Implementasi Varian Steganografi DNA yang Ditingkatkan dengan Menggunakan Algoritma RSA dan PRNG

Tri Sulton Adila – 13520033 (*Author*)
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: 3sulton@gmail.com

***Abstract*—Makalah ini bertujuan untuk memperkuat keamanan algoritma steganografi DNA yang ditingkatkan dengan menambahkan variansi baru. Variansi yang diusulkan melibatkan penggunaan algoritma RSA untuk mengenkripsi pesan sebelum disisipkan. Selain itu, PRNG juga digunakan untuk mengacak posisi sisipan pesan dalam sekuens DNA. Implementasi variansi telah dilakukan dengan menggunakan bahasa pemograman python. Hasil pengujian menunjukkan bahwa implementasi dapat berjalan dengan baik dalam melakukan proses encoding dan decoding.**

Keywords—steganografi; dna; prng; rsa; steganografi dna

I. PENDAHULUAN

Dalam dunia keamanan informasi, steganografi menjadi salah satu bidang yang menarik untuk dijelajahi, khususnya pada DNA. Steganografi DNA memanfaatkan sifat unik DNA sebagai media penyimpanan pesan rahasia. Dengan menggunakan teknik ini, pesan rahasia dapat disembunyikan dan tidak terdeteksi di dalam sekuens DNA, entah itu sekuens DNA sintetis maupun DNA organisme hidup. Keunikan ini sudah menarik minat peneliti untuk terus mengembangkan metode steganografi DNA untuk meningkatkan keamanan dan efektivitasnya.

Namun, seperti teknik steganografi pada umumnya, metode steganografi DNA juga memiliki beberapa tantangan dan kelemahan. Salah satu kelemahannya adalah potensi untuk diprediksi dan dideteksi oleh pihak yang tidak berwenang. Oleh karena itu, peningkatan keamanan dalam metode steganografi DNA sangatlah penting. Implementasi variansi pada metode ini dapat menjadi langkah yang efektif dalam meningkatkan tingkat keamanan pesan yang disembunyikan.

Makalah ini bertujuan untuk melakukan implementasi variansi pada Steganografi DNA yang telah ditingkatkan. Variansi yang dilakukan adalah dengan menambahkan algoritma RSA dan pengacakan pesan menggunakan PRNG. Diharapkan dengan ditambahkannya variansi terhadap algoritma steganografi DNA yang sudah ada dapat meningkatkan keamanan pesan yang ingin disembunyikan.

II. LANDASAN TEORI

A. DNA

Dalam bidang biologi, DNA (asam deoksiribonukleat) merupakan molekul yang sangat besar yang terdapat di dalam sel semua organisme hidup. DNA ini mengandung informasi genetik yang penting untuk menjalankan fungsi, reproduksi, dan evolusi suatu organisme. Struktur DNA terdiri dari banyak unit kecil yang disebut nukleotida. Ada empat jenis basa nukleotida dalam DNA, yaitu adenin (A), timin (T), guanin (G), dan sitosin (C). Dua untai DNA terhubung satu sama lain melalui ikatan antara basa-basa ini, dengan adenin berikatan dengan timin, dan sitosin berikatan dengan guanin. Setiap tiga nukleotida bertetangga membentuk apa yang disebut kodon, sehingga ada 64 kombinasi kodon yang berbeda.

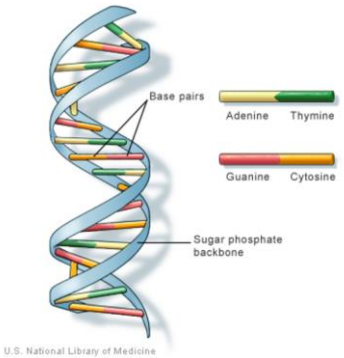

Fig. 1. Sruktur DNA *(U.S. National Library of Medicine)*

B. Steganografi

Steganografi berasal dari bahasa Yunani dan secara harfiah berarti "tulisan tersembunyi". Konsep steganografi melibatkan teknik penyembunyian data rahasia ke dalam media data lain dengan tujuan agar keberadaan data tersebut tidak diketahui atau disadari oleh pihak lain. Dalam merancang sistem steganografi, terdapat beberapa faktor yang harus dipertimbangkan, seperti imperceptibility (tidak terlihat), fidelity (kekakuan), capacity (kapasitas), robustness (ketahanan), recovery (kemampuan pemulihan), dan

