HenHash

Fungsi Hash berdasarkan Henon Map

Wiwit Rifa'i (13513073)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Jalan Ganesha 10-12, Bandung 40132, Indonesia
wiwitrifai@gmail.com

***Abstrak*—Fungsi hash banyak diaplikasikan dalam berbagai hal terutama dalam bidang kriptografi. Dalam perkembangannya, sudah terdapat algoritma hash yang banyak digunakan seperti MD2, MD4, MD5, SHA-1, dan SHA-3. Dalam makalah ini, penulis membahas sebuah desain algoritma hash yang menggunakan pendekatan sistem chaos. Chaotic map yang digunakan adalah Henon's map. Dari hasil percobaan dan analisis yang dilakukan, algoritma yang diusulkan memberikan hasil yang baik dan cukup aman.**

Kata Kunci—Hash; Message digest; Chaos; Henon Map; Kolisi

I. PENDAHULUAN

Fungsi hash merupakan salah satu bagian yang cukup penting dalam bidang kriptografi. Fungsi hash biasanya digunakan dalam algoritma lain yang berhubungan dengan kriptografi seperti algoritma penandatanganan digital. Fungsi hash dipakai untuk menjaga integritas dari suatu data. Selain digunakan dalam bidang kriptografi, fungsi hash juga banyak digunakan untuk berbagai macam operasi dalam komputer seperti *hash tables* yang dapat secara cepat menentukan lokasi dari sebuah *record* berdasarkan *query* tertentu, atau menemukan data yang terduplikasi dengan mudah.

Terdapat bermacam-macam algoritma yang sudah tersedia sebagai fungsi hash. Beberapa diantaranya adalah fungsi hash MD2, MD4, MD5, Secure Hash Algorithm (SHA), Snefru, dan N-hash.

Hasil keluaran dari fungsi hash biasanya berupa string dengan panjang tetap yang terlihat seperti acak. Ketika masukan diubah sedikit maka fungsi hash biasanya akan menghasilkan keluaran dengan perbedaan yang cukup signifikan jika dibandingkan dengan keluaran sebelumnya. Untuk menghasilkan hasil yang cenderung acak dan sensitif terhadap masukan, sistem chaos cukup cocok untuk digunakan dalam hal tersebut. Sistem chaos memiliki sifat sensitivitas yang cukup baik terhadap kondisi awal. Oleh karena itu, dalam makalah ini penulis mencoba melakukan pendekatan fungsi hash berdasarkan sistem chaos. Terdapat beberapa sistem chaos yang bisa digunakan, namun dalam makalah ini sistem chaos yang akan digunakan adalah Henon Map.

II. DASAR TEORI

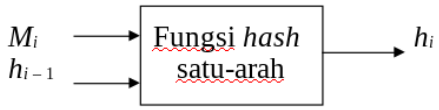
A. *Fungsi Hash*

Fungsi hash adalah sebuah fungsi yang menerima sejumlah pesan dengan panjang yang bisa bervariasi dan mengkonversikan pesan tersebut menjadi sebuah string dengan panjang tetap (umumnya lebih kecil dari ukuran awal). Hasil keluaran dari fungsi hash biasanya disebut dengan *message digest*. Fungsi hash bisa digunakan untuk berbagai hal termasuk dalam bidang kriptografi. Dalam kriptografi, fungsi hash biasanya digunakan untuk menjaga integritas dari suatu data. Sehingga untuk menentukan bahwa suatu pesan pernah diubah atau belum, cukup dengan membandingkan apakah nilai hash dari pesan tersebut sama dengan nilai hash dari pesan yang seharusnya. Dan karena ukuran hasil hash relatif kecil maka membandingakan dua nilai jauh lebih mudah dibandingkan membandingkan pesan secara langsung.

$$h = H(m) \tag{1}$$

Fungsi hash satu-arah adalah sebuah fungsi hash yang bekerja dalam satu satu arah yaitu pesan yang sudah dikonversi menjadi *message digest* tidak dapat dikonversikan lagi menjadi pesan asli hanya dari *message digest* tersebut. Karena *message digest* memiliki ukuran tetap sedangkan pesan asli memiliki ukuran sembarang, maka proses hashing biasanya dilakukan dengan membagi pesan tersebut dalam blok-blok terlebih dahulu dan setiap blok pesan diproses secara bertahap dengan skema seperti pada gambar 1 dan menggunakan persamaan yang berbentuk seperti persamaan 2.

