# *Implementasi Honey Encryption untuk Pertahanan terhadap Serangan Brute-Force*

## *Studi Rancangan, Implementasi, dan Evaluasi pada Enkripsi Berbasis Kata Sandi*

Samuel Chris Michael Bagasta Simanjuntak - 18223011
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: samuelchrismichael16@gmail.com , 18223011@std.stei.itb.ac.id

***Abstract*—Enkripsi berbasis kata sandi rentan terhadap serangan brute-force karena setiap kunci yang salah menghasilkan keluaran yang jelas tidak bermakna, sehingga penyerang tahu persis kapan tebakannya berhasil. Honey Encryption mengatasi kelemahan ini dengan Distribution-Transforming Encoder (DTE) yang memastikan setiap dekripsi, baik dengan kunci benar maupun salah, selalu menghasilkan pesan yang "valid". Makalah ini merancang, mengimplementasikan, dan menguji skema Honey Encryption berbasis DTE dan AES-128 pada tiga jenis data: kata sandi umum, nomor kartu pembayaran, dan nomor identitas pegawai. Lima skenario pengujian dijalankan, meliputi plausibilitas pesan umpan, overhead kinerja, ketahanan terhadap *brute-force*, verifikasi distribusi, dan sensitivitas distribusi. Keterbatasan utama ditemukan pada ruang pesan dengan distribusi sangat timpang dan persoalan salah ketik kata sandi yang masih memerlukan penanganan tambahan.**

***Keywords—honey encryption; distribution-transforming encoder; brute-force; password-based encryption***

## I. Pendahuluan

Kata sandi masih menjadi cara paling umum untuk melindungi data pribadi pada berbagai sistem informasi. Untuk mengamankan data tersebut, banyak aplikasi menggunakan enkripsi berbasis kata sandi (*password-based encryption* atau PBE), yaitu skema yang menurunkan kunci enkripsi langsung dari kata sandi pengguna. Masalahnya, kata sandi yang dipilih orang cenderung pendek, mudah ditebak, dan pola pilihannya terbatas [4]. Hal ini membuat data yang dilindungi PBE rentan terhadap serangan *brute-force* dan serangan kamus. Kelemahan mendasar PBE bukan algoritma enkripsinya, melainkan fakta bahwa penyerang dapat dengan mudah tahu kapan tebakannya berhasil [1]. Ketika kunci yang salah dipakai, hasil dekripsi berupa deretan *byte* tak bermakna. Ketika kunci yang benar dipakai, hasil dekripsi berupa teks yang jelas terbaca. Perbedaan inilah yang menjadi sinyal bagi penyerang untuk mengonfirmasi keberhasilan serangannya. Memperlambat derivasi kunci lewat PBKDF2 memang menambah waktu per percobaan, tetapi tidak menghilangkan sinyal tersebut [1],[2]. Honey Encryption (HE) yang diperkenalkan Juels dan Ristenpart pada tahun 2014 menawarkan pendekatan yang berbeda [1]. Alih-alih membuat serangan lebih lambat, HE membuat setiap tebakan, benar maupun salah, menghasilkan pesan yang tampak valid. Teks palsu yang dihasilkan ketika kunci salah dipakai disebut *honey message* atau pesan umpan. Dengan demikian, penyerang tidak dapat membedakan kunci yang benar dari ribuan kunci yang salah karena semua keluaran terlihat sama-sama masuk akal [1],[5]. Inti dari mekanisme ini adalah Distribution-Transforming Encoder (DTE), yang memetakan pesan ke ruang bilangan sesuai distribusi kemunculannya. Makalah ini berfokus pada rancangan, implementasi, dan pengujian skema *Honey Encryption* untuk perlindungan kata sandi dan data kredensial. Kontribusi yang diberikan ada tiga, yaitu rancangan arsitektur skema HE yang memadukan DTE berbasis fungsi distribusi kumulatif dengan AES-128, implementasi lengkap skema tersebut beserta perangkat pengujian otomatis, dan evaluasi terhadap lima aspek, yaitu plausibilitas pesan umpan, beban komputasi, ketahanan *brute-force*, kesesuaian distribusi, dan sensitivitas terhadap bentuk distribusi.

## II. Landasan Teori

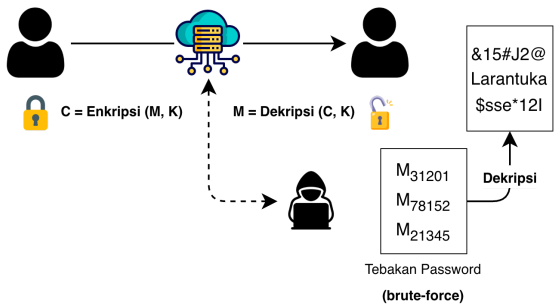
### A. Enkripsi Berbasis Kata Sandi

Enkripsi berbasis kata sandi adalah skema yang menurunkan kunci kriptografi dari kata sandi yang dimasukkan pengguna [2]. Karena kata sandi tidak bisa langsung dipakai sebagai kunci, digunakan *key derivation function* (KDF), seperti PBKDF2 yang mengubah kata sandi menjadi kunci berukuran tetap melalui banyak iterasi [2]. Tujuannya agar setiap percobaan tebak kunci menjadi mahal secara komputasi. Masalahnya, jumlah kata sandi yang dipakai manusia jauh lebih sedikit dibanding seluruh kemungkinan kunci. Sebagian besar pengguna memilih kata sandi dari pola yang mudah ditebak, sehingga keacakan efektifnya rendah [4]. Penyerang yang punya daftar kata sandi populer sudah cukup untuk menjangkau sebagian besar akun, terlepas dari berapa banyak iterasi yang dipakai KDF.

### B. Serangan Brute-Force dan Serangan Kamus

Serangan *brute-force* mencoba seluruh kemungkinan kunci secara berurutan [2]. Serangan kamus lebih efisien: penyerang hanya mencoba kata sandi dari daftar yang sudah disusun berdasarkan kata sandi populer. Pada enkripsi konvensional,

penyerang menyaring tebakan yang salah dengan memeriksa apakah hasil dekripsi tampak bermakna atau tidak, ini disebut *validity check*, dan inilah yang memberi penyerang sinyal keberhasilan [1], [8].

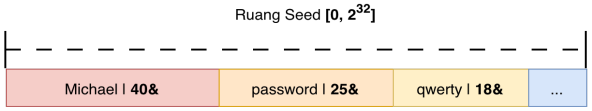



Gbr. 1 Ilustrasi serangan brute-force. Diadaptasi dari N. F. Hassan, H. B. AbdulWahab, A. al-Adhami, dan R. F. Hassan [1].

