IMPLEMENTASI ALGORITMA *BCRYPT* DAN *BASE64* DALAM ENKRIPSI KATA SANDI MENGGUNAKAN BAHASA PEMROGRAMAN *GO*

http://dx.doi.org/10.28932/jutisi.vXiX.X

Riwayat Artikel
Received: xx Bulan 20xx | *Final Revision*: xx Bulan 20xx | *Accepted*: xx Bulan 20xx

Arkjun Yudistira Pratama[#1], Pasca Arthurito[*2]

[#] *Teknik Informatika, Universitas Dipa MakassarJl. Perintis Kemerdekaan No.KM.9, Makassar, Kode-pos, Indonesia*
[1]nebo.arkjuniork.yudistira@gmail.com

[*]*Teknik Informatika, Universitas Dipa MakassarJl. Perintis Kemerdekaan No.KM.9, Makassar, Kode-pos, Indonesia*
[2]email.penulis2@domain.ekstensi

Corresponding author: email.penulis-corr@domain.extensi

***Abstrak* — Kata sandi yang disimpan dalam sistem harus dijaga dengan baik untuk mencegah akses yang tidak sah. Penggunaan enkripsi membantu melindungi kata sandi dari tebakan, tetapi ada celah keamanan yang perlu diperhatikan. Salah satu solusi adalah menambahkan lapisan keamanan dengan algoritma *Base64*. Namun, hal ini bisa mempengaruhi efisiensi sistem. Untuk meningkatkan efisiensi dalam proses enkripsi dan dekripsi, sistem akan diimplementasikan dengan bahasa pemrograman *Go*.**

***Kata kunci*— base64; bcrypt; go; kriptografi.**

BCRYPT AND BASE64 ALGORITHM IMPLEMENTATION ON PASSWORD ENCRIPTION USING GO PROGRAMMING LANGUAGE

***Abstract* — Passwords, crucial for security, require strong protection. Encryption helps prevent easy guessing, but vulnerabilities exist. To bolster security without compromising efficiency, implementing the Base64 algorithm adds an extra layer. However, this might impact system performance. To optimize both efficiency and performance in encryption and decryption, the system will use the Go programming language.**

***Keywords*— *base64*; *bcrypt*; *cryptography*; go.**

I. PENDAHULUAN

Maraknya perkembangan kecanggihan teknologi informasi semakin memudahkan kegiatan manusia dalam segala bidang, salah satunya kemudahan mengakses data ataupun informasi yang diinginkan. Namun, manfaat tersebut juga diikuti dengan munculnya kejahatan siber atau cybercrime. Cybercrime adalah bentuk kejahatan yang dilakukan melalui perangkat seperti komputer dan jaringan [1]. Munculnya berbagai macam tindak kejahatan siber mengakibatkan kekhawatiran dikalangan masyarakat, terkhusus pada keamanan data.

Keamanan data sudah sepatutnya mendapatkan prioritas dalam pemecahan masalah teknologi, mengingat banyaknya arus perputaran data dan informasi sehingga rawan terjadi cybercrime. Contoh jenis kejahatan siber yang mengancam kerahasiaan data adalah brute force, merupakan kegiatan penyerangan sistem keamanan perangkat melalui percobaan kunci yang dilakukan oleh hacker dengan teknik password cracker. Password cracker adalah teknik mengetahui kata sandi terenkripsi dengan algoritma tertentu [1].

Berdasarkan permasalahan yang muncul, diperlukan penyelesaian untuk memperkuat proteksi data dan informasi dalam perangkat elektronik. Berbagai solusi untuk memperketat keamanan data salah satunya melalui teknik kriptografi. Kriptografi merupakan ilmu yang mempelajari tentang cara proteksi informasi melalui perubahan ke dalam bentuk karakter yang tidak dapat dimengerti [2].

Dalam kejahatan brute force, serangan terjadi pada algoritma SHA-256 dan MD5 yang mengakibatkan collision. Namun, serangan-serangan tersebut tidak dapat menembus algoritma bcrypt. Bcrypt merupakan hashing kata sandi dengan jumlah ilustrasi yang lebih banyak untuk menimbulkan efek tahan lama terhadap serangan brute force [3].

