Eksperimen Steganalisis dengan Metode *Visual Attack* pada Citra Hasil EzStego Berformat GIF

Rinaldi Munir
Kelompok Keilmuan Informatika
Sekolah Teknik Elektro dan Informatika (STEI) ITB
Bandung, Indonesia
E-mail: rinaldi-m@stei.itb.ac.id

***Abstrak*—Di dalam makalah ini dipresentasikan hasil-hasil eksperimen steganalisis pada citra berformat GIF yang telah disisipi pesan. Penyisipan pesan menggunakan algoritma *EzStego* dan algoritma modifikasinya. Metode steganalisis yang digunakan adalah *Visual Attack*. Eksperimen dilakukan dengan bermacam ukuran dan tipe pesan, baik pesan normal maupun pesan acak, dan pada beberapa macam citra tipikal. Dari berbagai eksperimen diperoleh hasil pengamatan bahwa artefak yang terbentuk akibat penyisipan memiliki karakteristik yang berbeda, bergantung pada tipe pesan, ukuran, dan cara penyisipan (acak atau sekuensial).**

Kata kunci—Stegenalisis; citra GIF; EzStego; visual attack.

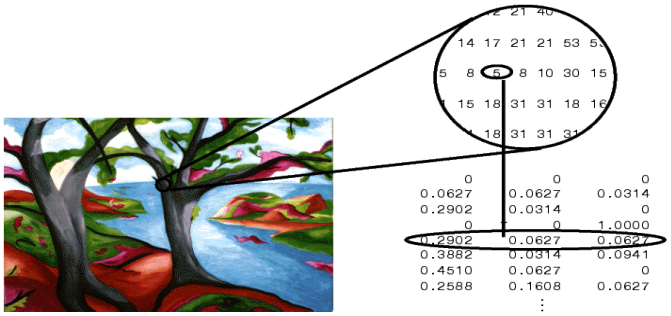
I. PENDAHULUAN

Steganalisis adalah ilmu dan seni untuk mendeteksi pesan yang tersembunyi di dalam data multimedia. Berbeda dengan steganografi yang tujuannya adalah untuk menyembunyikan pesan di di dalam data multimedia (teks, citra, audio, video), maka steganalisis adalah sebaliknya. Diberikan sebuah data multimedia yang dicurigai mengandung pesan rahasia, maka pertanyaan steganalisis adalah: apakah data multimedia tersebut mengandung pesan tersembunyi? Jawaban untuk pertanyaan tersebut adalah YA atau TIDAK. Oleh karena itu, basis serangan di dalam steganalisis adalah membedakan antara *cover object* (data multimedia yang tidak mengandung pesan tersembunyi) dengan *stego-object* (data multimedia yang mengandung pesan tersembunyi) [1]. Tidak seperti kriptanalisis di dalam bidang kriptologi yhang bertujua untuk mendekripsi pesan, steganalisis tidaklah bertujuan untuk mengekstraksi pesan yang tersembunyi di dalam *stego-object*, karena pekerjaan ini tidak *feasible*, sebab diperlukan pengetahuan tentang sistem steganografi yang digunakan dan kunci dekripsi (jika pesan dienkripsi sebelum disisipkan). Sekali diketahui sebuah media adalah *stego-object*, maka hasil ini akan mengarah pada ekstraksi pesan. Informasi tambahan seperti panjang pesan atau perkiraan lokasi penyisipan di dalam media adalah informasi yang berharga bagi steganalis untuk tahap selanjutnya (ekstraksi pesan).

Metode steganalisis pada citra digital secara umum dapat dibagi menjadi dua kelompok. Pertama *visual attack* dan kedua *statistical attack*. Kelompok metode pertama merupakan serangan berbasis pengamatan secara indrawi. Metode ini sederhana namun bersifat subyektif, sebab memerlukan pengamatan secara kasat mata untuk melihat artefak yang mencurigakan di dalam *stego-image*, lalu membandingkannya dengan citra asli (*cover image*). Kelompok metode kedua menggunakan analisis matematik pada citra untuk menemukan perbedaan antara *cover image* dan *stego image*. Metode ini lebih kompleks sebab didasarkan pada fakta bahwa penyembunyian pesan ke dalam citra menimbulkan artefak yang dapat dideteksi secara statistik sehingga dapat mengungkap penyembunyian pesan [2].

Visual attack merupakan metode awal di dalam riset steganalisis. Metode ini sangat efektif untuk mendeteksi pesan yang disisipkan dengan metode LSB (*Least Significant Bit*). Metode LSB merupakan metode paling dasar paling sederhana di dalam steganografi. Umumnya metode LSB diterapkan pada citra *bitmap* tak-terkompresi, misalnya citra dengan format BMP.

Salah satu format citra yang populer di dalam Internet adalah citra dengan format GIF (*Graphics Interchange Format*). Citra GIF adalah jenis citra terindeks (*indexed image*) yang pertama kali diperkenalkan oleh *Compuserve* pada tahun 1987. Format GIF biasanya digunakan untuk menyimpan citra grafika komputer, citra ikonik, kartun, logo, animasi, maupun citra natural. Sebuah citra terindeks mengunakan palet 256-warna yang disusun oleh 24-bit RGB dengan nilai berada di dalam selang [0, 1]. Nilai-nilai *pixel* menyatakan indeks ke nomor baris palet (Gambar 1). Warna sebuah *pixel* adalah kombinasi dari komponen *red* (R), *green* (G), dan *blue* (B).



