Jurnal EurekaMatika

Journal homepage: https://ejournal.upi.edu/index.php/JEM

# Penggunaan Fungsi Hitzl-Zele pada Implementasi Gabungan *Secret Sharing Shamir's* (*t*, *w*)-*Threshold Scheme* dan Steganografi Audio *Least Significant Bit*

*Miftah Fadillah Sopian, Siti Fatimah, dan Rini Marwati**

Program Studi Matematika, Fakultas Pendidikan Matematika dan Ilmu Pengetahuan Alam, Universitas Pendidikan Indonesia

*Correspondence e-mail: rinimarwati@upi.edu

## A B S T R A K

Seiring dengan kemudahan dan kecepatan pertukaran informasi melalui berbagai media, keamanan informasi menjadi semakin sulit dikendalikan. Untuk mengatasi risiko kebocoran informasi dan ancaman lainnya, dilakukan kombinasi antara kriptografi dan steganografi. Kriptografi *Shamir's* (*t*, *w*)-*Threshold Scheme* memungkinkan informasi rahasia dibagi menjadi beberapa bagian atau *share* yang didistribusikan kepada sejumlah pemegang kunci, sehingga tidak ada pemegang kunci tunggal. Steganografi audio dengan teknik *Least Significant Bit* (*LSB*) diterapkan untuk menyembunyikan *share* tersebut ke dalam file audio WAV, menggunakan fungsi Hitzl-Zele untuk menentukan lokasi penyematan dalam data audio. Kombinasi ini diimplementasikan pada aplikasi yang dibangun menggunakan Python, yang dinamakan "(*t*, *w*)-*Threshold Scheme and LSB Hitzl-Zele*". Aplikasi ini menunjukkan performa yang baik dalam menjaga kerahasiaan informasi, dengan nilai PSNR sebesar 112,43 desibel (dB).



## INFORMASI ARTIKEL

***Sejarah Artikel:***

*Diterima 13 September 2024*
*Direvisi 28 April 2025*
*Disetujui 1 Mei 2025*
*Tersedia online 2 Mei 2025*
*Dipublikasikan 2 Mei 2025*

***Kata Kunci:***
*Fungsi Hitzl-Zele,*
*Kriptografi,*
*Least Significant Bit,*
Skema Pembagian Rahasia,
*Steganografi Audio.*

## A B S T R A C T

*As information exchange becomes easier and faster through various media, ensuring information security has become increasingly difficult. To address the risks of information leakage and other threats, a combination of cryptography and steganography is employed. Shamir's (t, w)-Threshold Scheme in cryptography allows secret information to be divided into multiple parts, or shares, which are distributed among several key holders, so no single key holder possesses the entire information. Audio steganography using the Least Significant Bit (LSB) technique is applied to hide these shares in WAV audio files, utilizing the Hitzl-Zele function to determine the embedding locations within the audio data. This combination is implemented to an application built using Python, named "(t, w)-Threshold Scheme and LSB Hitzl-Zele." The application demonstrates a good performance in maintaining information confidentiality, achieving a PSNR value of 112.43 decibels (dB).*



***Keywords:***
*Audio Steganography,*
*Cryptography,*
*Hitzl-Zele Function,*
*Least Significant Bit,*
*Secret Sharing Scheme.*

## 1. PENDAHULUAN

Saat ini pertukaran informasi semakin mudah dan cepat dilakukan dan tidak dibatasi media informasinya. Hal tersebut mengakibatkan lalu lintas informasi-informasi tersebut sangat besar sehingga sulit dipantau keamanannya. Data dari perusahaan keamanan siber Surfshark dalam Katadata Insight Center yang ditayangkan pada laman https://databoks.katadata.co.id/datapublish/2022/09/13/indonesia-masuk-3-besar-negara-dengan-kasus-kebocoran-data-terbanyak-dunia dan diakses pada 5 Desember 2023 menjelaskan bahwa Indonesia menempati urutan ke-3 dengan jumlah kasus kebocoran data terbanyak di dunia setelah Rusia dan Prancis per kuartal III-2022. Oleh karena itu, informasi rahasia yang akan dibagikan harus dilindungi untuk menghindari kebocoran informasi dan ancaman keamanan informasi lainnya. Belakangan ini, banyak digunakan metode kriptografi dan steganografi dalam menjaga informasi rahasia dari ancaman keamanan informasi.

Kriptografi adalah istilah dari Bahasa Yunani dari kata "$cryptós$" yang berarti "*secret*" (rahasia) dan kata "$gráphein$" yang artinya "*writing*" (tulisan) (Amin, 2016). Di sisi lain, steganografi merupakan istilah dari Bahasa Yunani, *stegos* yang artinya "atap" atau "penutup" (*cover*) dan *graphia* yang artinya "tulisan" (*writing*). Dengan demikian, steganografi secara keseluruhan artinya "tulisan yang tertutup" (*covered writing*) (Dutta *et al*, 2019)

Metode kriptografi umumnya mengandalkan satu kunci untuk mendekripsi informasi yang terenkripsi, yang sering disimpan di lokasi aman seperti komputer atau ingatan manusia. Namun, jika kunci ini hilang, rusak, atau terlupakan, informasi tidak dapat diakses, bahkan oleh pemiliknya. Di sisi lain, penyimpanan kunci di banyak tempat meningkatkan risiko keamanan, seperti serangan siber dan kelalaian. Masalah ini dapat diatasi dengan skema pembagian rahasia Shamir (Shamir, 1979), yang membagi informasi rahasia menjadi beberapa bagian yang didistribusikan kepada beberapa pemegang kunci. Informasi asli hanya bisa diakses kembali jika sejumlah pemegang kunci bekerja sama, meningkatkan keamanan karena tidak ada satu pemegang kunci yang bisa mengakses informasi secara keseluruhan. Skema ini, yang dikenal sebagai *Shamir's (t, w)-Threshold Scheme* (skema (*t, w*)), memerlukan seorang *Dealer* untuk membagi rahasia menjadi *w* bagian, dengan *t* sebagai jumlah minimum pemegang kunci (*Participant*) yang diperlukan untuk merekonstruksi rahasia tersebut.

*Least Significant Bit* (*LSB*) audio adalah metode steganografi audio yang menyembunyikan informasi rahasia dalam bit terakhir dari setiap data audio (Anti *et al*., 2017). Penyematan bit dapat dilakukan secara berurutan atau menggunakan pemilihan lokasi data audio acak dengan *chaotic map*, seperti Hitzl-Zele, Ikeda, dan lainnya (Naik & Singh, 2022), yang meningkatkan keamanan informasi (Kordov & Stoyanov, 2017). Dalam penelitian ini, *LSB* digunakan dengan fungsi Hitzl-Zele untuk memilih lokasi data audio acak sebagai tempat penyematan bit.