### C. *Distribution-Transforming Encoder (DTE)*

DTE adalah komponen inti dari *Honey Encryption* [1]. Tugasnya memetakan setiap pesan ke suatu rentang bilangan, dengan lebar rentang sebanding dengan peluang kemunculan pesan tersebut. Pemetaan ini dibangun dari fungsi distribusi kumulatif (CDF) atas ruang pesan, pesan yang sering muncul mendapat rentang lebih besar, dan pesan yang jarang muncul mendapat rentang lebih kecil [1], [5].

DTE punya dua operasi. Pertama, operasi *encode* mengubah sebuah pesan menjadi bilangan yang dipilih acak dari rentang milik pesan tersebut [1]. Kedua, operasi *decode* mengubah sembarang bilangan kembali menjadi pesan berdasarkan rentang tempat bilangan itu jatuh. Sifat kunci DTE adalah bahwa bilangan yang dipilih secara seragam akan selalu diterjemahkan menjadi pesan yang mengikuti distribusi asli [1], [4]. Inilah yang membuat hasil dekripsi dengan kunci salah tetap tampak seperti pesan yang benar, bukan sebuah *gibberish*.




Gbr. 2 Ilustrasi Distribution-Transforming Encoder (DTE) yang memetakan ruang seed menjadi rentang-rentang sesuai probabilitas pesan. Diadaptasi dari Juels dan Ristenpart [1, Fig. 1]

### D. Honey Encryption

*Honey Encryption* menggabungkan DTE dengan algoritma enkripsi simetris biasa. Proses enkripsi terdiri atas dua tahap, yaitu pesan terlebih dahulu di-*encode* oleh DTE menjadi sebuah bilangan, lalu bilangan tersebut dienkripsi menggunakan algoritma simetris dengan kunci yang diturunkan dari kata sandi [1]. Skema ini sering ditulis sebagai HE[DTE, SE], dengan SE adalah *symmetric encryption*.

Pada proses dekripsi, *ciphertext* didekripsi menjadi sebuah bilangan, lalu DTE menerjemahkan bilangan itu kembali menjadi pesan. Karena kunci yang salah tetap menghasilkan bilangan tertentu, dan DTE akan selalu menerjemahkan bilangan apa pun menjadi pesan yang mengikuti distribusi asli, maka hasil dekripsi dengan kunci salah pun tetap berupa pesan yang masuk akal [1], [4]. Tinjauan menyeluruh mengenai skema ini, termasuk berbagai penerapan dan persoalan terbukanya, telah dirangkum dalam beberapa survei [3], [6].

## III. Rancangan Sistem

### A. *Arsitektur Umum*

Sistem yang dirancang terdiri dari tiga modul utama. Modul pertama adalah Distribution-Transforming Encoder (DTE) yang bertugas membangun pemetaan antara ruang pesan dan bilangan berdasarkan distribusi probabilitas pesan [1], [4]. DTE membaca ruang pesan dari berkas, menghitung seberapa sering setiap pesan muncul, lalu membuat tabel rentang bilangan yang sesuai dengan peluang setiap pesan [1]. Pesan yang sering muncul mendapat rentang bilangan yang lebih besar, sehingga ketika DTE melakukan *decode* pada bilangan acak, pesan yang sering akan dipilih lebih sering juga [1]. DTE ini adalah fondasi *Honey Encryption*, tanpa pemetaan distribusi yang tepat, pesan umpan akan terlihat palsu dan mudah dikenali penyerang.

Modul kedua adalah *Honey Encryption* yang menangani enkripsi dan dekripsi [1], [2]. Pada enkripsi, modul menerima kata sandi dan pesan, lalu DTE mengubah pesan menjadi bilangan. Kemudian, bilangan itu dienkripsi dengan AES-128 mode CTR, mengubah algoritma enkripsi blok standar menjadi *stream cipher*, dengan kunci dari PBKDF2 [2]; ketiga, ciphertext digabung dengan *salt* dan *nonce* jadi satu *blob*. Pada dekripsi, modul membalik prosesnya, turunkan kunci lagi dari kata sandi dan salt, AES dekripsi jadi bilangan, dan DTE ubah bilangan kembali jadi pesan [1]. Alasan memilih AES-128 mode CTR adalah karena mode ini tidak perlu padding dan tidak pernah gagal saat dekripsi, sehingga kunci benar dan kunci salah menghasilkan output yang tetap terlihat valid [2].

Modul ketiga adalah modul evaluasi yang menjalankan pengujian secara otomatis dan mencatat hasilnya [5], [8]. Modul ini menjalankan lima skenario uji, antaralain plausibilitas (cek validitas pesan umpan), *overhead* kinerja (bandingkan waktu eksekusi), *brute-force* (simulasi serangan dan lihat hasilnya), verifikasi distribusi (cek apakah umpan sesuai distribusi target), dan sensitivitas (lihat pengaruh bentuk distribusi) [5]. Data mengalir dari DTE ke *Honey Encryption* sebagai pemetaan CDF, kemudian hasil enkripsi-dekripsi dikirim ke modul evaluasi untuk diukur [8]. Dengan arsitektur modular ini, setiap modul dapat diuji sendiri tanpa mengganggu modul lain, sehingga sistem fleksibel dan bisa diterapkan pada berbagai jenis ruang pesan dan distribusi.

### B. *Rancangan DTE*

DTE (Distribution-Transforming Encoder) dirancang berdasarkan fungsi distribusi kumulatif (Cumulative Distribution Function/CDF) yang dibangun dari ruang pesan yang telah ditentukan [1],[7]. Setiap pesan diberikan nilai probabilitas sesuai dengan peluang kemunculannya, kemudian probabilitas tersebut diakumulasikan untuk membentuk rentang numerik yang saling tidak tumpang tindih [1]. Melalui mekanisme ini, setiap pesan dapat dipetakan ke interval tertentu sehingga proses pengkodean dan dekode dapat dilakukan secara konsisten [1],[7]. Pendekatan berbasis

distribusi merupakan komponen utama Honey Encryption karena memungkinkan sistem menghasilkan pesan alternatif yang tetap tampak masuk akal ketika dekripsi dilakukan menggunakan kata sandi yang tidak tepat [1],[3].