Gambar 1. Struktur citra GIF (Sumber: Matlab)

Terdapat beberapa algoritma (sekaligus kakas) steganografi khusus untuk citra GIF, tiga diantaranya yang populer adalah *EzStego*, *S-Tool*, dan *Gifshuffle*. *EzStego* dibuat oleh Machado [3] dan akan dijelaskan lebih lanjut pada bagian di bawah, *S-*

Tool dikembangkan oleh Andy Brown [4], sedangkan *Gifshuffle* dikembangkan oleh Matthew Kwan [5]. *S-Tool* mengenkripsi pesan sebelum disisipkan ke dalam citra dengan bermacam pilihan algoritma enkripsi seperti DES dan IDEA [3]. Ide *Gifshuffle* adalah dengan mengocok warna-warna di dalam palet citra GIF, pengocokan itu tidak mempengaruhi cara citra tersebut ditampilkan. Pesan dapat dikompresi atau dienkripsi sebelum disisipkan.

Di dalam eksperimen ini dipilih algoritma EzStego sebagai program steganografi dengan pertimbangan algoritma ini paling sederhana, tidak mengenkripsi atau mengkompresi pesan sebelum disisipkan. Algoritma EzStego aslinya tidak menggunakan kunci, sehingga siapapun dapat melakukan ekstraksi pesan dari dalam citra. Untuk menambah aspek keamanan pada *EzStego*, maka telah diusulkan modifikasi algoritma *EzStego* dengan penambahan fitur kunci [6]. Kunci bertujuan untuk membangkitkan bilangan-bilangan acak sebagai lokasi penyisipan pesan. Algoritma yang lain yang juga merupakan perbaikan dari *EzStego* diusulkan oleh Fridrich di dalam [7].

Di dalam makalah ini dipresentasikan hasil-hasil eksperimen terhadap citra stego yang dihasilkan dari algoritma *EzStego* dan algoritma modifikasinya. Eksperimen dilakukan dengan bermacam tipe dan ukuran pesan. Tipe pesan ada dua macam: pesan normal dan pesan teracak. Luaran yang akan diamati adalah karakteritik artefak yang timbul sebagai akibat penyisipan pesan.

II. ALGORITMA EZSTEGO

Algoritma *EzStego* menyisipkan bit-bit pesan pada bit LSB dari indeks palet. Akibat penyisipan tersebut, indeks palet dapat bertambah satu, tetap, atau berkurang satu. Oleh karena indeks palet merupakan *pointer* ke palet warna, maka indeks yang baru (setelah penyisipan LSB) menunjuk ke warna berikutnya atau ke warna sebelumnya di palet yang tentu saja secara visual berbeda signifikan. Hal ini tentu menimbulkan degradasi warna yang membuat citra stego berbeda jauh dengan citra *cover*.

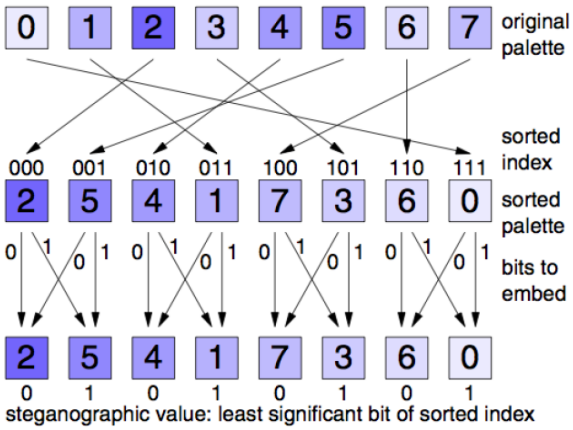
Untuk meminimalkan degradasi warna, maka langkah pertama di dalam algoritma *EzStego* adalah mengurutkan warna-warna di dalam palet sedemikian sehingga perbedaan dua warna yang bertetangga adalah minimal. Perbedaan dua warna dapat dihitung dengan rumus jarak Euclidean. Misalkan warna 1 dinyatakan sebagai vektor (R_1, G_1, B_1) dan warna 2 dinyatakan sebagai (R_2, G_2, dan B_2). Jarak Euclidean kedua warna tersebut dihitung dengan rumus:

$$d = \sqrt{(R_1 \quad R_2)^2 + (G_1 \quad G_2)^2 + (B_1 \quad B_2)^2} \qquad (1)$$

Jadi, proses pengurutan palet dilakukan dengan menghitung jarak antar warna di dalam palet, lalu mengurutkan palet berdasarkan jarak terkecil sedemikian sehingga akhirnya dua warna bertetangga memiliki jarak Euclidean yang kecil. Bit-bit pesan disisipkan pada bit LSB indeks dari palet yang terurut secara sekuensial.

