KRIPTANALISIS ALGORITMA KRIPTOGRAFI HIBRIDA BERBASIS MATRIKS MODIFIKASI HILL CIPHER-RSA

SKRIPSI

SALLY LIVIA KOSASIH

201401025

PROGRAM STUDI S-1 ILMU KOMPUTER

FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI

UNIVERSITAS SUMATERA UTARA

MEDAN

2024

KRIPTANALISIS ALGORITMA HIBRIDA BERBASIS MATRIKS MODIFIKASI HILL CIPHER DAN RSA

SKRIPSI

Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh Sarjana Ilmu Komputer

SALLY LIVIA KOSASIH

201401025

PROGRAM STUDI S-1 ILMU KOMPUTER

FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI

UNIVERSITAS SUMATERA UTARA

MEDAN

2024

PERSETUJUAN

Judul	:	KRIPTANALISIS ALGORITMA HIBRIDA BERBASIS MATRIKS MODIFIKASI HILL CIPHER – RSA
Kategori	:	SKRIPSI
Nama	:	SALLY LIVIA KOSASIH
Nomor Induk Mahasiswa	:	201401025
Program Studi	:	SARJANA (S1) ILMU KOMPUTER
Fakultas	:	ILMU KOMPUTER DAN TEKNOLOGI INFORMASI USU

Medan, 22 Desember 2023

Komisi Pembimbing:

Pembimbing II	Pembimbing I
Dian Rachmawati S.Si., M.Kom.	Dr. Mohammad Andri Budiman S.T., M.Com.Sc., M.E.M
NIP 198307232009122004	NIP 197510082008011011

Diketahui/disetujui oleh

Program Studi S1 Ilmu Komputer

Dr. Amalia S.T., M.T.

NIP 197812212014042001

PERNYATAAN

KRIPTANALISIS ALGORITMA HIBRIDA BERBASIS MATRIKS MODIFIKASI HILL CIPHER DAN RSA

SKRIPSI

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

Medan, 24 November 2023

Sally Livia Kosasih

201401025

PENGHARGAAN

Segala puji dan Syukur dipanjatkan kepada Tuhan yang Maha Esa ,penulis dapat berada di tahap penyusunan skripsi ini sebagai syarat untuk mendapatkan gelar Sarjana Komputer di Program Studi S-1 Ilmu Komputer, USU. Dengan penuh rasa hormat pada kesempatan ini penulis mengucapkan terima kasih kepada Ibu dan Ayah segala bentuk perjuangan, kasih sayang, dan perlindungan dengan doa yang dipanjatkan untuk penulis. Terima kasih untuk setiap dukungan yang telah diberikan sehingga penulis dapat berada di titik ini.

Penyusunan skripsi ini tidak terlepas dari bantuan, dukungan, dan bimbingan dari banyak pihak. Oleh karena itu, penulis mengucapkan banyak terima kasih kepada:

1. Bapak Prof. Dr. Muryanto Amin S.Sos., M.Si. selaku Rektor Universitas Sumatera Utara.
2. Bapak Dr. Mohammad Andri Budiman S.T., M.Comp.Sc., M.E.M. selaku Wakil Dekan I Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara dan Dosen Pembimbing I yang telah memberi banyak dukungan, motivasi, masukan, dan bimbingan spiritual yang membangun kepada penulis selama penyusunan skripsi ini.
3. Ibu Dr. Amalia, S.T., M.T. selaku Ketua Program Studi S-1 Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara serta dosen pembanding I yang telah memberikan saran dan kritik yang membangun kepada penulis terhadap penyusunan skripsi ini.
4. Ibu Sri Melvani Hardi S.Kom., M.Kom., selaku sekretaris Program Studi S-1 Ilmu Komputer, serta dosen pembanding II yang telah memberikan saran dan kritik yang membangun kepada penulis terhadap penyusunan skripsi ini.
5. Ibu Dian Rachmawati S.Si., M.Kom.. selaku Dosen Pembimbing Akademik yang telah memberi banyak dukungan dan motivasi kepada penulis serta dosen pembimbing II tugas akhir.
6. Seluruh bapak dan ibu dosen Fasilkom-TI USU, khususnya dosen Program Studi S-1 Ilmu Komputer yang telah mendidik dan memberi wawasan serta moral yang berharga, baik di bangku perkuliahan maupun setelah lulus.

7. Kakak serta sahabat yang senantiasa mendukung penulis dalam menyelesaikan tugas akhir.
8. Seluruh pegawai dan staf Fasilkom-TI USU yang telah memberi bantuan selama masa perkuliahan.
9. Teman-teman seperbimbingan ‘YOK BISA YOK’ yang senantiasa memberi semangat, saling mendoakan, mendukung dalam penulisan skripsi

Dan seluruh pihak yang telah memberi dukungan serta doa baik yang tidak dapat penulis sebutkan satu-persatu. Semoga Yang Maha Kuasa senantiasa melimpahkan keberkahan serta kebaikan atas semua dukungan yang telah diberikan kepada penulis.

Medan, 24 November 2023

Penulis,

Sally Livia Kosasih

ABSTRAK

Algoritma kriptografi simetris merupakan algoritma kriptografi yang memiliki kecepatan komputasi yang lebih tinggi namun memiliki kendala dalam pengolahan kunci simetris yang banyak. Sedangkan, algoritma kriptografi asimetris merupakan algoritma kriptografi yang memiliki tingkat keamanan yang lebih baik namun membutuhkan sumber daya komputasi yang lebih tinggi. Untuk mengatasi kelemahan dari algoritma kriptografi simetris dan asimetris, maka dibentuk algoritma kriptografi hibrida. Algoritma kriptografi hibrida merupakan gabungan algoritma simetris dan asimetris yang dibentuk untuk mengatasi kelemahan algoritma kriptografi simetris, misalkan algoritma Hill Cipher dan kriptografi asimetris, misalkan algoritma RSA (*Rivest-Shamir-Adleman*).

Salah satu contoh algoritma kriptografi hibrid adalah Algoritma Hasoun-Khlebus-Tayyeh. Walaupun algoritma kriptografi hibrida memiliki kemampuan dalam mengatasi kelemahan algoritma kriptografi simetris dan asimetris, perkembangan teknologi dan komputasi juga memberikan peluang bagi *hacker* dalam melakukan serangan terhadap kriptografi hibrida salah satu contohnya adalah algoritma modifikasi Hill Cipher-RSA. Untuk melakukan pengujian ketahanan algoritma modifikasi Hill Cipher-RSA, maka dilakukan pengujian dengan jenis serangan kriptografi pada umumnya yakni serangan *ciphertext-only* dan *brute-force*. Selain itu, pengujian juga dilakukan dengan menganalisa waktu yang diperlukan dalam proses enkripsi hingga kriptanalisis serta jumlah iterasi pada proses kriptanalisis. Hasil dari pengujian ini menunjukan bahwa proses keseluruhan sistem memiliki kompleksitas waktu θ dengan n merupakan nilai modulo yang digunakan pada algoritma Hill Cipher. Hal ini berarti semakin besar nilai n yang diuji, maka semakin lama sistem tersebut dalam menyelesaikan tahapan enkripsi dan uji coba kriptanalisis.

Kata Kunci: *Algoritma Kriptografi Hibrida, Kriptografi simetris, kriptografi asimetris, Hill Cipher, RSA, Brute-force, serangan ciphertext-only.*

CRYPTANALYSIS OF A HYBRID CRYPTHOGRAPHIC ALGORITHM BASED ON A MATRIX MODIFIED HILL CIPHER-RSA

ABSTRACT

A symmetric cryptographic algorithm is a cryptographic algorithm that has a higher computing speed but has problems in processing many symmetric keys. Meanwhile, an asymmetric cryptographic algorithm is a cryptographic algorithm that has a better level of security but requires higher computing resources. To overcome the weaknesses of symmetric and asymmetric cryptographic algorithms, a hybrid cryptographic algorithm was created. Hybrid cryptographic algorithms are a combination of symmetric and asymmetric algorithms which were formed to overcome the weaknesses of symmetric cryptographic algorithms, for example the Hill Cipher algorithm and asymmetric cryptography, for example the RSA (Rivest-Shamir-Adleman) algorithm.

One example of a hybrid cryptographic algorithm is the Hasoun-Khlebus-Tayyeh Algorithm. Although hybrid cryptographic algorithms have the ability to overcome the weaknesses of symmetric and asymmetric cryptographic algorithms, developments in technology and computing also provide opportunities for hackers to carry out attacks on hybrid cryptography, one example of which is the modified Hill Cipher-RSA algorithm. To test the resilience of the Hill Cipher-RSA modification algorithm, tests were carried out using general types of cryptographic attacks, namely ciphertext-only and brute-force attacks. Apart from that, testing is also carried out by analyzing the time required for the encryption process to cryptanalysis and the number of iterations in the cryptanalysis process. The results of this test show that the entire system process has a time complexity of $\theta(n^4)$ where n is the modulo value used in the Hill Cipher algorithm. This means that the greater the n value being tested, the longer it will take the system to complete the encryption and cryptanalysis testing stages.

Keywords: *Hybrid Cryptography Algorithm, Symmetric cryptography, asymmetric cryptography, Hill Cipher, RSA, Brute-force, ciphertext-only attack.*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I

PENDAHULUAN

1.1 Latar Belakang

Kriptografi merupakan suatu teknik dalam merahasiakan suatu pesan atau informasi, sehingga pesan atau informasi tersebut hanya dapat dipahami oleh pihak yang berwenang. Dalam merahasiakan suatu informasi atau pesan yang dikirim oleh pengguna (*sender*), konsep kriptografi akan mengubah pesan atau informasi tersebut menjadi suatu informasi yang acak dan tidak bermakna. Proses tersebut disebut dengan proses enkripsi atau *encipherment*. Pesan atau informasi yang diacak tersebut disebut dengan *ciphertext*. Penerima pesan atau informasi (*recipient*) yang berwenang akan mengubah *ciphertext* tersebut menjadi pesan atau informasi yang sebenarnya (*plaintext*). Proses tersebut disebut dengan proses dekripsi atau *decipherment*. Proses enkripsi-dekripsi dari suatu pesan atau informasi pada umumnya memerlukan suatu kunci yang dibentuk pada proses pembentukan algoritma kriptografi (Klima & Sigmon, 2019).

Pada umumnya, berdasarkan jenis kunci yang digunakan, kriptografi dapat dibedakan menjadi kriptografi simetris, kriptografi asimetris, serta kriptografi hibrida (Seputra & Saskara, 2020). Kriptografi simetris merupakan jenis kriptografi yang menggunakan satu jenis kunci dalam proses enkripsi dan dekripsi. Contoh dari algoritma kriptografi simetris adalah Hill Cipher, DES *(Data Encryption Standard*), *blowfish, twofish,* MARS, AES (*Advanced Encryption Standard*) (Arif & Nurokhman, 2023). Kriptografi asimetris merupakan jenis kriptografi yang menggunakan kunci yang berbeda dalam proses enkripsi dan dekripsi. Beberapa contoh dari algoritma kriptografi asimetris adalah RSA (Rivest-Shamir-Adleman), ECC (*Elliptic Curve Cryptography*) (Arif & Nurokhman, 2023).

Algoritma kriptografi simetris dan asimetris masing-masing memiliki kelebihan dan kelemahan yang berbeda. Algoritma kriptografi simetris memiliki

kecepatan komputasi yang lebih tinggi. Namun, algoritma kriptografi simetris memiliki kendala dalam pengolahan kunci simetris yang banyak. Algoritma kriptografi asimetris memiliki tingkat keamanan yang lebih baik. Namun, algoritma ini membutuhkan sumber daya komputasi yang lebih tinggi (Arif & Nurokhman, 2023)

Untuk mengatasi kelemahan masing-masing algoritma tersebut, salah satu solusi untuk mengatasi masalah kemampuan pengamanan data yang besar serta kebutuhan sumber daya komputasi yang tinggi adalah dengan menggunakan algoritma kriptografi hibrida. Algoritma kriptografi hibrida merupakan algoritma yang menggunakan dua jenis tingkatan kunci yakni kunci rahasia simetris serta kunci enkripsi asimetris yakni sepasang kunci publik dan kunci privat (Pangaribuan, 2018). Salah satu contoh penerapan algoritma hibrida adalah penggabungan algoritma kriptografi simetris Hill Cipher dengan RSA. Model modifikasi ini merubah konsep kunci rahasia Hill Cipher serta merubah metode enkripsi-dekripsi algoritma Hill Cipher dengan penambahan metode enkripsi-dekripsi RSA. Hill Cipher merupakan algoritma kunci simetris yang kuat terhadap serangan *brute force* dan serangan analisis frekuensi namun lemah terhadap serangan yang memanfaatkan persamaan linear. Sedangkan RSA merupakan kriptografi kunci asimetris yang memiliki pemfaktoran bilangan yang besar sehingga dapat meningkatkan keamanan pembentukan kunci (Jamaludin, 2018).

Walaupun algoritma kriptografi hibrida mampu mengatasi kelemahan yang dimiliki algoritma kriptografi simetris dan asimetris, namun seiring dengan perkembangan tingkat komputasi, potensi atau celah yang digunakan oleh *hacker* dalam melakukan penyerangan bagian kunci publik maupun kunci simetris juga semakin meningkat. Oleh sebab itu, penerapan konsep kriptanalisis dapat membantu memberikan identifikasi celah keamanan dari pemanfaatan suatu kriptografi sehingga suatu algoritma kriptografi dapat bertahan terhadap serangan yang semakin kompleks.

1.2 Rumusan Masalah

Sebagai salah algoritma hibrida, algoritma Hill Cipher-RSA merubah konsep pemanfaatan kunci rahasia yang digunakan pada tahapan enkripsi-dekripsi, menambah tahapan enkripsi-dekripsi dengan algoritma RSA, serta mengubah nilai modulo n pada Hill Cipher. Oleh sebab itu diperlukan pengujian algoritma modifikasi Hill Cipher-

RSA terhadap serangan *ciphertext-only* dan *brute-force* untuk mengevaluasi tingkat keamanan algoritma modifikasi Hill Cipher-RSA dibandingkan dengan algoritma Hill-Cipher klasik.

1.3 Batasan Masalah

Adapun batasan masalah yang akan dikaji dalam penelitian ini adalah sebagai berikut:

1. Ukuran matriks yang digunakan pada penerapan algoritma Kriptografi hibrida Hill Cipher-RSA adalah matriks dengan ukuran 2×2.
2. Pengujian jenis serangan terbatas pada serangan *ciphertext-only* dengan metode *brute force.*
3. Metode *brute-force* diterapkan dengan membangkitkan kunci simetris secara acak untuk diuji hasil dekripsi.

1.4 Tujuan Penelitian

Adapun tujuan dari penelitian yang dilakukan pada skripsi ini adalah menguji kemampuan algoritma kriptografi hibrida Hill Cipher – RSA dalam menghadapi serangan *ciphertext-only* dengan metode *brute force* dengan memanfaatkan ukuran kunci dari rentang 0 - 1024 serta ukuran *plaintext* 5-25 karakter berdasarkan waktu komputasi yang diperlukan serta jumlah percobaan *brute force* yang dihasilkan untuk menyelesaikan proses kriptanalisis.

1.5 Manfaat Penelitian

Adapun manfaat dari penelitian ini adalah untuk mengetahui kelebihan dan kelemahan algoritma kriptografi hibrida Hill Cipher-RSA melalui proses kriptanalisis.

1.6 Metodologi Penelitian

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

1. Studi Pustaka

   Tahapan awal yang akan dilaksanakan pada penelitian ini adalah mengumpulkan dan mempelajari berbagai informasi mengenai

karakteristik algoritma kriptografi berbasis matriks, algoritma kunci publik, algoritma kriptografi hibrida, dan metode kriptanalisis algoritma kriptografi berbasis matriks serta metode kriptanalisis algoritma kriptografi kunci publik berdasarkan berbagai sumber literatur seperti buku, jurnal, artikel, dan tinjauan pustaka lainnya.

2. Analisis dan Perancangan Sistem

   Pada tahapan analisis dan perancangan sistem dilakukan analisis model algoritma kriptografi hibrida Hill Cipher – RSA, mengidentifikasi dan memodelkan permasalahan dengan memanfaatkan diagram alir (*flowchart*). Selanjutnya dilakukan perancangan fungsionalitas sistem yang meliputi kriptanalisis dan prosedur pendukung lainnya.

3. Implementasi Sistem

   Pada tahapan implementasi sistem, dilakukan penerapan hasil analisis dan perancangan sistem melalui coding sistem

4. Pengujian Sistem

   Pada tahapan pengujian sistem dilakukan pengujian terhadap kinerja dan kemampuan sistem dalam mencapai tujuan. Tujuan pengujian ini adalah untuk mengidentifikasi kesalahan dan kekurangan sistem serta efisiensi kerja dari sistem dalam melakukan enkripsi dan kriptanalisis. Tahapan pengujian sistem dilakukan dengan memasukan inputan sesuai dengan fungsi yang dimiliki sistem dan menganalisis hasil output yang diberikan.

5. Dokumentasi

   Pada tahapan dokumentasi, dilaksanakan penyusunan skripsi melalui hasil analisis, perancangan, dan pengujian sistem sesuai dengan format yang berlaku.

1.7 Sistematika Penulisan

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

BAB I PENDAHULUAN

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

BAB II LANDASAN TEORI

Bab ini memuat kajian pustaka yang berkaitan dengan konsep Algoritma RSA, Hill Cipher, kriptografi, serangan kriptografi, teori matriks serta kajian pustaka mengenai metode faktorisasi bilangan prima.

BAB III ANALISIS DAN PERANCANGAN

Bab ini memuat tahapan analisis perancangan arsitektur sistem dan analisis keamanan Hill Cipher-RSA. Selanjutnya dilakukan perancangan sistem dan metode penyerangan kriptografi dengan memanfaatkan algoritma kriptanalisis yang telah ditetapkan.

BAB IV IMPLEMENTASI DAN PENGUJIAN SISTEM

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

BAB V KESIMPULAN DAN SARAN

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

BAB II

TINJAUAN PUSTAKA

2.1 Kriptografi

Kata Kriptografi (*cryptography*) berasal dari gabungan dua kata dalam bahasa Yunani yakni kata *kriptos* yang berarti rahasia dan kata *graphein* yang berarti tulisan. Pada masa lampau, kriptografi merupakan suatu teknik yang digunakan untuk mengamankan komunikasi informasi yang bersifat penting seperti komunikasi militer, mata-mata, diplomat, dan lain sebagainya. Namun seiring perkembangannya kriptografi tidak hanya terbatas pada pengamanan informasi. Pada saat ini, kriptografi modern juga meliputi kegiatan otentikasi data, memastikan integritas data, dan memastikan ketiadaan penyangkalan terhadap keterlibatan dalam proses komunikasi (Harahap, et al., 2021). Hal tersebut mengakibatkan adanya pengembangan makna kriptografi modern yakni sebagai suatu ilmu dan seni untuk mengamankan suatu pesan atau informasi dengan menerapkan ilmu Matematika dengan memperhatikan tingkat kerahasiaan, integritas data, autentikasi, dan ketiadaan penyangkalan (Yulianti & Wijaya, 2022).

Munculnya kriptografi modern disebabkan oleh penggunaan teknologi sehingga kriptografi klasik disesuaikan dengan pemanfaatan teknologi informasi untuk memaksimalkan pemanfaatannya (Murdowo, 2019). Namun, secara konseptual kriptografi klasik dan kriptografi modern memiliki konsep yang serupa yakni memanfaatkan suatu kunci dalam melakukan proses enkripsi dan dekripsi pesan atau informasi. Pada kriptografi klasik, kunci yang digunakan pada proses enkripsi dan dekripsi adalah sama. Sedangkan, pada algoritma kriptografi modern, kunci yang digunakan pada proses enkripsi dengan kunci yang digunakan dalam proses dekripsi adalah berbeda. Konsep algoritma tersebut dikenal dengan Algoritma Asimetris atau Algoritma Kunci Publik (Giri, et al., 2021).

2.2 Kriptografi Kunci Simetris

Kriptografi kunci simetris adalah kriptografi yang menggunakan satu kunci untuk melakukan proses enkripsi dan dekripsi. Kriptografi kunci simetris sering dikenal dengan kriptografi klasik (Putri & Rahayani, 2018). Kriptografi simetris banyak digunakan karena memiliki daya kerja yang cepat dan daya komputasi yang lebih kecil. Contoh algoritma kriptografi yang bersifat kriptografi kunci simetris yang telah ada saat ini adalah *Advanced Encryption Standard* (AES), Blowfish, *Triple Data Encryption Standard* (DES), dan *Rivest Code* (RC4) (Seputra & Saskara, 2020).

2.3 Kriptografi Kunci Asimetris

Kriptografi kunci asimetris atau yang sering disebut algoritma kunci publik merupakan algoritma kriptografi yang menggunakan dua kunci yang berbeda untuk melakukan proses enkripsi dan dekripsi (Seputra & Saskara, 2020). Pada umumnya, algoritma kriptografi kunci asimetris memanfaatkan 2 jenis kunci yakni kunci umum (*public key*) yakni kunci yang dapat diketahui oleh semua orang dan kunci privat (*private key*) yakni kunci yang bersifat rahasia dan hanya dapat diketahui oleh pihak yang berwenang (Putri & Rahayani, 2018).

2.4 Kriptanalisis

Kriptanalisis adalah suatu ilmu untuk mengungkapkan kunci yang digunakan pada algoritma kriptografi dengan *ciphertext* yang tersedia untuk memperoleh pesan atau informasi sebenarnya (*plaintext*) (Solin & Ramadhani, 2020). Pelaku kriptanalisis disebut dengan kriptanalis. Kriptanalisis bertujuan untuk menguji kemampuan suatu algoritma dan mengembangkan tingkat keamanan dari suatu algoritma jika diterapkan dengan strategi yang benar dan sesuai. (Chen, et al., 2020).

2.5 Serangan Kriptogtafi

Serangan kriptografi merupakan bentuk percobaan untuk memperoleh *plaintext* dari suatu *ciphertext* yang dimiliki atau menemukan kunci yang dapat mengubah *plaintext* dari *ciphertext*. Berdasarkan keterlibatan penyerang, jenis-jenis serangan kriptografi dapat dibedakan menjadi:

1. Serangan Pasif

Serangan pasif merupakan serangan yang tidak melibatkan seseorang secara langsung dalam komunikasi informasi antara penerima dan pengirim yang sah namun pada lalu lintas informasi untuk mengumpulkan data informasi yang dapat digunakan dalam proses melakukan kriptanalisis. Pada umumnya, bentuk serangan ini merupakan penyadapan terhadap informasi ketika proses komunikasi informasi, misalkan *Snooping* dan *Traffic analysis*.

