Steganografi Teks Menggunakan Metode Pencocokan LSB dan Karakter *Non-Breaking Space* Sebagai Penanda Pesan

(Steganography using LSB Matching and Non-breaking Space Character as Message Marker)

Ulil Amri, I Gede Pasek Suta Wijaya, Fitri Bimantoro
Jurusan Teknik Informatika, Fakultas Teknik, Universitas Mataram
Jl. Majapahit 62, Mataram, Lombok NTB, INDONESIA
Email: ulilmail24@gmail.com, gpsutawijya@unram.ac.id, bimo@unram.ac.id

***Abstract* - Steganography is one of methods used to secure an information. This study aimed to develop a steganography method applied in text media and evaluate the method used in message insertion. In this study, technique used to insert secret message into text media was LSB (Least Significant Bit) matching with non-breaking space character as message marker. To evaluate quality of steganographic, Peak Signal to Noise Ratio (PSNR) test, Jaro-Winkler distance score, questionnaire, and calculation of capacity ratio were done. The results give the PSNR value by a small margin, the Jaro-Winkler distance score with an average score of more than 0.8, the Mean Opinion Score (MOS) value of 2.36 and 1.79 for each test parameter, and the capacity ratio is less than 1%. Therefore, this method has a good ability to hide the message but has a relatively small capacity.**

Kata Kunci: Steganografi Teks, Pencocokan LSB, *non-breaking space*.

I. PENDAHULUAN

Steganografi adalah seni penyembunyian pesan ke dalam pesan lainnya yang telah ada sejak sebelum masehi [1]. Kini di tengah perkembangan serta kemajuan teknologi jaringan, steganografi banyak dimanfaatkan untuk mengirim pesan melalui jaringan *Internet* tanpa diketahui orang lain.

Steganografi dapat diklasifikasikan ke dalam steganografi gambar, audio, video, dan teks tergantung pada media *cover* yang digunakan untuk menyisipkan data rahasia. Dari beberapa jenis steganografi tersebut, steganografi teks dipercaya sebagai salah satu steganografi yang paling sulit digunakan. Hal ini dikarenakan kurangnya redundansi informasi seperti yang ada pada file gambar, audio, atau video. Struktur pada sebuah dokumen teks bersifat identik dengan apa yang terlihat, sedangkan pada jenis dokumen lainnya seperti pada gambar, struktur dokumennya berbeda dengan apa yang terlihat. Karena itulah pada jenis dokumen lainnya kita dapat menyembunyikan informasi dengan melakukan perubahan pada struktur dokumen tanpa membuat perubahan yang terlihat jelas pada hasil keluaran. Perubahan yang tak terlihat dapat dilakukan pada sebuah file gambar atau audio, tetapi pada file teks, bahkan sebuah huruf atau tanda baca tambahan dapat disadari dengan mudah oleh pembaca. Tetapi proses *sorting* pada dokumen teks hanya membutuhkan sedikit memori dan lebih cepat yang membuat file teks lebih baik jika dibandingkan dengan metode lainnya [2].

Atas dasar uraian di atas, maka paper ini akan membahas mengenai bagaimana mengamankan pesan teks dengan cara menyisipkan file pesan ke dalam file teks dengan format (.txt atau .docx) dengan metode pencocokan LSB (*Least Significant Bit*) dan penanda pesan. Penggunaan metode pencocokan LSB dianggap mampu mengatasi permasalahan pada file *cover* yang berupa teks. Hal ini karena pada pencocokan LSB nilai LSB tidak diubah tetapi hanya dicocokkan dengan bit pesan.

II. TINJAUAN PUSTAKA

Cara penyisipan pesan pada media teks dilakukan dengan beberapa teknik, yaitu teknik LSB, teknik manipulasi elemen teks, dan teknik *Natural Language Processing* (NLP). Teknik LSB ini menyisipkan pesan dengan cara mengganti bit-bit kurang berarti (*Least signifikan bit*) dari bit *cover* dengan bit pesan. Hasil stego-teks dari teknik ini akan sangat mencurigakan karena perubahan bit karakter pada berkas teks sedikit saja akan mengakibatkan perubahan yang cukup besar [3].

Perubahan format teks atau karakteristik dari sebuah karakter dapat menghasilkan penanda (*watermarking*) pada dokumen. Tujuan utamanya adalah membuat perubahan pada teks sedemikian rupa agar perubahan yang dilakukan tidak terlihat tetapi sandi yang ada dapat diuraikan. Perubahan dapat diterapkan pada dokumen yang direpresentasikan dalam bentuk gambar atau dalam file format dokumen [4].

Pencocokan bit pesan pada bit citra dilakukan dengan metode *divide and conquer* yang terdiri dari 3 proses yaitu *divide, conquer* dan *combine*. Artinya memecah masalah

menjadi beberapa bagian kecil (*divide*), kemudian secara rekursif menyelesaikan setiap masalah-masalah kecil tersebut (*conquer*). Selanjutnya solusi dari setiap masalah kecil tersebut hasilnya digabung menjadi satu solusi utama (*combine*)[5].

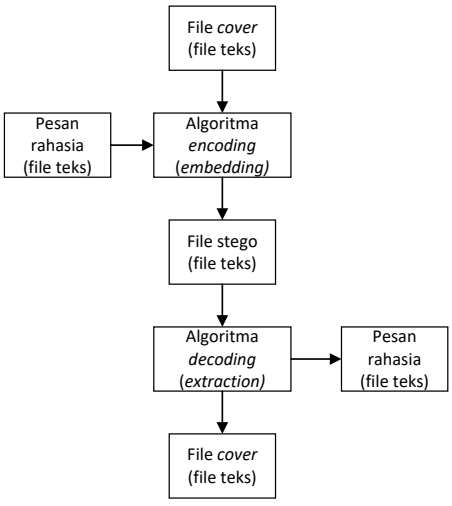
LSB (*Least Significant Bit*) *replacement* menanamkan sebuah pesan ke dalam *cover image* dengan mengganti nilai LSB dari *cover image* dengan nilai bit pesan dan menghasilkan stego *image*. Metode ini menambahkan 1 pada nilai *pixel* genap dan mengurangi 1 pada nilai *pixel* ganjil atau tidak melakukan perubahan nilai.

LSB *matching* juga mengubah nilai LSB dari *cover image* dalam menyisipkan pesan. LSB *matching* tidak langsung mengubah nilai LSB dari *cover image* seperti yang dilakukan pada LSB replacement. Di mana jika bit pesan tidak cocok dengan LSB dari *cover image*, lalu nilai 1 secara acak akan ditambahkan atau dikurangi dari nilai *pixel cover*. Nilai *pixel* tidak pernah diubah menjadi nilai di luar batas yang diperbolehkan [6]

Steganografi pada media teks dengan menggunakan metode LSB *replacement* dianggap sebagai metode steganografi yang buruk karena perubahan yang terjadi pada file stego akan sangat terlihat dan mencurigakan. Pada penelitian ini akan mencoba penyisipan pesan pada media teks dengan menggunakan metode LSB *matching* dan penanda pesan. Berdasarkan teori dan cara kerjanya, metode ini memungkinkan pesan disisipkan tanpa adanya perubahan mencolok pada file stego.

