ISSN: 2598-8565 (media cetak)
ISSN: 2620-4339 (media online)

PENYEMBUNYIAN PESAN *CHYPER* KEDALAM GAMBAR BERWARNA

[1]Frinto Tambunan, [2]Eviyanti Novita Purba
[1]Universitas Potensi Utama, [2]Universitas Methodist Indonesia
Email: [1]frintoaja@gmail.com, [2]eviyantinovita@gmail.com

DOI: https://doi.org/10.46880/jmika.Vol4No1.pp22-26

ABSTRAK

Pertukaran informasi menjadi hal yang sangat penting di era kemajuan saat ini. Hal ini didukung dengan teknologi informasi dan komunikasi yang terus berkembang pesat. Akan tetapi kemudahan mendapatkan informasi juga memberikan ancaman. Masalah keamanan dan kerahasiaan data merupakan salah satu aspek penting dari suatu informasi. Dengan berkembangnya teknik pengambilan informasi secara ilegal, banyak orang yang mencoba untuk mengakses informasi yang bukan haknya. Oleh karena itu diperlukan suatu sistem pengamanan data yang bertujuan untuk meningkatkan keamanan dan melindungi data sehingga tidak dimengerti oleh pihak yang tidak berwenang, Dengan menggabungkan metode kriptografi dan metode steganogrfi diharapkan mampu memenuhi kebutuhan akan keamanan dan melindungi data. Setelah dilakukan penelitian ini baik secara matematis dan implementasi kedalam aplikasi didapatkan sebuah hasil yang dapat di pertanggung jawabkan dimana hasil dari data asli yang di ubah ke bentuk lain melalui metode kriptografi kemudian hasilnya disisipkan kedalam sebuah gambar. Dari hasil pengamatan yang dilakukan terhadap hasil dari gambar ternyata kapasitas gambar tidaklah signifikan bertambah. Dengan kata lain penyisipan data kedalam gambar terlihat sempurna.

Kata Kunci: RC4, Kriptografi, LSB, Steganografi

PENDAHULUAN

Penggunaan data saat ini sangat berkembang terutama pada keseharian manusia, dimana penggunaan data ini adalah penggunaan data digital, pada perkembangan data digital sekarang ini mendukung untuk bertukar informasi yang sejalan dengan perkembangan *hardware* teknologi. Pada perkembangan yang sangat pesat ini, terkadang data yang sangat rahasia jatuh ke tangan yang tidak berwenang. Untuk memenuhi kebutuhan akan keamanan dan kerahasiaan data itu penulis mencoba memberikan pendekatan secara matematis yang di implementasikan melalui *kriptografi* dan *steganografi* sehigga dengan menggabungkan konsep tersebut dapat memenuhi akan kebutuhan masalah itu.

Dalam buku William Stallings (2017) Salah satu ilmu pengamanan data yang sangat terkenal adalah *kriptografi* dan *steganografi*. *Kriptografi* adalah studi tentang teknik-teknik matematika yang berhubungan dengan aspek-aspek pengamanan informasi seperti kerahasiaan (*confidentiality*), keutuhan data (*data integrity*), autentikasi entitas (*entity authentication*), dan autentikasi asal data (*data origin authentication*). Ilmu sekaligus seni untuk menjaga kerahasiaan data, atau informasi dengan cara menyamarkannya menjadi bentuk tersandi yang tidak mempunyai makna sama sekali. Dalam *kriptografi*, terdapat 2 proses utama, *enkripsi* dan *dekripsi*. *Enkripsi* adalah proses penyandian pesan asli atau *plainteks* menjadi *cipherteks* (teks tersandi). Sedangkan *dekripsi* adalah proses penyandian kembali *cipherteks* menjadi *plainteks*.

Sedangkan *steganografi* merupakan teknik yang bertujuan untuk memenuhi aspek kerahasiaan sebuah pesan sebagai teknik penyisipan pesannya. Metode ini mempunyai keunggulan yaitu pada tingkat keamanan yang tinggi untuk *steganografi* pada citra berbasis palet (*palette – based image*), yaitu citra *JPG*. (Putra, 2010)

Menggunakan fungsi *CheckSum*. *CheckSum* bekerja dengan cara menjumlahkan seluruh *byte* yang ada pada file. Fungsi ini digunakan ketika dilakukan proses pemasukan data *chyper* kedalam gambar kemudian dihitung jumlah *byte*-nya. Kemudian membandingkan nilai selisih antara gambar asli dengan gambar yang telah disisipi oleh *cyper*.

