KRIPTANALISIS ALGORITMA *PUBLIC KEY PAILLIER SCHEME 3* MENGGUNAKAN ALGORITMA *KRAITCHIK*

SKRIPSI

JOHANA PASKALINA SIHOTANG

211401035

PROGRAM STUDI S-1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2025

KRIPTANALISIS ALGORITMA *PUBLIC KEY PAILLIER SCHEME 3* MENGGUNAKAN ALGORITMA *KRAITCHIK*

SKRIPSI

Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh
Ijazah Sarjana Ilmu Komputer

JOHANA PASKALINA SIHOTANG

211401035

PROGRAM STUDI S-1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2025

PERSETUJUAN

Judul	: KRIPTANALISIS ALGORITMA *PUBLIC KEY PAILLIER SCHEME 3* MENGGUNAKAN ALGORITMA *KRAITCHIK*
Kategori	: SKRIPSI
Nama	: JOHANA PASKALINA SIHOTANG
Nomor Induk Mahasiswa	: 211401035
	: SARJANA (S-1) ILMU KOMPUTER
	: ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
Komisi Pembimbing	:

Pembimbing II	Pembimbing I
Amer Sharif, S.Si, M.Kom NIP. 196910212021011001	Dr. Mohammad Andri Budiman, S.T., M.Comp.Sc., M.E.M NIP. 197510082008011011

Diketahui/Disetujui Oleh
Program Studi Ilmu Komputer
Ketua

Dr. Amalia, S.T., M.T.
NIP. 197812212014042001

PERNYATAAN

KRIPTANALISIS ALGORITMA *PUBLIC KEY PAILLIER SCHEME 3* MENGGUNAKAN ALGORITMA *KRAITCHIK*

SKRIPSI

Saya mengakui bahwa skripsi ini adalah hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan masing-masing yang telah disebutkan sumbernya.

Medan, 25 Maret 2025

Johana Paskalina Sihotang
211401035

PENGHARGAAN

Segala puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa atas kasih dan penyertaan-Nya sehingga penulis dapat menyelesaikan skripsi ini sebagai bagian dari persyaratan untuk meraih gelar Sarjana Komputer di Program Studi S-1 Ilmu Komputer, Universitas Sumatera Utara.

Dalam penyusunan skripsi ini, penulis menerima banyak dukungan dari berbagai pihak. Oleh sebab itu, penulis ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada:

1. Bapak Prof. Dr. Muryanto Amin, S.Sos., M.Si., selaku Rektor Universitas Sumatera Utara.
2. Ibu Dr. Maya Silvi Lydia, B.Sc., M.Sc., selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
3. Bapak Dr. Mohammad Andri Budiman, S.T., M.Comp.Sc., M.E.M., selaku Wakil Dekan I Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
4. Ibu Dr. Amalia, S.T., M.T., selaku Ketua Program Studi S-1 Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
5. Ibu Sri Melvani Hardi, S.Kom., M.Kom., selaku Sekretaris Program Studi S-1 Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
6. Bapak Dr. Mohammad Andri Budiman, S.T., M.Comp.Sc., M.E.M., selaku Dosen Pembimbing I yang senantiasa meluangkan waktu untuk memberikan bimbingan dan saran kepada penulis dalam proses penyelesaian skripsi ini.
7. Bapak Amer Sharif, S.Si, M.Kom., selaku Dosen Pembimbing I yang senantiasa meluangkan waktu untuk memberikan bimbingan dan saran kepada penulis dalam proses penyelesaian skripsi ini.
8. Ibu Anandhini Medianty Nababan, S.Kom., M.T sebagai Dosen Penguji yang turut memberikan kritik membangun serta arahan yang membantu penulis dalam menyempurnakan karya ini.

9. Ibu Hayatunnufus, S.Kom, M.Cs sebagai Dosen Penguji yang telah memberikan masukan berharga dan saran konstruktif kepada penulis.
10. Ibu Desilia Selvida, S.Kom., M.Kom, selaku Dosen Pembimbing Akademik yang telah membimbing dan mengarahkan penulis sepanjang masa studi.
11. Seluruh Bapak dan Ibu Dosen dan Staf Pegawai Program Studi S-1 Ilmu Komputer, yang telah membimbing penulis sepanjang masa studi.
12. Kedua orang tua penulis yang terkasih, Juster Sihotang dan Manna Sitanggang, atas segala bentuk pengorbanan, kasih sayang, serta doa-doa tulus yang tiada henti-hentinya di setiap proses hidup penulis dan dukungannya yang selalu menemani di setiap langkah perjalanan ini.
13. Saudara kandung penulis, kakak Josepha Sihotang, dan Adik Mikael Sihotang, yang selalu mendukung serta mendoakan di setiap proses yang penulis hadapi.
14. Sahabat penulis Cahaya Tampubolon, Rodiatul Sitepu, Yessica Situmorang, yang selalu mendukung, memberikan motivasi serta bantuan kepada penulis sepanjang masa studi.
15. Sahabat penulis dalam masa perkuliahan Agatha Sinaga, Sammytha Siagian, dan Elin Br Ginting yang selalu memberikan semangat, dukungan, serta turut mendorong penulis dalam proses penyelesaian skripsi.
16. Christi Siagian, Natalia Sibuea, Sadela Panjaitan dan seluruh teman-teman penulis, yang selalu mendukung dan memberikan motivasi selama penyusunan skripsi.

Penulis menyadari bahwa masih ada kekurangan dalam penelitian ini, dan menerima segala masukan yang konstruktif dalam peningkatan kualitas penelitian ini. Penulis berharap penelitian ini dapat membantu dan bermanfaat bagi seluruh pihak yang membutuhkannya.

Medan, 25 Maret 2025

Johana Paskalina Sihotang

211401035

ABSTRAK

Algoritma *Paillier Scheme 3* merupakan varian dari skema enkripsi kunci publik dengan sifat homomorfik aditif yang dirancang untuk efisiensi proses dekripsi. Namun, meskipun memiliki basis matematika yang kompleks, algoritma ini tetap memerlukan pengujian terhadap potensi serangan kriptanalisis. Penelitian ini bertujuan untuk menganalisis kerentanan Algoritma *Paillier Scheme 3* terhadap serangan faktorisasi kunci publik menggunakan Algoritma *Kraitchik*. Penelitian ini mencakup pembangkitan kunci publik dan rahasia, proses enkripsi *plaintext*, dekripsi *ciphertext*, serta kriptanalisis dengan memfaktorkan modulus kunci publik. Efektivitas algoritma diukur berdasarkan waktu eksekusi serta keberhasilan dalam memperoleh kunci privat. Hasil menunjukkan bahwa Algoritma *Kraitchik* mampu membongkar kunci rahasia *Paillier Scheme 3* pada ukuran kunci tertentu. Selain itu, peningkatan panjang kunci dan ukuran dokumen *plaintext* berbanding lurus dengan waktu eksekusi proses enkripsi dan waktu eksekusi proses dekripsi.

Kata Kunci: Algoritma *Paillier Scheme 3*, Algoritma *Kraitchik*, Kriptanalisis, Kriptografi, Faktorisasi, Keamanan Data.

CRYPTANALYSIS OF PUBLIC KEY PAILLIER SCHEME 3 USING KRAITCHIK'S ALGORITHMA

ABSTRACT

The Paillier Scheme 3 algorithm is a variant of public key encryption with additive homomorphic properties, designed for efficient decryption. Despite its complex mathematical foundation, this algorithm still requires testing against potential cryptanalysis attacks. This study aims to analyze the vulnerability of the Paillier Scheme 3 algorithm to public key factorization attacks using the Kraitchik Algorithm. The research includes the generation of public and private keys, plaintext encryption, ciphertext decryption, and cryptanalysis through modulus factorization. The algorithm's effectiveness is measured based on execution time and the success in retrieving the private key. Results show that the Kraitchik Algorithm is capable of breaking the private key of Paillier Scheme 3 for certain key sizes. Additionally, an increase in key length and plaintext document size corresponds to longer encryption and decryption times.

***Keywords**: Paillier Scheme 3 Algorithm, Kraitchik Algorithm, Cryptanalysis, Cryptography, Factorization, Data Security.*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I

PENDAHULUAN

1.1. Latar Belakang

Dalam perkembangan pesat era digital, berbagi data dan informasi menjadi lebih praktis, tetapi juga meningkatkan risiko penyalahgunaan data oleh pihak yang tidak bertanggung jawab. Ancaman keamanan ini dapat menyebabkan kerugian finansial, pelanggaran privasi, serta kerahasiaan informasi selama proses pertukaran data (Hapsari & Pambayun, 2023). Karena itu, keamanan sebuah data kini telah menjadi hal yang krusial, di mana berbagai metode enkripsi digunakan untuk melindungi informasi dari ancaman siber yang terus berkembang (Suryawijaya, 2023). Enkripsi berperan penting dalam memastikan bahwa data yang dipertukarkan tetap rahasia dan aman dari akses tidak sah setiap saat, sehingga menjaga kepercayaan pengguna dan mendukung integritas sistem informasi secara keseluruhan (Google Cloud, 2024).

Kriptografi adalah metode untuk mengamankan informasi dan komunikasi melalui penggunaan kode khusus, sehingga hanya pihak tertentu yang dapat mengakses informasi tersebut. Untuk menguji keamanannya, kriptografi akan dianalisis menggunakan kriptanalisis, yaitu ilmu yang berfokus pada pemecahan sandi atau enkripsi. Kriptanalisis memungkinkan untuk mengidentifikasi kelemahan atau celah dalam suatu sistem enkripsi, sehingga *plaintext* dapat ditemukan dari *ciphertext* yang telah disembunyikan sebelumnya (Budiman et al., 2019).

Algoritma *Paillier Scheme 3* adalah salah satu skema enkripsi yang dirancang untuk memberikan keamanan tinggi melalui kompleksitas matematisnya. Meski memiliki dasar keamanan yang kuat, skema ini tetap memerlukan pengujian dan analisis berkelanjutan mengingat ancaman kriptanalisis yang terus berkembang. Dalam penulisan ini, Algoritma Kraitchik diterapkan untuk memfaktorkan kunci publik dari Paillier Scheme 3 dengan tujuan memperoleh kunci rahasianya.

Algoritma *Paillier* diperkenalkan oleh P. Paillier pada tahun 1999 dan merupakan salah satu algoritma asimetris dalam kriptografi kunci publik. Algoritma ini digunakan kompleksitas tinggi dalam proses komputasi enkripsi dan dekripsi. Di mana melibatkan dua kali pemangkatan, satu kali perkalian, dan satu kali operasi modulo. Kinerja keamanannya yang tinggi sangat bergantung pada kesulitan dalam faktorisasi bilangan. Dalam penelitiannya, Paillier menjelaskan dua sistem kriptografi parsial-homomorfik, yaitu *Scheme 1* dan *Scheme 3*. *Scheme 1* merupakan versi dasar dari skema *Paillier*, sementara *Scheme 3* adalah varian yang menawarkan proses dekripsi yang lebih cepat (Jost et al., 2015).

Algoritma *Kraitchik* adalah metode faktorisasi yang memanfaatkan sifat bilangan kuadrat dan merupakan bagian dari teknik yang lebih umum dikenal sebagai "metode kuadrat" atau "metode perbedaan kuadrat". Prinsip dasar algoritma ini adalah mencari pasangan bilangan yang kuadratnya memiliki selisih yang merupakan kelipatan dari modulus. Selisih ini digunakan untuk mengidentifikasi faktor-faktor dari dari bilangan tersebut. Mengingat bahwa keamanan *Paillier Scheme 3* sangat bergantung pada kesulitan dalam memfaktorkan modulus yang terdiri dari dua bilangan prima besar, Algoritma *Kraitchik* menjadi relevan dalam mengevaluasi potensi kerentanan skema ini terhadap serangan faktorisasi.

Dalam studi berjudul "*Secret Sharing Authentication Key Agreement*", skema *Paillier* dijelaskan secara rinci dalam tesis oleh Ryšavá (2022), yang membahas berbagai algoritma, termasuk *Skema 1* dan *Skema 3*. Selain itu, Ma et al. (2021) dalam penelitian mereka berjudul "*Optimized Paillier's Cryptosystem with Fast Encryption and Decryption*" mengusulkan optimasi baru untuk skema enkripsi homomorfik aditif *Paillier*. Mereka menetapkan sifat *one-wayness* (ketidakmungkinan membalikkan proses enkripsi tanpa kunci) dan keamanan semantik dari skema *Paillier* yang dioptimalkan.

Selanjutnya, penelitian oleh Lydia et al. (2021) dengan judul "*Factorization of Small RPrime RSA Modulus Using Fermat's Difference of Squares and Kraitchik's Algorithms in Python*" menerapkan teknik pemfaktoran dengan mencoba memfaktorkan modulus kecil dari *RPrime RSA* menggunakan dua algoritma, yaitu algoritma *Fermat's difference of squares* dan algoritma *Kraitchik*. Dalam penelitian ini, kedua algoritma tersebut terbukti efektif sebagai metode untuk memfaktorkan modulus kecil dari *RPrime RSA*.

Mengacu pada latar belakang yang telah dijelaskan, penelitian ini akan mengambil topik dengan judul "KRIPTANALISIS ALGORITMA *PUBLIC KEY PAILLIER SCHEME 3* DENGAN ALGORITMA *KRAITCHIK*". Penelitian ini memiliki tujuan untuk membongkar kunci rahasia dari algoritma *Paillier Scheme 3* dengan cara memfaktorkan kunci publik yang dimiliki algoritma *Paillier Scheme 3* menggunakan algoritma *Kraitchik*. Dengan menggunakan pendekatan ini, diharapkan dapat memperdalam pemahaman mengenai kerentanan algoritma *Paillier Scheme 3* dan kontribusi signifikan dalam bidang kriptanalisis untuk meningkatkan keamanan skema enkripsi kunci publik di masa depan.

1.2. Rumusan Masalah

Dalam era digital yang semakin kompleks dan dipenuhi dengan ancaman terhadap keamanan data, penting untuk terus mengembangkan dan menguji metode enkripsi yang ada. Algoritma *Paillier Scheme 3*, meskipun dikenal memiliki tingkat keamanan yang tinggi berkat kompleksitas matematisnya, tetap perlu dievaluasi secara berkala untuk memastikan ketahanannya terhadap berbagai teknik kriptanalisis yang terus berkembang. Oleh karena itu, penelitian ini akan menguji efisiensi dalam memecahkan kunci rahasia algoritma *Paillier Scheme 3* dengan cara memfaktorkan kunci publik menggunakan pendekatan algoritma *Kraitchik*.

1.3. Batasan Masalah

Berikut adalah batasan-batasan dalam penelitian ini untuk menjaga fokus agar tetap sesuai dengan rumusan masalah:

1. Fokus utama dari penelitian ini adalah penerapan algoritma *Paillier Scheme 3* dan algoritma *Kraitchik*.
2. Tipe data yang digunakan dalam penelitian ini adalah *file text* dengan format .doc/docx , .pdf.
3. Plaintext dan *ciphertext* berbasis ASCII.
4. Bahasa yang digunakan untuk membangun program adalah bahasa *Python*.

1.4. Tujuan Penelitian

Tujuan dari penelitian ini adalah memperoleh kunci rahasia dengan cara memfaktorkan kunci publik *n* algoritma *Paillier Scheme 3* menggunakan metode *Kraitchik*

1.5. Manfaat Penelitian

Adapun manfaat yang diharapkan dari penelitian ini antara lain:

1. Menilai efektivitas algoritma *Kraitchik* dalam membongkar kunci pada algoritma *Paillier Scheme 3* sebagai bahan evaluasi untuk pengembangan metode kriptanalisis yang lebih optimal.
2. Mengidentifikasi kelemahan algoritma *Paillier Scheme 3* untuk menghadapi serangan faktorisasi sebagai bahan evaluasi untuk mengembangkan dan memperkuat keamanan algoritma *Paillier Scheme 3*.

1.6. Metodologi Penelitian

1. Studi Pustaka

Tahap ini dilakukan dengan mengumpulkan data dan mempelajari literatur dari berbagai buku, artikel, jurnal, makalah, dan situs internet yang relevan dengan konsep kriptografi, algoritma *Paillier Scheme 3*, teknik-teknik kriptanalisis *Paillier Scheme 3*, serta dasar matematika terkait seperti teori bilangan dan pemfaktoran bilangan menggunakan algoritma *Kraitchik*.

2. Perancangan Sistem

Tahap ini mencakup masalah yang dihadapi dengan mengidentifikasi permasalahan, memodelkan masalah secara konseptual menggunakan *flowchart*, tujuan, dan solusi yang diusulkan. Kemudian dilakukan perancangan antarmuka aplikasi yang mencakup tahapan-tahapan operasi dalam proses pengolahan data dan prosedur untuk mendukung aplikasi tersebut.

3. Impelementasi Sistem

Tahap ini akan menerapkan hasil konseptual analisis dan perancangan sistem dengan pengkodean (*coding*) program.

4. Pengujian Sistem

Tahap ini mencakup pengujian pada sistem yang berguna untuk mengevaluasi semua kesalahan dan kekurangan yang ada dan mengumpulkan data behubungan dengan efisiensi waktu sistem. Pengujian yang dilakukan dengan menjalankan sistem memberikan berbagai masukan pada setiap fungsi dan melihat keluaran yang dihasilkan oleh sistem.

5. Dokumentasi

Penulis akan melakukan dokumentasi penelitian dengan menulis laporan berisi kesimpulan hasil analisa dan pengujian ke dalam bentuk skripsi, yang menggambarkan program yang dibangun serta hasil dari penelitian.

1.7. Sistematika Penulisan

Struktur skripsi yang diterapkan dalam penulisan penelitian ini terdiri dari lima bab utama, yaitu:

BAB 1 PENDAHULUAN

Bab ini menguraikan sejumlah aspek penting terkait penelitian ini, termasuk penjelasan mengenai latar belakang yang menjadi dasar dilakukannya penelitian ini, rumusan masalah yang dibahas, batasan masalah yang ditetapkan, tujuan dari penelitian ini, manfaat yang diharapkan, serta beberapa penelitian terkait yang digunakan sebagai rujukan. Bab ini juga menjelaskan metodologi yang diterapkan dan serta struktur penulisan skripsi yang akan digunakan dalam penyusunan laporan penelitian ini.

BAB 2 LANDASAN TEORI

Bab ini membahas kriptografi beserta komponennya, serta algoritma yang digunakan dalam kriptanalisis, yaitu Algoritma *Paillier Scheme 3* serta Algoritma *Kraitchik.*

BAB 3 ANALISIS DAN PERANCANGAN

Bab ini menjelaskan alur dan proses kerja kriptanalisis menggunakan Algoritma *Paillier Scheme 3* dan Algoritma *Kraitchik.*

BAB 4 IMPLEMENTASI DAN PENGUJIAN SISTEM

Bab ini meliputi penerapan sistem yang telah dibangun serta hasil uji coba sistem menggunakan data teks yang dimasukkan ke dalam sistem.

BAB 5 KESIMPULAN DAN SARAN

Bab ini memuat kesimpulan yang diambil dari pembahasan tiap bab, serta rekomendasi yang diberikan oleh peneliti sebagai kontribusi untuk penelitian di masa mendatang.

