# Penerapan Steganografi Dalam Memberikan Watermarking Pada Teks

Sigit Aji Nugroho 13510021
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*
*13510021@students.stei.itb.ac.id*

***Abstract*—Makalah ini menjabarkan tentang salah satu cara memberikan *watermarking* pada teks. Metode *watermarking* yang dipakai menggunakan prinsip steganografi pada teks. Pesan disimpan memanfaatkan keberadaan spasi pada teks. Spasi sebanyak satu buah menyimpan kode 0. Sedangkan spasi sebanyak dua buah menyimpan kode 1.**

**Format 0 dan 1 ini sama dengan bilangan biner. Jadi untuk menampung huruf abjad yang jumlahnya ada 26, dibutuhkan minimal lima buah bit biner ($2^5$=32). Masing-masing angka biner disimpan sebagai satu spasi**

***Index Terms*—teks rahasia, teks semu, teks stego, kunci stego.**

## I. PENDAHULUAN

Pada zaman yang kian modern ini, semakin banyak orang yang suka menulis. Semakin mudah dan banyaknya teknologi untuk menulis menjadi penunjang terjadinya fenomena ini. Dahulu seseorang hanya bisa menulis dengan pena atau pensil dan kemudian dengan mesin tik. Namun kini dengan berkembangnya komputer, laptop, tablet dan lainnya, semakin banyak fasilitas untuk menulis.

Berkembangnya internet, membuat penyebaran informasi semakin cepat dan luas. Termasuk juga penyebaran informasi berupa tulisan. Dengan segala fasilitas ini, seorang penulis bisa dengan mudah mempublikasikan tulisannya. Seseorang bisa dengan cepat pula mengirimkan tulisannya ke media massa.

Namun dibalik semua kemudahan ini, semakin banyak pula kejadian plagiasi terhadap tulisan. Untuk tulisan-tulisan yang telah didaftarkan hak ciptanya, tentu memiliki kekuatan untuk melawan plagiasi. Namun bagaimana dengan tulisan-tulisan yang belum memiliki hak cipta. Misalnya suatu puisi yang ditampilkan penulisnya ke dalam blog pribadi. Orang lain bisa meng-*copy* puisi tersebut dengan mudah. Kemudian puisi tersebut bisa dimasukkan ke dalam karyanya. Jika karya itu dikomersilkan, maka ada peluang bahwa penulis asli tidak bisa mendapatkan hak-haknya.

Kejadian seperti di atas tentunya merugikan bagi penulis asli. Hasil kekayaan intelektualnya telah disalahgunakan. Oleh karena itulah disusun algoritma ini sebagai salah satu cara untuk mengatasinya.

Penulisan kepemilikan karya teks tidak bisa diletakkan begitu saja di bagian awal atau akhirnya. Jika hanya dilakukan dengan cara tersebut, tentu saja dengan mudah bisa dihapus. Jejak penulis di karya tersebut tidak akan dapat ditemukan lagi.

Oleh karena itu, data kepemilikan ini harus di sembunyikan. Pembaca hanya akan melihat teks semunya (tulisan karya) saja. Pembaca yang jahat tidak akan menyadari bahwa di dalam tulisan tersebut sebenarnya telah disimpan teks rahasia berupa pesan kepemilikan.

## II. DAFTAR TEORI

*Watermarking* adalah penyisipan informasi yang menyatakan kepemilikan suatu karya. Informasi *(watermark)* bisa berupa teks, logo, audio, dan lainnya. Pada aplikasi ini, *watermark* berupa tulisan, misal nama pemilik karya. Tujuan dari *watermarking* adalah memberikan perlindungan terhadap karya seseorang layaknya *copyright*.

