

**IMPLEMENTASI *HYBRID CRYPTOSYSTEM* MENGGUNAKAN
ALGORITMA *MULTIPLICATIVE SUBSTITUTION*
CRYPTOSYSTEM DAN ALGORITMA
*MULTI-FACTOR RSA***

SKRIPSI

FARIZA AL DAFFA

191401118



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

MEDAN

2023

**IMPLEMENTASI *HYBRID CRYPTOSYSTEM* MENGGUNAKAN
ALGORITMA *MULTIPLICATIVE SUBSTITUTION*
CRYPTOSYSTEM DAN ALGORITMA
*MULTI-FACTOR RSA***

SKRIPSI

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

FARIZA AL DAFFA

191401118



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

2023

UNIVERSITAS SUMATERA UTARA

PERSETUJUAN

Judul : IMPLEMENTASI *HYBRID CRYPTOSYSTEM*
MENGUNAKAN ALGORITMA *MULTIPLICATIVE*
SUBSTITUTION CRYPTOSYSTEM DAN
ALGORITMA *MULTI-FACTOR RSA*

Kategori : SKRIPSI

Nama : FARIZA AL DAFFA

Nomor Induk Mahasiswa : 191401118

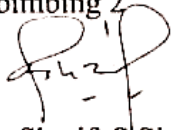
Program Studi : SARJANA (S-1) ILMU KOMPUTER

Fakultas : ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA

Telah diuji dan dinyatakan lulus di Medan, 27 Oktober 2023

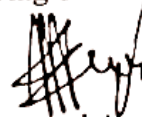
Komisi Pembimbing:

Pembimbing 2



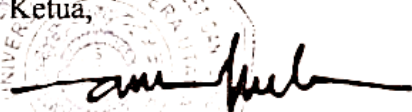
Amer Sharif, S.Si, M.Kom
NIP. 196910212021011001

Pembimbing 1



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

Diketahui/disetujui oleh
Program Studi S-1 Ilmu Komputer
Ketua,


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

PERNYATAAN

**IMPLEMENTASI *HYBRID CRYPTOSYSTEM* MENGGUNAKAN
ALGORITMA *MULTIPLICATIVE SUBSTITUTION*
CRYPTOSYSTEM DAN ALGORITMA
*MULTI-FACTOR RSA***

SKRIPSI

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

Medan, 27 Oktober 2023

Fariza Al Daffa

191401118

PENGHARGAAN

Segala puji dan syukur atas kehadiran Allah SWT yang telah memberikan rahmat dan hidayah-Nya, sehingga penulis dapat menyelesaikan penyusunan skripsi yang berjudul “Implementasi *Hybrid Cryptosystem* Menggunakan Algoritma *Multiplicative Substitution Cryptosystem* dan Algoritma *Multi-Factor RSA*” sebagai syarat memperoleh gelar Sarjana Komputer pada Program Studi S-1 Ilmu Komputer, Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.

Penulis mendoakan keberkahan kepada kedua orang tua tercinta Ibunda Mirla dan Ayahanda Enrico yang senantiasa memberikan do’a, dukungan secara moral dan material, serta kebahagiaan kepada penulis sehingga penulis dapat memperoleh pendidikan yang menjadi jalan untuk sampai kepada penyusunan skripsi ini. Dengan segala kerendahan hati, penulis ingin menyampaikan rasa hormat dan terima kasih yang sebesar-besarnya kepada semua pihak yang telah membantu dalam penyelesaian skripsi ini. Penulis mengucapkan terima kasih kepada :

1. Bapak Dr. Muriyanto Amin, S.Sos., M.Si selaku Rektor Universitas Sumatera Utara.
2. Ibu Dr. Maya Silvi Lydia, B.Sc., M.Sc. selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara.
3. Bapak Dr. Mohammad Andri Budiman, S.T., M.Comp.Sc., M.E.M., S.C.J.P. selaku Wakil Dekan 1 Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara dan selaku Dosen Pembimbing I yang telah memberikan bimbingan, saran, kritik serta motivasi kepada penulis dalam pengerjaan skripsi ini.
4. Ibu Dr. Amalia S.T., M.T. selaku Ketua Program Studi S-1 Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
5. Bapak Amer Sharif S.Si, M.Kom. selaku Dosen Pembimbing II yang juga telah memberikan bimbingan dan arahan yang luar biasa kepada penulis.

6. Bapak Prof. Dr. Syahril Efendi S.Si., M.IT. selaku Dosen Pembimbing Akademik yang selama ini telah memberikan arahan selama menempuh pendidikan di Program Studi S-1 Ilmu Komputer.
7. Seluruh dosen dan staf pegawai Program Studi S-1 Ilmu Komputer yang telah memberikan waktu dan tenaga untuk mengajar dan membimbing sehingga penulis dapat sampai kepada tahap penyusunan skripsi ini.
8. Anakin Al Yatsrib selaku Saudara penulis yang telah memberikan banyak doa, dukungan serta motivasi kepada penulis agar selalu semangat dalam menyelesaikan skripsi.
9. Teman-teman seperjuangan sekaligus sahabat penulis semasa kuliah yang telah banyak memberikan bantuan, dukungan, serta hiburan selama masa perkuliahan.
10. Dan semua pihak yang terlibat secara langsung maupun tidak langsung yang telah banyak membantu yang tidak dapat disebutkan satu per satu. Semoga segala kebaikan, dukungan serta motivasi yang telah diberikan kepada penulis mendapat berkah dari Allah SWT.

Medan, 27 Oktober 2023
Penulis,

Fariza Al Daffa

ABSTRAK

Masalah keamanan dan kerahasiaan informasi telah menjadi fokus utama dalam era digital saat ini, mengingat pertumbuhan pesat dalam pertukaran data dan informasi melalui jaringan komunikasi. Solusi untuk menyelesaikan masalah tersebut adalah dengan menerapkan kriptografi. Pada kriptografi modern biasanya dibedakan oleh dua algoritma, yaitu simetris dan asimetris. Namun kedua algoritma ini memiliki kelemahan dimana algoritma simetris sulit dalam manajemen keamanan dari kunci, sedangkan algoritma asimetris sulit untuk dikirim karena ukuran *ciphertext* yang dihasilkan lebih besar daripada ukuran *plaintext*. Untuk mengatasi masalah tersebut maka digunakan metode untuk mengamankan *file* dengan menggabungkan algoritma simetris dan algoritma asimetris yang biasa disebut *hybrid cryptosystem*. Dalam *hybrid cryptosystem*, *file* diamankan dengan menggunakan algoritma simetris dan kunci simetris diamankan dengan menggunakan algoritma asimetris. Pada penelitian ini, algoritma yang digunakan adalah *MSC* sebagai algoritma simetris dan *Multi-Factor RSA* sebagai algoritma asimetris. Hasil yang diperoleh dari penelitian ini menunjukkan bahwa penerapan metode *hybrid cryptosystem* menggunakan algoritma *MSC* dan algoritma *Multi-Factor RSA* untuk mengamankan *file* teks berhasil dilakukan. Hasil uji kecepatan proses algoritma memperlihatkan kenaikan waktu secara linear terhadap karakter yang dienkrpsi maupun didekripsi.

Kata kunci : Kriptografi, *Hybrid Cryptosystem*, *MSC*, *Multi-Factor RSA*

ABSTRACT

The issue of information security and confidentiality has become a major focus in the current digital era, considering the rapid growth in data and information exchange over communication networks. The solution to solve this problem is to apply cryptography. In modern cryptography, two algorithms are usually distinguished, namely symmetric and asymmetric. However, both of these algorithms have weaknesses in that the symmetric algorithm is difficult to manage the security of the key, while the asymmetric algorithm is difficult to send because the size of the resulting ciphertext is larger than the size of the plaintext. To overcome this problem, a method is used to secure files by combining symmetric algorithms and asymmetric algorithms which is usually called a hybrid cryptosystem. In a hybrid cryptosystem, files are secured using a symmetric algorithm and symmetric keys are secured using an asymmetric algorithm. In this research, the algorithms used are MSC as a symmetric algorithm and Multi-Factor RSA as an asymmetric algorithm. The results obtained from this research show that the application of the hybrid cryptosystem method using the MSC algorithm and the Multi-Factor RSA algorithm to secure text files was successful. The algorithm processing speed test results show a linear increase in time for encrypted and decrypted characters.

Keyword: Cryptography, Hybrid Cryptosystem, MSC, Multi-Factor RSA

DAFTAR ISI

PERSETUJUAN	i
PERNYATAAN	ii
PENGHARGAAN	iii
ABSTRAK.....	v
ABSTRACT.....	vi
DAFTAR ISI.....	vii
DAFTAR GAMBAR	x
DAFTAR TABEL.....	xii
BAB 1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Batasan Masalah.....	2
1.4 Tujuan Penelitian.....	3
1.5 Manfaat Penelitian.....	3
1.6 Metodologi Penelitian	3
1.7 Sistematika Penulisan.....	4
BAB 2 LANDASAN TEORI.....	5
2.1 Landasan Matematika Kriptografi.....	5
2.1.1 Bilangan Prima	5
2.1.2 Aritmatika Modulo.....	5
2.1.3 Euclidean Greatest Common Divisor (GCD)	5
2.1.4 Modulo Eksponensial (Square and Multiply).....	6
2.1.5 Inverse Modular	7
2.1.6 Fermat's Little Theorem	8
2.1.7 The Euler Totient Function	9
2.2 Kriptografi	9
2.2.1 Tujuan Kriptografi	10
2.2.2 Jenis-jenis Kriptografi.....	11
2.3 Hybrid Cryptosystem.....	13

2.4	Algoritma <i>Multiplicative Substitution Cryptosystem</i>	14
2.4.1	Proses Pembangkitan Kunci dengan Algoritma <i>MSC</i>	14
2.4.2	Proses Enkripsi Algoritma <i>MSC</i>	15
2.4.3	Proses Dekripsi Algoritma <i>MSC</i>	16
2.4.4	Proses Perhitungan Algoritma <i>MSC</i>	16
2.5	Algoritma <i>Multi-Factor RSA</i>	18
2.5.1	Proses Pembangkitan Kunci dengan Algoritma <i>Multi-Factor RSA</i>	18
2.5.2	Proses Enkripsi Algoritma <i>Multi-Factor RSA</i>	19
2.5.3	Proses Dekripsi Algoritma <i>Multi-Factor RSA</i>	19
2.5.4	Proses Perhitungan Algoritma <i>Multi-Factor RSA</i>	19
2.6	Algoritma <i>Brute Force</i>	20
2.7	Penelitian Relevan	21
BAB 3 ANALISIS DAN PERANCANGAN		23
3.1	Analisis Sistem	23
3.1.1	Analisis Masalah	23
3.1.2	Analisis Kebutuhan	24
3.1.3	Diagram Umum	25
3.2	Pemodelan Sistem	27
3.2.2	<i>Activity Diagram</i>	30
3.2.3	<i>Sequence Diagram</i>	34
3.3	<i>Flowchart</i>	35
3.3.1	<i>Flowchart sistem</i>	35
3.3.2	<i>Flowchart Algoritma Multi-Factor RSA</i>	36
3.3.3	<i>Flowchart Algoritma MSC</i>	37
3.3.4	<i>Flowchart Fermat's Little Theorem</i>	38
3.4	Perancangan <i>interface</i>	38
3.4.1	<i>Interface</i> Halaman Beranda	39
3.4.2	<i>Interface</i> Halaman Pengirim	40
3.4.3	<i>Interface</i> Halaman Penerima	41
3.4.4	<i>Interface</i> Halaman Bantuan	44
BAB 4 IMPLEMENTASI DAN PENGUJIAN		45

4.1	Implementasi Sistem	45
4.1.1	Halaman Beranda.....	45
4.1.2	Halaman Pembangkit Kunci	46
4.1.3	Halaman Enkripsi	46
4.1.4	Halaman Dekripsi	47
4.1.5	Halaman Bantuan	48
4.2	Pengujian Sistem	49
4.2.1	Pengujian Pembangkit Kunci <i>Multi-Factor RSA</i>	49
4.2.2	Pengujian Enkripsi	50
4.2.3	Pengujian Dekripsi	54
4.3	Waktu proses	59
4.3.1	Waktu Proses Enkripsi <i>File</i> Teks.....	59
4.3.2	Waktu Proses Dekripsi <i>File</i> Teks	61
4.3.3	Waktu Proses Enkripsi <i>File</i> Kunci	63
4.3.4	Waktu Proses Dekripsi <i>File</i> Kunci.....	64
4.4	Kompleksitas Algoritma	66
BAB 5 KESIMPULAN DAN SARAN		72
5.1	Kesimpulan.....	72
5.2	Saran.....	72
Daftar Pustaka		73

DAFTAR GAMBAR

Gambar 2.1 Proses Enkripsi dan Dekripsi	10
Gambar 2.2 Proses Enkripsi dan Dekripsi Kunci Simetris	11
Gambar 2.3 Proses Enkripsi dan Dekripsi Kunci Simetris	12
Gambar 3.1 Diagram Ishikawa Penelitian	24
Gambar 3.2 Diagram umum sistem	26
Gambar 3.3 <i>Use Case Diagram</i>	27
Gambar 3.4 <i>Activity Diagram</i> Pembangkit Kunci	31
Gambar 3.5 <i>Activity Diagram</i> Proses Enkripsi	32
Gambar 3.6 <i>Activity Diagram</i> Proses Dekripsi	33
Gambar 3.7 <i>Sequence Diagram</i> Sistem	34
Gambar 3.8 <i>Flowchart</i> sistem	35
Gambar 3.9 <i>Flowchart</i> Algoritma <i>Multi-Factor RSA</i>	36
Gambar 3.10 <i>Flowchart</i> Algoritma <i>MSC</i>	37
Gambar 3.11 <i>Flowchart</i> <i>Fermat's Little Theorem</i>	38
Gambar 3.12 Halaman Beranda	39
Gambar 3.13 Halaman Enkripsi	40
Gambar 3.14 Halaman Pembangkit Kunci	41
Gambar 3.15 Halaman Dekripsi	43
Gambar 3.16 Halaman Bantuan	44
Gambar 4.1 Halaman Beranda	45
Gambar 4.2 Halaman Pembangkit Kunci	46
Gambar 4.3 Halaman Enkripsi	47
Gambar 4.4 Halaman Dekripsi	48
Gambar 4.5 Halaman Bantuan	48
Gambar 4.6 Pengujian Pembangkit Kunci <i>Multi-Factor RSA</i>	49
Gambar 4.7 Pengujian enkripsi <i>plaintext</i>	51
Gambar 4.8 Pengujian enkripsi kunci	53
Gambar 4.9 Pengujian dekripsi kunci	55

Gambar 4.10 Pengujian dekripsi <i>ciphertext</i>	56
Gambar 4.11 Grafik Proses Enkripsi <i>MSC</i>	61
Gambar 4.12 Grafik Proses Dekripsi <i>MSC</i>	62
Gambar 4.13 Grafik Proses Enkripsi <i>Multi-Factor RSA</i>	64
Gambar 4.14 Grafik Proses Dekripsi <i>Multi-Factor RSA</i>	65
Gambar 4.15 Grafik Kriptanalisis <i>Brute Force</i> Pada <i>Multi-Factor RSA</i>	71

DAFTAR TABEL

Tabel 3.1 <i>Use Case Scenario</i> membangkitkan kunci <i>Multi-Factor RSA</i>	27
Tabel 3.2 <i>Use Case Scenario</i> membangkitkan kunci <i>MSC</i>	28
Tabel 3.3 <i>Use Case Scenario</i> mengenkripsi <i>file</i> teks.....	28
Tabel 3.4 <i>Use Case Scenario</i> mengenkripsi kunci <i>MSC</i>	29
Tabel 3.5 <i>Use Case Scenario</i> mendekripsi kunci <i>MSC</i>	29
Tabel 3.6 <i>Use Case Scenario</i> mendekripsi <i>ciphertext</i>	30
Tabel 4.1 Hasil Waktu Uji Coba Enkripsi <i>File</i> Teks	60
Tabel 4.2 Hasil Waktu Uji Coba Dekripsi <i>File</i> Teks.....	62
Tabel 4.3 Hasil Waktu Uji Coba Enkripsi <i>File</i> Kunci	63
Tabel 4.4 Hasil Waktu Uji Coba Dekripsi <i>File</i> Kunci	65
Tabel 4.5 Kompleksitas Algoritma Enkripsi <i>MSC</i>	66
Tabel 4.6 Kompleksitas Algoritma Dekripsi <i>MSC</i>	67
Tabel 4.7 Kompleksitas Algoritma Enkripsi <i>Multi-Factor RSA</i>	69
Tabel 4.8 Kompleksitas Algoritma Dekripsi <i>Multi-Factor RSA</i>	69
Tabel 4.9 Kriptanalisis <i>Brute Force</i>	70

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Pada era digital saat ini, komputer telah menjadi alat yang berguna dalam membantu dan meningkatkan efisiensi pekerjaan manusia, salah satunya dalam hal pertukaran data atau pesan. Pesan tersebut bisa saja memiliki informasi yang penting atau bersifat rahasia. Berdasarkan hal tersebut, tindakan pengamanan perlu dilakukan untuk memastikan kerahasiaan informasi terjaga dan tidak dapat disalahgunakan oleh pihak-pihak yang tidak berhak mengakses informasi tersebut. Maka dari itu, solusi yang bisa diterapkan dalam mengatasi permasalahan tersebut adalah dengan menerapkan konsep kriptografi.

Dalam kriptografi, terdapat beragam algoritma yang digunakan dalam proses enkripsi dan dekripsi data. Algoritma dalam kriptografi modern seringkali dikelompokkan menjadi dua jenis utama, yaitu algoritma asimetris dan algoritma simetris. Algoritma asimetris memanfaatkan sepasang kunci yang berbeda, yaitu kunci publik untuk enkripsi dan kunci privat untuk dekripsi. Sementara itu, algoritma simetris menggunakan kunci yang sama dalam proses enkripsi dan dekripsi.

Algoritma *MSC (Multiplicative Substitution Cryptosystem)* adalah algoritma simetris yang diperkenalkan oleh Vemulapalli Rajesh dan Panchami. Untuk mengenkripsi *plaintext* *MSC* menggunakan kunci yang dihasilkan dari fungsi *non-linear* pada kunci asli yang digunakan untuk memutihkan kunci. Analisis menunjukkan bahwa bila dibandingkan dengan *cipher* substitusi lain (*Caesar Cipher*, *Vigenere Cipher*, dan *Hill Cipher*) *MSC* menawarkan keamanan tinggi, kompleksitas waktu yang lebih sedikit, dan *throughput* yang tinggi (Rajesh, 2019).

Multi-Factor RSA adalah sebuah algoritma kriptografi asimetris yang merupakan hasil pengembangan dari algoritma *RSA* (Rivest-Shamir-Adleman). Keamanan dari algoritma *Multi-Factor RSA* bergantung pada tingkat kesulitan dalam melakukan faktorisasi bilangan bulat besar menjadi bilangan prima.

Hybrid Cryptosystem adalah teknik untuk meningkatkan keamanan data dengan mengombinasikan penggunaan algoritma simetris dan algoritma asimetris. Dalam *hybrid cryptosystem*, *file* diamankan dengan algoritma simetris dan kunci simetris diamankan dengan algoritma asimetris (Smart, 2016). Dengan menggunakan metode ini, pengirim memiliki *ciphertext* berukuran kecil karena *plaintext* diamankan oleh algoritma simetris. Kunci simetris juga lebih aman karena kunci dienkripsi dengan algoritma asimetris.

Berdasarkan latar belakang tersebut, maka akan dilakukan perancangan sebuah sistem pengamanan *file* teks menggunakan metode *hybrid cryptosystem* dengan menggabungkan dua algoritma yaitu, *MSC* dan *Multi-Factor RSA* untuk menjaga kerahasiaan dan keamanan data lebih optimal.

1.2 Rumusan Masalah

Rumusan masalah yang diajukan adalah algoritma *Multi-Factor RSA* merupakan algoritma asimetris yang ukuran *ciphertext* yang dihasilkan lebih besar daripada ukuran *plaintext* sehingga sulit untuk dikirim. Algoritma *MSC* adalah algoritma simetris yang menggunakan kunci yang sama untuk enkripsi dan dekripsi, maka untuk tiap pengiriman pesan dengan pengguna yang berbeda dibutuhkan kunci yang sama, sehingga akan terjadi kesulitan dalam manajemen keamanan dari kunci tersebut.