*Peak Signal-to-Noise Ratio* (*PSNR*) adalah ukuran yang digunakan untuk menilai kualitas rekonstruksi dari sinyal seperti gambar, audio, dan video setelah mengalami proses kompresi atau transmisi. *PSNR* dihitung berdasarkan perbandingan antara nilai maksimum yang bisa dihasilkan dari sinyal dengan besarnya derau pada sinyal tersebut yang dinyatakan dengan

galat (Herlinawati, 2016). Dalam steganografi audio, *PSNR* dapat digunakan untuk mengukur sejauh mana proses *embedding* mempengaruhi kualitas audio asli. Dengan kata lain, *PSNR* digunakan untuk mengevaluasi seberapa baik kualitas audio yang telah direkonstruksi, yaitu *stego-audio* dibandingkan dengan audio asli, yaitu *cover-audio* (Deshpande *et al*., 2018).

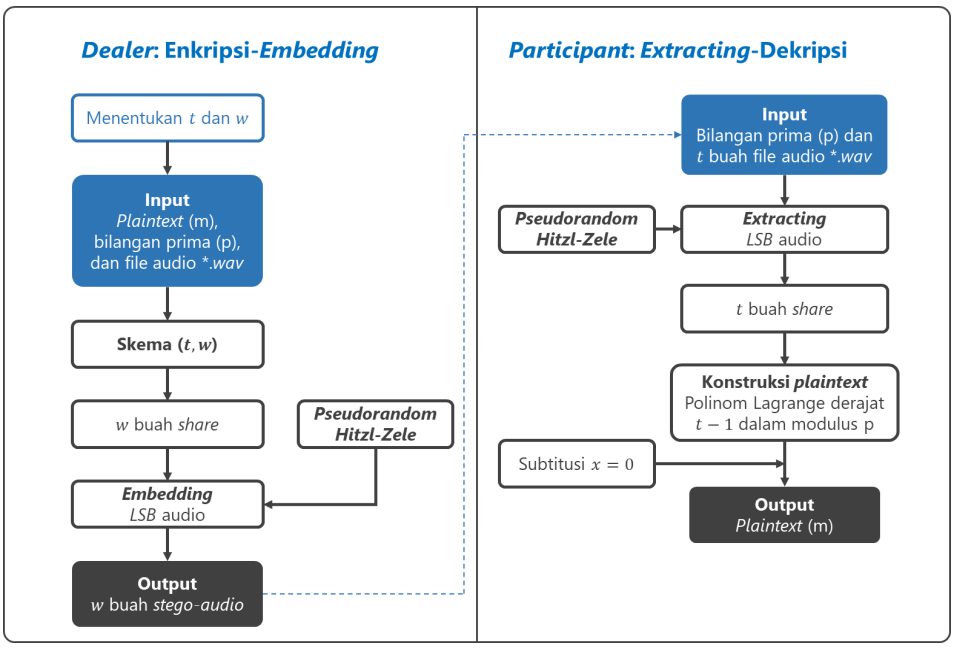
Salah satu penelitian terdahulu oleh Al-Hooti, dkk. (2016) membahas penggunaan metode *LSB* pada audio 16-bit untuk penyembunyian informasi rahasia dan mengevaluasi kualitas audio stego menggunakan metrik *PSNR*. Penelitian ini menjadi relevan untuk dibandingkan dalam mengukur kualitas steganografi audio berbasis *LSB*. Penelitian sebelumnya terkait *LSB* dengan fungsi Hitzl-Zele telah dilakukan oleh Kordov dan Stoyanov (2017), mereka meneliti meneliti tentang penyembunyian informasi pada media gambar. Penelitian tersebut menggunakan *LSB* untuk menyematkan informasinya pada *pixel* gambar yang dipilih dengan Hitzl-Zele *chaotic map*. Penelitian sebelumnya yang membahas penggabungan kriptografi dan steganografi adalah penelitian yang dilakukan Humaira *et al*. (2023) terkait kriptografi skema (*t*, *w*) dan steganografi *LSB* pada file audio WAV. Namun, penelitian tersebut membatasi masalah pada skema (3, 4) dan hanya mengakomodasi *plaintext* berupa bilangan bulat 6 digit serta penyematan *LSB* dilakukan pada data audio secara berurutan.

Pembatasan masalah pada penelitian sebelumnya menjadi dorongan dilakukannya penelitian ini. Oleh karena itu, pada penelitian ini mengkaji kriptografi skema (*t*, *w*), yaitu skema untuk membagi *plaintext* berupa bilangan bulat nonnegatif kepada *w participant* dengan jumlah minimum *participant* yang dapat mengonstruksi kembali *plaintext* adalah *t participant* dengan $w, t \in Z;\ 2 \leq t \leq w \leq 50$; dan steganografi audio *Least Significant Bit* menggunakan fungsi Hitzl-Zele serta implementasinya pada program komputer yang menghasilkan skema (*t*, *w*) dan disematkan pada *w* buah file audio WAV.

## 2. METODE

Metode yang digunakan dalam pengamanan *plaintext* pada penelitian ini adalah penggabungan kriptografi skema (*t*, *w*) dan steganografi audio *LSB* menggunakan fungsi Hitzl-Zele. *Plaintext* akan dienkripsi dengan skema (*t*, *w*) menjadi *w* buah bagian yang disebut sebagai *share*, kemudian *share* tersebut disematkan ke dalam file audio WAV menggunakan *LSB* dengan penyematan berdasarkan fungsi Hitzl-Zele, sehingga diperoleh *w* buah file audio WAV yang berisi *share*, disebut sebagai *stego-audio*. Untuk mendapatkan kembali *plaintext*, diperlukan minimum *t* buah *stego-audio* untuk dilakukan ekstraksi berdasarkan fungsi Hitzl-Zele, sehingga diperoleh *t* buah *share*, kemudian *share* tersebut didekripsi menggunakan polinomial Lagrange derajat *t-1*, dilakukan substitusi $x = 0$ ke polinomial tersebut sehingga diperoleh kembali *plaintext*.

Gambar 1 adalah skema penggabungan tersebut. Terdapat 5 proses utama untuk mengamankan *plaintext* dalam skema tersebut, antara lain enkripsi skema (*t*, *w*), *pseudorandom bit generator Hitzl-Zele*, *Embedding LSB*, *Extracting LSB*, dan dekripsi polinomial Lagrange.



