Aplikasi Teori Bilangan pada Fungsi Hash dan Enkripsi Dalam Sistem Keamanan Kata Sandi

Kevin Supendi - 13514094[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*13514094@std.stei.itb.ac.id*

***Abstract*—Kata sandi banyak digunakan untuk meningkatkan keamanan dan merahasiakan sesuatu. Agar kata sandi tidak diketahui orang lain, kata sandi tersebut diubah terlebih dahulu sebelum disimpan. Setidaknya ada dua metode yang bisa mengubah kata sandi atau teks, yaitu Hash dan Enkripsi. Keduanya menerapkan Teori Bilangan dalam cara kerjanya.**

***Kata Kunci*—Keamanan, Kata Sandi, Hash, Enkripsi**

I. PENDAHULUAN

Keamanan merupakan kebutuhan bagi setiap orang. Seiiring dengan perkembangan zaman, perkembangan teknologi yang semakin pesat juga mendorong system keamanan untuk ikut berkembang juga. Sistem keamanan dasar yang sering kita jumpai dalam kehidupan sehari-hari adalah verifikasi menggunakan kata sandi. Mulai dari kalangan terpelajar maupun kalangan awam semua sudah tidak asing dengan penggunaan kata sandi. Contohnya seperti layanan media sosial, layanan surat elektronik, memakai kata sandi untuk memasukki akun pengguna.

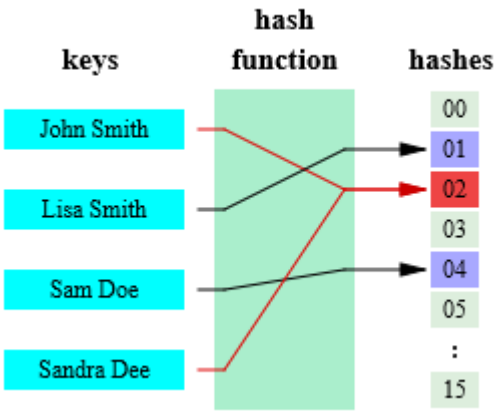
Keamanan kata sandi sangat krusial pengguna. Tentu saja kita tidak mau informasi-informasi pribadi yang kita miliki dilihat atau disebarluaskan orang lain. Terutama untuk informasi penting, pastilah kata sandi yang digunakan harus aman dari serangan digital.

Perusahaan-perusahaan besar harus bisa menyimpan seluruh kata sandi pengguna dengan baik dan menjamin keamanan dari kata sandi tersebut. Untuk menjamin keamanan kata sandi, tentu saja kata sandi tidak disimpan dalam bentuk yang mudah dibaca orang lain. Agar kata sandi tidak mudah diketahui orang lain, biasanya kata sandi diubah terlebih dahulu ke bentuk lain. Cara ini diharapkan mampu melindungi informasi pribadi milik pengguna. Meskipun penyerang dapat mengakses informasi mengenai kata sandi yang disimpan, kata sandi itu telah berubah dan tidak bisa langsung digunakan untuk mengakses informasi milik pengguna. Beberapa cara yang sering digunakan adalah Hash dan Enkripsi.

II. DASAR TEORI

A. Fungsi Hash

Hash adalah algoritma dapat yang mengubah input dengan jumlah besar menjadi output lain dengan jumlah yang tertentu dan jumlah lebih kecil. Input biasa disebut key, sementara output disebut juga hashes. Output yang dihasilkan belum tentu unik. Bisa saja input yang berbeda menghasilkan output yang sama. Hal ini disebut *collision.* Hash sering digunakan untuk mencari suatu data dari teks, *cache*, kriptografi Hash, dan lain-lain.



Gambar 1 Ilustrasi Hash Sumber: http://en.wikipedia.org/wiki/Hash_function

```
int hash(int key){
        int hashvalue;
        hashvalue = key % 5;
        return hashvalue;
}
```

Gambar 2 Ilustrasi Algoritma Hash Sederhana

Pada gambar 2 di atas algoritma tersebut akan menghasilkan nilai output antara 0 sampai 4 untuk input integer berapa pun. Fungsi di atas dapat ditulis dalam bentuk lain

$$f(x) = x \bmod 5$$

Dimana x adalah key dan hasilnya berupa hashvalue. Contoh input output fungsi hash tersebut dapat dilihat di gambar 3.

Input	Output
35	0
1	1
27	2
4	4
24	4
43	3

Gambar 3 Ilustrasi Hasil Hash

Algoritma Hash sederhana pada gambar 2 akan menghasilkan output di atas. Dapat dilihat output bisa saja menghasilkan nilai yang sama. Hal ini tentu dihindari untuk fungsi Hash dalam penyimpanan kata sandi.