*Watermarking* merupakan aplikasi dari steganografi. Steganografi berasal dari Bahasa Yunani *steganos* yang artinya tulisan tersembunyi. Steganografi adalah ilmu dan seni menyembunyikan pesan dengan cara menyisipkannya ke dalam pesan lain sehingga selain pengirim dan penerima tidak ada yang menyadari keberadaan pesan rahasia tersebut. Pesan yang disembunyikan bisa berupa teks, gambar, audio, dan video. Tempat penyembunyiannya juga bisa berbagai macam jenis. Hanya saja ukuran tempat penyembunyian harus lebih besar dari dari ukuran pesan itu sendiri. Tujuannya jelas agar pesan bisa tersembunyikan semuanya.

Suatu steganografi memiliki empat properti wajib. Properti pertama adalah *embedded message*. Untuk steganografi pada teks, properti ini disebut *hiddenteks* atau teks rahasia. Teks ini adalah pesan yang akan disembunyikan.

Properti kedua adalah *cover-object* atau *coverteks*. Nantinya dalam algoritma ini akan disebut sebagai teks semu. *Cover-object* adalah pesan yang digunakan untuk

menyembunyikan *embedded message*. Untuk kasus ini artinya *cover-object*-nya berupa tulisan karya.

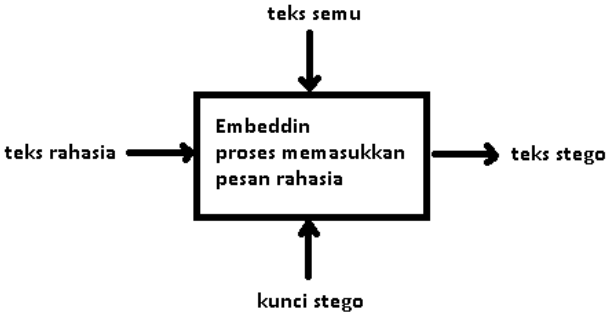
Properti ketiga adalah *stego-object* atau *stegotext*. Pada aplikasi ini menggunakan istilah teks stego. Teks stego adalah tulisan karya yang di dalamnya telah diberi teks rahasia.

Properti terakhir adalah *stego key* atau kunci stego. Kunci ini digunakan untuk menyisipkan pesan dan mengekstraksi pesan dari teks stego. Kunci ini digunakan untuk menetukan letak-letak spasi yang disisipi pesan. Hal ini harus dilakukan karena jika spasi yang diisi pesan berurutan, bisa dengan mudah diterjemahkan oleh pembaca jahat. Kemudian bisa saja pesan rahasia dihapus atau diganti.

## III. ANALISIS DAN IMPLEMENTASI

### *A. Skema Memasukkan Pesan Rahasia*

Aplikasi ini dapat digunakan untuk memasukan teks rahasia ke dalam teks semu. Properti yang dibutuhkan adalah teks rahasia, teks semu, dan kunci stego. Skemanya dapat dilihat dalam gambar di bawah.




Gambar 3.1 Skema proses memasukkan pesan rahasia

Saat hendak melakukan proses *embedding*, properti yang harus dimasukkan terlebih dahulu adalah teks semu. Hal ini karena teks karya akan dipublikasikan sehingga harus dijaga originalitasnya. Oleh karena itu, sebisa mungkin teks karya tidak mengalami perubahan.

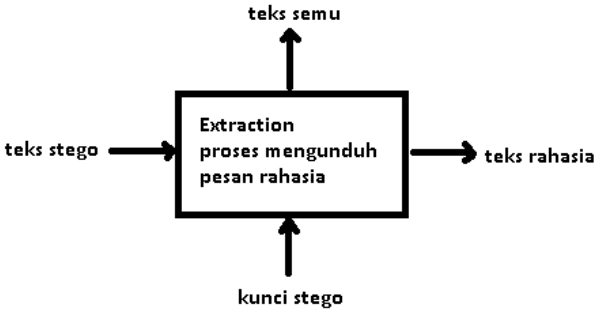
Selanjutnya yang harus dimasukkan adalah kunci stego. Kunci ini akan digunakan sebagai nilai pemicu bilangan acak semu yang berasal dari fungsi pembangkit. Bilangan acak semu ini akan digunakan untuk menentukan lokasi-lokasi spasi tempat menyimpan pesan rahasia.

