Implementasi dan Evaluasi Peningkatan Kapasitas Steganografi Teks Berbasis Unicode Whitespace pada Dokumen Digital

Daffari Adiyatma - 18222003
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: daffariadytm2507@gmail.com , 18222003@std.stei.itb.ac.id

***Abstract*—Keamanan informasi dalam komunikasi digital semakin menjadi perhatian utama seiring meningkatnya volume pertukaran dokumen secara elektronik. Steganografi teks merupakan salah satu teknik penyembunyian informasi yang bekerja dengan cara menyisipkan pesan rahasia ke dalam teks biasa tanpa mengubah makna maupun tampilan visualnya. Metode Innamark yang diusulkan oleh Hellmeier et al. (2025) menggunakan empat karakter spasi Unicode sebagai alfabet penggantian, menghasilkan kapasitas penyisipan sebesar 2 bit per spasi. Makalah ini mengusulkan perluasan alfabet penggantian menjadi 8 dan 16 karakter spasi Unicode yang valid secara visual, guna meningkatkan kapasitas penyisipan menjadi 3 hingga 4 bit per spasi. Implementasi dilakukan menggunakan bahasa pemrograman Python dengan mengadopsi kerangka Design Science Research Methodology (DSRM). Evaluasi eksperimental dilakukan pada tiga ukuran dokumen teks dengan total 9 kombinasi pengujian, seluruhnya lulus uji round-trip. Hasil menunjukkan peningkatan kapasitas sebesar 50% dan 100% dibandingkan baseline, dengan overhead ukuran file yang identik antar varian alfabet (~25–26%) — sebuah sifat overhead-free capacity increase yang merupakan kontribusi utama penelitian ini. Pengujian ketahanan mengungkap perilaku all-or-nothing dengan imunitas penuh terhadap serangan double-space autocorrect namun kerentanan total terhadap normalisasi Unicode, yang merupakan keterbatasan inheren metode berbasis substitusi karakter.**

Keywords—steganografi teks; Unicode; whitespace; kapasitas embedding; dokumen digital; kriptografi

I. PENDAHULUAN

Keamanan dan privasi informasi merupakan aspek kritis dalam komunikasi digital modern. Dua pendekatan utama yang lazim digunakan adalah kriptografi dan steganografi. Kriptografi mengubah isi pesan menjadi bentuk yang tidak dapat dibaca tanpa kunci tertentu [1], sedangkan steganografi menyembunyikan keberadaan pesan itu sendiri dengan cara menyisipkannya ke dalam medium cover yang tampak wajar sehingga pihak ketiga tidak menyadari adanya komunikasi rahasia [2]. Teknik steganografi teks secara umum diklasifikasikan menjadi tiga kategori: metode berbasis format (format-based), metode berbasis linguistik (linguistic-based), dan metode berbasis karakter (character-based) [4]. Masing-masing kategori memiliki keunggulan dan keterbatasan tersendiri yang dibahas secara lebih mendalam pada Bagian II.

Salah satu pendekatan yang menjanjikan dalam steganografi teks adalah pemanfaatan karakter spasi Unicode. Standar Unicode versi 16.0.0 mendefinisikan 17 karakter spasi dengan lebar yang bervariasi [6], yang secara visual hampir tidak dapat dibedakan dari spasi konvensional (U+0020) oleh pengamat manusia namun berbeda pada level mesin. Berbagai penelitian telah mengeksplorasi pendekatan ini, mulai dari WhiteSteg [12], UniSpaCh [7], AITSteg [8], hingga CovertSYS [13]. Metode paling komprehensif saat ini adalah Innamark [5] yang diusulkan oleh Hellmeier et al. (2025), menggunakan empat karakter spasi Unicode terpilih dan terbukti memiliki robustness terbaik dibandingkan sembilan metode pembanding pada dataset satu juta artikel Wikipedia.

Meskipun Innamark terbukti efektif, paper aslinya secara eksplisit menyebutkan kapasitas embedding sebagai keterbatasan utama yang perlu diatasi dalam penelitian lanjutan [5]. Dengan alfabet sebesar empat karakter, setiap spasi hanya mampu membawa 2 bit informasi, menghasilkan payload yang relatif kecil khususnya pada dokumen singkat. Peningkatan kapasitas dapat dicapai dengan memperluas alfabet penggantian, yakni menambahkan karakter spasi Unicode lain yang memenuhi kriteria serupa.

Makalah ini mengusulkan dan mengevaluasi perluasan alfabet penggantian Unicode pada metode steganografi teks berbasis whitespace. Secara spesifik, kontribusi makalah ini adalah: (1) implementasi varian perluasan alfabet menjadi 8 dan 16 karakter spasi Unicode yang dipilih berdasarkan kriteria visual dan robustness, (2) evaluasi eksperimental terhadap peningkatan kapasitas penyisipan dibandingkan metode baseline, dan (3) analisis ketahanan terhadap serangan normalisasi karakter pada berbagai skenario transformasi dokumen digital. Implementasi dilakukan menggunakan bahasa pemrograman Python dengan kerangka Design Science Research Methodology (DSRM). Sisa makalah ini disusun sebagai berikut: Bagian II memaparkan tinjauan pustaka, Bagian III menjelaskan metodologi, Bagian IV menyajikan

hasil eksperimen, Bagian V membahas analisis dan diskusi, dan Bagian VI berisi kesimpulan.

II. TINJAUAN PUSTAKA

A. Konsep Dasar Steganografi Teks

Steganografi merupakan ilmu dan seni menyembunyikan keberadaan sebuah pesan di dalam medium cover sedemikian rupa sehingga pihak ketiga tidak menyadari adanya komunikasi rahasia yang sedang berlangsung [2]. Berbeda dari kriptografi yang mengacak isi pesan sehingga tidak dapat dibaca, steganografi berfokus pada penyembunyian eksistensi pesan itu sendiri. Kedua teknik ini bersifat komplementer: steganografi dapat dikombinasikan dengan kriptografi untuk menghasilkan sistem keamanan informasi berlapis dengan melakukan enkripsi isi pesan sebelum disisipkan ke dalam medium cover [1].

Dalam taksonomi yang dikemukakan oleh Petitcolas et al. [2], information hiding sebagai bidang ilmu mencakup steganografi, watermarking, dan fingerprinting. Watermarking berfokus pada penyisipan tanda kepemilikan yang bersifat persisten dan tahan terhadap modifikasi, sedangkan steganografi mengutamakan imperceptibility atau ketidakterlihatan oleh pengamat manusia. Rizzo et al. [10] merangkum perbedaan konseptual ini secara singkat: kriptografi membuat informasi tidak dapat dibaca, steganografi menyembunyikan informasi dalam medium yang dapat dibaca, dan watermarking memastikan keaslian dan hak cipta konten digital.

