Journal of Artificial Intelligence and Software Engineering
Vol. 5, No. 3, September 2025, pp. 1278~1284
E-ISSN: 2777-001X, DOI: 10.30811/jaise.v5i3.7266

Implementation of Honey Encryption to Improve Resilience against Brute Force Attacks in Cloud-based Microservices Communication

Gabriel Nathanael Purba[1*], Bagus Gede Krishna Yudistira[2], Kadek Yota Ernanda Aryanto[3],
[1,2,3] Teknik Informatika, Fakultas Teknik dan Kejuruan, Universitas Pendidikan Ganesha, Singaraja, 81116, Indonesia

Informasi Artikel

Diterima : 28 Juni 2025
Revisi : 25 September 2025
Publikasi : 30 September 2025

Kata Kunci:

Microservices
Honey Encryption
Decoy Message

ABSTRAK

Arsitektur microservices berbasis cloud meningkatkan skalabilitas dan fleksibilitas sistem namun memperbesar risiko keamanan komunikasi antar layanan. Penelitian ini menerapkan Honey Encryption pada API Gateway di infrastruktur Amazon EC2, mengombinasikan Distribution Transforming Encoder dengan AES-CBC 256-bit untuk menjaga kerahasiaan dan integritas data. Hasil validasi menunjukkan bahwa dekripsi dengan kunci benar memulihkan data asli, sedangkan kunci salah menghasilkan decoy message yang plausible dan bervariasi per field, secara efektif menghambat upaya brute force. Evaluasi performa menegaskan bahwa waktu proses enkripsi dan dekripsi berbanding lurus dengan beban data namun tetap berada dalam batas latensi yang dapat diterima. Simulasi serangan memperlihatkan keunggulan decoy message dalam menyamarkan pesan asli, meningkatkan keamanan komunikasi data antar microservices.

ABSTRACT

Cloud-based microservices architecture enhances system scalability and flexibility but increases the security risks in inter-service communication. This research implements Honey Encryption on an API Gateway within Amazon EC_2 infrastructure, combining a Distribution Transforming Encoder with AES-CBC 256-bit encryption to ensure data confidentiality and integrity. Validation results show that decryption using the correct key successfully recovers the original data, while incorrect keys produce plausible and field-varying decoy messages, effectively thwarting brute-force attacks. Performance evaluation confirms that encryption and decryption processing times are proportional to data load but remain within acceptable latency limits. Attack simulations demonstrate the effectiveness of decoy messages in concealing the original message, enhancing the security of data communication between microservices.

Penulis Koresponden

Email[1] : gabriel@student.undiksha.ac.id

Cara sitasi IEEE:

G. N. Purba, B. G. K. Yudistira, & K. Y. E. Aryanto, "Implementation of Honey Encryption to Improve Resilience against Brute Force Attacks in Cloud-based Microservices Communication," *Journal of Artificial Intelligence and Software Engineering (J-AISE)*, vol. 5, no. 3, pp. 1274-1284, September 2025, doi: 10.30811/jaise.v5i3.7211

1. PENDAHULUAN

Arsitektur microservices telah menjadi pendekatan populer dalam pengembangan aplikasi modern berbasis cloud karena memungkinkan pemisahan layanan ke dalam unit-unit kecil yang dapat dikembangkan, di-deploy , dan diskalakan secara independen [1], [2], [3]. Penggunaan microservices sangat relevan dalam era Revolusi Industri 4.0 yang mendorong transformasi digital, integrasi teknologi informasi, serta otomatisasi sistem untuk meningkatkan efisiensi dan fleksibilitas layanan [1], [2], [4]. Kebutuhan akan solusi berbasis cloud yang menunjang aksesibilitas dan integrasi sistem secara efisien juga semakin menonjol dalam pengembangan sistem informasi modern [5], [6], [4]. Hal ini didukung oleh kemajuan teknologi cloud computing dan sistem repositori data [4], [25]. Namun, di balik fleksibilitas dan efisiensinya, arsitektur ini menimbulkan tantangan baru dalam aspek keamanan, khususnya dalam komunikasi antar layanan [7], [8], [19]. Pesan-pesan yang ditransmisikan antar microservices , terutama melalui protokol HTTP/REST, menjadi rentan terhadap ancaman seperti penyadapan [9], [6]. Ancaman ini mencakup juga brute-force attack [10], [17].