Setelah teks semu dan kunci stego dimasukkan, selanjutnya aplikasi akan menghitung panjang maksimal teks rahasia yang bisa dimasukkan. Penghitungan ini akan dilakukan oleh fungsi penghitungan panjang teks. Selanjutnya tinggal memasukkan pesan rahasia yang panjangnya maksimal seperti hasil fungsi penghitung panjang teks.

Setelah itu aplikasi akan menampilkan teks stego berupa tulisan karya yang telah diatur spasi antarkatanya. Penulis karya tinggal meng-*copy* tulisan stego ini. Selanjutnya tullisan bisa dipublikasikan.

### *B. Skema Mengunduh Pesan Rahasia*

Selain dapat digunakan untuk memasukkan teks rahasia, aplikasi ini juga dapat digunakan untuk mengunduh teks rahasia dari teks stego. Proses ini mungkin dibutuhkan jika suatu saat ternyata tulisan karya penulis dicuri dan disalahgunakan oleh orang lain. Agar dapat dibuktikan bahwa ini adalah karya asli Properti yang dibutuhkan adalah teks stego dan kunci stego. Skemanya dapat dilihat pada gambar di bawah.




Gambar 3.2 Skema proses mengunduh pesan rahasia

Langkah pertama yang harus dilakukan saat ingin mengunduh teks rahasia adalah memasukkan teks stego ke dalam aplikasi. Selanjutnya kunci stego bisa dimasukkan. Dengan kunci ini, lokasi-lokasi pesan rahasia di dalam teks stego dapat ditemukan.Fungsi ekstaksi akan membaca pesan rahasia yang tersimpan di dalam karya tersebut.

### *C. Metode Penentuan Lokasi Pesan Rahasia*

Pesan rahasia akan disimpan di dalam teks stego pada bagian spasinya. Untuk menentukan spasi mana yang menjadi lokasi pesan, digunakanlah suatu metode yang cukup sederhana. Metode ini berkaitan dengan angka yang dihasilkan oleh kunci pembangkit.

Lokasi pertama sesuai dengan angka pertama yang dihasilkan fungsi pembangkit. Jika fungsi pembangkit mengeluarkan nilai 3, artinya spasi pertama yang disisipi pesan adalah spasi yang ketiga jika dihitung dari awal teks semu. Karena satu huruf teks rahasia diterjemahkan sebagai lima karakter biner, maka satu huruf teks rahasia harus disimpan dalam lima buah spasi.

Guna memaksimalkan panjang teks rahasia yang dapat dimasukkan, maka karakter biner kedua sampai kelima dari huruf teks rahasia pertama disimpan di spasi-spasi selanjutnya secara berurutan. Artinya huruf rahasia pertama karakter biner kedua dimasukkan ke spasi keempat. Huruf rahasia pertama karakter biner ketiga disimpan ke spasi keempat, dan seterusnya.

| Nomor spasi | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| Teks semu | Pada | suatu | hari | hiduplah | seorang | pengembara | yang | bernama | Suketi. |
| Teks rahasia dalam biner | | | | 1a | 1b | 1c | 1d | 1e | |

Gambar 3.3 Contoh penempatan pesan rahasia

Guna menentukan lokasi huruf rahasia kedua disimpan, maka kembali diambil nilai dari fungsi pembangkit. Misal nilai yang dimunculkan adalah empat, maka karakter biner pertama dari huruf rahasia kedua disimpan di spasi keempat setelah spasi terakhir yang sudah digunakan untuk menyimpan. Contohnya dapat dilihat pada gambar di bawah.

| Nomor spasi | | 7 | | 8 | | 9 | | 10 | | 11 | | 12 | | 13 | | 14 | | 15 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Teks semu | yang | | bernama | | Suketi. | | Suketi | | berasal | | dari | | Garut. | | Usianya | | saat | | ini | | 25 |
| Lokasi teks rahasia | | 1e | | | | | | | | 2a | | 2b | | 2c | | 2d | | 2e | | | |

Gambar 3.4 Contoh 2 penempatan pesan rahasia

Huruf rahasia pertama karakter biner terakhir disimpan pada spasi ke-7. Karena nilai yang dihasilkan oleh fungsi pembangkit adalah empat, maka huruf rahasia kedua karakter biner pertama ditempatkan di spasi nomor 7+4 = 11. Karekter biner kedua disimpan di spasi selanjutnya. Begitu seterusrnya hingga karakter biner kelima.

