Penyimpanan Password Dengan Fungsi Hash

Felix Setiawan - 13518078
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13518078@std.stei.itb.ac.id

***Abstrak*—Algoritma hash umum digunakan untuk mengubah password menjadi hash yang secara teori tidak dapat dibaca oleh manusia secara langsung. Makalah ini membahas tentang aplikasi dari fungsi hash untuk menyimpan password ke dalam database. Bagian pertama membahas tentang dasar teori tentang hash dan password. Bagian selanjutnya membahas tentang penyimpanan password dengan metode hashing.**

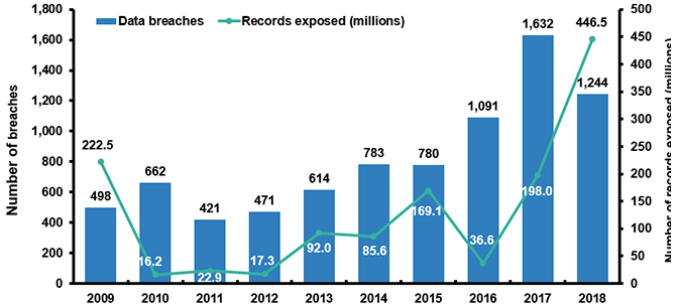
***Kata kunci*—fungsi, hash, password, penyimpanan.**

I. PENDAHULUAN

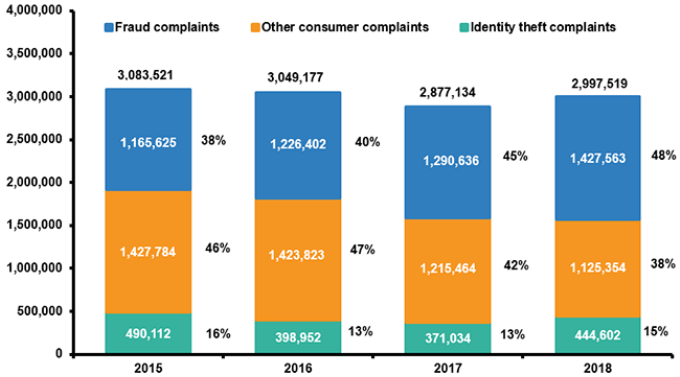
Salah satu kemajuan peradaban manusia yang sangat signifikan adalah merakyatnya internet. Banyaknya pengguna internet menuntut adanya personalisasi tampilan halaman web, atau yang biasa disebut dynamic webpage. Dari beberapa jenis dynamic web page, suatu web page yang berubah berdasar user yang mengunjungi pasti menggunakan sistem registrasi dan login akun. Untuk login diperlukan suatu identifier unik dan rangkaian karakter acak yang bebas sesuai keinginan pemilik akun. Identifier unik dapat berupa username atau alamat email, sedangkan rangkaian karakter acak umumnya disebut password.

Password adalah rangkaian karakter yang digunakan untuk autentikasi seorang user di suatu sistem. Password dari setiap pengguna disimpan dalam suatu database. Database ini sangat penting karena berisi kunci ke berbagai gembok yang ada. Oleh karena itu, faktor keamanan dari metode penyimpanan data authentikasi, salah satunya password, pada suatu database menjadi sangat penting untuk melindungi privasi penggunanya.

Kebocoran database yang berisi informasi pribadi merupakan salah satu yang ingin dicegah dengan penggunaan hashing. Dilansir dari https://www.iii.org/fact-statistic/facts-statistics-identity-theft-and-cybercrime, di tahun 2018 hingga 2019 ini, kebocoran data mengalami penurunan, dan kasus yang berhasil dipecahkan meningkat drastis.



