

**HYBRID CRYPTOSYSTEM DENGAN MARTINO HOMOMORPHIC
ENCRYPTION DAN GENERALIZATION OF THE ELGAMAL**

SKRIPSI

ANNISA KHAIRINA

211401042



PROGRAM STUDI S-1 ILMU KOMPUTER

**FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA**

MEDAN

2024

PERSETUJUAN

Judul : HYBRID CRYPTOSYSTEM DENGAN MARTINO
 HOMOMORPHIC ENCRYPTION DAN
 GENERALIZATION OF THE ELGAMAL

Kategori : SKRIPSI

Nama : ANNISA KHAIRINA

Nomor Induk Mahasiswa : 211401042

Program Studi : SARJANA (S-1) ILMU KOMPUTER

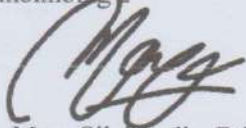
Fakultas : ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
 UNIVERSITAS SUMATERA UTARA

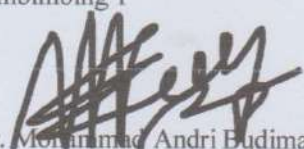
Telah diuji dan dinyatakan lulus di Medan, 09 Januari 2025.

Komisi Pembimbing :

Pembimbing 2

Pembimbing 1

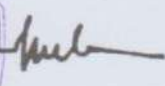

 Dr. Maya Silvi Lydia, B.Sc, M.Sc
 NIP. 197401272002122001


 Dr. Mohammad Andri Budiman, S.T., M.Comp. Sc., M.E.M.
 NIP. 197510082008011011

Diketahui/Disetujui Oleh

Program Studi S-1 Ilmu Komputer




 Dr. Amalia ST., M.T.

NIP. 197812212014042001

PERNYATAAN**HYBRID CRYPTOSYSTEM DENGAN MARTINO HOMOMORPHIC
ENCRIPTION DAN GENERALIZATION OF THE ELGAMAL****SKRIPSI**

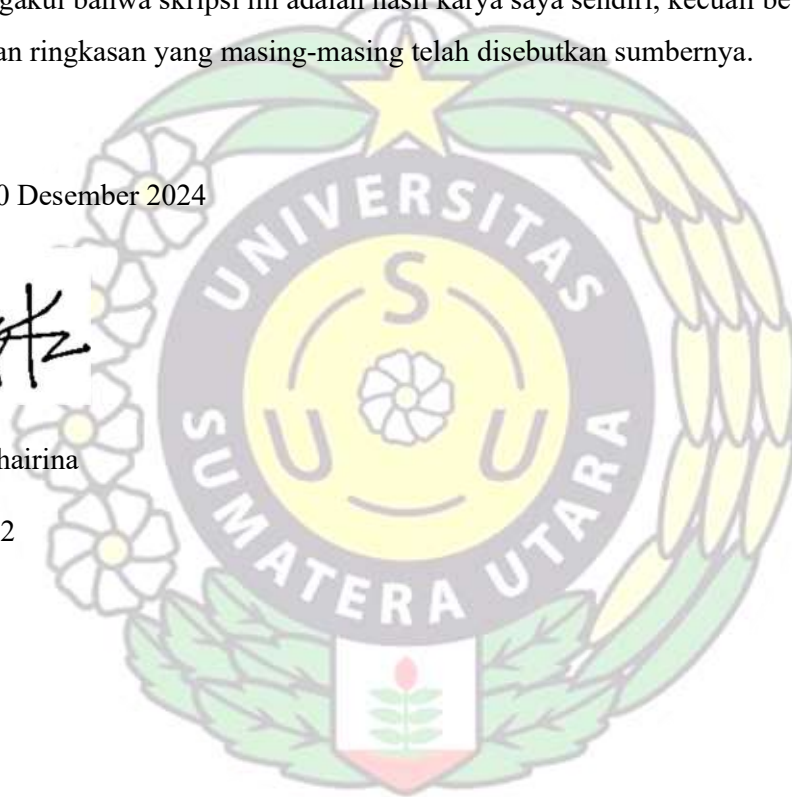
Saya mengakui bahwa skripsi ini adalah hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing telah disebutkan sumbernya.

Medan, 20 Desember 2024



Annisa Khairina

211401042



PENGHARGAAN

Bismillahirrahmanirrahim, puji dan syukur dipanjatkan kepada Allah *Subhanahu Wa Ta'ala* yang telah memberikan rahmat-Nya sehingga penulis dapat berada pada tahap penyusunan skripsi ini sebagai syarat untuk mendapatkan gelar Sarjana Komputer di Program Studi S-1 Ilmu Komputer, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara. Shalawat beserta salam kita panjatkan kepada Rasulullah *Shalallaahu 'Alayhi Wasallam* yang telah membimbing kita dari zaman kegelapan menuju zaman terang benderang saat ini.

Dengan penuh rasa hormat pada kesempatan ini penulis mengucapkan terima kasih sebesar-besarnya kepada Mama tercinta, Hj. Z. Siswanti S.S. atas segala bentuk cinta, kasih sayang, doa-doa yang dipanjatkan dan pengorbanan yang diberikan untuk penulis. Dan terima kasih kepada Almarhum Papa tercinta, H. Ir. Andry Yuzar atas dukungan dan kasih sayang yang membersamai di setiap langkah penulis. Terima kasih untuk setiap dukungan yang telah diberikan hingga penulis dapat berada di titik ini.

Penyusunan skripsi ini tidak terlepas dari bantuan, dukungan, dan bimbingan dari banyak pihak. Oleh karena itu, penulis mengucapkan banyak terima kasih 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. Bapak Dr. Mohammad Andri Budiman S.T., M.Comp.Sc., M.E.M. selaku Dosen Pembimbing I yang telah memberikan banyak bimbingan dan masukan yang berharga kepada penulis selama proses penyusunan skripsi ini.

6. Ibu Dr. Maya Silvi Lydia B.Sc., M.Sc. selaku dosen Pembimbing II yang telah memberikan bimbingan serta masukannya kepada penulis selama proses penyusunan skripsi ini.
7. Bapak Herriyance, S.T., M.Kom. selaku Dosen Pembimbing Akademik yang telah memberikan saran dan motivasi kepada penulis selama perkuliahan,
8. Seluruh Bapak dan Ibu Dosen Program Studi S-1 Ilmu Komputer, yang telah membimbing penulis selama masa perkuliahan hingga akhir masa studi.
9. Teristimewa kepada kedua orang tua penulis, Hj. Z. Siswanti S.S, dan H. Ir. Andry Yuzar (Alm) yang telah memberikan penulis kasih sayang yang tiada henti, ilmu yang bermanfaat, berbagai doa untuk penulis, memfasilitasi dan juga menyokong perkuliahan penulis sehingga penulis dapat menjalani perkuliahan dengan baik sampai bisa di titik ini.
10. Kepada Nenek tercinta, Hj. Basyariah, DRA (Alm), Tante Bebi Mushanah S.H., M.Kn., Om Ir. Indra Dermawan, serta sepupu penulis Alya Khalisa Zafira yang selalu mendukung dan mendoakan penulis dalam setiap langkah yang diambil.
11. Kepada teman seperjuangan penulis yang telah memberikan banyak semangat.

Dan seluruh pihak yang telah memberikan dukungan seta doa baik yang tidak dapat penulis sebutkan satu per-satu. Semoga Allah *Subhanahu Wa Ta'ala* senantiasa melimpahkan keberkahan serta kemudahan atas semua dukungan yang telah diberikan kepada penulis dan semoga temuan penelitian ini dapat memberikan inspirasi dan manfaat di masa mendatang.

Medan, 16 Desember 2024

Penulis,

Annisa Khairina

ABSTRAK

HYBRID CRYPTOSYSTEM DENGAN MARTINO HOMOMORPHIC ENCRYPTION DAN GENERALIZATION OF THE ELGAMAL

Keamanan data digital menjadi tantangan utama di era digital yang terus berkembang. Sistem kriptografi kunci asimetris, seperti Generalization of the ElGamal, memiliki keunggulan dalam hal keamanan yang tinggi karena didasarkan pada sulitnya menghitung logaritma diskrit dari modulus prima yang besar. Namun, algoritma ini kurang efisien dalam menangani dokumen berukuran besar dan memiliki kecepatan enkripsi yang lebih lambat. Di sisi lain, algoritma simetris seperti Martino Homomorphic Encryption menawarkan kecepatan enkripsi dan dekripsi yang tinggi, serta dilengkapi fitur *avalanche effect* yang memastikan perubahan kecil pada kunci atau plaintext menghasilkan perubahan signifikan pada ciphertext. Namun, algoritma ini menghadapi tantangan dalam pengelolaan kunci. Penelitian ini mengusulkan skema hybrid cryptosystem yang menggabungkan kelebihan dari kedua algoritma tersebut. Skema ini menggunakan algoritma Generalization of the ElGamal untuk mendistribusikan kunci simetris secara aman dan Martino Homomorphic Encryption untuk enkripsi data aktual. Dengan kombinasi ini, pendekatan hybrid cryptosystem mampu meningkatkan efisiensi dan keamanan dalam transmisi data digital, memberikan perlindungan optimal terhadap ancaman siber.

Kata Kunci: Hybrid cryptosystem, Generalization of the ElGamal, logaritma diskrit, Martino Homomorphic Encryption, *avalanche effect*, keamanan data, kriptografi simetris, kriptografi asimetris.

ABSTRACT

HYBRID CRYPTOSYSTEM WITH MARTINO HOMOMORPHIC ENCRYPTION AND GENERALIZATION OF THE ELGAMAL

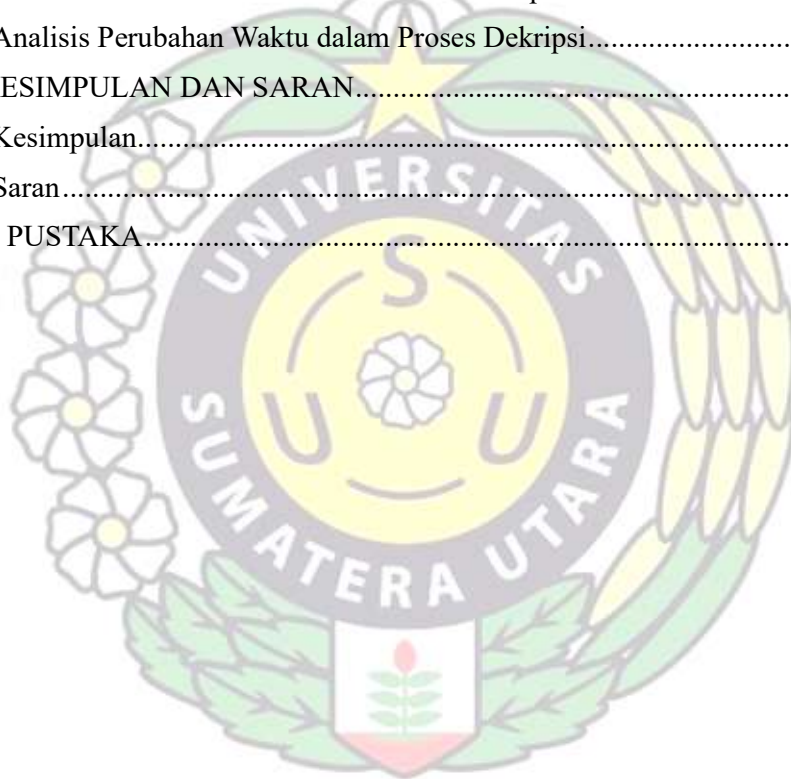
Digital data security has become a significant challenge in the ever-evolving digital era. Asymmetric cryptographic systems, such as the Generalization of the ElGamal, excel in providing high security due to the difficulty of solving discrete logarithms in large prime modulo. However, this algorithm is inefficient for handling large-sized documents and has slower encryption speeds. Conversely, symmetric algorithms like Martino Homomorphic Encryption offer high encryption and decryption speeds, with the added advantage of the *avalanche effect*, ensuring significant changes in ciphertext with minor alterations in the key or plaintext. Nonetheless, symmetric algorithms face challenges in secure key management. This study proposes a hybrid cryptosystem combining the strengths of these two algorithms. The Generalization of the ElGamal algorithm secures symmetric key distribution, while the Martino Homomorphic Encryption algorithm encrypts actual data efficiently. This hybrid cryptosystem can enhance the efficiency and security of digital data transmission, providing optimal protection against cyber threats.

Keywords: *Hybrid cryptosystem*, Generalization of the ElGamal, discrete logarithm, Martino Homomorphic Encryption, *avalanche effect*), data security, symmetric cryptography, asymmetric cryptography

DAFTAR ISI

PERSETUJUAN	ii
PERNYATAAN.....	iii
PENGHARGAAN	iv
ABSTRAK	vi
ABSTRACT.....	vii
DAFTAR TABEL	x
DAFTAR GAMBAR	xi
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah.....	3
1.3 Batasan Masalah.....	3
1.4 Tujuan Penelitian.....	4
1.5 Manfaat Penelitian	4
1.6 Metodologi Penelitian	4
1.7 Sistematika Penulisan.....	5
BAB II LANDASAN TEORI.....	6
2.1 Kriptosistem	6
2.2 Kriptosistem Hibrid.....	6
2.3 Enkripsi Kunci Simetris	7
2.4 Martino Homomorphic Encryption	7
2.5 Transformasi Matriks	7
2.6 Avalanche Effect	8
2.7 Enkripsi Kunci Asimetris	8
2.8 Generalization of the ElGamal	8
2.9 Fermat's Little Theorem	9
2.10 Akar Primitif	10
2.11 Modulo Eksponensial	11
2.12 Inversi Modulo	11
2.13 Extend Euclidean Algorithm	11
2.14 Contoh Kasus Penerapan Hybrid Cryptosystem dengan Algoritma Martino Homomorphic Encryption dan Generalization of the ElGamal	12
2.15 Penelitian yang Relevan	23

BAB III ANALISIS DAN PERANCANGAN	25
3.1 Analisis Sistem	25
3.2 <i>Use Case Diagram</i>	29
3.3 <i>Flowchart</i> Sistem	30
BAB IV IMPLEMENTASI DAN PENGUJIAN SISEM	38
4.1 Implementasi Sistem	38
4.2 Perhitungan <i>Avalanche Effect</i>	41
4.3 Perhitungan Kompleksitas Algoritma	42
4.4 Analisis Perubahan Waktu dalam Proses Enkripsi.....	56
4.5 Analisis Perubahan Data dalam Proses Enkripsi.....	57
4.6 Analisis Perubahan Waktu dalam Proses Dekripsi.....	58
BAB V KESIMPULAN DAN SARAN.....	59
5.1 Kesimpulan.....	59
5.2 Saran.....	59
DAFTAR PUSTAKA.....	60



DAFTAR TABEL

Tabel 2.1. Perhitungan Fermat's Little Theorem	9
Tabel 2.2. Pengujian Akar Primitif.....	10
Tabel 2.3. Bentuk ASCII dan biner dari setiap karakter.....	13
Tabel 2.4. Bentuk ASCII dan biner dari setiap karakter.....	22
Tabel 4.1. Hasil Percobaan Avalanche Effect.....	41
Tabel 4.2. Hasil Perhitungan Kompleksitas Algoritma Enkripsi.....	42
Tabel 4.3. Hasil Perhitungan Kompleksitas Algoritma Dekripsi.....	43
Tabel 4.4. Kompleksitas Algoritma Fungsi Pemeriksaan Bilangan Prima.....	43
Tabel 4.5. Kompleksitas Algoritma Fungsi Pemeriksaan Bilangan Akar Primitif.....	44
Tabel 4.6. Kompleksitas Algoritma Fungsi Pembangkit Kunci	45
Tabel 4.7. Kompleksitas Algoritma Fungsi Konversi String ke Matriks Biner.....	46
Tabel 4.8. Kompleksitas Algoritma Fungsi Rotasi <i>Shift Up</i>	46
Tabel 4.9. Kompleksitas Algoritma Fungsi Rotasi <i>Shift Right</i>	47
Tabel 4.10. Kompleksitas Algoritma Fungsi Rotasi <i>Shift Down</i>	47
Tabel 4.11. Kompleksitas Algoritma Fungsi Rotasi <i>Shift Left</i>	48
Tabel 4.12. Kompleksitas Algoritma Fungsi <i>Run-Length Encoding</i>	48
Tabel 4.13. Kompleksitas Algoritma Fungsi Enkripsi Simetris	49
Tabel 4.14. Kompleksitas Algoritma Fungsi Enkripsi Asimetris	50
Tabel 4.15. Kompleksitas Algoritma Fungsi Konversi String ke Matriks	50
Tabel 4.16. Kompleksitas Algoritma Fungsi Invers Modulo	51
Tabel 4.17. Kompleksitas Algoritma Fungsi <i>Run-Length Decoding</i>	52
Tabel 4.18. Kompleksitas Algoritma Fungsi Dekripsi Asimetris	53
Tabel 4.19. Kompleksitas Algoritma Fungsi Dekripsi Simetris	54
Tabel 4.20. Hasil Perubahan Waktu Enkripsi pada Skema Hibrid	55
Tabel 4.21. Hasil Perubahan Data Enkripsi pada Skema Hibrid	56
Tabel 4.22. Hasil Perubahan Waktu Dekripsi pada Skema Hibrid.....	58

DAFTAR GAMBAR

Gambar 2.1. Matriks “M” yang tersusun dari nilai biner setiap karakter	14
Gambar 2.2. Matriks ‘M1’ hasil dari rotasi matriks ‘M’	15
Gambar 2.3. Matriks ‘M2’ hasil dari rotasi matriks ‘M1’	16
Gambar 2.4. Matriks ‘MT’ hasil dari transpos matriks ‘M2’	16
Gambar 2.5. Array linier tunggal 'T'	17
Gambar 2.6 <i>Ciphertext</i>	17
Gambar 2.7 <i>Ciphertext</i>	19
Gambar 2.8. Array linier tunggal 'T'	19
Gambar 2.9. Matriks ‘MT’	19
Gambar 2.10. Matriks ‘M2’ hasil dari transpos matriks ‘MT’	20
Gambar 2.11. Matriks ‘M1’ hasil dari rotasi matriks ‘M2’	21
Gambar 2.12. Matriks ‘M1’ hasil dari rotasi matriks ‘M2’	22
Gambar 2.13. Matriks ‘M1’ hasil dari rotasi matriks ‘M2’	22
Gambar 3.1. Diagram <i>Ishikawa</i>	27
Gambar 3.2. Diagram Umum Sistem	28
Gambar 3.3. Use Case Diagram	29
Gambar 3.4. Flowchart Sistem Utama	30
Gambar 3.5. Flowchart Pembangkit Kunci (Key Generation)	31
Gambar 3.6. Flowchart Martino Homomorphic Encryption	32
Gambar 3.7. Flowchart Enkripsi Generalization of the ElGamal	33
Gambar 3.8. Flowchart Dekripsi Generalization of the ElGamal	34
Gambar 3.9. Flowchart Martino Homomorphic Decryption	35
Gambar 4.1. Halaman Awal Sistem	37
Gambar 4.2. Halaman Pembangkit Kunci	38
Gambar 4.3. Halaman Enkripsi	39
Gambar 4.4. Halaman Hasil Enkripsi	39
Gambar 4.5. Halaman Dekripsi	40
Gambar 4.6. Halaman Hasil Dekripsi	40
Gambar 4.7. Grafik Perubahan Waktu Enkripsi pada Skema Hibrid	55
Gambar 4.8. Grafik Perubahan Data Enkripsi pada Skema Hibrid	56
Gambar 4.9. Grafik Perubahan Waktu Dekripsi pada Skema Hibrid	57

BAB I

PENDAHULUAN

1.1 Latar Belakang

Keamanan data digital menjadi perhatian utama di era digital ini. Seiring dengan berkembangnya teknologi informasi, volume data yang disimpan dan dipertukarkan terus meningkat, membuka celah bagi para penjahat siber untuk melakukan aksi peretasan dan pencurian data. Untuk mengatasi hal ini, kriptografi hadir sebagai cara efektif untuk melindungi kerahasiaan dan integritas data.