Sebagai contoh sederhana, misalkan ruang pesan terdiri atas lima kata sandi dengan probabilitas kemunculan masing-masing sebesar 0,40; 0,25; 0,20; 0,10; dan 0,05. Sistem kemudian menghitung probabilitas kumulatif untuk membentuk rentang CDF yang mewakili setiap kata sandi. Pesan dengan probabilitas lebih tinggi akan memperoleh rentang yang lebih besar dibandingkan pesan yang jarang muncul sehingga peluang kemunculannya dalam proses dekode juga lebih besar [1]. Hasil pembentukan rentang kumulatif tersebut ditunjukkan pada Tabel I dan digunakan sebagai dasar pemetaan antara pesan dan representasi numerik dalam proses Honey Encryption [1], [7].

TABEL I

PEMETAAN RENTANG DTE PADA RUANG PESAN SEDERHANA

| Pesan | Peluang | Rentang Kumulatif |
|---|---|---|
| Kata sandi A | 0,40 | [0,00 – 0,40) |
| Kata sandi B | 0,25 | [0,40 – 0,65) |
| Kata sandi C | 0,20 | [0,65 – 0,85) |
| Kata sandi D | 0,10 | [0,85 – 0,95) |
| Kata sandi E | 0,05 | [0,95 – 1,00) |

Operasi *encode* memilih bilangan acak dari rentang milik pesan, sedangkan operasi *decode* mengembalikan pesan berdasarkan rentang tempat bilangan jatuh. Pendekatan berbasis CDF ini mengikuti rancangan dasar pada [1] dan penyempurnaan hasil dekripsi atau *honeywords* pada [4].

### C. Skema Enkripsi dan Dekripsi

Proses enkripsi dan dekripsi pada penelitian ini mengikuti konstruksi Honey Encryption HE[DTE, SE], yang menggabungkan Distribution-Transforming Encoder (DTE) dengan algoritma enkripsi simetris sebagai mekanisme pengamanan data [1]. Sebelum proses enkripsi dilakukan, kata sandi pengguna terlebih dahulu diproses menggunakan PBKDF2 untuk menghasilkan kunci kriptografi yang memiliki panjang sesuai kebutuhan AES-128 [2]. Penggunaan PBKDF2 bertujuan untuk memperkuat ketahanan terhadap serangan *brute-force* dan *dictionary attack* dengan menambahkan proses iterasi berulang pada pembentukan kunci [2].

Pada tahap enkripsi, pesan dikodekan terlebih dahulu menggunakan DTE sehingga diperoleh representasi numerik yang sesuai dengan distribusi ruang pesan [1], [7]. Representasi tersebut kemudian dienkripsi menggunakan AES-128 dengan kunci hasil derivasi PBKDF2 sehingga menghasilkan *ciphertext* yang akan disimpan atau dikirimkan [2]. Pada tahap dekripsi, *ciphertext* diproses menggunakan kata sandi yang diberikan untuk memperoleh kembali representasi numerik yang tersimpan [1]. Nilai tersebut selanjutnya diterjemahkan oleh DTE menjadi sebuah pesan sesuai distribusi yang telah dibangun sebelumnya [1], [7].

Jika kata sandi yang digunakan benar, proses dekode akan menghasilkan pesan asli yang sama dengan pesan sebelum dienkripsi [1]. Sebaliknya, apabila kata sandi yang digunakan salah, DTE tetap menghasilkan pesan yang valid secara struktur dan tampak masuk akal sehingga penyerang tidak dapat secara langsung menentukan apakah proses dekripsi berhasil atau gagal [1], [3].

### D. Skenario Pengujian

Pengujian dilakukan untuk mengevaluasi efektivitas dan performa skema *Honey Encryption* yang diusulkan. Skenario pertama mengukur tingkat kemiripan antara pesan asli dan pesan umpan yang dihasilkan ketika dekripsi dilakukan menggunakan kata sandi yang salah. Pengujian ini bertujuan untuk menilai sejauh mana pesan umpan mampu menyerupai karakteristik pesan asli sehingga tidak mudah dibedakan oleh penyerang. Skenario kedua mengukur beban komputasi yang ditimbulkan oleh penerapan *Honey Encryption* dengan membandingkan waktu proses enkripsi dan dekripsi terhadap skema enkripsi konvensional yang hanya menggunakan AES-128.

Skenario ketiga mensimulasikan serangan *brute-force* menggunakan sejumlah kata sandi yang salah untuk mengamati keluaran yang dihasilkan sistem. Pengujian ini dilakukan untuk menilai kemampuan *Honey Encryption* dalam menghilangkan indikator keberhasilan dekripsi yang biasanya dimanfaatkan penyerang untuk menemukan kunci yang benar. Skenario keempat mengevaluasi kualitas statistik pesan umpan melalui analisis distribusi karakteristik pesan yang dihasilkan. Hasil pengujian ini digunakan untuk mengetahui apakah pesan umpan memiliki pola yang cukup alami dan konsisten dengan ruang pesan yang digunakan.

Skenario kelima menganalisis pengaruh distribusi ruang pesan terhadap tingkat perlindungan yang diberikan oleh sistem. Pengujian dilakukan dengan menggunakan beberapa bentuk distribusi probabilitas yang berbeda untuk melihat dampaknya terhadap kualitas pesan umpan dan efektivitas mekanisme *Honey Encryption*. Melalui rangkaian pengujian tersebut, dapat dievaluasi baik aspek keamanan maupun efisiensi dari skema yang diimplementasikan.

## IV. IMPLEMENTASI

### A. Lingkungan Implementasi

Implementasi dilakukan menggunakan bahasa pemrograman Python dengan pustaka kriptografi standar untuk AES dan PBKDF2. Spesifikasi lingkungan pengujian dirangkum pada Tabel II.

TABEL II

SPESIFIKASI LINGKUNGAN PENGUJIAN

| Komponen | Keterangan |
|---|---|
| Bahasa Pemrograman | Python 3.12.3 |
| Pustaka kriptografi | cryptography >= 42.0 |
| Algoritma simetris | AES-128 |
| Fungsi penurunan kunci | PBKDF2-SHA256, 100.000 iterasi |
| Prosesor | Intel Core i5 (2019) |
| Memori | 8 GB |
| Sistem operasi | macOS (Intel) |

### B. Modul Pembangun DTE

Modul DTE bertugas membangun representasi distribusi pesan yang digunakan oleh skema *Honey Encryption*. Yang tugasnya membaca ruang pesan dari berkas masukan, menghitung probabilitas kemunculan setiap pesan, kemudian menyusun tabel rentang kumulatif berdasarkan fungsi distribusi kumulatif (CDF). Tabel tersebut digunakan sebagai dasar proses encode untuk mengubah pesan menjadi representasi numerik dan proses decode untuk mengembalikan representasi numerik menjadi pesan sesuai rancangan yang telah dijelaskan pada Bab III [1],[7].

