Analisis Keamanan Hash Password Berdasarkan Kasus Kebocoran Data LinkedIn

Nicholas Francis Aditjandra - 18221005[1,2]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]18221005@std.stei.itb.ac.id, [2]francisaditjandra@gmail.com

Abstrak*—Keamanan penyimpanan password merupakan salah satu aspek yang paling penting untuk sistem informasi modern. Meskipun begitu, banyaknya kasus kebocoran data telah menunjukkan bahwa mekanisme penyimpanan password yang salah justru dapat menimbulkan ancaman keamanan yang serius. Kebocoran data LinkedIn pada tahun 2012 adalah salah satu kasus kebocoran data terkait penyimpanan password yang paling terkenal. LinkedIn menyimpan password pengguna menggunakan fungsi hash SHA-1 tanpa salt, yang membuat penyimpanan yang tidak aman sehingga memungkinkan penyerang memecahkan jutaan password menggunakan rainbow table attack dan dictionary attack. Makalah ini akan berusaha menganalisis kelemahan dari pendekatan tersebut melalui eksperimen perbandingan metode hashing password tanpa salt, dengan salt, dengan iterasi, serta dengan salt dan iterasi. Hasil eksperimen menunjukkan bahwa penambahan salt dan iterasi berhasil meningkatkan ketahanan terhadap serangan rainbow table attack dan dictionary attack secara signifikan. Hasil ini menunjukkan betapa pentingnya menggunakan metode hashing password yang tepat untuk mencegah kompromi kredensial yang berulang dalam skala besar.

***Kata Kunci*—dictionary attack, fungsi hash, kebocoran data LinkedIn, keamanan password, rainbow table attack, salt**

I. Pendahuluan

Meskipun kasus kebocoran data dan kerentanan akibat penggunaan password telah banyak didokumentasi, nyatanya metode ini masih menjadi metode utama untuk mengamankan berbagai sistem informasi. Kelemahan dari sistem autentikasi berbasis password secara umum berasal dari dua sumber utama yaitu perilaku pengguna seperti pemilihan frasa yang lemah dan dapat diprediksi, serta dari aspek teknis penyimpanan dan pengelolaan password dari sistem informasi itu sendiri. Mengingat kesulitan dalam mengatur perilaku pengguna secara menyeluruh dan berarti, fokus mitigasi risiko perlu diarahkan pada penguatan aspek teknis sistem. Oleh karena itu, penting bagi penyedia sistem informasi untuk merancang dan menerapkan mekanisme penyimpanan password yang lebih canggih dan atau mengadopsi teknik autentikasi tambahan. Tujuan dari hal tersebut adalah untuk membangun lingkungan keamanan sistem informasi yang tidak hanya meningkatkan pengalaman pengguna tetapi juga memberikan perlindungan yang lebih tangguh bagi pemilik sistem.

Salah satu kasus nyata dari kebocoran data akibat penyimpanan password yang lemah terjadi pada tahun 2012, dimana LinkedIn mengalami salah satu serangan siber paling signifikan dalam sejarah platform media sosial. Serangan tersebut berhasil membobol lebih dari 6,5 juta hash password pengguna yang mencakup 5% pengguna pada waktu tersebut. Namun, pada tahun 2016 terungkap bahwa skala penyerangan sebenarnya jauh lebih besar dan terus menerus dengan lebih dari 167 juta kredensial pengguna dibocorkan secara online. Password tersebut telah dihash menggunakan algoritma SHA-1 tanpa menerapkan salt, sebuah teknik keamanan dasar untuk memperkuat enkripsi. Hal tersebut membuat hash tersebut relatif lebih rentan terhadap teknik pemecahan hash seperti rainbow tables attack. Insiden ini memaksa LinkedIn untuk segera melakukan reset password massal dan mengirimkan peringatan kepada jutaan penggunanya. Serangan ini menjadi pelajaran penting bagi industri teknologi tentang pentingnya menerapkan praktik hashing yang kuat. Selain menyebabkan kerugian reputasi, LinkedIn juga harus membayar jutaan dolar kepada pengguna yang terdampak sebagai kompensasi akibat serangan ini.