Gambar 2 memperlihatkan ilustrasi penyisipan pesan di dalam EzStego. Misalkan terdapat delapan palet yang berbeda warna. Angka 0-7 menyatakan nilai pixel di dalam citra GIF. Misalkan kita ingin menyisipkan "0" pada pixel bernilai 3 yang berkoresponden dengan indeks 5 (101) pada palet yang sudah terurut. Indeks 5 menjadi indeks 6 dengan LSB indeks 3 diganti dengan bit pesan "0" (101 → 100). Oleh karena itu, warna *pixel* 3 diganti dengan warna tetangga nya (indeks 6) dari palet yang terurut. Karena perbedaan warna antara dua warna bertetangga sangat kecil, perubahan tersebut hampir tidak bisa dilihat oleh mata.

Ekstraksi pesan sangat mudah dilakukan. Sebelum proses ekstraksi, palet warna diurutkan sedemikian sehingga perbedaan dua warna yang bertetangga adalah minimal. Selanjutnya bit pesan diekstraksi dari LSB indeks palet.



Gambar 2. Proses penyisipan pesan di dalam EzStego [8]

Algoritma *EzStego* di atas tidak membutuhkan kunci sehingga kurang aman untuk penyembunyian pesan, sehingga siapapun yang mengetahui algoritmanya dapat mengekstrak pesan. Untuk meningkatkan keamanan, modifikasi algoritma *EzStego* diusulkan di dalam [6] sehingga peyisipan dan ekstraksi pesan membutuhkan kunci (*stego-key*). Kunci berguna untuk membangkitkan posisi acak di dalam citra sebagai lokasi penyisipan bit-bit pesan.

III. EKSPERIMEN VISUAL ATTACK

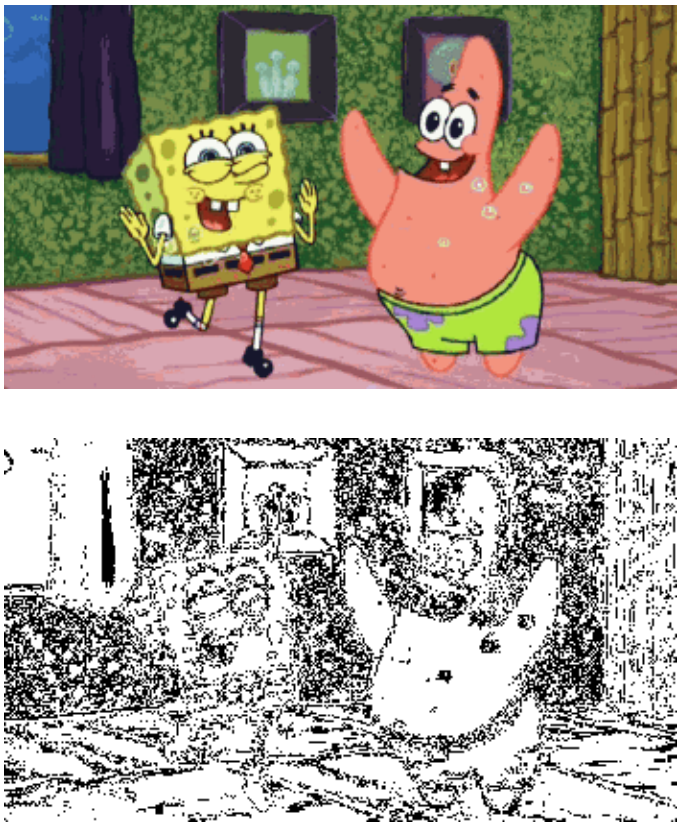
Metode *Visual Attack* dilakukan dengan mengekstrak *bitplane* LSB dari citra *suspect* (citra yang dicurigai membawa pesan rahasia). Pada citra *cover* atau citra orisinal, *bitplane* LSB adalah citra acak yang terlihat bersih. Sebaliknya pada citra stego yang sudah disisipi bit-bit pesan secara sekuensial, kita melihat artefak-artefak yang mencurigakan.

Dalam eksperimen ini, implementasi algoritma *EzStego* (sekuensial) dan algoritma modifikasinya (acak) dibuat kembali dengan menggunakan MATLAB. Citra GIF yang digunakan adalah GIF non-animasi, jadi hanya berupa citra tunggal saja.