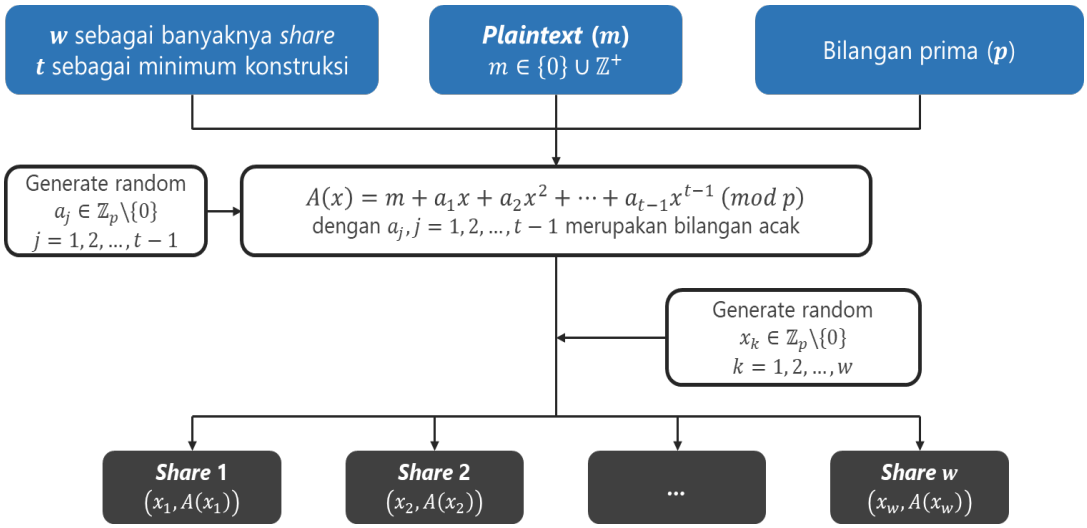

**Gambar 1.** Skema Penggabungan Kriptografi Skema (*t*, *w*) dan Steganografi Audio *LSB* Menggunakan Fungsi Hitzl-Zele

### 2.1 Enkripsi Skema (*t*, *w*)

Pada proses ini, *Dealer* membagi *plaintext* menjadi beberapa *share* dengan langkah-langkah sebagai berikut.

1. *Dealer* menentukan masukan yaitu *plaintext* $(m)$, banyaknya *share* $(w)$, minimum konstruksi $(t)$, dan bilangan prima $(p)$.
2. Dilakukan pengecekan terhadap masukan tersebut supaya memenuhi $w, t, m, p \in Z, 2 \leq t \leq w < p$, dan $0 \leq m < p$.
3. Masukan $m, t$, dan $p$ digunakan untuk membangun polinomial $A(x) = m + a_1 x + \cdots + a_{t-1}\, x^{t-1} \ (mod\ p)$, dengan $a$ adalah bilangan acak anggota $Z_p \backslash \{0\}$.
4. Polinomial $A(x)$ digunakan untuk membangun $w$ buah *share* dengan cara memilih bilangan acak $x_k \in Z_p \backslash \{0\}$, untuk $k = 1, .., w$. Selanjutnya substitusi masing-masing $x_k$ ke polinomial $A(x)$ sehingga diperoleh $w$ buah *share*, yaitu $(x_1, A(x_1)), \ldots, (x_w, A(x_w))$.

Langkah-langkah *Dealer* dalam proses enkripsi tersebut diilustrasikan oleh Gambar 2 sebagai berikut.




**Gambar 2.** Skema Enkripsi Menggunakan Skema (*t*, *w*)

### 2.2 *Pseudorandom Bit Generator Hitzl-Zele*

Fungsi Hitzl-Zele digunakan untuk membangun *stego-key* pada proses *Embedding LSB* dan *Extracting LSB*, yaitu barisan bilangan acak untuk menentukan lokasi bit disimpan pada data audio. Langkah-langkah membangun *stego-key* tersebut dijelaskan sebagai berikut:

1. Pada iterasi pertama, nilai awal $x(0), y(0), z(0)$ dikonstruksi menggunakan bilangan prima *p*, lalu tentukan berapa panjang rangkaian bit (*L*) yang akan dibangun.
2. Lakukan inisialisasi fungsi Hitzl-Zele sebanyak *L*, yaitu sampai diperoleh $x(L-1), y(L-1), z(L-1)$. Fungsi Hitzl-Zele adalah sistem dinamik dimensi tiga yang bergantung kepada waktu diskrit $i$ diberikan sebagai berikut:

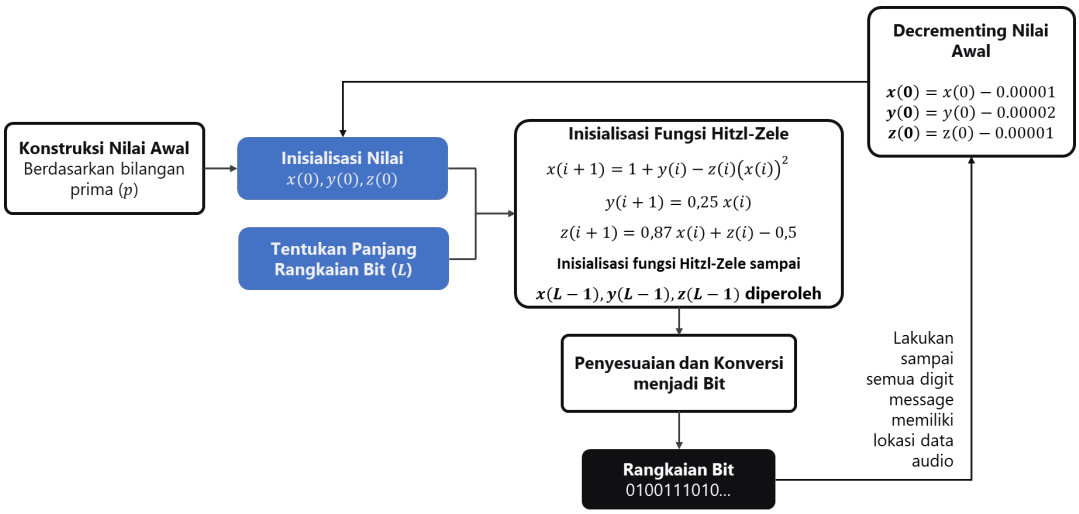
$$x(i+1) = 1 + y(i) - z(i)\big(x(i)\big)^2;$$

$$y(i+1) = 0{,}25\, x(i); \text{ dan}$$

$$z(i+1) = 0{,}87\, x(i) + z(i) - 0{,}5.$$

3. Penyesuaian dan konversi dilakukan untuk setiap titik yaitu $x(i), y(i), z(i)$, untuk $i = 0, \dots, L-1$ supaya diperoleh sebuah rangkaian bit dengan panjang $L$.
4. Konversi rangkaian bit tersebut ke dalam bilangan bulat, dilakukan pengecekan supaya bilangan yang dibangun unik.
5. Untuk iterasi kedua dan seterusnya, dilakukan pengubahan nilai awal supaya rangkaian bit yang akan dibangun selanjutnya berbeda.
6. Lakukan iterasi mulai dari langkah 2 dengan nilai awal yang baru. Iterasi dihentikan sampai diperoleh bilangan acak dengan jumlah yang diinginkan.

