Pemasangan Ikan Haring Merha untuk Kriptografi pada File Sistem Penyimpanan

Rayhan Ridhar Rahman (13522160)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13522160@std.stei.itb.ac.id, [2]rayhanridhar@gmail.com

***Abstract*—Kriptografi secara konvensional menjamin kerahasiaan isi pesan, namun tidak menyembunyikan keberadaan pesan itu sendiri. Kriptografi yang terbaik adalah yang dapat menjaga kuncinya yang terbaik sebab sifat dari enkripsi yang perlu diketahui public tidak memungkinkan algoritmanya tersembunyi. Dalam situasi pemilik data berada di bawah paksaan (coercion) untuk menyerahkan kunci enkripsi, keamanan data menjadi rentan. Maka diperlukan untuk pembuatan suatu system agar ciphertext menjadi plaintext yang aman untuk diserahkan. Makalah ini akan membahas beberapa penggunaan dari ikan haring merah pada kriptografi dan contoh-contoh implementasinya. Utamanya adalah system seperti Deniable Encryption (Enkripsi yang Dapat Disangkal) menggunakan skema Hidden Volume. Tujuannya adalah menciptakan mekanisme penyimpanan dengan keberadaan data rahasia tidak dapat dibuktikan secara matematis sehingga memberikan pemilik data kemampuan plausible deniability. Sebuah Proof of Concept (PoC) dikembangkan menggunakan bahasa pemrograman Python dengan algoritma AES-256 dan manajemen offset berbasis hash. Hasil pengujian menunjukkan bahwa data terenkripsi berhasil disamarkan sebagai random noise di dalam wadah penyimpanan, namun efektivitasnya sangat bergantung pada mitigasi kebocoran sistem operasi dan karakteristik media penyimpanan fisik (SSD vs HDD).**

***Keywords*—Deniable Encryption, Hidden Volume, Anti-Forensics, AES, Plausible Deniability, Cybersecurity.**

I. PENDAHULUAN

Keamanan data di era digital ini menitik-beratkan dalam sistem penjagaan kunci yang benar. Pemanfaatan *hash* merupakan salah satu dari bukti tersebut. Namun kunci mungkin saja dengan mudah didapatkan dengan beberapa metode, terutama jika tidak terenkripsi. Penyerangan pada individu, pengawasan yang semakin mengganggu privasi, dan kewajiban dapat menyebabkan kebocoran kunci yang dimaklumkan.

Ikan haring merah adalah sebuah perkata mengenai bukti yang bersifat untuk mengecoh. Dengan memberi kunci yang palsu disertai tidak adanya bukti yang menandakan kepalsuannya dapat membuat kriptoanalisis yang dilakukan seseorang untuk berhenti. Dengan ini kunci dan hasilnya dapat menjadi suatu harapan palsu bagi yang memberhentikan upaya kriptoanalisis.

Memang telah banyak metode enkripsi dan dekripsi yang sudah lebih baik. Tetapi dengan konsep *Plausible Deniability*, pemilik data dapat menyangkal keberadaan data rahasia secara meyakinkan. Teknik utama untuk mencapai ini adalah *Deniable Encryption*. Laporan ini bertujuan untuk merancang purwarupa perangkat lunak sederhana untuk mendemonstrasikan prinsip kerja *Hidden Volume*, seperti yang digunakan dalam beberapa sistem penyimpanan data-data sensitif.

Tujuan dari makalah ini adalah melakukan eksplorasi pada sistem enkripsi yang menggunakan teknik bisa disangkal ini

II. KAJIAN PUSTAKA

a. Kriptografi

Kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan. Merupakan bidang ilmu dengan tujuan utama untuk menjawab pemasalahan kerahasiaan (confidentiality), autentikasi (authentication), integritas data (data integrity), dan nir-penyangkalan (non-repudiation) [1].

Kriptografi digunakan sejak zaman terdahulu, dengan teknik-teknik kriptografi untuk saat ini sangat sederhana. Tujuan utama perkembangan kriptogrfi pada masa tersebut adalah untuk kerahasiaan. Terutama dalam penyusunan strategi militer dan diplomasi. Salah satu contoh kriptografi paling sederhana dan terkenal adalah *caesar cypher* yang telah digunakan banyak media sebagai *brain teaser*.