III. METODE USULAN

Bagian ini akan menjelaskan mengenai metode steganografi yang diusulkan. Metode steganografi pada paper ini adalah pencocokan LSB dan penggunaan karakter *non-breaking space* sebagai penanda pesan. Secara garis besar, tekniknya terbagi menjadi dua proses utama yaitu penyisipan (*embedding*) pesan dan ekstraksi (*extraction*) pesan yang tersembunyi. Hubungan antara kedua proses tersebut dapat dilihat pada Gambar 1.



Gambar 1 Diagram penyisipan dan ekstraksi pesan.

Proses penyisipan (*embedding*) pesan dimulai dengan memilih file teks yang akan digunakan sebagai *cover object* yang akan disisipkan pesan. Kemudian langkah selanjutnya adalah menuliskan pesan yang akan disisipkan ke dalam *cover object* dengan algoritma *embedding* yang digunakan. Sedangkan pada proses ekstraksi (*extraction*) pesan dimulai dengan memilih file stego (*stego object*). Kemudian jalankan proses ekstraksi pesan dengan menggunakan algoritma *extraction*. Dari proses ekstraksi tersebut diperoleh pesan rahasia yang telah disisipkan.

Berikut merupakan algoritma yang akan menjelaskan proses penyisipan pesan hingga menghasilkan *stego object* dan proses ekstraksi pesan rahasia yang telah disisipkan pada *stego object.*

A. Algoritma Penyisipan Pesan

Proses penyisipan pesan pada file dokumen dengan format TXT dan DOCX, dilakukan dengan menerapkan algoritma berikut.

- *Input*: teks pesan dan file *cover* berjenis (DOCX atau TXT).
- *Output*: file stego.
- Proses:

1. Masukkan teks pesan dan file *cover.*
2. Ubah teks pesan ke dalam bentuk biner dan tampung ke dalam sebuah variabel *array* satu dimensi.
3. Tampung file *cover* ke dalam variabel *array* satu dimensi.
4. Deklarasikan variabel boolean cek = false, int i = 0, int jumlahBit = 0.
5. If i < panjang *cover then* pindai karakter[i] pada file *cover.*
   1) *If* karakter[i] == 32 (spasi) *then*
      a) *If* cek == *false then* periksa
         - *If* i+1 < panjang *cover* && jumlahBit < panjang biner pesan *then if* karakter [i+1] < 255 *then if* karakter [i+1] % 2 == nilai pesan [jumlahBit] *then* karakter [i] = 160, cek = tua, jumlahBit++.

      b) *Else* cek = *false.*
   2) *Else if karakter*[i] == 160 *(non-breaking space) then* karakter [i] = 32.
   3) i++.
6. Ulangi langkah 5 sampai akhir file *cover*
7. *Else*
   1) If panjang pesan != jumlahBit *then* tampilkan "pesan tidak tersisipkan seluruhnya"

2) *Else* tampilkan "pesan tersisipkan seluruhnya"

8. Simpan *keluaran*.

Berdasarkan algoritma penyisipan pesan tersebut dapat bibuat sebuah ilustrasi, sebagai berikut.

Misalkan terdapat sebuah pesan dengan nilai biner 0, yang akan disisipkan pada file *cover* dengan nilai biner seperti yang terdapat pada Gambar 2.

```
01100001 00100000 01100010
```

Gambar 2 Nilai biner dari teks *cover*

Selanjutnya pada file cover tersebut dilakukan pemindaian karakter spasi dengan nilai desimal 32, setelah menemukan karakter spasi maka selanjutnya dilakukan pencocokan nilai LSB dari karakter setelah karakter spasi dengan bit pesan. Proses ini dapat dilihat pada Gambar 3, di mana nilai biner yang digaris bawahi merupakan karakter spasi dan nilai LSB karakter setelahnya.

```
01100001 00100000 0110001̲0̲
```

Gambar 3 Pemindaian karakter spasi dan nilai LSB karakter setelahnya.

Pada Gambar 3 dapat dilihat bahwa nilai LSB dari karakter setelah spasi dengan bit pesan yang akan di sisipkan sama. Sehingga karakter spasi di ubah menjadi *non-breaking space*, seperti pada Gambar 4.

```
01100001 10100000 0110001̲0̲
```

Gambar 4 Hasil penyisipan bit pesan dan perubahan karakter spasi.

Gambar 4 menunjukan perubahan nilai biner dari karakter spasi dengan nilai 32 menjadi karakter *non-breaking space* dengan nilai 160.

B. Algoritma Ekstraksi Pesan

Proses ekstraksi pesan pada file dokumen dengan format TXT dan DOCX, dilakukan dengan menerapkan algoritma berikut.Input: file cover berjenis (DOCX atau TXT).

- *Output*: file stego.
- Proses:
  1. Masukkan file stego *text*.
  2. Tampung file stego *text* ke dalam variabel *array* satu dimensi.
  3. Deklarasikan variabel *String* hasil="", int i = 0.
  4. If i < panjang stego *text then* pindai karakter[i] pada file stego *text*.
     1) *If* karakter[i] == 160 *(non-breaking space)*.
        a. *If* i+1 < panjang stego *text then*
           - int temp = karakter [i+1] % 2
           - hasil += ""+temp
           - karakter [i] = 32.
        2) i++.
  5. Ulangi langkah 4 sampai akhir file stego *text*.
  6. Rangkai hasil menjadi rangkaian per 8 bit.
  7. Konversi rangkaian 8 bit menjadi karakter
  8. Simpan keluaran berupa pesan rahasia dan file *cover*.

Berdasarkan algoritma ekstraksi pesan tersebut dapat bibuat sebuah ilustrasi, sebagai berikut.

Misalkan terdapat file stego dengan nilai biner seperti pada Gambar 5. Pada proses ekstraksi pesan, hal pertama yang dilakukan adalah memindai karakter *non-breaking space*.

```
01100001 10100000 01100010
```

Gambar 5 Pemindaian karakter *non-breaking space*

Pada Gambar 5, nilai biner yang berwarna merah merupakan karakter *non-breaking space*. Setelah proses menemukan karakter *non-breaking space*, maka dilakukan pengecekan dan pengambilan nilai LSB dari karakter setelahnya.

```
01100001 10100000 0110001̲0̲
```

Gambar 6 Pengecekan nilai LSB karakter setelah karakter *non-breaking space*

Gambar 6 menunjukan proses pengecekan dan pengambilan bit pesan yang disisipkan pada LSB karakter setelah karakter *non-breaking space*. Nilai LSB tersebut selanjutnya akan dirangkai menjadi pesan rahasia yang disisipkan.

Untuk mengetahui kualitas dari metode steganografi usulan, maka dilakukan pengujian untuk mengukur beberapa parameter yaitu PSNR, skor *Jaro-Winkler distance*, nilai rasio kapasitas, Mean Opinion Score (MOS).