Sifat-sifat Hash bermacam-macam, dan sifat ini ditentukan dari tujuan yang mau dicapai dari hash tersebut. Beberapa sifat dari algoritma Hash yang baik adalah :

- Deterministik : Fungsi Hash yang baik akan menghasilkan output yang tetap untuk input yang sama. Misalkan pertama Hash menerima input "abc" dan menghasilkan output 1. Jika berikutnya Hash diinput "abc", maka outputnya juga menghasilkan 1.
- Uniform : Suatu Hash sebaiknya uniform, yang berarti kemungkinan masing-masing output itu muncul sama besar.
- Batas Variabel : Fungsi Hash memiliki jumlah output tertentu, yang tetap atau bisa berubah saat program berjalan. Hash disebut *dynamic hash function* jika batasan nilai dari Hash tersebut bisa berubah.
- Normalisasi Data : Terkadang data yang digunakan sebagai input mempunyai sifat yang boleh diabaikan (tergantung dari kegunaan hash). Untuk mencari data nama orang, huruf besar dan kecil tidak dibedakan. Maka sebelum dimasukkan dalam fungsi Hash data tersebut akan dinormalisasi, diubah menjadi huruf kapital semua.
- Kontinu : Untuk implementasi *search*, fungsi Hash yang kontinu akan menghasilkan output yang sama atau berdekatan, jika data-data yang diinput berdekatan.

Hash yang digunakan untuk keperluan penyimpanan kata sandi adalah Hash kriptografik. Hash ini deterministik dan tidak kontinu. Hash ini juga terkadang menggunakan *salt*, yaitu suatu nilai atau string random yang ditambahkan ke input, berfungsi untuk menurunkan resiko terhadap serangan *brute force*.

Hash kriptografik dirancang untuk tidak bisa diubah kembali menjadi input awal. Contoh dari Hash ini adalah SHA-1 (Secure Hash Algorithm 1). SHA-1 menghasilkan output sebesar 160 bit hash value, yang biasa dituliskan dalam bentuk hexadecimal 40 digit.

B. Enkripsi

Enkripsi adalah proses yang mengubah input menjadi output lain, hanya beberapa orang saja yang bisa menerjemahkan input tersebut. Orang lain yang tidak mempunyai kunci enkripsi tidak akan bisa menerjemahkan data yang telah terenkripsi. Proses untuk menerjemahkan itu disebut dekripsi.

Perbedaannya dengan fungsi Hash adalah, enkripsi dirancang untuk bisa didekripsi, sementara Hash dirancang untuk *one-way*, sehingga nilai input hilang dalam proses Hash, dan tidak dapat dibalikkan.

Proses enkripsi sudah berkembang sejak zaman dahulu, dimana Julius Caesar menyampaikan pesan yang terenkripsi, sehingga pesan tersebut tidak mudah dibaca isinya, seandainya pesan itu jatuh ke tangan lawan. Dengan menganggap alfabet sebagai angka, tiap-tiap huruf diganti menjadi 3 huruf setelahnya. Proses enkripsi seperti ini disebut *shift cipher*.

```
int encrypt(int input, int key){
        int output;
        output = input + key;
        return output;
}
```

Gambar 4 Ilustrasi Enkripsi Sederhana

Ilustrasi di atas menggambarkan fungsi enkripsi, yaitu mengubah nilai input menjadi input+key. Nilai ini bisa dengan mudah didapatkan, yaitu dengan mengurangi nilai output dengan nilai key yang dirahasiakan.

Jenis *cipher* yang lain adalah *block cipher*. Jenis ini mengubah sekumpulan huruf untuk diganti dengan kumpulan huruf yang lain. Suatu Kalimat dipisah terlebih dahulu menjadi kumpulan huruf yang sama besar, kemudian urutan kumpulan huruf itu ditukar dengan kumpulan huruf lain.

Misalkan kalimat yang harus dienkripsi adalah "TOLONG AKU", kalimat ini akan dienkripsi dengan jumlah tiap blok adalah 3, yang akan menghasilkan 3 blok huruf. Selanjutnya blok akan dipindah berdasarkan permutasi σ dari set {1,2,3} dengan σ(1) = 3, σ(2) = 1, σ(3) = 2. Hal ini berarti blok pertama digeser ke posisi blok ketiga, blok ketiga ke posisi kedua, dan blok kedua ke posisi pertama. Enkripsi akan menghasilkan kalimat baru yaitu "ONGAKUTOL". Untuk mendeskripsikan *block cipher*, maka kalimat diubah menggunakan invers dari σ.