BAB II

LANDASAN TEORI

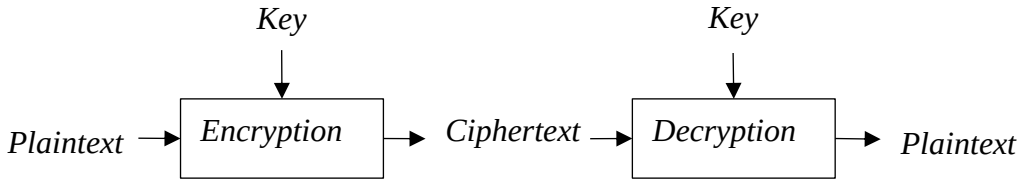
2.1. Kriptografi

Kata kriptografi berasal dari bahasa Yunani, yaitu *cryptos* yang berarti rahasia dan *graphein* yang berarti tulisan. Secara harfiah, kriptografi merujuk pada cara menulis pesan secara tersembunyi untuk menjaga kerahasiaan informasi yang disampaikan (Nanda et al., 2023). Kriptografi adalah cabang ilmu yang menggunakan teknik matematika berhubungan erat dengan perlindungan informasi termasuk kerahasiaan, keutuhan data, dan verifikasi entitas (Maulana Iqbal, 2024). Tujuan kriptografi adalah untuk melindungi, menjaga, serta memastikan kerahasiaan suatu informasi dari pihak-pihak yang tidak sah.

Pesan yang akan dikirim dan memuat data asli disebut *plaintext* . Melalui penggunaan teknik atau algoritma tertentu, *plaintext* diubah menjadi bentuk terenkripsi yang berbeda dari data aslinya. Proses ini dikenal sebagai enkripsi, dan hasil dari proses enkripsi dikenal sebagai *ciphertext*. Ciphertext merupakan pesan yang telah terenkripsi dan tidak dapat dipahami tanpa dekripsi terlebih dahulu. Pihak yang berwenang untuk mendapatkan data akan mengetahui algoritma yang diterapkan dan memiliki kunci untuk mengubah *ciphertext* kembali ke pesan asli. Proses ini dikenal dengan nama dekripsi. Data yang telah dienkripsi dibutuhkan untuk proses menyimpan atau mengirim data. Agar proses enkripsi dan dekripsi dapat dilakukan, baik pihak pengirim maupun pihak penerima harus memahami algoritma kriptografi yang digunakan dan mempunyai kunci yang tepat.

Tingkat perlindungan data terenkripsi terhadap upaya dekripsi paksa oleh pihak yang tidak berwenang ditentukan oleh seberapa kuat suatu algoritma yang diterapkan dan kerahasiaan kunci. Kuat tidaknya suatu algoritma enkripsi dan dekripsi sangat bergantung pada penerapan persamaan matematika.

Semakin kompleks dan banyak perhitungan dari persamaan matematika yang digunakan, semakin tinggi tingkat keamanan data yang terenkripsi. Penggunaan kecepatan dan ketepatan komputer sangat mendukung dalam proses ini.Kerahasiaan kunci bergantung pada metode penyimpanan dan pembagiannya kepada pihak yang berwenang, karena kunci ini diperlukan untuk melakukan dekripsi, semakin baik cara penyimpanan serta pembagiannya semakin aman data terenkripsi.



Gambar 2.1 Proses Enkripsi dan Dekripsi Data

2.1.1. Tujuan Kriptografi

Kriptografi memiliki empat tujuan dalam mencapai fundamental keamanan yaitu sebagai berikut:

1. Kerahasiaan (*Confidentiality*)

Kerahasiaan berfungsi untuk melindungi informasi dari setiap pihak yang tidak berwenang agar tidak dapat diakses. Kerahasiaan memastikan bahwa data tetap bersifat pribadi dalam tiga situasi berbeda yakni saat data disimpan *(at rest)*, saat data dikirim *(in transit)*, dan saat data digunakan *(in use)*

2. Autentikasi (*Authentication*)

Autentikasi merupakan verifikasi identitas. Hal ini penting untuk memastikan identitas pihak yang ingin mengakses sistem informasi (*entity authentication*) serta keaslian data dari sistem informasi itu sendiri (*data origin authentication*). Ketika mengirim atau menerima informasi, kedua belah pihak harus memverifikasi pengirim dari pesan merupakan individu yang sesuai dengan yang diklaim.

3. Integritas data (*Data integrity*)

Integritas data memiliki tujuan mencegah perubahan terhadap informasi tanpa izin. Agar integritas data terjaga, sistem informasi harus dapat mendeteksi adanya manipulasi data meliputi penyisipan, penghapusan, maupun perubahan data. Integritas data ditegakkan melalui penggunaan ringkasan pesan terenkripsi, yang

dikenal sebagai tanda tangan digital *(digital signatures)*, yang dibuat saat pesan dikirim. Penerima pesan cukup memverifikasi bahwa tanda tangan digital pesan tersebut valid, memastikan bahwa pesan tidak diubah selama transmisi. Integritas dapat ditegakkan oleh sistem kriptografi kunci publik maupun kunci rahasia.

4. *Nonrepudiation*

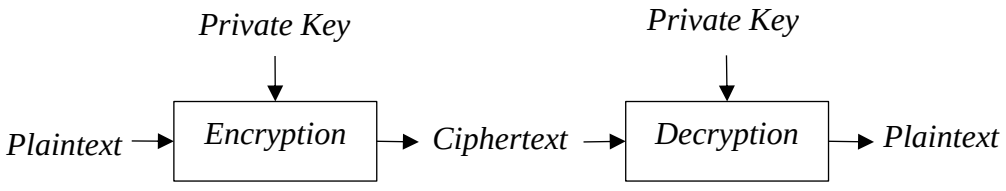
Nonrepudiation memastikan penerima bahwa pesan adalah benar berasal dari pengirim yang sah, bukan dari pihak yang menyamar. Hal ini menghindari kemungkinan baik pengirim ataupun penerima untuk membantah bahwa mereka telah mengirim atau menerima suatu pesan. Ketika pesan dikirim, penerima dapat membuktikan pesan tersebut adalah benar berasal dari pengirim yang tercatat. Sebaliknya, pengirim dapat membuktikan bahwa pesan yang dikirimnya diterima oleh pihak yang dituju.

2.1.2. Jenis Kunci Kriptografi

Terdapat dua jenis utama sistem kriptografi menerapkan kerahasiaan:

1. Sistem kriptografi simetris

Kunci simetris memanfaatkan satu kunci yang sama untuk kedua proses enkripsi dan proses dekripsi. Artinya kunci yang dipakai untuk mengenkripsi pesan juga digunakan untuk mendekripsi pesan tersebut. Sistem kriptografi ini umumnya lebih cepat dan efisien, tetapi memiliki tantangan dalam distribusi kunci karena kunci harus dijaga kerahasiaannya dan dikirim dengan aman kepada penerima. Algoritma kriptografi yang menggunakan jenis ini ada algoritma AES (*Advanced Encryption Standard*), algoritma DES (*Data Encryption Standard*), algoritma 3DES (*Triple DES*), dll.

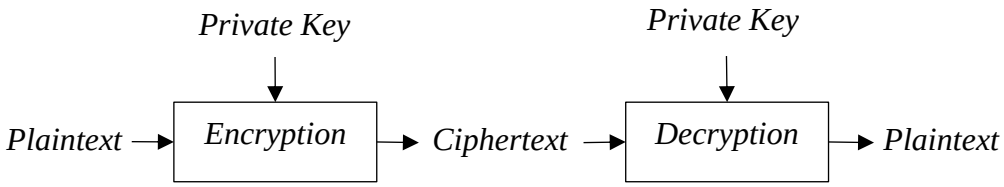


Gambar 2.2 Skema dari Kriptografi Kunci Simetris

2. Sistem kriptografi asimetris

Kriptografi asimetris memanfaatkan penggunaan pasangan kunci publik dan kunci

rahasia. Kunci publik digunakan untuk proses enkripsi dan kunci rahasia digunakan untuk proses dekripsi. Kunci publik dapat didistribusikan secara bebas, sementara kunci rahasia harus dijaga kerahasiaannya. Contoh Algoritma yang menggunakan system kriptografi asimetris adalah algoritma RSA (*Rivest-Shamir-Adleman*), algoritma ECC (*Elliptic Curve Cryptography*), algoritma DSA (*Digital Signature Algorithm*).



Gambar 2.3 Skema dari Kriptografi Kunci Asimetris

2.2. Kriptanalisis

Kriptanalisis diperkenalkan oleh seorang ilmuwan Arab pada abad ke-9 yang dikenal sebagai Al-Kindi (Munir, 2019). Kriptanalisis adalah bidang ilmu yang mempelajari cara untuk mengembalikan *ciphertext* menjadi *plaintext* tanpa memerlukan informasi tentang kunci yang digunakan. Orang yang melakukan kriptanalisis disebut kriptanalis. Dalam hal ini, kriptografer (*cryptographer*) mengubah *plaintext* menjadi *ciphertext* dengan algoritma dan kunci tertentu, sedangkan kriptanalis berupaya untuk membongkar *ciphertext* tersebut guna mencari *plaintext* atau kunci yang diterapkan.

Ketika membahas serangan terhadap sistem kriptografi, diasumsikan bahwa kriptanalis sudah memiliki pengetahuan tentang algoritma yang digunakan. Oleh karena itu, keamanan sistem kriptografi bergantung sepenuhnya pada kekuatan kunci. Prinsip ini berlandaskan pada Prinsip *Kerckhoff* (1883), yang menyatakan bahwa "*Algoritma kriptografi tidak perlu dirahasiakan dan dapat dianggap diketahui oleh musuh tanpa menimbulkan masalah. Namun, kunci yang digunakan dalam algoritma tersebut harus dianggap sebagai informasi yang rahasia*." Alasan pertama adalah bahwa lebih mudah bagi pihak terkait untuk menjaga kerahasiaan kunci yang dihasilkan secara acak daripada algoritma itu sendiri. Kedua, jika kunci berhasil diungkap, pihak terkait dapat dengan lebih mudah mengganti kunci tersebut dibandingkan harus mengganti algoritma (Muchlis et al., 2017).

2.3. Dasar Matematika Kriptografi

Terdapat beberapa dasar dari Matematika Kriptografi yang digunakan untuk memahami dan mengerjakan algoritma *Pailier Scheme* 3 dan algoritma *Kraitchik.*

2.3.1. Operasi Modulus

Operasi modulus adalah operasi matematika yang menentukan sisa pembagian dari dua bilangan. Dengan kata lain, jika kita membagi suatu bilangan *a* dengan bilangan *b*, hasil modulus *a* mod *b* adalah sisa dari pembagian tersebut.

Notasi Modulus:

$a \bmod b = \text{sisa dari } \frac{a}{b}$

Contoh: Berapa hasil 17 mod 5?

$17 \bmod 5 = \frac{17}{5} = 3 \text{ (dengan sisa 2)}$

Jadi, 17 mod 5 = 2

2.3.2. Bilangan Prima

Bilangan prima ialah suatu bilangan bulat positif yang lebih besar dari 1 (satu), yang hanya habis dibagi oleh 2 buah bilangan yaitu 1 dan bilangan itu sendiri. Contoh bilangan prima 2, 3, 5, 7, 11, 13, dan seterusnya.

2.3.3. *Prime Generator*

Untuk membuktikan suatu bilangan merupakan bilangan prima maka kita dapat menggunakan metode *prime generator*.

Teorema Wilson: Menyatakan bahwa setiap bilangan prima *p* membagi (*p*−1)!+1, di mana n! adalah notasi faktorial untuk 1×2×3×4×⋯× *n* (Britannica, 2024). Artinya, *p* adalah bilangan prima jika $(p-1)! + 1 \equiv 0 \pmod{p}$

Contoh : Apakah 23 bilangan prima?

p = 7

(*p* – 1)! = (7 - 1)! = 6!

= (6 x 5) (4 x 3) (2 x 1)

= (30) (12) (2)

= (30 mod 7) (12 mod 7) (2 mod 7)

= (2) (5) (2)

= $20 + 1 \equiv 0 \pmod 7$

= $21 \equiv 0 \pmod 7$

Maka 11 adalah bilangan prima.

2.3.4. *Greatest Common Divisor (GCD)*

Greatest common divisor (GCD) adalah faktor persekutuan terbesar dari dua bilangan bulat positif yang dapat membagi kedua bilangan tanpa menyisakan sisa. Untuk menghitung *Greatest Common Divisor* dari dua bilangan dapat menggunakan *Euclidean Algorithm.*

Algoritma ini bekerja dengan prinsip berikut:

- Jika $a > b$, maka:
  GCD(a, b) = GCD(b, a mod b)
- Proses ini diulangi hingga $b = 0$. Pada titik ini, nilai GCD adalah bilangan a yang tersisa.

Contoh: Berapa GCD dari 48 dan 18?

48 mod 18 = 12

18 mod 12 = 6

12 mod 6 = 0

Maka GCD dari 48 dan 18 adalah 64

2.3.5. *Least Common Multiple (LCM)*

Least Common Multiple (LCM) adalah bilangan bulat positif terkecil yang merupakan kelipatan dari dua bilangan. Jika GCD mencari faktor terbesar yang dapat membagi kedua bilangan, LCM mencari bilangan terkecil yang merupakan kelipatan dari kedua bilangan tersebut. Berikut rumus untuk menghitung LCM:

$$\text{LCM}\ (a, b) = \frac{|\ a \text{ x } b|}{\text{GCD}\ (a, b)}$$

Contoh: Berapa LCM dari 48 dan 18?

$$\text{LCM}\ (48, 18) = \frac{|\ 48 \text{ x } 18|}{\text{GCD}\ (48, 18)}$$

- Mulai dengan GCD(48, 18)

  48 mod 18 = 12

  18 mod 12 = 6

  12 mod 6 = 0

Jadi, GCD(48, 18) = 6

- Hitung LCM $(48, 18) = \frac{|48 \text{ x } 18|}{\text{GCD } (48,18)} = \frac{864}{6} = 144$

Jadi, LCM dari 48 dan 18 adalah 144.

2.3.6. *Invers Modulo*

Syarat *m* (mod *n*) memiliki invers adalah GCD (*m*,*n*) = 1. m^{-1} disebut invers dari *m* (mod *n*) apabila $m^{-1}m \equiv 1 \pmod{n}$.

2.3.6.1. Persamaan *Diophantine Linier (PDL)*

Persamaan dalam bentuk $ax + by = c$. Dimana, *x* dan *y* adalah peubah yang ingin dicari, *a*,*b*,*c* adalah konstanta yang ditentukan.

2.3.6.2. PDL dengan Identitas *Bizout* (IB)

PDL dengan Identitas *Bizout* (IB) adalah PDL dengan *c* = GCD (*a*,*b*).

Contoh: Apakah 31*x* + 11*y* = 1 adalah PDL dengan IB?

$a = 31$ dan $b = 11$

$\text{GCD}(31,11)$

- $31 \bmod 11 = 9$
- $11 \bmod 9 = 2$
- $9 \bmod 2 = 1$
- $2 \bmod 1 = 0$

$\text{GCD}(31,11)$ = 1, maka 31*x* + 11*y* = 1 adalah PDL dengan IB

2.3.6.3. Algoritma *Extended Euclidean*

Algoritma *Extended Euclidean* adalah algoritma untuk mencari solusi dari PDL dengan IB. Algoritma ini juga digunakan untuk menemukan invers modulo.

Contoh, berapa invers dari 31(mod 11)?

1. Tentukan GCD(31, 11) = 1, sehingga memenuhi syarat invers modulo.
2. Defenisikan 31(mod 11) menjadi Persamaan *Diophantine* Linier.
   ax + by = c, dimana c adalah GCD(31, 11)
   maka 31x + 11y = 1
3. Tentukan :
   a. Jika $x > y$ maka

x	y
1	0
0	1

   b. Jika $x < y$ maka

x	y
0	1
1	0

4. Nilai *r* pada dua baris pertama adalah hasil perkalian konstanta yang ditentukan dengan nilai peubah yang ingin dicari 31(1) + 11(0) dan 31(0) + 11(1). Karena 31 > 11 maka gunakan kondisi pertama.
5. Nilai *k* adalah kelipatan 11 yang mendekati 31.
6. Baris ketiga adalah hasil baris pertama dikurang baris kedua dikali nilai *k*. Contoh 1 – 0(2) = 1.
7. Jika *r* = 1 (GCD(a,b)) maka berhenti, jika tidak lakukan iterasi mencari nilai *k* hingga *r* = c atau GCD(a,b). Sehingga invers 31(mod 11) adalah 5.

x	**y**	**r**	**k**
1	0	31	
0	1	11	2
1	-2	9	1
-1	3	2	4
5	-14	1	

2.3.6.4. Algoritma *Square and Multiply (SAM)*

Algoritma ini digunakan untuk menyelesaikan modulo eksponensial. Algoritma *Square and Multiply* adalah sebagai berikut :

1. $x^y \bmod a$
2. Ubah *y* ke bentuk biner.
3. Ambil *z* = 1.
4. Telusuri *y* (biner) dari kiri ke kanan.
    1. Bila bertemu 0 maka update nilai *z* menjadi $z = z^2 \bmod a$
    2. Bila bertemu 1 maka update nilai *z* menjadi $z = xz^2 \bmod a$
5. Nilai *z* terakhir adalah nilai $x^y \bmod n$ yang dicari.

    Contoh: Berapa hasil dari $7^{18} \bmod 23$?

    $x = 7$

    $y = 18$, $y = 10010$

    $a = 23$

    $z = 1$

    - $1 \rightarrow z = xz^2 \bmod n = 7(1^2) \bmod 23 = 7$
    - $0 \rightarrow z = z^2 \bmod n = 7^2 \bmod 23 = 3$
    - $0 \rightarrow z = z^2 \bmod n = 3^2 \bmod 23 = 9$
    - $1 \rightarrow z = xz^2 \bmod n = 7(9^2) \bmod 23 = 15$
    - $0 \rightarrow z = z^2 \bmod n = 15^2 \bmod 23 = 18$

      Sehingga $7^{18} \bmod 23 = 18$

2.4. Kriptosistem *Paillier*

Algoritma *Paillier* adalah algoritma kriptografi yang memiliki sifat homomorfik aditif, ditemukan oleh Pascal Paillier pada tahun 1999 yang merupakan suatu algoritma asimetris probabilistic untuk kriptografi kunci publik. Pada algoritma paillier, kunci publik digunakan untuk mengenkripsi data dan kunci rahasia digunakan untuk mendekripsi data.

2.4.1. Algoritma *Paillier Scheme 1*

Pada Skema 1, kunci publik dan kunci rahasia dihasilkan oleh algoritma berikut:

Kunci Pembangkit:

1. Bangkitkan dua bilangan prima besar p dan q, secara acak. Pastikan bahwa $fpb\ ((pq),(p-1)\ (q-1))$ adalah 1.
2. Hitung $n = pq$.
3. Hitung $\lambda = kpk(p-1, q-1)$.
4. Bangkitkan sebuah bilangan bulat acak g dari himpunan $\mathbb{Z}_{n^2}^*$ (bilangan bulat antara 1 dan n^2), dan g adalah kelipatan dari n.
5. Kunci publik adalah n, g. Kunci rahasia adalah λ, μ.