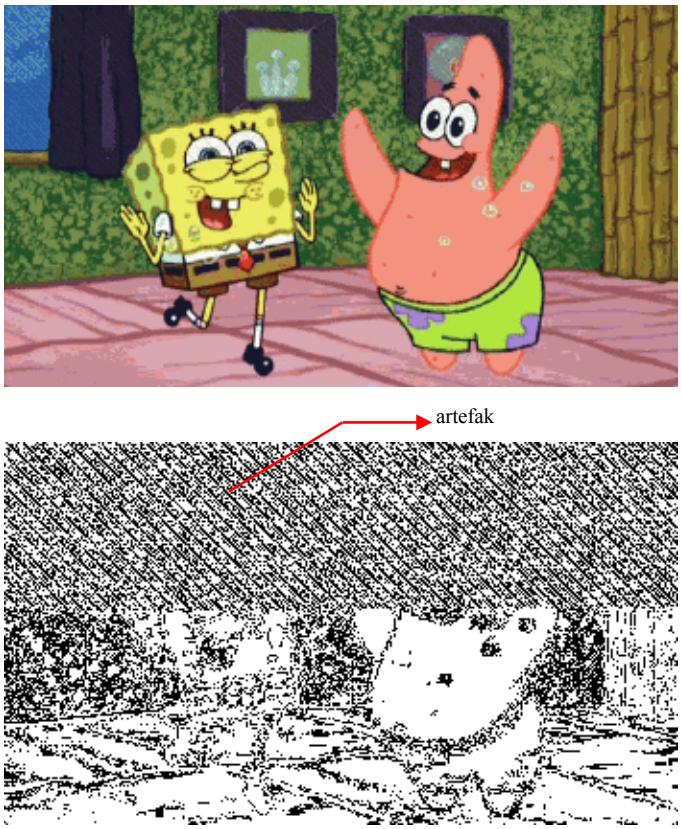
A. Penyisipan Secara Sekuensial

Gambar 3 memperlihatkan citra kartun original (*cover image*) *spongebob.gif* 351 × 200 pixel (37.8 KB) dan *bitplane* hasil ekstrak LSB-nya. Tidak ada artefak-artefak mencurigakan yang terlihat di dalam *bitplane* LSB, yang menandakan tidak ada pesan tersembunyi di dalam citra *cover* tersebut.

Sekarang citra *spongebob.gif* disisipi pesan teks (disimpan di dalam file *readme.txt*) yang berukuran 3.75 KB dengan algoritma *EzStego* sekuensial. Citra stegonya dan citra *bitplane* LSB dari citra stego tersebut diperlihatkan pada Gambar 4. Tampak pada citra *bitplane* LSB bagian atas muncul artefak baru, berupa pola-pola yang berbeda dengan bagian gambar lainnya. Artefak ini bukanlah bawaan dari citra *cover*. Artefak tersebut menandakan bahwa di lokasi artefak tersebut terdapat bit-bit pesan tersembunyi yang disisipkan secara sekuensial dari atas ke bawah. Berdasarkan pengamatan secara visual ini maka dapat dipastikan citra stego tersebut memang benar mengandung pesan tersembunyi.

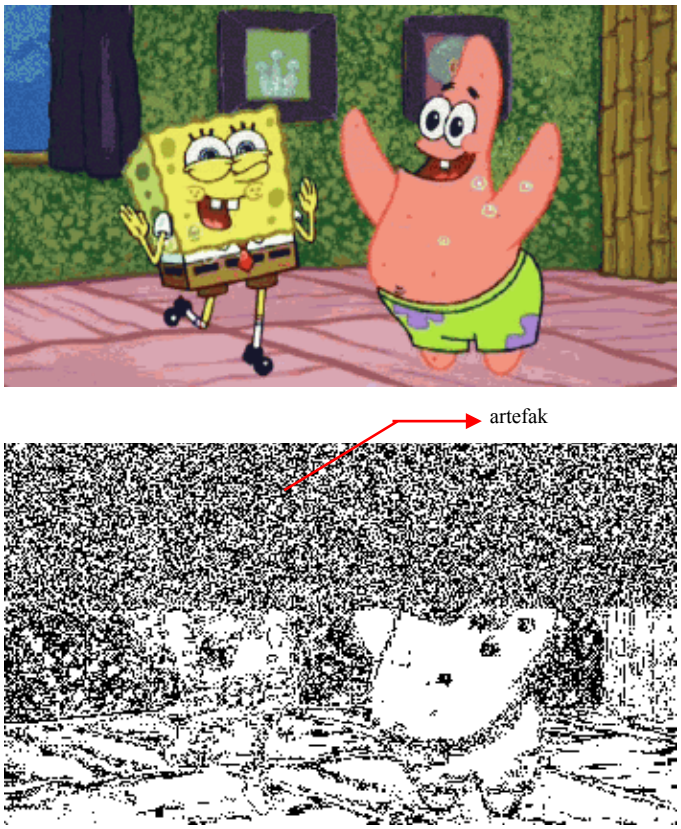


Gambar 3. Atas: *cover image* (*spongebob.gif*); Bawah: *bitplane* LSB



Gambar 4. Atas: *stego image*. Bawah: *bitplane* LSB. Pesan yang disisipkan ke dalam citra adalah pesan teks normal (tidak dienkripsi)

