SISTEM KRIPTOKOMPRESI MENGGUNAKAN ALGORITMA ASIMETRIS RABIN-P DAN ALGORITMA LOSSLESS COMPRESSION GOLBACH CODE

SKRIPSI

Diajukan Untuk Memenuhi Persyaratan Meraih Gelar Sarjana Komputer
Pada Program Studi Teknik Informatika Fakultas Teknik dan Komputer
Universitas Harapan Medan

TRI JOKO WARDANI
182350238

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK DAN KOMPUTER
UNIVERSITAS HARAPAN MEDAN
MEDAN
2022

SISTEM KRIPTOKOMPRESI MENGGUNAKAN ALGORITMA ASIMETRIS RABIN-P DAN ALGORITMA LOSSLESS COMPRESSION GOLBACH CODE

TRI JOKO WARDANI
182350238

SKRIPSI

Diajukan Untuk Memenuhi Persyaratan Meraih Gelar Sarjana Komputer
Pada Program Studi Teknik Informatika Fakultas Teknik dan Komputer
Universitas Harapan Medan

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK DAN KOMPUTER
UNIVERSITAS HARAPAN MEDAN
September, 2022

PERNYATAAN PEMBIMBING

Saya/Kami dengan ini menyatakan bahwa saya/kami telah memeriksa skripsi Mahasiswa ini dan menurut pendapat saya/kami, Skripsi Mahasiswa ini telah mencukupi untuk ruang lingkup dan kualitas untuk dianugerahkan gelar Sarjana Teknik/Komputer dalam bidang Teknik Informatika.

Medan,30 September 2022

Pembimbing 1	Pembimbing 2
(Imran Lubis, S.T., M.Kom)	**(Ir. Kalvin Chiuloto, M.Kes)**

PERNYATAAN MAHASISWA

Saya yang bertanda tangan di bawah ini:

Nama	: Tri Joko Wardani
Nama Orang Tua	: Jaudin
Program Studi	: Teknik Informatika
Jenjang Studi	: Strata 1
NPM	: 182350238

Menyatakan bahwa:

1. Skripsi ini merupakan gagasan, rumusan dan ide saya sendiri, tanpa bantuan dari pihak lain kecuali arahan dari Tim Dosen Pembimbing.
2. Skripsi ini belum pernah diajukan untuk mendapat gelar sarjana, baik di Fakultas Teknik dan Komputer Universitas Harapan Medan maupun di Perguruan Tinggi lain.
3. Dalam Skripsi ini tidak terdapat karya atau pendapat yang ditulis atau dipublikasikan orang lain, kecuali secara tertulis dengan mencantumkan sebagai acuan dalam naskah dengan menyebut nama pengarang dan dicantumkan dalam daftar pustaka.

Demikianlah pernyataan ini saya perbuat dengan sesungguhnya dan apabila dikemudian hari terbukti pernyataan ini tidak benar, maka saya bersedia menerima sanksi akademik berupa pencabutan gelar yang telah diberikan melalui karya tulis ini, serta sanksi lainnya sesuai dengan norma yang berlaku di Perguruan Tinggi ini.

Medan,7 Oktober 2022
Yang Menyatakan

Tri Joko Wardani
182350238

PERSETUJUAN

JUDUL	**: SISTEM KRIPTOKOMPRESI MENGGUNAKAN ALGORITMA ASIMETRIS RABIN-P DAN ALGORITMA LOSSLESS COMPRESSION GOLBACH CODE**
KATEGORI	**: SKRIPSI**
NAMA	**: TRI JOKO WARDANI**
NOMOR POKOK MAHASISWA	**: 182350238**
FAKULTAS	**: TEKNIK DAN KOMPUER**
PROGRAM STUDI	**: TEKNIK INFORMATIKA**
TAHUN TAMAT	**: 2022**

DISETUJUI OLEH

Komisi Pembimbing

Pembimbing 1	Pembimbing 2
Imran Lubis, S.T., M.Kom	**Ir. Kalvin Chiuloto, M.Kes**

Mengetahui,
Ketua Program Studi Teknik Informatika

Ilham Faisal, S.T., M.Kom

KATA PENGANTAR

Puji dan syukur penulis ucapkan kepada Allah SWT atas limpahan berkat, rahmat, serta kemudahan yang telah diberikan sehingga peneliti dapat menyelesaikan Skripsi ini yang merupakan syarat untuk mendapatkan gelar Sarjana Teknik/Komputer pada Program Studi Teknik Informatika, Fakultas Teknik dan Komputer, Universitas Harapan Medan. Tidak lupa juga shalawat serta salam kepada Nabi Muhammad SAW.

Dalam kurun waktu pengerjaan Skripsi ini penulis menyadari bahwa sangat banyak pihak yang berjasa turut membantu penulis dalam penyelesaian Skripsi ini. Dalam kesempatan ini penulis mengucapkan terima kasih kepada :

1. Untuk kedua orang tua tercinta yang selama ini telah membantu peneliti dalam bentuk perhatian, kasih sayang, semangat, serta do'a demi kesuksesan peneliti dalam menyelesaikan skripsi ini.
2. Bapak Abdul Jabbar Lubis, S.T., M.Kom selaku Dekan Fakultas Teknik dan Komputer
3. Bapak Ilham Faisal, S.T., M.Kom. selaku Ketua Program Studi Teknik Informatika Fakultas Teknik dan Komputer.
4. Ibu Haida Dafitri, S.T., M.Kom selaku Sekretaris Program Studi Teknik Informatika
5. Bapak Imran Lubis, S.T., M.Kom selaku pembimbing 1 dan Bapak Ir. Kalvin Chiuloto, M.Kes selaku pembimbing 2 yang telah meluangkan waktu membimbing penulis selama pengerjaan Skripsi ini.
6. Teman-teman seperjuangan mahasiswa Program Studi Teknik Informatika Fakultas Teknik dan Komputer angkatan 2018 yang telah memberikan motivasinya.
7. Semua pihak yang terlibat baik langsung maupun tidak langsung dalam pengerjaan Skripsi ini yang tidak penulis sebutkan satu persatu diucapkan terima kasih.

Penulis menyadari bahwa Skripsi ini masih belum sempurna sehingga kritik dan saran yang membangun sangat penulis harapkan. Akhir kata penulis berharap semoga Skripsi ini dapat bermanfaat.

Medan,7 Oktober 2022
Penulis

Tri Joko Wardani
182350238

ABSTRAK

Sejalan dengan perkembangan teknologi informasi maka keamanan informasi juga harus diperhatikan. Kemajuan teknologi informasi memberikan banyak keuntungan bagi keberlangsungan kehidupan manusia, tetapi keuntungan yang ditawarkan oleh teknologi juga menimbulkan kerugian seperti pencurian data. Data yang berasal dari dokumen pengolah kata menjadi salah satu bentuk dokumen yang umum digunakan untuk menyimpan informasi, baik yang sifatnya pribadi atau rahasia. Masalah keamanan merupakan salah satu aspek paling penting dalam dunia teknologi informasi yang menjadi kebutuhan manusia pada era globalisasi, misalnya keamanan dokumen. Oleh karena itu, keamanan dokumen sangat diperlukan dan harus dilindungi dari pihak-pihak yang tidak bertanggung jawab, baik saat disimpan pada ruang penyimpanan (storage) ataupun pada saat didistribusikan. Selain aspek keamanan, maka hal yang perlu diperhatikan juga adalah tentang media penyimpanan. Oleh karena itu, diperlukan langkah tambahan untuk mengefisiensikan media penyimpanan dengan melakukan kompresi pada data yang sudah diamankan supaya ukurannya menjadi lebih kecil. Dalam penelitian ini, akan menggabungkan teknik kriptografi dan teknik kompresi yang disebut dengan kriptokompresi. Sistem kriptokompresi dalam penelitian diterapkan dengan cara mengenkripsi data pada file text menggunakan kunci publik dari algoritma Rabin-p yang akan menghasilkan file text terenkripsi. Selanjutnya dikompresi menggunakan algoritma lossless compression Goldbach Code untuk memperkecil ukurannya. Dari hasil pengujian disimpulkan bahwa sistem kriptokompresi mampu menjaga kerahasiaan sebuah informasi karena telah dienkripsi serta dapat menghemat ruang penyimpanan yang lebih efisien karena sudah dikompresi terlebih dahulu. File text yang telah dienkripsi dan dikompresi dapat dikembalikan lagi kedalam file aslinya melalui proses dekompresi dan dekripsi sesuai dengan size dan jumlah karater yang terdapat pada file text aslinya.

Kata Kunci : *Kriptografi, Kompresi, Kriptokompresi, Rabin-p, Goldbach Code*

ABSTRACT

The in line line with the development of information technology, information security must also be considered. Advances in information technology provide many advantages for the survival of human life, but the advantages offered by technology also cause losses such as data theft. Data originating from word processing documents is one form of document that is commonly used to store information, whether private or confidential. The security issues are one of the most important aspects in the world of information technology which is a human need in the era of globalization, for example document security. Therefore, document security is very necessary and must be protected from irresponsible parties, either when stored in storage or when distributed. In addition to the security aspect, the thing that needs to be considered is also about the storage media. Therefore, additional steps are needed to streamline the storage media by compressing the data that has been secured so that its size becomes smaller. In this study, we will combine cryptographic techniques and compression techniques called cryptocompression. The cryptocompression system in this research is applied by encrypting data in text files using the public key of the Rabin-p algorithm which will produce an encrypted text file. Then it is compressed using Goldbach Code lossless compression algorithm to reduce its size. From the results of this test, it is concluded that the cryptocompression system is able to maintain the confidentiality of information because it has been encrypted and can save more efficient storage space because it has been compressed first. Text files that have been encrypted and compressed can be returned to the original file through a decompression and decryption process according to the size and number of characters contained in the original text file.

Keyword : *Cryptography, Compression, Cryptocompression, Rabin-p, Goldbach Code*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Teknologi informasi meliputi segala hal yang berkaitan dengan proses, penggunaan sebagai alat bantu, manipulasi, dan pengelolaan informasi. Sedangkan teknologi komunikasi adalah segala sesuatu yang berkaitan dengan penggunaan alat bantu untuk memproses dan mentransfer data dari perangkat yang satu ke lainnya (Huda, 2020). Sehingga teknologi informasi dan komunikasi merupakan dua buah konsep yang tidak terpisahkan yang mencakup semua peralatan teknis untuk memproses dan menyampaikan informasi. Teknologi komunikasi ditekankan pada bagaimana suatu hasil data dapat disampaikan ke tempat tujuan sedangkan teknologi informasi lebih ditekankan pada hasil data yang diperoleh. Teknologi informasi berkembang cepat seiring meningkatnya perkembangan komputer beserta perangkat-perangkat pendukung lainnya serta teknologi komunikasi berkembang cepat dengan meningkatnya perkembangan teknologi elektronika dan sistem transmisi sehingga suatu informasi dapat disampaikan dengan cepat dan tepat.

Sejalan dengan perkembangan informasi maka keamanan informasi juga harus diperhatikan. Kemajuan teknologi informasi memberikan banyak keuntungan bagi keberlangsungan kehidupan manusia, tetapi keuntungan yang ditawarkan oleh teknologi juga menimbulkan kerugian seperti pencurian data. Sehingga perkembangan ilmu untuk mengamankan data semakin ditingkatkan agar pengguna teknologi selalu merasa aman. Berbagai cara dilakukan untuk menjaga keamanan data seperti menyembunyikan data (teknik steganografi) dan penyandian data menjadi suatu kode-kode yang tidak dimengerti (teknik kriptografi), sehingga apabila dicuri atau disadap oleh orang lain akan kesulitan untuk mengetahui dan memahami informasi yang sebenarnya.

Pada zaman dahulu tepatnya Romawi Kuno, Julius Caesar telah menggunakan teknik kriptografi yang kemudian dijuluki *Caesar Cipher* untuk mengirim pesan secara rahasia. Kejadian yang sama juga terjadi pada perang dunia kedua, yaitu pihak sekutu mampu memecahkan kode mesin kriptografi Jerman yang disebut dengan *Enigma*. Keberhasilan tersebut banyak membantu pihak sekutu dalam memenangkan pertempuran. Pada dasarnya kriptografi dipahami sebagai ilmu tentang menyembunyikan pesan, tetapi seiring perkembangan zaman hingga saat ini pengertian kriptografi berkembang menjadi ilmu

tentang teknik matematis yang digunakan untuk menyelesaikan persoalan keamanan berupa privasi dan otentikasi. Teknik kriptografi merupakan salah satu alternatif solusi yang dapat diterapkan untuk menjaga keamanan data, yaitu dengan cara memanipulasi data ke dalam bentuk yang tidak dimengerti oleh banyak orang. Terdapat dua jenis algoritma kriptografi berdasarkan jenis kuncinya, yaitu algoritma simetris (konvensional) dan algoritma asimetris (kunci publik). Algoritma simetris adalah algoritma kriptografi yang menggunakan satu kunci untuk proses enkripsi dan dekripsi, sedangkan algoritma asimetris, kunci terbagi menjadi dua bagian yaitu kunci umum (*public key*) untuk proses enkripsi dan kunci pribadi (*private key*) untuk proses dekripsi. Dalam penelitian yang dilakukan oleh Laurentinus et al (2020) menyimpulkan bahwa algoritma simetris AES mempunyai kinerja waktu enkripsi dan dekripsi yang lebih baik daripada algoritma asimetris RSA seiring dengan besarnya ukuran karakter yang dienkripsi. Namun, kompresi algoritma *Huffman* terhadap *ciphertext* pada algoritma asimetris RSA lebih efisien daripada AES. Dari hasil penelitian tersebut dapat disimpulkan bahwa penerapan kompresi lebih efisien jika diterapkan pada algoritma kriptografi asimetris dibandingkan dengan algoritma simetris.

Penelitian terdahulu yang relevan dengan pokok permasalahan dalam penelitian ini dijadikan sebagai data pendukung. Penelitian terdahulu mengenai algoritma *Rabin-p* seperti yang dilakukan oleh Asbullah et al (2016), menyimpulkan bahwa algoritma *Rabin-p* bekerja lebih cepat dan menggunakan lebih sedikit penyimpanan dibandingkan dengan kriptosistem mirip *Rabin* lainnya. Penelitian lainnya juga dilakukan oleh Hidayat (2021), hasil penelitian dengan menggunakan lima sampel uji *file text* berformat *.txt* menyimpulkan bahwa algoritma *Rabin-p* lebih efisien dibandingkan algoritma RSA-CRT berdasarkan pengukuran waktu eksekusi komputasi dan kompleksitas algoritma. Penelitian terdahulu mengenai algoritma *Goldbach Code* seperti yang dilakukan oleh Yogie (2018), hasil yang dicapai dalam penelitian tersebut djelaskan bahwa algoritma *Goldbach Code* merupakan algoritma pengkompresian *file* yang cukup handal serta dapat menghemat space penyimpanan. Penelitian lainnya juga dilakukan oleh Tanjung and Nasution (2020), menyimpulkan bahwa hasil kompresi pada *file text* menggunakan *Goldbach Code* sangat bagus dan hasilnya tidak berkurang dari file asli atau tidak ada pengurangan.

File .txt merupakan dokumen teks standar yang berisi teks yang tidak diformat dan berguna untuk menyimpan informasi dalam teks biasa. *File .txt* dapat diproses oleh sebagian besar program perangkat lunak lain seperti *Notepad*. Sedangkan *file .docx* merupakan

dokumen yang berisi teks yang di format dan dapat diproses menggunakan perangkat lunak *Microsoft Word*. Data yang berasal dari dokumen pengolah kata seperti *Notepad* dan *Microsoft Word* menjadi salah satu bentuk dokumen yang umum digunakan untuk menyimpan informasi, baik yang sifatnya tidak penting maupun yang sifatnya pribadi atau rahasia. Oleh karena itu, keamanan dokumen sangat diperlukan dan harus dilindungi dari gangguan maupun serangan yang terjadi dari pihak-pihak yang tidak bertanggung jawab, baik saat disimpan pada ruang penyimpanan (*storage*) ataupun pada saat didistribusikan. Banyak hal yang dapat dilakukan untuk meningkatkan keamanan dokumen, salah satunya dengan menyandikan isi pesan menjadi suatu kode-kode yang tidak dimengerti atau yang sering dikenal dengan teknik kriptografi. Masalah keamanan merupakan salah satu aspek paling penting dalam dunia teknologi informasi yang menjadi kebutuhan manusia pada era globalisasi, misalnya keamanan dokumen. Dalam penelitian ini, akan menggabungkan teknik kriptografi dan teknik kompresi yang disebut dengan kriptokompresi menggunakan algoritma asimetris *Rabin-p* dan algoritma *lossless compression Goldbach Code*. Kriptokompresi dalam penelitian ini merupakan teknik yang memanfaatkan dua metode kedalam satu proses, artinya data yang akan diamankan yaitu berupa *file text* akan dienkripsi terlebih dahulu dengan algoritma asimetris *Rabin-p* untuk tujuan mengamankan data, kemudian dikompresi menggunakan algoritma lossless compression *Goldbach Code* guna memperkecil ukuran data. Dari penjabaran yang telah dipaparkan, maka penulis akan melakukan penelitian yang dituangkan dalam bentuk tugas akhir dengan judul penelitian "Sistem Kriptokompresi Menggunakan Algoritma Asimetris *Rabin-p* dan Algoritma *Lossless Compression Goldbach Code*".

1.2 Rumusan Masalah

Pada latar belakang yang telah dipaparkan penulis, maka penulis menyimpulkan rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana menerapkan sistem kriptokompresi dalam meningkatkan keamanan data serta memperkecil ukurannya dengan menggunakan algoritma asimetris *Rabin-p* dan algoritma *lossless compression Goldbach Code*.
2. Bagaimana hasil dokumen yang dienkripsi dari algoritma *Rabin-p* dan *Goldbach Code* mampu bertahan dalam menjaga keamanan kerahasiaan data.

1.3 Batasan Masalah

Berdasarkan uraian pada rumusan masalah diatas, agar tidak terjadi penyimpangan terhadap tujuan yang diharapkan dalam penelitian ini maka dibuat beberapa pembatasan masalah yaitu sebagai berikut:

1. *Input* data yang akan dienkripsi dan dikompresi hanyalah berupa karakter dalam *file text* yang berekstensi *.txt* dan *.docx*. Sedangkan *output* data berupa *file text* terkompresi yang akan disimpan dalam file berekstensi *.gc* (akronim dari algoritma kompresi *Goldbach Code*)
2. Alur kerja sistem pada penelitian ini dimulai dari *file text* yang akan dienkripsi terlebih dahulu dengan algoritma *Rabin-p,* kemudian dikompresi menggunakan algoritma *Goldbach Code*. Selanjutnya akan dilakukan proses dekompresi lalu didekripsi untuk mengembalikan *file text* aslinya.
3. Tidak melakukan enkripsi dan dekripsi data terhadap komponen lain seperti tabel atau gambar yang terdapat di dalam *file text* serta tidak membahas bagaimana proses pengiriman data.
4. Parameter yang digunakan untuk proses enkripsi dan dekripsi adalah *Running Time* dalam satuan *millisecond* (ms).
5. Parameter yang digunakan dalam proses kompresi pada penelitian ini adalah *Compression Ratio, Space Savings* serta waktu kompresi dan dekompresi.
6. Implementasi dari penelitian menggunakan bahasa pemrograman *Microsoft Visual C#.NET*.

1.4 Tujuan Penelitian

Berdasarkan rumusan masalah yang telah diuraikan sebelumnya, maka tujuan dari penelitian ini yaitu sebagai berikut:

1. Untuk meningkatkan kerahasiaan sebuah informasi yang terdapat didalam data *file text* serta memperkecil ukurannya dengan menerapkan sistem kriptokompresi menggunakan algoritma asimetris *Rabin-p* dan algoritma *lossless compression Goldbach Code*.
2. Untuk menguji hasil perbandingan data pada *file text* sebelum dan sesudah diproses dengan sistem kriptokompresi dengan pendekatan parameter *Compression Ratio, Space Savings* serta *Running Time*.

1.5 Manfaat Penelitian

Hasil dari penelitian ini diharapkan dapat memberikan manfaat bagi peneliti dan pembaca. Adapun manfaat yang diharapkan dari hasil penelitian ini adalah sebagai berikut:

1. Dengan menerapkan sistem kriptokompresi menggunakan algoritma asimetris *Rabin-p* dan algoritma *lossless compression Goldbach Code*.dapat meningkatkan keamanan data karena telah disandikan serta menghemat ruang penyimpanan yang lebih efisien karena sudah dikompresi terlebih dahulu.
2. Dengan menerapkan sistem kriptokompresi menggunakan algoritma asimetris *Rabin-p* dan algoritma *lossless compression Goldbach Code* dapat menjaga kerahasiaan sebuah informasi yang terdapat didalamnya sehingga lebih aman dalam proses pertukaran data serta mempercepat proses pengiriman data pada saat ditransmisikan.

1.6 Metodologi Penelitian

Dalam mendukung jalannya penelitian ini agar lebih terarah dan sistematis maka dibutuhkan suatu tahapan desain penelitian yang disusun dengan terstruktur. Penelitian ini menerapkan beberapa metode penelitian yang dapat dijelaskan yaitu:

1. Identifkasi Masalah

   Pada tahap ini penulis mengidentifikasi apa-apa saja yang menjadi permasalahan dan mengambil permasalahan tersebut menjadi topik penelitian sehingga peneliti dapat mencari solusi yang nantinya akan menjadi tujuan dari penelitian ini. Permasalahan yang diangkat dalam penelitian ini terkait dalam hal meningkatkan keamanan data pada *file text* serta bagaimana menghemat kebutuhan akan ruang penyimpanan (*storage*) data menjadi lebih efisian. Dalam penelitian ini, akan menggabungkan teknik kriptografi dan teknik kompresi yang disebut dengan sistem kriptokompresi sebagai alternatif solusi dari permasalahan tersebut.
2. Studi Pustaka

   Pada tahap ini bertujuan untuk mendapatkan landasan teori mengenai permasalahan yang akan diteliti sehingga dapat memahami permasalahan yang diteliti sesuai dengan pembahasan yang dilakukan. Sumber pustaka pada penelitian ini diperoleh dengan

membaca berbagai literatur seperti buku dan jurnal atau hasil kajian dari penelitian terdahulu.

3. Analisis dan Perancangan Sistem

   Pada tahap ini dilakukan analisis penerapan algoritma kriptografi asimetris *Rabin-p* dan kompresi *Goldbach Code* dalam skema sistem kriptokompresi. Terdapat dua proses utama yang dilakukan dalam tahap ini, yaitu enkripsi dan kompresi pada tahap pertama yang bertujuan untuk mentransformasi *file text* menjadi *ciphertext* dengan menggunakan kunci algoritma *Rabin-p*, selanjutnya akan dikompresi lagi dengan menggunakan kompresi *Goldbach Code* sehingga ukuran datanya menjadi lebih kecil. Proses kedua yaitu dekompresi dan dekripsi yang bertujuan untuk mengembalikan data ke bentuk aslinya. Sedangkan Perancangan sistem dilakukan dengan membuat *flowchart* sistem dan tampilan *interface* (antarmuka) sistem yang akan di integrasikan dengan aplikasi pada tahap implementasi sistem.

4. Implementasi dan Pengujian Sistem

   Pada tahap ini akan dilakukan pengkodean (*coding*) menggunakan bahasa pemrograman *Microsoft Visual C#.NET* yang mengacu pada perancangan sistem yang telah dibuat sebelumnya. Implementasi sistem dilakukan dengan menampilkan ke *user* (pengguna) mengenai hasil aplikasi berbasis *desktop* tentang sistem kriptokompresi yang telah dibuat. Sedangkan tahap pengujian dilakukan dengan tujuan untuk menjamin sistem yang dibuat sesuai dengan hasil analisis dan perancangan serta menghasilkan satu kesimpulan apakah sistem tersebut sesuai dengan yang diharapkan.