Dalam konteks ini, penggunaan teknik enkripsi menjadi solusi utama untuk menjaga kerahasiaan data [11], [16]. Ini esensial untuk memastikan integritas data yang dikirimkan [19], [2]. Algoritma yang banyak digunakan adalah Advanced Encryption Standard (AES) dalam mode Cipher Block Chaining (CBC) [11], [20]. Mode ini dikenal mampu menghasilkan ciphertext dengan entropi tinggi dan keamanan yang solid [21], [18]. Meskipun demikian, AES-CBC memiliki keterbatasan dalam menghadapi serangan eksplorasi kunci (key-guessing attack), karena hasil dekripsi dengan kunci yang salah tetap akan menghasilkan output yang membingungkan namun mudah dikenali sebagai salah oleh pihak ketiga [22], [19].

Sebagai solusi atas kelemahan ini, Honey Encryption (HE) diperkenalkan [14], [26]. Teknik ini dirancang agar setiap ciphertext yang didekripsi menggunakan kunci yang salah tetap memberikan hasil berupa decoy message [4], [19]. Decoy message ini tampak valid namun sebenarnya palsu, efektif mengelabui penyerang [14], [26], [4]. Dengan demikian, HE memperlambat proses eksplorasi kunci [16], [19]. Ini menyulitkan penyerang untuk memastikan apakah hasil dekripsi sudah benar atau belum [3], [19].

Dalam konteks teknis, HE bergantung pada mekanisme Distribution Transforming Encoder (DTE) [14], [26]. DTE mengubah pesan asli menjadi representasi probabilistik sebelum proses enkripsi dilakukan [20], [21]. Tantangan utama dari DTE adalah menjaga agar hasil dekripsi, baik menggunakan kunci yang benar maupun salah, tetap menghasilkan pesan yang berada dalam domain yang masuk akal [14], [19], [22]. Penelitian sebelumnya telah membuktikan efektivitas HE dalam perlindungan data pada proses migrasi cloud [22], [23]. Efektivitas ini juga menyoroti potensinya dalam keamanan cloud computing secara umum [14], [22], [23].

Berbagai penelitian sebelumnya telah mengeksplorasi metode untuk meningkatkan keamanan data dan komunikasi, khususnya dalam konteks enkripsi berbasis deception . Honey Encryption (HE) telah menjadi fokus utama, bertujuan menyajikan decoy message yang tampak sah kepada penyerang yang menggunakan kunci tidak valid [14], [4]. Studi Gharbi dan Nori, misalnya, memberikan tinjauan komprehensif tentang teknik keamanan HE dan fleksibilitasnya terhadap serangan brute-force [14]. Jain juga mengusulkan model hibrida yang mengintegrasikan Honey Encryption dengan algoritma Twofish untuk perlindungan kata sandi dan pesan, menyoroti efektivitas honeywords dalam menghasilkan data palsu yang valid [16]. Studi-studi ini secara kolektif menunjukkan perlunya solusi keamanan yang lebih canggih yang mampu menipu penyerang agar mereka tidak dapat membedakan data asli dari data palsu, terutama dalam skenario serangan brute-force .

Selain itu, meningkatnya kompleksitas serangan otomatis seperti brute-force attack dan vulnerability scanning menambah kebutuhan akan pendekatan enkripsi canggih [7], [9], [17]. Perkembangan alat-alat eksploitasi sistem yang mampu mendeteksi celah keamanan dengan cepat menandakan perlunya sistem perlindungan yang tidak hanya kuat secara matematis [17]. Sistem ini juga harus mampu menipu upaya penyerang [26], [19]. Dalam konteks keamanan jaringan, pendekatan berlapis menjadi kunci untuk mempertahankan keutuhan komunikasi antar layanan [24], [9]. Ini didukung oleh integrasi teknologi enkripsi [10], [19].