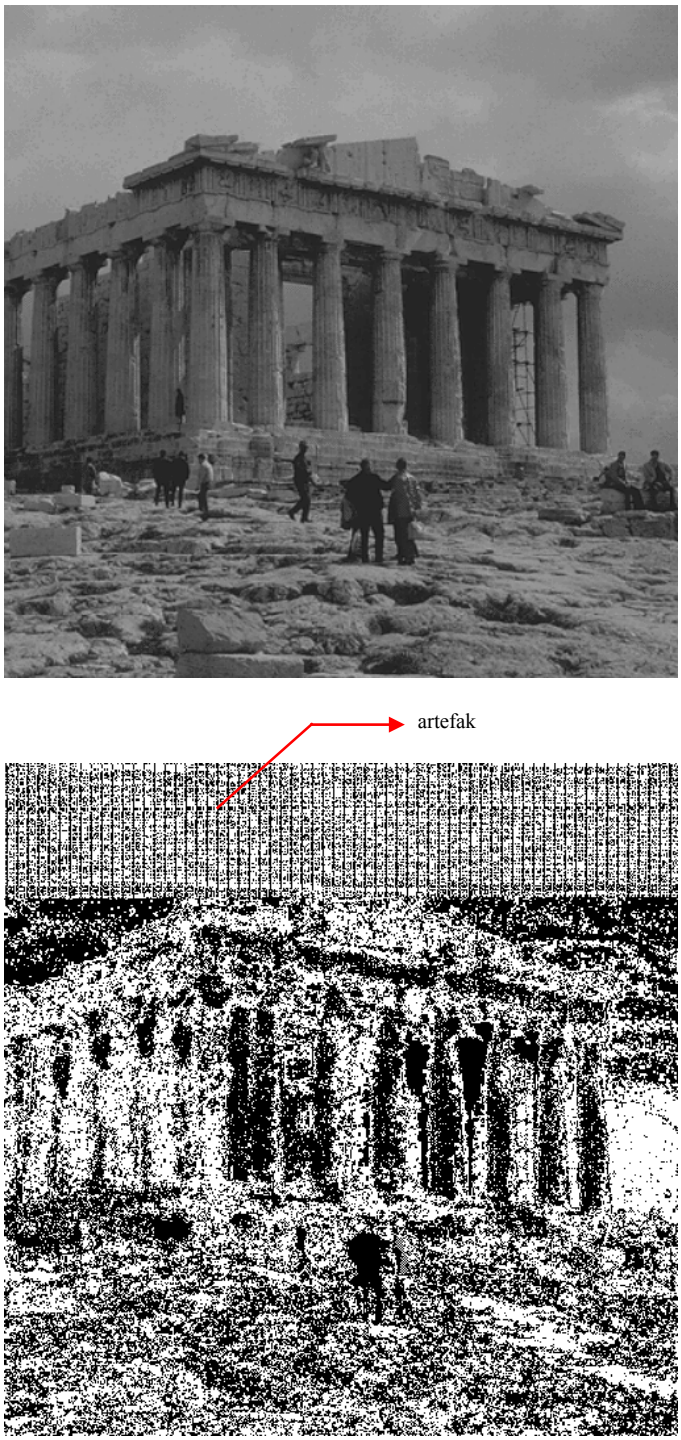
Pada eksperimen selanjutnya, citra *spongebob.gif* disisipi pesan yang berisi bit-bit acak. Bit-bit acak dapat diperoleh dengan mengenkripsi terlebih dahulu file *readme.txt* sebelum disisipkan ke dalam *spongebob.gif*. Enkripsi yang sederhana adalah dengan meng-XOR-kan bit-bit pesan dengan bit-bit yang dibangkitkan secara acak. Gambar 5 memperlihatkan citra stego dan citra *bitplane* LSB dari citra stego. Secara visual artefak yang tampak pada citra *bitplane* LSB memiliki pola acak pula. Hal ini sangat berbeda dengan artefak pada Gambar 4 yang memiliki pola-pola teratur.



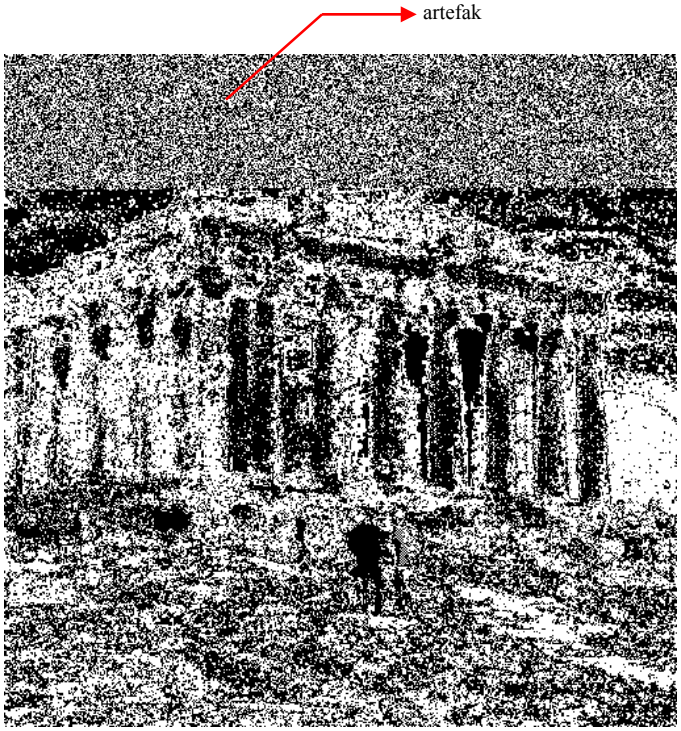
Gambar 5. Atas: *stego image*; Bawah: *bitplane* LSB. Pesan yang disisipkan ke dalam citra adalah pesan acak (hasil enkripsi)

Hal ini akan terlihat lebih jelas lagi pada eksperimen dengan citra natural bertipe *grayscale*. Gambar 6 memperlihatkan pola artefak yang timbul pada citra stego (*grayscale*) dari *roman.gif* yang berukuran 512 × 512 *pixel* (291 KB) jika disisipi pesan *father.txt* normal (tidak diacak), sedangkan Gambar 7 jika *father.txt* dienkripsi terlebih dahulu (menjadi bit-bit acak). Ukuran file *father.txt* adalah 6.38 KB.

