Implementasi Hash dengan Memanfaatkan Bogdanov Map

Radiyya Dwisaputra / 13515023
Program Studi Teknik Informatika
Sekolah Teknik Elektronika dan Informatika
Institut Teknologi Bandung
Jln. Ganesha 10 Bandung 40132, Indonesia
radiyyasaputra@gmail.com

Abstrak — Saat ini penggunaan teknologi informasi sangatlah luas dan tentu membutuhkan keamanan yang cukup baik. Salah satu algoritma pengamanan ialah hash. Hash banyak digunakan dalam mengamankan sebuah informasi dengan mengecek kevalidan informasi tersebut. Saat ini hash yang ada sudah cukup terkenal diantaranya ialah MD5, SHA-1, dan lain lain. Bogdanov map merupakan salah satu contoh *chaotic map* yang dapat digunakan untuk mengacak sebuah nilai tertentu karena menghasilkan *chaos behavior*. Bogdanov map dimanfatkan dalam implementasi fungsi hash ini untuk membuat fungsi hash acak dan aman dari berbagai serangan.

Kata kunci — Hash, Bogdanov Map

I. PENDAHULUAN

Saat ini dalam berbagai kebutuhan sehari hari hampir seluruh kegiatan berhubungan dengan sebuah teknologi informasi. Hal ini ditunjukkan dalam berbagai bidang seperti dalam pembayaran yang telah banyak terdapat alat pembayaran seperti gopay, ovo, dan lain lain. Bidang lain yang juga telah berhubungan dengan teknologi informasi ialah sosial media seperti facebook, twitter, dan lain lain. Bidang yang berhubungan dengan pemerintah pun juga telah berhubungan teknologi informasi.

Penggunaan teknologi informasi yang sudah sangat luas ini dikarenakan kemudahan yang dirasakan. Hal ini tentu berdampak pada teknologi informasi yang haruslah aman dari serangan serangan yang ada terutama serangan secara digital. Salah satu upaya yang dapat dilakukan ialah dengan menggunakan kriptografi untuk pengamanan sistem. Kriptografi merupakan sebuah kakas yang penting pada ilmu komputer untuk mengamankan pesan. Kriptografi sendiri terdiri dari 2 kata yakni cryptos dan graphein yang berarti "secret writing". Kata aman dalam kriptografi sendiri terdiri dari 4 aspek yaitu *confidentiality* (kerahasiaan pesan), *data integrity* (keaslian pesan), *authentication (*keaslian pengiriman dan penerima pesan), *non-repudiation* (anti penyangkalan). Dalam kriptografi sendiri pun juga terdiri dari berbagai macam bagian mulai dari *encryption, digital signature algorithm, hash*, dll. Oleh karena itu penerapan kriptografi pada teknologi informasi sangat penting untuk menjaga data yang ada dan juga kegiatan yang berkaitan dengan teknologi informasi selalu aman. Pada makalah ini, penulis akan membahas lebih lanjut mengenai hash.

Fungsi hash sendiri merupakan sebuah algoritma yang cukup sering dipakai dalam berbagai metode pengamanan pada teknologi informasi. Hash sendiri merupakan sebuah fungsi yang berfungsi untuk mengacak sebuah pesan dan bersifat satu arah. Walaupun demikian sebuah hash yang baik tidak boleh memiliki kolisi (nilai hasil hash yang sama dari dua pesan berbeda) dan juga harus dapat membuat efek perubahan yang besar pada hasil hash apabila terjadi perubahan kecil pada pesan (perubahan beberapa byte pada pesan).

