Implementasi Pengembangan Algoritma RSA dan *Pseudo-Random Prime-Number Generator* dalam Optimalisasi Steganografi

Michael Leon Putra Widhi - 13521108[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13521108@std.stei.itb.ac.id

***Abstrak*— Sepanjang sejarah manusia, kriptografi telah digunakan untuk menutupi berbagai informasi dari badan atau kelompok tertentu. Salah satu metode kriptografi yang banyak digunakan dalam hal pengiriman data dikenal dengan algoritma RSA. Algoritma yang telah digunakan puluhan tahun ini telah membuktikan betapa sulit dan kuatnya algoritma kriptografi ini. Selain dalam sebuah data, kriptografi dalam bentuk citra juga menjadi sebuah alternatif dewasa ini. Steganografi adalah ilmu dan seni menyembunyikan pesan rahasia (*hiding message*) sedemikian sehingga eksistensi sebuah pesan tidak terdeteksi oleh indera manusia [2]. Steganografi sudah kerap kali dikombinasikan dengan algoritma RSA, tetapi pada makalah ini akan dibahas mengenai bagaimana melakukan optimalisasi pada steganografi menggunakan algoritma RSA yang disempurnakan dengan pembangkit bilangan prima acak semu (*Pseudo-Random Prime-Number Generator*) untuk menghasilkan pola yang lebih acak. Selain itu, metode ini diharapkan dapat menjadi optimalisasi yang baik dari metode LSB (*Least Significant Bit)* yang seringkali dijadikan basis melakukan steganografi.**

***Kata Kunci*— Algoritma RSA, PRNG, Steganografi**

I. PENDAHULUAN

Perkembangan era digital yang semakin pesat saat ini telah berhasil membawa berbagai macam kemudahan bagi kita melakukan komunikasi dan mendapatkan akses informasi kapan pun dan dari manapun. Kemudahan ini tentu saja memberikan banyak sekali manfaat bagi kita, manusia dan kehidupannya. Namun seiring berjalannya waktu, kemudahan ini kerap kali dimanfaatkan oleh oknum tidak bertanggung jawab untuk melakukan peretasan data dan pencurian informasi, sehingga berbagai pihak di dunia membutuhkan sebuah solusi untuk melakukan pengiriman dan penerimaan data dengan lebih aman dengan prosedur keamanan yang lebih ketat.

Sepanjang puluhan tahun terakhir, telah banyak usaha yang dilakukan agar sebuah informasi dapat disampaikan dan tersampaikan dengan tingkat keamanan yang baik. Salah satu cara yang digunakan untuk melakukan pengiriman data melalui media internet adalah kriptografi. Kriptografi menurut catatan sejarah telah eksis sejak masa kejayaan Yunani atau kurang lebih sekitar tahun 400 Sebelum Masehi dan masih ada hingga saat ini [1]. Salah satu metode kriptografi paling terkenal adalah algoritma RSA. Algoritma kunci-nirsimetri ini membuat kunci yang dilakukan untuk melakukan enkripsi berbeda dengan kunci yang digunakan untuk melakukan dekripsi. Hal ini membuat algoritma RSA memiliki sistem keamanan yang sangat tinggi.

Selain dalam penyampaian pesan, kriptografi juga sudah dikembangkan dan diterapkan dalam sebuah citra. Adalah Steganografi, sebuah praktik yang merepresentasikan informasi dalam pesan atau objek fisik lain, sedemikian rupa sehingga keberadaan informasi tersebut tidak terlihat oleh pemeriksaan manusia. Steganografi menjadi suatu hal yang unik karena hampir setiap media digital dapat menjadi media yang digunakan dalam menyembunyikan pesan. Media-media tersebut seperti berkas citra digital, berkas audio, dan berkas video.

Pembangkit bilangan acak semu (*Pseudo-Random Number Generator*) memiliki karakteristik yang unik dalam membangun sebuah bilangan acak karena polanya akan terulang akibat menggunakan *seeds* yang statik. Karakteristik ini yang dapat dikembangkan menjadi sebuah pengacak bilangan prima sebagai dasar pembuatan kunci-nirsimetri. Selain itu, keamanan proses steganografi juga dapat ditingkatkan dengan memasukan hasil kriptografi dengan algoritma RSA pada citra, sehingga proses dekripsi memerlukan sebuah kunci tertentu yang relatif lebih sulit untuk ditemukan.

II. TEORI DASAR

A. Bilangan Bulat

Bilangan bulat adalah bilangan yang dapat dituliskan tanpa komponen desimal atau pecahan. Sebagai contoh, $2, 4, 0, -3, -67,$ dan -2048 merupakan bilangan bulat, sedangkan 9.75, $5\frac{1}{2}$, dan $\sqrt{5}$ bukan. Himpunan bilangan bulat terdiri dari angka 0, semua bilangan bulat positif serta semua bilangan bulat negatif. Dalam matematika, himpunan ini sering dilambangkan dengan $\mathbb{Z}$.

B. Pembagian dan Keterbagian Bilangan Bulat

Misalkan a dan b adalah dua buah bilangan bulat yang memenuhi $a \neq 0$. a dikatakan habis membagi b (a *divides* b) jika terdapat bilangan bulat c sedemikian sehingga $b = ac$ atau dapat dituliskan dalam notasi matematis sebagai berikut

$$a \mid b, \text{ jika } b = ac,\ c \in \mathbb{Z},\ a \neq 0$$

Melalui teori tersebut berkembanglah sebuah algoritma yang menggunakan sifat keterbagian yaitu algoritma Euclidean. Misalkan m dan n adalah dua buah bilangan bulat yang memenuhi $n > 0$. Jika m dibagi dengan n, maka hasil pembagiannya adalah q (quotient) dan sisanya r (remainder), sehingga

$$m = nq + r \text{ dengan } 0 \leq r < n$$

C. Pembagi Bersama Terbesar (PBB)

Misalkan a dan b adalah dua buah bilangan bulat taknol. Pembagi bersama terbesar (PBB – *greatest common divisor* atau *gcd*) dari a dan b adalah bilangan bulat terbesar d sedemikian hingga