Kasus LinkedIn tersebut kini telah menjadi pembelajaran klasik yang menunjukkan kegagalan dalam implementasi keamanan penyimpanan password. Insiden ini secara jelas berhasil mengungkapkan bahwa penggunaan fungsi hash kriptografis standar (dimana dalam kasus ini adalah SHA-1) saja tidak cukup untuk melindungi password. Diperlukan proses autentikasi tambahan atau setidaknya mekanisme tambahan seperti salt dan iterasi untuk meningkatkan ketahanan terhadap serangan. Oleh karena itu, makalah ini bertujuan untuk menganalisis keamanan hash password polos dengan hash yang telah diperkuat menggunakan salt dan iterasi, yang akan menunjukkan perbedaan drastis dalam sumber daya dan waktu yang dibutuhkan untuk membobolnya. Dengan begitu, eksperimen ini tidak hanya mendemonstrasikan kelalaian teknis yang terjadi pada LinkedIn, tetapi juga memberikan bukti kokoh tentang keperluan untuk menerapkan standar hashing yang kuat di era serangan siber yang semakin canggih ini.

II. Landasan Teori

Bagian ini membahas berbagai konsep kriptografi yang menjadi dasar dalam penelitian ini. Penjelasan diberikan untuk memberikan konteks teori yang memadai mengenai eksperimen yang dilakukan bagi pembaca makalah ini.

A. Fungsi Hash Kriptografis

Fungsi hash kriptografis adalah algoritma matematika yang mengubah data masukan menjadi urutan string heksadesimal dengan panjang tetap yang disebut sebagai nilai hash. Fungsi hash memiliki formula berikut:

$$h = H(m) \quad (1)$$

Dari formula tersebut, hash h adalah hasil diterapkannya fungsi hash H terhadap masukan m. Keamanan fungsi hash kriptografis ditentukan oleh sifat utama berikut:

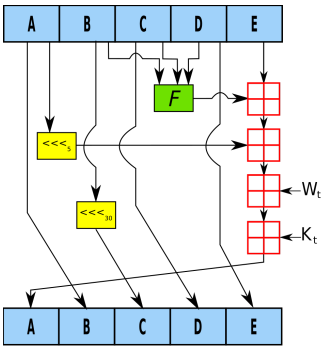
1. *Pre-image resistance*, yaitu sulit untuk menemukan masukan m dari nilai hash h = H(m).
2. *Second pre-image resistance*, yaitu sulit untuk menemukan masukan lain m' yang berbeda dari m sehingga H(m') = H(m).
3. *Collision resistance*, yaitu sulit untuk menemukan pasangan masukan berbeda (m1, m2) dengan H(m1) = H(m2).

Fungsi hash kriptografis dirancang agar cepat, deterministik, dan satu arah, dimana apabila terdapat perubahan sekecil apapun pada masukan, hash yang dihasilkan akan menjadi sangat berbeda. Fungsi ini penting untuk diterapkan dalam mengamankan informasi digital terutama dalam penerapannya pada proses verifikasi dan autentikasi data.

B. Secure Hash Algorithm (SHA)

Secure Hash Algorithm (SHA) merupakan keluarga fungsi hash kriptografis standar yang ditetapkan oleh *National Institute of Standards and Technology* (NIST). SHA-1 merupakan fungsi hash yang dibuat berdasarkan algoritma hash MD5 yang menghasilkan keluaran sepanjang 160 bit, fungsi ini dilakukan dengan membagi masukan menjadi beberapa blok tetap yang diproses secara iteratif melalui fungsi kompresi.