Teknik Martino Homomorphic Encryption sebagai kriptografi kunci simetris melakukan transformasi matriks dengan pergeseran, rotasi, dan transposisi nilai ASCII yang dikonversi menjadi Biner dari setiap karakter dalam *plaintext* (Rupa, C. et. al. 2023). Untuk enkripsi dan dekripsi, kriptografi simetris menggunakan kunci rahasia yang sama. Pada algoritma ini “*avalanche effect*” adalah fitur enkripsi simetris yang diinginkan dari semua metode enkripsi di mana dua kunci berbeda menghasilkan *ciphertext* terpisah untuk pesan yang sama. Perubahan kecil pada kunci atau plaintext akan menghasilkan perubahan besar pada *ciphertext*.

Di sisi lain, kriptografi kunci simetris menawarkan kecepatan enkripsi dan dekripsi yang tinggi. Kecepatan enkripsi simetris membuatnya sangat efisien untuk mengenkripsi data dalam jumlah besar, namun memiliki kelemahan dalam pengelolaan kunci. Kunci simetris perlu didistribusikan secara aman ke semua pihak yang berkepentingan, dan hal ini menjadi kerumitan tersendiri (Sasikumar, K., & Nagarajan, S. 2024).

Kriptosistem kunci publik seperti ElGamal dapat menyediakan transmisi dokumen digital yang aman. Keamanan pada algoritma ElGamal didasarkan pada sulitnya menghitung logaritma diskrit dari modulus prima yang besar (Zega, I. et. al. 2023). Generalization of the ElGamal merupakan varian dari ElGamal yang melibatkan proses enkripsi bergantung pada modulus eksponensial (Ranasinghe, R., & Athukorala, P. 2022). Algoritma ini juga merupakan algoritma asimetris yang memiliki pasangan kunci publik dan privat untuk enkripsi dan dekripsi.

Namun, untuk mengamankan dokumen berukuran besar biasanya tidak efisien jika menggunakan sistem kriptografi asimetris secara langsung. *Ciphertext* yang dihasilkan biasanya jauh lebih besar daripada dokumen asli itu sendiri, sehingga sulit untuk dikirimkan. Kriptografi kunci asimetris memiliki kelemahan dalam hal kecepatan. Enkripsi dan dekripsi dengan kriptografi kunci asimetris jauh lebih lambat dibandingkan dengan kriptografi kunci simetris (Sasikumar, K., & Nagarajan, S. 2024).

Oleh karena itu, penelitian ini bertujuan untuk mengatasi permasalahan tersebut dengan skema *hybrid cryptosystem*. Kombinasi algoritma Generalization of the ElGamal dan Martino Homomorphic Encryption akan digunakan untuk melakukan skema kriptosistem hibrid untuk keamanan data. Kriptografi hibrid dapat menjadi solusi untuk menggabungkan kelebihan dari kedua jenis kriptografi tersebut.

Hybrid cryptosystem mengintegrasikan metode enkripsi simetris dan asimetris untuk menciptakan cara yang aman dan efektif dari keunggulan masing-masing metode (Sasikumar, K., & Nagarajan, S. 2024). Dengan memanfaatkan kecepatan kriptografi kunci simetris untuk enkripsi dan dekripsi data aktual, memanfaatkan keamanan kriptografi kunci asimetris untuk mengamankan kunci simetris dan menggunakan kriptografi asimetris untuk pertukaran kunci simetris, hybrid cryptosystem menawarkan perlindungan yang lebih baik terhadap serangan kejahatan siber. Hal ini karena kunci publik dapat didistribusikan secara terbuka tanpa risiko, dan kunci simetris hanya didistribusikan dalam bentuk terenkripsi yang aman.

1.2 Rumusan Masalah

Sistem kriptografi asimetris dan simetris memiliki kelemahan masing-masing, yaitu kriptografi kunci asimetris yang tidak efisien untuk mengamankan dokumen berukuran besar dan kriptografi kunci simetris yang memiliki kelemahan dalam pengelolaan kunci. Skema Hybrid Cryptosystem dengan Martino Homomorphic Encryption dan Generalization of the ElGamal diharapkan dapat mengatasi permasalahan tersebut.

1.3 Batasan Masalah

Pada penelitian ini, peneliti membatasi ruang masalah yang akan diteliti. Batasan masalah pada penelitian ini adalah sebagai berikut:

1. Penelitian yang digunakan hanya pada implementasi Hybrid Cryptosystem dengan algoritma Martino Homomorphic Encryption dan Generalization of the ElGamal.
2. Implementasi algoritma menggunakan bahasa pemrograman Python.
3. Pengujian menggunakan data berupa file berekstensi pdf.
4. Kunci asimetris yang digunakan berupa bilangan prima.
5. Pembangkit kunci asimetris menggunakan fungsi *random* pada Python.
6. Pengujian bilangan prima yang telah dibangkitkan menggunakan algoritma *Fermat's Little Theorem*.
7. Pembangkit kunci simetris menggunakan algoritma pembangkit bilangan acak.
8. Karakter yang dienkripsi dibatasi hanya seluruh karakter pada ASCII.
9. Proses enkripsi dan dekripsi menggunakan file kunci yang berekstensi pdf.
10. Implementasi diterapkan pada aplikasi berbasis website.
11. Aplikasi yang dibangun menggunakan *backend* Flask.
12. Aplikasi yang dibangun menggunakan *frontend* HTML, CSS dan JavaScript.

1.4 Tujuan Penelitian

Penelitian ini bertujuan untuk mengembangkan algoritma yang dapat digunakan untuk membangun sebuah *hybrid cryptosystem* untuk mendistribusikan kunci enkripsi secara aman ke semua pihak yang berkepentingan dengan penggunaan enkripsi yang lebih ringan dan cepat sehingga mengurangi beban komputasi. Penelitian ini juga bertujuan untuk menguji apakah kedua algoritma yang digunakan dapat dikombinasikan untuk membangun sebuah *hybrid cryptosystem*.

1.5 Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini adalah sistem yang dibangun dapat dimanfaatkan untuk melakukan pengiriman data tanpa khawatir akan isi pesan, yang dapat meningkatkan rasa aman dengan menggunakan enkripsi yang lebih efisien.

1.6 Metodologi Penelitian

Adapun metodologi penelitian yang akan dilakukan dalam penelitian ini adalah sebagai berikut:

1. Studi Pustaka

Penelitian dimulai dengan melakukan studi literatur terhadap berbagai sumber ilmiah, termasuk jurnal, buku, dan publikasi lainnya yang terkait dengan kriptografi simetris, kriptografi asimetris, dan *hybrid cryptosystem*. Tujuan dari studi literatur ini adalah untuk memahami konsep dasar, sejarah perkembangan, serta keuntungan dan kelemahan dari masing-masing metode kriptografi.

2. Analisis dan Perancangan Sistem

Berdasarkan ruang lingkup penelitian, penulis melakukan analisa terhadap apa saja yang akan dibutuhkan dalam penelitian untuk segera dirancang dalam sebuah diagram alir (*flowchart*).

3. Implementasi Sistem

Pada tahap ini, membuat sebuah sistem dengan menggunakan Bahasa pemrograman python sesuai dengan diagram alir yang telah dirancang.

4. Pengujian Sistem

Pada tahap ini, sistem yang telah dirancang dilakukan uji coba untuk melakukan simulasi atau eksperimen untuk menguji kinerja *hybrid cryptosystem*.

5. Dokumentasi Sistem

Pada tahap ini, penelitian yang telah dilakukan, didokumentasikan mulai dari tahap analisa hingga tahap pengujian sistem yang kemudian akan ditulis dalam format laporan penelitian (skripsi).

1.7 Sistematika Penulisan

Sistematika penulisan dari skripsi ini terdiri dari 5 (lima) bab, yakni:

BAB I PENDAHULUAN

Bab ini memuat latar belakang peneltian, rumusan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, serta sistematika penulisan skripsi.

BAB II LANDASAN TEORI

Bab ini memuat latar belakang peneltian, rumusan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, serta sistematika penulisan skripsi.

BAB III ANALISIS DAN PERANCANGAN

Bab ini memuat latar belakang peneltian, rumusan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, serta sistematika penulisan skripsi.

BAB IV IMPLEMENTASI DAN PENGUJIAN SISTEM

Bab ini memuat tahapan implementasi dan pengujian sistem yang telah dirancang berdasarkan tahapan analisis dan perancangan.

BAB V KESIMPULAN DAN SARAN

Bab ini memuat kesimpulan dari peneltian yang dilakukan pada skripsi ini dan saran untuk peneltian selanjutnya.

BAB II

LANDASAN TEORI

2.1 Kriptosistem

Kriptosistem merupakan sistem yang menggunakan algoritma dan protokol untuk mengenkripsi dan mendekripsi informasi, memastikan keamanan data dari akses tidak sah. Kriptosistem terdiri dari tiga komponen utama: algoritma enkripsi, algoritma dekripsi, dan kunci kriptografi yang digunakan dalam proses enkripsi dan dekripsi. Kriptosistem modern dibagi menjadi kriptosistem kunci simetris, yang menggunakan satu kunci untuk enkripsi dan dekripsi, serta kriptosistem kunci asimetris, yang menggunakan kunci publik untuk enkripsi dan kunci privat untuk dekripsi. Kriptosistem dibagi menjadi beberapa jenis utama berdasarkan mekanisme dan tujuan enkripsinya, salah satunya yaitu kriptosistem hibrid.

2.2 Kriptosistem Hibrid

Kriptosistem hibrid adalah pendekatan dalam kriptografi yang menggabungkan kekuatan enkripsi kunci simetris dan asimetris untuk mencapai keamanan dan efisiensi yang lebih baik. Dalam kriptosistem ini, kunci asimetris digunakan untuk mengenkripsi kunci simetris sementara data atau pesan yang sebenarnya dienkripsi menggunakan kunci simetris. Hal ini karena enkripsi simetris lebih cepat dan efisien untuk data dalam jumlah besar, sedangkan enkripsi asimetris memberikan keamanan yang kuat untuk distribusi kunci. Kriptosistem hibrid mengatasi kelemahan masing-masing metode, kelemahan kunci simetris dalam distribusi aman dan kinerja yang lebih lambat dari kunci asimetris pada data besar (William Stallings, 2017). Dengan memadukan kedua metode, kriptosistem hibrid memberikan keseimbangan optimal antara keamanan dan efisiensi.

2.3 Enkripsi Kunci Simetris

Enkripsi kunci simetris adalah metode kriptografi di mana satu kunci yang sama digunakan untuk proses enkripsi dan dekripsi. Dalam sistem ini, baik pengirim maupun penerima harus memiliki kunci yang sama, dan keamanan pesan sangat bergantung pada kemampuan mereka untuk menjaga kerahasiaan kunci tersebut dari pihak ketiga. Enkripsi simetris sangat efisien dalam memproses data dalam jumlah besar, menjadikannya pilihan ideal untuk aplikasi yang memerlukan kecepatan tinggi (William Stallings, 2017). Metode ini memiliki sejarah panjang dalam kriptografi, dan meskipun aman dalam kondisi tertentu, enkripsi simetris menghadapi tantangan utama dalam distribusi kunci, yaitu bagaimana menyampaikan kunci secara aman kepada pihak yang berwenang tanpa terkompromi (David Khan, 1996).

2.4 Martino Homomorphic Encryption

Martino Homomorphic Encryption adalah teknik kriptografi simetris yang menggunakan transformasi matriks melalui pergeseran, rotasi, dan transposisi nilai ASCII dalam bentuk biner. Dengan kunci rahasia yang sama untuk enkripsi dan dekripsi, teknik ini menekankan "*avalanche effect*" di mana perubahan kecil pada kunci atau plaintext akan menyebabkan perubahan besar pada ciphertext, meningkatkan keamanan enkripsi.

2.5 Transformasi Matriks

Transformasi matriks adalah proses yang menggunakan matriks untuk mengubah bentuk, posisi, atau orientasi vektor dalam ruang. Dalam konteks ini, matriks bertindak sebagai operator linier yang, ketika diterapkan pada vektor, menghasilkan vektor baru yang telah mengalami transformasi tertentu, seperti skala (pengubahan ukuran), rotasi (pengubahan arah), refleksi (pencerminan), atau translasi (pergeseran posisi). Transformasi matriks memiliki peran penting dalam berbagai disiplin, termasuk grafika komputer, fisika, dan ilmu data, di mana mereka memungkinkan manipulasi objek dalam ruang dua atau tiga dimensi (Gilbert Strang, 2009). Transformasi matriks memungkinkan kita menerjemahkan konsep matematika ke dalam bentuk yang dapat dihitung dan divisualisasikan dengan efisiensi tinggi.

2.6 Avalanche Effect

Avalanche effect adalah sifat dalam kriptografi di mana perubahan kecil pada input, seperti perubahan satu bit, akan menghasilkan perubahan signifikan pada output, dengan sekitar 50% dari bit dalam hasil enkripsi berubah. Efek ini merupakan indikator bahwa algoritma kriptografi, memiliki ketahanan tinggi terhadap analisis diferensial dan serangan. Efek avalanche yang kuat adalah ciri penting dari cipher yang aman, karena hal ini membuat sulit bagi penyerang untuk memprediksi bagaimana perubahan kecil pada pesan akan mempengaruhi hasil akhir (Bruce Schneier, 1996). Efek ini sangat penting dalam menjaga integritas dan kerahasiaan data, serta dalam memastikan bahwa sistem kriptografi tidak memberikan petunjuk apa pun tentang input aslinya dari output yang dihasilkan.

2.7 Enkripsi Kunci Asimetris

Enkripsi kunci asimetris adalah teknik kriptografi yang menggunakan pasangan kunci berbeda untuk proses enkripsi dan dekripsi: kunci publik dan kunci privat. Kunci publik dapat dibagikan secara terbuka dan digunakan untuk mengenkripsi data, sedangkan kunci privat disimpan secara rahasia dan digunakan untuk mendekripsi data. Keunggulan utama enkripsi kunci asimetris adalah kemampuannya dalam memfasilitasi komunikasi yang aman tanpa perlu membagikan kunci rahasia terlebih dahulu, yang memecahkan masalah distribusi kunci dalam sistem simetris (William Stallings, 2017). Metode ini didasarkan pada masalah matematika yang kompleks, seperti faktorisasi bilangan besar (pada RSA) atau logaritma diskrit (pada algoritma Diffie-Hellman dan ElGamal), yang sulit dipecahkan oleh komputer modern (Alfred J. Menezes, 2017).

2.8 Generalization of the ElGamal

Generalization of the ElGamal merupakan varian dari ElGamal yang melibatkan proses enkripsi bergantung pada modulus eksponensial. Eksponensial modulus merupakan hasil dari operasi akar primitif yang diterapkan dua kali selama proses enkripsi, dengan mempertimbangkan angka-angka yang berbeda dan kunci enkripsi rahasia. Algoritma ini juga merupakan algoritma asimetris yang memiliki pasangan kunci publik dan privat untuk enkripsi dan dekripsi.

2.9 Fermat's Little Theorem

Fermat's Little Theorem adalah salah satu hasil fundamental dalam teori bilangan yang menyatakan bahwa jika p adalah bilangan prima dan a adalah bilangan bulat yang tidak habis dibagi oleh p , maka $a^{p-1} \equiv 1 \pmod{p}$, Dimana a adalah bilangan bulat yang memenuhi $1 \leq a < p$. Teorema ini sangat berguna dalam kriptografi modern, karena dapat digunakan untuk mempercepat perhitungan eksponen modular (David M. Burton, 2007). Teorema ini juga membentuk dasar dari uji primalitas, seperti uji Fermat, yang digunakan untuk menentukan apakah suatu bilangan adalah prima. *Fermat's Little Theorem* menyediakan hubungan penting antara sifat aritmatika bilangan prima dan eksponensiasi modular, yang menjadi salah satu pondasi utama dalam sistem kriptografi kunci publik (Neal Koblitz, 1987). Contoh percobaan Fermat's Little Theorem menggunakan sebuah bilangan p dapat dilihat pada tabel 2.1.