Salah satu akibat dari bocornya database ini adalah identity fraud. Seseorang yang menggunakan identitas orang lain untuk keuntungan pribadi. Tindakan ini sangat merugikan karena dapat merusak reputasi maupun keuangan seseorang. Dilansir dari https://www.iii.org/fact-statistic/facts-statistics-identity-theft-and-cybercrime, Identity fraud di Amerika Serikat mengalami peningkatan dari tahun 2016 ke 2018, penipuan lain-lain seperti pemalsuan kartu kredit juga meningkat.



Tindakan kriminal seperti identity fraud tersebut dapat dicegah dengan meningkatkan faktor keamanan penyimpanan data autentikasi. Dengan metode-metode tertentu, keamanan penyimpanan data dapat ditingkatkan sehingga lebih resistant terhadap serangan siber. Terdapat berbagai metode yang digunakan untuk menyimpan password di database dengan tingkat keamanan yang berbeda-beda. Salah satu metode penyimpanan password paling aman adalah dengan menggunakan fungsi hash yang dapat mengubah password menjadi bentuk lain yang bersifat satu-arah. Satu-arah berarti hasil konversi dari password asli tidak dapat didekripsi menggunakan rumus yang sama. Meskipun seorang hacker mendapat database yang sudah dikonversi, ia tidak akan dapat menggunakannya untuk mengakses informasi dari suatu akun. Pada makalah ini akan dibahas penggunaan fungsi hash dalam metode penyimpanan suatu password dan mekanismenya.

II. FUNGSI HASH

Fungsi Hash merupakan algoritma yang mengubah teks atau pesan (text or message) menjadi sederetan karakter acak yang memiliki karakter yang sama.

Hash juga termasuk salah satu bentuk teknik kriptografi tanpa menggunakan kunci (unkeyed cryptosystem). Selain itu hash memiliki nama lain yang juga dikenal yaitu “one-way function”.

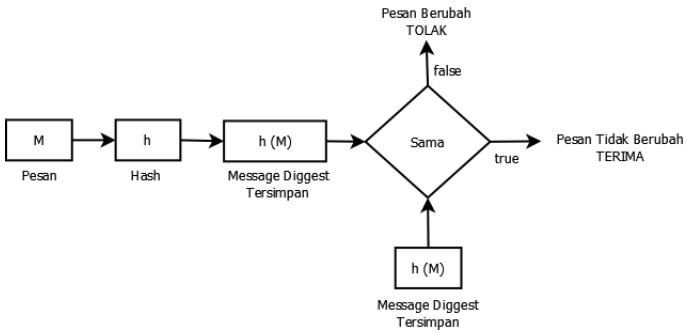
Fungsi Hash adalah fungsi yang menerima masukan string yang panjangnya sembarang selanjutnya

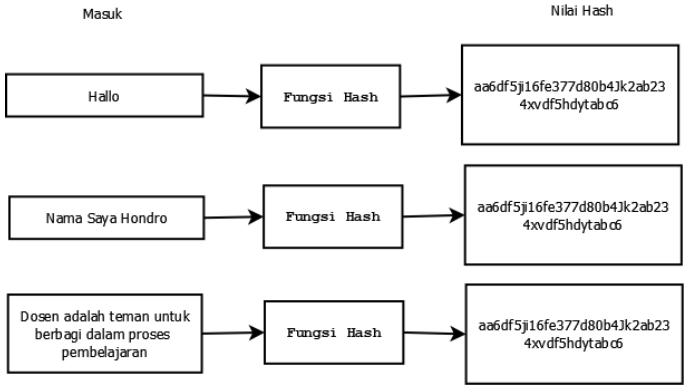
mentransformasikannya menjadi string keluaran yang panjangnya tetap (fixed) yang biasanya berukuran jauh lebih kecil daripada ukuran string semula.

Menurut Kaufman et. al. (2002), Fungsi hash dapat digunakan sebagai:

(1) Menyimpan Password, (2) Sebagai Message Integrity, (3) Sebagai Message Fingerprint

Pengujian Keutuhan Pesan





Persamaan Fungsi Hash

h = H(m)

Keterangan:

M = pesan ukuran sembarang

H = Fungsi Hash

h = nilai hash atau pesan ringkas (message-digest)

h<<<M

Contoh: size(M) = 1 MB –> size(h) = 128 bit !!!