Sedangkan algoritma Base64 merupakan jenis algoritma yang digunakan untuk encoding (penyandian) dan decoding data dalam bentuk ASCII, berdasarkan pada bilangan 64. Adanya fitur key dari proses encryption membuatnya lebih efektif digunakan untuk perlindungan data. Base64 sering digunakan sebagai format data untuk mentransmisikan informasi. Keuntungannya adalah hasil Base64 berbentuk teks sederhana, sehingga memudahkan pengiriman data dibandingkan dengan format biner [4].

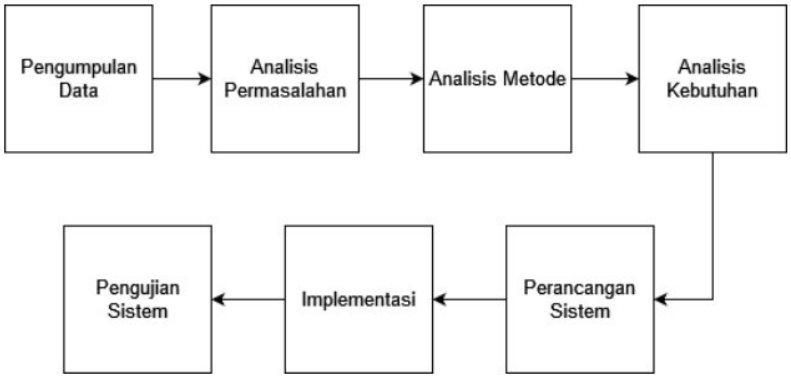
Penelitian yang dilakukan oleh Toras Pangidoan membuktikan bahwa kinerja keamanan algoritma Bcrypt tergolong sangat baik dalam menangkal serangan brute force untuk karakter campuran dalam kata sandi [3]. Sedangkan berdasarkan penelitian oleh Azlin, dkk menunjukkan bahwa algoritma Base64 efektif digunakan dalam mengamankan teks dengan mengubahnya menjadi karakter acak melalui enkripsi. Selanjutnya, proses ini memungkinkan pengembalian teks asli dari enkripsi tanpa mengubah sifat aslinya [4].

Dari uraian di atas, dibutuhkan implementasi yang efektif untuk menjaga keamanan data. Kajian ini ditujukan untuk menguji kombinasi algoritma Bcrypt dan Base64 dalam enkripsi kata sandi menggunakan Golang. Kombinasi direalisasikan dalam bentuk aplikasi web untuk menguji efektivitasnya.

II. METODE

Proses pengumpulan data dan informasi untuk mendukung pengkajian adalah melalui metode library search. Dimana metode ini memungkinkan peneliti mencari data tentang materi-materi yang dibutuhkan serta penelitian relevan untuk menambah referensi bacaan terkait Kriptografi algoritma Bcrypt dan Base64.

Metode yang digunakan dalam penelitian ini yakni metode kualitatif. Peneliti menerapkan pendekatan kualitatif karena komponen yang digunakan dalam pembuatan sistem yang bersifat deskriptif/ menjelaskan. Berikut ini adalah tahapan dalam penelitian kualitatif yang dilakukan, adalah:



Gambar 1.Tahap Penelitian

Proses diawali dengan pengumpulan data berupa materi dan teori yang diambil dari artikel, jurnal, buku dan sumber lainnya. Selanjutnya, permasalahan dapat diidentifikasi, analisis metode yang dapat diterapkan serta analisis kebutuhan dalam pemecahan masalah.

Proses perancangan sistem dilakukan dengan menggunakan bahasa pemrograman Go hingga didapatkan implementasi dari kombinasi algoritma yang efektif. Proses terakhir adalah pengujian yang dilakukan dengan metode blackbox. Pengujian terhadap sistem yang dibangun dalam bentuk aplikasi web dengan menggunakan data secara acak.

III. HASIL DAN PEMBAHASAN

Pengujian dilakukan menggunakan metode blackbox dengan cara melakukan hashing pada kata sandi, lalu melakukan komparasi untuk menentukan kesamaan kata sandi dan nilai hash. Hal yang perlu diperhatikan adalah pada proses komparasi pengguna dapat memilih untuk mengaktifkan komparasi dengan penyandian base64 atau tidak.