Gambar II.1 Roda Caesar Cipher
(Sugoilab, 2016)

Kriptografi modern berkembang dengan kemajuan teknologi komputer terutama untuk kerahasiaan jejak digital. Kriptografi modern lebih bergantung pada kerahasiaan kunci, bukan algoritma yang digunakan. Kriptografi modern terbagi menjadi tiga kategori utama,

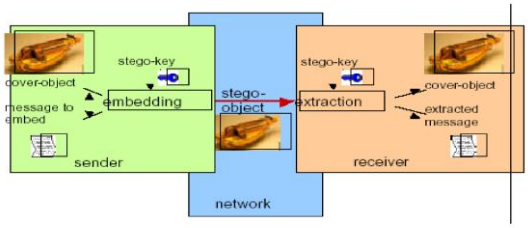
yaitu kriptografi simetris, kriptografi asimetris, dan fungsi hash. Kriptografi simetris menggunakan satu kunci untuk kepentingan enkripsi dan dekripsi. Kriptografi asimetris menggunakan pasangan kunci *public* dan kunci *privat*. Sedangkan fungsi hash mengkompresi pesan ukuran sembarang menjadi message-digest berukuran pasti.

Dengan adanya kriptografi, terdapat juga ilmu kriptoanalisis yang bertujuan untuk membongkar *cipher*. Ilmu ini sangat berguna untuk mengatasi beberapa hal seperti mendeteksi penyelundup, membongkar rahasia, dan pencarian data pada *disc* yang hilang. Namun hal ini pun bisa menjadi sumber dari kejahatan seperti mengungkapkan rahasia bisnis yang sangat merugikan.

b. Steganografi

Kata "steganografi" berasal dari bahasa Yunani, yaitu steganos yang berarti "tersembunyi" atau "tertutup" dan graphein yang berarti "menulis". Secara definisi, steganografi adalah seni dan ilmu komunikasi yang menyembunyikan keberadaan pesan. Sesuai dengan ikan haring merah, namun ciphertext memiliki wujud di mata manusia.

Tujuan utama steganografi adalah menyembunyikan data rahasia (payload) ke dalam media penyamar (cover) sehingga tidak menimbulkan kecurigaan. Media yang telah disisipi pesan disebut sebagai stego-object. Bergantung dengan ukuran penggunaannya, hasil akhir dari proses steganografi dapat memiliki fitur atau fungsi yang sama dengan .barang penyamar aslinya.



Gambar II.2 Alur Steganografi
(R. Munir, 2025)

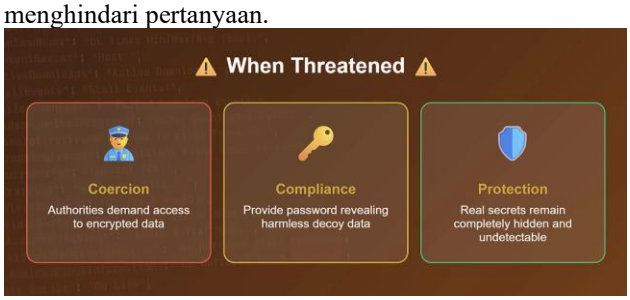
Steganografi bisa berupa media apa saja dengan syarat ukuran dari *cover* lebih (atau jauh lebih) besar daripada *stego object*. Kemudian untuk meningkatkan kompleksitas, dapat dilakukan dengan mengacak lokasi penyelundupan, ukuran bit setiap byte yang digunakan, dan pengacakan lokasi. Serupa dengan kriptoganalisis, stegoanalisis ada untuk mengungkan pesan asli dari *stego-object*.

c. Ikan Haring Merah

Red herring merupakan perkataan untuk klu yang memiliki maksud untuk mengelabui. Biasanya digunakan dalam pertunjukkan terutama dengan genre *mystery*. Memiliki tujuan utama untuk mengecoh penonton untuk terlalu fokus dengan hal tersebut sambil melupakan klu-klu penting.