C. PSNR pada File Teks

Kualitas file teks hasil dari proses steganografi dapat diukur dengan menggunakan metode PSNR. Pengujian tersebut dilakukan dengan menghitung nilai MSE menggunakan persamaan 1.

$$MSE = \frac{1}{M}\sum_{y=0}^{M}[I(y) - I'(y)]^2 \quad (1)$$

di mana :

- MSE = Nilai *Mean Square Error* citra steganografi
- M = Panjang teks stego
- I(y) = nilai desimal karakter dari teks *cover*
- I'(y) = nilai desimal karakter dari teks stego

Setelah diperoleh nilai MSE maka nilai PSNR dapat dihitung dari kuadrat nila karakter maksimum dibagi dengan nilai MSE. Secara matematis, nilai PSNR dirumuskan seperti pada persamaan 2.

$$PSNR = 10.\log\left(\frac{MAXi^2}{MSE}\right) \quad (2)$$

di mana:

- MSE = nilai perhitungan MSE
- MAXi = nilai maksimum dari karakter pada teks yang digunakan

Nilai maksimal (MAXi) dari karakter pada percobaan ini mengacu kepada nilai maksimal dari tabel ASCII 8-bit yang mengacu pada ISO 8859-1, yaitu 255. Semakin rendah nilai MSE maka akan semakin baik, dan semakin besar nilai PSNR maka semakin baik kualitas teks steganografi.

D. Algoritma Jaro-Winkler distance

Salah satu metode untuk mengetahui kualitas *stego object* dari metode steganografi yang menggunakan file *cover* berupa teks adalah algoritma Jaro-Winkler *distance.* Algoritma ini berfungsi untuk memeriksa tingkat kesamaan dari file *cover* dengan stego *text* yang dihasilkan. Algoritma ini akan menghasilkan skor dengan rentang nilai 0 sampai 1. Di mana nilai 0 menandakan tidak ada kesamaan, dan 1 menandakan sama persis.

Pada algoritma Jaro perhitungan jarak (d_j) antara dua *string* yaitu s_1 dan s_2 dapat diakukan dengan menggunakan persamaan 3.

$$d_j = \frac{1}{3} \times \left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) \quad (3)$$

di mana :

m = jumlah karakter yang sama persis
$|s_1|$ = panjang *string* 1
$|s_2|$ = panjang *string* 2
t = jumlah transposisi

Jarak teoritis dua buah karakter yang disamakan dapat dibenarkan jika memenuhi kriteria persamaan 4.

$$\left(\frac{\max(|s_1|,|s_2|)}{s}\right) < -1 \quad (4)$$

Jaro-Winkler *distance* menggunakan *prefix scale* (p) yang memberikan tingkat penilaian yang lebih, dan *prefix length* (l) yang menyatakan panjang awalan yaitu panjang karakter yang sama dari *string* yang dibandingkan sampai ditemukannya ketidaksamaan. Bila *string* s_1 dan s_2 yang diperbandingkan, maka nilai Jaro-Winkler *distance*-nya (d_w) dapat dihitung dengan persamaan 5.

$$d_w = d_j + \left(lp(1-d_j)\right) \quad (5)$$

di mana :

d_j = Jaro distance untuk *string*s s_1 dan s_2
l = panjang prefiks umum di awal *string* nilai maksimalnya 4 karakter (panjang karakter yang sama sebelum ditemukan perbedaan maksimal 4)
p = konstanta *scaling factor*. Nilai standar untuk konstanta ini menurut Winkler adalah p = 0,1.

***E.** Kapasitas*

Kapasitas atau daya tampung sebuah file *cover* merupakan salah satu aspek penting dalam pengembangan metode steganografi. Untuk mengetahui kapasitas sebuah file *cover* dapat dilakukan dengan menghitung rasio kapasitasnya. Rasio kapasitas sebuah file *cover* dapat di hitung dengan menggunakan persamaan 6.

$$\text{Rasio kapasitas} = \frac{(\text{jumlah byte yang disembunyikan})}{(\text{ukuran teks } \textit{cover} \text{ dalam byte})} \quad (6)$$

Jika diasumsikan satu karakter mewakili satu byte di dalam memori, kita dapat menghitung persentase kapasitas dengan cara persamaan 6 dikalikan 100.

IV. HASIL PENGUJIAN DAN PEMBAHASAN

Data yang digunakan untuk melakukan pengujian metode steganografi adalah 10 teks pesan dan 10 file *cover* berjenis TXT dan DOCX. Adapun daftar pesan yang digunakan dalam percobaan ini dapat dilihat pada Lampiran 1, sedangkan informasi mengenai file *cover* yang digunakan pada percobaan ini dapat dilihat pada Tabel I.

TABEL I. DAFTAR FILE *COVER* YANG DIGUNAKAN DALAM PROSES PENGUJIAN.

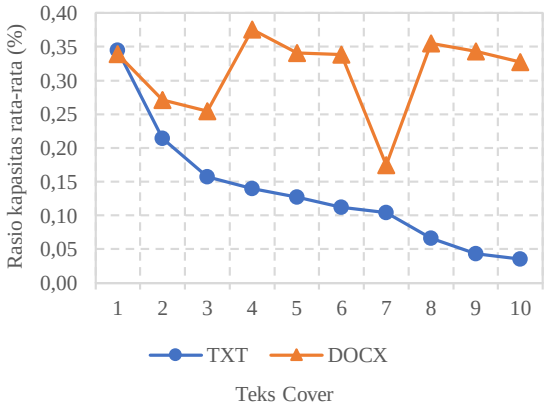
No.	File *Cover*	Jenis File	Ukuran File (KB)	Panjang Karakter
1	C1	TXT	27	26.629
2	C2	TXT	50	51.000
3	C3	TXT	89	90.144
4	C4	TXT	105	107.030
5	C5	TXT	117	118.831
6	C6	TXT	132	134.618
7	C7	TXT	142	144.518
8	C8	TXT	225	229.602
9	C9	TXT	343	350.372
10	C10	TXT	421	431.021
11	D1	DOCX	86	19.869
12	D 2	DOCX	93	45.640
13	D 3	DOCX	104	37.392
14	D 4	DOCX	105	17.939
15	D 5	DOCX	105	26.952
16	D 6	DOCX	107	24.729
17	D 7	DOCX	109	86.531
18	D 8	DOCX	144	16.725
19	D 9	DOCX	149	18.962
20	D 10	DOCX	152	29.463

A. Pengujian Kapasitas

Pengujian kapasitas bertujuan untuk mengetahui daya tampung sebuah file *cover*. Besarnya daya tampung sebuah file *cover* merupakan salah satu indikator baik atau

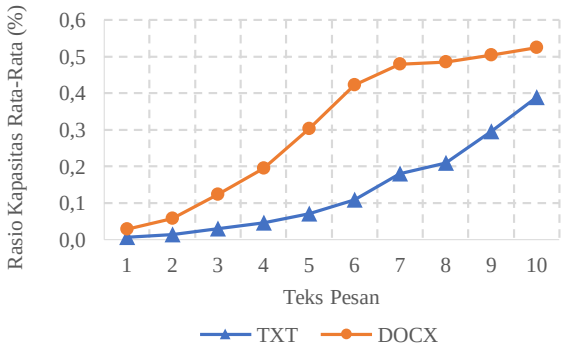
tidaknya mutu dari metode steganografi yang digunakan. Pengujian kapasitas dapat dilakukan dengan menghitung jumlah karakter yang berhasil disisipkan dan rasio kapasitas dari masing-masing *cover* file.