1.3 Batasan Masalah

Batasan masalah dalam penelitian ini adalah :

1. Metode kriptografi yang digunakan pada penelitian adalah *Hybrid Cryptosystem* dengan menggabungkan algoritma *MSC* sebagai algoritma simetris dan algoritma *Multi-Factor RSA* sebagai algoritma asimetris.
2. Proses pengamanan data dilakukan pada *file* teks.
3. C# adalah bahasa pemrograman yang digunakan pada penelitian.
4. Untuk uji keamanan kriptografi menggunakan Algoritma *Brute Force*.

1.4 Tujuan Penelitian

Adapun tujuan penelitian ini untuk mengimplementasikan kriptografi dalam membangun sistem keamanan pesan teks dengan menggabungkan algoritma *MSC* dan algoritma *Multi-Factor RSA*.

1.5 Manfaat Penelitian

Dengan dilakukannya penelitian ini diharapkan dapat memperoleh hasil sebuah sistem yang dapat mengamankan pesan *file* teks dengan tingkat keamanan yang tinggi.

1.6 Metodologi Penelitian

Penelitian ini menerapkan beberapa metode penelitian sebagai berikut :

1. Studi Kasus

Dalam tahap ini penulis akan melakukan pengumpulan referensi yang dibutuhkan terkait dengan algoritma *MSC* dan algoritma *Multi-Factor RSA*. Referensi yang diambil dalam bentuk jurnal, artikel, makalah skripsi dan e-book.

2. Analisis dan Perancangan

Berdasarkan ruang lingkup penelitian, penulis akan melakukan analisa terhadap hal-hal yang dibutuhkan dalam penelitian dan membuat rancangan sistem.

3. Implementasi

Pada tahap ini, akan dilakukan proses implementasi algoritma *MSC* dan algoritma *Multi-Factor RSA* ke dalam bahasa pemrograman C#.

4. Pengujian

Pada tahap ini, Di tahap pengujian akan dilakukan uji coba apakah keamanan data menggunakan implementasi kriptografi algoritma *MSC* dan algoritma *Multi-Factor RSA* sudah berjalan sesuai dengan kebutuhan.

5. Dokumentasi

Pada tahap ini, akan dilakukan proses dokumentasi mulai dari tahap analisa hingga tahap pengujian yang dibentuk dalam laporan penelitian(skripsi).

1.7 Sistematika Penulisan

Sistematika dalam penelitian disusun dalam format skripsi dan terbagi ke dalam beberapa bagian berikut:

BAB I PENDAHULUAN

Rangkuman mengenai latar belakang penelitian melibatkan rumusan masalah, tujuan penelitian, batasan masalah, manfaat penelitian, metode penelitian, dan sistematika penulisan.

BAB II LANDASAN TEORI

Berisi penjelasan mengenai penjelasan teori tentang kriptografi, Algoritma simetris dan asimetris, *Hybrid Cryptosystem*, dasar matematika kriptografi algoritma *MSC*, algoritma *Multi-Factor RSA*.

BAB III ANALISIS DAN PERANCANGAN

Membahas analisis dan perancangan terkait dengan masalah penelitian. Ini mencakup analisis kebutuhan yang diperlukan dalam pengembangan sistem serta proses perancangan sistem yang akan dibangun.

BAB IV IMPLEMENTASI DAN PENGUJIAN

Membahas penerapan algoritma *MSC*, dan algoritma *Multi-Factor RSA* pada aplikasi pengamanan pesan dan pembahasan mengenai hasil pengujian sistem yang telah dibangun serta analisis dari hasil-hasil tersebut.

BAB V KESIMPULAN DAN SARAN

Kesimpulan dari hasil penelitian yang telah dilakukan dan saran yang diharapkan dapat berguna untuk pengembangan selanjutnya.

BAB 2

LANDASAN TEORI

2.1 Landasan Matematika Kriptografi

Berikut ini merupakan beberapa prinsip matematika kriptografi yang digunakan untuk memahami dan mengerjakan proses algoritma *MSC* dan Algoritma *Multi-Factor RSA*.

2.1.1 Bilangan Prima

Bilangan prima merupakan bilangan bulat positif yang nilainya lebih besar dari 1 dan hanya mempunyai dua pembagi, yaitu 1 dan bilangan itu sendiri (Mollin, 2007). Sebagai contoh, 11 dapat dianggap sebagai bilangan prima karena hanya dapat dibagi habis oleh 11 dan 1. Sebagian besar bilangan prima adalah bilangan ganjil dan angka 2 adalah satu-satunya bilangan prima yang merupakan bilangan genap.

2.1.2 Aritmatika Modulo

Aritmatika modular adalah operasi matematika yang sering digunakan pada metode kriptografi (Christnatis, 2016). Aritmatika modulo adalah hasil sisa dari pembagian suatu bilangan dengan bilangan lainnya. Misalnya, jika a adalah bilangan bulat dan m adalah bilangan bulat positif. Maka $a \bmod m$ (dibaca “ a modulo m ”) akan memberikan sisa hasil bagi ketika a dibagi dengan m . Dengan kata lain, dapat dinyatakan bahwa $a \bmod m = r$ di mana $a = mq + r$, dengan $0 \leq r < m$

Contoh :

28 dibagi 5 memberikan hasil 5 dengan sisa 3, sehingga dapat ditulis:

$$28 \bmod 5 = 3$$

$$(28 = 5 \times 5 + 3)$$

2.1.3 Euclidean Greatest Common Divisor (GCD)

Apabila terdapat a dan b bilangan bulat tak nol, maka pembagi bersama terbesar (*greatest common divisor*) dari a dan b adalah bilangan bulat terbesar d memenuhi $d|a$ dan $d|b$ (Ariyus, 2006). Dalam hal ini dinyatakan bahwa $\gcd(a, b) = d$.

Untuk kasus dimana $\gcd(a, b) = 1$, dikatakan bahwa a relatif prima terhadap b dan begitu juga sebaliknya. *Euclidean GCD* merupakan suatu metode mencari *GCD* dengan algoritma *Euclidean* yang efisien untuk menghitung pembagi umum terbesar dari bilangan bulat positif tanpa meninggalkan sisa. Konsep dari algoritma ini adalah menghitung bilangan yang lebih besar (m) dibagi dengan bilangan yang lebih kecil (n). Langkah-langkah perhitungan *GCD* menggunakan algoritma *Euclidean* :

1. Jika $n = 0$ maka m adalah $\gcd(m, n)$. Tetapi jika $n \neq 0$ lanjutkan ke langkah 2.
2. Bagilah m dengan n dan misalkan r adalah sisanya.
3. Ganti nilai m dengan nilai n dan nilai n dengan nilai r , lalu ulang kembali ke langkah 1.

Contoh :

$\gcd(28, 5)$

$$28 \bmod 5 = 3$$

$$5 \bmod 3 = 2$$

$$3 \bmod 2 = 1$$

$$2 \bmod 1 = 0$$

Karena hasil adalah 0 maka $\gcd(28, 5)$ adalah bilangan sebelumnya yaitu 1.

Dan karena hasil 1 maka 28 relatif prima dengan 5.

2.1.4 Modulo Eksponensial (*Square and Multiply*)

Dalam kriptografi operasi modulo yang sering dijumpai adalah eksponensial. Eksponensial modulo adalah operasi dalam persamaan $x^y \bmod n$. Salah satu metode untuk menyelesaikan modulo eksponensial adalah *Square and Multiply*. Langkah-langkah *Square and Multiply* adalah sebagai berikut :

1. Inisialisasi nilai x , y dan n
2. Ubah y menjadi biner
3. Dari kiri ke kanan, lakukan iterasi di mana $z = 1$
4. Lakukan iterasi z dengan ketentuan
 - Jika bit = 0, maka nilai $z = z^2 \bmod n$
 - Jika bit = 1, maka nilai $z = x \cdot z^2 \bmod n$
5. Nilai z terakhir adalah nilai dari $x^y \bmod n$.

Contoh :

$$4^{35} \bmod 31$$

$$35 = 100011$$

$$4^{35} \bmod 31 = 4^{100011} \bmod 31$$

$$z = 4^1 \bmod 31 = 4 \cdot 1 \cdot 1 \bmod 31 = 4$$

$$z = 4^{10} \bmod 31 = 4 \cdot 4 \bmod 31 = 16$$

$$z = 4^{100} \bmod 31 = 16 \cdot 16 \bmod 31 = 8$$

$$z = 4^{1000} \bmod 31 = 8 \cdot 8 \bmod 31 = 2$$

$$z = 4^{10001} \bmod 31 = 4 \cdot 2 \cdot 2 \bmod 31 = 16$$

$$z = 4^{100011} \bmod 31 = 4 \cdot 16 \cdot 16 \bmod 31 = 1$$

$$4^{35} \bmod 31 = 1$$

2.1.5 Inverse Modular

Dalam *invers modulo*, *inverse* dari bilangan bulat a dalam modulo m adalah bilangan bulat x sedemikian rupa bahwa $ax \equiv 1 \pmod{m}$. Jika x adalah *invers* terkecil yang positif, maka kita menyebutnya *invers* terkecil dari bilangan bulat a dalam modulo m , dilambangkan dengan $x = a^{-1} \pmod{m}$ (Mollin, 2007).

Misalnya, kita ingin mencari *invers modulo* dari bilangan a dalam modulo m . *Invers modulo*, dilambangkan sebagai $a^{-1} \pmod{m}$, dimana bilangan bulat x yang memenuhi persamaan $xa \equiv 1 \pmod{m}$. Syarat-syarat yang harus terpenuhi untuk mendapatkan *invers modulo* adalah:

- Bilangan a dan m harus relatif prima, artinya bilangan tersebut tidak memiliki faktor persekutuan selain 1. Jika a dan m bukan relatif prima, maka *invers modulo* tidak akan ada.
- Modulo m harus lebih besar dari 1, yaitu $m > 1$. Jika $m = 1$, *invers modulo* tidak ada.

Untuk membuktikan keberadaan *invers modulo*, cari bilangan bulat x dan y sehingga $xa + ym = 1$. Karena $ym \equiv 0 \pmod{m}$, maka dapat disimpulkan bahwa $xa \equiv 1 \pmod{m}$. Jadi, dengan memenuhi syarat-syarat tersebut, dapat ditemukan *invers modulo* dari bilangan a dalam modulo m sebagai bilangan bulat x yang memenuhi persamaan $xa \equiv 1 \pmod{m}$.

Contoh :

Carilah nilai y yang memenuhi persamaan $5 * y \equiv 1 \pmod{11}$.

Dalam kasus ini, dilakukan percobaan nilai y secara berurutan, yaitu $y = 1, 2, 3, \dots, 11$, dan melihat jika $5 * y$ memiliki sisa pembagian 1 ketika dibagi dengan 11.

1. $y = 1; 5 * 1 \equiv 5 \pmod{11}$ (Tidak memenuhi persamaan)
2. $y = 2; 5 * 2 \equiv 10 \pmod{11}$ (Tidak memenuhi persamaan)
3. $y = 3; 5 * 3 \equiv 4 \pmod{11}$ (Tidak memenuhi persamaan)
4. $y = 4; 5 * 4 \equiv 9 \pmod{11}$ (Tidak memenuhi persamaan)
5. $y = 5; 5 * 5 \equiv 3 \pmod{11}$ (Tidak memenuhi persamaan)
6. $y = 6; 5 * 6 \equiv 8 \pmod{11}$ (Tidak memenuhi persamaan)
7. $y = 7; 5 * 7 \equiv 2 \pmod{11}$ (Tidak memenuhi persamaan)
8. $y = 8; 5 * 8 \equiv 7 \pmod{11}$ (Tidak memenuhi persamaan)
9. $y = 9; 5 * 9 \equiv 1 \pmod{11}$ (Memenuhi persamaan)

Ketika mencoba $y = 9$, $5 * 9$ memiliki sisa pembagian 1 ketika dibagi dengan 11. Jadi, *invers modulo* dari bilangan 5 dalam modulo 11 adalah 9.

2.1.6 Fermat's Little Theorem

Fermat's Little Theorem menyatakan bahwa jika p adalah bilangan prima dan a adalah bilangan bulat yang tidak habis dibagi oleh p , maka $a^{(p-1)} \equiv 1 \pmod{p}$ (Ariyus, 2006). Bila $a^{(p-1)} \pmod{p} = 1$ dimana $1 < a < p$ maka p adalah prima.

Contoh :

Apakah $p = 5$ adalah bilangan prima?

1. Ambil nilai a secara acak dengan syarat $1 < a < p$, $a = 2$
2. Hitung $2^{(5-1)} \pmod{5}$

$$\begin{aligned} 2^{(5-1)} \pmod{5} &= 2^4 \pmod{5} \\ &= 16 \pmod{5} \\ &= 1 \end{aligned}$$

Karena hasilnya adalah 1, maka dapat disimpulkan bahwa $p = 5$ adalah bilangan prima.

2.1.7 The Euler Totient Function

Fungsi *Euler Totient* atau *Euler Phi* diperkenalkan oleh Leonhard Euler. Untuk $n \geq 1$ fungsi *Phi* $\varphi(n)$ didefinisikan sebagai banyaknya bilangan bulat positif yang tidak lebih dari n dan relatif prima terhadap n (Ariyus, 2006).

Contoh :

$$\varphi(22) = 10$$

Hasil ini didapatkan dari bilangan bulat positif yang relatif prima dan tidak lebih dari 22 berjumlah sepuluh, hasilnya adalah 1, 3, 5, 7, 9, 13, 15, 17, 19, 21.

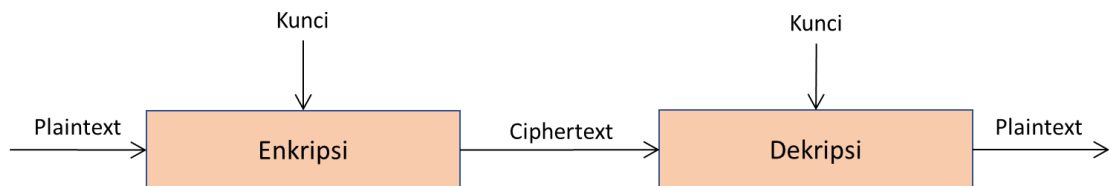
Jika n prima maka setiap bilangan bulat positif kurang dari n selalu relative prima terhadap n , sehingga $\varphi(n) = n - 1$. Jika n komposit (bukan prima), maka terdapat $1 \leq d \leq n$ sehingga $\gcd(d, n) \neq 1$. Dengan demikian, sedikitnya terdapat dua bilangan bulat positif diantara $1, 2, \dots, n$ yang tidak relatif prima terhadap n , yaitu d dan n . Oleh sebab itu didapat $\varphi(n) = n - 1$ jika dan hanya jika n prima.

2.2 Kriptografi

Kriptografi menurut Munir dapat didefinisikan sebagai seni dalam menyembunyikan pesan dari pihak yang tidak berhak mengakses pesan tersebut. Kata "kriptografi" berasal dari bahasa Yunani kuno, yaitu "*cryptós*" yang berarti "rahasia," dan "*gráphein*" yang berarti "tulisan," sehingga secara harfiah dapat diartikan sebagai "tulisan rahasia." Dalam konteks ini, kriptografi melibatkan konsep dan teknik untuk melindungi pesan dari akses yang tidak sah. Ariyus mengidentifikasi bahwa kriptografi memiliki tiga fungsi dasar (Ariyus, 2006), yaitu:

1. Enkripsi adalah sebuah proses di mana pesan asli, yang disebut *plaintext*, diubah menjadi kode-kode yang tidak dapat dipahami, yang sering disebut *ciphertext*. Untuk melakukan transformasi *plaintext* menjadi kode, diperlukan penggunaan algoritma kriptografi yang dapat mengkodekan data tersebut.

2. Dekripsi, merupakan kebalikan dari enkripsi, yaitu proses di mana pesan yang telah diubah menjadi *ciphertext* melalui enkripsi akan dikembalikan ke bentuk semula (*plaintext*),
3. Kunci, atau *key* berfungsi untuk melakukan enkripsi dan dekripsi. Kunci terbagi menjadi dua bagian, yaitu kunci umum (*public key*) dan kunci rahasia (*private key*).



Gambar 2.1 Proses Enkripsi dan Dekripsi

2.2.1 Tujuan Kriptografi

Tujuan dari kriptografi adalah mengirim pesan agar tidak diketahui oleh pihak musuh (Munir, 2006). Kriptografi memiliki tujuan untuk menyediakan layanan keamanan yang dapat dikenal dengan berbagai aspek keamanan berikut.

1. Kerahasiaan (*Confidentiality*)

Kriptografi dapat merahasiakan pesan dengan cara mengubah bentuk pesan asli (dapat dibaca dan dipahami) menjadi pesan terenkripsi (isi pesan tidak dapat dipahami).

2. Integritas (*Integrity*)

Kriptografi dapat menjamin bahwa pesan masih asli dan utuh setelah melewati proses enkripsi dan dekripsi.

3. Otentikasi (*Authentication*)

Kriptografi dapat digunakan untuk mengotentikasi identitas pihak yang berkomunikasi dan juga untuk mengotentikasi sumber pesan. Dalam proses kriptografi, hanya pihak-pihak yang sah yang terlibat dalam komunikasi. Otentikasi dalam aspek keamanan informasi dapat didukung melalui mekanisme tanda tangan digital.

4. *Non-repudiation*

Kriptografi dapat mencegah entitas yang berkomunikasi untuk melakukan penyangkalan, yang berarti pengirim pesan tidak dapat menyangkal bahwa mereka telah mengirim pesan, dan penerima pesan tidak dapat menyangkal

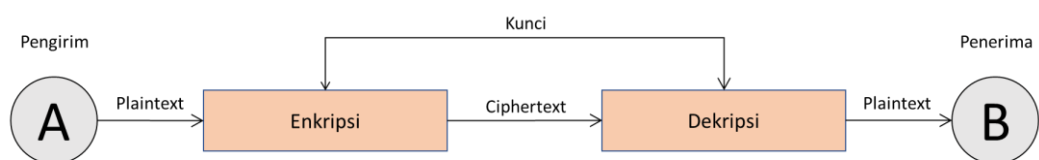
bahwa mereka telah menerima pesan. Aspek *Non-repudiation* dapat didukung melalui kriptografi dengan menggunakan algoritma tanda tangan digital (Agustina, 2009).

2.2.2 Jenis-jenis Kriptografi

Berdasarkan jenis kunci, algoritma kriptografi dikelompokkan menjadi dua bagian, yaitu:

1. Kriptografi Simetris

Algoritma kriptografi simetris adalah jenis algoritma kriptografi yang menggunakan kunci yang sama untuk enkripsi dan dekripsi pesan. Ini adalah salah satu jenis kriptografi yang paling umum digunakan. Dalam kriptografi simetris, kunci yang digunakan untuk mengamankan pesan adalah kunci yang sama dengan yang digunakan untuk membuka atau mendekripsi pesan tersebut (Basri, 2016). Terkadang istilah lain yang digunakan untuk menggambarkan kriptografi kunci simetris adalah kriptografi kunci privat. Dalam sistem kriptografi kunci simetris, pengirim dan penerima pesan diasumsikan telah memiliki kunci yang sama sebelum mereka dapat bertukar pesan. Keamanan dari sistem kriptografi simetris tergantung pada kerahasiaan kunci yang digunakan.



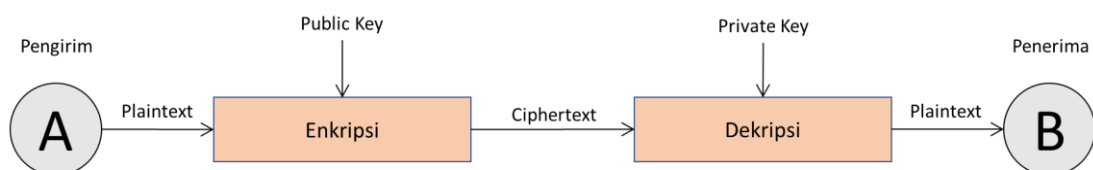
Gambar 2.2 Proses Enkripsi dan Dekripsi Kunci Simetris

Kriptografi tipe simetris memiliki keunggulan dalam hal kecepatan enkripsi dan dekripsi yang lebih tinggi dibandingkan dengan algoritma asimetris. Di sisi lain, kelemahannya terletak pada masalah distribusi kunci. Proses enkripsi dan dekripsi dalam kriptografi simetris menggunakan kunci yang sama, sehingga muncul isu dalam menjaga kerahasiaan kunci tersebut. Misalnya, ketika kunci perlu dikirim melalui media yang tidak aman seperti internet. Apabila kunci tersebut hilang atau diakses oleh pihak yang tidak berhak, maka keamanan sistem kriptografi tersebut dapat terancam.