5. Kesimpulan Penelitian

   Pada tahap ini diambil kesimpulan yang menjawab tujuan akhir dari penelitian berdasarkan hasil analisis sampai pengujian sistem yang telah dilakukan.

1.7 Sistematika Penulisan

Sistematika penulisan yang dipergunakan penulis dalam penyusunan skripsi ini terdiri dari lima bab, yaitu sebagai berikut:

BAB 1 : PENDAHULUAN

Bab ini menguraikan mengenai latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan sistematika penulisan.

BAB 2 : DASAR TEORI

Pada bab ini dijelaskan terkait teori-teori singkat yang berkaitan dengan penelitian seperti kriptografi, algoritma *Rabin-p*, kompresi data, dan algoritma *Goldbach Code*.

BAB 3 : ANALISIS DAN PERANCANGAN SISTEM

Pada bab ini menjelaskan tentang analisis pada masalah penelitian serta perancangn sistem yang akan dibangun sebagai solusi dari masalah tersebut.

BAB 4 : IMPLEMENTASI DAN PENGUJIAN SISTEM

Bab ini membahas implementasi dan pengujian yang dilakukan terhadap aplikasi sistem dan kasus-kasus uji terhadap program.

BAB 5 : KESIMPULAN DAN SARAN

Bab ini menjelaskan simpulan hasil penelitian yang selaras dengan rumusan masalah dan saran-saran yang dapat mengembangkan atau melanjutkan penelitian ini menjadi lebih baik lagi.

BAB 2
DASAR TEORI

2.1 Kriptografi

Kemajuan teknologi informasi telah memberikan dampak yang sangat luas, salah satunya sebagai media penyampaian informasi dari suatu tempat ke tempat lainnya, sehingga memudahkan orang dalam mengakses suatu informasi. Kemudahan pengaksesan media komunikasi oleh semua orang, tentunya akan memberikan dampak bagi keamanan informasi atau pesan yang menggunakan media komunikasi tersebut. Informasi menjadi sangat rentan untuk diketahui, diambil dan dimanipulasi oleh pihak-pihak yang tidak bertanggung jawab. Oleh sebab itu dibutuhkan suatu metode atau cara yang dapat menjaga kerahasiaan informasi ini, yang salah satunya dikenal dengan sebutan kriptografi.

Kata kriptografi berasal dari bahasa Yunani, "*cryptós*" yang berarti tersembunyi dan "*gráphein*" yang berarti tulisan (Meko, 2018). Teknik penulisan pesan rahasia ini digunakan oleh bangsa mesir sekitar 3000 tahun sebelum masehi. Penulisan rahasia ini disebut *hieroglyphcs* dimana mereka (bangsa mesir kuno) menyembunyikan tulisan supaya tidak dapat diketahui oleh pihak yang tidak diharapkan. Kriptografi tidak hanya mengenai penyembunyian pesan atau tulisan tetapi lebih pada sekumpulan teknik matematika untuk keamanan informasi yang bersifat kerahasiaan (Darwis et al, 2018).

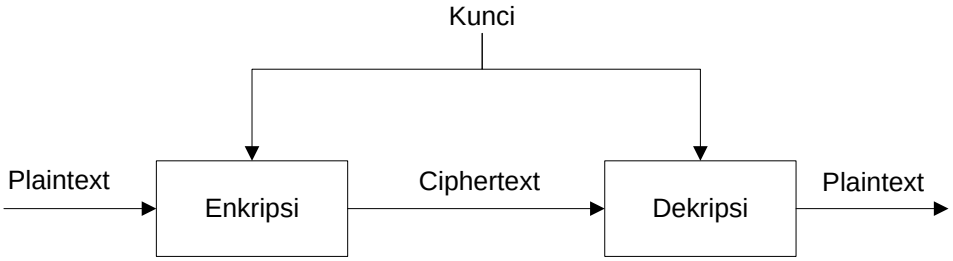
Kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi, seperti kerahasiaan data, keabsahan data, integritas data, serta autentikasi data (Hasugian, 2017). Kriptografi bertujuan untuk menjaga kerahasiaan informasi yang tergantung dalam data sehingga informasi tersebut tidak dapat diketahui oleh pihak yang tidak sah. Kriptografi merupakan bagian dari suatu cabang ilmu matematika yang disebut dengan kriptologi (*cryptology*) (Yogie, 2018).

Dalam ilmu kriptografi, terdapat dua buah proses yaitu melakukan enkripsi dan dekripsi. Enkripsi adalah suatu proses yang dilakukan untuk mengubah pesan asli (*plaintext*) menjadi pesan tersembunyi (*ciphertext*). Sedangkan suatu proses yang dilakukan untuk mengubah pesan tersembunyi menjadi pesan biasa (yang mudah dibaca) disebut dekripsi (Saputro et al, 2020). Berdasarkan sifat kuncinya, kriptografi dibagi menjadi dua, yaitu kriptografi simetris dan kriptografi asimetris. Pada kriptografi simetris, proses enkripsi dan dekripsi dilakukan menggunakan kunci rahasia yang sama. Sedangkan pada kriptografi

asimetris, proses enkripsi dan dekripsinya menggunakan kunci yang berbeda, yaitu kunci publik untuk enkripsi, dan kunci rahasia yang digunakan untuk dekripsi (Hasugian, 2017).

Penerapan kriptografi saat ini telah menjadi salah satu syarat penting dalam keamanan teknologi informasi terutama dalam pengiriman pesan rahasia. Pengiriman pesan rahasia sangat rentan terhadap serangan yang dilakukan oleh pihak ketiga, seperti penyadapan, pemutusan komunikasi, dan pengubahan pesan yang dikirim. Kriptografi dapat meningkatkan keamanan dalam pengiriman pesan atau komunikasi data dengan cara menyandikan pesan tersebut berdasarkan algoritma dan kunci tertentu yang hanya diketahui oleh pihak-pihak yang berhak atas data, informasi dan dokumen tersebut.

Algoritma kriptografi simetris disebut simetris karena memiliki *key* atau kunci yang sama dalam proses enkripsi dan dekripsi sehingga algoritma ini juga sering disebut algoritma kunci tunggal atau satu kunci. *Key* dalam algoritma ini besifat rahasia atau private *key* sehingga algoritma ini juga disebut dengan algoritma kunci rahasia (Meko, 2018). Kriptografi kunci simetris mengarah kepada metode enkripsi yang mana baik pengirim maupun yang dikirim saling memiliki kunci yang sama. Kriptografi kunci simetris merupakan bentuk kryptografi yang lebih tradisional, dimana sebuah kunci tunggal dapat digunakan untuk mengenkrip dan mendekrip pesan (Hasugian, 2017). Adapun skema algoritma kriptografi simetris dapat disajikan pada gambar 2.1.

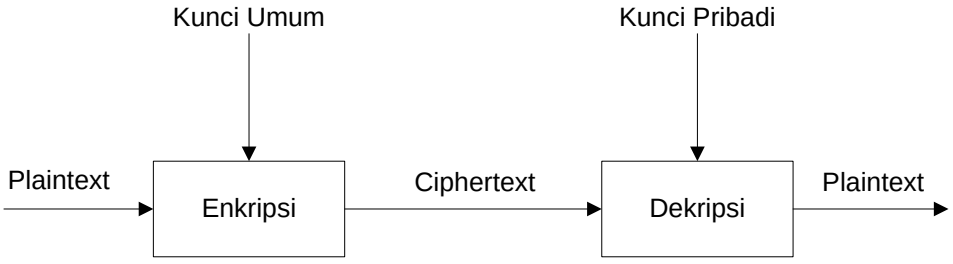


Gambar 2.1 Skema Algoritma Kriptografi Simetris
Sumber : Putri et al, 2018

Gambar 2.1 memperlihatkan algoritma kriptografi simetris yang biasa disebut juga sebagai kriptografi kunci konvensional. Dalam algoritma simetris, pengirim data dan penerima data menggunakan kunci yang sama. Jika kunci untuk proses enkripsi diketahui maka kunci untuk proses dekripsi dapat diperoleh karena menggunakan kunci yang sama. Oleh karena itu keamanan dari algoritma kriptografi simetris terletak pada kuncinya. Dalam penelitian yang dilakukan oleh Laurentinus et al (2020) menyimpulkan bahwa algoritma

simetris mempunyai kinerja waktu enkripsi dan dekripsi yang lebih baik daripada algoritma asimetris seiring dengan besarnya ukuran karakter yang dienkripsi.

Algoritma kriptografi asimetris (*Asymmetric* atau *Public key*) adalah algoritma yang menggunakan kunci yang berbeda untuk proses enkripsi dan dekripsi (Saputro, 2020). Sistem ini memiliki sepasang kunci yang disebut kunci publik yaitu kunci yang didistribusikan secara umum dan kunci privat yaitu kunci yang dirahasiakan yang hanya dimiliki oleh pihak yang berhak. Umumnya kunci publik digunakan untuk menyandi dan kunci privat digunakan untuk membuka sandi. Sistem sandi asimetrik bekerja lebih lambat dari sistem sandi simetris, sehingga sistem sandi ini lebih sering digunakan untuk menyandi data dengan ukuran bit yang kecil (Hasugian, 2017). Adapun skema algoritma kriptografi asimetris dapat disajikan pada gambar 2.2.



Gambar 2.2 Skema Algoritma Kriptografi Asimetris
Sumber : Putri et al, 2018

Gambar 2.2 memperlihatkan algoritma kriptografi asimetris dimana kunci terbagi menjadi dua bagian, yaitu kunci umum (*public key*) untuk proses enkripsi dan kunci pribadi (*private key*) untuk proses dekripsi. Dalam penelitian yang dilakukan oleh Laurentinus et al (2020) menyimpulkan bahwa algoritma simetris mempunyai kinerja waktu enkripsi dan dekripsi yang lebih baik daripada algoritma asimetris seiring dengan besarnya ukuran karakter yang dienkripsi. Namun, kompresi algoritma terhadap *ciphertext* algoritma asimetris lebih efisien daripada algoritma simetris. Dari hasil penelitian tersebut dapat disimpulkan bahwa penerapan kompresi lebih efisien jika diterapkan pada algoritma kriptografi asimetris dibandingkan dengan algoritma simetris. Algoritma kriptografi asimetris lebih kuat keamanannya dibanding dengan algoritma simetris (Saputro, 2020).

2.2 Algoritma *Rabin-p*

Algoritma *Rabin-p* merupakan varian dari algoritma *Rabin* yang dirancang oleh M. A. Asbullah dan M. R. K. Ariffin. Salah satu keuntungan yang jelas dari algoritma ini adalah

bahwa prosedur dekripsi dari algoritma *Rabin-p* hanya menghasilkan satu hasil yaitu pesan asli, sehingga tidak ada lagi kebingungan di pihak penerima tentang mengetahui pesan asli dari empat hasil dekripsi seperti pada algoritma Rabin (Budiman, et al, 2020). Algoritma *Rabin-p* bekerja lebih cepat dan menggunakan lebih sedikit penyimpanan dibandingkan dengan dua kriptosistem mirip Rabin lainnya, yaitu Rabin Takagi (Asbullah, et al, 2016).

Algoritma *Rabin-p* dinamai Rabin dengan tambahan *p* yang melambangkan bahwa skema yang diusulkan hanya menggunakan satu bilangan prima *p* sebagai kunci dekripsi. Berikut ini akan dijelaskan prosedur pembuatan kunci, enkripsi, dan dekripsi algoritma *Rabin-p* (Budiman, et al, 2020), yaitu sebagai berikut:

1. Pembangkitan Kunci Algoritma *Rabin-p*

   Untuk mengenkripsi dan dekripsi pesan dengan menggunakan algoritma *Rabin-p* terlebih dahulu membangkitkan sepasang kunci, yaitu kunci publik (*public key*) dan kunci privat (*private key*). Berikut ini algoritma penyelesaiannya yaitu:

   1). Tentukan k sebagai parameter keamanan.

   2). Pilih dua buah bilangan acak prima untuk nilai p dan q dimana $2^{\mathrm{k}} < \mathrm{p}, \mathrm{q} < 2^{\mathrm{k}+1}$ memenuhi $\mathrm{p} \equiv \mathrm{q} \equiv 3 (\mathrm{mod}\ 4)$.

   3). Hitung $\mathrm{N} = \mathrm{p}^2\mathrm{q}$

   4). Publikasikan N sebagai kunci publik (*public key*) untuk enkripsi

   5). Kunci dekripsi (*private key*) adalah nilai p dan sifatnya rahasia

2. Proses Enkripsi Algoritma *Rabin-p*

   Pengamanan data berdasarkan algoritma teknik kriptografi dilakukan dengan merubah pesan yang akan dirahasiakan (*plaintext*) menjadi sandi (*ciphertext*). Proses untuk mengkonversi *plaintext* menjadi *ciphertext* disebut dengan proses enkripsi. Adapun proses enkripsi pesan (*plaintext*) dengan menggunakan algoritma *Rabin-p* dapat dijelaskan tahapannya sebagai berikut:

   1). Langkah pertama ambil kunci publik (*public key*) algoritma *Rabin-p* yang telah dibangkitkan yaitu N dan parameter keamanan nilai k

   2). Pilih *plaintext* $0 < \mathrm{m} < 2^{2\mathrm{k}-1}$ dimana $\mathrm{GCD}(\mathrm{m}, \mathrm{N}) = 1$

   Dalam hal ini m adalah *plaintext* yang terdapat dalam *file text*.

   3). Hitung $\mathrm{c} \equiv \mathrm{m}^2 (\mathrm{mod}\ \mathrm{N})$

   Dalam hal ini c adalah hasil enkripsi *file text* (*ciphertext*).

3. Proses Dekripsi Algoritma *Rabin-p*

   Proses untuk mengkonversi *ciphertext* menjadi *plaintext* disebut dengan proses dekripsi. Proses dekripsi algoritma *Rabin-p* dengan menggunakan kunci privat dapat dapat dijelaskan tahapannya sebagai berikut:

   1). Langkah pertama ambil hasil enkripsi (*ciphertext*) yaitu c
   2). Langkah kedua ambil kunci privat (*private key*) *Rabin-p* yaitu p
   3). Hitung nilai $w \equiv c \bmod p$
   4). Hitung nilai $m_p \equiv w^{\frac{p+1}{4}} \bmod p$
   5). Hitung nilai $i = \frac{c - m_p^2}{p}$
   6). Hitung nilai $j \equiv \frac{i}{2m_p} \bmod p$
   7). Hitung nilai $m_1 = m_p + jp$
   8). Jika $m_1 < 2^{2k-1}$ maka $m = m_1$
   9). Selain itu, $m = p^2 - m_1$

Dasar penulis memilih algoritma ini adalah berdasarkan hasil penelitian terdahulu seperti yang dilakukan oleh Hidayat (2021), hasil penelitian dengan menggunakan lima sampel uji file teks berformat .txt menyimpulkan bahwa algoritma *Rabin-p* lebih efisien dibandingkan algoritma RSA-CRT berdasarkan pengukuran waktu eksekusi komputasi dan kompleksitas algoritma.

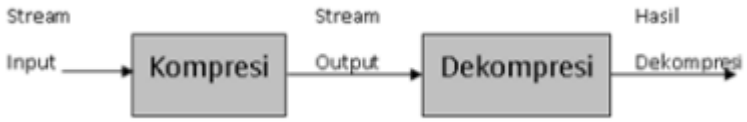
2.3 Kompresi Data

Kompresi data (data compression) adalah teknik untuk memperkecil jumlah ukuran data (hasil kompresi) dari data asli (Tanjung and Nasution, 2020). Kompresi data adalah sebuah cara untuk memadatkan data sehingga hanya memerlukan ruang penyimpanan lebih kecil untuk dapat lebih efisien dalam penyimpanan atau mempersingkat waktu pertukaran data dalam memaksimalkan *space* ataupun *storage*. Kegunaan metode ini sendiri muncul dikarenakan file yang semakin lama semakin besar sehingga menyebabkan banyak informasi yang harus ditampung oleh sebuah data (Siregar et al, 2020).

Dalam penerapan didalam komputer, kompresi data adalah suatu teknik untuk memadatkan data sehingga hanya memerlukan ruang penyimpanan lebih kecil sehingga

sangat lebih efisien dalam menyimpan atau mempersingkat waktu pada saat pertukaran data. Dekompresi merupakan proses mengembalikan data yang sudah terkompresi sebelumnya, data yang sudah dikompres dapat dikembalikan lagi ke dalam bentuk semula. Untuk dapat megembalikan data yang terkompres sebelumnya dapat dilakukan dengan cara yang berbeda seperti pada waktu proses kompres dilakukan. Jadi pada proses dekompres terdapat karakter *header* yang berupa *byte-byte* yang berisi karakter mengenai isi dari file tersebut (Pramadi et al, 2019).

Dengan merujuk pada beberapa definisi diatas, maka dapat disimpulkan bahwa kompresi data merupakan suatu cara untuk memadatkan data agar hanya memerlukan ruang penyimpanan lebih kecil sehingga lebih efisien dalam penyimpanan maupun dalam proses transmisi data. Gambar 2.3 berikut adalah proses kompresi dan dekompresi.

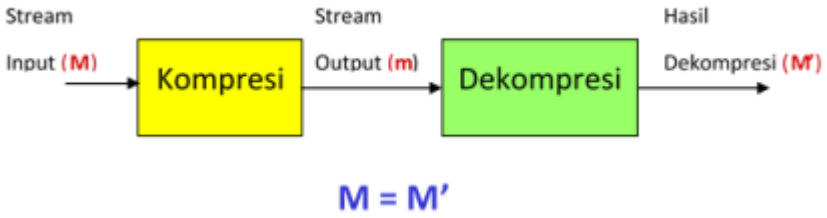


Gambar 2.3 Proses Kompresi dan Dekompresi
Sumber : Mangiri, 2018

Permasalahan mendasar dalam proses kompresi data adalah bagaimana cara mengkompres data (khususnya data teks) yaitu mendapatkan *file text* yang ukurannya lebih kecil dari ukuran aslinya, kemudian merekonstruksinya kembali ke data aslinya (dekompresi). Ada beberapa faktor yang sering menjadi pertimbangan dalam memilih suatu metode kompresi yang tepat, yaitu kecepatan kompresi, sumber daya yang dibutuhkan (memori dan kecepatan PC), ukuran file hasil kompresi, dan kompleksitas algoritma (Siregar et al, 2020). Algoritma kompresi yang digunakan dalam penelitian ini adalah algoritma *Goldbach Code*. Dasar penulis memilih algoritma ini adalah berdasarkan hasil penelitian terdahulu seperti yang dilakukan oleh Laswi (2019), menyimpulkan bahwa algoritma *Goldbach Code* cocok digunakan untuk mengoptimalkan *file text* sehingga dapat memperkecil penggunaan memori.

Pada kompresi data terdapat dua buah tipe teknik kompresi yang pertama teknik kompresi metode *lossless* (*Lossless Compression*) dan yang kedua adalah teknik kompresi metode *lossy* (*Lossy Compression*) (Mangiri, 2018). Perbedaan utama antara kompresi lossless dan lossy terletak pada kualitasnya.

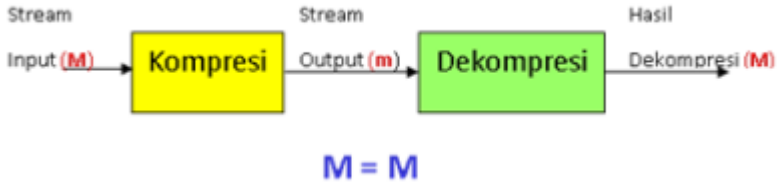
Lossy compression yaitu suatu metode kompresi data yang menghilangkan sebagian "informasi" dari data asli selama proses kompresi dengan tidak menghilangkan secara signifikan informasi yang ada dalam data secara keseluruhan. Kompresi jenis ini dapat mengurangi ukuran data secara signifikan, dan data yang sudah terkompresi dapat dikompres lagi sampai batas-batas tertentu (Pangesti et al, 2020). Melalui metode ini, stream input informasi yang dikompresi akan mempunyai kapasitas yang hampir sama dengan stream output informasi setelah melalui proses dekompresi ($M = M'$) (Mangiri, 2018) sebagaimana ditunjukkan pada gambar 2.4 di bawah ini.



Gambar 2.4 Blok Diagram Metode Lossy Compression
Sumber : Mangiri, 2018

Lossy compression biasanya digunakan dalam bidang multimedia karena sering digunakan untuk mengurangi ukuran data yang bersifat audio–visual. Misalnya untuk memperkecil ukuran file gambar, musik digital, dan untuk *encoding* film dari format *High Definition* seperti *Blu-Ray* menjadi format MPEG (*Moving Picture Experts Group*). Contoh metode *lossy compression* adalah *discrete cosine transform*, *vector quantization*, *waveled compression*, dan *distributed source coding* (Pangesti et al, 2020).

Lossless compression yaitu suatu metode kompresi data dengan tidak ada "informasi" data yang hilang atau berkurang jumlahnya selama proses kompresi. Sehingga setelah proses dekompresi jumlah *bit* (*byte*) data atau informasi dalam keseluruhan data hasil sama persis dengan data aslinya. Kompresi jenis ini tidak selalu dapat mengurangi ukuran data secara berarti, karena tidak semua data mengandung informasi yang tidak perlu. Selain itu, data yang telah dikompresi tidak dapat dikompresi lagi (Pangesti et al, 2020). Melalui metode ini, stream input informasi yang dikompresi akan mempunyai kapasitas yang sama dengan stream output informasi setelah melalui proses dekompresi ($M = M$) (Mangiri, 2018) sebagaimana ditunjukkan pada gambar 2.5 di bawah ini.



Gambar 2.5 Blok Diagram Metode Lossless Compression
Sumber : Mangiri, 2018

Lossless compression biasanya digunakan untuk mengurangi ukuran data-data yang penting, data-data yang isinya tidak boleh berubah sedikitpun, misalnya data-data yang berbentuk dokumen dan teks. Contoh metode *lossless compression* adalah *huffman coding*, *dynamic markov compression* (DMC), *Lempel-Ziv-Welch* (LZW), *aritmetic coding*, dan *Run-Length encoding* (Pangesti et al, 2020).

2.4 Algoritma Goldbach Code

Algoritma *Goldbach Code* adalah algoritma yang dibuat oleh Peter Fenwick yang dibuat menggunakan *conjecture Goldbach*. *Conjecture Goldbach* ini diciptakan oleh Christian Goldbach yaitu salah satu matematikawan terkenal pada abad 17. *Conjecture Goldbach* berisi setiap bilangan genap lebih besar dari 2 merupakan hasil penjumlahan dari 2 buah bilangan prima (Apriyanto and Hutrianto, 2020). Algoritma *Goldbach Code* adalah algoritma yang diasumsikan menggunakan teori *Goldbach Conjecture* yaitu semua bilangan genap positif yang lebih besar dari 2 merupakan penjumlahan dari dua bilangan prima (Almurtada and Syahrizal, 2018). Bilangan prima merupakan bilangan bulat yang hanya memiliki dua faktor yaitu 1 dan bilangan itu sendiri. Misalnya, bilangan bulat 5 adalah bilangan yang hanya dapat membagi bilangan 1 dan bilangan itu sendiri.

Konsep kerja algoritma *Goldbach Code* yaitu dengan menghitung jumlah frequensi kemunculan tiap karakter dari yang terbesar sampai yang terkecil, dan dilanjutkan dengan mencari *codeword* dengan cara mengkodekan bilangan bulat positif n dengan menghubahnya menjadi bilangan bulat positif genap dengan rumus $(2n + 3)$ dan kemudian menuliskan pasangan penjumlahan bilangan prima dalam keadaan terbalik (Irliansyah et al, 2017). Algoritma *Goldbach Code*s memiliki tiga kode, *Goldbach Code*s yang pertama dinamakan "G0" dan *Goldbach Code*s kedua dinamakan "G1" serta *Goldbach Code*s yang ketiga dinamakan "G2" adalah perluasan dari G1 *Codes* (Yogie, 2018). Adapun kode dari algoritma *Goldbach Code* G0 dapat disajikan pada tabel 2.1.