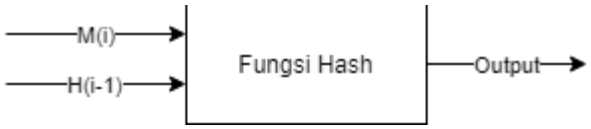
II. DASAR TEORI

A. *Fungsi Hash*

Fungsi Hash merupakan sebuah fungsi yang menerima masukan *string* yang panjangnya sembarang lalu mentransformasikannya menjadi sebuah *string* keluaran yang panjangnya tetap. Fungsi hash bersifat *irreversible* yang berarti hasil hash sebuah pesan *string* tidak dapat dikembalikan lagi menjadi pesan semula. Fungsi hash memiliki beberapa sifat seperti berikut:

1. Fungsi hash dapat diterpakan pada blok data berukuran berapa saja
2. Hasil hash selalu memiliki panjang yang tetap
3. Penghitungan hash tidak rumit
4. Irreversible
5. Tidak ada kolisi (untuk setiap x yang diberikan, tidak terdapat y sehingga H(y) = H(x)

Masukan pada sebuah fungsi hash meruapakn blok pesan dan keluaran dari hash block pesan sebelumnya. Skema fungsi hash dapat digambarkan sebagai berikut:



Gambar 1. Skema Umum Fungsi Hash

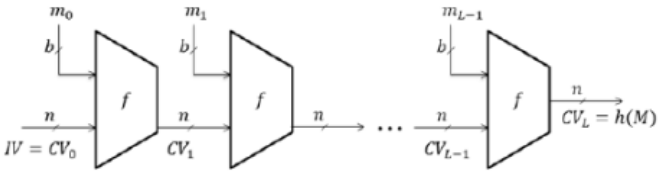
Seperti terlihat pada gambar masukan pada fungsi hash M(i) merupakan blok pesan ke i (saat ini) dan H(i-1) merupakan hasil hash pada blok pesan sebelumnya. Keluaran pada fungsi hash sendiri merupakan hasil pada blok pesan ke

I (saat ini) yang akan digunakan sebgai parameter pada fungsi hash berikutnya. Untuk fungsi hash yang pertama maka nilai hasil hash sebelumnya digantikan dengan sebuah *initial vector*.

Penggunaan fungsi hash dalam mengamankan sebuah teknologi informasi bervariatif diantaranya ialah untuk melakukan pengecekan kevalidan sebuah data/file, penyimpanan password pada basis data. Pengecekan kevalidan sebuah data/file dapat dilakukan dengan hash dikarenakan apabila nilai sebuah data/file berubah walaupun sangat sedikit seperti 1 byte saja akan tetap berdampak pada nilai hash yang berbeda jauh. Oleh karena sifat hash yang seperti itu sangat cocok digunakan dengan cara membandingkan nilai hasil hash antara kedua data/file untuk mengecek apakah kedua data/file tersebut sama. Hash juga dapat digunakan agar password pengguna sebuah teknologi informasi aman pada sistem yaitu dengan menyimpannya dalam bentuk hasil hash. Hal ini dapat membuat password aman karena bersifat irreversible yang membuat apabila basis data teknologi informasi didapatkan termasuk password pengguna oleh pihak yang tidak bertanggung jawab tidak dapat digunakan untuk mengakses akun pengguna. Pengecekan kebenaran password pengguna saat login dapat dilakukan dengan membandingkan hasil hash antara password yang dimasukkan pengguna dengan password yang telah dalam bentuk hasil hash di basis data.

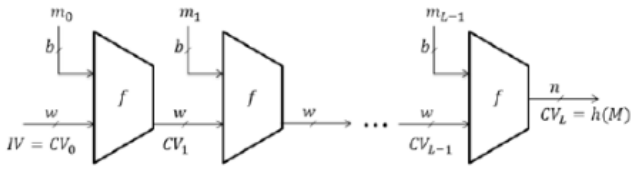
B. Konstruksi Hash

Dalam membuat sebuah hash terdapat beberapa konstruksi yang dapat digunakan untuk saat ini terdapat empat konstruksi yang cukup dikenal yaitu kontruksi Merkle Damgard, konstruksi Wide-pipe, konstruksi Sponge Functions, dan konstruksi HAIFA. Keempat konstruksi ini memiliki kelebihan dan kekurangan masing-masing. Berikut merupakan penjelasan singkat dari masing-masing konstruksi.