Penggunaan istilah ini pertama kali tercatat pada tahun 1686. Istilah ini digunakan untuk menunjukkan gangguan yang memperpanjang suatu aktivitas, misalnya ikan asap yang digunakan untuk mengaburkan jejak yang diikuti anjing pemburu selama perburuan. Pada tahun 1884, istilah ini telah menjadi perangkat sastra.

Di luar konteks bercerita, "red herring fallacy" adalah istilah yang digunakan dalam politik dan debat yang memiliki arti hampir sama. Namun, alih-alih elemen naratif, istilah ini merujuk pada argumen. Kekeliruan dalam konteks ini menunjukkan argumen yang tidak logis yang dimaksudkan untuk mengaburkan daripada menjawab. Ketika digunakan dalam konteks wawancara atau debat, pada dasarnya ini adalah taktik untuk menghindari pertanyaan.



Gambar II.3 Kasus Mengancam
(Sam Bent, 2025)

Dalam kriptografi hal ini telah menjadi gagasan untuk teknik enkripsi. Suatu enkripsi yang menyembunyikan pesan sesungguhnya di belakang pesan palsu yang mencolok. Digunakan untuk beberapa hal selain menjaga kerahasiaan dari pesan asli. Bisa juga digunakan untuk upaya melepaskan diri dari paksaan, menciptakan berkas sementara, dan *secure messaging*.

d. Deniable Encryption

Konsep *Deniable Encryption* pertama kali digagas secara formal oleh Canetti et al. pada tahun 1997. Berbeda dengan enkripsi standar yang hanya menjamin kerahasiaan (*confidentiality*) isi pesan, *deniable encryption* bertujuan untuk menyembunyikan eksistensi pesan itu sendiri. Dalam model ancaman kriptografi standar, musuh diasumsikan hanya dapat menyadap jalur komunikasi (*ciphertext-only attack*). Namun, dalam skenario dunia nyata, musuh sering kali memiliki kekuatan koersif untuk memaksa pengirim atau penerima mengungkapkan kunci dekripsi, sebuah metode yang dikenal sebagai *Rubber-hose Cryptanalysis* [3]

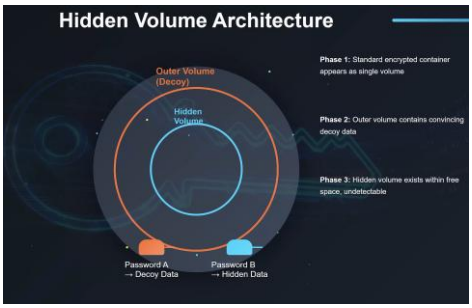
Secara teoritis, *deniable encryption* memungkinkan pengirim untuk menghasilkan *ciphertext* yang dapat didekripsi menjadi dua atau lebih *plaintext* yang berbeda, tergantung pada kunci yang digunakan. Jika dipaksa, pengguna dapat menyerahkan kunci palsu yang menghasilkan pesan "umpan" yang masuk akal (plausible decoy), tanpa penyerang dapat membuktikan secara matematis bahwa ada pesan lain yang tersembunyi [4]. Hal ini menciptakan *plausible deniability*, di mana penyangkalan pengguna terhadap keberadaan data rahasia tidak dapat dibantah dengan bukti teknis.

Dalam konteks kriptografi, *plausible deniability* memungkinkan seseorang untuk menyangkal kepemilikan data terenkripsi. Hal ini dicapai dengan membuat *ciphertext* (data terenkripsi) tidak dapat dibedakan dari data acak (*random noise*) yang biasanya mengisi ruang

kosong pada *hard drive*. Tentu untuk menyembunyikannya, perlu di perhitungkan beberapa indikator-indikator yang memberikan bukti cipher masih belum selesai dipecahkan.

e. Arsitektur Volume Tersembunyi

Metode paling praktis untuk menerapkan *deniable encryption* pada penyimpanan data adalah melalui skema *Hidden Volume*. Arsitektur ini memanfaatkan prinsip *steganographic file system*. Prinsip ini tersusun dari struktur bersarang (Nested structure), ciphertext indistinguishability, dan manajemen blok serta risiko korupsi data.



