IMPLEMENTASI MULTITHREADING PROGRAMMING CONCEPT UNTUK EFISIENSI PROSES STEGANOGRAFI METODE LSB

Paskalis Andrianus Nani
Program Studi Teknik Informatika Universitas Katolik Widya Mandira
Jl. Jend. Achmad Yani No.50-52 Kupang 85225 - NTT
email : paskalisnani@gmail.com

Abstrak

Steganografi metode Least Significant Bit (LSB) adalah metode peyembunyian pesan rahasia dengan cara menyisipkan pesan ke dalam bit rendah (least significant bit) pada data piksel yang menyusun file citra penampung. Steganografi menggunakan metode LSB merupakan teknik yang relatif mudah dimengerti. Namun, proses enkripsi pesan yang berukuran besar membutuhkan waktu yang tidak sedikit. Semakin banyak jumlah bit yang akan diproses, maka semakin banyak pula waktu yang dibutuhkan. Sumberdaya komputasi dapat digunakan semaksimal mungkin agar proses komputasi yang panjang dapat dipersingkat.

Penelitian ini mengimplementasikan konsep pemrograman multithread pada proses steganografi metode LSB dengan tujuan meningkatkan efisiensi proses steganografi dengan memanfaatkan bagian prosesor yang idle.

Hasil penelitian ini menunjukkan bahwa penggunaan multithreading programming concept dalam proses steganografi metode LSB sangat efektif dalam meningkatkan efisiensi proses komputasi yang dikerjakan. Hal ini ditunjukkan oleh selisih waktu yang sangat signifikan yaitu berkisar antara 438 sampai dengan 8.568 milliseconds.

Kata Kunci :
Multithreading, Steganografi, Least Significant Bit.

1. Pendahuluan

Kemajuan teknologi informasi memiliki banyak keuntungan dalam kehidupan manusia. Namun demikian, aspek negatifnya juga banyak terjadi, antara lain kejahatan komputer, yang meliputi pencurian, penipuan, pemerasan dan lain sebagainya. Jatuhnya informasi ke tangan pihak lain (misalnya pihak lawan bisnis) bisa menimbulkan kerugian bagi pemilik informasi. Untuk itu, keamanan dari informasi harus terjamin dalam batas tertentu.

Salah satu cara yang dapat dilakukan untuk menyembunyikan informasi adalah menggunakan teknik steganografi. Steganografi merujuk pada sebuah seni untuk menyembunyikan informasi. Steganografi menyembunyikan informasi rahasia di dalam informasi lain sehingga informasi tersebut tidak dapat diketahui oleh orang lain yang tidak berkepentingan[4].

Salah satu metode steganografi yang paling sederhana adalah metode *Least Significant Bit* (LSB)[1,2,3,4,6,8,13]. Metode ini meyembunyikan pesan rahasia dengan cara menyisipkan pesan ke dalam bit rendah (*least significant bit*) pada data piksel yang menyusun *file* citra penampung[9]. Metode ini merupakan teknik substitusi pada steganografi.

Pada penelitian sebelumnya[10,11] telah dilakukan implementasi algoritma *Blowfish* untuk mengenkripsi informasi yang akan dikirim lalu menyamarkan informasi tersebut menggunakan teknik steganografi. Dari hasil penelitian tersebut, dapat disimpulkan bahwa dengan melakukan enkripsi terlebih dahulu tehadap pesan yang ingin disisipkan, keamanan pesan dapat semakin terjaga walaupun pesan yang terdapat dalam stego dapat dengan mudah diekstrak dengan *software steganalysis* yang beredar luas di dunia maya saat ini. Namun, aspek efisiensi proses belum terlalu diperhatikan. Akibatnya jika pesan yang akan disembunyikan berukuran besar, maka mesin (komputer) akan melalukan proses *looping* atau perulangan yang sangat panjang.

Penelitian ini akan mengimplementasikan konsep pemrograman *multithread* pada proses steganografi metode LSB dengan tujuan meningkatkan efisiensi proses steganografi dengan memanfaatkan bagian prosesor yang *idle*.