Tabel 2.1. Perhitungan Fermat's Little Theorem

$$p = 11$$

a	$a^{p-1} \equiv 1 \pmod{p}$
1	$1^{10} \bmod 11 = 1$
2	$2^{10} \bmod 11 = 1$
3	$3^{10} \bmod 11 = 1$
4	$4^{10} \bmod 11 = 1$
5	$5^{10} \bmod 11 = 1$
6	$6^{10} \bmod 11 = 1$
7	$7^{10} \bmod 11 = 1$
8	$8^{10} \bmod 11 = 1$
9	$9^{10} \bmod 11 = 1$
10	$10^{10} \bmod 11 = 1$

Berdasarkan perhitungan *Fermat's Little Theorem* pada tabel di atas, dapat dilihat bahwa $a^{11-1} \equiv 1 \pmod{11}$, untuk semua a dari 1 hingga 10 yang relatif prima terhadap 11. Maka hasil ini membuktikan bahwa 11 adalah bilangan prima. ($p = 11$ bilangan prima).

2.10 Akar Primitif

Akar primitif dalam teori bilangan adalah bilangan bulat g sedemikian rupa sehingga bilangan tersebut dapat menghasilkan semua bilangan bulat lain yang relatif prima terhadap modulus n melalui perpangkatan dalam aritmetika modular. Dengan kata lain, g adalah akar primitif modulo n jika setiap bilangan bulat yang relatif prima terhadap n dapat direpresentasikan sebagai $g^k \pmod{n}$ untuk suatu bilangan bulat k . Akar primitif sangat penting dalam teori bilangan karena konsep ini digunakan dalam struktur grup siklik dan juga dalam algoritma kriptografi, di mana kekuatan akar primitif untuk menghasilkan semua nilai modular memungkinkan pembangkitan kunci yang aman dan acak (Kenneth H. Rosen, 2011). Akar primitif memberikan dasar bagi eksponensiasi modular, yang merupakan komponen penting dalam berbagai sistem keamanan digital. Sebagai contoh, untuk bilangan prima $p = 7$, bilangan $g = 3$, dapat dicoba sebagai kandidat akar primitive seperti pada tabel 2.2.

Tabel 2.2. Pengujian Akar Primitif

k	3^k	$3^k \pmod{7}$
1	$3^1 = 3$	$3 \pmod{7} = 3$
2	$3^2 = 9$	$9 \pmod{7} = 2$
3	$3^3 = 27$	$27 \pmod{7} = 6$
4	$3^4 = 81$	$81 \pmod{7} = 4$
5	$3^5 = 243$	$243 \pmod{7} = 5$
6	$3^6 = 729$	$729 \pmod{7} = 1$

Berdasarkan tabel di atas, akar primitif dari suatu bilangan prima p adalah bilangan g yang ketika dipangkatkan secara berurutan dari 1 hingga $p - 1$ dalam modulus p , menghasilkan seluruh bilangan dari 1 hingga $p - 1$. Urutan ini mencakup seluruh bilangan dari 1 hingga 6 tanpa pengulangan, menunjukkan bahwa 3 menghasilkan semua bilangan dari 1 hingga $p - 1$ dalam modulus 7. Dengan demikian, 3 memenuhi syarat sebagai akar primitif dari 7 karena menghasilkan seluruh bilangan dalam interval tersebut.

2.11 Modulo Eksponensial

Modulo eksponensial adalah operasi matematika yang melibatkan penghitungan hasil eksponen dalam suatu modulus tertentu, yang ditulis sebagai $a^b \pmod n$. Modulo eksponensial memungkinkan penghitungan bilangan besar secara efisien dengan cara mengeliminasi bilangan yang berulang dalam modulus tertentu, sehingga sangat efisien meski ukuran eksponennya besar (Neal Koblitz, 1987). Penggunaan metode pengurangan ini penting dalam aplikasi yang membutuhkan keamanan tinggi, karena hasilnya sulit diprediksi tanpa mengetahui basis dan modulus yang tepat (Kenneth H. Rosen, 2011).

2.12 Inversi Modulo

Inversi modulo adalah suatu konsep dalam teori bilangan yang mengacu pada bilangan, yang jika dikalikan dengan bilangan lain dalam modulus tertentu, menghasilkan nilai 1 sebagai hasil akhirnya. Inversi modulo dari bilangan a modulo n adalah bilangan b sedemikian rupa sehingga $a \times b \equiv 1 \pmod n$, dengan syarat bahwa a dan n saling prima (memiliki faktor persekutuan terbesar 1) (Rosen, 2011). Untuk menemukan inversi modulo, metode seperti Algoritma Euclidean yang diperluas (Extended Euclidean Algorithm) sering digunakan karena dapat menangani bilangan besar dan efisien dalam aplikasi kriptografi.

2.13 Extend Euclidean Algorithm

Algoritma Euclidean yang Diperluas (Extended Euclidean Algorithm) adalah perpanjangan dari Algoritma Euclidean yang tidak hanya menemukan pembagi terbesar (GCD) dari dua bilangan bulat a dan b , tetapi juga koefisien bilangan bulat x dan y yang memenuhi persamaan identitas Bézout: $ax + by = gcd(a, b)$. Algoritma ini sangat penting dalam teori bilangan dan aplikasi kriptografi, terutama dalam menemukan invers modulo, yang merupakan syarat penting untuk operasi dalam sistem kriptografi. Extended Euclidean Algorithm memberikan cara efisien untuk menghitung invers modulo dengan memanfaatkan sifat identitas Bézout, karena Solusi x dari persamaan tersebut memberikan invers dari a modulo b jika a dan b relatif prima (David M. Burton, 2007).

2.14 Contoh Kasus Penerapan Hybrid Cryptosystem dengan Algoritma Martino Homomorphic Encryption dan Generalization of the ElGamal

Dalam contoh kasus, anggap bahwa pesan "Hello World @01" ingin dikirimkan kepada seorang penerima pesan. Pengiriman pesan dilakukan dengan melalui serangkaian proses. Pertama-tama, membuat kunci simetris sementara (misalnya, kunci simetris: "6412") untuk mengenkripsi pesan "Hello World @01". Pada kasus ini, pesan dienkripsi dengan menggunakan Martino Homomorphic Encryption. Proses ini menghasilkan teks terenkripsi yang tidak dapat dibaca tanpa kunci simetris yang sesuai.

Selanjutnya, untuk mengirim kunci simetris ini dengan aman, kunci simetris dienkripsi menggunakan kunci publik asimetris milik penerima pesan, misalnya pada kasus ini dengan menggunakan Generalization of the ElGamal. Penerima pesan kemudian menerima kedua bagian pesan: teks terenkripsi dari pesan "Hello World @01" dan kunci simetris yang telah dienkripsi. Dengan kunci privat Generalization of the ElGamal-nya, penerima pesan mendekripsi kunci simetris tersebut, dan kemudian menggunakan kunci itu untuk mendekripsi pesan "Hello World @01". Pendekatan ini menggabungkan efisiensi enkripsi simetris untuk pesan dan keamanan enkripsi asimetris untuk pertukaran kunci, memberikan keamanan lebih tanpa mengorbankan kecepatan.

2.14.1 Pembangkit kunci (key generation) Generalization of the ElGamal

Penerima pesan memulai dengan membuat sepasang kunci: kunci publik dan kunci privat. Kunci publik dirancang untuk dapat dibagikan secara bebas dan tidak mengandung informasi rahasia, sehingga dapat diberikan kepada siapa saja, termasuk kepada pengirim pesan, sedangkan kunci privat disimpan rahasia hanya untuk penerima pesan. Kunci publik dan kunci privat ini berpasangan dan dibuat menggunakan algoritma kriptografi asimetris, pada contoh kasus ini akan digunakan algoritma pembangkit kunci Generalization of the ElGamal. Pertama, pilih bilangan prima (p) yang lebih besar dari 255, ($p > 255$). Kedua, pilih bilangan akar primitif g modulo p . Ketiga, pilih kunci dekripsi privat (x), sehingga $1 < x < p - 1$. Terakhir, hitung $a \equiv g^x \pmod{p}$. Kunci publik: (p, g, a) dan kunci privat: x . Contoh:

Misalkan $p = 257$; $g = 17$; dan $x = 11$, masing-masing menjadi bilangan prima, akar primitif dan kunci dekripsi privat. Hitunglah $a \equiv g^x \pmod{p}$.

$$a \equiv 17^{11} \pmod{257}$$

$$a = 223$$

Public encryption keys = $(p, g, a) = (257, 17, 223)$

Private decryption key = $(x) = (11)$

1. Proses Enkripsi

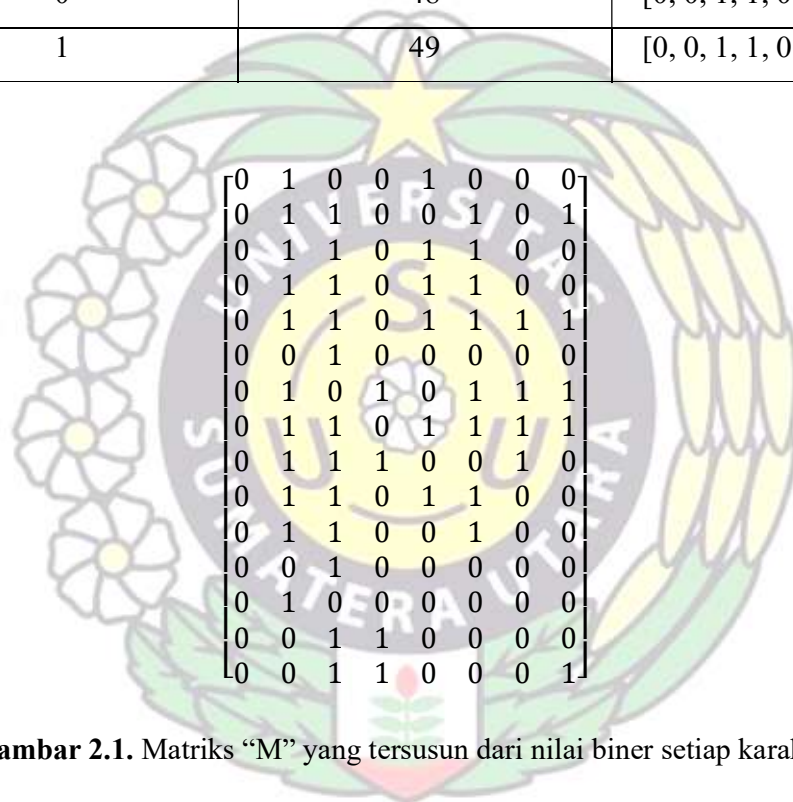
Berikut merupakan cara kerja Martino Homomorphic Encryption dalam mengenkripsi suatu pesan. Contoh pesan (*plaintext*) atau (P) yang akan digunakan untuk dienkripsi yaitu “Hello World @01”. Lalu, contoh kunci simetris (K) yang akan digunakan untuk mengenkripsi suatu pesan (P) harus terdiri dari 4 digit, yaitu “6412”. Contoh:

Pertama, masukkan *plaintext* (P) dan kunci simetris (K) yang dibutuhkan dengan ukuran 4 digit. Lalu setiap karakter dalam (P) akan diubah menjadi nilai ASCII yang sesuai, kemudian dikonversikan ke bentuk biner. Jika panjang biner kurang dari 8 bit, maka nol ditambahkan di depan hingga mencapai 8 bit seperti pada tabel 2.3. Selanjutnya, Matriks 'M' akan disusun dari nilai biner setiap karakter tersebut seperti pada gambar 2.1.

Tabel 2.3. Bentuk ASCII dan biner dari setiap karakter

Input	ASCII	Biner
H	72	[0, 1, 0, 0, 1, 0, 0, 0]
E	69	[0, 1, 1, 0, 0, 1, 0, 1]
L	76	[0, 1, 1, 0, 1, 1, 0, 0]
L	76	[0, 1, 1, 0, 1, 1, 0, 0]
O	79	[0, 1, 1, 0, 1, 1, 1, 1]

	32	[0, 0, 1, 0, 0, 0, 0, 0]
W	87	[0, 1, 0, 1, 0, 1, 1, 1]
O	79	[0, 1, 1, 0, 1, 1, 1, 1]
R	82	[0, 1, 1, 1, 0, 0, 1, 0]
L	76	[0, 1, 1, 0, 1, 1, 0, 0]
D	68	[0, 1, 1, 0, 0, 1, 0, 0]
	32	[0, 0, 1, 0, 0, 0, 0, 0]
@	64	[0, 1, 0, 0, 0, 0, 0, 0]
0	48	[0, 0, 1, 1, 0, 0, 0, 0]
1	49	[0, 0, 1, 1, 0, 0, 0, 1]



Gambar 2.1. Matriks “M” yang tersusun dari nilai biner setiap karakter

Mulai fase rotasi berdasarkan kunci. Jika digit pertama pada kunci adalah 'GENAP,' operasi 'Putar Pergeseran Atas' atau 'Rotate Shift Up' (RSU) pada matriks 'M' akan dilakukan sebanyak digit kedua pada kunci (K). Jika digit pertama adalah 'GANJIL,' operasi 'Putar Pergeseran Bawah' atau 'Rotate Shift Down' (RSD) pada matriks 'M' akan dilakukan sesuai angka kedua pada kunci. Hasilnya disimpan dalam matriks 'M1'.

Pada kasus ini, dapat dilihat bahwa digit pertama pada kunci K ($K = "6412"$) merupakan angka 6 (angka genap). Oleh karena itu matriks harus diputar pergeseran atas (RSU) sebanyak digit kedua. Digit kedua merupakan angka 4, maka matriks "M" harus diputar pergeseran atas (RSU) sebanyak empat kali. Hasil setelah rotasi dapat dilihat pada gambar 2.2.



0	1	1	0	1	1	1	1
0	0	1	0	0	0	0	0
0	1	0	1	0	1	1	1
0	1	1	0	1	1	1	1
0	1	1	1	0	0	1	0
0	1	1	0	1	1	0	0
0	1	1	0	0	1	0	0
0	0	1	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	0	1	1	0	0	0	1
0	1	0	0	1	0	0	0
0	1	1	0	0	1	0	1
0	1	1	0	1	1	0	0
0	1	1	0	1	1	0	0

Gambar 2.2. Matriks 'M1' hasil dari rotasi matriks 'M'

Jika digit ketiga pada kunci adalah 'GENAP,' operasi 'Putar Pergeseran Kiri' atau 'Rotate Shift Left' (RSL) pada matriks 'M1' akan dilakukan sebanyak digit keempat pada kunci (K). Jika digit ketiga adalah 'GANJIL,' operasi 'Putar Pergeseran Kanan' atau 'Rotate Shift Right' (RSR) pada matriks 'M1' akan dilakukan sesuai angka keempat pada kunci. Hasilnya disimpan dalam matriks 'M2'.

Pada kasus ini, dapat dilihat bahwa digit ketiga pada kunci K (K = "6412") merupakan angka 1 (angka ganjil). Oleh karena itu matriks harus diputar pergeseran kanan (RSR) sebanyak digit keempat. Digit keempat merupakan angka 2, maka matriks "M1" harus diputar pergeseran kanan (RSR) sebanyak dua kali. Hasil setelah rotasi dapat dilihat pada gambar 2.3.

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Gambar 2.3. Matriks 'M2' hasil dari rotasi matriks 'M1'

Transpos pada matriks 'M2' dilakukan. Hasilnya disimpan pada matriks 'MT' seperti pada gambar 2.4. Kemudian, matriks 'MT' diratakan dengan mengubahnya menjadi array linier tunggal 'T' dengan membaca baris per baris, seperti pada gambar 2.5.

$$\begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Gambar 2.4. Matriks 'MT' hasil dari transpos matriks 'M2'

$$b = 16^{15} \bmod 257$$

$$b = 241$$

$$C = [[54[223^{8 \cdot 15} \bmod 257]] \bmod 257], [[52[223^{8 \cdot 15} \bmod 257]] \bmod 257], \\ [[49[223^{8 \cdot 15} \bmod 257]] \bmod 257], [[50[223^{8 \cdot 15} \bmod 257]] \bmod 257]]$$

$$C = [93, 61, 13, 29]$$

$$b = 241, c = [93, 61, 13, 29]$$

2. Proses Dekripsi

Proses ketiga adalah dekripsi pesan. Pihak penerima berwenang mendekripsi *ciphertext*, kunci simetris yang dienkripsi, menggunakan kunci asimetris dekripsi privat x . Berikut langkah-langkahnya. Kembalikan nilai m menggunakan persamaan berikut: $m \equiv c \cdot (\overline{b^x}) \pmod{m}$, di mana $(\overline{b^x})$ adalah invers dari b^x modulo m . Contoh:

1. Nilai b dan *ciphertext* telah diterima dari pengirim

$$(b = 241, c = [93, 61, 13, 29])$$

2. Hitung z

$$z = 241^{11} \bmod 257 \\ = 16$$

3. Hitung w

$$w = 16^{-1} \bmod 257 \\ = 241$$

4. Hitung m

$$m = [[93 \times 241 \bmod 257], [61 \times 241 \bmod 257], [13 \times 241 \bmod 257], \\ [29 \times 241 \bmod 257]] \\ = ['6', '4', '1', '2'] \rightarrow 6412$$

Jadi, dari nilai yang didekripsi: [93, 61, 13, 29], diperoleh: [6, 4, 1, 2] sebagai kunci simetris untuk mendekripsi pesan. Kunci simetris ini dapat digunakan untuk mendekripsi pesan yang terenkripsi. Berikut adalah cara kerja Martino Homomorphic Encryption dalam mendekripsi *ciphertext*.

Proses dimulai dengan memisahkan *ciphertext* 'C', pada gambar 2.7 menggunakan pemisah ('#') menjadi elemen-elemen 't'. Untuk setiap elemen dalam 't', baca nilainya kecuali digit terakhir, lalu simpan nilai tersebut sebanyak digit terakhir dalam elemen tersebut ke dalam 'T' seperti pada gambar 2.8. Selanjutnya, baca secara terus-menerus setiap 'q' digit biner dari 'T' dan bentuk sebagai baris dalam matriks 'MT', di mana nilai 'q' adalah panjang 'T'/8. Transposisikan matriks 'MT' dan simpan hasilnya ke dalam 'M2'.

