

Analisis Ketahanan *SHA-256+ Salt* dan *Bcrypt* terhadap *Offline Dictionary Attack* dan *Brute Force Attack* pada Sistem Autentikasi

Keisha Tulauditta^{*1}, Muhammad Rizky Ramadhan², Muhammad Fahmi Hamzah³, I Wayan Rangga Pinastawa⁴

^{1,2,3,4}Program Studi Informatika, Universitas Pembangunan Nasional “Veteran” Jakarta

E-mail: ^{*1}2410511001@mahasiswa.upnvj.ac.id, ²2410511016@mahasiswa.upnvj.ac.id,

³2410511021@mahasiswa.upnvj.ac.id, ⁴rangga@upnvj.ac.id

Abstrak

Keamanan penyimpanan password merupakan aspek penting dalam sistem autentikasi karena kebocoran hash password dapat dimanfaatkan untuk melakukan serangan offline password cracking. Penelitian ini bertujuan menganalisis dan membandingkan ketahanan *SHA-256 + Salt*, *bcrypt cost 10*, dan *bcrypt cost 12* terhadap *dictionary attack* dan *brute force attack*. Hash password diperoleh dari sistem autentikasi berbasis web, kemudian digunakan sebagai target pengujian secara offline. Pengujian dilakukan terhadap tiga kategori kompleksitas password, yaitu *low*, *medium*, dan *high*. *Dictionary attack* menggunakan wordlist berukuran 100, 500, dan 1.000 kandidat, sedangkan *brute force attack* dibatasi hingga 1.000 percobaan untuk setiap target. Evaluasi dilakukan berdasarkan waktu pengujian, jumlah percobaan, keberhasilan serangan, *attempts per second*, penggunaan CPU dan memori, serta estimasi waktu eksplorasi ruang pencarian. Hasil *dictionary attack* menunjukkan bahwa seluruh password berhasil ditemukan karena tersedia dalam wordlist, tetapi *bcrypt* membutuhkan waktu verifikasi lebih tinggi dibandingkan *SHA-256 + Salt*. Pada *brute force attack*, seluruh target belum ditemukan hingga batas 1.000 percobaan. *SHA-256 + Salt* menghasilkan throughput lebih dari dua juta percobaan per detik, sedangkan *bcrypt cost 10* berada pada kisaran 15–18 percobaan per detik dan *bcrypt cost 12* sekitar 3–5 percobaan per detik. Hasil penelitian menunjukkan bahwa *bcrypt cost 12* memberikan ketahanan tertinggi di antara konfigurasi yang diuji, dengan konsekuensi biaya komputasi yang lebih besar.

Kata kunci - *SHA-256*, *Bcrypt*, *Brute Force Attack*, *Dictionary Attack*, *Offline Password Cracking*

1. PENDAHULUAN

Perkembangan teknologi informasi telah meningkatkan penggunaan sistem autentikasi digital dalam berbagai layanan berbasis komputer dan internet. Password masih menjadi salah satu metode autentikasi yang umum digunakan karena mudah diterapkan pada sistem informasi. Namun, penyimpanan password yang tidak aman dapat meningkatkan risiko kebocoran data dan serangan seperti *brute force attack* dan *dictionary attack*. Oleh karena itu, diperlukan mekanisme perlindungan password melalui teknik hashing [1]. Sohidin et al. menjelaskan bahwa lemahnya perlindungan password pada sistem informasi dapat menyebabkan kebocoran dan penyalahgunaan data [2]. Dalam konteks aplikasi web, serangan berbasis password dapat mengakibatkan akses tidak sah terhadap informasi sensitif sehingga penerapan hashing pada proses penyimpanan dan verifikasi password menjadi bagian penting dalam sistem autentikasi [16].

Hashing merupakan proses satu arah yang mengubah data menjadi nilai hash sehingga sulit dikembalikan ke bentuk aslinya. Salah satu algoritma yang banyak digunakan adalah *SHA-256*

karena memiliki proses komputasi yang cepat dan menghasilkan hash dengan panjang tetap [3]. Penelitian Sitorus et al. menunjukkan bahwa SHA-256 memiliki keseimbangan antara efisiensi dan keamanan pada berbagai aplikasi keamanan data [4]. Namun, karakteristik komputasinya yang cepat dapat menjadi kelemahan ketika digunakan untuk penyimpanan password karena memungkinkan kandidat password diuji dengan cepat [5]. Ngemba et al. juga menyebutkan bahwa SHA-256 memiliki keamanan lebih baik dibandingkan MD5, tetapi tetap membutuhkan mekanisme tambahan untuk meningkatkan perlindungan [6]. Penelitian Batubara dan Aritonang melakukan pencocokan terhadap hash menggunakan teknik *brute force* atau *dictionary attack* serta mengukur waktu yang diperlukan untuk menemukan kembali *plaintext*, sehingga waktu serangan dapat digunakan sebagai salah satu parameter untuk mengevaluasi ketahanan hash [17].

Berbeda dengan SHA-256, bcrypt dirancang untuk penyimpanan password dengan mekanisme *salt* dan *adaptive cost factor* yang meningkatkan biaya komputasi proses hashing [5], [7]. Penelitian sebelumnya menunjukkan bahwa bcrypt dapat meningkatkan ketahanan password terhadap proses *cracking* karena setiap kandidat memerlukan biaya komputasi yang lebih tinggi [8]. Zulma et al. menjelaskan bahwa peningkatan jumlah ronde hashing pada bcrypt dapat meningkatkan ketahanan terhadap *brute force attack* dan *rainbow table attack* [9]. Ramadhan et al. juga menjelaskan bahwa bcrypt menggunakan *salt* dan *cost*, dengan peningkatan *cost* menyebabkan proses hashing membutuhkan waktu lebih lama [16]. Oleh karena itu, penelitian ini menggunakan cost factor 10 dan 12 sebagai dua tingkat biaya komputasi untuk menganalisis pengaruh peningkatan *cost* terhadap kecepatan verifikasi kandidat dan penggunaan sumber daya. Penelitian lain juga menunjukkan penerapan bcrypt pada keamanan password dan sistem login berbasis web [10], [12].