Teknik steganografi teks secara umum dapat diklasifikasikan ke dalam tiga kategori utama berdasarkan cara penyisipan informasinya [4]. Pertama, metode berbasis format (format-based) menyembunyikan informasi melalui modifikasi atribut visual teks seperti pergeseran baris, variasi jarak antarkata, atau perubahan ukuran dan warna huruf. Metode ini tidak dapat diterapkan pada dokumen teks polos dan mudah terdeteksi saat terjadi konversi format. Kedua, metode berbasis linguistik (linguistic-based) menggantikan kata dengan sinonim atau menghasilkan teks baru yang mengandung pesan tersembunyi. Meskipun dapat diterapkan pada teks polos, perubahan semantik yang terjadi berpotensi mengubah makna teks asli, khususnya dalam konteks dokumen hukum atau teknis [5]. Ketiga, metode berbasis karakter (character-based) memanfaatkan substitusi karakter individual, termasuk penggunaan karakter Unicode yang secara visual identik namun berbeda pada level kode.

B. Metode Steganografi Berbasis Whitespace Sebelum Innamark

TABLE I. PERBANDINGAN METODE STEGANOGRAFI TEKS BERBASIS WHITESPACE (WHITESTEG S.D. METODE USULAN)

Metode	Tahun	Carrier	BPS	Robustness	Kelemahan Utama
Whitesteg	2008	Spasi ganda antar kata	1	Rendah	DASH attack
UniSpaCh	2012	4 karakter unicode	1-2	Sedang	DASH attack
AITSteg	2019	Zero-width space	2	Rendah	Filter Platform
CovertSYS	2022	Zero-width space	2	Sedang	Filter Platform
Innamark	2025	4 Unicode space (A+)	2	Tinggi	Kapasitas Terbatas
Usulan	**2026**	**8/16 Unicode space**	**3-4**	**TBD**	**-**

Pemanfaatan karakter spasi sebagai carrier telah menjadi fokus penelitian sejak awal era komputasi digital, dimulai dari SNOW yang menyisipkan tab dan spasi di akhir baris namun mudah terdeteksi karena anomali spasi berlebih [5]. Por et al. [7] mengusulkan WhiteSteg (2008) yang menggantikan satu spasi dengan satu atau dua spasi, tetapi kapasitasnya sangat terbatas (~13 bit per baris kosong). Pengembangannya, UniSpaCh [7], memanfaatkan karakter spasi Unicode secara langsung dan mencapai kapasitas hingga 80% ruang spasi, namun rentan terhadap serangan DASH. Ahvanooey et al. [8] kemudian mengusulkan AITSteg (2019) berbasis zero-width space (U+200B) dengan kapasitas 2 bit per posisi; namun karena ZWSP dikategorikan sebagai format control character—bukan space character [6]—metode ini rentan terhadap normalisasi dan pemfilteran aktif oleh platform seperti Twitter dan WhatsApp [5]. Ringkasan perbandingan disajikan pada Tabel I.

Secara umum, metode-metode sebelum Innamark menghadapi dua keterbatasan yang saling berkaitan: kapasitas rendah (1–2 bit per spasi) akibat terbatasnya alfabet penggantian, dan ketahanan lintas platform yang tidak konsisten karena karakter khusus sering dinormalisasi saat copy-paste [5].

C. Metode Innamark

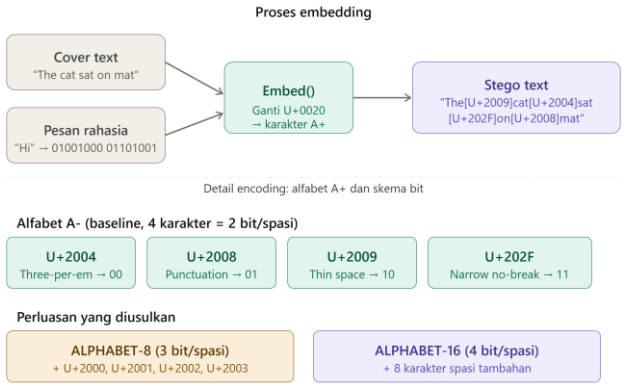
Innamark merupakan metode information hiding berbasis penggantian karakter spasi Unicode yang diusulkan oleh Hellmeier et al. [5] dari Fraunhofer ISST, Jerman. Metode ini merupakan pengembangan dari pendekatan awal yang dipresentasikan pada HICSS 2025 [9] dan dipublikasikan sebagai preprint pada Februari 2025. Innamark bekerja dengan menggantikan setiap karakter spasi konvensional (U+0020) dalam teks cover dengan salah satu karakter spasi Unicode dari alfabet yang telah dikurasi, yang disebut A+ (A-plus). Pemilihan karakter pengganti ditentukan oleh bit-bit pesan rahasia yang ingin disisipkan.

Proses seleksi alfabet A+ dalam paper Innamark dilakukan secara sistematis melalui evaluasi semua 17 karakter spasi yang didefinisikan oleh standar Unicode v16.0.0 [6]. Evaluasi dilakukan berdasarkan dua kriteria utama: (1) non-noticeability, yakni ketidakterlihatan perbedaan visual antara

karakter pengganti dengan spasi konvensional U+0020 oleh pengamat manusia, dan (2) robustness lintas platform, yakni kemampuan karakter untuk bertahan saat teks dipindahkan antara berbagai format file (.txt, .docx, .pdf) dan aplikasi (email, Microsoft Teams). Dari evaluasi ini, diperoleh lima karakter yang memenuhi kedua kriteria: U+2004 (three-per-em space), U+2008 (punctuation space), U+2009 (thin space), U+202F (narrow no-break space), dan U+205F (medium mathematical space) [5].

Innamark menggunakan empat dari lima karakter di atas sebagai alfabet encoding A-, sementara satu karakter (U+202F) difungsikan sebagai separator phi untuk menandai batas-batas struktur pesan. Dengan alfabet A- berukuran empat karakter, setiap spasi dalam teks cover dapat mengkodekan 2 bit informasi ($\log_2(4) = 2$). Innamark juga mendukung fungsionalitas opsional seperti kompresi payload, enkripsi, hashing, dan error-correcting codes melalui struktur pesan yang disebut InnamarkTags. Evaluasi pada dataset 1.000.000 artikel Wikipedia menunjukkan bahwa Innamark memiliki performa robustness terbaik dibandingkan sembilan algoritma pembanding, termasuk UniSpaCh, AITSteg, CovertSYS, dan StegCloak [5].

Fig. 1. Mekanisme embedding steganografi berbasis Unicode whitespace (diadaptasi dari [5])



Paper Innamark secara eksplisit menyebutkan kapasitas penyisipan sebagai keterbatasan yang perlu diatasi dalam penelitian lanjutan. Kapasitas sebesar 2 bit per spasi menghasilkan payload yang relatif kecil pada dokumen pendek, dan paper tersebut menyarankan eksplorasi terhadap penambahan karakter alfabet sebagai salah satu cara untuk meningkatkan kapasitas [5]. Inilah yang menjadi titik berangkat penelitian dalam makalah ini.