2. Kriptografi Asimetris

Berbeda dengan kriptografi simetris yang menggunakan satu kunci untuk proses enkripsi dan dekripsi, kriptografi asimetris memanfaatkan dua kunci yang berbeda untuk melaksanakan enkripsi dan dekripsi. Kunci yang digunakan dalam proses enkripsi disebut sebagai kunci publik, dan kunci ini bersifat terbuka dan dapat diketahui oleh siapa saja. Sebaliknya, untuk proses dekripsi, kunci yang digunakan disebut kunci privat, yang hanya diketahui oleh penerima pesan (Saputro, 2020). Oleh karena itu, kriptografi asimetris juga sering dikenal dengan sebutan kriptografi kunci publik.

Pada jenis kriptografi ini, setiap individu yang berinteraksi memiliki sepasang kunci, yakni kunci privat dan kunci publik. Ketika seorang pengirim ingin mengirimkan pesan kepada penerima, pengirim akan menggunakan kunci publik penerima untuk mengenkripsi pesan tersebut. Namun, hanya penerima yang memiliki pengetahuan mengenai kunci privatnya sendiri yang dapat melakukan dekripsi terhadap pesan tersebut. Penting untuk dicatat bahwa hanya memiliki pengetahuan mengenai kunci publik saja tidak akan memungkinkan seseorang untuk menemukan kunci rahasia. Pasangan kunci publik dan kunci rahasia ini digunakan untuk melakukan dua transformasi yang bersifat saling terbalik satu sama lain, tetapi kunci rahasia tidak dapat diturunkan atau dihitung dari kunci publik. Dalam sistem kriptografi kunci publik ini, proses enkripsi dan dekripsi menggunakan dua kunci yang berbeda, namun kunci-kunci tersebut memiliki hubungan matematis, oleh karena itu sistem ini juga disebut sebagai sistem kriptografi asimetris.



Gambar 2.3 Proses Enkripsi dan Dekripsi Kunci Simetris

Keunggulan kriptografi asimetris adalah bahwa tidak perlu memiliki kunci rahasia sebanyak jumlah orang yang ingin berkomunikasi secara rahasia. Cukup dengan membuat dua jenis kunci, yaitu kunci publik yang dapat digunakan oleh semua pihak untuk mengenkripsi pesan, dan kunci privat yang hanya dimiliki

oleh penerima pesan untuk mendekripsi pesan tersebut. Kunci publik dapat aman dikirimkan kepada penerima melalui saluran yang sama dengan saluran yang digunakan untuk mengirim pesan. Biasanya, saluran pengiriman pesan tersebut tidak terjamin keamanannya. Namun, proses algoritma dalam kriptografi asimetris tergolong lambat jika dibandingkan dengan algoritma kunci simetris, dan ini adalah kelemahan utama. Selain itu, untuk mencapai tingkat keamanan yang setara, ukuran rata-rata kunci dalam kriptografi asimetris harus lebih besar dibandingkan dengan ukuran kunci dalam kriptografi simetris.

2.3 Hybrid Cryptosystem

Hybrid Cryptosystem, menurut Schneier, adalah kombinasi dari dua jenis kriptosistem, yaitu kriptosistem asimetris dan kriptosistem simetris. Algoritma *hybrid* adalah algoritma yang memanfaatkan dua tingkat kunci. Pertama, terdapat kunci simetris yang sering disebut sebagai "*session key*" (kunci sesi), yang digunakan untuk mengenkripsi dan mendekripsi data. Kedua, terdapat pasangan kunci, yaitu kunci rahasia dan kunci publik, yang digunakan untuk melindungi dan mengamankan kunci simetris tersebut (Ariyus, 2006).

Dalam *Hybrid Cryptosystem*, proses enkripsi atau dekripsi pesan dilakukan menggunakan kriptografi kunci simetris, sementara kunci simetris itu sendiri dienkripsi atau didekripsi menggunakan kunci publik. Kunci simetris ini biasanya dihasilkan oleh salah satu pihak dan digunakan untuk mengenkripsi pesan. Selanjutnya, kunci simetris tersebut dienkripsi dengan menggunakan kunci publik penerima, dan hasil enkripsi tersebut dikirimkan bersama dengan pesan yang sudah dienkripsi. Ketika penerima menerima pesan, langkah pertama yang dilakukannya adalah mendekripsi kunci sesi dengan menggunakan kunci privatnya sendiri. Setelah mendapatkan kunci sesi, penerima dapat mendekripsi pesan itu sendiri dengan menggunakan kunci sesi yang telah diterima. Dengan cara ini, pesan dapat diamankan selama proses pengiriman dan hanya dapat dibuka oleh penerima yang memiliki kunci privat yang sesuai.

Hybrid Cryptosystem sering digunakan karena menggabungkan keunggulan kecepatan pemrosesan data oleh algoritma simetris dengan kemudahan transfer

kunci menggunakan algoritma asimetris. Dengan cara ini, sistem dapat mencapai peningkatan kecepatan tanpa mengorbankan kenyamanan atau keamanan. *Hybrid cryptosystem* juga memungkinkan untuk mengombinasikan algoritma simetrik, asimetrik, *hash function*, dan *digital signature* dalam satu kali pemrosesan data, sehingga keempat aspek keamanan dapat dijamin.

2.4 Algoritma *Multiplicative Substitution Cryptosystem*

MSC (Multiplicative Substitution Cryptosystem) adalah algoritma simetris yang diperkenalkan oleh Vemulapalli Rajesh dan Panchami V pada tahun 2019. Untuk mengenkripsi *plaintext*, *MSC* menggunakan kunci yang dihasilkan dari fungsi *non-linear* pada kunci asli yang digunakan untuk memutihkan kunci. Analisis menunjukkan bahwa bila dibandingkan dengan *cipher* substitusi lain (*Caesar Cipher*, *Vigenere Cipher*, dan *Hill Cipher*) *MSC* menawarkan keamanan tinggi, kompleksitas waktu yang lebih sedikit, dan *throughput* yang tinggi (Rajesh, 2019).

2.4.1 Proses Pembangkitan Kunci dengan Algoritma *MSC*

Pada kasus panjang kunci kurang dari 256, jika panjang teks biasa kurang dari 256, maka kunci dapat diperluas dengan pengulangan agar sesuai dengan panjang Teks biasa untuk enkripsi dan dekripsi. Jika panjang teks biasa lebih besar dari 256. Kunci diperluas hingga panjang 256 dengan pengulangan dan kemudian kunci yang diperlukan lebih lanjut dihasilkan dari kunci yang ada.

1. Mulai
2. Baca *Plaintext* p
3. Hitung Panjang dari *Plaintext* p
4. $n = \text{panjang}(p)$
5. $j = n - 256$
6. Jika $(n \leq 256)$
 - a. Atur panjang kunci menjadi $1 \leq k \leq 256$
 - b. Perluas kunci dengan pengulangan agar sesuai dengan panjang dari *Plaintext*
 - c. Kembalikan nilai K
7. Jika $(n > 256)$
 - a. Perluas kunci dengan pengulangan hingga panjangnya 256

b. jika ($i \leq j$)

- $ki + 256 = ki \oplus ki + 1$
- $i = i + 1$
- Kembalikan nilai $ki + 256$

8. Berhenti

2.4.2 Proses Enkripsi Algoritma MSC

Input : Plaintext p, Key k, Initialize a = 1

Output : Ciphertext c

Algoritma:

1. Perluas kunci menggunakan fungsi *key generation*.
2. Mengkonversi karakter *Plaintext p* dan *Key k* menjadi nilai *ASCII*.
3. Simpan nilai di p dan k
4. Inisialisasi $a = 1$
5. Baca setiap nilai dari p dan k
6. Hitung Panjang dari *Plaintext p*
 - a. $n = panjang(p)$
7. Untuk ($a \leq n$)
 - a. Hitung $q = k^3 + a * k$
 - b. Jika q adalah genap
 - Inisialisasi $b = 1$
 - Kembalikan nilai q, $q = q + b$
 - c. Jika q adalah ganjil
 - Inisialisasi $b = 0$
 - Kembalikan nilai q, $q = q + b$
 - d. Hitung $c = p * q \bmod 256$
 - e. $a = a + 1$
 - f. Kembalikan nilai c
8. *c ciphertext*

2.4.3 Proses Dekripsi Algoritma MSC

Proses input pada dekripsi adalah c , k . *Ciphertext* c , *key* k adalah karakter, dan *key* diperluas dengan pengulangan agar cocok dengan panjang *ciphertext*. Karakter - karakter ini dikonversi menjadi nilai *ASCII* yang sesuai yang berkisar dari 0 hingga 255.

$$q = (k^3 + a * k) \bmod 256$$

Di mana k adalah nilai *key*, a adalah nomor pengindeksan yang mana berada dalam rentang panjang *ciphertext*. Jika karakter pertama dari *ciphertext* sedang didekripsi maka nilai a mengambil 1, jika karakter kedua maka nilai a mengambil 2 juga. b nilainya 0 atau 1. Jika nilai q genap, maka nilai b mengambil 1 selain 0. $d = q + b$. Nilai d dihitung dari nilai q dimana $(d * q) \bmod 256 = 1$. Nilai d adalah *inverse* perkalian dari $q \bmod 256$. *Plaintext* dihitung dari persamaan $p = (c * d) \bmod 256$.

$$\text{Dekripsi : } p = (c * d) \bmod 256$$

2.4.4 Proses Perhitungan Algoritma MSC

a. Pembangkitan kunci :

1. Input plainteks dan kunci, contoh $p = \text{"DAFFA"}$ dan $k = \text{"AL"}$
2. Input $n = \text{Panjang}(p)$; $n = 5$
3. Karena $n \leq 256$ maka perluas kunci dengan pengulangan hingga sepanjang plainteks (p).
4. $k = \text{"ALALA"}$

b. Enkripsi :

1. Ubah karakter *Plaintext* p dan *Key* k menjadi nilai *ASCII*.

$$p : D = 68$$

$$A = 65$$

$$F = 70$$

$$F = 70$$

$$A = 65$$

$$k : A = 65$$

$$L = 76$$

$$A = 65$$

$$L = 76$$

$$A = 65$$

2. Hitung $q_i = k^3 + a * k$, dimana $a = i$.

$$q_1 = 65^3 + 1 * 65 = 274.690$$

$$q_2 = 76^3 + 2 * 76 = 439.128$$

$$q_3 = 65^3 + 3 * 65 = 274.820$$

$$q_4 = 65^3 + 4 * 76 = 439.280$$

$$q_5 = 65^3 + 5 * 65 = 274.950$$

3. Jika q genap, maka $q + 1$.

$$q_1 = 274.690 + 1 = 274.691$$

$$q_2 = 439.128 + 1 = 439.129$$

$$q_3 = 274.820 + 1 = 274.821$$

$$q_4 = 439.280 + 1 = 439.281$$

$$q_5 = 274.950 + 1 = 274.951$$

4. Hitung $c = p * q \bmod 256$

$$c_1 = 68 * 274.691 \bmod 256 = 204$$

$$c_2 = 65 * 439.129 \bmod 256 = 153$$

$$c_3 = 70 * 274.821 \bmod 256 = 94$$

$$c_4 = 70 * 439.281 \bmod 256 = 230$$

$$c_5 = 65 * 274.951 \bmod 256 = 199$$

5. Hasil *Ciphertext* adalah

$$c = 204\ 153\ 094\ 230\ 199$$

c. Dekripsi :

1. Ambil nilai c

$$c = 204\ 153\ 094\ 230\ 199$$

2. Hitung $(d * q) \bmod 256 = 1$.

$$(d_1 * q_1) \bmod 256 = 1 ; d_1 = 171$$

$$(d_2 * q_2) \bmod 256 = 1 ; d_2 = 233$$

$$(d_3 * q_3) \bmod 256 = 1 ; d_3 = 77$$

$$(d_4 * q_4) \bmod 256 = 1 ; d_4 = 17$$

$$(d_5 * q_5) \bmod 256 = 1 ; d_5 = 183$$

$$d = 171\ 233\ 77\ 17\ 183$$

3. Hitung $p = (c * d) \bmod 256$.

$$p_1 = (c * d) \bmod 256 = 204 * 171 \bmod 256$$

$$= 68$$

$$p_2 = (c * d) \bmod 256 = 153 * 233 \bmod 256$$

$$= 65$$

$$p_3 = (c * d) \bmod 256 = 94 * 77 \bmod 256$$

$$= 70$$

$$p_4 = (c * d) \bmod 256 = 230 * 17 \bmod 256$$

$$= 70$$

$$p_5 = (c * d) \bmod 256 = 199 * 183 \bmod 256$$

$$= 65$$

$$p = 68\ 65\ 70\ 70\ 65$$

4. Ubah p menjadi karakter *ASCII*

$$p = D\ A\ F\ F\ A$$

2.5 Algoritma *Multi-Factor RSA*

Multi-Factor RSA adalah algoritma kriptografi asimetris yang merupakan hasil pengembangan dari algoritma *RSA*. Pada *Multi-Factor RSA* ditemukan hasil waktu proses 2,25 kali lebih cepat dari algoritma *RSA* (Boneh, 2002).

2.5.1 Proses Pembangkitan Kunci dengan Algoritma *Multi-Factor RSA*

1. Hasilkan nilai b di mana b adalah bilangan prima dan $b > 3$.
2. Menghasilkan bilangan prima sebanyak b , lalu cek prima dengan *Fermat's Little Theorem*.
 - a. Bangkitkan nilai p .
 - b. Ambil nilai a secara acak, dengan $1 < a < p$.
 - c. Bila $a^{(p-1)} \bmod p = 1$ dimana $1 < a < p$, maka p adalah prima.
 - d. Jika tidak, p adalah bilangan komposit.

3. Hitung nilai dari $n \leftarrow \prod_{i=1}^b p_i$
4. Cari nilai dari $\phi(n) \leftarrow \prod_{i=1}^b (p_i - 1)$
5. Tentukan nilai e sesuai standar *RSA public key* yaitu $e = 65537$.
6. Hasilkan nilai d yang merupakan invers dari $e \pmod{\phi(n)}$.

2.5.2 Proses Enkripsi Algoritma *Multi-Factor RSA*

1. Ambil kunci publiknya yaitu N dan e .
2. Enkripsi pesan dengan rumus $c = m^e \pmod{n}$
3. Kirim c ke penerima yang dituju.

2.5.3 Proses Dekripsi Algoritma *Multi-Factor RSA*

1. Ambil pesan terenkripsi yaitu c , dan siapkan private key yang sudah dihasilkan $(d, p_1, p_2, p_3, \dots, p_n)$.
2. Dekripsi pesan menggunakan rumus

$$m = c^d \pmod{n}.$$

2.5.4 Proses Perhitungan Algoritma *Multi-Factor RSA*

a. Pembangkitan kunci :

1. Input $p_1 p_2 p_3$ dimana p bilangan prima.

$$p_1 = 7 ; p_2 = 13 ; p_3 = 11$$
2. Hitung $n = p_1 * p_2 * p_3$

$$n = 7 * 13 * 11$$

$$n = 1001$$
3. Hitung $\phi(n) = (p_1 - 1) * (p_2 - 1) * (p_3 - 1)$

$$\phi(n) = (p_1 - 1) * (p_2 - 1) * (p_3 - 1)$$

$$= 6 * 12 * 10$$

$$= 720$$
4. Tentukan nilai e . Standar *RSA public key* yaitu $e = 65537$, e bisa menggunakan nilai lain tetapi e harus relative prima dengan $\phi(n)$.

$$e = 23$$
5. Hasilkan nilai d yang merupakan invers dari $e \pmod{\phi(n)}$.

$$d = e^{-1} \pmod{\phi(n)}$$

$$d = 407$$

b. Enkripsi :

1. Input *plaintext* m.

$$m = "A"$$

Lalu ubah karakter *plaintext* ke nilai *ASCII*, $A = 65$

2. Ambil kunci publik yaitu n dan e.

$$n = 1001 \quad e = 23$$

3. Enkripsi pesan dengan rumus $c = m^e \pmod{n}$

$$\begin{aligned} c &= m^e \pmod{n} \\ &= 65^{23} \pmod{1001} \\ &= 494 \end{aligned}$$

4. Kirim c ke penerima yang dituju.

c. Dekripsi :

1. Ambil pesan terenkripsi yaitu $c = 494$

$$n = 1001 \quad e = 23$$

2. Dekripsi pesan dengan rumus $m = c^d \pmod{n}$

$$\begin{aligned} m &= c^d \pmod{n} \\ &= 494^{407} \pmod{1001} \\ &= 21710754 \pmod{1001} \\ &= 65 \end{aligned}$$

3. Ubah m menjadi karakter *ASCII*.

$$m = "A"$$

2.6 Algoritma *Brute Force*

Metode *Brute Force* adalah metode kriptanalisis yang paling sering berhasil dalam mendapatkan hasil (felix, 2006). Kelebihan dari *Brute Force* adalah sederhana, mudah, dan pasti menemukan jawabannya. Masalah utama dari metode ini adalah waktu komputasi yang sangat lama. Karena algoritma ini menghasilkan dan menguji semua kandidat solusi yang mungkin, algoritma brute force disebut juga *exhaustive search* (Budiman, 2018).

2.7 Penelitian Relevan

Beberapa penelitian terdahulu yang relevan antara lain :

1. Berdasarkan penelitian yang dilakukan oleh Dian Dwi Prayoga dengan judul *“Implementation Of Combination Of Multi-Factor Algorithm RSA And Cipher 4x4 In Hybrid Cryptography Scheme For Security Text Messages On Instant Messaging Application”* (2021) disimpulkan bahwa hasil penelitian menunjukkan proses enkripsi dan dekripsi pesan teks berbanding lurus dengan panjang pesan terkirim dan mencatat total waktu yang diperlukan untuk enkripsi dan dekripsi pesan teks lebih dari 3000 karakter adalah 1,034 detik.
2. Berdasarkan penelitian Fauza Badratul Chairiah yang berjudul *“VMPC (Variably Modified Permutation Composition) And MultiFactor RSA In Text File Security”* (2020) disimpulkan bahwa proses enkripsi dan dekripsi metode hybrid dari algoritma *Multi-Factor RSA* dan algoritma *VMPC* memenuhi kriteria integritas data, ukuran *file ciphertext* setelah proses enkripsi menjadi lebih kecil dari ukuran *file plaintext*, pertumbuhan waktu enkripsi dan dekripsi dari algoritma *VMPC* dan algoritma *Multi-Factor RSA* berbanding lurus. Semakin panjang karakter suatu file, semakin besar pula waktu yang dibutuhkan untuk enkripsi dan dekripsi proses. Dan proses enkripsi pada Algoritma *Multi-Factor RSA* membutuhkan waktu yang lebih lama dibandingkan proses dekripsi, sedangkan proses dekripsi dalam algoritma *VMPC* membutuhkan waktu lebih lama dari proses enkripsi.
3. Berdasarkan penelitian yang dilakukan oleh Vemulapalli Rajesh dan Panchami V dengan judul *“A Novel Multiplicative Substitution Cryptosystem”* (2019) disimpulkan bahwa pada *Cipher* menggunakan simbol selain alfabet dan angka. Dalam penelitian telah dikembangkan *Multiplicative Substitution Cryptosystem* yang menggunakan Algoritma *MSC* untuk mengenkripsi plaintext menggunakan kunci yang dihasilkan dari fungsi non-linear pada kunci asli yang digunakan untuk memutihkan kunci. Berlaku untuk berbagai karakter (*UTF8*) 256 jumlahnya. Karena dari berbagai karakter yang memecahkan sandi sangat keras bila dibandingkan dengan *cipher* substitusi primitif. *Cipher* ini cukup cepat dan dapat diterapkan pada jangkauan yang luas aplikasi. Ini menawarkan keamanan yang sangat tinggi.