Nama lain fungsi hash adalah:

– fungsi kompresi (compression function)

– cetak-jari (fingerprint)

– cryptographic cheksum

– message integrity check (MIC)

– manipulation detection code (MDC)

Fungsi Hash Satu-Arah

Fungsi hash satu-arah (one-way function) adalah fungsi hash yang bekerja dalam satu arah. Satu arah pesan yang diubah menjadi message digest tidak dapat dikembalikan lagi menjadi pesan semula (irreversible).

Masukan fungsi hash adalah blok pesan (M) dan keluaran dari hashing blok pesan sebelumnya,

hi = H(Mi, hi-1)

skema fungsi hash ditunjukkan pada gambar berikut ini,

Beberapa contoh algoritma kriptografi dengan menggunakan kunci fungsi hash, antara lain:

– MD2, MD4, MD5,

– secure Hash Function (SHA),

– Snefru,

– N-hash,

– RIPE-MD, dan lain-lain

Catatan: (MD singkatan dari Message Digest)

Spesifikasi beberapa fungsi hash:

Algorit ma	Ukuran *message digest* (bit)	Ukuran blok pesan	Kolisi
MD2	128	128	Ya
MD4	128	512	Hampir
MD5	128	512	Ya
RIPEMD	128	512	Ya
RIPEMD -128/256	128/256	512	Tidak
RIPEMD -160/320	160/320	512	Tidak
SHA -0	160	512	Ya
SHA -1	160	512	Ada cacat
SHA -256/224	256/2 24	512	Tidak
SHA -512/384	512/384	1024	Tidak
WHIRLPOOL	512	512	Tidak

Layanan yang diberikan Fungsi Hash:

1. Menjaga Integritas Data.

Dengan fungsi hash sangat peka terhadap perubahan 1 bit pada pesan, Pesan berubah 1 bit, nilai hash berubah sangat signifikan, Bandingkan nilai hash baru dengan nilai hash lama. Jika sama, pesan masih asli. Jika tidak sama, pesan sudah dimodifikasi.

2. Menghemat waktu pengiriman.

Misalkan untuk memverifikasi sebuah salinan arsip dengan arsip asli. Salinan dokumen berada di tempat yang jauh dari basis data arsip asli. Ketimbang mengirim data salinan arsip tersebut secara keseluruhan ke komputer pusat (yang membutuhkan waktu transmisi lama), lebih baik mengirimkan message digest-nya. Jika message digest salinan arsip sama dengan message digest arsip asli, berarti salinan arsip tersebut sama dengan arsip master.

3. Menormalkan panjang data yang beraneka ragam.

Misalkan password panjangnya bebas (minimal 8 karakter). Password disimpan di komputer host (server) untuk keperluan otentikasi pemakai komputer. Password disimpan di dalam basisdata. Untuk menyeragamkan panjang field

password di dalam basisdata, password disimpan dalam bentuk nilai hash (panjang nilai hash tetap).[9]

D. Algortima Hash Umum

Terdapat dua algoritma *hash* yang umum digunakan, yaitu *Message Digest 5* (MD5) dan *Secure Hash Algorithm* (SHA). Kedua algoritma tersebut umumnya digunakan untuk memverifikasi *digital signature* dan integritas data dari suatu *file*. Berikut penjelasan untuk kedua algoritma *hash* tersebut [4].