Gambar II.4 Arsitektuk volume tersembunyi (Sam Bent, 2025)

Sistem ini menggunakan dua lapisan kontainer. Lapisan pertama adalah Outer Volume yang berisi data non-sensitif. Lapisan kedua adalah Hidden Volume yang disisipkan di dalam area alokasi bebas (free space) dari Outer Volume. Kedua kontainer akan tampak menjadi sati arsitektur.

Keamanan skema ini bergantung pada properti bahwa data terenkripsi harus tampak tidak dapat dibedakan dari data acak seragam (*uniform random noise*). Saat inisialisasi, seluruh *Outer Volume* diisi dengan data acak.

$$E(K, M) \approx Random(U) \quad (1)$$

E adalah fungsi enkripsi, K adalah kunci, dan M adalah pesan. Karena Hidden Volume yang terenkripsi juga terlihat sebagai data acak, penyerang yang menganalisis free space pada Outer Volume tidak dapat membedakan antara ruang kosong yang diisi noise atau ruang yang berisi Hidden Volume [5]. Arsitektur modern sekarang pun masih kesulitan membedakan *disk space* yang tidak digunakan dengan *hidden data*.

Tantangan utama dalam arsitektur ini adalah risiko tertimpanya. Karena sistem operasi "luar" tidak mendeteksi *Hidden Volume* sehingga penulisan file baru ke *Outer Volume* dapat secara tidak sengaja menimpa sektor tempat *Hidden Volume* berada. Software modern seperti VeraCrypt mengatasi ini dengan mekanisme proteksi *header* khusus, namun pengguna disarankan untuk tidak menulis data ke *Outer Volume* saat *Hidden Volume* sedang aktif untuk menjaga integritas data [6].

III. IMPLEMENTASI

Implementasi yang dilakukan adalah penggunaan infrastruktur volume tersembunyi sederhana. Dengan membuat suatu file binary yang memiliki ukuran 10 MB. Command line akan digunakan untuk navigasi opsi dan tidak secara langsung membuka file.

a. Spesifikasi Program

Simulasi dilakukan menggunakan Python dengan spesifikasi:

- **Algoritma:** AES (Advanced Encryption Standard) mode CBC.
- **Hashing:** SHA-256 untuk derivasi *key* dan penentuan lokasi penyimpanan (*offset*).
- **Library:** cryptography.

b. Logika Program

Program tidak akan membuat suatu sistem file baru tetapi semua data akan ada di dalam satu file binary berukuran 10 MB. File ini akan diinisialisasikan dengan binary random untuk menghasilkan entropi yang cukup tinggi. File memiliki kemungkinan sangat besar untuk menjadi file tanpa format yang benar dan menghasilkan *gibberish* jika dibuka. Hal tersebutlah yang akan menjadi suatu *cover* untuk data sesungguhnya.

```
def create_container():
    if os.path.exists(CONTAINER_NAME):
        print(f"[!] File {CONTAINER_NAME} already
exist. Please don't overwrite your progress.")
        return

    print(f"[*] Membuat wadah {CONTAINER_NAME}
ukuran {CONTAINER_SIZE/1024/1024} MB...")
    with open(CONTAINER_NAME, "wb") as f:
        f.write(os.urandom(CONTAINER_SIZE))
```

Pengguna dapat memasukkan file ke dalam binary tersebut. Tetapi dalam implementasi, tidak akan seperti infrasturktur aslinya. Program tidak membedakan antara lokasi *hidden volume* dan *outer volume*. Maka pesan yang disembunyikan memiliki peluang yang sama untuk penempatannya yang membuatnya sangat mungkin saling menimpa. Maka pengguna disarankan untuk menyembunyikan pesan asli diakhir agar tidak tertimpa sama sekali.