11#10#31#100#11#10#21#60#11#10#11#170#11#10#51#10#11#20#61#10
 #51#10#21#10#31#20#11#10#11#40#21#40#11#20#11#10#11#50#11#10
 #31#10#21#10#21#50#31

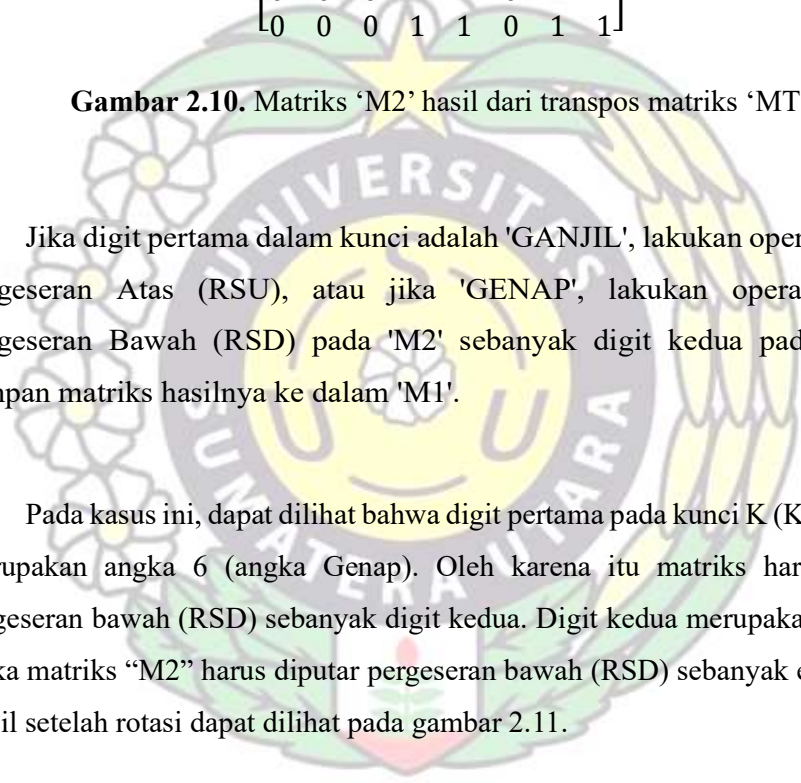
Gambar 2.7. *Ciphertext*

$T \leftarrow [1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1,$
 $0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1,$
 $1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,$
 $1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1]$

Gambar 2.8. Array linier tunggal 'T'

1	0	1	1	1	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	1	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	1	1	1	0	1	0	0	1	1	1	1
1	1	0	1	1	1	1	1	0	1	1	0	1	1	1
0	0	1	0	1	0	0	0	0	1	1	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	1	0	1	1
1	0	1	1	0	1	1	0	0	0	0	0	1	1	1

Gambar 2.9. Matriks 'MT'

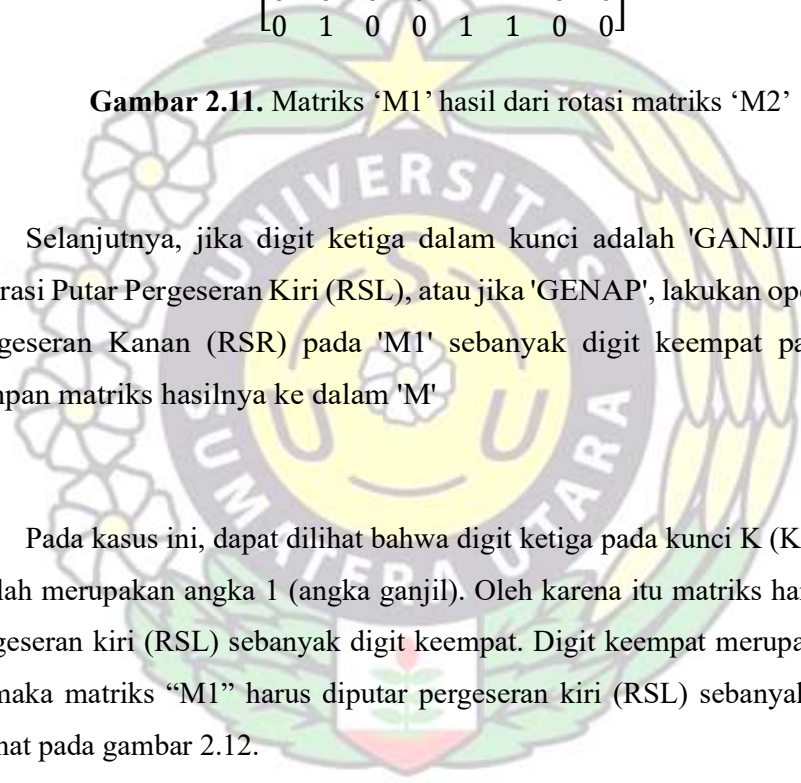


1	1	0	1	1	0	1	1
0	0	0	0	1	0	0	0
1	1	0	1	0	1	0	1
1	1	0	1	1	0	1	1
1	0	0	1	1	1	0	0
0	0	0	1	1	0	1	1
0	0	0	1	1	0	0	1
0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0
0	0	0	0	1	1	0	0
0	1	0	0	1	1	0	0
0	0	0	1	0	0	1	0
0	1	0	1	1	0	0	1
0	0	0	1	1	0	1	1
0	0	0	1	1	0	1	1

Gambar 2.10. Matriks 'M2' hasil dari transpos matriks 'MT'

Jika digit pertama dalam kunci adalah 'GANJIL', lakukan operasi Putar Pergeseran Atas (RSU), atau jika 'GENAP', lakukan operasi Putar Pergeseran Bawah (RSD) pada 'M2' sebanyak digit kedua pada kunci. Simpan matriks hasilnya ke dalam 'M1'.

Pada kasus ini, dapat dilihat bahwa digit pertama pada kunci K (K = "6412") merupakan angka 6 (angka Genap). Oleh karena itu matriks harus diputar pergeseran bawah (RSD) sebanyak digit kedua. Digit kedua merupakan angka 4, maka matriks "M2" harus diputar pergeseran bawah (RSD) sebanyak empat kali. Hasil setelah rotasi dapat dilihat pada gambar 2.11.



0	0	0	1	0	0	1	0
0	1	0	1	1	0	0	1
0	0	0	1	1	0	1	1
0	0	0	1	1	0	1	1
1	1	0	1	1	0	1	1
0	0	0	0	1	0	0	0
1	1	0	1	0	1	0	1
1	1	0	1	1	0	1	1
1	0	0	1	1	1	0	0
0	0	0	1	1	0	1	1
0	0	0	1	1	0	0	1
0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0
0	0	0	0	1	1	0	0
0	1	0	0	1	1	0	0

Gambar 2.11. Matriks 'M1' hasil dari rotasi matriks 'M2'

Selanjutnya, jika digit ketiga dalam kunci adalah 'GANJIL', lakukan operasi Putar Pergeseran Kiri (RSL), atau jika 'GENAP', lakukan operasi Putar Pergeseran Kanan (RSR) pada 'M1' sebanyak digit keempat pada kunci. Simpan matriks hasilnya ke dalam 'M'

Pada kasus ini, dapat dilihat bahwa digit ketiga pada kunci K (K = "6412") adalah merupakan angka 1 (angka ganjil). Oleh karena itu matriks harus diputar pergeseran kiri (RSL) sebanyak digit keempat. Digit keempat merupakan angka 2, maka matriks "M1" harus diputar pergeseran kiri (RSL) sebanyak dua kali. dilihat pada gambar 2.12.

L	76	[0, 1, 1, 0, 1, 1, 0, 0]
L	76	[0, 1, 1, 0, 1, 1, 0, 0]
O	79	[0, 1, 1, 0, 1, 1, 1, 1]
	32	[0, 0, 1, 0, 0, 0, 0, 0]
W	87	[0, 1, 0, 1, 0, 1, 1, 1]
O	79	[0, 1, 1, 0, 1, 1, 1, 1]
R	82	[0, 1, 1, 1, 0, 0, 1, 0]
L	76	[0, 1, 1, 0, 1, 1, 0, 0]
D	68	[0, 1, 1, 0, 0, 1, 0, 0]
	32	[0, 0, 1, 0, 0, 0, 0, 0]
@	64	[0, 1, 0, 0, 0, 0, 0, 0]
0	48	[0, 0, 1, 1, 0, 0, 0, 0]
1	49	[0, 0, 1, 1, 0, 0, 0, 1]

2.15 Penelitian yang Relevan

1. Penelitian berjudul "Novel secure data protection scheme using Martino homomorphic encryption" berfokus pada perlindungan data *plaintext* melalui metode enkripsi homomorfik. Data yang digunakan dalam penelitian ini adalah *plaintext* yang diubah menjadi nilai ASCII biner. Konversi ke bentuk biner ini memudahkan penerapan transformasi matriks, pergeseran, rotasi, dan transposisi, yang merupakan elemen utama dalam proses enkripsi untuk meningkatkan keamanan data. Dalam hal performa, algoritma ini dirancang untuk memiliki waktu respons yang efisien.

Selain performa, algoritma ini menampilkan *avalanche effect* yang tinggi, di mana perubahan kecil pada input menghasilkan perubahan signifikan pada output atau teks sandi, meningkatkan tingkat keamanan. Hasil penelitian ini menegaskan bahwa metode enkripsi yang diusulkan tidak hanya lebih aman dibandingkan teknik tradisional tetapi juga memiliki performa yang memadai untuk diterapkan pada lingkungan komputasi awan dan perangkat IoT yang memerlukan keamanan data tinggi dan efisiensi waktu.

2. Penelitian berjudul “Generalization of the ElGamal public-key cryptosystem” mengembangkan versi baru dari sistem ElGamal dengan menambahkan proses eksponensiasi modular ganda, yang didasarkan pada kunci enkripsi rahasia. Penggunaan kunci publik dalam skema ini mencakup parameter seperti G , p , g , dan g^x untuk memastikan komunikasi yang aman. Selain itu, proses enkripsi yang lebih kompleks memungkinkan sistem baru ini cocok untuk pesan kecil, seperti distribusi kunci simetris, karena efisiensinya untuk data pendek.



BAB III

ANALISIS DAN PERANCANGAN

3.1 Analisis Sistem

Tahapan analisis sistem adalah langkah untuk mengidentifikasi keperluan pembangunan sistem agar komponen-komponen dapat berinteraksi dengan benar. Hal ini bertujuan agar sistem dapat bekerja dengan baik, sesuai dengan yang diharapkan. Analisis sistem dibagi menjadi tiga jenis utama, antara lain analisis masalah, analisis kebutuhan, dan analisis proses.

3.1.1 Analisis masalah

Analisis masalah merupakan tahapan untuk mengidentifikasi permasalahan yang dibahas dan dikaji pada penelitian. Proses ini diperlukan untuk memecahkan permasalahan tersebut. Fokus permasalahan pada penelitian ini adalah kriptografi kunci simetris yang memiliki kelemahan dalam pengelolaan kunci dan kriptografi kunci asimetris yang tidak efisien untuk mengamankan dokumen berukuran besar.



Gambar 3.1. Diagram Ishikawa

Sebuah diagram bernama diagram *Ishikawa* dapat mempermudah proses identifikasi penyebab masalah dengan membaginya menjadi beberapa klasifikasi permasalahan. Diagram *Ishikawa* terdapat pada **Gambar 3.1**. Diagram menunjukkan penyebab terjadinya masalah dibagi menjadi empat kategori, yaitu material, metode, manusia, dan mesin. Perincian setiap kategori digambarkan melalui panah yang mengarah ke panah kategori masalah. Algoritma *Hybrid Cryptosystem* dengan Martino Homomorphic Encryption dan Generalization of the ElGamal menjadi algoritma kriptosistem hibrida yang dipilih dalam mengatasi kelemahan masing-masing kriptografi.

Pada kategori Material, masalah utama meliputi ukuran dokumen yang besar, yang menyulitkan proses enkripsi dan dekripsi, serta penyimpanan kunci yang rentan terhadap kebocoran atau serangan keamanan. Dalam kategori Manusia, keterbatasan dalam memahami dan menangani algoritma asimetris menjadi kendala, terutama dalam penerapan sistem yang kompleks.

Kategori Metode mencakup kurangnya pengukuran efisiensi dalam penggunaan algoritma kriptografi serta protokol distribusi kunci yang tidak efisien, yang dapat menghambat kinerja sistem keamanan. Terakhir, pada kategori Mesin, keterbatasan sistem dalam memproses dokumen berukuran besar menjadi kendala utama, karena baik enkripsi simetris maupun asimetris memiliki kelemahan dalam menangani data dalam jumlah besar secara efisien.

3.1.2 Analisis kebutuhan fungsional dan non-fungsional

Analisis kebutuhan merupakan tahapan yang dilakukan untuk mengidentifikasi dan mengetahui hal-hal yang diperlukan pada pembangunan sistem yang akan dirancang. Proses ini diperlukan agar sistem yang dibuat dapat berjalan dengan benar. Analisis kebutuhan terdiri dari dua bagian, yaitu analisis kebutuhan fungsional dan analisis kebutuhan non-fungsional. Kebutuhan fungsional merupakan kebutuhan yang mencakup proses yang dilaksanakan pada sistem. Kebutuhan non-fungsional merupakan fitur-fitur atau fungsi-fungsi suatu sistem yang bersifat sebagai nilai tambahan agar sistem dapat berjalan dengan lebih baik. Pada penelitian ini, kebutuhan fungsional yang dibutuhkan antara lain:

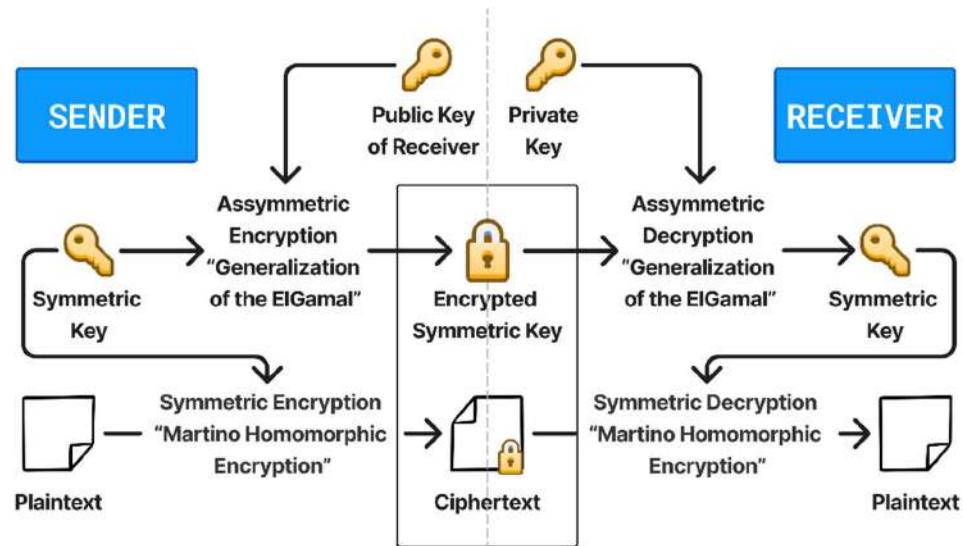
1. Sistem mampu melaksanakan pembangkitan kunci privat dan kunci publik.
2. Sistem mampu memberikan keluaran kunci privat dan kunci publik pada proses pembangkitan kunci publik.
3. Sistem mampu melaksanakan enkripsi pesan menjadi *ciphertext*, dan dekripsi *ciphertext* kembali menjadi pesan asli (*plaintext*) dengan menerapkan algoritma Martino Homomorphic Encryption dan Generalization of the ElGamal.
4. Sistem mampu menghasilkan hasil enkripsi dan dekripsi yang konsisten, artinya tidak ada perbedaan dalam hasil atau informasi yang hilang ketika data tersebut diproses di seluruh siklus enkripsi dan dekripsi.

Kebutuhan non-fungsional pada sistem antara lain:

1. *User Friendly*
Sistem yang dibuat menampilkan tampilan antarmuka yang relatif sederhana.
2. *Usability*
Sistem memiliki kinerja dan fungsi yang tetap terjaga, seperti memberikan informasi sukses tidaknya sebuah proses yang dijalankan oleh penggunaanya
3. *Performance*
Sistem dibangun agar dapat berjalan dalam waktu yang relatif cepat tanpa mengurangi kualitas baik pada proses maupun keluaran sistem.
4. *Compatibility*
Sistem menerima dokumen berformat *pdf* yang bisa diunggah oleh pengguna untuk diproses, dan memberikan keluaran berupa dokumen dengan format *pdf*.

3.1.3 Analisis proses

Analisis proses adalah penjelasan cara kerja sistem yang dibuat. Analisis proses biasanya direpresentasikan menggunakan diagram umum. Diagram umum merupakan diagram perancangan sistem yang menggambarkan alur, proses dan interaksi antar komponen dalam sistem. Diagram umum terdapat pada **Gambar 3.2.** berikut.



Gambar 3.2. Diagram Umum Sistem

Diagram ini menggambarkan sistem kriptografi *hybrid*, yang menggabungkan kriptografi simetris dan asimetris untuk meningkatkan keamanan pertukaran informasi antara pengirim (*sender*) dan penerima (*receiver*). Proses ini terdiri dari dua tahap utama. Tahap pertama yaitu Enkripsi yang dilakukan oleh pengirim dan dekripsi yang dilakukan oleh penerima.