1. *Message Digest 5* (MD5)

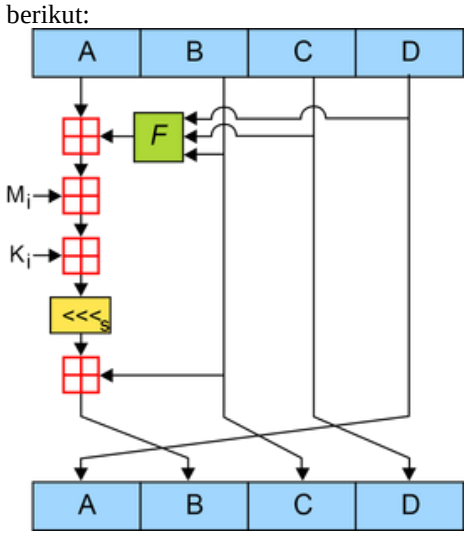
MD5 merupakan salah satu jenis Cryptographically secure hashing untuk proses pembuatan suatu "sidik jari" (Fingerprint atau kerap juga disebut digest untuk suatu naskah. Algoritma MD5 dirancang oleh Ron Rivest dan penggunaannya sangat populer dikalangan komunitas open source sebagai checksum untuk file yang dapat di download. MD5 juga kerap digunakan untuk menyimpan password dan juga digunakan dalam digital signature dan certificate. Spesifikasi lengkap untuk algoritma MD5 ada pada suatu RFC (request for comment).

Besarnya blok untuk MD5 adalah 512 bit sedangkan digest size adalah 128 bit. Karena word size ditentukan sebesar 32 bit, satu blok terdiri dari 16 word sedangkan digest terdiri dari 4 word. Indeks untuk bit dimulai dari 0.

Preprocessing dimulai dengan padding sebagai berikut :

1. Bit dengan nilai 1 ditambahkan setelah ahir naskah.
2. Deretan bit dengan nilai 0 ditambahkan setelah itu, sehingga besar dari naskah mencapai nilai 448 (mod 512) (sedikitnya 0 dan sebanyaknya 511 bit dengan nilai 0 ditambahkan, sehingga tersisa 64 bit untuk diisi agar besar naskah menjadi kelipatan 512).
3. 64 bit yang tersisa diisi dengan besar naskah asli dalam bit.

Skema pemrosesan MD5 dapat dilihat di gambar berikut:



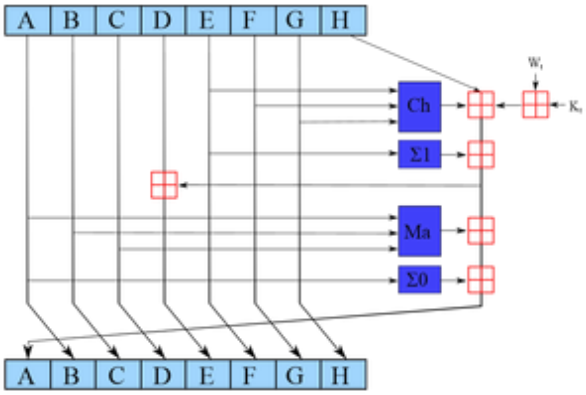
Sumber: https://kl2217.wordpress.com/2011/07/21/common-hashing-algorithms/

2. *Secure Hash Algorithm* (SHA)

Tiga algoritma SHA memiliki struktur berbeda-beda, yaitu *SHA-0*, *SHA-1*, dan *SHA-2*. *SHA-1* sangat mirip dengan S*HA-0*, tetapi mengoreksi kesalahan yang terdapat di spesifikasi hash original yang menyebabkan beberapa kelemahan. *SHA-0* dan *SHA-1* tidak *collision resistant*, terdapat kemungkinan kolisi pada kedua algoritma tersebut. Oleh karena itu, *SHA-2* lebih umum digunakan dan lebih aman karena kemungkinan kolisi yang sangat kecil. *SHA-2* terdiri dari *SHA-224*, *SHA-256*, *SHA-385*, dan *SHA-512* yang menghasilkan *hash value* sebesar nomornya.