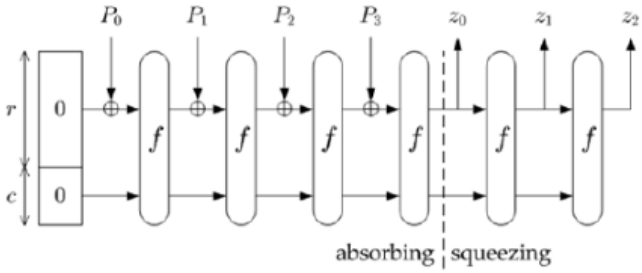
Gambar 2. Merkle Damgard

Merkle Damgard merupakan konstruksi hash berulang yang mana seperti skema umum fungsi hash dengan ukuran input nilai n (initialitation vector pada urutan pertama) sesuai dengan ukuran output yang diharapkan.



Gambar 3. Wide Pipe

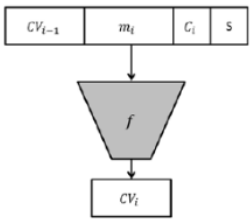
Wide pipe memiliki konstruksi yang sangan mirip dengan merkle damgard hanya saja dilakukan sebuah perubahan pada ukuran input yang dimasukkan yang lebih besar dari ukuran output yang akan diperoleh. Hal ini dilakukan untuk mencegah hash tidak dapat mengalami *length extension attack*.



Gambar 4. Sponge Functions

Sponge Function sedikit berbeda dibandingkan dengan konstruksi sebelumnya yang mana pada konstruksi ini terdapat dua fase utama yaitu *absorbing phase*, dan *squeezing phase*. Fungsi hash terkenal yang menggunakan konstruksi ini adalah SHA-3 atau dikenal juga sebagai nama Keccak.

$$CV_i = C(CV_{i-1}, m_i, C_i, S).$$



Gambar 5. HAIFA

Konstuksi HAIFA (The HAsh Iterative FrAmework) banyak menyelesaikan berbagai masalah mengenai kolisi internal yang dialami oleh konstruksi Merkle Damgard. Konstruksi HAIFA sendiri sangat mirip dengan Wide Pipe.

C. Bogdanov Map

Bogdanov map merupakan sebuah *chaotic map. Chaotic map* merupakan sebuah map yang berguna untuk menunjukan sebuah perilaku acak yang berarti akan mengacak sebuah nilai pada titik tertentu. Bogdanov map sendiri memiliki formula yang sangat sederhana karena tidak ada penghitungan yang kompleks di dalamnya. Bogdanov map sendiri memiliki 2 parameter yakni x dan y. Operasi yang dilakukan pada Bogdanov map merupakan operasi perkalian sederhana. Berikut formula Bogdanov Map :

$$\begin{cases} x_{n+1} = x_n + y_{n+1} \\ y_{n+1} = y_n + \epsilon y_n + k x_n (x_n - 1) + \mu x_n y_n \end{cases}$$

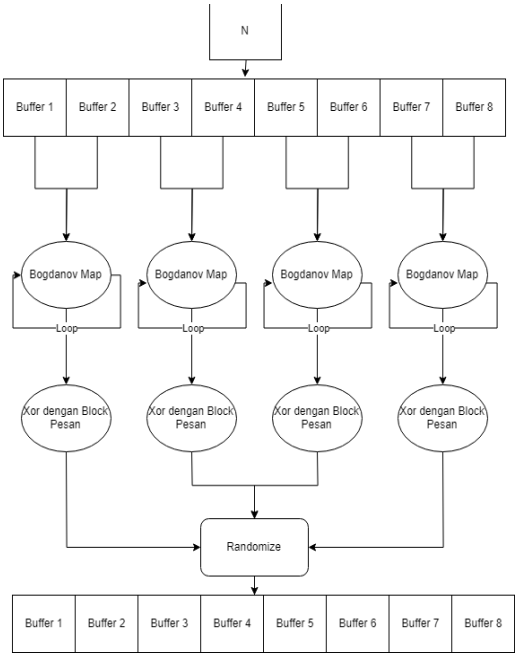
ϵ, μ , dan k merupakan sebuah konstanta yang digunakan untuk melakukan pengacakan nilai x dan y. Map ini berkaitan dengan *Bogdanov-Takens Bifurcation* yang merupakan salah satu contoh bifurkasi pada dua dimensi (x dan y).