METODE PENELITIAN

Algoritma RC4

Terdapat dua proses untuk memperkuat aliran kunci algoritma RC4 yaitu *Key Scheduling Algorithm* (KSA) dan *Pseudo-Random Generator Algorithm* (PRGA). *Key Scheduling Algorithm* (KSA) merupakan tahapan pertama dalam pemberian nilai

awal berdasarkan kunci enkripsi. Letak dari nilai awal tersebut berupa *array* dengan representasi permutasi 256 *byte* (dengan indeks 0 sampai dengan 255) dinamakan *array* S. Menggunakan rentang tersebut karena RC4 mengenkripsi pada mode *byte* (255=2^8 dan 8 *bit* = 1 *byte*). Artinya maksimal panjang kunci yang dapat tersimpan pada *array* U adalah 256 karakter. Permutasi terhadap nilai *array* S dilakukan dengan *pseudo-code* Setelah *keystream* tercipta, kemudian *keystream* tersebut dimasukkan dalam operasi *XOR* dengan *plaintext*.

Least Significant Bit (LSB)

Metode Least Significant Bit (LSB) merupakan metode yang tidak terlalu berarti, penyimpanan pesan pada *cover object* juga lumayan besar. awal metode ini adalah bilangan biner yaitu angka 0 dan 1, karena pada data *digital* merupakan susunan angka 0 dan 1 maka proses penerapannya menjadi sangat mudah. selanjut metode ini sangat berhubungan erat dengan ukuran 1 *bit* dan ukuran 1 *byte* dimana 1 *byte* data terdiri dari 8 *bit* data dan *bit* pada posisi paling kanan disebut dengan LSB (Andrian, 2013). Metode LSB diganti dengan *bit* yang akan disembunyikan. Karena *bit* yang akan diubah hanya *bit* yang paling akhir, maka *stego image* yang dihasilkan tidak terlalu berubah secara signifikan dan tidak dapat dilihat dengan mata secara langsung.

HASIL DAN PEMBAHASAN

Ini adalah cara perhitungan algoritma RC4 dengan mode 4 *byte* (untuk lebih men- sederhanakan dalam perhitungan manual) serta dalam kebutuhan waktu yang sangat terbatas. State *Box* dengan panjang 4 *byte*, dengan S[0] = 0, S[1] = 1, S[2] = 2 dan S[3] = 3 sehingga *array* S menjadi: 0 1 2 3

Inisialisasi 4 *byte* kunci *array*, K. Misalkan kunci Ulang kunci sampai memenuhi seluruh adalah 1 2 3 4, sehingga *array* K berisi 1 2 3 4 dan akan mencoba untuk mengenkripsi kata KAMU.

Inisialisasi i dan j dengan 0 selanjutnya akan dilakukan KSA (*Key-scheduling algorithm*) agar terbentuk *state-array* yang acak. Penjelasan iterasi selanjutnya akan dijelaskan sebagai berikut:

Iterasi 1
i = 0
j = (0 + S[0] + K [0 mod 4]) mod 4
= (0 + 0 + 1) mod 4 = 1 Swap (S[0],S[1])
Hasil Array S 1 0 2 3

Iterasi 2
i = 1
j = (1 + S[1] + K [1 mod 4]) mod 4
= (1 + 0 + 2) mod 4 = 3 Swap (S[1],S[3])
Hasil Array S 1 3 2 0

Iterasi 3
i = 2
j = (3 + S[2] + K [2 mod 4]) mod 4
= (3 + 2 + 3) mod 4 = 0 Swap (S[2],S[0])
Hasil 2 3 1 0

Iterasi 4
i = 3
j = (0 + S[3] + K [3 mod 4]) mod 4
= (0 + 0 + 4) mod 4 = 0 Swap (S[3],S[0])
Hasil Array S 2 3 1 0

Setelah dilakukan KSA, selanjutnya dilakukan proses PRGA (*Pseudo-random generation algorithm*). PRGA akan dilakukan sebanyak 4 kali karena plainteks yang akan dienkripsi berjumlah 4 karakter. Hal ini disebakan karena sangat dibutuhkan 1 kunci dan 1 kali pengoperasian *XOR* untuk tiap tiap karakter pada plainteks. Selanjutnya adalah tahapan proses penghasilan kunci enkripsi dengan PRGA.