$$h_{(n+1)} = H(M_n, h_n) \tag{2}$$



Gambar 1. Skema Fungsi hash satu-arah

B. *Chaos*

Sistem Chaos adalah sebuah sistem dinamis nirlanjar yang menunjukan fenomena chaos. Sistem chaos memiliki kepekaan

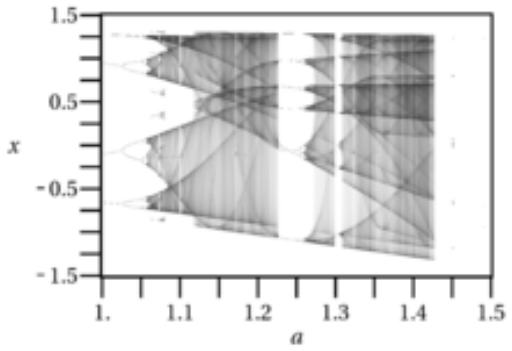
terhadap kondisi awal. Sehingga kepekaan terhadap kondisi awal tersebut membuat chaos terlihat muncul secara acak. Meskipun pada dasarnya sistem chaos memiliki sifat yang deterministik dan tidak memiliki parameter acak. Sistem Chaos biasanya dipakai sebagai Pseudo Random Number Generator (PRNG) dalam bidang kriptografi.

Beberapa contoh sistem chaos adalah Logistic Map, Henon Map, dan Arnold's Cat Map. Henon Map adalah salah satu chaotic map yang diusulkan oleh M. Henon (1976). Henon Map merupakan bentuk sederhana dari Sistem Lorenz namun memiliki sifat yang menyerupai Sistem Lorenz. Persamaan yang digunakan dalam Henon Map adalah sebagai berikut.

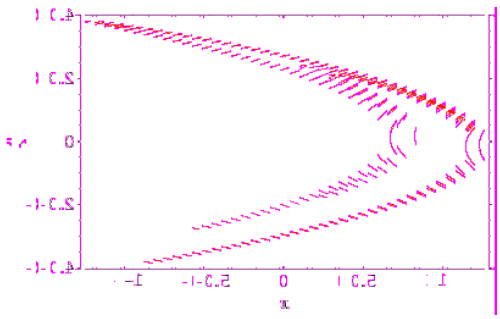
$$x_{(n+1)} = 1 - a x_n^2 + y_n \quad (3)$$

$$y_{(n+1)} = b\, x_n \quad (4)$$

Gambar 2 menunjukkan Bifurcation Diagram untuk Henon Map dengan b = 0.3. Semakin gelap suatu titik di gambar tersebut menunjukkan peluang yang lebih tinggi agar memperoleh nilai x tersebut untuk nilai a yang bersesuaian. Untuk beberapa paramater a dan b, Henon Map memiliki sejumlah *Strange Attractor*. Yaitu untuk sejumlah titik awal, titik-titik yang dihasilkan selanjutnya akan konvergen terhadap suatu *Attractor* dan sedangkan titik-titik awal lain akan menjauhi *Attractor* dan menuju tak-hingga. Gambar 3 menunjukkan titik-titik yang menjadi *Strange Attractor* untuk Henon Map dengan parameter a = 1.4 dan b = 0.3.



Gambar 2. Orbit Diagram dari Henon Map dengan parameter b = 0.3.



Gambar 3. *Strange Attractor* dari Henon Map dengan parameter a = 1.4 dan b = 0.3

III. Rancangan Algoritma

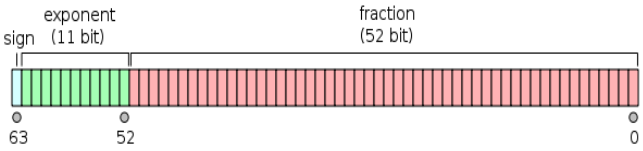
Algoritma HenHash menerima masukan sebuah pesan dengan panjang sembarang dan akan menghasilkan sebuah message digest dengan ukuran 96-bit. Algoritma ini menggunakan double floating point (64-bit) untuk mengoperasikan Henon Map. Parameter yang dipakai dalam Henon Map adalah parameter klasik yang dianggap cukup baik yaitu a = 1.4 dan b = 0.3. Algoritma ini memproses pesan dengan secara bertahap dan setiap blok 48-bit diproses secara bergantian.