Enkripsi:

1. Bangkitkan bilangan acak r dimana $0 < r < n$.
2. Hitung *ciphertext* $c = g^m r^n \bmod n^2$.

Dekripsi:

1. Hitung invers multiplikatif modular $\mu = (L(g^\lambda \bmod n^2))^{-1} \bmod n$. Dimana $L(x) = \frac{(x-1)}{n}$
2. Hitung *plaintext* $m = L(c^\lambda \bmod n^2)\mu \bmod n$

2.4.2. Algoritma *Paillier Scheme 3*

Ini adalah varian dari skema *Paillier asli*, dengan dekripsi yang lebih cepat. Alih-alih bekerja dalam seluruh grup $\mathbb{Z}_{n^2}^*$, kita bekerja dalam subgrup (g) yang dihasilkan oleh elemen (g) dengan ordo αn. Hal ini memungkinkan dekripsi dilakukan dengan eksponensiasi menggunakan eksponen α daripada λ, yang mempercepat dekripsi tergantung pada ukuran α.

Kunci Pembangkit:

1. Bangkitkan dua bilangan prima besar p dan q, secara acak. Pastikan bahwa $gcd\ ((pq),(p-1)\ (q-1))$ adalah 1.
2. Hitung $n = pq$.
3. Hitung $\lambda = lcm(p-1, q-1)$.
4. Bangkitkan α di mana $1 \leq \alpha \leq \lambda$.

5. Bangkitkan sebuah bilangan bulat acak g dari himpunan $\mathbb{Z}_{n^2}^*$ (bilangan bulat antara 1 dan n^2), dan g adalah orde dari αn.
6. Kunci publik n, g. Kunci rahasia adalah p, q, α.

Enkripsi:

1. Bangkitkan bilangan acak r di mana $r < \alpha$.
2. Hitung *ciphertext* $c = g^m (g^n)^r \bmod n^2$.

Dekripsi:

1. Hitung $\mu = L(g^\alpha \bmod n^2)^{-1}$
2. Hitung *plaintext* $m = L(c^\alpha \bmod n^2)\ \mu \bmod n$.

Contoh:

Kunci Pembangkit:

1. Bangkitkan p = 11 dan q = 13.
2. Hitung n = = 11 x 13

   n = 143
3. Hitung $\lambda = lcm(11-1, 13-1)$.

   $\lambda = lcm(10,12)$.

   λ = 60
4. Bangkitkan α = 6
5. Bangkitkan g = 2201.
6. Kunci publik: n = 143, g = 2201.

Enkripsi:

1. Bangkitkan r = 3.
2. Hitung *chipertext* $c = g^m (g^n)^r \bmod n^2$.

   Misalkan m = "A" → ASCII = 65

   $c = 2201^{65} (2201^{143})^3 \bmod 143^2$

   $c = 2201^{65} \bmod 143^2 \,.\, ((2201^{143}) \bmod 143^2)^{\,3} \bmod 143^2$

   Menggunakan algoritma *Square and Multply*, hitung modulo eksponensial $2201^{65} \bmod 143^2$,

   x = 2201

   y = 65, y = 1000001

   a = 20449

   z = 1

   - $1 \rightarrow z = xz^2 \bmod n = 2201(1^2) \bmod 20449 = 2201$

- $0 \rightarrow z = z^2 \bmod n = 2201^2 \bmod 20449 = 18437$
- $0 \rightarrow z = z^2 \bmod n = 18437^2 \bmod 20449 = 19691$
- $0 \rightarrow z = z^2 \bmod n = 19691^2 \bmod 20449 = 1992$
- $0 \rightarrow z = z^2 \bmod n = 1992^2 \bmod 20449 = 958$
- $0 \rightarrow z = z^2 \bmod n = 958^2 \bmod 20449 = 18008$
- $1 \rightarrow z = xz^2 \bmod n = 2201(18008^2) \bmod 20449 = \mathbf{18613}$

Menggunakan algoritma *Square and Multply*, hitung modulo eksponensial $2201^{143} \bmod 143^2$,

x = 2201

y = 143, *y* = 10001111

a = 20449

z = 1

- $1 \rightarrow z = xz^2 \bmod n = 2201(1^2) \bmod 20449 = 2201$
- $0 \rightarrow z = z^2 \bmod n = 2201^2 \bmod 20449 = 18437$
- $0 \rightarrow z = z^2 \bmod n = 18437^2 \bmod 20449 = 19691$
- $0 \rightarrow z = z^2 \bmod n = 19691^2 \bmod 20449 = 1992$
- $1 \rightarrow z = xz^2 \bmod n = 2201(1992^2) \bmod 20449 = 2311$
- $1 \rightarrow z = xz^2 \bmod n = 2201(2311^2) \bmod 20449 = 3312$
- $1 \rightarrow z = xz^2 \bmod n = 2201(3312^2) \bmod 20449 = 5314$
- $1 \rightarrow z = xz^2 \bmod n = 2201(5314^2) \bmod 20449 = \mathbf{9318}$

$c = 18613 . (9318)^3 \bmod 143^2$

$c = 18613 . 19603 \bmod 143^2$

$c = \mathbf{19581}$

Dekripsi:

1. Hitung $\mu = L(2201^6 \bmod 143^2)^{-1}$. Dimana $L(x) = \frac{(x-1)}{n}$

   Hitung modulus eksponensialnya

   x = 2201

   y = 6, *y* = 110

   a = 20449

   z = 1

- $1 \rightarrow z = xz^2 \bmod n = 2201(1^2) \bmod 20449 = 2201$
- $1 \rightarrow z = xz^2 \bmod n = 2201(2201^2) \bmod 20449 = 9021$

- $0 \rightarrow z = z^2 \bmod n = 9021^2 \bmod 20449 = 11870$

$L(x) = \dfrac{(11870 - 1)}{143} = 83^{-1}$

2. Hitung *plaintext* $m = L(19581^6 \bmod 143^2)$.

Hitung modulus eksponensialnya

x = 19581

y = 6, y = 110

a = 20449

z = 1

- $1 \rightarrow z = xz^2 \bmod n = 19581(1^2) \bmod 20449 = 19581$
- $1 \rightarrow z = xz^2 \bmod n = 19581(19581^2) \bmod 20449 = 7437$
- $0 \rightarrow z = z^2 \bmod n = 7437^2 \bmod 20449 = 14873$

$L(x) = \dfrac{(14873 - 1)}{143} = 104$

$m = 104 \,.\, 83^{-1} \, mod \, 143$

Hitung PDL dengan IB $83^{-1} \, mod \, 143$

$83x + 143y = 1$

x	y	r	k
0	1	143	
1	0	83	
-1	1	60	1
2	-1	23	1
-5	3	14	2
7	-4	9	1
-12	7	5	1
19	-11	4	1
-31	18	1	

Maka, $83^{-1} \, mod \, 143 = -31 + 143 = 112$

$m = 104 \,.112 \, mod \, 143$

m= 65 → “A”

2.5. Algoritma *Kraitchik*

Algoritma *Kraitchik* adalah algoritma faktorisasi yang didasarkan pada konsep kongruansi, diusulkan pertama kali oleh Maurice Kraitchik pada tahun 1920-an. Algoritma ini merupakan perbaikan dari algoritma *Fermat's difference of square.* Algoritma *Kraitchik* adalah salah satu metode kriptanalisis yang menggunakan konsep bilangan kuadrat sempurna dan *Greatest common divisor Euclidean.* Berikut adalah Algoritma faktorisasi Kraitchik.

Berikut adalah Algoritma faktorisasi Kraitchik

1. Tentukan n yang akan difakorkan
2. Hitung $x = |\sqrt{n}|$
3. Jika faktor *n* tidak ditemukan maka:
   a. Tentukan $k = 1$
   b. Jika $x^2 - kn \geq 0$ maka:
      i. $y = \sqrt{x^2 - kn}$
      ii. Jika *y* adalah kuadrat sempurna dan $(x + y) \bmod n \neq 0$ dan $(x - y) \bmod n \neq 0$ kemudian faktor dari *n* adalah $p = \gcd(x + y, n)$ dan $q = \gcd(x - y, n)$, maka berhenti.
      iii. Jika tidak $k = k + 2$
   c. $x = x + 1$

Contoh, faktorkan 143 menggunakan metode *Kraitchik*

1. $n = 143$
2. Hitung $x = \left|\sqrt{143}\right|$
3. Akar kuadrat dari 22 sekitar 11.96 jadi, $x = 11$
   a. $k = 1$
   b. Hitung $x^2 - kn$

      $11^2 - 1(143)$

      $= 121 - 143$

      $= -22 \leq 0$, karena tidak memenuhi maka:
   c. $x = x + 1$

      $x = 11 + 1 = 12$
4. Karena faktor *n* belum ditemukan maka
   a. $k = 1$

b. Hitung $x^2 - kn$

$12^2 - 1(143)$

$= 144 - 143$

$= 1 \leq 0$, karena memenuhi maka:

i. $y = x^2 - kn$

$y = \sqrt{1} = 1$

ii. Karena *y* adalah kuadrat sempurna, periksa (*x* + *y*) mod *n* ≠ 0 dan (*x* – *y*) mod *n* ≠ 0

(12 + 1) mod 143 ≠ 0, p = 13

(12 – 1) mod 143 ≠ 0, q = 11

Jadi faktor dari *n* = 143 adalah 13 dan 11

2.6. Penelitian Relevan

Di bawah ini merupakan sejumlah riset yang berhubungan terkait Algoritma *Paillier Scheme 3* dengan Algoritma *Kraitchik* sebagai berikut:

1. Dalam penelitian (Jost et al., 2015), "*Encryption Performance Improvements of the Paillier Cryptosystem*" membahas optimasi enkripsi homomorfik yang mampu meningkatkan throughput enkripsi secara signifikan, mencapai 89,483 enkripsi per detik untuk kunci 2048-bit dengan menggunakan *Scheme 3* dari Paillier Cryptosystem, di mana skema tersebut menawarkan dekripsi lebih besar.
2. Penelitian (Purba, 2019) yang berjudul "*Kriptanalisis kunci publik algoritma Rabin menggunakan metode Kraitchik*" menyimpulkan bahwa metode *Kraitchik* berhasil memecahkan kunci publik algoritma *Rabin* sehingga ditemukan hasil kunci rahasianya dan juga panjang kunci publik tidak selalu menentukan lamanya waktu yang dibutuhkan untuk pemecahan, sehingga nilai kunci publik yang lebih besar tidak selalu menghasilkan waktu proses yang lebih lama atau lebih cepat.
3. Berdasarkan penelitian (Lydia et al., 2021) yang berjudul "*Factorization of Small RPrime RSA Modulus Using Fermat's Difference of Squares and Kraitchik's Algorithms in Python*" Algoritma *Kraitchik* mampu digunakan untuk memfaktorkan modulus dari RPrime RSA, yang merupakan produk dari dua atau lebih bilangan prima. Algoritma Kraitchik memiliki keterbatasan dalam hal skalabilitas. Ketika menangani modulus yang lebih besar, waktu pemrosesan

meningkat secara eksponensial.

4. Penelitian (Ma et al., 2021) "*Optimized Paillier's Cryptosystem with Fast Encryption and Decryption*" memperkenalkan optimasi pada *Paillier Cryptosystem,* khususnya pada *Scheme 3*. Dalam penelitian tersebut, digunakan subkelompok yang dipilih secara khusus pada ruang acak yang memungkinkan percepatan proses enkripsi hingga 7.5 kali lebih cepat dibandingkan dengan Paillier's Scheme dasar. Optimasi ini menawarkan peningkatan efisiensi tanpa mengorbankan keamanan kriptografis, menjadikannya ideal untuk aplikasi yang memerlukan enkripsi dan dekripsi cepat dalam skenario yang besar
5. Penelitian (Mulya et al., 2021) berjudul "*Analisis Dan Perancangan Simulasi Algoritma Paillier Cryptosystem Pada Pesan Text Dengan Presentation Format Binary, Octal, Hexadecimal dan Base64*" mengkaji bagaimana algoritma Paillier dapat digunakan untuk mengenkripsi pesan teks secara efisien dengan simulasi yang dibangun menggunakan perangkat lunak Cryptool2. Hasil penelitian menunjukkan bahwa format *binary* merupakan yang tercepat dalam proses enkripsi dan dekripsi, sedangkan base64 adalah yang paling lambat.

BAB III

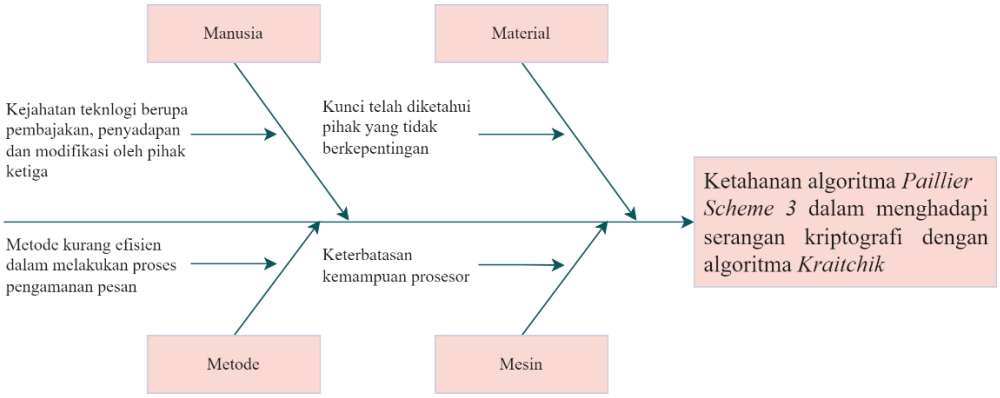
ANALISIS DAN PERANCANGAN SISTEM

3.1. Analisis Sistem

Tahapan analisis sistem merupakan salah satu tahapan yang bertujuan untuk mengidentifikasikan kebutuhan yang dapat dimanfaatkan agar sistem dapat bekerja secara optimal sesuai dengan yang diharapkan. Analisis sistem terdiri dari tiga fase antara lain analisis masalah, analisis kebutuhan dan analisis proses. Analisis masalah dilakukan untuk identifikasi masalah. Analisis kebutuhan dilakukan untuk menggambarkan fungsi yang disediakan dan yang dapat dilakukan oleh sistem. Tujuan dari analisis proses adalah untuk memodelkan perilaku sistem.

3.1.1. Analisis Masalah

Analisis masalah merupakan tahapan yang dilakukan untuk mengidentifikasikan masalah yang dapat muncul pada sistem yang akan dibangun. Tujuannya adalah mengecek sistem dapat berjalan sesuai kebutuhan pengguna. Dalam konteks ini, permasalahan yang dihadapi adalah bagaimana ketahanan algoritma *Paillier Scheme 3* dalam menghadapi serangan kriptografi dengan algoritma *Kraitchik* dan algoritma *Brute-force*. Untuk menyelesaikan permasalahan sistem yang akan dibangun, digunakan *fishbone diagram* (Diagram Ishikawa). Diagram Ishikawa merupakan alat visual yang menunjukkan hubungan antara suatu masalah dengan berbagai kemungkinan penyebabnya. Gambar 3.1. menampilkan Diagram Ishikawa sistem.



Gambar 3.1 Diagram Ishikawa

Gambar 3.1 menampilkan 4 kelompok pemicu masalah pada kriptanalisis algoritma *Paillier Scheme 3* menggunakan algoritma *Kraitchik*. Kategori-kategori tersebut terdiri dari material, manusia, metode, dan mesin, yang masing-masing direpresentasikan oleh sebuah panah yang menunjuk ke pusat masalah atau tulang punggung. Setiap kategori memiliki tanda panah unik yang menghubungkannya dengan kategori yang sesuai.

3.1.2. Analisis Kebutuhan

Pada tahapan analisis kebutuhan dilakukan identifikasi data-data yang diperlukan pada suatu sistem dan alur yang akan dirancang pada suatu sistem sehingga dapat mencapai tujuan yang diharapkan. Pada penelitian ini diuraikan analisis kebutuhan menjadi dua kelompok yaitu:

1. Kebutuhan Fungsional, merupakan kebutuhan yang mencakup proses yang dilaksanakan pada sistem. Adapun kebutuhan fungsional yang diperlukan dalam perancangan sistem ini adalah:
    a. Sistem mampu melakukan pembangkitan kunci rahasia dan kunci publik.
    b. Sistem mampu melakukan enkripsi pesan menjadi *ciphertext* dengan menerapkan algoritma *Paillier Scheme 3*.
    d. Sistem mampu melakukan pengujian pencarian kunci rahasia dan memperoleh *plaintext* dengan metode *Kraitchik* dan *Brute-force*.
    e. Sistem mampu menjalankan proses dekripsi dengan algoritma *Paillier Schem 3* menggunakan kunci publik dan kunci rahasia yang diperoleh

dengan *Kraitchik* dan *Brute-force*

3. Kebutuhan non-fungsional, merupakan kebutuhan yang mencakup batasan pengembangan sistem serta batasan kemampuan sistem. Adapun kebutuhan non-fungsional yang diperlukan dalam pengembangan sistem ini adalah sebagai berikut:
    a. Menampilkan antarmuka yang berinteraksi dengan baik dengan pengguna, dan sistem yang dibuat juga dapat menampilkan dan menyimpan hasil dari fungsi kriptografi yang dilakukan.
    b. Menampilkan pesan kesalahan ketika pengguna melakukan kesalahan seperti *input* yang tidak lengkap atau persyaratan *input* yang tidak dapat diterima.
    c. Jenis serangan pengujian pada proses kriptanalisis terbatas pada serangan *kraitchik* dan *brute-force.*

3.2. Perancangan Sistem

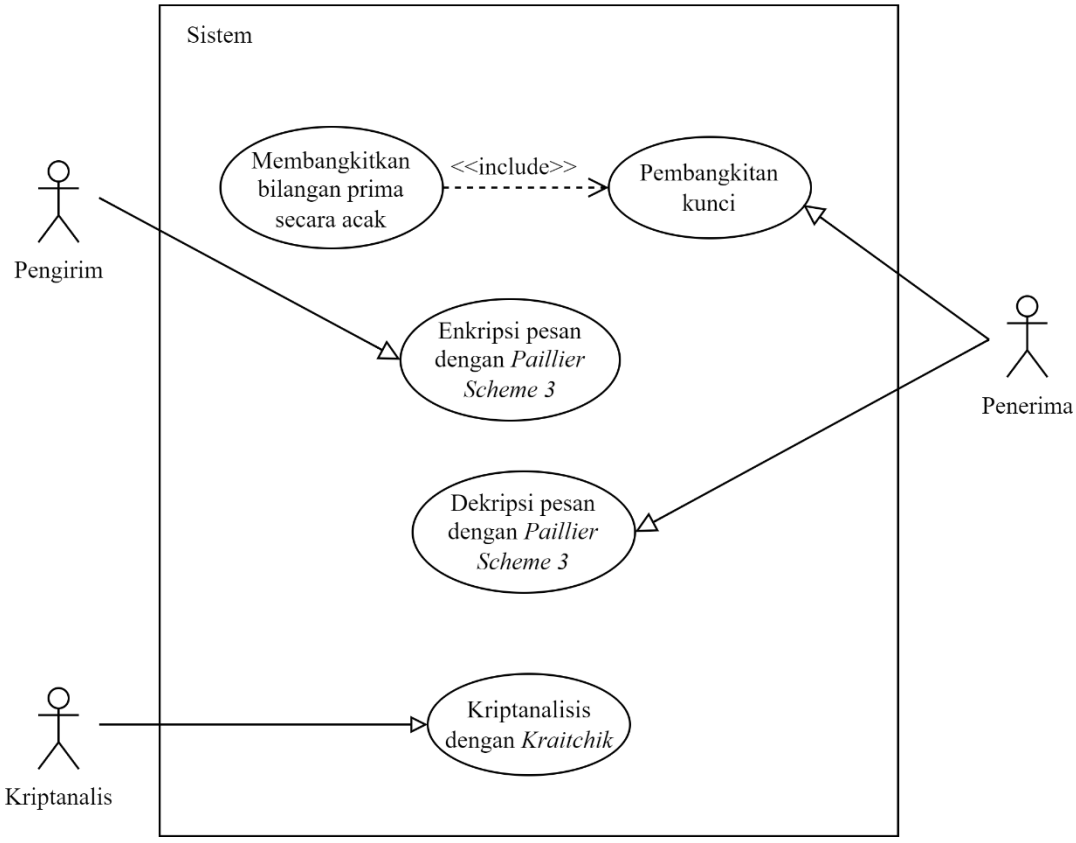
Perancangan sistem merupakan salah satu langkah yang bertujuan untuk merancang sistem dengan tujuan membuat sistem tersebut beroperasi dengan fokus meningkatkan tingkat efektivitas dan efisiensi. Perancangan sistem dalam penelitian ini akan memanfaatkan berbagai jenis diagram, termasuk diagram umum, diagram *use case*, diagram aktivitas, diagram urutan, dan diagram alir (*flowchart)*

3.2.1. Use Case Diagram

Use case diagram merupakan jenis graf pada pemodelan perangkat lunak yang digunakan untuk memvisualisasikan hubungan antara sistem perangkat lunak dan berbagai aktor (pengguna atau entitas eksternal lainnya) yang terlibat dalam sistem. *Use case* diagram membantu mengidentifikasi, mendeskripsikan, serta memahami bagaimana sistem yang akan dibangun beroperasi dalam konteks situasi dunia nyata.