Penelitian ini berfokus pada penerapan Honey Encryption untuk mengamankan komunikasi antar microservices dalam lingkungan cloud berbasis Infrastructure-as-a-Service (IaaS) menggunakan Amazon EC2. Platform ini dipilih karena memberikan fleksibilitas tinggi dalam pengelolaan jaringan, virtual machine , dan container orchestration , yang sangat sesuai untuk simulasi komunikasi layanan secara realistis dalam jaringan cloud privat [7], [2], [4]. Pendekatan yang digunakan dalam penelitian ini menyisipkan proses HE secara transparan pada API Gateway. Komponen ini akan menangani proses enkripsi dan dekripsi pesan antara microservices , memastikan bahwa layanan internal hanya berkomunikasi dalam plaintext . Jika terjadi serangan brute-force terhadap ciphertext , sistem akan tetap memberikan decoy message yang tampak sah, sehingga mengacaukan upaya eksploitasi kunci.

Tujuan utama dari penelitian ini adalah merancang dan mengimplementasikan mekanisme komunikasi yang lebih aman antar layanan microservices dalam lingkungan cloud menggunakan HE, serta mengevaluasi

performa enkripsi, dekripsi, dan ketahanan sistem terhadap serangan brute-force. Diharapkan, hasil dari penelitian ini dapat memberikan kontribusi nyata terhadap pengembangan sistem komunikasi yang aman dan modern untuk infrastruktur microservices berbasis cloud.

2. METODE

2.1. Arsitektur Sistem dan Lingkungan Penelitian

Penelitian ini menerapkan arsitektur microservices dalam lingkungan cloud dengan menggunakan Amazon EC2 sebagai platform IaaS. Sistem yang dikembangkan terdiri dari tiga komponen utama:

- API Gateway: Bertanggung jawab untuk menangani proses enkripsi dan dekripsi payload pesan menggunakan Honey Encryption. API Gateway berfungsi sebagai titik masuk utama bagi komunikasi antar-mikroservis.
- Service A (Pengirim): Layanan yang mengirimkan pesan dalam bentuk plaintext ke API Gateway.
- Service B (Penerima): Layanan yang menerima pesan plaintext dari API Gateway setelah proses dekripsi.

Semua komponen dikemas dalam container Docker dan diorkestrasi menggunakan Docker Compose. Lingkungan penelitian diuji di EC2 untuk mensimulasikan kondisi penerapan sistem pada cloud.

2.2. Implementasi Honey Encryption dalam API Gateway

2.2.1. Desain dan Alur Kerja Honey Encryption

Honey Encryption (HE) diterapkan untuk mengamankan payload komunikasi antar-mikroservis dengan menghasilkan ciphertext yang plausible apabila dekripsi dilakukan dengan kunci yang salah. Mekanisme yang digunakan melibatkan dua tahap utama:

1. Encoding dengan Distribution Transforming Encoder (DTE):
    - Apabila pesan asli terdapat dalam dataset honey words, DTE menghasilkan seed dengan format "I:<index>".
    - Jika pesan tidak terdapat dalam dataset, DTE mengembalikan seed dengan format "C:" yang diikuti pesan asli. Pendekatan ini memungkinkan sistem memulihkan pesan asli secara konsisten saat kunci yang benar digunakan dan menghasilkan decoy message yang plausible secara individual untuk setiap field payload, apabila kunci yang salah digunakan.
2. Enkripsi Simetris dengan AES-CBC:
    - Seed yang dihasilkan melalui DTE diproses dengan metode padding menggunakan skema PKCS#7.
    - Enkripsi dilakukan menggunakan algoritma AES dalam modus CBC dengan kunci sepanjang 32 byte untuk mendukung AES-256.
    - Hasil enkripsi, yang merupakan ciphertext dikombinasikan dengan Inisialisasi Vektor (IV), dikodekan dalam format Base64.