Penyembunyian pesan dilakukan dengan menentukan path ke file dan memasukkan password sebagai *seed* dari algoritma hash. Hash yang dilakukan menggunakan SHA-256. 32-byte hasil hash menjadi kunci AES, sementara nilai integer dari hash tersebut akan menjadi *Offset* untuk tempat penyembunyian. Tidak akan dilakukan pengacakan lokasi bytes selanjutnya. Untuk menjaga integritas data, 4 bytes pertama digunakan untuk penunjuk ukuran dari ciphertext. Data diproses menggunakan AES-256 mode CBC (*Cipher Block Chaining*) dengan PKCS7 padding.

```
def store_file(filepath, password):
    if not os.path.exists(CONTAINER_NAME):
        print("[!] Container not found, please create one!")
        return

    with open(filepath, "rb") as f:
        plaintext = f.read()

    key, iv, offset = derive_key_and_offset(password)
```

```
    data_length = len(plaintext).to_bytes(4, 'big')

    padder = padding.PKCS7(128).padder()
    padded_data = padder.update(data_length + plaintext)
+ padder.finalize()

    cipher = Cipher(algorithms.AES(key), modes.CBC(iv),
backend=default_backend())
    encryptor = cipher.encryptor()
        ciphertext = encryptor.update(padded_data) +
encryptor.finalize()

    cipher_len_bytes = len(ciphertext).to_bytes(4, 'big')

    if offset + 4 + len(ciphertext) > CONTAINER_SIZE:
            print("[!] Error: File is overflowing with this
offset.")
        return

    with open(CONTAINER_NAME, "r+b") as f:
        f.seek(offset)
        f.write(cipher_len_bytes)
        f.write(ciphertext)
```

Sistem enkripsi bersifat simetris sehingga serupa dengan penyembunyian, memerlukan fungsi **derive_key_and_offset**(). Dengan offset digunakan untuk lokasi ciphertext yang diduga. Kemudian byte headers dibaca dan jika masuk akal proses dekripsi akan dilanjutkan. Jika password salah, offset akan menunjuk ke lokasi acak. Saat didekripsi, data acak tersebut akan menghasilkan format *padding* yang tidak valid (PKCS7 error).

```
def retrieve_file(output_path, password):
    if not os.path.exists(CONTAINER_NAME):
        print("[!] Container not found, please create one!")
        return

    key, iv, offset = derive_key_and_offset(password)

    try:
        with open(CONTAINER_NAME, "rb") as f:
            f.seek(offset)

            size_bytes = f.read(4)
                if not size_bytes: raise ValueError("Offset is
empty or EOF")

            cipher_len = int.from_bytes(size_bytes, 'big')

            if cipher_len > CONTAINER_SIZE or cipher_len
<= 0:
                raise ValueError("Invalid data length")

            encrypted_chunk = f.read(cipher_len)

                    cipher = Cipher(algorithms.AES(key),
modes.CBC(iv), backend=default_backend())
        decryptor = cipher.decryptor()
                                    decrypted_padded =
decryptor.update(encrypted_chunk) +
decryptor.finalize()

        unpadder = padding.PKCS7(128).unpadder()
            raw_data = unpadder.update(decrypted_padded) +
unpadder.finalize()

        file_len = int.from_bytes(raw_data[:4], 'big')
        original_content = raw_data[4:4+file_len]

        with open(output_path, "wb") as f:
            f.write(original_content)

            print(f"[+] Successfully Decrypted, saving to:
'{output_path}'")

    except Exception as e:
        print(f"[-] Failed to Decrypt (Error: {e})")
```

IV. PENGUJIAN DAN ANALISIS

a. Skenario Uji

1. Memasukkan file yang terlalu besar untuk offset. Memastikan program yang dibuat hanya dapat menulis pada file, *ciphertext* yang bisa ditampung.
2. Tabrakan antara data *outer volume* dengan *hidden volume*. Menganalisis risiko ketika dua *password* berbeda menghasilkan area penyimpanan yang tumpang tindih yang merupakan risiko dari arsitektur *hidden volume*.
3. Skenario normal dengan memasukkan kunci yang palsu dan kunci yang benar. Memastikan file memang benar tersimpan pada file storage.