Snooping merupakan kegiatan pencurian informasi data melalui hak akses ilegal terhadap jaringan lalu lintas komunikasi informasi yang seharusnya. *Traffic analysis* merupakan jenis serangan yang dilakukan dengan menganalisis lalu lintas komunikasi data untuk memperoleh data-data informasi (Thahara & Siregar, 2021).

Selain *Snooping* dan *Traffic Analysis*, terdapat beberapa jenis serangan pasif, yakni:

a. *Wiretapping* merupakan serangan terhadap informasi dengan melakukan pencegatan informasi pada saluran lalu lintas komunikasi dengan memanfaatkan perangkat keras.
b. *Electromagnetic eavesdropping* merupakan serangan dengan memanfaatkan media nirkabel atau *wireless* seperti radio.
c. *Acoustic Eavesdropping* merupakan serangan dengan memanfaatkan gelombang yang diperoleh dari suara manusia (Purba, et al., 2020).

2. Serangan Aktif

Serangan aktif merupakan bentuk serangan yang dapat mengubah informasi yang dikirim serta mempengaruhi sistem dan aliran komunikasi informasi. Beberapa contoh bentuk serangan aktif adalah sebagai berikut:

a. *Masquerade* merupakan jenis serangan yang dilakukan dengan cara mengambil alih peranan pengirim atau penerima.
b. *Modification* merupakan jenis serangan yang mengambil alih jalur komunikasi sehingga penyerang dapat menginterupsi proses pengiriman informasi.

c. *Denial of Service* merupakan jenis serangan yang memiliki tujuan untuk merusak sistem sehingga sistem tidak dapat bekerja sebagaimana mestinya (Thahara & Siregar, 2021).

d. *Man-in-middle attack* adalah jenis penyerangan dimana penyerang menempatkan posisi diantara dua pihak yang sedang berkomunikasi dengan tujuan untuk memantau, memanipulasi, atau mencuri informasi yang dikirim (Purba, et al., 2020).

Berdasarkan teknik yang digunakan dalam penyerangan kriptografi, jenis-jenis serangan dapat dibedakan menjadi:

a. *Brute Force* atau *Exhaustive attack* merupakan jenis serangan kriptografi yang memanfaatkan pendekatan langsung untuk memecahkan suatu masalah. Konsep ini pada umumnya memanfaatkan semua kemungkinan kunci untuk mengungkap keseluruhan *plaintext*.

b. *Analytical Attack* merupakan jenis serangan dengan memanfaatkan cara menganalisis kelemahan algoritma kriptografi untuk mengurangi kemungkinan kunci-kunci yang tidak digunakan. Untuk menghadapi serangan ini, diperlukan suatu algoritma kriptografi yang kompleks.

Berdasarkan kuantitas informasi yang diperoleh kriptanalis, jenis-jenis serangan kriptografi dapat dibedakan menjadi:

a. *Ciphertext-only attack* merupakan jenis kriptanalisis dimana seorang kriptanalis memiliki *ciphertext* dari pesan yang dienkripsi dengan suatu algoritma yang diketahui untuk menemukan kunci yang akan digunakan untuk proses dekripsi. Metode ini akan memanfaatkan teknik untuk mengubah *plaintext* menjadi *ciphertext* atau teknik analisis frekuensi untuk mencari kunci yang sesuai.

b. *Known-plaintext* attack merupakan metode kriptanalisis dengan seorang kriptanalis mempunyai *ciphertext* dan *plaintext* pesan.

c. *Chosen-plaintext attack* merupakan metode penyerangan dengan seorang kriptanalis memiliki akses terhadap *ciphertext* dan

plaintext dari beberapa pesan dan dapat memilih beberapa penggalan dari *plaintext* yang dimiliki.

d. *Adaptive-Chosen-plaintext attack* merupakan metode penyerangan dengan seorang kriptanalis dapat memilih satu bagian kecil dari blok *plaintext* yang dapat digunakan untuk menemukan bagian *plaintext* yang lain. Kriptanalis akan mengulangi langkah-langkah tersebut hingga kriptanalis dapat menemukan seluruh informasi.

e. *Chosen-ciphertext attack* merupakan metode penyerangan dengan seorang kriptanalis yang mampu memilih informasi yang telah dienkripsi serta mengetahui plaintext hasil dekripsi yang sebenarnya.

f. *Chosen-key attack* merupakan metode kriptanalisis dengan seorang kriptanalis memiliki informasi mengenai kunci yang digunakan untuk mendekripsi pesan serta mengetahui hubungan antar kunci (Purba, et al., 2020).

2.6 Matriks

Matriks merupakan kumpulan angka-angka yang disusun dalam bentuk persegi atau persegi panjang yang pada umumnya dikelompokan dalam tanda kurung siku. Definisi ukuran matriks merupakan jumlah baris dan jumlah kolom yang dimiliki pada suatu matriks. Setiap anggota atau angka pada matriks dilambangkan dengan huruf kecil variabel yang melambangkan suatu matriks kemudian diikuti dengan posisi baris dan kolom yang ditempati bilangan tersebut (Klima & Sigmon, 2019).

2.6.1 Penjumlahan dan Pengurangan Matriks

Dua buah matriks dapat dijumlahkan atau dikurangkan jika dan hanya jika ukuran kedua matriks tersebut adalah sama (Klima & Sigmon, 2019). Misalnya terdapat dua buah Matriks A dan B yang memiliki nilai,

$$A = \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix}, B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

Maka, operasi matriks $A + B$ dapat dilakukan karena ukuran matriks A sama dengan matriks B. Operasi matriks $A + B$ akan menghasilkan:

$$A + B = \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix} + \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

$$= \begin{bmatrix} 7 & 9 \\ 11 & 13 \end{bmatrix}$$

2.6.2 Perkalian Matriks

Dua buah (Matriks A dan Matriks B) matriks dapat dikalikan jika jumlah baris Matriks A sama dengan jumlah kolom Matriks B serta jumlah kolom Matriks A sama dengan jumlah baris Matriks B (Klima & Sigmon, 2019). Misalkan terdapat dua buah matriks A dan B yang memiliki nilai:

$$A = \begin{bmatrix} 2 & 3 \\ 4 & 5 \\ 6 & 7 \end{bmatrix}, B = \begin{bmatrix} 2 & 3 & 4 \\ 5 & 6 & 7 \end{bmatrix}$$

Maka, operasi $A \cdot B$ dapat dilakukan karena jumlah baris matriks A sama dengan jumlah kolom matriks B serta jumlah kolom matriks A sama dengan jumlah baris matriks B. Hasil operasi perkalian matriks $A \cdot B$ akan menghasilkan

$$A \cdot B = \begin{bmatrix} 2 & 3 \\ 4 & 5 \\ 6 & 7 \end{bmatrix} \cdot \begin{bmatrix} 2 & 3 & 4 \\ 5 & 6 & 7 \end{bmatrix}$$

$$= \begin{bmatrix} 2 \cdot 2 + 3 \cdot 5 & 2 \cdot 3 + 3 \cdot 6 & 2 \cdot 4 + 3 \cdot 7 \\ 4 \cdot 2 + 5 \cdot 5 & 4 \cdot 3 + 5 \cdot 6 & 4 \cdot 4 + 5 \cdot 7 \\ 6 \cdot 2 + 7 \cdot 5 & 6 \cdot 3 + 7 \cdot 6 & 6 \cdot 4 + 7 \cdot 7 \end{bmatrix}$$

$$= \begin{vmatrix} 19 & 24 & 29 \\ 33 & 42 & 51 \\ 47 & 60 & 73 \end{vmatrix}$$

2.6.3 Matriks Invers dan Matriks Identitas

Matriks identitas merupakan matriks persegi yang memiliki elemen yang bernilai 1 (satu) secara diagonal dari ujung kiri atas hingga ujung kanan bawah sedangkan elemen lainnya bernilai 0 (nol). Matriks tersebut dikenal sebagai matriks identitas karena matriks tersebut berfungsi sebagai elemen identitas bagi suatu matriks. Suatu matriks persegi A dengan ukuran tertentu jika dilakukan operasi perkalian dengan suatu matriks identitas I maka akan memenuhi persamaan berikut: (Klima & Sigmon, 2019)

$$A \cdot I = I \cdot A = A$$

Suatu matriks A, memiliki suatu invers matriks B jika hasil perkalian matriks A dan B menghasilkan matriks Identitas I. Adapun terdapat beberapa karakteristik invers dari suatu matriks adalah sebagai berikut:

1. Suatu matriks B dapat dibuktikan sebagai invers dari matriks A jika memenuhi syarat $AB = I$, dengan I adalah matriks Identitas
2. Suatu invers dari suatu matriks bersifat unik untuk matriks tertentu, artinya jika matriks B merupakan invers matriks dari matriks A, maka B merupakan satu-satunya matriks yang akan memenuhi persamaan:
$$A \cdot B = I$$
3. Oleh karena suatu invers dari suatu matriks bersifat unik untuk matriks tersebut maka untuk menyatakan suatu matriks B merupakan invers dari matriks A dapat dinyatakan dengan:
$$B = A^{-1}$$
4. Suatu matriks dengan invers dari matriks tersebut berpasangan, artinya jika matriks A^{-1} merupakan invers dari matriks A, maka matriks A merupakan invers dari matriks A^{-1} (Klima & Sigmon, 2019).

2.6.4 Matriks dengan Operasi Modulo

Operasi-operasi pada matriks dapat digabungkan dengan aritmatika modulo dengan melakukan operasi tersebut secara umum kemudian melakukan operasi modulo pada setiap himpunan anggota matriks tersebut. Misalkan terdapat matriks A dan matriks B yang memiliki nilai sebagai berikut:

$$A = \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix}, \qquad B = \begin{bmatrix} 6 & 7 \\ 8 & 9 \end{bmatrix}$$

Untuk memperoleh nilai $(A + B)\ mod\ 3$ maka dapat dihitung dengan metode sebagai berikut:

$$\begin{aligned} (A + B)\ mod\ 3 &= \left(\begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix} + \begin{bmatrix} 6 & 7 \\ 8 & 9 \end{bmatrix} \right) mod\ 3 \\ &= \begin{bmatrix} 8 & 10 \\ 12 & 14 \end{bmatrix} mod\ 3 \\ &= \begin{bmatrix} 8\ mod\ 3 & 10\ mod\ 3 \\ 12\ mod\ 3 & 14\ mod\ 3 \end{bmatrix} \\ &= \begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix} \end{aligned}$$

Konsep invers matriks juga dapat diterapkan pada matriks modulo. Matriks B dapat dinyatakan sebagai invers dari matriks A pada modulo m, ($B = A^{-1}\ mod\ m$) jika $A \cdot B\ mod\ m = I$ (Klima & Sigmon, 2019).

2.7 Bilangan Prima

Bilangan prima dapat didefinisikan sebagai bilangan bulat positif lebih besar dari atau sama dengan 2 (dua), yang tidak memiliki bilangan pembagi positif selain bilangan 1 (satu) dan bilangan tersebut (Plymen, 2020). Pada umumnya, bilangan prima tersebar secara tidak merata dan acak pada suatu rentang bilangan bulat yang sangat besar. Hal ini mengakibatkan kesulitan dalam mencari bilangan-bilangan prima dalam rentang bilangan yang sangat besar (Wang, 2021). Permasalahan tersebut dapat dijadikan sebagai karakteristik unik dari bilangan prima. Pada umumnya, bilangan prima banyak dimanfaatkan dalam bidang kriptografi. Hal tersebut disebabkan oleh pencarian nilai faktor prima dari suatu bilangan bulat yang eksponensial hingga saat ini masih membutuhkan waktu yang relatif lama (Puspita, et al., 2015).

2.7.1. Faktorisasi Fermat

Faktorisasi Fermat merupakan salah metode untuk memfaktorisasi dua buah bilangan prima pembentuk suatu bilangan bulat eksponensial. Adapun tahapan algoritma Faktorisasi Fermat dapat diuraikan sebagai berikut: (Bahiq, et al., 2020).

1. Misalkan n adalah sebuah bilangan bulat komposit yang akan difaktorisasi menjadi p_1 dan p_2
2. Hitung nilai $q_1 = \lfloor \sqrt{n} \rfloor$ dan $q_2 = q_1^2 - n$.
3. Ulangi Langkah (2) selama q_2 bukanlah merupakan akar kuadrat sempurna dengan meningkatkan nilai q_1 menjadi $q_1 = q_1 + 1$.
4. Jika q_2 merupakan bilangan akar kuadrat sempurna maka definisikan nilai p_1 dan p_2 dengan:

$$p_1 = q_1 + \sqrt{q_2}$$
$$p_2 = q_1 - \sqrt{q_2}$$

2.8 Fungsi Totient Euler (*Euler's phi function*)

Fungsi Totient Euler merupakan suatu fungsi aritmatika yang digunakan untuk menghitung banyaknya bilangan bulat positif dari 1 (satu) hingga n yang bersifat relatif prima dengan bilangan n. Fungsi Totient Euler pertama kali ditemukan oleh Leonhard Euler pada tahun 1763 (Sonea & Cristea, 2023). Terdapat beberapa karakteristik untuk menentukan nilai Totient Euler yakni: (Rangasamy, 2019)

1. Jika p merupakan nilai prima dan faktor dari bilangan n maka nilai dari $\phi(p \cdot n)$ dapat ditentukan dengan perhitungan berikut:
$$\phi(p \cdot n) = p \cdot \phi(n)$$
2. Jika p merupakan nilai prima dan bukan merupakan faktor dari bilangan n, maka nilai dari $\phi(p \cdot n)$ dapat ditentukan dengan perhitungan berikut:
$$\phi(p \cdot n) = (p-1) \cdot \phi(n)$$
3. Jika n merupakan bilangan bulat positif serta n merupakan bilangan komposit, maka nilai dari $\phi(n!)$ dapat ditentukan dengan perhitungan berikut:
$$\phi(n!) = n \cdot \phi((n-1)!)$$
4. Jika n merupakan bilangan bulat positif serta n merupakan bilangan prima, maka nilai dari $\phi(n!)$ dapat ditentukan dengan perhitungan berikut:
$$\phi(n!) = (n-1) \cdot \phi((n-1)!)$$
5. Jika n merupakan bilangan bulat positif dan bilangan a merupakan bilangan eksponensial terhadap n, maka nilai dari $\phi(n^a)$ dapat ditentukan dengan perhitungan berikut:
$$\phi(n^a) = n^{a-1}\,\phi(n)$$

2.9 Algoritma Euklides (*Euclidean Algorithm*) untuk menentukan Faktor Pembagi Terbesar (FPB) atau *Greatest Common Divisor* (GCD) dari dua bilangan

Algoritma Euklides (*Euclidean Algorithm*) dalam menentukan Faktor Pembagi Terbesar (FPB) dari dua bilangan bulat memanfaatkan nilai sisa bagi (*reminder*) (Harahap, 2016). Adapun cara kerja dari algoritma ini dapat diuraikan sebagai berikut:

1. Misalkan, bilangan bulat i dan j, dengan $i > j$ merupakan bilangan yang akan ditentukan nilai Faktor Pembagi Terbesar dari kedua bilangan tersebut (FPB).
2. Hitung k yang merupakan sisa hasil pembagian i terhadap j.
3. Ubah nilai i menjadi j dan nilai j menjadi k, kemudian ulangi tahapan (2) dan (3) hingga k mencapai nilai nol.

4. Jika k mencapai nilai nol, maka i merupakan nilai dari Faktor Pembagi Terbesar (FPB) dari bilangan i dan j (Lestari, 2017).

Bentuk penulisan *pseudocode* dari algoritma Euclidean-GCD dapat dinotasikan sebagai berikut:

```
Fungsi Euclidean-GCD (i, j)
int k;
k = i mod j;
if (k = 0) {
   return j
}
return Euclidean-GCD (j, k)
```

Gambar 1 *Psuedocode* Algoritma Euclidean-GCD

Kompleksitas waktu merupakan pengukuran terhadap jumlah tahapan yang diperlukan untuk menjalankan suatu algoritma (Lestari, 2017). Berdasarkan *pseudocode* yang dinyatakan pada Gambar 1, kompleksitas algoritma Euclidean-GCD dipengaruhi oleh nilai i dan j serta kompleksitas terburuk dari algoritma ini terjadi ketika i dan j merupakan bilangan Fibonacci yang berurutan sehingga kompleksitas algoritma Euclidean-GCD adalah $O(\log(N))$ atau Kompleksitas waktu logaritmik (Saputra, 2011).

2.10 Hill Cipher

Hill Cipher merupakan jenis algoritma kriptografi kunci publik yang pertama kali ditemukan oleh Lester S. Hill pada tahun 1929 pada artikelnya yang berjudul *Cryptology in an Algebraic Alphabet* (Klima & Sigmon, 2019). Hill Cipher termasuk jenis kriptografi polialfabetik. Kriptografi polialfabetik merupakan jenis kriptografi yang memanfaatkan metode penyandian beberapa huruf substitusi. Adapun beberapa tahapan algoritma Hill Cipher adalah sebagai berikut:

1. Tentukan nilai batasan bilangan bulat yang akan digunakan, misalkan nilai tersebut dilambangkan dengan n.
2. Bentuk suatu matriks enkripsi A yang bersifat invertible
3. Ubah *plaintext* menjadi bentuk matriks, misalkan matriks B.
4. Uraikan matriks B menjadi beberapa bagian yang ukurannya kolomnya menyesuaikan ukuran baris matriks A
5. Lakukan proses enkripsi untuk memperoleh *ciphertext* dengan melakukan perhitungan:

$$C_i = B_i \cdot A \ mod \ n$$

6. Proses dekripsi pada algoritma Hill Cipher dapat dilakukan dengan perhitungan sebagai berikut:

$$B_i = C_i \cdot A^{-1} \ mod \ n$$

2.11 RSA (Rivest-Shamir-Adleman)

Algoritma kunci publik RSA (Rivest-Shamir-Adleman) merupakan salah satu algoritma kunci public yang dikembangkan oleh 3 peneliti dari MIT (Massachusetts Institute of Technology) yakni Ron Rivest, Adi Shamir, dan Leonard Adleman pada tahun 1997 (Simarmata, et al., 2019). Seperti algoritma kriptografi lainnya, algoritma kunci publik RSA juga memanfaatkan operasi aritmatika modular dengan operasi perpangkatan eksponensial serta bilangan prima sebagai salah satu kekuatan dalam pengamanan informasi (Klima & Sigmon, 2019).

Adapun tahapan yang dilakukan dalam menggunakan algoritma kunci public RSA adalah sebagai berikut:

a. Pembentukan kunci (*Key Generation*)
    1. Pilih dua buah bilangan prima yang berbeda, misalkan p dan q
    2. Hitung nilai $n = p \times q$
    3. Hitung nilai $m = (p-1) \times (q-1)$. Setelah menghitung nilai m, maka p dan q akan dihapus untuk mencegah diketahui oleh pihak lain.
    4. Bangkitkan bilangan bulat e yang relatif prima terhadap m.
    5. Bangkitkan suatu bilangan bulat d yang merupakan invers modulo e dalam modulo m, yakni $e \equiv d^{-1}(mod\ m)$.

Keseluruhan tahapan diatas dibangkitkan oleh penerima pesan (*recipient*). Kunci yang dikirim melalui jalur tidak aman atau kunci public terdiri dari N dan e. Sedangkan kunci yang dijaga kerahasiaannya adalah kunci m dan d (Simarmata, et al., 2019).

b. Enkripsi

Setelah pengirim pesan (*sender*) memperoleh kunci publik dari penerima pesan melakukan tahapan enkripsi yakni:

1. Ubah pesan yang berbentuk teks ke bentuk kumpulan blok (m_i) numerik yang nilainya berada pada rentang $[0, n-1]$.
2. Lakukan enkripsi dengan persamaan $C_i = m_i^e \ mod \ n$, dimana C_i adalah blok *ciphertext* dan P_i adalah blok *plaintext*.
3. Hasil blok *ciphertext* kemudian dikirim melalui jalur tidak aman ke penerima pesan (*recipient*) (Simarmata, et al., 2019)

c. Dekripsi

Setelah penerima pesan (*recipient*) memperoleh *ciphertext*, maka penerima pesan akan melakukan dekripsi dengan persamaan $m_i = C_i^d \ mod \ n$ (Simarmata, et al., 2019).

2.12 Algoritma kriptografi Hibrida Hasoun-Khlebus-Tayyeh

Algoritma Kriptografi Hibrida Hasoun-Khlebus-Tayyeh merupakan algoritma kriptografi hibrida yang menggabungkan algoritma kriptografi simetris, yakni Hill Cipher dan algoritma kriptografi asimetris, yakno RSA dalam satu konsep algoritma. Konsep tersebut memberikan peningkatan kemampuan pada algoritma Hill Cipher dalam ketahanan terhadap serangan kriptografi *known*-plaintext serta meningkatkan keamanan. Hal ini disebabkan karena algoritma ini bergantung pada kerahasiaan kunci K yang merupakan matriks invertibel, nilai modulo n yang besar, serta faktorisasi prima dari bilangan bulat N yang sangat besar. (Hasoun, et al., 2021) Adapun tahapan penggunaan algoritma hibrida Hill Cipher dan RSA yang diusulkan adalah sebagai berikut:

a. Pembangkitan Kunci (*Key Generation*)

1. Bangkitkan sebuah matriks involutori K. Matriks Involutori adalah matriks yang memiliki nilai invers yang menyerupai nilai matriks awal atau yang memiliki persamaan $K = K^{-1}$. Dalam matriks ukuran $2 \ x \ 2$, setiap elemen $a, b, c, d \in K$ memiliki karakteristik sebagai berikut:
   - $a^2 + bc = 1$
   - $a = -d$
2. Pilih dua buah bilangan prima acak yang berbeda p dan q.