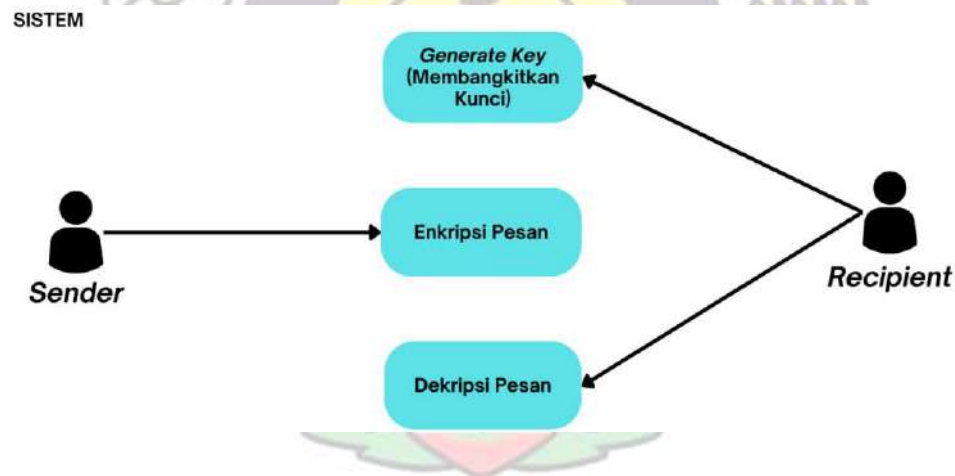
Pada tahap enkripsi, pengirim terlebih dahulu menghasilkan kunci simetris yang akan digunakan untuk mengenkripsi data. Setelah kunci simetris dibuat, pengirim mengenkripsi *plaintext* menggunakan algoritma Martino Homomorphic Encryption, yang merupakan salah satu metode enkripsi simetris. Hasil dari proses ini adalah *ciphertext*, yaitu data yang sudah terenkripsi dan tidak dapat dibaca tanpa kunci dekripsi yang sesuai. Pengirim mengenkripsi kunci tersebut menggunakan *public key* milik penerima dengan algoritma Generalization of the ElGamal. Hasil dari enkripsi ini adalah *encrypted symmetric key* yang dikirim bersama *ciphertext* kepada penerima.

Setelah menerima *ciphertext* dan *encrypted symmetric key*, penerima memulai proses dekripsi. Langkah pertama yang dilakukan adalah mendekripsi *encrypted symmetric key* dengan menggunakan *private key* miliknya melalui algoritma Generalization of the ElGamal. Dengan cara ini, penerima dapat memperoleh kembali kunci simetris asli yang sebelumnya digunakan oleh pengirim untuk mengenkripsi *plaintext*.

Setelah mendapatkan kunci simetris, penerima kemudian menggunakan kunci tersebut untuk mendekripsi *ciphertext* dengan algoritma Martino Homomorphic Encryption, sehingga *plaintext* asli dapat dipulihkan. Kriptografi *hybrid* memastikan bahwa informasi tetap aman selama proses transmisi. Kriptografi simetris digunakan untuk mengenkripsi data karena kecepatannya, sedangkan kriptografi asimetris digunakan untuk mengamankan kunci simetris sebelum dikirimkan kepada penerima.

3.2 Use Case Diagram

Use Case Diagram merupakan sebuah diagram yang mempresentasikan hubungan interaksi dan perilaku sistem dengan aktor/pengguna. Diagram ini menunjukkan bagaimana aktor menggunakan sistem untuk mencapai tujuan tertentu melalui serangkaian tindakan. Use Case Diagram pada penelitian ini terdapat pada gambar berikut.



Gambar 3.3. *Use Case Diagram*

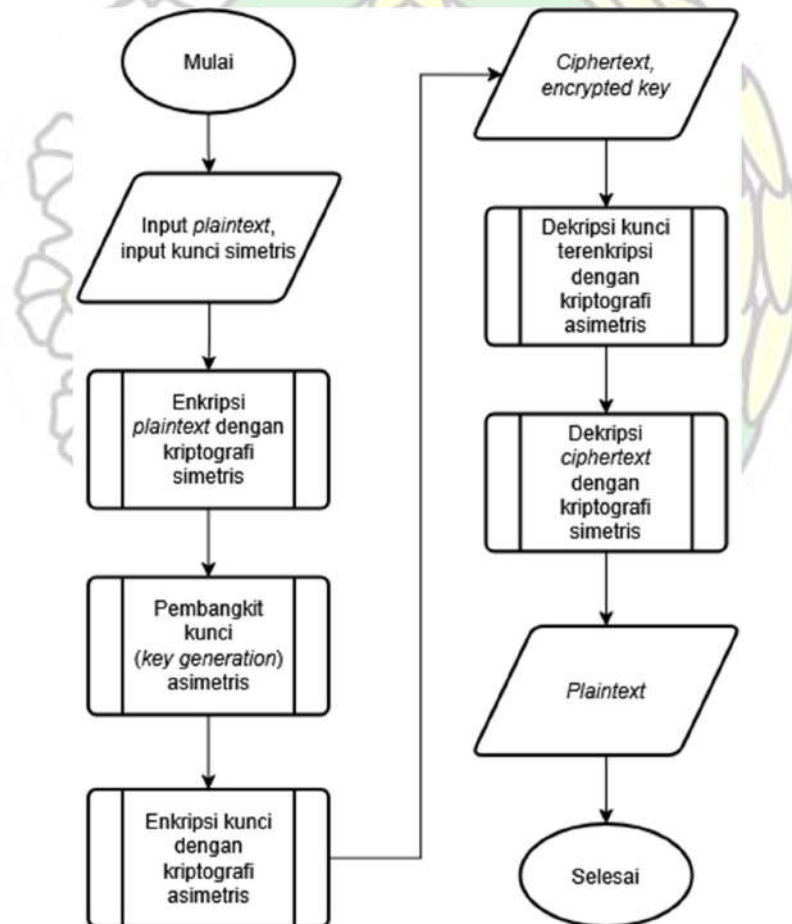
Pada **Gambar 3.3.**, *Use Case Diagram* menunjukkan interaksi antara dua aktor utama, *Sender* dan *Recipient*. *Recipient* terlebih dahulu melakukan proses *Generate Key* (Membangkitkan Kunci) untuk menghasilkan kunci yang akan digunakan dalam enkripsi dan dekripsi pesan. *Sender* kemudian mengenkripsi pesan menggunakan kunci tersebut dalam proses Enkripsi Pesan sebelum mengirimkannya ke *Recipient*. Setelah menerima pesan terenkripsi, *Recipient* melakukan proses Dekripsi Pesan untuk mendapatkan kembali pesan asli yang dikirim oleh *Sender*.

3.3 Flowchart Sistem

Flowchart merupakan representasi dari urutan proses yang digambarkan dalam bentuk berbagai simbol untuk menunjukkan hubungan antara suatu proses dengan 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 Utama

Berikut adalah *flowchart* alur sistem utama yang menunjukkan langkah-langkah terstruktur dari input data hingga hasil akhir berupa *plaintext* yang telah dipulihkan.

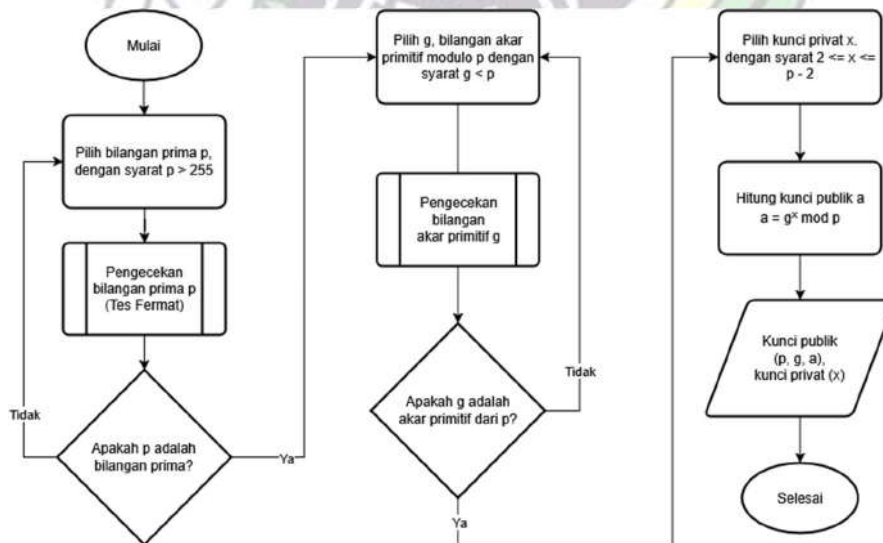


Gambar 3.4. Flowchart Sistem Utama

Alur sistem yang dijabarkan pada *flowchart* **Gambar 3.4.** yaitu, proses diawali dengan pengguna atau sistem yang memasukkan *plaintext* (data yang akan dienkripsi) beserta kunci simetris. Selanjutnya, *plaintext* tersebut dienkripsi menggunakan algoritma kriptografi simetris, untuk menghasilkan *ciphertext*. Setelah itu, sistem melakukan pembuatan pasangan kunci asimetris (*public key* dan *private key*). Kunci simetris yang digunakan sebelumnya kemudian dienkripsi menggunakan *public key* dalam skema kriptografi asimetris, sehingga hanya pemilik *private key* yang dapat mendekripsinya. Hasil dari proses enkripsi ini berupa *ciphertext* dan *encrypted key*, yang kemudian dikirimkan kepada penerima. Pada tahap dekripsi, penerima pesan terlebih dahulu harus mendekripsi kunci simetris yang terenkripsi menggunakan *private key* dalam skema kriptografi asimetris. Setelah berhasil mendapatkan kembali kunci simetris, penerima dapat menggunakannya untuk mendekripsi *ciphertext* menggunakan algoritma kriptografi simetris, sehingga *plaintext* asli dapat diperoleh kembali. Dengan demikian, proses dekripsi berhasil mengembalikan data ke bentuk semula.

3.3.2 Pembangkit kunci (key generation) Generalization of the ElGamal

Berikut adalah *flowchart* yang menggambarkan alur sistem Pembangkit Kunci (Key Generation) Generalization of the ElGamal.

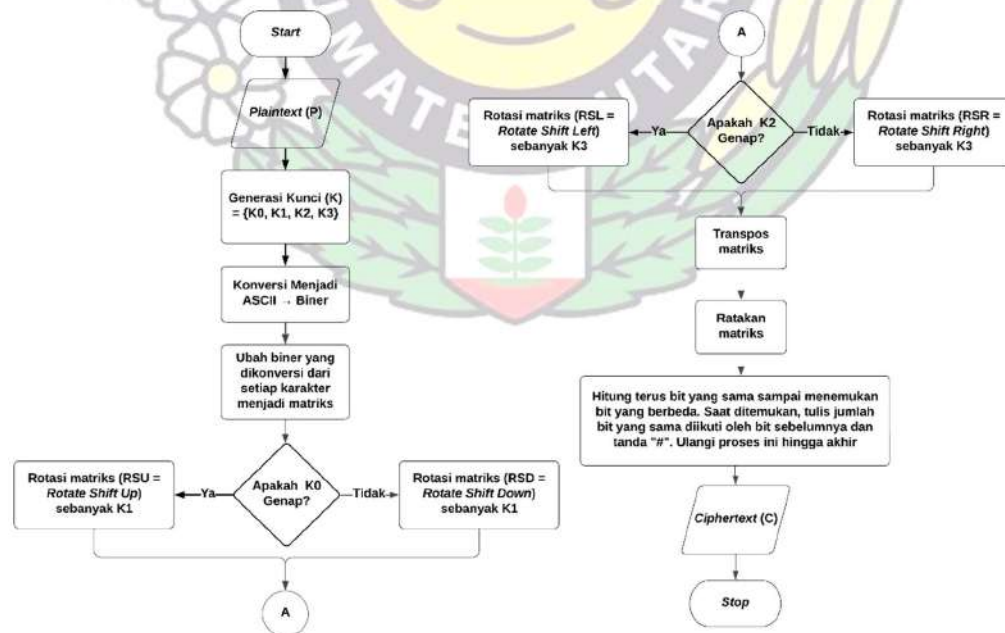


Gambar 3.5. Flowchart Pembangkit Kunci (Key Generation)

Alur sistem yang dijabarkan pada *flowchart* **Gambar 3.5.** yaitu, proses dimulai dengan memilih bilangan prima (p) yang lebih besar dari 255, ($p > 255$). Setelah itu, bilangan prima (p) diuji menggunakan Tes Fermat untuk memastikan bahwa bilangan tersebut benar-benar prima. Jika bilangan tidak lolos uji primalitas, maka pemilihan bilangan p diulang. Jika p adalah bilangan prima, langkah berikutnya adalah memilih bilangan akar primitif g modulo p dengan syarat $g < p$. Setelah memilih g , sistem memverifikasi apakah g benar-benar merupakan akar primitif dari p . Jika g tidak memenuhi syarat sebagai akar primitif, maka pemilihan bilangan g diulang hingga menemukan yang sesuai. Setelah validasi akar primitif selesai, proses berlanjut ke tahap berikutnya. Pengguna memilih kunci privat (x), yang harus berada dalam rentang $2 \leq x \leq p - 2$. Kunci publik kemudian dihitung menggunakan rumus $a \equiv g^x \pmod{p}$, dimana (p, g, a) menjadi kunci publik dan x menjadi kunci privat.

3.3.3 Martino Homomorphic Encryption

Berikut adalah *flowchart* yang menggambarkan alur sistem enkripsi Martino Homomorphic Encryption.



Gambar 3.6. Flowchart Martino Homomorphic Encryption

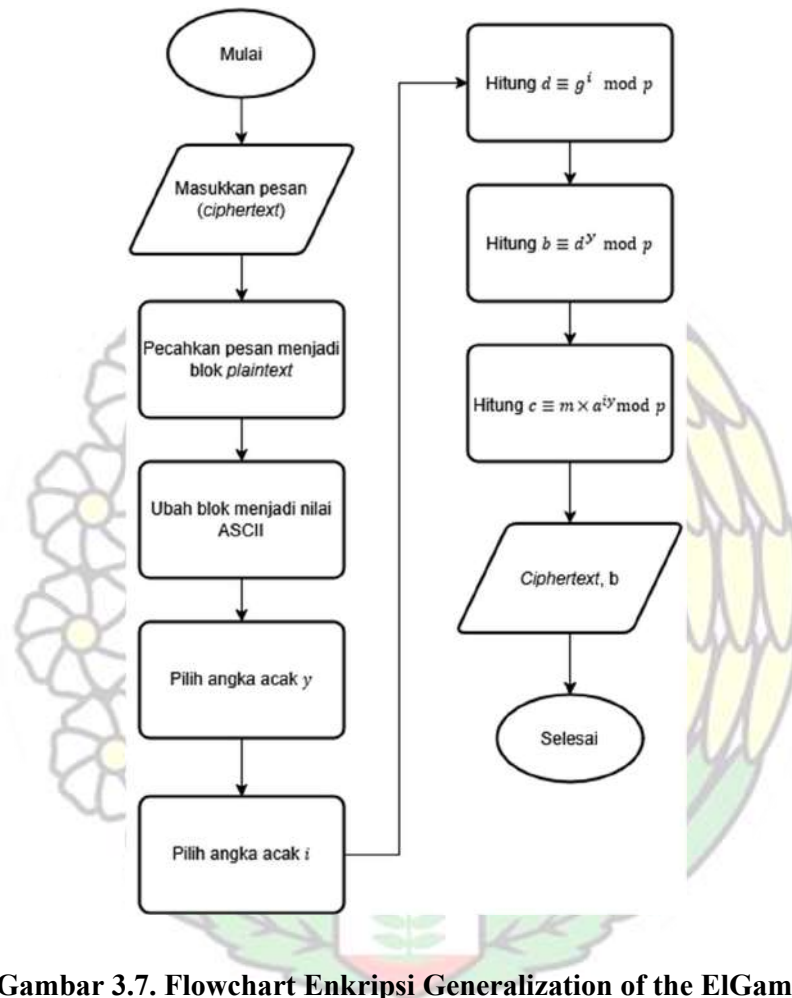
Alur sistem yang dijabarkan pada *flowchart* **Gambar 3.6** yaitu, Langkah pertama adalah memasukkan *plaintext* (P), yaitu teks asli yang akan dienkripsi agar tidak dapat dibaca langsung oleh pihak yang tidak berwenang. Selanjutnya, dilakukan generasi kunci (K) yang terdiri dari beberapa parameter, yaitu K0, K1, K2, dan K3. Kunci ini akan digunakan dalam berbagai tahap proses enkripsi, termasuk rotasi dan transposisi matriks, untuk meningkatkan keamanan data yang akan dienkripsi. Setelah kunci dihasilkan, *plaintext* dikonversi ke dalam format ASCII, lalu diubah menjadi representasi biner agar dapat diproses lebih lanjut dalam bentuk matriks. Setelah dikonversi ke biner, setiap karakter dari *plaintext* direpresentasikan sebagai elemen dalam sebuah matriks.

Kemudian, sistem memeriksa apakah nilai K0 merupakan bilangan genap atau ganjil. Jika K0 genap, maka matriks akan mengalami rotasi ke atas (*Rotate Shift Up* - RSU) sebanyak K1 kali. Jika K0 tidak genap, maka matriks akan mengalami rotasi ke bawah (*Rotate Shift Down* - RSD) sebanyak K1 kali. Setelah rotasi pertama ini selesai, alur akan berlanjut ke tahap pemrosesan selanjutnya, yaitu melakukan rotasi matriks kembali, kali ini dengan metode *Rotate Shift Left* (RSL) sebanyak K3 kali. Setelah itu, sistem akan mengevaluasi apakah nilai K2 merupakan bilangan genap. Jika K2 genap, maka matriks akan diratakan atau disusun ulang ke dalam format yang lebih sesuai. Jika K2 tidak genap, maka dilakukan rotasi dengan metode *Rotate Shift Right* (RSR) sebanyak K3 kali sebelum matriks diratakan.

Setelah matriks dalam keadaan rata, sistem melakukan transposisi matriks, di mana elemen-elemen dalam matriks ditukar posisinya berdasarkan aturan tertentu. Kemudian, sistem akan melakukan penghitungan jumlah bit yang sama secara berurutan hingga menemukan bit yang berbeda. Saat bit yang berbeda ditemukan, jumlah bit yang sama sebelumnya dituliskan sebagai bagian dari *ciphertext*, diawali dengan tanda "*". Proses ini diulang terus-menerus hingga seluruh *plaintext* dikonversi menjadi *ciphertext*. Tahap terakhir adalah menyimpan hasil akhir dari proses enkripsi sebagai *ciphertext* (C). *Ciphertext* ini merupakan bentuk terenkripsi dari *plaintext* yang tidak dapat langsung dibaca tanpa proses dekripsi menggunakan kunci yang sesuai.