Perbedaan artefak tersebut dapat dijelaskan sebagai berikut: pesan teks terdiri dari karakter-karakter ASCII. Kelompok karakter ASCII yang berdekatan (misalnya kelompok huruf) memiliki beberapa bit yang sama (3 bit sama pada kelompok huruf kecil). Jadi, kita akan melihat pola berulang setiap 8 bit karakter ASCII. Pada citra GIF, kita menyisipkan tiga bit pada tiga buah *pixel*, jadi kita akan melihat pola yang sama berulang setiap ((8/3)×3) *pixel*, yaitu setiap 8 *pixel* [6]. Pada pesan acak kita tidak menemukan pola demikian sehingga artefaknya juga telihat acak.

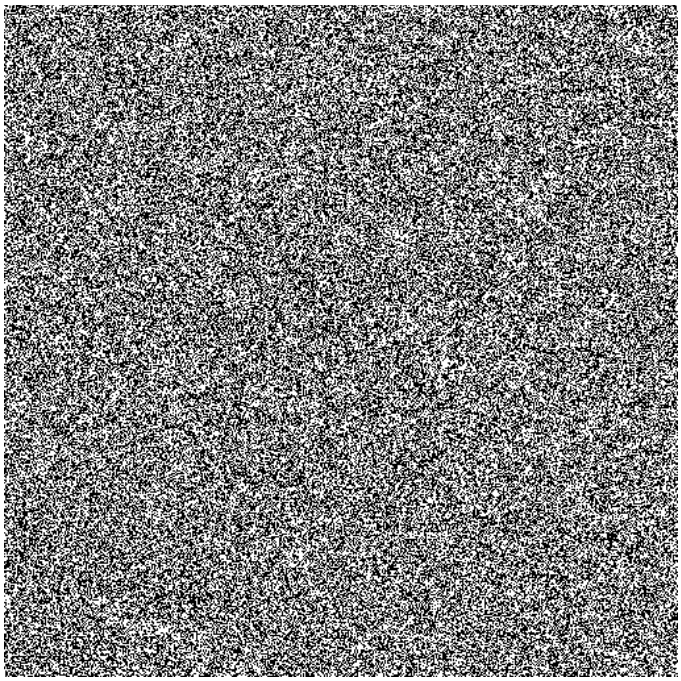


Gambar 6. Atas: *stego image* dari citra *roman.gif*; Bawah: *bitplane* LSB. Pesan yang disisipkan ke dalam citra adalah pesan teks normal (tidak dienkripsi)



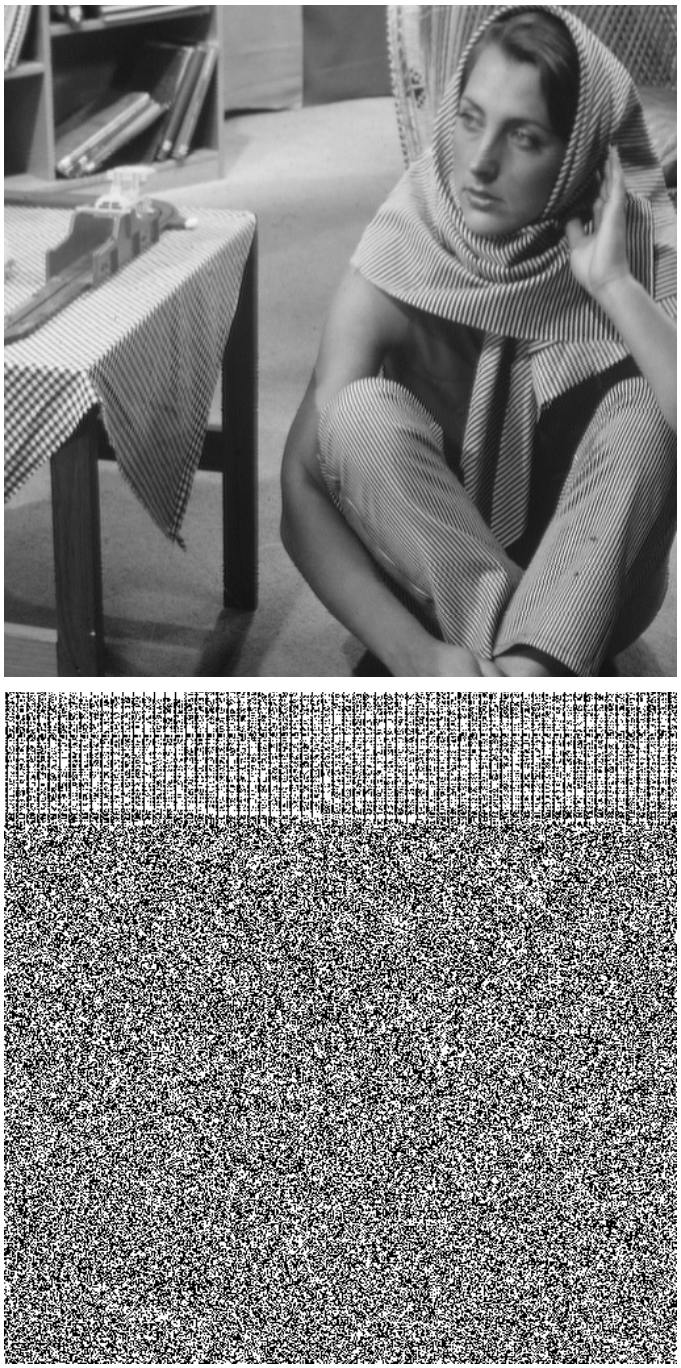
Gambar 7. Atas: *stego image* dari citra *roman.gif*; Bawah: *bitplane* LSB. Pesan yang disisipkan ke dalam citra adalah pesan acak (hasil enkripsi)

Hasil yang lebih ekstrim terlihat pada citra *barbara.gif* di bawah ini. Jika ke dalam citra *barbara.gif* disisipkan bit-bit acak (pesan terenkripsi), maka artefak yang mencurigakan tidak dapat diamati pada *bitplane* LSB dari citra stegonya. Hal ini karena artefak yang berupa pola acak sama dengan bitplane LSB yang juga terlihat acak (Gambar 8).