2. Tinjauan Pustaka

Multithreading

Multithreading adalah cara pengeksekusian yang mengizinkan beberapa *thread* terjadi dalam sebuah proses, saling berbagi sumber daya tetapi dapat dijalankan secara independen [5].

Pada dasarnya, *task multithreading* ditujukan untuk membuat pengunaan *resource* komputer secara lebih bijak, dengan memungkinkan *resource* yang sedang digunakan dapat digunakan oleh berbagai macam varian kecil dari proses yang sama secara simultan. Konsep dasar dari *multithreading* telah berada dalam waktu yang cukup lama, namun baru populer pada dekade tahun 90-an [7].

Dengan memungkinkan seubah program untuk menangani lebih dari satu *task* dengan model *multithreading*, sistem tidak perlu mengikuti dua program terpisah untuk menginisiasi dua proses terpisah

dan harus menggunakan *file* yang sama secara bersamaan.

Keuntungan dari sistem yang menerapkan *multithreading* dapat dikategorikan menjadi 4 bagian: [7]

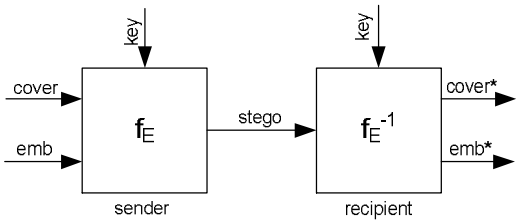
a. Responsif. Aplikasi interaktif menjadi tetap responsif meskipun sebagian dari program sedang diblok atau melakukan operasi lain yang panjang. Umpamanya, sebuah *thread* dari *web browser* dapat melayani permintaan pengguna sementara *thread* yang lain berusaha menampilkan gambar.
b. Berbagi sumber daya. Beberapa *thread* yang melakukan proses yang sama akan berbagi sumber daya. Keuntungannya adalah mengizinkan sebuah aplikasi untuk mempunyai beberapa *thread* yang berbeda dalam lokasi memori yang sama.
c. Ekonomis. Pembuatan sebuah proses memerlukan pengalokasian memori dan sumber daya. Alternatifnya adalah dengan menggunakan *thread*, karena *thread* membagi memori dan sumber daya yang dimilikinya sehingga lebih ekonomis untuk membuat *thread* dan *context switching thread*. Akan susah mengukur perbedaan waktu antara *thread* dan *switch*, tetapi secara umum pembuatan dan pengaturan proses akan memakan waktu lebih lama dibandingkan dengan *thread*.
d. Utilisasi arsitektur multiprosesor. Keuntungan dari *multithreading* dapat sangat meningkat pada arsitektur multiprosesor, dimana setiap *thread* dapat berjalan secara paralel di atas prosesor yang berbeda. Pada arsitektur prosessor tunggal, CPU menjalankan setiap *thread* secara bergantian tetapi hal ini berlangsung sangat cepat sehingga menciptakan ilusi paralel, tetapi pada kenyataanya hanya satu *thread* yang dijalankan CPU pada satu-satuan waktu.

Steganografi

Steganografi (*steganography*) adalah teknik menyembunyikan data rahasia di dalam wadah (media) digital sehingga keberadaan data rahasia tersebut tidak diketahui oleh orang[9].

Penyembunyian data rahasia ke dalam citra digital akan mengubah kualitas citra tersebut. Kriteria yang harus diperhatikan dalam penyembunyian data adalah: [15]

a. *Fidelity*. Mutu citra penampung tidak jauh berubah. Setelah penambahan data rahasia, citra hasil steganografi masih terlihat dengan baik. Pengamat tidak mengetahui kalau di dalam citra tersebut terdapat data rahasia.
b. *Robustness*. Data yang disembunyikan harus tahan terhadap manipulasi yang dilakukan pada citra penampung (seperti pengubahan kontras, penajaman, pemampatan, rotasi, perbesaran gambar, pemotongan (cropping), enkripsi, dan sebagainya). Bila pada citra dilakukan operasi pengolahan citra, maka data yang disembunyikan tidak rusak.
c. *Recovery*. Data yang disembunyikan harus dapat diungkapkan kembali (*recovery*). Karena tujuan steganografi adalah data hiding, maka sewaktu-waktu data rahasia di dalam citra penampung harus dapat diambil kembali untuk digunakan lebih lanjut.