Selain bcrypt, Argon2 merupakan algoritma *password hashing* modern yang memiliki karakteristik *memory-hard*, sehingga meningkatkan kebutuhan memori dan dirancang untuk meningkatkan ketahanan terhadap serangan berbasis GPU maupun perangkat keras khusus [18]. Saputra dan Akbi membandingkan bcrypt, Argon2, dan PBKDF2 pada autentikasi berbasis web berdasarkan waktu hashing, waktu verifikasi, penggunaan sumber daya, dan *dictionary attack*. Hasil penelitian tersebut menunjukkan bahwa Argon2 menghasilkan waktu pemrosesan lebih tinggi dan ketahanan *dictionary attack* yang lebih baik pada lingkungan pengujian mereka, sedangkan bcrypt menjadi alternatif yang lebih seimbang antara keamanan dan performa [18]. Meskipun demikian, penelitian ini difokuskan pada SHA-256 + Salt dan bcrypt untuk membandingkan karakteristik fungsi hash yang cepat dengan *password hashing* berbasis *adaptive cost* melalui skenario pengujian yang sama.

Beberapa penelitian terdahulu telah membahas implementasi bcrypt pada sistem autentikasi dan keamanan password [2], [13], serta membandingkan algoritma hashing terhadap serangan berbasis password [14]. Penelitian lain memanfaatkan SHA-256 dalam konteks keamanan dan integritas data [15], sementara penelitian terbaru telah membandingkan bcrypt dengan algoritma modern seperti Argon2 dan PBKDF2 pada autentikasi berbasis web [18]. Penelitian Saputra dan Akbi menunjukkan bahwa perbedaan karakteristik algoritma dapat memengaruhi performa, penggunaan sumber daya, dan ketahanan terhadap serangan password. Berdasarkan penelitian terdahulu tersebut, penelitian ini melakukan pengujian komparatif SHA-256 + Salt, bcrypt cost 10, dan bcrypt cost 12 pada lingkungan yang sama terhadap *offline dictionary attack* dan *offline brute force attack*, dengan mempertimbangkan waktu serangan, kecepatan pengujian kandidat, penggunaan sumber daya, dan kompleksitas password.

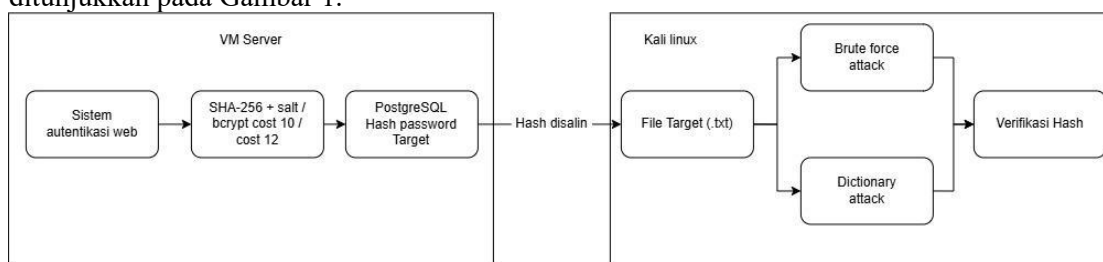
Berdasarkan permasalahan tersebut, penelitian ini bertujuan untuk menganalisis dan membandingkan ketahanan SHA-256 + Salt, bcrypt cost 10, dan bcrypt cost 12 terhadap *offline dictionary attack* dan *offline brute force attack* menggunakan hash password yang dihasilkan oleh sistem autentikasi berbasis web. Evaluasi dilakukan berdasarkan waktu pengujian, jumlah percobaan, *attempts per second*, penggunaan CPU dan memori, keberhasilan serangan, serta estimasi waktu eksplorasi ruang pencarian. Hasil penelitian diharapkan dapat memberikan gambaran mengenai *trade-off* antara ketahanan terhadap serangan password dan kebutuhan komputasi sebagai pertimbangan dalam pemilihan mekanisme penyimpanan password.

2. METODE PENELITIAN

2.1 Metode dan Lingkungan Penelitian

Penelitian ini menggunakan metode eksperimen komparatif untuk membandingkan ketahanan SHA-256 + Salt, bcrypt cost 10, dan bcrypt cost 12 terhadap *offline dictionary attack* dan *offline brute force attack*. Pengujian dilakukan menggunakan sistem autentikasi berbasis web yang dijalankan pada VM Server dan PostgreSQL sebagai basis data, sedangkan Kali Linux digunakan sebagai lingkungan pengujian serangan. Seluruh lingkungan berada pada jaringan internal yang terisolasi dan digunakan khusus untuk keperluan eksperimen. Pada SHA-256, password dikombinasikan dengan *salt* acak sebesar 16 byte sebelum dilakukan proses hashing dan disimpan dalam format yang memuat *salt* serta nilai hash. Sementara itu, bcrypt menggunakan mekanisme *salt* internal dengan variasi *cost factor* 10 dan 12. Sebelum pengujian serangan dilakukan, fungsi registrasi dan login diuji untuk memastikan bahwa proses penyimpanan dan verifikasi password pada sistem autentikasi berjalan sesuai dengan algoritma yang dikonfigurasi.

Pengujian serangan menggunakan skenario *offline password cracking* yang merepresentasikan kondisi ketika hash password pengguna telah diperoleh dari penyimpanan sistem. Hash password yang dihasilkan pada VM Server disalin dan disimpan dalam file target pada Kali Linux. Selanjutnya, kandidat password pada *dictionary attack* dan *brute force attack* diverifikasi secara langsung terhadap hash target tanpa melakukan permintaan autentikasi ke halaman login atau *endpoint* sistem. Dengan demikian, pengujian difokuskan pada ketahanan masing-masing mekanisme hashing ketika hash password telah terekspos. Arsitektur pengujian ditunjukkan pada Gambar 1.