Langkah-langkah yang diperlukan dalam algoritma ini adalah sebagai berikut.

1. Pesan masukan ditambahkan 32-bit yang berisi panjang pesan tersebut, kemudian ditambahkan padding bit 0 sebelum 32-bit terakhir yang berisi panjang pesan tersebut sampai ukuran sekarang merupakan kelipatan 48. Pesan tersebut kemudian akan dibagi dalam blok-blok berukuran 48-bit untuk diproses secara berurutan.

2. Inisialisasai nilai x dan y dengan suatu nilai awal yang tetap misalnya x = 0.631354477 dan y = 0.189406343.

3. Untuk masing-masing blok pesan, blok tersebut akan di XOR dengan 48-bit pertama dari bit-bit fraction dari representasi biner IEEE 754 double-precision binary floating-point variabel x. Operasi XOR hanya dilakukan pada bit-bit fraction untuk menghindari perubahaan nilai x yang terlalu signifikan yang berpotensi mengubah nilai x menjadi tak terhingga atau NaN (Not a Number). Bit-bit fraction yaitu bit ke-0 sampai bit ke-51 dari 64-bit double floating point seperti pada gambar 4.



Gambar 4. Representasi *double floating point*

4. Kemudian dengan Henon Map dihitung nilai titik selanjutnya dengan rumus seperti pada persamaan (3) dan (4). Untuk meningkatkan keacakan, perhitungan dilakukan dalam beberapa iterasi.

5. Langkah 3 dan 4 dilakukan untuk setiap blok pesan dari blok pertama hingga blok terakhir.

6. Kemudian langkah 3 dan 4 dilakukan lagi untuk blok-blok pesan tersebut tetapi dengan urutan berkebalikan yaitu dari blok terakhir hingga blok pertama. Hal ini dilakukan agar meningkatkan tinggat keamanan dari fungsi hash.

7. Kemudian message digest diambil dari 48-bit pertama dari bit-bit fraction x dan 48-bit pertama dari bit-bit fraction y. Sehingga diperoleh message digest total berukuran 96-bit.

IV. Percobaan Dan Analisis

Analisis dilakukan dengan melakukan sejumlah percobaan terhadap Algoritma HenHash yang telah diimplementasikan dalam bahasa C++. Percobaan dilakukan dengan komputer dengan spesifikasi CPU Intel Core i7-4720HQ 2.60GHz, RAM 4 GB, dan beroperasi pada sistem operasi Fedora 25. Berikut ini adalah hasil analisis dari percobaan yang telah dilakukan.

A. Analisis Performansi

Tabel 1 menunjukkan hasil performansi dari algoritma HenHash yang dicatat dari sejumlah percobaan dengan beberapa ukuran masukan yang berbeda. Dari hasil tersebut dapat dilihat bahwa algoritma ini berjalan dengan cukup cepat dan efisien. Dapat dilihat juga bahwa waktu eksekusi berbanding lurus dengan ukuran file.

TABLE I. Waktu eksekusi untuk ukuran file yang berbeda

Ukuran (kilo byte)	Waktu (ms)
4	0.1530
16	0.5580
64	1.7830
256	7.3180
1024	31.6430
4096	117.4200
16384	450.6870
65536	1792.0730
262144	7193.8210

B. Analisis Keamanan

Analisis Keamanan dievaluasi berdasarkan analisis sensitivitas fungsi hash terhadap perubahan masukan, analisis statistik dari diffusion dan confusion, dan analisis kemungkinan terjadinya kolisi.

1) Analisis Sensitivitas

Percobaan dilakukan dengan menggunakan sejumlah pesan berbeda dan pesan yang mirip namun dengan beberapa perubahan kecil. Hasil percobaan dapat dilihat pada tabel 2.

TABLE II. Beberapa hasil *message digest* dengan sejumlah pesan berbeda.