Proses SHA-1 dimulai dengan pesan masukan yang kemudian dipadding agar panjangnya memenuhi kelipatan 512 bit. Pesan yang telah dipadding kemudian akan dibagi menjadi blok dan word yang diperluas untuk proses komputasi. Selanjutnya, nilai hash awal diinisialisasi dan setiap blok diproses dan dikompresi melalui 80 putaran operasi. Hasil dari putaran tersebut ditambahkan ke nilai hash awal sehingga menghasilkan output berupa nilai hash akhir sepanjang 160 bit.



Gambar 2.1 Satu Iterasi Fungsi Kompresi SHA-1
Sumber :
https://commons.wikimedia.org/wiki/File:SHA-1.svg

Meskipun SHA-1 telah memenuhi seluruh sifat dasar fungsi hash kriptografis, desainnya yang terlalu terfokus pada efisiensi komputasi, justru akan menjadi kelemahan karena memungkinkan penyerang melakukan perhitungan hash dalam jumlah besar.

C. Salt dan Iterasi pada Fungsi Hash

Untuk meningkatkan keamanan, dapat digunakan nilai salt yang digabungkan dengan masukan sebelum hashing. Proses hashing dengan salt dapat dirumuskan sebagai h = H(m||s) dimana salt s akan digabungkan dengan masukan m sebelum dilakukan fungsi hash H.

Iterasi, atau *key stretching*, merupakan teknik untuk meningkatkan biaya komputasi hashing dengan menerapkan fungsi hash secara berulang. Proses ini dapat dilihat dengan formula:

$$hi = Hi(\dots H1(H(m)) \quad (2)$$

Dimana hashing dilakukan berulang sebanyak i + 1 untuk mendapatkan hash yang berbeda.

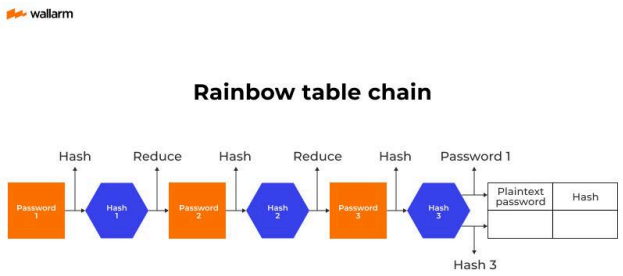
Penggunaan salt dan iterasi dalam pemecahan hash berfungsi untuk meningkatkan ketahanan password terhadap berbagai jenis serangan seperti brute force attack dan rainbow table attack. Dengan digunakannya salt, sebuah hash yang sudah ditemukan passwordnya tidak dapat digunakan kembali untuk menyerang pengguna dengan password yang sama sehingga secara garis besar akan mencegah penggunaan lookup table. Sedangkan penggunaan iterasi akan memperlambat setiap percobaan penebakan password secara linier, sehingga penyerang membutuhkan waktu komputasi yang jauh lebih besar untuk mencoba setiap kemungkinan sehingga penyerangan bisa dianggap tidak efisien secara praktis.

D. Rainbow Table Attack

Rainbow table attack adalah sebuah teknik pertukaran memori dan waktu yang diterapkan dalam kriptanalisis hash password. Dibandingkan menghitung hash untuk setiap tebakan password secara langsung seperti *brute force attack*, penyerang melakukan komputasi terlebih dahulu untuk menghasilkan struktur data yang tidak hanya berukuran lebih kecil tetapi juga memiliki proses

pencarian hash yang lebih cepat saat serangan dilakukan.

Rainbow table tidak menyimpan semua pasangan masukan dan hash secara eksplisit, melainkan menyimpan rantai yang direpresentasikan oleh nilai awal dan nilai akhir. Rantai tersebut dibentuk berdasarkan dua fungsi utama yaitu fungsi hash dan fungsi reduksi. Fungsi reduksi adalah sebuah fungsi spesifik yang berusaha memasangkan sebuah hash dengan sebuah calon pesan masukan. Fungsi reduksi ini tidak berusaha membalikkan hash, tetapi hanya berupa pemetaan deterministik dari hash ke ruang pesan.