Guna menentukan lokasi huruf rahasia ketiga, maka kembali diambil nilai dari fungsi pembangkit. Jika huruf rahasia telah disimpan semua, maka proses penentuan lokasi bisa dihentikan. Jadi ada kemungkinan spasi-spasi di bagian akhir teks semu tidak menyimpan teks rahasia.

### *D. Metode Penempatan Pesan Rahasia*

Penyimpanan teks rahasia ke dalam bilangan biner mirip implementasi bilangan hexa maupun ASCII. Hanya saja bilangan hexa hanya terdiri dari empat bit biner dan ASCII sebesar satu byte atau delapan bit biner. Sementara pada algoritma ini masing-masing huruf teks rahasia diterjemahkan sebagai lima bit biner.

Pada aplikasi ini karakter yang bisa digunakan sebagai teks rahasia dibatasi hanya berupa huruf abjad dan spasi. Tujuan pembatasan ini agar panjang bit untuk masing-masing karakter tidak berlebihan. Sehingga jumlah teks rahasia yang dimasukkan bisa lebih banyak.

Total karakter yang bisa digunakan ada 27. Huruf abjad sebanyak 26 ditambah spasi. Maka panjang bit minimal:

$$x = \lceil {}^{2}\log 27 \rceil$$

Artinya masih ada 5 slot ruang yang bisa dijadikan sebagai karakter dari teks rahasia. Nantinya bisa diisi dengan tanda baca yang sering digunakan. Berikut adalah tabel kode yang digunakan.

Tabel 1.Karakter Teks Rahasia

| Karakter | Desimal | 5 bit |
|---|---|---|
| A | 0 | 00000 |
| B | 1 | 00001 |
| C | 2 | 00010 |
| D | 3 | 00011 |
| E | 4 | 00100 |
| F | 5 | 00101 |
| G | 6 | 00110 |
| H | 7 | 00111 |
| I | 8 | 01000 |
| J | 9 | 01001 |
| K | 10 | 01010 |
| L | 11 | 01011 |
| M | 12 | 01100 |
| N | 13 | 01101 |
| O | 14 | 01110 |
| P | 15 | 01111 |
| Q | 16 | 10000 |
| R | 17 | 10001 |
| S | 18 | 10010 |
| T | 19 | 10011 |
| U | 20 | 10100 |
| V | 21 | 10101 |
| W | 22 | 10110 |
| X | 23 | 10111 |
| Y | 24 | 11000 |
| Z | 25 | 11001 |
| <spasi> | 26 | 11010 |
| | 27 | 11011 |
| | 28 | 11100 |
| | 29 | 11101 |
| | 30 | 11110 |
| | 31 | 11111 |

Berdasar tabel di atas, maka misal huruf pertama teks rahasia adalah “Y”, maka contoh pada gambar 3.1 hasilnya menjadi:

“**Pada suatu hari hiduplah seorang pengembara yang bernama Suketi.**”

Pada contoh teks stego di atas, spasi yang disisipi pesan adalah spasi ke-3 sanpai ke-7. Karena nilai biner dari huruf “Y” adalah 11000, artinya lokasi untuk menyimpan bit biner ke-1 dan ke-2, jumlah spasinya harus dua. Di contoh hasil tampak spasi ke-3 dan ke-4 jumlahnya dua buah.