4. Berdasarkan penelitian yang dilakukan oleh Mohammad Andri Budiman, Poltak Sihombing dan I A Fikri yang berjudul “*A cryptocompression system with Multi-Factor RSA algorithm and Levenstein code algorithm*” (2021) disimpulkan bahwa dengan skema kompres lalu enkripsi dan skema enkripsi lalu kompres di algoritma *Multi-Factor RSA* and *Levenstein code* menghasilkan pengurangan ukuran data. Namun, skema enkripsi lalu kompres lebih baik daripada skema kompres lalu enkripsi dalam hal rasio kompresi. Di sisi lain, skema kompres lalu enkripsi lebih cepat daripada skema enkripsi lalu kompres jika kinerja waktu dipertimbangkan.
5. Berdasarkan penelitian yang dilakukan oleh D Rachmawati, M A Budiman dan D F Perangin-angin dengan judul “*A hybrid cryptosystem approach for information security by using RC4 algorithm and LUC algorithm*” (2019) disimpulkan bahwa dengan menggunakan konsep *hybrid cryptosistem* yaitu algoritma *RC4* untuk pengamanan pesan dan algoritma *LUC* untuk mengamankan kunci *RC4*, pesan menjadi lebih aman daripada menggunakan algoritma tunggal. Setelah proses dekripsi, pesan dapat dikembalikan ke bentuk aslinya. *Ciphertext* dan *cipher key* yang telah dienkripsi akan berubah menjadi bentuk angka desimal. Berdasarkan percobaan dilakukan pada algoritma *RC4* dan algoritma *LUC*, waktu enkripsi dan dekripsi yang dibutuhkan pada kedua algoritma tersebut berbanding lurus, yaitu semakin lama pesan dan kunci yang digunakan, semakin lama waktu yang dibutuhkan lebih lama.

BAB 3

ANALISIS DAN PERANCANGAN

3.1 Analisis Sistem

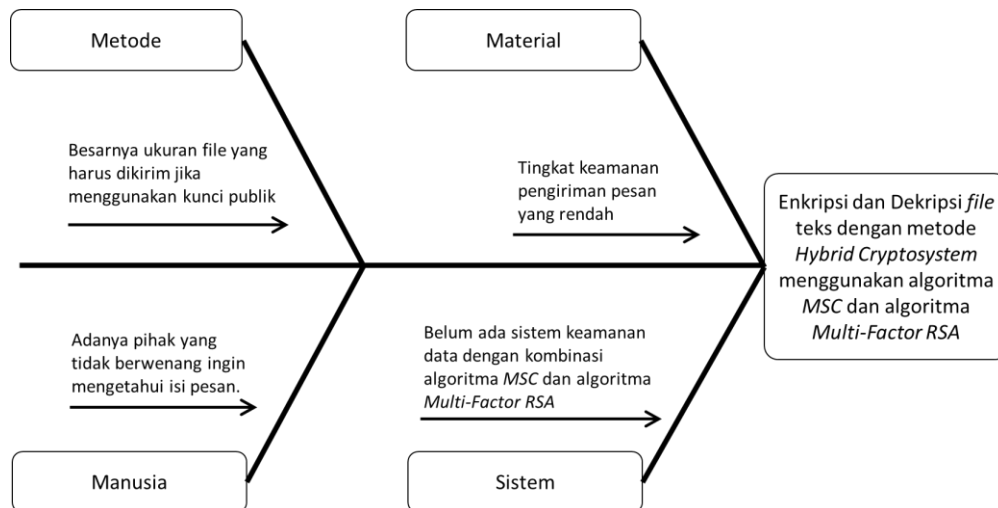
Perancangan dan evaluasi yang efektif adalah prasyarat bagi sebuah sistem yang baik. Tahap awal dalam pengembangan sistem adalah analisis, yang melibatkan identifikasi kebutuhan yang diperlukan untuk membangun sistem tersebut. Melalui analisis sistem, tujuan sistem dapat tercapai dengan baik, sehingga sistem dapat beroperasi secara optimal.

3.1.1 Analisis Masalah

Pada tahap awal, penting untuk menganalisis sistem yang akan dirancang agar sistem tersebut dapat lebih efektif dan sesuai dengan ide yang ada. Melakukan analisis akan memudahkan dalam proses pengembangan sistem dan memungkinkan lebih mudahnya menemukan dan memperbaiki masalah yang tak terduga di masa depan.

Untuk mengurangi risiko penyebaran pesan kepada pihak yang tidak bertanggung jawab, diperlukan suatu sistem yang mampu menjaga pesan agar kerahasiaannya terjaga hingga pesan sampai kepada penerima yang berhak membacanya. Meskipun demikian, terdapat potensi bahwa pihak yang tidak berwenang dapat mengakses pesan terenkripsi jika kunci untuk membuka pesan sudah diketahui. Oleh karena itu, diperlukan suatu algoritma yang dapat menjaga keamanan kunci tersebut.

Setiap masalah akan dianalisis dengan menggunakan diagram Ishikawa, yang juga dikenal sebagai *Fishbone Diagram*. Masalah-masalah tersebut akan diuraikan dalam diagram ini untuk mengidentifikasi faktor-faktor yang berperan dalam terjadinya masalah tersebut.



Gambar 3.1 Diagram Ishikawa Penelitian

Pada Gambar 3.1 dapat dilihat bahwa diagram ini memiliki bentuk seperti ikan, dengan rumusan masalah ditempatkan pada bagian kepala ikan yang terletak pada bagian paling kanan, sementara penyebab masalah ditempatkan pada bagian tulang ikan. Masalah dari penelitian yang ingin diselesaikan adalah enkripsi dan dekripsi *file* teks dengan metode *Hybrid Cryptosystem* menggunakan algoritma *MSC* dan algoritma *Multi-Factor RSA*. Penyebab masalah pada penelitian ini terbagi menjadi empat kategori, yaitu Metode, Material, Manusia, dan Sistem.

3.1.2 Analisis Kebutuhan

Analisis kebutuhan dibagi menjadi dua, yaitu :

1. Kebutuhan fungsional sistem

Kebutuhan fungsional yang wajib dimiliki oleh sistem adalah:

- Sistem dapat menerima masukan *file* berekstensi *.txt.
- Sistem mempunyai kemampuan untuk membangkitkan *public key* dan *private key* menggunakan algoritma *Multi-Factor RSA*.
- Sistem dapat mengenkripsi *file* berekstensi *.txt berupa *string* menggunakan algoritma *MSC*.
- Sistem dapat mengenkripsi kunci algoritma *MSC* menggunakan *public key Multi-Factor RSA* dan mendekripsi kunci algoritma *MSC* dengan *private key Multi-Factor RSA*.
- Sistem dapat mendekripsi *file* sehingga menjadi bentuk semula dengan menggunakan algoritma *MSC*.

- f. Sistem bisa menyimpan *file* yang terenkripsi maupun terdekripsi dan menyimpan *public key* dan *private key*.

2. Kebutuhan Non-Fungsional

Kebutuhan non-fungsional adalah kebutuhan yang menjelaskan tentang fitur, karakteristik, dan batasan lainnya. Kebutuhan non-fungsional dalam sistem adalah sebagai berikut:

a. Performa

Sistem dapat menampilkan hasil proses enkripsi file teks dan dapat mengembalikan menjadi file asli melalui proses dekripsi.

b. User Friendly

Sistem dibuat agar mudah untuk digunakan oleh pengguna.

c. Kontrol

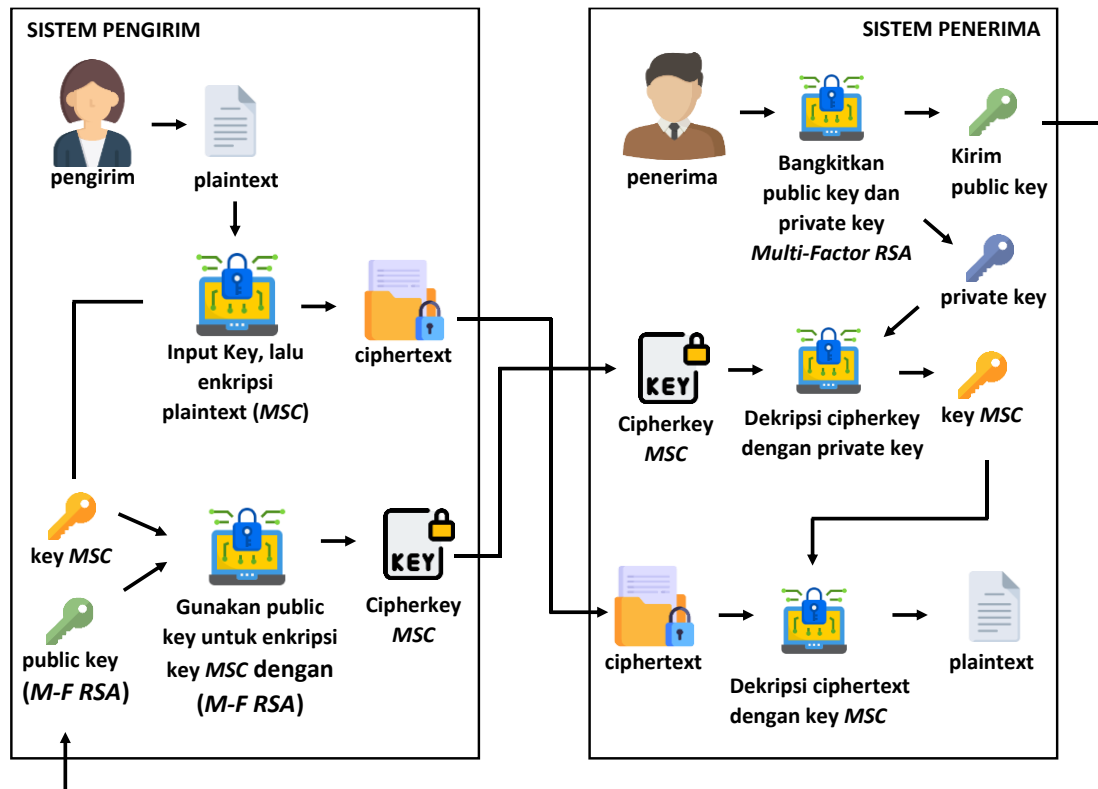
Sistem dapat menampilkan pesan peringatan kesalahan (*error message*) apabila ada kesalahan pada sistem.

d. Kualitas

Sistem dapat menghasilkan output yang benar dan akurat dalam proses pembangkitan kunci, enkripsi dan dekripsi.

3.1.3 Diagram Umum

Diagram umum sistem merupakan gambaran bagaimana konsep kerja dari sistem yang dibuat. Diagram umum sistem tersebut dapat dilihat pada gambar berikut:



Gambar 3.2 Diagram umum sistem

Penjelasan alur proses diagram umum sistem pada gambar adalah sebagai berikut:

- **Pengirim :**

1. Pengirim menginputkan key (MSC), lalu mengenkripsi file teks (Plaintext) menggunakan algoritma MSC yang akan menghasilkan Ciphertext.
2. Pengirim mengenkripsi key (MSC) dengan algoritma Multi-Factor RSA menggunakan public key (Multi-Factor RSA) yang didapatkan dari penerima, sehingga menghasilkan Cipherkey.
3. Cipherkey dan Ciphertext dikirim ke penerima.

- **Penerima :**

1. Penerima membangkitkan public key dan private key menggunakan algoritma Multi-Factor RSA. Kemudian public key dikirim ke pengirim.
2. Penerima mendapatkan Ciphertext dan Cipherkey dari pengirim yang menggunakan public key miliknya.

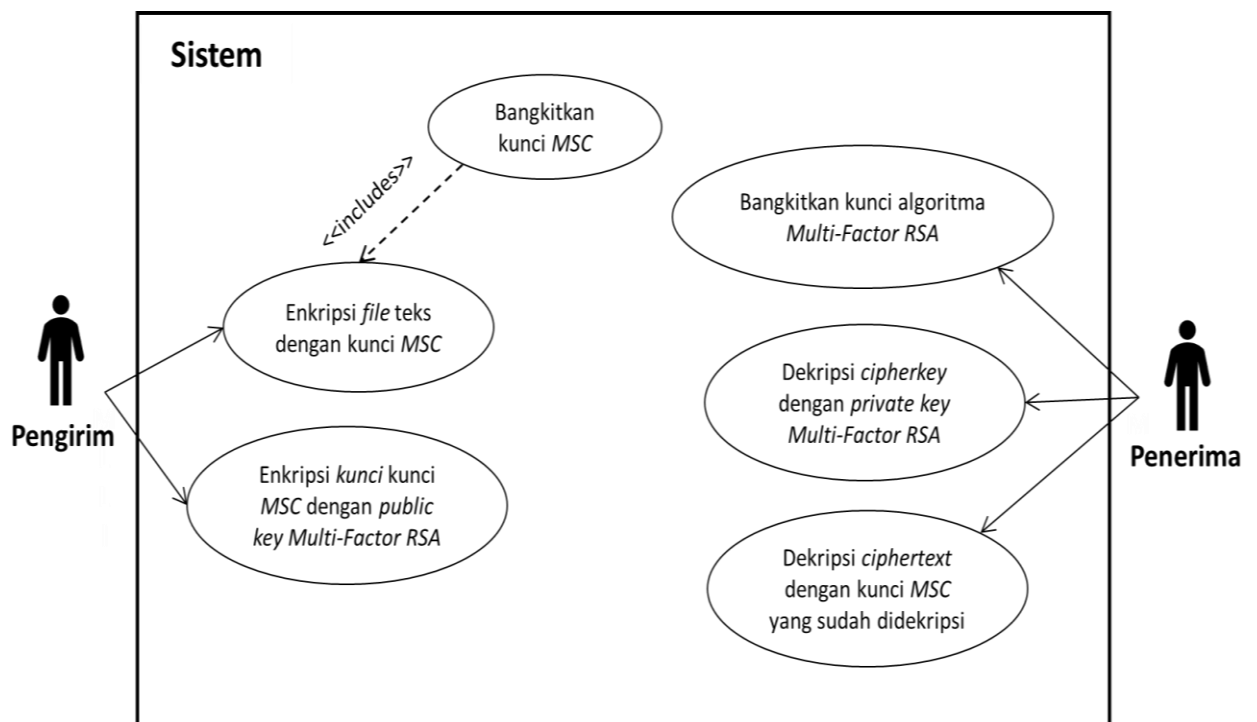
3. Penerima mendekripsi Cipherkey menggunakan private key (*Multi-Factor RSA*) yang menghasilkan key (*MSC*) untuk membuka Ciphertext.
4. Ciphertext dapat didekripsi dengan key (*MSC*) untuk melihat file teks (*Plaintext*) yang dikirim oleh pengirim.

3.2 Pemodelan Sistem

Pemodelan yang akan digunakan pada sistem ini adalah *use case diagram*, *activity diagram*, *sequence diagram*, *flowchart* dan perancangan *interface*.

3.2.1 Use Case Diagram

Use case diagram adalah diagram yang menggambarkan interaksi dari aktor yang terlibat dengan sistem. Interaksi ini menjelaskan apa saja yang dapat dilakukan oleh masing-masing aktor. *Use case diagram* pada sistem ditunjukkan oleh gambar berikut.



Gambar 3.3 Use Case Diagram

Tabel 3.1 Use Case Scenario Bangkitkan kunci *Multi-Factor RSA*

Nama	Bangkitkan kunci <i>Multi-Factor RSA</i>
Aktor	Penerima
Deskripsi	Proses Pembangkitan kunci <i>Multi-Factor RSA</i>

	Aksi Aktor	Reaksi Sistem
Skenario Normal	1. Penerima menekan tombol bangkitkan kunci <i>Multi-Factor RSA</i>	2. Sistem menampilkan <i>public key</i> dan <i>private key Multi-Factor RSA</i>
Skenario Alternatif	1. Penerima menekan tombol bangkitkan kunci <i>Multi-Factor RSA</i>	2. Sistem menampilkan <i>public key</i> dan <i>private key Multi-Factor RSA</i>

Tabel 3.2 *Use Case Scenario* Bangkitkan kunci *MSC*

Nama	Bangkitkan kunci <i>MSC</i>	
Aktor	Pengirim	
Deskripsi	Proses Pembangkitan kunci <i>MSC</i>	
	Aksi Aktor	Reaksi Sistem
Skenario Normal	1. Pengirim menekan tombol bangkitkan kunci <i>MSC</i>	2. Sistem menampilkan kunci <i>MSC</i>
Skenario Alternatif	1. Pengirim menekan tombol bangkitkan kunci <i>MSC</i>	2. Sistem menampilkan pesan <i>error</i>

Tabel 3.3 *Use Case Scenario* Enkripsi *file* teks

Nama	Enkripsi <i>file</i> teks	
Aktor	Pengirim	
Deskripsi	Proses enkripsi <i>file</i> teks menggunakan <i>MSC</i>	
	Aksi Aktor	Reaksi Sistem
Skenario Normal	1. Pengirim menginput <i>file</i> teks	2. Sistem menampilkan <i>file</i> teks
	3. Pengirim menginput kunci <i>MSC</i>	4. Sistem menampilkan kunci <i>MSC</i>
	5. Pengirim menekan tombol enkripsi	6. Sistem menampilkan <i>file</i> hasil enkripsi
Skenario Alternatif	1. Pengirim menginput <i>file</i> teks	2. Sistem menampilkan <i>file</i> teks

	3. Pengirim menekan tombol enkripsi	4. Sistem menampilkan pesan <i>error</i>
--	-------------------------------------	--

Tabel 3.4 *Use Case Scenario* Enkripsi kunci *MSC*

Nama	Enkripsi kunci <i>MSC</i>	
Aktor	Pengirim	
Deskripsi	Proses enkripsi kunci <i>MSC</i> menggunakan <i>Multi-Factor RSA</i>	
	Aksi Aktor	Reaksi Sistem
Skenario Normal	1. Pengirim menginput kunci <i>MSC</i>	2. Sistem menampilkan kunci <i>MSC</i>
	3. Pengirim menginput <i>public key Multi-Factor RSA</i>	4. Sistem menampilkan <i>public key Multi-Factor RSA</i>
	5. Pengirim menekan tombol enkripsi	6. Sistem menampilkan <i>cipherkey</i>
Skenario Alternatif	1. Pengirim menginput kunci <i>MSC</i>	2. Sistem menampilkan kunci <i>MSC</i>
	3. Pengirim menekan tombol enkripsi	4. Sistem menampilkan pesan <i>error</i>

Tabel 3.5 *Use Case Scenario* Dekripsi kunci *MSC*

Nama	Dekripsi kunci <i>MSC</i>	
Aktor	Penerima	
Deskripsi	Proses dekripsi kunci <i>MSC</i> menggunakan <i>Multi-Factor RSA</i>	
	Aksi Aktor	Reaksi Sistem
Skenario Normal	1. Pengirim menginput <i>cipherkey</i>	2. Sistem menampilkan <i>cipherkey</i>
	3. Pengirim menginput <i>private key Multi-Factor RSA</i>	4. Sistem menampilkan <i>private key Multi-Factor RSA</i>
	5. Pengirim menekan tombol dekripsi	6. Sistem menampilkan kunci <i>MSC</i>

Skenario Alternatif	1. Pengirim menginput <i>cipherkey</i>	2. Sistem menampilkan <i>cipherkey</i>
	3. Pengirim menekan tombol dekripsi	4. Sistem menampilkan pesan <i>error</i>