Gambar 2.2 Diagram Rantai Pengisian Rainbow Table
Sumber :
https://www.wallarm.com/what/rainbow-table-attack

Pada tahap pencarian hash target, penyerang memulai dengan mengambil nilai hash yang ingin dipecahkan, kemudian menerapkan fungsi reduksi yang sesuai dengan kolom terakhir pada rainbow table untuk mengubah hash tersebut menjadi kandidat password. Kandidat password ini selanjutnya diproses kembali menggunakan fungsi hash, lalu direduksi lagi menggunakan fungsi reduksi pada kolom sebelumnya, dan proses ini diulang secara berurutan hingga menghasilkan sebuah nilai yang berpotensi cocok dengan nilai akhir yang tersimpan di dalam rainbow table. Jika ditemukan kecocokan dengan salah satu end point, penyerang kemudian merekonstruksi seluruh rantai dari nilai awal terkait untuk melacak kembali langkah demi langkah hingga menemukan password asli yang menghasilkan hash target tersebut.

D. Dictionary Attack

Dictionary attack adalah metode pemecahan password dengan cara mencoba daftar password yang telah disiapkan sebelumnya, bukan seluruh kemungkinan karakter seperti pada *brute force attack*. Daftar ini biasanya berisi kata umum, frase populer, variasi sederhana, password yang sering muncul pada kebocoran data sebelumnya, atau bahkan kata password. Tujuan utama dari metode ini adalah untuk mengeksploitasi kebiasaan manusia yang cenderung menggunakan password mudah diingat dan mudah diprediksi.

III. IMPLEMENTASI

Implementasi eksperimen ini akan dilakukan dengan dua skenario penyerangan. Kedua skenario tersebut akan diimplementasi menggunakan python dalam skala kecil untuk menunjukkan perbedaan resource yang diperlukan untuk memecah hash untuk masing-masing skenario.

A. Rainbow Table Attack

Eksperimen *rainbow table attack* akan dilakukan dengan pembuatan dua program utama yaitu program pembuatan rainbow table dan program lookup rainbow table. Pada penyerangan ini password akan dibuat berdasarkan karakter alfanumerik sebanyak 62 karakter.

1. Pembuatan Rainbow Table

Pada program pembuatan rainbow table, password akan pertama-tama dibuat berdasarkan panjang karakter dan tipe karakternya. Kemudian dari password tersebut akan diterapkan fungsi hash SHA-1 yang diambil dari library hashlib dan fungsi reduksi dengan formula:

$$R(H, i) = BaseC(int(H) + i) \quad (3)$$

Pada fungsi tersebut hash H akan diubah menjadi integer dan dijumlahkan dengan nilai iterasi chain i dan diterapkan fungsi BaseC() sebanyak jumlah karakter di dalam password. Fungsi BaseC itu sendiri akan dilakukan dengan:

$$BaseC(j) = j \bmod C \quad (4)$$

Dimana nilai j (hasil penjumlahan nilai integer hash dan nilai iterasi chain saat ini) akan dimodulokan dengan panjang set karakter C (pada eksperimen adalah 62) dan dipilih karakter tersebut pada set karakter. Langkah ini akan dilakukan sehingga didapatkan sebuah password deterministik dari hash yang dimiliki.

Fungsi hash dan reduksi kemudian akan dilakukan berulang sebanyak panjang chain. Setelah chain terbentuk, akan disimpan password awal dan hash akhirnya pada sebuah table. Langkah pembuatan chain akan dilakukan berulang sebanyak jumlah chain yang ditentukan dimana chain dengan hash akhir yang sama (menandakan kolisi) tidak akan disimpan.