### *E. Fungsi Pembangkit*

Pada aplikasi ini, algoritma lain telah dibuat sederhana dengan tujuan agar dengan panjang teks semu yang sama bisa menampung teks rahasia yang lebih banyak. Jadi, di fungsi inilah kesempatan membuat suatu algoritma yang rumit. Tujuannya tentu agar algoritma yang dimiliki aplikasi ini menjadi sulit dipecahkan.

Agar tetap memenuhi prinsip dapat menampung teks rahasia sebanyak mungkin, maka di akhir fungsi yang rumit tinggal diberi modulo. Tujuannya jelas, berapapun nilai yang dihasilkan kalkulasi sebelumnya, rentang hasil tidak lebih dari nilai modulo yang diberikan. Hanya saja agar tidak diperlukan fungsi untuk mengatasi penghitungan modulo pada bilangan negatif, maka harus dipastikan bahwa nilai yang dihasilkan pada fungsi

sebelumnya selalu bernilai positif.

Misal jarak antara bit kelima karakter sebelumnya tak ingin lebih dari sepuluh dengan bit pertama karakter yang akan dimasukkan, maka penghitungan rumit tadi tinggal dimodulo sepuluh. Hasil yang dikeluarkan hanya antar 0-9. Tetapi meski rentang hasil kecil, fungsi random semu ini tetap sulit dideteksi. Sehingga nampak seperti fungsi pembangkit bilangan random yang sesungguhnya.

Sebagai input pertama fungsi ini adalah stego key yang dimasukkan user. Agar lebih mudah, maka dibatasi nilai yang dimasukkan user adalah bilangan asli. Tapi sebenarnya bisa saja fungsi yang dimasukkan berupa karakter ASCII atau unicode. Nantinya masing-masing karakter tinggal diterjemahkan nilai desimalnya. Kemudian masing-masing nilai desimal bisa dijumlahkan atau dikalikan atau dilakukan operasi lain yang lebih rumit. Kerumitan kunci juga akan meningkatkan kerumitan fungsi pembangkit secara keseluruhan.

Ada beberapa catatan yang sempat diamati dalam jalannya aplikasi. Jika fungsi yang dimasukkan seperti rumus (1) di bawah, maka operasi yang dilakukan jika dikerjakan pada komputer biasa akan membutuhkan waktu yang lama. Lebih baik menggunakan operasi yang menggunakan logaritma atau dilengkapi konstanta-kontanta yang panjang misalnya π (phi).

$$N_{i+1}=((N^6+4N^3+2n)\%10)+1 \quad (1)$$

$$N_{i+1}=(\lceil {}^{7}\log(6N+14\pi)\rceil\%10)+1 \quad (2)$$

Hasil nilai modulo kemudian ditambah dengan satu. Tujuannya agar tidak ada nilai akhir fungsi pembangkit sama dengan 0 (nol). Jika 0 maka lokasi penempatan saat ini sama dengan lokasi spasi yang sudah ditempati bit terakhir karakter sebelumnya. Selain dengan ditambah satu, bisa digunakan cara atau ketentuan lain, misal jika nilainya 0 dianggap sebagai 1.

Input dari fungsi pembangkit yang selanjutnya, di dapat dari hasil fungsi pembangkit saat ini. Karena inputnya selalu berbeda, maka hasilnya juga berbeda. Untuk menjamin ini maka harus dipastikan bahwa jika inputnya misal 3 hasilnya tidak boleh 3 juga. Tujuannya agar tidak terjadi pengulangan menghasilkan nilai 3 sampai akhir fungsi digunakan.

### *F. Fungsi Penghitung Panjang Teks*

Fungsi ini digunakan untuk mengetahui berapa panjang teks rahasia yang bisa dimasukkan. Sebelum panjang pesan rahasia bisa diketahui, harus diketahui terlebih dahulu teks semu dan kunci stegonya. Teks stego digunakan untuk mencari banyaknya spasi yang bisa disisipi. Kunci digunakan untuk mengetahui spasi mana saja yang bisa digunakan untuk menyimpan bit biner teks rahasia.