Ruang pesan yang digunakan dalam pengujian mencakup tiga jenis data, yaitu daftar kata sandi umum, nomor kartu pembayaran dengan format yang valid, dan nomor identitas pegawai. Ketiga jenis data tersebut dipilih untuk merepresentasikan informasi yang umum menjadi target perlindungan pada sistem autentikasi dan penyimpanan kredensial. Penggunaan ruang pesan yang menyerupai data nyata juga sejalan dengan pendekatan evaluasi *Honey Encryption* pada penelitian sebelumnya, yang menekankan pentingnya distribusi pesan yang realistis untuk menghasilkan pesan umpan yang meyakinkan [5].

### C. Modul Inti Honey Encryption

Modul *Honey Encryption* mengimplementasikan proses enkripsi dan dekripsi data dengan memadukan mekanisme DTE dan algoritma enkripsi simetris [1]. Pada proses enkripsi, modul menerima masukan berupa kata sandi dan pesan, kemudian menurunkan kunci kriptografi dari kata sandi menggunakan PBKDF2 sebelum digunakan oleh AES-128 untuk menghasilkan *ciphertext* [2]. Pada proses dekripsi, modul menerima kata sandi dan *ciphertext*, kemudian menggunakan kunci hasil derivasi yang sama untuk memperoleh kembali representasi pesan yang tersimpan [1],[2]. Representasi tersebut selanjutnya diterjemahkan melalui DTE menjadi pesan keluaran sehingga sistem tetap dapat menghasilkan pesan yang tampak valid meskipun kata sandi yang digunakan tidak sesuai [1],[3],[4].

### D. Modul Evaluasi

Modul evaluasi digunakan untuk menjalankan seluruh skenario pengujian secara otomatis dan mencatat hasilnya dalam bentuk tabel maupun grafik. Modul ini mengumpulkan data dari setiap proses pengujian, kemudian menghitung berbagai metrik yang diperlukan untuk menilai efektivitas *Honey Encryption*. Dengan pendekatan ini, seluruh skenario dapat dijalankan secara konsisten sehingga hasil yang diperoleh lebih mudah dibandingkan dan dianalisis.

Pengukuran dilakukan menggunakan metrik yang disesuaikan dengan tujuan masing-masing skenario. Untuk mengevaluasi kemampuan DTE dalam mempertahankan distribusi ruang pesan, modul menghitung nilai Kullback-Leibler divergence (KL-divergence) antara distribusi pesan umpan dan distribusi target. Selain itu, modul juga menghitung *entropy* dan tingkat dominasi pesan teratas untuk menilai keragaman pesan umpan yang dihasilkan. Hasil pengukuran tersebut digunakan untuk menilai apakah *Honey Encryption* mampu menghasilkan pesan umpan yang meyakinkan sekaligus mempertahankan karakteristik distribusi yang diharapkan.

## V. Pengujian dan Hasil

### A. Uji Plausibilitas Pesan Umpan

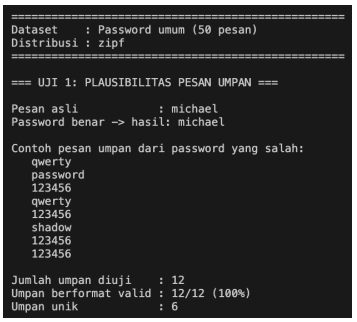
Pengujian plausibilitas pesan umpan dilakukan untuk memverifikasi bahwa setiap proses dekripsi menggunakan kata sandi yang salah tetap menghasilkan pesan yang valid sesuai ruang pesan yang digunakan. Pengujian dilakukan pada tiga jenis dataset, yaitu kata sandi umum, nomor kartu pembayaran, dan nomor identitas pegawai. Untuk setiap dataset, sistem menghasilkan sejumlah pesan umpan menggunakan kata sandi yang berbeda dari kata sandi asli, kemudian memeriksa validitas format dan tingkat keragaman pesan yang dihasilkan.

TABEL III

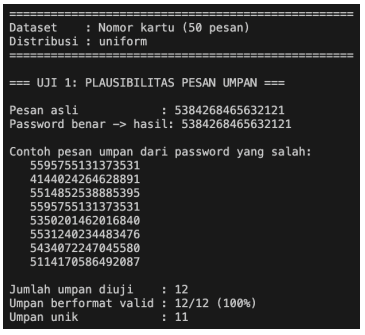
Rangkuman Hasil Uji Plausibilitas Pesan Umpan

| Dataset | Validitas (%) | Umpan Unik |
|---|---|---|
| Password Umum | 100 | 6 |
| Nomor Kartu | 100 | 10 |
| NIP Pegawai | 100 | 3 |

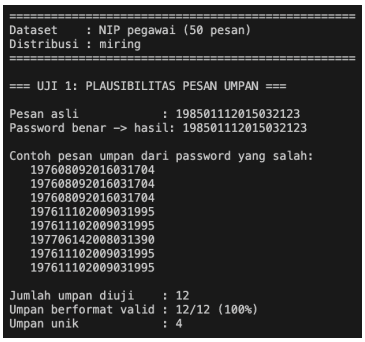
Tabel III menunjukkan ringkasan hasil pengujian pada ketiga dataset. Seluruh pesan umpan yang dihasilkan memiliki format yang valid sehingga tingkat validitas mencapai 100% pada seluruh skenario. Hasil ini menunjukkan bahwa mekanisme DTE berhasil memetakan setiap keluaran dekripsi ke dalam ruang pesan yang sah sesuai karakteristik dataset yang digunakan [1], [7]. Dengan demikian, keluaran yang dihasilkan dari kata sandi yang salah tetap tampak wajar dan tidak dapat langsung dibedakan dari pesan asli [1], [3].