undetectability (ketidakdapatan deteksi). Tingkat keberhasilan suatu sistem steganografi akan dinilai berdasarkan sejauh mana faktor-faktor tersebut dapat terpenuhi dengan tingkat yang tinggi. Namun, dalam implementasinya, sulit untuk mencapai tingkat keberhasilan yang tinggi karena faktor-faktor tersebut seringkali saling berkompetisi. Ketika satu faktor ditingkatkan, faktor lainnya mungkin akan mengalami penurunan kualitas, sehingga setiap sistem steganografi memiliki kelebihan dan kekurangan masing-masing.

Dalam dunia steganografi, terdapat beberapa istilah yang umum digunakan. Embedded message atau secret message adalah pesan yang disembunyikan dalam steganografi dan dapat berupa teks, gambar, audio, atau video. Cover object adalah media digital yang digunakan untuk menyembunyikan embedded message dan juga bisa berupa teks, gambar, audio, atau video. Stego object atau stego data adalah cover object yang telah mengandung embedded message setelah dilakukan proses penyisipan. Stego key adalah kunci yang digunakan untuk menyisipkan embedded message ke dalam cover object dan mengekstraksi embedded message dari stego object.

Beberapa metode steganografi telah dikembangkan dalam beberapa tahun terakhir, mulai dari metode yang sederhana hingga yang kompleks dengan menggunakan berbagai persamaan matematika. Salah satu metode sederhana adalah teknik penyisipan Least Significant Bit (LSB) yang bekerja pada ranah spasial. Terdapat juga teknik dalam ranah frekuensi yang memanfaatkan Discrete Cosine Transform (DCT), yang mirip dengan algoritma JPEG. Selain itu, ada juga metode yang menggunakan ranah wavelet dan memanfaatkan sistem chaos yang dikenal sebagai "logistic map". Dengan menggabungkan dan memodifikasi berbagai teknik yang telah ada, diharapkan dapat dikembangkan sistem steganografi yang memiliki kapasitas penyimpanan yang besar dan tingkat ketahanan yang tinggi.

C. Algoritma RSA

RSA adalah salah satu algoritma kunci publik yang paling terkenal dan banyak digunakan dalam enkripsi data sehari-hari. Nama "RSA" sendiri diambil dari inisial tiga peneliti dari MIT yang mengembangkannya, yaitu Ronald Rivest, Adi Shamir, dan Len Adleman pada tahun 1976. Keamanan algoritma RSA terletak pada kesulitan dalam memecahkan faktorisasi bilangan bulat yang besar menjadi faktor-faktor primanya. Dalam kata lain, sulit untuk mengembalikan faktor-faktor prima dari hasil perkalian bilangan bulat besar yang digunakan dalam RSA.

D. PRNG

Generator Bilangan Acak Pseudo (PRNG) adalah suatu metode yang digunakan untuk menciptakan serangkaian angka dengan sifat acak, sehingga angka-angka selanjutnya tidak dapat diprediksi. Penggunaan metode ini sangat luas dalam bidang kriptografi, simulasi, dan permainan elektronik. Metode ini disebut "pseudo" karena PRNG memerlukan "seed" (nilai awal) yang sangat mempengaruhi hasil angka acak yang dihasilkan. Jika seed yang digunakan adalah bilangan yang benar-benar acak, maka angka-angka yang dihasilkan juga akan benar-benar acak tanpa pola yang dapat dikenali.

Metode PRNG menggunakan algoritma matematika untuk menciptakan urutan angka yang terlihat acak, meskipun pada kenyataannya angka-angka tersebut dihasilkan secara deterministik. Hal ini berarti bahwa jika seed yang sama digunakan pada PRNG yang sama, maka urutan angka yang dihasilkan akan selalu identik. Dengan kata lain, PRNG mengikuti aturan matematis yang konsisten, sehingga angka-angka yang dihasilkan dapat diprediksi jika seed-nya diketahui. Meskipun urutan angka yang dihasilkan tidak acak sejati, PRNG masih berguna dalam banyak aplikasi yang membutuhkan kebutuhan acak yang lebih sederhana dan cepat.