Gambar 1. Arsitektur Lingkungan Pengujian

Berdasarkan Gambar 1, VM Server tidak berperan sebagai target yang menerima serangan secara langsung, tetapi sebagai lingkungan implementasi sistem autentikasi yang menghasilkan hash password target. Hash tersebut kemudian disalin ke file target pada Kali Linux untuk dilakukan pengujian secara *offline*. Dengan skenario tersebut, penelitian berfokus pada evaluasi ketahanan mekanisme penyimpanan password ketika nilai hash telah terekspos. Pendekatan yang sama diterapkan pada ketiga konfigurasi algoritma agar hasil pengujian dapat dibandingkan pada kondisi yang setara.

2. 2 Skenario dan Parameter Pengujian

Pengujian menggunakan sembilan password target yang dibagi menjadi tiga kategori kompleksitas, yaitu *low*, *medium*, dan *high*, dengan masing-masing kategori terdiri atas tiga password. Password target yang sama digunakan pada SHA-256 + Salt, bcrypt cost 10, dan bcrypt cost 12 sehingga perbedaan hasil pengujian tidak disebabkan oleh penggunaan password target yang berbeda. Pada *dictionary attack*, kandidat password diperoleh dari tiga ukuran *wordlist*, yaitu 100, 500, dan 1.000 kandidat. Untuk menjaga konsistensi pengujian, password target ditempatkan mendekati bagian akhir *wordlist*, yaitu pada posisi 92–100 untuk *wordlist* 100, posisi 492–500 untuk *wordlist* 500, dan posisi 992–1.000 untuk *wordlist* 1.000. Kandidat dibaca secara berurutan dan diverifikasi terhadap hash target sampai ditemukan kecocokan. Parameter yang dicatat meliputi jumlah percobaan, waktu yang diperlukan untuk menemukan password, dan keberhasilan serangan.

Pada *brute force attack*, kandidat tidak berasal dari *wordlist*, melainkan dibangkitkan berdasarkan kombinasi karakter sesuai kategori password. Kategori *low* menggunakan kombinasi

huruf kecil dan angka dengan panjang enam karakter, sedangkan kategori *medium* menggunakan huruf besar dan huruf kecil dengan panjang enam karakter. Kategori *high* menggunakan kombinasi huruf besar, huruf kecil, angka, dan karakter khusus dengan panjang enam atau sembilan karakter sesuai password target. Setiap pengujian *brute force* dibatasi hingga maksimum 1.000 percobaan. Apabila password belum ditemukan setelah batas tersebut tercapai, hasil dicatat sebagai *limit reached* dan tidak diartikan bahwa password tidak dapat ditemukan apabila ruang pencarian dilanjutkan. Ringkasan skenario dan parameter yang digunakan dalam pengujian ditunjukkan pada Tabel 1.

Tabel 1. Skenario dan Parameter Pengujian

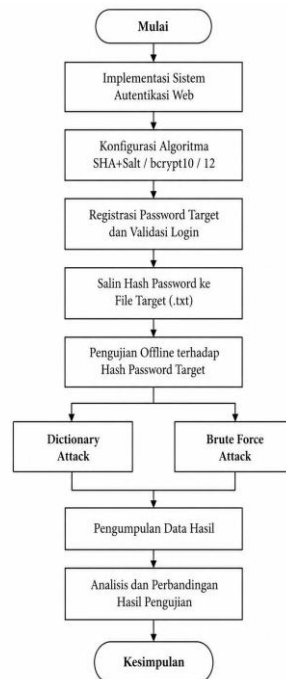
Parameter	Dictionary Attack	Brute Force Attack
Target Pengujian	Hash password	Hash password
Metode kandidat	Kandidat dari <i>wordlist</i>	Kombinasi karakter
Algoritma	SHA-256 + Salt, bcrypt-10, bcrypt-12	SHA-256 + Salt, bcrypt-10, bcrypt-12
Kompleksitas	<i>Low, medium, high</i>	<i>Low, medium, high</i>
Target per kategori	3 password	3 password
Ukuran <i>wordlist</i>	100, 500, 1000	-
Batas <i>brute force</i>	-	1000 percobaan
Parameter pengukuran	Percobaan, waktu, keberhasilan	Waktu, <i>attempts/s</i> , CPU, memori
Analisis tambahan	-	Estimasi eksplorasi ruang pencarian

Berdasarkan Tabel 1, kedua metode serangan menggunakan hash password yang sama sebagai target, tetapi berbeda dalam proses pembentukan kandidat. *Dictionary attack* menggunakan kandidat yang telah tersedia dalam *wordlist*, sedangkan *brute force attack* membangkitkan kandidat berdasarkan ruang karakter yang ditentukan. Penggunaan tiga kategori kompleksitas dan tiga konfigurasi algoritma memungkinkan pengujian dilakukan pada kondisi yang sama sehingga pengaruh algoritma, kompleksitas password, dan metode serangan dapat dianalisis secara lebih terstruktur.

2. 3 Prosedur Pengujian

Proses penelitian dimulai dengan implementasi sistem autentikasi berbasis web dan konfigurasi algoritma hashing yang akan diuji. Password target kemudian diregistrasikan pada sistem sehingga menghasilkan hash sesuai konfigurasi SHA-256 + Salt, bcrypt cost 10, atau bcrypt cost 12. Setelah fungsi registrasi dan login berhasil divalidasi, hash password target yang dihasilkan sistem disalin dan disimpan dalam file target pada Kali Linux. Pada pengujian *dictionary attack*, program membaca kandidat password dari *wordlist* secara berurutan. Untuk target SHA-256 + Salt, *salt* yang terdapat pada data target digunakan untuk menghitung SHA-256 dari setiap kandidat, kemudian hasilnya dibandingkan dengan hash target. Pada bcrypt, setiap kandidat diverifikasi terhadap hash bcrypt sesuai parameter yang tersimpan di dalam hash. Proses dilanjutkan sampai ditemukan kandidat yang sesuai dengan password target. Waktu pengujian dan jumlah percobaan pada saat password ditemukan kemudian dicatat sebagai hasil pengujian.