Misal **M** adalah banyaknya spasi dalam teks semu. **N** adalah nilai kunci stego dari pengguna atau selanjutnya adalah nilai keluaran dari fungsi pembangkit. Selanjutnya **A** adalah penghitung panjang teks rahasia yang awalnya diinisiasi dengan nilai 0 (nol). Integer **J** digunakan untuk memegang spasi yang saat ini sedang dicek, awalnya juga diinisiasi dengan 0 (nol).

```
Integer A=0;
Integer J=0;
While (J<M) do
        N=Pembangkit(N);
        J=J+N; (3)
        J=J+4; (4)
        If (J<M) do (5)
                A=A+1; (6)
Return A;
```

Rumus (3) di atas digunakan untuk menentukan lokasi bit pertama dari karakter rahasia tersebut. Rumus (4) digunakan untuk menentukan bit terakhir karakter tersebut. Rumus (6) digunakan untuk menambah jumlah teks rahasia yang bisa ditampung. Syaratnya hingga bit ke lima masih muat dimasukkan ke dalam teks semu (5). Jika sudah tidak muat, maka penghitungan akan dihentikan karena syarat iterasi (**while**) bahwa **J** harus lebih kecil dari **M**. Fungsi tinggal mengembalikan nilai **A** sebagai panjang teks rahasia yang bisa dimasukkan.

### *F. Metode Mengunduh Pesan Rahasia*

Layaknya saat menempatkan, proses mengunduh juga diawali dengan memasukkan kunci ke dalam fungsi pembangkit. Nilai yang dihasilkan merupakan lokasi bit pertama huruf rahasia pertama. Kemudia dicatat spasinya satu atau dua. Jika satu disimpan sebagai bit 0. Jika dua disimpan sebagai bit 1. Kemudian dicek juga empat spasi selanjutnya. Dari lima bit yang sudah terkumpul kemudian dibandingkan menggunakan Tabel Karakter Teks Rahasia. Diperolehlah karakter pertama teks rahasia.

Guna mengunduh karakter kedua, kembali dipanggil fungsi pembangkit. Nilai masukan dari fungsi pembangkit saat ini adalah nilai keluaran dari fungsi pembangkit sebelumnya. Selanjutnya nilai keluaran tersebut digunakan untuk menentukan jarak bit pertama karakter kedua dari bit terakhir karakter pertama. Kemudian kembali digali jumlah spasinya.

Proses ini terus diulang hingga spasi pada stego teks sudah habis. Fungsi ini masih memiliki kelemahan. Jika panjang teks rahasia yang dimasukkan di awal lebih kecil dari pada kapasitas maksimal, maka saat diunduk sisa kapasitas akan berisi karakter ‘A’ yang berulang. Hal ini bisa terjadi karena sisa spasi tak pernah disisipi sehingga jumlahnya semua satu. Tapi saat diunduh tetap digali sampai akhir. Karena spasinya semua satu, bit yang dihasilkan ‘00000’ yang diterjemahkan sebagai karakter ‘A’.

Hal ini kurang menjadi masalah jika teks rahasia berisi nama seseorang. Batas antara teks rahasia dan teks sisa kapasitas masih bisa dilihat. Sehingga teks rahasianya masih jelas. Sedangkan jika teks rahasia yang dimasukkan adalah kode bebas misal "LLIIIIBBAAA", maka batas antara teks rahasia dan teks sisa kapasitas tidak jelas. Oleh karena itu, algoritma ini perlu diperbaiki.

Cara lain untuk mengatasi masalah di atas ialah menyimpan panjang teks rahasia ke dalam teks stego. Cara untuk memasukkannya ada bermacam-macam. Hanya saja tetap harus dipastikan bahwa keorisinalitasan karya tidak berubah.

## IV. PENGUJIAN

### *A. Penghapusan Spasi Ganda*

Hasil teks stego berupa karya tulisan tersebut jika diamati oleh orang yang teliti sesungguhnya cukup mencolok keanehannya. Beberapa spasi ada yang berganda. Apalagi jika karya ditulis dalam format **justify**, maka jarak spasi pada suatu baris bisa semakin lebar. Keadaan ini tentu cukup mengganggu bagi seorang penulis atau pembaca yang teliti.