Tabel 2.1 Tabel Goldbach Codes G0

n	2 (n+3)	Primes	Codeword
1	8	3+5	11
2	10	3+7	101
3	12	5+7	011
4	14	3+11	1001
5	16	5+11	0101
6	18	7+11	0011
7	20	7+13	00101
8	22	5+17	010001
9	24	11+13	00011
10	26	7+19	0010001
11	28	11+17	000101
12	30	13+17	000011
13	32	13+19	0000101
14	34	11+23	00010001

Sumber : Nasution et al, 2017

Adapun tahapan yang dilakukan dalam proses kompresi data pada *file text* dengan menggunakan algoritma *Goldbach Code* (Apriyanto and Hutrianto, 2020) dapat dijelaskan sebagai berikut:

1. Langkah pertama mencari frekuensi kemunculan setiap karakter.
2. Kemudian urutkan karakter berdasarkan frekuensi kemunculan dari yang terbesar hingga terkecil.
3. Selanjutnya cari nilai $(2n + 3)$.
4. Setelah nilai dari $n2$ ditemukan, langkah berikutnya menentukan bilangan prima dan *codeword* G0 algoritma *Goldbach Code* dari nilai $n2$.
5. Langkah selanjutnya adalah mencari nilai bit tiap karakter, dimana jumlah digit *codeword* mewakili nilai bit karakter.
6. Hasil dari algoritma *Goldbach Code* adalah teks sebelum dikompresi kemudian dikompresi menggunakan *codeword* yang mewakili setiap karakter.

Sedangkan tahapan yang dilakukan dalam proses dekompresi data pada *file text* dengan menggunakan algoritma *Goldbach Code* (Apriyanto and Hutrianto, 2020) dapat dijelaskan sebagai berikut:

1. Mengambil nilai *biner* 1 dengan urutan ke-2 diikuti dengan nilai biner dibelakangnya sebagai identitas setiap karakter, begitu selanjutnya. Misalkan biner 1001101, ambil nilai biner 1 diurutan ke 2 diikuti dengan nilai biner sebelumnya maka didapat 1001. Nilai *biner* "1001" tersebut mewakili identitas sebuah karakter.
2. Kemudian setelah identitas karakter didapat, pembacaan karakter tersebut melalui proses header atau pembacaan ulang karakter yang menjadi kunci dalam proses dekompresi.

Algoritma *Goldbach Code* G0 adalah dasar dari algoritma *Goldbach G1 Code*. *Goldbach G1 Code* memiliki prinsip untuk menentukan dua bilangan prima P_i dan *Pj* (di mana $i \leq j$) yang jumlahnya menghasilkan bilangan bulat yang diberikan *n*, dan mengkodekan pair $(i, j - i + 1)$ dengan dua kode gamma (Siregar, 2019). Adapun kode dari algoritma *Goldbach Code* G1 dapat disajikan pada tabel 2.2.

Tabel 2.2 Tabel Goldbach Codes G1

N	Penjumlahan	Index	Pair	Kode	Panjang
2	1+1	1, 1	1, 1	1:1	2
4	1+3	1, 2	1, 2	1:010	4
6	3+3	2, 2	2, 1	010:1	4
8	3+5	2, 3	2, 2	010:010	6
10	5+5	3, 3	3, 1	011:1	4
12	5+7	3, 4	3, 2	011:010	6
14	7+7	4, 4	4, 1	00100:1	6
16	5+11	3, 5	3, 3	011:011	6
18	7+11	4, 5	4, 2	00100:010	8
20	7+13	4, 6	4, 3	00100:011	8

Sumber : Siregar, 2019

Pada kompresi terdapat beberapa faktor penting yang perlu diperhatikan sebagai bahan pertimbangan untuk mengukur kualitas dari suatu metode kompresi, serta

mendapatkan hasil perbandingan dari metode yang diuji. Parameter yang digunakan dalam proses kompresi pada penelitian ini adalah *Ratio of Compression* (RC), *Compression Ratio* (CR), *Space Savings* (SS) serta *Time Compression* (TM) yang dapat dijelaskan yaitu:

1. *Ratio of Compression* (RC)

   Ratio of Compression atau rasio kompresi adalah menghitung kinerja dari repersentase data yang sudah dikompresi dan sebelum dikompresi (Pramadi et al, 2019). Secara matematis rasio kompresi dapat dituliskan dengan persamaan (2.1).

   $$RC = 100\% - \left(\frac{Compressed\ bits}{Uncompressed\ bits}\right) * 100\% \quad (2.1)$$

   Misalkan rasio kompresi adalah 10% artinya 10% dari data semula telah berhasil dimampatkan.

2. *Compression Ratio* (CR)

   Compression Ratio atau kompresi rasio adalah persentase perbandingan antara data yang sudah dikompresi dan sebelum dikompresi (Tanjung and Nasution, 2020). Secara matematis kompresi rasio dapat dituliskan dengan persamaan (2.2).

   $$CR = \frac{Compressed\ bits}{Uncompressed\ bits} \quad (2.2)$$

3. *Space Saving* (SS)

   Space Savings adalah perbedaan antara data sebelum dikompresi dan data terkompresi (Tanjung and Nasution, 2020). *Space savings* merupakan persentase penghematan ruang (memori) setelah file dikompresi dengan mencari persentase selisih antara data awal sebelum dikompresi dengan hasil data yang telah dikompresi. Secara matematis *space savings* dapat dituliskan dengan persamaan (2.3).

   $$SS = \left(1 - \frac{Compressed\ bits}{Uncompressed\ bits}\right) x\ 100 \quad (2.3)$$

4. *Time Compression* (TM)

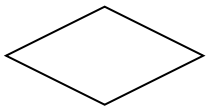
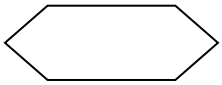
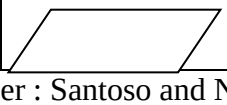
   Time Compression adalah perhitungan waktu yang diperoleh ketika algoritma melakukan kompresi atau dekompresi (Tanjung and Nasution, 2020). *Time Compression* pada pengujian sistem akan diukur dalam satuan *millisecond* (ms). Semakin sedikit waktu yang diperlukan sistem untuk melakukan kompresi, maka semakin efektif metode kompresi yang digunakan.

2.5 Flowchart

Flowchart atau diagram alir adalah representasi secara simbolik dari suatu algoritma atau prosedur untuk menyelesaikan suatu masalah, dengan menggunakan *flowchart* akan memudahkan pengguna melakukan pengecekan bagian-bagian yang terlupakan dalam analisis masalah (Santoso and Nurmalina, 2017). *Flowchart* merupakan gambar atau bagan yang memperlihatkan urutan dan hubungan antar proses beserta instruksinya yang dinyatakan dengan simbol. Simbol-simbol dalam flowchart menggambarkan proses tertentu, sedangkan hubungan antar proses digambarkan dengan garis penghubung.

Berikut ini adalah simbol-simbol yang digunakan dalam *flowchart* seperti yang terlihat pada table 2.3.

Tabel 2.3 Simbol-simbol Flowchart

No.	Simbol	Fungsi
1.		Permulaan sub program
2.		Perbandingan, pernyataan, penyeleksian data yang memberikan pilihan untuk langkah selanjutnya
3.		Penghubung bagianbagian flowchart yang berada pada satu halaman.
4.		Penghubung bagianbagian flowchart yang berada pada halaman berbeda
5.		Permulaan/akhir program
6.		Arah aliran program
7.		Proses inisialisasi/pemberian harga awal
8.		Proses penghitung/proses pengolahan data
9.		Proses input/output data

Sumber : Santoso and Nurmalina, 2017

2.6 Penelitian Terdahulu

Dalam penelitian ini, penulis sedikit banyak terinspirasi dan mereferensi dari beberapa penelitian-penelitian sebelumnya yang berkaitan erat dengan latar belakang masalah pada penelitian ini. Berikut ini beberapa penelitian yang berkaitan dengan teknik kriptografi dengan algoritma *Rabin-p* dan teknik kompresi dengan algoritma *Goldbach Code* yaitu:

1. Penelitian yang dilakukan oleh Hidayat (2021), hasil penelitian dengan menggunakan lima sampel uji file teks berformat *.txt* menyimpulkan bahwa algoritma *Rabin-p* lebih efisien dibandingkan algoritma RSA-CRT berdasarkan pengukuran waktu eksekusi komputasi dan kompleksitas algoritma.
2. Penelitian yang dilakukan oleh Yogie (2018), hasil yang dicapai dalam penelitian tersebut djelaskan bahwa algoritma *Goldbach Code* merupakan algoritma pengkompresian file yang cukup handal serta dapat menghemat space penyimpanan.
3. Penelitian yang dilakukan oleh Laswi (2019), menyimpulkan bahwa algoritma *Goldbach Code* cocok digunakan untuk mengoptimalkan *file text* sehingga dapat memperkecil penggunaan memori pada perangkat keras.
4. Penelitian yang dilakukan oleh Tanjung and Nasution (2020), menyimpulkan bahwa hasil kompresi pada *file text* menggunakan *Goldbach Code* sangat bagus dan hasilnya tidak berkurang dari file asli atau tidak ada pengurangan.
5. Penelitian yang dilakukan oleh Irliansyah et al (2017), menyimpulkan bahwa rasio yang dihasilkan berdasarkan proses kompresi data pada *file text* menggunakan kombinasi algoritma *Deflate* dan algoritma *Goldbach Code*s menghasilakan file dengan ukuran lebih kecil dari sebelum dilakukan proses kompresi.
6. Penelitian yang dilakukan oleh Laurentinus et al (2020), menyimpulkan bahwa algoritma simetris AES mempunyai kinerja waktu enkripsi dan dekripsi yang lebih baik daripada algoritma asimetris RSA seiring dengan besarnya ukuran karakter yang dienkripsi. Namun, kompresi algoritma *Huffman* terhadap *ciphertext* algoritma asimetris RSA lebih efisien daripada AES, maka dapat disimpulkan bahwa penerapan kompresi lebih efisien jika diterapkan pada algoritma kriptografi asimetris dibandingkan dengan algoritma kriptografi simetri

BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

Analisis sistem merupakan tahapan yang dilakukan untuk mengetahui hal-hal apa yang diperlukan dalam pembuatan sistem, serta meminimalisir kesalahan saat pembuatan sistem. Dengan adanya analisis sistem, diharapkan perangkat lunak atau aplikasi yang akan dibangun menjadi tepat guna dan sesuai dengan harapan. Pada latar belakang, telah dijelaskan permasalahan yang diteliti dalam penelitian ini adalah untuk meningkatkan kerahasiaan sebuah informasi yang terdapat didalam data *file text* serta memperkecil ukurannya dengan menerapkan sistem kriptokompresi menggunakan algoritma asimetris *Rabin-p* dan algoritma *lossless compression Goldbach Code*.

Dalam penelitian ini, akan menggabungkan teknik kriptografi dan teknik kompresi yang disebut dengan kriptokompresi dengan menggunakan algoritma asimetris *Rabin-p* dan algoritma lossless compression *Goldbach Code*. Kriptokompresi dalam penelitian ini merupakan teknik yang memanfaatkan dua metode kedalam satu proses, artinya data yang akan diamankan yaitu berupa *file text* akan dienkripsi terlebih dahulu dengan algoritma asimetris *Rabin-p* untuk tujuan mengamankan data, kemudian dikompresi menggunakan algoritma *lossless compression Goldbach Code* guna memperkecil ukuran data.

3.2 Analisis Kebutuhan Sistem

Dalam analisis kebutuhan sistem terdapat dua bagian yang akan dibahas, yaitu analisis kebutuhan fungsional sistem dan analisis kebutuhan non-fungsional sistem. Kebutuhan fungsional sistem mendeskripsikan fungsi-fungsi yang harus dilakukan oleh sebuah sistem untuk mencapai tujuan. Sedangkan kebutuhan non-fungsional sistem mendeskripsikan fitur lain seperti performa sistem, efisiensi sistem, dokumentasi sistem, kontrol sistem, dan manajemen kualitas agar sistem yang dibangun dapat berjalan dan berfungsi dengan sukses sesuai dengan yang diharapkan.

3.2.1 Kebutuhan Fungsional Sistem

Kebutuhan fungsional sistem menjelaskan aktifitas yang disediakan suatu sistem dalam melakukan pelayanannya dan kebutuhan yang harus dipenuhi oleh sistem. Untuk dapat menerapkan sistem kriptokompresi dalam pengamanan *file* teks berbasis *desktop* ini, maka kebutuhan fungsional sistem yang harus dipenuhi antara lain yaitu:

1. Sistem harus dapat membangkitkan kunci publik (*public key*) dan kunci privat (*private key*) dari algoritma *Rabin-p* secara acak (*random*) untuk selanjutnya digunakan dalam mengenkripsi dan dekripsi file teks.
2. Sistem harus dapat membaca *string* dalam file teks yang berformat *.txt* dan *.docx* yang akan diinputkan oleh pengguna.
3. Sistem harus dapat melakukan proses enkripsi terhadap setiap karakter yang terdapat dalam file teks dengan menggunakan algoritma kriptografi *Rabin-p* serta harus dapat melakukan proses dekripsi terhadap *ciphertext* hasil dari proses enkripsi menggunakan algoritma yang sama dengan proses enkripsi yaitu algoritma *Rabin-p*. *String* yang dihasilkan pada proses dekripsi harus sama dengan *string* yang terdapat pada file teks sebelum dilakukan proses enkripsi.
4. Sistem harus dapat melakukan kompresi terhadap file teks yang telah dienkripsi atau *ciphertext* dengan menggunakan algoritma kompresi *Goldbach Code*. Selain itu sistem juga harus dapat melakukan dekompresi terhadap *string* yang telah dilakukan proses dekripsi dan menghasilkan *string* yang sama dengan *string* yang terdapat pada file teks sebelum dienkripsi.

3.2.2 Kebutuhan Non-fungsional Sistem

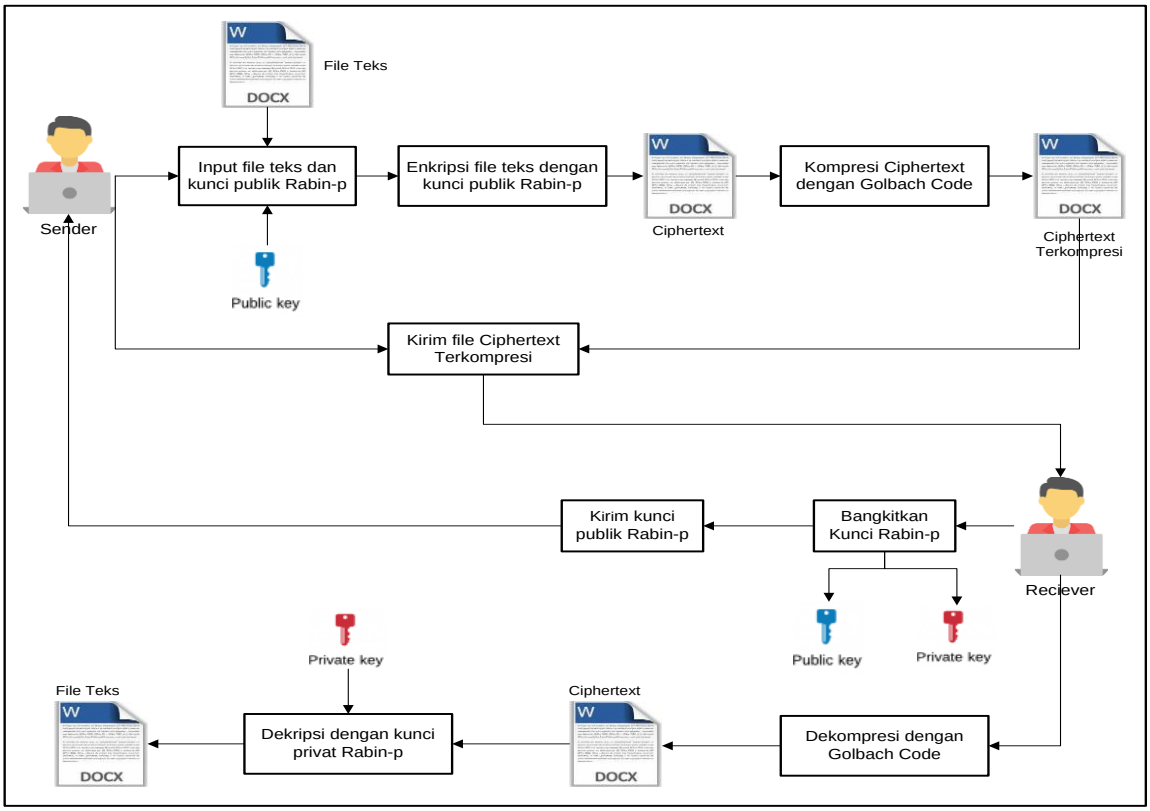
Kebutuhan non-fungsional sistem mendeskripsikan batasan layanan dan fungsi yang dimiliki oleh sistem seperti performa sistem, efisiensi sistem, dokumentasi, kontrol sistem, dan manajemen kualitas. Adapun kebutuhan non fungsional dari sistem kriptokompresi dengan menggunakan algoritma asimetris *Rabin-p* dan algoritma lossless compression *Goldbach Code* antara lain yaitu:

1. Performa, sistem atau perangkat lunak yang dibangun dapat menunjukkan serta menyimpan hasil dari fungsi kriptografi dan kompresi yang dilakukan oleh sistem.

2. Efisiensi, sistem dibangun dengan sesederhana mungkin agar pengguna dapat lebih mudah dalam menggunakannya.
3. Dokumentasi, sistem yang dibangun dapat menyimpan data hasil enkripsi dan kompresi maupun data hasil dekompresi dan dekripsi.
4. Kontrol, sistem yang dibangun akan menampilkan pesan jika penyimpanan *ciphertext* dan plaintext berhasil dan pesan error untuk setiap input yang tidak sesuai.
5. Manajemen kualitas, sistem yang dibangun memiliki kualitas yang baik, dimana proses pembangkitan kunci yang cepat serta proses enkripsi, kompresi, dekompresi dan dekripsi yang akurat.

3.3 Analisis Proses Kriptokompresi

Kriptokompresi dalam penelitian ini merupakan teknik yang memanfaatkan dua metode kedalam satu proses, artinya data yang akan diamankan yaitu berupa *file text* akan dienkripsi terlebih dahulu dengan algoritma asimetris *Rabin-p* untuk tujuan mengamankan data, kemudian dikompresi menggunakan algoritma *lossless compression Goldbach Code* guna memperkecil ukuran data.



Gambar 3.1.Skema Sistem Kriptokompresi

Skema sistem kriptokompresi pada gambar 3.1 dapat dijelaskan untuk proses pertama kali penerima pesan (*reciever*) membangkitkan kunci *Rabin-p* yang akan menghasilkan sepasang kunci, yaitu kunci publik (*public key*) dan kunci privat (*private key*). Langkah selanjutnya *reciever* akan mengirimkan *public key* ke pengirim pesan (*sender*). *Sender* melakukan proses enkripsi file teks mnggunakan kunci publik *Rabin-p* dan menghasilkan *ciphertext* yang selanjutnya akan dikompresi menggunakan algoritma *Goldbach Code* yang akan menghasilkan file *ciphertext* yang terkompresi. Setelah itu akan dikirmkan ke *reciever*.

Setelah *reciever* menerima file *ciphertext* yang terkompresi, maka tahap pertama yang dilakukan yaitu proses dekompresi menggunakan algoritma *Goldbach Code* yang akan menghasilkan file *ciphertext*. Selanjutnya akan di dekripsi menggunakan kunci privat algoritma *Rabin-p* dan menghasilkan *file* teks yang asli.

3.3.1 Analisis Proses Enkripsi dan Kompresi

Proses enkripsi dan kompresi dalam skema sistem kriptokompresi seperti yang telah dijelaskan pada gambar 3.1 dimulai dengan membangkitkan kunci publik dan kunci privat algoritma *Rabin-p* yang dilakukan oleh penerima pesan. Algoritma *Rabin-p* merupakan algoritma kriptografi asimetris dimana kunci enkripsi tidak sama dengan kunci dekripsinya. Untuk mengenkripsi dan dekripsi pesan dengan menggunakan algoritma *Rabin-p*, maka terlebih dahulu membangkitkan sepasang kunci, yaitu kunci publik (*public key*) dan kunci privat (*private key*).

Adapun algoritma untuk membangkitkan kunci publik (*public key*) dan kunci privat (*private key*) algoritma *Rabin-p* yaitu sebagai berikut:

1. Langkah pertama adalah menentukan parameter keamanan k. Misalkan dipilih untuk nilai $k = 7$
2. Langkah kedua menentukan bilangan prima p dan q dimana $2^k < p, q < 2^{k+1}$ memenuhi syarat $p \equiv q \equiv 3\ mod\ 4$ atau $p\ mod\ 4 = 3$ dan $q\ mod\ 4 = 3$

   Misalkan dipilih bilangan prima p dan q yaitu sebagai berikut:

   $p = 131$ dan $q = 43$

   Pembuktian untuk bilangan prima $p = 131$

   $2^k < p$

$2^7 < 131$

$128 < 131$

$131 \equiv 3\ mod\ 4 \rightarrow$ Memenuhi syarat, karena $131\ mod\ 4 = 3$

Pembuktian untuk bilangan prima $q = 43$

$q < 2^{k+1}$

$43 < 2^{7+1}$

$43 < 2^8$

$43 < 256$

$43 \equiv 3(mod\ 4) \rightarrow$ Memenuhi syarat, karena $43\ mod\ 4 = 3$

3. Langkah ketiga adalah menghitung nilai $N = p^2q$

   $N = p^2q$

   $N = (131^2) * 43$

   $N = 737923$

4. Sehingga pasangan kunci enkripsi dan dekripsi hasil dari algoritma algoritma *Rabin-p* di atas adalah sebagai berikut:

   1). Kunci enkripsi (*public key*) adalah nilai $N = 737923$

   2). Kunci dekripsi (*private key*) adalah nilai $p = 131$

Setelah kunci publik (*public key*) dan kunci privat (*private key*) algoritma *Rabin-p* dibangkitkan, maka proses enkripsi dapat dilakukan dengan menggunakan persamaan (2.1). Adapun tahapan yang dilakukan dalam proses enkripsi dengan menggunakan algoritma *Rabin-p* dapat dijelaskan yaitu sebagai berikut:

1. Langkah pertama ambil kunci publik (*public key*) algoritma *Rabin-p* yang telah dibangkitkan yaitu $N = p^2q = 737923$ dan parameter keamanan nilai $k = 7$
2. Langkah kedua adalah menentukan *plaintext* (m) yang akan di enkripsi, dimana $0 < m < 2^{2k-1}$ dan $GCD(m, N) = 1$

. Pada contoh kasus yang digunakan dalam penelitian ini ditentukan *plaintext* (m) yang akan dienkripsi adalah sebagai berikut:

Plaintext (m) : TRIJOKO

Sebelum *plaintext* dapat di enkripsi maka terlebih dahulu setiap karakter *plaintext* dikonversi kedalam nilai desimal pada tabel ASCII sehingga diperoleh hasil konversi seperti disajikan pada tabel 3.1 berikut.