3. Hitung nilai $N = p * q$, N akan digunakan sebagai bilangan modulo untuk kunci publik dan kunci privat.
4. Hitung nilai Totient Euler (*Euler phi function*) $\phi(N) = (p-1)(q-1)$.
5. Pilih sebuah bilangan bulat e dimana e berada pada rentang $1 < e < \phi(N)$.
6. Bangkitkan nilai d yang merupakan nilai invers modulo dari e pada modulo $\phi(N)$.
7. Publikasi kunci (e, N) sebagai kunci publik, dan d sebagai kunci privat.

b. Enkripsi

1. Misalkan terdapat blok *plaintext* $P = \begin{pmatrix} P_1 \\ \vdots \\ P_n \end{pmatrix}$ dengan $P_i \in \{0, 1, 2, \dots, 256\}$.
2. Lakukan enkripsi blok dengan metode Hill Cipher dengan persamaan:

$$C_H = \begin{pmatrix} C_{h_1} \\ \vdots \\ C_{H_n} \end{pmatrix} = K \times P \ (mod\ 256)$$

3. Lakukan enkripsi blok dengan metode RSA dengan persamaan:

$$C_R = \begin{pmatrix} C_{R_1} \\ \vdots \\ C_{R_n} \end{pmatrix} = (C_H)^e \ (mod\ N)$$

c. Dekripsi

1. Dengan memanfaatkan kunci privat d, lakukan perhitungan dekripsi dengan algoritma RSA dengan persamaan:

$$D(C_R) = (C_R)^d \equiv \begin{pmatrix} C_{R_1} \\ \vdots \\ C_{R_n} \end{pmatrix}^d (mod\ N) = C_H$$

2. Dengan memanfaatkan kunci rahasia matriks K, dimana K adalah matriks involutori sehingga $K = K^{-1}$ dilakukan dekripsi dengan algoritma Hill Cipher dengan persamaan berikut:

$$D(C_H) = P \equiv C_H \times K \ (mod\ 256).$$

(Hasoun, et al., 2021).

2.13 Penelitian yang Relevan

1. Menurut penelitian "*Kriptografi hybrida algoritma Hill Cipher dan Rivest Shamir Adleman Sebagai Pengembanagan kriptografi kunci simetris (Studi Kasus: Nilai Mahasiswa AMIK MBP"* oleh (Pangaribuan, 2018), penggunaan kriptografi kunci Hill Cipher dan RSA mampu meningkatkan jumlah kemungkinan untuk menebak kunci Hill setelah dimodifikasi dengan algoritma RSA namun juga meningkatkan waktu dekripsi.
2. Menurut penelitian "*Rancang Bangun Kombinasi Hill Cipher dan RSA Menggunakan Metode Hybrid Cryptosystem"* oleh (Jamaludin, 2018), penerapan kombinasi algoritma Hill Cipher dan RSA memungkinkan peningkatan keamanan enkripsi dan dekripsi dari model Hill Cipher serta peningkatan pengamanan kunci pada algoritma kriptografi simetris dan asimetris.

BAB III

ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

Tahapan analisis sistem merupakan salah satu tahapan yang bertujuan untuk mengidentifikasi kebutuhan yang dapat dimanfaatkan agar sistem dapat bekerja secara optimal sesuai dengan yang diharapkan. Adapun terdapat beberapa jenis analisis sistem yang dapat dilakukan adalah analisis kebutuhan sistem yang terdiri dari kebutuhan fungsional dan non-fungsional, serta analisis masalah.

3.1.1 Analisis Masalah

Analisis masalah merupakan tahapan yang dilakukan untuk mengidentifikasi permasalahan yang dibahas dan dikaji pada penelitian sehingga penelitian ini. Fokus permasalahan yang akan dikaji pada penelitian ini adalah pengujian kemampuan algoritma Hill Cipher – RSA sebagai algoritma kriptosistem hibrida dalam menghadapi serangan kriptografi *ciphertext-only* dengan metode *brute-force*.

3.1.2 Analisis Kebutuhan

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

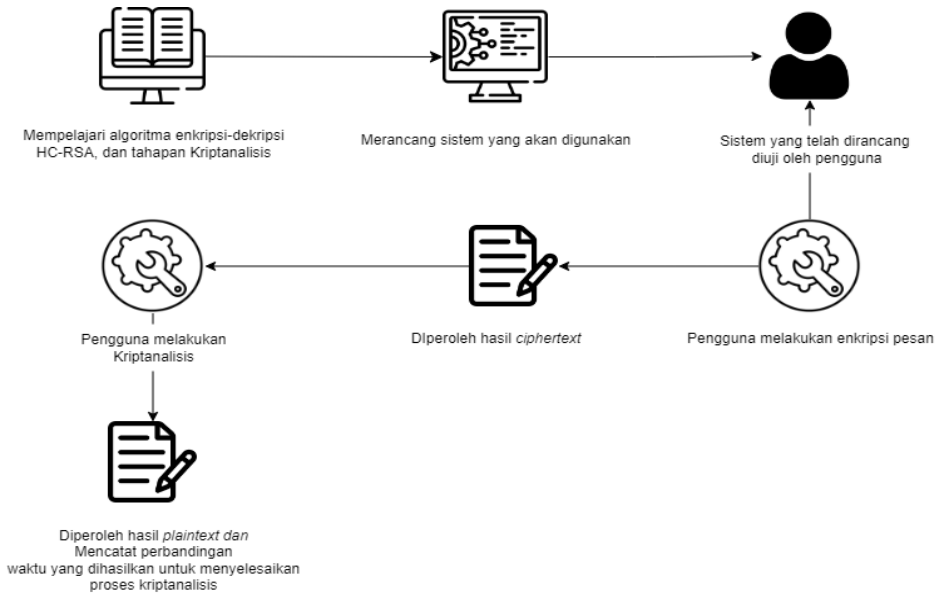
1. Kebutuhan Fungsional, merupakan kebutuhan yang mencakup proses yang dilaksanakan pada sistem. Adapun kebutuhan fungsional yang diperlukan dalam perancangan sistem ini adalah:
    a. Sistem mampu melakukan pembangkitan kunci privat dan kunci publik.

b. Sistem mampu melakukan enkripsi pesan menjadi *ciphertext* dengan menerapkan algoritma HC-RSA.
c. Sistem mampu melakukan pengujian pencarian kunci privat dan memperoleh *plaintext* dengan metode serangan *ciphertext-only* dan *brute-force.*

2. Kebutuhan non-fungsional, merupakan kebutuhan yang mencakup batasan pengembangan sistem serta batasan kemampuan sistem. Adapun kebutuhan non-fungsional yang diperlukan dalam pengembangan sistem ini adalah sebagai berikut:
   a. Mampu menampilkan hasil enkripsi dan hasil akhir dari penyerangan kriptografi serta menampilkan waktu (*running time*) yang diperlukan untuk menyelesaikan kriptanalisis.
   b. Ukuran matriks yang digunakan pada algoritma HC-RSA terbatas pada matriks 2×2.
   c. Jenis serangan untuk pengujian pada proses kriptanalisis terbatas pada serangan *brute-force* dan *ciphertext-only.*

3.1.3 Diagram Umum Sistem

Diagram umum sistem merupakan ilustrasi alur, proses, dan hubungan antarkomponen dalam sistem. Adapun perancangan alur kerja sistem digambarkan pada diagram di bawah ini



Gambar 2 Diagram Umum Sistem

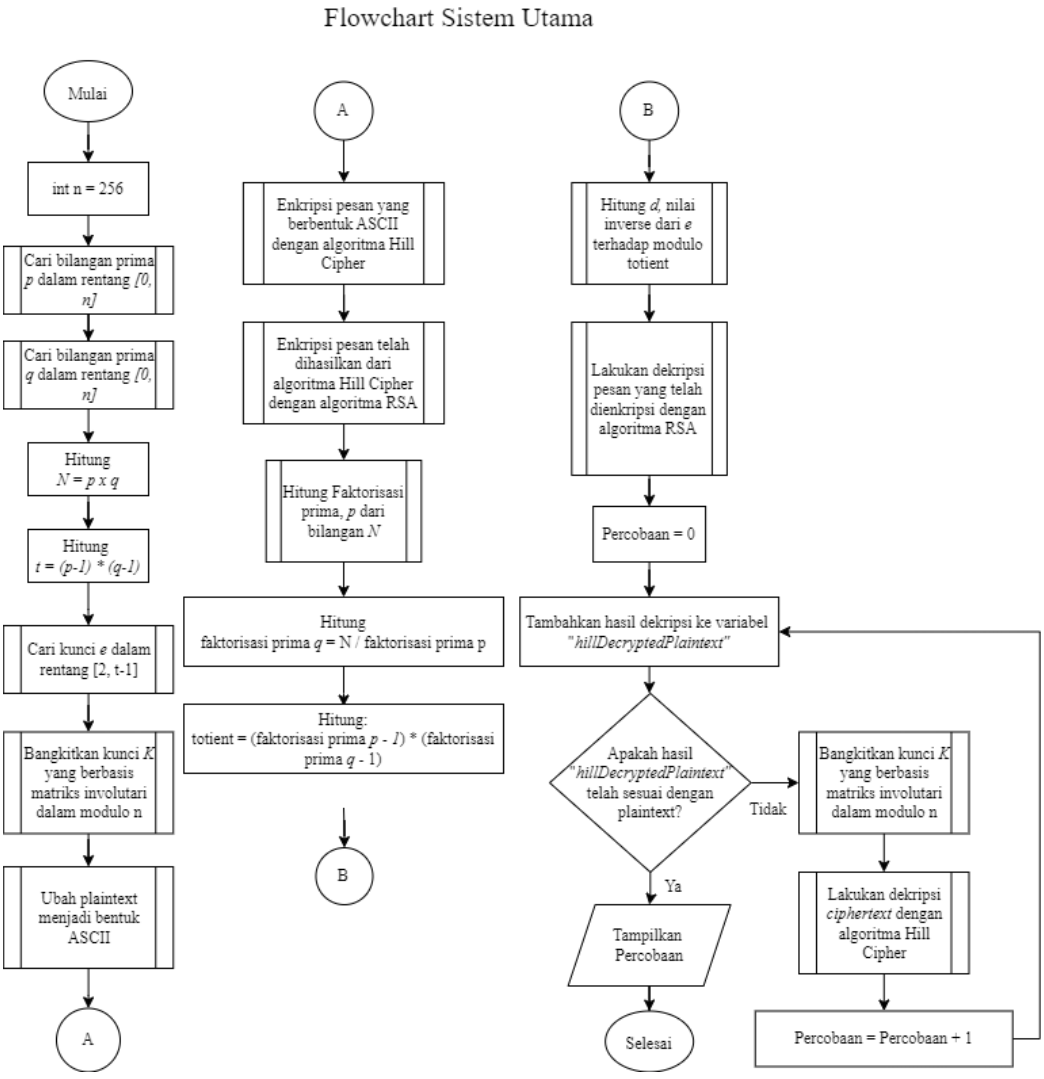
Secara umum, diagram umum sistem dapat diuraikan sebagai berikut:

1. Mempelajari algoritma enkripsi, dekripsi, dan pembangkitan kunci Hill Cipher-RSA melalui kajian literatur, mempelajari cara faktorisasi algoritma kunci publik Hill C-RSA melalui berbagai sumber seperti jurnal, buku, literatur, dan sebagainya,
2. Setelah memperoleh informasi mengenai cara kerja algoritma HC-RSA dan metode faktorisasi yang dapat digunakan untuk memfaktorisasi kunci publik algoritma HC-RSA, dilakukan perancangan sistem berdasarkan cara kerja algoritma tersebut menggunakan bahasa pemrograman C++.
3. Setelah sistem dirancang, pengguna akan melakukan pengujian sistem dengan melakukan enkripsi suatu pesan dengan menampilkan hasil *ciphertext*.
4. Setelah menguji hasil *ciphertext*, dilakukan pengujian serangan kriptografi yakni *brute-force attack* dan *ciphertext-only attack* untuk memperoleh waktu yang diperlukan untuk memecahkan *plaintext* yang dienkripsi dengan algoritma Hill Cipher-RSA.

3.2 *Flowchart*

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

3.2.1. Flowchart Sistem utama



Gambar 3 *Flowchart* Sistem utama

Flowchart sistem utama merupakan diagram yang menggambarkan urutan proses yang dijalankan pada saat penggunaan sistem. Pada sistem ini, terdapat beberapa fungsi utama yang juga dijabarkan menjadi berbagai *flowchart* lainnya yakni *flowchart* Fungsi Pembangkit Bilangan Prima, *flowchart* Fungsi Pembangkit Bilangan Acak dengan Rentang Awal dan Rentang Akhir, *flowchart* Fungsi Pemeriksaan Bilangan Prima, *flowchart* Fungsi Pengujian Primalitas, *flowchart* Fungsi Pencarian Nilai Modulo Eksponensial, *flowchart* Fungsi Konversi Bilangan ke Bentuk Biner, *flowchart* Fungsi Pembangkit Kunci *e* RSA, *flowchart* Fungsi Pembangkit Kunci Rahasia *K* Hill Cipher, *flowchart* Fungsi Invers Modulo, *flowchart* Fungsi Konversi *Plaintext* ke Bentuk ASCII, *flowchart* Fungsi Enkripsi dengan Algoritma Hill

Cipher, *flowchart* Fungsi Enkripsi dengan Algoritma RSA, *flowchart* Fungsi Faktorisasi Prima, *flowchart* Fungsi Dekripsi dengan Algoritma RSA, dan *flowchart* Fungsi Dekripsi dengan Algoritma Hill Cipher.

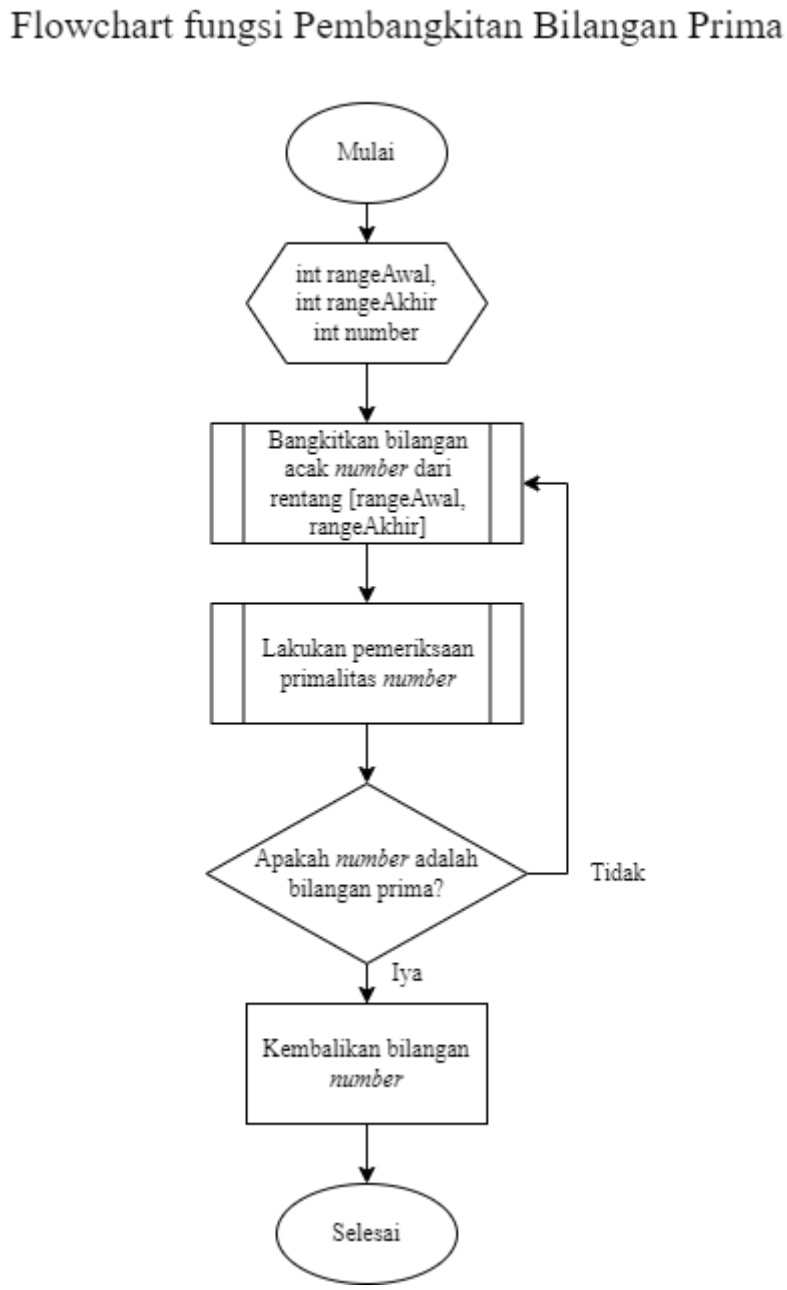
Adapun alur sistem yang dijabarkan pada *flowchart* sistem di atas adalah sebagai berikut:

a. Tahapan pembangkitan Kunci (*Key Generation*) dan Enkripsi
    1. Pada saat program dijalankan, sistem akan menentukan nilai $n = 256$ yang digunakan untuk proses operasi modulo algoritma Hill Cipher.
    2. Selanjutnya sistem akan melakukan pembentukan dua buah bilangan prima yakni p dan q.
    3. Sistem selanjutnya akan melakukan perhitungan Kunci N dengan persamaan $N = p \times q$.
    4. Sistem selanjutnya akan melakukan perhitungan nilai $\phi(N) = (p-1) \times (q-1)$
    5. Selanjutnya sistem akan melakukan pencarian Kunci e bilangan ganjil yang memiliki rentang $[2,\ \phi(N) - 1]$
    6. Sistem kemudian akan membangkitkan kunci K yang merupakan matriks involutari. Kunci K merupakan kunci rahasia yang akan digunakan pada tahapan enkripsi dan dekripsi dengan metode Hill Cipher.
    7. Sistem kemudian akan melakukan konversi *plaintext* yang diberikan ke bentuk numerik dengan memanfaatkan bilangan ASCII.
    8. Selanjutnya sistem akan melakukan proses enkripsi dengan *plaintext* yang telah dikonversi ke bentuk numerik dengan memanfaatkan algoritma enkripsi Hill Cipher.
    9. Setelah proses enkripsi dengan Hill Cipher, hasil *ciphertext* yang dihasilkan, akan dienkripsi dengan menggunakan algoritma enkripsi RSA.

b. Tahapan Pengujian Kriptanalisis
    1. Sistem akan melakukan perhitungan nilai faktorisasi prima p dan q dari bilangan N yang diperoleh sebelumnya.

2. Sistem selanjutnya akan melakukan perhitungan nilai $\phi(N)$ dari hasil faktorisasi p dan q yang diperoleh
3. Selanjutnya dilakukan perhitungan kunci $d \equiv e^{-1} \ (mod\ \phi(N))$
4. Setelah memperoleh keseluruhan kunci yang diperlukan untuk dekripsi algoritma RSA, maka sistem akan melakukan dekripsi *ciphertext* yang dihasilkan pada tahapan enkripsi dengan algoritma RSA.
5. Selanjutnya sistem akan melakukan pengujian dekripsi dengan metode *brute force* untuk mencari kunci K yang digunakan dengan membangkitkan kunci K secara acak dan dilakukan pengujian dekripsi dengan algoritma Hill Cipher hingga ditemukannya *plaintext* yang sesuai.
6. Sistem kemudian akan menampilkan jumlah percobaan yang dilakukan serta waktu yang diperlukan untuk sistem berhasil melakukan enkripsi hingga kriptanalisis.

3.2.2. Flowchart Fungsi Pembangkit Bilangan Prima



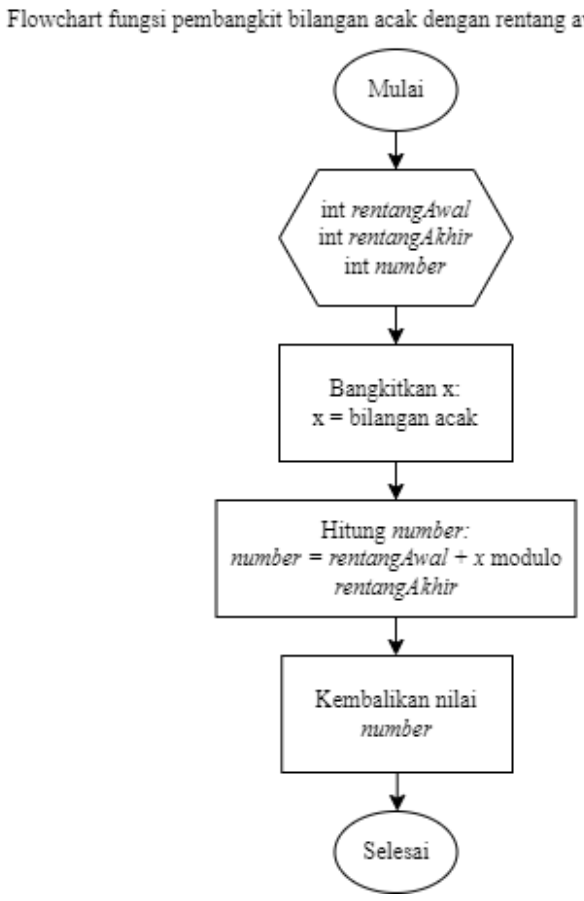
Gambar 4 *Flowchart* Fungsi Pembangkit Bilangan Prima

Flowchart Fungsi Pembangkit Bilangan Prima merupakan diagram alir yang menunjukan proses pembentukan bilangan prima yang dirancang pada sistem penelitian. Adapun tahapan atau proses yang dilaksanakan Fungsi Pembangkit Bilangan Prima adalah sebagai berikut:

1. Lakukan pembangkitan bilangan acak dari rentang yang ditentukan

2. Kemudian lakukan pemeriksaan primalitas dari bilangan yang dihasilkan
3. Periksa hasil yang diberikan dari uji primalitas apakah bilangan tersebut termasuk bilangan prima atau tidak. Jika bilangan tersebut adalah bilangan prima, maka kembalikan hasil bilangan yang diperoleh. Namun, jika bilangan yang dihasilkan bukan bilangan prima, maka dilakukan pembangkitan bilangan kembali seperti pada langkah (1).