Pada Gambar 3.2, terdapat aktor yang berinteraksi dengan sistem, yaitu pengguna, pengirim, penerima, dan kriptanalis. Penerima memiliki peran untuk menghasilkan pasangan kunci dengan menggunakan bilangan prima yang dihasilkan secara acak. Setelah kunci publik dan kunci rahasia tersedia, pengirim dapat melakukan proses enkripsi terhadap pesan menggunakan kunci publik penerima,

menghasilkan *ciphertext*. Selanjutnya, penerima dapat mendekripsi *ciphertext* tersebut menggunakan kunci rahasia yang dimilikinya. Sementara itu, kriptanalis dapat memanfaatkan kunci publik untuk melakukan analisis kriptografi terhadap ciphertext untuk mengungkap informasi yang terkandung di dalamnya.

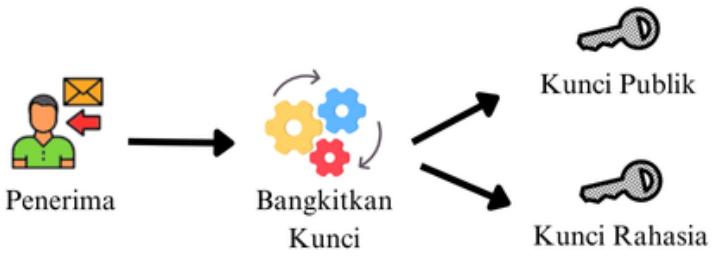


Gambar 3.2 *Use Case* Diagram

3.2.2. Diagram Umum Sistem

Diagram umum sistem merupakan ilustrasi alur, proses, dan hubungan antar komponen dalam sistem. Diagram umum sistem dalam penelitian ini terbagi menjadi 3, yaitu diagram umum pembangkitan kunci, enkripsi dan dekripsi, serta kriptanalisis.

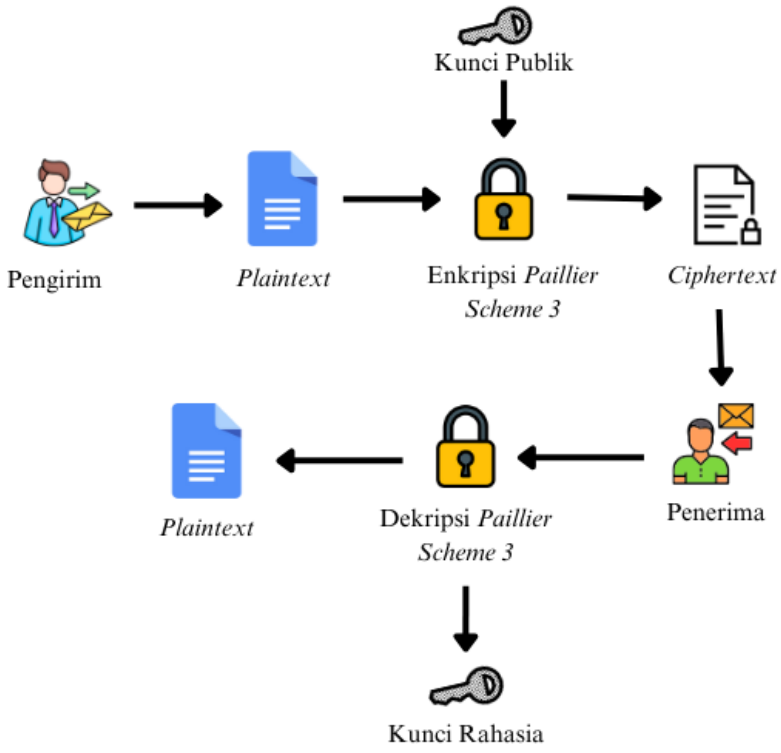
3.2.2.1. Diagram Umum Pembangkitan Kunci



Gambar 3.3 Diagram Umum Pembangkitan Kunci

Pada Gambar 3.3 menunjukkan bagaimana proses pembangkitan kunci yang digunakan. Penerima akan membangkitkan kunci publik (n, g) dan kunci rahasia (μ,α) yang nantinya akan digunakan pada proses enkripsi dan dekripsi.

3.2.2.2. Diagram Umum Enkripsi dan Dekripsi

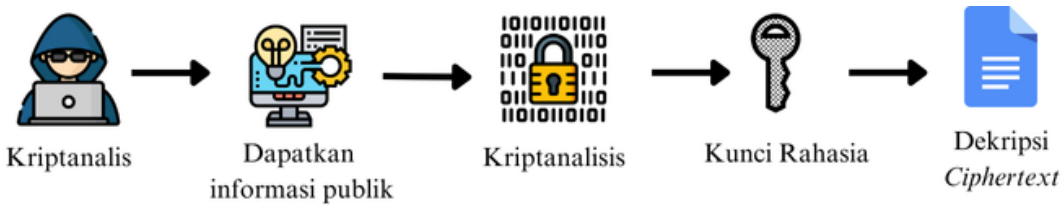


Gambar 3.4. Diagram Umum Enkripsi dan Dekripsi

Pada Gambar 3.4, menunjukkan bagaimana proses enkripsi dan dekripsi dilakukan pada sistem. Pengirim memasukkan pesan yang akan di enkripsi menggunakan algoritma *Paillier Scheme 3*. Lalu pengirim memasukkan kunci

publik yang telah dibangkitkan sebelumnya. Pesan kemudian akan dienkripsi sehingga menghasilkan *ciphertext.* Setelah penerima menerima *ciphertext* yang dikirim oleh pengirim, penerima akan melakukan dekripsi pesan menggunakan kunci rahasia untuk mendapatkan pesan asli

3.2.2.3. Diagram Umum Kriptanalisis



Gambar 3.5 Diagram Umum Kriptanalisis

Pada Gambar 3.5 menunjukkan bagaimana proses kriptanalisis dilakukan pada sistem.Kriptanalisis akan mengumpulkan informasi publik lalu memasukkan informasi tersebut ke dalam sistem. Selanjutnya dilakukan proses kriptanalisis ntuk melakukan proses dekripsi pada *ciphertext* tanpa memiliki pengetahuan tentang kunci rahasia.

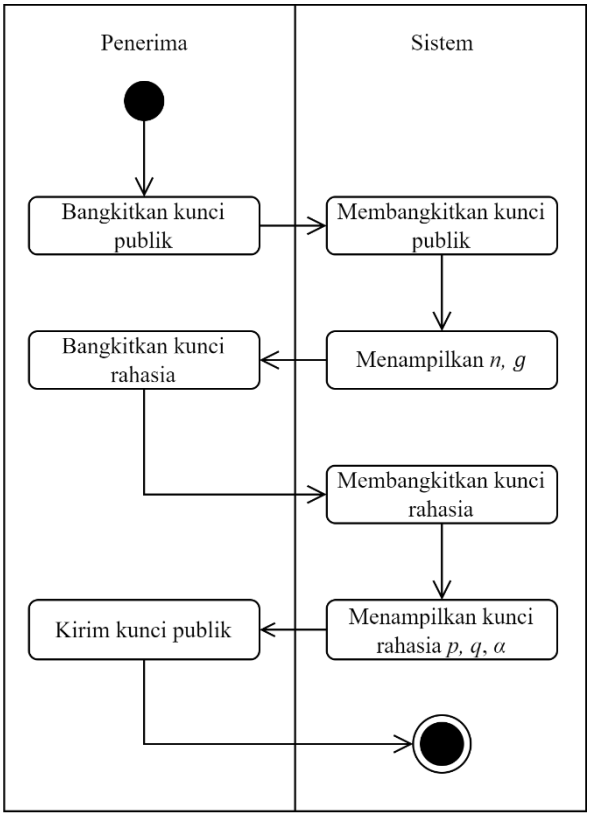
3.2.3. *Activity Diagram*

Diagram aktivitas adalah diagram yang menggambarkan berbagai aliran aktivitas dari sistem yang direncanakan, bagaimana setiap aliran dimulai, keputusan apa yang dapat dibuat, dan bagaimana akhirnya. Suatu aktivitas dapat diimplementasikan dengan satu atau lebih *use case*, atau dengan kata lain, diagram aktivitas adalah diagram yang menjelaskan tindakan pengguna terhadap sistem dalam situasi yang dijelaskan dalam diagram *use case* dari langkah sebelumnya.

3.2.3.1. *Activity Diagram* Pembangkitan Kunci

Pada Gambar 3.6 menggambarkan graf aktivitas pada fase pembangkitan kunci pada sistem yang digunakan dalam penelitian ini. Penerima akan melakukan proses pembangkitan kunci publik sehingga sistem akan menanggapi untuk melakukan proses tersebut dan mengembalikan nilai n dan g. Selanjutnya penerima akan membangkitkan kunci rahasia. Sistem akan melakukan proses pembangkitan kunci rahasia dan mengembalikan

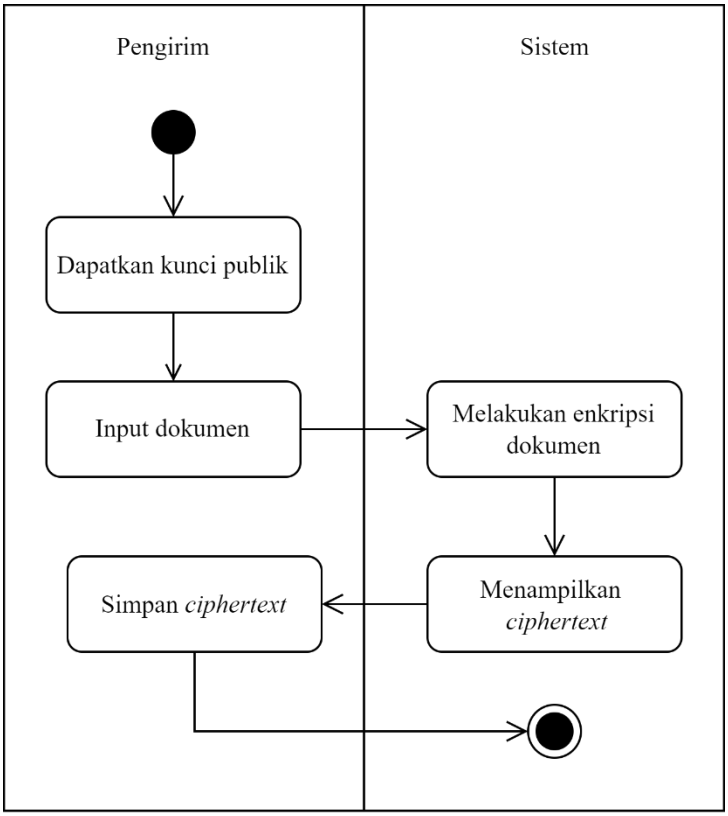
keluaran kunci rahasia $p, q,$ dan α yang akan dipakai di fase dekripsi. Penerima akan mengirim kunci publik yang nantinya akan di gunakan pengirim pada fase enkripsi.



Gambar 3.6 *Activity Diagram* Pembangkitan Kunci *Paillier Scheme 3*

3.2.3.2. *Activity Diagram* Enkripsi

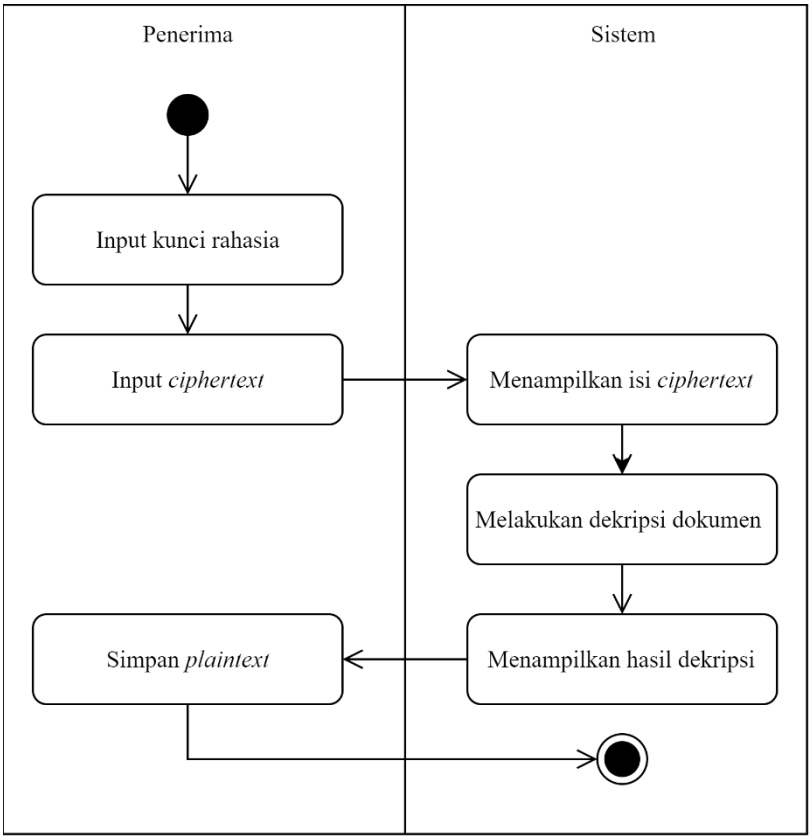
Pada Gambar 3.7 menggambarkan graf aktivitas pada fase enkripsi pada sistem yang digunakan dalam penelitian ini. Pengirim mendapatkan kunci publik yang telah dibangkitkan penerima sebelumnya. Pengirim akan memasukkan dokumen yang akan dienkripsi menggunakan algoritma Pillier Scheme 3. Sistem akan menanggapi dengan melakukan enkripsi terhadap dokumen yang dimasukkan oleh pengirim. Setelah proses enkripsi berhasil, sistem akan menampilkan ciphertext. Pengguna kemudian dapat menyimpan ciphertext.



Gambar 3.7 *Activity Diagram* Enkripsi

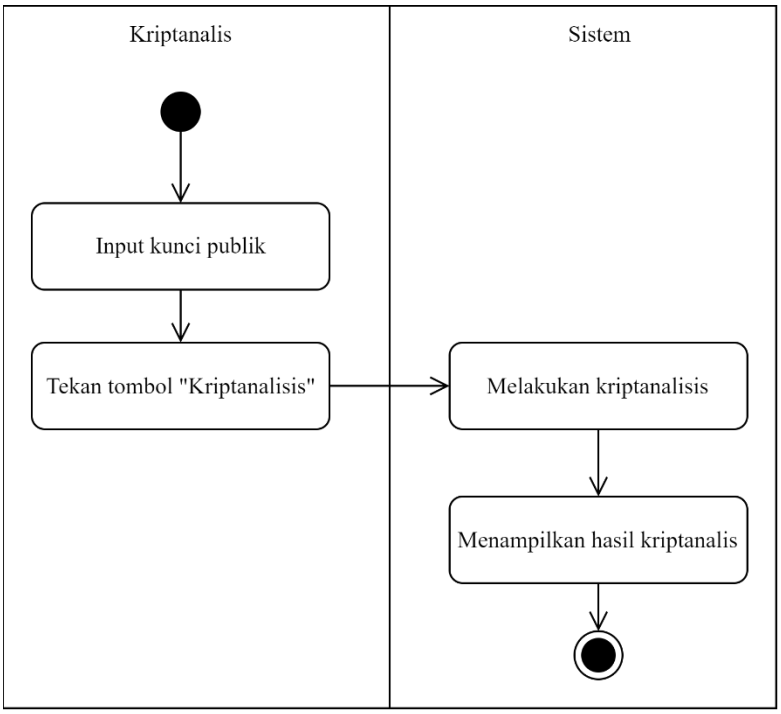
3.2.3.3. *Activity Diagram* Dekripsi

Pada Gambar 3.8 menggambarkan graf aktivitas pada fase dekripsi pada sistem yang digunakan dalam penelitian ini. Penerima memasukkan kunci publik yang didapatkan dari proses pembangkitan kunci. Kemudian memasukkan *ciphertext* yang didapat dari proses enkripsi sebelumnya. Setelah itu sistem menanggapi dengan menampilkan isi dari *ciphertext*. Sistem kemudian akan melakukan proses dekripsi menggunakan informasi publik yang dimasukkan oleh penerima. Sistem akan menampilkan hasil dekripsi berupa *plaintext* . Pengguna kemudian dapat menyimpan *plaintext* .