3.3.4 Enkripsi Generalization of the ElGamal

Berikut adalah *flowchart* yang menggambarkan alur sistem enkripsi Generalization of the ElGamal.

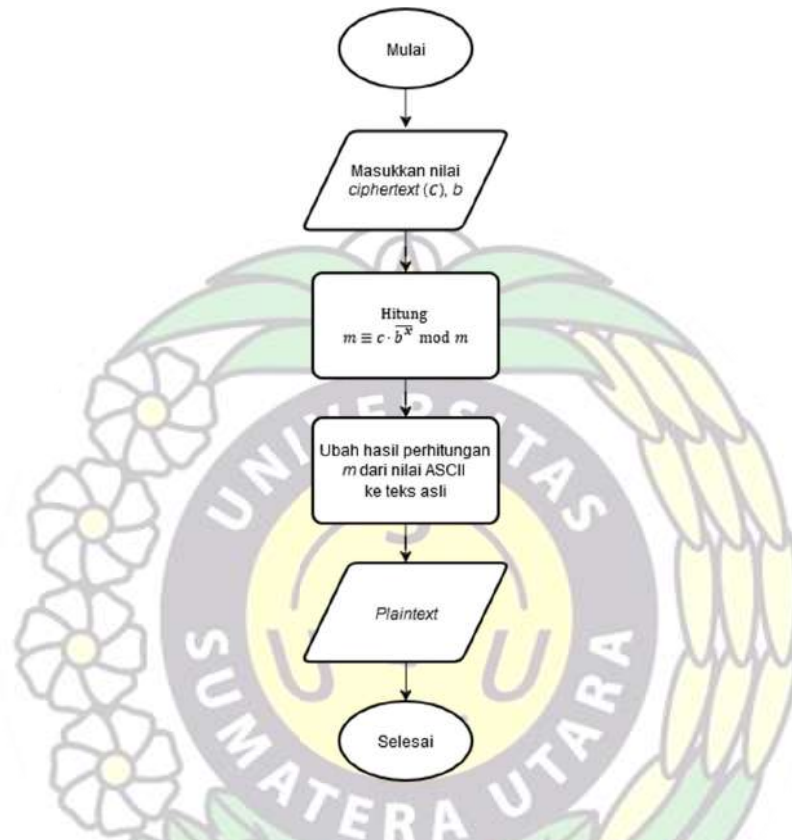


Gambar 3.7. Flowchart Enkripsi Generalization of the ElGamal

Alur sistem yang dijabarkan pada *flowchart* **Gambar 3.7**, yaitu, dimulai dengan memasukkan pesan yang akan dienkripsi, lalu dipecah menjadi blok-blok kecil dan dikonversi ke nilai ASCII. Setelah itu, dua angka acak, y dan i , dipilih untuk memastikan keamanan enkripsi. Perhitungan dilakukan dengan menghitung nilai d dan b . Selanjutnya, *ciphertext* dihitung dan menghasilkan nilai c , di mana m adalah nilai *plaintext* dalam bentuk blok pesan ASCII, dan a adalah kunci publik. Hasil akhirnya adalah pasangan nilai *ciphertext* dan b , yang dikirim kepada penerima pesan sebagai hasil enkripsi.

3.3.5 Dekripsi Generalization of the ElGamal

Berikut adalah *flowchart* yang menggambarkan alur sistem dekripsi Generalization of the ElGamal.

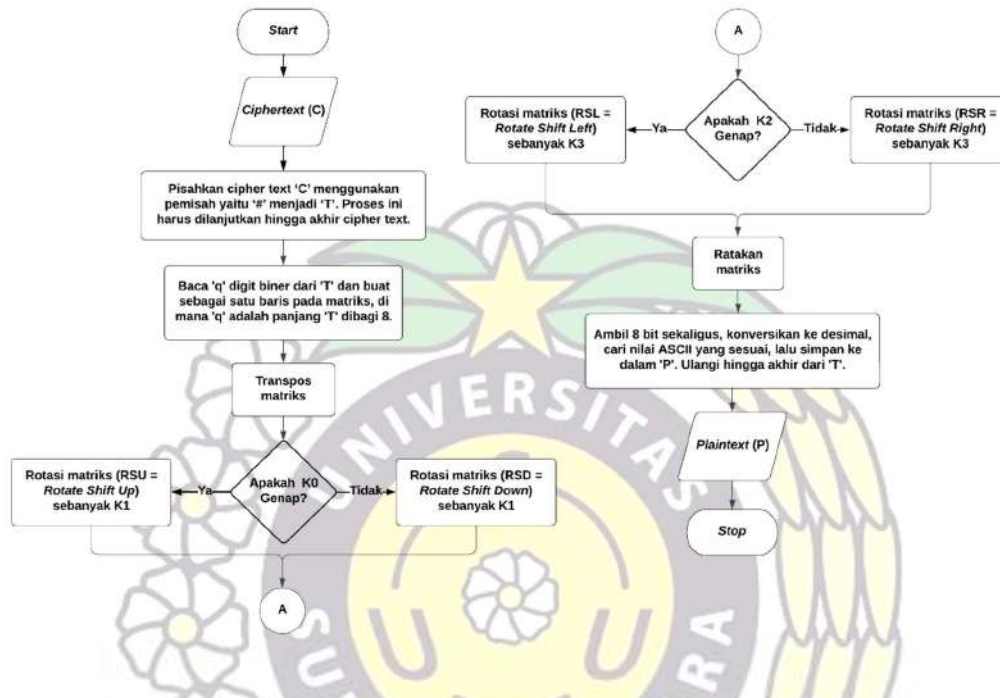


Gambar 3.6. Flowchart Dekripsi Generalization of the ElGamal

Alur sistem yang dijabarkan pada *flowchart* **Gambar 3.6** yaitu, pertama masukkan nilai *ciphertext* (*c*) yaitu pesan terenkripsi yang ingin didekripsi serta nilai *b*, yang kemungkinan merupakan bagian dari kunci dekripsi. Selanjutnya nilai *m* dikembalikan menggunakan persamaan $m \equiv c \cdot (\overline{b^x}) \pmod{m}$, di mana $(\overline{b^x})$ adalah invers dari b^x modulo *m*. Setelah mendapatkan nilai *m*, hasil perhitungan tersebut masih berbentuk angka yang mewakili karakter dalam format tertentu, seperti ASCII. Oleh karena itu, tahap berikutnya adalah mengonversi nilai *m* dari bentuk numerik ke teks asli yang dapat dibaca oleh pengguna. Proses ini melibatkan pencocokan nilai numerik dengan tabel karakter yang sesuai agar *plaintext* dapat direkonstruksi dengan benar.

3.3.6 Martino Homomorphic Decryption

Berikut adalah *flowchart* yang menggambarkan alur sistem dekripsi Martino Homomorphic Decryption.



Gambar 3.7. Flowchart Martino Homomorphic Decryption

Alur sistem yang dijabarkan pada *flowchart* **Gambar 3.7** yaitu, langkah pertama adalah memasukkan *ciphertext* (C) yang akan diproses lebih lanjut untuk dikembalikan ke bentuk aslinya. Selanjutnya, *ciphertext* (C) dipisahkan menggunakan pemisah tertentu, yaitu karakter "#", menjadi beberapa bagian yang disebut "T". Proses ini dilakukan secara berulang hingga seluruh *ciphertext* terbagi ke dalam bagian-bagian yang lebih kecil. Setelah itu, setiap bagian T dibaca dalam bentuk representasi biner, yang kemudian disusun menjadi baris dalam sebuah matriks.

Setelah berhasil, langkah selanjutnya adalah melakukan transposisi matriks. Setelah transposisi dilakukan, sistem akan menentukan apakah nilai K0 merupakan bilangan genap. Jika K0 genap, maka matriks akan dirotasi menggunakan metode *Rotate Shift Up* (RSU) sebanyak K1 kali. Jika K0 tidak genap, maka matriks akan dirotasi menggunakan metode *Rotate Shift Down* (RSD) sebanyak K1 kali. Setelah proses rotasi pertama ini selesai, alur akan berlanjut ke tahap pemrosesan berikutnya.

Selanjutnya adalah melakukan rotasi matriks kembali, tetapi kali ini dengan metode *Rotate Shift Left* (RSL) sebanyak K3 kali. Setelah itu, sistem akan mengevaluasi apakah nilai K2 merupakan bilangan genap. Jika K2 genap, maka matriks akan diratakan atau disusun ulang ke dalam format yang lebih sesuai untuk proses berikutnya. Namun, jika K2 tidak genap, maka dilakukan rotasi menggunakan metode *Rotate Shift Right* (RSR) sebanyak K3 kali sebelum matriks diratakan.

Setelah matriks dalam keadaan rata, tahap selanjutnya adalah mengambil 8 bit secara bertahap, mengonversinya ke bilangan desimal, dan mencocokkannya dengan karakter ASCII yang sesuai. Hasil dari proses ini akan disimpan ke dalam *plaintext* P. Proses ini terus diulang hingga seluruh bagian T dalam *ciphertext* telah diproses dan dikonversi kembali menjadi *plaintext*. Setelah semua data berhasil dikonversi, hasil akhirnya adalah teks asli yang dapat dibaca.

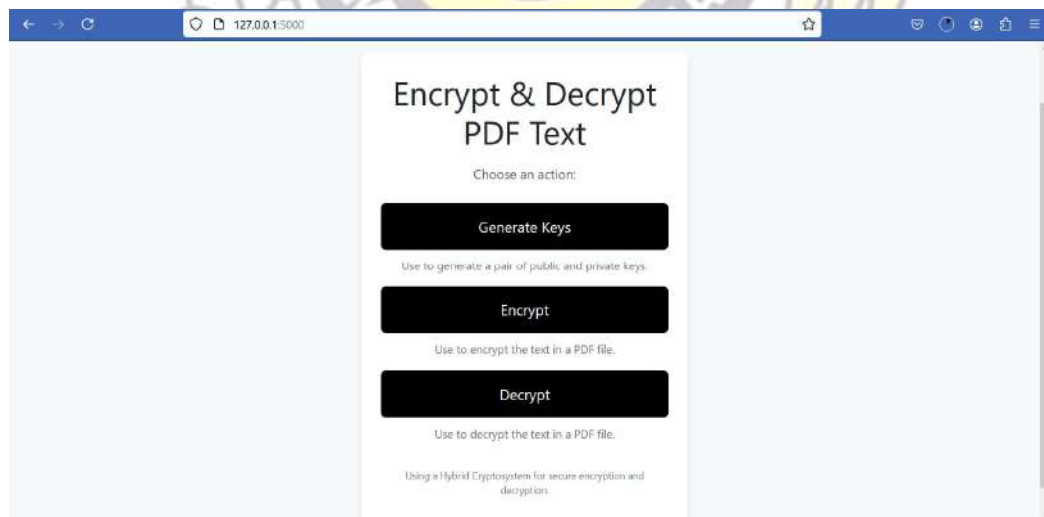


BAB IV

IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1 Implementasi Sistem

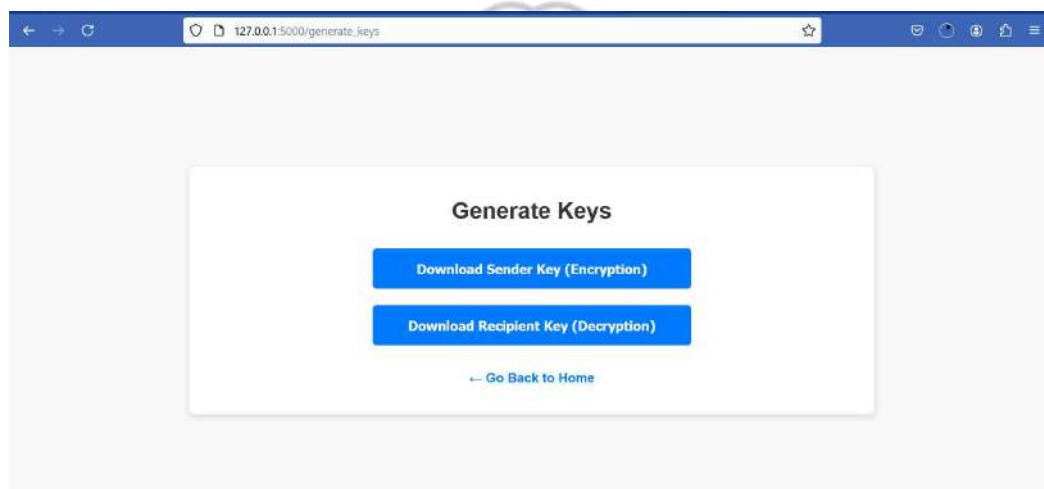
Sistem kriptosistem *hybrid* dibangun menggunakan Python Flask yang dirancang untuk mengintegrasikan algoritma simetris dan asimetris dalam menjaga keamanan data. Seperti pada **Gambar 4.1** Sistem terdiri dari beberapa halaman, yaitu halaman pembangkit kunci, halaman enkripsi untuk memproses *plaintext* menjadi *ciphertext* serta mengenkripsi kunci simetris, serta halaman dekripsi untuk memulihkan *plaintext* dari *ciphertext* menggunakan kunci simetris yang telah didekripsi. Antarmuka pengguna dibangun dengan HTML, CSS, dan JavaScript untuk mempermudah interaksi dengan pengguna. Backend Flask mengelola komunikasi antara frontend dan modul pembangkit kunci/enkripsi/dekripsi melalui REST API dengan endpoint utama seperti */key generation*, */encrypt* dan */decrypt*.



Gambar 4.1. Halaman Awal Sistem

4.1.1 Halaman pembangkit kunci

Membangkitkan kunci merupakan titik awal dalam proses. Pasangan kunci publik dan kunci privat dihasilkan menggunakan algoritma asimetris Generalization of the ElGamal. Seperti pada **Gambar 4.2**, ketika pengguna mengakses halaman ini, sistem akan secara otomatis membuat kunci publik dan privat dalam bentuk file PDF, dan dapat diunduh oleh pengguna untuk digunakan dalam proses enkripsi dan dekripsi.



Gambar 4.2. Halaman Pembangkit Kunci

4.1.2 Halaman enkripsi

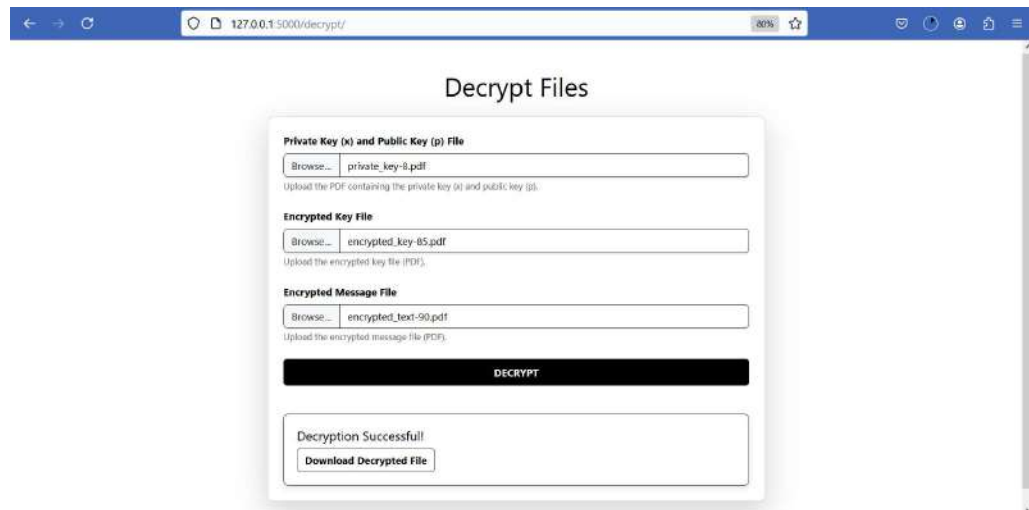
Halaman enkripsi dirancang untuk mengenkripsi data yang dimasukkan oleh pengguna. Seperti pada **Gambar 4.3**, pada halaman ini, pengguna dapat mengunggah *plaintext* berupa file PDF melalui *form* yang tersedia. Setelah tombol enkripsi ditekan, *plaintext* dienkripsi menggunakan algoritma simetris Martino Homomorphic Encryption, yang akan menghasilkan *ciphertext*. Selanjutnya, kunci simetris yang digunakan dalam proses ini akan dienkripsi menggunakan kunci publik yang telah dibangkitkan sebelumnya melalui algoritma Generalization of the ElGamal. Hasil akhir berupa *ciphertext* dan kunci simetris terenkripsi akan disimpan pada sistem, sementara *ciphertext* ditampilkan kepada pengguna atau dapat diunduh dalam format file seperti pada **Gambar 4.4**.

Gambar 4.3. Halaman Enkripsi

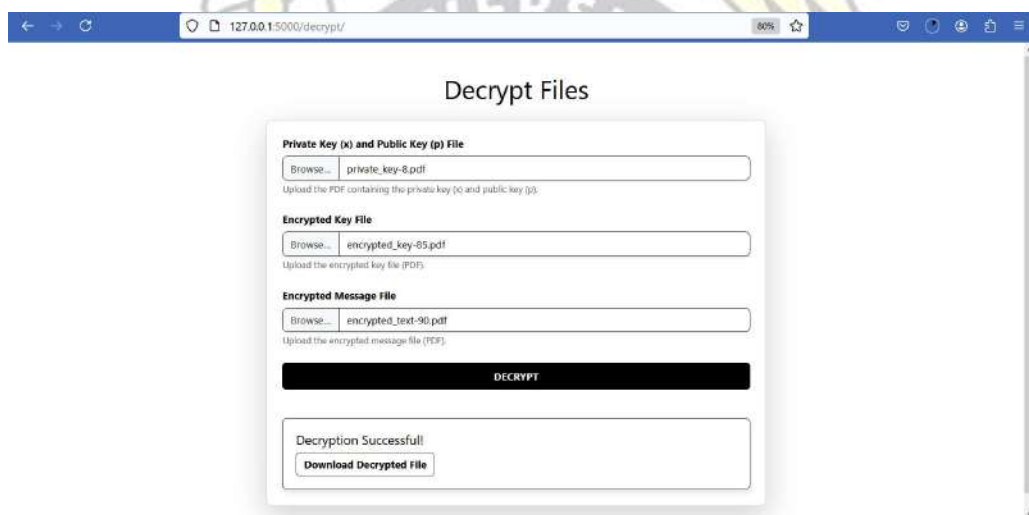
Gambar 4.4. Halaman Hasil Enkripsi

4.1.3 Halaman Dekripsi

Halaman dekripsi berfungsi untuk memulihkan *plaintext* dari *ciphertext*. Seperti pada **Gambar 4.5**, pada halaman ini, pengguna mengunggah file PDF berupa *ciphertext* beserta kunci simetris terenkripsi yang diperoleh dari proses sebelumnya. Sistem akan terlebih dahulu mendekripsi kunci simetris menggunakan kunci privat. Setelah kunci simetris berhasil diperoleh, *ciphertext* akan didekripsi menggunakan algoritma simetris, sehingga *plaintext* asli dapat dipulihkan. File PDF hasil dekripsi akan tersedia untuk diunduh oleh pengguna. Dokumen asli dapat diakses kembali dengan format dan isi yang sama seperti sebelum proses enkripsi seperti pada **Gambar 4.6**.



