Penggunaan Fungsi *Hash* BLAKE2 dalam Pendeteksian Virus Komputer

Muhamad Visat Sutarno 13513037
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
visat@students.itb.ac.id

***Abstrak*—Makalah ini membahas tentang penerapan fungsi *hash* BLAKE2 dalam mendeteksi sebuah virus komputer. Salah satu metode untuk mengidentifikasi sebuah virus komputer adalah dengan pendeteksian berbasis *signature* yang disimpan di dalam sebuah basis data. Basis data yang digunakan di dalamnya didefinisikan *signature* virus, nama virus, dan informasi terkait virus lainnya. *Signature* virus komputer pada umumnya direpresentasikan dengan sebuah *string* yang merupakan nilai *hash* dari berkas suatu virus. Nilai *hash* didapatkan dengan suatu fungsi *hash*. Fungsi *hash* yang akan dibahas dalam makalah ini adalah BLAKE2.**

***Kata Kunci*—BLAKE2, Fungsi Hash, Komputer, Virus.**

I. PENDAHULUAN

Dewasa ini, teknologi informasi berkembang dengan sangat pesat. Salah satu teknologi informasi yang dengan pesat berkembang adalah komputer. Dalam kehidupan sehari-harinya, kegiatan manusia sangat erat dengan penggunaan komputer. Penggunaan komputer tersebut meliputi banyak hal, mulai dari hanya untuk menjelajah dunia maya, bermain, sampai digunakan untuk bekerja secara profesional. Dalam penggunaan komputer tersebut, biasanya terjadi perpindahan maupun pertukaran data dari sebuah komputer ke komputer lainnya. Transfer data dari suatu komputer ke komputer lainnya dapat melewati banyak perantara, antara lain melalui internet, *removable disks* seperti *flash disk*, dll. Akan tetapi, tidak selamanya data yang ditransfer adalah data yang diinginkan. Tidak tertutup kemungkinan data yang sedang ditransfer malah merupakan virus komputer.

Virus komputer adalah program jahat yang ketika dijalankan akan menggandakan dirinya dengan cara menyisipkan salinan dirinya (yang mungkin telah diubah) ke dalam program komputer lainnya, berkas-berkas data, ataupun *boot sector* dari sebuah *hard drive*[1]. Virus komputer mempunyai tujuan-tujuan yang berbeda. Ada virus komputer yang bertujuan merusak komputer, memperlambat kinerja komputer, mencuri data-data yang bersifat pribadi, ataupun hanya merupakan hasil keisengan dari si pembuat virus. Walaupun terkadang ada yang sama sekali tidak berbahaya, virus komputer tetaplah mengganggu pengguna yang berhasil terinfeksi olehnya.

Virus-virus komputer menyebar dan berkembang dengan pesat. Pada kuartal ketiga pada tahun 2014 saja, Kaspersky Lab telah mendeteksi sebanyak 116.710.804 program-program berbeda yang berbahaya dan tidak diinginkan[13]. Untuk mengatasi hal tersebut, diciptakanlah sebuah piranti lunak untuk melawannya yang biasa disebut antivirus.

Peranti lunak antivirus pada awalnya hanya diciptakan untuk mendeteksi dan membasmi virus-virus komputer. Akan tetapi, antivirus-antivirus modern bukan hanya melakukan hal tersebut saja, melainkan juga melindungi pengguna dari serangan maya lainnya. Untuk mengidentifikasi virus-virus komputer yang jumlahnya jutaan, peranti lunak antivirus biasanya mempunyai metode-metode tersendiri. Namun, pada umumnya metode-metode tersebut secara umum sama. Metode yang paling awal diciptakan dan tetap digunakan antivirus modern adalah metode pendeteksian virus berbasis *signature*.

Metode pendeteksian virus berbasis *signature* adalah pendeteksian virus dengan cara mendapatkan sidik jari dari sebuah berkas dan mencocokkannya dengan yang ada di basis data yang telah didefinisikan. Jika ditemukan, maka berkas tersebut dinyatakan sebagai virus komputer. Untuk mendapatkan sidik jari tersebut biasanya antivirus menggunakan fungsi *hash*. Fungsi *hash* yang paling umum digunakan adalah MD5 dan SHA-1.

