PENERAPAN KRIPTOGRAFI DAN STEGANOGRAFI PADA CITRA DIGITAL MENGGUNAKAN ALGORITMA RSA BERBASIS ANDROID

Aries Maesya [1] **, Mochammad Iqbal Permana** [2]

[1), 2)] *Ilmu Komputer Universitas Pakuan*
Jl. Pakuan PO BOX 452, Bogor Telp/Fax (0251) 8375 547
Email : a.maesya@gmail.com[1)], iqbal.perman@gmail.com[2)]

Abstrak

Pengiriman pesan melalui mobile dapat di terima secara langsung oleh penerima pesan, tetapi pengiriman dengan cara ini juga rentan terhadap bahaya pencurian data. Untuk mengatasi pengiriman pesan tersebut, maka digunakan kriptografi dan steganografi. Dalam penelitian ini, media yang diajukan adalah penggunaan media image seperti file JPG dan PNG sebagai data masukan media pembawa pesan rahasia (secret-text). Dengan menggunakan metode Least Significant Bit diharapkan nantinya kualitas image yang dihasilkan dari proses embeding data tidak akan terlalu bertambah secara signifikan. Selain itu, agar menjaga data lebih aman digunakan algortima enkripsi RSA. Setelah melalui proses embedding data, output image hasil dari penelitian ini tidak mengalami penurunan kualitas. Begitu juga untuk proses pengekstrakan kembali dari berkas stego, informasi rahasia dapat diungkapkan kembali tanpa mengalami kerusakan sedikitpun.

Kata kunci : *Kriptografi, Algoritma RSA, Steganografi.*

1. Pendahuluan

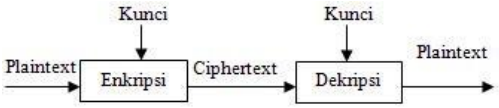
Proses pertukaran informasi termasuk pengiriman pesan dapat dilakukan dalam berbagai macam cara selain itu, pesan yang dapat dikirim tidak terbatas pada pesan yang berbentuk data teks, tetapi juga berbentuk data audio maupun data video. Pengiriman pesan dapat diterima secara langsung oleh penerima pesan, tetapi pengiriman dengan cara ini juga rentan terhadap bahaya pencurian data. Oleh karena itu diperlukan suatu sistem pengamanan data yang dapat melindungi pesan pesan yang bersifat pribadi dan rahasia supaya sampai ke tangan orang yang berhak menerima pesan tersebut.

Untuk mengatasi pengiriman informasi rahasia, maka digunakan teknik kriptografi. Kriptografi mencakup enkripsi dan dekripsi. Enkripsi adalah suatu proses mentransformasikan pesan yang hendak dikirim (plaintext) sedemikian sehingga menjadi suatu pesan yang tidak dapat dimaknai (ciphertext). Dekripsi merupakan invers dari enkripsi, yaitu suatu proses mengembalikan ciphertext menjadi plaintext. Enkripsi dan dekripsi memerlukan bantuan yang disebut kunci. Beberapa proses enkripsi-dekripsi menggunakan kunci yang sama (simetris), namun ada pula yang menggunakan kunci yang berbeda (asimetris). Masing-masing proses enkripsi, dekripsi, dan pembangkitan kunci dibuat berdasarkan suatu algoritma. Suatu algoritma dibuat berdasarkan fungsi-fungsi matematika. Penggunaan kriptografi saja pada pengiriman pesan rahasia dirasa memiliki kekurangan dalam hal keamanan. Steganografi dianggap dapat menutupi kekurangan yang ada pada kriptografi..

2. Tinjauan Pustaka

2.1 Kriptografi

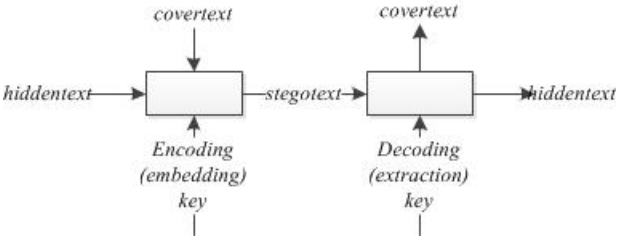
Kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan data, keabsahan data, integritas data, serta autentikasi data[1].