Jika kemudian seluruh spasi ganda dihapus dan dijadikan spasi tunggal, maka semua teks rahasia akan lenyap. Saat diunduh, teks rahasia yang didapat hanya karakter 'A' yang berulang. Jal ini karena semua karakter hanya akan memiliki nilai bit '00000' yang di dapat dari lima buat spasi tunggal.

### *B. Pengubahan Salah Ketik*

Terkadang dalam karya yang telah dimasukkan ke dalam aplikasi masih ada salah ketik. Kemudian penulis ingin melakukan perbaikan. Pembenaran penulisan terhadap suatu kata tidak akan menjadi masalah. Hal ini karena tidak terjadi perubahan jumlah dan posisi spasi yang dimiliki teks stego. Oleh karena itu, keberadaan teks rahasia masih baik-baik saja.

### *C. Penambahan Atau Pengurangan Kata.*

Setelah dipublikasikan, terkadang ada beberapa pihak yang melakukan plagiasi. Plagiator bisa saja melakukan beberapa perubahan terhadap isi karya, hal ini agar karya nampak berbeda dari tulisan aslinya. MIsalnya dengan menambah beberapa kata atau bahkan kalimat. Bbisa juga dengan mengurangi jumlah kata atau kalimatnya. Akibatnya akan terjadi kerusakan pada teks rahasia.

Hingga sebelum lokasi spasi yang mulai mengalami perubahan, karakter teks rahasia masih aman. Tapi di titik yang mengalami perubahan, misal penambahan satu kata, maka lokasi yang di cek akan bergeser. Contoh kasus dapat dilihat pada gambar di bawah.

| Nomor spasi | | 10 | | 11 | 12 | | 13 | | 14 | | 15 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Teks stego asli | Suketi | berasal | 0 | dari | 1 | Garut. | 1 | Usianya | 0 | saat | 1 | ini |
| Teks stego palsu | Suketi | berasal | 0 | dari | 1 | Kabupaten | 0 | Garut. | 1 | Usianya | 0 | saat |
| Lokasi teks rahasia | | | | 2a | 2b | | 2c | | 2d | | 2e | |

Gambar 3.4 Contoh 2 penempatan pesan rahasia

Kode 0 dan 1 di dalam baris teks stego asli menunjukkan karakter bit teks rahasia yang disimpan di lokasi tersebut. Kemudia teks mengalami perubahan dengan penambahan kata "Kabupaten" sebelum kata "Garut". Akhirnya muncul spasi baru di posisi ke-13. Spasi di sana yang awalnya ganda kini menjadi tunggal. Akibatnya saat spasi ke-13 digali bit yang dihasilkan adalah 0, padahal seharusnya adalah 1. Begitu juga dengan spasi ke-14. Saat digali bit yang diperoleh adalah 1 padahal semestinya adalah 0. Hal yang sama terjadi pada spasi ke-15 tempat menyimpan bit kelima karakter rahasia kedua.

Karakter rahasia dua seharusnya memiliki nilai bit '01101' dan diterjemahkan sebagai huruf 'N'. Namu akibat perubahan pada teks stego, nilai bit yang digali menjadi '01010'. Saat diterjemahkan menjadi huruf 'K'.

Perubahan ini juga berimbas pada karakter-karakter setelahnya. Spasi-spasi yang disisipi pesan bergeser satu kebelakang. Akibatnya saat digali akan terjadi kesalahan pembacaan nili bit dari yang seharusnya.

## V. KESIMPULAN

Ilmu stgenografi dan *watermarking* merupakan cabang ilmu kriptografi. Ilmu kriptografi sering berkaitan dengan persandian dan pesan rahasia. Ilmu ini sering digunakan untuk opreasi militer atau bahkan terorisme. Oleh karena itu, selalu ada pihak yang berusaha untuk memecahkan sandi yang telah diciptakan guna menangkap pesan rahasia yang disampaikan.