$$d \mid a \text{ dan } d \mid b$$

maka diperoleh $\text{PBB}(a, b) = d$. Selain itu, Misalkan m dan n adalah dua buah bilangan bulat dengan syarat $n > 0$ sedemikian hingga

$$m = nq + r \text{ dengan } 0 \leq r < n$$

maka $\text{PBB}(m, n) = \text{PBB}(n, r)$

D. Relatif Prima

Dua buah bilangan bulat a dan b dikatakan relatif prima jika $\text{PBB}(a, b) = 1$. Jika dikaitkan dengan konsep kombinasi lanjar, jika a dan b relatif prima, maka terdapat bilangan bulat m dan n sedemikian sehingga

$$ma + nb = 1$$

E. Aritmetika Modulo dan Sifatnya

Misalkan a dan m adalah bilangan bulat dengan $m > 0$. Maka operasi $a \bmod m$ akan memberikan sisa a jika dibagi dengan m atau dapat dituliskan dalam notasi matematis sebagai berikut

$$(a \bmod m = r) \mid (a = mq + r) \text{ dengan } 0 \leq r < m$$

m disebut sebagai modulus atau modulo dan hasil aritmetika modulo m terletak pada himpunan $\{0, 1, 2, \ldots, m-1\}$. Beberapa sifat dasar aritmetika modulo bilangan bulat positif, misalkan m adalah sebuah bilangan bulat positif, maka :

a. Jika $a \equiv b \pmod m$ dan c adalah sembarang bilangan bulat, maka
   i. $(a + c) \equiv (b + c) \pmod m$
   ii. $ac \equiv bc \pmod m$
   iii. $a^p \equiv b^p \pmod m$ dengan p bilangan bulat tak negatif
b. Jika $a \equiv b \pmod m$ dan $c \equiv d \pmod m$, maka
   i. $(a + c) \equiv (b + d) \pmod m$
   ii. $ac \equiv bd \pmod m$

F. Kekongruenan dan Kongruensi Lanjar

Misalkan a dan b adalah bilangan bulat dan m adalah bilangan bulat yang memenuhi $m > 0$. Maka operasi

$$a \equiv b \pmod m \Leftrightarrow m \mid (a - b)$$

Jika a tidak kongruen terhadap b dalam modulus m, maka dituliskan $a \not\equiv b \pmod m$.

Kekongruenan lanjar (*linear congruence*) berbentuk

$$ax \equiv b \pmod m$$

dengan a dan b adalah sembarang bilangan bulat, $m > 0$, dan x adalah peubah bilangan bulat, maka solusinya adalah

$$ax = b + km \Longrightarrow x = \frac{b + km}{a}$$

lalu lakukan percobaan untuk $k = 0, 1, 2, \ldots$ dan $k = -1, -2, -3, \ldots$ hingga menghasilkan nilai x yang bulat.

G. Invers Modulo

Jika a dan m relatif prima dan $m > 1$, maka terdapat balikan (invers) dari $a \pmod m$, yaitu terdapat bilangan bulat x sehingga memenuhi kondisi

$$ax \equiv 1 \pmod m$$

atau dapat dinyatakan dalam notasi $a^{-1} \pmod m = x$. Adapun cara menentukan nilai dari invers modulo adalah dengan menyelesaikannya dalam notasi 'sama dengan' berikut

$$ax = 1 + km \Longrightarrow x = \frac{1 + km}{a}$$

lalu lakukan percobaan untuk $k = 0, 1, 2, \ldots$ dan $k = -1, -2, -3, \ldots$ dengan solusinya adalah semua bilangan bulat yang memenuhi.

H. Bilangan Prima dan Karakteristiknya

Bilangan bulat positif p dengan $p > 1$ disebut bilangan prima jika pembaginya hanya 1 dan p. Seluruh bilangan prima adalah bilangan ganjil kecuali bilangan prima pertama yaitu 2. Bilangan bulat selain bilangan prima disebut bilangan komposit.

Karakteristik bilangan prima ini seringkali dimanfaatkan dalam teori bilangan, salah satunya adalah Teorema Fermat yang menyatakan bahwa jika p adalah bilangan prima dan a adalah bilangan bulat yang memebuhi kondisi $\text{PBB}(a, p) = 1$, maka berlaku

$$a^{p-1} \equiv 1 \pmod p$$

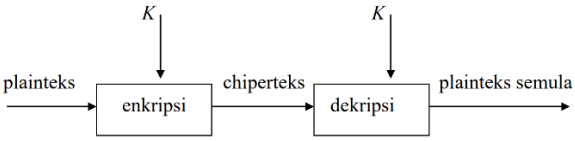
Teorema ini dapat diimplementasikan untuk menyelesaikan persoalan perpangkatan yang besar.

I. Kriptografi

Kata kriptografi atau *cryptography* diketahui berasal dari bahasa Yunani, *kripto* dan *graphia*. Kata *kripto* memiliki arti menyembunyikan, sedangkan *graphia* berarti tulisan. Kriptografi merupakan ilmu yang mempelajari teknik-teknik matematika yang berkaitan dengan aspek keamanan informasi. Contohnya seperti keabsahan data, kerahasiaan data, kredibilitas data, integritas data, dan autentikasi data [1].

Dalam kriptografi, pesan yang dirahasiakan dinamakan plainteks, sedangkan pesan hasil penyandian disebut cipherteks. Untuk mengembalikan pesan yang sudah disandikan ke pesan semula, seseorang harus mengetahui metode penyandian dan kunci penyandian pesan tersebut sehingga hanya orang yang berhak yang bisa mengembalikannya ke keadaan awal. Proses menyandikan

plainteks menjadi cipherteks disebut enkripsi, sedangkan proses membalikkan cipherteks menjadi plainteks disebut dekripsi.