Pada tahap dekripsi, ciphertext didekode dan didekripsi dengan kunci yang digunakan. Setelah penghapusan padding, seed yang diperoleh dianalisis untuk menentukan apakah memiliki format yang valid (dimulai dengan "I:" atau "C:"). Jika valid, AdaptiveDTE digunakan untuk memulihkan pesan asli atau, apabila kunci salah, untuk menghasilkan decoy message. Proses dekripsi ini memastikan bahwa output yang dihasilkan berupa JSON dengan masing-masing field payload mengembalikan pesan asli (jika kunci benar) atau decoy message yang unik (jika kunci salah).

2.2.2. Pembuatan Decoy Message dalam Format JSON

Pada implementasi HE dalam penelitian ini, decoy message dirancang dalam format JSON agar konsistensi struktur data tetap terjaga. Setiap field pada payload pesan dienkripsi secara individual. Saat dekripsi dengan kunci yang salah, sistem menggunakan parameter tambahan berupa "salt" dalam perhitungan hash untuk memastikan bahwa setiap field menghasilkan decoy message yang berbeda. Decoy message dipilih dari dataset yang telah ditentukan dan dikembalikan dalam struktur JSON yang sama dengan pesan asli. Pendekatan ini memastikan bahwa struktur data hasil dekripsi tetap konsisten dengan payload asli. Pemanfaatan salt berupa nama field dalam perhitungan hash membuat decoy message berbeda antar-field, sehingga meningkatkan kompleksitas analisa bagi pihak yang mencoba melakukan brute force.

2.2.3. Alat dan Teknologi yang Digunakan

- Bahasa Pemrograman dan Framework: Prototype sistem dikembangkan menggunakan Python dan framework Flask untuk membangun API Gateway.

- Enkripsi: Library pycryptodome digunakan untuk implementasi AES-CBC.
- Containerization dan Orkestrasi: Docker digunakan untuk mengemas setiap komponen, dengan koordinasi menggunakan Docker Compose.
- Platform Cloud: Amazon EC2 digunakan sebagai platform Infrastructure as a Service (IaaS) untuk deployment dan pengujian di lingkungan cloud, menggunakan instans t2.micro (Amazon Linux 2023) di wilayah US East (N. Virginia) (us-east-1d), dengan volume root EBS 10 GiB.

2.3. Metode Pengujian dan Evaluasi

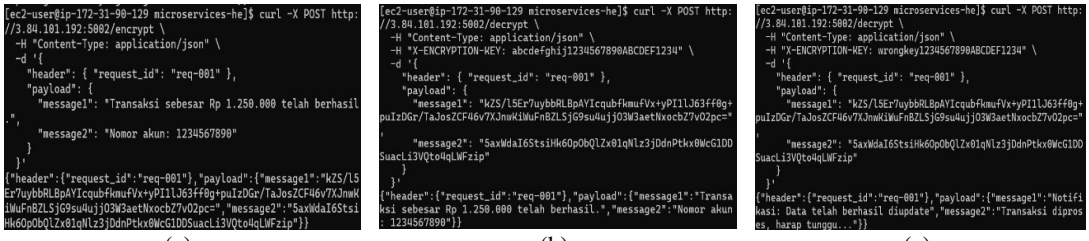
Pengujian dan evaluasi dalam penelitian ini dilakukan untuk memastikan bahwa implementasi Honey Encryption (HE) pada arsitektur microservices berjalan dengan benar serta dapat meningkatkan keamanan komunikasi antar-layanan. Pengujian dilakukan melalui empat tahapan utama berikut:

1. Deployment Sistem Microservices
   Sistem dikembangkan dalam tiga layanan utama: Service A (pengirim data), API Gateway (tempat enkripsi dan dekripsi menggunakan Honey Encryption), dan Service B (penerima data). Ketiga layanan dijalankan dalam container menggunakan Docker Compose dan kemudian di-deploy ke Amazon EC2 untuk pengujian dalam lingkungan cloud.
2. Pengujian Fungsi Enkripsi dan Dekripsi
   Data uji dikirim dari Service A ke API Gateway untuk dienkripsi, dan dari Service B ke API Gateway untuk didekripsi. Format data yang digunakan berupa struktur JSON bertingkat, terdiri dari beberapa field payload yang diuji secara individual. Pengujian bertujuan untuk memastikan:
   - Jika kunci enkripsi dan dekripsi benar, hasil dekripsi harus identik dengan plaintext asli.
   - Jika kunci dekripsi salah, sistem akan mengembalikan decoy message yang berbeda dari pesan asli
3. Evaluasi Performa Sistem
   Evaluasi dilakukan untuk mengukur efisiensi sistem setelah penerapan Honey Encryption. Pengukuran dilakukan dengan mengukur waktu enkripsi dan dekripsi berdasarkan jumlah payload.
4. Analisis Keamanan terhadap Serangan Brute Force
   Sistem diuji dengan upaya dekripsi menggunakan sejumlah kunci yang salah secara berulang untuk mensimulasikan serangan brute force. Setiap output dievaluasi untuk memastikan bahwa Honey Encryption berhasil menghasilkan decoy message yang berbeda untuk setiap kunci salah, sehingga menyulitkan penyerang dalam mengidentifikasi pesan asli melalui percobaan berulang.

3. HASIL DAN PEMBAHASAN

3.1 Validasi Honey Encryption

Pengujian dilakukan dengan mengirimkan payload dari Service A ke API Gateway untuk dienkripsi, kemudian hasil enkripsi dikirim ke Service B untuk didekripsi menggunakan API Gateway.
Hasil menunjukkan bahwa ketika payload didekripsi menggunakan kunci yang benar, sistem berhasil mengembalikan plaintext sesuai dengan data asli, seperti tampak pada Gambar 1.a dan Gambar 1.b. Sebaliknya, saat menggunakan kunci yang salah, sistem menghasilkan output berupa decoy message, yang secara sintaksis valid namun kontennya salah, sebagaimana terlihat pada Gambar 1.c.



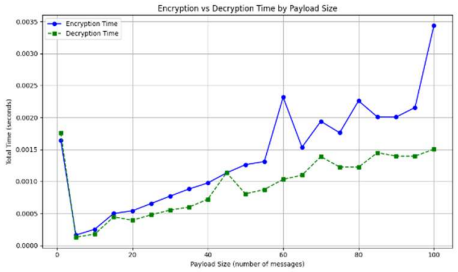
(a) (b) (c)

Gambar 1. a) Hasil Enkripsi Payload, b) Hasil Dekripsi dengan Kunci yang Benar, c) Hasil Dekripsi dengan Kunci yang Salah

Hal ini membuktikan bahwa implementasi Honey Encryption berjalan sesuai dengan konsep dasarnya, yaitu mengelabui penyerang dengan pesan palsu jika kunci dekripsi salah.

3.2 Pengukuran Performa Enkripsi dan Dekripsi

Pengukuran waktu dilakukan terhadap proses enkripsi dan dekripsi pada API Gateway, dengan jumlah payload bervariasi mulai dari 1 hingga 100. Setiap skenario dilakukan sebanyak satu kali untuk setiap ukuran payload berikut: [1, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100]. Hasil pengujian divisualisasikan pada Gambar 2.