Sebelum melakukan perhitungan rasio kapasitas dari sebuah file *cover*. Perlu dihitung terlebih dahulu jumlah karakter yang berhasil disisipkan pada masing-masing file *cover*. Setelah mendapatkan hasil perhitungan jumlah karakter yang dapat disisipkan pada masing-masing file *cover*, dapat dilakukan perhitungan rasio kapasitas dengan menggunakan persamaan 6. Hasil perhitungan rata-rata nilai rasio kapasitas pada percobaan ini disajikan dalam bentuk grafik dan dapat dilihat pada Gambar 7.



Gambar 7 Grafik rasio kapasitas rata-rata pada setiap file cover.

Berdasarkan data grafik pada Gambar 7, dapat dilihat bahwa pada file berjenis TXT semakin besar ukuran file *cover* yang digunakan akan semakin kecil rasio kapasitas yang dihasilkan. Sedangkan pada file berjenis DOCX besarnya ukuran file yang digunakan tidak berpengaruh besar terhadap nilai rasio kapasitas dari file *cover* yang digunakan. Selain itu nilai rasio kapasitas yang dihasilkan pada file berjenis DOCX lebih bervariatif. Hal ini dikarenakan pada file berjenis DOCX tidak hanya berisi karakter, tetapi juga berisi gambar atau konten lainnya yang tidak digunakan dalam proses penyisipan, sehingga menyebabkan sedikitnya jumlah karakter pada file DOCX jika dibandingkan dengan file TXT. Sedikitnya karakter pada file *cover* mengakibatkan nilai rasio kapasitas yang dihasilkan cenderung lebih tinggi.



Gambar 8 Grafik rasio kapasitas rata-rata pada setiap teks pesan.

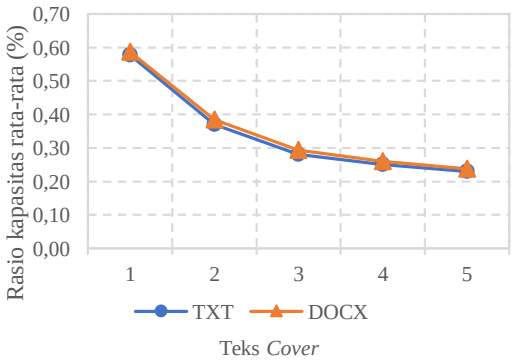
Grafik pada Gambar 8 menunjukkan bahwa rasio kapasitas dari sebuah file *cover* akan semakin besar jika selisih perbandingan antara jumlah karakter pesan yang disisipkan dengan jumlah karakter pada file *cover* semakin kecil. Berdasarkan grafik pada Gambar 8 dapat dilihat perbedaan antara nilai rasio kapasitas pada file berjenis DOCX lebih besar nilai rasio pada file berjenis TXT.

Selain menggunakan file *cover* pada Tabel I, pada pengujian ini juga dilakukan pengujian rasio kapasitas pada file *cover* berjenis TXT dan DOCX dengan isi file yang sama. Keterangan mengenai file *cover* yang akan digunakan dapat dilihat pada Tabel II.

TABEL II. DAFTAR FILE COVER DENGAN ISI SAMA PADA PROSES PENGUJIAN RASIO KAPASITAS.

No.	File *Cover*	Jenis File	Ukuran File (KB)	Panjang Karakter
1	C1	TXT	27	26.629
2	C2	TXT	50	51.000
3	C3	TXT	89	90.144
4	C4	TXT	105	107.030
5	C5	TXT	117	118.831
6	C1	DOCX	21	25.985
7	C2	DOCX	36	48.832
8	C3	DOCX	54	86.638
9	C4	DOCX	65	102.518
10	C5	DOCX	67	114.313

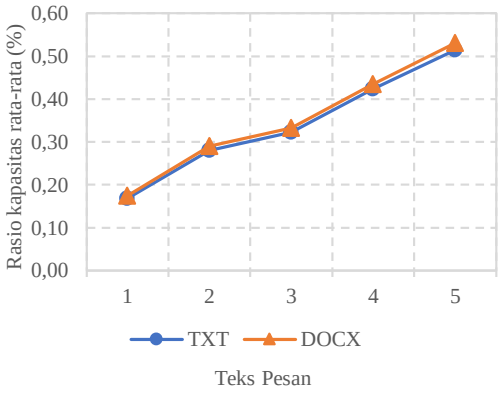
Sama seperti pengujian kapasitas rasio sebelumnya, perlu dilakukan perhitungan jumlah karakter yang berhasil disisipkan pada masing-masing file *cover*. Kemudian dilakukan perhitungan rasio kapasitas dengan menggunakan persamaan 6. Hasil perhitungan rata-rata nilai rasio kapasitas pada percobaan ini disajikan dalam bentuk grafik dan dapat dilihat pada Gambar 9.



Gambar 9 Grafik rasio kapasitas rata-rata pada setiap file cover.

Berdasarkan data grafik pada Gambar 9, dapat dilihat bahwa pada pengujian dengan menggunakan file cover berjenis TXT dan DOCX dengan isi sama, menghasilkan rata-rata nilai rasio yang hampir sama. Dalam grafik tersebut dapat dilihat bahwa file berjenis DOCX menghasilkan rata-rata rasio kapasitas lebih besar. Perbedaan tersebut dikarenakan adanya perbedaan jumlah karakter yang dihasilkan pada file berjenis DOCX dengan file berjenis TXT, di mana pada file DOCX jumlah karakter

yang terbaca lebih sedikit meskipun isi yang digunakan pada file berjenis TXT dan DOCX sama. Hal tersebut disebabkan oleh perbedaan proses *encoding* pada kedua jenis file.



Gambar 10 Grafik rasio kapasitas rata-rata pada setiap teks pesan.

Grafik pada Gambar 10 menunjukkan bahwa rasio kapasitas dari sebuah file *cover* akan semakin besar jika selisih perbandingan antara jumlah karakter pesan yang disisipkan dengan jumlah karakter pada file *cover* semakin kecil.

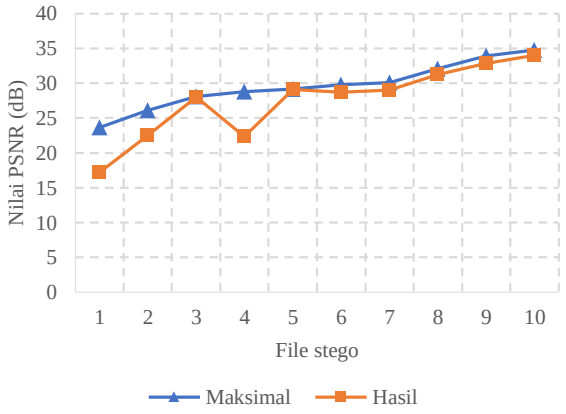
Secara keseluruhan rasio kapasitas yang diperoleh dengan menggunakan metode steganografi pada percobaan ini tergolong kecil, di mana nilai rasio kapasitas dari kedua jenis file *cover* tersebut kurang dari 1%. Hal ini dikarenakan metode penyisipan yang digunakan hanya dapat menyisipkan satu bit pesan pada satu karakter tertentu.