Fungsi *hash* MD5 yang sering digunakan sudah tergolong cepat. Akan tetapi, telah ditemukan adanya kecacatan pada fungsi *hash* MD5. Dua buah berkas yang berbeda dapat mempunyai sidik jari yang sama jika menggunakan fungsi *hash* MD5. Berkas-berkas di komputer pengguna pun semakin hari semakin bertambah besar ukurannya dan semakin banyak jumlahnya. Untuk menangani berkas-berkas tersebut dibutuhkan suatu fungsi *hash* yang bisa lebih cepat dari fungsi *hash* MD5. Untuk itu, suatu tim mengembangkan fungsi *hash* baru. Tim tersebut akhirnya berhasil dan menamakan fungsi *hash* temuan mereka dengan nama BLAKE2.

II. LANDASAN TEORI

A. Fungsi Hash

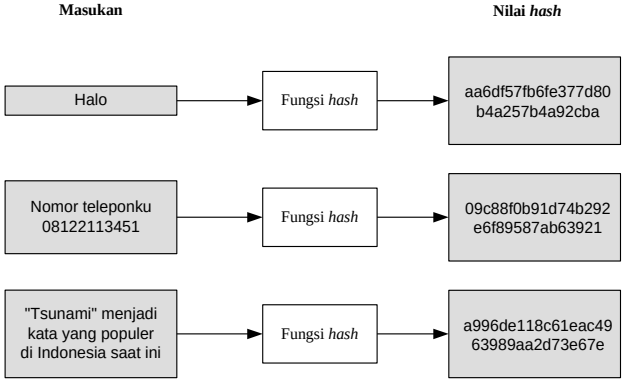
Fungsi *hash* adalah sebuah algoritme matematika yang menghasilkan sebuah nilai atau hasil yang ukurannya tetap dari sebuah data yang ukurannya sembarang[2]. Fungsi *hash* dituliskan dalam notasi persamaan:

$$h = H(M)$$

h : nilai *hash* yang ukurannya tetap
H : fungsi *hash*
M : pesan dengan ukuran sembarang

$$h \ll M$$

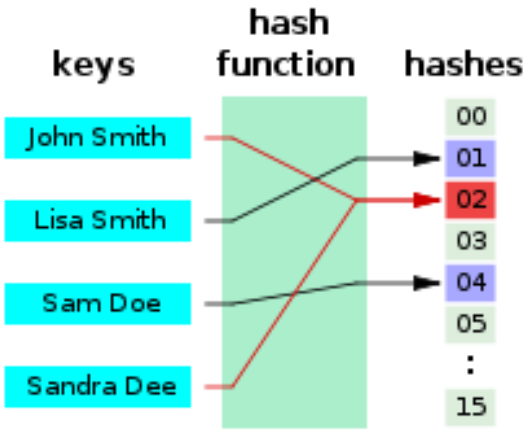
Nilai *hash* yang dihasilkan dapat disebut pesan ringkas atau *message digest*. Fungsi *hash* dapan mengubah pesan yang berukuran berapa saja menjadi *message digest* yang berukuran tetap dan biasanya lebih kecil dari ukuran pesan[3].



Gambar 2.1.1. Fungsi *Hash*
Sumber: Referensi [3]

Fungsi *hash* merupakan fungsi yang bersifat satu arah. Hal ini berarti akan sulit sekali mencari nilai balikan dari hasil fungsi ini, sehingga kita sulit untuk menemukan nilai input x yang memenuhi persamaan H(x) = h. Jika diberikan sebuah pesan x untuk dikompresi dalam suatu fungsi *hash*, kita akan sulit mencari pesan y yang akan menyamai dari nilai H(x), yang berarti sulit juga untuk mencari H(x) = H(y). Maka dapat kita katakan bahwa fungsi *hash* x tidak mungkin bertabrakan dengan fungsi *hash* y. Bisa dikatakan akan sangat sulit untuk menemukan kesamaan dua pesan dalam fungsi *hash*. Jadi fungsi *hash* adalah fungsi yang bersifat *strongly collision-free* (bebas dari tabrakan).