Gambar 1. Proses Enkripsi dan Dekripsi

2.2 Steganografi

Steganografi adalah ilmu dan seni menyembunyikan pesan rahasia (hiding message) sedemikian sehingga keberadaan (eksistensi) pesan tidak terdeteksi oleh indera manusia[2].



Gambar 2. *Encoding* dan *Decoding* steganografi

2.3 Algoritma RSA

Algoritma RSA dibuat oleh tiga orang peneliti dari MIT (Massachussets Institute of Technology) pada tahun 1976, akronim RSA berasal dari nama belakang penemunya yaitu Ron Rivest, Adi Shamir dan Leonard Adleman. RSA adalah salah teknik kriptografi dimana

kunci untuk melakukan enkripsi berbeda dengan kunci untuk melakukan dekripsi. Kunci untuk melakukan enkripsi disebut sebagai kunci publik, sedangkan kunci untuk melakukan dekripsi disebut sebagai kunci privat. Orang yang mempunyai kunci publik dapat melakukan enkripsi tetapi yang dalam melakukan dekripsi hanyalah orang yang memiliki kunci privat. Kunci publik dapat dimiliki oleh sembarang orang, tetapi kunci privat hanya dimiliki oleh orang tertentu saja[3].

Ada beberapa langkah dalam membangkitkan nilai dari kunci publik dan kunci privat dalam RSA, yaitu:

1. Ambil dua bilangan prima secara acak, sebagai contoh p=23 dan q=11.
2. Hitung n = p*q = 23*11 = 253.
3. Hitung m = (p-1)*(q-1) = (23-1)*(11-1) = 220.
4. Pilih sebuah bilangan prima e yang memiliki syarat: 1 < e < m, yang relatif prima terhadap m (catatan: m tidak bisa dibagi oleh e). Sebagai contoh kita ambil e = 13.
5. Lalu n = 253 dan e = 13 dijadikan sebagai kunci publik / enkripsi.
6. Untuk menghitung kunci privat, kita harus mengetahui nilai d (dengan syarat d juga bilangan prima) dari persamaan berikut :

   e * d = 1 (mod n)
   13 * d = 1 (mod 253)

Dimana e * d menghasilkan 1 jika dimoduluskan dengan n, artinya (13 * d) mod 253 = 1. Setelah dicoba dengan d = 1,2,3, ...dst, maka didapatkan nilai d = 17 sehingga e*d = 221 karena 221 dimoduluskan 253 menghasilkan nilai 1. Jadi d dan n yang dijadikan sebagai kunci privat / dekripsi.

Ini merupakan langkah dalam mengenkripsi pesan dengan kunci publik. Contoh pesan yang akan dikirim berupa *plaintext* P = AKU, jika diubah kedalam *American Standard Code for Information Interchange* (ASCII), yaitu kode standar yang digunakan dalam pertukaran informasi pada komputer komputer hanya dapat memahami nomor.

Tabel 1. *Representasi Plaintext "AKU"*

Char	Hex	Dec
A	41	65
K	4B	75
U	55	85

Dengan menggunakan rumus P ^ e = E (mod n), dimana P adalah plaintext yang akan dienkripsi, e dan n adalah kunci publik, sedangkan E adalah ciphertext. Berikut langkah – langkah yang dilakukan:

1. Buat plaintext yang telah diubah ke ASCII Desimal menjadi blok-blok dua digit:
   P1 = 65 P3 = 85 P2 = 75
   Perlu diketahui bahwa nilai Pi masih dalam jarak antara 0 sampai n – 1 (253 - 1).
2. Pi ^ e = E (mod n)
   - P1 = 6513 mod 253 = 76
   - P2 = 7513 mod 253 = 36
   - P3 = 8513 mod 253 = 72

Maka, ciphertext adalah E = 76 36 72 , karena ciphertext tersebut masih dalam bentuk nilai ASCII Desimal maka jika dikonversi ke dalam karakter.