A. Enkripsi

Penggunaan algoritma *base64* menambah lapisan keamanan pada proses enkripsi kata sandi, proses enkripsi kata sandi dengan menggunakan algoritma *base64* dan *bcrypt* dimulai dengan mengirimkan *payload* dalam hal ini adalah kata sandi yang kemudian disandikan pada proses *encoding* menggunakan algoritma *base64 yang* nantinya akan memberikan hasil berupa teks acak, hasil encoding kemudian akan di-hash menggunakan *bcrypt*.

TABEL 1
HASIL ENKRIPSI

Kata Sandi	*Cost*	*Hash*	Waktu
katasandi123	10	$2a$10$.tekmi3n8v8Zs/uJ3FK3XOfJGKSev8ZcNQlG0u2HWTCdMFw99L2zK	131.246058ms
katasandi123	12	$2a$12$6KQwufOlWm.0mN5XQ6TINukKv6MlxuwtJszjsxbWzMSek51OZdrJa	523.025374ms
katasandi123	14	$2a$14$kgkeb42rRAquMKJkWDoEte/qtTPuaDbJOlHu0wCmEeROc7badbize	2.079322446s

Tabel diatas menunjukkan hasil pengolahan kata sandi yang sama menjadi *hash* dengan jumlah *cost* yang berbeda, jumlah *cost* berdampak pada waktu yang digunakan oleh sistem untuk memperoleh *hash* dari kata sandi, semakin tinggi nilai *cost* yang diberikan maka waktu yang dibutuhkan semakin banyak. Nilai *cost* yang biasanya digunakan adalah 10 dengan nilai minimal 4 dan maksimalnya adalah 14. Go sebagai bahasa pemrograman yang digunakan untuk pengujian ini memiliki kecepatan yang baik dimana untuk nilai *cost* maksimal hanya membutuhkan waktu sekitar 2 detik.

Dibawah ini daftar kata sandi dengan berbagai jenis kombinasi dengan panjang karakter minimal 8 yang dapat digunakan untuk menguji waktu yang dibutuhkan oleh sistem untuk menghasilkan *hash* dengan nilai *cost* maksimal yaitu 14.

TABEL 2
PENGUJIAN WAKTU

Kata Sandi	*Cost*	*Hash*	Waktu
zcVcs2Sf	14	$2a$14$IxEaXNBTtBFYSbIfJF/Y/eOsyM5MFkFDdff61arIQtJ/LaTDgmeCa	2.084519861s

\3g%9-6OSY~a	14	$2a$14$OlmO30vq2ErDYh.xQ.ZUtuMmL1n05PTbeKz1Fki3s57uVlCntzf3O	2.081127304s
k3"}g-MY267/8gq(\Sr	14	$2a$14$.wRDsPkVm5sgG0HhoQz0xe3J.QVjkNuWtZXfLbwvfaNdKusObtMuO	2.081553736s
/<4%|}dTAXOU48X4sR/Pt;KB>b	14	$2a$14$GplEyXk/CqUoXEuZuOYVR.KdHfJAgEUQGhFgWC.mzZntQDuumVBoy	2.078920655s

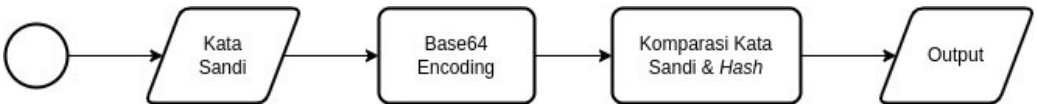
Dari hasil pengujian pada tabel diatas, waktu yang dibutuhkan oleh bahasa pemrograman Go untuk menghasilkan *hash* pada kata sandi dari yang paling mudah sampai pada kata sandi dengan panjang dan kombinasi karakter yang sangat kuat memberikan waktu yang konsisten yaitu sekitar 2 detik. Hal ini tentu saja akan menambah performa dari sistem apabila diimplementasikan ke dalam sistem berskala kecil hingga besar.