Collision sendiri merupakan gejala tabrakan yang terjadi ketika dua input menghasilkan *hash* yang sama. Tabrakan ini dapat dihindari dengan *Collision Resolution*. *Collision Resolution* merupakan proses untuk menangani kejadian dua atau lebih *key* di-*hash* ke alamat tabel yang sama. Cara yang dilakukan jika terjadi *collision* adalah mencari lokasi yang kosong dalam tabel *hash* secara terurut. Cara lainnya adalah dengan menggunakan fungsi *hash* yang lain untuk mencari lokasi kosong tersebut.

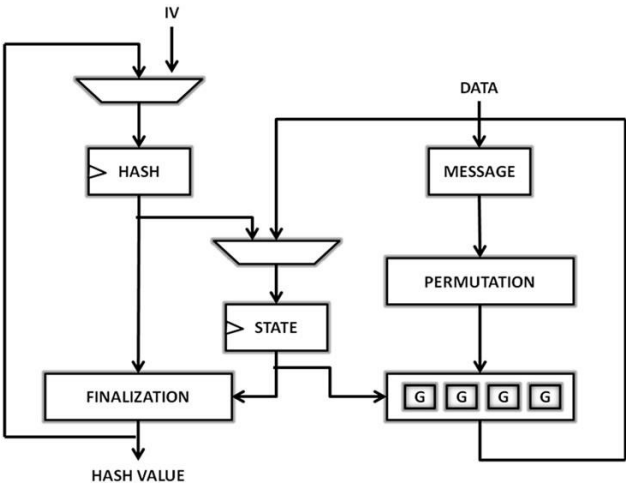


Gambar 2.1.2 Ilustrasi *Collision*
Sumber: http://id.wikipedia.org/wiki/Hash

Penggunaan fungsi *hash* ini erat kaitannya dengan dunia kriptografi karena sifat satu arah yang sulit untuk dikembalikan[4]. Contoh penggunaan yang paling terkenal dari fungsi *hash* adalah MD5 dan SHA.

B. BLAKE dan BLAKE2

Fungsi *hash* BLAKE didesain oleh Jean-Philippe Aumasson, Luca Henzen, Willi Meier, dan Raphael C.-W. Phan[5]. BLAKE adalah salah satu dari lima finalis kompetisi yang diadakan National Institute of Standards and Technology (NIST) untuk mengembangkan sebuah fungsi *hash* baru yang nantinya akan dinamakan SHA-3. Pada 2 Oktober 2012, diumumkan pemenang kompetisi tersebut yaitu Keccak, yang menjadi standar baru untuk SHA-3[6]. Walaupun tidak terpilih sebagai pemenang, BLAKE mempunyai tingkat keamanan yang tinggi, performa yang sangat baik, dan menarik perhatian banyak kriptoanalis[7].



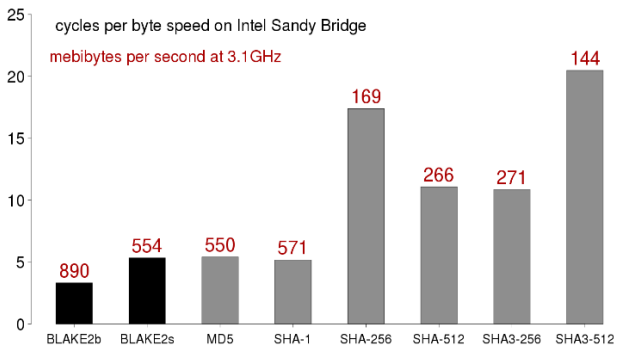
Gambar 2.2.1. Diagram Blok Algoritme BLAKE
Sumber: http://file.scirp.org/Html/1-7800059_18767.htm

Pada 21 Desember 2012, BLAKE2 yang didesain oleh Jean-Philippe Aumasson, Samuel Neves, Zooko Wilcox-