Dalam proses enkripsi ada dua jenis key, yaitu *public key* dan *private key*. Perbedaan dari kedua key tersebut adalah dalam proses dekripsi. Dalam *private key cryptosystem*, setiap orang yang mempunyai *key* tersebut dapat dengan mudah mendekripsi pesan yang dirahasiakan. Karena itu key disimpan agar tidak dicuri orang lain dan hanya pihak-pihak tertentu saja yang boleh memiliki key.

Hal ini tidak berlaku untuk *public key*, dimana orang yang telah mengetahui key tersebut belum tentu bisa mendekripsikan pesan rahasia. Setiap orang bisa melakukan enkripsi untuk mengirim pesan, tetapi hanya orang tertentu saja yang memilliki *decryption key* (*private*

key) yang bisa menerjemahkan pesan itu. Contoh dari *public key* ada pada RSA Cryptosystem.

Dalam sistem RSA, setiap orang memiliki kunci enkripsi (n,e), n adalah bilangan prima yang sangat besar, dengan faktor p dan q, sementara e adalah eksponen yang relatif prima dengan hasil perkalian (p-1)(q-1). Proses membuat kunci RSA adalah sebagai berikut :

1. Cari bilangan prima p dan q
2. Hitung n = pq
3. Cari eksponen e
4. Hitung d yaitu $e^{-1}\ (mod\ (p-1)(q-1))$, d adalah *decryption key*

Untuk mengenkripsi, langkah-langkahnya mirip dengan *block cipher*, hanya saja setiap blok diubah dengan rumus

$$C = M^e\ mod\ n$$

M adalah representasi angka dari blok yang ingin dienkripsi, menghasilkan blok baru yaitu C.

Untuk proses dekripsi, si pembaca pesan harus mempunyai *decryption key* untuk membuka pesan tersebut. Penyerang harus terlebih dahulu mencari *decryption key*, dengan mencari faktor p dan q. Hal ini sangat sulit karena bilangan yang digunakan untuk n biasanya sangat besar, sehingga mencari nilai p dan q hanya dengan mengetahui nilai n dan e memakan waktu yang tidak masuk akal.

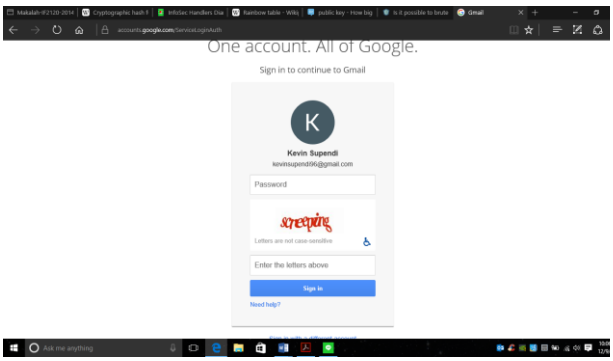
Proses dekripsi menggunakan nilai d, caranya dengan mengubah tiap blok dengan persamaan

$$M = C^d\ mod\ n$$

Dimana C adalah blok yang ingin diubah.

C. Algoritma Brute Force

Seluruh kata sandi rentan terhadap serangan *brute force*. Perbedaannya adalah apakah sistem keamanan kata sandi dapat mencegah atau memperlambat proses *brute force* tersebut, sehingga waktu dan tenaga yang dibutuhkan untuk memecahkan kata sandi tidak sebanding dengan hasilnya.



Gambar 5 Layanan surel yang meminta input tambahan untuk mengecek apakah pengguna bukan robot, setelah belasan kali gagal memasukkan kata sandi

Misalkan kata sandi membutuhkan minimal 8 karakter. Maka kemungkinan kata sandi yang bisa dibuat adalah 94^8, atau setara dengan 6.0956×10^{15}. Dengan kemampuan komputasi masa kini, dan penelitian terbaru mengatakan kata sandi dengan 8 karakter saja sudah tidak aman, karena dapat diproses dalam 6 jam [4]. Hal ini dapat diakali dengan menambah jumlah karakter, namun sayangnya kata sandi yang terlalu panjang akan sulit diingat, bila dicatat di kertas juga bisa saja kertas itu hilang atau dicuri.

Kemudian hal ini membuat kita berpikir, tentu saja tidak semua orang mempunyai kapasitas dan resource yang sedemikian besar, sehingga bisa menemukan kata sandi dengan kombinasi karakter dan simbol dalam waktu singkat. Sangat disayangkan, karena *brute force* dapat dilengkapi dengan *dictionary*, kumpulan kata-kata yang sering dipakai dalam kata sandi, seperti misalnya "rahasia", "password", "hello", dan lain sebagainya. Dengan mengecek kata-kata dalam *dictionary* terlebih dahulu, waktu pemecahan kata sandi dapat meningkat dengan cepat. Bahkan dalam *dictionary* bisa ditambahkan aturan-aturan, sehingga jika salah satu karakter diganti dengan angka (contoh : "h3llo), algoritma *brute force* dapat menemukannya dengan cepat.