TABLE II. KARAKTER SPASI UNICODE TERPILIH ALFABET A+ (DIADAPTASI DARI [5])

Kode Unicode	Nama	Lebar (perkiraan)	Peran di Innamark
U+2004	Three-per-em space	1/3 em	A- (encoding)
U+2008	Punctuation space	Lebar titik	A- (encoding)
U+2009	Thin space	1/5 em	A- (encoding)
U+202F	Narrow no-break space	~1/4 em	phi (separator)
U+205F	Medium math space	4/18 em	A- (encoding)

D. Standar Unicode Whitespace

Unicode adalah standar pengkodean karakter universal yang dikembangkan dan dikelola oleh Unicode Consortium. Standar Unicode versi 16.0.0, yang dirilis pada September 2024, mendefinisikan sebanyak 154.998 karakter dari berbagai sistem penulisan di seluruh dunia [6]. Di antara karakter-karakter tersebut, terdapat 17 karakter yang berfungsi sebagai karakter spasi dengan lebar positif (positive-width space characters), yang dideskripsikan dalam kategori Unicode sebagai Zs (Space Separator).

Karakter-karakter spasi Unicode ini bervariasi dalam hal lebar tampilan, mulai dari em space (U+2003) yang lebarnya setara dengan satu em (lebar huruf kapital M), hingga hair space (U+200A) yang sangat tipis. Spasi konvensional U+0020 memiliki lebar sekitar 1/4 em, sehingga karakter-karakter yang lebarnya mendekati rentang 1/3 hingga 1/5 em secara visual tidak dapat dibedakan darinya oleh manusia dalam konteks teks normal [5]. Penting untuk dicatat bahwa zero-width space (U+200B), meskipun disebut space dalam namanya, secara teknis tidak termasuk dalam kategori Zs dan bukan merupakan karakter spasi berwidth positif, sehingga tidak termasuk dalam alfabet yang relevan untuk metode berbasis substitusi spasi visual [6].

Dari 17 karakter spasi berwidth positif yang tersedia, tidak semua memenuhi kriteria untuk digunakan dalam steganografi berbasis substitusi. Karakter dengan lebar yang sangat berbeda dari U+0020, seperti em space (U+2003) atau ideographic space (U+3000), akan menimbulkan anomali visual yang dapat dideteksi oleh pengamat manusia. Selain itu, robustness karakter terhadap normalisasi oleh aplikasi bervariasi secara signifikan; beberapa karakter seperti U+00A0 (no-break space) dikonversi secara otomatis oleh aplikasi pengolah kata tertentu, sementara yang lain dipertahankan [5]. Pemilihan subset karakter yang tepat berdasarkan kedua kriteria ini menjadi faktor kunci dalam merancang metode steganografi whitespace yang efektif.

E. Metrik Evaluasi Steganografi Teks

Evaluasi kualitas suatu metode steganografi secara umum didasarkan pada tiga dimensi utama yang saling berkaitan [11]. Pertama, kapasitas embedding (embedding capacity) mengukur seberapa banyak informasi rahasia yang dapat disisipkan ke dalam medium cover. Dalam konteks steganografi teks berbasis substitusi karakter, kapasitas lazim diukur dalam satuan bits per character (BPC) atau bits per space (BPS), yakni jumlah bit pesan yang dapat dikodekan per karakter spasi dalam teks cover. Kapasitas total dalam bit diperoleh dengan mengalikan BPS dengan jumlah spasi yang tersedia dalam dokumen.

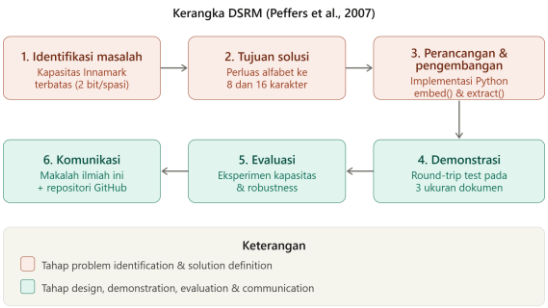
Kedua, imperceptibility atau ketidakterlihatan mengukur sejauh mana keberadaan pesan tersembunyi tidak dapat dideteksi oleh pengamat manusia maupun oleh perangkat analisis otomatis. Dalam steganografi teks berbasis Unicode whitespace, imperceptibility terhadap manusia dievaluasi melalui inspeksi visual terhadap dokumen stego untuk memastikan tidak ada anomali spasi yang terlihat. Imperceptibility terhadap perangkat otomatis diukur melalui ketahanan terhadap serangan steganalisis, yakni kemampuan metode untuk menghindari deteksi oleh perangkat yang secara aktif mencari keberadaan informasi tersembunyi [2].

Ketiga, robustness mengukur ketahanan pesan tersembunyi terhadap berbagai jenis modifikasi atau serangan yang dilakukan pada dokumen stego. Dalam konteks steganografi teks digital, serangan yang paling relevan adalah serangan normalisasi (normalization attack) yang dilakukan dengan cara menggantikan kembali semua karakter spasi Unicode dengan spasi konvensional U+0020. Metrik yang digunakan untuk mengukur robustness adalah survival rate, yakni persentase bit payload yang berhasil dipulihkan setelah dokumen stego melewati suatu transformasi atau serangan [5]. Dalam makalah ini, ketiga metrik tersebut digunakan secara komprehensif untuk mengevaluasi metode yang diusulkan.

III. METODOLOGI

Penelitian ini mengadopsi Design Science Research Methodology (DSRM) yang diusulkan oleh Peffers et al. [18] sebagai kerangka metodologi utama. DSRM merupakan pendekatan penelitian berorientasi artefak yang secara sistematis mencakup enam tahap: (1) identifikasi masalah dan motivasi, (2) definisi tujuan solusi, (3) perancangan dan pengembangan, (4) demonstrasi, (5) evaluasi, dan (6) komunikasi [18]. Pendekatan ini dipilih karena sesuai dengan sifat penelitian ini yang bertujuan merancang, mengimplementasikan, dan mengevaluasi sebuah artefak perangkat lunak berupa sistem steganografi teks dengan kapasitas yang ditingkatkan. Gambar 1 mengilustrasikan pemetaan keenam tahap DSRM ke dalam konteks penelitian ini.

Fig. 2. Pemetaan enam langkah DSRM (Peffers et al. [18]) ke konteks penelitian ini



A. *Identifikasi Masalah dan Motivasi*

Metode Innamark [5] terbukti unggul dalam imperceptibility dan robustness lintas platform, namun paper aslinya secara eksplisit mengidentifikasi kapasitas embedding sebesar 2 bit per spasi sebagai keterbatasan utama yang perlu diatasi. Penelitian sebelumnya mendukung perluasan alfabet carrier sebagai arah yang menjanjikan: Shazzad-Ur-Rahman et al. [15] menunjukkan bahwa penggunaan lebih banyak karakter Unicode per posisi spasi meningkatkan kapasitas secara signifikan, sementara Ahvanooey et al. [14] mengonfirmasi kapasitas rendah sebagai tantangan bersama metode berbasis substitusi karakter. Penelitian ini termotivasi untuk menguji apakah perluasan alfabet Innamark dapat meningkatkan kapasitas secara berarti tanpa mengorbankan imperceptibility dan robustness.