Array S 2 3 1 0
Inisialisasi 1
i = 0
j = 0
Iterasi 1
i = (0 + 1) mod 4 = 1
j = (0 + S[1]) mod 4 = (0+ 3) mod 4
= 3 swap (S[1],S[3])
2 1 0 3
K1 = S[(S[1]+S[3]) mod 4] = S[2 mod 4] = 2 K1 = 00000000

Iterasi 2
i = (1 + 1) mod 4 = 2
j = (2 + S[2]) mod 4 = (2+ 0) mod 4
= 2 swap (S[2],S[2])
2 1 0 3
K2 = S[(S[2]+S[2]) mod 4] = S[0 mod 4] = 0 K2 = 00000000

Iterasi 3
i = (2 + 1) mod 4 = 3
j = (2 + S[3]) mod 4 = (2+ 3) mod 4
= 1 swap (S[3],S[1])
3 1 0 2
K3 = S[(S[3]+S[1]) mod 4] = S[5 mod 4] = 1 K3 = 00000001

Iterasi 4
i = (3 + 1) mod 4 = 0
j = (1 + S[0]) mod 4 = (1+ 3) mod 4
= 0 swap (S[0],S[0])
3 1 0 2
K4 = S[(S[0]+S[0]) mod 4] = S[6 mod 4] = 2
K4 = 00000010

Sesudah menemukan kunci untuk tiap karakter, maka akan dilakukan operasi *XOR* antara karakter pada *plaintext* dengan kunci yang didapat. Berikut adalah tabel *ASCII* untuk tiap- tiap karakter pada *plaintext* yang akan digunakan.

Tabel 1. Huruf Kode *ASCII* (*Binary* 8 *bit*)

Kata	K	A	M	U
Hexadesimal	4B	41	4D	55
Biner	01010011	01000001	01011001	01000001

Berikut adalah proses peng*XOR*an dari *plaintext* dengan *key* yang telah didapat:

	K	A	M	U
Plaintext	: 01001011	01000001	01001101	01010101
Key	: 00000000	00000000	00000001	00000010 ⊕
Ciphertext	: 01001011	01000001	01001100	01010111
	K	A	L	W

Setelah di dapat hasil enkripsi menggunakan algoritma RC4 selanjutnya *ciphertext* yang didapat akan disisipkan ke dalam sebuah gambar menggunakan algoritma LSB.

Gambar 1. Gambar asli yang akan disisipkan pesan

Setelah proses zoom gambar di dapat nilai dari masing-masing *pixel* yang akan digunakan untuk menyembunyikan pesan. Nilainya adalah sebagai berikut pada Tabel 2:

Tabel 2. Nilai extraksi gambar

R	G	B	R	G	B	R	G	B
255	0	0	255	0	0	255	1	0
255	0	0	255	0	0	255	1	0
255	0	1	255	0	1	255	1	0
255	0	1	255	0	1	255	0	1

Nilai RGB tersebut selanjutnya diubah kedalam bentuk biner seperti pada Tabel 3:

Tabel 3. Nilai Extrasi Hexa ke Biner

R	G	B	R	G	B	R	G	B
1111 1111	0000 0000	0000 0000	1111 1111	0000 0000	0000 0000	1111 1111	0000 0001	0000 0000
1111 1111	0000 0000	0000 0000	1111 1111	0000 0000	0000 0000	1111 1111	0000 0001	0000 0000
1111 1111	0000 0000	0000 0001	1111 1111	0000 0000	0000 0001	1111 1111	0000 0001	0000 0000
1111 1111	0000 0000	0000 0001	1111 1111	0000 0000	0000 0001	1111 1111	0000 0000	0000 0001

pesan yang akan disisipkan adalah KALW, yang merupakan hasil enkripsi dari algoritma RC4 maka sebelum proses penyisipan pesan akan diubah ke dalam bentuk biner sehingga dihasilkan :

K:01001011 A:01000010
L:01001100 W:01010111

Dalam algoritma LSB cara menyispkan pesan adalah dengan mengganti *bit* ke 8, 16 dan 24 setiap *pixel* gambar dengan nilai biner dari pesan yang akan disisipkan, sehingga berdasarkan ketentuan tersebut hasil penyisipan dapat dilihat pada *bit* yang ditebalkan (*bold*) seperti pada Tabel 4.