III. Analisis Sistem Keamanan Kata Sandi yang Menggunakan Hash atau Enkripsi

A. Sistem Keamanan Hash

Hash adalah proses satu jalan. Sekali input di-hash, maka tidak ada cara untuk mengembalikan nilai input tersebut. Dalam sistem verifikasi kata sandi menggunakan hash, bentuk kata sandi yang sesungguhnya tidak pernah disimpan dalam database. Sebelum disimpan, kata sandi diproses terlebih dahulu menjadi *hash*. Saat pengguna ingin memasukki akun, input dari user pada saat itu di-*hash*, lalu dibandingkan dengan nilai *hash* yang ada di database layanan itu. Jika ternyata cocok, maka verifikasi berhasil dan pengguna boleh mengakses akun miliknya.

Seandainya penyerang dapat mengakses database dan mengambil informasi kata sandi, penyerang tidak dapat menggunakan informasi itu untuk mengakses akun pengguna. Verifikasi akan gagal karena input yang dimasukkan akan di-*hash* dua kali. Cara lain yang bisa digunakan adalah dengan menggunakan *brute force* untuk mencari kata sandi yang memilki nilai *hash* yang sama.

Untuk menangkal serangan *brute force* yang menggunakan *dictionary*, biasanya kata sandi akan diperkuat menggunakan *salt*. Untuk memecahkan kata sandi yang telah ditambahkan *salt*, maka penyerang harus mengubah *dictionary* untuk memperhitungkan *salt*. Jika setiap kata sandi menggunakan nilai *salt* yang berbeda, maka untuk setiap kata sandi *dictionary* yang harus dibuat juga berbeda. Belum lagi posisi *salt* yang bisa di awal, di tengah, atau di akhir kata sandi. Cara ini dapat menurunkan resiko dari serangan *brute force*.

Kekuatan dari fungsi Hash juga bergantung dari algoritma yang dipakai. Jika algoritma yang dipakai adalah algoritma yang memakan waktu, maka waktu yang diperlukan untuk *brute force* juga akan meningkat. SHA-1 dianggap sudah tidak aman, meskipun masih dipakai banyak orang.

B. Sistem Keamanan Enkripsi

Untuk keamanan enkripsi, tergantung dari algoritma enkripsi yang digunakan. Algoritma sederhana seperti *shift cipher* dapat dengan mudah didekripsikan. Dengan menganalisa frekuensi dari jumlah karakter, *brute force* tidak diperlukan. Cukup dengan menebak karakter nilai pergeseran.

Untuk enkripsi yang lebih rumit, seperti dalam enkripsi RSA, tidak mudah untuk mendekripsikan kata sandi. Misalkan penyerang sudah mempunyai *public key* yaitu n, untuk mencari nilai dari *decryption key* maka diperlukan faktor prima n yaitu p dan q. Nilai n yang digunakan adalah biasanya adalah angka yang sangat besar, sekitar 400 digit. Sementara saat ini angka yang bisa difaktorkan dengan cepat hanya mencapai sekitar 200 digit. Mendekripsikan enkripsi RSA tanpa *decryption key* tidak mungkin, dan mencari faktor dari n juga memakan waktu yang sangat lama dan tidak realistis. Karena itu *decryption key* (*private* key) harus disimpan dengan baik agar tidak diketahui banyak orang.

IV. Kelebihan dan Kekurangan Masing-Masing Sistem Keamanan

Sistem keamanan menggunakan fungsi Hash mempunyai tingkat keamanan yang lebih baik. Kata sandi yang sudah dimasukkan tidak akan bisa diubah kembali. *Database* hanya menyimpan nilai *hash*, dan penyerang tidak bisa menggunakan nilai itu untu memasukki akun pengguna. Kelebihan dari fungsi Hash juga sulit diserang oleh *dictionary attack*, karena dapat dengan mudah ditangkal oleh penambahan *salt* pada kata sandi.

Kelemahan dari sistem keamanan Hash adalah kata sandi tidak pernah disimpan dalam *database*, tetapi langsung hilang begitu diproses dalam fungsi Hash. Hal ini menjadi masalah bila kata sandi yang digunakan tidak boleh hilang. Untuk keperluan sehari-hari seperti layanan elektronik, akun media sosial, hal ini tidak begitu berpengaruh.