Adapun alur proses enkripsi/*hashing* kata sandi menggunakan *bcrypt* dan *base64* dilakukan dengan cara yang sederhana dan bekerja secara *synchronous* yaitu proses ini dilakukan secara bertahap, dimulai dengan mengirimkan *payload* berupa *Text Form* yang berisi teks kata sandi, selanjutnya *payload* kemudian akan disandikan oleh sistem menggunakan algoritma penyandian *(encoding) base64* sehingga menghasilkan teks acak dengan panjang karakter yang tetap. Hasil penyandian inilah yang kemudian diolah oleh algoritma *bcrypt* untuk menghasilkan *hash.*

Gambar 2.Alur proses enkripsi kata sandi menggunakan algoritma *bcrypt* dan *base64*

B. Dekripsi

Pengujian dekripsi dilakukan dengan cara membandingkan (komparasi) kata sandi dengan *hash* yang dihasilkan. Proses komparasi kata sandi dengan *hash* dilakukan dengan alur yang hampir sama dengan proses enkripsi, perbedaannya adalah apabila proses hashing menghasilkan keluaran berupa teks acak dan dapat dapat disimpan, sedangkan proses komparasi mengambil kata sandi dan hasil *hash* yang telah tersimpan untuk kemudian dibandingkan, keluaran dari proses ini berupa *boolean* yang bernilai benar (*true*) atau salah (*false*).



Gambar 3.Alur proses dekripsi kata sandi menggunakan algoritma *bcrypt* dan *base64*

Apabila hasil komparasi yang keluar bernilai salah maka kata sandi dan hash dianggap tidak memiliki kesamaan, sedangkan untuk hasil bernilai benar menyatakan bahwa kata sandi yang dibandingkan benar adanya. Pada pengujian dekripsi akan ditampilkan teks yang merepresentasikan nilai keluaran yang dihasilkan pada proses komparasi, seperti dibawah ini.

TABEL 3
REPRESENTASI NILAI HASIL KOMPARASI

Nilai	Teks
True	Cocok
False	Tidak Cocok
Error	Terjadi Kesalahan

Untuk nilai dengan nilai *Error* menandakan bahwa telah terjadi kesalahan pada sistem ketika proses komparasi berlangsung, Pada bahasa pemrograman Go, setiap langkah yang menghasilkan *error* harus ditangani secara manual dan sebaiknya ditampilkan secara jelas baik pada tampilan aplikasi, respon ataupun pada *log*. Namun demi kesederhanaan sistem yang dibangun, saat ini semua pesan kesalahan akan diwakilkan oleh teks “Terjadi Kesalahan”.

Untuk pengujian dekripsi/komparasi penulis akan menampilkan hasil komparasi baik dengan algoritma *base64* dan tidak. Dibawah ini merupakan hasil komparasi kata sandi dari yang paling mudah dan singkat hingga kata sandi dengan kombinasi yang rumit dan panjang berdasarkan tabel 2.

TABEL 4
PENGUJIAN DEKRIPSI KATA SANDI

Kata Sandi	*Base64*	*Hash*	Waktu	Output
zcVcs2Sf	Aktif	$2a$14$IxEaXNBTtBFYSbIfJF/Y/eOsyM5MFkFDdff61arIQtJ/LaTDgmeCa	2.082501197s	Cocok
zcVcs2Sf	Tidak Aktif	$2a$14$IxEaXNBTtBFYSbIfJF/Y/eOsyM5MFkFDdff61arIQtJ/LaTDgmeCa	2.08405743s	Tidak Cocok
\3g%9-6OSY~a	Aktif	$2a$14$OlmO30vq2ErDYh.xQ.ZUtuMmL1n05PTbeKz1Fki3s57uVlCntzf3O	2.081027433s	Cocok
\3g%9-6OSY~a	Tidak Aktif	$2a$14$OlmO30vq2ErDYh.xQ.ZUtuMmL1n05PTbeKz1Fki3s57uVlCntzf3O	2.087668474s	Tidak Cocok
k3"}g-MY267/8gq(\Sr	Aktif	$2a$14$.wRDsPkVm5sgG0HhoQz0xe3J.QVjkNuWtZXfLbwvfaNdKusObtMuO	2.089078224s	Cocok
k3"}g-MY267/8gq(\Sr	Tidak Aktif	$2a$14$.wRDsPkVm5sgG0HhoQz0xe3J.QVjkNuWtZXfLbwvfaNdKusObtMuO	2.081059799s	Tidak Cocok
/<4%|}dTAXOU48X4sR/Pt;KB>b	Aktif	$2a$14$GplEyXk/CqUoXEuZuOYVR.KdHfJAgEUQGhFgWC.mzZntQDuumVBoy	2.094909255s	Cocok
/<4%|}dTAXOU48X4sR/Pt;KB>b	Tidak Aktif	$2a$14$GplEyXk/CqUoXEuZuOYVR.KdHfJAgEUQGhFgWC.mzZntQDuumVBoy	2.086243617s	Tidak Cocok