3.2.3. Flowchart Fungsi Pembangkit Bilangan Acak dengan rentang awal dan rentang akhir



Gambar 5 *Flowchart* Fungsi Pembangkit Bilangan Acak dengan Rentang Awal dan Rentang Akhir

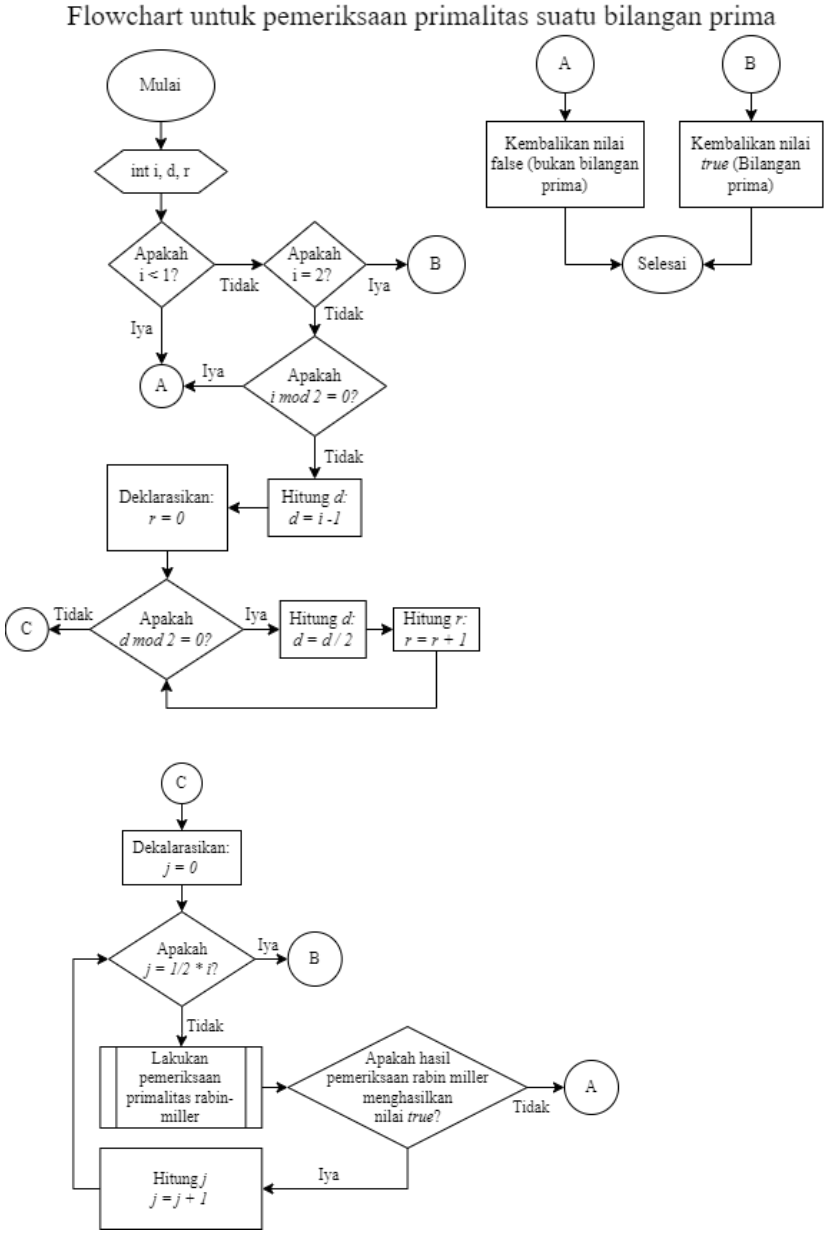
Flowchart Gambar 5 menjelaskan alur atau langkah Fungsi Pembangkit Bilangan Acak yang mampu mengembalikan nilai acak dalam rentangan nilai batasan awal dan batasan akhir. Adapun penjelasan tahapannya adalah sebagai berikut:

1. Lakukan pembangkitan suatu bilangan acak x.
2. Ubah nilai atau rentang x menjadi berada di dalam rentangan batasan awal dan batasan akhir dengan:

   $$number = batasan\ awal + x\ modulo\ batasan\ akhir$$

3. Sistem akan mengembalikan nilai $number$ dari fungsi ini ke sistem utama.

3.2.4. Flowchart Fungsi Pemeriksaan Bilangan Prima (Menentukan apakah suatu bilangan merupakan bilangan prima atau komposit)

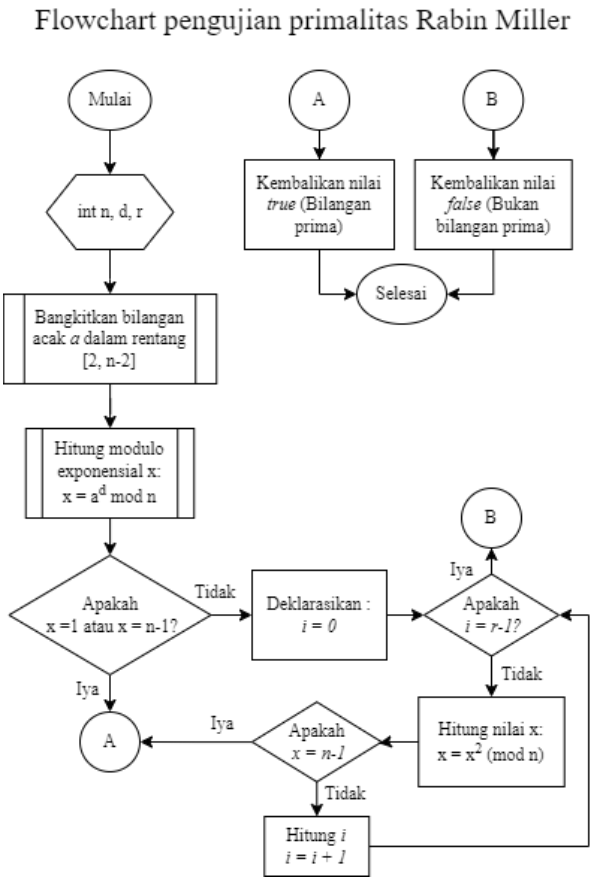


Gambar 6 *Flowchart* Fungsi Pemeriksaan Bilangan Prima

Flowchart pada Gambar 6 berfungsi untuk melakukan pemeriksaan apakah sebuah bilangan dapat dikategorikan sebagai bilangan prima atau tidak. Fungsi di atas juga menambahkan validasi prima dengan metode Rabin-Miller. Adapun rangkaian alur yang dirancang pada fungsi tersebut yang diterapkan pada sistem adalah sebagai berikut:

1. Misalkan terdapat sebuah bilangan N yang akan diuji apakah bilangan tersebut termasuk bilangan prima atau tidak.
2. Jika $N \leq 1$, maka sudah pasti N bukan merupakan bilangan prima, maka sistem akan mengembalikan nilai *false.*
3. Jika $N = 2$, maka sudah pasti N merupakan bilangan prima, maka sistem akan mengembalikan nilai *true.*
4. Jika N merupakan bagian dari bilangan genap, maka sudah pasti N bukan merupakan bilangan prima, maka sistem akan mengembalikan nilai *false*.
5. Jika N merupakan bilangan ganjil, maka lakukan perhitungan $d = i - 1$ serta deklarasikan nilai $r = 0$.
6. Selanjutnya lakukan perhitungan $d = d \div 2$ dan $r = r + 1$ hingga $d \; modulo \; 2 \neq 0$.
7. Selanjutnya dilakukan perulangan sebanyak $\frac{N}{2}$ kali dalam pengujian validasi dengan algoritma Rabin-Miller
8. Jika dari hasil validasi algoritma Rabin-Miller menghasilkan nilai prima, maka bilangan tersebut adalah prima sehingga sistem akan mengembalikan nilai *true*.
9. Jika dari hasil validasi algoritma Rabin-Miller menghasilkan nilai bukan prima, maka bilangan tersebut bukanlah bilangan prima, sehingga sistem akan mengembalikan nilai *false*.

3.2.5. Flowchart Fungsi Pengujian Primalitas dengan metode Rabin-Miller



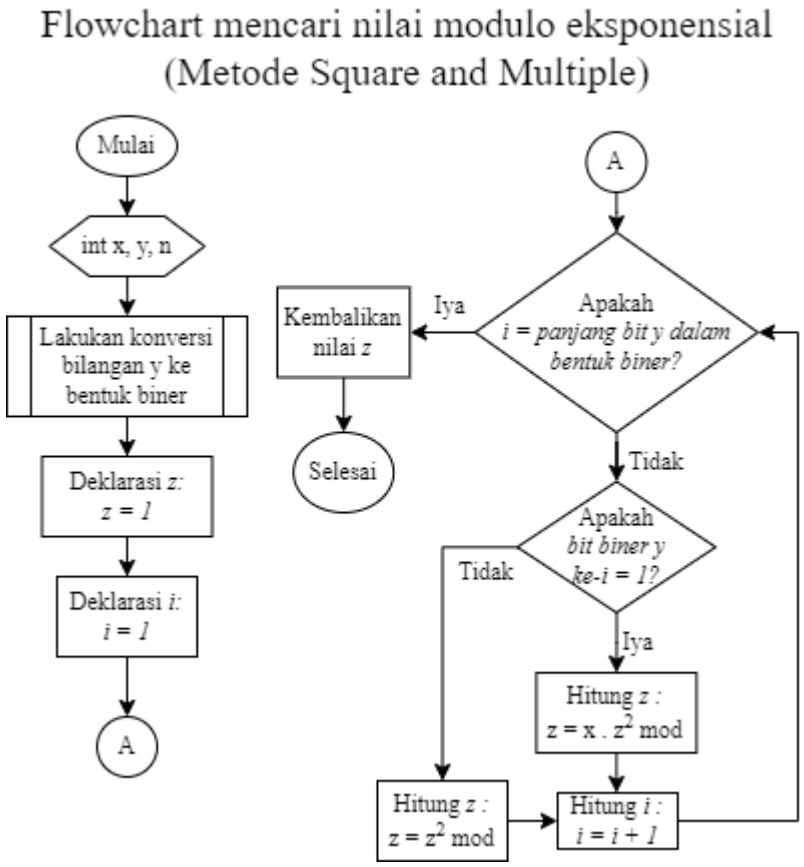
Gambar 7 *Flowchart* Fungsi Pengujian Primalitas dengan metode Rabin-Miller

Flowchart Fungsi Pengujian Primalitas dengan metode Rabin-Miller merupakan diagram yang merepresentasikan alur kerja perancangan sistem pemeriksaan primalitas suatu bilangan dengan memanfaatkan teori primalitas Rabin-Miller. Adapun tahapan alur kerja fungsi pemeriksaan primalitas bilangan dengan metode Rabin-Miller yang dirancang pada sistem ini adalah sebagai berikut:

1. Lakukan pembangkitan bilangan acak, misalkan bilangan a yang memiliki rentang $2 \leq a \leq n-2$ dimana n merupakan bilangan yang akan diperiksa primalitasnya.
2. Lakukan perhitungan nilai modulo eksponensial $x = a^d \ mod \ n$.
3. Jika nilai x yang diperoleh bernilai $x = 1$ atau $x = n-1$, maka bilangan n dianggap prima.

4. Namun jika x tidak memperoleh nilai $x = 1$ atau $x = n - 1$, maka dilakukan iterasi sebanyak $r - 1$ kali pada perhitungan $x = x^2 \ (mod\ n)$.
5. Jika dari perhitungan tersebut diperoleh nilai $x = n - 1$, maka bilangan n dianggap prima.
6. Jika pada tahapan akhir iterasi nilai $x = n - 1$ tidak ditemukan, maka n dianggap bukan bilangan prima.

3.2.6. Flowchart Fungsi Pencarian Nilai Modulo Eksponensial dengan Metode Square and Multiply



Gambar 8 *Flowchart* Fungsi Pencarian Nilai Modulo Eksponensial dengan Metode *Square and Multiply*

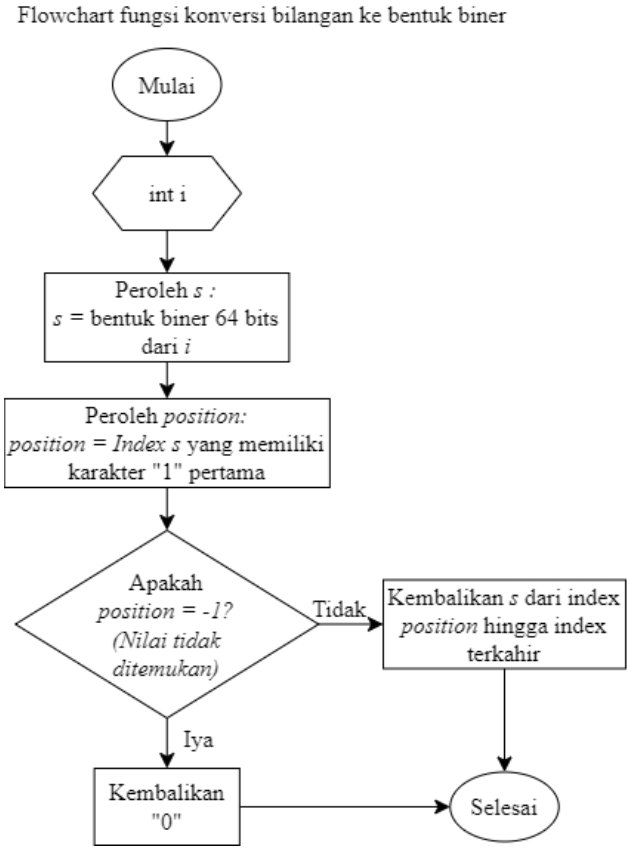
Flowchart pada Gambar 8 menggambarkan rancangan alur kerja algoritma *Square and Multiply* dalam menyelesaikan pencarian nilai modulo eksponensial. Adapun tahapan yang dijelaskan pada *flowchart* tersebut dapat diuraikan menjadi:

1. Misalkan diketahui sebuah modulo exponensial yang besar yang dimasukan pada fungsi tersebut yakni:

$$x^y \ mod \ m$$

2. Ubah nilai y menjadi bentuk biner
3. Tetapkan nilai awal untuk z, yakni $z = 1$
4. Lakukan perulangan sepanjang jumlah digit biner y, jika ditemukan digit '1', maka ubah nilai z menjadi $z = x \, . z^2 \ (mod \ m)$. Sedangkan jika digit '0' ditemukan, maka ubah nilai z menjadi $z = z^2 \ (mod \ m)$.
5. Selanjutnya sistem akan mengembalikan nilai z yang diperoleh.

3.2.7. Flowchart Fungsi Konversi Bilangan ke Bentuk Biner



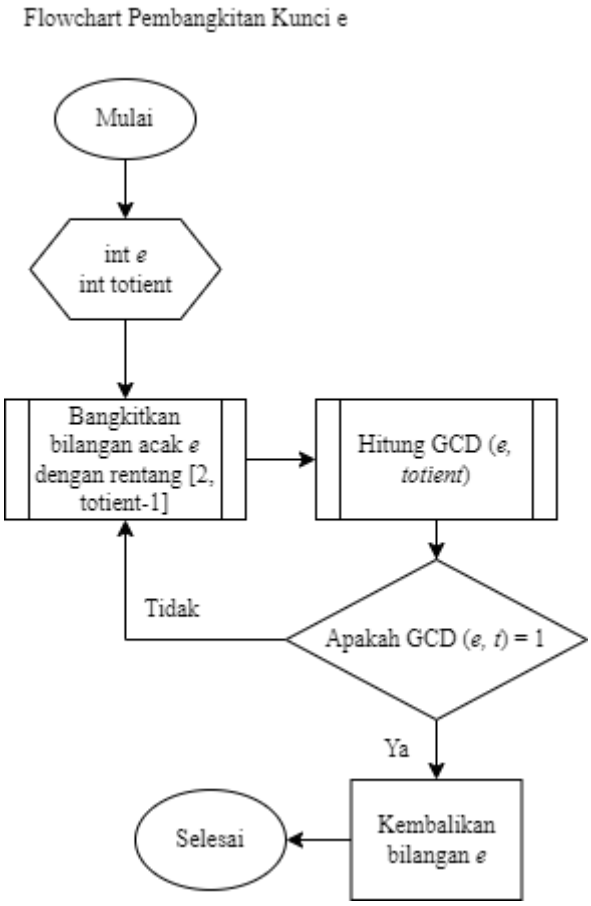
Gambar 9 *Flowchart* Fungsi Konversi Bilangan ke Bentuk Biner

Flowchart pada Gambar 9 mendeskripsikan alur dari fungsi untuk melakukan konversi suatu bilangan ke bentuk biner. Adapun tahapan yang digambarkan pada *flowchart* tersebut adalah sebagai berikut:

1. Ubah suatu bilangan, misalkan bilangan i menjadi bentuk biner dengan ukuran 64 bits dalam bentuk teks (*string*).
2. Sistem kemudian akan melakukan pencarian karakter "1" sepanjang ukuran biner.

3. Jika karakter “1” ditemukan, maka fungsi akan menampung posisi karakter “1” ditemukan pada biner tersebut. Kemudian fungsi akan mengembalikan nilai biner dalam bentuk string dari posisi karakter “1” pertama hingga posisi terakhir pada biner.
4. Jika karakter “1” tidak ditemukan, fungsi tersebut akan mengembalikan karakter “0”.

*3.2.8. Flowchart Fungsi Pembangkit Kunci **e** RSA*



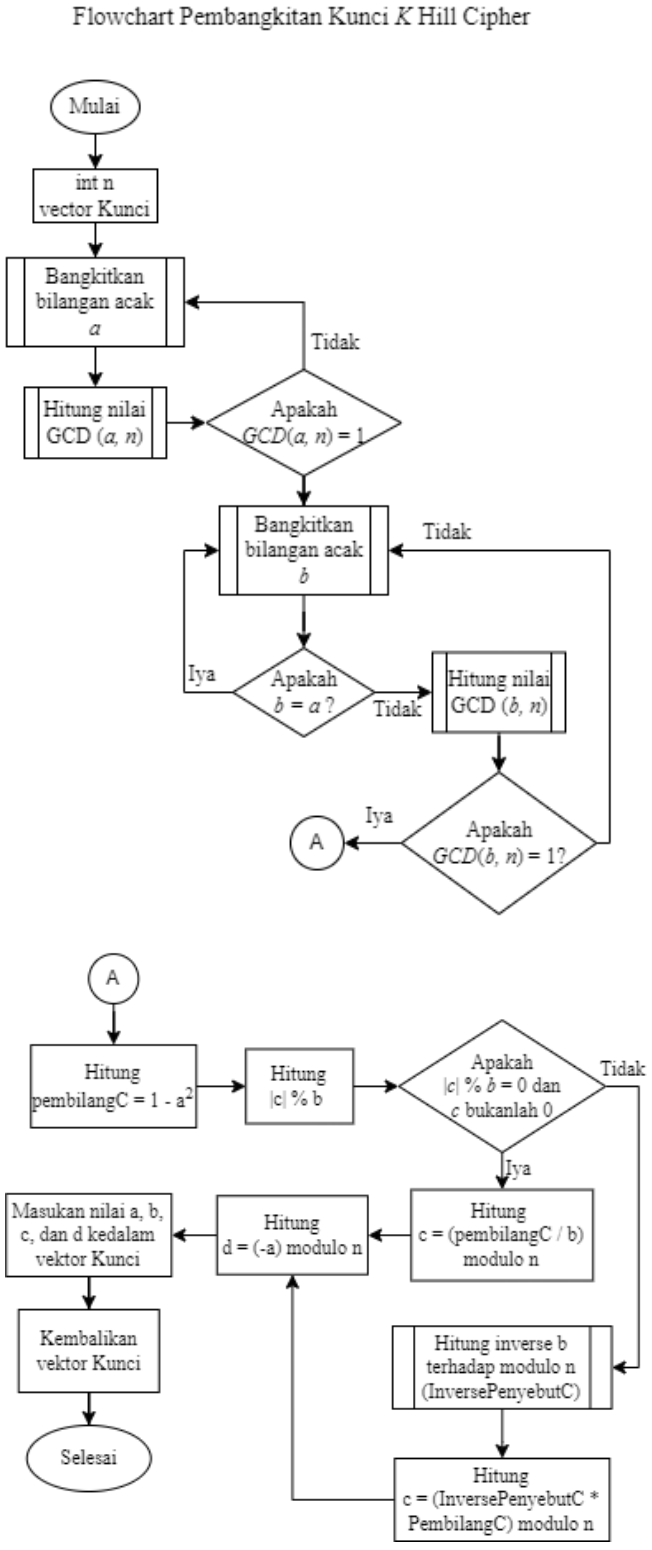
Gambar 10 *Flowchart* Fungsi Pembangkit Kunci *e* RSA

Flowchart Fungsi Pembangkit Kunci *e* RSA merupakan diagram yang merepresentasikan alur dan tahapan yang dirancang dalam pembangkitan kunci publik *e* RSA. Adapun uraian dari tahapan yang dirancang adalah sebagai berikut:

1. Bangkitkan sebuah bilangan acak e dengan rentang $2 \leq e \leq totient - 1$
2. Kemudian lakukan perhitungan $GCD(e, totient)$. Jika hasil perhitungan $GCD(e, totient)$ = 1 atau e relatif prima terhadap

totient, maka kembalikan bilangan *e* tersebut. Jika tidak, maka ulangi pembangkitan bilangan acak *e*.