Tabel 3.1 Konversi Karakter Pada Tabel ASCII

Karakter Plaintext (m)	Kode ASCII
T	84
R	82
I	73
J	74
O	79
K	75
O	79

Pembuktian untuk $m = 84 \rightarrow 0 < m < 2^{2k-1}$

$0 < 84 < 2^{2*7-1}$

$0 < 84 < 2^{13}$

$0 < 84 < 8192$

$m = 84$ memenuhi syarat $0 < m < 2^{2k-1}$

Pembuktian untuk $GCD(m, N) = 1$

$GCD(84, 737923) = 84\ mod\ 737923 = 84$

$737923\ \ mod\ 84 = 67$

$84\ \ mod\ 67 = 17$

$67\ \ mod\ 17 = 16$

$17\ \ mod\ 16 = 1$

Karena memenuhi syarat $GCD(m, N) = 1$, maka dapat digunakan $m = 84$

Lakukan hal yang sama untuk semua karakter yang akan di enkripsi dan dapat dipastikan bahwa semua karakter (*plaintext*) yang digunakan dapat memenuhi syarat.

3. Langkah ketiga adalah melakukan proses enkripsi dengan menggunakan persamaan (2.1) sehingga diperoleh hasil enkripsi (*ciphertext*), yaitu sebagai berikut:

   1). Untuk karakter "T" dengan nilai desimal pada tabel ASCII yaitu 84

   $c = m^2 mod\ N$

   $c = 84^2\ mod\ 737923$

   $c = 7056\ mod\ 737923$

   $c = 7056$

2). Untuk karakter “R” dengan nilai desimal pada tabel ASCII yaitu 82

$c = m^2 \ mod \ N$

$c = 82^2 \ mod \ 737923$

$c = 6724 \ mod \ 737923$

$c = 6724$

3). Untuk karakter “I” dengan nilai desimal pada tabel ASCII yaitu 73

$c = m^2 \ mod \ N$

$c_3 = 73^2 \ mod \ 737923$

$c_3 = 5329 \ mod \ 737923$

$c_3 = 5329$

4). Untuk karakter “J” dengan nilai desimal pada tabel ASCII yaitu 74

$c = m^2 \ mod \ N$

$c = 74^2 \ mod \ 737923$

$c = 5476 \ mod \ 737923$

$c = 5476$

5). Untuk karakter “O” dengan nilai desimal pada tabel ASCII yaitu 79

$c = m^2 \ mod \ N$

$c = 79^2 \ mod \ 737923$

$c = 6241 \ mod \ 737923$

$c = 6241$

6). Untuk karakter “K” dengan nilai desimal pada tabel ASCII yaitu 75

$c = m^2 \ mod \ N$

$c = 75^2 \ mod \ 737923$

$c = 5625 \ mod \ 737923$

$c = 5625$

7). Untuk karakter “O” dengan nilai desimal pada tabel ASCII yaitu 79

$c = m^2 \ mod \ N$

$c_7 = 79^2 \ mod \ 737923$

$c_7 = 6241 \ mod \ 737923$

$c_7 = 6241$

Berdasarkan hasil perhitungan diatas maka diperoleh hasil enkripsi (*ciphertext*) dengan menggunakan algoritma *Rabin-p* yaitu seperti disajikan pada tabel 3.2 berikut.

Tabel 3.2 Hasil Enkripsi Algoritma Rabin-p

Karakter Plaintext (m)	**Kode ASCII**	***Ciphertext* (c)**
T	84	7056
R	82	6724
I	73	5329
J	74	5476
O	79	6241
K	75	5625
O	79	6241

Berdasarkan hasil enkripsi (*ciphertext*) pada tabel 3.2 maka dalam penelitian ini akan ditambahkan karakter spasi diantara setiap nilai *ciphertext* yang difungsikan sebagai penanda antara satu *ciphertext* dengan *ciphertext* yang lain. Adapun hasil dari *ciphertext* yaitu : 7056 6724 5329 5476 6241 5625 6241.

Setelah hasil enkripsi (*ciphertext*) didapatkan maka tahap selanjutnya dilakukan proses kompresi untuk memperkecil ukuran data pada *ciphertext*. Metode kompresi yang digunakan dalam penelitian ini yaitu metode *lossless compression* dengan menggunakan algoritma *Goldbach Code*. Tahap awal kompresi adalah menghitung jumlah frekuensi yang terdapat dalam *ciphertext* yang hasilnya dapat ditunjukkan pada tabel 3.3.

Tabel 3.3 Ciphertext Sebelum Dikompresi

Char	**Freq**	**ASCII Desimal**	**ASCII *Biner***	**Bit**	**Bit * Freq**
7	3	55	00110111	8	24
0	1	48	00110000	8	8
5	5	53	00110101	8	40
6	6	54	00110110	8	48
spasi	6	32	00100000	8	48
2	5	50	00110010	8	40
4	4	52	00110100	8	32
3	1	51	00110011	8	8
9	1	57	00111001	8	8
1	2	49	00110001	8	16
Total Bit					**272**

Berdasarkan tabel 3.3, satu karakter bernilai 8 bit bilangan *biner* dalam kode ASCII. Sehingga 33 karakter pada *string ciphertext* mempunyai nilai *biner* sebanyak 272 bit, artinya *string ciphertext* sebelum dikompresi adalah 34 byte, dimana 1 karakter adalah 1 byte, dan 1 byte adalah 8 bit maka total ukuran file teks yang akan di kompresi = 272 bit.

Setelah menghitung jumlah frekuensi yang terdapat dalam *ciphertext*, maka tahap selanjutnya adalah dengan membangkitkan codeword *Goldbach Code* yang dimulai dengan jumlah frekuensi terbesar yang diurutkan secara ascending seperti pada tabel 3.4.

Tabel 3.4 Ciphertext Setelah Dikompresi

Char	**Freq**	**N**	**N**	**Penjumlahan**	**Index**	**Pair**	**Kode**	**Bit**	**Bit * Freq**
Spasi	6	1	2	1+1	1, 1	1, 1	1:1	2	12
6	6	2	4	1+3	1, 2	1, 2	1:010	4	24
5	5	3	6	3+3	2, 2	2, 1	010:1	4	20
2	5	4	8	3+5	2, 3	2, 2	010:010	6	30
4	4	5	10	5+5	3, 3	3, 1	011:1	4	16
7	3	6	12	5+7	3, 4	3, 2	011:010	6	18
1	2	7	14	7+7	4, 4	4, 1	00100:1	6	12
3	1	8	16	5+11	3, 5	3, 3	011:011	6	6
9	1	10	20	7+13	4, 6	4, 3	00100:010	8	8
0	1	9	18	7+11	4, 5	4, 2	00100:011	8	8
Total Bit									**154**

Tahap selajutnya yaitu menyusun kode-kode yang telah dihasilkan sesuai dengan posisi karakter pada *string ciphertext* berdasarkan tabel 3.4. Diketahui bahwa *string ciphertext* adalah 7056 6724 5329 5476 6241 5625 6241 sehingga menjadi:

7	0	5	6	spasi	6	7	2	4	Spasi
011010	00100011	0101	1010	11	1010	011010	010010	0111	11
5	3	2	9	spasi	5	4	7	6	Spasi
0101	011011	010010	00100010	11	0101	0111	011010	1010	11
6	2	4	1	spasi	5	6	2	5	Spasi
1010	010010	0111	001001	11	0101	1010	010010	0101	11
6	2	4	1						
1010	010010	0111	001001						

Setelah *string ciphertext* disusun berdasarkan *codeword* sesuai dengan karakternya masing-masing, sehingga akan menjadi *string* bit berikut ini:

01101000100011010110101110100110100100100111110101011011010010001000101101010111011010101011101001001001110010011101011010010010010111101001001001110010011100100

Dalam ASCII satu karakter direpresentasikan sebanyak 8 bit dengan bilangan *biner*. Namun, jika *bit* tersebut bukan kelipatan 8, maka dilakukan penambahan *padding bit* dan *flag bit*. *Padding* yaitu menambahkan *bit* 0 pada hasil kompresi yang bukan merupakan kelipatan 8. Sedangkan *flag* yaitu untuk menjelaskan berapa jumlah *bit* yang ditambahkan dalam melakukan *padding*. Adapun penambahan *bit* dilakukan dengan menghitung sisa jumlah *string bit* dibagi 8, lalu hasilnya diselisihkan dengan 8. Adapun jumlah *string bit* hasil kompresi adalah 154. Jika 154 dibagi 8 akan menyisakan 2, lalu selisihkan dengan 8 maka hasilnya 6. Maka hasil *padding bit* yaitu 000000, dan diperoleh *flag* 00000110 (*biner* dari 6). Dengan demikian hasil kompresi setelah menambahkan *padding bit* dan *flag bit* adalah sebagai berikut:

01101000100011010110101110100110100100100111110101011011010010001000101101010111011010101011101001001001110010011101011010010010010111101001001001110010011100100000000000000110

Setelah dilakukan penambahan *padding bit* dan *flag bit* maka diperoleh total *string bit* hasil kompresi yaitu 154 (*string bit*) + 6 (*padding*) + 8 (*flag*) = 168 *bit*. Selanjutnya akan dilakukan perhitungan untuk mengetahui kualitas dari algoritma kompresi yang digunakan. Adapun parameter yang digunakan dalam proses kompresi pada penelitian ini adalah *Ratio of Compression* (RC), *Compression Ratio* (CR), dan *Space Savings* (SS). Sebagaimana diketahu bahwa ukuran data sebelum dan setelah kompresi yaitu:

Ukuran data sebelum di kompresi (*uncompressed bits*) = 272 *bit*

Ukuran data setelah di kompresi (*compressed bits*) = 168 *bit*

Dengan menggunakan persamaan (2.3) maka dapat diperoleh *Ratio of Compression* (RC) yaitu sebagai berikut:

$$RC = 100\% - \left(\frac{Compressed\ bits}{Uncompressed\ bits}\right) * 100\%$$

$$RC = 100\% - \left(\frac{168}{272}\right) * 100\% = 0.382 = 38{,}23\%$$

Dengan menggunakan persamaan (2.4) maka dapat diperoleh *Compression Ratio* (CR) yaitu sebagai berikut:

$$CR = \frac{Compressed\ bits}{Uncompressed\ bits}$$

$$CR = \frac{168}{272} = 0.6175$$

Dengan menggunakan persamaan (2.5) maka dapat diperoleh *Space Saving* (SS) yaitu sebagai berikut:

$$SS = \left(1 - \frac{Compressed\ bits}{Uncompressed\ bits}\right) x\ 100$$

$$SS = \left(1 - \frac{168}{272}\right) x\ 100 = 38{,}235$$

Berdasarkan hasil perhitungan diatas maka diperoleh parameter kompresi untuk *Ratio of Compression* (RC) yaitu 38,23% yang artinya 42.4% dari data semula telah berhasil diperkecil ukurannya. Sedangkan untuk *Compression Ratio* (CR) yaitu 0.617 serta *Space Saving* (SS) yaitu 38,23.

3.3.2 Analisis Proses Dekompresi dan Dekripsi

Proses dekompresi dan dekripsi dalam skema sistem kriptokompresi seperti yang telah dijelaskan pada gambar 3.1 dimulai dengan melakukan proses dekompresi terlebih dahulu untuk mendapatkan kembali data *ciphertext*, kemudvan dilanjutkan dengan melakukan dekripsi unutk mendapatkan kembali data aslinya. Proses dekompresi dilakukan dengan terlebih dahulu menentukan panjang *string bit* yang harus dibaca dengan menghitung *padding* dan *flagging*. Pada tabel 3.4, *string bit* yang dihasilkan adalah:

01101000100011010110101110100110100100100111110101011011010010001000
01011010101110110101010111010010010011100100111010110100100100101111
10100100100111001001000000000000110

Flagging pada *string bit* tersebut adalah 00000110 (8 *bit* terakhir pada *string bit*) yang bila dikonversi ke dalam desimal akan bernilai 6, maka *padding* 000000. Sehingga panjang *string bit* yang harus dibaca adalah panjang *string bit* (168 *bit*) terkompresi dikurang total jumlah panjang *padding* dan *flagging* (6+8). Maka diperoleh nilai panjang *string bit* yang harus dibaca adalah 168 - (6+8) = 154 *bit* sehingga akan menghasilkan *string bit* berikut:

01101000100011010110101110100110100100100111110101011011010010001000101110101011101101010101110100100100111001001110101101001001001011110100100100111010

Proses dekompresi dilakukan dengan pembacaan pada *string bit* yang didapat dari proses kompresi pada tabel 3.4 yang dimulai dari indeks terawal sampai dengan indeks terakhir dengan terus menambahkan nilai pada indeks sebelumnya yang tidak mewakili karakter pada tabel 3.4. Kemudian cocokkan kembali nilai pada *string bit* sesuai dengan tabel *codeword* yang digunakan pada proses kompresi *string bit* ini. Untuk melakukan dekompresi, maka pembacaan *string bit* dimulai dari indeks paling kecil yang dapat diuraikan sebagai berikut:

01101000100011010110101110100110100100100111110101011011010010001000101110101011101101010101110100100100111001001110101101001001001011110100100100111010

Indeks ke-0 adalah **0**, pada tabel 3.4, kode **0** tidak mewakilkan karakter apapun.

011010001000110101101011101001101001001001111101010110110100100010001011101010111011010101011101001001001110010011101011010010010010111101001001001110100

Maka dikuti oleh indeks ke-1 yaitu **1**. Pada tabel 3.4, kode **01** tidak mewakilkan karakter apapun.

0110100010001101011010111010011010010010011111010101101101001000100010111010101110110101010111010010010011100100111010110100100100101111010010010011101001

Maka dikuti oleh indeks ke-2 yaitu **1**. Pada tabel 3.4, kode **011** tidak mewakilkan karakter apapun.

01101000100011010110101110100110100100100111110101011011010010001000101110101011101101010101110100100100111001001110101101001001001011110100100100111010

Maka dikuti oleh indeks ke-3 yaitu **0**. Pada tabel 3.4, kode **0110** tidak mewakilkan karakter apapun.

011010001000110101101011101001101001001001111101010110110100100010001011101010111011010101011101001001001110010011101011010010010010111101001001001110100

Maka dikuti oleh indeks ke-4 yaitu **1**. Pada tabel 3.4, kode **01101** tidak mewakilkan karakter apapun.

0110100010001101011010111010011010010010011111010101101101001000100101101010111011010101011101001001001110010011101011010010010010111101001001001110010011

Maka dikuti oleh indeks ke-5 yaitu **0**. Pada tabel 3.4, kode **011010** mewakili karakter "7".

Pembacaan selanjutnya dimulai pada indeks ke-6 yaitu **0**

011010**0**0100011010110101110100110100100100111110101011011010010001001011010101110110101010111010010010011100100111010110100100100101111010010010011100100111

Langkah tersebut akan dilakukan terus berlangsung hingga *string bit* habis sehingga didapatkan hasil dekompresi yaitu "7056 6724 5329 5476 6241 5625 6241" yang merupakan *ciphertext* yang telah dienkripsi sebelumnya.

Setelah *ciphertext* diperoleh, maka selanjutnya akan dikirimkan ke penerima untuk proses dekripsi yang bertujuan untuk mengembalikan pesan sudah di enkripsi pada langkah sebelumnya. Proses dekripsi algoritma *Rabin-p* dengan menggunakan kunci privat dapat dilakukan dengan menggunakan persamaan (2.2), yaitu sebagai berikut:

1. Langkah pertama ambil hasil enkripsi (*ciphertext*) yaitu sebagai berikut:

   $c = 7056\ 6724\ 5329\ 5476\ 6241\ 5625\ 6241$

   Kemudian bagi *ciphertext* menjadi beberapa bagian yaitu sebagai berikut:

   $c_1 = 7056$

   $c_2 = 6724$

   $c_3 = 5329$

   $c_4 = 5476$

   $c_5 = 6241$

   $c_6 = 5625$

   $c_7 = 6241$
2. Langkah kedua ambil kunci privat (*private key*) *Rabin-p* yaitu $p = 131$
3. Langkah ketiga adalah melakukan proses dekripsi sehingga didapatkan hasilnya yaitu sebagai berikut:

   Untuk *ciphertext* $c_1 = 7056$

Menghitung $w \equiv c \; mod \; p$

$$w \equiv 7056 \; mod \; 131$$

$$w \equiv 113$$

Menghitung $m_p \equiv w^{\frac{p+1}{4}} mod \; p$

$$m_p \equiv 113^{\frac{131+1}{4}} mod \; 131$$

$$m_p \equiv 113^{\frac{132}{4}} mod \; 131$$

$$m_p \equiv 113^{33} mod \; 131$$

$$m_p \equiv 84$$

Menghitung $i = \frac{c - m_p^2}{p}$

$$i = \frac{7056 - 84^2}{131}$$

$$i = \frac{7056 - 7056}{131}$$

$$i = \frac{0}{131} = 0$$

Menghitung $j \equiv \frac{i}{2m_p} mod \; p$

$$j \equiv \frac{0}{2 * 84} mod \; 131$$

$$j \equiv \frac{0}{168} mod \; 131 = 0$$

Menghitung $m_1 = m_p + jp$

$$m_1 = 84 + 0 * 131$$

$$m_1 = 84 + 0$$

$$m_1 = 84$$

Menentukan nilai m, jika $m_1 < 2^{2k-1}$ maka $m = m_1$

Selain itu, $m = p^2 - m_1$

$m_1 < 2^{2k-1}$

$84 < 2^{2*7-1}$

$84 < 2^{13}$

$84 < 8192$

Karena $84 < 8192$, maka memenuhi syarat $m_1 < 2^{2k-1}$, sehingga hasil dekripsinya adalah $m = 84$ dan jika dikonversi ke tabel ASCII adalah karakter huruf "T" yang merupakan pesan asli (*plaintext*) pada karakter pertama.

4. Lakukan hal yang sama untuk semua karakter *ciphertext* sehingga diperoleh hasil dekripsi dengan menggunakan algoritma *Rabin-p* yaitu seperti disajikan pada tabel 3.5 berikut.

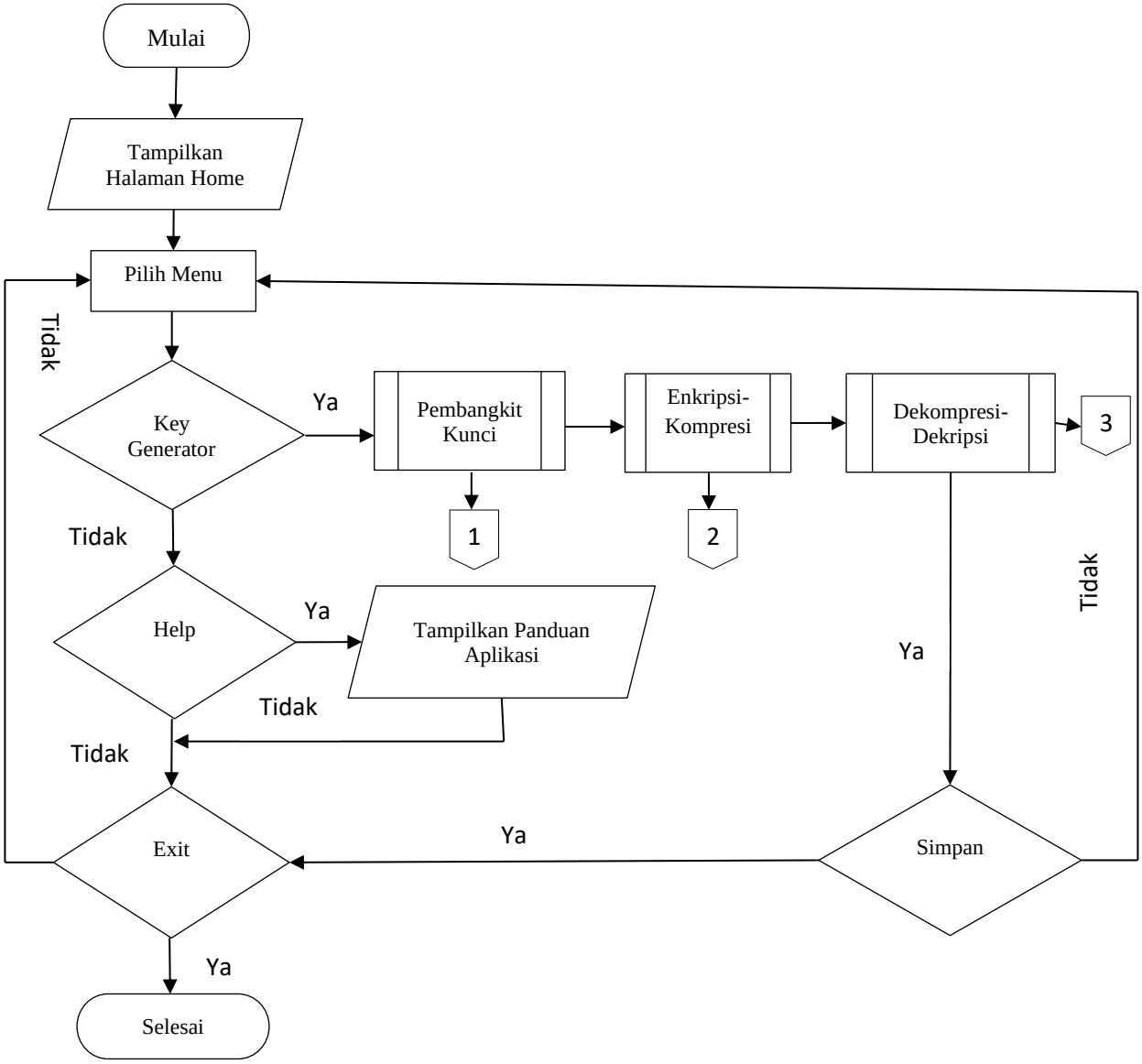
Tabel 3.5 Hasil Dekripsi Algoritma Rabin-p

***Ciphertext* (c)**	**Hasil Dekripsi (m)**	**Karakter**
7056	84	T
6724	82	R
5329	73	I
5476	74	J
6241	79	O
5625	75	K
6241	79	O

Berdasarkan hasil dekripsi pada tabel 3.5 maka diperoleh *plaintext* hasil dekripsi yang merupakan pesan asli yang sebenarnya menjadi "TRIJOKO". Dari hasil proses dekompresi dan dekripsi pada contoh kasus yang telah diuraikan, maka dapat dinyatakan bahwa hasil proses dekompresi dan dekripsi sudah sesuai berdasarkan aturan dari algoritma kompresi *Goldbach Code* dan algoritma kriptografi *Rabin-p* yang digunakan dalam penelitian ini.

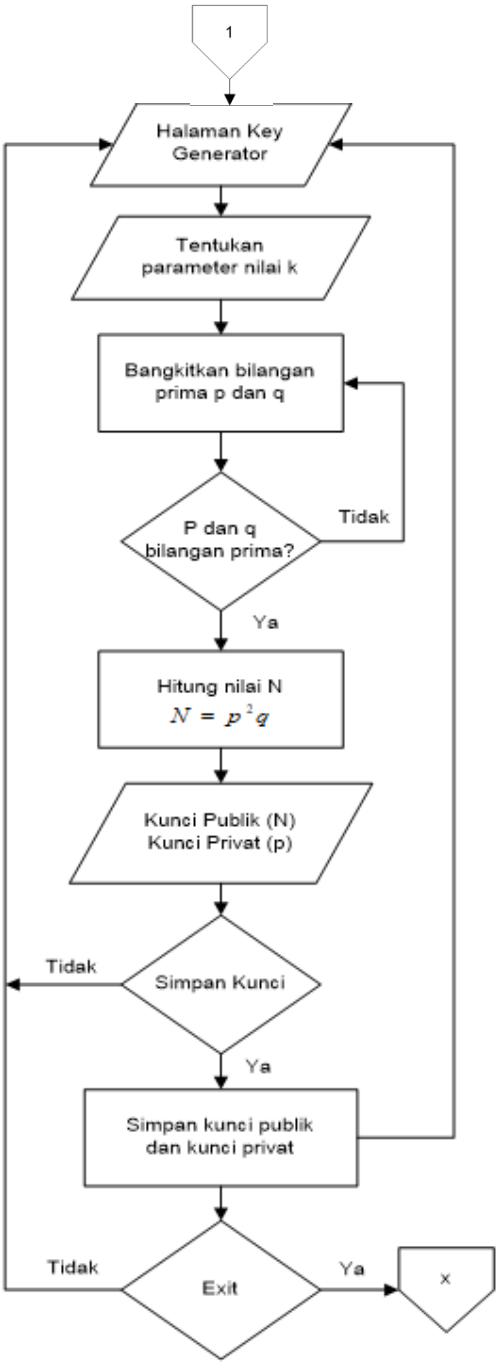
3.4 Flowchart Sistem

Flowchart atau diagram alir merupakan gambar atau bagan yang pada penelitian ini akan menggambarkan urutan dan hubungan antar proses yang terdapat dalam sistem dengan menggunakan simbol-simbol tertentu. Dalam penelitian ini, *flowchart* sistem terdiri dari *flowchart* pembangkitan kunci algoritma *Rabin-p*, *flowchart* proses enkripsi dan kompresi, dan *flowchart* proses dekompresi dan dekripsi. Adapun untuk *flowchart* sistem dapat dilihat pada gambar 3.2.



