

Trial = percobaan

- ***Skema Password***

Agar suatu skema *password* menjadi lebih baik biasanya dalam skema tersebut ditambahkan suatu mekanisme yang mendukung keamanan fungsi penyandinya. Mekanisme tambahan yang diterapkan pada skema ini adalah penggunaan file *database* untuk menyimpan pasangan nama pengguna, *salt* publik, dan *password* yang sudah dienkripsi. Mekanisme yang lain yang digunakan sebagai fungsi pendukung adalah aturan *password* dan *password salt*. Dalam skema ini digunakan dua buah *salt,* yaitu *salt* publik yang nilainya diambil dari waktu terakhir pengguna login ke sistem dan *salt* rahasia yang nilainya didapat dari pembangkitan bilangan secara acak.

Analisis Keamanan

- *Algoritma*

Penentuan aman tidaknya suatu algoritma kriptografi dilihat dari lamanya waktu yang dibutuhkan oleh suatu jenis serangan untuk mematahkan algoritma tersebut. Semakin lama waktu yang diperlukan berarti semakin baik pula algoritma tersebut. Pada **tabel 4.** yang memperlihatkan perbandingan waktu dan jumlah minimal percobaan yang di-perlukan untuk mematahkan algoritma-algoritma *MD5, SHA1* dan *RIPEMD-160* dengan menggunakan serangan acak dan serangan tanggal lahir.

Dari tabel dapat dilihat bahwa lamanya waktu yang diperlukan untuk mematahkan algoritma *SHA1* dan *RIPEMD-160* adalah sama, yaitu untuk serangan acak waktunya sekitar 10^{34} tahun dengan jumlah minimal percobaan yang dilakukan sekitar 2^{160} percobaan, sedangkan seorang kriptanalis yang menggunakan serangan tanggal lahir membutuhkan jumlah minimal percobaan sebanyak 2^{80} percobaan dengan lamanya waktu percobaan sekitar 10^{10} tahun. Untuk *MD5* waktu yang diperlukan sekitar 10^{25} tahun dengan jumlah minimal percobaan sekitar 2^{128} percobaan jika mengunakan serangann acak dan 6.10^{5} tahun dengan jumlah minimal percobaan sekitar 2^{64} percobaan jika menggunakan serangan tanggal lahir.

Dalam sebuah jurnalnya, perancang algoritma *RIPEMD-160* ini mengatakan bahwa algoritma ini diperkirakan dapat tahan dari serangan-serangan yang mungkin dilakukan terhadap algoritma kriptografi dan aman untuk digunakan sampai 20 tahun mendatang.

- *Skema Password*

Secara teori, penggunaan suatu algoritma *hash* yang diimplementasikan sebagai fungsi penyandi sudah cukup untuk membuat suatu skema *password* yang aman. Namun dengan menggunakan mekanisme pendukung akan lebih menjamin keamanan skema tersebut.

Adapun manfaat dari penggunaan mekanisme pendukung dalam skema ini adalah :

1. Mekanisme penggunaan file database sehingga bisa dimanfaatkan fungsi yang terintegrasi didalamnya.
2. Penyimpanan *password* dalam bentuk nilai *hash*-nya agar *password* tidak bisa langsung digunakan jika penyerang berhasil mendapatkan file *password.*

3. Penerapan aturan pembuatan *password* berfungsi untuk menjamin pengguna menggunakan *password* yang baik.
4. *Salt* publik membuat *password* menjadi sulit ditebak. Sedangkan *salt* rahasia membuat waktu yang dibutuhkan untuk memecahkan suatu *password* menjadi lebih lama karena untuk setiap tebakan *password,* seseorang harus mencoba semua kemungkinan dari nilai *salt* rahasia.

Analisis Algoritma

Analisis algoritma dilakukan dengan asumsi bahwa mesin yang digunakan adalah model *Random Access Machine (RAM),* berprosesor tunggal. Operasi yang dianalisis adalah operasi aritmatika dasar, yaitu penjumlahan, pengurangan, perkalian, pembagian dan operasi penugasan (*assignment*), yang dieksekusi baris per baris dengan lamanya waktu eksekusi setiap operasi tersebut dianggap satu satuan waktu.