B. *Definisi Tujuan Solusi*

Berdasarkan identifikasi masalah di atas, tujuan solusi yang hendak dicapai dalam penelitian ini dirumuskan sebagai berikut. Pertama, merancang varian perluasan alfabet penggantian Unicode dari 4 karakter menjadi 8 dan 16 karakter spasi Unicode yang memenuhi kriteria non-noticeability dan robustness yang telah ditetapkan oleh Hellmeier et al. [5]. Kedua, mengimplementasikan algoritma embedding dan extraction untuk ketiga varian alfabet tersebut dalam bahasa pemrograman Python. Ketiga, mengevaluasi peningkatan kapasitas embedding (dalam satuan bits per space) yang dihasilkan oleh masing-masing varian alfabet dibandingkan dengan baseline Innamark. Keempat, mengevaluasi ketahanan (robustness) masing-masing varian terhadap serangan normalisasi karakter pada berbagai skenario transformasi dokumen digital.

C. *Perancangan dan Pengembangan*

Tahap perancangan dimulai dengan seleksi karakter spasi Unicode yang akan digunakan sebagai perluasan alfabet. Mengacu pada tabel evaluasi karakter Unicode yang telah disusun oleh Hellmeier et al. [5], dari 17 karakter spasi berwidth positif yang didefinisikan dalam Unicode v16.0.0 [6], dipilih subset yang memenuhi dua kriteria: (1) lebar karakter mendekati lebar spasi konvensional U+0020 (sekitar 1/3 hingga 1/5 em) sehingga tidak menimbulkan anomali visual, dan (2) karakter bertahan saat teks dipindahkan antar format file dan aplikasi umum. Berdasarkan kriteria ini, dirancang tiga varian alfabet sebagai berikut.

Varian pertama, ALPHABET-4, merupakan alfabet baseline Innamark yang terdiri dari empat karakter: U+2004 (three-per-em space), U+2008 (punctuation space), U+2009 (thin space), dan U+202F (narrow no-break space). Alfabet ini menghasilkan kapasitas 2 bit per spasi. Varian kedua, ALPHABET-8, memperluas alfabet dengan menambahkan empat karakter tambahan: U+2000 (en quad), U+2001 (em quad), U+2002 (en space), dan U+2003 (em space), sehingga total alfabet berjumlah delapan karakter dan kapasitas meningkat menjadi 3 bit per spasi ($\log_2(8) = 3$). Varian ketiga, ALPHABET-16, memperluas lebih jauh dengan menambahkan

delapan karakter lagi, menghasilkan alfabet enam belas karakter dengan kapasitas teoritis 4 bit per spasi ($\log_2(16) = 4$). Pemilihan karakter tambahan pada setiap varian mempertimbangkan trade-off antara peningkatan kapasitas dan potensi penurunan robustness akibat penggunaan karakter dengan lebar yang lebih berbeda dari U+0020 [5], [6].

TABLE III. VARIAN ALFABET UNICODE YANG DIUSULKAN DALAM PENELITIAN INI

Varian	Jumlah Karakter	Karakter Tambahan	BPS (Teoritis)
ALPHABET-4(baseline)	4	U+2004, U+2008,U+2009, U+202F	2 bit
ALPHABET-8 (usulan 1)	8	+ U+2000, U+2001,U+2002, U+2003	3 bit
ALPHABET-16(usulan 2)	16	+ U+2006, U+200A,U+205F, dan 5 lainnya	4 bit

Implementasi dilakukan menggunakan bahasa pemrograman Python 3. Untuk setiap varian alfabet, diimplementasikan dua fungsi utama: embed(cover_text, secret_message, alphabet) dan extract(stego_text, alphabet). Fungsi embed mengonversi pesan rahasia ke representasi biner, kemudian menggantikan setiap karakter spasi U+0020 dalam teks cover dengan karakter dari alfabet yang dipilih berdasarkan kelompok bit yang bersesuaian. Fungsi extract melakukan operasi invers: membaca karakter spasi Unicode dari stego text, memetakannya kembali ke kelompok bit, dan merekonstruksi pesan rahasia asli. Implementasi juga mencakup fungsi capacity_analysis(text, alphabet) untuk menghitung kapasitas embedding pada teks cover yang diberikan, serta modul eksperimen untuk menjalankan pengujian secara otomatis pada berbagai ukuran dokumen.

D. Demonstrasi

Tahap demonstrasi dilakukan dengan menjalankan proses embedding dan extraction secara end-to-end pada tiga dokumen teks uji dengan ukuran berbeda: dokumen pendek (~100 kata), dokumen sedang (~500 kata), dan dokumen panjang (~1.000 kata). Dokumen uji menggunakan teks berbahasa Inggris agar jumlah spasi antar kata dapat dikontrol dan dibandingkan secara konsisten. Pada setiap kombinasi dokumen dan varian alfabet, dilakukan pengujian round-trip (embed kemudian extract) untuk memverifikasi bahwa pesan rahasia yang disisipkan dapat dipulihkan kembali secara sempurna tanpa kehilangan bit. Keberhasilan round-trip ini menjadi prasyarat validitas sebelum eksperimen kapasitas dan robustness dilanjutkan.

E. Evaluasi

Tahap evaluasi terdiri dari dua eksperimen terpisah yang masing-masing mengukur dimensi berbeda dari kualitas sistem yang diusulkan, mengacu pada kerangka evaluasi tiga dimensi steganografi yang dikemukakan oleh Majeed et al. [11].

Eksperimen 1: Analisis Kapasitas. Eksperimen ini mengukur kapasitas embedding aktual dari ketiga varian alfabet pada ketiga ukuran dokumen uji. Metrik yang digunakan adalah bits per space (BPS), total kapasitas dalam bytes, dan persentase overhead ukuran file antara dokumen asli dan stego text. Hasil eksperimen disajikan dalam bentuk tabel perbandingan dan grafik batang untuk memvisualisasikan peningkatan kapasitas antar varian alfabet secara intuitif. Sebagai baseline pembanding digunakan hasil kapasitas ALPHABET-4 yang merepresentasikan metode Innamark asli [5].