Pada pengujian *brute force attack*, program membangkitkan kandidat berdasarkan *charset* dan panjang password sesuai kategori pengujian. Kandidat tersebut diverifikasi secara langsung terhadap hash target menggunakan mekanisme yang sesuai dengan masing-masing algoritma. Proses dibatasi maksimum 1.000 percobaan untuk setiap target. Selama pengujian dicatat waktu eksekusi, jumlah percobaan per detik (*attempts per second*), penggunaan CPU, dan penggunaan memori. *Throughput* yang diperoleh selanjutnya digunakan untuk memperkirakan waktu yang diperlukan apabila seluruh ruang pencarian dieksplorasi dengan kecepatan yang sama. Keseluruhan prosedur penelitian disajikan pada Gambar 2.



Gambar 2. Tahapan Penelitian

Berdasarkan Gambar 2, pengujian serangan baru dilakukan setelah sistem autentikasi berhasil diimplementasikan dan fungsi registrasi serta login telah divalidasi. Hash yang dihasilkan sistem kemudian menjadi input pada tahap pengujian *offline*. Hasil *dictionary attack* dan *brute force attack* dikumpulkan untuk selanjutnya dibandingkan berdasarkan parameter yang telah ditentukan. Pemisahan antara tahap implementasi sistem dan tahap serangan juga memastikan bahwa waktu pengujian yang diperoleh merupakan waktu verifikasi kandidat terhadap hash secara *offline*, bukan waktu autentikasi *end-to-end* melalui jaringan.

2. 4 Variabel dan Analisis Data

Variabel bebas dalam penelitian meliputi algoritma hashing yang terdiri atas SHA-256 + Salt, bcrypt cost 10, dan bcrypt cost 12; jenis serangan yang terdiri atas *dictionary attack* dan *brute force attack*; kategori kompleksitas password; serta ukuran *wordlist* pada *dictionary attack*. Variabel terikat meliputi waktu pengujian, jumlah percobaan, keberhasilan menemukan password, *attempts per second*, penggunaan CPU, penggunaan memori, dan estimasi waktu eksplorasi ruang pencarian. Data *dictionary attack* dianalisis berdasarkan jumlah percobaan dan waktu yang diperlukan untuk menemukan password pada masing-masing ukuran *wordlist*. Karena setiap kategori terdiri atas tiga password target, hasil waktu pengujian dirata-ratakan berdasarkan kategori *low*, *medium*, dan *high* untuk mempermudah perbandingan antarkonfigurasi algoritma. Keberhasilan serangan juga dianalisis berdasarkan apakah kandidat password target tersedia dan berhasil ditemukan di dalam *wordlist*.

Pada *brute force attack*, hasil dianalisis berdasarkan waktu yang diperlukan untuk menjalankan maksimum 1.000 percobaan dan *throughput* dalam bentuk *attempts per second*. Penggunaan CPU dan memori digunakan untuk melihat kebutuhan sumber daya selama proses pengujian. Seluruh target yang belum ditemukan setelah 1.000 percobaan dinyatakan mencapai batas pengujian (*limit reached*), bukan sebagai kegagalan permanen untuk menemukan password.

Selain hasil aktual dari 1.000 percobaan, *throughput* pengujian *brute force* digunakan untuk mengestimasi waktu eksplorasi seluruh ruang pencarian. Ruang pencarian ditentukan berdasarkan panjang password dan jumlah karakter yang tersedia pada masing-masing kategori. Nilai tersebut digunakan untuk memperkirakan biaya waktu apabila seluruh kemungkinan kandidat diuji dengan kecepatan yang sama seperti pada eksperimen. Oleh karena itu, estimasi eksplorasi ruang pencarian merupakan proyeksi berdasarkan hasil pengujian dan bukan waktu cracking aktual.

3. HASIL DAN PEMBAHASAN

3. 1 Hasil Pengujian Fungsional Sistem

Sebelum pengujian serangan dilakukan, sistem autentikasi diuji melalui proses registrasi dan login secara manual. Pengujian ini bertujuan memastikan bahwa password dapat diproses menggunakan konfigurasi hashing yang dipilih, disimpan ke dalam PostgreSQL, dan diverifikasi kembali ketika pengguna melakukan login. Hasil Pengujian menunjukkan bahwa proses registrasi dan login dapat berjalan dengan baik pada ketiga konfigurasi, yaitu SHA-256 + salt, bcrypt-10, dan bcrypt-12.

Pada implementasi SHA-256, setiap password disertai *random salt* disimpan bersama hasil hash. Sementara itu, bcrypt menghasilkan hash yang telah memuat informasi salt dan *cost factor*. Setelah fungsi autentikasi berhasil diverifikasi, hash password yang dihasilkan oleh sistem pada masing-masing konfigurasi disalin dan disimpan dalam file target untuk digunakan pada pengujian *offline dictionary attack* dan *offline brute force attack*.

3. 2 Hasil Dictionary Attack

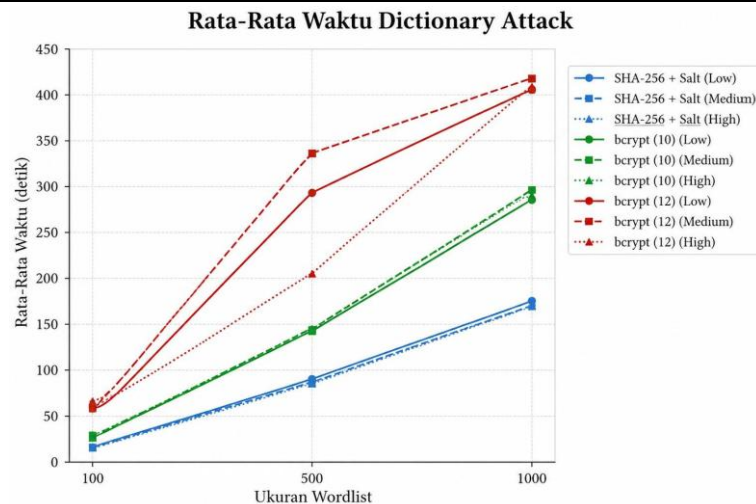
Pengujian *dictionary attack* dilakukan terhadap SHA-256 + Salt, bcrypt cost 10, dan bcrypt cost 12 menggunakan *wordlist* berukuran 100, 500, dan 1.000 kandidat. Setiap algoritma diuji terhadap tiga kategori kompleksitas password (*low*, *medium*, dan *high*), dengan masing-masing kategori terdiri atas tiga password target. Untuk menyederhanakan penyajian, waktu pengujian dirata-ratakan berdasarkan tiga password pada setiap kategori. Password target ditempatkan mendekati bagian akhir *wordlist*, yaitu pada posisi 92–100, 492–500, dan 992–1.000. Seluruh password berhasil ditemukan karena tersedia di dalam *wordlist*.