E. Steganografi DNA yang Ditingkatkan

Penulis mengambil referensi algoritma steganografi DNA milik Malathi, et al. Dalam algoritmanya, digunakan dua buah key. K1 sebagai initial value yang akan di-XOR dengan karakter terakhir pada pesan, hasilnya akan di-XOR kembali dengan pesan sebelum pesan terakhir, begitu seterusnya. Key lainnya yaitu K2 digunakan untuk membagi sekuens DNA menjadi segmen yang panjangnya sama sebesar K2.

Berikut merupakan tahapan encoding beserta ilustrasinya.

1. Bagi M (pesan) menjadi karakternya dan setiap karakter dikonversi ke binary dengan panjang 8-bit berdasarkan ASCII
2. Secara random, bangkitkan nilai K1 dalam rentang 0 sampai 255. Konversikan K1 ke dalam binary 8-bit.
3. Lakukan XOR karakter terakhir M dengan K1.
4. Hasilnya di-XOR kembali dengan karakter sebelum karakter terakhir. Hasilnya di-XOR kembali dengan karakter sebelumnya lagi, begitu seterusnya sampai semua karakter berhasil dikonversi lalu simpan hasilnya ke dalam A.
5. Konversikan A ke dalam bentuk sekuens DNA.
6. Siapkan cover DNA S dan konversikan menjadi sekuens binary. Proses konversi mengikuti tabel berikut ini.

TABLE I. KONVERSI DNA KE BINER

DNA nucleotide	***Binary***
A	00
C	01
G	10
T	11

7. Bangkitkan nilai K2, disarankan nilainya kecil karena nilai ini akan membagi sekuens DNA S menjadi segmen. Panjang segmen adalah sebesar K2.
8. Tambahkan binary pertama milik A di awal segmen binary, lalu masukkan binary kedua milik A di awal segmen kedua, begitu seterusnya.

9. Gabung semua sekuens binary segmen dan konversikan ke dalam sekuens DNA.

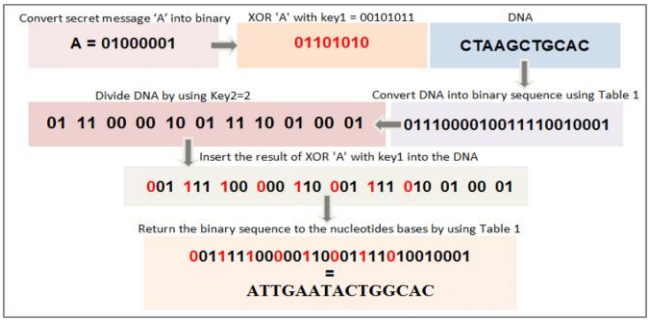

Fig. 2. Ilustrasi Proses Encoding *(Al-Harbi, 2020)*

Berikut merupakan tahapan encoding beserta ilustrasinya.

1. Konversi sekuens DNA palsu ke dalam sekuens binary berdasarkan Table 1.
2. Bagi binary DNA ke dalam segmen. Setiap segmennya memiliki panjang sebesar K2 + 1.
3. Ambil bit pertama dari setiap segmen dan gabungkan semuanya menjadi B.
4. Hitung nilai XOR 8 bit pertama dari B dengan K1 dan hitung XOR 8-bit kedua dengan 8-bit sebelumnya, begitu seterusnya.
5. Konversikan sekuens binary DNA ke dalam teks berkarakter ASCII.

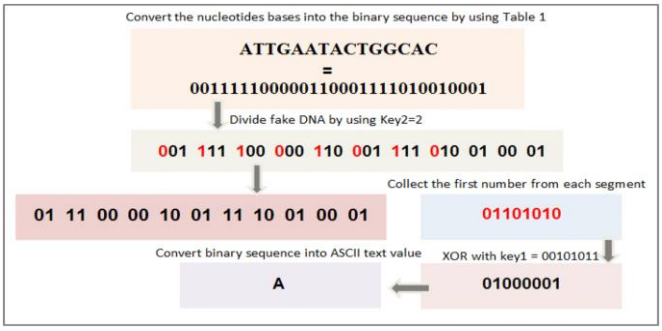

Fig. 3. Ilustrasi Proses Decoding *(Al-Harbi, 2020)*

III. IMPLEMENTASI

Pada makalah ini akan diimplementasikan variansi dari steganografi DNA yang ditingkatkan. Varian yang ditambahkan adalah dengan melakukan pengacakan pesan menggunakan PRNG dan sebelum dimasukkan ke dalam sekuens DNA cover, pesan yang telah diacak tersebut dilakukan enkripsi dengan menggunakan algoritma RSA. Jumlah key semula yang hanya terdapat dua buah, sekarang menjadi enam buah. Empat tambahan kunci lainnya disebabkan oleh pengacakan PRNG yang membutuhkan satu kunci untuk seed dan tiga buah kunci untuk enkripsi dekripsi RSA.