B. Pengujian Nilai PSNR

Pengujian nilai PSNR (*Peak Signal to Noise Ratio*) dilakukan untuk mengetahui kualitas file stego. Proses pengujian ini akan menghasilkan nilai MSE (*Mean Square Error*) dan nilai PSNR dari sebuah file stego. Semakin kecil nilai MSE yang dihasilkan maka semakin baik kualitas file stego. Sedangkan nilai PSNR yang baik adalah nilai yang besar dan mendekati nilai PSNR maksimal dari file stego tersebut.

File *cover* yang digunakan pada pengujian ini merupakan file yang diperoleh dari hasil proses ekstraksi pesan di mana setelah proses ekstraksi dilakukan akan menghasilkan dua *output*, yaitu file pesan dan file *cover*. Pengujian pada file *cover* ini bertujuan untuk memeriksa kesamaan file *cover* sebelum dan setelah digunakan dalam metode steganografi ini. Selain itu pengujian ini juga digunakan untuk mengukur tingkat keberhasilan algoritma ekstraksi yang digunakan. Jika hasil pengujian MSE semakin mendekati nol maka akan semakin baik. Berdasarkan hasil pengujian file *cover* yang dilakukan dalam percobaan ini semua nilai MSE yang dihasilkan adalah 0. Hal tersebut menandakan bahwa seluruh file *cover* hasil proses ekstraksi pesan dapat kembali ke bentuk sebelumnya seperti file *cover* sebelum proses penyisipan pesan.

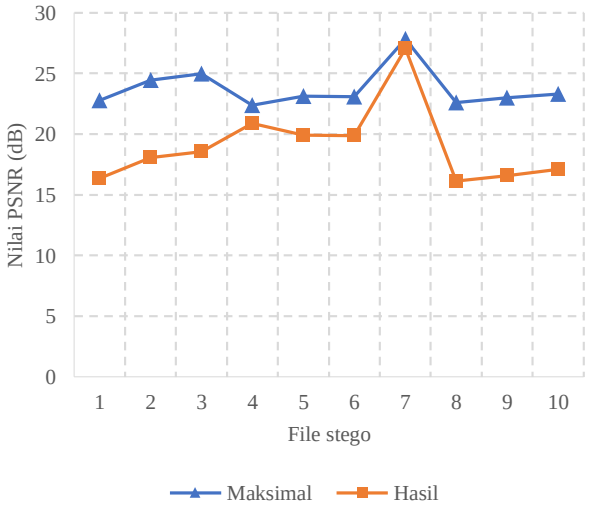
Berdasarkan data perhitungan nilai MSE dan PSNR rata-rata yang dilakukan pada file berjenis TXT, diperoleh grafik dari nilai PSNR seperti pada Gambar 11.



Gambar 11 Grafik Perbandingan nilai rata-rata PSNR maksimal dan hasil dari setiap file stego pada file TXT.

Grafik perbandingan nilai PSNR rata-rata pada file berjenis TXT dapat dilihat pada Gambar 11. Gambar tersebut menunjukkan, dalam percobaan ini file stego 1, file stego 2, dan file stego 4 memiliki selisih nilai yang cukup besar. Sedangkan untuk file stego lainnya nilai PSNR rata-rata hasil yang diperoleh mendekati nilai PSNR rata-rata maksimal dari file stego tersebut.

Berdasarkan data perhitungan nilai MSE dan PSNR rata-rata yang dilakukan pada file berjenis DOCX, diperoleh grafik dari nilai PSNR seperti pada Gambar 7.



Gambar 12 Grafik perbandingan nilai rata-rata PSNR hasil dan maksimal setiap file stego pada file DOCX.

Grafik perbandingan nilai PSNR rata-rata pada file berjenis DOCX dapat dilihat pada Gambar 12. Gambar tersebut menunjukkan, dalam percobaan ini file stego 4 dan file stego 7 memiliki selisih nilai terkecil. Sedangkan untuk

file stego lainnya nilai PSNR rata-rata hasil yang diperoleh memiliki nilai selisih yang cukup besar.

Nilai MSE dan PSNR rata-rata pada kedua jenis file dalam percobaan ini dapat dipengaruhi oleh berbagai faktor, seperti nilai terbesar dari karakter pada file *cover* dan perbandingan jumlah karakter pada file *cover* dengan jumlah karakter pesan yang disisipkan. Semakin besar selisih antar keduanya maka nilai MSE yang dihasilkan akan semakin kecil, dan begitu pula sebaliknya. Semakin kecil selisih antara keduanya maka semakin besar nilai MSE yang dihasilkan.

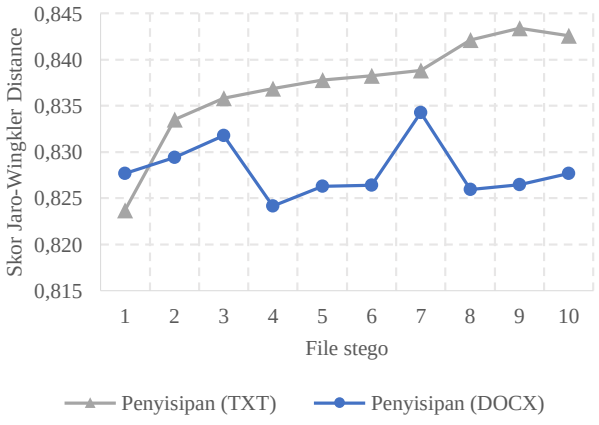
Secara umum pengujian dengan metode PSNR pada file teks tidak begitu akurat. Hal ini karena penggunaan metode PSNR pada file teks memiliki batasan nilai karakter yang dapat diuji, yaitu 255. Batasan nilai ini menyebabkan apabila terdapat karakter dengan nilai lebih besar dari 255, maka karakter tersebut tidak akan diproses.

C. Pengujian Jaro-Winkler Distance

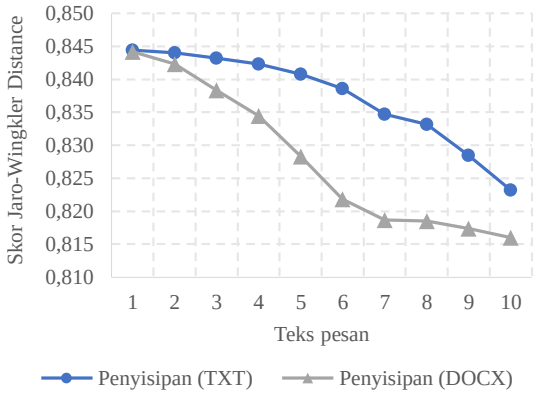
Pengujian Jaro-Winkler *Distance* ini dilakukan untuk mengetahui tingkat kesamaan file stego dengan file *cover* sebelum proses penyisipan. Pengujian ini dibutuhkan untuk mengetahui seberapa besar pengaruh metode yang digunakan terhadap perubahan yang terjadi pada file stego dan file *cover* setelah dilakukan proses ekstraksi pesan.