Pesan Masukan	*Message digest*
ab	`9587B3D8AC2779E461762F46`
ac	`C6D2DFD1AC276BC9A27A2F46`
aba	`C668C5FC18CCC29B9E8A106D`
abc	`18D794DC05CCE67BE0381A28`
abcd	`29F48A64E9813B9BB7DED1EC`
Hello world!!	`A5C298776A08ED3F019BB16C`
Hello world!!!	`4148EAE2425EB317A33EDD1A`
Hello World!!	`9033EF2AD40759952054626C`
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789~!@#$%^&*()_+`-=;':",./<>?	`5157EE9FDE01ADEF0C1011B2`
XbcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789~!@#$%^&*()_+`-=;':",./<>?	`902021A689C3921B12943CD8`
bcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789~!@#$%^&*()_+`-=;':",./<>?	`AD01682976861B9D79226D64`

Dari tabel 2 tersebut dapat dilihat bahwa untuk pesan yang cukup panjang, perubahan kecil pada pesan tersebut akan membuat perubahan yang cukup signifikan pada message digest. Namun untuk perubahan kecil pada pesan yang terlalu pendek tidak menghasilkan perubahan yang terlalu signifikan, meskipun begitu masih tetap ada perubahaan yang cukup banyak pada message digest.

2) Analisis Difusi dan Konfusi

Difusi dan konfusi adalah prinsip umum yang digunakan untuk mendesain fungsi hash. Percobaan dilakukan dengan mengubah satu bit secara acak dari pesan. Kemudian dihitung berapa banyaknya perubahan bit antara nilai hash dari pesan sebelum dan sesudah diubah. Percobaan dilakukan sebanyak N kali. Percobaan dilakukan dengan sejumlah nilai N berbeda yaitu 128, 256, 512, 1024, dan 2048.

Misalkan C_n menyatakan banyaknya perubahaan bit pada message digest yang terjadi dalam satu kali perngubahaan pesan. Untuk setiap percobaan dilakukan perhitungan beberapa nilai statistik berikut.

Rata-rata perubahaan bit $$\mu = \frac{1}{N}\sum_{k=1}^{N} C_k \quad (5)$$

Rata-rata peluang perubahan bit $$p = \frac{\mu}{ukuran\ pesan} \quad (6)$$

Simpangan baku sampel $$s = \sqrt{\frac{1}{N-1}\sum_{k=1}^{N}(C_k - \mu)^2} \quad (7)$$

Percobaan dilakukan dengan beberapa pesan awal yang memiliki ukuran yang bermacam-macam. Percobaan pertama dilakukan dengan pesan awal berukuran 32-bit berupa “abcd”. Hasil dari percobaan pertama adalah seperti pada tabel 3.

TABLE III. HASIL STATISTIK PERUBAHAAN BIT PERCOBAAN 1

N	μ	p	s
128	45.757812	47.029622%	6.385379
256	45.886719	47.798665%	6.036398
512	45.255859	47.141520%	6.217122
1024	45.289062	47.176107%	6.555757
2048	45.383301	47.274272%	6.396220

Percobaan kedua dilakukan dengan pesan awal dengan ukuran yang lebih panjang dari percobaan pertama yaitu “abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789~!@#$%^&*()_+`-=;':",./<>?” dengan ukuran 88 karakter (704 bit). Hasil dari percobaan kedua adalah seperti pada tabel 4.

TABLE IV. HASIL STATISTIK PERUBAHAAN BIT PERCOBAAN 2

N	μ	p	s
128	47.718750	49.707031%	5.014840
256	48.191406	50.199382%	6.341158
512	47.681641	49.668376%	5.836840
1024	46.273438	48.201497%	10.205773
2048	46.514648	48.452759%	9.250505

Percobaan ketiga dilakukan dengan pesan awal dengan ukuran yang lebih dari percobaan pertama dan kedua yaitu dengan pesan dengan panjang 1312 karakter (10496 bit) yaitu

> Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?

Dan hasil dari percobaan ketiga adalah seperti pada tabel 5.