Gambar 1. Steganographic System[14]

Gambar 1 di atas menunjukkan sebuah sistem steganografi umum dimana di bagian pengirim pesan(*sender*) dilakukan proses *embedding*(f_E) pesan yang hendak dikirim secara rahasia(*emb*) ke dalam data *cover* sebagai tempat menyimpannya(*cover*), dengan menggunakan kunci tertentu(*key*), sehingga dihasilkan data dengan pesan tersembunyi di dalamnya(*stego*).

Di bagian penerima pesan (*recipient*), dilakukan proses *extracting*(f_E^{-1}) pada stego untuk memisahkan pesan rahasia (*emb**) dan data penyimpan (*cover**) tadi menggunakan kunci (*key*) yang sama seperti pada proses *embedding*.

Least Significant Bit

Metode ini meyembunyikan pesan rahasia dengan cara menyisipkan pesan ke dalam bit rendah (*least significant bit*) pada data *pixel* yang menyusun file citra penampung[9].

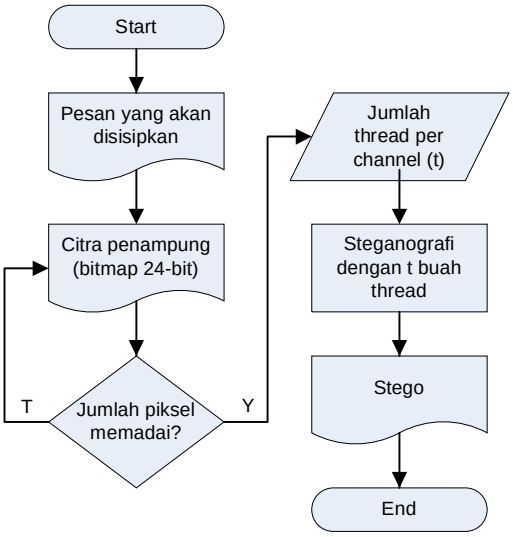
Pada susunan bit di dalam sebuah byte (1 byte = 8 bit), ada bit yang paling berarti (*most significant bit* atau MSB) dan bit yang paling kurang berarti (*least significant bit* atau LSB)[12].

Bit yang cocok untuk diganti adalah bit LSB, sebab perubahan tersebut hanya mengubah nilai *byte* satu lebih tinggi atau satu lebih rendah dari nilai sebelumnya. Misalkan *byte* tersebut menyatakan warna merah, maka perubahan satu bit LSB tidak mengubah warna merah tersebut secara berarti. Lagi pula, mata manusia tidak dapat membedakan perubahan yang kecil tersebut. Sebuah file bitmap (BMP) mampu menyembunyikan file yang berukuran besar [13].

3. Metode Penelitian

Penelitian ini akan mengimplementasikan konsep pemrograman *multithread* pada proses steganografi metode LSB dengan tujuan meningkatkan efisiensi proses steganografi dengan memanfaatkan bagian prosesor yang *idle*.

Proses implementasi *multithread* pada steganografi dapat dilihat pada Gambar 2 di bawah ini:



Gambar 2. Diagram alir implementasi multithread pada proses steganografi

Diagram alir di atas dapat dijelaskan sebagai berikut: pertama-tama kita menginput pesan yang akan disisipkan lalu memilih citra penampung. Citra penampung haruslah citra RGB dengan format bitmap 24-bit. Kemudian, sistem akan memeriksa apakah citra tersebut memiliki jumlah piksel yang dapat menampung seluruh pesan yang akan disisipkan.