Tabel 2. Rata-Rata Waktu *Dictionary Attack*

Algoritma	Kompleksitas	Wordlist 100(s)	Wordlist 500(s)	Wordlist 1000(s)
SHA-256 + Salt	Low	16,53	89,44	174,49
	Medium	16,83	86,55	170,86
	High	17,02	87,18	174,90
Bcrypt (10)	Low	27,33	143,68	286,34
	Medium	30,89	145,91	296,28
	High	30,69	146,19	293,28
Bcrypt(12)	Low	57,14	292,74	407,20
	Medium	58,57	335,19	416,23
	High	63,58	204,37	407,80

Berdasarkan Tabel 2, peningkatan ukuran *wordlist* secara umum meningkatkan waktu serangan karena umlah kandidat yang harus diuji dan diverifikasi terhadap hash target semakin banyak. SHA-256 + Salt cenderung menghasilkan waktu terendah, diikuti bcrypt cost 10 dan bcrypt cost 12. Sebagai contoh, pada *wordlist* 100 kategori *low*, rata-rata waktu meningkat dari 16,53 detik pada SHA-256 + Salt menjadi 27,33 detik pada bcrypt cost 10 dan 57,14 detik pada bcrypt cost 12. Hal ini menunjukkan bahwa peningkatan *cost factor* bcrypt meningkatkan biaya komputasi dalam proses verifikasi kandidat.

Perbedaan kategori *low*, *medium*, dan *high* tidak menunjukkan peningkatan waktu yang selalu konsisten. Pada bcrypt cost 12 dengan *wordlist* 500, misalnya, kategori *high* menghasilkan rata-rata 204,37 detik, lebih rendah dibandingkan *low* sebesar 292,74 detik dan *medium* sebesar 335,19 detik. Hal ini menunjukkan bahwa kompleksitas karakter password tidak secara langsung menentukan waktu *dictionary attack* apabila password telah tersedia dalam *wordlist*. Jumlah percobaan terutama ditentukan oleh posisi password dalam daftar kandidat, sedangkan perbedaan waktu pada jumlah percobaan yang berdekatan dapat dipengaruhi oleh variasi kondisi eksekusi sistem. Untuk memperjelas kecenderungan waktu serangan pada masing-masing algoritma dan ukuran *wordlist*, hasil rata-rata divisualisasikan pada Gambar 4.



Gambar 4. Perbandingan Rata-Rata Waktu *Dictionary Attack*

Gambar 4 memperlihatkan kecenderungan peningkatan waktu *dictionary attack* seiring bertambahnya ukuran *wordlist*. bcrypt cost 12 secara umum menghasilkan waktu serangan tertinggi, sedangkan SHA-256 + Salt menghasilkan waktu terendah. Perbedaan tersebut menunjukkan bahwa mekanisme *cost factor* pada bcrypt memperbesar biaya verifikasi setiap kandidat sehingga memperlambat proses serangan. Meskipun demikian, seluruh password tetap berhasil ditemukan karena tersedia di dalam *wordlist*, sehingga penggunaan algoritma hashing yang lebih lambat meningkatkan biaya serangan tetapi tidak menghilangkan risiko terhadap password yang terdapat dalam kamus serangan. Waktu yang dicatat pada pengujian merupakan waktu proses *offline dictionary attack*, yaitu waktu yang diperlukan untuk membaca kandidat dari *wordlist* dan memverifikasinya secara langsung terhadap hash target. Oleh karena itu, waktu pengujian tidak mencakup komunikasi jaringan, akses *endpoint* login, maupun pemrosesan autentikasi pada VM Server.

3. 3 Hasil Brute Force Attack

Pengujian *brute force* dilakukan dengan batas maksimum 1000 percobaan terhadap setiap password target. Tidak terdapat password yang berhasil ditemukan dalam batas tersebut sehingga seluruh pengujian menghasilkan status *limit reached*. Kondisi tersebut tidak menunjukkan bahwa password tidak dapat dipecahkan, tetapi hanya menunjukkan bahwa kandidat yang sesuai belum ditemukan dalam 1.000 percobaan yang dilakukan. Perbedaan yang sangat jelas ditemukan pada kecepatan pengujian ketiga algoritma. Untuk memperoleh gambaran yang ringkas, Tabel 3 menampilkan rata-rata *attempts per second* berdasarkan tiga password pada setiap kategori.

Tabel 3. Rata-Rata Kecepatan *Brute Force*

Algoritma	Low	Medium	High
SHA-256 + Salt	≈2.413.056/s	≈2.292.256/s	≈2.281.216/s
Bcrypt-10	≈17,83/s	≈16,84/s	≈15,20/s
Bcrypt-12	≈3,76/s	≈4,33/s	≈3,65/s

Hasil tersebut memperlihatkan perbedaan throughput yang sangat besar. SHA-256 + Salt mampu melakukan lebih dari dua juta percobaan per detik pada lingkungan pengujian, sedangkan bcrypt cost 10 hanya berada pada kisaran 15–18 percobaan per detik dan bcrypt cost 12 sekitar 3–5 percobaan per detik. Penambahan salt pada SHA-256 berperan dalam menghasilkan hash yang berbeda untuk password yang sama ketika salt berbeda. Namun, ketika salt dan hash target telah tersedia pada skenario pengujian *offline*, kandidat tetap dapat diverifikasi dengan menghitung SHA-256 menggunakan salt tersebut. Oleh karena itu, kecepatan dasar SHA-256 masih menghasilkan throughput yang tinggi pada pengujian ini.