Gambar 2. Perbandingan Waktu Enkripsi dan Dekripsi Berdasarkan Ukuran Payload

Secara umum, baik waktu enkripsi maupun dekripsi mengalami kenaikan seiring bertambahnya jumlah payload. Hal ini sejalan dengan ekspektasi, mengingat semakin banyak pesan yang harus diproses. Namun, terdapat fluktuasi atau deviasi kecil pada beberapa titik, di mana jumlah payload yang lebih tinggi justru menghasilkan waktu proses yang lebih cepat dibandingkan jumlah yang lebih rendah. Anomali ini disebabkan oleh berbagai faktor non-deterministik pada sistem, seperti:

- Optimasi caching internal Python,
- Variasi kondisi resource CPU, seperti thread scheduling atau latar belakang proses lain pada EC2 instance,
- Efisiensi batch operation, di mana overhead inisialisasi relatif lebih besar saat jumlah payload kecil, sehingga waktu rata-rata per payload terlihat lebih tinggi.
- Tren umum tetap menunjukkan bahwa waktu enkripsi lebih tinggi dibanding waktu dekripsi karena enkripsi melibatkan penambahan noise atau struktur honey data untuk setiap payload.

3.3 Simulasi Serangan Brute Force

Simulasi brute force dilakukan untuk mengevaluasi kemampuan Honey Encryption dalam menyamarkan pesan asli ketika kunci dekripsi yang salah digunakan. Pengujian dilakukan dengan mencoba 20 kunci acak terhadap satu ciphertext valid. Pemilihan 20 kunci didasarkan pada pertimbangan efisiensi sumber daya dan kebutuhan representasi eksperimen yang memadai. Jumlah ini dianggap cukup untuk membuktikan bahwa setiap kunci yang salah menghasilkan decoy message yang unik dan plausible tanpa membebani sumber daya secara berlebihan. Setiap hasil dekripsi dievaluasi untuk memastikan bahwa sistem mengembalikan pesan decoy yang tampak logis dan tidak menunjukkan pesan error. Berikut adalah tabel hasil lengkap dari simulasi tersebut:

Tabel 1. Hasil Simulasi Serangan Brute Force terhadap Honey Encryption

No	Kunci Salah	Decoy Message
1	i7h4iSEycaA4ZwXwFaQOpBHthKp6Td	Sukses! Pengaturan Anda telah disimpan
2	c3kzfP9oQim0NqRfFT5NqtM504lwYY	Notifikasi: Sistem telah diperbarui
3	dbdBVC6CySj4uWFG47GUQntW5hDxEJ	Server overload, harap tunggu beberapa saat
4	KlajvcIA7P9Om3qXU1vYBXducJNDun	Login berhasil dari IP 192.168.1.20
5	xSjQsnty5uWBQ5TDG4tZpczfnBfUEO	Perubahan email telah dikonfirmasi
6	1j8ywX2JKulumGCZ3MAKTAELyPZHdm	Pemberitahuan: Email Anda telah dikonfirmasi
7	vvu9oxwRD521xTQ81YYG3xmeDCPzSK	Akun berhasil dibuat, selamat datang!
8	Ek1rJezXGo97jBEIBRqsqyCWUrlaUo	Autentikasi gagal, coba lagi
9	5goulzY52EFepXz50xlZWW1uwGgFju	Batas kuota API telah tercapai
10	qGm0XTyvtqyYBXZZVWjFyEM0VEeWi7	Sukses! Data telah diekspor ke file CSV
11	H4ya2vcg5wbk2bZrCXXaE7iIRMW2CN	Peringatan: Kata sandi Anda terlalu lemah
12	Giatzl2h6wTPjqpfXViR3SYp5mHqkE	Kesalahan dalam parsing data
13	EumTnRZz4OfZFQBsL3LKJkSUkpncJT	Permintaan API diterima, status: 200 OK
14	TpfjDYwRN2hg6iO2mTTbtd5BkB9rzS	Token akses telah diperbarui
15	oHhm7Mt4MX9O5qPxzMOqIWYRYam00b	Koneksi ke database gagal, silakan coba lagi
16	4c3fUeUuisebs9cBw0haIPqoLlHdBj	Pembayaran berhasil, ID transaksi: TXN8945
17	DR3rnMAWLk1vatJftPUm0M5dObNpEM	Data yang Anda masukkan tidak valid
18	ILYWYYFLXOTVbzEoHydbzSjlKYXt4c	Notifikasi baru: Anda mendapat pesan baru
19	kNovz92DcZEs96ViDb2fwP3bJLAnKG	Pengguna tidak memiliki izin untuk mengakses sumber daya ini
20	WrbIREPZpjUMUkUk7hrA5Ly46be6V8	Konfigurasi sistem telah diperbarui