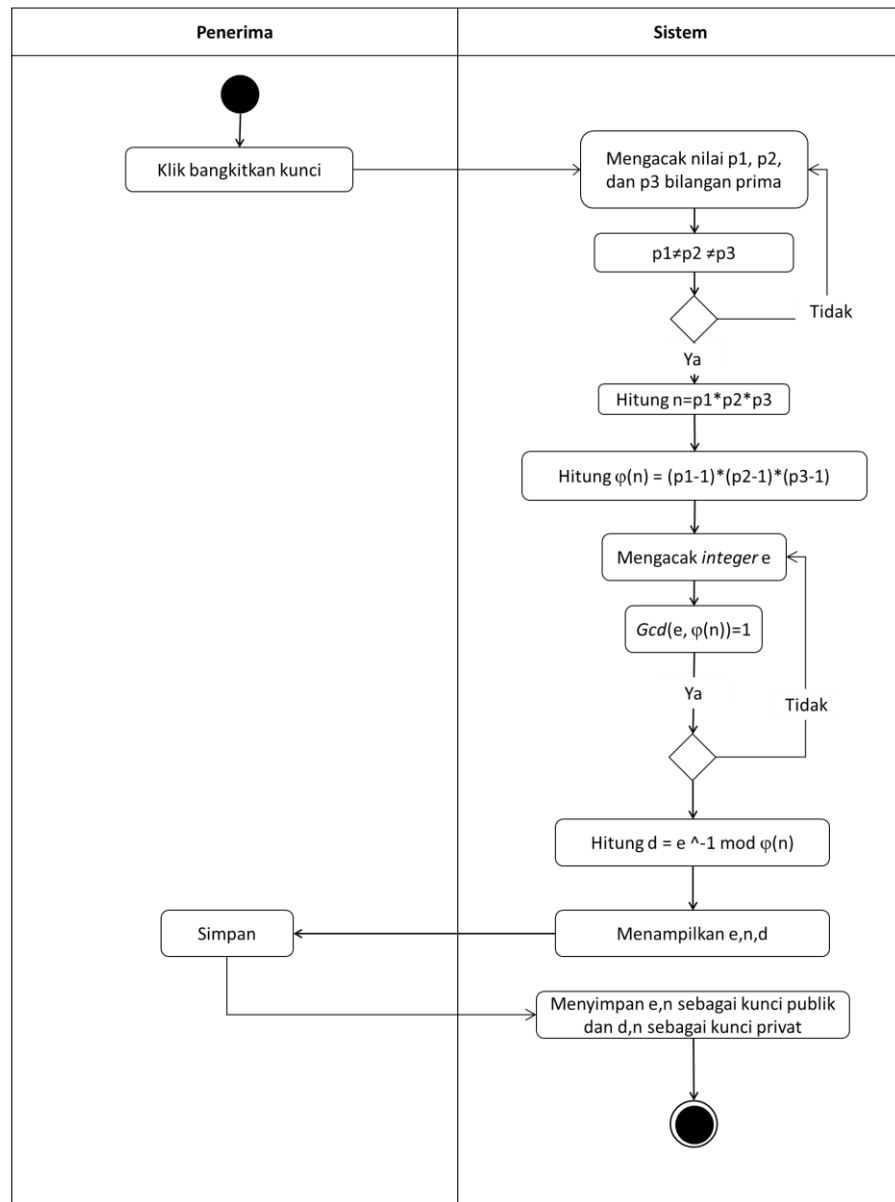
Tabel 3.6 *Use Case Scenario* Dekripsi *ciphertext*

Nama	Dekripsi <i>ciphertext</i>	
Aktor	Penerima	
Deskripsi	Proses dekripsi <i>ciphertext</i> menggunakan <i>MSC</i>	
	Aksi Aktor	Reaksi Sistem
Skenario Normal	1. Pengirim menginput <i>ciphertext</i>	2. Sistem menampilkan <i>ciphertext</i>
	3. Pengirim menginput kunci <i>MSC</i>	4. Sistem menampilkan kunci <i>MSC</i>
	5. Pengirim menekan tombol dekripsi	6. Sistem menampilkan <i>file</i> teks
Skenario Alternatif	1. Pengirim menginput <i>ciphertext</i>	2. Sistem menampilkan <i>ciphertext</i>
	3. Pengirim menekan tombol dekripsi	4. Sistem menampilkan pesan <i>error</i>

3.2.2 Activity Diagram

Activity Diagram adalah suatu representasi grafis yang memberikan gambaran tentang urutan dan interaksi yang terjadi antara pengguna dan sistem secara bertahap, mulai dari awal hingga akhir.

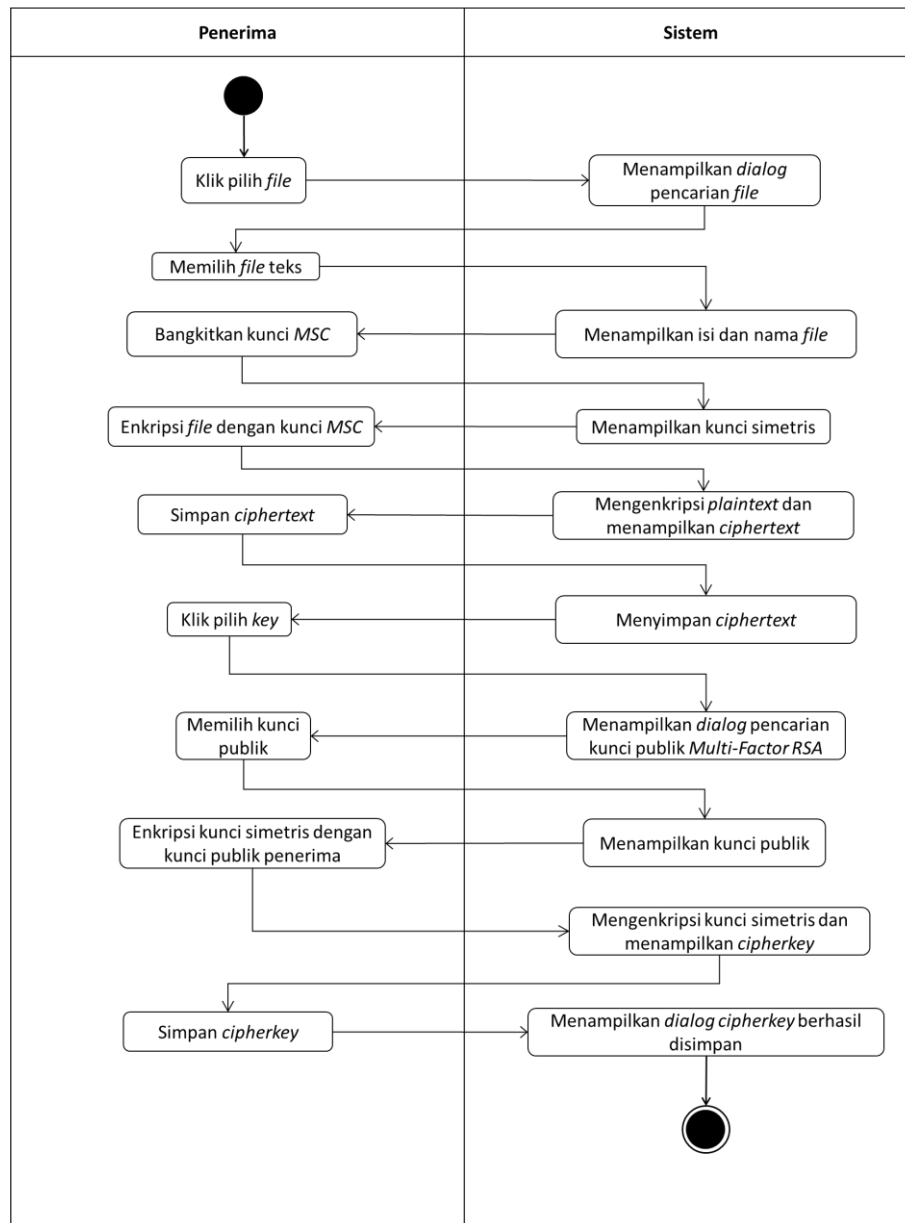
- *Activity diagram* Pembangkit Kunci



Gambar 3.4 Activity Diagram Pembangkit Kunci

Pada Gambar 3.4 dapat dilihat dalam pembangkitan kunci, penerima menekan tombol bangkitkan kunci agar sistem melakukan proses perhitungan yang nantinya akan menghasilkan nilai e, n, d. Setelah nilai dihasilkan, sistem menampilkan nilai e, n, d. Kemudian penerima menekan tombol simpan maka sistem akan menyimpan e dan n sebagai *public key* kemudian menyimpan d dan n sebagai *private key*.

- Activity diagram Proses Enkripsi

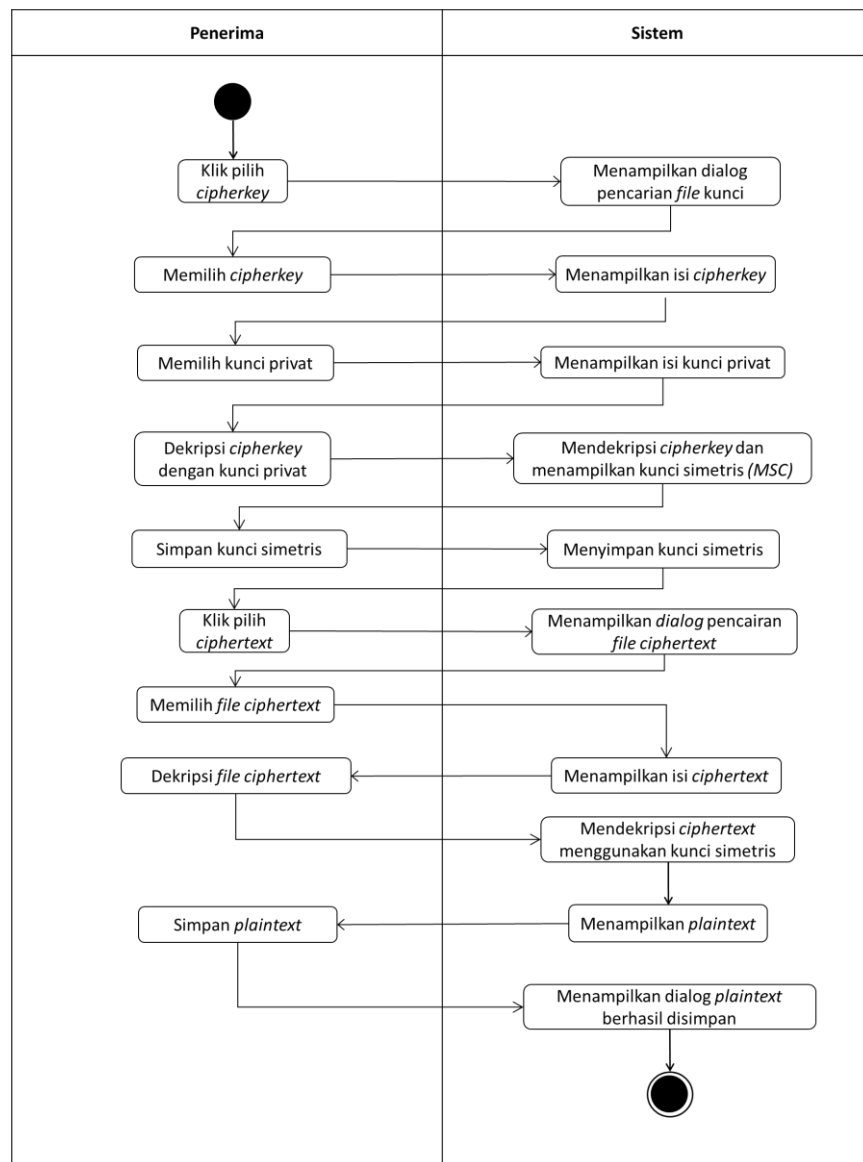


Gambar 3.5 Activity Diagram Proses Enkripsi

Pada Gambar 3.5 dapat dilihat pada proses enkripsi dimulai dari pengirim menekan tombol pilih *file* teks lalu sistem akan menampilkan *dialog* pemilihan *file*, pengirim akan menentukan *file* teks dan sistem kemudian menampilkan isi dan nama *file*. Lalu pengirim akan membangkitkan kunci *MSC* dan memberi perintah untuk mengenkripsi *file*, sistem akan melakukan enkripsi dan menampilkan hasil enkripsi yang berupa *ciphertext*. Kemudian tekan simpan dan sistem menyimpan *file ciphertext*. Selanjutnya pengirim menekan tombol pilih *file public key* lalu sistem akan menampilkan *dialog* pemilihan *file* dan pengirim akan memilih *file*

public key dan sistem menampilkan isi *file*, pengirim akan memerintahkan enkripsi kunci simetris menggunakan algoritma *Multi-Factor RSA*. Sistem akan melakukan enkripsi dan menampilkan hasil enkripsi yang berupa *cipherkey*. Kemudian tekan simpan dan sistem menyimpan *file cipherkey*.

- *Activity diagram* Proses Dekripsi

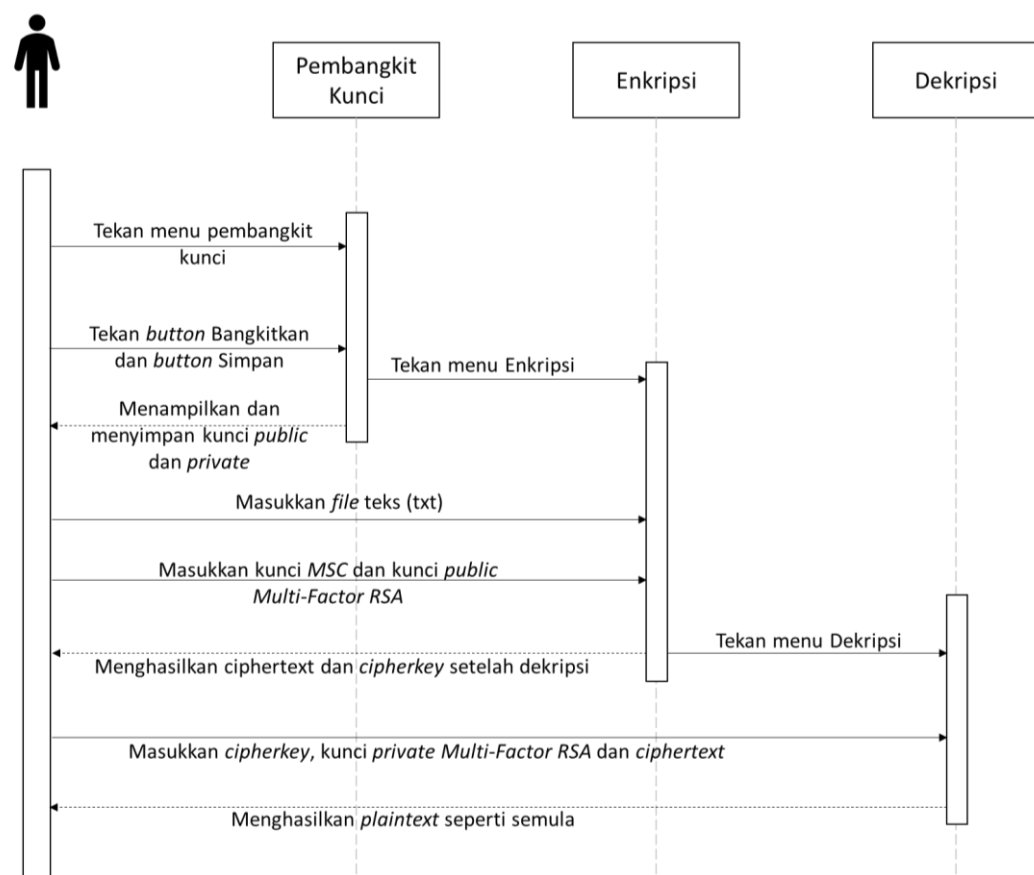


Gambar 3.6 Activity Diagram Proses Dekripsi

Pada Gambar 3.6 dapat dilihat pada proses dekripsi dimulai dari penerima menekan tombol pilih *file* lalu sistem akan menampilkan *dialog* pemilihan *file*, penerima akan menentukan *file cipherkey* dan *private key*. Sistem kemudian menampilkan isi *file*. Lalu penerima menekan tombol dekripsi kunci, sistem akan

melakukan dekripsi dan menampilkan hasil dekripsi yang berupa kunci *MSC*. Selanjutnya penerima menekan tombol pilih *file* untuk memilih *ciphertext*. Sistem akan menampilkan *dialog* pemilihan *file*, dan penerima akan menentukan *file ciphertext*. Sistem menampilkan isi *file*, lalu penerima menekan tombol dekripsi *ciphertext*. Sistem akan melakukan dekripsi dan menampilkan hasil dekripsi yang berupa *plaintext*. Kemudian penerima tekan simpan dan sistem menyimpan *file plaintext*.

3.2.3 Sequence Diagram



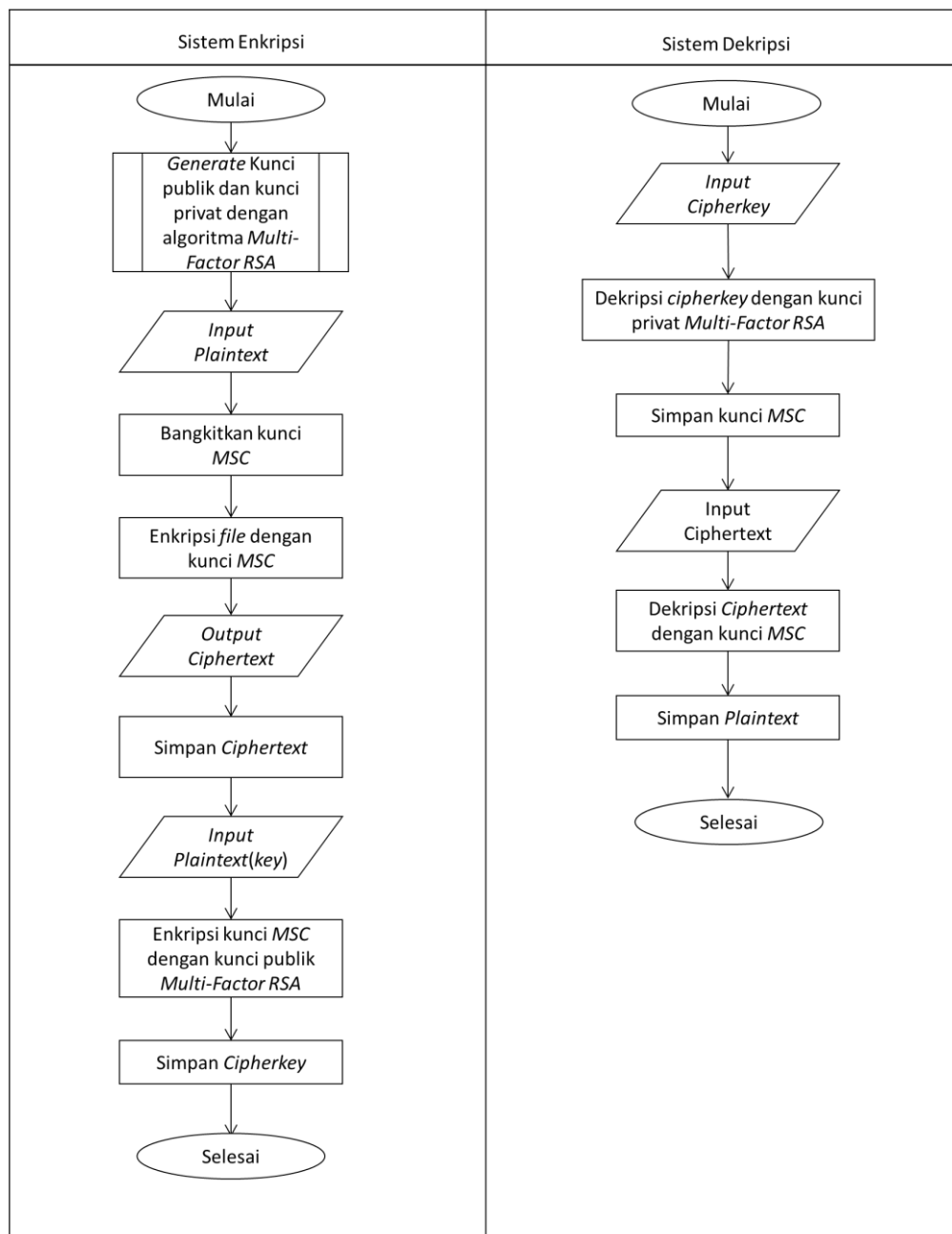
Gambar 3.7 Sequence Diagram Sistem

Pada Gambar 3.7 menunjukkan interaksi pengguna dengan sistem secara *sequence*. Saat menggunakan sistem, pengguna akan berinteraksi dengan halaman beranda, halaman pembangkit kunci, halaman enkripsi, halaman dekripsi dan halaman bantuan.