2. Cakupan Rainbow Table

Pembuatan rainbow table akan dilakukan untuk 1-4 karakter password dan akan dibandingkan waktu dan jumlah chain yang dibentuk per detik. Penentuan panjang chain dan jumlah chain akan dilakukan untuk masing-masing panjang karakter agar dapat mencakupi sebanyak $\approx 80\%$ coverage keyspace karakter alfanumerik yang ada pada tabel berikut.

Panjang Karakter (n)	Keyspace (62^n)

1	62
2	3.844
3	238.328
4 (Maksimal panjang password diuji)	14.776.336
5	916.132.832
6	56.800.235.584
7	3.521.614.606.208
8 (Minimum panjang password umum)	218.340.105.584.896

Tabel 3.1 Tabel Keyspace Karakter Alfanumerik

Coverage chain terhadap itu sendiri akan dikalkulasikan menggunakan formula berikut:

$$Coverage = 1 - e^{-M/N} \quad (5)$$

Dimana N merupakan keyspace untuk jumlah karakter password dan M merupakan total percobaan hash yang didapatkan dengan mengalikan panjang chain dan jumlah chain yang disimpan pada rainbow table. Coverage ini tidak hanya menandai jumlah cakupan keyspace tetapi juga memprediksi kemungkinan ditemukannya password dari hash tersebut.

3. Rainbow Table Lookup

Setelah rainbow table terbentuk, eksperimen dilanjutkan dengan fase lookup untuk melakukan simulasi serangan terhadap hash target. Pada tahap ini, hash target diasumsikan dapat berada pada posisi mana pun di dalam sebuah rantai, sehingga proses lookup dilakukan dengan memproyeksikan hash target ke berbagai kemungkinan hash akhir rantai melalui kombinasi fungsi hash dan reduksi. Apabila hash hasil proyeksi ditemukan dalam rainbow table, maka rantai tersebut akan direkonstruksi kembali dari password awal dan setiap hash yang dihasilkan akan diverifikasi sehingga ditemukan hash yang sesuai dengan hash target. Proses ini memastikan bahwa hasil yang diperoleh merupakan password yang valid dan bukan hasil kolisi.

4. Konfigurasi Pengujian

Pengujian akan dilakukan dengan mengatur informasi yang dimiliki oleh penyerang, dimana ditentukan bahwa saat ini hanya berupa jumlah karakter password. Hal ini akan dipengaruhi oleh jumlah karakter salt dimana dengan ditambahkannya karakter salt semakin lama juga pemecahan hashnya karena keyspace nya yang semakin besar. Selain itu, juga akan diketahui jumlah iterasi hash yang diperkirakan akan memperpanjang proses membuat rainbow table secara signifikan.

Pengujian pengaruh salt akan dilihat dengan perbandingan rainbow table attack dimasing-masing panjang password. Sedangkan pengujian pengaruh iterasi hash akan dilihat dengan perbandingan proses rainbow table attack untuk password tiga karakter tanpa iterasi dan dengan sepuluh dan seratus iterasi hash.

Secara umum, penerapan salt dan iterasi hash akan secara langsung menghancurkan rainbow table attack. Namun, pada eksperimen ini saya berusaha membandingkan seberapa besar sumber daya yang terpakai apabila diterapkan salt maupun iterasi meskipun sudah diketahui oleh penyerang panjang karakter salt dan jumlah iterasi hash.

B. Dictionary attack

Eksperimen dictionary attack akan dilakukan dengan membuat sebuah kamus kecil sebesar sepuluh password umum. Kemudian sebuah password masukan akan dihash menggunakan empat pengaturan berbeda yaitu tanpa salt maupun iterasi, dengan salt, dengan iterasi, serta dengan salt dan iterasi. Hash password tersebut akan dicocokan dengan melakukan lookup pada kamus yang telah dibuat dengan melakukan metode hash yang sesuai dengan pengaturan untuk masing-masing password di kamus. Parameter yang diukur adalah waktu yang diperlukan untuk memecahkan hash password dalam detik.