Gambar 2.1 Ilustrasi Skema dalam Kriptografi
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Teori-Bilangan-2020-Bagian3.pdf

Misalkan terdapat sebuah cipherteks yang dilambangkan C dan sebuah plainteks yang dilambangkan dengan P. Notasi matematisnya adalah sebagai berikut :

a. Fungsi enkripsi E yang memetakan P ke C dapat dinyatakan sebagai
$$E(P) = C$$
b. Fungsi dekripsi D yang memetakan C ke P dapat dinyatakan sebagai
$$D(C) = P$$

Adapun proses enkripsi dan dekripsi memerlukan sebuah kunci untuk dapat dipecahkan. Berdasarkan kunci yang digunakan dalam proses enkripsi dan dekripsi, kriptografi dibedakan menjadi kriptografi kunci-simetri (*symetric-key cryptography*) yang biasa disebut kriptografi kunci-privat dan kriptografi kunci-nirsimetri (*asymetric key cryptography*) yang biasa disebut kriptografi kunci-publik [2].

Pada kriptografi kunci-publik terdapat sepasang kunci, satu kunci untuk proses enkripsi dan satu kunci untuk proses dekripsi. Kunci untuk proses enkripsi bersifat umum atau tidak dirahasiakan (kunci publik), oleh karena itu semua orang dapat menggunakannya. Sedangkan kunci untuk proses dekripsi bersifat rahasia (kunci rahasia) dan hanya digunakan oleh penerima pesan. Oleh karena itu, kunci enkripsi tidak boleh sama dengan kunci dekripsi. Salah satu algoritma kriptografi kunci-publik yang paling populer adalah algoritma RSA.

J. Algoritma Kriptografi RSA

Algoritma ini pertama kali dibuat dan diperkenalkan oleh tiga orang peneliti dari MIT (*Massachussets Institute of Technology*), yaitu Ronald Rivest, Adi Shamir, dan Leonard Adleman, pada tahun 1976 [6]. Algoritma ini termasuk dalam algoritma kriptografi nir-simetri dimana kunci untuk melakukan enkripsi berbeda dengan kunci yang digunakan untuk melakukan dekripsi. Dalam algoritma ini, setiap pengguna memiliki sepasang kunci yaitu kunci publik e untuk melakukan enkripsi dan kunci privat p untuk melakukan dekripsi. Kunci publik tidak bersifat rahasia, tetapi kunci privat hanya diketahui oleh pemilik kunci. Berikut adalah beberapa prosedur yang dilakukan untuk menggunakan algoritma kriptografi RSA :

1. Prosedur pembangkitan pasangan kunci

Prosedur	Rahasia/Tidak
Pilih dua buah bilangan prima p dan q	Keduanya rahasia
Hitung nilai n dengan $n = pq$	Tidak
Hitung nilai m dengan $m = (p-1)(q-1)$	Rahasia
Pilih sebuah kunci publik e yang relatif prima terhadap m, yaitu $\text{PBB}(e, m) = 1$	Tidak
Hitung kunci dekripsi d melalui kekongruenan $ed \equiv 1 \pmod m$	Rahasia

Tabel 2.1 Prosedur Pembangkitan Pasangan Kunci
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Teori-Bilangan-2020-Bagian3.pdf

2. Prosedur enkripsi
   a. Nyatakan pesan dalam beberapa blok plainteks $p_1, p_2, p_3, \ldots$ sedemikian hingga $p_i \leq n-1$.
   b. Hitung blok cipherteks c_i untuk blok plainteks p_i dengan persamaan $c_i = p_i^e \pmod n$ dengan e adalah kunci publik. Satukan blok hasil partisi setalah dienskripsi.
3. Prosedur dekripsi
   a. Nyatakan pesan yang terenkripsi dalam sejumlah blok yang jumlahnya sama dengan blok partisi sewaktu melakukan enkripsi.
   b. Hitung blok plainteks p_i untuk blok cipherteks c_i dengan persamaan $p_i = c_i^d \pmod n$ dengan d adalah kunci dekripsi pribadi.

K. Pembangkit Bilangan Acak Semu

Pembangkit Bilangan Acak Semu atau *Pseudo-random Number Generator* atau yang biasa disingkat PRNG adalah sebuah algoritma untuk menghasilkan urutan bilangan yang sifat-sifatnya mendekati sifat-sifat urutan bilangan acak. Urutan yang dihasilkan PRNG tidak benar-benar acak, karena sepenuhnya ditentukan oleh nilai awal, yang disebut *seed* yang mungkin termasuk nilai yang benar-benar acak. Meskipun urutan yang mendekati benar-benar acak dapat dihasilkan menggunakan sebuah alat yang canggih, pembangkit bilangan acak semu ini juga penting dalam praktiknya karena kecepatannya dalam pembuatan nomor dan reproduktifitasnya.

Salah satu metode pembangkitan sebuah bilangan acak semu adalah dengan menggunakan basis kongruensi lanjar atau *Linear Congruential Generator* (LCG) melalui persamaan

$$X_n = (aX_{n-1} + b) (\text{mod } m)$$

Dengan X_n adalah bilangan acak ke-n dari deretnya, X_{n-1} adalah bilangan acak sebelumnya, a adalah factor pengali, b adalah nilai increment, m adalah modulus, dan X_0 adalah umpan (seed).

L. Steganografi

Steganografi berasal dari bahasa Yunani *steganos*, yang artinya tersembunyi atau terselubung, dan *graphein*, yang

artinya menulis. Steganografi adalah ilmu, teknik atau seni menyembunyikan pesan rahasia (*hiding messages*) atau tulisan rahasia (*covered writing*) sehingga keberadaan pesan tidak terdeteksi orang lain kecuali pengirim dan penerima pesan tersebut [3].