Cara kerja SHA-2 mirip dengan MD5, dengan perbedaannya terletak pada kondisi 256-bit (yang dipecah menjadi 8 pecahan 32-bit, yaitu A, B, C, D, E, F, G, dan H), fungsi yang digunakan, dan dilakukan dalam lima *rounds*.

Pemrosesan satu ronde operasi SHA-2 dapat dilihat di bawah ini.



Sumber: https://kl2217.wordpress.com/2011/07/21/common-hashing-algorithms/

Komponen biru pada skema tersebut merupakan fungsi-fungsi di bawah ini.

Ch (E, F, G) = (E ∧ F) ⊕ (¬E ∧ G)

Ma (A, B, C) = (A ∧ B) ⊕ (A ∧ C) ⊕ (B ∧ C)

$\sum_0$ (A) = (A ⋙ 2) ⊕ (A ⋙ 13) ⊕ (A ⋙ 22)

$\sum_1$ (E) = (E ⋙ 6) ⊕ (E ⋙ 11) ⊕ (E ⋙ 25)

III. PASSWORD

A. Definisi *Password*

Password adalah rangkaian karakter yang digunakan untuk autentikasi seorang user di suatu sistem computer. Contohnya, Anda mungkin memiliki suatu akun di komputer Anda yang mewajibkan Anda untuk log in. Agar dapat mengakses akun Anda, Anda harus memasukkan pasangan username dan password yang tepat. Kombinasi ini sering disebut suatu login. Umumnya username adalah informasi publik, sedangkan password merupakan privasi setiap pengguna.

Kebanyakan password terdiri dari beberapa karakter yang umumnya mengandung huruf, angka, dan simbol, tetapi tidak spasi. Password yang mudah

diingat sering menjadi pilihan banyak orang, tetapi jangan membuatnya terlalu simpel hinga dapat ditebak dengan mudah oleh orang lain. Password yang paling aman menggunakan kombinasi huruf, angka, dan simbol, dan tidak mengandung kata yang benar-benar ada.[5]

B. Strong Password

Faktanya adalah meskipun mayoritas website sudah aman, namun selalu ada kemungkinan orang lain berniat jahat mencoba untuk mengakses atau mencuri informasi-informasi Anda. Tindakan seperti ini umumnya dikenal sebagai hacking (meretas). Kata sandi yang kuat adalah salah satu cara untuk mempertahankan akun dan informasi pribadi Anda dari hacker.

Kata sandi (password) yang kuat hendaknya mudah Anda ingat tetapi sulit ditebak orang lain. Mari kita lihat beberapa hal penting untuk dipertimbangkan saat membuat password.

- Jangan pernah menggunakan informasi pribadi Anda seperti nama, ulang tahun, username, atau alamat email. Informasi-informasi ini umumnya dapat diakses publik, sehingga orang lain lebih mudah menebak password Anda.
- Gunakan password yang panjang. Setidaknya panjang password minimal 6 digit, meskipun dapat lebih panjang untuk keamanan ekstra.
- Jangan memakai password yang sama untuk semua akun Anda. Jika seseorang menemukan kata sandi pada satu akun Anda, maka akun-akun Anda lainnya akan terancam.
- Gunakan kombinasi angka, simbol, huruf kapital dan kecil.
- Hindari menggunakan kata-kata yang dapat ditemukan di dalam kamus. Misalnya, berenang1 merupakan password yang lemah.
- Password acak adalah password yang terkuat. Jika Anda mengalami kesulitan membuat password acak ini, maka Anda dapat menggunakan aplikasi password generator.[6]

C. Metode Penyimpanan Password

Ada beberapa cara suatu situs menyimpan password Anda, dan beberapa cara dianggap lebih aman dari yang lainnya. Berikut merupakan penjelasan singkat dari beberapa metode yang popular.