Langkah-langkah tersebut dapat diilustrasikan oleh Gambar 3 sebagai berikut.




**Gambar 3.** Skema *Pseudorandom Bit Generator* Hitzl-Zele

### 2.3 *Embedding LSB*

Pada proses *embedding*, *Dealer* menyematkan *share* ke dalam file audio WAV sehingga diperoleh *stego-audio* melalui proses sebagai berikut:

1. Penyesuaian dilakukan pada *w* buah *share,* masing-masing *share* diubah menjadi sebuah *string* dengan penambahan karakter "," sebagai berikut: $(x_k, A(x_k)) \rightarrow "x_k, A(x_k)" \rightarrow "x_k, A(x_k),"$ di mana $k = 1, \dots, w$. Penambahan "," berguna sebagai kriteria terminasi pada proses *extracting*. Kemudian *string* tersebut dikonversi berdasarkan *ASCII* menjadi *bitstring*, yang disebut sebagai *bitshare*. Konversi menjadi *bitshare* tersebut dilakukan supaya *share* bisa disematkan pada data audio.
2. *Dealer* memilih *w* buah file audio WAV, kemudian masing-masing file audio WAV dikonversi menjadi data audio.
3. *Stego-key* dibangun menggunakan *pseudorandom bit generator* Hitzl-Zele dengan nilai awalnya dikonstruksi berdasarkan *p*, panjang rangkaian bit adalah sepanjang data audio, dan banyaknya bilangan acak yang dibangun adalah sebanyak *bitshare*.
4. Untuk masing-masing *bitshare*, lakukan *embedding* ke masing-masing data audio dengan cara mengganti bit terakhir dari rangkaian bit data audio oleh bit dari *bitshare*.
5. Setelah semua *bitshare* berhasil di-*embedding*, dilakukan konversi pada masing-masing data audio yang tersematkan menjadi file audio WAV. Dengan demikian, diperoleh *w* buah *stego-audio*.

### 2.4 *Extracting LSB*

Pada proses *extracting*, *participant* mengekstraksi *bitshare* dari *stego-audio* dengan langkah-langkah sebagai berikut.

1. *Participant* menentukan masukan yaitu bilangan prima (*p*) dan *n* buah file audio WAV, selanjutnya masing-masing file audio WAV dikonversi menjadi data audio.

2. *Stego-key* dibangun menggunakan *pseudorandom bit generator* Hitzl-Zele dengan nilai awalnya dikonstruksi berdasarkan *p*, panjang rangkaian bit adalah sepanjang data audio, dan banyaknya bilangan acak yang dibangun adalah sebanyak 3000.
3. Untuk masing-masing data audio, lakukan proses *extracting* yaitu mengambil bit terakhir dari rangkaian bit pada data audio. Pemilihan lokasi rangkaian bit dilakukan berdasarkan *stego-key*.
4. Setiap 8-bit diperoleh, konversi menjadi *string* pada *ASCII* dan cek apakah karakter "," ditemukan sebanyak dua kali. Jika ditemukan dua kali, maka hentikan proses dan lanjutkan ke langkah 5. Jika tidak, lanjutkan proses *extracting*. Proses *extracting* akan berhasil jika masukan *p* adalah kunci yang sesuai sehingga diperoleh *n* buah *bitshare*.
5. Konversi masing-masing $n$ buah *bitshare* berdasarkan *ASCII* menjadi *string*, yaitu "$x_1, A(x_1),$", … , "$x_n, A(x_n),$". Kemudian lakukan penyesuaian dengan mengubah $n$ buah *string* tersebut menjadi $n$ buah *share* dengan menghilangkan karakter "," terakhir sebagai berikut: "$x_k, A(x_k),$" → "$x_k, A(x_k)$" → $\left(x_k, A(x_k)\right)$ di mana $k = 1, \dots, n$.

### 2.5 Dekripsi Polinomial Lagrange

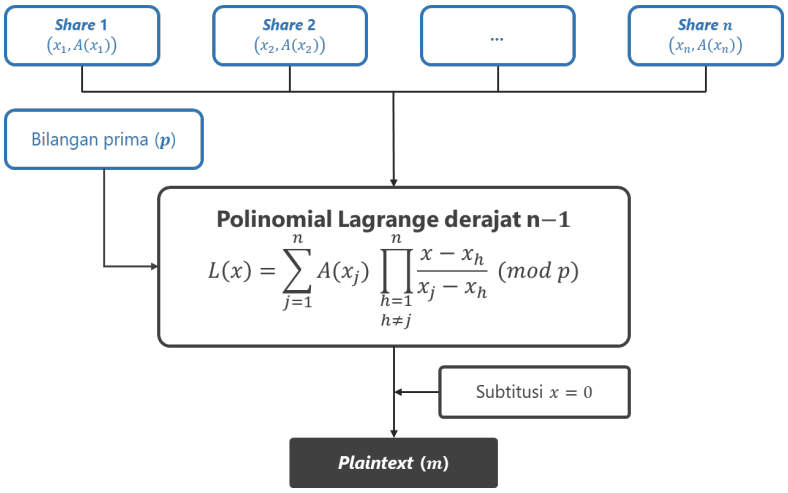
Pada proses dekripsi, *participant* mengonstruksi kembali *plaintext* menggunakan *share* dengan langkah-langkah sebagai berikut:

1. Masukan $p$ dan $n$ buah *share* digunakan untuk membangun polinomial $L(x)$, yaitu polinomial Lagrange derajat $n-1$ dalam modulus $p$ yang diberikan sebagai berikut:

$$L(x) = \sum_{j=1}^{n} A(x_j) \prod_{\substack{h=1 \\ h \neq j}}^{n} \frac{x - x_h}{x_j - x_h} \ (mod\ p).$$