*3.2.9. Flowchart Fungsi Pembangkit Kunci Rahasia **K** Hill Cipher*

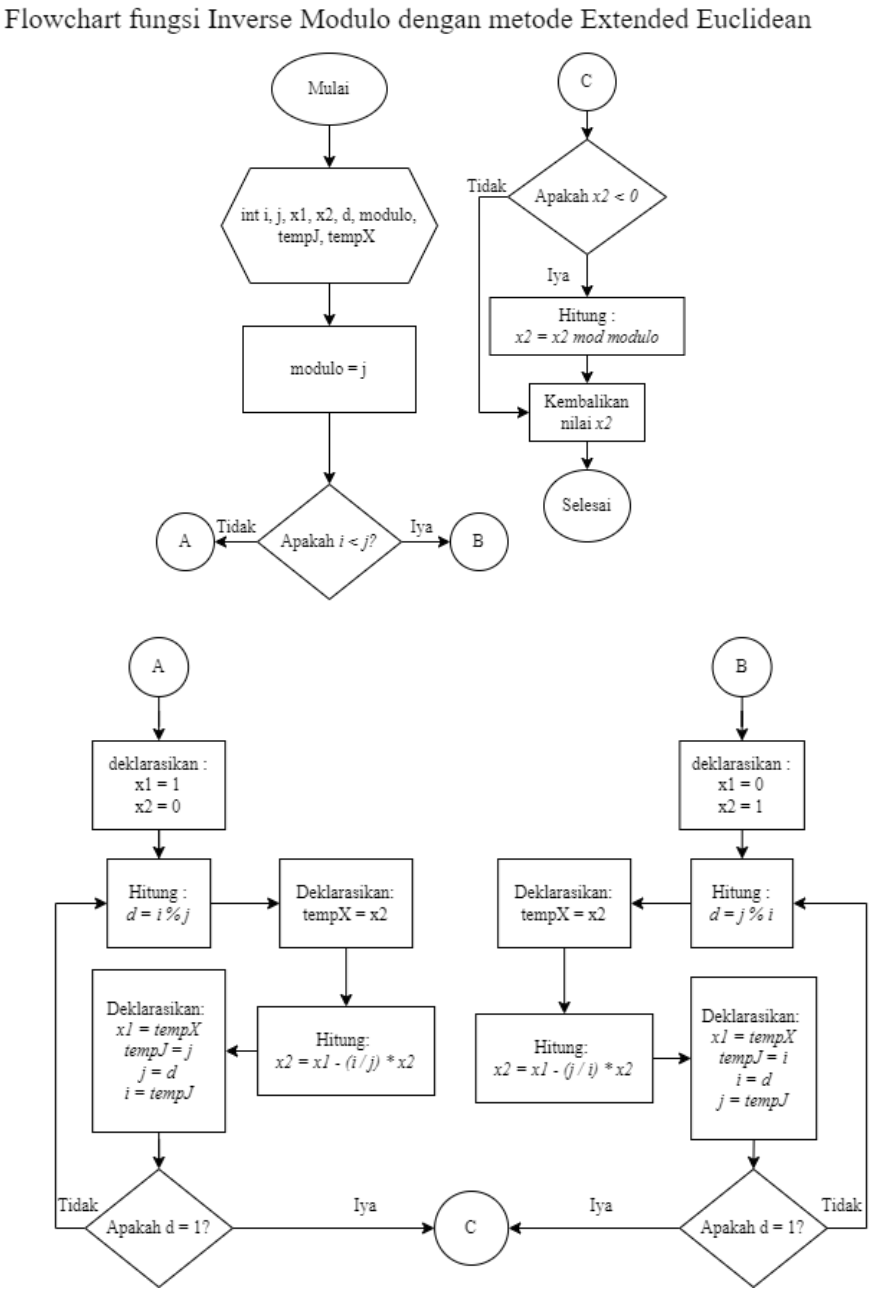


Gambar 11 *Flowchart* Fungsi Pembangkit Kunci K Hill Cipher

Flowchart Fungsi Pembangkit Kunci K pada Gambar 11, merepresentasikan alur sistem dalam pembentukan Kunci K yang berbasis matriks Involuatari. Adapun tahapan atau alur yang dirancang dalam pembentukan kunci K yang diuraikan pada *flowchart* di atas adalah sebagai berikut:

1. Sistem akan melakukan pembangkitan nilai a secara acak namun dengan syarat a harus relatif prima dengan n
2. Selanjutnya sistem akan melakukan pembangkitan nilai b secara acak namun dengan syarat b harus relatif prima dengan n serta $b \neq a$.
3. Sistem akan melakukan perhitungan terhadap nilai pembilang pembentuk c yakni dengan persamaan $upperC = 1 - a^2$.
4. Lakukan pemeriksaan apakah bilangan c yang dibentuk merupakan bilangan pecahan atau bilangan bulat dengan melakukan pemeriksaan apakah $upperC$ modulo $b = 0$
5. Jika c yang dihasilkan adalah bilangan pecahan, maka lakukan perhitungan invers modulo terhadap penyebut c (*lowerC*) kemudian hasil invers modulo dikalikan dengan pembilang c (*UpperC*)
6. Jika c yang dihasilkan adalah bilangan bulat, maka c hanya perlu dioperasikan dengan operasi modulo.
7. Selanjutnya hitung nilai $d = -a \ (mod\ n)$
8. Masukan seluruh nilai a, b, c, d ke variabel vector yang telah disediakan
9. Sistem akan mengembalikan vektor tersebut.

3.2.10. Flowchart Fungsi Invers Modulo dengan metode Extended Euclidean



Gambar 12 Flowchart Fungsi Invers modulo dengan metode *Extended Euclidean*

Flowchart Invers Modulo merupakan representasi fungsi mencari nilai invers modulo yang digunakan pada sistem. Fungsi tersebut digunakan untuk menghitung nilai invers suatu bilangan dalam modulo bilangan tertentu dengan memanfaatkan metode *Extended Euclidean*. Adapun alur atau tahapan yang diterapkan dalam pembentukan fungsi invers modulo:

1. Pada tahapan awal, sistem akan mendeklarasikan beberapa variabel yang akan digunakan serta menetapkan nilai dasar yang akan digunakan pada perhitungan selanjutnya
2. Kemudian sistem akan melakukan pemeriksaan apakah nilai modulo, misalkan j lebih besar daripada bilangan yang akan dilakukan perhitungan inversnya, misalkan i
3. Jika $i > j$ maka:
   i. Deklarasikan nilai awal konstanta $X_1 = 1$ serta $X_2 = 0$
   ii. Kemudian hitung nilai $d = i \ mod \ j$.
   iii. Deklarasikan sebuah variabel sementara, misalkan $tempX$ untuk menampung nilai dari konstanta X_2
   iv. Ubah nilai konstanta $X_2 = X_1 - (i \ \div j) * X_2$
   v. Ubah nilai konstanta X_1 menjadi nilai X_2 yang sebelumnya dengan memanfaatkan nilai dari variabel sebelumnya
   vi. Deklarasikan sebuah variabel sementara untuk menampung nilai j, misalkan $tempJ$. Kemudian masukan nilai j ke variabel $tempJ$ ($tempJ = j$).
   vii. Ubah nilai j menjadi nilai d serta ubah nilai i menjadi nilai j yang sebelumnya dengan memanfaatkan variabel sementara $tempJ$.
   viii. Langkah (i) hingga (vii) dilakukan berulang kali hingga nilai $d = 1$.
   ix. Konstanta X_2 merupakan hasil akhir yang akan digunakan sebagai solusi pencarian nilai invers modulo. Jika nilai X_2 yang diperoleh adalah bilangan positif, maka sistem akan langsung mengembalikan nilai tersebut. Namun jika nilai X_2 yang diperoleh adalah bilangan negatif, maka nilai X_2 harus dilakukan operasi modulo sebelum dikembalikan hasilnya.
4. Jika $i < j$, maka:
   i. Deklarasikan nilai awal konstanta $X_1 = 0$ serta $X_2 = 1$
   ii. Kemudian hitung nilai $d = j \ mod \ i$.

iii. Deklarasikan suatu variabel sementara, misalkan $tempX$ untuk menampung nilai dari konstanta X_2

iv. Ubah nilai konstanta $X_2 = X_1 - (j \div i) * X_2$

v. Ubah nilai konstanta X_1 menjadi nilai X_2 yang sebelumnya dengan memanfaatkan nilai dari variabel sebelumnya

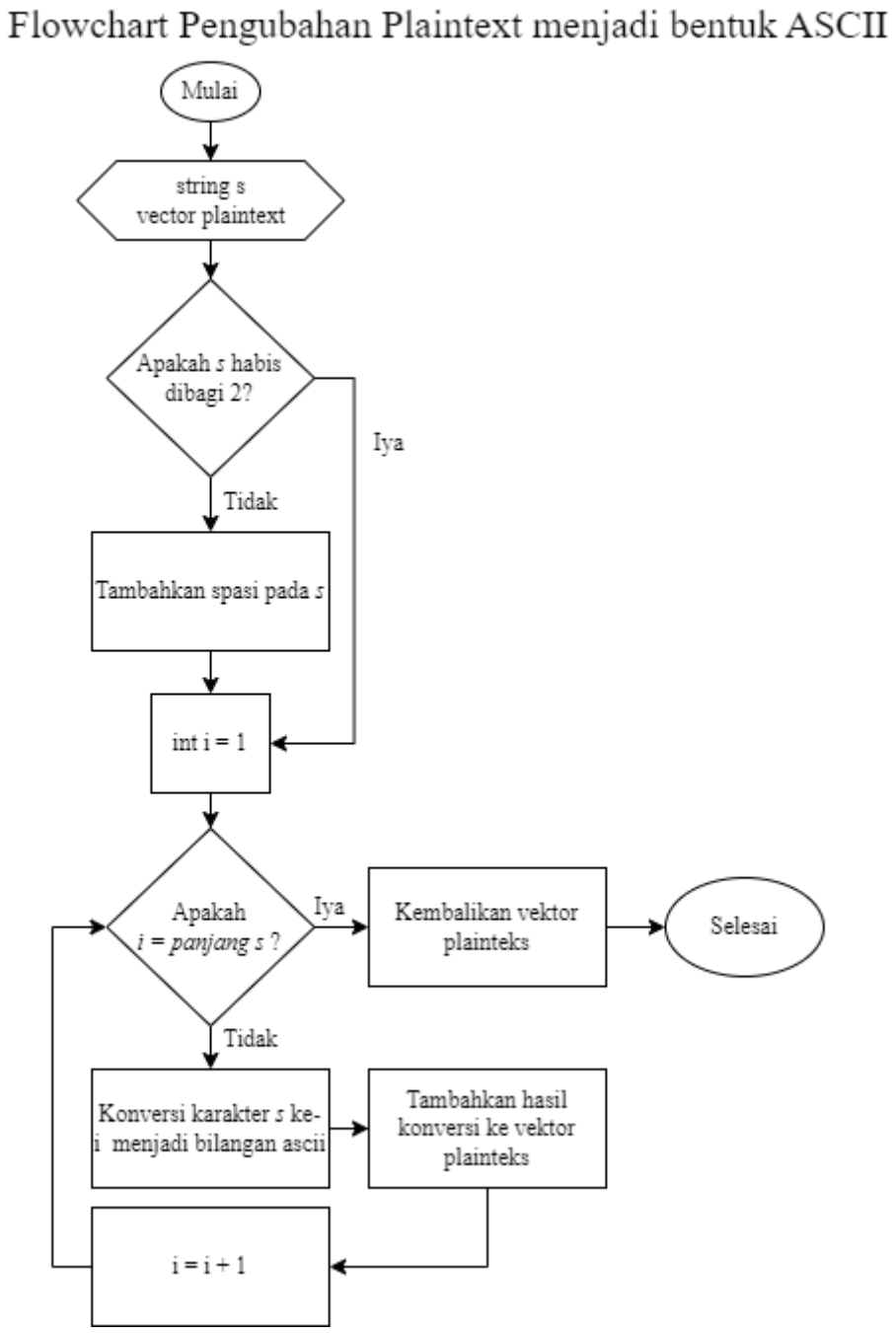
vi. Deklarasikan suatu variabel sementara untuk menampung nilai j, misalkan $tempJ$. Kemudian masukan nilai i ke variabel $tempJ$ ($tempJ = i$).

vii. Ubah nilai i menjadi nilai d serta ubah nilai j menjadi nilai j yang sebelumnya dengan memanfaatkan variabel sementara $tempJ$.

viii. Langkah (i) hingga (vii) dilakukan berulang kali hingga nilai $d = 1$.

ix. Konstanta X_2 merupakan hasil akhir yang akan digunakan sebagai solusi pencarian nilai invers modulo. Jika nilai X_2 yang diperoleh adalah bilangan positif, maka sistem akan langsung mengembalikan nilai tersebut. Namun jika nilai X_2 yang diperoleh adalah bilangan negatif, maka nilai X_2 harus dilakukan operasi modulo sebelum dikembalikan hasilnya.

3.2.11. Flowchart Fungsi Konversi Plaintext ke Bentuk ASCII



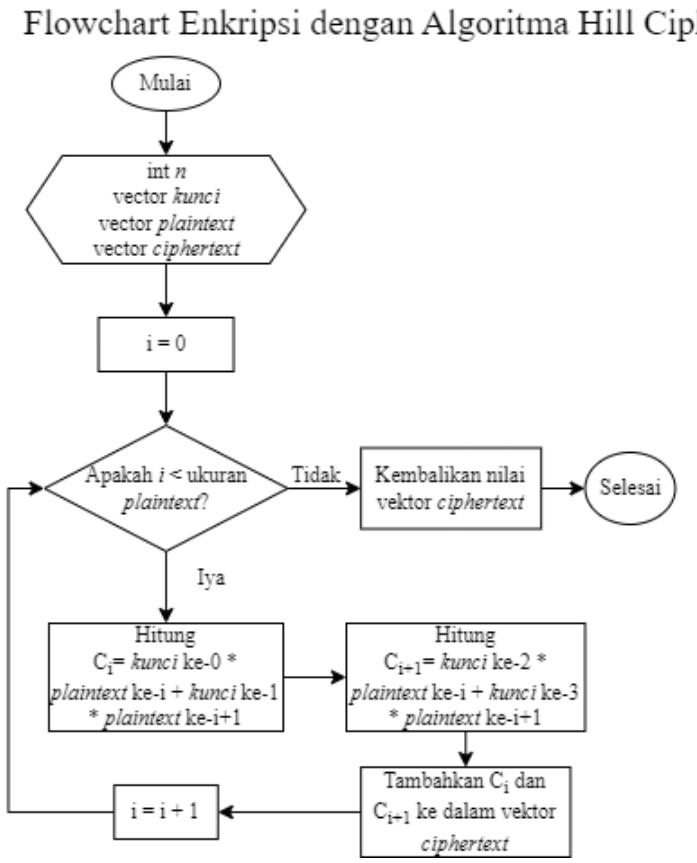
Gambar 13 *Flowchart* Fungsi Konversi *Plaintext* menjadi Bentuk ASCII

Flowchart pada Gambar 13 menggambarkan rancangan proses konversi suatu *plaintext* dari bentuk tulisan ke bentuk numerik dengan memanfaatkan nilai ASCII dari karakter tulisan. Adapun rincian tahapan yang dirancang adalah sebagai berikut:

1. Misalkan terdapat sebuah *plaintext s* yang akan dikonversi ke dalam bentuk ASCII

2. Sistem akan melakukan perhitungan Panjang s. Jika panjang teks s adalah ganjil, maka sistem akan menambahkan spasi " " ke teks s sehingga s akan memiliki panjang genap.
3. Kemudian sistem akan melakukan iterasi sepanjang panjang dari teks s dan dilakukan konversi ke bentuk ASCII per karakter teks s
4. Kemudian hasil konversi tersebut akan ditampung pada variabel dan dikembalikan nilai fungsinya.

3.2.12. Flowchart Fungsi Enkripsi Algoritma Hill Cipher



Gambar 14 *Flowchart* Fungsi Enkripsi Algoritma Hill Cipher

Flowchart pada Gambar 14 merepresentasikan rancangan alur enkripsi dengan algoritma Hill Cipher pada sistem yang dapat dijabarkan sebagai berikut:

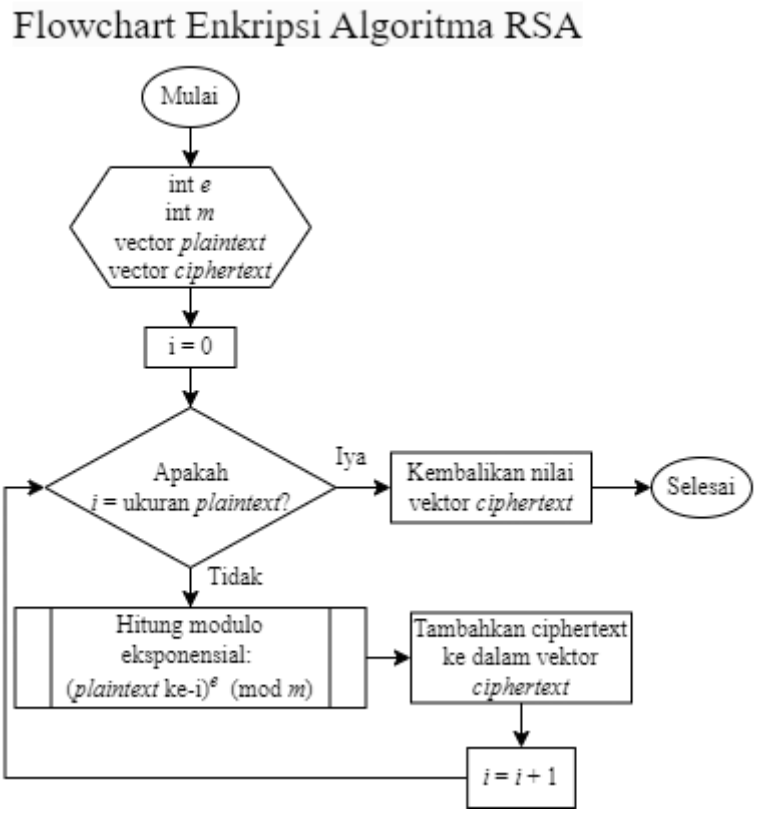
1. Lakukan iterasi sepanjang *plaintext* yang diberikan.
2. Untuk setiap komponen *plaintext* lakukan perhtiungan enkripsi yakni:

$$C_i = Kunci_0 * Plaintext_i + Kunci_1 * plaintext_{i+1}$$

$$C_{i+1} = Kunci_2 * plaintext_i + Kunci_3 * plaontext_{i+1}$$

3. Setiap hasil enkripsi ditambahkan ke variabel vektor yang akan menjadi nilai kembalian suatu fungsi.

3.2.13. Flowchart Fungsi Enkripsi Algoritma RSA



Gambar 15 *Flowchart* Fungsi Enkripsi Algoritma RSA

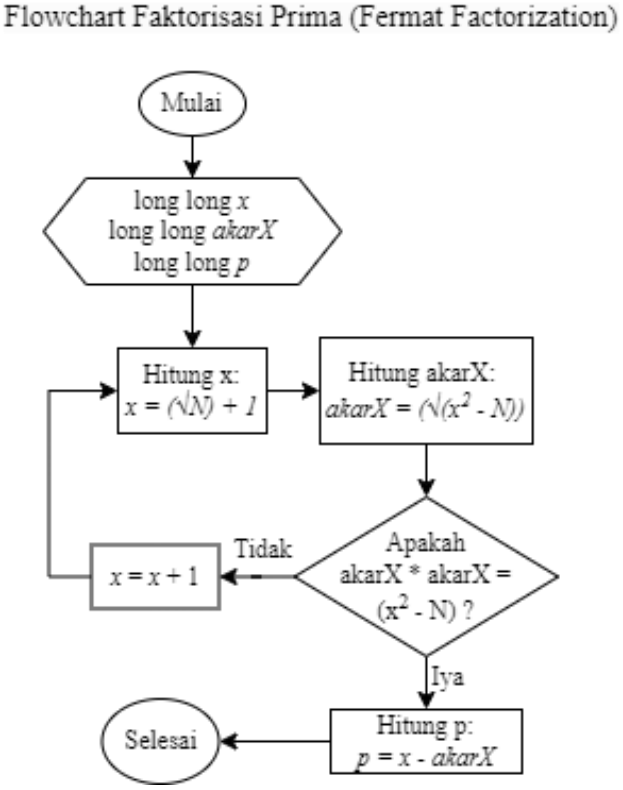
Flowchart pada Gambar 14 merepresentasikan rancangan alur enkripsi dengan algoritma RSA pada sistem yang dapat dijabarkan sebagai berikut:

1. Lakukan iterasi sepanjang *plaintext* yang diberikan.
2. Untuk setiap komponen *plaintext* lakukan perhitungan enkripsi yakni:

$$C_i = plaintext_i^e \ (mod\ N)$$

3. Perhitungan pada langkah kedua dilakukan dengan menggunakan metode pencarian modulo eksponensial. Pada sistem ini metode yang digunakan adalah "*Square and Multiply".*
4. Untuk setiap hasil *ciphertext* yang diperoleh, sistem akan menampungnya pada variabel yang akan digunakan sebagai nilai kembalian fungsi.

3.2.14. Flowchart Fungsi Faktorisasi Prima dengan Metode Fermat's Factorization



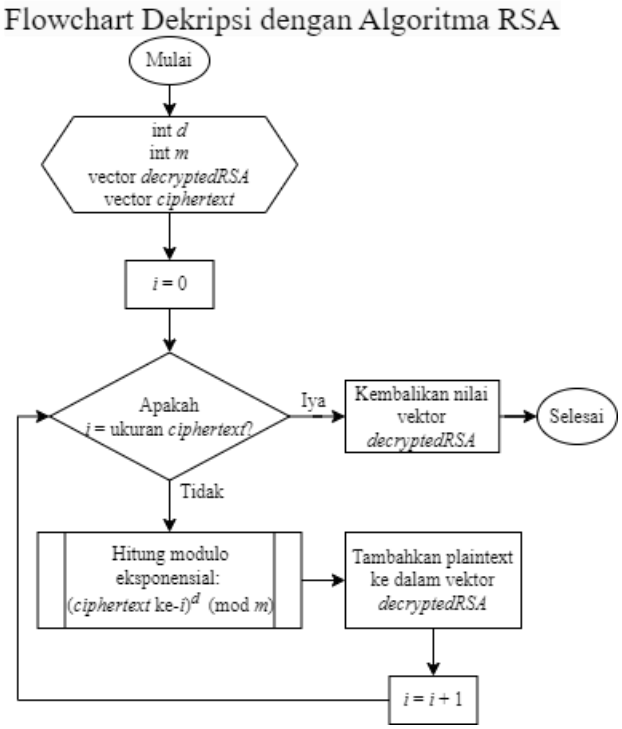
Gambar 16 Flowchart Fungsi Faktorisasi Prima dengan Metode *Fermat's factorization*

Flowchart faktorisasi prima adalah diagram yang menunjukan proses yang dirancang pada sistem untuk melakukan faktorisasi bilangan prima dari suatu bilangan komposit. Metode yang digunakan untuk merancang fungsi faktorisasi ini adalah dengan memanfaatkan metode *Fermat Factorization.* Tahapan alur dan cara kerja sistem dalam membentuk sistem ini adalah sebagai berikut:

1. Misalkan terdapat bilangan N yang akan dilakukan faktorisasi bilangan prima pembentuk N. Selanjutnya lakukan perhitungan $x = \sqrt{N} + 1$.
2. Kemudian lakukan perhitungan $akarX = \sqrt{x^2 - N}$.
3. Selanjutnya dilakukan pemeriksaan apakah akarX merupakan akar kuadrat sempurna. Jika, akarX menghasilkan akar kuadrat sempurna, maka hitung nilai $p = x - akarX$ sedangkan jika

akarX bukan merupakan akar kuadrat sempurna, maka $x = x + 1$.