Tabel 4. Penyisipan Pesan Biner

R	G	B	R	G	B	R	G	B
1111 1110	0000 0001	0000 0000	1111 1110	0000 0001	0000 0000	1111 1111	0000 0001	0000 0000
1111 1111	0000 0000	0000 0000	1111 1110	0000 0000	0000 0001	1111 1110	0000 0000	0000 0000
1111 1111	0000 0000	0000 0000	1111 1111	0000 0001	0000 0000	1111 1110	0000 0000	0000 0001
1111	0000	0000	1111	0000	0000	1111	0000	0000

1110	0001	0000	1111	0001	0001	1111	0000	0001

R	G	B	R	G	B	R	G	B
1111 1110	0000 0001	0000 0000	1111 1110	0000 0001	0000 0000	1111 1111	0000 0001	0000 0000
1111 1111	0000 0000	0000 0000	1111 1110	0000 0000	0000 0001	1111 1110	0000 0000	0000 0000
1111 1111	0000 0000	0000 0000	1111 1111	0000 0001	0000 0000	1111 1110	0000 0000	0000 0001
1111 1110	0000 0001	0000 0000	1111 1111	0000 0001	0000 0001	1111 1111	0000 0000	0000 0001

Proses pengambilan pesan menggunakan algoritma LSB adalah dengan mengambil nilai tiap-tiap *bit* paling kanan dari nilai biner *pixel* gambar. Sehingga dari hasil penyembunyian pesan diatas, dari proses pengambilan tiap-tiap *bit* paling kanan akan dihasilkan nilai yang tampak pada Tabel 5:

Tabel 5. Proses Pengambilan Pesan Biner

R	G	B	R	G	B	R	G	B
1111 1110	0000 0001	0000 0000	1111 1110	0000 0001	0000 0000	1111 1111	0000 0001	0000 0000
1111 1111	0000 0000	0000 0000	1111 1110	0000 0000	0000 0001	1111 1110	0000 0000	0000 0000
1111 1111	0000 0000	0000 0000	1111 1111	0000 0001	0000 0000	1111 1110	0000 0000	0000 0001
1111 1110	0000 0001	0000 0000	1111 1111	0000 0001	0000 0001	1111 1111	0000 0000	0000 0001

R	G	B	R	G	B	R	G	B
1111 1110	0000 0001	0000 0000	1111 1110	0000 0001	0000 0000	1111 1111	0000 0001	0000 0000
1111 1111	0000 0000	0000 0000	1111 1110	0000 0000	0000 0001	1111 1110	0000 0000	0000 0000
1111 1111	0000 0000	0000 0000	1111 1111	0000 0001	0000 0000	1111 1110	0000 0000	0000 0001
1111 1110	0000 0001	0000 0000	1111 1111	0000 0001	0000 0001	1111 1111	0000 0000	0000 0001

Pada saat ekstrasi data diambil tiap *bit* dari LSB dari setiap *byte pixel* pada gambar, selanjutnya *bit* tersebut di satukan dan *konversi* ke dalam karakter.

K:01001011 A:01000010
L:01001100 W:01010111

Setelah pesan diambil dari gambar langkah selanjutnya akan dilakukan proses pengembalian pesan asli (dekripsi), Proses ini sama untuk proses *key-schedule*-nya. Untuk mendapatkan pesan asli

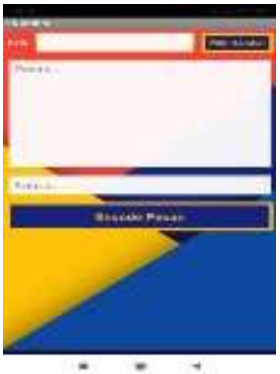
(*plaintext)*, *ciphertext* yang diperoleh di *XOR*kan dengan *pseudo random byte* yang diambil sebelumnya. Maka hasilnya adalah *plaintext* atau teks asli.