2. Substitusikan $x = 0$ ke dalam polinomial $L(x)$ sehingga diperoleh *plaintext* $(m)$.

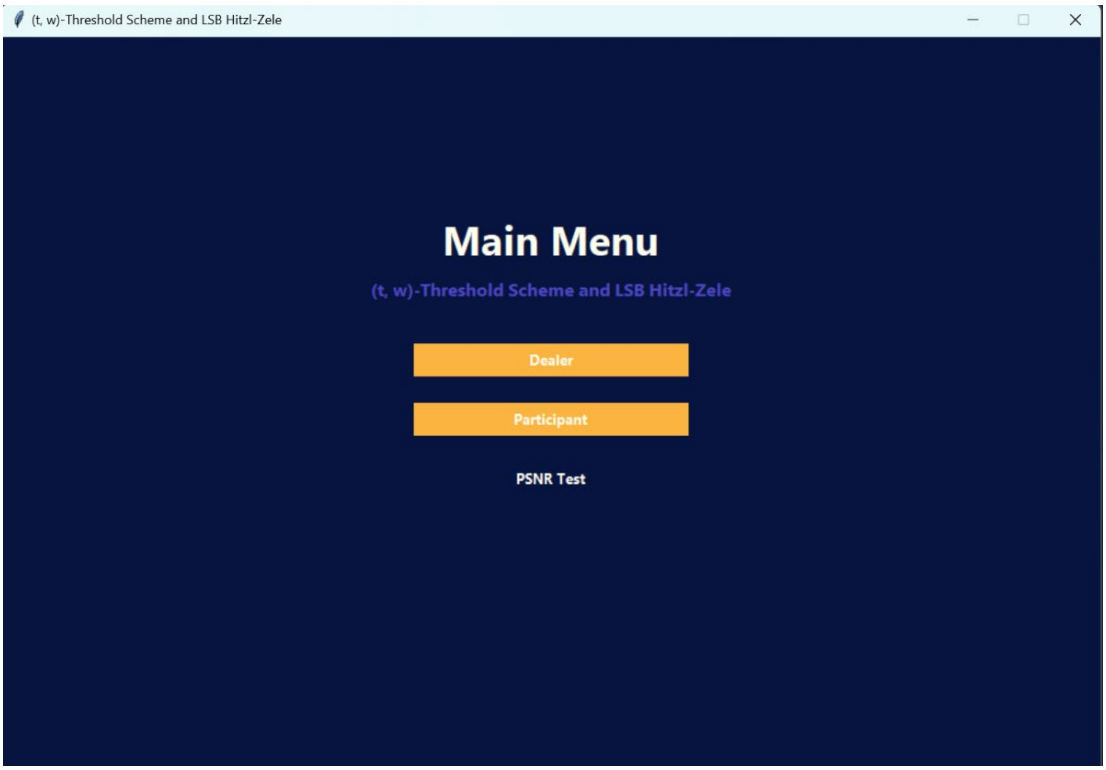
Langkah-langkah *participant* dalam proses dekripsi tersebut diilustrasikan oleh Gambar 4 sebagai berikut:




**Gambar 4.** Skema Dekripsi Menggunakan Polinomial Lagrange

## 3. HASIL DAN PEMBAHASAN

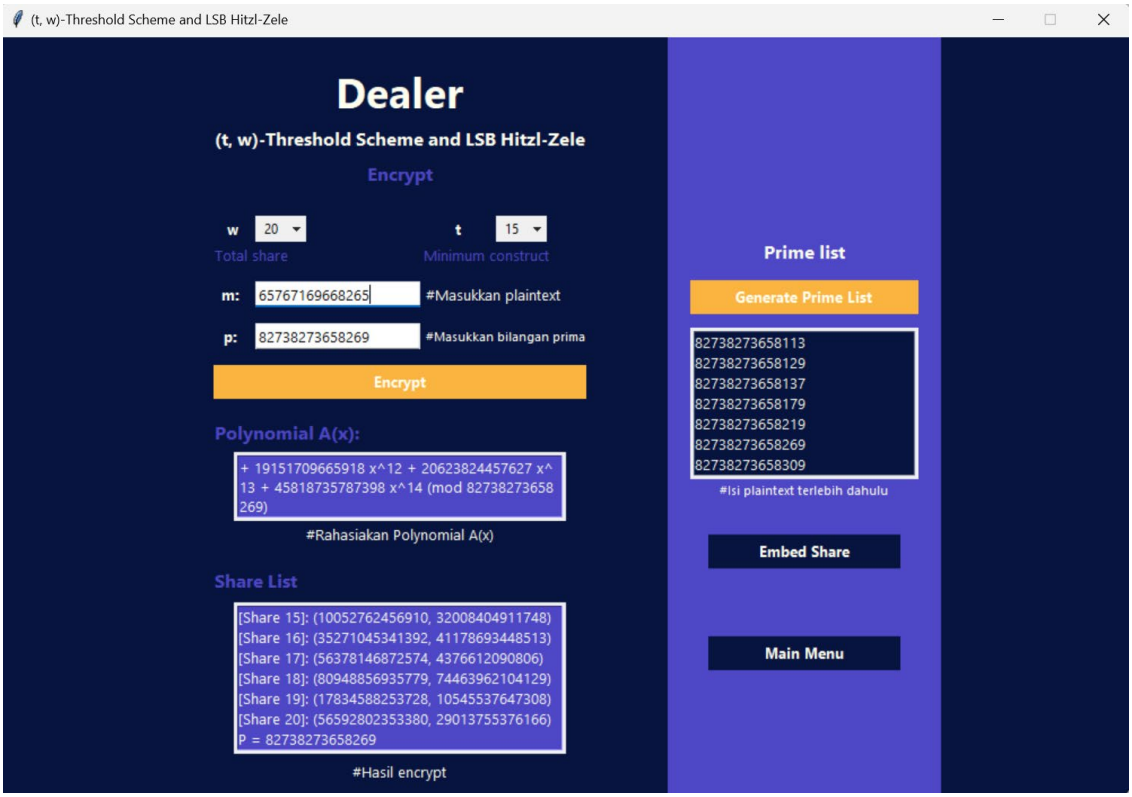
Untuk mempermudah pengamanan *plaintext* menggunakan penggabungan kriptografi skema (*t*, *w*) dan steganografi audio *LSB* menggunakan fungsi Hitzl-Zele, dikonstruksi program aplikasi yang diberi nama "*(t, w)-Threshold Scheme and LSB Hitzl-Zele*". Program aplikasi tersebut dikonstruksi menggunakan bahasa pemrograman Python 3.12. Terdapat 2 menu primer, yaitu Menu *Dealer* dan Menu *Participant*. Tampilan Menu Utama program aplikasi "*(t, w)-Threshold Scheme and LSB Hitzl-Zele*" ditampilkan pada Gambar 5.




**Gambar 5.** Tampilan Program Aplikasi pada Menu Utama

### 3.1 Menu *Dealer*

Setelah Menu *Dealer* dipilih, maka tampilan awal yang muncul adalah subbagian Enkripsi. Subbagian Enkripsi berfungsi untuk menjalankan proses enkripsi, yaitu membagi *plaintext* menjadi beberapa buah *share*. Tampilan subbagian Enkripsi dapat dilihat pada Gambar 6.