3.3 Flowchart

Flowchart (diagram alir) ialah suatu diagram yang disusun secara sistematis dan bersifat logis yang langkah - langkahnya diwakilkan dalam bentuk simbol-simbol grafis yang urutannya dihubungkan dengan anak panah. *Flowchart* berfungsi untuk menyelesaikan masalah-masalah secara berurutan langkah demi langkah sampai mendapatkan solusi yang sesuai keinginan

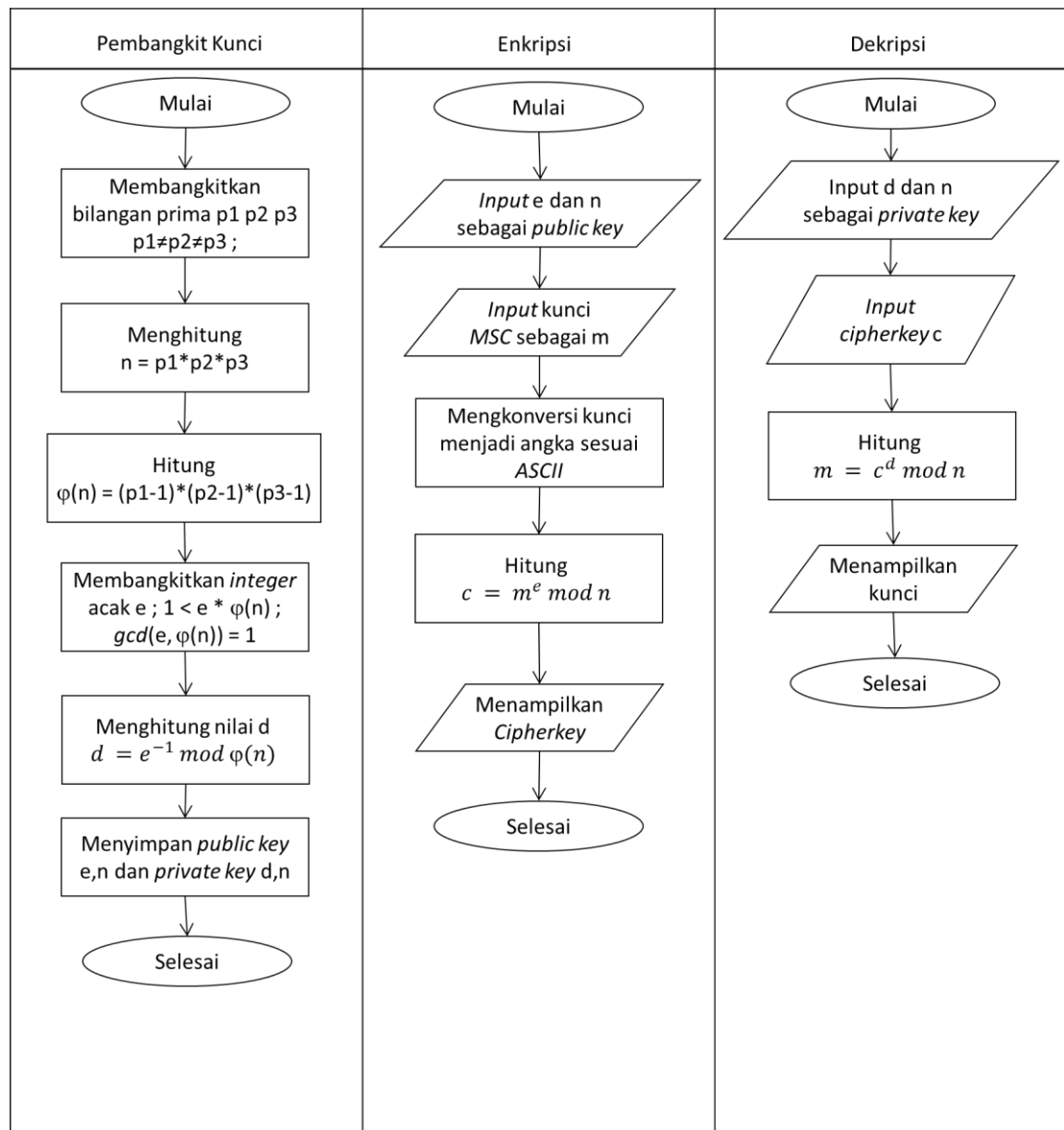
3.3.1 Flowchart sistem



Gambar 3.8 Flowchart sistem

Pada Gambar 3.8 menunjukkan gambaran umum proses sistem secara keseluruhan dari proses pembangkitan kunci, proses enkripsi *file* teks dan kunci hingga proses dekripsi *ciphertext* dan *cipherkey*.

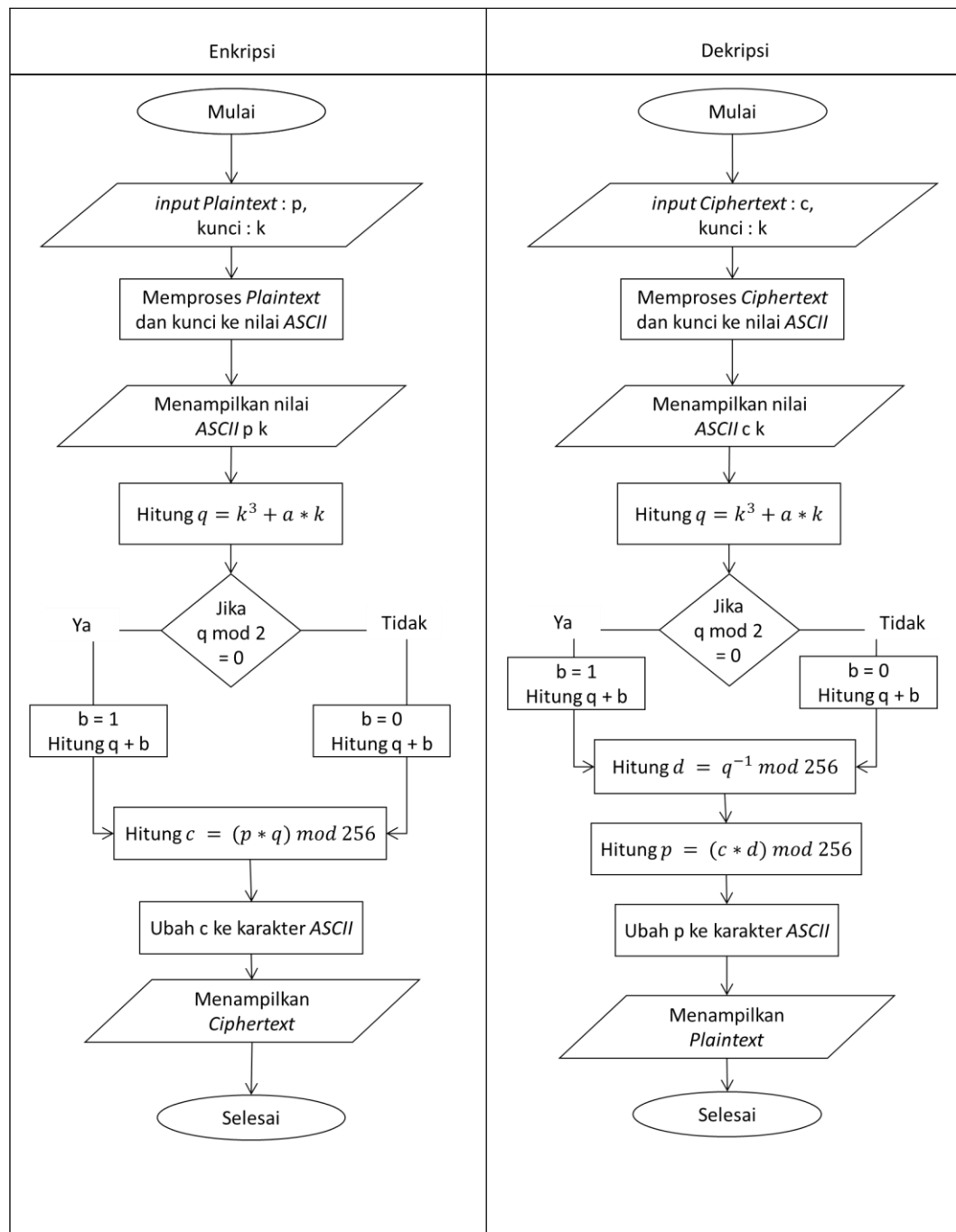
3.3.2 Flowchart Algoritma Multi-Factor RSA



Gambar 3.9 Flowchart Algoritma Multi-Factor RSA

Pada Gambar 3.9 menunjukkan gambaran proses pembangkitan *public key* dan *private key*, enkripsi dan dekripsi menggunakan algoritma *Multi-Factor RSA*.

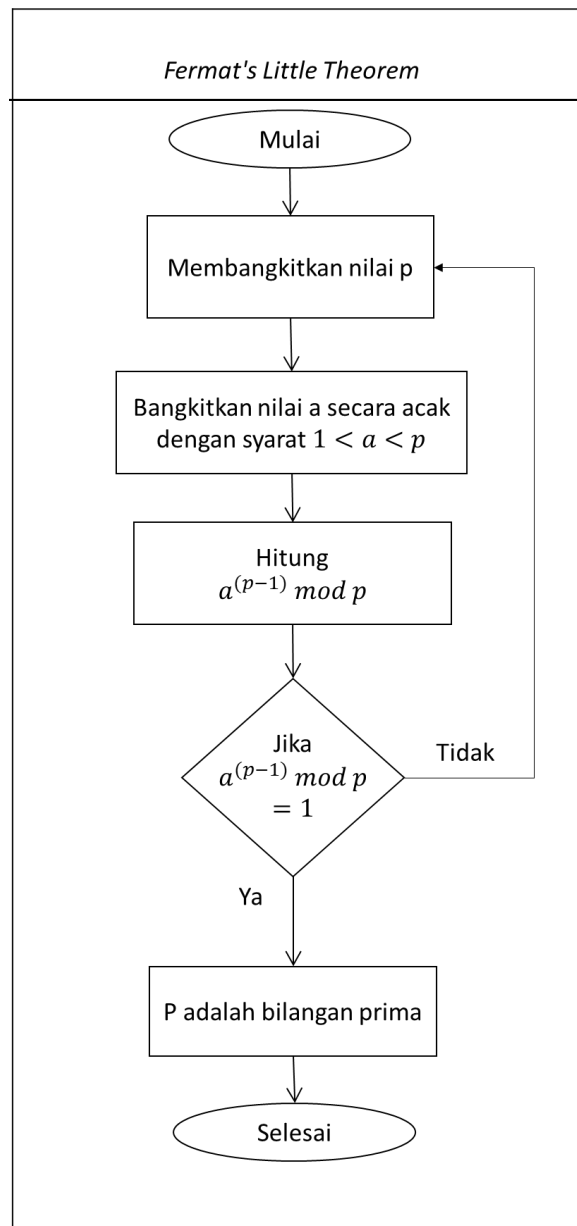
3.3.3 Flowchart Algoritma MSC



Gambar 3.10 Flowchart Algoritma MSC

Pada Gambar 3.10 menunjukkan gambaran proses enkripsi dan dekripsi menggunakan algoritma MSC.

3.3.4 Flowchart Fermat's Little Theorem



Gambar 3.11 Flowchart Fermat's Little Theorem

Pada Gambar 3.11 menunjukkan gambaran proses pembangkitan bilangan prima menggunakan *Fermat's Little Theorem*.

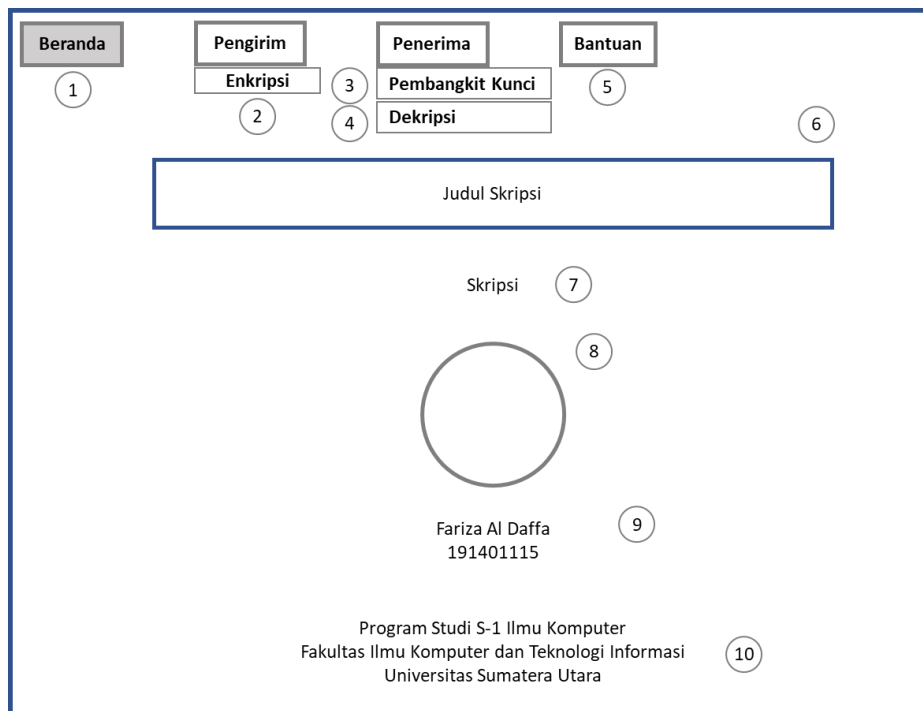
3.4 Perancangan *interface*

Perancangan *interface* merupakan suatu cara yang penting untuk diperhatikan. Interface ialah tampilan antar muka yang menghubungkan user ke dalam sistem. Interface haruslah menarik agar user tertarik untuk menggunakannya. Interface

harus juga mudah dimengerti agar user mudah menggunakan aplikasi. Pada penelitian ini sistem dibuat berbasis Desktop agar mudah di akses dimana saja. Sistem ini memiliki lima halaman yaitu beranda, pembangkit kunci, enkripsi, dekripsi dan bantuan.

3.4.1 Interface Halaman Beranda

Halaman beranda merupakan tampilan awal dari sistem yang akan dibuat.



Gambar 3.12 Halaman Beranda

Keterangan gambar:

1. *Button*, membuka halaman Beranda.
2. *Button*, membuka halaman Enkripsi untuk pengirim.
3. *Button*, membuka halaman Pembangkit Kunci untuk penerima.
4. *Button*, membuka halaman Dekripsi untuk penerima.
5. *Button*, membuka halaman Bantuan.
6. *Label*, berisi Judul Skripsi.
7. *Label*, berisi Skripsi.
8. *PictureBox*, menampilkan logo Universitas.
9. *Label*, berisi nama dan NIM penulis.
10. *Label*, berisi program studi penulis.

3.4.2 Interface Halaman Pengirim

3.4.2.1 Interface Halaman Enkripsi

Pada halaman enkripsi, pengirim akan mengenkripsi *file* teks menggunakan algoritma *MSC* yang nantinya akan menghasilkan *ciphertext*, lalu kunci *MSC* akan dienkripsi dengan *public key Multi-Factor RSA* menjadi *cipherkey*.

Gambar 3.13 Halaman Enkripsi

Keterangan gambar:

1. Label, Enkripsi file.
2. Label, File Teks.
3. *TextBox*, berisi nama *file* teks.
4. *Button*, untuk membuka *dialog* pemilihan *file* teks.
5. Label, Plaintext.
6. *RichTextBox*, menampilkan isi dari *file* teks.
7. Label, Kunci.
8. *RichTextBox*, menampilkan kunci simetris.
9. *Button*, untuk membangkitkan kunci simetris secara acak.
10. *Button*, untuk melakukan enkripsi *file* teks.
11. *Button*, untuk menyimpan kunci simetris.
12. Label, Ciphertext.

13. *RichTextBox*, menampilkan isi dari *ciphertext*.
14. *Button*, untuk menyimpan *ciphertext*.
15. *Label*, Enkripsi kunci.
16. *Label*, Kunci.
17. *TextBox*, berisi kunci simetris.
18. *Label*, Kunci Publik.
19. *RichTextBox*, menampilkan isi kunci publik.
20. *Button*, untuk membuka *dialog* pemilihan *file* kunci publik.
21. *Button*, untuk melakukan enkripsi kunci simetris.
22. *Label*, Cipherkey.
23. *RichTextBox*, menampilkan isi dari *cipherkey*.
24. *Button*, untuk menyimpan *cipherkey*.

3.4.3 Interface Halaman Penerima

3.4.3.1 Interface Halaman Pembangkit Kunci

Pada halaman Pembangkit Kunci, penerima akan membangkitkan *public key* dan *private key* untuk digunakan dalam proses enkripsi dan dekripsi *file*.

Gambar 3.14 Halaman Pembangkit Kunci

Keterangan gambar:

1. *Label*, Nilai p1.

2. *Label*, Nilai p_2 .
3. *Label*, Nilai p_3 .
4. *TextBox*, berisi nilai p_1 .
5. *TextBox*, berisi nilai p_2 .
6. *TextBox*, berisi nilai p_3 .
7. *Label*, Nilai n .
8. *Label*, Nilai e .
9. *Label*, Nilai $\phi(n)$.
10. *TextBox*, berisi nilai n .
11. *TextBox*, berisi nilai e .
12. *TextBox*, berisi nilai $\phi(n)$.
13. *Label*, *private key*.
14. *Label*, Nilai d .
15. *Label*, Nilai n .
16. *TextBox*, berisi nilai d .
17. *TextBox*, berisi nilai n .
18. *Label*, *Public Key*.
19. *Label*, Nilai e .
20. *Label*, Nilai n .
21. *TextBox*, berisi nilai e .
22. *TextBox*, berisi nilai n .
23. *Button*, sebagai pembangkit kunci publik dan kunci privat.
24. *Button*, untuk *reset* isi dari semua *textbox*.
25. *Button*, untuk menyimpan kunci publik dan kunci privat.

3.4.3.2 Interface Dekripsi

Pada halaman ini penerima akan mendekripsi *cipherkey* dengan menggunakan *private key Multi-Factor RSA* yang menghasilkan kunci simetris (*MSC*). Kunci simetris digunakan untuk mendekripsi *ciphertext* menjadi *plaintext* agar isi *file* teks dapat dibaca.

Gambar 3.15 Halaman Dekripsi

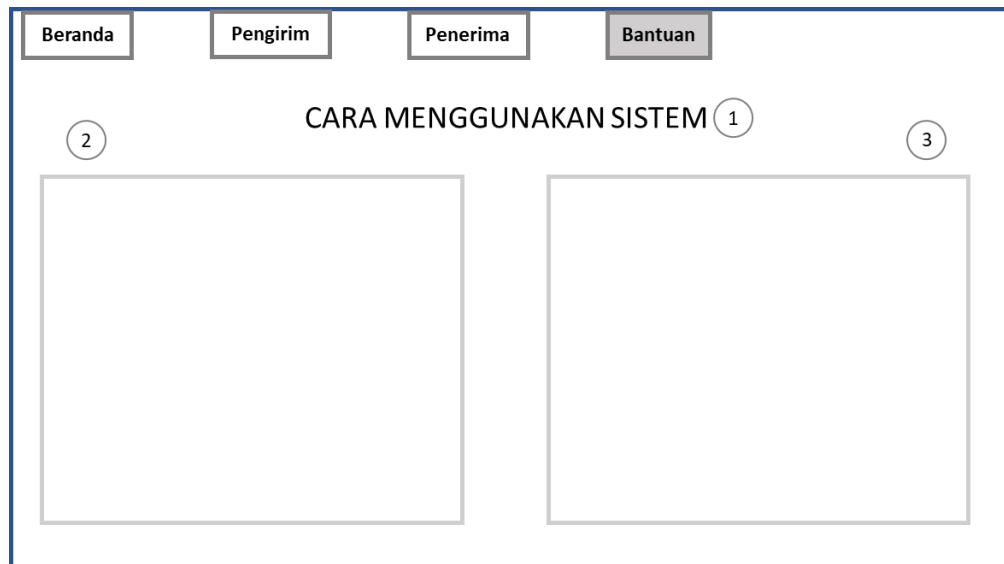
Keterangan gambar:

1. *Label*, Dekripsi kunci.
2. *Label*, Cipherkey.
3. *TextBox*, menampilkan isi cipherkey.
4. *Button*, untuk membuka *dialog* pemilihan *file* cipherkey.
5. *Label*, Kunci Privat.
6. *RichTextBox*, menampilkan isi kunci privat.
7. *Button*, untuk membuka *dialog* pemilihan *file* kunci privat.
8. *Button*, untuk mendekripsi *cipherkey*.
9. *Label*, Kunci.
10. *RichTextBox*, menampilkan kunci simetris yang sudah didekripsi.
11. *Button*, untuk menyimpan kunci simetris.
12. *Label*, Dekripsi *file*.
13. *Label*, *File* teks.
14. *TextBox*, berisi nama *file ciphertext*.
15. *Button*, untuk membuka *dialog* pemilihan *file ciphertext*.
16. *Label*, *Ciphertext*.