Proses *XOR pseudo random byte* dengan *ciphertext* pada dekripsi yaitu:

	K	A	L	W
Ciphertext	: 01001011	01000001	01001100	01010111
Key	: 00000000	00000000	00000001	00000010 ⊕
Plaintext	: 01001011	01000001	01001101	01010101
	K	A	M	U

Halaman Encode Pesan

Halaman Encode pesan dapat terlihat proses Penyembunyian Pesan Ke Dalam Gambar terlihat pada gambar 2.

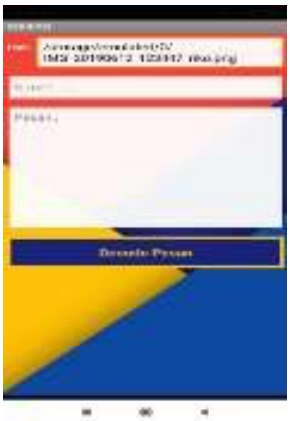


Gambar 2. Halaman *Encode* Pesan

Penjelasan: Pilih gambar yang akan disipkan kemudian input pesan yang akan dirahasiakan serta menginput kunci agar bisa terjadi proses penyembunyian pesan ke dalam gambar.

Halaman *Decode* Pesan

Digunakan untuk proses mengambil pesan dari sebuah gambar dan melakukan dekripsi untuk mengembalikan pesan ke dalam bentuk aslinya. Gambar tampilan halaman *Decode* pesan ditunjukkan pada gambar 3.



Gambar 3. Halaman *Decode* pesan

Penjelasan: pilih gambar yang sudah kita sisipkan pesan rahasia, kemudian input kunci yang kita masukan sebelumnya, maka keluarlah hasil pesan berbentuk pesan asli (*plaintext).*

KESIMPULAN

Berdasarkan hasil pambahasan dan uji coba yang telah dilakukan, dapat disimpulkan:

1. Aplikasi dapat berjalan dengan baik pada smartphone android.
2. Aplikasi dapat digunakan untuk menyembunyikan pesan ke dalam gambar dan juga mengambil pesan dari sebuah gambar.
3. Pesan yang akan disembunyikan sebelumnya akan di enkripsi menggunakan algoritma RC4 lalu disembunyikan ke dalam gambar menggunakan algoritma LSB berjalan dengan baik.

DAFTAR PUSTAKA

Andrian, Y. (2013). Modifikasi Metode Least Significant Bits (LSB) pada Steganografi Citra Digital. *Prosiding Seminar Nasional Ilmu Komputer (SNIKOM) 2013*. pp. 274-279. FIKOM Universitas Methodist Indonesia. Medan.

Buchmann, J. A. (2004). *Introduction to Cryptography,* 2nd Edition. New York: Springer.

Jamaluddin, J., Zarlis, M., & Tulus, T. (2014). Pengamanan Data dengan Kombinasi Teknik Kriptografi Rabin dan Teknik Steganografi Chaotic LSB. *Prosiding SNASTIKOM 2014*. STT Harapan. Medan.

Munir, R. (2004). *Pengolahan Citra Digital*. Bandung: Informatika.

Putra, D. (2010). *Pengolahan Citra Digital*. Yogyakarta: Penerbit Andi.

Rahul, L. S. (2013). Image Steganography With LSB. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), 2* (1).

Sinaga, M. D. et al., (2018). Hybrid Cryptography WAKE (Word Auto Key Encryption) and Binary Caesar Cipher Method For Data Security. *Proceeding of 6th International Conference on Cyber and IT Service Management (CITSM).*

ISSN: 2598-8565 (media cetak)
ISSN: 2620-4339 (media online)

Stallings, W. (2016). *Cryptography and Network Security, Principles and Practice.* 7th Edition. Essex: Pearson.

Tambunan, F. (2016). Perancangan aplikasi chatting mime base64 dengan koneksi wireless. *Prosiding Seminar Nasional Informatika (SNIf) 2016*. Univ. Potensi Utama. Medan

Zebua, T. (2015). Penerapan Metode Lsb- 2 Untuk Menyembunyikan Ciphertext Pada Citra Digital. *Pelita Informatika Budi Darma*, *10* (3), 135-140.