IV. Hasil Eksperimen

A. Hasil Eksperimen Rainbow Table Attack

Berikut adalah hasil data pengujian rainbow table attack untuk password 1-4 karakter. Untuk mendapatkan waktu lookup, proses tersebut akan dijalankan sebanyak 5 kali dan dimasukkan rata-ratanya.

Panjang Password	Panjang Chain	Jumlah end point	Coverage	Ukuran Tabel (KB)	Waktu pembuatan (s)	Waktu Lookup (s)
1	10	10	80,1 %	1	0,0017	0,00081
2	100	72	84,6 %	4	0,7919	0,00352
3	100	4089	82,0 %	184	8.1554	0,0072
4	1000	22833	82.5 %	1184	594,28	0,2088

Tabel 4.1 Tabel Hasil Eksperimen Rainbow Table Attack

Dari hasil tersebut dapat terlihat bahwa waktu pembuatan mengalami peningkatan jauh dalam ukuran tabel dan waktu pembuatan rainbow table tetapi untuk jumlah karakter kecil ini, tidak ada perubahan waktu

lookup yang berarti.

B. Hasil Eksperimen Rainbow Table Attack dengan Iterasi

Berikut adalah hasil data eksperimen rainbow attack untuk password 3 karakter dengan 1, 10 dan 100 iterasi hash. Untuk mendapatkan waktu lookup, proses tersebut akan dijalankan sebanyak 5 kali dan dimasukkan rata-ratanya.

Jumlah Iterasi Hash	Ukuran Tabel (KB)	Waktu pembuatan (s)	Waktu Lookup (s)
1	184	8.1554	0,0072
10	273	37.9715	0,0208
100	282	347.1692	0,6649

Tabel 4.2 Tabel Hasil Eksperimen Rainbow Table Attack

Penggunaan iterasi dalam melakukan hash pada password akan mempengaruhi waktu pembuatan rainbow table secara drastis tetapi tidak mengubah ukuran tabel maupun waktu lookup secara signifikan.

C. Hasil Eksperimen Dictionary Attack

Berikut adalah hasil data eksperimen dictionary attack untuk pengaturan tanpa salt maupun iterasi, dengan salt, dengan iterasi, serta dengan salt dan iterasi. Salt akan ditentukan secara acak di awal eksperimen dan sama untuk setiap pengaturan terkait. Iterasi akan ditentukan sebanyak 100000 iterasi hash. Untuk mendapatkan waktu pemecahan, proses tersebut akan dijalankan sebanyak 5 kali dan dimasukkan rata-ratanya.

Skema Hash	Waktu pemecahan hash (s)	Hash/detik
Hash	0.000029421	33989
Hash + salt	0.000028753	34778
Hash + 100000 iterasi	0.9762	102438
Hash + salt + 100000 iterasi	0.9546	104755

Tabel 4.3 Tabel Hasil Eksperimen Dictionary Attack

Dari data tersebut dapat terlihat jelas bahwa iterasi dapat secara signifikan memperlambat dictionary attack. Sedangkan salt tidak memiliki pengaruh besar terhadap satu password dengan salt yang telah diketahui tetapi dengan adanya salt hash yang didapatkan tidak dapat digunakan lagi untuk mendapatkan password yang sama dari pengguna lain karena salt yang berbeda untuk masing-masing pengguna.

V. Kesimpulan

Berdasarkan hasil eksperimen dan analisis yang telah dilakukan, dapat disimpulkan bahwa metode penyimpanan password menggunakan hash SHA-1 saja seperti pada kasus kebocoran data LinkedIn 2012 sudah tidak cukup lagi untuk diterapkan di masa modern ini. Eksperimen rainbow table attack menunjukkan bahwa hash SHA-1 tanpa salt maupun iterasi dapat dipecahkan secara efisien dengan sumber daya komputasi yang relatif kecil terutama untuk password dengan panjang karakter pendek. Hal ini menunjukkan bahwa penerapan hash yang terpaku pada kecepatan akan cepat untuk dipecahkan juga.