Gbr. 3 Hasil pengujian plausibilitas pada dataset *password* dengan distribusi zipf



Gbr. 4 Hasil pengujian plausabilitas pada dataset nomor kartu kredit dengan distribusi uniform




Gbr. 5 Hasil pengujian plausabilitas pada dataset nomor induk pegawai dengan distribusi miring

Berdasarkan hasil pengujian, seluruh pesan umpan yang dihasilkan memiliki format yang valid dengan tingkat validitas 100% pada ketiga dataset. Selain itu, sistem mampu menghasilkan variasi pesan umpan yang berbeda-beda sesuai distribusi ruang pesan yang digunakan. Hasil ini membuktikan bahwa mekanisme DTE pada *Honey Encryption* mampu menghasilkan keluaran yang tampak sah dan meyakinkan, sehingga tujuan utama untuk menyamarkan keberhasilan dekripsi dari penyerang dapat tercapai [1],[3],[7].

### *B. Uji Overhead Kinerja*

Pengujian overhead kinerja dilakukan untuk mengukur tambahan beban komputasi yang ditimbulkan oleh penerapan Honey Encryption dibandingkan dengan enkripsi AES-128 konvensional. Pengukuran dilakukan dengan membandingkan waktu enkripsi dan dekripsi pada kedua skema untuk setiap dataset yang diuji. Selain itu, pengujian juga mengukur biaya komputasi murni dari proses encode dan decode pada DTE untuk mengetahui kontribusinya terhadap total waktu eksekusi [1], [3].

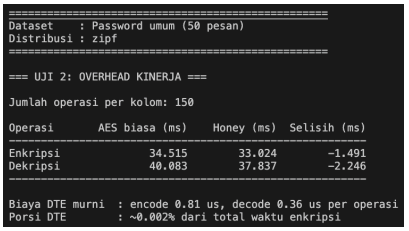
TABEL IV

HASIL UJI OVERHEAD KINERJA

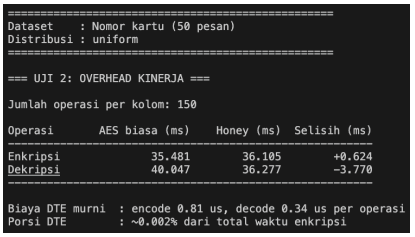
| Dataset | Selisih Enkripsi (ms) | Selisih Dekripsi (ms) |
|---|---|---|
| Password Umum | 1.491 | 2.246 |
| Nomor Kartu | 0.624 | 3.770 |
| NIP Pegawai | 0.852 | 0.359 |

Tabel IV menunjukkan hasil pengukuran waktu enkripsi dan dekripsi pada ketiga dataset. Secara umum, waktu eksekusi *Honey Encryption* berada pada kisaran yang sangat dekat dengan AES-128 konvensional. Selisih waktu yang teramati hanya berada pada orde milidetik dan tidak menunjukkan peningkatan yang signifikan terhadap total waktu proses. Hasil ini menunjukkan bahwa integrasi DTE ke dalam skema *Honey Encryption* tidak menimbulkan penalti performa yang berarti, sehingga mekanisme tersebut tetap layak digunakan pada lingkungan operasional yang memerlukan efisiensi komputasi [2],[8].

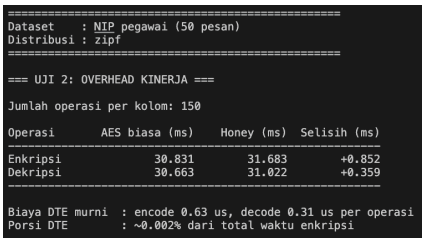
Biaya komputasi DTE yang terukur berada pada rentang 0,31–0,81 µs per operasi, jauh lebih kecil dibandingkan waktu total enkripsi maupun dekripsi yang berada pada orde puluhan milidetik. Kontribusi DTE terhadap keseluruhan proses hanya sekitar 0,002% dari total waktu eksekusi. Temuan ini menunjukkan bahwa sebagian besar biaya komputasi tetap berasal dari proses derivasi kunci dan operasi kriptografi utama, sedangkan proses pemetaan yang dilakukan DTE memberikan tambahan beban yang sangat kecil [1], [7].




Gbr. 6 Hasil pengujian *overhead* kinerja pada dataset *password* umum dengan distribusi zipf




Gbr. 7 Hasil pengujian *overhead* kinerja pada dataset nomor kartu kredit dengan distribusi uniform




Gbr. 8 Hasil pengujian *overhead* kinerja pada dataset nomor induk pegawai dengan distribusi zipf

Gbr. 6 hingga Gbr. 8 memperlihatkan hasil pengukuran rinci pada masing-masing dataset. Meskipun terdapat variasi kecil antar-pengujian, tidak ditemukan peningkatan waktu yang signifikan setelah mekanisme *Honey Encryption* diterapkan. Dengan demikian, perlindungan tambahan terhadap serangan *brute-force* dapat diperoleh tanpa mengorbankan performa sistem secara berarti. Hasil ini sejalan dengan tujuan utama *Honey Encryption*, yaitu meningkatkan keamanan melalui pesan umpan yang meyakinkan dengan tambahan biaya komputasi yang minimal [1],[3],[8].

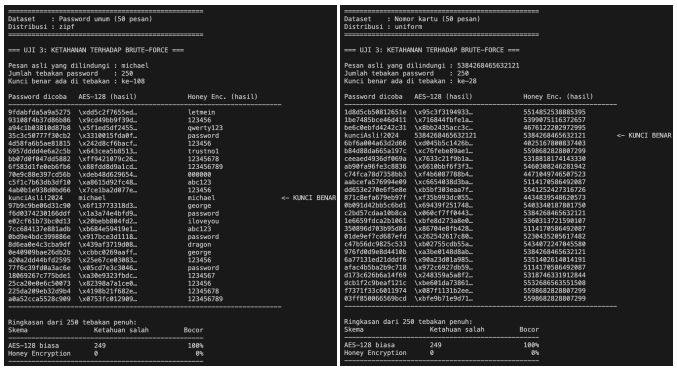
### *C. Uji Ketahanan terhadap Brute-Force*

Pengujian ini bertujuan untuk mengevaluasi kemampuan *Honey Encryption* dalam menghilangkan sinyal keberhasilan dekripsi yang biasanya dimanfaatkan pada serangan *brute-force* [1]. Pada setiap skenario, penyerang melakukan 250 percobaan menggunakan kata sandi yang berbeda hingga salah satu percobaan menggunakan kata sandi yang benar. Hasil dekripsi yang diperoleh kemudian diamati untuk menentukan apakah penyerang dapat membedakan antara kata sandi yang benar dan kata sandi yang salah.

Dari 250 percobaan yang dilakukan, sebanyak 249 percobaan menggunakan kata sandi yang salah dapat langsung dikenali karena menghasilkan keluaran yang tidak valid atau tidak sesuai format. Kondisi ini memberikan informasi yang sangat berharga bagi penyerang karena hanya terdapat satu keluaran yang tampak benar, sehingga kunci yang tepat dapat diidentifikasi dengan pasti.