Implementasi dari variasi steganografi DNA ini menggunakan bahasa pemrograman Python. Python dipilih karena memiliki berbagai fitur yang berguna untuk memanipulasi dan memproses string, seperti sekuens DNA. Tersedia juga beberapa modul bawaan yang mendukung sehingga pernulis tidak perlu menuliskan ulang semua implementasi fungsi dari nol. Implementasi dilakukan mulai dari membuat beberapa fungsi utilitas, fungsi PRNG, fungsi RSA, dan fungsi utama untuk melakukan encoding pesan dan decoding steganografi. Berikut merupakan rincian implementasi tersebut.

A. Implementasi Fungsi Utilitas

Fungsi utilitas berikut ini merupakan kumpulan dari beberapa fungsi yang dibuat untuk mengerjakan tugas-tugas tertentu yang cukup sering digunakan dalam implementasi steganografi DNA ini.

- binary(c): Mengembalikan representasi karakter ASCII sebagai bilangan biner dengan panjang 8-bit. Contohnya fungsi ini akan mengembalikan string "01000101" apabila masukannya adalah "E"
- asciiCode(b): Mengembalikan kode ASCII integer suatu binary string. Contohnya fungsi ini akan mengembalikan 69, yaitu nilai integer dari "01000101".
- xor(b1, b2) : Mengembalikan hasil operasi XOR dari dua buah binary sepanjang 8-bit.
- seqDNA2Bin(seqDNA): Mengembalikan hasil konversi dari sekuens DNA menjadi sekuens binary deengan bantuan dictionary dnaDict. Nilai DNA A menjadi 00, C menjadi 01, G menjadi 10, dan T menajdi 11.
- seqBin2DNA(seqBin): Mengembalikan hasil konversi dari sekuens binary menjadi sekuens DNA deengan bantuan dictionary dnaDict. Nilai DNA 00 menjadi A, 01 menjadi C, 10 menjadi G, dan 11 menajdi T.

```
dnaDict = {"A" : "00", "C" : "01", "G" : "10", "T" :
"11", "00" : "A", "01" : "C", "10" : "G", "11" : "T"}

def binary(c):
    if type(c) == str:
        c = ord(c)
    return bin(c)[2:].zfill(8)

def asciiCode(b: str):
    return int(b, 2)

def xor(b1: str, b2: str) -> str:
    return format(int(b1, 2) ^ int(b2, 2), '08b')

def seqDNA2Bin(seqDNA):
    dnaBinary = ""
    for dna in seqDNA:
        dnaBinary += dnaDict[dna]
    return dnaBinary

def seqBin2DNA(seqBin):
    steganoSeq = ""
    for i in range(0, len(seqBin), 2):
        steganoSeq += dnaDict[seqBin[i:i+2]]
    return steganoSeq
```

B. Implementasi PRNG

Beberapa fungsi ini bertanggung-jawab untuk menghasilkan angka acak berdasarkan seed yang diberikan. Fungsi random_integer_list mengembalikan list yang berisi bilangan bulat dari 0 hingga panjang tertentu. Urutan angka pada list dalam keadaan teracak. Hasil dari list ini akan dijadikan sebagai acuan indeks untuk fungsi randomize_word dalam mengacak suatu string dan restore_word dalam memulihkan string acak ke posisi awal sebelum terjadi pengacakan.

```
import random

def random_integer_list(length, seed):
    random.seed(seed)
    integer_list = list(range(length))
    random.shuffle(integer_list)
    return integer_list

def randomize_word(word, seed):
    randomized_word = ""
    index_list = random_integer_list(len(word), seed)
    for i in index_list:
        randomized_word += word[i]
    return randomized_word

def restore_word(randomized_word, seed):
    restored_word = list(randomized_word)
    index_list =
random_integer_list(len(randomized_word), seed)
    j = 0
    for i in index_list:
        restored_word[i] = randomized_word[j]
        j += 1
    return ''.join(restored_word)
```