Eksperimen 2: Analisis Robustness. Eksperimen ini mengevaluasi ketahanan payload yang tersembunyi ketika stego text mengalami transformasi yang umum terjadi dalam skenario komunikasi digital. Tiga jenis serangan yang disimulasikan adalah: (a) serangan normalisasi (normalization attack), yakni penggantian semua karakter spasi Unicode kembali ke U+0020; (b) serangan penghapusan karakter non-ASCII (strip attack); dan (c) serangan pemotongan teks (truncation attack) pada berbagai persentase pemotongan. Metrik yang digunakan adalah survival rate, yakni persentase bit payload yang berhasil dipulihkan setelah serangan, sebagaimana didefinisikan dalam evaluasi robustness pada paper Innamark [5]. Hasil eksperimen disajikan dalam bentuk tabel dan grafik batang untuk memudahkan perbandingan ketahanan antar varian alfabet dan antar jenis serangan.

F. Komunikasi

Tahap komunikasi dari DSRM [18] dalam penelitian ini direalisasikan melalui penulisan makalah ilmiah ini. Seluruh tahapan DSRM yang telah dilalui, mulai dari identifikasi masalah hingga hasil evaluasi eksperimen, didokumentasikan secara sistematis untuk memungkinkan replikasi dan pengembangan lebih lanjut oleh peneliti lain. Kode implementasi Python yang digunakan dalam penelitian ini juga dipublikasikan di repositori GitHub publik untuk mendukung transparansi dan reproduksibilitas penelitian.

IV. HASIL EKSPERIMEN

Seluruh implementasi dilakukan dalam bahasa pemrograman Python 3 pada sistem operasi Windows. Eksperimen dibagi menjadi dua bagian utama: analisis kapasitas embedding dan analisis ketahanan terhadap serangan, serta dua eksperimen tambahan yang dilakukan untuk memperkaya temuan penelitian.

A. Setup Eksperimen

Tiga dokumen teks uji disiapkan dalam bahasa Inggris dengan ukuran yang bervariasi: short.txt (97 kata, 89 spasi, 712 byte), medium.txt (348 kata, 320 spasi, 2.486 byte), dan long.txt (811 kata, 748 spasi, 5.775 byte). Teks berisi konten akademik tentang kriptografi dan steganografi untuk menjaga relevansi kontekstual. Ketiga varian alfabet yang diuji adalah ALPHABET-4 (baseline Innamark, 2 bit/spasi), ALPHABET-8

(3 bit/spasi), dan ALPHABET-16 (4 bit/spasi). Seluruh 9 kombinasi dokumen-alfabet pada eksperimen kapasitas dinyatakan lulus pengujian round-trip, yakni pesan yang disisipkan dapat diekstrak kembali secara sempurna tanpa kehilangan satu bit pun.

B. Eksperimen 1 — Analisis Kapasitas Embedding

Eksperimen pertama mengukur kapasitas embedding aktual dari ketiga varian alfabet pada ketiga dokumen uji. Hasil lengkap disajikan pada Tabel 4. Peningkatan kapasitas yang dihasilkan oleh perluasan alfabet bersifat linear dan dapat diprediksi: ALPHABET-8 menghasilkan kapasitas 1,5 kali lipat dibandingkan ALPHABET-4, sedangkan ALPHABET-16 menghasilkan kapasitas tepat 2 kali lipat. Pada dokumen panjang (long.txt) dengan 748 spasi, ALPHABET-4 hanya mampu menampung 187 byte pesan rahasia, sedangkan ALPHABET-16 mampu menampung hingga 374 byte pada dokumen yang sama.

TABLE IV. HASIL ANALISIS KAPASITAS EMBEDDING PER TEKS PER ALFABET (9 KOMBINASI, SEMUA ROUND-TRIP OK)

Teks	Kata	Spasi	Alfabet	Bit/Spasi	Kapasitas (bit)	Kapasitas (byte)	\ Overhead (%)	Round-trip
short	97	89	ALPHABET_4	2	178	22	25	TRUE
short	97	89	ALPHABET_8	3	267	33	25	TRUE
short	97	89	ALPHABET_16	4	356	44	25	TRUE
medium	348	320	ALPHABET_4	2	640	80	25.7442	TRUE
medium	348	320	ALPHABET_8	3	960	120	25.7442	TRUE
medium	348	320	ALPHABET_16	4	1280	160	25.7442	TRUE
long	811	748	ALPHABET_4	2	1496	187	25.9048	TRUE
long	811	748	ALPHABET_8	3	2244	280	25.9048	TRUE
long	811	748	ALPHABET_16	4	2992	374	25.9048	TRUE

Temuan yang paling signifikan dari eksperimen ini adalah bahwa perluasan alfabet tidak menambah overhead ukuran file sama sekali. Ketiga varian alfabet menghasilkan overhead yang identik pada teks yang sama: 25,00% untuk short.txt, 25,74% untuk medium.txt, dan 25,90% untuk long.txt (lihat Gambar 3). Hal ini disebabkan oleh fakta bahwa hampir semua karakter spasi Unicode yang digunakan dalam ketiga alfabet berukuran 3 byte dalam pengkodean UTF-8, menggantikan karakter spasi konvensional U+0020 yang hanya berukuran 1 byte, sehingga setiap substitusi menambahkan tepat 2 byte overhead terlepas dari alfabet yang digunakan. Dengan demikian, overhead ukuran file ditentukan semata-mata oleh kepadatan spasi dalam teks bukan oleh pilihan alfabet. Ini merupakan keunggulan signifikan dari pendekatan perluasan alfabet dibandingkan pendekatan alternatif seperti multi-substitusi per spasi yang berpotensi menambah overhead.

Overhead juga menunjukkan peningkatan yang sangat gradual seiring bertambahnya panjang teks (25,00% hingga 25,90%), yang disebabkan oleh kecenderungan teks yang lebih panjang untuk memiliki proporsi spasi yang sedikit lebih tinggi. Grafik kapasitas per teks per alfabet disajikan pada Gambar 3, dan grafik overhead ukuran file disajikan pada Gambar 4.

Fig. 3. Bar chart kapasitas embedding

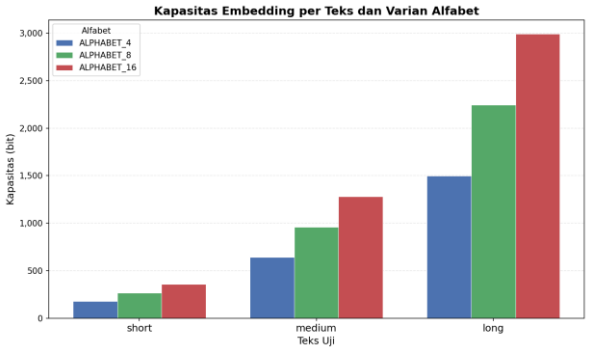
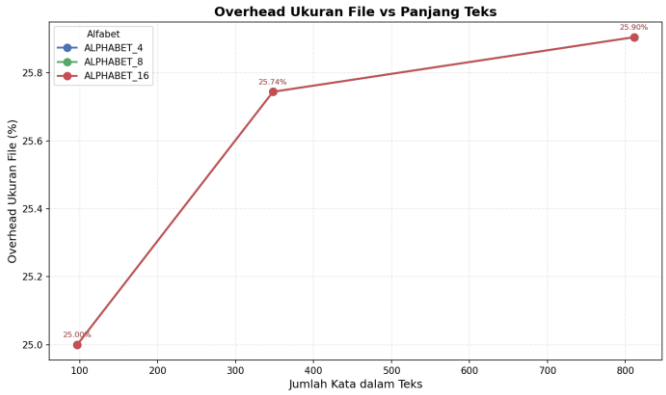