Berdasarkan tabel diatas dengan mengaktifkan algoritma *base64* maka komparasi akan menghasilkan nilai "Cocok" hal ini membuktikan bahwa penggunaan algoritma *base64* tidak akan merusak ataupun mengubah nilai orisinil dari kata sandi yang akan enkripsi, sebaliknya kata sandi yang tidak mengaktifkan algoritma *base64* akan menghasilkan teks "Tidak Cocok", dengan *output* tersebut maka dapat dipastikan sistem bekerja sesuai dengan kehendak dan tidak mengalami kesalahan.

Adapun waktu yang dibutuhkan oleh sistem untuk menyelesaikan proses dekripsi berdasarkan aktif atau tidaknya algoritma *base64* berkisar sekitar 2 detik, hal ini didasarkan pada fakta bahwa semua kata sandi yang telah dienkripsi menggunakan nilai *cost* maksimum yaitu 14, sehingga wajar apabila waktu yang dibutuhkan untuk melakukan dekripsi kata sandi memiliki persamaan dengan waktu yang digunakan untuk enkripsi.

Untuk perbandingan waktu antara kata sandi yang dikomparasi dengan mengaktifkan algoritma *base64* dan tidak dapat dilihat lebih jelas pada tabel dibawah.

TABEL 5
WAKTU DEKRIPSI

Base64	Rata-rata
Aktif	2.08687902725s
TIdak Aktif	2.08475733s

Berdasarkan tabel diatas didapatkan data yaitu rata-rata waktu yang dibutuhkan untuk melakukan komparasi dengan mengaktifkan algoritma *base64* lebih tinggi dibandingkan dengan proses yang menonaktifkan algoritma tersebut, namun perlu dicatat bahwa perbedaan waktu yang tercatat tidaklah mencolok karena hanya berbeda beberapa *milisecond* dan perbedaan tersebut bisa jadi tidak akan disadari secara langsung.

IV. SIMPULAN

Penggunaan algoritma penyandian *base64* pada enkripsi kata sandi menggunakan algoritma *bcrypt* dapat menambah lapisan keamanan pada kata sandi sehingga tidak mudah untuk didekripsi atau diretas. Dari segi efisiensi waktu penambahan algoritma *base64* tidak menambah durasi waktu yang digunakan dalam melakukan enkripsi dan dekripsi kata sandi bahkan dengan pemakaian nilai *cost* maksimum, hal ini disebabkan karena implementasi sistem yang dibuat menggunakan bahasa pemrograman Go.

UCAPAN TERIMA KASIH

Judul untuk bagian ucapan terima kasih dan daftar pustaka tidak perlu dinomori dan dibuat rata tengah. Ucapan terima kasih biasanya diberikan pada institusi atau perusahaan yang mendanai riset anda.

DAFTAR PUSTAKA

[1]“Giffary and Ramadhani - 2022 - Implementasi Bcrypt dengan SHA-256 pada Password P.pdf.”

[2]“Anggoro et al. - 2019 - PENERAPAN ALGORITMA KNAPSACK DAN FUNGSI HASH PADA .pdf.”

[3]T. P. Batubara, S. Efendi, and E. B. Nababan, “Analysis Performance BCRYPT Algorithm to Improve Password Security from Brute Force,” J. Phys.: Conf. Ser., vol. 1811, no. 1, p. 012129, Mar. 2021, doi: 10.1088/1742-6596/1811/1/012129.

[4]“Musadat and Nur - 2018 - APLIKASI KRIPTOGRAFI KEAMANAN DATA MENGGUNAKAN ALG.pdf.”