III. IMPLEMENTASI

Implementasi fungsi hash yang dengan memanfaatkan Bogdanov map menggunakan konstruksi Merkle Damgard. Konstruksi Merkle Damgard dipilih karena secara teori imlpementasi yang dilakukan dapat lebih sederhana. Fungsi f

hash yang digunakan pun menggunakan beberapa operasi sederhana. Langkah-langkah hash adalah sebagai berikut. Masukan pada fungsi berupa n dan blok pesan dengan initialitation vector sepanjang 64 byte. Panjang initialitation vector disamakan dengan output yang diharapkan. Pesan akan dibagi menjadi beberapa blok dengan panjang 4 byte, apabila pesan memiliki panjang yang tidak habis dibagi 4 maka akan dilakukan padding terlebih dahulu pada pesan. Proses pada fungsi hash ialah dengan membagi n menjadi 8 buffer dengan masing-masing memiliki panjang 8 byte. Nilai x dan y didapatkan dari urutan buffer secara berurutan lalu membagi nya dengan nilai tertentu untuk membuat nilai x dan y masing-masing menjadi kurang dari 1. Hal ini dilakukan karena Bogdanov map akan dilakukan secara berulang sehingga nilai x dan y harus bernilai antara 0 dan 1 agar tidak bernilai sangat kecil nantinya. Banyaknya pengulangan Bogdanov map ditentukan dengan mengubah blok menjadi nilai ASCII, serta nilai x dan y lalu dilakukan modulo menjadi 56 agar proses tidak terlalu lama. Setelah dilakukan Bogdanov map berulang pada nilai x dan y selanjutnya dilakukan xor dengan blok pesan saat ini. Akhirnya akan dilakukan sebuah randomize pada ke 8 buffer untuk menukarkan posisi penukaran ke 8 buffer dilakukan secara konsisten.

Berikut gambaran proses fungsi f hash yang telah dibuat:



Gambar 6 Proses pada fungsi f dalam hash

Proses pada fungsi f ini dilakukan hingga seluruh blok pesan telah di proses. Nilai konstanta yang digunakan dalam Bogdanov Map adalah $\epsilon = 0.01$, $\mu = -0.01$, dan $k = 0.30375$. Pemilihan konstanta didasarkan pada penelitian yang telah dilakukan sebelumnya.

IV. Analisa Dan Eksperimen

Program dieksekusi dengan menggunakan library Python 3 dengan lingkungan Windows 10, processor intel core i5-5200U dan RAM berukuran 8 GB.

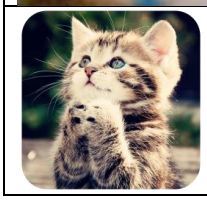
Berikut merupakan contoh masukan dan keluaran dari program.