Perbandingan hasil perhitungan skor Jaro-Winkler *Distance* dari proses penyisipan pesan yang telah dilakukan dapat ditampilkan dalam bentuk grafik. Grafik yang dihasilkan dibagi menjadi dua, yaitu grafik skor rata-rata berdasarkan setiap file stego yang digunakan dan grafik skor rata-rata berdasarkan teks pesan yang digunakan. Grafik tersebut dapat dilihat pada Gambar 13 dan Gambar 14.

Berdasarkan grafik pada Gambar 13 dan Gambar 14 dapat dilihat bahwa besar kecilnya skor yang dihasilkan tergantung pada perbandingan jumlah karakter pesan dengan jumlah karakter pada file cover. Semakin besar selisih perbandingannya maka skor yang dihasilkan akan semakin besar.



Gambar 13 Grafik perbandingan skor rata-rata dari proses penyisipan pesan pada file TXT dan DOCX berdasarkan file stego yang digunakan.



Gambar 14 Grafik Perbandingan skor rata-rata dari proses penyisipan pesan pada file TXT dan DOCX berdasarkan teks pesan yang digunakan.

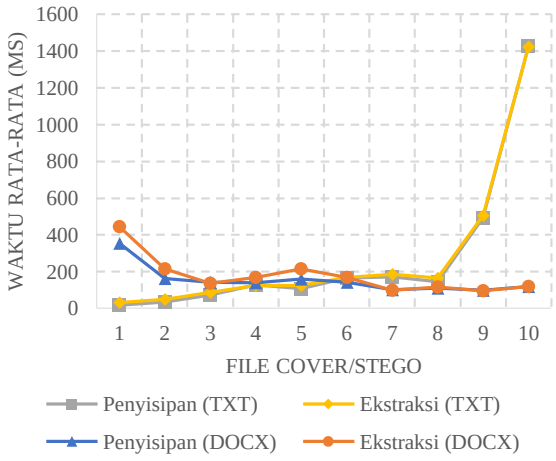
Berdasarkan pengujian dengan metode Jaro-Winkler *Distance* tersebut dapat diketahui proses penyisipan pesan mengakibatkan terjadinya perubahan pada file stego, tetapi perubahan yang terjadi tidak terlalu besar dan masih mendekati bentuk aslinya. Berdasarkan pengujian tersebut dapat pula diketahui bahwa setelah pesan yang terkandung pada file stego diekstrak atau diambil maka file tersebut akan kembali ke bentuk awalnya.

D. Pengujian Waktu Komputasi

Pengujian waktu komputasi cukup penting dilakukan untuk mengetahui waktu komputasi rata-rata yang dibutuhkan oleh sistem untuk melakukan proses penyisipan ataupun ekstraksi pesan. Pengujian waktu komputasi pada percobaan ini dilakukan dengan cara menghitung waktu yang dibutuhkan oleh sistem untuk melakukan proses penyisipan dan ekstraksi pesan. Pengujian ini dilakukan pada dua jenis file *cover* yaitu TXT dan DOCX dengan menggunakan file yang telah disisipkan.

Hasil pengujian waktu komputasi yang dilakukan menunjukkan bahwa jumlah karakter pada file yang digunakan mempengaruhi waktu komputasi yang dihasilkan. Faktor lainnya yang dapat mempengaruhi waktu komputasi yang dibutuhkan adalah tingkat kecocokan antara file yang digunakan dengan teks pesan yang disisipkan dan jumlah pesan yang berhasil disisipkan.

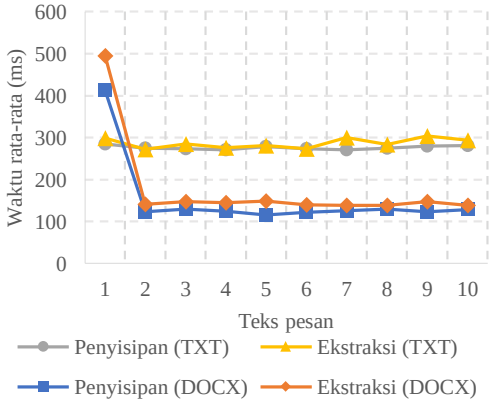
Sama seperti pada pengujian-pengujian lainnya yang dilakukan pada file DOCX. Jumlah karakter dan keberadaan konten lain pada file berjenis DOCX juga mempengaruhi hasil pengujian waktu komputasinya. Hal ini mengakibatkan terjadinya variasi waktu komputasi pada file *cover*. Faktor lainnya yang dapat mempengaruhi waktu komputasi yang dibutuhkan tidak berbeda dengan file berjenis TXT, yaitu tingkat kecocokan antara file yang digunakan dengan teks pesan yang disisipkan dan jumlah pesan yang berhasil disisipkan. Grafik waktu komputasi rata-rata pada setiap file *cover* dapat dilihat pada Gambar 15.



Gambar 15 Grafik waktu komputasi rata-rata pada setiap file *cover*.

Berdasarkan Gambar 15 dapat dilihat perbedaan waktu komputasi antara file berjenis TXT dengan file berjenis DOCX, serta perbedaan waktu komputasi antara proses penyisipan dan ekstraksi pesan pada masing-masing jenis file. Pengujian waktu komputasi pada file berjenis TXT menghasilkan perbedaan signifikan antara file *cover* 1 dan file stego 1 dengan file *cover* 10 dan file stego 10, di mana file *cover* 1 dan file stego 1 memiliki jumlah karakter terkecil sedangkan file *cover* 10 dan file stego 10 memiliki jumlah karakter terbesar. Sedangkan pada file berjenis DOCX terjadi variasi waktu komputasi yang disebabkan oleh variasi jumlah karakter pada setiap file *cover* berjenis DOCX.

Berdasarkan grafik tersebut juga terlihat jelas pada file berjenis DOCX terjadi perbedaan waktu komputasi antara file *cover* 1 atau file stego 1 dengan file lainnya, yang merupakan efek penggunaan *library* Java pada file DOCX. Efek yang disebabkan oleh penggunaan *library* tersebut dapat dilihat dengan lebih jelas pada grafik waktu komputasi rata-rata pada setiap teks pesan yang disajikan dalam Gambar 16.



Gambar 16 Grafik waktu komputasi rata-rata pada setiap teks pesan.

Berdasarkan Gambar 16, nilai rata-rata yang dihasilkan oleh teks pesan 1 pada file berjenis DOCX memiliki nilai terbesar. Hal ini terjadi karena teks pesan 1 merupakan pesan yang pertama kali diproses, dan pada saat pemrosesan tersebutlah terjadi proses pengaktifan dan pengalokasian memori untuk *library* Java yang digunakan. Sedangkan untuk file selanjutnya waktu komputasi rata-rata yang dibutuhkan tidak lagi dipengaruhi oleh proses pengaktifan *library* tersebut.