Fig. 4. Line chart overhead ukuran file vs panjang teks



C. Eksperimen 2 — Analisis Ketahanan terhadap Serangan

Eksperimen kedua mengevaluasi ketahanan payload terhadap berbagai jenis serangan yang umum terjadi dalam skenario komunikasi digital. Eksperimen ini menggunakan ALPHABET-8 sebagai varian uji dengan medium.txt (320 spasi) sebagai cover text dan pesan rahasia sepanjang 32 byte. Dari 320 slot spasi yang tersedia, hanya 91 slot yang dibutuhkan untuk menyisipkan payload beserta header 16-bit (91/320 = 28,4%). Hasil lengkap survival rate disajikan pada Tabel 5 dan Gambar 5.

TABLE V. SURVIVAL RATE PAYLOAD PER JENIS SERANGAN (ALPHABET-8, SECRET 32 BYTE, MEDIUM.TXT)

attack name	stego_chars_after	stego_loss_pct	exact_match	\ survival_rate_pct

attack name	stego_chars_after	stego_loss_pct	exact_match	\ survival_rate_pct
baseline	320	0	TRUE	100
normalize	0	100	FALSE	0
strip_unicode	0	100	FALSE	0
double_space	320	0	TRUE	100
truncate_75	248	22.5	TRUE	100
truncate_50	159	50.31	TRUE	100
truncate_25	83	74.06	FALSE	0

Dari tujuh kondisi yang diuji, empat menghasilkan survival rate 100% dan tiga menghasilkan survival rate 0%. Tidak ditemukan hasil parsial di antara kedua nilai ekstrem tersebut. Serangan baseline (tanpa serangan) mengonfirmasi sistem berfungsi normal dengan survival 100%. Serangan normalisasi Unicode (mengganti semua karakter stego kembali ke U+0020) dan serangan penghapusan karakter non-ASCII (strip_unicode) keduanya menghancurkan payload sepenuhnya dengan survival rate 0%. Serangan double-space autocorrect tidak berdampak sama sekali (survival 100%) karena metode embedding TREND mengganti seluruh U+0020 dalam teks dengan karakter stego, sehingga tidak ada pasangan spasi ASCII yang tersisa setelah embedding. Ini merupakan sifat imunitas bawaan dari skema substitusi total.

Untuk serangan pemotongan teks (truncation), sistem berhasil mempertahankan payload hingga pemotongan 50% (survival 100%), namun gagal pada pemotongan 25% (survival 0%). Analisis granular yang dilakukan pada eksperimen tambahan (Eksperimen 3b) mengidentifikasi bahwa ambang batas pemotongan yang tepat, berada pada 28% panjang teks asli, tepat saat 93 slot stego masih tersedia (melampaui 91 yang dibutuhkan). Di bawah 28%, sistem gagal total karena protokol length-prefix 16-bit mengharuskan seluruh bitstream payload tersedia untuk melakukan ekstraksi.

Fig. 5. Bar chart survival rate per serangan

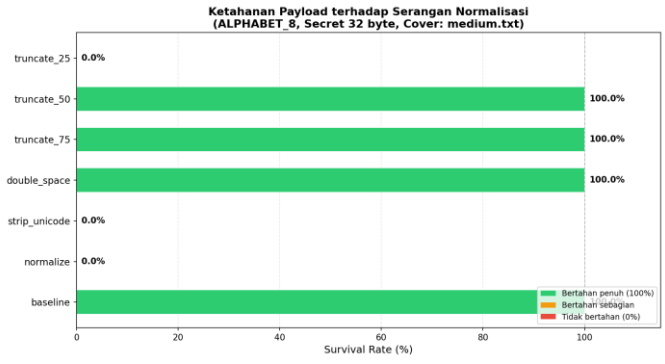
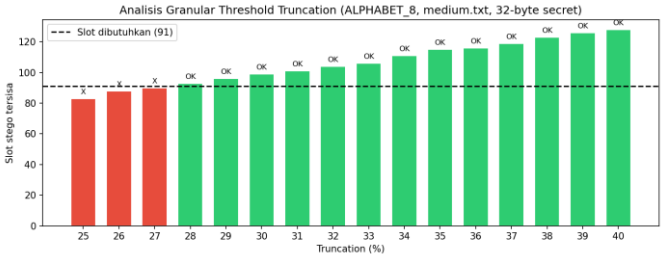


Fig. 6. Granular threshold truncation 25-40%



D. *Eksperimen Tambahan — Teks Cover Bahasa Indonesia*

Sebagai eksperimen tambahan di luar rancangan awal, dilakukan pengujian pada teks cover berbahasa Indonesia (indonesian.txt, 245 kata, 240 spasi, 1.851 byte UTF-8) untuk mengevaluasi sifat language-agnostic dari sistem yang diimplementasikan. Seluruh 9 kombinasi (3 pesan rahasia x 3 alfabet) berhasil melewati pengujian round-trip, termasuk pesan rahasia yang mengandung karakter non-ASCII multibyte (huruf Yunani alfaBetagamma dalam UTF-8). Teks berbahasa Indonesia menunjukkan rasio spasi per kata yang sedikit lebih tinggi dibandingkan teks Inggris (98% vs 92%), namun kapasitas total lebih kecil karena jumlah spasi absolut yang lebih sedikit. Hasil ini mengonfirmasi bahwa sistem benar-benar language-agnostic dan hanya bergantung pada jumlah karakter U+0020 dalam teks cover, bukan pada bahasa atau konten semantiknya.

V. Analisis dan Diskusi

A. *Implikasi Perluasan Alfabet terhadap Kapasitas*

Hasil eksperimen mengonfirmasi hipotesis utama penelitian ini: perluasan alfabet penggantian Unicode secara langsung meningkatkan kapasitas embedding sesuai dengan hubungan logaritmik log2(|alfabet|). Peningkatan dari 2 ke 3 bit per spasi (ALPHABET-8) dan dari 2 ke 4 bit per spasi (ALPHABET-16) menghasilkan peningkatan kapasitas masing-masing sebesar 50% dan 100% dibandingkan baseline Innamark. Temuan ini selaras dengan prediksi teoritis yang didasarkan pada teori informasi Shannon, dan secara empiris memvalidasi argumen Hellmeier et al. [5] bahwa perluasan alfabet merupakan jalur yang menjanjikan untuk mengatasi keterbatasan kapasitas metode baseline.