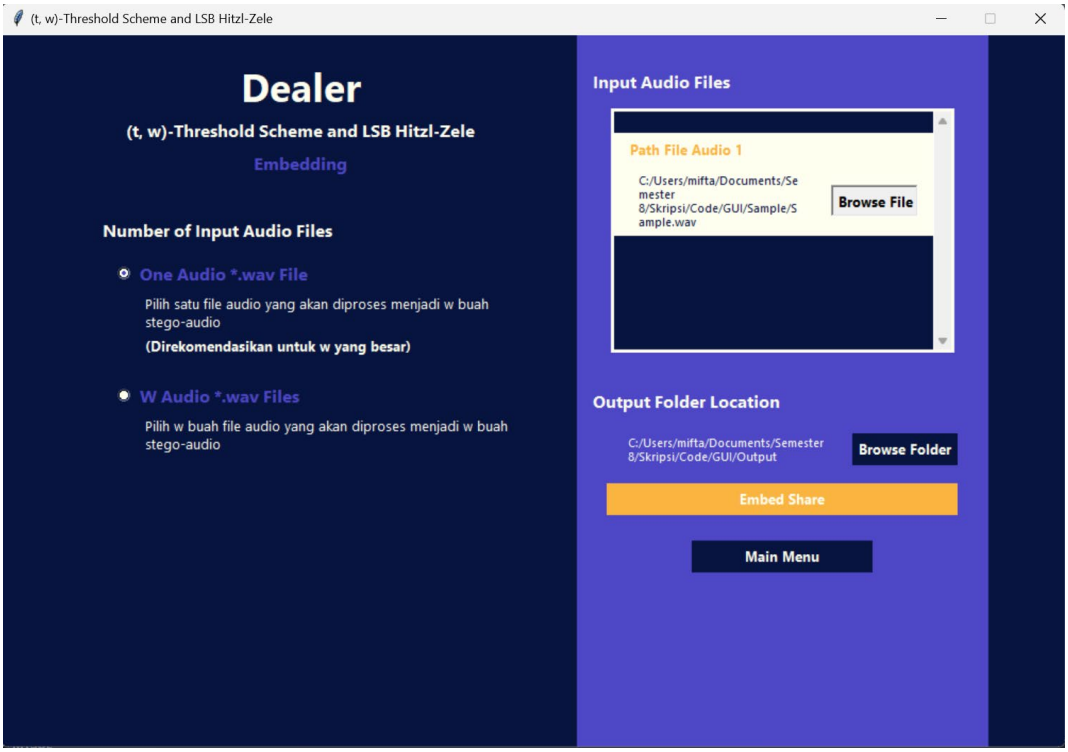

**Gambar 6.** Tampilan Program Aplikasi pada Menu *Dealer* subbagian Enkripsi

Dipilih skema (15, 20) dan pesan yang akan dienkripsi dan di-*embedding* adalah "ALGEBRA". Karena *plaintext* pada penelitian ini hanya menerima bilangan bulat nonnegatif, maka dilakukan konversi terhadap *plaintext* tersebut berdasarkan sistem *ASCII* untuk alfabet kapital supaya mudah dikonversi kembali. Diperoleh pesan "ALGEBRA" menjadi *plaintext* (*m*) yaitu 65767169668265 dan dipilih bilangan prima (*p*) yaitu 82738273658269. Setelah menekan tombol "*Encrypt*", diperoleh 20 buah *share* yang diuraikan pada Tabel 1.

**Tabel 1.** Keluaran *Share* Hasil Subbagian Enkripsi

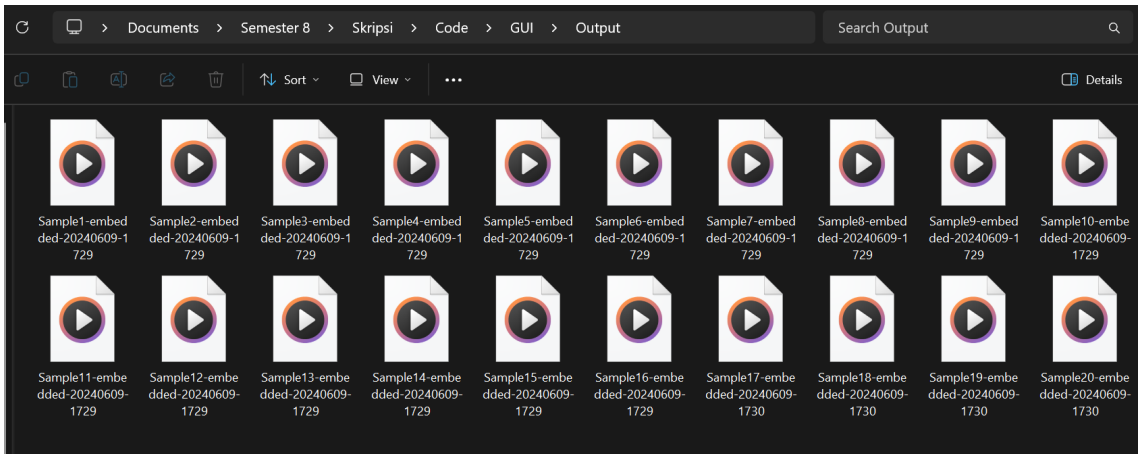
| $k$ | $x_k$ | $A(x_k)$ | $k$ | $x_k$ | $A(x_k)$ |
|---|---|---|---|---|---|
| **1** | 62384890714037 | 66641281265435 | **11** | 29209755993754 | 67717685484636 |
| **2** | 38435653508216 | 33200411338072 | **12** | 8084838395431 | 40307829596084 |
| **3** | 13254571387793 | 56392527237818 | **13** | 5464888974800 | 35525369303470 |
| **4** | 81939558018707 | 62942952795643 | **14** | 72358563820974 | 34780109396213 |
| **5** | 73809835413635 | 81255830419790 | **15** | 10052762456910 | 32008404911748 |
| **6** | 71418827394137 | 14057494702244 | **16** | 35271045341392 | 41178693448513 |
| **7** | 76888615219567 | 70661458426982 | **17** | 56378146872574 | 4376612090806 |
| **8** | 40101155252131 | 12502430722496 | **18** | 80948856935779 | 74463962104129 |
| **9** | 63494147416063 | 34796426238554 | **19** | 17834588253728 | 10545537647308 |
| **10** | 31060543197074 | 41926607642049 | **20** | 56592802353380 | 29013755376166 |

Untuk lanjut ke proses *embedding*, dapat menekan tombol "*Embed Share*" pada kolom sebelah kanan. Tampilan selanjutnya bagian *Dealer* adalah subbagian *Embedding*. Pada subbagian ini dilakukan proses *embedding*, yaitu penyematan *share* ke *file* audio WAV. Tampilan subbagian ini dapat dilihat pada Gambar 7.