Karena langkah 1, 2, dan 4 memiliki waktu eksekusi konstan, tidak terpengaruh oleh besar atau kecilnya ukuran input, maka yang menjadi fokus analisis ini adalah bagian proses kompresinya. Pada proses kompresi ini, untuk setiap pengolahan blok M_i diperlukan 160 operasi. Jadi untuk pesan yang terdiri dari n blok input diperlukan waktu eksekusi sebesar 160n.

Secara keseluruhan waktu eksekusi algoritma *RIPEMD-160* adalah 160n + a, dengan a yaitu suatu konstanta yang merupakan akumulasi waktu dari langkah 1, 2 dan 4. Jadi bentuk notasi–*O* untuk kasus terburuk dari proses *hash* pesan dengan menggunakan algoritma *RIPEMD-160* adalah 160n + a $\in$ *O*(n). Sedangkan waktu eksekusi untuk algoritma *MD5* adalah 64n + a $\in$ *O*(n), dan algoritma *SHA1* adalah 80n + a $\in$ *O*(n).

Analisis Hasil Implementasi dan Kecepatan

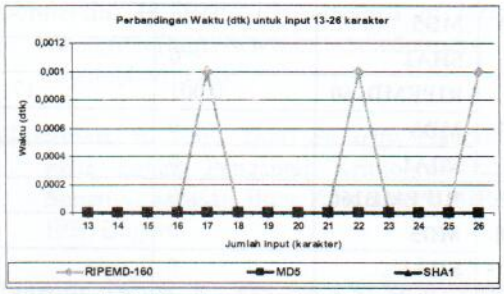
- *Algoritma*

Hasil perhitungan nilai *hash* untuk masing-masing algoritma terdapat pada **Tabel 5.**

Sedangkan **tabel 6** merupakan hasil pengujian algoritma dengan input antara 13 - 26 karakter, dilakukan sebanyak 40 kali percobaan untuk masing-masing algoritma, pada komputer *Pentium IV 1.7 GHz, RAM 256 MB.*

Tabel 5. *Hasil perhitungan nilai hash*

Algoritma	Input	Nilai *hash*
RIPEMD 160	""	9C1185A5C5E9FC5461280 8977EE8F548B2258D31$_X$
	"abc"	8EB208F7E05D987A9B044 A8E98C6B087F15A0BFC$_X$
MD5	""	D41D8CD98F00B204E9800 998ECF8427E$_X$
	"abc"	900150983CD24FB0D6963 F7D28E17F72$_X$
SHA1	""	DA39A3EC5E6B4B0D3255 BFEF95601890AFD80709$_X$
	"abc"	A9993E364706816ABA3C2 5717850C26C9CD0D89D$_X$



Gambar 5. *Perbandingan waktu ketiga algoritma untuk input 13-26 karakter*

Waktu proses yang berkisaran dinilai 0 pada tabel diatas didapat karena ketiga algoritma tersebut memproses inputnya per 64 karakter sehingga pada input 13 – 26 karakter hanya dilakukan satu kali proses.

Untuk memperlihatkan perbedaan kecepatan antara ketiga algoritma diatas, maka dilakukan pengujian tambahan menggunakan input 38.4 - 192 juta karakter yang hasilnya seperti pada **tabel 7.**

- ***Skema Password***

Pengujian hasil implementasi dan kecepatan skema *password* dilakukan untuk memperlihatkan pengaruh perbedaan spesifikasi komputer yang digunakan terhadap batas atas nilai *salt* rahasia yang dapat dipergunakan.