Temuan yang lebih menarik secara ilmiah adalah identikalnya overhead ukuran file di antara ketiga varian alfabet. Berbeda dari metode-metode sebelumnya seperti UniSpaCh [7] yang overhead-nya bergantung pada jumlah karakter yang disubstitusi, sistem berbasis Unicode whitespace memiliki overhead yang ditentukan semata-mata oleh ukuran UTF-8 dari karakter pengganti. Selama karakter-karakter dalam alfabet yang diperluas juga berukuran 3 byte dalam UTF-8, perluasan alfabet memberikan kapasitas tambahan tanpa biaya overhead apapun. Ini adalah sifat yang sangat menguntungkan dan menjadikan perluasan alfabet sebagai pendekatan yang efisien secara biaya (cost-free capacity increase).

Perlu dicatat bahwa dua karakter dalam ALPHABET-16, yaitu U+3000 (ideographic space, yang lebarnya secara visual jauh lebih besar dari spasi konvensional) dan U+200B (zero-width space, yang memiliki lebar nol), berpotensi melanggar kriteria non-noticeability yang ditetapkan oleh Hellmeier et al. [5]. Pengujian imperceptibility secara sistematis terhadap karakter-karakter ini belum dilakukan dalam penelitian ini dan merupakan salah satu arah penelitian lanjutan yang penting.

B. Analisis Ketahanan dan Perilaku All-or-Nothing

Perilaku all-or-nothing yang ditemukan dalam eksperimen ketahanan merupakan konsekuensi langsung dari desain protokol length-prefix 16-bit yang digunakan. Sistem hanya akan berhasil melakukan ekstraksi jika seluruh bitstream payload, termasuk 16 bit header, tersedia secara utuh. Ketiadaan mekanisme forward error correction (FEC) atau error correcting code (ECC) berarti sistem tidak toleran terhadap kehilangan bit sekecil apapun. Innamark sendiri [5] menyediakan fungsionalitas opsional untuk integrasi ECC melalui InnamarkTags, namun fitur tersebut tidak diimplementasikan dalam penelitian ini demi mempertahankan kesederhanaan implementasi.

Ketahanan nol terhadap serangan normalisasi Unicode (normalize dan strip_unicode) merupakan keterbatasan inheren dari semua metode berbasis substitusi karakter Unicode, tidak terbatas pada sistem yang diusulkan dalam penelitian ini. Ahvanooey et al. [14] mengidentifikasi normalisasi sebagai tantangan fundamental dalam steganografi teks berbasis karakter. Solusi yang paling realistis adalah menggabungkan embedding dengan enkripsi payload sebelum penyisipan, sebagaimana disarankan oleh Hellmeier et al. [5] dan sejalan dengan rekomendasi pada paper Sofian et al. yang dikutip dalam literatur related work Innamark. Dengan enkripsi, bahkan jika serangan normalisasi berhasil mengekstrak payload, isi pesan tetap tidak dapat dibaca tanpa kunci.

Imunitas terhadap serangan double-space, yang merupakan temuan yang tidak diantisipasi dalam rancangan eksperimen awal, memberikan wawasan yang relevan tentang sifat skema substitusi total (full substitution scheme). Berbeda dari metode seperti WhiteSteg [12] yang hanya memodifikasi sebagian spasi, metode berbasis TREND mengganti seluruh U+0020 dalam teks sehingga tidak meninggalkan spasi ASCII yang dapat menjadi target serangan normalisasi berbasis spasi ganda. Ini adalah contoh di mana sifat yang awalnya tampak sebagai keterbatasan (seluruh spasi dimodifikasi) justru memberikan keuntungan ketahanan yang tidak terduga.

C. Formula Ambang Batas Pemotongan dan Implikasinya

Eksperimen granular ambang batas pemotongan (Eksperimen 3b) berhasil memvalidasi secara empiris formula teoritis ambang batas persentase teks minimum yang harus dipertahakan diberikan oleh:

$$p_{\min} = \frac{\lceil (16 + 8N)/b \rceil}{S} \tag{1}$$

dengan N adalah panjang pesan dalam byte, b adalah bit per spasi alfabet yang digunakan, dan S adalah total spasi dalam cover text. Pada konfigurasi eksperimen (N=32, b=3, S=320), formula menghasilkan ambang batas teoritis 28,44%, sedangkan pengujian empiris mengidentifikasi ambang batas 28% (karakter teks). Deviasi kurang dari 1% ini mengonfirmasi bahwa distribusi spasi dalam teks uji cukup merata sehingga persentase karakter teks yang dipertahankan hampir identik dengan persentase slot stego yang tersisa.

Formula ini memiliki implikasi praktis yang signifikan: untuk sistem berbasis TREND, keamanan payload terhadap serangan pemotongan dapat diperhitungkan secara deterministik berdasarkan parameter yang diketahui (panjang pesan, alfabet, dan jumlah spasi cover). Pengguna sistem dapat memilih alfabet yang tepat untuk memaksimalkan redundansi: dengan ALPHABET-16 (4 bit/spasi), ambang batas pemotongan yang sama hanya membutuhkan sekitar 21% teks yang dipertahankan (dibandingkan 28% untuk ALPHABET-8), memberikan ketahanan truncation yang lebih baik sebagai efek samping dari kapasitas yang lebih tinggi.

D. Perbandingan dengan Metode Sebelumnya

Dibandingkan metode yang ada, pendekatan ini unggul dalam robustness lintas platform terhadap UniSpaCh [7] (sebagaimana ditunjukkan benchmark Innamark [5]) meskipun dengan overhead lebih tinggi (~25%), serta lebih tahan dibandingkan AITSteg [8] dan CovertSYS [13] karena karakter spasi berwidth positif tidak difilter sebagian besar platform modern—walau ketiganya berbagi kerentanan terhadap normalisasi Unicode. Kontribusi utama penelitian ini adalah demonstrasi empiris bahwa perluasan alfabet merupakan mekanisme peningkatan kapasitas yang bebas overhead untuk metode steganografi berbasis substitusi Unicode whitespace, memperkuat dan memvalidasi argumen teoritis Hellmeier et al. [5].

E. Keterbatasan Penelitian

Penelitian ini memiliki beberapa keterbatasan yang perlu diakui. Pertama, pengujian imperceptibility dilakukan secara visual informal dan tidak melalui user study formal yang melibatkan partisipan manusia, sehingga klaim non-noticeability untuk karakter-karakter tambahan di ALPHABET-16 belum diverifikasi secara sistematis. Kedua, eksperimen ketahanan hanya menggunakan ALPHABET-8 sebagai varian uji; perbandingan ketahanan lintas varian alfabet belum dilakukan. Ketiga, pengujian robustness lintas platform nyata (copy-paste ke Gmail, WhatsApp, Telegram) dilakukan secara manual dan tidak dikuantifikasi, berbeda dari metodologi benchmark Innamark yang menggunakan dataset berskala besar [5]. Keempat, implementasi ini tidak mengintegrasikan enkripsi payload, sehingga kerahasiaan pesan yang tersembunyi tidak terjamin jika serangan normalisasi berhasil dilakukan.