Gambar 3.8 *Activity Diagram* Dekripsi

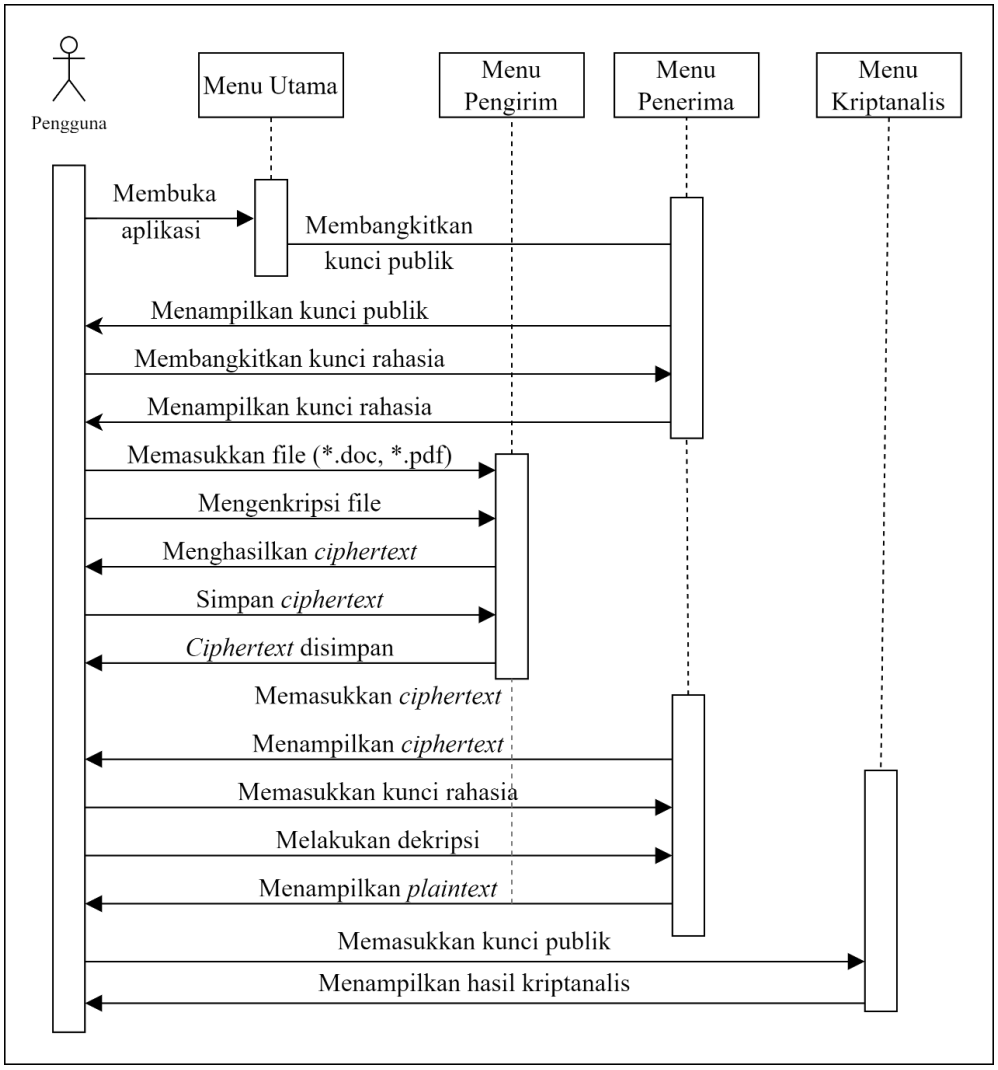
3.2.3.4. *Activity Diagram* Kriptanalisis



Gambar 3.9 *Activity Diagram* Kriptanalisis

Pada Gambar 3.9 menggambarkan graf aktivitas pada fase kriptanalisis pada sistem yang digunakan dalam penelitian ini. Kriptanalis memasukkan kunci publik. Sistem kemudian akan menampilkan hasil kriptanalis berupa *plaintext* . Pengguna kemudian dapat menyimpan *plaintext* .

3.2.3.5. Sequence Diagram



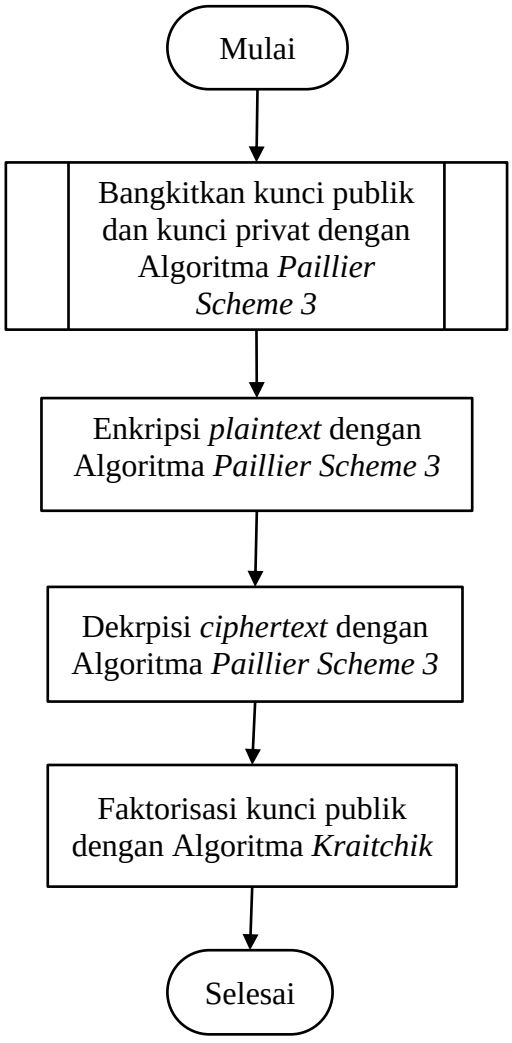
Gambar 3.10 *Sequence Diagram*

Pada Gambar 3.10 memperlihatkan interaksi antara pengguna terhadap pengguna secara *sequence.* Pada saat melakukan proses dekripsi pengguna akan berinteraksi dengan menu utama, menu pengirim, dan menu penerima.

3.3. Flowchart

Flowchart merupakan representasi dari urutan proses yang digambarkan dalam bentuk berbagai simbol untuk menunjukkan hubungan antara suatu proses degan proses lainnya pada suatu sistem. Alur dan proses sistem yang dirancang pada penelitian ini dapat diuraikan menjadi beberapa jenis *flowchart* yakni sebagai berikut:

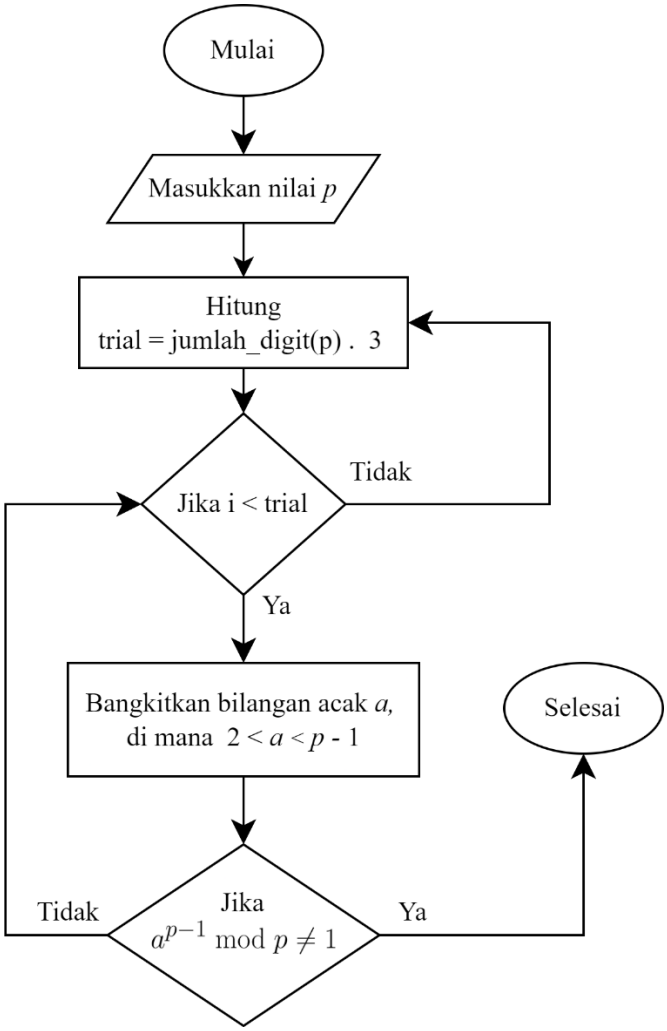
3.3.1. *Flowchart* Sistem



Gambar 3.11 *Flowchart* Sistem

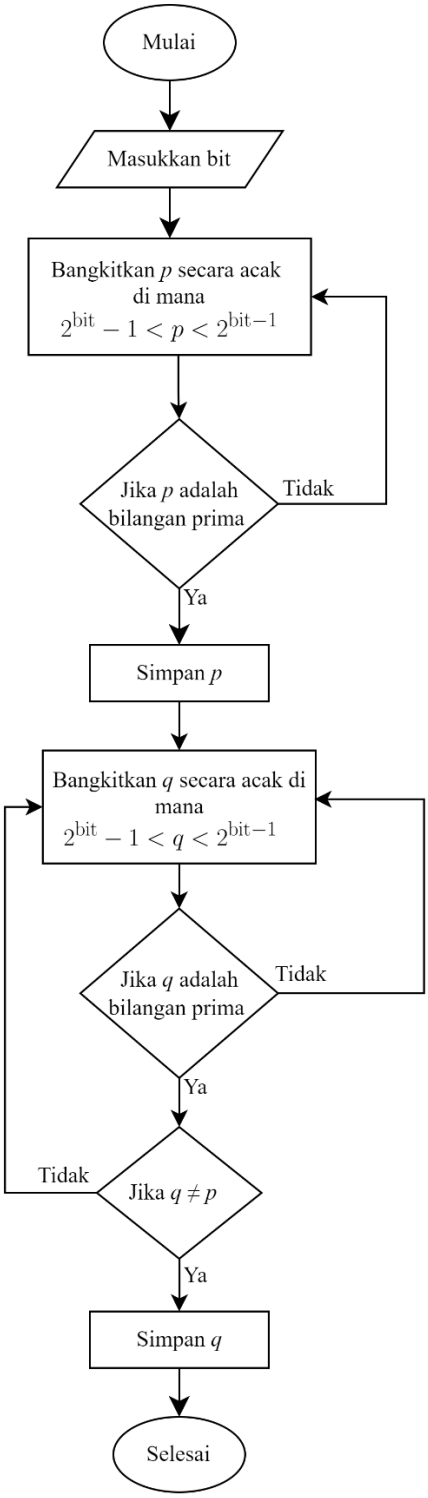
Pada Gambar 3.11 memperlihatkan proses sistem yang digunakan dalam penelitian ini.

3.3.2. *Flowchart* Pembangkitan Bilangan Prima *Fermat's Little Theorem*



Gambar 3.12 *Flowchart* Pembangkit Kunci

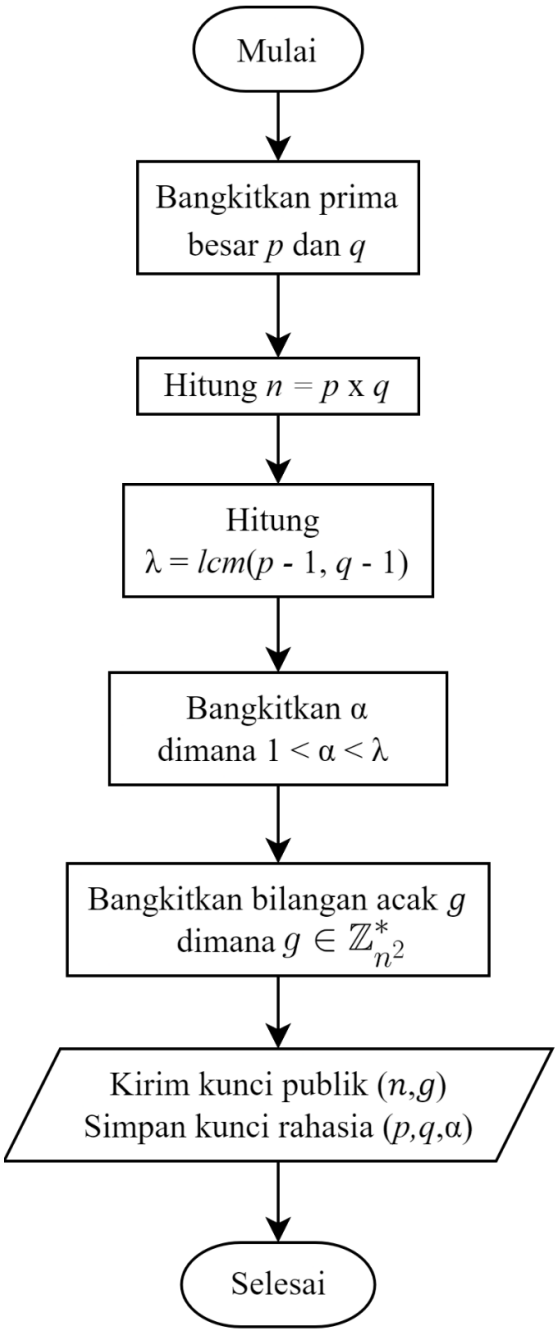
Pada gambar 3.12 memperlihatkan pembangkitan bilangan acak prima menggunakan *fermat's little theorem*. Percobaan akan dilakukan sebanyak 3 kali panjang bilangan *p*. Sistem akan mencari atau menghitung bilangan a secara acak dalam rentang 2 sampai p-1. Jika $a^{p-1}\ mod\ p \neq 1$, benar maka perulangan akan dilanjutkan, jika tidak maka program selesai.



Gambar 3.13 *Flowchart* Pembangkit Random Prima

Pada gambar 3.13 memperlihatkan *flowchart* dalam pembangkitan bilangan acak. Panjang bit yang digunakan pada sistem akan dimasukkan. Sistem akan mencari bilangan acak antara $2^{(bit\_length\ -1)}$ sampai $2^{bit\_length\ -1}$, jika benar maka nilai akan dicek keprimaannya. Jika benar maka nilai akan disimpan, dengan tambahan $p \neq q$.

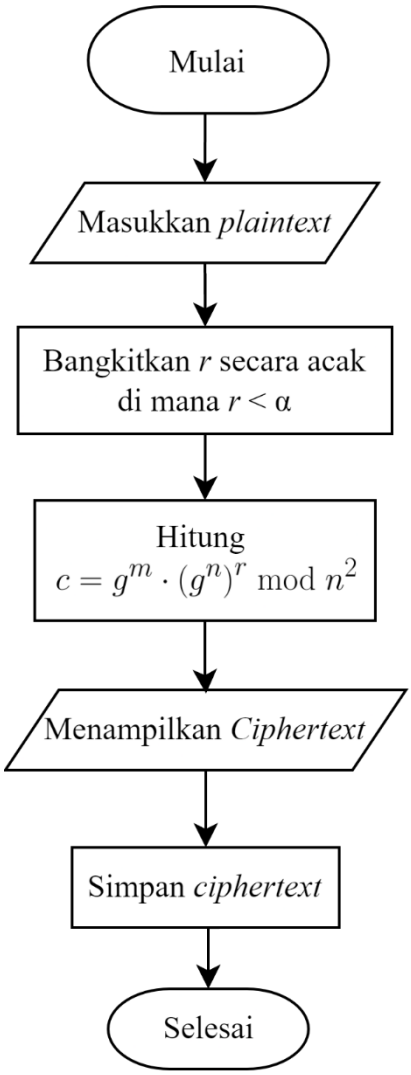
3.3.3. *Flowchart* Pembangkitan Kunci pada Algoritma *Paillier Scheme 3*



Gambar 3.14 *Flowchart* Pembangkitan Kunci pada Algoritma *Paillier Scheme 3*

Pada gambar 3.14 menunjukkan proses pembangkitan kunci publik dan kunci rahasia menggunakan algoritma *Paillier Scheme 3,* yang digunakan untuk mengenkripsi dan mendekripsi pesan (*plaintext).*

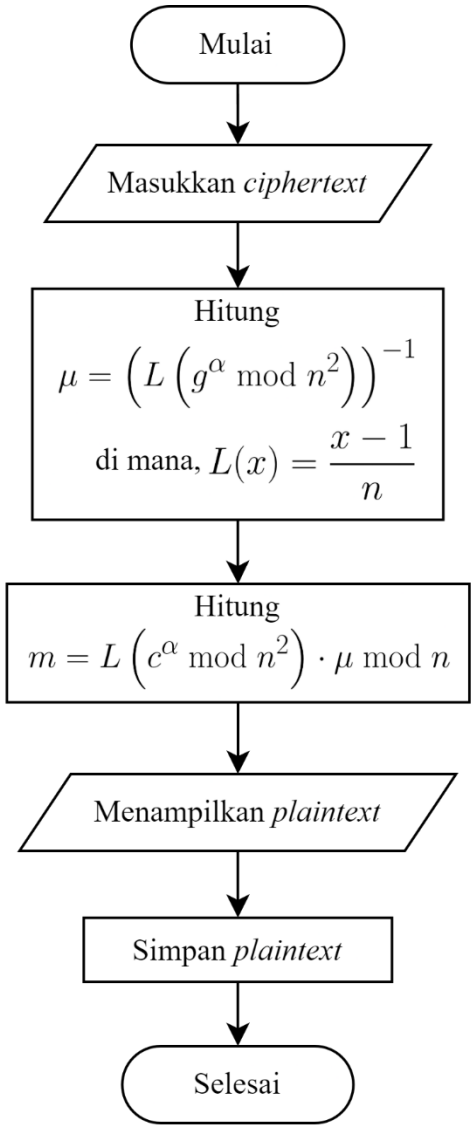
3.3.4. *Flowchart* Algoritma Enkripsi *Paillier Scheme 3*



Gambar 3.15 *Flowchart* Algoritma Enkripsi

Pada gambar 3.15 memperlihatkan *flowchart* algoritma enkripsi pada *Paillier Scheme 3.*

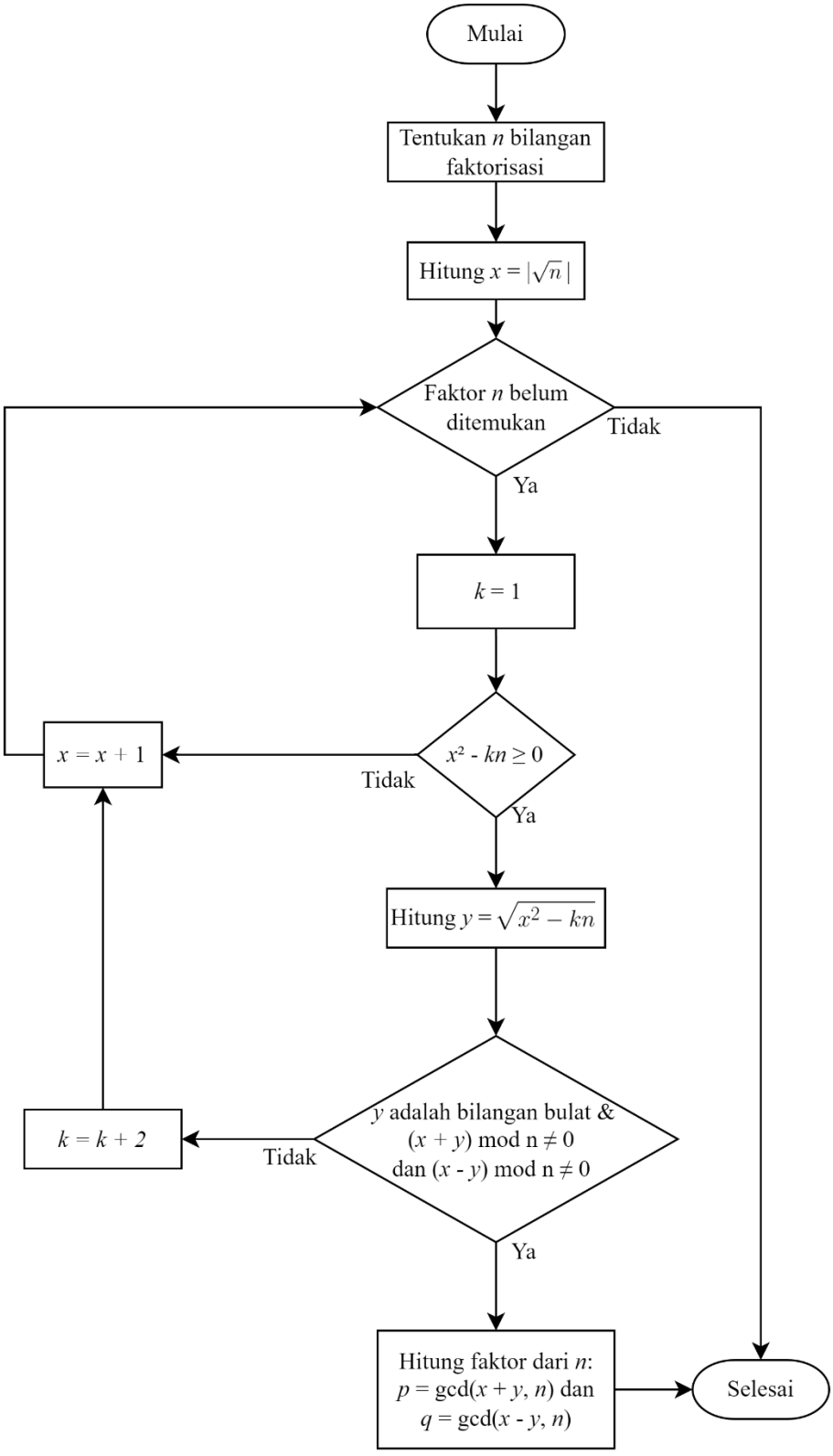
3.3.5. *Flowchart* Algoritma Dekripsi *Paillier Scheme 3*