O'Hearn, dan Christian Winnerlein diumumkan[8][9]. BLAKE2 merupakan pembaharuan dari fungsi *hash* pendahulunya, BLAKE. Keccak, yang merupakan standar baru SHA-3, sebenarnya 20% lebih cepat dari SHA-2. Akan tetapi, beberapa perangkat lunak membutuhkan fungsi *hash* yang lebih cepat dari itu. Beberapa perangkat lunak pun tetap menggunakan fungsi *hash* MD5 untuk pengidentifikasian berkas dan dan pengecekan integritas data, walaupun MD5 telah ditemukan kecacatan dalam keamanannya. Pengembang BLAKE2 tersebut pun berusaha mengoptimasi BLAKE, sehingga terciptalah fungsi *hash* BLAKE2. Keunggulan BLAKE2 antara lain:

- Lebih cepat dari MD5 pada sistem Intel 64-bit
- Membutuhkan RAM 32% lebih sedikit daripada BLAKE
- Dukungan langsung terhadap:
  − Paralelisme, untuk performa *hashing* berkali-kali lebih cepat pada prosesor *multicore* atau SIMD
  − *Tree hashing*, untuk memverifikasi berkas berukuran besar
  − Prefix-MAC, untuk *authentication* yang lebih sederhana dan lebih cepat daripada HMAC
  − Personalisasi, untuk mendefinisikan *hash* yang unik untuk setiap aplikasi
- *Padding* yang minimal sehingga lebih mudah dan cepat untuk diimplementasikan

BLAKE2 dan SHA-3 saat ini belum ditemukan adanya masalah keamanan, sedangkan MD5 dan SHA-1 rentan terhadap *collision attacks*. Saat diuji coba dengan prosesor Intel modern, BLAKE2 pun menjadi yang tercepat mengalahkan MD5, SHA-1, SHA-2, dan SHA-3.



Gambar 2.2.2. Perbandingan Performa Fungsi *Hash*
Sumber: Referensi [11]

C. Pendeteksian Virus

Ada empat metode pendeteksian virus[10], yaitu:

1. Pendeteksian Virus Berbasis *Signature*

   Pendeteksian ini mengandalkan basis data yang berisi sidik jari statis dari virus-virus komputer yang telah diketahui sebelumnya. Sidik jari virus tersebut bisa merupakan rangkaian *byte* dalam berkasnya ataupun nilai *hash* dari sebuah fungsi *hash*. Pendeteksian virus dengan metode ini merupakan aspek penting dari sebuah antivirus sejak pertama kali antivirus diciptakan, dan masih banyak digunakan pada antivirus-antivirus modern sekarang. Kelemahan terbesar pada metode pendeteksian virus berbasis *signature* adalah tidak dapat mendeteksi virus-virus baru yang belum ada di basis data *signature*. Mengetahui kelemahan ini, pembuat virus modern sering kali mengubah *signature* virus setiap kali mereplikasi dirinya.

2. Pendeteksian Virus Berbasis *Heuristic*

   Pendeteksian dengan cara ini bertujuan untuk mendeteksi virus-virus komputer baru dengan cara mengidentifikasi berkas dengan karakteristik yang mencurigakan tanpa harus cocok dengan *signature*. Sebagai contoh, antivirus akan menganggap adanya instruksi-instruksi komputer yang jarang digunakan atau instruksi yang tidak menghasilkan apapun sebagai suatu karakteristik virus. Antivirus juga mungkin mengemulasikan apa yang akan dilakukan program tersebut jika dijalankan, tentunya tanpa memperlambat kinerja sistem komputer. Satu karakteristik mencurigakan tidaklah cukup untuk mengatakan berkas tersebut adalah virus atau tidak. Biasanya antivirus akan mengumpulkan karakteristik virus sampai jumlah tertentu, barulah antivirus dapat menentukan berkas tersebut merupakan virus. Kelemahan terbesar dari pendeteksian virus berbasis *heuristic* adalah dapat salah mendeteksi berkas sebagai virus, padahal bukan. Kejadian ini sering disebut sebagai *false alarm.*