Fungsi Hash dalam kata sandi juga sulit untuk diubah atau ditingkatkan keamanannya. Bayangkan jika layanan media sosial telah menetapkan algoritma yang tetap untuk mengakses akun pengguna. Misalkan pertama sistem keamanan tersebut menggunakan algoritma SHA-1 pada awalnya. Beberapa tahun setelahnya ternyata SHA-1 sudah tidak aman, dan perusahaan tersebut mau mengganti algoritma hash menjadi SHA-2, maka berarti nilai *hash* dari akun-akun lama pengguna sudah tidak valid lagi. Karena itu untuk mengubah keamanan dalam fungsi Hash tidak mudah.

Sistem keamanan enkripsi memiliki tingkat kesulitan lebih untuk dipecahkan. Enkripsi RSA menggunakan angka yang besar tidak bisa dipecahkan begitu saja. Malah mustahil untuk mendekripsikan tanpa mengetahui nilai dari faktor prima n.

Sistem keamanan enkripsi mempunyai kelemahan yang fatal, terutama dalam sistem keamanan kata sandi. Suatu pesan yang bisa dienkripsi pasti bisa didekripsikan juga. Kemudian masih ada kemungkinan *decryption key* dapat diketahui orang lain, karena kita tidak bisa menjamin *key* yang disimpan akan selalu aman.

Dibandingkan dengan menggunakan *hash*, hal ini menjadi perbedaan besar dalam sistem keamanan kata sandi. Nilai *hash* tidak berguna dan tidak menguntungkan penyerang. Kemungkinan paling buruk adalah nilai *hash* hilang dan seluruh akun pengguna tidak bisa diakses. Bahkan pemilik layanan tersebut tidak bisa mengetahui kata sandi apa yang dimasukkan hanya dengan melihat nilai *hash* saja.

V. Kesimpulan

Dalam sistem keamanan kata sandi, fungsi Hash terbukti lebih aman dan efektif untuk verifikasi. Enkripsi tidak aman karena keberadaan kunci dekripsi yang bisa saja sewaktu-waktu diserang. Aplikasi enkripsi lebih cocok digunakan untuk mengirim pesan rahasia, karena isi pesan di dalamnya harus bisa didapatkan kembali, dan ini bisa dilakukan dengan proses dekripsi.

Inilah mengapa dalam layanan akun media sosial, kita tidak bisa mendapat kata sandi lama kita. Jika kita lupa dengan kata sandi, maka layanan tersebut akan meminta kita memasukkan kata sandi baru, ini berarti layanan media sosial itu mempunyai keamanan kata sandi yang baik, karena layanan tersebut tidak menyimpan kata sandi yang kita masukkan.

VI. Ucapan Terima Kasih

Saya ingin mengucapkan terimakasih kepada Tuhan Yang Maha Esa sehingga makalah ini dapat selesai dengan baik. Saya ingin berterimakasih kepada kedua orang tua saya yang telah membesarkan saya dengan baik, menyekolahkan saya sampai saya kuliah, saya sungguh-sungguh bersyukur. Tidak lupa juga terimakasih untuk Pak Rinaldi, dosen mata kuliah IF2120, yang mengajarkan banyak hal, dari materi kuliah sampai hal-hal dalam kehidupan sehari-hari. Terimakasih juga kepada teman-teman saya yang telah membantu saya dalam mengerti materi kuliah. Terimakasih untuk semua orang yang telah memotivasi dan menginspirasi saya dalam pembuatan makalah ini.

Daftar Pustaka

[1] Kenneth H. Rosen. "Discrete Mathematics and Its Application", 7th Edition, Mc Graw-Hill, New York, 2012.

[2] http://www.scribd.com/doc/199819816/Fundamental-Data-Structures diakses tanggal 9 Desember 2015 pk. 09.21 WIB

[3] https://ssd.eff.org/en/module/what-encryption diakses tanggal 9 Desember 2015 pk. 09.22 WIB

[4] http://www.dshield.org/diary/Hashing+Passwords/11110 diakses tanggal 9 Desember 2015 pk 09.34 WIB

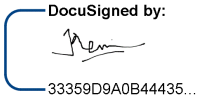
[5] http://security.stackexchange.com/questions/43683/is-it-possible-to-brute-force-all-8-character-passwords-in-an-offline-attack diakses tanggal 9 Desember 2015 pk 21.22 WIB

[6] http://www.lockdown.co.uk/?pg=combi&s=articles diakses tanggal 9 Desember 2015 pk 21.30 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, 9 Desember 2015



ttd

Kevin Supendi 13514094