3.2.15. Flowchart Fungsi Dekripsi Algoritma RSA



Gambar 17 *Flowchart* Fungsi Dekripsi Algoritma RSA

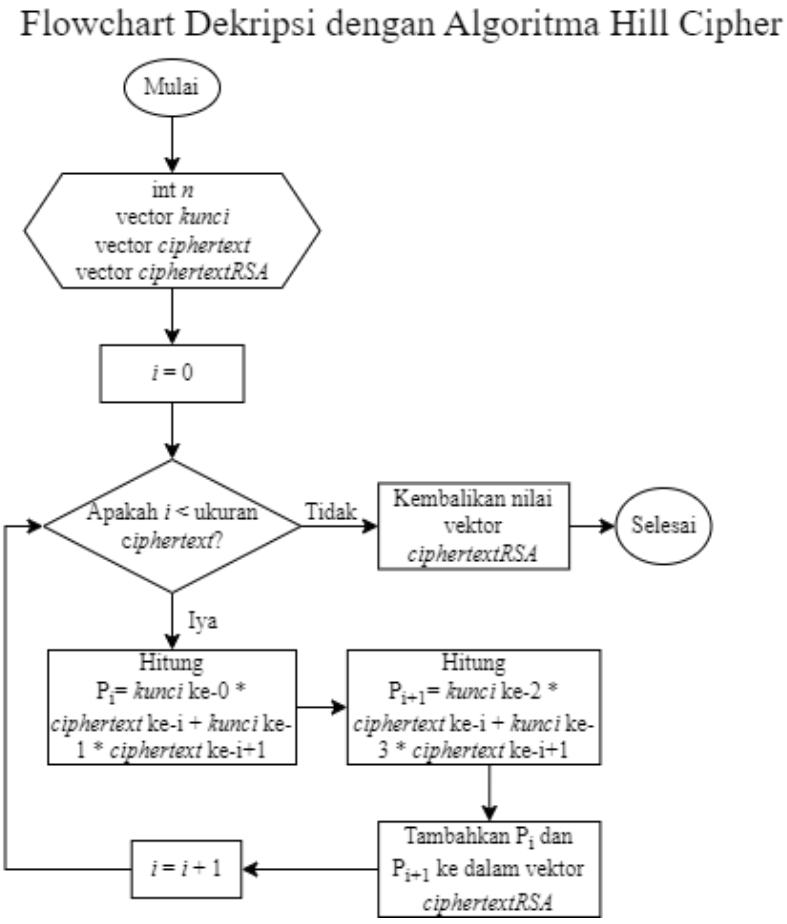
Flowchart pada Gambar 17 merepresentasikan rancangan alur enkripsi dengan algoritma RSA pada sistem yang dapat dijabarkan sebagai berikut:

1. Lakukan iterasi sepanjang *ciphertext* yang diberikan.
2. Untuk setiap komponen *ciphertext* lakukan perhitungan dekripsi yakni:

$$P_i = ciphertext_i^d \ (mod\ N)$$

3. Perhitungan pada langkah kedua dilakukan dengan menggunakan metode pencarian modulo eksponensial. Pada sistem ini metode yang digunakan adalah "*Square and Multiply".*
4. Untuk setiap hasil *plaintext* yang diperoleh, sistem akan menampungnya pada variabel yang akan digunakan sebagai nilai kembalian fungsi.

3.2.16. Flowchart Fungsi Dekripsi dengan Algoritma Hill Cipher



Gambar 18 *Flowchart* Fungsi Dekripsi dengan Algoritma Hill Cipher

Flowchart diatas merepresentasikan rancangan alur dekripsi dengan algoritma Hill Cipher pada sistem yang dapat dijabarkan sebagai berikut:

1. Lakukan iterasi sepanjang *ciphertext* yang diberikan.
2. Untuk setiap komponen *ciphertext* lakukan perhtiungan dekripsi yakni:

$$P_{Hi} = Kunci_0\ Ciphertext_i + Kunci_1 * Ciphertext_{i+1}$$

$$P_{Hi+1} = Kunci_2 * Ciphertext_i + Kunci_3 * Ciphertext_{i+1}$$

3. Setiap hasil dekripsi ditambahkan ke variabel vektor yang akan menjadi nilai kembalian suatu fungsi.

BAB IV

IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1. Perhitungan Kompleksitas Algoritma

Kompleksitas Algoritma berfokus pada analisis struktur dan kinerja suatu algoritma. Suatu algoritma tidak dapat diukur menggunakan satuan waktu seperti pada pengukuran lainnya. Hal tersebut disebabkan oleh kinerja operasi suatu algoritma dapat dipengaruhi oleh faktor kemampuan prosesor suatu mesin. Oleh sebab itu, pengukuran kompleksitas algoritma dapat membantu pemahaman tingkat efisiensi suatu algoritma dalam menyelesaikan suatu permasalahan.

4.1.1. Kompleksitas Algoritma Fungsi Random-Number

Tabel 1 Kompleksitas Algoritma Fungsi Random-Number

Fungsi randomNumber(int rangeAwal, rangAkhir)	C	#	C#
int number = rangeAwal + rand() % rangeAkhir;	C1	1	C1
return number;	C2	1	C2

$$T(m,n) = C1 + C2$$

$$= \theta(1)$$

4.1.2. Kompleksitas Algoritma Fungsi GCD

Tabel 2 Fungsi GCD (*Greatest Common Divisor*)

Fungsi GCD(i, j)

int k = i % j;
if (k == 0) {
  return j;
}
return GCD(j, k);

Berdasarkan pembahasan kompleksitas Algoritma Euclidean-GCD pada Landasan Teori, maka kompleksitas Fungsi GCD adalah $T(i,j) = \theta(\log(\min(i,j))$

4.1.3. Kompleksitas Algoritma Fungsi Inverse-Modulo

Tabel 3 Kompleksitas Algoritma Fungsi Inverse-Modulo

Fungsi InverseModulo(int i, int j)	C	#	C#
int x1, x2, tempJ, tempX, d, modulo;	C1	1	C1
modulo = j;	C2	1	C2
if (i > j){	C3	1	C3
x1 = 1;	C2	1	C2
x2 = 0;	C2	1	C2
do {	C4	i+1	(i+1)C4
d = i % j;	C2	i+1	(i+1)C2
tempX = x2;	C2	i+1	(i+1)C2
x2 = x1 - (i / j) * x2;	C2	i+1	(i+1)C2
x1 = tempX;	C2	i+1	(i+1)C2
tempJ = i;	C2	i+1	(i+1)C2
i = d;	C2	i+1	(i+1)C2
j = tempJ;	C2	i+1	(i+1)C2
} while(d != 1);	C4	i+1	(i+1)C4
}			
else {	C5	1	C5
x1 = 0;	C2	1	C2
x2 = 1;	C2	1	C2
do {	C2	1	C2
d = j % i;	C4	j+1	(j+1)C4
tempX = x2;	C2	j+1	(j+1)C2
x2 = x1 - (j / i) * x2;	C2	j+1	(j+1)C2
x1 = tempX;	C2	j+1	(j+1)C2
tempJ = i;	C2	j+1	(j+1)C2
i = d;	C2	j+1	(j+1)C2
j = tempJ;	C2	j+1	(j+1)C2
} while(d != 1);	C4	j+1	(j+1)C4
}			
if (x2 < 0) {	C3	1	C3
return (x2 % modulo + modulo) % modulo;	C5	1	C5
}			
return x2;	C5	1	C5
}			

$$T(i,j) = C1 + C2 + C3 + C2 + C2 + (i+1)C4 + (i+1)C2 + (i+1)C2$$
$$+ (i+1)C2 + (i+1)C2 + (i+1)C2 + (i+1)C2 + (i+1)C2$$
$$+ (i+1)C4 + C5 + C2 + C2 + C2 + (j+1)C4 + (j+1)C2$$
$$+ (j+1)C2 + (j+1)C2 + (j+1)C2 + (j+1)C2 + (j+1)C2$$
$$+ (j+1)C2 + (j+1)C2 + (j+1)C4 + C3 + C5 + C5$$
$$= C1 + 5C2 + 7(i+1)C2 + 8(j+1)C2 + 2C3 + 2(i+1)C4$$
$$+ 2(j+1)C4 + 3C5$$
$$= C1 + (7i + 8j + 20)C2 + 2C3 + (2i + 2j + 4)C4 + 3C5$$
$$= \theta(7i + 8j) \approx \theta(i + j)$$

4.1.4. Kompleksitas Algoritma Fungsi Number-To-Binary

Tabel 4 Kompleksitas Algoritma Fungsi Number-To-Binary

Fungsi NumberToBinary(int i)	C	#	C#
string binaryIn64Bits = bitset<64>(i).to_string();	C1	64	64C1
int position = binaryIn64Bits.find("1");	C2	64	64C2
if (position != -1) {	C3	1	C3
return (binaryIn64Bits.substr(position)); }	C4	1	C3
return "0"; }	C4	1	C4

$$T(i) = 64C1 + 64C2 + C3 + C4 + C4$$
$$= \theta(64C1 + 64C2) \approx \theta(C1 + C2) \approx \theta(1)$$

4.1.5. Kompleksitas Algoritma Fungsi Square-and-Multiply

Tabel 5 Kompleksitas Algoritma Fungsi *Square-And-Multiply* untuk perhitungan modulo Eksponensial

Fungsi SAM(x, y, m){	C	#	C#
string binaryOfY = numberToBinary(y);	C1	1	.C1
long long z = 1;	C1	1	C1
for (int i = 0; i < binaryOfY.length(); i++) {	C2	N	N. C2
if (binaryOfY[i] == '1') {	C3	N	N . C3
z = (x * z * z) % m; }	C1	N	N . C1
else {	C4	N	N . C4

z = (z * z) % m;
}
}
	C1	N	N . C1

return z; }	C5	1	C5

$$T(x,y,m) = C1 + C1 + N\,.\,C2 + N\,.\,C3 + N\,.\,C1 + N\,.\,C4 + N\,.\,C1 + C5$$
$$= 2C1 + 2N\,.\,C1 + N\,.\,C2 + N\,.\,C3 + N\,.\,C4 + C5$$
$$= C1 + N(2C1 + C2 + C3 + C4) + C5$$
$$= \theta(N)$$, dengan N adalah panjang karakter *binaryOfY* (karakter y dalam bentuk biner).

4.1.6. Kompleksitas Algoritma Rabin-Miller

Tabel 6 Kompleksitas Algoritma Fungsi Rabin-Miller

Fungsi RabinMiller(n, d, r){	C	#	C#
int a = randomNumber(2, (n-2));	C1	1	C1
int x = SAM(a, d, n)	C1	N	N . C1
if (x == 1 || x == n-1) {	C2	1	C2
return true; }	C3	1	C3
for (int i = 0; i < r-1; i++) {	C4	r	r . C4
x = (x * x) % n;	C3	r	r . C3
if (x == n-1) {	C2	r	r . C2
return true; }	C3	1	C3
return false; }	C3	1	C3

$$T(n,d,r) = C1 + N\,.\,C1 + C2 + C3 + r\,.\,C4 + r\,.\,C3 + r\,.\,C2 + C3 + C3$$
$$= C1(1+N) + C2(1+r) + C3(3+r) + rC4$$
$$= \theta(N+r)$$, dengan N adalah panjang karakter d dalam bentuk biner.

4.1.7. Kompleksitas Algoritma Fungsi Pemeriksaan Bilangan Prima (Is-Prime-Number)

Tabel 7 Kompleksitas Algoritma Fungsi Pemeriksaan Bilangan Prima

Fungsi isPrimeNumber(i){	C	#	C#
int d, r;	C1	1	C1
if (i == 1) {	C2	1	C2
return false; }	C3	1	C3
else if (i == 2) {	C4	1	C4
return true; }	C3	1	C3
else if (i % 2 == 0) {	C4	1	C4
return false; }	C3	1	C3

d = i – 1;	C5	1	C5
r = 0;	C5	1	C5
while (d % 2 == 0) {	C6	log (i)	(log(i))C6
d /= 2	C5	log(i)	(log (i))C5
r += 1; }	C5	log (i)	(log(i))C5
for (int j = 0; j < i / 2; j++) {	C7	i	i . C7
if (!rabinMillerTest(i, d, r)) {	C2	$i(N+r)$	$i(N+r)C2$
return false; }	C3	$i(N+r)$	$i(N+r)C3$
return true; }	C3	i	i . C3

$$T(i) = C1 + C2 + C3 + C4 + C3 + C4 + C3 + C5 + C5 + \log(i)\, C6 + \log(i)\, C5 + \log(i)\, C5 + i\,.C7 + i(N+r)C2 + i(N+r)C3 + i\,.C3$$

$$= C1 + \big(1 + i(N+r)\big)C2 + (3 + i(N+r) + i)C3 + 2C4 + (2 + 2\log(i))C5 + \log(i)\, C6 + i.C7$$

$= \theta\,(3 + i(N+r) + i) \approx \theta\big(i(N+r)\big) \approx \theta(i^2)$, dengan N adalah panjang karakter d dalam bentuk biner.

4.1.8. Kompleksitas Algoritma Fungsi Pembangkit Bilangan Prima

Tabel 8 Kompleksitas Algoritma Fungsi Pembangkit Bilangan Prima

Fungsi primeNumber(a, b)	C	#	C#
int i;	C1	1	C1
do {	C2	i^2	i^2 . C2
i = randomNumber(a, b);	C3	i^2	i^2 . C3
} while (IsPrimeNumber(i) == 0);	C2	i^2	i^2 . C2
return i; }	C4	1	C4

$$T(a,b) = C1 + i^2C2 + i^2C3 + i^2C2 + C4$$

$$= C1 + 2i^2C2 + i^2C3 + C4$$

$$= \theta(i^2) \approx \theta(b^2)$$

4.1.9. Kompleksitas Algoritma Fungsi Key-K

Tabel 9 Kompleksitas Algoritma fungsi Key-K

Fungsi GenerateKeyK(int n){	C	#	C#
vector<int> K;	C1	1	C1
int a, b, upperC, inverseLowerC, c, d;	C1	1	C1
do {	C2	log (min(a,n))	log(min(a,n)) . C2

a = randomNumber(0, n);	C3	$\log(\min(a,n))$	$\log(min(a,n))$. C3
} while (GCD(a, n) != 1);	C2	$\log(\min(a,n))$	$\log(min(a,n))$ C4
do {	C2	$\log(\min(a,n))$	$\log(min(a,n))$. C3
b = randomNumber(0, n);	C3	$\log(\min(b,n))$	$\log(min(b,n))$. C2
} while (GCD(b, n) !=1 && (b != a));	C2	$\log(\min(b,n))$	$\log(min(b,n))$. C3
upperC = 1 - (a * a);	C3	1	C3
if ((abs(upperC) % b !=0 && upperC != 0)) {	C4	1	C4
inverseLowerC = inverseModulo(b, n);	C3	$(7b + 7n)$	$(7b + 7n)$. C3
c = ((inverseLowerC * upperC) % n + n) % n; }	C3	1	C3
else {	C5	1	C5
c = ((upperC / b) % n + n) & n; }	C3	1	C3
d = ((-a) % n + n) % n;	C3	1	C3
K.push_back(a);	C6	1	C6
K.push_back(b);	C6	1	C6
K.push_back(c);	C6	1	C6
K.push_back(d);	C6	1	C6
return K; }	C7	1	C7

$$T(n) = C1 + C1 + \log(\min(a,n)) \,.\, C2 + \log(\min(a,n)) \,.\, C3 + \log(min(a,n)) \,.\, C4 + \log(min(a,n)) \,.\, C3 + \log(min(a,n)) \,.\, C2 + \log(\min(a,n)) \,.\, C3 + C3 + C4 + (7b + 7n) \,.\, C3 + C3 + C5 + C3 + C3 + C6 + C6 + C6 + C6 + C7$$

$$= 2C1 + 2\log(\min(a,n) \,.\, C2 + 3\log(min(a,n)) \,.\, C3 + (7b + 7n) \,.\, C3 + 4C3 + \log(\min(a,n)) \,.\, C4 + C4 + C5 + 4C6 + C7$$

$$= 2C1 + \log(min(a,n))\,(2C2 + 3C3 + C4) + (7b + 7n + 4)C3 + C4 + C5 + 4C6 + C7$$

$$= \theta\big(\log(\min(a,n)) + (7b + 7n)\big)$$

4.1.10. Kompleksitas Algoritma Fungsi Key-E

Tabel 10 Kompleksitas Algoritma Fungsi Key-E

Fungsi GenerateKeyE(int n){	C	#	C#
int e;	C1	1	C1
do {	C2	$n + 1$	$(n + 1)$. C2
e = randomNumber(2, (totient-1));	C3	$n + 1$	$(n + 1)$. C3

} while (GCD(e, totient) != 1);	C2	$n+1$	$(n+1)$. C2
return e; }	C4	1	C4

$$T(n) = C1 + (n+1)C2 + (n+1)C3 + (n+1)C2 + C4$$
$$= C1 + 2(n+1)C2 + (n+1)C3 + C4$$
$$= \theta(2n+2)$$
$$= \theta(2n)$$

4.1.11. Kompleksitas Algoritma Fungsi Plaintext-To-ASCII

Tabel 11 Kompleksitas Algoritma Fungsi Plaintext-To-ASCII

Fungsi GeneratePlaintextToASCII(string s){	C	#	C#
vector<int> plaintextInAscii;	C1	1	C1
if (s.length() % 2 != 0) {	C2	1	C2
s.append(" "); }	C3	1	C3
for (int i = 0; i < s.length(); i++) {	C4	N	N . C4
int asciiValue = (int)(s[i])+100;	C5	N	N . C3
plaintextInAscii.push_back(asciiValue); }	C6	N	N . C6
return plaintextInAscii; }	C7	1	C7

$$T(s) = C1 + C2 + C3 + N\,.C4 + N\,.C3 + N\,.C6 + C7$$
$$= C1 + C2 + (1+N)\,.C3 + N\,.C4 + N\,.C6 + C7$$
$$= \theta(1+N) \approx \theta(N)$$, dengan N adalah panjang karakter s.

4.1.12. Kompleksitas Algoritma Fungsi Hill-Encryption

Tabel 12 Kompleksitas Algoritma Fungsi Hill-Encryption

Fungsi GenerateHillEncryption(k, p, n){	C	#	C#
int i = 0;	C1	1	C1
vector<int> ciphertext;	C2	1	C2
while (i < p.size() - 1) {	C3	N	N . C2
ciphertext.push_back(((k[0] * p[i]) + (k[1] * p[i+1])) % n);	C4	N	N . C4
ciphertext.push_back(((k[2] * p[i]) + (k[3] * p[i+1])) % n);	C4	N	N . C4
i+=2; }	C1	N	N . C1
return ciphertext;	C5	1	C5

$$T(k,p,n) = C1 + C2 + N\,.C2 + N\,.C4 + N\,.C1 + C5$$
$$= (1+N)C1 + (1+N)C2 + (N)C4 + C5$$

$= \theta(1 + N) \approx \theta(N)$, dengan N adalah panjang karakter p.

4.1.13. Kompleksitas Algoritma Fungsi RSA-Encryption

Tabel 13 Kompleksitas Algoritma Fungsi RSA-Encryption

Fungsi GenerateRSAEncryption(e, m, c){	C	#	C#
vector<int> rsaCiphertext;	C1	1	C1
for (int i = 0; i < c.size(); i++) {	C2	N	N . C2
rsaCiphertext.push_back((SAM(c[i], e, m))); }	C3	N	N . C3
return rsaCiphertext; }	C4	1	C4

$$T(e, m, c) = C1 + N \,.\, C2 + N \,.\, C3 + C4$$

$= \theta(N)$ dengan N adalah panjang karakter c.

4.1.14. Kompleksitas Algoritma Fungsi RSA-Decryption

Tabel 14 Kompleksitas Algoritma Fungsi RSA-Decryption

Fungsi GenerateRSADecryption(d, m, c){	C	#	C#
vector<int> decryptedRsa;	C1	1	C1
for (int i = 0; i < c.size(); i++) {	C2	N	N . C2
decryptedRsa.push_back((SAM(c[i], d, m))); }	C3	N	N . C3
return decryptedRsa; }	C4	1	C4

$$T(d, m, c) = C1 + N \,.\, C2 + N \,.\, C3 + C4$$

$= \theta(N)$ dengan N adalah panjang karakter c.

4.1.15. Kompleksitas Algoritma Fungsi Hill-Decryption

Tabel 15 Kompleksitas Algoritma Fungsi Hill-Decryption

Fungsi GenerateHillDecryption(k, c, n){	C	#	C#
int i = 0;	C1	1	C1
vector<int> ciphertextRSA;	C2	1	C2
while (i < c.size() - 1) {	C3	N	N . C2
ciphertextRSA.push_back(((k[0] * c[i]) + (k[1] * p[i+1])) % n);	C4	N	N . C4
ciphertextRSA.push_back(((k[2] * c[i]) + (k[3] * c[i+1])) % n);	C4	N	N . C4
i+=2; }	C1	N	N . C1
return ciphertext;	C5	1	C5

$$T(k, c, n) = C1 + C2 + N \,.\, C2 + N \,.\, C4 + N \,.\, C4 + N \,.\, C1 + C5$$

$$= (1 + N)C1 + (1 + N)C2 + 2N \,.\, C4 + C5$$

$= \theta(2N)$ dengan N merupakan panjang karakter c.