b. Hasil Skenario

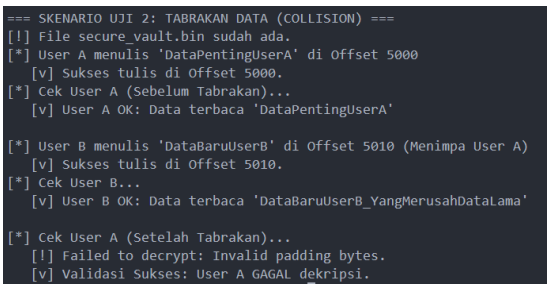
Skenario uji 1 dilakukan dengan memasukkan suatu *array of characters* dengan panjang 100 bytes dengan offset diatur pada 10 bytes terakhir pada penyimpanan. Hasilnya adalah sebagai berikut.

```
=== SKENARIO UJI 1: FILE TERLALU BESAR (OVERFLOW) ===
[!] File secure_vault.bin sudah ada.
[*] Mencoba menulis 100 bytes di sisa ruang 10 bytes...
[*] Offset: 10485750, Ukuran Container: 10485760
   [RESULT] GAGAL TULIS (Expected). Data melebihi batas container.
```

Gambar IV-1 Pengujian Skenario 1

Seperti yang diduga, data input tidak bisa disimpan pada file penyimpanan. Hal ini menunjukkan salah satu kelemahan dari sistem sederhana ini. Dengan password tertentu dapat menghasilkan *offset* yang tidak memperbolehkan penyimpanan file yang diinginkan. Walaupun data yang akan diisi berukuran jauh lebih kecil dari ukuran penyimpanan, sistem tidak memperbolehkannya. Masalah ini mungkin bisa diselesaikian dengan pengacakan *bytes* penyembunyian data. Namun metode tersebut dapat menimbulkan penimpaan file yang tidak diketahui, dibanding rancangan yang terukur ini.

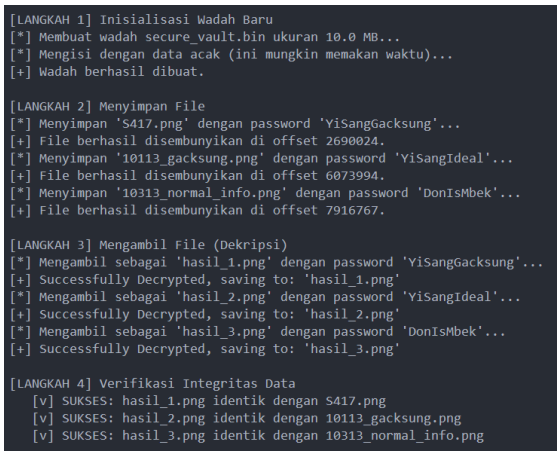
Skenario uji 2 dilakukan dengan persona Alice (A) dan Bob (B). Mula-mula Alice menyimpan data rahasianya, “DataPentingUserA” pada *offset* 5000. Kemudian Alice mendekripsi data tersebut.



Gambar IV-2 Pengujian Skenario 2

Dari hasil skenario bisa ditraik kesimpulan bahwa tanpat memisahkan lokasi *hidden volume* dan *outer volume*, data akan lebih rentan untuk ditimpa. Alice kehilangan data pentingnya karena ditimpa oleh data dari Bob. Pada *hidden volume* sesungguhnya, seharusnya bisa dibedakan dengan *random noise* di lapisan luar.

Skenario uji 3 adalah tes normal dengan 3 file yang disimpan pada tempatnya dengan sesuai. Langkah pertama adalah melakukan penyimpanan rahasia dari ketiga file tersebut. Kemudian dilanjut dengan melakukan dekripsi dengan password masing-masing. Hasil dari dekripsi pun dibandingkan dengan file aslinya yang disamarkan.



Gambar IV-3 Pengujian Skenario Normal

Hasil sesuai dengan skenario yang optimal dengan masing-masing file menempati lokasinya sendiri. Kombinasi dari password dan ukuran file perlu diperhitungkan agar tidak terjadi *overflow* dan penimpaan. Dalam skenario normal, password dan file telah diatur sedemikian rupa agar tidak menghasilkan kesalahan.