Tabel 2. *Representasi Ciphertext "L$H"*

Dec	Hex	Char
76	4C	L
36	24	$
72	48	H

Jadi, *ciphertext* E = L$H

Untuk dekripsi (mengubah ciphertext menjadi plaintext) maka digunakan kunci privat d = 17 dan n = 253 dengan cara sebagai berikut :
Ei ^ d = P (mod n)
E1 = 7617 mod 253 = 65
E2 = 3617 mod 253 = 75
E3 = 7217 mod 253 = 85
Maka diperoleh plaintext P = 65 75 85
dalam karakter ASCII P = AKU.

2.4 Least Significant Bit Insertion

Least Significant Bit (LSB) Insertion adalah salah satu metode steganografi yang menyembunyikan pesan dengan memanfaatkan bit-bit paling kanan (bit terlemah) dari sebuah file gambar[4]. Jenis-jenis format pewarnaan di dalam media gambar, seperti grayscale, RGB, dan CMYK. Sebagai contoh pewarnaan *monochrome bitmap* (menggunakan 1 bit untuk tiap *pixel*), RGB - 24 bit (R=8 bit, G=8 bit, B=8 bit), *Grayscale*-8 bit (menentukan tingkat kehitaman suatu *pixel* berdasarkan nilai bitnya)

Gambar 3. Abstrak.jpg

Gambar 3 menggunakan pewarnaan RGB yang terdiri dari 5 *pixel* x 5 *pixel*, berikut nilai RGB pada masing-masing *pixel*-nya.

Tabel 3. Nilai Warna RGB per *Pixel*

	1	2	3	4	5
1	247, 197, 200	186, 148, 147	96, 101, 97	169, 223, 187	218, 253, 229
2	254, 116, 114	244, 120, 120	197, 210, 192	88, 243, 141	156, 250, 188
3	254, 219, 217	251, 201, 204	240, 234, 244	204, 235, 229	195, 211, 201
	254,	247,	207,	156,	41,

4	192, 195	89, 90	159, 221	132, 210	40, 36
5	253, 215, 214	246, 140, 150	196, 154, 230	164, 136, 239	170, 166, 181

Karena metode LSB Insertion membutuhkan bit dalam melakukan penyisipan datanya, maka nilai RGB tiap piksel yang ada pada gambar perlu diubah kedalam biner.

Tabel 4. Hasil Konversi Nilai RGB ke Data Biner

11110111, 11000101, 11001000	10111010, 10010100, 10010011	01100000, 01100101, 01100001	10101001, 11011111, 10111011	11011010, 11111101, 11100101
11111110, 01110100, 01110010	11110100, 01111000, 01111000	11000101, 11010010, 11000000	01011000, 11110011, 10001101	10011100, 11111010, 10111100
11111110, 11011011, 11011001	11111011, 11001001, 11001100	11110000, 11101010, 11110100	11001100, 11101011, 11100101	11000011, 11010011, 11001001
11111110, 11000000, 11000011	11110111, 01011001, 01011010	11001111, 10011111, 11011101	10011100, 10000100, 11010010	00101001, 00101000, 00100100
11111101, 11010111, 11010110	11110110, 10001100, 10010110	11000100, 10011010, 11100110	10100100, 10001000, 11101111	10101010, 10100110, 10110101

Jika Embedded Message yang disisipkan adalah "L$H", Embedded Message tersebut telah dienkripsi dengan kunci public e,n (13, 253) dan kunci privat d,n (17, 253) lalu diubah ke dalam nilai ASCII Biner

Tabel 5. Data Biner *Embedded Message "**AKU**"*

	1	2	3	4	5	6	7	8
L	0	1	0	0	1	1	0	0
$	0	0	1	0	0	1	0	0
H	0	1	0	0	1	0	0	0

Maka setiap bit dari karakter "L$H" disisipkan pada LSB dari kumpulan deretan biner di atas (dari kiri ke kanan), sehingga deretan biner tersebut mengalami perubahan nilai.