4. KESIMPULAN

Penelitian ini berhasil merancang dan mengimplementasikan Honey Encryption pada API Gateway untuk mengamankan komunikasi antar microservices di lingkungan cloud berbasis Amazon EC2. Validasi fungsional menunjukkan bahwa dekripsi dengan kunci yang benar selalu menghasilkan plaintext asli, sedangkan penggunaan kunci yang salah menghasilkan decoy message sintaksis valid namun kontennya keliru sehingga secara efektif mengacaukan upaya brute force attack. Analisis performa menegaskan peningkatan waktu enkripsi dan dekripsi sebanding dengan jumlah payload antara 1 hingga 100 field, dengan overhead enkripsi yang sedikit lebih tinggi namun masih dalam kisaran latensi yang dapat diterima untuk aplikasi microservices modern. Penerapan menggunakan Docker Compose pada lingkungan EC2 membuktikan kelayakan solusi ini dalam infrastruktur IaaS nyata dengan memanfaatkan skalabilitas dan fleksibilitas platform untuk mensimulasikan kondisi produksi

Kekuatan utama pendekatan ini terletak pada kemampuan decoy message yang dihasilkan ketika kunci dekripsi salah. Setiap field payload memunculkan pesan palsu unik yang mempertahankan struktur data sehingga penyerang mengalami kesulitan dalam membedakan pesan asli dari pesan palsu dan proses brute force attack menjadi terhambat. Namun penelitian ini juga memiliki batasan. Pengujian hanya dilakukan pada satu tipe instance EC2 tanpa variasi mesin. Simulasi brute force attack dibatasi pada 20 kunci acak. Dataset untuk decoy message masih bersifat statis dan belum mencakup skenario payload yang lebih beragam. Fluktuasi performa yang diamati dipengaruhi faktor non deterministik seperti caching internal Python dan variasi kondisi resource EC2 sehingga diperlukan studi lebih lanjut untuk mengisolasi variabel yang mempengaruhi hasil

Penelitian selanjutnya dapat difokuskan pada pengujian sistem di platform cloud lain seperti Google Cloud atau Azure guna mengevaluasi performa Honey Encryption di lingkungan infrastruktur yang berbeda. Selain itu, pengembangan mekanisme decoy message yang lebih dinamis dan kontekstual disarankan agar sistem lebih tangguh dalam menghadapi brute force attack dan semakin sulit dibedakan dari pesan asli.

UCAPAN TERIMAKASIH

Penulis mengucapkan terima kasih kepada Bagus Gede Krishna Yudistira dan Kadek Yota Ernanda Aryanto, selaku dosen pembimbing sekaligus Penulis Kedua dan Penulis Ketiga, atas bimbingan dan kontribusi ilmiah selama proses penulisan artikel ini.