Berdasarkan data pada Gambar 16 tersebut dapat dilihat bahwa, tidak terjadi perbedaan signifikan dari waktu komputasi yang dibutuhkan oleh setiap teks pesan meskipun teks pesan memiliki panjang yang berbeda-beda. Panjang teks pesan yang akan disisipkan tidak begitu mempengaruhi waktu komputasi. Hal tersebut disebabkan cepat atau tidaknya waktu komputasi lebih dipengaruhi oleh tingkat kecocokan teks pesan dengan file *cover* yang digunakan dan jumlah karakter pada file *cover*. Secara umum waktu komputasi rata-rata yang dibutuhkan untuk melakukan proses penyisipan dan ekstraksi pesan pada file DOCX lebih cepat jika dibandingkan dengan file TXT. Hal tersebut disebabkan oleh penggunaan *library* pada file berjenis DOCX.

E. Hasil Ekstraksi Pesan

Dalam sebuah metode steganografi, kemampuan *recovery* merupakan salah satu aspek penting yang mempengruhi tingkat keberhasilan sebuah metode steganografi. Salah satu cara yang dapat dilakukan untuk melihat dan mengukur kemampuan *recovery* dari sebuah metode steganografi, yaitu dengan cara memeriksa kesamaan pesan yang di sisipkan dengan pesan yang telah diekstrak. Metode steganografi yang baik akan menghasilkan keluaran yang sama dengan pesan rahasia yang disisipkan dan tidak mengandung karakter asing ataupun karakter yang tidak disisipkan.

Pada percobaan yang telah dilakukan, didapatkan bahwa hasil semua ekstraksi pesan tidak ada yang berbeda dengan pesan sebelum disisipkan ke dalam file cover. Namun ada beberapa pesan yang tidak dapat ditampilkan semua, hal ini terjadi karena kurangnya daya tampung file cover sehingga pesan tidak sepenuhnya berhasil disisipkan.

F. Data Hasil Angket Kuesioner

Pengujian aspek *impercepbility* dari metode steganografi yang dikembangkan dilakukan dengan menyebarkan angket kuesioner kepada beberapa responden. Kuesioner tersebut berisi beberapa pertanyaan dan contoh keluaran yang dihasilkan dari proses steganografi. Pengisian kuesioner dilakukan dengan membandingkan tiga buah file, yaitu file *cover*, file stego, dan file *cover* hasil dari proses ekstraksi pesan.

Survei dilakukan dengan membagikan kuesioner kepada 30 orang mahasiswa sebagai responden. Berdasarkan data yang diperoleh dalam percobaan ini akan dilakukan perhitungan nilai MOS *(Mean Opinion Score)*. MOS merupakan metode evaluasi yang menggunakan pendapat responden sebagai dasar penilaian. Perhitungan skor dari metode MOS dilakukan berdasarkan skala standar penilaian yang dimulai dari angka 1 hingga 5. Perhitungan

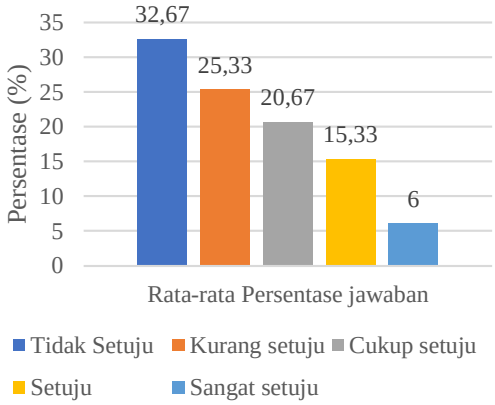
nilai MOS dilakukan terhadap setiap pertanyaan dengan menghitung jumlah responden pada setiap skor kemudian mengalikannya dengan skor masing-masing dan menjumlahkan seluruh hasil perkalian pada masing-masing skor. Setelah itu hasil penjumlahan tersebut akan dibagi dengan jumlah keseluruhan responden dan akan diperoleh nilai MOS. Selain menghitung nilai MOS akan dihitung pula persentase respon yang diberikan pada masing-masing pertanyaan.

Proses perhitungan MOS dilakukan dengan mengelompokkan pertanyaan yang ada berdasarkan tujuan dari masing-masing pertanyaan. Berikut ini merupakan hasil perhitungan MOS dan persentase pada setiap kelompok pertanyaan.

Pertanyaan 1 sampai 5 adalah pertanyaan yang bertujuan untuk mengetahui tingkat kesadaran responden terhadap perbedaan dan perubahan yang terjadi pada file stego. Semakin rendah nilai yang diberikan terhadap setiap pertanyaan maka akan semakin baik. Nilai yang rendah menandakan kecilnya tingkat kesadaran responden terhadap keberadaan pesan ataupun perubahan yang terjadi pada file stego. Hasil perhitungan MOS pada pertanyaan 1 sampai 5, pertanyaan 1 sampai 4 skor yang diperoleh berkisar di antara 2 hingga 3. Pertanyaan 1 sampai 4 merupakan pertanyaan mengenai perubahan yang terjadi pada file stego, seperti perubahan makna kata, format penulisan, dan penambahan serta pengurangan karakter. Skor yang diperoleh pada pertanyaan tersebut menandakan responden menyadari bahwa terdapat beberapa perubahan yang terjadi terhadap file stego. Sedangkan pertanyaan ke-5 memperoleh skor 1,73. Pertanyaan ke-5 merupakan pertanyaan yang digunakan untuk mengetahui tingkat kemampuan responden untuk mengetahui isi pesan rahasia ataupun tanda-tanda keberadaannya. Skor yang diperoleh pada pertanyaan ke-5 menandakan mayoritas responden tidak dapat mengetahui isi pesan rahasia yang disisipkan pada file stego. Secara keseluruhan skor rata-rata yang diperoleh pada pengujian tersebut adalah 2,37. Ini menandakan bahwa perubahan pada file stego tidak terlalu besar dan berpengaruh terhadap kemampuan responden untuk mengetahui isi pesan rahasia. Persentase rata-rata jawaban responden dari pertanyaan 1 sampai 5 dapat dilihat pada Gambar 17.

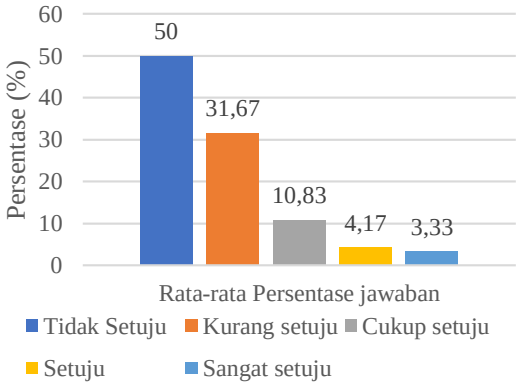
Gambar 17 menunjukkan bahwa pada pengujian tersebut diperoleh persentase sebesar 32,67% tidak setuju, 25,33% kurang setuju, 20,67% cukup setuju, 15,33% setuju, dan 6% sangat setuju. Berdasarkan data tersebut dapat dikatakan bahwa perubahan yang terjadi pada file stego tidak berpengaruh besar terhadap kemampuan responden untuk mengetahui pesan rahasia yang tersimpan di dalam file stego.