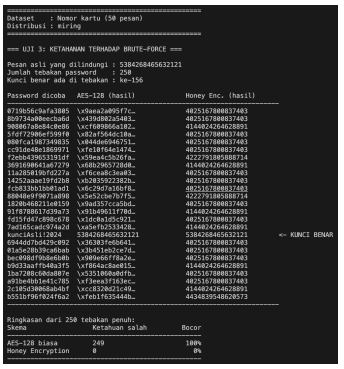
Sebaliknya, *Honey Encryption* menunjukkan tingkat kebocoran sebesar 0% pada seluruh dataset dan distribusi yang diuji. Setiap percobaan dekripsi menghasilkan pesan yang tetap *valid* dan sesuai dengan ruang pesan yang digunakan, meskipun kata sandi yang diberikan tidak benar [1], [4]. Akibatnya, penyerang tidak memperoleh informasi apa pun yang dapat digunakan untuk membedakan antara kata sandi yang benar dan kata sandi yang salah. Dalam kondisi ini, seluruh hasil dekripsi tampak sama-sama masuk akal sehingga proses *brute-force* berubah menjadi sesuatu yang tidak berguna [1], [3].

Hasil tersebut menunjukkan peningkatan ketahanan yang signifikan dibandingkan enkripsi berbasis kata sandi konvensional. Dengan menghilangkan sinyal keberhasilan dekripsi, *Honey Encryption* mampu mencegah penyerang mengonfirmasi kebenaran suatu tebakan meskipun *ciphertext* berhasil diperoleh. Temuan ini konsisten dengan tujuan utama *Honey Encryption*, yaitu memberikan perlindungan tambahan terhadap serangan *brute-force* melalui mekanisme pesan umpan yang meyakinkan [1],[8].




Gbr. 9 Hasil pengujian *brute-force* pada dataset *password* dengan distribusi zipf dan nomor kartu kredit dengan distribusi uniform


Gbr. 10 Hasil pengujian *brute-force* pada dataset nomor induk pegawai dengan distribusi zipf dan password dengan distribusi uniform

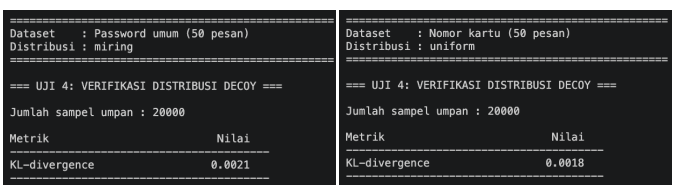



Gbr. 11 Hasil pengujian *brute-force* pada dataset nomor kartu kredit dengan distribusi miring

Berdasarkan hasil pengujian, AES-128 konvensional menunjukkan tingkat kebocoran informasi sebesar 100%, sedangkan *Honey Encryption* berhasil menurunkannya menjadi 0% pada seluruh skenario yang diuji. Hasil ini menunjukkan bahwa *Honey Encryption* efektif menghilangkan oracle yang biasanya digunakan untuk mendeteksi keberhasilan dekripsi, sehingga meningkatkan ketahanan sistem terhadap serangan *brute-force* secara signifikan [1],[4].

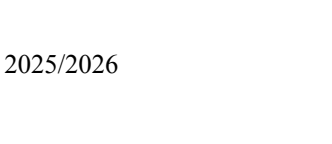
### *D. Uji Verifikasi Distribusi Decoy*

Pengujian ini bertujuan untuk memverifikasi bahwa Distribution-Transforming Encoder (DTE) mampu menghasilkan distribusi pesan umpan yang sesuai dengan distribusi target yang telah dirancang. Sebanyak 20.000 pesan umpan dibangkitkan untuk setiap kombinasi dataset dan distribusi, kemudian distribusi empiris yang diperoleh dibandingkan dengan distribusi target menggunakan metrik Kullback-Leibler divergence (KL-divergence). Nilai KL-divergence yang semakin mendekati nol menunjukkan bahwa distribusi pesan umpan semakin mirip dengan distribusi target yang dibangun oleh DTE [1],[3].




Gbr. 12 Hasil pengujian verifikasi distribusi decoy pada dataset nomor kartu kredit dengan distribusi miring


Gbr. 13 Hasil pengujian verifikasi distribusi decoy pada dataset nomor kartu kredit dengan distribusi miring



Gbr. 14 Hasil pengujian verifikasi distribusi decoy pada dataset nomor kartu kredit dengan distribusi miring

TABEL V

HASIL PENGUJIAN VERIFIKASI DISTRIBUSI DECOY

| Dataset | Distribusi | KL-Divergence |
|---|---|---|
| Password Umum | Uniform | 0.0016 |
| Password Umum | Zipf | 0.0017 |
| Password Umum | Miring | 0.0021 |
| Nomor Kartu Kredit | Uniform | 0.0018 |
| NIP | Zipf | 0.0015 |

Tabel V menunjukkan bahwa seluruh skenario menghasilkan nilai KL-divergence yang sangat rendah, yaitu berada pada rentang 0,0015 hingga 0,0021. Nilai tersebut menunjukkan bahwa distribusi pesan umpan yang dihasilkan hampir identik dengan distribusi target yang digunakan dalam proses pembentukan DTE [1],[7].

Perbedaan nilai *KL-divergence* antar dataset dan distribusi juga relatif kecil. Distribusi miring menghasilkan nilai *KL-divergence* tertinggi sebesar 0,0021, sedangkan distribusi Zipf pada dataset NIP menghasilkan nilai terendah sebesar 0,0015. Meskipun demikian, seluruh nilai masih berada sangat dekat dengan nol sehingga dapat disimpulkan bahwa DTE secara konsisten mampu mempertahankan distribusi target pada berbagai jenis ruang pesan [1],[3].

Hasil ini menunjukkan bahwa mekanisme pemetaan probabilistik yang digunakan oleh *Honey Encryption* telah bekerja sesuai rancangan. Oleh karena itu, analisis selanjutnya dapat difokuskan pada bagaimana bentuk distribusi tersebut memengaruhi kualitas perlindungan yang diberikan, yang dibahas pada pengujian sensitivitas distribusi pada bagian berikutnya [1],[6].