V. KESIMPULAN DAN SARAN

Deniable Encryption adalah teknik pertahanan terakhir untuk menjaga privasi data di lingkungan yang bermusuhan (*hostile environment*). Implementasi kode dibuat berhasil membuktikan konsep bahwa data terenkripsi dapat disembunyikan di dalam noise tanpa meninggalkan metadata yang terlihat.

Namun, keamanan teknik ini bergantung berat pada disiplin pengguna dan tidak banyak untuk algoritma. Tanpa strategi umpan yang meyakinkan dan pemilihan hardware yang tepat, penyangkalan keberadaan data (*deniability*) akan gagal di hadapan analisis forensik mendalam. Maka perlu diperhintungkan ukuran dari sistem file penyimpanan tersebut terhadap ukuran berkas dalam dan luar.

Untuk meningkatkan keamanan, keandalan, dan fungsionalitas sistem dapat dikembangkan kembali dalam berbagai hasl seperti berikut:

1. Saat ini, sistem menggunakan *offset* acak yang berisiko menimpa data yang sudah ada jika dua password menghasilkan *hash* yang berdekatan. Disarankan untuk mengimplementasikan peta alokasi (*bitmap allocation*) yang terenkripsi atau mekanisme pengecekan "ruang kosong" sebelum penulisan, tanpa membocorkan peta tersebut kepada pihak luar.
2. Dalam implementasi saat ini, 4-byte penanda panjang data ditulis secara linear sebelum *ciphertext*. Analis forensik tingkat lanjut mungkin dapat mencari pola integer 4-byte ini. Pengembangan selanjutnya sebaiknya mengenkripsi header panjang ini secara terpisah atau file memiliki ukuran blok yang tetap.
3. Dapat dikembangkan lagi untuk *secure wiping* dengan cara yang cukup mudah. Fitur ini berfungsi untuk memastikan tidak ada sisa magnetik dari data sebelumnya.

UCAPAN TERIMA KASIH

Puji dan syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa, karena atas rahmat dan karunia-Nya, makalah ini dapat diselesaikan dengan baik. Makalah ini disusun untuk memenuhi salah satu tugas mata kuliah IF4020 Kriptografi.

Pada kesempatan ini, penulis ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada Bapak Dr. Ir. Rinaldi Munir, MT, selaku dosen pengampu mata kuliah IF4020 Kriptografi yang telah memberikan ilmu, arahan, dan bimbingan selama perkuliahan berlangsung.

Penulis menyadari bahwa makalah ini masih jauh dari kesempurnaan. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan demi penyempurnaan karya tulis di masa mendatang.

REFERENSI

[1] R. Munir, “Pengantar kriptografi,” 2025, slide Kuliah, Program Studi Teknik Informatika STEI ITB.

[2] ——, “Steganografi Bagian 1,” 2025, slide Kuliah, Program Studi Teknik Informatika STEI ITB.

[3] Canetti, R., C. Dwork, M. Naor, and R. Ostrovsky (1997). “Deniable encryption.” *Advances in Cryptology—CRYPTO'97, Proceedings* Santa Barbara, CA, USA, August 17–21 (Lecture Notes in Computer Science vol. 1294), ed. B.S. Kaliski, Springer-Verlag, Berlin, 90–104.

[4] A. Juels and T. Ristenpart, “Honey Encryption: Encryption beyond the Brute-Force Barrier,” *IEEE Security & Privacy*, vol. 12, no. 4, pp. 59–62, Jul. 2014, doi: https://doi.org/10.1109/msp.2014.67.

[5] M. Kedziora, Y.-W. Chow, and W. Susilo, “Defeating Plausible Deniability of VeraCrypt Hidden Operating Systems,” *Communications in computer and information science*, pp. 3–13, Jan. 2017, doi: https://doi.org/10.1007/978-981-10-5421-1_1.

[6] VeraCrypt, “VeraCrypt - Free Open source disk encryption with strong security for the Paranoid,” *Veracrypt.io*, 2025. https://veracrypt.io/en/Documentation.html

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

Rayhan Ridhar Rahman dan 13522160