1. *Plaintext Passwords*

Metode paling sederhana dalam penyimpanan *password* adalah dalam bentuk *plaintext*. Itu berarti di suatu tempat di server mereka, terdapat database dengan nama pengguna dan kata sandi Anda dalam bentuk yang dapat dibaca manusia (yaitu, jika kata sandi Anda testing123, disimpan dalam database sebagai testing123). Ketika Anda memasukkan identitas login Anda di situs, identitas Anda dicocokkan dengan database. Ini adalah metode yang paling buruk, dalam hal keamanan, dan sebagian besar situs web terkemuka tidak menyimpan kata sandi dalam teks biasa.

Apakah kata sandi yang kuat penting? Tidak sama sekali. Tidak peduli berapa lama atau kuat kata sandi Anda, jika disimpan dalam teks biasa dan situs diretas, kata sandi Anda mudah diakses oleh siapa pun, tidak perlu usaha berlebih. Itu masih penting dalam hal menyembunyikan kata sandi Anda dari, katakanlah, teman-teman Anda, atau orang lain yang dapat dengan mudah menebaknya, tetapi itu tidak akan membuat perbedaan jika situs diretas.

2. *Basic Password Encryption*

Untuk menambahkan perlindungan ke kata sandi Anda dari yang disediakan oleh teks biasa, sebagian besar situs mengenkripsi kata sandi Anda sebelum mereka menyimpannya di server mereka. Enkripsi, bagi Anda yang tidak tahu, menggunakan kunci khusus untuk mengubah kata sandi Anda menjadi string teks acak. Jika seorang hacker mendapatkan untaian teks acak ini, mereka tidak akan bisa masuk ke akun Anda kecuali mereka juga memiliki kunci, yang kemudian dapat mereka gunakan untuk mendekripsi.

Masalahnya adalah, kuncinya sering disimpan pada server yang sama dengan kata sandi, jadi jika server diretas, seorang hacker tidak perlu melakukan banyak pekerjaan untuk mendekripsi semua kata sandi, yang berarti metode ini masih sangat tidak aman.

Apakah kata sandi yang kuat penting? Tidak. Karena dengan mudah kata sandi dapat didekripsi dengan kunci, kata sandi yang kuat juga tidak akan membuat perbedaan di sini. Sekali lagi: ini dalam hal situs diretas; jika Anda memiliki teman atau anggota keluarga yang ingin tahu rahasia Anda, kata sandi yang kuat dapat mencegah mereka menebaknya.

3. *Hashed Passwords*

Hashed mirip dengan enkripsi dalam arti mengubah kata sandi Anda menjadi serangkaian huruf dan angka yang panjang untuk membuatnya tetap tersembunyi. Namun, tidak seperti enkripsi, hashing adalah jalan satu arah: Jika Anda memiliki hash, Anda tidak dapat menjalankan algoritma invers untuk mendapatkan kata sandi asli. Ini berarti seorang peretas harus mendapatkan hash dan kemudian mencoba sejumlah kombinasi kata sandi yang berbeda untuk melihat mana yang berfungsi.

Namun, ada kerugian dari metode ini. Sementara seorang peretas tidak dapat memecahkan kode hash kembali ke kata sandi asli, mereka dapat mencoba banyak kata sandi yang berbeda sampai satu cocok dengan hash yang mereka miliki. Komputer dapat melakukan ini dengan sangat cepat, dan dengan bantuan sesuatu yang disebut rainbow tables — yang

pada dasarnya adalah daftar trilyunan hash yang berbeda dan kata sandi yang cocok. Coba ketikkan e38ad214943daad1d64c102faec29de4afe9da3d ke Google. Anda akan dengan cepat menemukan bahwa itu adalah hash SHA-1 untuk "password1".