Sebaliknya, bcrypt sengaja dirancang dengan proses yang lebih mahal secara komputasi. Hasil penelitian menunjukkan bahwa peningkatan *cost factor* dari 10 menjadi 12 semakin menurunkan jumlah kandidat yang dapat diverifikasi per detik. Dengan demikian, penyerang membutuhkan waktu yang lebih panjang untuk menguji jumlah kandidat yang sama. Perbedaan tersebut juga terlihat dari waktu yang diperlukan untuk menyelesaikan 1.000 percobaan. SHA-256 + Salt hanya memerlukan waktu kurang dari satu milidetik pada pengujian lokal, sedangkan bcrypt cost 10 membutuhkan sekitar 56–69 detik dan bcrypt cost 12 membutuhkan sekitar 208–361 detik tergantung target yang diuji.

3.4 Penggunaan CPU dan Memori

Selain kecepatan serangan, pengujian *brute force* juga mengukur penggunaan CPU dan memori proses. Pada bcrypt cost 10, penggunaan CPU selama pengujian berada pada kisaran 99–100%, dan pola serupa juga ditemukan pada bcrypt cost 12. Hasil tersebut menunjukkan bahwa proses verifikasi bcrypt memberikan beban komputasi CPU yang tinggi selama pengujian *brute force* berlangsung. Penggunaan memori relatif stabil selama pengujian. Pada bcrypt cost 10 dan bcrypt cost 12, penggunaan memori proses berada di sekitar 14 MB dan hanya mengalami perubahan kecil antara awal dan akhir pengujian. Dengan demikian, pada konfigurasi pengujian yang digunakan, peningkatan *cost factor* bcrypt lebih terlihat pada peningkatan waktu komputasi, sedangkan penggunaan CPU pada kedua konfigurasi sama-sama mendekati 100% dan konsumsi memori relatif stabil. Pada SHA-256 + Salt, pengukuran CPU menghasilkan nilai 0% pada pengujian 1.000 percobaan. Nilai tersebut tidak menunjukkan bahwa proses SHA-256 tidak menggunakan sumber daya CPU. Durasi pengujian SHA-256 yang sangat singkat, yaitu kurang dari satu milidetik, diduga menyebabkan penggunaan CPU tidak tertangkap secara memadai oleh metode pengukuran yang digunakan. Oleh karena itu, nilai CPU SHA-256 tidak digunakan sebagai dasar untuk menyimpulkan bahwa algoritma tersebut tidak membutuhkan sumber daya CPU.

3.5 Estimasi Eksplorasi Ruang Pencarian

Throughput yang diperoleh dari *brute force attack* selanjutnya digunakan untuk memperkirakan waktu yang diperlukan apabila seluruh kombinasi dalam ruang pencarian diuji. Estimasi ini digunakan untuk menggambarkan pengaruh kompleksitas password dan kecepatan verifikasi algoritma terhadap besarnya biaya serangan.

Tabel 4. Estimasi Waktu Eksplorasi *Full Keyspace*

Algoritma	Low(6)	Medium (6)	High(6)	High(9)
SHA-256 + Salt	≈14–16 menit	≈2,4 jam	≈22–23 jam	≈1.273 tahun
Bcrypt (10)	≈3,8–3,9 tahun	≈36–40 tahun	≈384–424 tahun	≈176 juta tahun
Bcrypt(12)	≈14–25 tahun	≈143–148 tahun	≈1.394–2.004 tahun	≈760 juta tahun

Tabel 4 menunjukkan bahwa algoritma hashing dan kompleksitas password sama-sama memberikan pengaruh terhadap estimasi waktu eksplorasi ruang pencarian. Pada password dengan ruang pencarian yang sama, bcrypt menghasilkan estimasi yang jauh lebih tinggi dibandingkan SHA-256 + Salt karena throughput verifikasinya lebih rendah. Pengaruh panjang password juga terlihat sangat besar pada kategori *high*. Pada SHA-256 + Salt, password *high* sepanjang enam karakter masih memiliki estimasi eksplorasi penuh sekitar 22–23 jam, sedangkan password sepanjang sembilan karakter meningkat menjadi sekitar 1.273 tahun karena pertumbuhan jumlah kombinasi yang harus diuji.

Pola yang sama terlihat pada bcrypt. Untuk password *high* sepanjang enam karakter, estimasi bcrypt cost 10 berada pada ratusan tahun, sedangkan target sembilan karakter mencapai sekitar 176 juta tahun. Pada bcrypt cost 12, estimasi meningkat lebih jauh hingga sekitar 760 juta tahun untuk target sembilan karakter. Namun, angka tersebut tidak menunjukkan waktu aktual yang dibutuhkan untuk menemukan password. Nilai tersebut merupakan proyeksi waktu untuk

mengeksplorasi seluruh *keyspace* berdasarkan throughput yang diperoleh selama pengujian 1.000 kandidat. Password dapat ditemukan sebelum seluruh ruang pencarian selesai dieksplorasi tergantung posisi kandidat yang benar.