C. Implementasi RSA

Implementasi yang dilakukan pada RSA cukup sederhana karena hanya menerapkan proses perhitungan num^e mod N. Diasumsikan pengguna implementasi ini telah memiliki kunci untuk enkripsi dan dekripsi yang valid sehingga tidak diperlukan pengecekan kunci dan hanya berfokus pada proses enkripsi dekripsi saja.

```
def rsa(num: int, e, N) -> int:
    return pow(num, e, N)
```

D. Implementasi Encoding

Proses encoding terdiri atas beberapa tahap dan melibatkan beberapa parameter. Berikut penjelasan parameter yang digunakan.

- cover: Sekuens DNA berupa string yang digunakan sebagai wadah untuk menyimpan pesan. Pastikan cover cukup panjang untuk menampung pesan yang akan disembunyikan.
- M: Pesan berupa string yang akan disisipkan.
- k1: Kunci sebagai initial value yang akan di-XOR dengan baris terakhir pesan. Batasan nilai k1 adalah 0 hingga 255, sesuai dengan rentang karakter ASCII.
- k2: Kunci yang menyatakan panjang setiap segmen pada cover. Diasumsikan pengguna memilih nilai k2 yang memenuhi persamaan berikut ini.

$$\lfloor 2length(\text{cover}) / k_2 \rfloor \geq 8length(\text{msg})$$

Hal ini untuk memastikan tersedianya jumlah segmen untuk menampung pesan. Contoh untuk cover degnan ukuran 84 dan panjang pesan adalah 7, nilai maksimal k2 yang dapat memenuhi adalah 3. Tidak bisa mengambil nilai k2 di atas 3 karena akan membuat jumlah segmen kurang dari pesan yang ingin disisipkan.

- k3: Kunci sebagai seed dalam pengacakan isi pesan.
- kRSA: Pasangan 2 buah nilai kunci RSA yaitu e dan N. Dalam implementasi ini terdapat batasan nilai N, yaitu nilai N harus lebih kecil atau sama dengan 256. Batasan ini muncul akibat hasil enkripsi akan dimasukkan ke dalam cover sebagai binary dengan panjang delapan bit (kode integer ASCII. Nilai maksimal ASCII adalah 255. Apabila hasil dari enkripsi melebihi nilai 255, ini akan membuat reprentasi bit tidak lagi dalam cakupan karakter ASCII dan berisiko membuat panjang bit menjadi lebih dari besar dari delapan.

Berikut merupakan tahap-tahap yang dilakukan.

1. Mengonversi cover yang semula berupa sekuens DNA menjadi bentuk binary.
2. Mengacak urutan pesan menggunakan k3 sebagai seed.
3. Mengenkripsi setiap karakter pesan menggunakan algoritma RSA
4. Menghitung nilai XOR. Awalnya karakter terakhir pesan di-XOR dengan k1. Hasil dari XOR awal akan digunakan untuk perhitungan XOR pada karakter berikutnya (sebelum karakter pesan terakhir). Lakukan ini sampai karakter pesan pertama telah dilakukan operasi XOR.
5. Membagi cover sehingga terdiri atas potongan segmen. Setiap segmen memiliki panjang sebesar k2.
6. Menambahkan satu bit dari hasil tahap 4 ke posisi awal segmen. Bit pertama dimasukkan ke segmen pertama. Bit kedua dimasukkan ke segment kedua. Begitu seterusnya hingga bit dan segmen terakhir.
7. Menggabungkan setiap segmen yang telah ditambahkan bit hasil XOR. Segmen yang tidak memperoleh kesempatan untuk digabung dengan hasil XOR akan diabaikan dan tidak akan digunakan pada tahap selanjutnya.
8. Mengonversi hasil gabungan yang semula adalah sekuens binary menjadi sekuens DNA.

```
def encode(cover, M, k1, k2, k3, kRSA):
    dnaBinary = seqDNA2Bin(cover)

    randomizedM = randomize_word(M, k3)

    res = binary(k1)
    resTotal = ""
    for m in reversed(randomizedM):
        m = rsa(ord(m), kRSA[0], kRSA[1])
```

```
        res = xor(binary(m),res)
        resTotal += res

    chunked_cover = [dnaBinary[i:i+k2] for i in
range(0, len(dnaBinary), k2)]
    i = 0
    for bm in resTotal:
        chunked_cover[i] = bm + chunked_cover[i]
        i += 1

    steganoBin = ''.join([chunk for chunk in
chunked_cover if len(chunk) == k2+1])

    steganoSeq = seqBin2DNA(steganoBin)
    return steganoSeq
```