**Gambar 7.** Tampilan Program Aplikasi pada Menu *Dealer* subbagian *Embedding*

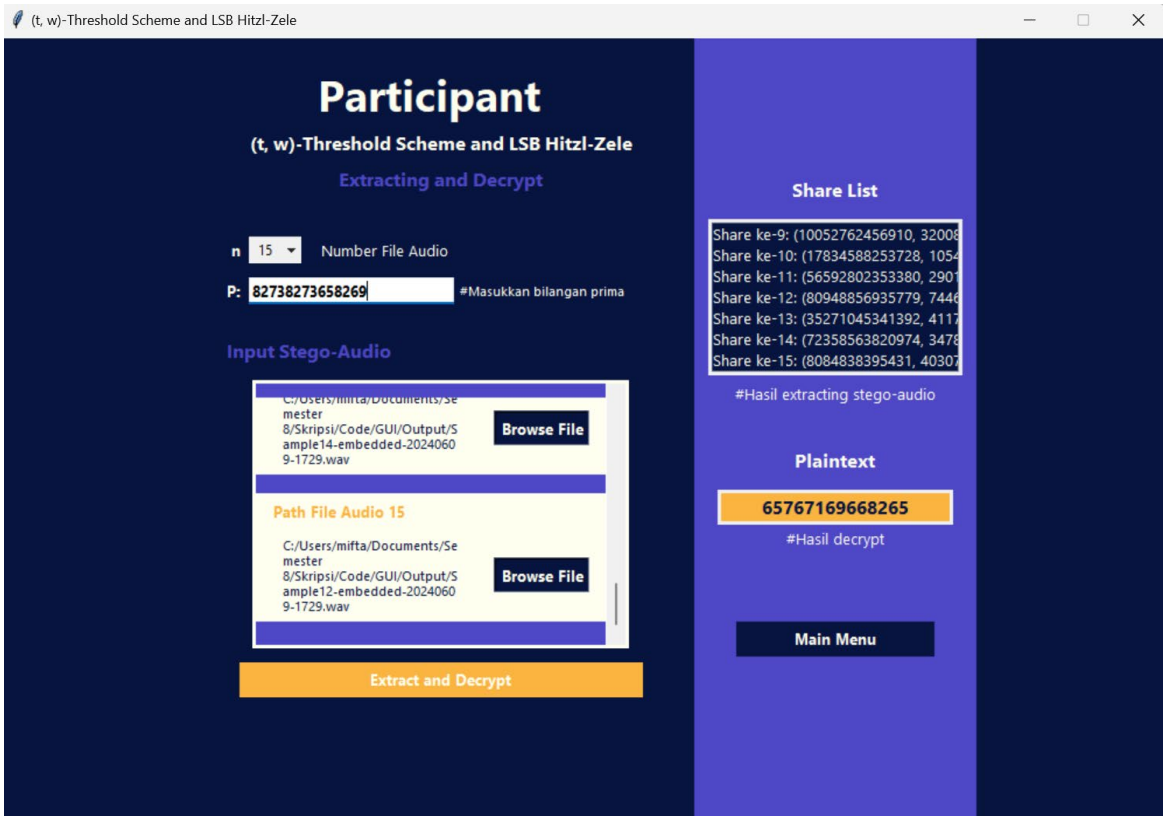
Proses *embedding* ini, dipilih opsi hanya menggunakan 1 buah *file* audio WAV, yaitu "Sample.wav". Setelah lokasi keluaran ditentukan dan tombol "*Embed Share*" ditekan, proses *embedding* dimulai. Keluaran proses tersebut adalah 20 buah *stego-audio*, yaitu *file* audio yang sudah disematkan *share* yang dapat dilihat pada Gambar 8.




**Gambar 8.** Keluaran *Stego-Audio* Hasil Subbagian *Embedding*

### 3.2 Menu *Participant*

Menu *Participant* digunakan untuk memperoleh kembali *plaintext* dari *stego-audio* melalui proses *extracting* dan dekripsi. Tampilan bagian ini dapat dilihat pada Gambar 9.




**Gambar 9.** Tampilan Program Aplikasi pada Menu *Participant*

Untuk memperoleh kembali *plaintext*, diperlukan paling sedikit masukan 15 *stego-audio* dan bilangan prima (*p*) pada proses sebelumnya. Masukan 15 *stego-audio* dipilih secara acak dari 20 *stego-audio* pada proses sebelumnya.

Dapat dilihat bahwa *plaintext* 65767169668265 dapat dikonstruksi kembali, jika setiap 2 digit *plaintext* tersebut dikonversi berdasarkan alfabet kapital dalam sistem *ASCII*, maka diperoleh *plaintext* "ALGEBRA".

### 3.3 Tes *PSNR*

Tes *PSNR* dilakukan pada keluaran *stego-audio* dari program aplikasi untuk mengevaluasi performa program aplikasi dalam menyembunyikan pesan. Perhitungan nilai *PSNR*, dinyatakan dalam desibel (dB) diberikan sebagai berikut:

$$PSNR = 10\left(\frac{R^2}{MSE}\right)$$

di mana $R$ adalah nilai maksimum yang dapat dicapai sinyal (*peak signal*), dihitung dengan $R = 2^b - 1$, di mana $b$ adalah *bit depth* sinyal dan *MSE* adalah *Mean Squared Error*, yaitu parameter galat untuk mengukur derau pada sinyal (*noise*), diberikan sebagai berikut:

$$MSE = \frac{1}{M}\sum_{i=0}^{M-1} (x_i - y_i)^2$$

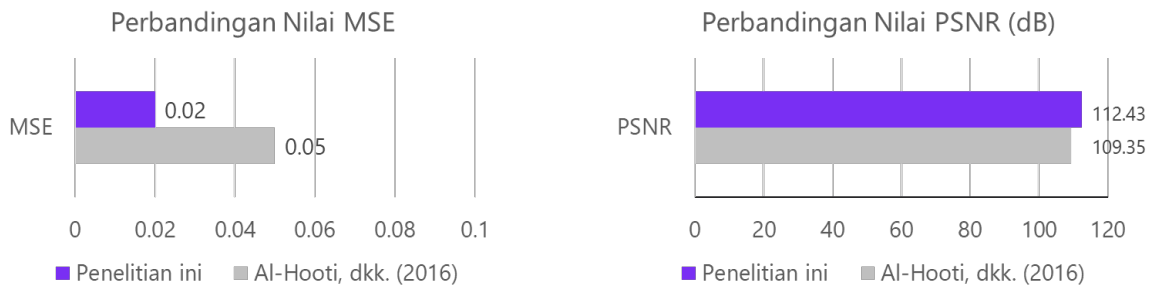
di mana $M$ adalah panjang data audio, $x_i$ adalah data *cover-audio* ke-$i$, dan $y_i$ adalah data *stego-audio* ke-$i$. Semakin kecil nilai *MSE*, maka semakin besar nilai *PSNR,* sehingga semakin besar nilai *PSNR*, maka semakin baik kualitas audio.