3. Pendeteksian Virus Berdasarkan Perilaku Program

   Dengan metode ini, antivirus mengamati bagaimana program berjalan. Antivirus mencoba mengidentifikasi virus dari perilaku-perilaku yang dikategorikan mencurigakan. Perilaku-perilaku tersebut contohnya adalah memodifikasi berkas-berkas sistem di komputer pengguna, mengintai setiap pengguna menekan tombol di *keyboard/mouse*, ataupun mencoba menjalankan instruksi-instruksi yang mencurigakan seperti mereplikasi dirinya. Sama seperti pendeteksian virus berbasis *heuristic*, suatu perilaku mencurigakan yang dijalankan program tidak dapat langsung ditentukan program tersebut merupakan sebuah virus komputer atau tidak. Akan tetapi, jika ada beberapa perilaku program mencurigakan yang dijalankan dapat menjadi indikasi bahwa program tersebut merupakan virus komputer.

4. Pendeteksian Virus Berbasis *Cloud*

   Pendeteksian dengan metode ini mengidentifikasi virus komputer dengan cara mengumpulkan data-data dari komputer pengguna dan mengirimkannya ke *cloud* untuk dianalisis. Biasanya cara ini dilakukan dengan mengirimkan informasi-informasi yang terkait dengan berkas tersebut dan hal apa saja yang dilakukan jika berkas tersebut merupakan program. Kemudian *cloud* akan menganalisis bukan hanya dari informasi

komputer tersebut saja, melainkan juga dari komputer-komputer lainnya yang terhubung dengan *cloud*. Karena hampir semua analisis dilakukan di *cloud* dan bukan di komputer pengguna, antivirus dengan metode ini ringan dijalankan di komputer. Kelemahan dari metode ini adalah jika komputer pengguna tidak terkoneksi dengan internet, pendeteksian virusnya tidak sehebat ketika terkoneksi internet karena basis datanya tidak terbarui.

III. ANALISIS

A. Implementasi BLAKE2

Source code dari BLAKE2 dapat diunduh secara gratis dan siap untuk dipakai, sehingga kita tidak perlu menulis ulang kode untuk algoritme BLAKE2[11]. Berikut ini adalah notasi algoritmik[12] sederhana untuk implementasi BLAKE2 dalam mendeteksi virus berbasis *signature*:

```
procedure cekSignature (input filename: string,
output hasil: boolean, output virusname: string)
{ I.S. Sembarang }
{ F.S. jika input adalah virus maka hasil bernilai
true dan virusname bernilai nama virusnya, jika
bukan virus hasil bernilai false dan virusname
bernilai string kosong }

Kamus Lokal
result: boolean
resultname: string

Algoritme
checkDatabase(blake2_stream(filename), result,
resultname)
```

```
Program avSangatSederhana
{ Implementasi pendeteksian virus berbasis
signature dengan fungsi hash BLAKE2 }

Kamus Lokal
result: boolean
resultname: string
namafolder: string
i, n: integer

Algoritme
inisialisasiDatabase()
n ← 0
input(namafolder)
i traversal [1..JumlahFile(namafolder)]
  namafile ← namafolder[i]
  cekSignature(namafile, result, resultname)
  if (result) then
    n ← n+1
    output(n, namafile, resultname)
```

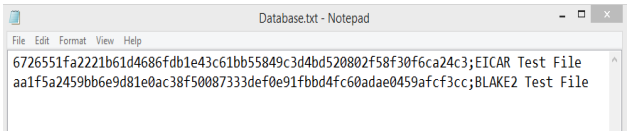
B. Hasil Pendeteksian Virus

Pada saat pengujian, penulis ingin mendeteksi dua buah virus komputer, yaitu berkas yang masing-masing berisi *string*:

Isi Berkas	Nilai *Hash*
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*	6726551fa2221b61d4686fdb1e43c61bb55849c3d4bd520802f58f30f6ca24c3
Ini adalah file virus.	aa1f5a2459bb6e9d81e0ac38f50087333def0e91fbbd4fc60adae0459afcf3cc