Gambar 8. Atas: *stego image* dari citra *barbara.gif*; Bawah: *bitplane* LSB. Pesan yang disisipkan ke dalam citra adalah pesan acak (hasil enkripsi). Artefak tidak dapat diamati karena polanya sama dengan bagian lain di dalam *bitplane* LSB

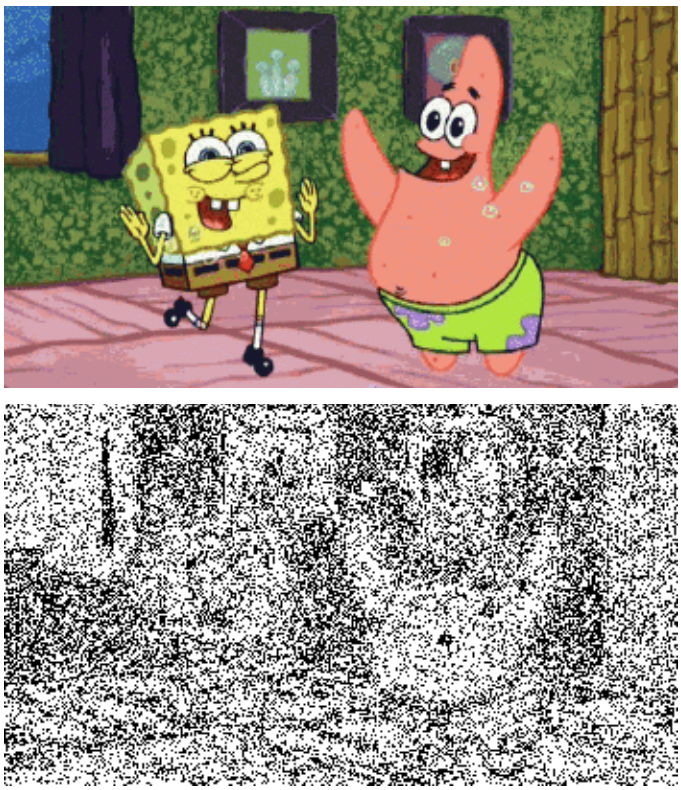
Artefak pada citra stego dari *barbara.gif* masih dapat diamati jika pesan yang disisipkan adalah pesan teks normal, seperti yang ditunjukkan pada Gambar 9. Artefak yang terbentuk berupa pola-pola khas yang teratur dan berulang pada bagian atas bitplane LSB, sangat kontras dengan baian *bitplane* LSB lainnya yang terlihat acak.



Gambar 9. Atas: *stego image* dari citra *barbara.gif*; Bawah: *bitplane* LSB. Pesan yang disisipkan ke dalam citra adalah pesan teks normal (tidak dienkripsi). Artefak yang mencurigakan dapat diamati karena polanya berbeda dengan bagian lain di dalam *bitplane* LSB

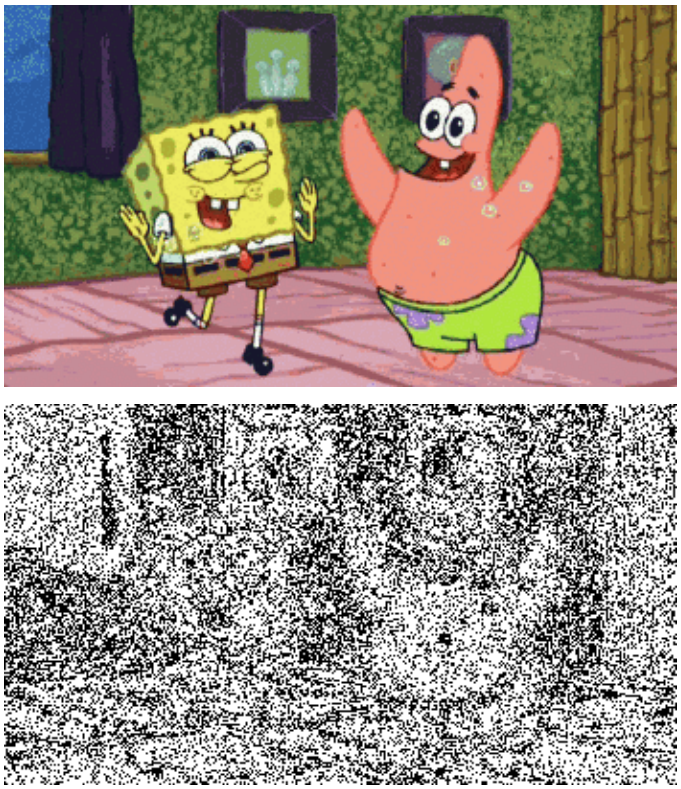
B. Penyisipan Secara Acak

Eksperimen selanjutnya menggunakan program *EzStego* yang dimodifikasi sehingga bit-bit pesan disisipkan pada lokasi yang acak di dalam citra GIF [4]. Kunci untuk pembangkitan bilangan acak adalah 1234. Gambar 8 memperlihatkan citra stego dan citra *bitplane* LSB-nya. Karena bit-bit disisipkan pada posisi acak, maka tidak dapat diamati artefak yang memiliki pola berbeda dengan bagian lainnya di dalam *bitplane* LSB.