17. *RichTextBox*, menampilkan isi *file ciphertext*.
18. *Label*, Kunci.
19. *TextBox*, menampilkan isi kunci simetris.
20. *Button*, untuk membuka *dialog* pemilihan *file* kunci simetris.
21. *Button*, untuk mendekripsi *ciphertext*.
22. *Label*, *Plaintext*.
23. *RichTextBox*, menampilkan *plaintext* setelah didekripsi.
24. *Button*, untuk menyimpan *plaintext*.

3.4.4 Interface Halaman Bantuan

Pada halaman Bantuan terdapat panduan dalam menggunakan sistem, termasuk cara menggunakan pembangkit kunci, enkripsi, dan dekripsi.



Gambar 3.16 Halaman Bantuan

Keterangan gambar :

1. *Label*, CARA MENGGUNAKAN SISTEM.
2. *Label*, Panduan untuk pengirim.
3. *Label*, Panduan untuk penerima.

BAB 4

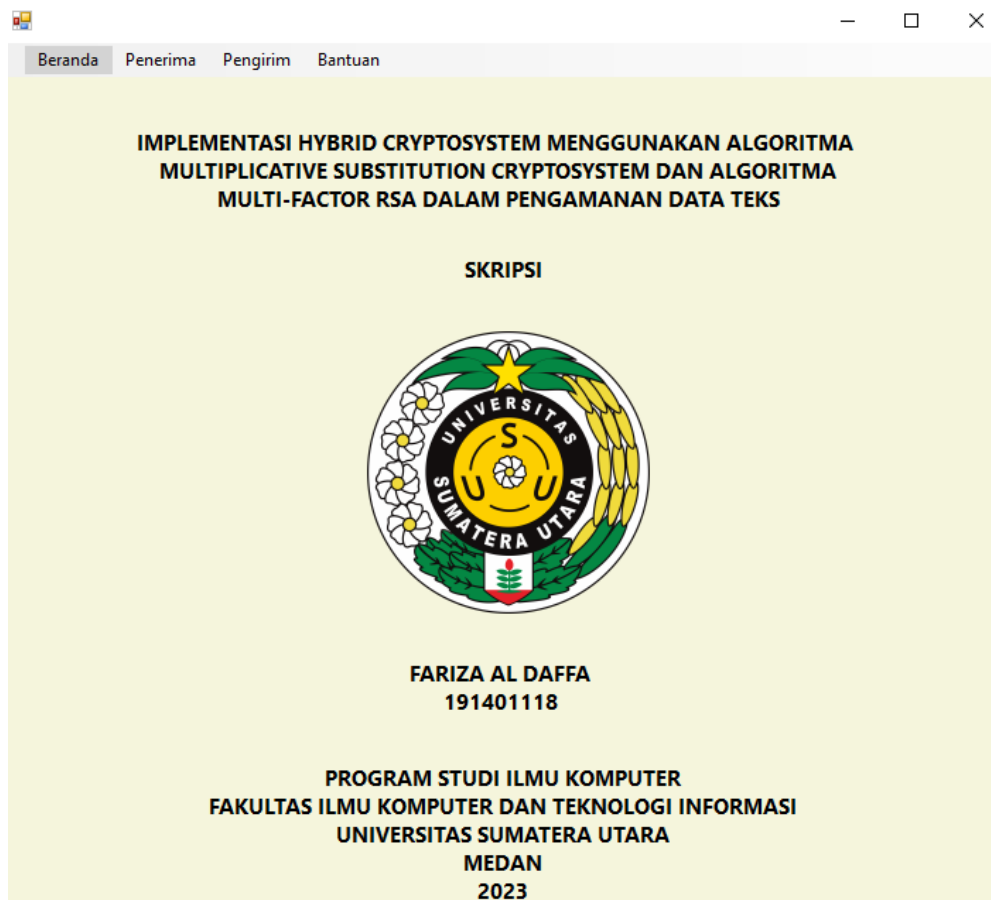
IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Sistem

Pada tahap ini membahas implementasi sistem yang merupakan hasil analisis dan perancangan sistem ke dalam bahasa pemrograman. Sistem dibangun menggunakan bahasa program *C#*, dimana sistem ini memiliki lima *form* yaitu halaman beranda, halaman pembangkit kunci, halaman enkripsi, halaman dekripsi, dan halaman bantuan

4.1.1 Halaman Beranda

Halaman yang pertama kali muncul saat pengguna menjalankan sistem. Pada halaman ini berisi judul skripsi, nama dan NIM penulis, logo, dan instansi penulis.



Gambar 4.1 Halaman Beranda

4.1.2 Halaman Pembangkit Kunci

Halaman Pembangkit Kunci menampilkan proses pembangkitan kunci *Multi-Factor RSA*. Pembangkitan kunci menggunakan nilai-nilai yang akan dihasilkan secara acak, kemudian diproses untuk menghasilkan kunci privat dan kunci publik. Halaman ini memiliki tombol *Reset* yang berfungsi untuk menghapus semua nilai setelah proses pembangkitan kunci selesai. Setelah pengguna berhasil membangkitkan kunci, pengguna dapat menyimpan kunci publik dan kunci privat tersebut.

Gambar 4.2 Halaman Pembangkit Kunci

4.1.3 Halaman Enkripsi

Pada halaman ini terdapat dua proses enkripsi, yaitu bagian pertama untuk enkripsi *file* teks dan bagian kedua untuk enkripsi kunci simetris. Pertama, pengguna memilih *file* teks yang akan dienkripsi. Kemudian, bangkitkan kunci simetris secara acak dengan menekan tombol "bangkitkan" untuk mendapatkan kunci simetris acak. Selanjutnya, pengguna menyimpan kunci simetris terlebih dahulu untuk proses enkripsi selanjutnya menggunakan kunci publik dengan menekan tombol "simpan". Setelah *file* teks dan kunci simetris terunggah, pengguna menekan tombol "enkripsi" untuk mengenkripsi teks tersebut dengan kunci simetris. Proses enkripsi akan berjalan dan menghasilkan *ciphertext*, yang kemudian dapat disimpan

oleh pengguna. Selanjutnya, pengguna mengunggah kunci simetris yang telah disimpan sebelumnya dengan mengeklik tombol "*browse*" dan juga memasukkan kunci publik yang telah dibangkitkan sebelumnya. Pengguna kemudian mengenkripsi kunci simetris menggunakan kunci publik dengan menekan tombol "enkripsi" untuk mendapatkan *cipherkey*. Selanjutnya, *cipherkey* akan disimpan dengan menekan tombol "simpan".

Gambar 4.3 Halaman Enkripsi

4.1.4 Halaman Dekripsi

Pada halaman ini terdapat dua proses, yaitu proses untuk dekripsi kunci simetris dan proses untuk dekripsi file *ciphertext*. Pertama, pengguna memasukkan *cipherkey* yang telah dienkripsi bersama dengan kunci privat. Kemudian, tekan tombol "dekripsi" yang nantinya akan menghasilkan kunci simetris. Hasil dari dekripsi kunci akan disimpan oleh pengguna. Selanjutnya, pengguna memasukkan kembali *ciphertext* yang akan dienkripsi menggunakan kunci simetris yang dihasilkan sebelumnya. Proses ini akan mengembalikan *plaintext* seperti semula, dan tekan tombol "simpan" untuk menyimpan hasil dekripsi *ciphertext*.

Gambar 4.4 Halaman Dekripsi

4.1.5 Halaman Bantuan

Halaman ini berguna dalam memberikan panduan langkah-langkah untuk menggunakan aplikasi. Halaman ini berisi informasi yang berguna bagi pengguna untuk memahami dan mengoperasikan sistem dengan benar.

CARA MENGGUNAKAN APLIKASI

Pembangkit Kunci :

1. Buka menu Pembangkit Kunci
2. Tekan tombol bangkitkan Kunci untuk membangkitkan kunci secara otomatis
3. Simpan kunci public dan kunci private dengan menekan tombol simpan kunci

Enkripsi :

1. Buka menu enkripsi
2. Pada bagian enkripsi file, tekan tombol browse untuk mengupload file berekstensi .txt yang akan dienkripsi
3. Isi dari file (plaintext) akan muncul pada textbox plaintext
4. Bangkitkan kunci simetris dengan menekan tombol bangkitkan
5. Simpan kunci simetris yang sudah dibangkitkan dengan menekan tombol simpan pada bagian enkripsi file.
6. Enkripsi file menggunakan kunci simetris dengan menekan tombol enkripsi pada bagian enkripsi file.
7. Hasil enkripsi akan muncul pada textbox ciphertext, klik simpan untuk menyimpan ciphertext.
8. Pada bagian enkripsi kunci, tekan tombol browse untuk mengupload kunci simetris yang sudah disimpan sebelumnya.
9. Lalu tekan browse untuk mengupload public key.
10. Enkripsi kunci simetris dengan menekan tombol enkripsi pada bagian enkripsi kunci.
11. Cipherkey akan muncul pada textbox cipherkey.
12. Tekan simpan untuk menyimpan cipherkey.

Dekripsi :

1. Pilih menu dekripsi
2. Pada bagian dekripsi kunci, tekan tombol browse untuk mengupload cipherkey.
3. Tekan tombol browse private key setelahnya untuk mengupload private key yang sudah disimpan sebelumnya.
4. Dekripsi cipherkey dengan menekan tombol dekripsi.
5. Kunci simetris akan muncul pada textbox kunci.
6. Simpan kunci simetris dengan menekan tombol simpan.
7. Pada bagian dekripsi file, tekan tombol browse untuk mengupload file Ciphertext.
8. Ciphertext akan muncul pada textbox ciphertext.
9. Upload kunci simetris yang sudah didekripsi sebelumnya dengan menekan tombol browse.
10. Dekripsi file ciphertext menggunakan kunci simetris dengan menekan tombol dekripsi.
11. Plaintext semula akan muncul pada textbox plaintext.
12. Tekan simpan untuk menyimpan file plaintext.

Gambar 4.5 Halaman Bantuan

4.2 Pengujian Sistem

Pengujian sistem dilakukan untuk menguji apakah aplikasi dapat melakukan proses enkripsi dan dekripsi *file* teks dengan menggunakan algoritma *MSC* dan algoritma *Multi-Factor RSA*. Berikut adalah kriteria pengujian sistem:

1. *File* diuji yaitu berekstensi *.txt.
2. *Ciphertext* proses enkripsi maupun dekripsi berekstensi *.ciphertext dan *Cipherkey* proses enkripsi maupun dekripsi berekstensi *.cipherkey.
3. Kunci publik disimpan dengan ekstensi *.publickey, kunci privat disimpan dengan ekstensi *.privatekey dan kunci simetris akan disimpan dengan ekstensi *.sym.
4. Pada analisis *running time* dimulai saat proses enkripsi dan dekripsi dilakukan dengan satuan *millisecond (ms)*.
5. Pengujian sistem menggunakan *personal computer*, spesifikasi *Processor* Intel(R) Core(TM) i3-6006U CPU @ 2.00GHz 1.99 GHz, RAM 4,00GB.

4.2.1 Pengujian Pembangkit Kunci *Multi-Factor RSA*

Pada halaman ini , pengguna menekan button Bangkitkan Kunci untuk membangkitkan seluruh nilai bilangan acak, kunci privat dan kunci publik, lalu pengguna menyimpan masing-masing kunci dengan menekan button simpan kunci.

Penerima	
Nilai p1 :	941
Nilai p2 :	421
Nilai p3 :	367
Nilai n :	145391087
Nilai e :	57629461
Nilai $\phi(n)$:	144496800

Private Key		Public Key	
Nilai d :	64010941	Nilai e :	57629461
Nilai n :	145391087	Nilai n :	145391087

Gambar 4.6 Pengujian Pembangkit Kunci *Multi-Factor RSA*

Ketika *button* Bangkitkan Kunci ditekan, maka sistem akan membangkitkan nilai $p_1, p_2, p_3, e, \phi(n), n, d$. Nilai e dan n adalah *public key* sedangkan nilai d dan n adalah *private key*. Dalam membuktikan apakah sistem sudah sesuai dapat dilakukan dengan perhitungan manual berikut:

$$1. \quad p_1 = 941 ; p_2 = 421 ; p_3 = 367$$

Nilai $p_1 p_2 p_3$, p adalah bilangan prima yang dibangkitkan secara acak menggunakan *Fermat's Little Theorem*.

$$2. \quad \text{Hitung } n = p_1 * p_2 * p_3$$

$$n = 941 * 421 * 367$$

$$n = 145391087$$

$$3. \quad \text{Hitung } \phi(n) = (p_1 - 1) * (p_2 - 1) * (p_3 - 1)$$

$$\phi(n) = (p_1 - 1) * (p_2 - 1) * (p_3 - 1)$$

$$= 940 * 420 * 366$$

$$= 144496800$$

$$4. \quad \text{Bangkitkan nilai } e, \text{ dimana } e \text{ harus } \textit{relative} \text{ prima dengan } \phi(n).$$

$$e = 57629461$$

$$\text{GCD}(e, \phi(n))$$

$$\text{GCD}(57629461, 144496800) = 1$$

$$5. \quad \text{Hasilkan nilai } d \text{ yang merupakan invers dari } e \text{ mod } \phi(n).$$

$$d = e^{-1} \text{ mod } \phi(n)$$

$$d = 57629461^{-1} \text{ mod } 144496800$$

$$d = 64010941$$

4.2.2 Pengujian Enkripsi

Setelah pengguna berhasil membangkitkan kunci privat dan kunci publik, pengguna memasuki halaman enkripsi untuk mengenkripsi *file* teks dan juga untuk membangkitkan kunci simetris secara acak.

Gambar 4.7 Pengujian enkripsi *plaintext*

Dapat dilihat pada Gambar 4.7 menunjukkan bahwa pengguna telah memasukkan *file* teks dan menghasilkan kunci simetris secara acak. Selanjutnya, pengguna menekan tombol "enkripsi" dan sistem menghasilkan *ciphertext*. Untuk membuktikan apakah sistem sudah sesuai, dapat dilakukan perhitungan manual berikut:

1. *Plaintext* : Ilmu Komputer
Kunci : Yc0hKIqwF
2. Sesuaikan Panjang kunci sama dengan plaintext dengan melakukan perulangan. Kunci : Yc0hKIqwFYc0h
3. Ubah karakter *Plaintext* p dan *Key* k menjadi nilai *ASCII*.

p : *Ilmu Komputer*

p : 73 108 109 117 32 75 111 109 112 117 116 101 114

k : Yc0hKIqwFYc0h

k : 89 99 48 104 75 73 113 119 70 89 99 48 104

4. Hitung $q_i = k^3 + a * k$, dimana $a = i$.

$$q_1 = 89^3 + 1 * 89 = 705058$$

$$q_2 = 99^3 + 2 * 99 = 970497$$

$$q_3 = 48^3 + 3 * 48 = 110736$$

$$q_4 = 104^3 + 4 * 104 = 1125280$$

$$q_5 = 75^3 + 5 * 75 = 422250$$

$$\begin{aligned}
q_6 &= 73^3 + 6 * 73 = 389455 \\
q_7 &= 113^3 + 7 * 113 = 1443688 \\
q_8 &= 119^3 + 8 * 119 = 1686111 \\
q_9 &= 70^3 + 9 * 70 = 343630 \\
q_{10} &= 89^3 + 10 * 89 = 705859 \\
q_{11} &= 99^3 + 11 * 99 = 971388 \\
q_{12} &= 48^3 + 12 * 48 = 111168 \\
q_{13} &= 104^3 + 13 * 104 = 1126216
\end{aligned}$$

5. Jika q genap, maka $q + 1$.

$$\begin{aligned}
q_1 &= 705058 + 1 = 705059 \\
q_2 &= 970497 \\
q_3 &= 110736 + 1 = 110737 \\
q_4 &= 1125280 + 1 = 1125281 \\
q_5 &= 422250 + 1 = 422251 \\
q_6 &= 389455 \\
q_7 &= 1443688 + 1 = 1443689 \\
q_8 &= 1686111 \\
q_9 &= 343630 + 1 = 343631 \\
q_{10} &= 705859 \\
q_{11} &= 971388 + 1 = 971389 \\
q_{12} &= 111168 + 1 = 111169 \\
q_{13} &= 1126216 + 1 = 1126217
\end{aligned}$$

6. Hitung $c = p * q \bmod 256$

$$\begin{aligned}
c_1 &= 73 * 705059 \bmod 256 = 251 \\
c_2 &= 108 * 970497 \bmod 256 = 108 \\
c_3 &= 109 * 110737 \bmod 256 = 189 \\
c_4 &= 117 * 1125281 \bmod 256 = 149 \\
c_5 &= 32 * 422251 \bmod 256 = 96 \\
c_6 &= 75 * 389455 \bmod 256 = 37 \\
c_7 &= 111 * 1443689 \bmod 256 = 135
\end{aligned}$$

$$c_8 = 109 * 1686111 \bmod 256 = 115$$

$$c_9 = 112 * 343631 \bmod 256 = 144$$

$$c_{10} = 117 * 705859 \bmod 256 = 159$$

$$c_{11} = 116 * 971389 \bmod 256 = 164$$

$$c_{12} = 101 * 111169 \bmod 256 = 165$$

$$c_{13} = 114 * 1126217 \bmod 256 = 130$$

7. Hasil *Ciphertext* adalah

$$c = 251\ 108\ 189\ 149\ 96\ 37\ 135\ 115\ 144\ 159\ 164\ 165\ 130$$

Setelah pengirim mengenkripsi *plaintext* dan menyimpannya, selanjutnya pengirim akan mengenkripsi kunci simetris *MSC* menggunakan kunci publik *Multi-Factor RSA*.

Gambar 4.8 Pengujian enkripsi kunci

Gambar 4.8 menunjukkan bahwa *user* telah menginput kunci simetris beserta kunci publik, lalu *user* menekan button enkripsi dan sistem akan menghasilkan *cipherkey*. Untuk membuktikan apakah sistem sudah sesuai maka dilakukan perhitungan manual dengan langkah-langkah sebagai berikut:

1. Input kunci *MSC*.

$$m = \text{"Yc0hKlqwF"}$$

Lalu ubah ke nilai *ASCII*,

$$m = 89\ 99\ 48\ 104\ 75\ 73\ 113\ 119\ 70$$

2. Ambil kunci publik yaitu *n* dan *e*.

$$n = 145391087 \quad e = 57629461$$

3. Gabung setiap 3 karakter kunci

Proses 1

$$0 * 256 + 89 = 89$$

$$89 * 256 + 99 = 22883$$

$$22883 * 256 + 48 = 5858096$$

Proses 2

$$0 * 256 + 104 = 104$$

$$104 * 256 + 75 = 26699$$

$$26699 * 256 + 73 = 6835017$$

Proses 3

$$0 * 256 + 113 = 113$$

$$113 * 256 + 119 = 29047$$

$$29047 * 256 + 70 = 7436102$$

4. Enkripsi pesan dengan rumus $c = m^e \pmod{n}$

$$\begin{aligned} c_1 &= 5858096^{57629461} \pmod{145391087} \\ &= 25224938 \end{aligned}$$

$$\begin{aligned} c_2 &= 6835017^{57629461} \pmod{145391087} \\ &= 55903409 \end{aligned}$$

$$\begin{aligned} c_3 &= 7436102^{57629461} \pmod{145391087} \\ &= 106978184 \end{aligned}$$

$$\text{Maka } c = 25224938 \ 55903409 \ 106978184$$

Kirim c ke penerima yang dituju.