Pengujian ini dilakukan dengan cara membuat suatu program sederhana yang merupakan implementasi dari skema *password* yang dideskripsikan diatas dan kemudian dipilih nilai maksimal yang masih dapat digunakan tanpa melewati batas waktu yang diijinkan untuk proses autentikasi yaitu ½ detik *(Manber, 1994).*

Pengujian ini dilakukan pada 2 komputer, yang memiliki spesifikasi berbeda, dengan hasil sebagai berikut :

***Tabel** 6. Hasil pengujian waktu dan kecepatan proses untuk input 13 – 26 karakter*

Ukuran Input (dalam Bytes)		Waktu (detik)	V_{rata2} (byte/ detik)
13	RIPEMD160	0	
	MD5	0	
	SHA1	0	
14	RIPEMD160	0	
	MD5	0	
	SHA1	0	
15	RIPEMD160	0	
	MD5	0	
	SHA1	0	
16	RIPEMD160	0	
	MD5	0	
	SHA1	0	
17	RIPEMD160	0.001	17
	MD5	0	
	SHA1	0	
18	RIPEMD160	0	
	MD5	0	
	SHA1	0	
19	RIPEMD160	0	
	MD5	0	
	SHA1	0	
20	RIPEMD160	0	
	MD5	0	
	SHA1	0	
21	RIPEMD160	0	
	MD5	0	
	SHA1	0	
22	RIPEMD160	0.001	22
	MD5	0	
	SHA1	0	
23	RIPEMD160	0	
	MD5	0	
	SHA1	0	
24	RIPEMD160	0	
	MD5	0	
	SHA1	0	
25	RIPEMD160	0	
	MD5	0	
	SHA1	0	
26	RIPEMD160	0.001	26
	MD5	0	
	SHA1	0	

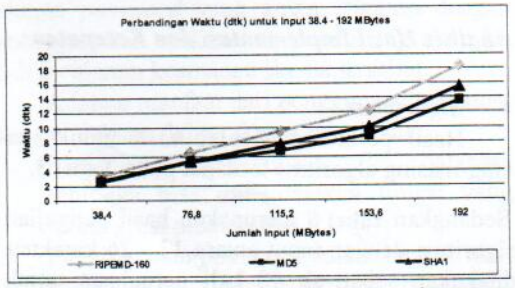
Tabel 6 dapat digambarkan grafiknya seperti pada **Gambar 5.**

- Pada komputer 1 yang memiliki spesifikasi berupa *prosesor intel pentium IV 1.7 GHz*, dengan *RAM 256 Mbytes*, batas atas nilai *salt* rahasia yang bisa digunakan adalah 3900.
- Sedangkan untuk Komputer 2 dengan spesifikasi berupa *prosesor AMD Athlon 650 MHz* dan *RAM 640 Mbytes*, batas atas yang bisa dipergunakan adalah 2100.

Dari **tabel 7** dapat dilihat bahwa algoritma *RIPEMD-160* memiliki waktu yang paling lama dalam proses perhitungan nilai *hash*nya dengan rata-rata waktu proses 9.989 detik sebagaimana diperlihatkan pada **gambar 6.**

***Tabel** 7. Hasil pengujian waktu dan kecepatan proses untuk input 38.4 – 192 juta karakter*

Ukuran Input (dalam MBytes)		Waktu (detik)	V_{rata2} (byte/detik)
38.4	RIPEMD160	3.36	11448234
	MD5	2.439	15775099
	SHA1	2.78	14183956
76.8	RIPEMD160	6.562	11850355
	MD5	5.273	16461963
	SHA1	5.374	14395239
115.2	RIPEMD160	9.39	12300834
	MD5	6.678	17281753
	SHA1	7.659	15153269
153.6	RIPEMD160	12.285	12567971
	MD5	8.853	17362490
	SHA1	10.028	15316461
192	RIPEMD160	18.346	10452208
	MD5	13.527	14696114
	SHA1	15.496	12858894



***Gambar** 6. Perbandingan waktu ketiga algoritma untuk input 38.4 – 192 MBytes*

KESIMPULAN DAN SARAN

Kesimpulan

Berdasarkan hasil pembahasan diatas dapat ditarik kesimpulan sebagai berikut:

1. Rangkaian rantai paralel yang dimilikinya, menjadikan algoritma *RIPEMD–160* lebih aman untuk digunakan meskipun waktu proses perhitungan nilai *hash*nya lebih lama jika dibandingkan dengan *MD5* dan *SHA1*. Hal ini dapat dilihat dari kompleksitas algoritmanya pada kasus terburuk, dimana kompleksitas algoritma *RIPEMD–160* adalah 160n + a $\in O(n)$, algoritma *MD5* adalah 64n + a $\in O(n)$ sedangkan algoritma *SHA1* adalah 80n + a $\in O(n)$. Selain itu pada tabel 6 juga dapat dilihat bahwa untuk setiap jenis input *RIPEMD–160* memiliki waktu yang paling lama dengan rata-rata waktu proses 9.989 detik.
2. Setiap mekanisme pendukung memiliki manfaat masing-masing dalam setiap penggunaannya. Beberapa diantaranya :
   - Penggunaan database dengan fungsi-fungsi yang terintegrasi menjamin keamanan dan membuat manajemen data yang tersimpan didalamnya menjadi lebih mudah jika dibandingkan dengan penyimpanan data dalam file teks biasa.
   - Penyimpanan *password* dalam bentuk nilai *hash*nya berfungsi untuk mempersulit orang yang ingin mengetahui password seorang pengguna.
   - Aturan pembuatan *password* berfungsi untuk memastikan bahwa *password* yang digunakan oleh pengguna merupakan *password* yang baik.
   - *Salt* publik membuat suatu *password* sulit untuk ditebak sedangkan *salt* rahasia berfungsi untuk memperlambat kecepatan dalam memecahkan suatu *password*.
3. Semakin baik spesifikasi komputer yang digunakan, maka semakin besar pula nilai maksimal yang mungkin digunakan untuk *salt* rahasia.

Saran

Pada penelitian selanjutnya disarankan untuk melakukan penelitian terhadap algoritma-algoritma lain yang merupakan pengembangan dari algoritma *RIPEMD-160* seperti algoritma *MAC*. Saran lainnya adalah dilakukan penelitian untuk pengembangan suatu skema *password* yang lebih aman yang dapat diintegrasikan secara langsung dengan skema yang yang sudah ada sehingga skema yang lama tetap dapat digunakan.

DAFTAR PUSTAKA

Bakhtiari, S., Naini, R. S., Pieprzyk, J. 1995. Cryptographic Hash Functions : A Survey. citeseer.nj.nec.com/bakhtiari95cryptographic.html

Dobbertin, H. 1996. *Cryptanalysis of MD5 Compress*.(http://www.cs.ucsd.edu/users/bsy/dobbertin.ps)

Guritman, S. 2003. *Data Security*. Handout mata kuliah Pengantar Kriptografi/Data Security. Jurusan Ilmu Komputer FMIPA IPB. Bogor.

Indrajit, Dr. R. E., Prastowo M.Sc, Drs. B. N., dan Yuliardi, R. 2002. *Buku Pintar Linux -Memahami Security Linux*. PT. Elex Media Komputindo Kelompok Gramedia Jakarta. Jakarta.

Manber, U. 1994. *A Simple Scheme to Make Passwords Based on One-Way Functions Much Harder to Crack*. (http://www.citeseer.nj.nec.com/manber96simple.html)

Menezes, A. P. van Oorschot and S. Vanstone. 1996. *Handbook of Applied Cryptography*. CRC Press, New York.

Prasetya, M. 2001. *Message Digest 5 (MD5) dan Secure Hash Algorithm 1 (SHA1) untuk Autentikasi Pesan*. Skripsi. Jurusan Ilmu Komputer FMIPA IPB, Bogor.

Preneel, B., Bosselaers, A., and Dobberin, H. 1997. *The Cryptographic Hash Function RIPEMD-160*. (ftp://ftp.rsasecurity.com/pub/cryptobytes/crypto3n2.pdf)

Preneel, B., Bosselaers, A., and Dobberin, H. 1997. *RIPEMD-160 : A strengthened Version of RIPEMD*. (www.esat.kuleuven-.ac.be/~cosicart/pdf/AB-9601/AB9601-.pdf)