4.1.16. Kompleksitas Algoritma Fungsi Fermat-Prime-Factorization

Tabel 16 Kompleksitas Algoritma Fungsi Fermat-Prime-Factorization

Fungsi FermatPrimeFactorization(N){	C	#	C#
long long x = $sqrt(N) + 1$;	C1	1	C1
long long akarX = $sqrt(x^2 - N)$	C1	1	C1
while ($akarX^2 != x^2 - N$) {	C2	log ($sqrt(N)$)	$\log(sqrt(N))$. C2
x+= 1;	C3	log ($sqrt(N)$)	$\log(sqrt(N))$. C3
akarX = $sqrt(x^2 - N)$; }	C1	log ($sqrt(N)$)	$\log(sqrt(N))$. C1
long long p = $x - akarX$;	C3	1	C3
return p; }	C4	1	C4

$$T(N) = C1 + C1 + \log(sqrt(N)) . C2 + \log(sqrt(N)) . C3 + \log(sqrt(N)) . C1 + C3 + C4$$

$$= 2C1 + \log(sqrt(N)) C1 + \log(sqrt(N)) C2 + C3 + \log(sqrt(N)) C3 + C4$$

$$= (2 + \log(sqrt(N)))C1 + \log(sqrt(N)) C2 + (1 + \log(sqrt(N)))C3 + C4$$

$$= \theta(\log(100 . sqrt(N)))$$

$$= \theta(\log(sqrt(N)))$$

4.1.17. Kompleksitas Algoritma Fungsi Is-Vector-Equal

Tabel 17 Kompleksitas Algoritma Fungsi Is-Vector-Equal

Fungsi IsVectorEqual(p, q){	C	#	C#
if (p.size() == q.size()) {	C1	1	C1
if (equal(p.begin(), p.end(), q.begin())) {	C1	N	N . C1
return true; } }	C2	1	C2
return false; }	C2	1	C2

$$T(p) = C1 + N . C1 + C2 + C2$$

$$= (1 + N)C1 + C2$$

$= \theta(1 + N) \approx \theta(N)$ dengan N adalah panjang karakter p.

4.1.18. Kompleksitas Algoritma Fungsi ASCII-To-Character

Tabel 18 Kompleksitas algoritma funsgi ASCII-To-Character

Fungsi AsciiToCharacter(p){	C	#	C#
cout<<"Plaintext hasil kriptanalisis : [";	C1	1	C1
for (int i = 0; i < p.size(); i++) {	C2	N	N . C2
cout<<char(p[i]-100)<<" ";	C1	N	N . C1
}			
cout<<"]"<<endl;	C1	1	C1
}			

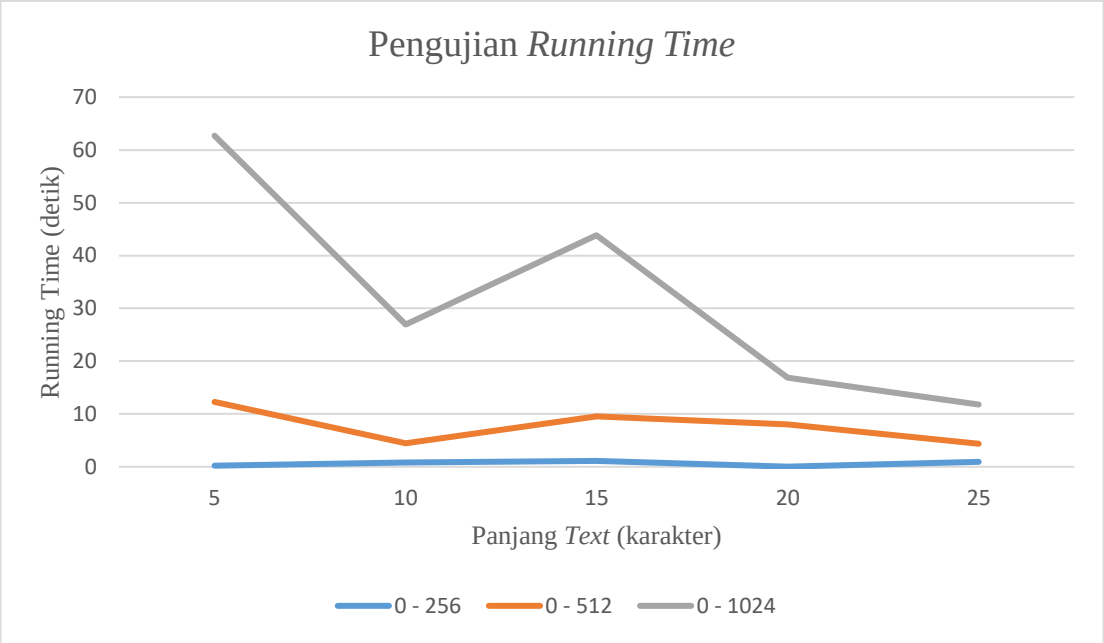
$$T(p) = C1 + N\,.C2 + N\,.C1 + C1$$
$$= (2 + N)C1 + N\,.C2$$
$$= \theta(2 + N) \approx \theta(N), \text{ dengan } N \text{ adalah panjang karakter } p.$$

4.2. Pengujian *Running Time* dalam Proses enkripsi dan kriptanalisis Algoritma Kriptografi Hibrida Hill Cipher-RSA

Pengujian *running time* pada tahapan enkripsi dan proses kriptanalisis algoritma Kriptografi Hibrida Hill Cipher-RSA bertujuan untuk menganalisis waktu proses yang diperlukan sistem untuk melakukan proses enkripsi dan proses kriptanalisis algoritma dalam satuan waktu terhadap berbagai rentang ukuran kunci privat, publik, dan kunci rahasia yang digunakan dengan menggunakan ukuran *plaintext* (p) yang berbeda.

Tabel 19 Hasil pengujian *running time* sistem terhadap perubahan ukuran kunci dan panjang *plaintext* (p)

Ukuran Kunci (bilangan)	Percobaan I (p = 5)	Percobaan II (p = 10)	Percobaan III (p = 15)	Percobaan IV (p = 20)	Percobaan V (p = 25)
0 – 256	0.218000 detik	0.824000 detik	1.112000 detik	0.026000 detik	0.963000 detik
0 – 512	12.06000 detik	3.648000 detik	8.411000 detik	7.996000 detik	3.402000 detik
0 – 1024	50.413000 detik	22.481000 detik	34.346000 detik	8.830000 detik	7.419000 detik



Gambar 19 Grafik pengujian *running time* terhadap terhadap perubahan panjang *plaintext* berdasarkan ukuran kunci

Pada Gambar 19, ditunjukan grafik pengukuran *running time* masing-masing ukuran kunci terhadap perubahan panjang *plaintext*. Berdasarkan hasil Gambar 19, dapat disimpulkan jika semakin besar ukuran kunci yang digunakan, maka semakin besar *running time* yang diperlukan untuk suatu sistem dappat menyelesaikan proses enkripsi dan kriptanalisis. Selain itu, ukuran atau panjang teks (*plaintext*) tidak mempengaruhi waktu *running time* dari sistem.

BAB V

KESIMPULAN DAN SARAN

5.1. Kesimpulan

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

1. Algoritma Modifikasi Hill Cipher-RSA yang menggunakan nilai modulo $n = 256$ masih lemah terhadap serangan *ciphertext-only* dan serangan *brute-force*.
2. Waktu dan jumlah iterasi yang diperlukan sistem dalam melakukan serangan *brute force* bergantung pada besar nilai modulus n algoritma Hill Cipher dan tidak bergantung pada panjang *plaintext* berdasarkan percobaan pada Bab 4.
3. Kompleksitas proses kriptanalisis algoritma Hill-Cipher dan RSA dari sistem yang dirancang adalah $\theta(n^4)$, dengan n merupakan nilai modulo Hill Cipher yang diuji.

5.2. Saran

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

1. Pemanfaatan metode serangan kriptografi lainnya, misalkan *Chosen-Ciphertext*, Kraitchik, dan lain sebagainya dapat diuji pada penelitian selanjutnya.
2. Pemanfaatan metode serangan *brute force* dapat dikembangkan menjadi lebih terstruktur untuk menghasilkan waktu dan jumlah percobaan yang lebih stabil.
3. Pengujian dengan nilai modulus n pada Hill Cipher yang lebih besar dapat dilakukan untuk memperoleh hasil yang lebih optimal.

DAFTAR PUSTAKA

Arif, Z. & Nurokhman, A., 2023. Analisis perbandingan algoritma kriptografi simetris dan asimetris dalam meningkatakan keamanan sistem informasi. *JTSI,* 4(2), pp. 394-405.

Bahiq, H. M., Bahiq, H. M. & Kotb, Y., 2020. Fermat Factorization using a Multi-Core System. *International Journal of Advanced Computer Science and Applications,* 11(4), pp. 323-330.

Chen, Y., Tang, C. & Ye, R., 2020. Cryptanalysis and improvement of medical mage encryption using high-speed scrambling and pixel adaptive diffusion. *Signal Processing,* Volume 167, pp. 1-12.

Giri, K. J., Parah, S. A., Bashir, R. & Muhammad, K., 2021. *Multimedia Security, Algorithm Development, Analysis and Applications.* Singapore: Springer Nature Singapore Pte Ltd. 2021.

Harahap, L., Panggabean, J. F. R. & Situmorang, S., 2021. Implementasi Metode Kriptografi Stream Cipher Gifford Untuk Enkripsi Intensitas Warna Piksel Pada Citra Digital Rahasia. *Jurnal Sains dan Teknologi ISTP,* 15(2), pp. 203-210.

Harahap, M. K., 2016. Membangkitkan bilangan prima marsenne dengan metode bilangan prima probabilistik Solovay - Strassen. *Publikasi Jurnal & Penelitian Teknik Informatika,* 1(1), pp. 1-4.

Hasoun, R. K., Khlebus, S. F. & Tayyeh, H. K., 2021. A new Approach of Classical Hill Cipher in Public Key Cryptography. *Int. J. Nonlinear Anal. Appl.,* 12(2), pp. 1071-1081.

Jamaludin, 2018. Rancang Bangun Kombinasi Hill Cipher dan RSA dengan Menggunakan Metode Hybrid Cryptosystem. *SinKron,* 2(2), pp. 86-93.

Klima, R. & Sigmon, N., 2019. *Cryptology Classical and Modern (Second Edition).* Boca Ranton: Taylor & Francis Group LLC.

Lestari, F. D., 2017. Analisa algoritma faktor persekutuan terbesar (FPB) menggunakan bahasa pemrograman C++. *Jurnal Evolusi,* 5(1), pp. 64-68.

Murdowo, S., 2019. Mengenal Kriptografi Modern Sederhana dengan Menggunakan Electronik Code Book (ECB). *INFOKAM,* 15(1), pp. 29-37.

Pangaribuan, L. J., 2018. Kriptografi hibrida algoritma hill cipher dan rivest shamir adleman (rsa) sebagai pengembangan kriptografi simetris (studi kasus: nilai mahasiswa amik mbp). *jurnal teknologi informasi dan komunikasi,* 7(1), pp. 11-26.

Plymen, R., 2020. *The Great Prime Number Race.* United States of America: The American Mathematical Society.

Purba, B., Gulo, F. A., Utami, N. I. & Sihotang, Y. A., 2020. *Pengamanan File Teks Menggunakan Algoritma RC4.* Medan, s.n., pp. 420-425.

Puspita, S., Noviani, E. & Prihandono, B., 2015. Metode Solovay-Starassen untuk Pengujian Bilangan Prima. *Buletin Ilmmiah Mat. Sat. dan Terapannya (Bimaster),* 04(1), pp. 85-94.

Putri, G. G. & Rahayani, R. D., 2018. Analisis Kriptografi Simetris AES dan Kriptografi Asimetris RSA pada Enkripsi Citra Digital. *Ethos: Jurnal Penelitian dan Pengabdian Masyarakat,* 6(2), pp. 197-207.

Rangasamy, B. P., 2019. Some Extentions on Numbers. *Advances in Pure Mathematics,* Volume 9, pp. 944-958.

Saputra, O., 2011. *Kompleksitas Algoritma Euclidean dan Stein (FPB),* Bandung: Makalah IF2091 Struktur Diskrit.

Seputra, K. A. & Saskara, G. A. J., 2020. Kriptografi simetris RC4 pada transaksi online booking engine system. *Jurnal pendidikan teknologi dan kejuruan,* 17(2), pp. 286-295.

Seputra, K. A. & Saskara, G. A. J., 2020. Kriptografi Simetris RC4 Pada Transaksi Online Booking Engine System. *Jurnal Pendidikan Teknologi dan Kejuruan,* 17(2), pp. 286-295.

Simarmata, J., S. & Rahim, R., 2019. *Kriptografi Teknik Keamanan Data dan Informasi.* Yogyakarta: ANDI.

Solin, R. & Ramadhani, P., 2020. Modifikasi Pembangkit Kunci Algoritma Elgamal denga Menerapkan Algoritma Freivalds. *KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer),* 4(1), pp. 351-356.

Sonea, A. & Cristea, I., 2023. Euler's Totient function applied to complete hypergroups. *AIMS Mathematics,* 8(4), pp. 7731-7746.

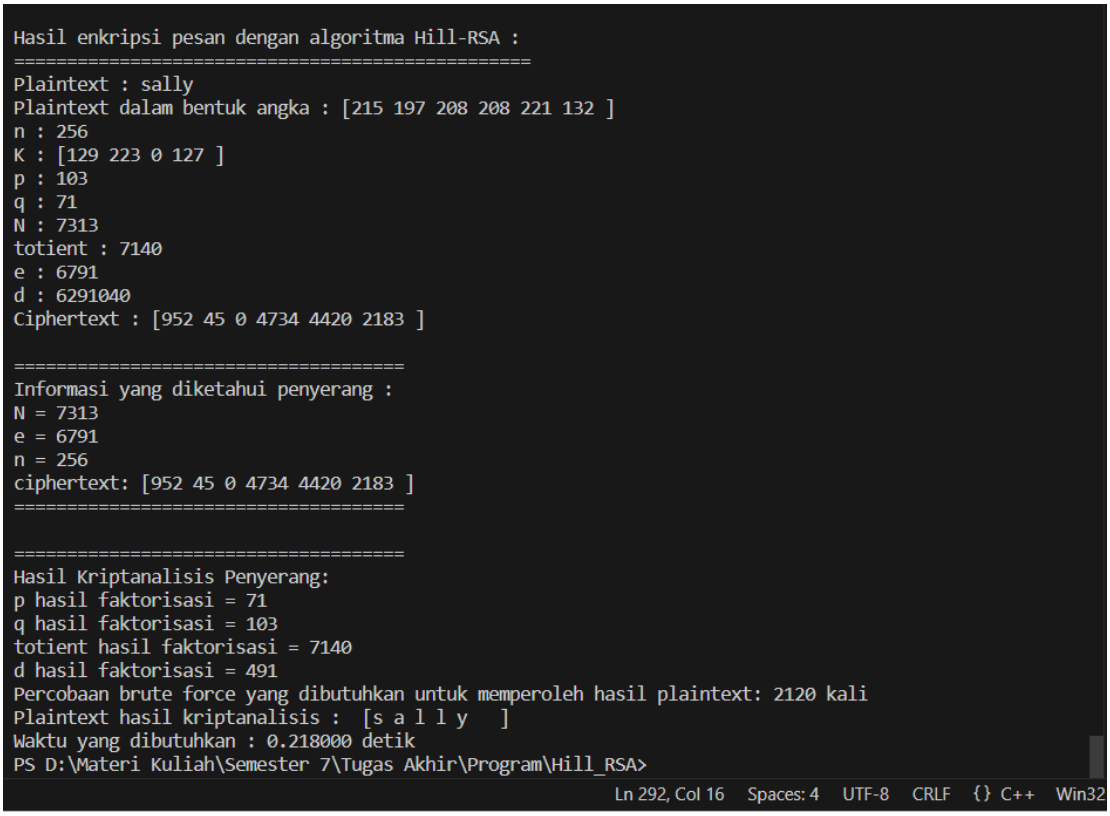
Thahara, A. & Siregar, I. T., 2021. Implementasi Kriptografi untuk Keamanan Data dan Jaringan menggunakan Algoritma DES. *JURTI,* 5(1), pp. 31-38.

Wang, X., 2021. The Genesis of Prime Number - Revealing the Underlying Periodicity of Prime Numbers. *Advances in Pure Mathematics,* Volume 11, pp. 12-18.

Yulianti, N. & Wijaya, A., 2022. Desain Kriptografi Klasik dengan Konsep Koset Grup dalam Teori Aljabar. *UNNES Journal of Mathematics,* 11(1), pp. 16-26.

LAMPIRAN

Lampiran 1. Hasil Percobaan $p = 5$ dengan Ukuran kunci 0-256



```
Hasil enkripsi pesan dengan algoritma Hill-RSA :
================================================
Plaintext : sally
Plaintext dalam bentuk angka : [215 197 208 208 221 132 ]
n : 256
K : [129 223 0 127 ]
p : 103
q : 71
N : 7313
totient : 7140
e : 6791
d : 6291040
Ciphertext : [952 45 0 4734 4420 2183 ]

====================================
Informasi yang diketahui penyerang :
N = 7313
e = 6791
n = 256
ciphertext: [952 45 0 4734 4420 2183 ]
====================================

====================================
Hasil Kriptanalisis Penyerang:
p hasil faktorisasi = 71
q hasil faktorisasi = 103
totient hasil faktorisasi = 7140
d hasil faktorisasi = 491
Percobaan brute force yang dibutuhkan untuk memperoleh hasil plaintext: 2120 kali
Plaintext hasil kriptanalisis :  [s a l l y   ]
Waktu yang dibutuhkan : 0.218000 detik
PS D:\Materi Kuliah\Semester 7\Tugas Akhir\Program\Hill_RSA>
```

Lampiran 2. Hasil Percobaan $p = 5$ dengan ukuran kunci 0 – 512

```
Hasil enkripsi pesan dengan algoritma Hill-RSA :
================================================
Plaintext : sally
Plaintext dalam bentuk angka : [215 197 208 208 221 132 ]
n : 512
K : [399 303 480 113 ]
p : 277
q : 29
N : 8033
totient : 7728
e : 43
d : 6291040
Ciphertext : [7849 1632 1720 1508 1596 2562 ]

====================================
Informasi yang diketahui penyerang :
N = 8033
e = 43
n = 512
ciphertext: [7849 1632 1720 1508 1596 2562 ]
====================================

====================================
Hasil Kriptanalisis Penyerang:
p hasil faktorisasi = 29
q hasil faktorisasi = 277
totient hasil faktorisasi = 7728
d hasil faktorisasi = 3235
Percobaan brute force yang dibutuhkan untuk memperoleh hasil plaintext: 134298 kali
Plaintext hasil kriptanalisis :  [s a l l y   ]
Waktu yang dibutuhkan : 12.060000 detik
PS D:\Materi Kuliah\Semester 7\Tugas Akhir\Program\Hill_RSA>
```

Lampiran 3. Hasil Percobaan $p = 5$ dengan ukuran kunci 0 – 1024

```
Hasil enkripsi pesan dengan algoritma Hill-RSA :
================================================
Plaintext : sally
Plaintext dalam bentuk angka : [215 197 208 208 221 132 ]
n : 1024
K : [695 187 144 329 ]
p : 179
q : 41
N : 7339
totient : 7120
e : 2419
d : 6291040
Ciphertext : [3542 6391 420 225 5127 4177 ]

====================================
Informasi yang diketahui penyerang :
N = 7339
e = 2419
n = 1024
ciphertext: [3542 6391 420 225 5127 4177 ]
====================================

====================================
Hasil Kriptanalisis Penyerang:
p hasil faktorisasi = 41
q hasil faktorisasi = 179
totient hasil faktorisasi = 7120
d hasil faktorisasi = 1819
Percobaan brute force yang dibutuhkan untuk memperoleh hasil plaintext: 607864 kali
Plaintext hasil kriptanalisis :  [s a l l y   ]
Waktu yang dibutuhkan : 50.413000 detik
PS D:\Materi Kuliah\Semester 7\Tugas Akhir\Program\Hill_RSA>
```

Lampiran 4. Hasil Percobaan $p = 10$ dengan ukuran kunci 0 – 256

```
Hasil enkripsi pesan dengan algoritma Hill-RSA :
================================================
Plaintext : sallylivia
Plaintext dalam bentuk angka : [215 197 208 208 221 208 205 218 205 197 ]
n : 256
K : [117 241 8 139 ]
p : 181
q : 223
N : 40363
totient : 39960
e : 8263
d : 6291040
Ciphertext : [38317 25969 224 21476 17035 12829 8635 39984 9922 25513 ]

====================================
Informasi yang diketahui penyerang :
N = 40363
e = 8263
n = 256
ciphertext: [38317 25969 224 21476 17035 12829 8635 39984 9922 25513 ]
====================================

====================================
Hasil Kriptanalisis Penyerang:
p hasil faktorisasi = 181
q hasil faktorisasi = 223
totient hasil faktorisasi = 39960
d hasil faktorisasi = 4807
Percobaan brute force yang dibutuhkan untuk memperoleh hasil plaintext: 9793 kali
Plaintext hasil kriptanalisis :  [s a l l y l i v i a ]
Waktu yang dibutuhkan : 0.824000 detik
PS D:\Materi Kuliah\Semester 7\Tugas Akhir\Program\Hill_RSA>
```

Ln 292, Col 16 Spaces: 4 UTF-8 CRLF {} C++ Win32

Lampiran 5. Hasil Percobaan $p = 10$ dengan ukuran kunci 0 – 512

```
Hasil enkripsi pesan dengan algoritma Hill-RSA :
================================================
Plaintext : sallylivia
Plaintext dalam bentuk angka : [215 197 208 208 221 208 205 218 205 197 ]
n : 512
K : [17 227 416 495 ]
p : 193
q : 101
N : 19493
totient : 19200
e : 13679
d : 6291040
Ciphertext : [12081 13721 17367 12348 3367 13151 7697 822 17920 17816 ]

====================================
Informasi yang diketahui penyerang :
N = 19493
e = 13679
n = 512
ciphertext: [12081 13721 17367 12348 3367 13151 7697 822 17920 17816 ]
====================================

====================================
Hasil Kriptanalisis Penyerang:
p hasil faktorisasi = 101
q hasil faktorisasi = 193
totient hasil faktorisasi = 19200
d hasil faktorisasi = 5519
Percobaan brute force yang dibutuhkan untuk memperoleh hasil plaintext: 30209 kali
Plaintext hasil kriptanalisis :  [s a l l y l i v i a ]
Waktu yang dibutuhkan : 2.604000 detik
PS D:\Materi Kuliah\Semester 7\Tugas Akhir\Program\Hill_RSA>
```