Gambar 4.5. Halaman Dekripsi



Gambar 4.6. Halaman Hasil Dekripsi

4.2 Perhitungan *Avalanche Effect*

Avalanche effect mengacu pada kondisi di mana perubahan kecil dalam input (*plaintext* atau kunci) menghasilkan perubahan yang signifikan pada output (*ciphertext*). Idealnya, perubahan satu bit pada *plaintext* atau kunci harus menghasilkan perubahan sekitar 50% pada *ciphertext* agar algoritma kriptografi dianggap aman. Efek ini penting karena menunjukkan bahwa algoritma tersebut sensitif terhadap input dan dapat mencegah prediktabilitas.

Avalanche effect menjadi salah satu faktor yang dievaluasi dalam berbagai algoritma kriptografi. Penelitian ini mengukur seberapa efektif algoritma Martino Homomorphic Encryption dalam menghasilkan avalanche effect. Hasil pengujian dapat dilihat pada **Tabel 4.1**.

Tabel 4.1. Hasil Percobaan Avalanche Effect

No.	Ukuran Data (Byte)	Total Bit yang Berubah	Avalanche Effect (%)
1	16	64	50.00%
2	32	62	48.44%
3	64	66	51.56%
4	128	60	46.88%

Avalanche Effect pada algoritma Martino Homomorfik Encryption diuji menggunakan berbagai ukuran data, yaitu 16 *byte*, 32 *byte*, 64 *byte*, dan 128 *byte*. Pengujian dilakukan dengan cara mengubah satu bit pada *plaintext* atau kunci, kemudian menghitung persentase perubahan bit antara *ciphertext* sebelum dan setelah perubahan. Hasil pengujian menunjukkan bahwa untuk ukuran data 16 *byte*, Avalanche Effect sebesar 50.00%. Untuk ukuran 32 *byte*, persentase perubahan bit adalah 48.44%, sedangkan untuk data 64 *byte* dan 128 *byte* masing-masing menghasilkan Avalanche Effect sebesar 51.56% dan 46.88%. Nilai-nilai ini berada dalam rentang ideal antara 40% hingga 60%.

4.3 Perhitungan Kompleksitas Algoritma

Kompleksitas algoritma mengukur seberapa cepat atau efisien suatu algoritma dapat menyelesaikan suatu masalah. Notasi Big-O menjadi alat utama dalam menghitung kompleksitas algoritma, di mana notasi ini menggambarkan hubungan antara ukuran input dan waktu eksekusi atau penggunaan ruang. Berikut adalah tabel kompleksitas algoritma enkripsi Hybrid Cryptosystem dengan Martino Homomorphic Encryption dan Generalization of the ElGamal:

Tabel 4.2. Hasil Perhitungan Kompleksitas Algoritma Enkripsi

Fungsi	Langkah	Kompleksitas
Enkripsi Simetris	Konversi String ke Matriks Biner	$O(n)$
	Rotasi <i>Shift Up/Down</i>	$O(n)$
	Rotasi <i>Shift Right/Left</i>	$O(n)$
	<i>Transpose</i> Matriks	$O(n)$
	<i>Flatten</i> Matriks	$O(n)$
	<i>Run-Length Encoding</i>	$O(n)$
Total Enkripsi Simetris		$O(n)$
<i>Generate keys</i>	Pemilihan Bilangan Prima p	$O(\log p)$
	Memeriksa Akar Primitif g	$O(\sqrt{n} \log(n) + x \log(p))$
	Menghasilkan Kunci Privat x	$O(1)$
	Menghitung Kunci Publik a	$O(\log p)$
Total <i>Generate keys</i>		$O(\sqrt{n} \log(n) + x \log(p))$
Enkripsi Asimetris	Menghasilkan Nilai Acak i dan y	$O(1)$
	Menghitung Nilai d dan b	$O(\log p)$
	Menghitung Nilai C	$O(n \log p)$
Total Enkripsi Asimetris		$O(n \log p)$
Total Enkripsi		$O(n) + O(\sqrt{n} \log(n) + x \log(p)) + O(n \log p)$

Tabel 4.3. Hasil Perhitungan Kompleksitas Algoritma Dekripsi

Fungsi	Langkah	Kompleksitas
Dekripsi Simetris	Konversi String ke Matriks	$O(n)$
	Rotasi <i>Shift Up/Down</i>	$O(n)$
	Rotasi <i>Shift Right/Left</i>	$O(n)$
	<i>Transpose</i> Matriks	$O(n)$
	<i>Flatten</i> Matriks	$O(n)$
	<i>Run-Length Decoding</i>	$O(n)$
Total Dekripsi Simetris		$O(n)$
Total Dekripsi Asimetris		$O(1)$
Total Dekripsi		$O(n) + O(1)$

4.3.1 Kompleksitas algoritma fungsi pemeriksaan bilangan prima

Tabel 4.4. Kompleksitas Algoritma Fungsi Pemeriksaan Bilangan Prima

Fermat(p):	C	#	C#
$t = 10$	C1	1	C1
For i from 1 to t Do:	C2	t	$t \cdot C2$
$z = \text{Random integer in the range } [2, p - 1]$	C3	t	$t \cdot C3$
If $(z^{(p-1)} \bmod p) \neq 1$ Then	C4	$t \cdot \log(p)$	$t \cdot \log(p) \cdot C4$
return False	C5	1	C5
return True	C5	1	C5

$$T(p) = C1 + t C2 + t C3 + t \log(p) C4 + C5 + C5$$

$$T(p) = C1 + t (C2 + C3) + \log(p) C4 + 2 C5$$

$$T(p) \approx \theta(t \log p)$$

4.3.2 Kompleksitas algoritma fungsi pemeriksaan bilangan akar primitif

Tabel 4.5. Kompleksitas Algoritma Fungsi Pemeriksaan Bilangan Akar Primitif

Primitive Root(g, p):	C	#	C#
$n = p - 1$	C1	1	C1
Initialize factors as empty list	C1	1	C1
For i from 2 to \sqrt{n} Do:	C2	\sqrt{n}	$\sqrt{n} \cdot C2$
While n is divisible by i Do:	C3	$\sqrt{n} \log(n)$	$\sqrt{n} \log(n) \cdot C3$
Add i to factors	C4	k	$k \cdot C4$
Divide n by i	C5	$\log(n)$	$\log(n) \cdot C5$
If $n > 1$ Then	C6	1	C6
Add n to factors	C4	1	C4
For each q in factors Do:	C2	x	$x \cdot C2$
If $g^{\frac{n}{q}} \bmod p == 1$ Then	C6	$x \cdot \log(p)$	$x \cdot \log(p) \cdot C6$
return False	C7	$\log(x)$	$\log(x) \cdot C7$
return True	C7	1	C7

$$T(g, p) = C1 + C1 + \sqrt{n} C2 + \sqrt{n} \log(n) C3 + k C4 + \log(n) C5 + C6 + C4 + x C2 + x \log(p) C6 + \log(x) C7 + C7$$

$$T(g, p) = 2 C1 + x C2 + \sqrt{n} C2 + \sqrt{n} \log(n) C3 + C4 + k C4 + \log(n) C5 + x \log(p) C6 + C7 + \log(x) C7$$

$$T(g, p) \approx \theta(\sqrt{n} \log(n) + x \log(p))$$

4.3.3 Kompleksitas algoritma fungsi pembangkit kunci

Tabel 4.6. Kompleksitas Algoritma Fungsi Pembangkit Kunci

Generate_Keys():	C	#	C#
$p = \text{random integer}$	C1	1	C1

while p is not a prime number:	C2	$k \cdot t \cdot \log(p)$	$k \cdot t \cdot \log(p) \cdot C2$
p = random integer	C1	k	$k \cdot C1$
g = random integer	C1	1	$C1$
While g is not a primitive root of p (g, p):	C2	$m \cdot (\sqrt{n} \cdot \log(n) + x \cdot \log(p))$	$m \cdot (\sqrt{n} \cdot \log(n) + x \cdot \log(p)) \cdot C2$
g = random integer	C1	m	$m \cdot C1$
x = random integer	C1	1	$C1$
$a = g^x \bmod p$	C3	$\log(p)$	$\log(p) \cdot C3$
return (p, g, a), (x)	C4	1	$C4$

$$T = C1 + k t \log(p) C2 + k C1 + C1 + m (\sqrt{n} \log(n) + x \log(p)) C2 + m C1 + C1 + \log(p) C3 + C4$$

$$T = (3 + k + m) C1 + (k t \log(p) + m (\sqrt{n} \log(n) + x \log(p))) C2 + \log(p) C3 + C4$$

$$T \approx \theta(m \cdot \sqrt{n} \cdot \log(n) + (m \cdot x + k \cdot t) \cdot \log(p))$$

4.3.4 Kompleksitas algoritma fungsi konversi string ke matriks biner

Tabel 4.7. Kompleksitas Algoritma Fungsi Konversi String ke Matriks Biner

string_to_binary_matrix(string):	C	#	C#
binary matrix = [empty list]	C1	1	C1
for each character in string:	C2	n	$n \cdot C2$
binary_rep = format(ord(char), '08b')	C1	n	$n \cdot C1$
binary_matix.append([int(bit) for bit in binary_rep])	C3	n	$n \cdot C3$
return binary matrix	C4	1	C4

$$T(s) = C1 + n C2 + n C1 + n C3 + C4$$

$$T(s) = C1 + n (C1 + C2 + C3) + C4$$

$$T(s) \approx \theta(n)$$

4.3.5 Kompleksitas algoritma fungsi rotasi shift up

Tabel 4.8. Kompleksitas Algoritma Fungsi Rotasi Shift Up

rotate_shift_up(matrix, times):	C	#	C#
each iteration from 1 to `times`:	C1	n	$n \cdot C1$
first_row = matrix.pop(0)	C2	n	$n \cdot C2$
matrix.append(first_row)	C3	n	$n \cdot C3$
return matrix	C4	1	$C4$

$$T(m, t) = n C1 + n C2 + n C3 + C4$$

$$T(m, t) = n (C1 + C2 + C3) + C4$$

$$T(m, t) \approx \theta(n)$$

4.3.6 Kompleksitas algoritma fungsi rotasi shift right

Tabel 4.9. Kompleksitas Algoritma Fungsi Rotasi Shift Right

Rotate_shift_right(matrix, times):	C	#	C#
For each iteration from 1 to `times`:	C1	t	$t \cdot C1$
For each row in the `matrix`:	C1	$t \cdot m$	$t \cdot m \cdot C1$
last_element = row.pop()	C2	$t \cdot m$	$t \cdot m \cdot C2$
row.insert(0, last_element)	C3	$t \cdot m$	$t \cdot m \cdot C3$
return matrix	C4	1	$C4$

$$T(m, t) = t C1 + t m C1 + t m C2 + t m C3 + C4$$

$$T(m, t) = t (C1 + (C2 + C3) m) + C4$$

$$T(m, t) \approx \theta(t m)$$

$$\approx \theta(n)$$

4.3.7 Kompleksitas algoritma fungsi rotasi shift down

Tabel 4.10. Kompleksitas Algoritma Fungsi Rotasi Shift Down

rotate_shift_down(matrix, times):	C	#	C#
For each iteration from 1 to `times`:	C1	n	$n \cdot C1$
last_row = matrix.pop()	C2	n	$n \cdot C2$
matrix.insert(0, last_row)	C3	n	$n \cdot C3$
return matrix	C4	1	$C4$

$$T(m, t) = n C1 + n C2 + n C3 + C4$$

$$T(m, t) = n (C1 + C2 + C3) + C4$$

$$T(m, t) \approx \theta(n)$$

4.3.8 Kompleksitas algoritma fungsi rotasi shift left

Tabel 4.11. Kompleksitas Algoritma Fungsi Rotasi Shift Left

rotate_shift_left(matrix, times):	C	#	C#
For each iteration from 1 to `times`:	C1	m	$m \cdot C1$
For each row in the `matrix`:	C1	$m \cdot t$	$m \cdot t \cdot C1$
first_element = row.pop(0)	C2	$m \cdot t$	$m \cdot t \cdot C2$
row.append(first_element)	C3	$m \cdot t$	$m \cdot t \cdot C3$
return matrix	C4	1	$C4$

$$T(m, t) = t C1 + t m C1 + t m C2 + t m C3 + C4$$

$$T(m, t) = t (C1 + (C2 + C3) m) + C4$$

$$T(m, t) \approx \theta(t m)$$

$$\approx \theta(n)$$

4.3.9 Kompleksitas algoritma fungsi run-length encoding

Tabel 4.12. Kompleksitas Algoritma Fungsi Run-Length Encoding

run_length_encode(<i>bits</i>):	C	#	C#
If bits is empty:	C1	1	C1
return empty string	C2	1	C2
encoded = [empty list]	C3	1	C3
current_bit = bits[0]	C3	1	C3
count = 1	C3	1	C3
for each bits[1:]:	C4	n	$n \cdot C4$
if bit = current_bit:	C5	n	$n \cdot C5$
Increment 'count' by 1	C3	n	$n \cdot C3$
else: encoded.append(f'{count}{current_bit}#')	C6	n	$n \cdot C6$
current_bit = bit	C3	n	$n \cdot C3$
count = 1	C3	n	$n \cdot C3$
encoded.append(f'{count}{current_bit}')	C6	1	C6
Return join(encoded)	C2	k	$k \cdot C2$

$$T(b) = C1 + C2 + C3 + C3 + C3 + n C4 + n C5 + C3 + n C6 + n C3 + n C3 + C6 + k C2$$

$$T(b) = C1 + C2 + 3 C3 + C6 + n (3 C3 + C4 + C5 + C6) + k C2$$

$$T(b) \approx \theta(n)$$

4.3.10 Kompleksitas algoritma fungsi enkripsi simetris

Tabel 4.13. Kompleksitas Algoritma Fungsi Enkripsi Simetris

Symmetric_Encrypt(symmetric_key, public_key):	C	#	C#
binary_matrix = string_to_binary_matrix(data)	C1	n	$n \cdot C1$
If symmetric_key[0] is even:	C2	1	C2
rotations = symmetric_key[1]	C1	1	C1
Perform RotateShiftUp(binary_matrix, rotations)	C1	n	$n \cdot C1$
Else: rotations = symmetric_key[1]	C1	1	C1
Perform RotateShiftDown(binary_matrix, rotations)	C1	n	$n \cdot C1$
If symmetric_key[2] is odd:	C2	1	C2
rotations = symmetric_key[3]	C1	1	C1
Perform RotateShiftRight(binary_matrix, rotations)	C1	n	$n \cdot C1$
Else: rotations = symmetric_key[3]	C1	1	C1
Perform RotateShiftLeft(binary_matrix, rotations)	C1	n	$n \cdot C1$
transposed_matrix = transpose(binary_matrix)	C1	$n \cdot p$	$n \cdot p \cdot C1$
flattened_array = flatten(transposed_matrix)	C1	$n \cdot p$	$n \cdot p \cdot C1$
encoded_string = run_length_encode(flattened_array)	C1	k	$k \cdot C1$
return encoded string	C2	1	C2

$$T(s) = n \cdot C1 + C2 + C1 + n \cdot C1 + C1 + n \cdot C1 + C2 + C1 + n \cdot C1 + C1 + n \cdot C1 + n \cdot p \cdot C1 + n \cdot p \cdot C1 + k \cdot C1 + C2$$

$$T(s) = 4 C1 + n(5 C1 + (2 C1) p) + k C1 + 3 C2$$

$$T(s) \approx \theta(n)$$

4.3.11 Kompleksitas algoritma fungsi enkripsi asimetris

Tabel 4.14. Kompleksitas Algoritma Fungsi Enkripsi Asimetris

assymmetric_encrypt (<i>symmetric_key</i> , <i>public_key</i>):	C	#	C#
p, g, a = public_key	C1	1	C1
c = symmetric_key	C1	1	C1
y = random integer	C2	1	C2
i = random integer	C2	1	C2
$d = g^i \bmod p$	C3	$\log(p)$	$\log(p) C3$
$b = g^x \bmod p$	C3	$\log(p)$	$\log(p) C3$
$C = a^i \bmod y$	C3	$\log(p)$	$\log(p) C3$
$C = ((C \% p) * c) \% p$	C4	1	C4
return C, b	C5	1	C5

$$T(a) = C1 + C1 + C2 + C2 + \log(p) C3 + \log(p) C3 + \log(p) C3 + C4 + C5$$

$$T(a) = 2 (C1 + C2) + 3 \log(p) C3 + C4 + C5$$

$$T(a) = \theta(\log p)$$

4.3.12 Kompleksitas algoritma fungsi enkripsi asimetris

Tabel 4.15. Kompleksitas Algoritma Fungsi Konversi String ke Matriks

string_to_matrix(<i>bits</i> , <i>q</i>):	C	#	C#
matrix = [empty list]	C1	1	C1
for i in range(0, len(bits), q):	C2	n/q	$n/q \cdot C2$
bits[i:i + q]	C3	n	$n \cdot C3$
matrix.append(bits[i:i + q])	C4	n/q	$n/q \cdot C4$
return matrix	C5	1	C5