Apakah kata sandi yang kuat penting? Dalam hal ini, ya. Tabel pelangi terdiri dari kata sandi yang telah diuji terhadap hash, yang berarti yang sangat lemah akan dipecahkan dengan sangat cepat. Namun, kelemahan terbesar mereka bukanlah kompleksitas, tetapi panjang. Sebaiknya Anda menggunakan kata sandi yang sangat panjang (seperti " correct horse battery staple" yang terkenal di XKCD) daripada yang pendek dan rumit (seperti kj$fsDl #).

4. *Hashed Passwords with a Dash of Salt*

Salting a hash berarti menambahkan serangkaian karakter acak - disebut "salt" - ke awal atau akhir kata sandi Anda sebelum membuat hash. Metode ini menggunakan salt yang berbeda untuk setiap kata sandi, dan bahkan jika salt disimpan di server yang sama, itu akan membuatnya sangat sulit untuk menemukan hash salt itu di tabel pelangi, karena masing-masing panjang, kompleks, dan unik. LinkedIn terkenal tidak menggunakan hash salt, yang membuat mereka di bawah pengawasan ketat setelah peretasan terakhir mereka — seandainya mereka menggunakan salt, pengguna mereka akan lebih aman.

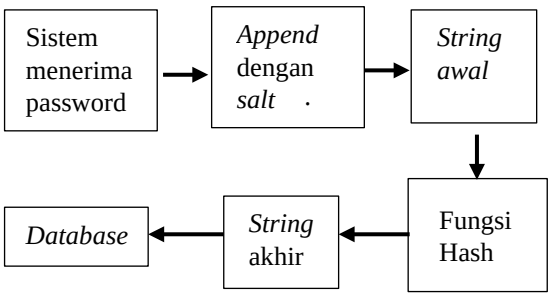
Apakah kata sandi yang kuat penting? Pasti! Sayangnya, bagaimanapun, kita telah mencapai titik komputer begitu cepat sehingga banyak yang mampu meretas bahkan hash salt. Ini bisa memakan waktu yang sangat lama — tentu saja lebih lama daripada menggunakan tabel pelangi — tetapi masih bisa dilakukan. Ini berarti kekuatan kata sandi Anda masih penting, karena semakin lama dan semakin kompleks, semakin lama waktu yang dibutuhkan untuk memecahkan dalam serangan brute force.[7]

IV. PENYIMPANAN PASSWORD KE DATABASE DENGAN METODE HASHING WITH DASH OF SALT

Penyimpanan *password* ke *database* di *server* terdiri dari beberapa tahap, yaitu (misalnya dengan metode hash dan penambahan salt):

1. Sistem menerima *input password*.
2. Penambahan salt dilakukan dengan *append salt* ke awal atau akhir *password*.
3. Melakukan *hashing password* tergantung metode penyimpanan.
4. Menyimpan *password* ke *database*.

Alur penyimpan *password* adalah sebagai berikut.



Gambar 5. Skema alur penyimpanan *password*.

Langkah pencocokan identitas login pengguna dengan database mirip dengan mekanisme penyimpanan password. Perbedaan keduanya terletak di langkah terakhir, yaitu membandingkan nilai yang disimpan pada *database* dengan masukkan pengguna. Jika nilai masukan sama dengan yang tersimpan di *database*, maka sistem akan memberi akses ke pengguna.

Sebagai contoh, akan digunakan sebuah *strong password*, yaitu “^Bv8fU[.!p+cRMk!”. Dengan metode *Hashed Passwords with Salt*, *password* tersebut akan ditambahkan dengan sebuah *salt* acak, misalnya “d2Clm0dUA5ZT1bSS”. Hasil dari *string* yang telah ditambahkan *salt* dan sebelum dimasukkan ke fungsi *hash* adalah “^Bv8fU[.!p+cRMk! d2Clm0dUA5ZT1bSS”. *String* tersebut kemudian dimasukkan ke suatu fungsi *hash*, misalnya SHA256. *Hash value* yang dihasilkan dari fungsi tersebut adalah FC5D1620DDD2E8AFBD2DB9DE772C1241C3A8E494A67504D73B8AB5687911DE26. *Hash value* ini akan disimpan dalam *database*.