REFERENSI

[1] L. De Lauretis, “From monolithic architecture to microservices architecture,” 2019, pp. 93–96.
[2] G. Liu, “Microservices: architecture, container, and challenges,” 2020, pp. 629–635.
[3] R. Sood dan H. Kaur, “A literature review on RSA, DES and AES encryption algorithms,” 2023, pp. 57–63.
[4] R. Saini dan R. Behl, “An introduction to AWS EC2 (elastic compute cloud),” 2020, pp. 99–102.
[5] S. K. A, Paramartha, A. A. G. Y, dan Wijaya, “Implementasi google drive cloud storage pada sistem repositori AL-Daring,” SINTECH Journal , vol. 3, Art. no. 1, 2020.
[6] K. S. M. Wikrama, “DDoS attack using GoldenEye, DAVOSET, and PyLoris tools,” CoreIT , vol. 9, Art. no. 2, 2023.
[7] T. Yarygina dan B. A. H, “Overcoming security challenges in microservice architectures,” 2018, pp. 11–20.
[8] J. M. A, “Perbandingan DDoS attack menggunakan tools LOIC, HOIC dan HULK dalam melakukan penyerangan pada suatu website,” JINTEKS (Jurnal Informatika Teknologi dan Sains) , vol. 4, Art. no. 4, 2022.
[9] M. H. A. Jacinto dan M. F. H, “Enhanced security implementation of hybrid image steganography technique using low-contrast LSB and AES-CBC cryptography,” International Journal of Advanced Computer Science and Applications , vol. 13, Art. no. 8, 2022.
[10] R. Lindholm, “Honey Encryption: implementation challenges and solutions,” 2019.
[11] O. A. E, A. Jantan, dan A. O. I, “A comprehensive review of honey encryption scheme,” Indonesian Journal of Electrical Engineering and Computer Science , vol. 13, Art. no. 2, 2019.
[12] K. V. R, “Honey encryption based hybrid cryptographic Algorithm:A fusion ensuring enhanced security,” 2020, pp. 490–494.
[13] T. Win dan Moe, “Protecting private data using improved honey encryption and honeywords generation algorithm,” Advances in Science, Technology and Engineering Systems Journal , vol. 3, Art. no. 5, 2018.
[14] A. A. M. Gharbi dan N. A. S, “Honey encryption security techniques: A review paper,” Al-Rafidain Journal of Computer Sciences and Mathematics (RJCM) , vol. 16, Art. no. 1, 2022.
[15] S. Jain, “Honey2Fish - An enhanced hybrid encryption method for password and messages,” 2022.
[16] R. Sahu dan A. M. S, “Securing messages from brute force attack by combined approach of honey encryption and blowfish,” International Research Journal of Engineering and Technology (IRJET) , vol. 4, Art. no. 9, 2017.
[17] I. M. P. Utama, “Perbandingan tools vulnerability scanning pada pengujian sebuah website,” Jurnal Informatik , vol. 18, Art. no. 3, 2022.
[18] C. Tan, X. Deng, dan L. Zhang, “Identification of block ciphers under CBC mode,” Procedia Computer Science , vol. 131, pp. 65–71, 2018.

[19] H. M. E, “Comparative study of several operation modes of AES algorithm for encryption ECG biomedical signal,” International Journal of Electrical and Computer Engineering (IJECE) , vol. 9, Art. no. 6, 2019.
[20] S. K. A dan Saskara, G. A. J, “Kriptografi simetris RC4 pada transaksi online booking engine system,” Jurnal Pendidikan Teknologi dan Kejuruan , vol. 17, Art. no. 2, 2020.
[21] I. M. P. Utama, “Analisis perbandingan kinerja tool website directory brute force dengan target website DVWA,” Jurnal Informatik , vol. 18, Art. no. 3, 2022.
[22] K. Ravindranadh, “Data migration in cloud computing using honey encryption,” International Journal of Engineering & Technology , vol. 7, Art. no. 2.8, 2018.
[23] N. Mehta dan N. M. V, “An approach of security on cloud computing using honey encryption technique,” International Journal of Current Science , vol. 14, Art. no. 2, 2024.
[24] Saskara, G. A. J, Indrawan, I. P. O, dan P. P. M, “Keamanan jaringan komputer nirkabel dengan captive portal dan WPA/WPA2 di politeknik ganesha guru,” Jurnal Pendidikan Teknologi dan Kejuruan , vol. 16, Art. no. 2, 2019.
[25] K. V. R. Choudhary, V. P. K, dan P. Rai, “A walkthrough of amazon elastic compute cloud (amazon EC2): A review,” International Journal for Research in Applied Science and Engineering Technology , vol. 9, Art. no. 11, 2021.
[26] E. Mok, A. Samsudin, dan F. T. S, “Implementing the honey encryption for securing public cloud data storage,” 2017.
[27] J. Veijanen, “Implementation of security best practices on AWS Cloud,” 2020.