$$T(b) = C1 + \frac{n}{q} \cdot C2 + n \cdot C3 + \frac{n}{q} \cdot C4 + C5$$

$$T(b) = C1 + \frac{n}{q} (C2 + C4) + n \cdot C3 + C5$$

$$T(b) \approx \theta(n)$$

4.3.13 Kompleksitas Algoritma Fungsi Invers Modulo

Tabel 4.16. Kompleksitas Algoritma Fungsi Invers Modulo

def mod_inverse(z, p)	C	#	C#
X0, X1 = 0, 1	C1	1	C1
D, r = p, z	C1	1	C1
while r ≠ 0:	C2	$\log(p)$	$\log(p) C2$
q = D // r	C3	$\log(p)$	$\log(p) C3$
X0, X1 = X1, X0 - q * X1	C4	$\log(p)$	$\log(p) C4$
D, r = r, D mod r	C4	$\log(p)$	$\log(p) C4$
if X0 < 0:	C5	1	C5
Increment 'X0' by p	C3	1	C3
return X0	C6	1	C6

$$T(a) = C1 + C1 + \log(p) C2 + \log(p) C3 + \log(p) C4 + \log(p) C4 + C5 + C3 + C6$$

$$T(a) = 2 C1 + \log(p) (C2 + C3 + 2C4) + C5 + C3 + C6$$

$$T(a) \approx \theta(\log p)$$

4.3.14 Kompleksitas algoritma fungsi run-length decoding

Tabel 4.17. Kompleksitas Algoritma Fungsi Run-Length Decoding

RunLengthDecode(bits)	C	#	C#
Decoded = [empty list]	C1	1	C1

Join = [empty list]	C1	1	$C1$
count = 0	C1	1	$C1$
bits_index = 0	C1	1	$C1$
For each bits:	C2	n	$n \cdot C2$
If bit is \neq '#' Then	C3	n	$n \cdot C3$
Increment count by 1	C1	l	$l \cdot C1$
Else:	C1	m	$m \cdot C1$
temp = count			
While temp > 0:	C4	j	$j \cdot C4$
If temp = 1 Then Break	C3	j	$j \cdot C3$
Append bits[bits_index - temp] to Join	C5	j	$j \cdot C5$
Decrement temp by 1	C1	j	$j \cdot C1$
changer = Join elements joined into a single string	C1	m	$m \cdot C1$
converted = Convert changer to integer	C1	m	$m \cdot C1$
While converted > 0:	C4	k	$k \cdot C4$
Append bits[bits_index - 1] to decoded	C5	k	$k \cdot C5$
Decrement converted by 1	C1	k	$k \cdot C1$
Clear Join	C6	m	$m \cdot C6$
count = 0	C1	m	$m \cdot C1$
Increment bits_index by 1	C1	n	$n \cdot C1$
size = Length of bits	C1	1	$C1$
If bits[size - 1] \neq '#':	C3	1	$C3$
temp = count	C1	1	$C1$
While temp > 0:	C4	p	$p \cdot C4$
If temp = 1 Then Break	C3	p	$p \cdot C3$
Append bits[bits_index - temp] to Join	C5	p	$p \cdot C5$
Decrement temp by 1	C1	p	$p \cdot C1$
changer = Concatenate characters in Join	C1	1	$C1$
converted = Convert changer to integer	C1	1	$C1$
While converted > 0:	C4	q	$q \cdot C4$

Append bits[bits_index - 1] to decoded	C5	q	$q \cdot C5$
Decrement converted by 1	C1	q	$q \cdot C1$
Return Join elements of decoded as a single string	C7	1	$C7$

$$T(r) = C1 + C1 + C1 + C1 + n C2 + n C3 + l C1 + m C1 + j C4 + j C3 + j C5 + j C1 + m C1 + m C1 + k C4 + k C5 + k C1 + m C6 + m C1 + n C1 + C1 + C3 + C1 + p C4 + p C3 + p C5 + p C1 + C1 + C1 + q C4 + q C5 + q C1 + C7$$

$$T(r) = 8 C1 + l C1 + j C1 + 3 m C1 + k C1 + n C1 + p C1 + q C1 + n C2 + C3 + n C3 + j C3 + p C3 + j C4 + k C4 + p C4 + q C4 + j C5 + k C5 + p C5 + q C5 + m C6 + C7$$

$$T(r) \approx \theta(n)$$

4.3.15 Kompleksitas algoritma fungsi dekripsi asimetris

Tabel 4.18. Kompleksitas Algoritma Fungsi Dekripsi Asimetris

decrypt_key(b, x, p, C):	C	#	C#
$z = b^x \bmod p$	C1	1	$C1$
$w = \text{mod_inverse}(z, p)$	C1	1	$C1$
$m = \text{char}((C * w) \bmod p)$	C2	1	$C2$
return m	C5	1	$C5$

$$T(b, x, p, C) = C1 + C1 + C2 + C5$$

$$T(b, x, p, C) = 2C1 + C2 + C5$$

$$T(b, x, p, C) \approx \theta(1)$$

4.3.16 Kompleksitas algoritma fungsi dekripsi simetris

Tabel 4.19. Kompleksitas Algoritma Fungsi Dekripsi Simetris

SymmetricDecrypt(cipher_text, symmetric_key)	C	#	C#
decoded_bits = run_length_decode(cipher_text)	C1	n	$n \cdot C1$
q = Length of decoded_bits divided by 8	C1	1	C1
matrix = string_to_matrix(decoded_bits, q)	C1	n	$n \cdot C1$
transposed_matrix = transpose(matrix)	C1	n	$n \cdot C1$
If symmetric_key[0] is odd Then	C2	1	C2
rotations = Convert symmetric_key[1] to integer	C1	1	C1
transposed_matrix = RotateShiftUp (transposed_matrix, rotations)	C1	n	$n \cdot C1$
Else rotations = Convert symmetric_key[1] to integer	C1	1	C1
transposed_matrix = RotateShiftDown (transposed_matrix, rotations)	C1	n	$n \cdot C1$
If symmetric_key[2] is odd Then	C2	1	C2
rotations = Convert symmetric_key[3] to integer	C1	1	C1
transposed_matrix = RotateShiftLeft (transposed_matrix, rotations)	C1	n	$n \cdot C1$
Else rotations = Convert symmetric_key[3] to integer	C1	1	C1
transposed_matrix = RotateShiftRight (transposed_matrix, rotations)	C1	n	$n \cdot C1$
flattened_bits = Flatten(transposed_matrix)	C1	$n \cdot m$	$n \cdot m$
Initialize plain_text as empty	C1	1	C1
For i from 0 to Length of flattened_bits in steps of 8 Do:	C3	n	$n \cdot C3$
binary_value = Subarray of flattened_bits from i to i + 7	C1	1	C1

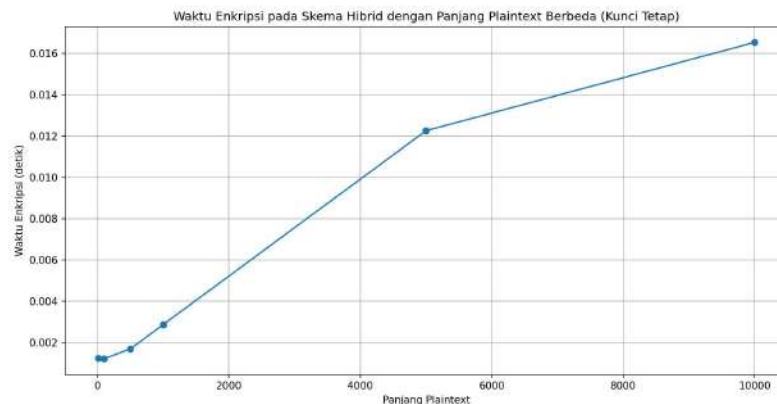
decimal_value = BinaryToDecimal (binary_value)	C1	1	C1
Append Character corresponding to decimal_value to plain_text	C1	1	C1
Return plain_text	C4	1	C4

$$T(s) = n C1 + C2 + C1 + n C1 + C1 + + n C1 + C2 + C1 + n C1 + C1 + n C1 + n m + C1 + n C3 + C1 + C1 + C4$$

$$T(s) = 7 C1 + n 5 C1 + 2 C2 + n C3 + C4$$

$$T(s) \approx \theta(n)$$

4.4 Analisis Perubahan Waktu dalam Proses Enkripsi

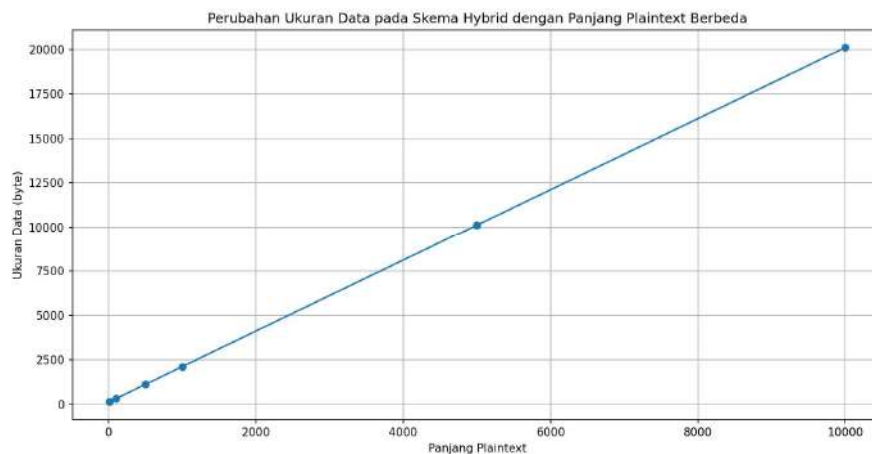


Gambar 4.7. Grafik Perubahan Waktu Enkripsi pada Skema Hibrid

Seperti pada **Gambar 4.6** hasil pengujian menunjukkan bahwa waktu enkripsi pada skema *hybrid* memiliki hubungan linier dengan panjang plaintext. Semakin panjang plaintext yang dienkrpsi, semakin lama waktu yang dibutuhkan untuk proses enkripsi. Namun, peningkatan waktu ini relatif kecil.

Bahkan untuk plaintext sepanjang 10.000 karakter, sehingga membuktikan bahwa skema hybrid cukup efisien untuk digunakan dalam berbagai skenario, terutama pada data dengan ukuran kecil hingga sedang. Dengan penggunaan kunci simetris tetap sepanjang 4 digit, performa enkripsi tetap konsisten tanpa adanya variasi signifikan pada waktu enkripsi selain yang disebabkan oleh panjang plaintext.

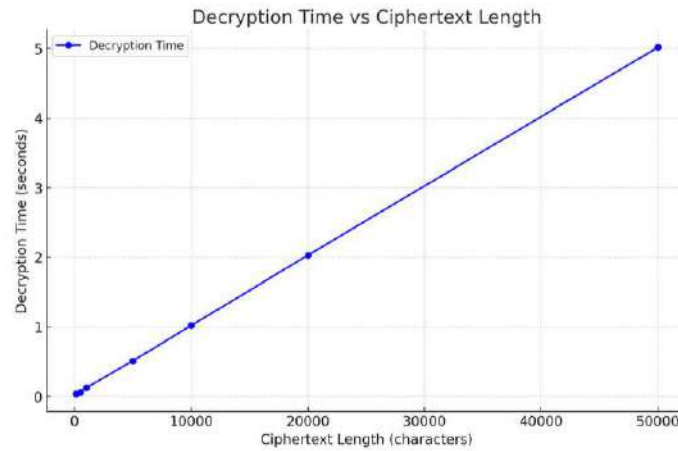
4.5 Analisis Perubahan Data dalam Proses Enkripsi



Gambar 4.8. Grafik Perubahan Data Enkripsi pada Skema Hibrid

Seperti pada **Gambar 4.7** dapat disimpulkan bahwa ukuran ciphertext meningkat secara linear seiring dengan panjang plaintext. Setiap tambahan karakter pada plaintext menghasilkan pertambahan yang konsisten pada ukuran ciphertext. Sebagai contoh, untuk plaintext dengan panjang 10 karakter, ukuran ciphertextnya adalah 40 byte, sementara untuk plaintext sepanjang 10000 karakter mencapai 40000 byte. Hal ini menunjukkan bahwa proses enkripsi menyebabkan pertambahan ukuran data yang signifikan, yang wajar mengingat skema enkripsi yang digunakan membutuhkan lebih banyak data untuk menyimpan informasi terkait keamanan.

4.6 Analisis Perubahan Waktu dalam Proses Dekripsi



Gambar 4.9. Grafik Perubahan Waktu Dekripsi pada Skema Hibrid

Tabel 4.22. Hasil Perubahan Waktu Dekripsi pada Skema Hibrid

Ciphertext Length (characters)	Decryption Time (seconds)
100	0.035354
500	0.062266
1,000	0.123388
5,000	0.513604
10,000	1.019372
20,000	2.029565
50,000	5.018477

Seperti pada **Gambar 4.8** dan **Tabel 4.21**, rasio yang stabil menunjukkan bahwa algoritma dekripsi memiliki efisiensi yang baik meskipun panjang ciphertext meningkat secara signifikan.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Adapun Kesimpulan yang dapat diperoleh dari penelitian setelah melewati tahapan analisis hingga implementasi adalah sebagai berikut:

1. Dari percobaan yang telah dilakukan, algoritma Martino Homomorphic Encryption menghasilkan avalanche effect sekitar 50,34%.
2. Skema hybrid cukup efisien untuk digunakan pada data dengan ukuran kecil hingga sedang.
3. Kompleksitas total enkripsi yang diperoleh $O(n) + O(\sqrt{n} \cdot \log(n) + x \cdot \log(p)) + O(n \log p)$
4. Kompleksitas total *Generate keys* yang diperoleh $O(\sqrt{n} \cdot \log(n) + x \cdot \log(p))$
5. Kompleksitas total dekripsi yang diperoleh $O(n) + O(1)$

5.2 Saran

Adapun saran yang dapat ditambahkan untuk penelitian selanjutnya adalah sebagai berikut:

1. Pemanfaatan metode serangan kriptografi untuk menguji sistem yang dapat dilakukan pada penelitian selanjutnya.
2. Pengujian dengan menggunakan bilangan prima yang lebih besar dapat dilakukan untuk memperoleh hasil yang lebih optimal.

DAFTAR PUSTAKA

- Abid, M. R., & Munir, R. (2020). Enhancing data security using hybrid cryptosystem in cloud computing. *Journal of Cloud Computing: Advances, Systems and Applications*, 9(3), 23-31.
- Abdulraheem, M., Awotunde, J. B., Jimoh, R. G., & Oladipo, I. D. (2021). An efficient lightweight cryptographic algorithm for IoT security. In *Misra, S., & Muhammad-Bello, B. (Eds.), Information and Communication Technology and Applications. ICTA 2020. Communications in Computer and Information Science*, vol. 1350.
- Budiman, M. A., & Zarlis, M. (2021, June). Implementation of hybrid cryptosystem using Rabin-p algorithm and One Time Pad to secure images. In *Journal of Physics: Conference Series (Vol. 1898, No. 1, p. 012037)*. IOP Publishing.
- Chandra, S., Paira, S., Alam, S. S., & Sanyal, G. (2014). A comparative survey of symmetric and asymmetric key cryptography. In *2014 International Conference on Electronics, Communication and Computational Engineering (ICECCE)*.
- Diffie, W., & Hellman, M. E. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6), 644-654.
- Hasan, M. K., et al. (2021). Lightweight encryption technique to enhance medical image security on Internet of Medical Things applications. *IEEE Access*, 9, 47731–47742.
- Kadhim, A. N., & Manaa, M. E. (2022). Improving IoT data security using compression and lightweight encryption technique. In *2022 5th International Conference on Engineering Technology and Its Applications (IICETA), Al-Najaf, pp. 187–192*.
- Kahn, D. (1996). *The codebreakers: The comprehensive history of secret communication from ancient times to the internet*. Scribner.
- Koblitz, N. (1987). *A course in number theory and cryptography*. Springer.
- Kumar, P., & Singh, D. (2019). A review on hybrid cryptosystem for data security in cloud computing. *Journal of Information Security and Applications*, 47, 41-51.

- Lai, J.-F., & Heng, S.-H. (2022, September). Secure file storage on cloud using hybrid cryptography. *Journal of Informatics and Web Engineering*, 1(2), 1–18.
- Pooja, S., et al. (2023). Security and privacy in smart Internet of Things environments for well-being in the healthcare industry. In *Medical Information Processing and Security: Techniques and Applications*, pp. 307.
- Prakash, V., Singh, A. V., & Kumar Khatri, S. (2019). A new model of lightweight hybrid cryptography for Internet of Things. In *2019 3rd International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pp. 282-285.
- Ranasinghe, R., & Athukorala, P. (2022). A generalization of the ElGamal public-key cryptosystem. *Journal of Discrete Mathematical Sciences and Cryptography*, 25(8), 2395-2403.
- Rosen, K. H. (2011). *Discrete mathematics and its applications* (7th ed.). McGraw-Hill.
- Rupa, C., Greeshmanth, & Shah, M. A. (2023). Novel secure data protection scheme using Martino homomorphic encryption. *Journal of Cloud Computing*, 12, 47.
- Sasikumar, K., & Nagarajan, S. (2024). Comprehensive review and analysis of cryptography techniques in cloud computing. *IEEE Access*.
- Schneier, B. (1996). *Applied cryptography: Protocols, algorithms, and source code in C* (2nd ed.). Wiley.
- Sow, D., Robert, L., & Lafourcade, P. (2020). Linear generalized ElGamal encryption scheme. *Cryptology ePrint Archive*.
- Stallings, W. (2017). *Cryptography and network security: Principles and practice* (7th ed.). Pearson.
- Strang, G. (2009). *Introduction to linear algebra* (4th ed.). Wellesley-Cambridge Press.
- Zega, M. A., Budiman, M., & Syahril, M. (2023). Comparative analysis of ciphertext enlargement on generalization of the ElGamal and multi-factor RSA. *Data Science: Journal of Computing and Applied Informatics (JoCAI)*, 7(1), 44-50.