3.6 Pembahasan Perbandingan Algoritma

Secara keseluruhan, hasil kedua jenis serangan menunjukkan karakteristik yang berbeda antara SHA-256 + Salt dan bcrypt. Pada *dictionary attack*, seluruh password berhasil ditemukan karena kandidat tersedia di dalam wordlist. Meskipun demikian, bcrypt memperlambat proses serangan karena setiap kandidat memerlukan proses verifikasi yang lebih mahal dibandingkan SHA-256 + Salt. bcrypt cost 12 secara umum memberikan waktu pengujian paling tinggi, diikuti bcrypt cost 10 dan SHA-256 + Salt. Perbedaan menjadi lebih jelas pada pengujian *brute force* lokal. SHA-256 + Salt menghasilkan throughput lebih dari dua juta kandidat per detik, sedangkan bcrypt cost 10 hanya menghasilkan belasan kandidat per detik dan bcrypt cost 12 berada pada kisaran beberapa kandidat per detik. Hasil tersebut menunjukkan bahwa keberadaan salt pada SHA-256 tidak dengan sendirinya membuat proses verifikasi kandidat menjadi lambat. Salt terutama mencegah penggunaan hash yang sama untuk password identik dan menghambat penggunaan tabel hash yang telah dihitung sebelumnya, sedangkan bcrypt menambahkan biaya komputasi pada setiap percobaan password.

Kompleksitas password juga menjadi faktor penting. Pada *dictionary attack*, password dengan kategori kompleksitas tinggi tetap dapat ditemukan apabila tersedia di dalam wordlist. Sebaliknya, pada *brute force*, peningkatan jumlah karakter yang mungkin digunakan dan panjang password memperbesar ruang pencarian secara signifikan. Dengan demikian, ketahanan password tidak hanya ditentukan oleh algoritma hashing, tetapi juga oleh pemilihan password yang tidak mudah diprediksi serta memiliki ruang pencarian yang besar. Dari sisi performa, bcrypt membutuhkan sumber daya komputasi lebih tinggi dibandingkan SHA-256 + Salt, khususnya pada penggunaan CPU dan waktu verifikasi. Peningkatan cost dari 10 menjadi 12 semakin memperbesar biaya tersebut. Kondisi ini menunjukkan adanya *trade-off* antara keamanan dan biaya komputasi. Peningkatan *cost factor* memperlambat verifikasi kandidat pada pengujian ini, tetapi penerapannya pada sistem autentikasi juga perlu mempertimbangkan kemampuan sumber daya sistem.

Berdasarkan keseluruhan hasil, bcrypt cost 12 menunjukkan ketahanan tertinggi terhadap pengujian password guessing yang dilakukan, diikuti bcrypt cost 10, sedangkan SHA-256 + Salt memiliki throughput verifikasi paling tinggi sehingga memungkinkan pengujian kandidat secara jauh lebih cepat pada skenario *offline*. Namun, hasil ini berlaku pada konfigurasi perangkat keras, perangkat lunak, batas percobaan, dan skenario eksperimen yang digunakan dalam penelitian ini.

4. KESIMPULAN

Berdasarkan hasil pengujian, SHA-256 + Salt, bcrypt cost 10, dan bcrypt cost 12 menunjukkan perbedaan ketahanan terhadap *offline password cracking*. Pada *dictionary attack*, seluruh password target berhasil ditemukan karena telah tersedia di dalam *wordlist*. Peningkatan ukuran *wordlist* secara umum meningkatkan waktu serangan, sedangkan kompleksitas karakter password tidak secara langsung menentukan ketahanan apabila password telah tercantum dalam daftar kandidat. bcrypt membutuhkan waktu verifikasi yang lebih tinggi dibandingkan SHA-256 + Salt, terutama pada cost 12, sehingga meningkatkan biaya komputasi yang diperlukan untuk menguji setiap kandidat password.

Pada *brute force attack* dengan batas 1.000 percobaan, seluruh password target belum berhasil ditemukan. SHA-256 + Salt menghasilkan *throughput* pengujian paling tinggi, sedangkan bcrypt cost 10 dan bcrypt cost 12 menghasilkan *throughput* yang jauh lebih rendah dengan penggunaan CPU yang tinggi. Estimasi eksplorasi ruang pencarian menunjukkan bahwa peningkatan panjang dan kompleksitas password memperbesar waktu yang diperlukan untuk

menguji seluruh kemungkinan kombinasi. Berdasarkan keseluruhan pengujian *offline* terhadap hash target, bcrypt cost 12 menunjukkan ketahanan tertinggi di antara konfigurasi yang diuji, tetapi membutuhkan biaya komputasi yang lebih besar.

Penelitian ini memiliki keterbatasan berupa pengujian yang dilakukan secara *offline* terhadap hash password yang disalin dari sistem autentikasi, sehingga belum merepresentasikan serangan langsung terhadap layanan autentikasi pada lingkungan produksi. Pengujian juga menggunakan *wordlist* maksimum 1.000 kandidat, membatasi *brute force* hingga 1.000 percobaan, serta dilakukan pada konfigurasi perangkat keras dan perangkat lunak tertentu. Oleh karena itu, estimasi eksplorasi seluruh ruang pencarian merupakan proyeksi berdasarkan *throughput* pengujian dan bukan waktu *cracking* aktual. Berdasarkan *trade-off* keamanan dan komputasi yang diperoleh, bcrypt cost 10 dapat dipertimbangkan untuk sistem dengan sumber daya komputasi lebih terbatas, sedangkan bcrypt cost 12 lebih sesuai ketika prioritas keamanan lebih tinggi dan tersedia sumber daya yang memadai. SHA-256 + Salt memiliki proses verifikasi yang jauh lebih cepat, tetapi karakteristik tersebut juga memungkinkan kandidat diuji dengan kecepatan yang jauh lebih tinggi ketika hash dan *salt* telah diperoleh pada skenario *offline*, sehingga dalam konteks penelitian ini kurang direkomendasikan sebagai mekanisme utama penyimpanan password dibandingkan bcrypt.