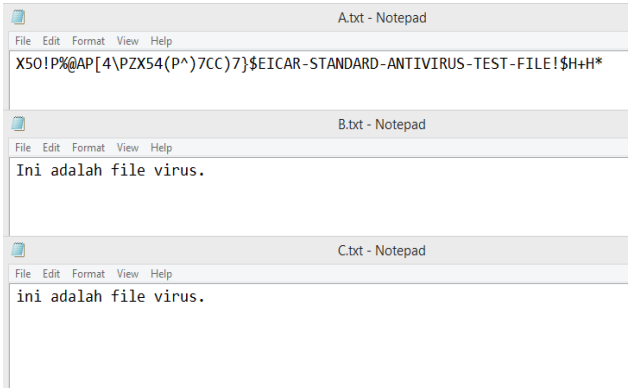
Tabel 3.2.1. Isi Berkas dan Nilai Hash dari Virus

Kemudian penulis mendefinisikan ke dalam basis data nilai *hash* BLAKE2 dua buah berkas tadi dengan menamakan virus tersebut EICAR Test File dan BLAKE2 Test File untuk dideteksi sebagai virus komputer. Basis data ini disimpan dalam bentuk berkas eksternal bernama "Database.txt".



Gambar 3.2.1. Isi Basis Data

Lalu penulis menyediakan tiga buah berkas yang disimpan dalam satu folder yang sama, yaitu berkas bernama "A.txt", "B.txt", dan "C.txt". Dua di antara berkas tersebut yaitu "A.txt" dan "B.txt" merupakan dua virus komputer yang telah didefinisikan sebelumnya pada basis data.

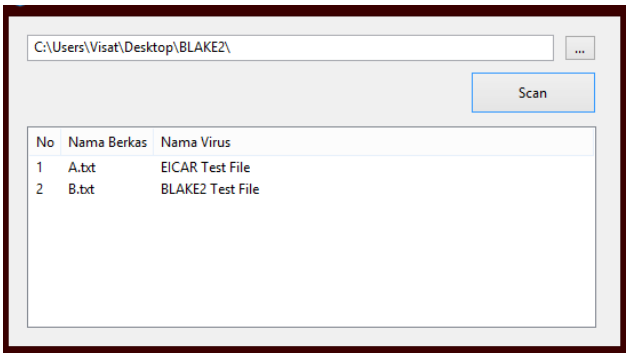


Gambar 3.2.2. Tiga Berkas untuk Diuji

Walaupun sekilas terlihat sangat mirip dengan berkas "B.txt", berkas "C.txt" merupakan berkas yang berbeda. Perbedaan dalam berkas-berkas tersebut terletak pada *byte* pertamanya. Dengan hanya perbedaan satu *byte*, nilai *hash* yang dihasilkan oleh fungsi *hash* BLAKE2 sangat jauh berbeda.

Isi Berkas	Nilai *Hash*
Ini adalah file virus.	aa1f5a2459bb6e9d81e0ac38f50087333def0e91fbbd4fc60adae0459afcf3cc
ini adalah file virus.	4400aa00bc81484ea80b328bf86f2677ab681ef7987fde646127a45fec5592f0

Tabel 3.2.2. Perbedaan Nilai *Hash* pada Isi Berkas yang Mirip



Gambar 3.2.3. Hasil Pendeteksian Virus

Seperti hasil yang diharapkan, setelah selesai dijalankan program dapat mendeteksi semua virus yang telah didefinisikan sebelumnya pada basis data, yaitu berkas "A.txt" dan "B.txt".

C. Kelebihan Pendeteksian Virus dengan BLAKE2

Kelebihan paling menonjol dari pendeteksian virus berbasis *signature* dengan BLAKE2 dibandingkan dengan fungsi *hash* yang lainnya terletak pada kecepatan. Referensi [9] telah menyebutkan bahwa performa kecepatan fungsi *hash* BLAKE2 adalah yang terbaik dari fungsi-fungsi *hash* yang ada. Selain itu, kelebihan lain yang dimiliki fungsi *hash* BLAKE2 adalah belum ditemukannya kelemahan terhadap *collision*, sehingga belum dimungkinkan dua berkas yang berbeda menghasilkan nilai *hash* yang sama.