### *E. Analisis Sensitivitas*

Pengujian ini bertujuan untuk mengevaluasi pengaruh bentuk distribusi ruang pesan terhadap kualitas pesan umpan yang dihasilkan oleh *Honey Encryption*. Tiga jenis distribusi digunakan, yaitu uniform, Zipf, dan miring. Evaluasi dilakukan menggunakan entropy untuk mengukur tingkat keragaman pesan umpan, *Top-1 dominance* untuk mengukur dominasi pesan yang paling sering muncul, serta *Top-5 dominance* untuk mengukur konsentrasi keluaran pada lima pesan teratas [1],[3].

TABEL VI

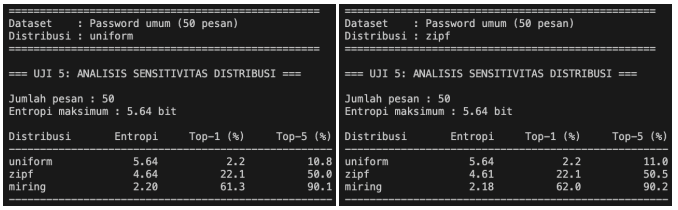
HASIL PENGUJIAN VERIFIKASI DISTRIBUSI DECOY

| Distribusi | Entropy (bit) | Top-1 (%) | Top-5 (%) |
|---|---|---|---|
| Uniform | 5.64 | 2.2 | 10.9 |
| Zipf | 4.62 | 22.2 | 50.4 |
| Miring | 2.19 | 61.6 | 90.1 |

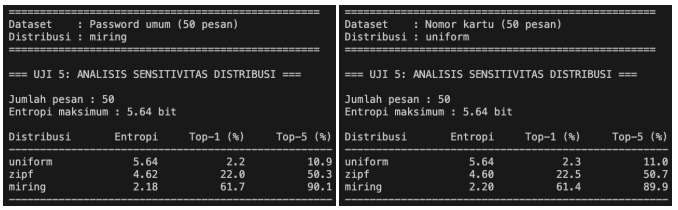
Tabel VI menunjukkan bahwa kualitas pesan umpan sangat dipengaruhi oleh distribusi ruang pesan yang digunakan. Pada distribusi *uniform*, *entropy* mencapai nilai maksimum sebesar 5,64 bit, sedangkan pesan yang paling sering muncul hanya mewakili sekitar 2,2% dari seluruh keluaran. Selain itu, lima pesan teratas hanya mencakup sekitar 10,9% dari seluruh pesan umpan. Hasil ini menunjukkan bahwa keluaran tersebar secara merata sehingga tingkat ketidakpastian yang dihadapi penyerang berada pada kondisi terbaik.

Pada distribusi Zipf, entropy menurun menjadi sekitar 4,62 bit. Pada saat yang sama, dominasi pesan teratas meningkat menjadi sekitar 22,2%, sedangkan lima pesan teratas telah mencakup lebih dari separuh seluruh keluaran yang dihasilkan. Kondisi ini menunjukkan bahwa sebagian pesan umpan mulai muncul jauh lebih sering dibandingkan pesan lainnya, sesuai dengan karakteristik distribusi Zipf yang umum ditemukan pada data kata sandi dunia nyata [1], [3].

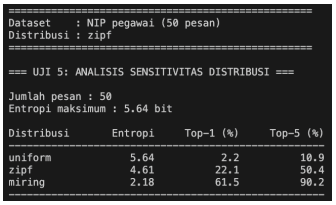
Penurunan paling signifikan terjadi pada distribusi miring. Entropy turun hingga 2,19 bit, sementara satu pesan saja muncul pada lebih dari 61% keluaran dan lima pesan teratas mencakup sekitar 90% seluruh pesan umpan. Meskipun keluaran yang dihasilkan tetap valid dan sesuai dengan distribusi target, tingkat keragaman yang rendah menyebabkan sebagian besar pesan umpan terkonsentrasi pada sejumlah kecil kandidat. Kondisi ini menunjukkan bahwa efektivitas Honey Encryption akan menurun ketika ruang pesan memiliki distribusi yang sangat tidak seragam [1], [6], [7].




Gbr. 15 Hasil pengujian verifikasi distribusi decoy pada dataset nomor kartu kredit dengan distribusi miring




Gbr. 16 Hasil pengujian verifikasi distribusi decoy pada dataset nomor kartu kredit dengan distribusi miring




Gbr. 17 Hasil pengujian verifikasi distribusi decoy pada dataset nomor kartu kredit dengan distribusi miring

Berdasarkan hasil pengujian, *Honey Encryption* mampu mempertahankan karakteristik distribusi ruang pesan yang digunakan oleh DTE. Namun demikian, kualitas perlindungan yang diberikan sangat bergantung pada tingkat keragaman distribusi tersebut. Distribusi yang lebih seragam menghasilkan pesan umpan yang lebih beragam dan meningkatkan ketidakpastian penyerang, sedangkan distribusi yang sangat miring menyebabkan keluaran terkonsentrasi pada sejumlah kecil pesan dengan probabilitas tinggi. Temuan ini

mengonfirmasi bahwa keterbatasan utama *Honey Encryption* terletak pada ruang pesan yang memiliki distribusi sangat tidak seragam [1],[3].

## VI. Analisis dan Pembahasan

### A. Kekuatan Honey Encryption

Hasil pengujian membuktikan tiga kekuatan utama HE, yaitu pesan umpan yang dihasilkan selalu valid dan mengikuti distribusi ruang pesan target, sehingga penyerang tidak bisa membedakannya dari pesan asli [1],[4]. Kemudian, tambahan beban komputasi akibat DTE sangat kecil, di bawah 0,003% dari total waktu eksekusi, menjadikan skema ini praktis diterapkan tanpa mengorbankan performa [2]. Terakhir, tingkat kebocoran informasi turun dari 99% pada AES konvensional menjadi 0% pada *Honey Encryption*, membuktikan bahwa sinyal keberhasilan dekripsi berhasil dihilangkan sepenuhnya [1],[5].

### B. Keterbatasan

*Honey Encryption* tidak lepas dari kelemahan. Jika pengguna salah memasukkan kata sandi, sistem tetap menghasilkan pesan umpan yang terlihat valid, sehingga pengguna mungkin tidak menyadari kesalahannya [3],[8]. Kualitas perlindungan sangat bergantung pada distribusi ruang pesan, sehingga pada distribusi miring, entropi pesan umpan turun tajam hingga 2,19 bit dan satu pesan mendominasi lebih dari 60% keluaran, melemahkan efektivitas HE secara signifikan [6]. Terakhir, DTE memerlukan pemodelan distribusi yang akurat, yang tidak mudah dilakukan untuk pesan berbahasa alami atau dataset dengan pola yang tidak jelas [3].