Gambar 8. Atas: *stego image*. Bawah: *bitplane* LSB. Pesan yang disisipkan ke dalam citra adalah pesan teks normal (tidak dienkripsi)

Hasil pengamatan visual yang sama juga diperoleh jika pesan dienkripsi terlebih dahulu sehingga menghasilkan bit-bit pesan yang acak, seperti yang diperlihatkan pada Gambar 9. Tidak terlihat artefak yang menimbulkan kecurigaan sehingga pesan yang tersembunyi di dalam gambar sulit dideteksi.



Gambar 9. Atas: *stego image*. Bawah: *bitplane* LSB. Pesan yang disisipkan ke dalam citra adalah pesan acak (hasil enkripsi)

IV. Diskusi Hasil-Hasil Visual Attack

Berdasarkan hasil-hasil eksperimen di atas telah dibuktikan bahwa steganalisis dengan metode *visual attack* dapat diterapkan pada citra berformat GIF yang dicurigai mengandung pesan tersembunyi. Penyisipan bit-bit pesan secara sekuensial dengan Algoritma *EzStego* dan secara acak dengan Algoritma *EzStego* modifikasi memberikan hasil pengamatan yang berbeda. Pada penyisipan bit-bit pesan secara sekuensial, artefak yang mencurigkan dapat dilihat secara kasat mata pada *bitplane* LSB dari citra stegonya. Artefak di dalam citra stego memiliki pola-pola yang dapat dibedakan dengan bagian citra lainnya, sehingga dapat menjadi indikasi bahwa citra dicurigai mengandung pesan tersembunyi.

Sebaliknya, jika pesan disisipkan pada posisi yang acak di dalam citra, maka artefak yang timbul sangat sulit diamati karena tidak dapat dibedakan dengan bagian lainnya di dalam *bitplane* SLB.

Pola-pola artefak di dalam *bitplane* LSB juga bergantung pada keacakan pesan. Pesan yang berupa bit-bit acak (misalnya pesan hasil enkrips) memiliki pola artefak yang juga acak. Sebaliknya pesan normal (tidak acak) memiliki pola-pola artefak yang khas yang disebabkan oleh kesamaan beberapa bit pada representasi binernya, sebagaimana sudah sudah dijelaskan pada bagian III di atas.

Metode *visual attack* di atas juga dapat diterapkan untuk melakukan steganalisis citra dengan format *bitmap* (BMP). Algoritma steganografi yang umum untuk citra *bitmap* adalah algoritma modifikasi LSB. Algoritma modifikasi LSB menyisipkan pesan pada bit-bit LSB dari *pixel-pixel bitmap*. Dengan mengekstrak bit-bit LSB dari citra *suspect*, kita dapat mengamati secara visual pola-pola artefak yang muncul di dalam *bitplane* LSB-nya.

V. Kesimpulan

Di dalam makalah ini telah dipresentasikan hasil-hasil eksperimen steganalisis dengan metode *visual attack* pada citra GIF yang disisipi pesan dengan algoritma *EzStego* dan algoritma modifikasinya. Secara umum hasil-hasil *visual attack* sangat bergantung pada natural dari bit-bit pesan (acak atau tidak acak), pola penyisipan pesan (sekuensial atau acak), kartakterisik citra GIF (citra natural atau citra hasil grafika komputer), dan subyektifitas pengamat. *Visual attack* dapat dijadikan sebagai pre-steganalisis. Hasil-hasil yang lebih akurat dapat diperoleh dengan metode steganalisis yang lebih kualitatif seperti *statistical attack.*

Daftar Pustaka

[1] Philip Batemen (2008), *Image Steganography and Steganalysis*, University of Surrey, 2008.

[2] Wen Chen (2005), *Study of Steganalysis Method*, New Jersey Institute of Technology.

[3] R. Machado, EZStego, http://www.stego.com

[4] N.F. Johnson and S. Jajodia (1998), 'Exploring Steganography: Seeing the Unseen, George Mason University', IEEE Computers.

[5] The Gifshuffle Hompage, http://www.darkside.com.au/gifshuffle/

[6] Rinaldi Munir (2015), *Chaos-based Modified "EzStego" Algorithm for Improving Security of Message Hiding in GIF Image*, Proceeding of IC3INA 2015, Bandung.

[7] J. Fridrich (1999), A New Steganographic Method for Palette Images, IS&T PICS : Savannah, Georgia, USA.

[8] A. Westfeld and A. Pfitzmann (1999). "Attack on Steganographic System", Lecture Notes in Computer Sciences, vol. 1768, pp. 61-76.

[9] Guillermito (2004), A few tools to discover hidden data, http://www.guillermito2.net/stegano/tools/.