Pertanyaan 6 sampai 9 adalah pertanyaan yang bertujuan untuk mengetahui tingkat perbedaan dan perubahan yang terdapat di antara file *cover* dengan file *cover* hasil dari proses ekstraksi pesan. Semakin rendah nilai yang diberikan terhadap setiap pertanyaan maka akan semakin baik. Nilai yang rendah menandakan kecilnya tingkat perbedaan atau perubahan yang terjadi.



Gambar 17 Grafik persentase jawaban rata-rata pada pertanyaan 1 sampai 5.

Berdasarkan data hasil perhitungan MOS yang dilakukan, pada pertanyaan 6 sampai 9 skor yang diperoleh berkisar di antara 1 hingga 2. Pertanyaan 6 sampai 9 merupakan pertanyaan mengenai perubahan yang terjadi pada file *cover* hasil proses ekstraksi, seperti perubahan makna kata, format penulisan, dan penambahan serta pengurangan karakter. Skor yang diperoleh pada pertanyaan tersebut menandakan mayoritas responden tidak melihat adanya perbedaan. Persentase rata-rata jawaban responden dari pertanyaan 6 sampai 9 dapat dilihat pada Gambar 18.



Gambar 18 Grafik persentase jawaban rata-rata pertanyaan 6 sampai 9.

Gambar 18 menunjukkan bahwa pada pengujian tersebut diperoleh persentase sebesar 50% tidak setuju, 31,67% kurang setuju, 10,83% cukup setuju, 4,17% setuju, dan 3,33% sangat setuju. Berdasarkan data tersebut dapat dikatakan bahwa perubahan yang terjadi pada file stego tidak berpengaruh besar terhadap kemampuan responden untuk mengetahui pesan rahasia yang tersimpan di dalam file stego.

V. KESIMPULAN DAN SARAN

Dari hasil implementasi dan hasil pengujian sistem yang dilakukan pada percobaan ini, maka dapat diambil kesimpulan sebagai berikut:

a. Metode pencocokan LSB dan penggunaan karakter penanda pesan dapat diterapkan dalam steganografi pada file teks tanpa merusak atau mengubah karakter penampung pesan.
b. Penggunaan karakter *non-breaking space* mengakibatkan perubahan terhadap tampilan dari isi file stego, tetapi tidak mengakibatkan perubahan karakter, makna kata atau informasi yang ada. Hal ini dapat dilihat berdasarkan hasil Pengujian nilai PSNR yang menghasilkan rata-rata selisih yang kecil antara nilai PSNR maksimum dengan PSNR hasil steganografi, skor Jaro-Winkler *Distance* dengan skor rata-rata lebih dari 0,8 pada file stego dan skor rata-rata 1 pada file yang telah diekstraksi, dan hasil kuesioner dengan nilai rata-rata MOS 2,36 dan 1,79 untuk masing-masing katagori pertanyaan. Hasil pengujian tersebut menujukan perubahan yang terjadi tidak terlalu besar dan File dapat kembali ke bentuk semula setelah diekstraksi.
c. Berdasarkan pengujian kapasitas yang telah dilakukan, daya tampung dari file *cover* yang digunakan pada metode tergolong kecil. Rasio kapasitas rata-rata yang dihasilkan masih kurang dari 1%. Hal ini disebabkan penggunaan karakter spasi sebagai acuan penyisipan pesan dan pencocokan LSB menyebabkan hanya karakter tertentu pada file *cover* yang dapat disisipkan pesan.
d. Waktu komputasi yang dibutuhkan dapat dipengaruhi oleh beberapa faktor seperti, jumlah karakter pada file, tingkat kecocokan pesan dengan file *cover*, dan penggunaan bantuan *library* pada sistem. Berdasarkan percobaan yang dilakukan waktu komputasi pada file berjenis DOCX relatif lebih cepat dari file berjenis TXT. Hal ini disebabkan oleh penggunaan *library* Java untuk pemrosesan file DOCX.

Adapun saran yang dapat disampaikan untuk pengembangan metode dan sistem lebih lanjut adalah sebagai berikut:

a. Untuk meningkatkan keamanan dari metode ini dapat dilakukan proses kriptografi terhadap teks pesan sebelum proses penyisipan dilakukan.
b. Untuk meningkatkan daya tampung dari metode ini dapat dilakukan penambahan karakter penanda yang digunakan.
c. Metode penanda pesan dengan karakter dapat diubah menjadi penggunaan file kunci. File kunci tersebut dapat berisi pemetaan lokasi karakter yang mengandung pesan.
d. Berdasarkan hasil penelitian yang dilakukan, metode pencocokan LSB (*Least Significant Bit*) tidak mengubah nilai bit yang digunakan sebagai wadah pesan, sehingga bit penampung pesan yang dapat digunakan tidak hanya LSB, tetapi seluruh bit pada karakter.

DAFTAR PUSTAKA

[1] P. Alatas, "Implementasi Teknik Steganografi dengan Metode LSB pada Citra Digital," Fakultas Ilmu Komputer & Teknologi Informasi, Universitas Gunadarma, Depok, 2009.

[2] M. Argawal, "Text Steganographic Approaches: A Comparison," *International Journal of Network Scurity & Its Applications (IJNSA),* vol. 5, no. 1, p. 92, 2013.

[3] I. Andiniarti, "Implementasi Steganografi pada Media Teks dengan Metode Line-Shift Coding dan Metode Centroid," Fakultas Matematika dan ilmu Pengetahuan Alam, Institut Pertanian Bogor, Bogor, 2009.

[4] R. Popa, "An Analysis of Steganographic Techniques," The Politehnica University of Timisoara, Faculty of Automatics and computers, Department of computer science and Software Engineering, 1998.

[5] B. Prasetiyo, "Kombinasi Steganografi Bit Matching dan Kriptografi DES untuk Pengamanan Data," Universitas Diponegoro, Semarang, 2013.

[6] J. Mielikainen, "LSB Matching Revisited," *IEEE Signal Processing Letters,* vol. 13, no. 5, pp. 285-287, 2006.

VI. LAMPIRAN DAFTAR PERNTANYAAN MOS

No	Pernyataan
1.	Apakah secara umum terdapat perbedaan signifikan antara file cover dengan file stego?
2.	Apakah terdapat perubahan huruf/karakter, makna kata atau informasi pada file stego (lampiran 2)?
3.	Apakah terdapat penambahan atau pengurangan karakter atau kata pada file stego (lampiran 2)?
4.	Apakah terjadi perubahan format penulisan pada file stego (lampiran 2)? (ukuran karakter, letak karakter, margin, dll.)
5.	Apakah pesan rahasia yang disisipkan terlihat jelas?
6.	Apakah file *cover* (lampiran 1) sama persis dengan file setelah ekstraksi pesan (lampiran 3)?
7.	Apakah terdapat perubahan huruf/karakter, makna kata atau informasi pada file setelah ekstraksi pesan (lampiran 3)?
8.	Apakah terdapat penambahan atau pengurangan karakter atau kata pada file setelah ekstraksi pesan (lampiran 3)?
9.	Apakah terjadi perubahan format penulisan pada file setelah ekstraksi pesan (lampiran 3)? (ukuran karakter, letak karakter, margin, dll.)