Gambar 3.2 Flowchart Sistem

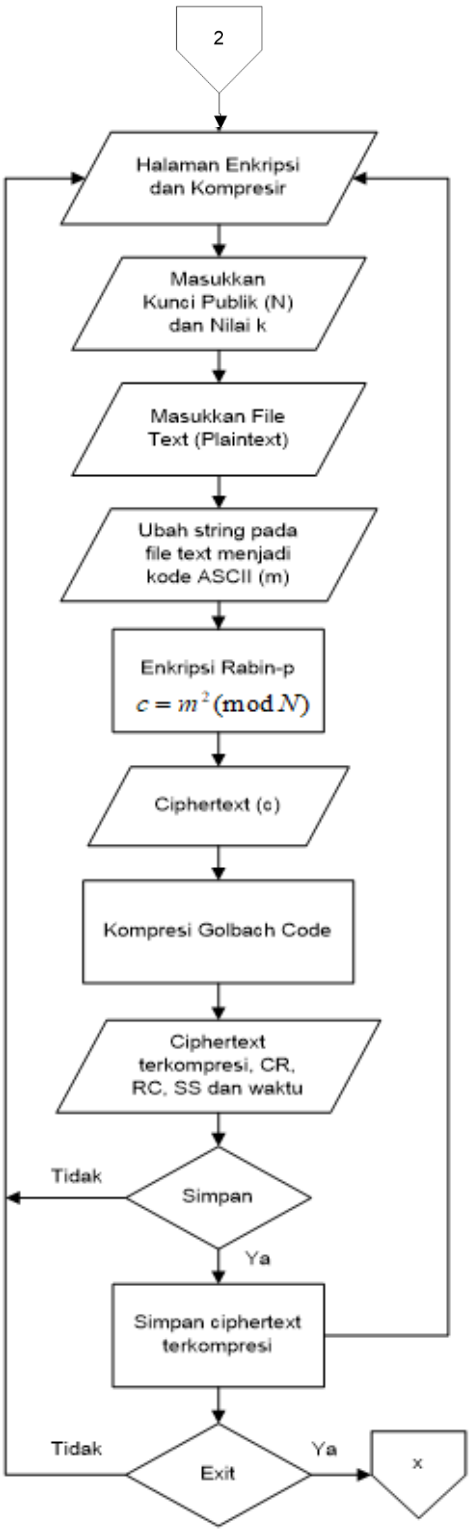
Proses pembangkitan kunci (*key generator*) dengan algoritma *Rabin-p* dilakukan dengan menentukan parameter keamanan k lalu mengambil bilangan secara acak untuk p dan q kemudian dicek apakah bilangan tersebut termasuk dalam persayaratan bilangan prima. Selanjutnya menghitung nilai N sehingga didapatkan pasangan kunci, yaitu kunci publik (*public key*) dan kunci privat (*private key*). Adapun *flowchart* pembangkitan kunci algoritma *Rabin-p* dapat dilihat pada gambar 3.3.



Gambar 3.3 Flowchart Pembangkitan Kunci Algoritma *Rabin-p*

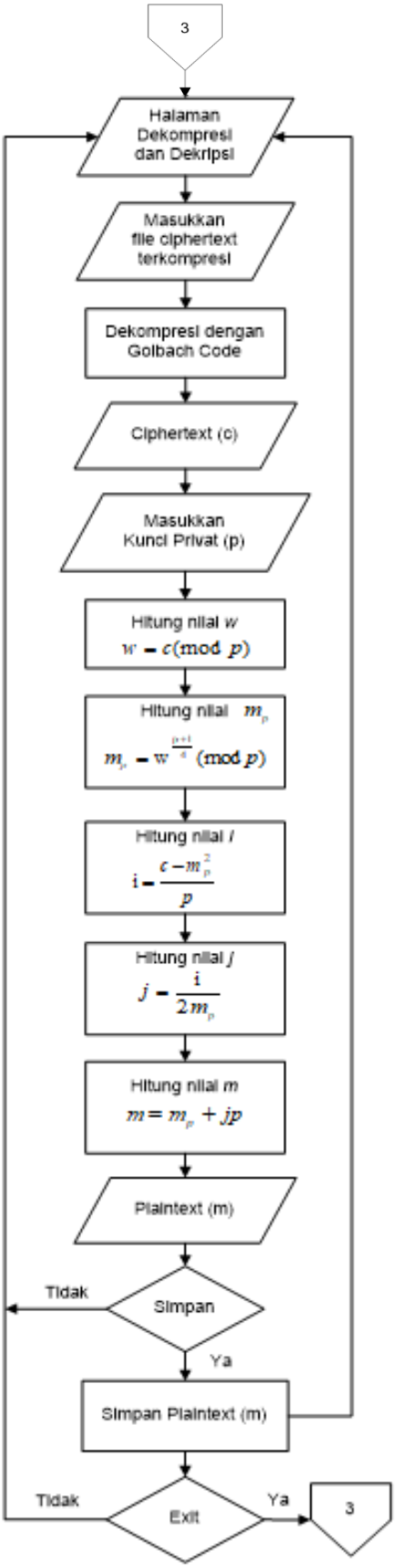
Flowchart enkripsi dan kompresi dimulai dengan menginputkan kunci publik dan *plaintext* (file teks) yang akan di enkripsi. Kemudian plaintext tersebut akan di enkripsi dengan algoritma *Rabin-p* sehingga diperoleh hasil enkripsi (*ciphertext*) yang selanjutnya dilakukan proses kompresi dengan menggunakan algoritma *Goldbach Code* sehinnga ukuran *ciphertext* menjadi lebih kecil dan akan ditampilkan file terkompresi, RC (*Ratio Compression*), CR (*Compression Ratio*), SS (*Space Savings*) dan waktu (*running time*),

kemudian disimpan. Adapun *flowchart* proses enkripsi dan kompresi dengan algoritma *Rabin-p* dan algoritma *Goldbach Code* dapat dilihat pada gambar 3.4.



Gambar 3.4 Flowchart Proses Enkripsi dan Kompresi

Flowchart proses dekompresi dan dekripsi dengan algoritma *Goldbach Code* dan algoritma *Rabin-p* dapat dilihat pada gambar 3.5.



Gambar 3.5 Flowchart Proses Dekompresi dan Dekripsi

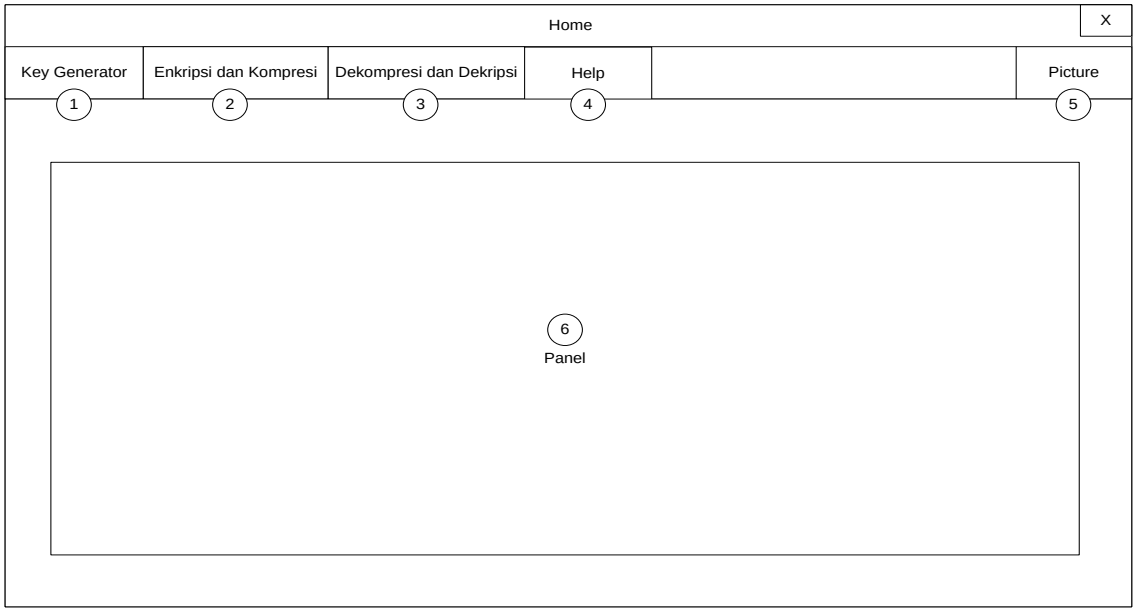
Flowchart dekompresi dan dekripsi dimulai dengan menginputkan file terkompresi kemudian dilakukan dekompresi menggunakan algoritma *Goldbach Code* sehingga menghasilkan file hasil dekompresi yang merupakan file *ciphertext*. Selanjutnya dilakukan proses dekripsi dengan algoritma *Rabin-p* dengan menggunakan kunci privat sehingga menjadi *plaintext* yang sama dengan *plaintext* sebelum dilakukan proses enkripsi.

3.5 Perancangan Interface Sistem

Perancangan *interface* atau antarmuka merupakan perancangan tampilan suatu sistem yang akan dibangun. Tujuan dari perancangan antarmuka yaitu membuat tampilan sistem yang sederhana dan mudah digunakan (*user friendly*) sehingga user dapat lebih mudah dan nyaman dalam menjalankan sistem. Interface yang akan dirancang pada sistem ini memiliki enam bagian utama, yaitu halaman utama (*home*), halaman *key generator* (bangkitkan kunci), halaman enkripsi dan kompresi, halaman dekompresi dan dekripsi, serta halaman *help*.

3.5.1 Rancangan Halaman Home

Halaman *home* merupakan halaman utama dari sistem. Halaman *home* terdiri dari beberapa menu yang membuat *user* dapat mengakses halaman lain. Rancangan halaman utama (*home*) dapat dilihat pada gambar 3.6.



Gambar 3.6 Rancangan Interface Halaman Home

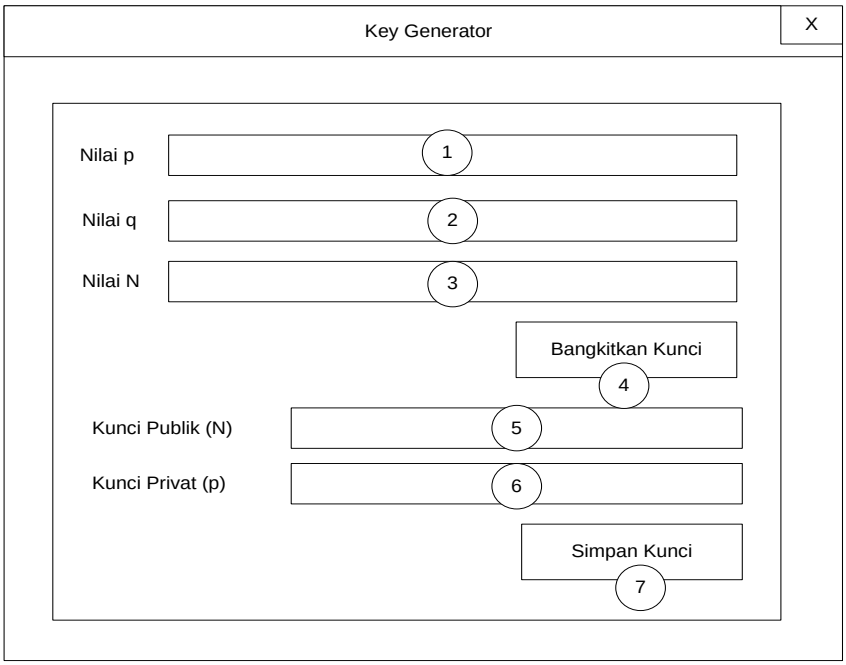
Berikut ini merupakan keterangan rancangan antarmuka pada halaman utama (*home*) seperti dijelaskan pada tabel 3.6.

Tabel 3.6 Keterangan Rancangan Interface Halaman Home

No.	Komponen	Name of Properties	Keterangan
1.	*Button1*	btnKeyGenerator	menu "Key Generator" berfungsi untuk menampilkan halaman bangkitan kunci publik dan kunci privat *Rabin-p*
2.	*Button2*	btnEnkripKompres	menu "Enkripsi dan Kompresi" berfungsi untuk menampilkan halaman enkripsi dan kompresi
3.	*Button3*	btnDekompresDekrip	menu "Dekompresi dan Dekripsi" berfungsi untuk menampilkan halaman dekompresi dan dekripsi
4.	*Button4*	btnHelp	menu "Help" berfungsi untuk menampilkan halaman help
5.	*PictureBox1*	pictureBox1	untuk menampilkan *icon user*
6.	*Panel1*	panel1	untuk menampilkan *form* dari menu yang dipilih

3.5.2 Rancangan Halaman Key Generator

Halaman *key generator* merupakan halaman yang bertujuan untuk proses pembangkitan kunci publik (*public key*) dan kunci privat (*private key*) algoritma *Rabin-p* yang nantinya diperlukan dalam proses enkripsi dan dekripsi. Pada halaman *key generator* terdapat dua buah tombol utama, yaitu tombol untuk membangkitkan kunci publik dan kunci privat dan tombol untuk menyimpan kunci yang telah dibangkitkan. Adapun untuk rancangan halaman *key generator* dapat dilihat pada gambar 3.7.



Gambar 3.7 Rancangan Interface Halaman Key Generator

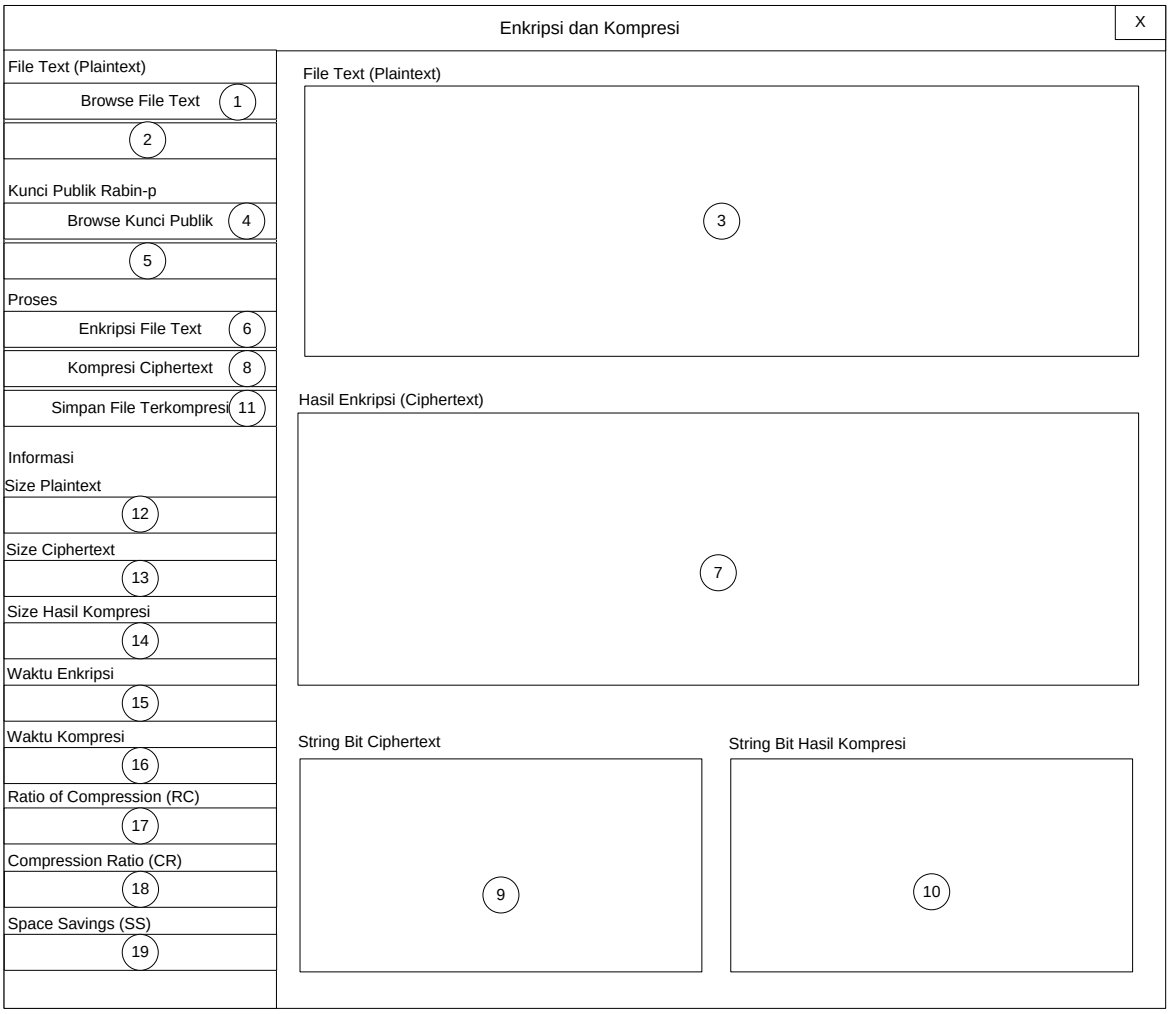
Berikut ini merupakan keterangan rancangan antarmuka pada halaman *key generator* seperti dijelaskan pada tabel 3.7.

Tabel 3.7 Keterangan Rancangan Interface Halaman Key Generator

No.	Komponen	Name of Properties	Keterangan
1.	*TextBox1*	txtNilai_p	untuk menampilkan bilangan prima *p*
2.	*TextBox2*	txtNilai_q	untuk menampilkan bilangan prima *q*
3.	*TextBox3*	txtNilai_N	untuk menampilkan niai *N*
4.	*Button1*	btnBangkitkanKunci	untuk proses membangkitkan kunci
5.	*TextBox4*	txtKunciPublik	untuk menampilkan kunci publik
6.	*TextBox5*	txtKunciPrivat	untuk menampilkan kunci privat
7.	*Button2*	btnSimpanKunci	untuk menyimpan kunci publik dan privat

3.5.3 Rancangan Halaman Enkripsi dan Kompresi

Halaman enkripsi dan kompresi merupakan halaman yang berisikan proses enkripsi dengan algoritma *Rabin-p* dan proses kompresi dengan algoritma *Goldbach Code* pada file teks. Untuk rancangan halaman enkripsi dan kompresi dapar dilihat pada gambar 3.8.



Gambar 3.8 Rancangan Interface Halaman Enkripsi dan Kompresi

Berikut ini merupakan keterangan rancangan antarmuka pada halaman enkripsi dan kompresi seperti dijelaskan pada tabel 3.8.

Tabel 3.8 Keterangan Rancangan Interface Halaman Enkripsi dan Kompresi

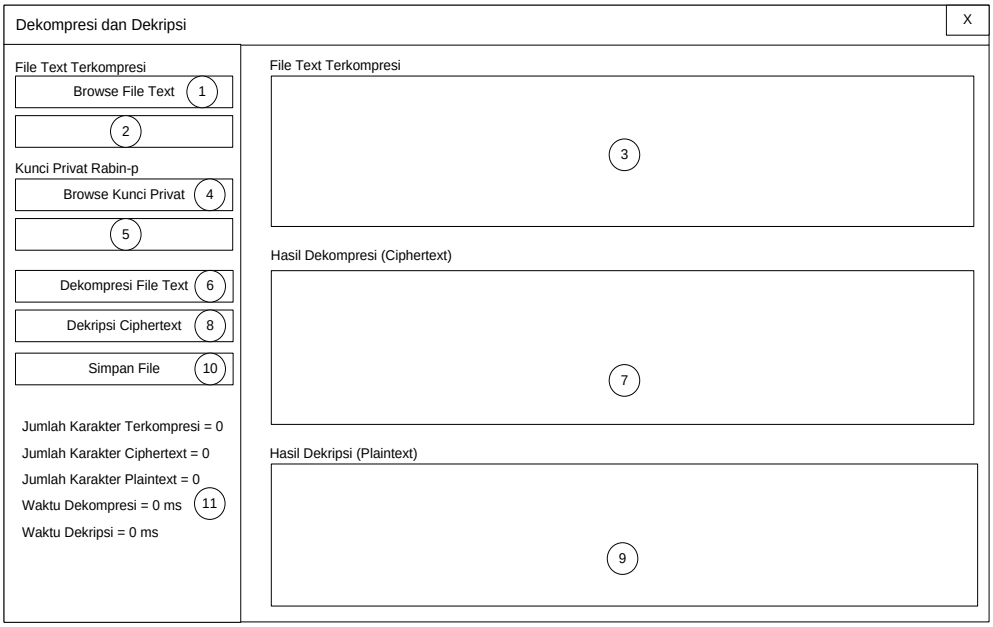
No.	Komponen	Name of Properties	Keterangan
1.	*Button1*	btnBrowseFile	untuk mengimport *file* teks
2.	*TextBox1*	txtDirFile	untuk menampilkan direktori *file* teks
3.	*RichTextBox1*	rtbPlaintext	untuk menampilkan isi dari *file* teks
4.	*Button2*	btnBrowseKunci	untuk mengimport kunci publik *Rabin-p*
5.	*TextBox2*	txtKunciPublik_N	untuk menampilkan kunci publik *Rabin-p*
6.	*Button3*	btnEnkripsi	untuk mengenkripsi *file* teks
7.	*RichTextBox2*	rtb*Ciphertext*	untuk menampilkan hasil enkripsi

Lanjutan Tabel 3.8 Keterangan Rancangan Interface Halaman Enkripsi dan Kompresi

8.	*Button4*	btnKompresi	untuk mengkompresi hasil enkripsi
9.	*RichTextBox3*	rtb*String*Bit*Ciphertext*	untuk menampilkan *string* bit *ciphertext*
10.	*RichTextBox4*	rtbTerkompresi	untuk menampilkan hasil kompresi
11.	*Button4*	btnSimpan	untuk menyimpan hasil kompresi
12.	*TextBox3*	txtSizePlaintext	untuk menampilkan ukuran *plaintext*
13.	*TextBox4*	txtSize*Ciphertext*	untuk menampilkan ukuran *ciphertext*
14.	*TextBox5*	txtSizeTerkompresi	untuk menampilkan ukuran terkompresi
15.	*TextBox6*	txtWaktuEnkripsi	untuk menampilkan waktu enkripsi
16.	*TextBox7*	txtWaktuKompresi	untuk menampilkan waktu kompresi
17.	*TextBox8*	txtRC	untuk menampilkan rasio kompresi
18.	*TextBox9*	txtCR	untuk menampilkan kompresi rasio
19.	*TextBox10*	txtSS	untuk menampilkan space savings

3.5.4 Rancangan Halaman Dekompresi dan Dekripsi

Halaman dekompresi dan dekripsi merupakan halaman yang berisikan proses dekompresi dengan algoritma *Goldbach Code* dan proses dekripsi dengan algoritma *Rabin-p* pada file teks. Untuk rancangan halaman dekompresi dan dekripsi dapar dilihat pada gambar 3.9.