E. *Implementasi Decoding*

Pada implementasi decoding, parameter yang digunakan dan tahap-tahap yang dilewati memiliki kemiripan. Parameter cover serta M dihilangkan dan diganti dengan stegano yang merupakan sekuens DNA steganografi. Urutan tahap-tahap yang dilewati berkebalikan dengan implementasi encoding. Berikut merupakan tahap-tahap yang dilakukan.

Berikut merupakan tahap-tahap yang dilakukan.

1. Mengonversi stegano yang semula berupa sekuens DNA menjadi bentuk binary.
2. Membagi hasil konversi sehingga terdiri atas potongan segmen. Setiap segmen memiliki panjang sebesar k2 + 1.
3. Mengambil most-significant bit (MSB) dari setiap segmen lalu menggabungkan setiap bit menjadi satu sekuens binary.
4. Menghitung nilai XOR untuk setiap segmen sebesar delapan bit pada kumpulan MSB hasil tahap tiga. Awalnya segmen pertama di-XOR dengan k1. Selanjutnya, untuk setiap segmen mulai dari segmen kedua akan dihitung nilai XOR-nya dengan segment sebelumnya. Lakukan ini sampai segmen terakhir telah dilakukan operasi XOR. Hasil dari XOR merupakan segmen yang berisi karakter pesan terenkripsi.
5. Mendekripsi setiap karakter pesan menggunakan algoritma RSA
6. Mengonversi hasil gabungan yang semula adalah sekuens binary menjadi sekuens DNA.
7. Memulihkan kata yang semulanya masih teracak ke dalam bentuk awalnya dengan menggunakan k3 sebagai seed.

```
def decode(stegano, k1, k2, k3, kRSA):
    dnaBinary = seqDNA2Bin(stegano)

    msgBinary = ''.join([dnaBinary[i] for i in
range(len(dnaBinary)) if i % (k2+1) == 0])

    msgSegment = [msgBinary[i:i+8] for i in range(0,
len(msgBinary), 8)]
    msgSegmentRes = [xor(binary(k1), msgSegment[0])]
    for i in range(1, len(msgSegment)):
        msgSegmentRes.append(xor(msgSegment[i-1],
msgSegment[i]))
```

```
    randomized_word = "".join([chr(rsa(asciiCode(b),
kRSA[0], kRSA[1])) for b in reversed(msgSegmentRes)])

    M = restore_word(randomized_word, k3)
    return M
```

IV. PENGUJIAN

Pengujian dilakukan dengan menggunakan parameter sebagai berikut.

- M = “WELCOME”
- cover = “ACGGTTCCAATGCCTAAGCTAACGACGGTTCCAATGACGGTTCCAATGACGGTTCCAATGACGGTTCCAATGACGGTTCCAATG”
- k1 = 60, k2 = 3, k3 = 42
- eRSA = 5, dRSA = 77, N = 119

A. *Encoding*

Tahap 1: Mengonversi sekuens DNA cover menjadi binary menghasilkan

0001101011110101000011100101110000100111000000110000110101111010100001110000110101111010100001110000110101111010100001110000110101111010100001110000110101111010100001110

Tahap 2: Mengacak pesan menghasilkan

ECOLEWM

Tahap 3: Mengenkripsi pesan dan melakukan operasi XOR menghasilkan

0001011001100011001001100100100100010110000001100100 0011

Tahap 4: Menyisipkan bit hasil XOR ke dalam segmen dan menggabungkannya menghasilkan

0000011001011111001011001001011000101111100000010001101001000111000000110110101110010110010010110000011100101011111010010000111100000011001011111001011001001011000001100101011100101100100101100000111001010111001001001001110

Tahap 5: Mengonversi sekuens binary menjadi sekuens DNA menghasilkan

AACGCCTTAGTAGCCGAGTTGAAGATCAGATGAACGTCCTAGTAGCCGAATGCCCTGGCAACTGAACGCCTTAGTAGCCGAACGCCCTAGTAGCCGAATGCCCTAGCAGCTG

Sampai tahap ini, stegano DNA telah berhasil dibentuk.