Keuntungan steganografi dibandingkan kriptografi saja adalah bahwa pesan rahasia yang dimaksudkan tidak menarik perhatian dirinya sendiri sebagai objek pengawasan. Singkatnya kriptografi adalah praktik melindungi isi sebuah pesan, sedangkan steganografi juga berkaitan dengan penyembunyian fakta bahwa sebuah pesan rahasia sedang dikirim.

III. ANALISIS PERMASALAHAN

A. Implementasi Fungsi Acak dalam Bahasa Pemrograman

Fungsi acak dalam berbagai bahasa pemrograman didasarkan pada apa yang disebut "fungsi *chaos*". Fungsi *chaos* pada dasarnya adalah fungsi yang polanya liar dan sulit diprediksi. Setiap bahasa pemrograman menggunakan fungsi *chaos* yang berbeda-beda, tetapi berikut beberapa di antaranya :

a. Implementasi fungsi *rand()* di C akan bergantung pada versi stdlib mana yang digunakan. Secara umum ini merupakan pembangkit dengan basis kongruensi lanjar (LCG) [7].
b. Class *Random* pada Java memiliki ide yang mirip dengan C. Implementasinya akan tergantung pada JDK apa yang digunakan, tetapi versi *open source* menggunakan yang lagi-lagi menggunakan pembangkit dengan basis LCG [8].
c. Fungsi *built-in random* milik Python menggunakan Algoritma *Mersenne Twister*. Pendekatannya cukup menarik, tentang bagaimana pencipta algoritma ini bisa sampai pada konfigurasi "acak" yang optimal. Ini adalah pembangkit bilangan acak yang sangat efektif (periode perulangan berada pada rentang $2^{607} - 1$ hingga $2^{216091} - 1$) meskipun tidak cukup aman dalam cakupan kriptografi masif karena pola yang masih dapat terdeteksi pada beberapa rentang bilangan [9].

Tentu saja, tidak ada pembangkit bilangan acak yang benar-benar menghasilkan bilangan acak. Hanya saja perilaku dari masing-masing algoritma tersebut yang sulit untuk diprediksi. Pada dasarnya implementasi fungsi acak pada setiap bahasa pemrograman merupakan sebuah algoritma PRNG dan untuk algoritma PRNG apa pun yang dibuat dengan sebuah *seed*, maka akan menghasilkan urutan bilangan yang sejatinya hasilnya dapat diprediksi.

B. Pengembangan Algoritma RSA dengan PRNG

Dengan memanfaatkan sifat dari PRNG kompleks dalam algoritma *Mersenne Twister* yang dimiliki oleh Bahasa pemrograman Python, dapat dikembangkan implementasi dari algoritma kriptografi RSA yang telah dibahas sebelumnya. Pada bagian ini diimplementasikan pembangkit bilangan prima acak semu atau *Pseudo-Random Prime-Number Generator* yang menjadi dasar pembangkitan pasangan kunci enkripsi dan dekripsi yang akan digunakan dalam proses kriptografi. Langkah yang dilakukan adalah melakukan *import* modul *random* yang dimiliki Python, kemudian melakukan penyaringan bilangan terkait apakah bilangan tersebut merupakan bilangan prima atau tidak. Jika tidak maka akan dilaksanakan prosedur rekursif untuk memanggil kembali fungsi *random* hingga ditemukan sebuah bilangan prima di dalam rentang bilangan yang telah didefinisikan di dalam fungsi *random*.

Karena algoritma ini merupakan PRNG, maka ada kemungkinan bahwa angka yang sama akan terulang kembali pada rentang pengulangan tertentu, akan tetapi dengan melakukan penyaringan terhadap bilangan yang masuk, maka pola pengulangan ini dapat sangat diminimalisasi. Oleh sebab itu, diimplementasikan perbandingan kondisional menggunakan *if* dan *else* untuk mengecek apakah kedua kunci yang dibangkitkan bernilai sama atau tidak. Jika sama, maka fungsi *random* akan kembali dipanggil hingga memenuhi kondisi yang diharapkan.

C. Skema Optimalisasi Steganografi

Skema implementasi steganografi yang seringkali digunakan adalah dengan metode *Least Significant Bit* (LSB). Kemudahan dalam proses memasukkan data maupun mengambil data membuat metode ini menjadi salah satu yang sering digunakan. Metode ini bekerja dengan cara mengganti bit terakhir dari representasi susunan bit objek pembangun media [10]. Misalkan akan digunakan sebuah media citra digital berukuran $m \times n$ piksel, maka artinya terdapat mn piksel pada citra tersebut. Setiap piksel terdiri dari kombinasi tiga warna yaitu *red*, *green*, dan *blue* (RGB) yang setiap warnanya dibangun atas 8 bit. Bit terakhir dalam susunan 8 bit tersebut akan diganti untuk menyimpan pesan yang ingin dikirimkan. Dengan implementasi tersebut, tiap piksel dapat menampung 3 bit pesan rahasia. Penggunaan metode LSB ini bertujuan untuk dapat mengurangi perbedaan yang timbul akibat penyisipan pesan. Untuk lebih jelasnya misal terdapat sebuah citra yang memiliki kode biner 3 piksel pertama dengan LSB 011 010 101 sebagai berikut :

R	G	B
11110010	10101111	00010101
10110010	11111111	00101100
01000101	01010000	10100011

Tabel 3.1 Contoh 3 piksel pertama sebuah citra dengan LSB 011 010 101
Sumber : Dokumen pribadi

Jika kemudian disisipkan sebuah pesan yang dalam biner dapat dinyatakan sebagai 111 111 111 ke dalam citra tersebut, maka bit terakhir dari setiap sel (baik dari R, G, maupun B) dalam citra tersebut harus diubah sesuai dengan pesan yang akan dimasukkan. Hasil akhirnya adalah sebagai berikut.

R	G	B
11110011	10101111	00010101
10110011	11111111	00101101
01000101	01010001	10100011

Tabel 3.2 Hasil akhir steganografi dengan mengubah tiga piksel pertama sebuah citra menjadi LSB 111 111 111
Sumber : Dokumen pribadi

Untuk melakukan optimalisasi pada proses steganografi menggunakan metode LSB ini, akan digunakan algoritma RSA dan *Pseudo-Random Prime-Number Generator*. Berikut adalah skema implementasinya.