Tabel 1. Contoh Masukan dan Keluaran Berupa Text

Masukan	Keluaran
Kriptografi	726d124fc74648079b46e 05bd83ef552feff0079e1e 1e2d110b5b18764d10a77
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer a sodales mauris. Nunc sapien nisl, egestas sit amet risus id, molestie sagittis elit. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Phasellus rhoncus diam id ultricies accumsan. Nullam quis porttitor eros. Vivamus in eros nec lectus viverra auctor. Integer vulputate maximus tellus, vitae bibendum enim fermentum sit amet. Nulla id magna eget ex mollis rutrum eu a turpis. Proin sodales facilisis urna in auctor.	124841a283b6629dcb8b3 eadbe7d2b4f329e2dc2c4a b2f6f66a6d96ce385b86e
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer a sodales mauris. Nunc sapien nisl, egestas sit amet risus id, molestie sagittis elit. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Phasellus rhoncus diam id ultricies accumsan. Nullam quis porttitor eros. Vivamus in eros nec lectus viverra auctor. Integer vulputate maximus tellus, vitae bibendum enim fermentum sit amet. Nulla id magna eget ex mollis rutrum eu a turpis. Proin sodales facilisis urna in auctor. Duis lorem enim, tristique pellentesque enim eu, accumsan faucibus ex. Aenean risus sem, rhoncus ut metus et, malesuada sodales enim. Integer justo leo, egestas nec ipsum a, bibendum vestibulum purus. Mauris pretium lacus erat, ornare venenatis neque congue vel. Donec in ante nec sapien vestibulum placerat. Proin vehicula odio ut diam blandit pharetra. Pellentesque quis	f6218dac392019b0d094e 8bebc2ec05b1b18f24013 1c851a85372619250dfd8f

dolor eget tortor malesuada aliquam.	

Tabel 2. Contoh Masukan dan Keluaran Berupa File

Masukan	Keluaran
	6464630b69a9a3b0fe64db3c969c2c91 2323257e96161bfe505052f3d4140e1c
	88d5525f60041625c70d61e5d266be5b 2df840f01014e51ee923d3a8e3d9039d

Pada contoh masukan dan keluaran pada tabel diatas dapat terlihat bahwa hash yang telah dibuat sesuai dengan karakteristik hash yang seharusnya yaitu memiliki panjang output yang selalu sama, dalam hash yang telah dibuat ini output selalu berukuran 512 bit walaupun masukan memiliki ukuran yang berbeda beda.

Selanjutnya akan dilakukan beberapa Analisa pada fungsi hash yang telah dibuat.

1. Perubahan karakter atau byte pada masukan.

Analisa dilakukan dengan menggunakan contoh masukan yang mirip dengan merubah beberapa bagian tertentu

Masukan 1: Kriptografi penting dalam kehidupan
Keluaran:
12e049b1ff3debac681bf05223c513cb37b1660b7203743916 d3124cf92b20bd

Masukan 2: **C**riptografi penting dalam kehidupan
Keluaran:
339ec7823f3b89c0d60cade62a53ac6d3900ea8d781f6d5c16 4e95897521a741

Masukan 3: Kriptografi penting dalam kehidupa**m**
Keluaran:
759060e8ce7f7d846b0472d2718eccf7124109e81014f5b679 54138c1138239e

Masukan 4: Kriptografi p**w**nting dalam kehidupan
Keluaran:
ace31da7a2e644c4549daaac8ed4f3d740f2691c255918c7b0 1a6ab2730147ac

Dari keempat masukan tersebut dapat kita lihat setelah dilakukan perubahan kecil berupa penggantian sebuah karakter dari masukan 1 seperti pada masukan 2, masukan 3, dan masukan 4. Hasil hash yang diperoleh sangat jauh berbeda. Bahkan tidak terdapat pola yang bisa didapatkan.

2. Waktu eksekusi program

Pengukuran waktu eksekusi program akan dilakukan pada beberapa ukuran masukan yaitu 1 KB, 4 KB dan 136 KB.

Tabel 3. Pengukuran Waktu Eksekusi

Ukuran Masukan	Waktu Eksekusi (dalam detik)
1 KB	0.07295799255371094
4 KB	0.11793208122253418
136 KB	3.8218069076538086

Waktu eksekusi fungsi hash sudah cukup cepat namun untuk file – file besar masih cukup membutuhkan waktu. Hal ini masih menjadi kelemahan fungsi hash yang telah dibuat ini karena pada fungsi hash yang telah ada seperti MD5, SHA1, SHA-256 dapat memproses masukan dengan ukuran yang cukup besar kurang dari 1 detik.