VI. KESIMPULAN

Penelitian ini mengusulkan perluasan alfabet penggantian karakter spasi Unicode pada metode Innamark untuk mengatasi keterbatasan kapasitas embedding. Dengan memperluas alfabet dari 4 karakter (2 bit/spasi) menjadi 8 karakter (3 bit/spasi) dan 16 karakter (4 bit/spasi), penelitian berhasil mendemonstrasikan peningkatan kapasitas masing-masing sebesar 50% dan 100% dibandingkan baseline. Temuan paling signifikan adalah sifat overhead-free: ketiga varian menghasilkan overhead ukuran file identik (~25–26%) karena hampir seluruh karakter spasi yang digunakan berukuran 3 byte dalam UTF-8.

Eksperimen ketahanan menunjukkan perilaku all-or-nothing sebagai konsekuensi protokol length-prefix 16-bit: sistem imun terhadap serangan double-space autocorrect (survival 100%) namun rentan terhadap normalisasi Unicode (survival 0%), keterbatasan inheren semua metode substitusi karakter. Eksperimen tambahan pada teks Indonesia mengonfirmasi sifat language-agnostic sistem.

Penelitian ini merekomendasikan ALPHABET-16 untuk skenario berkapasitas maksimal, dengan catatan imperceptibility karakter tambahannya perlu diverifikasi melalui user study formal. Untuk penggunaan praktis, disarankan menggabungkan embedding dengan enkripsi payload (mis. AES-256) dan error-correcting code (mis. Reed-Solomon). Arah penelitian lanjutan mencakup: (1) user study imperceptibility, (2) evaluasi robustness lintas platform kuantitatif berskala besar, (3) integrasi enkripsi dan ECC, serta (4) metode seleksi alfabet sistematis berbasis pengujian robustness per karakter.

KODE SUMBER

https://github.com/jackund25/kripto-research

VIDEO LINK AT YOUTUBE

https://youtu.be/cDG8pKcH9z0

ACKNOWLEDGMENT

Penulis mengucapkan puji dan rasa Syukur kepada Allah SWT. karena telah memberikan penulis kemampuan akal dan kesehatan sehingga penelitian ini dapat terlaksanakan. Terima kasih penulis ucapkan kepada Bapak Dr. Ir. Rinaldi Munir, M.T. selaku dosen pengampu mata kuliah II4021 Kriptografi, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, atas bimbingan, materi perkuliahan, dan tugas makalah yang telah mendorong penulis untuk mengeksplorasi topik steganografi teks berbasis Unicode secara mendalam.

REFERENCES

[1] R. J. Anderson, Security Engineering: A Guide to Building Dependable Distributed Systems, 2nd ed. Indianapolis: Wiley, 2008.

[2] F. A. P. Petitcolas, R. J. Anderson, dan M. G. Kuhn, "Information hiding—a survey," Proc. IEEE, vol. 87, no. 7, pp. 1062–1078, Jul. 1999.

[3] M. N. Kamaruddin, A. Kamsin, L. Y. Por, dan H. Rahman, "A review of text watermarking: Theory, methods, and applications," IEEE Access, vol. 6, pp. 8011–8028, 2018.

[4] R. B. Krishnan, P. K. Thandra, dan M. S. Baba, "An overview of text steganography," in Proc. 4th Int. Conf. Signal Process., Commun. Netw. (ICSCN), 2017, pp. 1–6.

[5] M. Hellmeier, H. Norkowski, E.-C. Schrewe, H. Qarawlus, dan F. Howar, "Innamark: A whitespace replacement information-hiding method," arXiv preprint arXiv:2502.12710, Feb. 2025.

[6] The Unicode Consortium, The Unicode Standard, Version 16.0.0. Mountain View, CA: Unicode Consortium, 2024. [Online]. Available: https://www.unicode.org/versions/Unicode16.0.0/

[7] L. Y. Por, K. Wong, dan K. O. Chee, "UniSpaCh: A text-based data hiding method using Unicode space characters," J. Syst. Softw., vol. 85, no. 5, pp. 1075–1082, 2012.

[8] M. T. Ahvanooey, Q. Li, J. Hou, H. D. Mazraeh, dan J. Zhang, "AITSteg: An innovative text steganography technique for hidden transmission of text message via social media," IEEE Access, vol. 7, pp. 19866–19879, 2019.

[9] M. Hellmeier dan F. von Scherenberg, "A hidden digital text watermarking method using Unicode whitespace replacement," in Proc. 58th Hawaii Int. Conf. Syst. Sci. (HICSS), Jan. 2025.

[10] M. Rizzo, F. Bertini, dan D. Montesi, "Content-preserving text watermarking through unicode homoglyph substitution," in Proc. 20th Int. Database Eng. Appl. Symp. (IDEAS), 2016, pp. 97–104.

[11] M. A. Majeed, R. Sulaiman, Z. Shukur, dan M. K. Hasan, "A review on text steganography techniques," Mathematics, vol. 9, no. 21, p. 2829, Nov. 2021.

[12] L. Y. Por, T. F. Ang, dan B. Delina, "WhiteSteg: A new scheme in information hiding using text steganography," WSEAS Trans. Comput., vol. 7, no. 6, pp. 735–745, 2008.

[13] M. T. Ahvanooey, M. X. Zhu, W. Mazurczyk, Q. Li, M. Kilger, K.-K. R. Choo, dan M. Conti, "CovertSYS: A systematic covert communication approach for providing secure end-to-end conversation via social networks," J. Inf. Secur. Appl., vol. 71, p. 103368, 2022.

[14] M. T. Ahvanooey, M. X. Zhu, W. Mazurczyk, dan M. Bendechache, "Information hiding in digital textual contents: Techniques and current challenges," Computer, vol. 55, no. 6, pp. 56–65, Jun. 2022.

[15] N. Shazzad-Ur-Rahman, M. S. Islam, dan M. A. R. Ahad, "An efficient text steganography scheme using Unicode space characters," in Proc. Int. Conf. Comput., Commun., Chem., Mater. Electron. Eng. (IC4ME2), 2017, pp. 1–4.

[16] M. Hellmeier, J. Pampus, H. Qarawlus, dan F. Howar, "Security and detectability analysis of Unicode text watermarking methods against large language models," arXiv preprint arXiv:2512.13325, Dec. 2024.

[17] M. T. Ahvanooey, Q. Li, J. Hou, A. R. Rajput, dan Y. Chen, "Modern text hiding, text steganalysis, and applications: A comparative analysis," Entropy, vol. 21, no. 4, p. 355, 2019.

[18] K. Peffers, T. Tuunanen, M. A. Rothenberger, dan S. Chatterjee, "A design science research methodology for information systems research," J. Manag. Inf. Syst., vol. 24, no. 3, pp. 45–77, 2007.

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

Daffari Adiyatma (13522003)