TABLE V. HASIL STATISTIK PERUBAHAAN BIT PERCOBAAN 2

N	μ	p	s
128	47.320312	49.291992%	5.298839
256	45.597656	47.497559%	11.255945
512	45.222656	47.106934%	11.567772
1024	44.048828	45.884196%	13.688818
2048	40.161621	41.835022%	18.245590

Berdasarkan hasil percobaan tersebut, dapat dilihat bahwa peluang perubahan bit pada *message digest* yang disebabkan oleh perubahan satu bit pada pesan asli memiliki nilai berkisar antara 41% hingga 50%. Sehingga bisa dibilang bahwa algoritma memiliki difusi dan konfusi yang cukup baik.

3) Analisis Kolisi

Salah satu masalah yang penting dalam fungsi hash adalah terjadinya kolisi. Kolisi adalah keadaan dimana 2 buah masukan berbeda menghasilkan satu *message digest* yang sama. Banyaknya *message digest* yang mungkin dihasilkan oleh Algoritma HenHash ini adalah 2^{96} karena ukuran *message digest* yang selalu 96-bit. Sehingga jika kita memiliki input sebanyak lebih dari 2^{96} buah pesan berbeda pasti akan terjadi kolisi. Sehingga kolisi bisa saja terjadi meskipun peluangnya relatif kecil.

Serangan yang biasanya terjadi pada fungsi hash adalah mencoba mencari 2 pesan berbeda yang saling kolisi. Pada operasi Henon Map, kita bisa dengan mudah untuk membalikkan titik menjadi titik sebelumnya dengan operasi aljabar sederhana. Sehingga jika operasi hash hanya dilakukan dari blok pesan pertama hingga terakhir saja (tanpa melanjutkan hash dengan arah sebaliknya dari blok terakhir sampai blok pertama) maka dengan mudah kita bisa mendapatkan pesan yang saling kolisi dengan cara membalikkan operasi pada Henon Map secara terus menerus hingga didapatkan titik awal. Namun hal itu sudah diatasi dengan melanjutkan operasi hash tersebut bukan hanya dari blok pertama sampai blok terakhir, tapi juga dilanjutkan dengan memproses ulang blok-blok tersebut dengan arah berkebalikan. Sehingga serangan mencari 2 pesan berbeda yang saling kolisi menjadi sulit untuk dilakukan. Selain itu sebelum dilakukan perhitungan hash pesan masukan ditambahkan dengan informasi panjang bit dari pesan tersebut sehingga kolisi menjadi lebih sulit untuk ditemukan.

V. SIMPULAN DAN SARAN

Algoritma Fungsi Hash HenHash yang dikembangkan memiliki performa yang cukup baik. Selain itu dari segi keamanan algoritma ini juga sudah cukup baik. Perubahan kecil pada pesan bisa membuat perubahan yang cukup signifikan pada message digest yang dihasilkan. Serangan mencari 2 pesan berbeda yang memiliki nilai hash sama juga cukup sulit dilakukan terhadap algoritma ini. Meskipun ada kemungkinan bahwa suatu saat algoritma ini memiliki celah yang bisa digunakan untuk mendapatkan 2 pesan bebeda yang

memiliki nilai hash yang sama. Untuk meningkatkan faktor keamanan, pengembangan dari algoritma ini bisa dilakukan dengan mengombinasikan pendekatan ini dengan algoritma fungsi hash lain dan/atau dengan memperbesar ukuran dari message digest yang dihasilkan.

UCAPAN TERIMA KASIH

Penulis ingin mengucapkan rasa syukur atas rahmat dan karunia Allah SWT sehingga makalah ini bisa terselesaikan. Penulis juga ingin mengucapkan terima kasih kepada Bapak Dr. Ir. Rinaldi Munir, M.T. selaku dosen pengampu IF4020 Kriptografi yang telah membimbing penulis dalam mempelajari kriptografi dalam satu semester. Dan tak lupa pula penulis ingin mengucapkan terima kasih kepada keluarga dan teman seperjuangan yang telah memberikan dorongan selama ini.

DAFTAR PUSTAKA

[1] Munir, Rinaldi. "Diktat Kuliah IF5054 Kriptografi", Departemen Teknik Informatika ITB ,2005.

[2] Weisstein, Eric W. "Chaos." From MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/Chaos.html

[3] Hénon, M. "A Two-Dimensional Mapping with a Strange Attractor." Comm. Math. Phys. 50, 69-77, 1976.

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, 21 Desember 2016

Wiwit Rifa'i