Tabel 6. Data Biner Hasil LSB *Insertion*

11110110, 11000101, 11001000	10111010, 10010101, 10010011	01100000, 01100100, 01100000	10101000, 11011111, 10111010	11011010, 11111101, 11100100
11111110, 01110100, 01110011	11110100, 01111000, 01111001	11000100, 11010010, 11000000	01011000, 11110011, 10001101	10011100, 11111010, 10111100
11111110, 11011011, 11011001	11111011, 11001001, 11001100	11110000, 11101010, 11110100	11001100, 11101011, 11100101	11000011, 11010011, 11001001
11111110, 11000000, 11000011	11110111, 01011001, 01011010	11001111, 10011111, 11011101	10011100, 10000100, 11010010	00101001, 00101000, 00100100
11111101, 11010111, 11010110	11110110, 10001100, 10010110	11000100, 10011010, 11100110	10100100, 10001000, 11101111	10101010, 10100110, 10110101

Dari hasil tersebut ada sedikit perubahan pada *Cover Object* tersebut yang ditandai dengan angka tebal, namun ada beberapa juga piksel yang tidak berubah dikarenakan nilai bit terakhirnya sama dengan nilai bit terakhir dari Embedded Message.

Steganografi dengan metode LSB hanya mampu menyimpan informasi dengan ukuran yang sangat terbatas. Seperti pada gambar 3 diatas digunakan sebagai wadah untuk menyimpan data berukuran 24 bit, jika masing – masing komponen warnanya (RGB) digunakan satu pixel untuk menyimpan informasi rahasia tersebut, maka setiap pixelnya disimpan 3 bit informasi, sehingga setidaknya dibutuhkan citra wadah berukuran 25 piksel atau setara 25 x 3 x 8 = 600 bit (25 kali lipat). Jadi suatu citra 24-bit hanya mampu menampung pesan maksimum berukuran 1/25 dari ukuran citra penampung tersebut

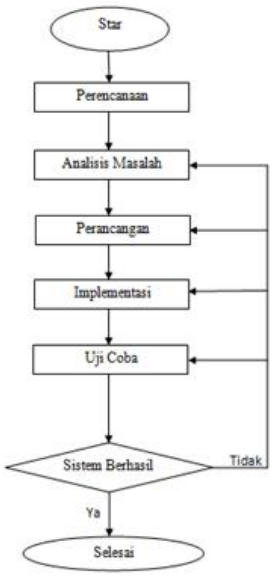
2.5 Android

Android merupakan perangkat bergerak pada sistem operasi untuk telepon seluler yang berbasis linux[4]. Android merupakan OS (Operating System)

Mobile yang tumbuh ditengah OS lainnya yang berkembang dewasa ini OS lainnya seperti Windows Mobile, IPhone OS, Symbian, dan masih banyak lagi. Akan tetapi, OS yang ada ini berjalan dengan memprioritaskan aplikasi inti yang dibangun sendiri tanpa melihat potensi yang cukup besar dari aplikasi pihak ketiga.Oleh karena itu, adanya keterbatasan dari aplikasi pihak ketiga untuk mendapatkan data asli ponsel, berkomunikasi antar proses serta keterbatasan distribusi aplikasi pihak ketiga untuk platform mereka.

3. Metode Penelitian

Metode dalam penelitian ini menggunakan metode Siklus Hidup Pengembangan Sistem (System Development Life Cycle - SDLC).



Gambar 3. Tahap Pengembangan SDLC

3.1. Perancangan Secara Umum

Dengan pertimbangan terhadap latar belakang bahwa kriptografi tanpa steganografi atau steganografi tanpa kriptografi masih rentan terhadap pembobolan maka aplikasi steganografi akan mengenkripsi pesan/file rahasia menjadi sebuah file chipper/hidden, kemudian akan dienkripsi kembali (encode) dengan metode steganografi yang akan menghasilkan suatu file stego-object

Gambar 4. Kerangka Sistem

3.2. Proses Pembangkitan Kunci

Keamanan algoritma RSA terletak pada sulitnya memfaktorkan bilangan yang besar menjadi faktor-faktor prima. Pemfaktoran dilakukan untuk memperoleh kunci pribadi. Selama pemfaktoran bilangan besar menjadi faktor-faktor prima belum ditemukan algoritma yang mangkus, maka selama itu pula keamanan algoritma RSA tetap terjamin.