A. Skema Enkripsi

1. Melakukan pembangkitan kunci dengan mengambil dua buah bilangan prima acak, misalkan p dan q, menggunakan pembangkit bilangan prima acak semu yang telah dioptimalisasi sebelumnya.
2. Melakukan kalkulasi nilai n dengan $n = p \times q$ yang nilainya tidak perlu dirahasiakan dan nilai m dengan $m = (p-1)(q-1)$ yang nilainya dirahasiakan.
3. Memilih sebuah bilangan acak e yang memenuhi kondisi $\text{PBB}(e, m) = 1$ sebagai kunci publik.
4. Meminta alamat citra dan pesan yang akan disisipkan ke dalam citra tersebut.
5. Melakukan konversi masukan pesan ke dalam nilai ASCII untuk setiap karakternya.
6. Melakukan perhitungan untuk masing-masing nilai cipherteks c_i untuk blok plainteks p_i hasil konversi menjadi ASCII tersebut dengan persamaan $c_i = p_i^e \pmod n$ dengan e adalah kunci publik.
7. Satukan hasil perhitungan dari kode cipherteks dengan *padding* yang hanya diketahui oleh pembuat pesan. Akan tercipta sebuah pesan dengan hasil kriptografi yang jauh lebih panjang dan kompleks.
8. Implementasikan skema steganografi dengan melakukan penyisipan pesan pada bit terakhir citra dengan metode LSB.
9. Simpan citra dan implementasi steganografi selesai dilaksanakan. Jangan lupa untuk melakukan penyimpanan terhadap kunci publik dan kunci privat yang telah dibangkitkan sebelumnya.

B. Skema Dekripsi

1. Meminta alamat citra yang akan didekripsi.
2. Melakukan dekripsi dari steganografi untuk mendapatkan sebuah pesan rahasia. Akan tetapi ingat, pesan yang didapat bukanlah pesan yang sebenarnya sebab itu adalah pesan hasil implementasi algoritma RSA.
3. Lakukan partisi kembali sesuai dengan partisi pada saat proses enkripsi. Seseorang yang tidak mengetahui *padding* enkripsi akan mengalami kesulitan pada tahap ini sebab hasil dekripsi akan berbeda jika *padding* yang dikodekan berbeda.
4. Meminta masukan dua buah nilai yaitu nilai n dan kunci publik e.
5. Melakukan perhitungan kunci dekripsi d yang memenuhi persamaan $ed \equiv 1 \pmod m$ dengan nilai m yang hanya diketahui oleh pembuat pesan dan orang yang hendak diberikan pesan.
6. Melakukan perhitungan dekripsi untuk masing-masing nilai blok plainteks p_i dengan blok cipherteks c_i melalui persamaan $p_i = c_i^d \pmod n$ dengan e adalah kunci dekripsi yang telah dicari pada tahap sebelumnya.
7. Akan diperoleh sejumlah karakter satukan hasil perhitungan kode plainteks, maka pesan yang disampaikan melalui steganografi telah berhasil terdekripsi.

Proses penyisipan pesan pada steganografi ini akan melakukan banyak perubahan ke dalam sebuah citra, tetapi dengan perubahan yang jauh lebih cepat dan lebih tidak terlihat. Hal ini bisa terjadi karena proses perubahan karakter dilakukan hanya dalam rentang angka $0-9$ saja dan tidak mengandung karakter yang membuat implementasi LSB perlu mengubah $2-3$ bit dari citra.

Proses dekripsi akan lebih memakan waktu lagi karena pencarian pasangan bilangan prima yang memenuhi kondisi tersebut bukanlah hal yang mudah jika rentang bilangan yang dimaksudkan berada diatas 10^5. Hal ini membuat proses dekripsi sangat memerlukan waktu dan tentu saja hasilnya tidak akan benar jika kunci dan *padding* yang diberikan dan dinyatakan tidak sesuai dengan yang seharusnya. Yang membuatnya lebih menarik lagi, bilangan prima yang dibangkitkan melalui PRNG ini memerlukan periode yang sangat panjang untuk dapat terulang kembali dan pasangan bilangan prima berbeda akan selalu muncul untuk setiap masukan, sehingga akan sulit untuk menyatakan sebuah kunci yang statik setiap kali pesan dimasukkan ke dalam sebuah citra.

IV. IMPLEMENTASI

Pada makalah ini, penulis telah berhasil mengimplementasikan analisis permalasahan dan solusi diatas dalam bahasa pemrograman Python. Alasan pemilihan Bahasa pemrograman python adalah karena banyaknya *library* yang dapat memudahkan proses pengimplementasian algoritma seperti PIL untuk pengolahan citra, *random* untuk implementasi PRNG, dan *stepic* untuk pembuatan steganografi. Berikut merupakan penjelasan lengkap mengenai fungsi, prosedur, serta implementasi algoritma dalam Python.

1. primeNumberGenerator

Pada *source code* berikut, dibangkitkan sebuah bilangan acak menggunakan algoritma PRNG *Mersenne Twister* milik Bahasa pemrograman Python, kemudian dilakukan skema validasi bilangan prima. Tujuan dari pembuatannya adalah untuk membangkitkan pasangan kunci yang acak pada setiap masukannya.

```
# 1. Random prime number generator
def primeNumberGenerator():
    num = random.randint(2, 1000)
    # Checking whether the random integer is prime or not
    for i in range(2,num):
        if (num % i) == 0:
            return primeNumberGenerator()
    return num
```

Gambar 4.1 Fungsi Psedo-Random Prime-Number Generator
Sumber : Dokumen pribadi

2. gcd

Fungsi ini digunakan untuk mencari Pembangi Bersama

Terbesar (PBB) dari dua buah bilangan. Adapun algoritma yang digunakan adalah Algoritma Euclidean.

```
# 2. GCD of two numbers
def gcd(p,q):
    while q != 0:
        p, q = q, p % q
    return p
```