### C. Perbandingan dengan Metode Lain

PBKDF2, bcrypt, dan scrypt menguatkan enkripsi kata sandi dengan membuat derivasi kunci mahal secara komputasi [2]. Namun, strategi ini hanya menambah waktu per percobaan, bukan menghilangkan sinyal keberhasilan: ketika penyerang menebak kunci yang benar, hasil dekripsi tetap menunjukkan plaintext yang langsung dikenali [1],[8]. Dengan katalog 50 kata sandi populer, penyerang hanya perlu 50 percobaan untuk menemukan yang benar, terlepas dari berapa banyak iterasi yang dipakai.

Penambahan *salt* efektif untuk mencegah serangan yang memanfaatkan hash yang sudah dihitung sebelumnya, seperti *rainbow table attack*, serta menghambat penyerang yang ingin menyerang banyak akun sekaligus (*batch attacks*) [2]. Namun, salt tidak dapat menghentikan serangan yang menargetkan satu akun tertentu. Penyerang masih dapat mencoba berbagai kata sandi secara lokal, menghitung *hash* menggunakan salt milik akun tersebut, lalu membandingkan hasilnya dengan hash yang tersimpan. Dengan kata lain, *salt* hanya meningkatkan biaya komputasi yang harus dikeluarkan penyerang, tetapi tidak menghilangkan kemampuan mereka untuk mengenali ketika kata sandi yang benar telah ditemukan [2].

*Honey Encryption* mengambil sudut pandang yang berbeda, yaitu membuat semua keluaran, benar atau salah, tampak sama-sama valid [1]. Dengan cara ini, sinyal keberhasilan dihilangkan sepenuhnya. Dengan demikian, ketiga pendekatan ini bukan pesaing, melainkan pertahanan berlapis yang saling melengkapi, PBKDF2/bcrypt mengatasi biaya komputasi penyerang, *salt* mengatasi *batch attacks*, dan HE menghilangkan sinyal keberhasilan pada serangan individual [8]. *Honey Encryption* paling cocok untuk skenario spesifik, yaitu perlindungan *password manager* atau sistem *credential storage* dengan ruang pesan terbatas dan terdefinisi [7].

## VII. Kesimpulan

Makalah ini telah mengimplementasikan dan mengevaluasi skema *Honey Encryption* berbasis DTE dan AES-128 untuk perlindungan enkripsi berbasis kata sandi. Dari lima skenario pengujian diperoleh tiga temuan utama, yaitu validitas pesan umpan mencapai 100% di seluruh dataset, tambahan beban komputasi DTE di bawah 0,003% dari total waktu eksekusi, dan tingkat kebocoran informasi turun dari 99% pada AES konvensional menjadi 0% pada *Honey Encryption*. Temuan ini mengonfirmasi bahwa HE berhasil menghilangkan sinyal keberhasilan yang selama ini menjadi kelemahan mendasar PBE, tanpa mengorbankan performa sistem secara berarti.

Meskipun demikian, efektivitas *Honey Encryption* sangat bergantung pada distribusi ruang pesan yang digunakan. Pada distribusi miring, entropi pesan umpan turun drastis hingga 2,19 bit dan satu pesan mendominasi 61,6% keluaran, yang berarti penyerang masih bisa melakukan tebakan terarah. Persoalan salah ketik kata sandi yang menghasilkan pesan umpan tanpa peringatan juga memerlukan penanganan tambahan, misalnya dengan *Multi-Factor Authentication* (MFA). Untuk pengembangan selanjutnya, disarankan penelitian tentang *typo-tolerant* HE, rancangan DTE untuk distribusi tidak seragam, dan integrasi HE dengan KDF modern sebagai sistem perlindungan berlapis yang lebih menyeluruh.

## Video Link at Youtube *(Heading 5)*

https://youtu.be/hFyU74Dlq6k


## Acknowledgment *(Heading 5)*

Penulis mengucapkan terima kasih kepada dosen pengampu mata kuliah II4021 Kriptografi atas penjelasan menyeluruh tentang algoritma AES-128 di perkuliahan, serta kepada penulis-penulis literatur acuan yang karyanya menjadi dasar analisis pada makalah ini.


## References


[1] A. Juels and T. Ristenpart, "Honey encryption: Encryption beyond the brute-force barrier," IEEE Security & Privacy, vol. 12, no. 4, pp. 59–62, 2014.

[2] W. Yin, J. Indulska, and H. Zhou, "Protecting private data by honey encryption," Security and Communication Networks, vol. 2017, Article ID 6760532, 2017, doi: 10.1155/2017/6760532.

[3] A. E. Omolara, A. Jantan, and O. I. Abiodun, "A comprehensive review of honey encryption scheme," Indonesian Journal of Electrical Engineering and Computer Science, vol. 13, no. 2, pp. 649–656, 2019, doi: 10.11591/ijeecs.v13.i2.pp649-656.

[4] T. Win and K. S. M. Moe, "Protecting private data using improved honey encryption and honeywords generation algorithm," Advances in Science, Technology and Engineering Systems Journal, vol. 3, no. 5, pp. 311–320, 2018, doi: 10.25046/aj030537.

[5] A. E. Omolara, A. Jantan, O. I. Abiodun, H. Arshad, K. V. Dada, and E. Emmanuel, "HoneyDetails: A prototype for ensuring patient's information privacy and thwarting electronic health record threats based on decoys," Health Informatics Journal, vol. 26, no. 3, pp. 2083–2104, 2020, doi: 10.1177/1460458219894479.

[6] A. A. M. Gharbi and A. S. Naser, "Honey encryption security techniques: A review paper," Al-Rafidain Journal of Computer Sciences and Mathematics, vol. 16, no. 1, pp. 21–32, 2022.

[7] A. E. Omolara and A. Jantan, "Modified honey encryption scheme for encoding natural language message," International Journal of Electrical and Computer Engineering (IJECE), vol. 9, no. 3, pp. 1871–1878, 2019, doi: 10.11591/ijece.v9i3.pp1871-1878.

[8] D. R. Purba et al., "Implementation of honey encryption to improve resilience against brute force attacks in cloud-based microservices communication," Journal of Artificial Intelligence and Software Engineering (JAISE), vol. 5, no. 1, 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, 19 Juni 2026

Samuel Chris Michael B.S
18223011