Setelah itu, kita menginputkan jumlah *thread* yang akan digunakan pada masing-masing *channel* warna (R, G dan B). Jika jumlah thread yang di-*input* adalah 2, maka channel R, G dan B masing-masing akan diproses menggunakan 2 buah *thread* jadi total ada 6 buah thread yang bekerja. Jika jumlah thread yang di-*input* hanya 1 maka hanya akan 3 thread yang dibuat. Jadi secara default, tiga buah thread akan dibentuk untuk menangani masing-masing *channel* warna.

Selanjutnya, bit-bit pesan akan dipecah (dibagi) sesuai dengan jumlah *thread* lalu proses steganografi dilakukan.

Piksel terakhir pada channel B akan digunakan untuk menyimpan informasi jumlah *thread*, jumlah bit pesan per *thread* dan jumlah piksel yang ditangani oleh sebuah *thread*. Informasi ini akan sangat dibutuhkan saat proses ekstraksi pesan.

Jumlah piksel yang akan diproses oleh setiap *thread* dapat diperoleh menggunakan rumus di bawah ini:

$$p = round\left(\frac{m \times n}{t}\right) \quad \text{..... (1)}$$

Secara matematis, masing-masing *thread* (asumsi terdapat 3 buah thread) pada setiap *channel* warna akan melakukan proses steganografi dengan persamaan sebagai berikut:

$$f(x_1) = \sum_{a=0}^{p} S_a \quad \text{..... (2)}$$

$$f(x_2) = \sum_{a=p+1}^{2p} S_a \quad \text{..... (3)}$$

$$f(x_3) = \sum_{a=2p+1}^{3p} S_a \quad \text{..... (4)}$$

4. Hasil Dan Pembahasan

Persiapan

Terdapat 6(enam) *file* yang akan dipakai dalam proses steganografi ini dengan komposisi 5(lima) *file* yang akan disembunyikan dan sebuah *file* citra berformat bitmap sebagai *cover*. Keenam file yang digunakan dapat dilihat pada Tabel 1 berikut:

Tabel 1. File-file yang digunakan

No	Nama File	Ukuran (kb)
1	File1_Image.jpeg	733
2	File2_Image.jpeg	943
3	File3_Image.jpeg	1.204
4	File4_Audio.mp3	3.209
5	File5_Audio.mp3	4.167
6	Host.bmp	35.645

File "Host.bmp" memiliki resolusi 4.272 x 2.848 piksel atau setara dengan 12.166.656 piksel. *File* ini yang nantinya akan menjadi *cover*.

Implementasi *Thread*

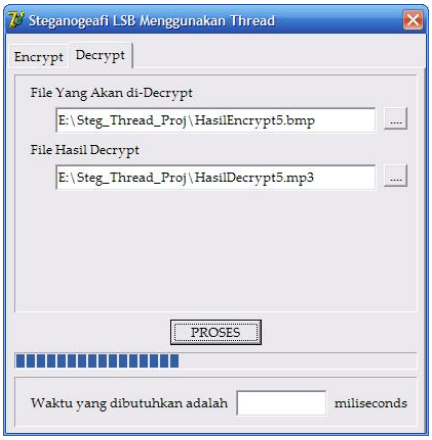
Aplikasi yang dikembangkan ini menggunakan *Borland Delphi 7*. Untuk menghitung lama proses, digunakan fungsi *MilliSecondsBetween()*. Berikut potongan kode yang digunakan untuk menghitung lama proses:

```
awal:=now;
Enkripsi(citra,file,tujuan,jumThread);
akhir:=now;
lama:=MilliSecondsBetween(awal,akhir);
```

Gambar 3 di bawah ini merupakan tampilan aplikasi saat melakukan proses penyembunyian pesan "File5_Audio.mp3" menggunakan 3 (tiga) buah *thread* dan hasilnya disimpan sebagai "HasilEncrypt5.bmp".