Gambar 4.2 Fungsi untuk menghitung nilai PBB dua bilangan
Sumber : Dokumen pribadi

3. coprimeGenerator

Fungsi ini digunakan untuk mencari sembarang nilai yang merupakan ko-prima dari sutau bilangan. Fungsi gcd digunakan disini untuk mencari kunci publik berdasarkan nilai p dan q yang acak.

```
# 3. Co-prime to a number generator
def coprimeGenerator(num):
    m = random.randint(2, 1000)
    if gcd(num, m) != 1:
        return coprimeGenerator(num)
    else :
        return m
```

Gambar 4.3 Fungsi untuk mencari sembarang bilangan yang ko-prima
Sumber : Dokumen pribadi

4. egcd

Source code dibawah mengimplementasikan fungsi egcd atau *Extended Euclidean Algorithm*. Adapun algoritma tersebut akan menerima dua buah bilangan, misal b dan n, yang mengembalikan tuple (PBB(b,n), a, m) dan memenuhi kondisi Identitas Bezout $ab + mn =$ PBB(b,n).

```
# 4. EGCD of 2 numbers
def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, y, x = egcd(b % a, a)
        return (g, x - (b // a) * y, y)
```

Gambar 4.4 Fungsi untuk mencari nilai egcd dua bilangan
Sumber : Dokumen pribadi

5. modinv

Fungsi ini dibuat untuk melakukan mekanisme balikan (invers) modulo. Fungsi egcd digunakan disini.

```
# 5. Modulo Inverse
def modinv(a, m):
    g, x, y = egcd(a, m)
    if g != 1:
        raise Exception('Modular inverse does not exist')
    else:
        return x % m
```

Gambar 4.5 Fungsi untuk mencari nilai invers modulo
Sumber : Dokumen pribadi

6. Encrypt

Fungsi ini melaksanakan skema enkripsi yang telah dibahas dalam analisis permasalahan. Fungsi ini juga dilengkapi fitur tambahan yaitu waktu eksekusi sehingga bisa diamati perubahannya untuk masing-masing variasi variabel uji coba. Berikut implementasinya.

```
# 6. Encrypting Code
def Encrypt(image_path, s):
    # xx. Starting time
    timestart = time.time()

    # a. Defining 2 number, relatively prime
    p = primeNumberGenerator()
    q = primeNumberGenerator()

    # b. Validating is 2 number is same and setup for p and q value
    sama = (p == q)
    while (sama):
        q = primeNumberGenerator()
        sama = (p == q)

    if (p < q):
        # Swap procedure
        temp = q
        q = p
        p = temp

    # c. Setup value of m and n
    n = p * q
    m = (p -1) * (q - 1)

    # d. Choosing e
    e = coprimeGenerator(m)

    # e. Insert image paths after validating scheme
    img = Image.open(image_path)

    # f. Convert every single alphabet in string to ASCII
    list = [ord(c) for c in s]

    # g. Encrypt the code
    plt = [0 for i in range (len(list))]
    for i in range (len(list)):
        plt[i] = int((list[i] ** e) % n)

    # h. Encrypted words
    res1 = ""
    for i in plt:
        res1 = res1 + chr(i)

    # i. Encrypted codes
    res = ""
    for i in plt:
        res = res + str('{:06}'.format(i))

    # j. Convert the message into UTF-8 format:
    message = res.encode()

    # k. Adding messages encrypted on images
    encoded_img = stepic.encode(img, message)

    # xx. Finish time
    timefinish = time.time()

    # l. Request image file to save
    image_save = input("Insert image path to save the encoded images: ")
    encoded_img.save(image_save)

    # m. Outputting
    print("")
    print("Here is your private key")
    print("n =",n,"e =",e)
    print("")
    print("Save this key whenever you want to decrypt the words!")
    with open('Encrypted.txt', 'w') as file:
        file.write(f'{res}')
    print("Result image was written on",image_save)
    print("The text result is also written on Encrypted.txt\n")
    print("Encryption time:",timefinish-timestart,"second(s).")
```

Gambar 4.6 Fungsi yang mengimplementasikan skema enkripsi
Sumber : Dokumen pribadi

7. Decrypt

Fungsi ini melaksanakan skema dekripsi yang telah dibahas dalam analisis permasalahan. Fungsi ini juga dilengkapi fitur tambahan yaitu waktu eksekusi sehingga bisa diamati perubahannya untuk masing-masing variasi variabel uji coba. Berikut implementasinya.

```python
# 7. Decrypting Code
def Decrypt(image_path, n, e):
    # xx. Starting time
    timestart = time.time()

    # a. Image path to analyze stenography after validating
    image = Image.open(image_path)

    # b. Getting words from picture of steganophy
    kata = stepic.decode(image)

    # c. Splitting the input to array
    array = ['' for i in range (int(len(kata)/6))]

    # d. Parsing the input into few segments
    i = 0
    while i < int(len(kata)):
        array[int(i/6)] = kata[i:i+6]
        i += 6

    # e. Converting to integer, Raising error management
    arrayint = [0 for i in range (len(array))]

    for i in range (len(array)):
        try:
            arrayint[i] = int(array[i])
        except ValueError:
            raise ValueError("There\'s no secret found here")

    # f. Setup value of m using n input
    for i in range(2,n):
        if (n % i == 0):
            p = i
```

```python
    q = n // p
    m = (p - 1) * (q - 1)

    # g. Choosing decription key, fulfilled ed = 1 (mod m)
    d = modinv(e, m)

    # h. Decrypting process
    fin = [0 for i in range (len(arrayint))]

    for i in range (len(arrayint)):
        fin[i] = int((arrayint[i] ** d) % n)

    # i. Resulting the decryption process
    res = ""
    for i in fin :
        res = res + chr(i)

    # xx. Finish time
    timefinish = time.time()

    print("")
    print("Result of Decryption:")
    print(res)
    print("")
    print("Decryption time:",timefinish-timestart,"second(s).")
```