Penelitian terkait yang digunakan sebagai perbandingan adalah penelitian yang dilakukan Al-Hooti, dkk. (2016). Penelitian tersebut dipilih karena menggunakan metode *LSB* pada file audio dengan *bit depth* 16-bit dan mengevaluasi kualitas hasil steganografi menggunakan nilai *PSNR*, sehingga relevan untuk dijadikan acuan perbandingan dengan hasil penelitian ini. Gambar 10 menunjukkan perbandingan tersebut. Perbandingan tersebut berdasarkan nilai *PSNR* dari persentase panjang data bit yang disematkan dan panjang data audio yang sama, yaitu 10% dan pada file audio dengan *bit depth* yang sama, yaitu 16-bit yang diuraikan pada Tabel 2.

**Tabel 2.** Uraian Properti Data Audio untuk Perbandingan

| Penelitian | Panjang *Bitshare* | Panjang Data Audio | Persentase Panjang *Bitshare* dan Panjang Data Audio | *Bit Depth* |
|---|---|---|---|---|
| **Al-Hooti, dkk. (2016)** | 15.717 | 157.176 | 10% | 16-bit |
| **Penelitian ini** | 2.400 | 23.493 | 10% | 16-bit |

Berdasarkan Gambar 10, nilai *PSNR* yang diperoleh dari penelitian ini adalah 112,43 dB. Nilai *PSNR* tersebut sedikit lebih tinggi dibandingkan dengan nilai *PSNR* yang diperoleh Al-Hooti, dkk. (2016), yaitu 109,35 dB. Dengan demikian, performa program aplikasi dapat dikatakan lebih baik dalam menyisipkan pesan dibandingkan dengan penelitian tersebut.




**Gambar 10.** Hasil Perbandingan Nilai *MSE* dan *PSNR* dengan penelitian terkait

## 4. KESIMPULAN DAN SARAN

Berdasarkan penelitian yang dilakukan, skema pengamanan *plaintext* dengan penggabungan kriptografi skema $(t, w)$ dan steganografi *LSB* menggunakan fungsi Hitzl-Zele pada *file* audio WAV berhasil dirancang. Terdapat 2 pihak yang terlibat, yaitu *Dealer* dan *Participant*. *Dealer* adalah pihak pengirim, meliputi proses enkripsi dan *embedding*. *Participant* adalah pihak penerima, meliputi proses *extracting* dan dekripsi. Skema pengamanan *plaintext* tersebut diimplementasikan ke dalam program aplikasi yang diberi nama "*(t, w)-Threshold Scheme and LSB Hitzl-Zele*" yang dikonstruksi menggunakan bahasa pemrograman Python 3.12. Program aplikasi tersebut berisi 2 menu primer, yaitu Menu *Dealer* untuk enkripsi dan *embedding* dan Menu *Participant* untuk *extracting* dan dekripsi. Hasil tes PSNR menunjukkan bahwa program aplikasi memperlihatkan performa yang lebih baik dalam menyisipkan pesan berdasarkan nilai PSNR dibandingkan dengan penelitian Al-Hooti, dkk. (2016).

Untuk penelitian yang akan datang, terdapat saran untuk dikembangkan dan diperbaiki, di antaranya:

1. Menemukan dan mengembangkan metode supaya kriptografi skema $(t, w)$ dan steganografi *LSB* dengan menggunakan fungsi Hitzl-Zele dapat memproses *plaintext* yang berbentuk media lain, misalnya teks, gambar, audio, atau video.
2. Meneliti penggunaan fungsi acak (*chaotic map*) lain untuk melihat perbandingan efektivitas dan keamanan dalam penerapan kriptografi skema $(t, w)$ dan steganografi *LSB*.
3. Melakukan kriptanalisis untuk menguji kekuatan dan kelemahan dari implementasi kriptografi skema $(t, w)$ dan steganografi *LSB* menggunakan fungsi Hitzl-Zele. Tujuan dari kriptanalisis ini adalah untuk mengidentifikasi potensi serangan dan mencari cara untuk meningkatkan keamanan algoritma yang digunakan.

## 5. DAFTAR PUSTAKA

Al-Hooti, M. H. A., Djanali, S., & Ahmad, T. (2016). Audio data hiding based on sample value modification using modulus function. *Journal of Information Processing Systems*, *12*(3), 525-537.

Amin, M. M. (2016). Implementasi kriptografi klasik pada komunikasi berbasis teks. *Pseudocode*, *3*(2), 129-136.

Anti, U. A., Kridalaksana, A. H., & Khairina, D. M. (2017). Steganografi pada video menggunakan metode Least Significant Bit (LSB) dan End of File (EoF). *Jurnal Informatika Mulawarman*, *12*(2), 104-111

Deshpande, R. G., Ragha, L. L., & Sharma, S. K. (2018). Video quality assessment through PSNR estimation for different compression standards. *Indonesian Journal of Electrical Engineering and Computer Science*, *11*(3), 918-924.

Dutta, H., Das, R. K., Nandi, S. & Prasanna, S.R. M. (2019). An overview of digital audio steganography. *IETE Technical Review*. *37*(6), 632-650.

Herlinawati, H. (2016). Steganografi video H263 dengan metode Discrete Cosine Transform. Electrician: *Jurnal Rekayasa dan Teknologi Elektro*, *10*(1), 11-20.

Humaira, A. F., Marwati, R., & Yulianti, K. (2023). Implementasi kriptografi secret sharing scheme dan steganografi audio Least Significant Bit (LSB). *JMT: Jurnal Matematika dan Terapan*, *5*(1), 1-11.

Kordov, K. M., & Stoyanov, B. (2017). Least Significant Bit Steganography using Hitzl-Zele Chaotic. *International Journal of Electronics and Telecommunications*, *63*(4); 417-422.

Naik, R. B., & Singh, U. (2022). A Review on applications of chaotic maps in Pseudo-Random Number Generators and encryption. *Annals of Data Science*, *11*(1), 1-26.

Shamir, A. (1979). How to share a secret. *Communications of the ACM*, *22*(11), 612-61.