Ln 292, Col 16 Spaces: 4 UTF-8 CRLF {} C++ Win32

Lampiran 6. Hasil Percobaan $p = 10$ dengan ukuran kunci 0 – 1024

```
Hasil enkripsi pesan dengan algoritma Hill-RSA :
================================================
Plaintext : sallylivia
Plaintext dalam bentuk angka : [215 197 208 208 221 208 205 218 205 197 ]
n : 1024
K : [211 711 296 813 ]
p : 421
q : 41
N : 17261
totient : 16800
e : 15959
d : 6291040
Ciphertext : [5296 12923 5454 14913 1475 13665 16038 3542 5975 14701 ]

====================================
Informasi yang diketahui penyerang :
N = 17261
e = 15959
n = 1024
ciphertext: [5296 12923 5454 14913 1475 13665 16038 3542 5975 14701 ]
====================================

====================================
Hasil Kriptanalisis Penyerang:
p hasil faktorisasi = 41
q hasil faktorisasi = 421
totient hasil faktorisasi = 16800
d hasil faktorisasi = 839
Percobaan brute force yang dibutuhkan untuk memperoleh hasil plaintext: 262797 kali
Plaintext hasil kriptanalisis :  [s a l l y l i v i a ]
Waktu yang dibutuhkan : 22.481000 detik
PS D:\Materi Kuliah\Semester 7\Tugas Akhir\Program\Hill_RSA>
```

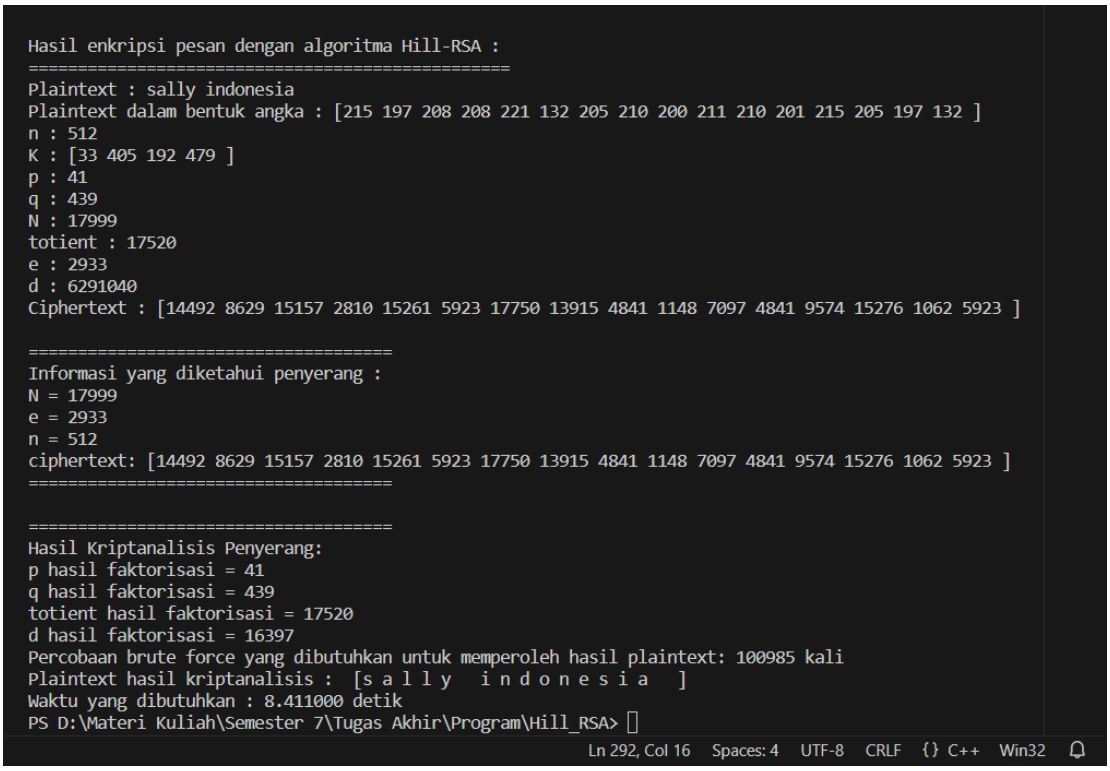
Ln 292, Col 17 Spaces: 4 UTF-8 CRLF {} C++ Win32

Lampiran 7. Hasil Percobaan $p = 15$ dengan ukuran kunci 0 – 256

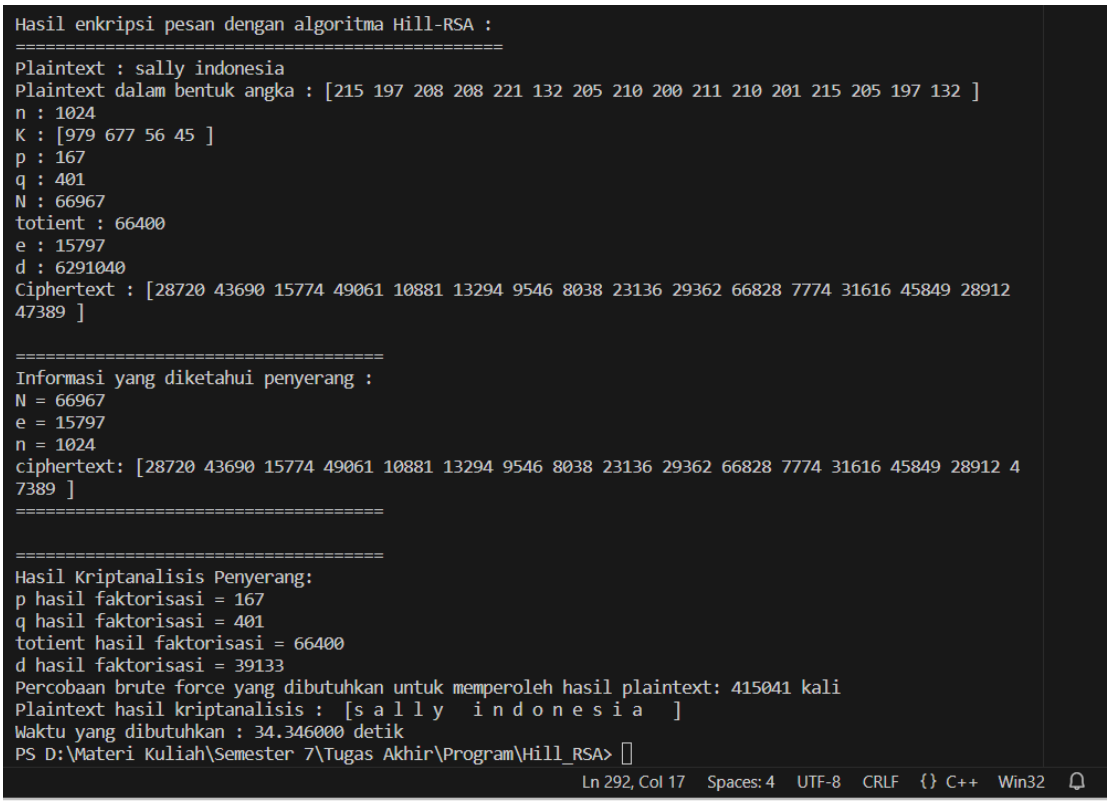
```
Hasil enkripsi pesan dengan algoritma Hill-RSA :
================================================
Plaintext : sally indonesia
Plaintext dalam bentuk angka : [215 197 208 208 221 132 205 210 200 211 210 201 215 205 197 132 ]
n : 256
K : [117 67 216 139 ]
p : 5
q : 233
N : 1165
totient : 928
e : 425
d : 6291040
Ciphertext : [105 585 1058 247 636 226 362 439 266 861 1009 733 234 208 1009 73 ]

====================================
Informasi yang diketahui penyerang :
N = 1165
e = 425
n = 256
ciphertext: [105 585 1058 247 636 226 362 439 266 861 1009 733 234 208 1009 73 ]
====================================

====================================
Hasil Kriptanalisis Penyerang:
p hasil faktorisasi = 5
q hasil faktorisasi = 233
totient hasil faktorisasi = 928
d hasil faktorisasi = 345
Percobaan brute force yang dibutuhkan untuk memperoleh hasil plaintext: 13335 kali
Plaintext hasil kriptanalisis :  [s a l l y   i n d o n e s i a  ]
Waktu yang dibutuhkan : 1.112000 detik
PS D:\Materi Kuliah\Semester 7\Tugas Akhir\Program\Hill_RSA>
```

Ln 306, Col 40 Spaces: 4 UTF-8 CRLF {} C++ Win32

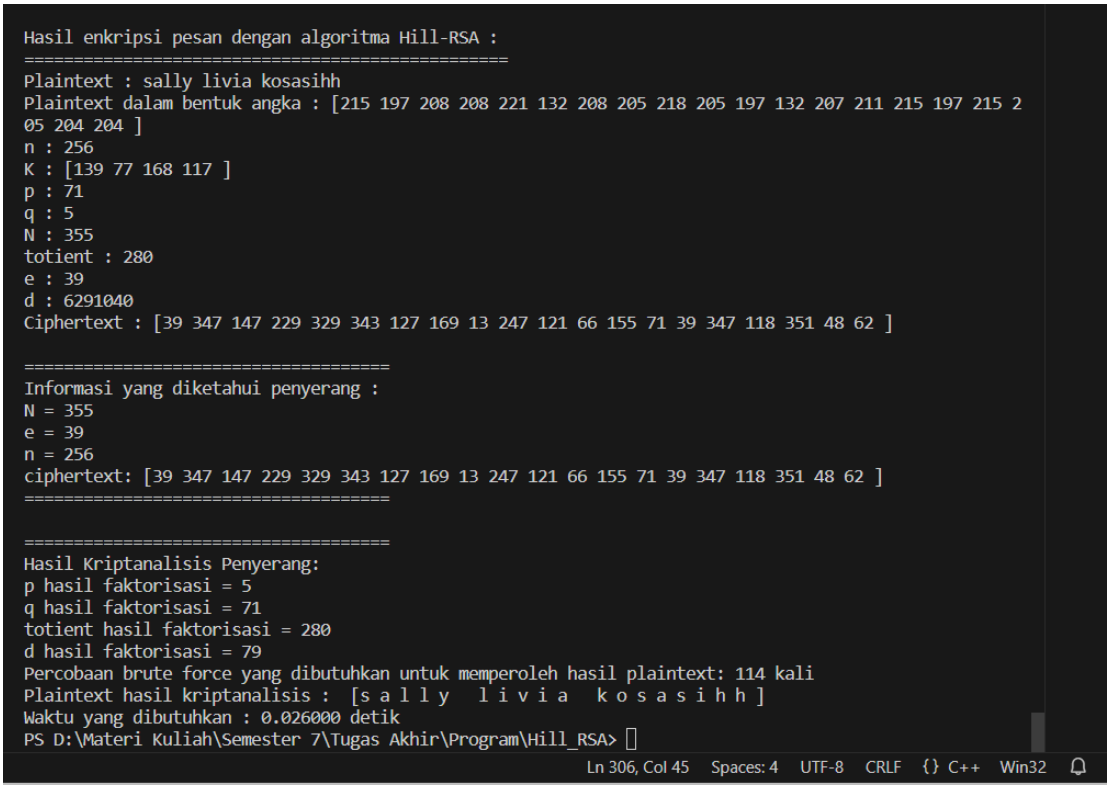
Lampiran 8. Hasil Percobaan $p = 15$ untuk ukuran kunci 0 – 512



Lampiran 9. Hasil Percobaan $p = 15$ untuk ukuran kunci 0 – 1024



Lampiran 10. Hasil Percobaan $p = 20$ untuk ukuran kunci 0 – 256



Lampiran 11. Hasil percobaan $p = 20$ untuk ukuran matriks 0 – 512

```
Hasil enkripsi pesan dengan algoritma Hill-RSA :
================================================
Plaintext : sally livia kosasihh
Plaintext dalam bentuk angka : [215 197 208 208 221 132 208 205 218 205 197 132 207 211 215 197 215 2
05 204 204 ]
n : 512
K : [275 59 72 237 ]
p : 3
q : 269
N : 807
totient : 536
e : 43
d : 6291040
Ciphertext : [638 451 385 437 434 638 160 514 458 677 671 508 785 789 638 451 625 515 409 15 ]

====================================
Informasi yang diketahui penyerang :
N = 807
e = 43
n = 512
ciphertext: [638 451 385 437 434 638 160 514 458 677 671 508 785 789 638 451 625 515 409 15 ]
====================================

====================================
Hasil Kriptanalisis Penyerang:
p hasil faktorisasi = 3
q hasil faktorisasi = 269
totient hasil faktorisasi = 536
d hasil faktorisasi = 187
Percobaan brute force yang dibutuhkan untuk memperoleh hasil plaintext: 86587 kali
Plaintext hasil kriptanalisis :  [s a l l y   l i v i a   k o s a s i h h ]
Waktu yang dibutuhkan : 7.996000 detik
PS D:\Materi Kuliah\Semester 7\Tugas Akhir\Program\Hill_RSA>
```

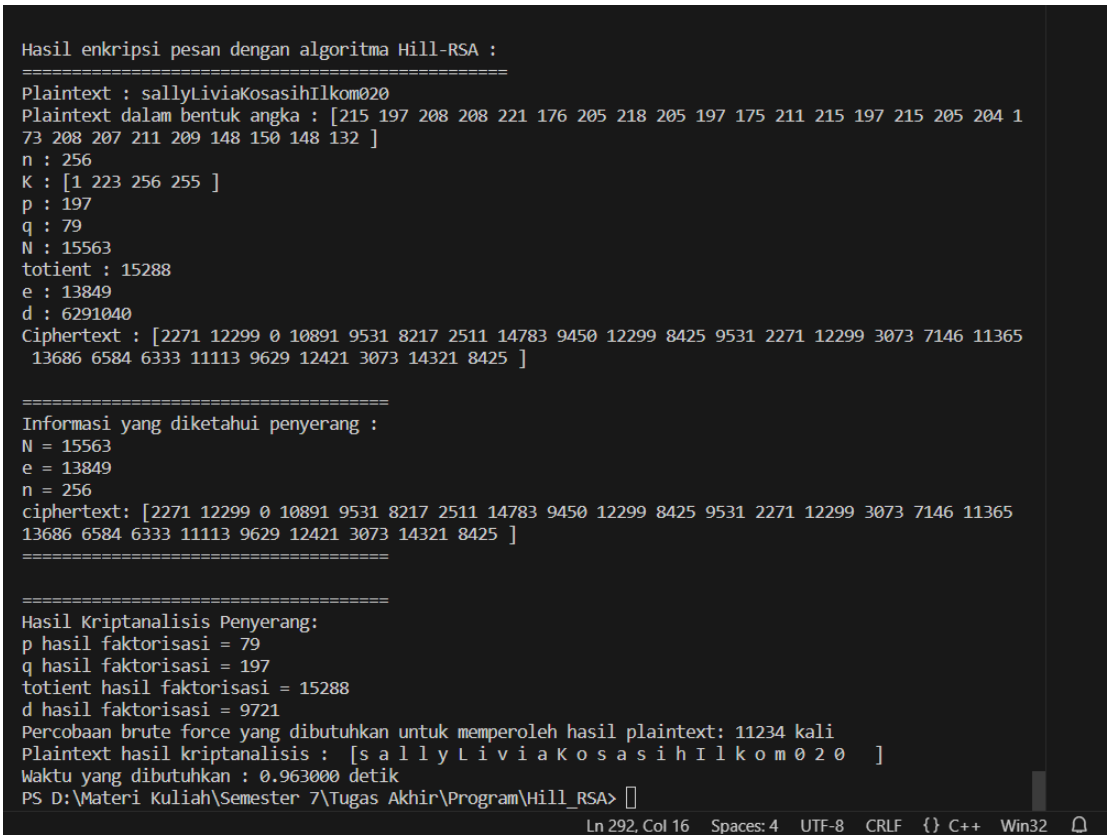
Ln 292, Col 16 Spaces: 4 UTF-8 CRLF {} C++ Win32

Lampiran 12. Hasil Percobaan $p = 20$ untuk ukuran kunci 0 – 1024

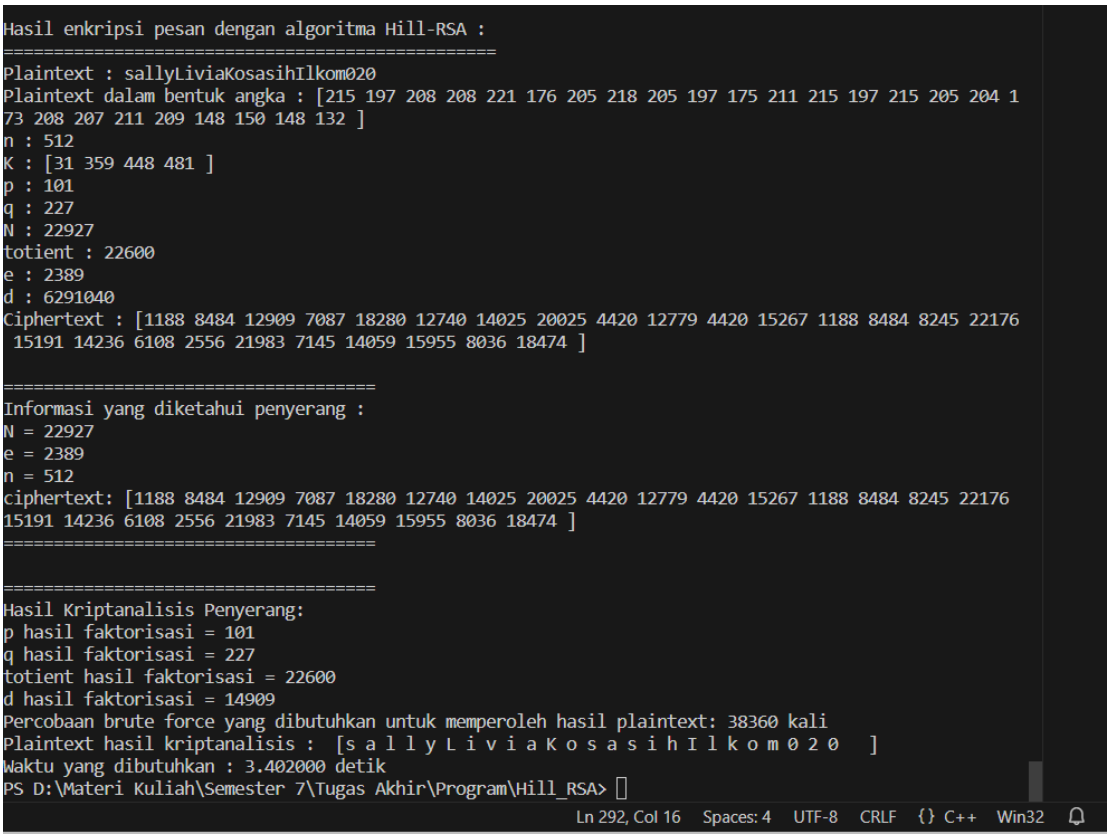
```
Hasil enkripsi pesan dengan algoritma Hill-RSA :
================================================
Plaintext : sally livia kosasihh
Plaintext dalam bentuk angka : [215 197 208 208 221 132 208 205 218 205 197 132 207 211 215 197 215 2
05 204 204 ]
n : 1024
K : [589 659 712 435 ]
p : 953
q : 283
N : 269699
totient : 268464
e : 20071
d : 6291040
Ciphertext : [22984 176374 203108 178464 219056 264402 68473 152139 245647 20457 10141 39913 88699 20
9352 22984 176374 9666 86364 169590 124716 ]

====================================
Informasi yang diketahui penyerang :
N = 269699
e = 20071
n = 1024
ciphertext: [22984 176374 203108 178464 219056 264402 68473 152139 245647 20457 10141 39913 88699 209
352 22984 176374 9666 86364 169590 124716 ]
====================================

====================================
Hasil Kriptanalisis Penyerang:
p hasil faktorisasi = 283
q hasil faktorisasi = 953
totient hasil faktorisasi = 268464
d hasil faktorisasi = 158743
Percobaan brute force yang dibutuhkan untuk memperoleh hasil plaintext: 100956 kali
Plaintext hasil kriptanalisis :  [s a l l y   l i v i a   k o s a s i h h ]
Waktu yang dibutuhkan : 8.830000 detik
PS D:\Materi Kuliah\Semester 7\Tugas Akhir\Program\Hill_RSA>
```

Ln 292, Col 17 Spaces: 4 UTF-8 CRLF {} C++ Win32

Lampiran 13. Hasil Percobaan $p = 25$ untuk ukuran kunci 0 – 256



Lampiran 14. Hasil Percobaan $p = 25$ untuk ukuran kunci 0 – 512



Lampiran 15. Hasil Percobaan $p = 25$ untuk ukuran kucni 0 – 1024

```
Hasil enkripsi pesan dengan algoritma Hill-RSA :
================================================
Plaintext : sallyLiviaKosasihIlkom020
Plaintext dalam bentuk angka : [215 197 208 208 221 176 205 218 205 197 175 211 215 197 215 205 204 1
73 208 207 211 209 148 150 148 132 ]
n : 1024
K : [327 171 752 697 ]
p : 359
q : 643
N : 230837
totient : 229836
e : 18301
d : 6291040
Ciphertext : [78327 85853 4755 180644 201192 155106 64611 15293 218354 165606 223787 100626 78327 858
53 207116 172963 164745 200812 73036 12967 104800 192252 145026 158176 203844 218091 ]

==================================
Informasi yang diketahui penyerang :
N = 230837
e = 18301
n = 1024
ciphertext: [78327 85853 4755 180644 201192 155106 64611 15293 218354 165606 223787 100626 78327 8585
3 207116 172963 164745 200812 73036 12967 104800 192252 145026 158176 203844 218091 ]
==================================

==================================
Hasil Kriptanalisis Penyerang:
p hasil faktorisasi = 359
q hasil faktorisasi = 643
totient hasil faktorisasi = 229836
d hasil faktorisasi = 58021
Percobaan brute force yang dibutuhkan untuk memperoleh hasil plaintext: 81551 kali
Plaintext hasil kriptanalisis :  [s a l l y L i v i a K o s a s i h I l k o m 0 2 0  ]
Waktu yang dibutuhkan : 7.419000 detik
PS D:\Materi Kuliah\Semester 7\Tugas Akhir\Program\Hill_RSA>
                                                Ln 292, Col 17  Spaces: 4  UTF-8  CRLF  {} C++  Win32
```