Meskipun berdasarkan eksperimen, penambahan salt tidak berpengaruh untuk mencegah pemecahan satu password tertentu, perlu diketahui bahwa salt dapat secara efektif menggagalkan penggunaan rainbow table maupun dictionary secara ulang terhadap banyak pengguna, karena setiap password menghasilkan hash yang unik berdasarkan penggunanya meskipun isi passwordnya sama. Sementara itu, penerapan iterasi hash secara signifikan meningkatkan biaya komputasi bagi penyerang, baik pada skenario rainbow table attack maupun dictionary attack.

Makalah ini saya harapkan dapat digunakan sebagai referensi dan pengingat bagi pemilik sistem untuk menerapkan praktik terbaik dalam pengamanan password terutama dalam bagian penyimpanan dengan tujuan menghadapi ancaman siber yang terus berkembang di masa modern ini satu dekade setelah kasus kebocoran data LinkedIn. Untuk kedepannya eksperimen ini dapat di ekstensi dengan membandingkan standar keamanan hash ini dengan standar keamanan modern seperti bcrypt, scrypt, atau Argon2. Selain itu, eksperimen ini dapat diperluas dengan memperbesar skala dan menggunakan data password dummy yang tersedia di internet.

VI. Ucapan Terima Kasih

Penulis mengucapkan syukur kepada Tuhan Yang Maha Esa yang telah membantu dalam setiap langkah penulisan makalah ini. Ucapan terima kasih juga penulis sampaikan kepada Bapak Dr. Ir. Rinaldi Munir, M.T. selaku dosen pengampu mata kuliah IF4020 Kriptografi atas bimbingan, arahan serta wawasan yang telah diberikan selama perkuliahan sehingga makalah ini dapat disusun dengan baik. Penulis juga menyampaikan apresiasi kepada seluruh pihak yang telah memberikan dukungan dan masukan selama penyusunan makalah ini.

References

[1] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano, “Passwords and the evolution of imperfect authentication,” *Commun. ACM*, vol. 58, no. 7, pp. 78–87, Jun. 2015. [*Online*]. Tersedia: https://dl.acm.org/doi/10.1145/2699390. Diakses: 18 Desember 2025.

[2] T. Cohen Wood, "The LinkedIn hack: Understanding why it was so easy to crack the passwords," Inspired eLearning, LLC, 2016. [*Online*]. Tersedia: https://inspiredelearning.com/wp-content/uploads/2017/05/LinkedInHack_Guide_Proof.pdf. Diakses: 18 Desember 2025.

[3] P. Oechslin, "Making a faster cryptanalytic time-memory trade-off," Lab. de Securite et de Cryptogr. (LASEC), Ecole Polytech. Fed. de Lausanne, Lausanne, Switzerland, 2003. [Online]. Tersedia: https://infoscience.epfl.ch/record/152274. Diakses: 18 Desember 2025.

[4] *Secure Hash Standard (SHS)*, Nat. Inst. Stand. Technol., Gaithersburg, MD, USA, Federal Information Processing Standards Publication (FIPS) 180-4, Aug. 2015. [*Online*]. Tersedia: https://csrc.nist.gov/pubs/fips/180-4/upd1/final. Diakses: 18 Desember 2025.

[5] *SHA-1 Hash*. GeeksforGeeks, Sep. 27, 2018 (diperbarui: Jul. 18, 2024). [Online]. Tersedia: https://www.geeksforgeeks.org/java/sha-1-hash-in-java/. Diakses: 18 Desember 2025.

[6] *Rainbow Table Attack*. Wallarm.com, 6 Apr. 2025. [*Online*]. Tersedia: https://www.wallarm.com/what/rainbow-table-attack. Diakses: 18 Des. 2025.

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, 24 Desember 2025

Nicholas Francis Aditjandra
18221005