Gambar 4.7 Fungsi yang mengimplementasikan skema dekripsi
Sumber : Dokumen pribadi

8. Program Utama (*main*)

Pada bagian ini, semua fungsi yang ada pada program, diimplementasikan sesuai dengan kerja dan karakteristik dari masing-masing fungsi. Program ini memiliki tampilan dan memungkinkan pengguna untuk melakukan enkripsi dan dekripsi pada sebuah citra melalui pengaplikasian steganografi beradasarkan masukan dan *control flow* yang diinginkan oleh pengguna. Berikut implementasinya.

```python
# ALGORITHM - MAIN PROGRAM
print("Welcome to RSA Stenography Conv!")
print("Decide your choice")
print("1. Encrypting Image")
print("2. Decrypting Image\n")
check = int(input("Your input: "))

# Input processing
valid = False
while (not valid):
    if check == 1:
        image_path = input("Insert image path to encode: ")
        if isExist(image_path):
            s = input("Insert words to encrypt: ")
            Encrypt(image_path,s)
            valid = True
        else :
            print("Image file is not exist")
            valid = False
    elif check == 2 :
        image_path = input("Insert image path to decode: ")
        if isExist(image_path):
            n = int(input("Insert the n value: "))
            e = int(input("Insert the public key: "))
            Decrypt(image_path,n,e)
            valid = True
        else :
            print("Image file is not exist")
            valid = False
    else :
        valid = False
        print("Your input is Invalid!\n")
        check = int(input("Your input : "))
```

Gambar 4.8 Program utama yang mampu melakukan enkripsi dan dekripsi steganografi
Sumber : Dokumen pribadi

V. UJI COBA

Berikut adalah mekanisme uji coba yang dilakukan untuk melihat efektivitas hasil optimalisasi steganografi. Akan digunakan sebuah citra masukan sebagai berikut.

Gambar 5.1 Citra alam musim gugur
Sumber : https://wallpaperaccess.com/colorful-beautiful-nature

Citra asli ini memiliki dimensi 1800×1350 piksel dan disimpan sebagai nature.png.

Pada bagian ini akan dilakukan uji coba untuk dua metode steganografi, metode LSB dan optimalisasi yang dibahas pada makalah ini. Berikut adalah plainteks dengan 200 karakter yang akan dimasukkan ke dalam citra.

> But I must explain to you how all this mistaken idea of denouncing pleasure and praising pain was born and I will give you a complete account of the system, and expound the actual teachings of the gre

1. **Steganografi biasa**
   Steganografi yang biasa ini dilakukan tanpa menggunakan algoritma RSA dan mekanisme *Pseudo-Random Number Generator*, berikut adalah hasilnya.

Gambar 5.2. Citra hasil steganografi biasa
Sumber : Dokumen pribadi

```
Welcome to RSA Stenography - Basic Version!
Decide your choice
1. Encrypting Image
2. Decrypting Image

Your input: 1
Insert image path to encode: nature.png
Insert words to encrypt:
But I must explain to you how all this mistaken idea of denouncing pleasure and
praising pain was born and I will give you a complete account of the system, and
 expound the actual teachings of the gre
Insert image path to save the encoded images: nature_stegbasic.png

Result image was written on nature_stegbasic.png

Encryption time: 1.0819964408874512 second(s).
```

Gambar 5.3. Proses enkripsi memakan waktu 1.082 sekon
Sumber : Dokumen pribadi

```
Welcome to RSA Stenography - Basic Version!
Decide your choice
1. Encrypting Image
2. Decrypting Image

Your input: 2
Insert image path to decode: nature_stegbasic.png

Result of Decryption:
But I must explain to you how all this mistaken idea of denouncing pleasure and
praising pain was born and I will give you a complete account of the system, and
 expound the actual teachings of the gre

Decryption time: 0.23299574851989746 second(s).
```

Gambar 5.4. Proses dekripsi memakan waktu 0.233 sekon
Sumber : Dokumen pribadi

2. **Steganografi yang dioptimalisasi**
   Pada steganografi yang dioptimalisasi ini telah digunakan segala algoritma yang terdapat pada makalah ini. Berikut adalah hasilnya.

```
0083341556860816100048100675540048101196191556861657590816100048101584281476251752711392161
5842800298700481011151107525315842811111800421402388981556860048101178780388980171420048101
1111181392161392160048100816101178781115111657590048101196191115111657590816101111181707701
5842800298700481011151107525315842811111800410038898080620004810075253158428002987038898155
6860029870726541115110029871547070048101751271392161584281111181657591556860182251584280048
1011111800298707525300481017527101822511111811511165759111511002987154707004810175271111811
1511002987004810017142111181657590048100382590388980182250029870048101111180029870752530048
1006755400481001714211151113921613921600481008161011787811151116575908161011111817077015842
8002987004810111511083613158428004810063214038898155686004810111118004810072654038898119
```

Gambar 5.5. Kata hasil enkripsi dengan metode RSA yang dioptimalisasi dengan sistem padding 6 bit
Sumber : Dokumen pribadi

Gambar 5.6. Citra hasil steganografi yang dioptimalisasi
Sumber : Dokumen pribadi

```
Welcome to RSA Stenography - Optimized Version!
Decide your choice
1. Encrypting Image
2. Decrypting Image

Your input: 1
Insert image path to encode: nature.png
Insert words to encrypt:
But I must explain to you how all this mistaken idea of denouncing pleasure and
praising pain was born and I will give you a complete account of the system, and
 expound the actual teachings of the gre
Insert image path to save the encoded images: nature_stegoptim.png

Here is your private key
n = 185047 e = 499

Save this key whenever you want to decrypt the words!
Result image was written on nature_stegoptim.png
The text result is also written on Encrypted.txt

Encryption time: 0.24799609184265137 second(s).
```