Gambar 3.16 *Flowchart* Algoritma Dekripsi

Pada gambar 3.16 memperlihatkan *flowchart* algoritma dekripsi pada *Paillier Scheme 3.*

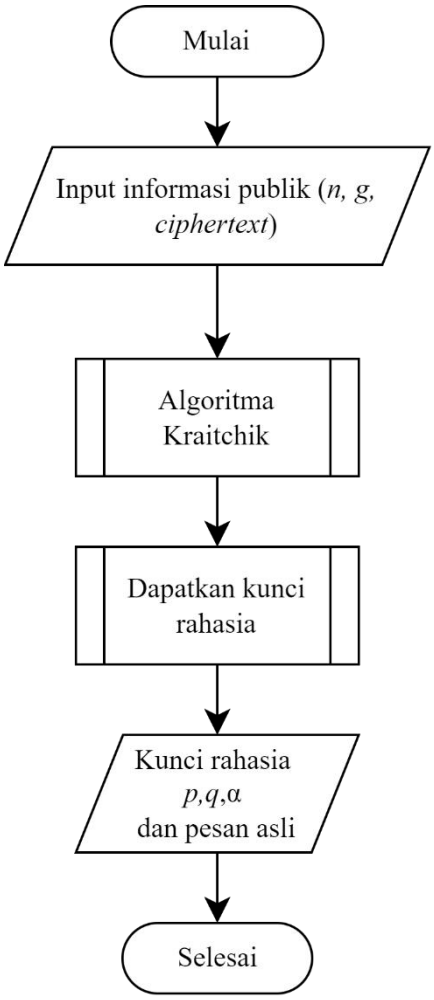
3.3.6. *Flowchart* Algoritma *Kraitchik*



Gambar 3.17 *Flowchart* Algoritma *Kraitchik*

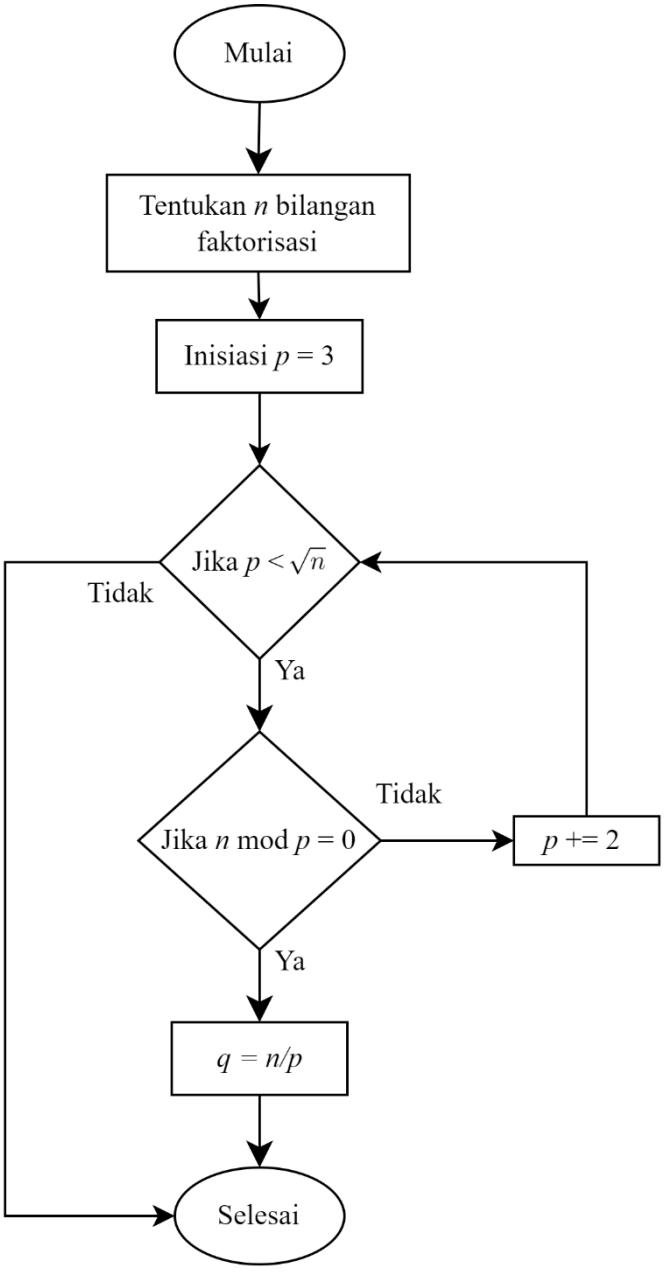
Pada gambar 3.17 memperlihatkan alur proses algoritma Kraitchik.

3.3.7. *Flowchart* Kriptanalisis



Gambar 3.18 *Flowchart* Kritptanalisis

Pada gambar 3.18. memperlihatkan proses kriptanalisis. *Flowchart* menjelaskan bagaimana nantinya proses kriptanalisis berjalan pada sistem yang dikembangkan. Masukan yang dibutuhkan pada proses ini ada 3, yaitu kunci publik *n* dan kunci publik *g* serta *ciphertext*. Dilanjutkan dengan algoritma *Kraitchik.*



Gambar 3.19 *Flowchart* Algoritma *Brute-Force*

Pada gambar 3.19 memperlihatkan bagaimana proses pada algoritma *Brute-force.* Proses faktorisasi dengan *Brute-force* dimulai dengan menerima masukan kunci publik *n*. Selanjutnya dilakukan iterasi dalam rentang 3 hingga p < $\sqrt{n}$, dengan langkah 2. Pada setiap iterasi, algoritma akan memeriksa apakah *p* adalah faktor *n* dengan menggunakan operasi modulus *n* mod *p* = 0, Jika hasilnya nol maka *p* adalah faktor dari *n.* Jika ditemukan *p* adalah faktor dari *n* maka *q* dihitung dengan pembagian *n /p.* Setelah pasangan faktor *p* dan *q* ditemukan maka algoritma berhenti.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1. Implementasi Sistem

Dalam penelitian ini, sistem dikembangkan menggunakan bahasa pemrograman *Python* dengan menggunakan *framework Django* sebagai *back-end* dan bahasa pemrograman *Javascript* dengan menggunakan *framework React.Js* sebagai *front-end* dengan *Visual Studio Code* sebagai IDE. Pada sistem yang dikembangkan memiliki 6 laman utama, yakni laman *Home*, laman *Dashboard*, laman Enkripsi, laman Dekripsi, Laman *Kraitchik*, dan Laman *Brute Force*.

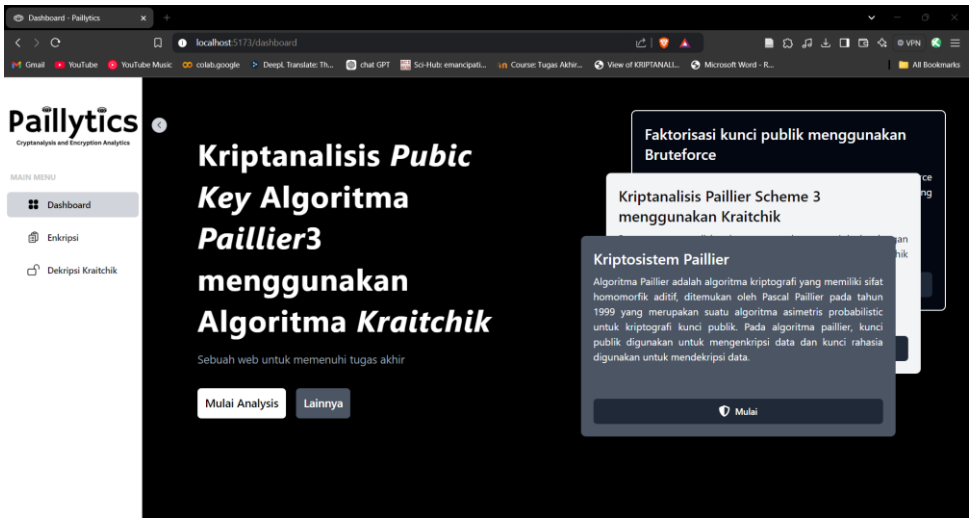
4.1.1. Laman *Home*

Laman *Home* adalah laman yang ditampilkan pertama kali ketika website dibuka.

Gambar 4.1 Laman *Home*

4.1.2. Laman *Dashoard*

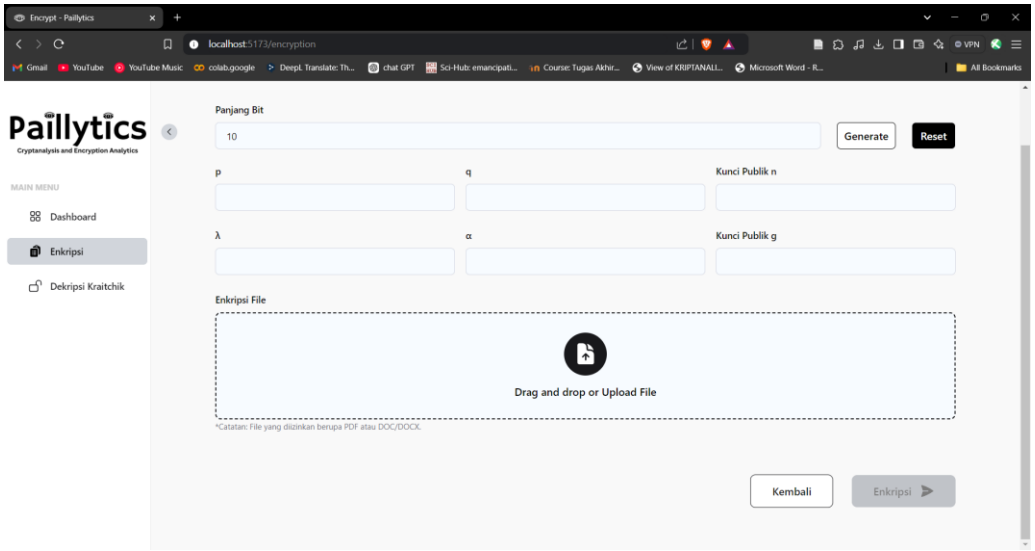
Laman *Dashoard* adalah laman yang berisi fitur-fitur yang ada pada website.



Gambar 4.2 Laman *Dashboard*

4.1.3. Laman Enkripsi

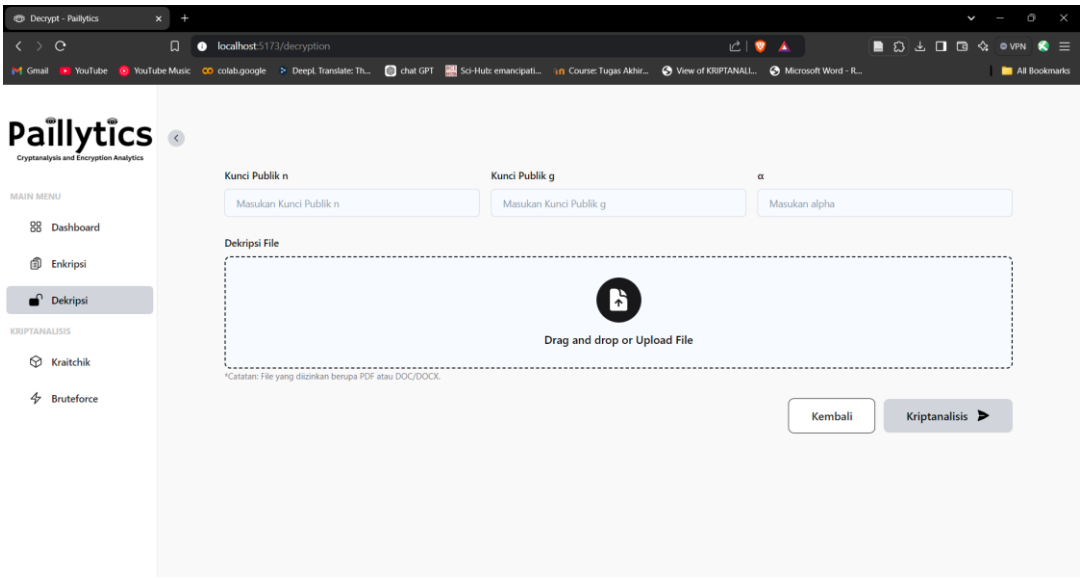
Pada gambar 4.3 menunjukkan laman Enkripsi. Pada halaman ini menampilkan pembangkitan kunci dan enkripsi yang bisa diakses oleh pengirim. Pada laman ini akan memberikan keluaran kunci publik, kunci rahasia, dan *ciphertext.*



Gambar 4.3 Laman Enkripsi

4.1.4. Laman Dekripsi

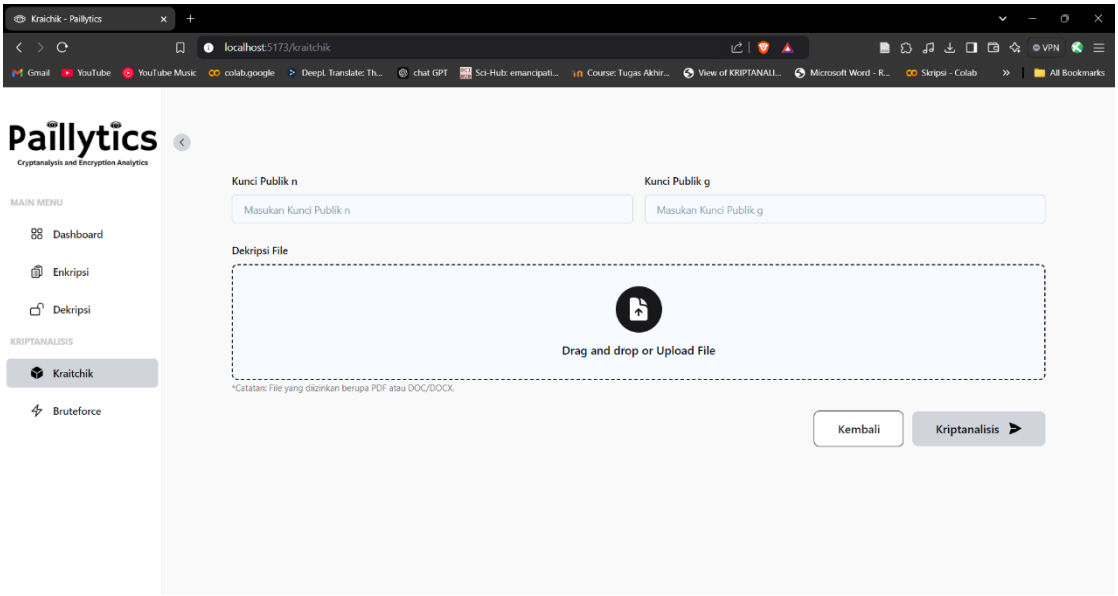
Laman dekripsi dapat diakses oleh penerima untuk melakukan dekripsi pada *ciphertext*. Keluaran dari laman ini adalah pesan asli



Gambar 4.4 Laman Dekripsi

4.1.5. Laman *Kraitchik*

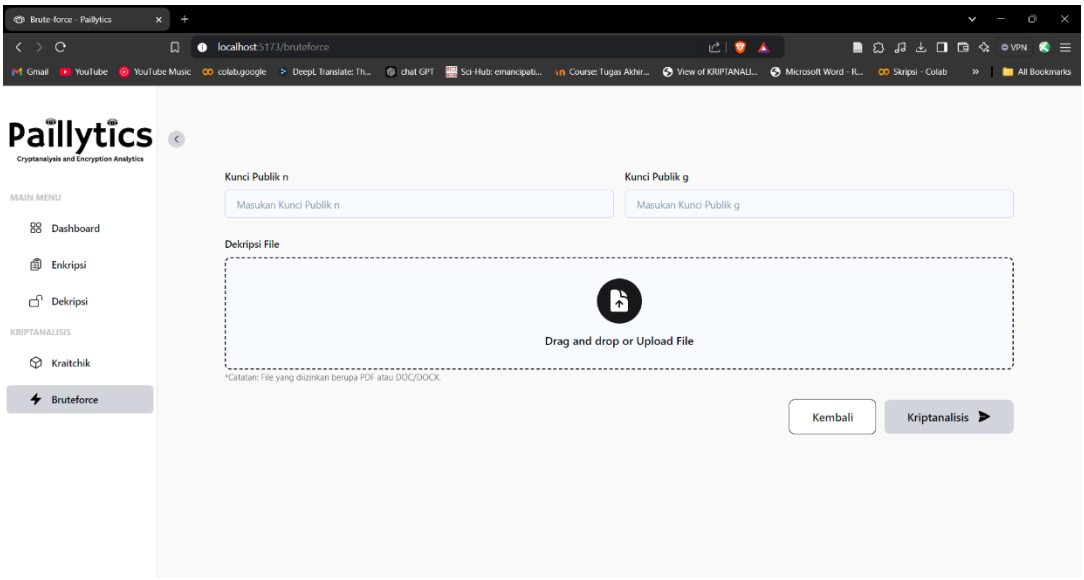
Laman *Kraitchik* dapat diakses oleh kriptanalisis untuk melakukan kriptanalisis menggunakan algoritma *Kraitchik* terhadap kunci publik. Keluaran dari laman ini adalah hasil faktorisasi dari kunci publik yakni bilangan prima *p* dan *q*.



Gambar 4.5 Laman *Kraitchi*k

4.1.6. Laman Brute Force

Laman *Brute Force* dapat diakses oleh kriptanalisis untuk melakukan kriptanalisis menggunakan algoritma *Brute-force* terhadap kunci publik. Keluaran dari laman ini adalah hasil faktorisasi dari kunci publik yakni bilangan prima *p* dan *q*.