Demi menjaga kerahasiaan pesan, maka algoritma yang digunakan untuk menyandikan dibuat serumit mungkin. Pada algoritma ini, bagian yang menentukan tingkat kerumitan algoritma adalah fungsi pembangkit. Semakin rumit operasi yang ada di dalam fungsi pembangkit, nilai yang dihasilkan akan semakin nampak acak. Akibatnya akan semakin susah seseorang untuk membaca teks rahasia yang dimasukkan.

Fungsi pembangkit menjadi salah satu penentu kinerja dari aplikasi. Jika kompleksitas dari fungsi pembangkit terlalu besar, maka satu kali proses pemanggilan bisa membutuhkan waktu yang cukup lama. Apalagi fungsi dipangil secara iteratif.

Kelemahan dari aplikasi ini salah satunya terletak pada fungsi penghitung panjang teks. Dalam fungsi ini, juga dipanggil fungsi pembangkit secara iteratif. Akibatnya kompleksitas algoritma menjadi dua kali lipat. Seharusnya ada metode lain untuk mengatasi ini. Misalnya hasil setiap keluaran fungsi pembangkit yang dipanggil pada

fungsipenghitung panjang teks disimpan dalam sebuah *array*. Sehingga saat hendak menentukan lokasi pesan rahasia tidak perlu lagi memanggil fungsi pembangkit. Melainkan cukup dengan membaca nilai di dalam *array*.

Kelemahan lain dari aplikasi ini terdapat dalam fungsi pengunduh teks rahasia. Belum ditemukan cara bagaimana agar teks rahasia bisa digali dengan tepat. Salah satu cara yang mungkin digunakan adalah menyimpan panjang teks rahasia ke dalam stego teks. Namun karena stego teks berupa karya tulisan, harus dipertimbangkan bagaimana cara menyimpannya tanpa merusak orisinalitas dari karya.

Kelemahan yang paling utama dari aplikasi ini ialah rentan terhadap perubahan.Pegangan utama dari algoritma ini adalah panjang dan posisi spasi. Jika ada perubahan satu spasi saja, entah itu berkurang atau bertambah, akan mengakibatkan perubahan posisi penyimpanan bit teks rahasia di dalam teks stego. Akibatnya lokasi spasi yang digali tidak lagi tepat pada posisi bit karakter yang seharusnya.

## VI. ACKNOWLEDGMENT

Penulis ingin mengucapkan terima kasih yang paling besar kepada Allah SWT. Dialah yang senantiasa memberikan kesehatan dan kesempatan bagi penulis sehingga bisa menyusun makalah ini hingga selesai. Alhamdulillah.

Ucapan terima kasih selanjutnya ingin penulis sampaikan kepada dosen mata kuliah IF3058, Bapak Rinaldi Munir. Berkat bimbingan beliau penulis bisa memahami ilmu-ilmu dalam bidang Kriptografi. Semoga kedepannya, ilmu ini akan terus bermanfaat bagi penulis dan orang-orang di sekitarnya.

Ucapan terima kasih terakhir diberikan kepada teman-teman sekelas penulis. Khususnya ucapan terima kasih yang sangat besar kepada rekan sekelompok tugas. Atas kerja samanya, tugas besar dan tugas kecil yang diberikan pada mata kuliah ini bisa dikerjakan dengan baik.

## REFERENCES

[1] Muntaha, Amir. 2008."*Studi Implementasi Steganografi Pesan tulisan dengan Menggunakan Pembangkit Kalimat*". Makalah kuliah Kriptografi, ITB.

[2] Wayne, Peter. 2009. *Disappearing Cryptography 3rd Editon: Information Hidding: Steganography & Watermarking*. Amsterdam: MK/Morgan Kaufmann Publishers.

## 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, 26 Maret 2013

Sigit Aji Nugroho
13510021