D. Kekurangan Pendeteksian Virus dengan BLAKE2

Kekurangan dari pendeteksian virus dengan BLAKE2 sama dengan kekurangan dari metode pendeteksian virus berbasis *signature* lainnya, yaitu tidak dapat mendeteksi virus yang belum didefinisikan pada basis data. Kelemahan ini sangat mungkin dimaanfatkan para pembuat virus untuk menghindar dari deteksi dengan cara mengubah potongan *byte* program seperti *icon* program. Alternatif lainnya adalah dengan cara menambah instruksi-instruksi yang jika ditambahkan tidak menimbulkan efek apa-apa, seperti instruksi *nop*. Dengan cara tersebut, virus akan mempunyai nilai *hash* yang berbeda karena ada potongan *byte* yang berbeda sehingga virus lolos dari deteksi. Untuk mengatasi kelemahan ini, keakuratan pendeteksian virus dapat ditingkatkan dengan cara menggabungkan metode ini dengan metode seperti pendeteksian virus berbasis *heuristic*, dan metode pendeteksian virus lainnya.

IV. Ucapan Terima Kasih

Penulis mengucapkan puji dan syukur kepada Allah SWT untuk segala rahmat-Nya sehingga penulis bisa menyelesaikan makalah ini. Penulis juga mengucapkan terima kasih kepada dosen pengajar Matematika Diskrit, Ibu Harlili dan Bapak Rinaldi Munir, untuk segala pengajaran yang telah diberikan mengenai Matematika Diskrit, khususnya mengenai Teori Bilangan yang menjadi salah satu dasar penting dalam penulisan makalah ini.

Referensi

[1] Aycock, John. 2006. *Computer Viruses and Malware*. New York: Springer. hal. 14.

[2] Carroll, Ovie; Krotoski, Mark. 2014. *Using 'Digital Fingerprints' (or Hash Values) for Investigations and Cases Involving Electronic Evidence*. 62 United States Attorneys' Bulletin 44-82 (May 2014)

[3] Munir, Rinaldi. 2013. *Fungsi Hash*. Bahan Kuliah IF3058 Kriptografi. Bandung: Program Studi Teknik Informatika Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung.

[4] Masykuri, Luthfi Hamid. 2013. *Penggunaan Fungsi Hash MD5 dalam Enkripsi PHP Session dan Cookies*. Makalah IF2120 Matematika Diskrit – Sem. I Tahun 2012/2013. Bandung: Program Studi Teknik Informatika Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung.

[5] http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/submissions_rnd3.html diakses 7 Desember 2014 pukul 14.33 WIB

[6] http://csrc.nist.gov/groups/ST/hash/index.html diakses 7 Desember 2014 pukul 14.39 WIB

[7] Guo, Jian; Karpman, Pierre; Nikoli, Ivica; Wang, Lei; Wu, Shuang. 2014. *Analysis of BLAKE2*. Nanyang Technological University, Singapore & Ecole normale sup´erieure de Rennes, France.

[8] http://www.paritynews.com/2012/12/25/531/blake2-an-alternative-to-sha-3-sha-2-and-md5-announced/ diakses 7 Desember 2014 pukul 15.09 WIB

[9] Aumasson, Jean-Philippe; Neves, Samuel; Wilcox-O'Hearn, Zooko; Winnerlein, Christian. 2013. *BLAKE2: simpler, smaller, fast as MD5*.

[10] http://searchsecurity.techtarget.com/tip/How-antivirus-software-works-Virus-detection-techniques diakses 7 Desember 2014 pukul 16.28 WIB

[11] https://blake2.net diakses 7 Desember 2014 pukul 19.13 WIB

[12] Liem, Inggriani. 2007. *Diktat Kuliah Dasar Pemrograman*. Bandung: Program Studi Teknik Informatika Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung.

[13] http://securelist.com/analysis/quarterly-malware-reports/67637/it-threat-evolution-q3-2014/ diakses 8 Desember 2014 pukul 08.43 WIB

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, 8 Desember 2014

Muhamad Visat Sutarno
13513037