Gambar 3. Tampilan Proses Enkripsi

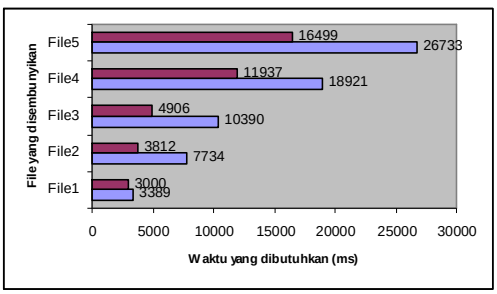
Berikut ini (Gambar 4) adalah *screenshot* saat aplikasi menjalankan proses dekripsi terhadap *file* ”HasilEncrypt5.bmp” yang hasilnya disimpan sebagai ”HasilDecrypt5.mp3”.



Gambar 4. Tampilan Proses Dekripsi

Pembahasan

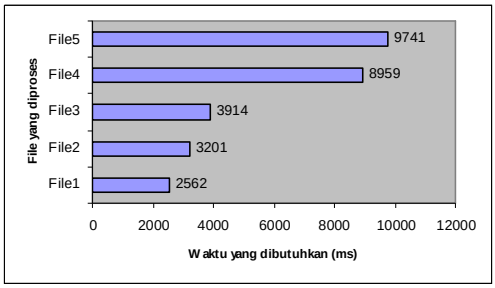
Grafik berikut ini (Gambar 3) menunjukkan perbandingan lama waktu proses yang dibutuhkan tanpa menggunakan *thread* untuk kelima *file* yang digunakan.



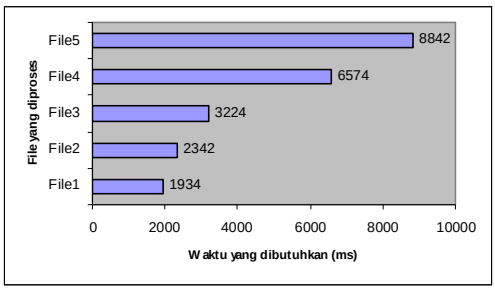
Ket: Proses Substitusi Bit saja
Proses Substitusi Bit dan Konstruksi Citra Baru

Gambar 3. Grafik perbandingan lama waktu proses steganografi lima file tanpa thread

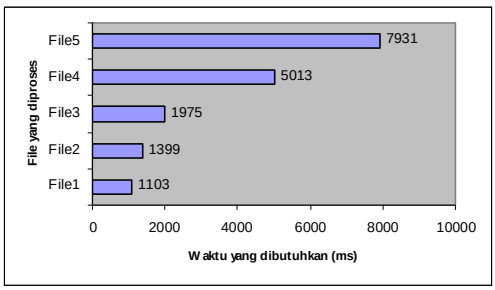
Setelah aplikasi dimodifikasi dengan menambahkan fitur *multithread* pada proses steganografi, maka diperoleh hasil sebagai berikut untuk penggunaan 3, 6 dan 9 *thread* (Gambar 4, 5 dan 6):



Gambar 4. Grafik perbandingan lama waktu proses steganografi lima file dengan 1 thread per channel warna



Gambar 5. Grafik perbandingan lama waktu proses steganografi lima file dengan 2 thread per channel warna



Gambar 6. Grafik perbandingan lama waktu proses steganografi lima file dengan 3 thread per channel warna

Dari hasil percobaan yang dilakukan terhadap kelima *file* di atas, dapat dilihat bahwa semakin banyak *thread* yang digunakan maka semakin singkat waktu yang dibutuhkan untuk proses steganografi. Hal ini ditunjukkan dengan selisih waktu yang dibutuhkan yaitu antara 438 dan 8.568 *milliseconds.*

5. Kesimpulan

Steganografi menggunakan metode LSB merupakan teknik yang relatif mudah dimengerti. Namun, proses enkripsi pesan yang berukuran besar membutuhkan waktu yang tidak sedikit. Semakin banyak jumlah bit yang akan diproses, maka semakin banyak pula waktu yang dibutuhkan. Sumberdaya komputasi dapat digunakan semaksimal mungkin agar proses komputasi yang panjang dapat dipersingkat.