Saat seorang pengguna ingin mengakses *account* miliknya, *password* yang dimasukkan akan melalui proses pencocokan, kemudian sistem akan membandingkan hasilnya dengan *hash value* yang disimpan di *database*.

Dengan menyimpan *password* dalam bentuk *hash values*, seorang *hacker* akan mengalami kesulitan dalam mencoba untuk mendapatkan *password* aslinya. Untuk menemukan *password* asli dari sebuah *hash value* SHA-512 tanpa *salt* secara *bruteforce*, dibutuhkan waktu 2^240 * 2^-2 = 2^238 ~ 10^72s ~ 3,17 * 10^64 tahun [8]. Dengan penambahan *salt*, waktu yang dibutuhkan akan bertambah secara eksponensial. Serangan melalui *rainbow table* pun tidak akan mudah karena penambahan *salt* membuat *hash value* menjadi lebih panjang, kompleks, dan unik.

V. KESIMPULAN

Tingkat keamanan penyimpanan *password* sangat penting untuk menjaga informasi pribadi dari setiap orang. Salah satu metode penyimpanan paling aman adalah Hashed password with dash of salt yang menambah sejumlah karakter ke awal atau akhir password anda kemudian mengubah *string password* masukan menjadi sebuah *hash value*. *Hash value* tidak dapat didekripsi dengan kunci yang sama ketika enkripsi. Oleh karena itu, penggunaan *hash* untuk menyimpan *password* meningkatkan keamanan dari metode penyimpanannya.

VI. UCAPAN TERIMA KASIH

Penulis ingin mengucapkan terima kasih kepada Ibu Fariska Zakhralativa Ruskanda, S.T., M.T. selaku dosen pengajar mata kuliah Matematika Diskrit kelas 3 yang telah memberikan ilmu yang digunakan dalam penulisan makalah ini. Penulis juga ingin mengucapkan terima kasih kepada semua penulis dari berbagai referensi yang digunakan pada penulisan makalah ini.

REFERENSI

[1] Munir, Rinaldi. 2004. Diktat IF5054 Kriptografi. Bandung: Departemen Teknik Informatika.

[2] Techterms. *Hash*. Diakses pada 5 Desember 2019 dari https://techterms.com/definition/hash.

[3] Thakral, Kushagra. *Applications of Hashing*. Diakses pada 5 Desember 2019 dari https://www.geeksforgeeks.org/applications-of-hashing/.

[4] C. J. Kaufman, Rocky Mountain Research Lab., Boulder, CO, private communication, May 1995.

[5] Christensson, P. (2006). Password Definition. Retrieved 2019, Dec 5, from https://techterms.com

[6] Anonymous. *Keamanan Internet: Membuat Kata Sandi (Password) yang Kuat.* Diakses pada 6 Desember 2019 dari https://edu.gcfglobal.org/en/keamanan-internet/membuat-kata-sandi-password-yang-kuat/1/

[7] Gordon, Whitson. *How Your Passwords Are Stored on the Internet (and When Your Password Strength Doesn't Matter).* Diakses pada 6 Desember 2019 dari https://lifehacker.com/5919918/how-your-passwords-arestored-on-the-internet-and-when-your-password-strength-doesnt-matter.

[8] Stack Overflow. *How long to brute force a salted SHA-512 hash? (salt provided).* Diakses pada 6 Desember 2019 dari https://stackoverflow.com/questions/6776050/how-long-to-brute-force-asalted-sha-512-hash-salt-provided.

[9] Hondro, Rivalri Kristianto. *Fungsi Hash dalam Algoritma Kriptografi.* Diakses pada 6 Desember 2019 dari https://rivalryhondro.wordpress.com/2018/04/04/fungsi-hash/

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, 6 Desember 2019

Felix Setiawan 13518078