Gambar 5.7. Proses enkripsi memakan waktu 0.248 sekon
Sumber : Dokumen pribadi

```
Welcome to RSA Stenography - Optimized Version!
Decide your choice
1. Encrypting Image
2. Decrypting Image

Your input: 2
Insert image path to decode: nature_stegoptim.png
Insert the n value: 185047
Insert the public key: 499

Result of Decryption:
But I must explain to you how all this mistaken idea of denouncing pleasure and
praising pain was born and I will give you a complete account of the system, and
 expound the actual teachings of the gre

Decryption time: 40.4009964466095 second(s).
```

Gambar 5.8. Proses dekripsi memakan waktu 40.401 sekon
Sumber : Dokumen pribadi

Melalui hasil uji coba yang dilakukan, terlihat bahwa berdasarkan waktu yang diperlukan untuk melakukan enkripsi, metode steganografi yang telah dioptimalisasi lebih unggul karena memerlukan waktu enkripsi yang lebih singkat. Hal yang menarik adalah pada saat yang sama, proses penyisipan pesan pada steganografi dengan metode yang dioptimalisasi akan melakukan perubahan yang lebih banyak ke dalam sebuah citra karena lebih banyak karakter yang disisipkan. Hal ini bisa terjadi karena proses perubahan karakter dilakukan hanya dalam

rentang angka 0 – 9 saja, berbeda dengan proses steganografi biasa yang mengandung karakter sehingga membuat implementasi LSB perlu mengubah 2 – 3 bit dari citra yang membuat waktu enkripsi lebih lambat. Secara umum hasil steganografi kedua metode cukup baik dengan hasil yang masih tersamarkan. Hal ini bisa terjadi karena proses perubahan yang terjadi pada citra jauh lebih kecil dari dimensi citra.

Selanjutnya jika dipandang dari proses dekripsi, metode steganografi yang dioptimalisasi kembali unggul karena diperlukan waktu dekripsi yang jauh sekali lebih lambat dibandingkan dengan steganografi biasa. Hal ini dikarenakan pada proses dekripsi dengan metode steganografi yang dioptimalisasi perlu dicari pasangan bilangan prima yang memenuhi kondisi yang diharapkan untuk dapat membuka kunci. Selain itu, proses ini akan jauh lebih aman karena diperlukan sejumlah aspek dan parameter yang hanya dapat diketahui oleh pembuat pesan dan penerima pesan, seperti kunci dekripsi yang bersifat privat, *padding* partisi yang akan menghasilkan hasil yang sangat berbeda jika tidak tepat, serta beberapa konsep teori bilangan lain yang belum tentu diketahui dengan mudah oleh orang banyak.

Uji coba dan pengembangan lebih lanjut dapat dilakukan dengan mengakses laman *github* penulis berikut :

https://github.com/mikeleo03/Optimized-Steganography

VI. KESIMPULAN

Steganografi merupakan sebuah metode kriptografi yang cukup sering digunakan beberapa tahun terakhir untuk menyampaikan informasi secara rahasia antara pengirim dan penerima pesan. Metode optimalisasi steganografi yang dilakukan menggunakan algoritma RSA dan *Pseudo-Random Prime-Number Generator* ini membuat sebuah pesan yang disimpan di dalam citra lebih tersamarkan dengan waktu enkripsi yang lebih efisien dan waktu dekripsi yang lebih lama. Selain itu, pesan yang disamarkan tidak akan dengan mudah tertebak oleh orang lain yang tidak mengetahui parameter rahasia dari proses pengiriman pesan ini. Adapun *link source code* terlampir harapannya dapat dikembangkan lebih lajut untuk menciptakan metode steganografi yang lebih efisien lagi kedepannya.

VII. UCAPAN TERIMA KASIH

Puji Syukur hanya kepada Tuhan Yang Maha Esa karena hanya atas berkat dan limpahan rahmatNya, penulis dapat menyelesaikan makalah ini dengan baik. Terima kasih juga penulis sampaikan kepada Bu Fariska Zakhralativa, S.T, M.T sebagai dosen pengampu dalam mata kuliah IF2120 Matematika Diskrit kelas K02 atas ilmu yang telah diberikan kepada penulis sehingga penulis dapat menyelesaikan makalah ini. Tidak lupa terima kasih juga penulis sampaikan kepada orang tua yang senantiasa memberikan dukungan dan motivasi kepada penulis.

REFERENSI

[1] https://lp2m.uma.ac.id/2022/04/26/mengenal-kriptografi-definisi-tujuan-dan-jenis-jenisnya/, diakses pada 3 Desember 2022
[2] Munir, Rinaldi. "*Kriptografi*". Informatika. Bandung, 2006.
[3] Suhendra, Adhe. Steganografi Pada Citra Terkompresi Metode Huffman. Jurnal Media Informasi Analisa dan Sistem, Volume 1, 2016. Diambil dari https://media.neliti.com/media/publications/282448
[4] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Teori-Bilangan-2020-Bagian1.pdf, diakses pada 3 Desember 2022
[5] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Teori-Bilangan-2020-Bagian2.pdf, diakses pada 3 Desember 2022
[6] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Teori-Bilangan-2020-Bagian3.pdf, diakses pada 3 Desember 2022
[7] https://stackoverflow.com/questions/4768180/rand-implementation diakses pada 3 Desember 2022
[8] https://developer.classpath.org/doc/java/util/Random-source.html diakses pada 3 Desember 2022
[9] https://stackoverflow.com/questions/41998399/how-exactly-does-random-random-work-in-python diakses pada 3 Desember 2022
[10] Steganography Tutorial: Least Significant Bit (LSB). https://www.tutorialspoint.com/what-is-least-significant-bit-algorithm-in-information-security diakses pada 5 Desember 2022
[11] *https://wallpaperaccess.com/colorful-beautiful-nature* diakses pada 5 Desember 2022

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, 6 Desember 2022

Michael Leon Putra Widhi
13521108