Berdasarkan hasil penelitian ini, dapat disimpulkan bahwa penggunaan *thread* dalam pemrograman pada umumnya dan proses steganografi

khususnya sangat efektif dalam meningkatkan efisiensi proses komputasi yang dikerjakan.

Teknik yang sama perlu dicoba untuk diimplementasikan pada perangkat *mobile* yang support *multithread*.

Daftar Pustaka

[1] Aditya, Yogie, dkk, 2010, *Studi Pustaka Untuk Steganografi Dengan beberapa Metode*, Prosiding Seminar Nasional Aplikasi Teknologi Informasi Yogyakarta 9 Juni 2010.

[2] Alatas, Putri, 2009, *Implementasi Teknik Steganografi Dengan Metode LSB Pada Citra Digital*, Jakarta, Universitas Gunadarma.

[3] Anggaraini, Ema Utami, 2007, *Analisis Penyisipan Data pada Citra Bitmap Menggunakan Metode Bit Plane Complexity Segmentation*, Prosiding Seminar Nasional Teknologi Yogyakarta 24 November 2007.

[4] B. Karthikeyan, V. Vaithiyanathan, B. Thamotharan, M. Gomathymeenakshi and S. Sruti, 2012, *LSB Replacement Stegnography in an Image using Pseudorandomised Key Generation*, Research Journal of Applied Sciences, Engineering and Technology, vol.4, no.5, hal. 491-494, Maret 2012, ISSN: 2040-7467.

[5] Bo I Sandn, 2011, *Design Of Multi-threading Software: The Entity-Life Modelling Approach*, IEEE Computer Society Pr.

[6] Hakim, Muhammad, *Studi Dan Implemen-tasi Steganografi Metode LSB Dengan Preprocessing Kompresi Data Dan Ekspansi Wadah*, Bandung, STEI ITB.

[7] Indriyawan, Eko, 2008, *Membangun Sistem Andal Dengan Delphi*, Yogyakarta, ANDI.

[8] Krisnawati, 2008, *Metode Least Significant Bit (LSB) dan End Of File (EOF) Untuk Menyisipkan Teks ke Dalam Citra Grayscale*, Prosiding Seminar Nasional Informatika @ UPN Veteran Yogyakarta 24 Mei 2008.

[9] Munir, Rinaldi, 2004, *Pengolahan Citra Digital Dengan Pendekatan Algoritmik*, Bandung, INFORMATIKA.

[10] Nani, Paskalis Andrianus, 2011, *Penerapan Enkripsi Algoritma Blowfish Pada Proses Steganografi Metode EOF*, Prosiding SNATIKA 2011, hal. 236-241, ISSN: 2089-1083.

[11] Nani, Paskalis Andrianus, 2012, *Implemen-tasi Steganografi Menggunakan Algoritma Blowfish dan Least Significant Bit*, Prosiding KNSI 2012, ISBN: 978-602-98768-0-2.

[12] Pitas, Ioannis, 1993, *Digital Image Processing Algorithms*. UK, Prentice-Hall.

[13] Pratiksha Y. Pawar and S. H. Gawande, 2012, *M-Commerce Security Using Random LSB Steganography and Cryptography*, International Journal of Machine Learning and Computing, Vol.2, No.4, Agustus 2012, hal. 427-430.

[14] Putra, Dharma, 2010, *Pengolahan Citra Digital*, Yogyakarta, ANDI.

[15] Sutoyo, dkk. 2009, *Teori Pengolahan Citra Digital*, Yogyakarta, ANDI.

Biodata Penulis

Paskalis Andrianus Nani, memperoleh gelar Sarjana Teknik (ST), Program Studi Teknik Informatika Unika Widya Mandira, lulus tahun 2008. Aktif menulis pada seminar-seminar bertaraf Nasional. Saat ini bertugas sebagai Kepala Bagian Pelayanan Konten Online UPT TI Unika Widya Mandira.