Gambar 3.9 Rancangan Interface Halaman Dekompresi dan Dekripsi

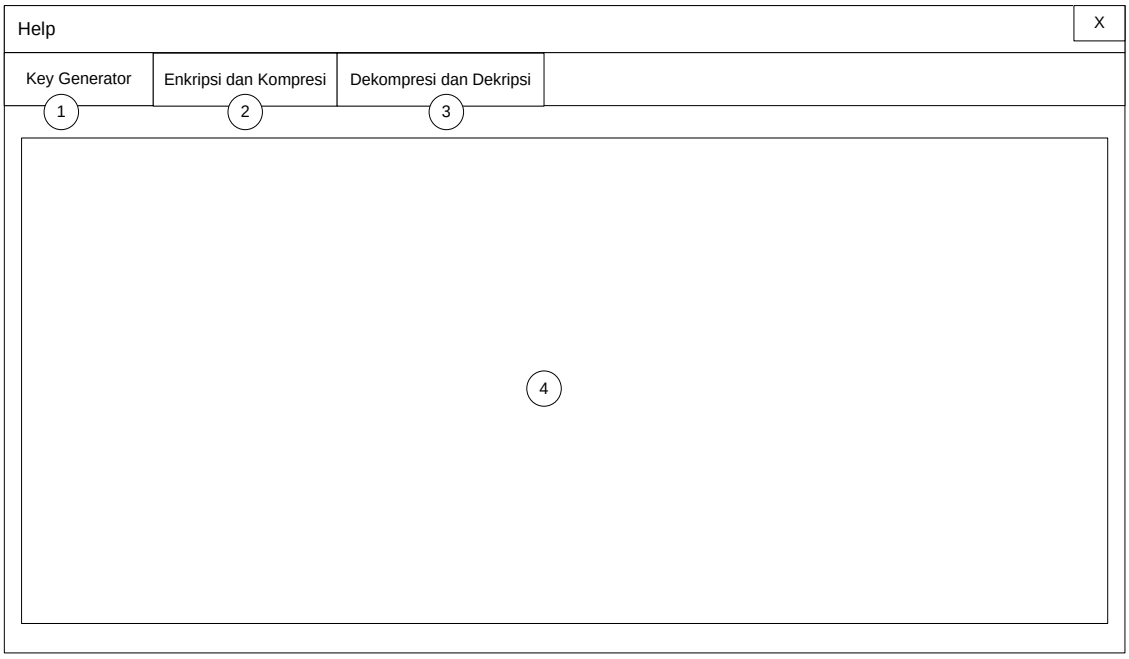
Berikut ini merupakan keterangan rancangan antarmuka pada halaman dekompresi dan dekripsi seperti dijelaskan pada tabel 3.9.

Tabel 3.9 Keterangan Rancangan Interface Halaman Dekompresi dan Dekripsi

No.	Komponen	Name of Properties	Keterangan
1.	*Button1*	btnBrowseFile	untuk mengimport *file* teks terkompresi
2.	*TextBox1*	txtDirFile	untuk menampilkan direktori *file* teks
3.	*RichTextBox1*	rtbTerkompresi	untuk menampilkan isi dari *file* teks terkompresi
4.	*Button2*	btnBrowseKunci	untuk mengimport kunci privat
5.	*TextBox2*	txtKunciPrivat_P	untuk menampilkan kunci privat
6.	*Button3*	btnDekompresi	untuk melakukan dekompresi *file* teks
7.	*RichTextBox2*	rtb*Ciphertext*	untuk menampilkan hasil dekompresi
8.	*Button3*	btnDekripsi	untuk mendekripsi hasil dekompresi
9.	*RichTextBox3*	rtbPlaintext	untuk menampilkan hasil dekripsi
10.	*RichTextBox4*	rtb*String*bitPlaintext	untuk menampilkan *string* bit plaintext
11.	*Button4*	btnSimpan	untuk menyimpan hasil dekripsi
12.	*TextBox3*	txtSizeTerkompresi	menampilkan ukuran terkompresi
13.	*TextBox4*	txtSize*Ciphertext*	menampilkan ukuran *ciphertext*
14.	*TextBox5*	txtSizePlaintext	menampilkan ukuran *plaintext*
15.	*TextBox6*	txtJumkarPlaintext	menampilkan jumlah karakter *plaintext*
16.	*TextBox7*	txtWaktuDekompresi	menampilkan waktu proses dekompresi
17.	*TextBox8*	txtWaktuDekripsi	menampilkan waktu proses dekripsi

3.5.5 Rancangan Halaman Help

Halaman *help* merupakan halaman yang berisikan bantuan untuk cara penggunaan sistem. Dengan adanya halaman ini pengguna akan lebih mengerti bagaimana cara dalam menjalankan sistem. Adapun tampilan untuk rancangan pada halaman *help* dapat dilihat pada gambar 3.10.



Gambar 3.10 Rancangan Interface Halaman Help

Berikut ini merupakan keterangan rancangan antarmuka halaman *help* seperti dijelaskan pada tabel 3.10.

Tabel 3.10 Keterangan Rancangan Interface Halaman Help

No.	Komponen	Name of Properties	Keterangan
1.	*TabControl1*	*tabPage1*	untuk menampilkan panduan penggunaan pada halaman bangkitkan kunci
2,	*TabControl1*	*tabPage2*	untuk menampilkan panduan penggunaan pada halaman enkripsi dan kompresi
3.	*TabControl1*	*tabPage3*	untuk menampilkan panduan penggunaan pada halaman dekompresi dan dekripsi
4.	*TextBox1*	*textBox1*	untuk menampilkan informasi dari menu yang dipilih

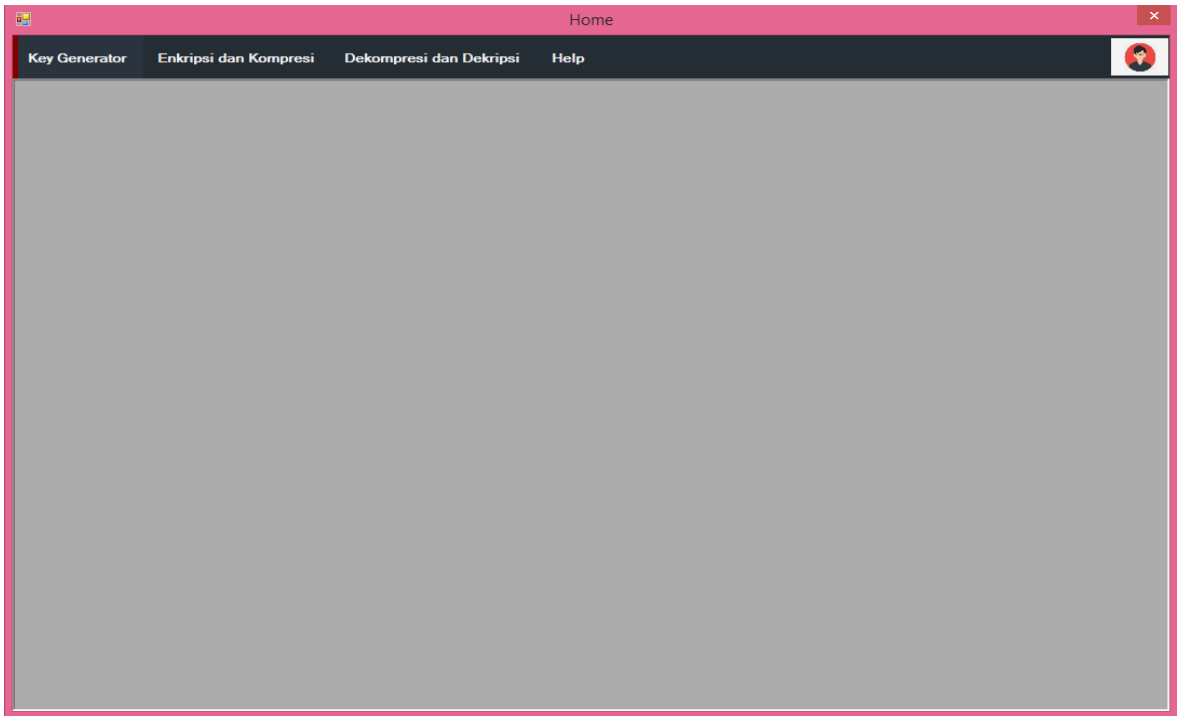
BAB 4
IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1 Implementasi Sistem

Implementasi sistem merupakan lanjutan dari tahap analisis dan perancangan sistem. Pada penelitian ini, sistem kriptokompresi yang dibuat dibangun dengan menggunakan bahasa pemograman *C#*. Berdasarkan ilustrasi pada tahap rancangan sistem yang telah dipaparkan, sistem kriptokompresi ini terdiri dari lima buah *form*, yaitu *form* utama, *form key generator*, *form* enkripsi dan kompresi, *form* dekompresi dan dekripsi, dan *form help*.

4.1.1 Implementasi Form Utama

Form utama merupakan tampilan utama pada saat aplikasi dijalankan. Pada *form* utama terdapat empat buah menu yang terdiri dari menu *key generator* yang berfungsi untuk melakukan pembangkitan kunci (*key generator*) dari algoritma *Rabin-p*, menu enkripsi dan kompresi, menu dekompresi dan dekripsi, dan menu *help*. Gambar 4.1 merupakan tampilan dari *form* utama.



Gambar 4.1 Tampilan Form Utama

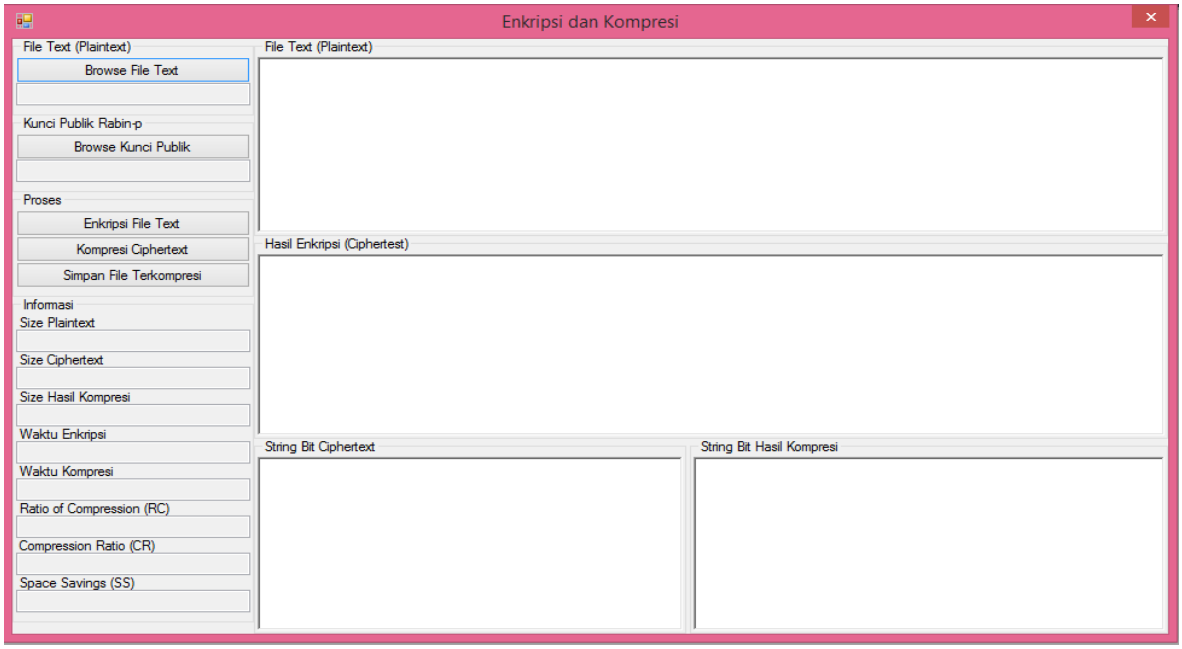
4.1.2 Implementasi Form Key Generator

Form ini berfungsi untuk melakukan proses pembangkitan kunci (*key generator*) dari algoritma *Rabin-p*. Kunci yang dibangkitkan terdiri dari sepasang kunci, yakni kunci publik (*public key*) yang digunakan untuk melakukan proses enkripsi pada *file* teks dan kunci privat (*private key*) yang digunakan untuk melakukan proses dekripsi. Gambar 4.2 merupakan tampilan dari *form key generator*.

Gambar 4.2 Tampilan Form Key Generator

4.1.3 Implementasi Form Enkripsi dan Kompresi

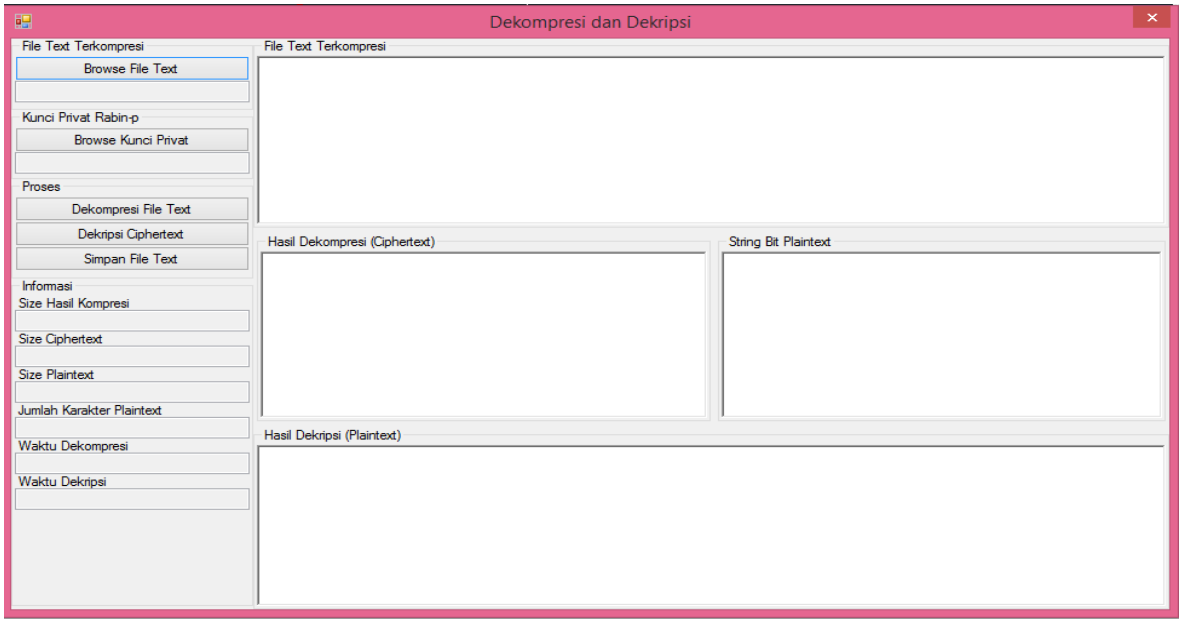
Form ini berfungsi untuk melakukan proses enkripsi pada *file* teks yang diinputkan oleh pengguna dengan menggunakan kunci publik dari algoritma asimetris *Rabin-p* yang telah dibangkitkan sebelumnya. Hasil enkripsi (*ciphertext*) kemudian akan dikompresi yang bertujuan untuk memperkecil ukuran data pada *ciphertext* dengan menggunakan algoritma kompresi *Goldbach Code*. Setelah *ciphertext* berhasil dikompresi, sistem akan menampilkan hasil kompresi beserta informasi mengenai jumlah *bit*, *Ratio of Compression*, *Compression of Ratio*, *Space Savings*, waktu kompresi. Gambar 4.3 merupakan tampilan dari *form* enkripsi dan kompresi.



Gambar 4.3 Tampilan Form Enkripsi dan Kompresi

4.1.4 Implementasi Form Dekompresi dan Dekripsi

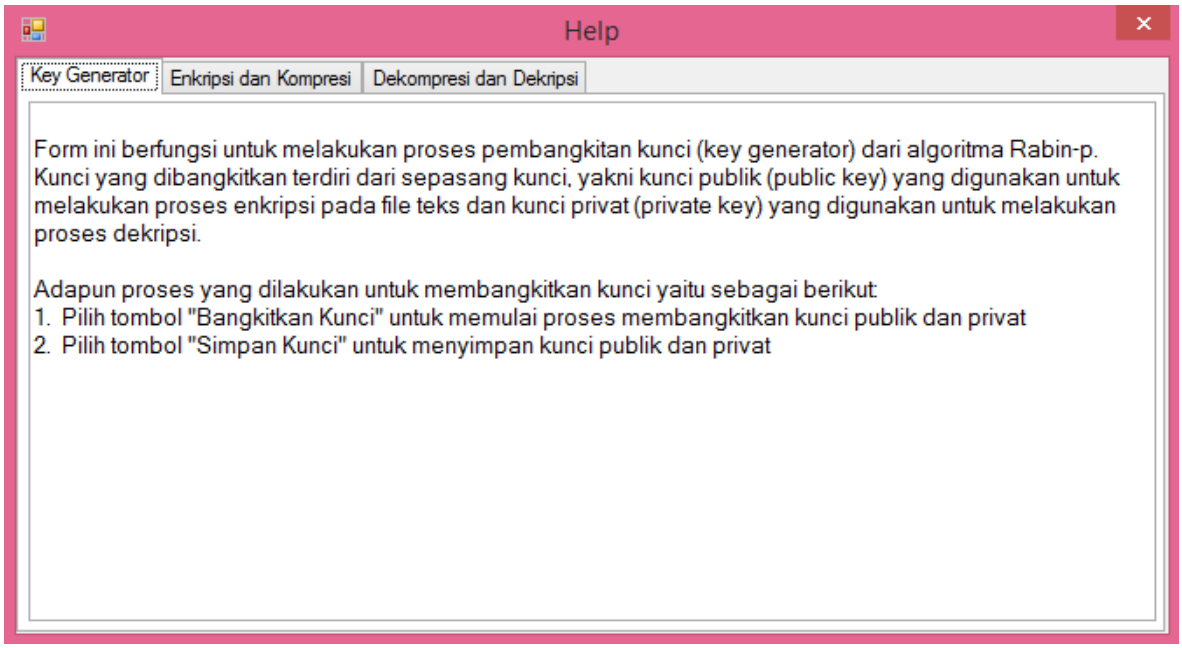
Form ini berfungsi untuk melakukan proses dekompresi pada *file* teks hasil kompresi sebelumnya dengan menggunakan algoritma *Goldbach Code*. Hasil dekompresi selanjutnya akan di dekripsi untuk mendapatkan kembali *file* teks aslinya dengan menggunakan kunci privat algoritma *Rabin-p* yang telah dibangkitkan sebelumnya. Gambar 4.4 merupakan tampilan dari *form* dekompresi dan dekripsi.



Gambar 4.4 Tampilan Form Dekompresi dan Dekripsi

4.1.5 Implementasi Form Help

Form ini berfungsi untuk menampilkan informasi mengenai alur dari penggunaan aplikasi sistem kriptokompresi. Gambar 4.5 merupakan tampilan dari *form help*.



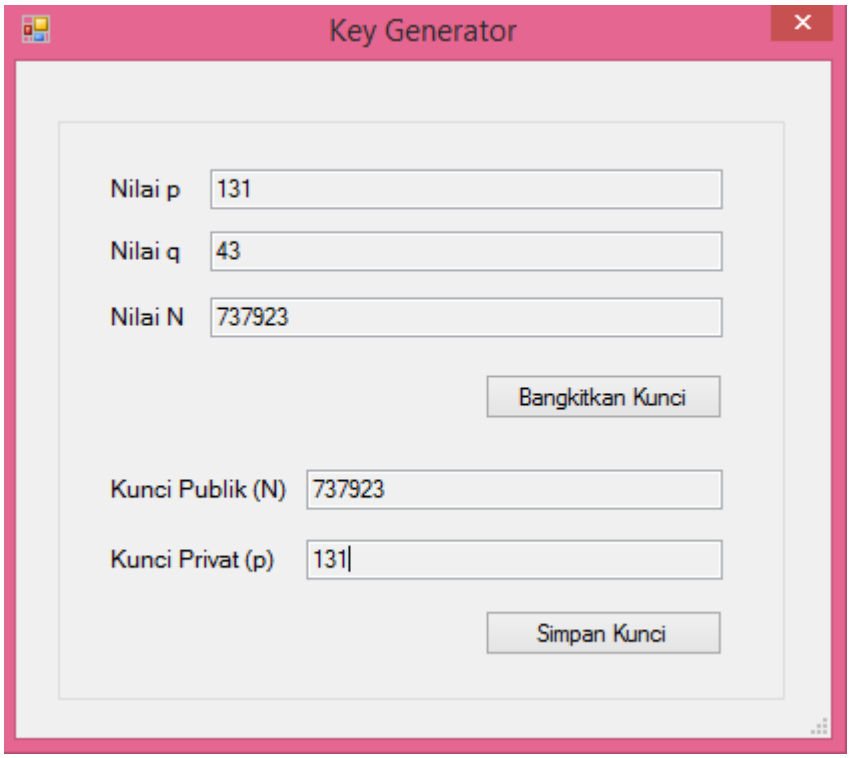
Gambar 4.5 Tampilan Form Help

4.2 Pengujian Sistem

Pengujian sistem adalah tahapan yang bertujuan untuk mengidentifikasi serta menguji sistem yang telah dibangun. Pada tahap ini, sistem akan di identifikasi, apakah sistem yang dibangun telah memenuhi kriteria yang telah ditentukan pada tahap sebelumnya dan apakah sistem telah berhasil menjalankan fungsi-fungsinya dengan benar. Pengujian ini dilakukan dalam lima proses yakni proses pembangkitan kunci (*key generator*), proses enkripsi, proses kompresi, proses dekompresi dan proses dekripsi.

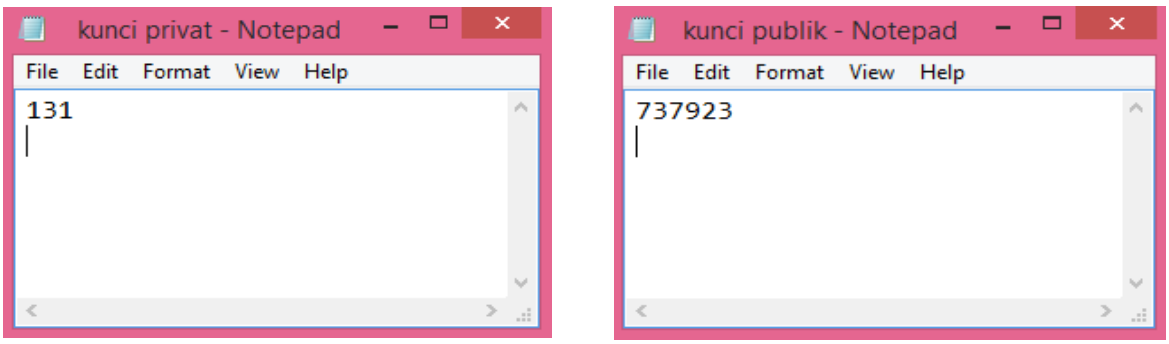
4.2.1 Pengujian Proses Key Generator

Proses membangkitkan kunci publik (*public key*) dan kunci privat (*private key*) algoritma asimetris *Rabin-p* dimulai dengan memilih tombol “Bangkitkan Kunci”, sistem akan mengacak bilangan prima p dan q yang digunakan untuk menghasilkan pasangan kunci publik dan kunci privat. Adapun hasil dari pasangan kunci yang dibangkitkan seperti terlihat pada gambar 4.6.