Gambar 5. Alur Kerja Membangkitkan Kunci

3.3. Proses Enkripsi Pesan

Proses enkripsi pesan memerlukan masukan dari pengguna berupa pesan rahasia dan kunci public.

Gambar 6. Proses Enkripsi Pesan

3.4 Proses Penyisipan Pesan

Proses penyisipan pesan pada aplikasi steganografi memerlukan masukan dari pengguna berupa cover object dan pesan rahasia yang sudah dienkripsi

ISSN : 2302-3805

Gambar 7. Proses Penyisipan Pesan

3.5. Proses Pengungkapan Pesan

Proses pengungkapan pesan hanya memerlukan satu masukan, yaitu stego- object.

G

ambar 8. Proses Pengungkapan Pesan

3.6. Proses Dekripsi Pesan

Proses dekripsi pesan hanya memerlukan dua masukan, yaitu: pesan rahasia yang berhasil diekstrak dari stego-object (stego- text) dan kunci privat.

Gambar 9. Proses Dekripsi Pesan

3.7. Desain Antar Muka

Perancangan antarmuka pengguna merupakan salah satu bagian yang perlu diperhatikan untuk perancangan *user interface*. Perancangan ini dipaparkan melalui desain tampilan aplikasi sistem. Dalam desain antar muka ini, penggunaan system antar muka dibedakan menjadi 3 bagian utama yaitu bagian untuk meng-*generate key,*bagian untuk menyembunyikan informasi (*encoding data*), dan bagian untuk mengambil informasi dari file stego (*decoding data*).

Gambar 10. Tampilan Utama

4. Hasil dan Pembahasan

4.1. Proses *Generate Key*

Proses ini dilakukan untuk memilih pasangan kunci publik-privat. Karena kunci yang dipakai harus menggunakan hitungan dari algoritma RSA.

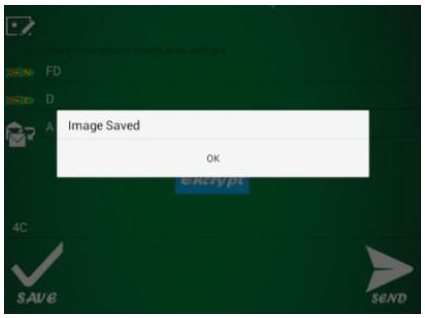
Gambar 11. Tampilan *Generate Key*

4.2. Proses Encoding Data

Proses Encoding dilakukan untuk menyisipkan secret-text ke dalam cover-object. Berkas gambar yang digunakan sebagai cover-object adalah "-Kara-kara-18981458-1024-768.jpg" dengan ukuran 23,7KB, secret-text yang akan disisipkan adalah "girlband korea", dan public key yang dipakai merupakan hasil generate key.Hasil dari encoding yaitu stego-object berupa gambar dengan ekstensi PNG.

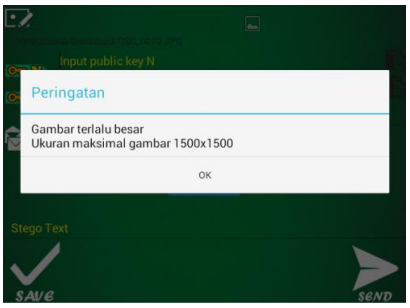
Gambar 12. Proses Enkripsi – *Encoding*

Jika proses *encoding* selesai, maka aplikasi memberikan pemberitahuan berikut.



Gambar 13. Pesan sukses *encoding*

Gambar yang digunakan berekstensi jpg dan memiliki resolusi maksimal 1500px X 1500px. Jika gambar memiliki resolusi lebih dari 1500px X 1500px, maka akan muncul peringatan seperti ini.



Gambar 14. Pesan *error* ukuran gambar

4.3. Proses *Decoding Data*

Proses *decoding* dilakukan untuk mengembalikan atau mengektraksi informasi dari stego-object. Nama berkas stego-object ini yaitu "-Kara-kara-18981458-1024-768_235_JpegStego.png" dan masih menggunakan private key dari proses generate key yang sama.