5. SARAN

Penelitian selanjutnya disarankan menggunakan lingkungan pengujian dengan variasi spesifikasi perangkat keras dan kondisi sistem yang lebih beragam untuk mengetahui konsistensi performa masing-masing algoritma. Pengujian juga dapat dikembangkan dengan menggunakan *wordlist* yang lebih besar dan lebih representatif serta meningkatkan batas percobaan *brute force* agar pengukuran *throughput* dan estimasi waktu eksplorasi ruang pencarian dapat diperoleh dari jumlah percobaan yang lebih luas. Selain itu, pengujian dapat dilakukan secara berulang pada setiap skenario untuk mengurangi pengaruh variasi kondisi eksekusi dan memperoleh hasil statistik yang lebih representatif. Penelitian berikutnya juga dapat memperluas perbandingan dengan algoritma *password hashing* lain, seperti Argon2, serta mengevaluasi lebih lanjut pengaruh konfigurasi algoritma terhadap waktu autentikasi dan penggunaan sumber daya pada kondisi beban pengguna yang berbeda. Dengan pengembangan tersebut, penelitian selanjutnya diharapkan dapat memberikan rekomendasi konfigurasi *password hashing* yang lebih komprehensif berdasarkan kebutuhan keamanan dan kemampuan sumber daya sistem.

DAFTAR PUSTAKA

- [1] I. Algoritma *et al.*, “Implementation Of Bcrypt Algorithm On Website-Based Hashing Generator Using Laravel Framework,” vol. 7, no. 2, 2023, doi: 10.52362/jisicom.v7i2.1130.
- [2] S. Kasus, U. Langlangbuana, T. Informatika, F. Teknik, and U. Langlangbuana, “Penerapan Algoritma Bcrypt untuk Pengamanan Password pada Sistem Informasi Akademik (SIAK),” vol. 6, no. 2, pp. 1–5, 2025.
- [3] S. Teknik, “Analisis Penggunaan Fungsi Hash BCrypt untuk Keamanan Kata Sandi”.
- [4] A. Sha-, N. Sitorus, J. Sharon, G. Sinaga, and S. L. Samosir, “Analisis Kinerja Algoritma Hash pada Keamanan Data : Perbandingan,” vol. 2, no. 2, 2024.
- [5] H. Md *et al.*, “Perbandingan Waktu Pemecahan Password Menggunakan Algoritma yang perlu dikaji lebih lanjut , khususnya dalam konteks analisis empiris terhadap ketahanan efektif suatu algoritma hash dalam menahan serangan tersebut . Oleh karena itu , penelitian ini,” no. November 2025, 2026.
- [6] J. T. Informasi, F. Teknik, and U. Tadulako, “Implementasi Enkripsi Data MD5 dan SHA-256 pada Sistem Informasi Peminjaman Buku Tanah,” vol. 22, no. 3, pp. 654–664, 2023.
- [7] K. Nur, D. Suhartono, M. Thoriq, and A. Qothrunnada, “Implementasi Pengamanan Data Menggunakan Teknik Bcrypt Hashing Password dan Algoritma Advanced Encryption

-
- Standard (AES) Implementation of Data Security Using Bcrypt Hashing Password Technique and Advanced Encryption Standard (AES) Algorithm,” vol. 13, no. 1, pp. 101–108, 2025, doi: 10.26418/justin.v13i1.84997.
- [8] R. S. Giffary and E. Ramadhani, “Implementasi Bcrypt dengan SHA-256 pada Password Pengguna Aplikasi Golek Kost,” vol. 3, pp. 543–546, 2022, doi: 10.30865/json.v3i4.4285.
 - [9] G. Divva, M. Zulma, H. B. Seta, and T. Yuniati, “Implementasi Algoritma AES Dan Bcrypt untuk Pengamanan File Dokumen,” vol. 4221, pp. 163–176, 2022.
 - [10] R. M. Liauren, B. Zaman, and S. Bahri, “Implementasi Algoritma Aes Dan Bcrypt Untuk Pengamanan,” no. 01, pp. 57–71, 2025.
 - [11] A. H. Md, R. Dafi, A. Azhar, and I. S. Widiati, “Evaluasi Keamanan Penyimpanan Password Menggunakan,” pp. 1302–1305, 2025.
 - [12] Y. Arafat, T. Hidayat, and M. Khozin, “Bcrypt Dan Passkey Pada Sistem Login Web Di Smk Negeri 1 XYZ,” vol. 11, no. 1, pp. 70–76, 2026.
 - [13] N. D. Aji, T. T. Sujaka, O. Asroni, and K. A. Latif, “Analisis dan Implementasi Algoritma Bcrypt dengan Affine Cipher untuk Pengamanan Password pada Aplikasi Web Analysis and Implementation Of Bcrypt Algorithm with Affine Cipher for Password Security in Web Applications,” vol. 8, no. 1, pp. 1–9, 2025.
 - [14] P. Serangan, B. Kata, and S. Studi, “Tentang Implementasi Hash Pada Aplikasi Web Prevention Of Password-Based Attacks: A Comprehensive Study Of Hash Implementation In Web ApplicationS,” vol. 7, pp. 920–925, 2024.
 - [15] J. Hutagalung, P. S. Ramadhan, S. J. Sihombing, and P. Korespondensi, “Keamanan Data Menggunakan Secure Hashing Algorithm (SHA) - 256 Dan Rivest Shamir Adleman (RSA) Pada Digital Signature Implementation Of Secure Hashing Algorithm (SHA) -256 And Rivest Shamir Adleman (RSA) On Digital Signature,” vol. 10, no. 6, 2023, doi: 10.25126/jtiik.2023107319.
 - [16] M. A. Ramadhan, Arpinda, D. Saputra, and D. I. Mulyana, “Pencegahan serangan berbasis kata sandi: Studi komprehensif tentang implementasi hash pada aplikasi web,” *Journal of Information Technology and Computer Science (INTECOMS)*, vol. 7, no. 3, pp. 920–925, 2024.
 - [17] T. P. Batubara and M. A. S. Aritonang, “Perbandingan algoritma kriptografi hash SHA 256 dan SHA 512 untuk keamanan sandi,” *Jurnal Quancom*, vol. 3, no. 1, pp. 8–12, Jun. 2025, doi: 10.62375/jqc.v3i1.627.
 - [18] M. G. D. Saputra and D. R. Akbi, “Perbandingan Bcrypt, Argon2, dan PBKDF2 pada keamanan SIMPEG berbasis web,” *Jurnal Informatika: Jurnal Pengembangan IT*, vol. 11, no. 2, 2026, doi: 10.30591/jpit.v11i2.10280.
-