Gambar 4.6 Hasil Pengujian Pasangan Kunci yang Dibangkitkan

Sesuai gambar 4.6, nilai bilangan prima yang dihasilkan setelah diacak untuk $p = 131$ dan $q = 43$. Sedangkan untuk nilai $N = 737923$ yang merupakan hasil dari p^2q. Kunci publik $N = 737923$ dan kunci privat merupakan $p = 131$. Pasangan kunci yang berhasil dibangkitkan selanjutnya akan disimpan untuk keperluan proses enkripsi dan dekripsi. Untuk menyimpan pasangan kunci dapat dilakukan dengan memilih tombol "Simpan Kunci" dan selanjutnya sistem akan menampilkan *diaog box* untuk proses menyimpan kunci. Adapun tampilan dari pasangan kunci publik dan kunci privat *Rabin-p* setelah berhasil disimpan dapat dilihat pada gambar 4.7.

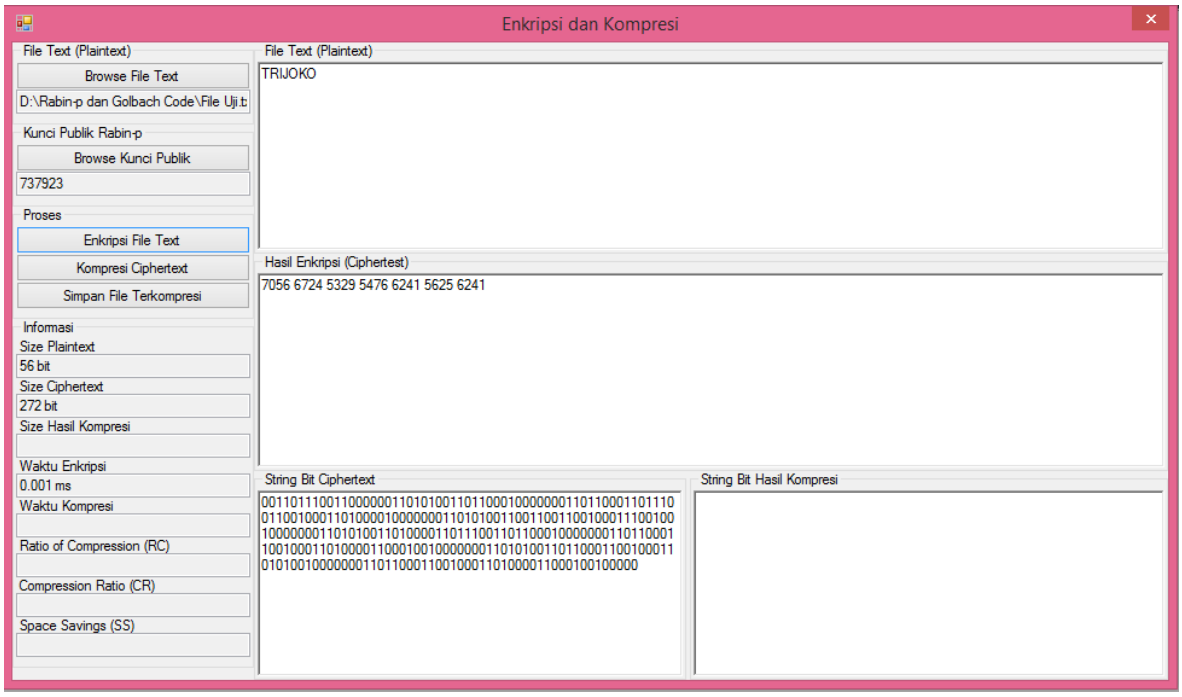


(a) (b)

Gambar 4.7 Hasil Kunci Algoritma Rabin-p (a) Kunci Publik (b) Kunci Privat

4.2.2 Pengujian Proses Enkripsi

Setelah memilih tombol "Enkripsi" pada halaman utama, maka sistem akan menampilkan halaman enkripsi yang digunakan untuk proses enkripsi dan kompresi. Proses enkripsi dimulai dengan memilih tombol "*Browse File Text*" untuk mengimport *file* teks dengan memilih *format file .txt* atau *.docx* yang akan dienkripsi yang selanjutnya sistem akan menampilkan preview isi pada *file* teks. Setelah itu dilanjutkan dengan memasukkan kunci publik *Rabin-p* dengan memilih tombol "Browse Kunci Publik" dan sistem akan menampilkan kunci yang di inputkan. Setelah memasukkan *file* teks dan kunci publik, maka langkah selanjutnya adalah memilih tombol "Enkripsi File Text" dan sistem akan menampilkan hasil enkripsi (*ciphertext*) seperti terlihat pada gambar 4.8.



Gambar 4.8 Hasil Pengujian Enkripsi

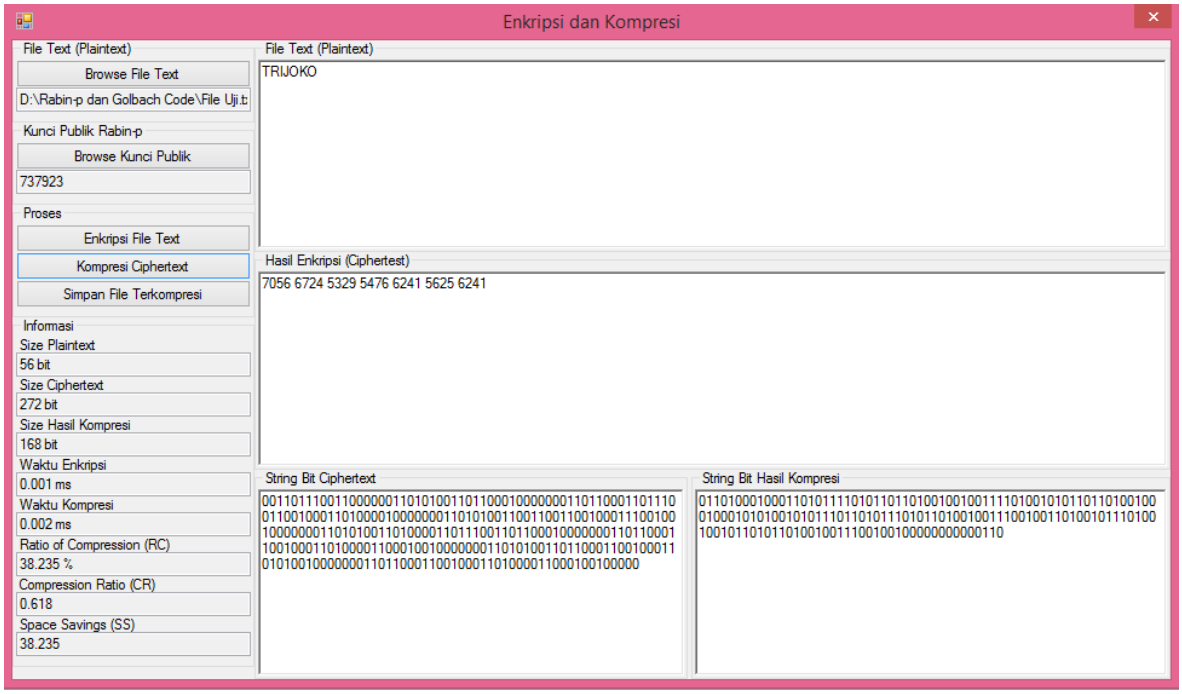
Gambar 4.8 merupakan hasil pengujian enkripsi pada *file* teks dengan menggunakan kunci publik algoritma *Rabin-p*. Pada contoh pengujian ini menggunakan *plaintext* "TRIJOKO" dan kunci publik $N = 737923$. *File* teks yang di uji menggunakan sampel data dengan *size* 56 *bit*. Karakter (*string*) yang terdapat pada *file* teks yang telah di enkripsi akan menghasilkan *ciphertext* dalam bentuk bilangan desimal "7056 6724 5329 5476 6241 5625 6241" dengan *size* 272 *bit* serta estimasi waktu enkripsi selama 0.001 ms (*millisecond*).

Dapat dilihat bahwa panjang karakter dari proses enkripsi dengan menggunakan algoritma *Rabin-p* akan menghasilkan *ciphertext* yang dua kali lebih besar dari *file* aslinya

(*plaintext*). Oleh karena itu algoritma kompresi diperlukan guna memperkecil ukuran data dari hasil proses enkripsi sehingga dapat menghemat ruang penyimpanan data (*storage*) jika akan disimpan. Pada penelitian ini, hasil enkripsi berupa *ciphertext* kemudian akan dikompresi dengan menggunakan algoritma *lossless compression Goldbach Code* untuk tujuan memperkecil ukuran dari *ciphertext*.

4.2.3 Pengujian Proses Kompresi

Proses kompresi merupakan lanjutan dari proses enkripsi sebelumnya yang berfungsi untuk memperkecil ukuran dari hasil enkripsi (*ciphertext*). Proses kompresi dapat dilakukan dengan terlebih dahulu memilih tombol "Kompresi Ciphertext" dan sistem akan menampilkan hasil kompresi seperti terlihat pada gambar 4.9.



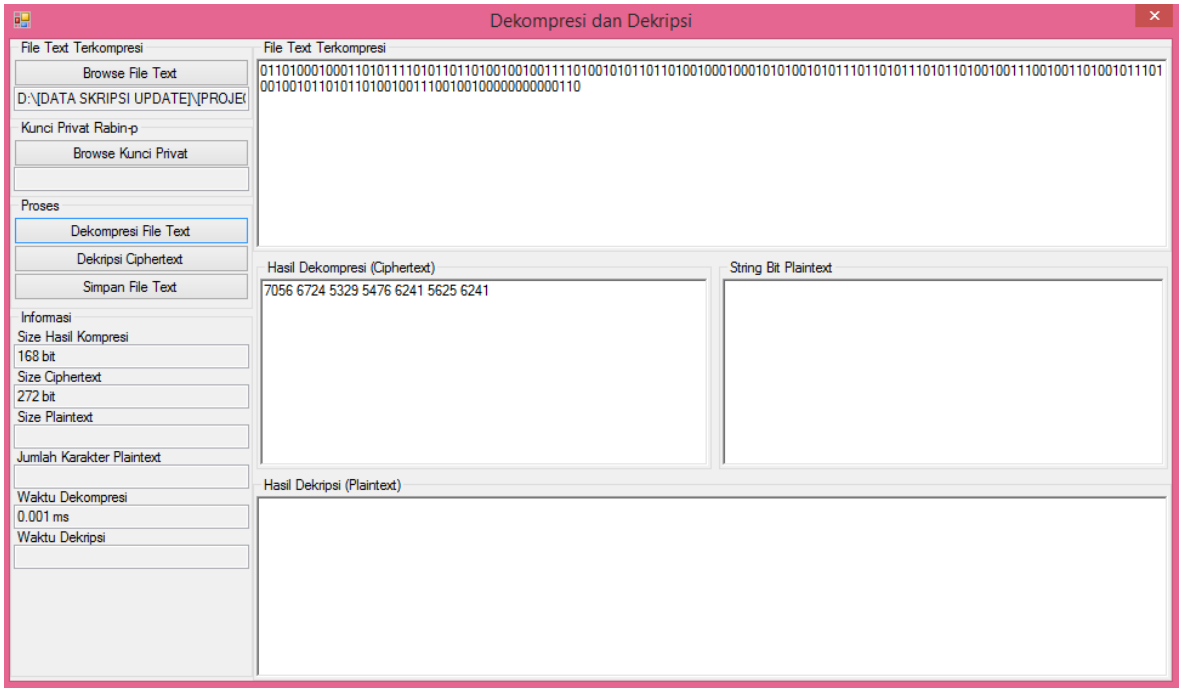
Gambar 4.9 Hasil Pengujian Kompresi

Pada gambar 4.9, setelah *ciphertext* dikompresi dengan menggunakan algoritma *Goldbach Code* maka ukuran datanya dapat diperkecil, yang mana ukuran *ciphertext* sebelum dikompresi adalah 272 *bit* dan setelah dikompresi menjadi 168 *bit*. Berdasarkan parameter yang digunakan dalam proses kompresi pada penelitian ini, diperoleh *Ratio of Compression* (RC) sebesar 38,235, *Compression of Ratio* (CR) sebesar 0,618, *Space Savings* (SS) sebesar 38,235% dan waktu kompresi selama 0.002 ms (*millisecond*). Hasil dari kompresi kemudian akan disimpan untuk tujuan pengjian pada proses dekompresi. Untuk

menyimpan hasil kompresi dapat dilakukan dengan memilih tombol “Simpan File Terkompresi” dan selanjutnya sistem akan menampilkan *dialog box* untuk proses menyimpan hasil kompresi. Hasil proses kompresi berbentuk *string bit* dan akan disimpan dengan format *file .gc* (akronim dari algoritma *Goldbach Code*).

4.2.4 Pengujian Proses Dekompresi

Proses dekompresi merupakan tahapan yang dilakuan untuk mengembalikan data dari hasil kompresi. Proses dekompresi dimulai dengan memilih tombol “*Browse File Text*” untuk mengimport *file* hasil kpmpresi sebelumnya dan sistem akan menampilkan preview dari *file* teks terkompresi. Selanjutnya pilih tombol “Dekompresi File Text” untuk memulai proses dan sistem akan menampilkan hasil dekompresi seperti terlihat pada gambar 4.10.



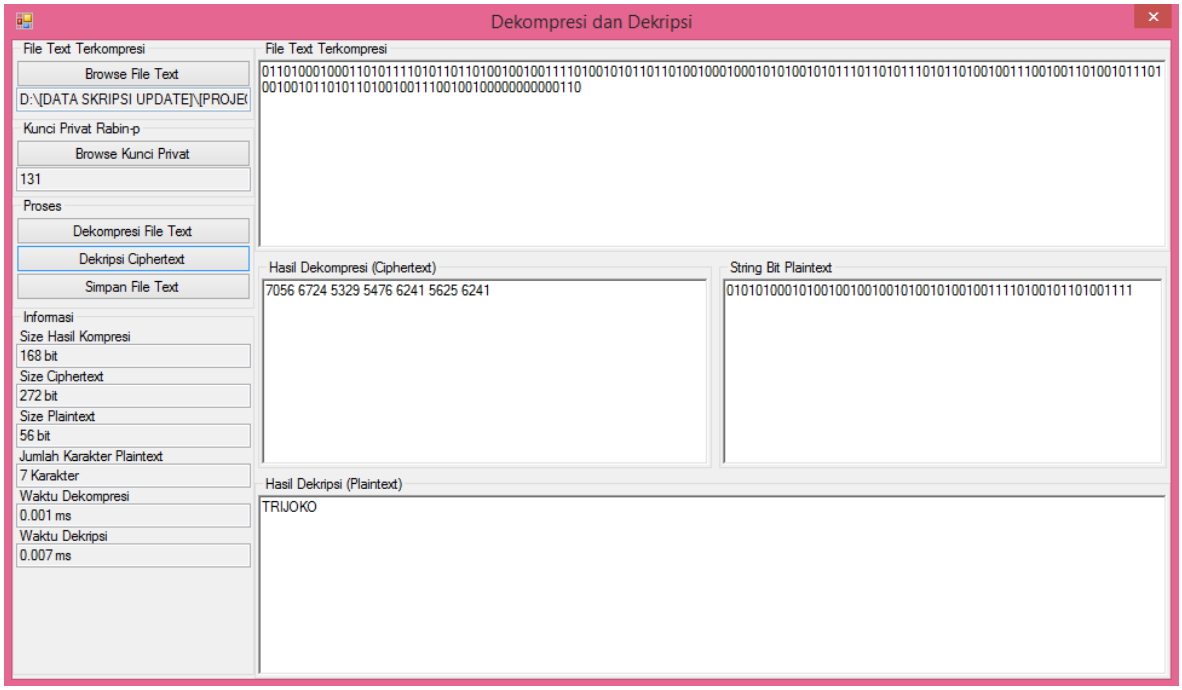
Gambar 4.10 Hasil Pengujian Dekompresi

Gambar 4.10 merupakan hasil pengujian dari proses dekompresi yang bertujuan untuk mengembalikan hasil kompresi sebelumnya yang merupakan *file* hasil enkripsi (*ciphertext*). Pengujian dekompresi menunjukkan bahwa *size* kompresi sebelumnya adalah 168 *bit* dan setelah di dekompresi akan menghasilkan size 272 *bit* yang sama dengan *size ciphertext* yang telah dikompresi sebelumnya dengan waktu dekompresi selama 0.001 ms (*millisecond*). Hal ini menunjukkan bahwa fungsional sistem berjalan sesuai dengan yang

diharapkan, yang mana setelah proses dekompresi, jumlah *bit* dalam keseluruhan *file* teks hasil dekompresi sama persis dengan jumlah *bit* pada *file* teks aslinya.

4.2.5 Pengujian Proses Dekripsi

Setelah melakukan dekompresi dengan algoritma *Goldbach Code* maka akan menghasilkan *ciphertext* yang selanjutnya akan di dekripsi dengan menggunakan kunci privat algoritma *Rabin-p*. Proses dekripsi dimulai dengan memasukkan kunci privat algoritma *Rabin-p* dengan memilih tombol "Browse Kunci Privat". Setelah itu pilih tombol "Dekripsi Ciphertext" dan sistem akan menampilkan hasilnya seperti terlihat pada gambar 4.11.

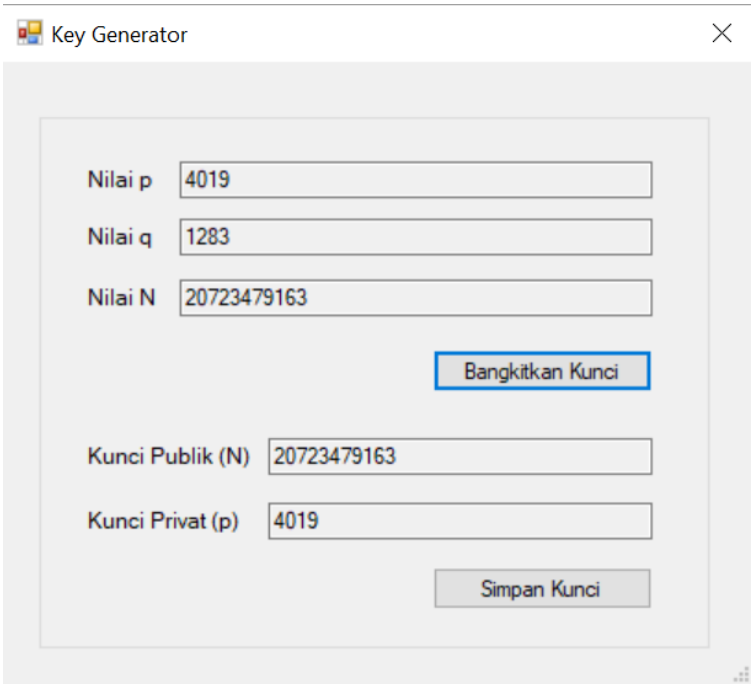


Gambar 4.11 Hasil Pengujian Dekripsi

Gambar 4.11 merupakan hasil pengujian dari proses dekripsi setelah dilakukan dekompresi sebelumnya. Hasil dekompresi akan menghasilkan *ciphertext*, sehingga perlu didekripsi untuk mendapatkan isi dari *file* teks aslinya. Setelah proses dekripsi jumlah *bit* dan panjang karakter hasil dekripsi (*plaintext*) sama persis dengan *file* teks aslinya yaitu 56 *bit* dengan waktu dekripsi selama 0.007 ms (*millisecond*).

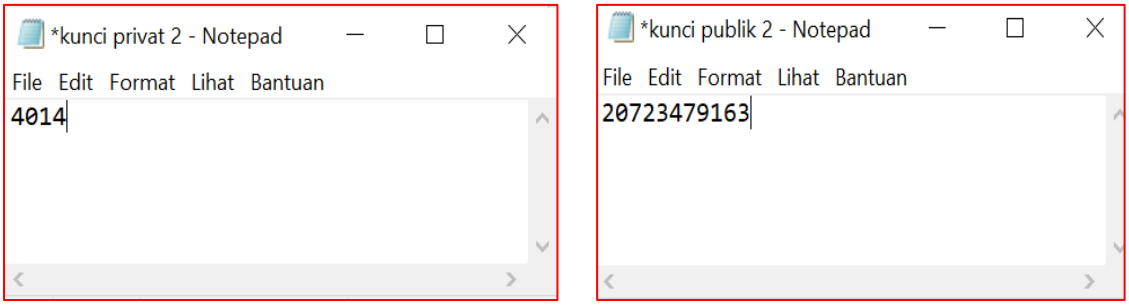
4.2.6 Pengujian Enkripsi dan Kompresi Menggunakan Kunci Berbeda

Proses membangkitkan kunci publik (*public key*) dan kunci privat (*private key*) algoritma asimetris *Rabin-p* dimulai dengan memilih tombol "Bangkitkan Kunci", sistem akan mengacak bilangan prima p dan q yang digunakan untuk menghasilkan pasangan kunci publik dan kunci privat. Adapun hasil dari pasangan kunci yang dibangkitkan seperti terlihat pada gambar 4.12



Gambar 4.12 Hasil Pasangan kunci yang berbeda di bangkitkan

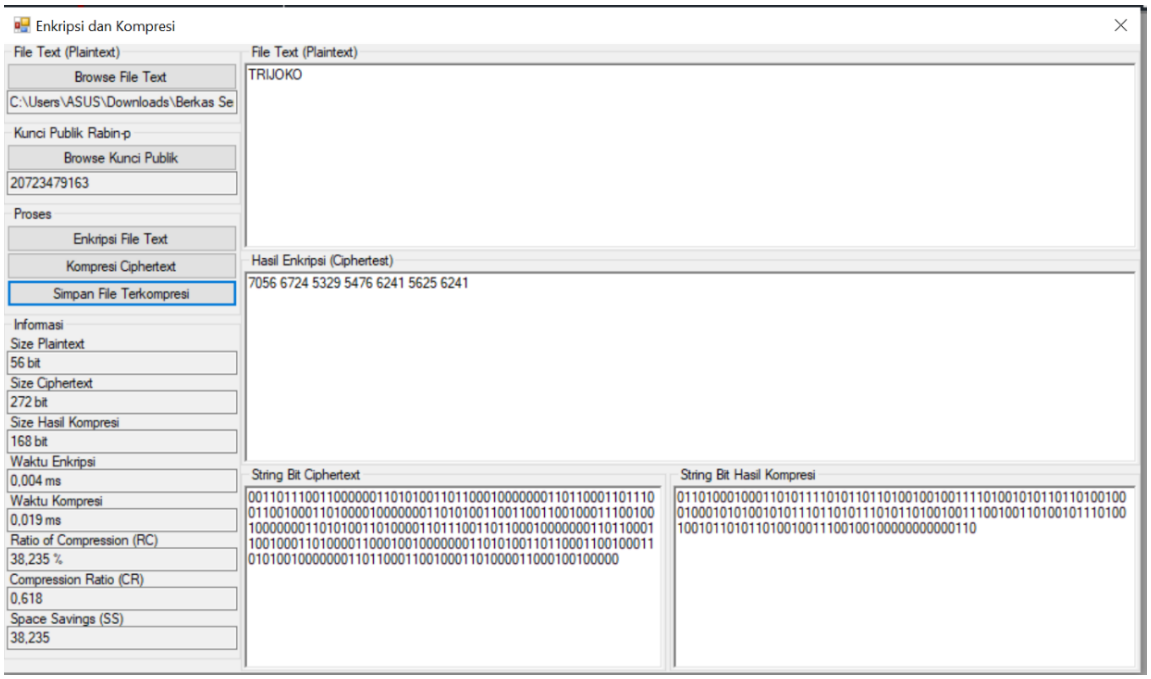
Sesuai gambar 4.12 nilai bilangan prima yang dihasilkan setelah diacak untuk $p = 4019$ dan $q = 1283$. Sedangkan untuk nilai $N = 20723479163$ yang merupakan hasil dari p^2q. Kunci publik $N = 20723479163$ dan kunci privat merupakan $p = 4019$. Pasangan kunci yang berhasil dibangkitkan selanjutnya akan disimpan untuk keperluan proses enkripsi dan dekripsi. Untuk menyimpan pasangan kunci dapat dilakukan dengan memilih tombol "Simpan Kunci" dan selanjutnya sistem akan menampilkan *diaog box* untuk proses menyimpan kunci. Adapun tampilan dari pasangan kunci publik dan kunci privat *Rabin-p* setelah berhasil disimpan dapat dilihat pada gambar 4.13.