Gambar 15. Proses Dekripsi – *Decoding*

Dari proses *decoding* diatas didapatkan pesan tersembunyi (*hidden-text*) didalam *stego-object* yaitu adalah "girlband korea". Tetapi jika didalam pemilihan gambar yang akan *decoding* tidak terdapat *hidden-text*, maka akan muncul peringatan berikut.

Gambar 16. Pesan *error* tidak ada *hidden text*

5. Hasil Pengujian

Dalam melakukan pengujian kami memanfaatkan beberapa berkas *cover- object* dan beberapa pesan rahasia (*secret- text*) yang akan disimpan dalam *cover-object*. Berkas–berkas yang akan digunakan tesebut dan *output* yang dihasilkan adalah sebagai berikut :

Tabel 7. Berkas *Cover-Object*

No	Nama	Ukuran	Tipe
1	worlds_best_cat3.jpg	432x288 (104kb)	JPEG (24bit)

2	-Kara-kara-18981458-1024-768.jpg	280x210 (23,7kb)	JPEG (24bit)

Tabel 8. Berkas *Stego-object*

No	Nama	Ukuran	Tipe
1	worlds_best_cat3_971_JpegStego.png	432x288 (219kb)	PNG (32bit)
2	-Kara-kara-18981458-1024-768_235_JpegStego.png	280x210 (99,9kb)	PNG (32bit)

Tabel 9. Pesan & Kunci Publik-Privat yang digunakan

No	*Cover Object*	*Secret Text*	N Mod	Kunci Publik	Kunci Privat	*Stego Object*
1	1	"A"	FD	D	11	1
2	2	"girlband korea"	DEDE 58F6E 150DA 0E817 CBB88 89F55 E6AF7 209A3 EE6A1 997419 B741B C260A D5A9	EBA 121F 060C 874B 39C0 D39 7F73 A3D 18E8 F43 CBE D57 342F A21 027F 5E1F 36B3 C2F	A8F30 05F05 E9DC8 8EAB2 2262C 644E8 7ED33 0358D 75D6B 7D7F7 5AEE0 3B269 09BF	2

6. Simpulan dan Saran

Berdasarkan analisa dari hasil kedua uji coba yang diterangkan pada bagian sebelumnya, kesimpulan yang didapatkan adalah :

a. Aplikasi ini mampu menyisipkan dan mengambil pesan rahasia yang ada pada gambar.
b. Aplikasi ini menawarkan fitur pengamanan data dengan menggunakan Algoritma RSA yang menggunakan kunci publik-privat.
c. Gambar yang dihasilkan proses encoding (stego-object) menggunakan ekstensi PNG
d. Resolusi gambar cover-object dengan stego-object tidak mengalami perubahan, namun terjadi perubahan kedalaman bit dari 24bit ke 32bit.
e. Semakin panjang pesan yang disisipkan, semakin besar ukuran gambar stego object-nya.

7. Daftar Pustaka

[1]**Menezes A, Oorschot, van P, & Vanstone, S.** 1996. *Handbook of pplied Cryptography*. CRC Press.
[2]**Alatas, Putri**. 2009. *Implementasi Teknik Steganografi Dengan Metode LSB Pada Citra Digital*. Tugas Akhir. Jurusan Sistem Informasi Fakultas Ilmu Komputer & Teknologi Informasi Universitas Gunadarma, Depok.
[3]**Kromodimoeljo, Sentot.** 2010. *Teori dan Aplikasi Kriptografi.* SPK IT Consulting.
[4]**Arifanto, Teguh.** 2011. *Membuat Interface Aplikasi Android Lebih Keren dengan LWUIT*. Andi Publisher, Yogyakarta

Biodata Penulis

Aries Maesya memperoleh gelar Sarjana Komputer (S.Kom), Jurusan Teknik Informatika STMIK AMIKOM Yogyakarta, lulus tahun 2006. Memperoleh gelar Magister Komputer (M.Kom) Program Pasca Sarjana Magister Teknik Informatika Universitas Gajah Mada Yogyakarta, lulus tahun 2011.Saat ini menjadi Dosen di STMIK AMIKOM Yogyakarta.