3. Ketahanan terhadap serangan bruteforce

Pada fungsi hash yang telah dibuat memiliki panjang output berupa 64 byte atau 512 bit sehingga untuk melakukan serangan secara *bruteforce* pada fungsi hash dibutuhkan sekitar 2^{512} operasi dan apabila sebuah komputer dapat melakukan perhitungan sebanyak 10^8 kali, waktu yang dibutuhkan untuk melakukan serangan ialah sebesar $1{,}134 * 10^{146}$ detik yang setara dengan $3{,}593 * 10^{141}$ tahun. Karena waktu yang dibutuhkan sangat besar maka hamper tidak mungkin untuk seseorang melakukan serangan bruteforce pada fungsi hash ini.

4. Ketahanan terhadap serangan kolisi

Serangan kolisi dapat dilakukan dengan berbagai cara, cara yang akan dianalisa ialah dengan penggunaan *birthday attack*. Dengan menggunakan *birthday attack*, kolisi dapat ditemukan pada $2^{n/2}$ dengan 2^n adalah kompleksitas *bruteforce*. Maka dapat disimpulkan *birthday attack* dapat melakukan serangan apabila telah melakukan operasi selama $1{,}796 * 10^{141}$ tahun yang mana hal ini hampir sangat mustahil untuk dapat dilakukan untuk saat ini.

V. KESIMPULAN DAN SARAN

Bogdanov map sebagai salah satu chaotic map dapat dimanfaatkan dengan baik dalam membuat sebuah fungsi hash karena akan aman dari berbagai serangan dan waktu yang dibutuhkan juga tidak terlalu lama. Walaupun demikian perlu ada nya penelitian lebih lanjut apakah terdapat sebuah algoritma yang dapat membuat waktu eksekusi fungsi menjadi lebih kecil sehingga dapat dengan mudah diterapkan untuk masukan masukan yang memiliki ukuran besar. Selain itu mungkin bisa dilakukan sebuah penelitian lebih lanjut tentang chaotic map yang dapat menghasilkan fungsi hash yang jauh lebih baik daripada fungsi hash yang telah dibuat ini.

VI. UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Tuhan Yang Maha Esa atas segala rahmat dan kehendaknya makalah ini dapat selesai pada waktunya. Tidak lupa, penulis ucapkan terima kasih kepada Bapak Rinaldi Munir selaku dosen kelas IF4020 Kriptografi yang telah mengajarkan banyak ilmu yang bermanfaat dalam mengerjakan makalah ini dan juga terima kasih kepada teman-teman yang telah memberikan dukungan agar makalah ini dapat selesai dengan baik.

REFERENSI

[1] Munir, Rinaldi. Pengantar Kriptografi. Slide Presentasi Kuliah IF4020, Diakses pada 9 Mei 2019.

[2] Munir, Rinaldi. Fungsi Hash. Slide Presentasi Kuliah IF4020, Diakses pada 9 Mei 2019.

[3] Munir, Rinaldi. Algoritma MD5. Slide Presentasi Kuliah IF4020, Diakses pada 9 Mei 2019.

[4] Munir, Rinaldi. Secure Hash Algorithm SHA. Slide Presentasi Kuliah IF4020, Diakses pada 9 Mei 2019.

[5] Denton, Adhami. Modern Hash Function Construction, Diakses pada 9 Mei 2019

[6] https://en.wikipedia.org/wiki/Cryptographic_hash_function, Diakses pada tanggal 9 Mei 2018

[7] https://en.wikipedia.org/wiki/Birthday_attack, Diakses pada tanggal 10 Mei 2018

VII. PERNYATAAN

Dengan ini kami menyatakan bahwa makalah yang kami tulis ini adalah tulisan kami sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 10 Mei 2019

Radiyya Dwisaputra
13515023