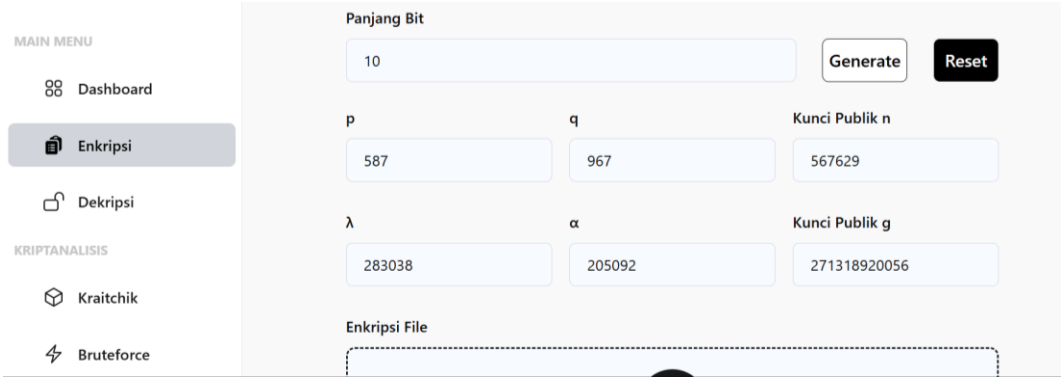
Gambar 4.6 Laman *Brute Force*

4.2. Pengujian Sistem

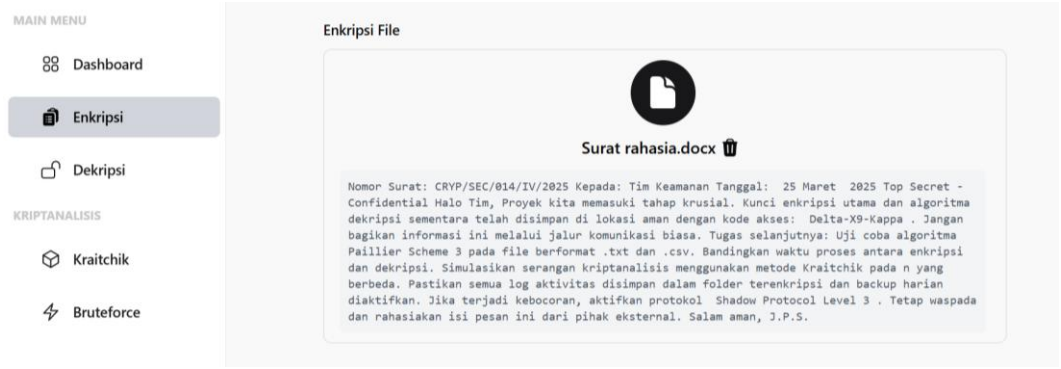
Dalam fase pengujian sistem, sistem yang berhasil dikembangkan akan diuji guna memastikan bahwa sistem dapat melakukan enkripsi pada pesan serta mengembalikannya menjadi bentuk semula menggunakan algoritma *Paillier Scheme 3* dan *Kraitchik.*

4.2.1. Pengujian Enkripsi

Dalam tahap ini akan dimulai dengan memasukkan nilai panjang bit kemudian menekan tombol "*Generate*" untuk membangkitkan *p* dan *q* sehingga menghasilkan nilai kunci publik n serta membangkitkan nilai α dan λ untuk menghasilkan nilai kunci publik g. Kunci yang telah dibangkitkan kemudian dikirim untuk digunakan pada proses enkripsi dan dekripsi. Selanjutnya proses enkripsi dilakukan dengan memasukkan dokumen asli. Sistem akan memberi keluaran isi dari dokumen. Kemudian menekan tombol "*Enkripsi*".



Gambar 4.7 Pembangkitan Kunci

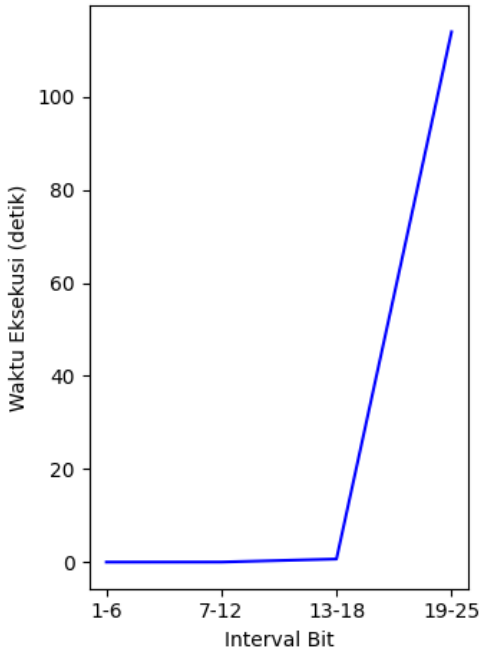


Gambar 4.8 Enkripsi Pesan

Pada gambar 4.7 menunjukkan bagaimana pembangkitan kunci dilakukan. Pembangkitan bilangan prima *p* dan *q* pada sistem dilakukan dengan cara memilih angka secara acak dalam panjang bit tertentu, lalu menguji keprimaan angka tersebut dengan menggunakan algoritma *Fermat's Prime*. Waktu eksekusi untuk menguji keprimaan sebuah angka dengan algoritma *Fermat's Prime* ditunjukkan pada Tabel 4.1.

Table 4.1 Waktu eksekusi algoritma *Fermat's Prime*

Panjang Interval Bit	Kemungkinan Prima	Waktu (detik)
1-6	17	0.000000
7-12	546	0.012836
13-18	22436	0.822886
19-25	2040714	120.961622



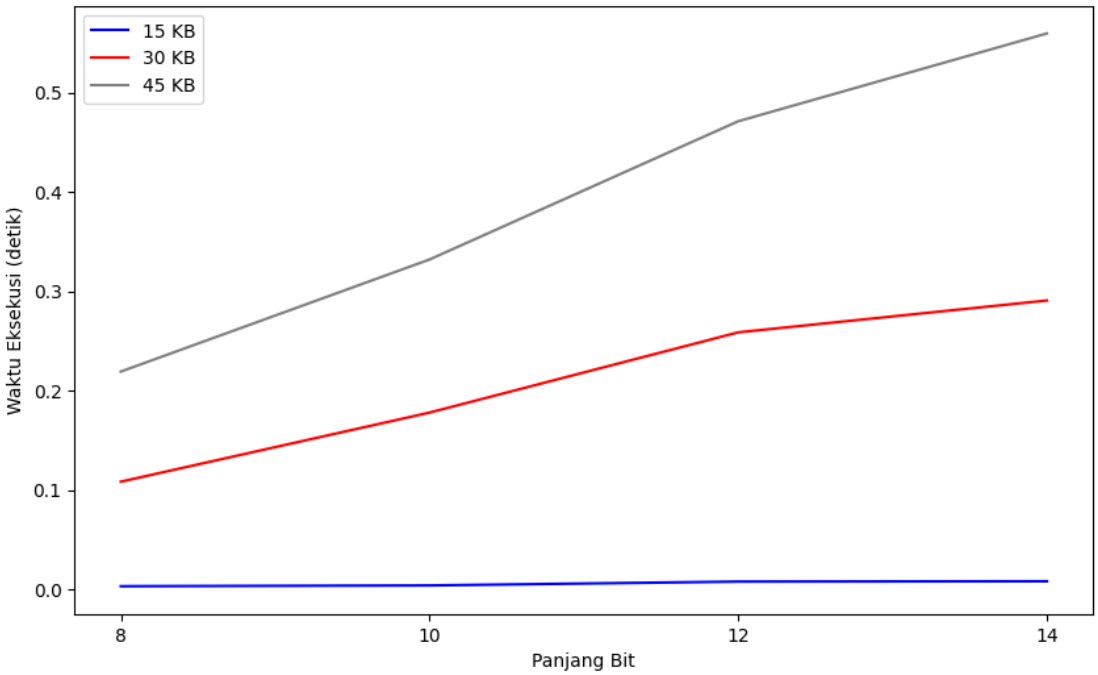
Gambar 4.9 Grafik perbandingan panjang bit dengan rata-rata waktu eksekusi algoritma *Fermat's Prime*

Pada Gambar 4.9 menunjukkan bahwa semakin panjang interval bit, semakin lama waktu eksekusi yang dibutuhkan untuk mengevaluasi bilangan prima, dengan lonjakan signifikan terjadi pada interval bit 13-18 ke 19-25.

Pada Tabel 4.2 menunjukkan waktu eksekusi untuk proses enkripsi menggunakan algoritma *Paillier Scheme 3.*

Tabel 4.2 Waktu eksekusi proses enkripsi algoritma *Paillier Scheme 3* terhadap perubahan ukuran kunci dan ukuran dokumen *plaintext*

Panjang Bit	Jumlah Karakter (p = 869, 15KB)	Jumlah Karakter (p = 36468,30KB)	Jumlah Karakter (p = 67375, 45KB)
8	0.00300 detik	0.10825 detik	0.21911 detik
10	0.00380 detik	0.17782 detik	0.33193 detik
12	0.00767 detik	0.25860 detik	0.47112 detik
14	0.00799 detik	0.29067 detik	0.55953 detik

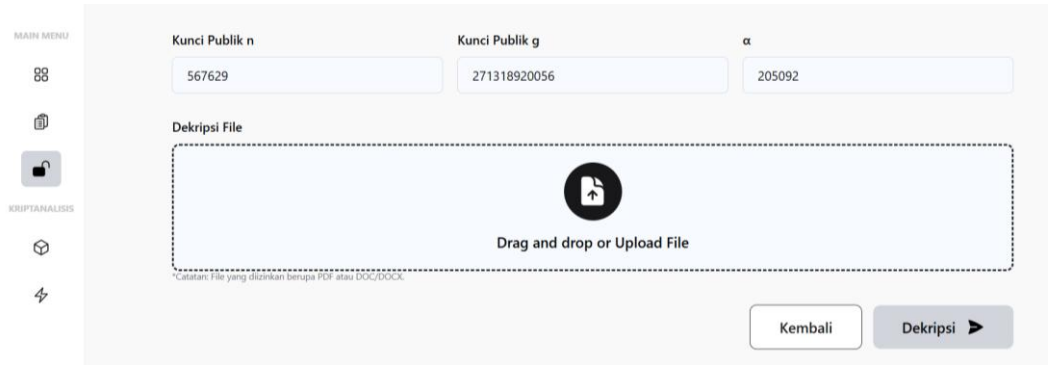


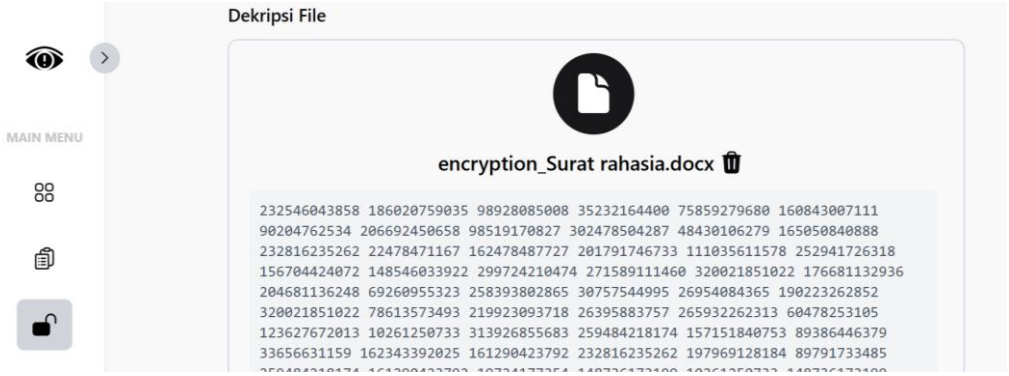
Gambar 4.10 Grafik pengujian waktu eksekusi enkripsi algoritma *Paillier Scheme 3* terhadap perubahan ukuran kunci dan ukuran dokumen *plaintext*

Pada Gambar 4.10 menunjukkan grafik pengujian waktu eksekusi enkripsi yang dipengaruhi oleh variasi ukuran kunci (panjang bit) dan ukuran dokumen *plaintext* (15 KB, 30 KB, dan 45 KB), di mana waktu eksekusi meningkat seiring bertambahnya ukuran kunci dan dokumen.

4.2.2. Pengujian Dekripsi

Pada tahap pengujian dekripsi kunci publik n dan g serta α memuat data kunci yang telah dibangkitkan pada proses pembangkitan kunci pada laman Enkripsi. Proses dekripsi dilakukan dengan memasukkan dokumen yang telah di enkripsi kemudian menekan tombol "*Dekripsi*"



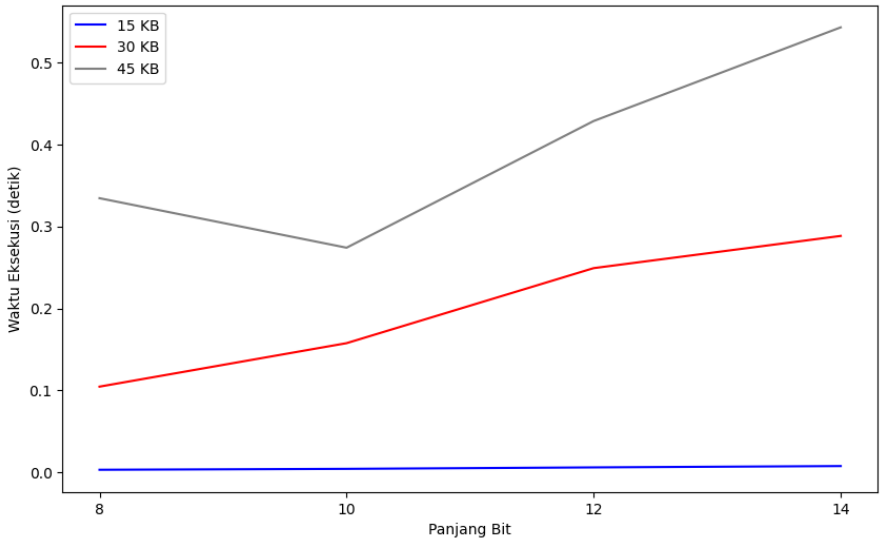


Gambar 4.11 Dekripsi Pesan

Pada Tabel 4.3 menunjukkan waktu eksekusi untuk proses dekripsi menggunakan algoritma *Paillier Scheme 3.*

Tabel 4.3 Waktu eksekusi proses dekripsi algoritma *Paillier Scheme 3* terhadap perubahan ukuran kunci dan ukuran dokumen *plaintext*

Panjang Bit	Jumlah Karakter (p = 869, 15KB)	Jumlah Karakter (p = 36468,30KB)	Jumlah Karakter (p = 67375, 45KB)
8	0.00314 detik	0.10453 detik	0.33438 detik
10	0.00425 detik	0.15756 detik	0.27399 detik
12	0.00603 detik	0.24907 detik	0.42852 detik
14	0.00765 detik	0.28836 detik	0.54276 detik



Gambar 4.12 Grafik pengujian waktu eksekusi dekripsi algoritma *Paillier Scheme 3* terhadap perubahan ukuran kunci dan ukuran dokumen *ciphertext*

Pada Gambar 4.12 menunjukkan hubungan antara waktu eksekusi proses dekripsi dengan panjang bit kunci dan ukuran dokumen *ciphertext*, menunjukkan bahwa peningkatan panjang bit kunci dan ukuran dokumen menyebabkan waktu eksekusi menjadi lebih lama.

4.2.3. Pengujian Kriptanalisis

Pada proses kriptanalisis kunci publik *n* didapatkan melalui proses pembangkitan kunci atau memasukkan nilai kunci publik *n* ke dalam *form*, kemudian menekan tombol "*Submit*". Proses kriptanalisis dilakukan menggunakan algoritma *Kraitchik* dan algoritma *Brute-force.*

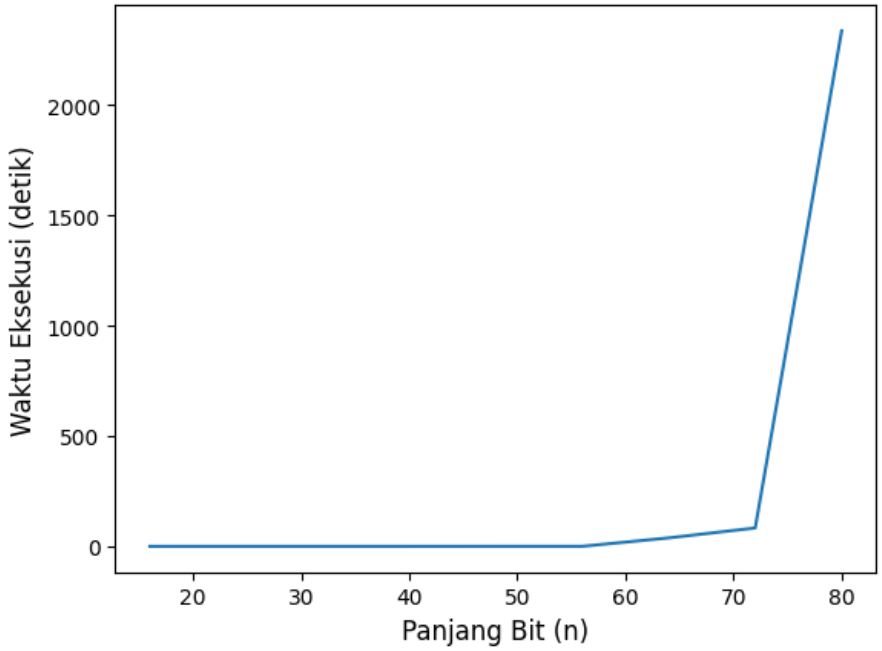
Gambar 4.13 Kriptanalisis menggunakan algoritma *Kraitchik*

Gambar 4.14 Kriptanalisis menggunakan algoritma *Brute-force*

Pada Gambar 4.13 dan Gambar 4.14 menunjukkan kunci publik berhasil di faktorisasi menggunakan algoritma *Kraitchik* dan algoritma *Brute-force.*

Tabel 4.4 Waktu eksekusi algoritma *Kraitchik*

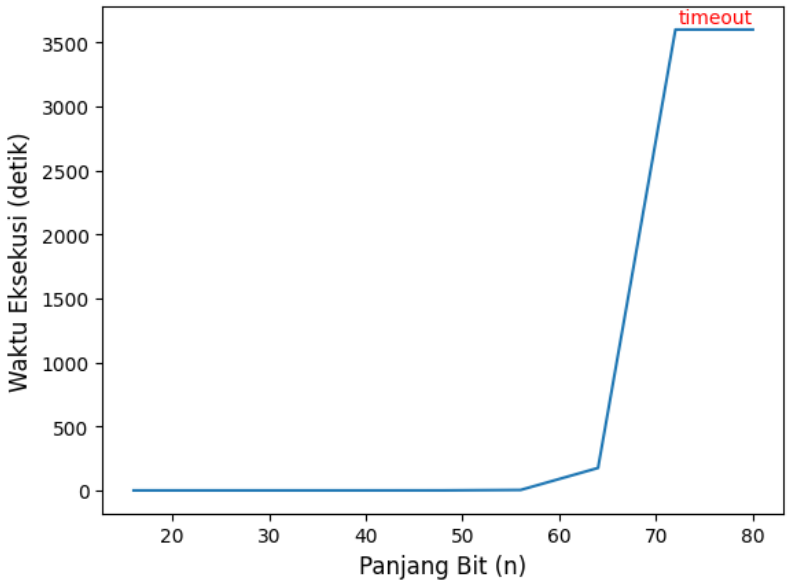
Panjang Bit	Kunci Publik (*n*)	*p*	*q*	Waktu Eksekusi (detik)
16	34277	227	151	0.000000
24	11571739	3967	2917	0.000000
32	2351306359	63671	36929	0.001006
40	792155202253	1025939	772127	0.004045
48	178470579851699	15314653	11653583	0.064009
56	40893118404262483	212129413	192774391	0.107388
64	11971710164707810513	4048347067	2957184739	38.296825
72	2808100171950000724891	56237968079	49932461429	83.215281
80	6470571028282868756339 59	884214651241	731787357199	2337.463865



Gambar 4.15 Grafik perbandingan panjang bit dengan waktu eksekusi algoritma *Kraitchik*

Tabel 4.5 Waktu eksekusi algoritma *Brute-force*

Panjang Bit	Kunci Publik (*n*)	*p*	*q*	Waktu Eksekusi (detik)
16	34277	227	151	0.000412
24	11571739	3967	2917	0.000000
32	2351306359	63671	36929	0.001092
40	792155202253	1025939	772127	0.016136
48	178470579851699	15314653	11653583	0.236060
56	40893118404262483	212129413	192774391	3.513936
64	11971710164707810513	4048347067	2957184739	175.238676
72	2808100171950000724891	56237968079	49932461429	3600(*time out*)
80	6470571028282868756339 59	884214651241	731787357199	3600(*time out*)



Gambar 4.16 Grafik perbandingan panjang bit dengan waktu eksekusi algoritma *Brute-force*

Berdasarkan hasil percobaan pada Tabel 4.4 dan Tabel 4.5 perbandingan antara algoritma *brute-force* dan *Kraitchik* menunjukkan perbedaan yang signifikan dalam hal waktu eksekusi. Untuk panjang bit kecil (seperti 16 dan 24), *brute-force* dapat

menyelesaikan faktorisasi dengan sangat cepat, bahkan mencapai waktu eksekusi 0 detik, namun seiring bertambahnya panjang bit, waktu eksekusinya meningkat tajam, terutama pada bit 64 dan 80. Sebaliknya, *Kraitchik* menunjukkan peningkatan waktu eksekusi yang lebih stabil meskipun tetap lebih lambat dibandingkan *brute-force* pada panjang bit kecil. *Kraitchik* lebih efisien dalam menangani bilangan besar dan memberikan hasil yang lebih konsisten, meskipun pada panjang bit 80 waktu eksekusinya masih cukup lama (2337.4 detik). Dengan demikian, *Kraitchik* lebih unggul untuk faktor bilangan yang lebih besar, sementara *brute-force* lebih efektif pada bilangan kecil.