Gambar 4.13 Hasil kunci Algoritma Rabin-p Kunci Publik dan Kunci Privat

4.2.7 Pengujian Proses Enkripsi dan Kompresi

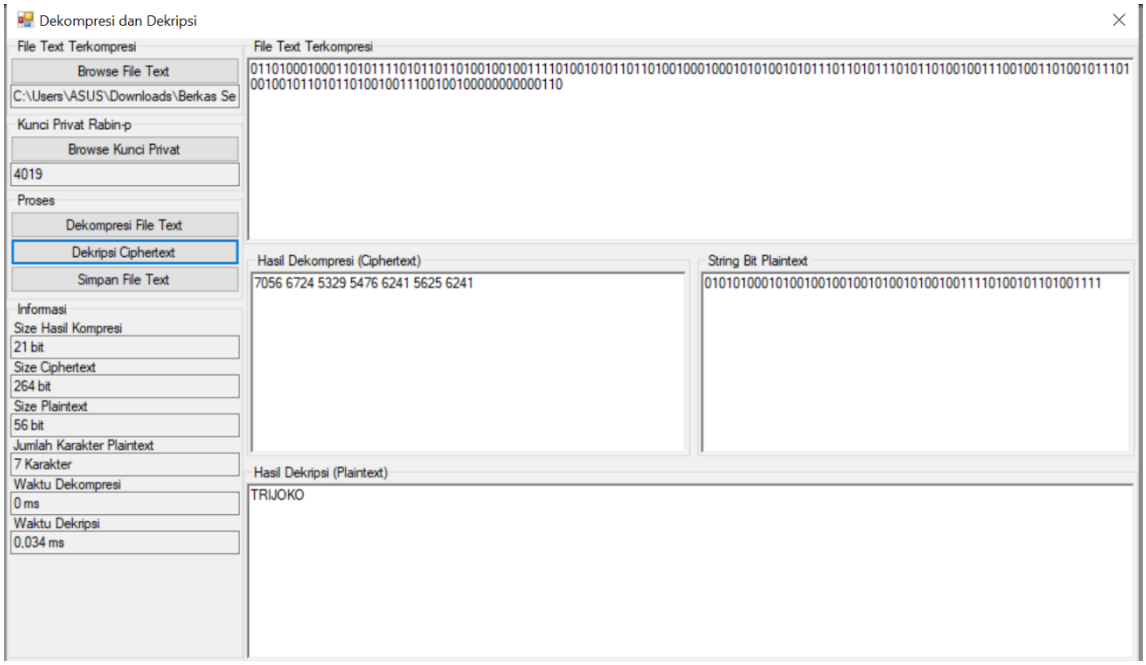
Proses Enkripsi dimulai dengan memilih tombol "*Browse File Text*" untuk mengimport *file* teks dengan memilih *format file .txt* atau *.docx* yang akan dienkripsi yang selanjutnya sistem akan menampilkan preview isi pada *file* teks. Setelah itu dilanjutkan dengan memasukkan kunci publik *Rabin-p* dengan memilih tombol "Browse Kunci Publik" dan sistem akan menampilkan kunci yang di inputkan. Setelah memasukkan *file* teks dan kunci publik, maka langkah selanjutnya adalah memilih tombol "Enkripsi File Text" dan pilih juga tombol kompresi pada chipertext dan sistem akan menampilkan hasil enkripsi dan kompresi pada (*ciphertext*) seperti terlihat pada gambar 4.13



Gambar 4.14 Hasil pengujian Enkripsi dan Kompresi

4.2.8 Pengujian Dekompresi dan Dekripsi

Proses dekompresi dimulai dengan memilih tombol "*Browse File Text*" untuk mengimport *file* hasil kpmpresi sebelumnya dan sistem akan menampilkan preview dari *file* teks terkompresi. Selanjutnya pilih tombol "Dekompresi File Text" untuk memulai proses lalu pilih tombol "Dekripsi Ciphertext" dan sistem akan menampilkan hasil dekompresi dan Dekripsi seperti terlihat pada gambar 4.13.



Gambar 4.15 Hasil pengujian Dekompresi dan Dekripsi

4.2.9 Perbandingan Hasil Pengujian Enkripsi dan Kompresi

Pengujian enkripsi dan kompresi pada penelitian ini dilakukan untuk mengetahui perbandingan hasil penggunaan kunci publik *Rabin-p* yang berbeda terhadap *size* hasil enkripsi dan kompresi. Adapun hasil pengujiannya dapat disajikan pada tabel 4.1.

Tabel 4.1 Perbandingan Hasil Pengujian Enkripsi dan Kompresi

Data Uji		**Hasil Enkripsi**			**Hasil Kompresi**				
Jumlah Karater	**Size (bit)**	**Kunci Publik (N)**	**Size (bit)**	**RT (ms)**	**Size (bit)**	**RT (ms)**	**RC**	**CR**	**SS**
7	56	737923	272	0.001	168	0.002	38.235	0.618	38.235
7	56	20723479163	272	0.004	168	0.019	38.235	0.618	38.235

Lanjutan Tabel 4.2 Hasil Pengujian Dekompresi dan Dekripsi

Data Uji		Hasil Enkripsi			Hasil Kompresi				
Jumlah Karater	**Size (bit)**	**Kunci Publik (N)**	**Size (bit)**	**RT (ms)**	**Size (bit)**	**RT (ms)**	**RC**	**CR**	**SS**
7	56	37158745111	272	0.004	168	0.005	38.235	0.618	38.235
7	56	16216844387	272	0.001	168	0.003	38.235	0.618	38.235
7	56	38940931963	272	0.001	168	0.001	38.235	0.618	38.235

Tabel 4.1 merupakan hasil perbandingan pengujian enkripsi dan kompresi dengan menggunakan kunci publik *Rabin-p* yang berbeda. Pengujian dilakukan menggunakan data uji dengan jumlah karakter dan *size* yang sama, namun kunci publik *Rabin-p* yang berbeda. Adapun perbandingan hasil pengujiannya dapat dijelaskan sebagai berikut:

1. Pada pengujian pertama dengan *size* 56 *bit* menggunakan kunci publik (N) sepanjang 6 digit (737923) akan menghasilkan *ciphertext* dengan *size* sebesar 272 *bit* dan estimasi waktu enkripsi selama 0.001 ms (*millisecond*). Setelah dikompresi akan menghasilkan *size* sebesar 168 *bit* dengan *Ratio of Compression* (RC) sebesar 38,235, *Compression of Ratio* (CR) sebesar 0,618, *Space Savings* (SS) sebesar 38,235% dan waktu kompresi selama 0.002 ms (*millisecond*).
2. Pada pengujian kedua dengan *size* 56 *bit* menggunakan kunci publik (N) sepanjang 11 digit (20723479163) akan menghasilkan *ciphertext* dengan *size* sebesar 272 *bit* dan estimasi waktu enkripsi selama 0.004 ms (*millisecond*). Setelah dikompresi akan menghasilkan *size* sebesar 168 *bit* dengan *Ratio of Compression* (RC) sebesar 38,235, *Compression of Ratio* (CR) sebesar 0,618, *Space Savings* (SS) sebesar 38,235% dan waktu kompresi selama 0.019 ms (*millisecond*).

Berdasarkan hasil pengujian pada tabel 4.1 dapat disimpulkan bahwa perbedaan jumlah digit pada kunci publik *Rabin-p* akan tetap menghasilkan *size* hasil enkripsi dan kompresi yang sama persis, namun yang membedakan hanya terletak pada waktu proses (*running time*) yang dibutuhkan selama proses enkripsi dan kompresi. Semakin besar digit kunci publik *Rabin-p* yang digunakan maka semakin lama juga waktu enkripsi yang dibutuhkan karena proses kalkulasi menggunakan angka yang besar.

4.3.0 Hasil Pengujian Dekompresi dan Dekripsi

Pengujian dekompresi dan dekripsi pada penelitian ini dilakukan untuk mengetahui apakah data yang sudah di enkripsi dan dikompresi dapat dikembalikan sesuai dengan size data aslinya. Pada pengujian ini juga akan menggunakan kunci privat *Rabin-p* yang berbeda. Adapun hasil pengujiannya dapat disajikan pada tabel 4.2.

Tabel 4.2 Hasil Pengujian Dekompresi dan Dekripsi

Hasil Dekompresi		**Hasil Dekripsi**			**Data Uji**	
Size (bit)	**RT (ms)**	**Kunci Privat (p)**	**Size (bit)**	**RT (ms)**	**Size (bit)**	**Jumlah Karater**
272	0.001	131	56	0.007	56	7
272	0.001	4014	56	0.014	56	7
272	0.002	4447	56	0.008	56	7
272	0	3119	56	0.019	56	7
272	0	4567	56	0.027	56	7

Tabel 4.2 merupakan hasil perbandingan pengujian dekompresi dan dekripsi menggunakan kunci privat *Rabin-p* yang berbeda. Pada pengujian pertama dengan kunci privat (p) sepanjang 3 digit (131) akan menghasilkan *size* hasil dekompresi dan dekripsi yang sama persis dengan menggunakan kunci privat (p) sepanjang 4 digit (4014). Hal ini menunjukkan bahwa *size* dari proses dekompresi dan dekripsi dapat dikembalikan sesuai dengan *size* dan panjang karater pada data uji.

4.3.1 Pengujian Blackbox Testing

Pengujian *blackbox* (*blackbox testing*) adalah salah satu metode pengujian perangkat lunak yang berfokus pada sisi fungsionalitas). Pengujian dengan metode *blackbox testing* dilakukan dengan cara memberikan sejumlah *input* pada program. *Input* tersebut kemudian diproses sesuai dengan kebutuhan fungsionalnya untuk melihat apakah program aplikasi dapat menghasilkan *output* yang sesuai dengan yang diinginkan dan sesuai pula dengan fungsi dasar dari program tersebut. Apabila dari *input* yang diberikan, proses dapat menghasilkan *output* yang sesuai dengan kebutuhan fungsionalnya, maka program yang dibuat sudah benar, tetapi apabila *output* yang dihasilkan tidak sesuai dengan kebutuhan

fungsionalnya, maka masih terdapat kesalahan pada program tersebut, dan selanjutnya dilakukan penelusuran perbaikan untuk memperbaiki kesalahan yang terjadi.

Adapun hasil dari pengujian sistem dengan menggunakan metode *blackbox testing* dapat diuraikan sebagai berikut:

1. *Black Box Testing Key Generator*

   Hasil *black box testing* pada proses *key generator* dapat disajikan pada tabel 4.3.

Tabel 4.3 Black Box Testing Key Generator

No.	Kasus Uji	Langkah Uji	Hasil	Status
1.	Bangkitkan kunci	membangkitkan sepasang kunci dengan memilih tombol "Bangkitkan Kunci"	sistem dapat menampilkan pasangan kunci publik dan kunci privat melalui dua buah bilangan prima *p* dan *q*	Berhasil
2.	Simpan kunci	menyimpan kunci publik dan kunci privat dengan memilih tombol "Simpan Kunci Publik"	sistem dapat menyimpan *file* kunci publik kedalam format *file .pubkey* dan kunci privat kedalam format *file .privkey*	Berhasil

Tabel 4.3 pengujian *black box testing* proses bangkitkan kunci, kasus pengujian yang dilakukan antara lain "bangkitkan kunci" dan "simpan kunci" dengan status hasil pengujian berhasil.

2. *Black Box Testing* Enkripsi

   Hasil *black box testing* pada proses enkripsi yang dapat disajikan pada tabel 4.4.

Tabel 4.4 Black Box Testing Enkripsi

No.	Kasus Uji	Langkah Uji	Hasil	Status
1.	*Input file text*	mengimport *file text* dengan memilih tombol "*Browse File Text*"	sistem hanya dapat menerima format *file .txt* dan *.docx*, sistem dapat menampilkan isi dan *size* dari *file text*	Berhasil

Lanjutan Tabel 4.4 Black Box Testing Enkripsi

No.	Kasus Uji	Langkah Uji	Hasil	Status
2.	*Input file* kunci	mengimport *file* kunci publik dengan memilih tombol "*Browse* Kunci Publik"	sistem hanya dapat menerima format *file .pubkey* dan sistem dapat menampilkan isi dari *file* kunci publik *Rabin-p*	Berhasil
3.	Enkripsi	mengenkripsi *file text* dengan memilih tombol "Enkripsi *File Text*"	sistem dapat menampilkan hasil enkripsi (*ciphertext*), *string bit ciphertext*, *size ciphertext*, dan waktu proses enkripsi	Berhasil

Tabel 4.4 pengujian *black box testing* proses enkripsi, kasus pengujian yang dilakukan antara lain "*input file text*", "*input file* kunci", dan "enkripsi" dengan status hasil pengujian berhasil.

3. *Black Box Testing* Kompresi

   Hasil *black box testing* pada proses kompresi dapat disajikan pada tabel 4.5.

Tabel 4.5 Black Box Testing Kompresi

No.	Kasus Uji	Langkah Uji	Hasil	Status
1.	Kompresi	melakukan kompresi pada *ciphertext* dengan memilih tombol "Kompresi *Ciphertext*"	sistem dapat menampilkan hasil kompresi berbentuk *string bit*, *size* hasil kompresi, *RC*, *CR*, *SS* dan waktu proses kompresi	Berhasil
2.	Simpan	menyimpan hasil kompresi dengan memilih tombol "Simpan File Terkompresi"	sistem dapat menyimpan hasil kompresi kedalam format *.gc*	Berhasil

Tabel 4.5 pengujian *black box testing* proses kompresi, kasus pengujian yang dilakukan antara lain "kompresi" dan "simpan" dengan status hasil pengujian berhasil.

4. *Black Box Testing* Dekompresi

   Hasil *black box testing* pada proses dekompresi daapt disajikan pada tabel 4.6.

Tabel 4.6 Black Box Testing Dekompresi

No.	Kasus Uji	Langkah Uji	Hasil	Status
1.	*Input file text*	mengimport *file text* terkompresi dengan memilih tombol "*Browse File Text*"	sistem hanya dapat menerima *file text* terkompresi dengan format *.gc* dan sistem dapat menampilkan isi dan *size* dari *file text* terkompresi	Berhasil
2.	Dekompresi	melakukan dekompresi dengan memilih tombol memilih tombol "Dekompresi *File Text*"	sistem dapat menampilkan hasil dekompresi, *size*, dan waktu proses dekompresi	Berhasil

Tabel 4.6 pengujian *black box testing* proses dekompresi, kasus pengujian yang dilakukan antara lain "*input file text*" dan "dekompresi" dengan status berhasil.

5. *Black Box Testing* Dekripsi

   Adapun hasil *black box testing* pada proses dekripsi dengan menggunakan kunci privat *Rabin-p* yang disajikan pada tabel 4.7.

Tabel 4.7 Black Box Testing Dekripsi

No.	Kasus Uji	Langkah Uji	Hasil	Status
1.	*Input file* kunci	mengimport *file* kunci privat dengan memilih tombol "*Browse* Kunci Privat"	sistem hanya dapat menerima format *file .privkey* dan sistem dapat menampilkan isi dari *file* kunci privat *Rabin-p*	Berhasil
2.	Dekripsi	mendekripsi *file text* hasil dekompresi dengan memilih tombol "Dekripsi *File Text*"	sistem dapat menampilkan hasil dekripsi (*plaintext*), *size*, dan waktu proses dekripsi	Berhasil

Lanjutan Tabel 4.7 Black Box Testing Dekripsi

3.	Simpan	memilih tombol "Simpan *File Text*"	sistem dapat menyimpan hasil dekripsi (*plaintext*) dalam format *.txt* atau *.docx*.	Berhasil

Tabel 4.7 pengujian *black box testing* proses dekripsi, kasus pengujian yang dilakukan antara lain "*input file* kunci", "dekripsi" dan "simpan" dengan status hasil pengujian berhasil.

BAB 5
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil analisis, implementasi, dan pengujian sistem, maka kesimpulan yang dapat diperoleh dari penelitian sistem kriptokompresi menggunakan algoritma asimetris *Rabin-p* dan algoritma *lossless compression Goldbach Code*, yaitu sebagai berikut:

1. Sistem kriptokompresi dalam penelitian diterapkan dengan cara mengenkripsi data pada *file text* menggunakan kunci publik dari algoritma *Rabin-p* yang akan menghasilkan *file text* terenkripsi. Selanjutnya dikompresi menggunakan algoritma *lossless compression Goldbach Code* untuk memperkecil ukuran *file text* terenkripsi. Dengan demikian sistem kriptokompresi dapat meningkatkan keamanan data serta memperkecil ukurannya.
2. Data pada dokumen *file text* dari hasil sistem kriptokompresi mampu menjaga kerahasiaan sebuah informasi karena telah dienkripsi serta dapat menghemat ruang penyimpanan yang lebih efisien karena sudah dikompresi terlebih dahulu. *File text* yang telah dienkripsi dan dikompresi dapat dikembalikan lagi kedalam *file* aslinya melalui proses dekompresi dan dekripsi sesuai dengan *size* dan jumlah karater yang terdapat pada *file text* aslinya.

5.2 Saran

Adapun saran yang dapat penulis berikan untuk pengembangan selanjutnya dari penelitian ini adalah sebagai berikut:

1. Sistem kriptokompresi yang telah dibangun hanya mendukung pembacaan *string* saja, diharapkan untuk penelitian selajutnya dapat melakukan pembacaan terhadap audio dan video.
2. Sistem kriptokompresi dalam penelitian ini bekerja dengan cara mengenkripsi data lalu melakukan kompresi, maka untuk penelitian selajutnya dapat membandingkan dengan menggunakan kinerja sebaliknya agar dapat mengetahui proses mana yang lebih efisien.

DAFTAR PUSTAKA

Almurtada, I and Syahrizal, M., "Penerapan Algoritma Golbach Codes Pada Kompresi File Teks Terenkripsi Hill Cipher," *Jurnal Pelita Informatika*, vol. 6, no. 1, 2018.

Apriyanto, M and Hutrianto, "Analisa Penerapan Algoritma Golbach Codes dan Metode Shannon-Fano Pada Kompresi File Teks," *Bina Darma Conference on Computer Science*, vol. 2, no. 5, 2020.

Asbullah, M. A., Ariffin, M. R. K., Mahad, Z., "Analysis On The Rabin-p Cryptosystem," *Citation: AIP Conference Proceedings 1787*, 080012, 2016.

Asbullah, M. A., Ariffin, M. R. K., Mahad, Z., "Design of Rabin-Like Cryptosystem without Decryption Failure," *Malaysian Journal of Mathematical Sciences 10(S)* August: 1–18, 2016.

Budiman, M. A., Saputra, M. Y., Handrizal., "A Hybrid Cryptosystem Using Vigenère Cipher and Rabin-p Algorithm in Securing BMP Files," *Journal of Computing and Applied Informatics (JoCAI)*, vol. 4, no. 2, 2020.

Darwis, D., Prabowo, R., Hotimah, N., "Kombinasi Gifshuffle, Enkripsi AES dan Kompresi Data Huffman Untuk Meningkatkan Keamanan Data," *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK)*, vol. 5, no. 4, 2018.

Hasugian, B. S., "Penerapan Kriptografi Sebagai Keamanan Sistem Informasi Pada Usaha Kecil dan Menengah," Jurnal Warta Edisi : 53 Juli, 2017.

Hidayat, R., "Analisis Perbandingan Efisiensi Algoritma Kunci Publik Rabin-p dan Algoritma Kunci Publik RSA-CRT Pada Pengamanan Pesan," *Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Sriwijaya*, 2021.

Huda, I. A., "Perkembangan Teknologi Informasi dan Komunikasi (TIK) Terhadap Kulaitas Pembelajaran di Sekolah Dasar," *Jurnal Pendidikan dan Konseling Research & Learning in Primary Education (JPdK)*, vol. 2, no. 1, 2020.

Irliansyah, M. R., Nasution, S. D., Ulfa, K., "Penerapan Metode Deflate dan Algoritma Golbach Codes Dalam Kompresi File Teks," *KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer)*, vol. 1, no. 1, 2017.

Laswi, A. S., "Pemanfaatan Algoritma Rice Codes dan Golbach Codes Untuk Mengoptimalkan Media Penyimpanan Pada Memori Berupa File Text," *Jurnal Informatika Sains dan Teknologi (INSTEK)*, vol. 4, no. 2, 2019.

Laurentinus, Pradana, H. A., Sylfania, D. Y., Juniawan, F. P., "Perbandingan Kinerja RSA dan AES Terhadap Kompresi Pesan SMS Menggunakan Algoritme Huffman," *Jurnal Teknologi dan Sistem Komputer*, vol. 8, no. 3, 2020.

Mangiri, H. S., "Pembelajaran Kompresi Text Dengan Menggunakan Metode Shanon-Fano," *Joined Jurnal*, vol. 1, no. 1, 2018.

Meko, D. A., "Perbandingan Algoritma DES, AES, IDEA Dan Blowfish dalam Enkripsi dan Dekripsi Data," *Jurnal Teknologi Terpadu*, vol. 4, no. 1, 2018.

Nasution, S. D., Ginting, G. L., Syahrizal, M., Rahim, R., "Data Security Using Vigenere Cipher and Goldbach Codes Algorithm," *International Journal of Engineering Research & Technology (IJERT)*, vol. 6, issue. 01, 2017.

Pangesti, W. E., Widagdo, G., Riana, D., Hadianti, S., "Implementasi Kompresi Citra Digital Dengan Membandingkan Metode Lossy dan Lossless Compression Menggunakan Matlab," *Jurnal Khatulistiwa Informatika*, vol. 8, no. 1, 2020.

Pramadi, A. A., Nasution, S. D., Purba, B., "Penerapan Algoritma Even-Rodeh Pada Aplikasi Kompresi File Gambar," *KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer)*, vol. 3, no. 1, 2019.

Putri, G. G., Styorini, W., Rahayani, R. D., "Analisis Kriptografi Simetrvs AES dan Kriptografi RSA Pada Enkripsi Citra Digital," *Jurnal Penelitian dan Pengabdian (Sains & Teknologi)*, vol. 6, no. 2, 2018.

Santoso and Nurmalina, R., "Perencanaan dan Pengembangan Aplikasi Absensi Mahasiswa Menggunakan Smart Card Guna Pengembangan Kampus Cerdas (Studi Kasus Politeknik Negeri Tanah Laut)," *Jurnal Integrasi*, vol. 9, no. 1, 2017.

Saputro, T. H., Hidayati, N., Ujianto, E. I. H., "Survei Tentang Algoritma Kriptografi Asimetris," JIP (Jurnal Informatika Polinema), 2020.

Siregar, R., "Analvsis Perbandingan Kinerja Algoritma Start/Stop Code dan Algoritma Golbach G1 Code Pada Kompresi File Teks," *Repository Universitas Sumatera Utara*, 2019.

Siregar, A., Siregar, R., Handoko, D., Tommy., Elhanafi, A. M., “Analisis Perbandingan Kompresi Data Teks Dengan Menggunakan Algoritma Diferensiasi ASCII dan Half-Byte,” *SNASTIKOM Universitas Harapan Medan*, 2020.

Tanjung, A. S and Nasution, S. D., “Comparison Analysis with Huffman Algorithm and Goldbach Codes Algorithm in File Compression Text Using the Method Exponential Comparison,” *The IJICS (International Journal of Informatics and Computer Science)*, vol. 4, no. 1, 2020.

Yogie, M., “Penerapan Algoritma Golbach Codes Pada Kompresi File Gambar Terenkripsi Vigenee Cipher,” *Jurnal Pelita Informatika*, vol. 7, no. 1, 2018.