B. *Decoding*

Tahap 1: Mengonversi sekuens DNA stegano menjadi binary menghasilkan

000001100101111100101100100101100010111110000010001 101001000111000000110110101110010110010010110000011 100101011110100100000111100000011001011111001011001 001011000000110010101110010110010010110000011100101 01110010010010011110

Tahap 2: Membagi hasil konversi menjadi segmen dan mengambil MSB setiap segmen menghasilkan

00010110011000110010011001001001000101100000011001000011

Tahap 3: Menghitung nilai XOR setiap segmen, mendekripsi, dan mengoversinya menjadi karakter ASCII menghasilkan

ECOLEWM

Tahap 4: Memulihkan pengacakan kata ke kondisi semula menghasilkan

WELCOME

Sampai tahap ini, pesan tersembunyi dalam stegano telah berhasil ditemukan.

V. Kesimpulan dan Saran

Berdasarkan hasil pengujian encoding dan decoding menggunakan varian steganografi yang ditingkatkan yang telah dilakukan, dapat disimpulkan bahwa hasil implementasi dapat berjalan dengan baik. Proses encoding berhasil menyisipkan pesan menjadi stegano DNA yang tersembunyi dalam DNA cover. Proses decoding juga berhasil mengembalikan pesan asli dari stegano DNA. Steganografi DNA memberikan keamanan tambahan karena pesan tersembunyi dalam sekuens DNA cover sehingga akan sulit dideteksi oleh orang lain. Ditambah lagi, penggunaan enkripsi RSA dan juga pengacakan kata dapat membuat keamanannya lebih terjaga.

Saran yang dapat diberikan adalah supaya dilakukan pengjuian lebih lanjut dengan ukuran pesan yang lebih besar dan variasi sekuens DNA cover supaya dapat dilihat bagaimana implementasi ini dapat digunakan dalam berbagai kondisi. Selanjutnya dapat juga dengan menguji performa dengan menggunakan metrik tertentu. Terakhir adalah dengan memikirkan kemungkinan serangan sekaligus metode pendeteksiannya terhadap steganografi DNA.

Ucapan Terima Kasih

Pertama-tama penulis mengucapkan puji syukur kepada Allah Yang Maha Esa karena berkat limpahan nikmat, rahmat, dan karunia-Nya, penulis dapat menyelesaikan makalah berjudul “Implementasi Varian Steganografi DNA yang Ditingkatkan dengan Menggunakan Algoritma RSA dan PRNG”. Penulis mengucapkan terima kasih kepada dosen pengajar II4031 Kriptografi dan Koding 1, yaitu Dr. Ir. Rinaldi, M.T. karena berkat beliau, penulis mendapatkan bimbingan dan ilmu tentang kriptografi khususnya pada materi steganografi. Tidak lupa penulis mengucapkan terima kasih kepada orang tua yang telah memberikan dukungan baik moral maupun material. Penulis juga mengucapkan terima kasih kepada para penulis sumber referensi yang telah menyediakan berbagai informasi yang dibutuhkan agar makalah ini dapat terselesaikan.

Referensi

[1] Al-Harbi, O.A., Alahmadi, W.E. & Aljahdali, A.O. Security analysis of DNA based steganography techniques. SN Appl. Sci. 2, 172 (2020). https://doi.org/10.1007/s42452-019-1930-1

[2] Malathi, P., Manoaj, M., Manoj, R., Raghavan, V., & Vinodhini, R. E. (2017). Highly improved DNA based steganography. Procedia Computer Science, 115, 651-659.

[3] R. Munir, “Steganografi”, Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2022-2023/05-Steganografi-2023.pdf. Retrieved 22 May 2023

[4] R. Munir, “Algoritma RSA”, Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2022-2023/10-Algoritma-RSA-2023.pdf. Retrieved 22 May 2023

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, 22 Mei 2023

Tri Sulton Adila 13520033