4.3. Perhitungan Avalanche Effect

Avalanche Effect adalah metode untuk mengukur seberapa besar perubahan pesan selama proses enkripsi, dengan membandingkan rasio antara jumlah bit yang berubah pada *ciphertext* dan jumlah bit pada *plaintext* sebelum modifikasi. Semakin tinggi persentase perubahan, semakin baik kualitas enkripsi yang dihasilkan. Pengujian *Avalanche Effect* dianggap optimal jika perubahan bit berada dalam rentang 45-60%, dengan 50% sebagai angka ideal (Kangmouse & Handoko, 2022). Perubahan sebesar 50% menciptakan tantangan signifikan bagi penyerang, sehingga meningkatkan keamanan data yang dilindungi sesuai dengan rumus yang digunakan. Pada tabel 4.4 menunjukkan pengujian seberapa efektif algoritma *Paillier Scheme 3* dalam menghasilkan *Avalanche Effect.*

Tabel 4.6 Hasil Percobaan *Avalanche Effect*

Plaintext (Byte)	Jumlah Bit Berbeda	*Avalanche Effect*
16	36	56.25%
32	58	45.31%
64	153	59.77%
128	259	50.59%

Hasil pengujian *Avalanche Effect* pada algoritma *Paillier Scheme 3* menunjukkan bahwa algoritma ini efektif dalam menghasilkan perubahan signifikan pada *ciphertext* ketika satu bit diubah pada *plaintext*. Pada 16 *byte plaintext*, terjadi 36 bit perubahan

dengan *avalanche effect* sebesar 56.25%, menunjukkan perubahan yang cukup besar. Pada 32 *byte plaintext*, meskipun persentase turun sedikit menjadi 45.31%, ini masih dalam rentang yang dapat diterima. Hasil terbaik ditemukan pada 64 *byte plaintext* dengan 59.77% perubahan bit, yang mendekati angka ideal 50%. Sementara itu, 128 *byte plaintext* menghasilkan 50.59% perubahan bit, yang masih berada dalam rentang optimal. Secara keseluruhan, hasil pengujian ini menunjukkan bahwa algoritma *Paillier Scheme 3* memiliki performa yang baik dalam menyebarkan perubahan bit dan menjaga keamanan *ciphertext*.

4.4. Kompleksitas Algoritma

Kompleksitas algoritma digunakan untuk mengukur efisiensi algoritma berdasarkan waktu yang dibutuhkan dan memori yang digunakan seiring bertambahnya ukuran masukan (n). Kompleksitas waktu menunjukkan kecepatan eksekusi algoritma, sedangkan kompleksitas ruang menggambarkan jumlah memori yang diperlukan. Biasanya, kompleksitas dinyatakan menggunakan notasi Big-O, seperti $O\ (1)$ untuk konstanta, $O\ (n)$ untuk linear, dan $O\ (n^2)$ untuk kuadratik, yang menunjukkan bagaimana kinerja algoritma berubah dengan ukuran masukan. Kompleksitas algoritma akan dijelaskan dalam proses pembangkitan kunci, enkripsi, dan dekripsi, yang dianalisa berdasarkan efisiensi waktu yang diperlukan.

Tabel 4.7 Kompleksitas Pembangkitan Kunci

Kode Program	C	#	C#
def is_prime(p):			
assert p > 2 and p % 2 == 1	C1	1	C1
t = len(str(p)) * 3	C2	$log\ (p)$	C2 $log\ (p)$
for i in range(t):	C3	t	C3 t
a = random.randint(2, p - 1)	C4	t	C4 t
if pow(a, p - 1, p) != 1:	C5	$log^2\ (p)$	C5 $log^2\ (p)$
return False	C6	1	C6
return True	C7	1	C7
Kompleksitas total fungsi: $= \mathrm{C1} + \mathrm{C2}\log(p) + \mathrm{C3}\ t + \mathrm{C4}\ t + \mathrm{C5}\ log^2\ (p) + \mathrm{C6} + \mathrm{C7}$			

$= O\ (log^2\ (p))$

Kode Program	C	#	C#
def generate_primes(bit_len):			
low = 2 ** (bit_len - 1)	C1	1	C1
high = 2 ** bit_len - 1	C1	1	C1
def find_prime(low, high):			
while True:	C2	$log\ (range)$	C2 $log\ (range)$
p = random.randrange(low | 1, high, 2)	C3	1	C3
if is_prime(p):	C4	$log^2\ (p)$	C4 $log^2\ (p)$
return p	C5	1	C5
p1 = find_prime(low, (low + high) // 2)	C6	$log(range)$ $.log^2\ (p)$	C6 $log(range)$ $.log^2\ (p)$
p2 = find_prime((low + high) // 2 + 1, high)	C6	$log(range)$ $.log^2\ (p)$	C6 $log(range)$ $.log^2\ (p)$
return p1, p2	C7	1	C7

Kompleksitas total fungsi:

$= \mathrm{C1}\ +\ \mathrm{C2}\log(range) +\ \mathrm{C3}\ t\ +\ \mathrm{C4}\ log^2\ (p) + \mathrm{C5} +\ \mathrm{C6}\ log(range)$

$.\ log^2\ (p) +\ \mathrm{C6}\ log\ (range)\ .log^2\ (p)\ +\ \mathrm{C7}$

$= O\ (log\ (range)\ .log^2\ (p))$

Karena range = [low, high], di mana low = 2^{bit_len-1} dan high = $2^{bit\_len} - 1$, ukuran range adalah kira-kira range $\sim 2^{bit\_len}$. Dalam kode program p adalah angka acak dalam range $[2^{bit\_len-1}, 2^{bit\_len} - 1]$, nilainya memiliki skala yang sama dengan range $\sim 2^{bit\_len}$.

Sehingga $O(log(p)) = O(log(2^{bit_{len}})) = O(log(bit\_len))$

Kompleksitas total fungsi: $O\ (log^3\ (p))$

Kode Program	C	#	C#
def generate_g(n, alpha):			
n_squared = n ** 2	C1	$log\ (n)$	C1 $log\ (n)$
while True:	C2	$log\ (n)$	C2 $log\ (n)$

g = random.randint(1, n_squared - 1)	C3	1	C3
if math.gcd(g, n_squared) == 1:	C4	$log\ (n)^3$	C4 $log\ (n)^3$
if pow(g, alpha * n, n_squared) == 1:	C5	$log\ (n)^3$	C5 $log\ (n)^3$
return g	C6	1	C6
Kompleksitas total fungsi: $= C1\ log\ (n)\ + C2\ log\ (n) + \ C3 + C4\ log\ (n)^3 + C5\ log\ (n)^3 + \ C6$ $= O\ (log\ (n)^3)$			
Kode Program	**C**	**#**	**C#**
def generate_paillier_keys(bit_len=10):			
p, q = generate_primes (bit_len)	C1	$log^3(bit\_len)$	C3 $log^3(bit\_len)$
n = p * q	C2	$log^2(bit\_len)$	C2 $log^2(bit\_len)$
lambda_ = math.lcm(p - 1, q - 1)	C3	$log\ (bit\_len)$	C3 $log\ (bit\_len)$
alpha = random.randint(1, lambda_ - 1)	C4	$log\ (bit\_len)$	C4 $log\ (bit\_len)$
g = generate_g(n, alpha)	C5	$log^3(bit\_len)$	C5 $log^3(bit\_len)$
return p, q, n, lambda_, alpha, g	C6	1	C6
Kompleksitas total fungsi: $= C1\ log^3(bit\_len) + C2\ log^2(bit\_len) + \ C3\ log\ (bit\_len) + C4\ log\ (bit\_len)$ $+C5\ log^3(bit\_len) + \ C6$ $= O\ (log^3(bit\_len))$			

Tabel 4.8 Kompleksitas Enkripsi

Kode Program	**C**	**#**	**C#**
def encryption_paillier(g, n, m, r):			
c = pow(g,m+(n*r), n**2)	C1	$log\ (n)\ .log^2\ (n)$	C1 $log\ (n)\ .log^2\ (n)$
return c	C2	1	C2
Kompleksitas total fungsi: $= \ C1\ log\ (n)\ .log^2\ (n)\ + C2$ $= O\ (log\ (n)\ .log^2\ (n))$			
Kode Program	**C**	**#**	**C#**
def encrypt_text(text, g, n, alpha):			

encrypted_text = []	C1	1	C1
for char in text:	C2	t	C2 t
m_ascii = ord(char)	C3	1	C3
r = random.randint(1, alpha - 1)	C4	$t \,.\, log\ (alpha)$	C4 $t \,.\, log\ (alpha)$
c = encryption_paillier(g, n, m_ascii, r)	C5	$t \,.\, \log(n) \,.\, log^2\ (n)$	C5 $t \,.\, log\ (n) \,.\, log^2\ (n)$
encrypted_text.append(c)	C6	1	C6
return encrypted_text	C7	1	C7
Kompleksitas total fungsi: $= \mathrm{C1} + \mathrm{C2}\ t + \mathrm{C3} + \mathrm{C4}\ t \,.\, log(alpha) + \mathrm{C5}\ t \,.\, log\ (n) \,.\, log^2\ (n) + \mathrm{C6} + \mathrm{C7}$ $= O\ (t \,.\, log\ (n) \,.\, log^2\ (n))$			

Tabel 4.9 Kompleksitas Dekripsi

Kode Program	**C**	**#**	**C#**
def L(x, n):			
return (x - 1) // n	C1	1	C1
Kompleksitas total fungsi: $= \mathrm{C1}$ $= O\ (1)$			
Kode Program	**C**	**#**	**C#**
def decryption_paillier(c, g, alpha, n):			
n_squared = n ** 2	C1	1	C1
L1 = L(pow(c, alpha, n_squared), n)	C2	$log(alpha) \cdot log^2(n)$	C2 $log(alpha) \cdot log^2(n)$
L2 = L(pow(g, alpha, n_squared), n)	C3	$log(alpha) \cdot log^2(n)$	C3 $log(alpha) \cdot log^2(n)$
m = (L1 * pow(L2, -1, n)) % n	C4	$log^2(n)$	C4 $log^2(n)$
if c < n_squared:	C5	1	C5 1
return m	C6	1	C6
Kompleksitas total fungsi: $= \mathrm{C1} + \mathrm{C2}\ log(alpha) \cdot log^2(n) + \mathrm{C3}\ log(alpha) \cdot log^2(n) + \mathrm{C4}\ log^2(n)$ $+\mathrm{C5} + \mathrm{C6}$ $= O\ (log(alpha) \cdot log^2(n))$			

Kode Program	C	#	C#
def decrypt_text(encrypted_text, g, alpha, n):			
decrypted_text = ""	C1	1	C1
for c in encrypted_text:	C2	t	C2 t
m_decrypted_ascii = decryption_paillier(c, g, alpha, n)	C3	$log(alpha) \cdot log^2(n)$	C3 $log(alpha) \cdot log^2(n)$
decrypted_text += chr(m_decrypted_ascii)	C4	1	C4
return decrypted_text	C5	1	C5
Kompleksitas total fungsi: $= C1 + C2\ t + C3\ log(alpha) \cdot log^2(n) + C4 + C5$ $= O\ (t\ .log(alpha) \cdot log^2(n))$			

Tabel 4.10 Kompleksitas *Kraitchik*

Kode Program	C	#	C#
def check_is_integer(z):			
if isinstance(z, float) and not(z.is_integer()):	C1	1	C1
return True	C2	1	C2
elif isinstance(z, (int, float)):	C1	1	C1
return False	C2	1	C2
Kompleksitas total fungsi: $= C1 + C2$ $= O\ (1)$			
Kode Program	**C**	**#**	**C#**
def kraitchik_factorization(n):			
while check_is_integer(math.sqrt(n)):	C1	$log^2(n)$	C1 $log^2(n)$
x = math.ceil(math.sqrt(n))	C2	$log^2(n)$	C2 $log^2(n)$
while True:	C3	$log^2(n).I$	C3 $log^2(n).I$
k = 1	C4	1	C4
while x**2 - k * n >= 0:	C5	$log^2(n).I\ .t$	C5 $log^2(n).I\ .t$

y = math.sqrt(x**2 - k * n)	C6	$log^2(n).I.t$	C6 $log^2(n).I.t$
if y.is_integer():	C7	1	C7
y = int(y)	C4	1	C4
if (x + y) % n != 0 and (x - y) % n != 0:	C8	$log\ (n)$	C8 $log\ (n)$
p = math.gcd(x + y, n)	C9	$log\ (n)$	C9 $log\ (n)$
q = math.gcd(x - y, n)	C9	$log\ (n)$	C9 $log\ (n)$
if p * q == n:	C10	$log\ (n)$	C10 $log\ (n)$
return p, q	C11	1	C11
k += 2	C4	1	C4
x += 1	C4	1	C4
Kompleksitas total fungsi: $= \mathrm{C1}\ log^2(n) + \mathrm{C2}\ log^2(n) + \mathrm{C3}\ log^2(n).I + \mathrm{C4} + \mathrm{C5}\ log^2(n).I + \mathrm{C6}\ log^2(n).I + \mathrm{C7} + \mathrm{C8}\ log\ (n) + \mathrm{C9}\ log\ (n) + \mathrm{C10}\ log\ (n) + \mathrm{C11}$ $= O\ (log^2(n).I.t)$			

BAB V

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan hasil analisis dan pengujian implementasi sistem Kriptanalisis Algoritma *Paillier Scheme 3* menggunakan *Kraitchik* dan *Brute-force* pada pesan berekstensi *.docx atau *.pdf, dapat disimpulkan:

1. Berdasarkan pengujian, *Paillier Scheme 3* memiliki tingkat keamanan yang bergantung pada panjang kunci. Pada 80 bit, *Kraitchik* dapat memecahkan enkripsi dalam 2337.46 detik (39 menit), sementara *Brute-force* mengalami *timeout* setelah 3600 detik. Pada 64 bit, enkripsi masih rentan karena dapat dipecahkan dalam 38.29 detik (*Kraitchik*) dan 175.23 detik (*Brute-force*). Semakin panjang kunci, semakin tinggi keamanannya terhadap serangan kriptanalisis.
2. Proses enkripsi dan dekripsi pada algoritma *Paillier Scheme 3* menunjukkan peforma yang baik untuk dokumen berukuran hingga 65KB (67375 karakter atau lebih).
3. Waktu eksekusi untuk enkripsi, dekripsi, dan kriptanalisis menggunakan Algoritma *Paillier Scheme 3* meningkat seiring dengan bertambahnya ukuran dokumen dan jumlah karakter, membentuk grafik waktu eksekusi yang cenderung menaik.

5.2. Saran

Penulis memberikan beberapa saran untuk penelitian lanjutan:

1. Penelitian selanjutnya diharapkan dapat mendukung proses enkripsi dan dekripsi untuk dokumen dengan format selain *.docx dan *.pdf dan menguji sistem menggunakan berbagai metode serangan kriptografi lainnya untuk meningkatkan ketahanan sistem terhadap ancaman yang lebih kompleks.
2. Disarankan untuk menggunakan bilangan prima yang lebih besar dalam pengujian enkripsi, guna meningkatkan performa serta keandalan sistem terhadap serangan.

DAFTAR PUSTAKA

Britannica. (2024, August 9). *Wilson's theorem | Number Theory, Prime Numbers & Congruence | Britannica*. https://www.britannica.com/science/Wilsons-theorem

Budiman, M. A., Rachmawati, D., & Lydia, S. (2019). The implementation of the Pollard factorization method in the cryptanalysis of the Rabin public key algorithm. *Journal of Physics: Conference Series*, *1235*(1), 012086. https://doi.org/10.1088/1742-6596/1235/1/012086

Google Cloud. (2024). *Apa itu enkripsi dan bagaimana cara kerjanya?* Google Cloud. https://cloud.google.com/learn/what-is-encryption

Hapsari, R. D., & Pambayun, K. G. (2023). Ancaman Cybercrime di Indonesia. *Fakultas Perlindungan Masyarakat IPDN Jatinangor*. https://doi.org/10.33701/jk.v5i1.3208

Jost, C., Lam, H., Maximov, A., & Smeets, B. (2015). *Encryption Performance Improvements of the Paillier Cryptosystem* (No. 2015/864). Cryptology ePrint Archive. https://eprint.iacr.org/2015/864

Kangmouse, M., & Handoko, B. (2022). Pengujian Avalanche Effect pada Kriptografi Teks Menggunakan Autokey Cipher. *ResearchGate*. https://doi.org/10.51903/semnastekmu.v2i1.162

Lydia, M. S., Budiman, M. A., & Rachmawati, D. (2021). Factorization of Small RPrime RSA Modulus Using Fermat's Difference of Squares and Kraitchik's Algorithms in Python. . . *Vol., 11*.

Ma, H., Han, S., & Lei, H. (2021). Optimized Paillier's Cryptosystem with Fast

Encryption and Decryption. *Annual Computer Security Applications Conference*, 106–118. https://doi.org/10.1145/3485832.3485842

Mulya, M. F., Rismawati, N., & Trisanto, D. (2021). Analisis Dan Perancangan Simulasi Algoritma Paillier Cryptosystem Pada Pesan Text Dengan Presentation Format Binary, Octal, Hexadecimal dan Base64. *Faktor Exacta, 13*(4), Article 4. https://doi.org/10.30998/faktorexacta.v13i4.7429

Purba, S. (2019). Kriptanalisis Kunci Publik Algoritma Rabin Menggunakan Metode Kraitchik. *Jurnal Sains Dan Teknologi ISTP*, *11*(2), Article 2. https://doi.org/10.59637/jsti.v11i2.25

Suryawijaya, T. W. E. (2023). Memperkuat Keamanan Data melalui Teknologi Blockchain: Mengeksplorasi Implementasi Sukses dalam Transformasi Digital di Indonesia. *Jurnal Studi Kebijakan Publik, 2*(1), Article 1. https://doi.org/10.21787/jskp.2.2023.55-68