4.2.3 Pengujian Dekripsi

Pada halaman dekripsi, penerima akan melakukan dekripsi pada kunci untuk membuka *file* teks menggunakan kunci privat yang dimilikinya. Setelah kunci simetris berhasil didekripsi, penerima akan melanjutkan dengan melakukan dekripsi pada *file* teks menggunakan kunci simetris yang telah didekripsi sebelumnya.

Gambar 4.9 Pengujian dekripsi kunci

Pada Gambar 4.9 memperlihatkan bahwa pengguna telah menginput *cipherkey* beserta kunci privat, lalu pengguna melakukan dekripsi *cipherkey* dan sistem akan menghasilkan kembali kunci simetris. Untuk membuktikan apakah sistem sudah sesuai dapat dilakukan perhitungan manual sebagai berikut:

1. Ambil *cipherkey*, $c = 25224938 \ 55903409 \ 106978184$

$$d = 64010941 \quad n = 145391087$$

2. Dekripsi pesan dengan rumus $m = c^d \pmod{n}$

$$\begin{aligned} m_1 &= 494^{64010941} \pmod{145391087} \\ &= 5858096 \end{aligned}$$

$$\begin{aligned} m_2 &= 494^{64010941} \pmod{145391087} \\ &= 6835017 \end{aligned}$$

$$\begin{aligned} m_3 &= 494^{64010941} \pmod{145391087} \\ &= 7436102 \end{aligned}$$

3. Pisah setiap 3 karakter kunci.

Proses 1

$$5858096 \pmod{256} = 48 ; 5858096 \div 256 = 22883$$

$$22883 \pmod{256} = 99 ; 22883 \div 256 = 89$$

$$89 \pmod{256} = 89$$

$$m_1 = 89 \ 99 \ 48$$

Proses 2

$$6835017 \bmod 256 = 73 ; 6835017 \div 256 = 26699$$

$$26699 \bmod 256 = 75 ; 26699 \div 256 = 104$$

$$104 \bmod 256 = 104$$

$$m_2 = 104\ 75\ 73$$

Proses 3

$$7436102 \bmod 256 = 70 ; 7436102 \div 256 = 29047$$

$$29047 \bmod 256 = 119 ; 29047 \div 256 = 113$$

$$113 \bmod 256 = 113$$

$$m_3 = 113\ 119\ 70$$

$$m = 89\ 99\ 48\ 104\ 75\ 73\ 113\ 119\ 70$$

4. Ubah m menjadi karakter *ASCII*.

$$m = \text{"Yc0hKlqwF"}$$

Setelah penerima mendekripsi *cipherkey* dan menyimpannya, selanjutnya penerima akan mendekripsi *ciphertext* menggunakan kunci simetris *MSC* yang sudah didekripsi sebelumnya.

The image shows a software interface for decryption, divided into two main sections: 'Penerima' (Receiver) and 'Dekripsi File' (Decrypt File).

Penerima Section:

- Dekripsi Kunci:** A section for decrypting the key. It includes a 'Cipherkey' input field with a dropdown menu showing '25224938 55903409' and '106978184', a 'Browse' button, and a 'Kunci Private' input field with a dropdown menu showing '64010941' and '145391087', also with a 'Browse' button.
- Dekripsi:** A button to perform the decryption.
- Kunci:** An input field showing the result 'Yc0hKlqwF'.
- Running Time:** A display showing '0.9301' and a 'Simpan' (Save) button.

Dekripsi File Section:

- File Teks:** An input field showing 'ciphertext.ciphertext' and a 'Browse' button.
- Ciphertext:** An input field showing the ciphertext '251 108 189 149 96 37 135 115 144 159 164 165 130'.
- Kunci:** An input field showing the key 'Yc0hKlqwF' and a 'Browse' button.
- Dekripsi:** A button to perform the file decryption.
- Plaintext:** An input field showing the result 'Ilmu Komputer'.
- Running Time:** A display showing '2.4727' and a 'Reset' button.
- Simpan:** A button to save the result.

Gambar 4.10 Pengujian dekripsi *ciphertext*

Pada Gambar 4.10 memperlihatkan bahwa user telah menginput *ciphertext* beserta kunci simetris yang sudah didekripsi sebelumnya, pengguna melakukan dekripsi *ciphertext* dan sistem akan menghasilkan kembali *plaintext*. Untuk membuktikan apakah sistem sudah sesuai dapat dilakukan perhitungan manual dengan langkah-langkah sebagai berikut:

1. *Ciphertext*: 251 108 189 149 96 37 135 115 144 159 164 165 130
Kunci : Yc0hKIqwF
2. Sesuaikan Panjang kunci sama dengan *ciphertext* dengan melakukan perulangan. Kunci : Yc0hKIqwFYc0h
3. Ubah karakter kunci k menjadi nilai *ASCII*
k : Yc0hKIqwFYc0h
k : 89 99 48 104 75 73 113 119 70 89 99 48 104
4. Hitung $q_i = k^3 + a * k$, dimana $a = i$.

$$q_1 = 89^3 + 1 * 89 = 705058$$

$$q_2 = 99^3 + 2 * 99 = 970497$$

$$q_3 = 48^3 + 3 * 48 = 110736$$

$$q_4 = 104^3 + 4 * 104 = 1125280$$

$$q_5 = 75^3 + 5 * 75 = 422250$$

$$q_6 = 73^3 + 6 * 73 = 389455$$

$$q_7 = 113^3 + 7 * 113 = 1443688$$

$$q_8 = 119^3 + 8 * 119 = 1686111$$

$$q_9 = 70^3 + 9 * 70 = 343630$$

$$q_{10} = 89^3 + 10 * 89 = 705859$$

$$q_{11} = 99^3 + 11 * 99 = 971388$$

$$q_{12} = 48^3 + 12 * 48 = 111168$$

$$q_{13} = 104^3 + 13 * 104 = 1126216$$
5. Jika q genap, maka $q + 1$.

$$q_1 = 705058 + 1 = 705059$$

$$q_2 = 970497$$

$$q_3 = 110736 + 1 = 110737$$

$$q_4 = 1125280 + 1 = 1125281$$

$$q_5 = 422250 + 1 = 422251$$

$$q_6 = 389455$$

$$q_7 = 1443688 + 1 = 1443689$$

$$q_8 = 1686111$$

$$q_9 = 343630 + 1 = 343631$$

$$q_{10} = 705859$$

$$q_{11} = 971388 + 1 = 971389$$

$$q_{12} = 111168 + 1 = 111169$$

$$q_{13} = 1126216 + 1 = 1126217$$

6. Hitung $(d * q) \bmod 256 = 1$.

$$(d_1 * q_1) \bmod 256 = 1 ; d_1 = 139$$

$$(d_2 * q_2) \bmod 256 = 1 ; d_2 = 1$$

$$(d_3 * q_3) \bmod 256 = 1 ; d_3 = 113$$

$$(d_4 * q_4) \bmod 256 = 1 ; d_4 = 97$$

$$(d_5 * q_5) \bmod 256 = 1 ; d_5 = 67$$

$$(d_6 * q_6) \bmod 256 = 1 ; d_6 = 175$$

$$(d_7 * q_7) \bmod 256 = 1 ; d_7 = 217$$

$$(d_8 * q_8) \bmod 256 = 1 ; d_8 = 159$$

$$(d_9 * q_9) \bmod 256 = 1 ; d_9 = 175$$

$$(d_{10} * q_{10}) \bmod 256 = 1 ; d_{10} = 107$$

$$(d_{11} * q_{11}) \bmod 256 = 1 ; d_{11} = 213$$

$$(d_{12} * q_{12}) \bmod 256 = 1 ; d_{12} = 193$$

$$(d_{13} * q_{13}) \bmod 256 = 1 ; d_{13} = 249$$

$$d = 139 \ 1 \ 113 \ 97 \ 67 \ 175 \ 217 \ 159 \ 175 \ 107 \ 213 \ 193 \ 249$$

7. Hitung $p = (c * d) \bmod 256$.

$$\begin{aligned} p_1 &= (c * d) \bmod 256 = 251 * 139 \bmod 256 \\ &= 73 \end{aligned}$$

$$\begin{aligned} p_2 &= (c * d) \bmod 256 = 108 * 1 \bmod 256 \\ &= 108 \end{aligned}$$

$$p_3 = (c * d) \bmod 256 = 189 * 113 \bmod 256$$

$$\begin{aligned}
&= 109 \\
p_4 &= (c * d) \bmod 256 = 149 * 97 \bmod 256 \\
&= 117 \\
p_5 &= (c * d) \bmod 256 = 96 * 67 \bmod 256 \\
&= 32 \\
p_6 &= (c * d) \bmod 256 = 37 * 175 \bmod 256 \\
&= 75 \\
p_7 &= (c * d) \bmod 256 = 135 * 217 \bmod 256 \\
&= 111 \\
p_8 &= (c * d) \bmod 256 = 115 * 159 \bmod 256 \\
&= 109 \\
p_9 &= (c * d) \bmod 256 = 144 * 175 \bmod 256 \\
&= 112 \\
p_{10} &= (c * d) \bmod 256 = 159 * 107 \bmod 256 \\
&= 117 \\
p_{11} &= (c * d) \bmod 256 = 164 * 213 \bmod 256 \\
&= 116 \\
p_{12} &= (c * d) \bmod 256 = 165 * 193 \bmod 256 \\
&= 101 \\
p_{13} &= (c * d) \bmod 256 = 130 * 249 \bmod 256 \\
&= 114 \\
p &: 73\ 108\ 109\ 117\ 32\ 75\ 111\ 109\ 112\ 117\ 116\ 101\ 114
\end{aligned}$$

8. Ubah p menjadi karakter *ASCII*

$$\begin{aligned}
p &= 73\ 108\ 109\ 117\ 32\ 75\ 111\ 109\ 112\ 117\ 116\ 101\ 114 \\
p &= \text{Ilmu Komputer}
\end{aligned}$$

4.3 Waktu proses

Tahap ini merupakan perhitungan waktu yang digunakan dalam proses enkripsi dan dekripsi dari *file* teks. Sistem dinilai dengan memperhatikan variasi panjang karakter teks yang akan dienkripsi atau didekripsi, dan waktu proses dihitung dalam satuan *milisecond (ms)*.

4.3.1 Waktu Proses Enkripsi *File* Teks

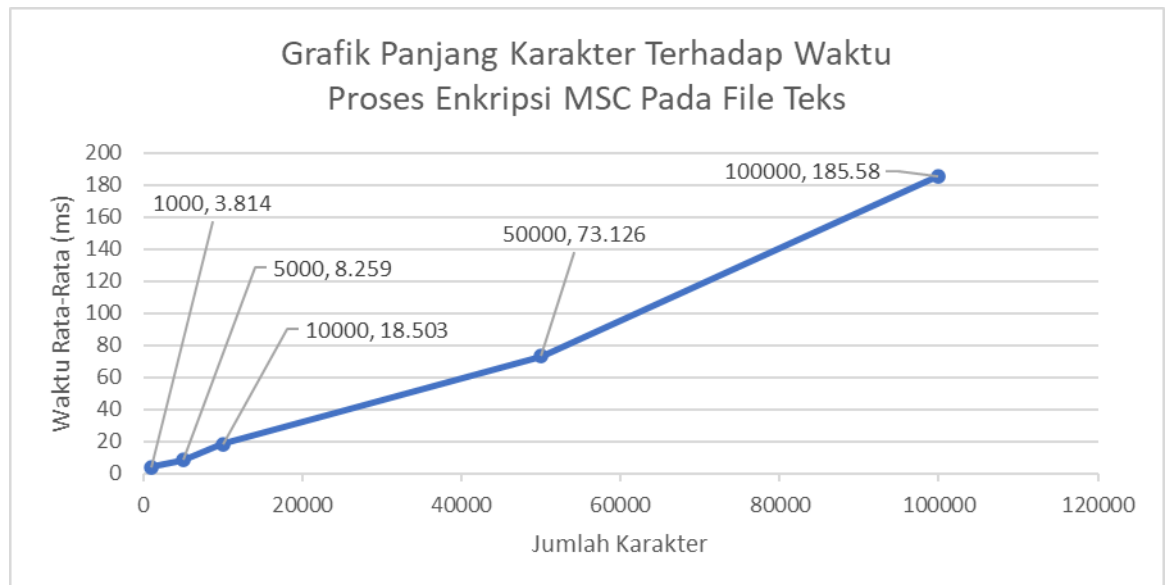
Perhitungan waktu proses enkripsi pada *file* teks dilakukan dengan menghitung waktu yang dibutuhkan untuk mengenkripsi menggunakan algoritma *MSC*. Tabel

4.1 menunjukkan hasil dari pengujian waktu proses untuk enkripsi *file* teks, serta waktu rata-rata untuk setiap jumlah karakter.

Tabel 4.1 Hasil Waktu Uji Coba Enkripsi *File* Teks

Jumlah Karakter	Uji Coba	Waktu Proses (ms)	Waktu Rata – rata (ms)
1000 karakter	1	6.933	3.814
	2	2.154	
	3	3.658	
	4	2.5796	
	5	3.7472	
5000 karakter	1	11.1956	8.259
	2	10.6258	
	3	6.4739	
	4	6.5173	
	5	6.4866	
10000 karakter	1	31.6005	18.503
	2	16.9207	
	3	14.7946	
	4	15.4765	
	5	13.7264	
50000 karakter	1	81.8638	73.126
	2	65.0839	
	3	70.1772	
	4	77.3138	
	5	71.195	
100000 karakter	1	209.42	185.580
	2	192.1811	
	3	180.1228	
	4	182.0039	
	5	164.1748	

Gambar 4.11 berikut merupakan penggambaran dari rata-rata waktu enkripsi dengan jumlah karakter.



Gambar 4.11 Grafik Proses Enkripsi MSC

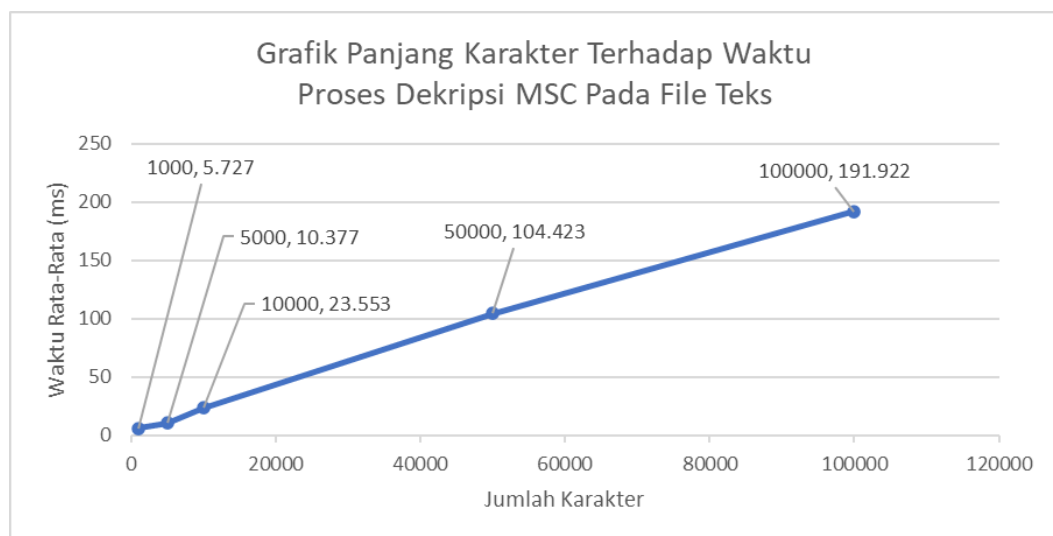
4.3.2 Waktu Proses Dekripsi *File* Teks

Perhitungan waktu proses dekripsi pada *file* teks dilakukan dengan menghitung waktu yang dibutuhkan untuk mendekripsi menggunakan algoritma *MSC*. Tabel 4.2 menunjukkan hasil dari pengujian waktu proses untuk dekripsi *file* teks, serta waktu rata-rata untuk setiap jumlah karakter.

Tabel 4.2 Hasil Waktu Uji Coba Dekripsi *File Teks*

Jumlah Karakter	Uji Coba	Waktu Proses (ms)	Waktu Rata – rata (ms)
1000 karakter	1	15.5814	5.727
	2	4.4955	
	3	4.9141	
	4	1.8488	
	5	1.7976	
5000 karakter	1	12.1386	10.377
	2	8.0809	
	3	10.6904	
	4	12.8164	
	5	8.1628	
10000 karakter	1	38.7769	23.553
	2	17.4019	
	3	21.6148	
	4	18.0519	
	5	21.9206	
50000 karakter	1	160.2372	104.423
	2	86.7317	
	3	102.7685	
	4	85.8851	
	5	86.4969	
100000 karakter	1	248.1547	191.922
	2	205.9729	
	3	168.32	
	4	161.4958	
	5	175.6672	

Gambar 4.12 berikut merupakan penggambaran dari rata-rata waktu dekripsi dengan jumlah karakter.

**Gambar 4.12** Grafik Proses Dekripsi *MSC*

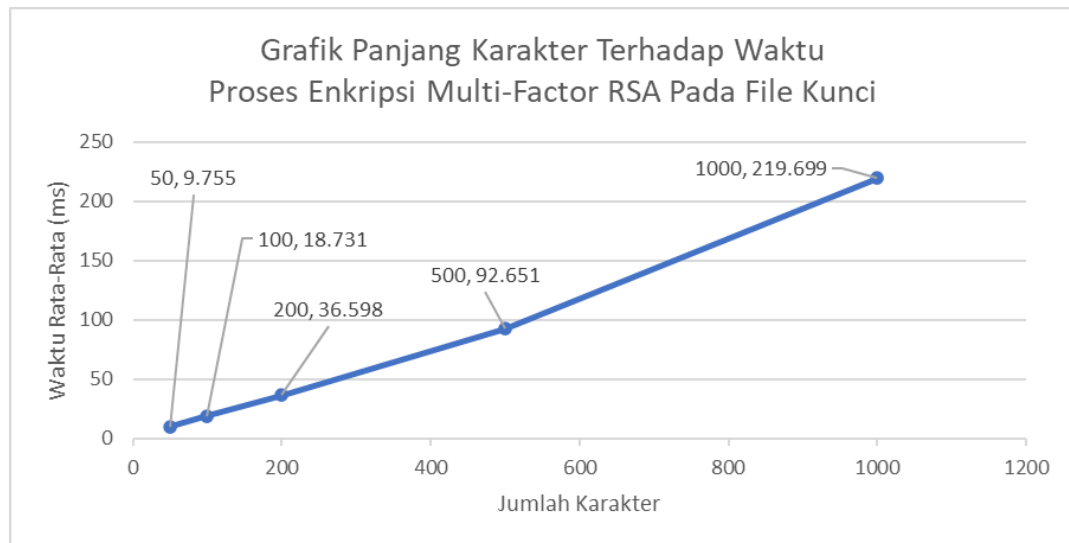
4.3.3 Waktu Proses Enkripsi *File* Kunci

Pada enkripsi *file* kunci dilakukan proses menghitung waktu yang dibutuhkan untuk mengenkripsi menggunakan algoritma *Multi-Factor RSA*. Tabel 4.3 menunjukkan hasil dari pengujian waktu proses untuk enkripsi *file* kunci, serta waktu rata-rata untuk setiap jumlah karakter.

Tabel 4.3 Hasil Waktu Uji Coba Enkripsi *File* Kunci

Jumlah Karakter	Uji Coba	Waktu Proses (ms)	Waktu Rata – rata (ms)
50 karakter	1	7.6168	9.755
	2	9.2452	
	3	7.5329	
	4	9.2612	
	5	15.122	
100 karakter	1	17.1015	18.731
	2	13.3636	
	3	16.7727	
	4	27.9841	
	5	18.4378	
200 karakter	1	36.784	36.598
	2	41.6141	
	3	42.6704	
	4	33.4037	
	5	28.5221	
500 karakter	1	89.0244	92.651
	2	96.8017	
	3	98.5085	
	4	93.9774	
	5	84.9446	
1000 karakter	1	208.6222	219.699
	2	254.2844	
	3	227.1287	
	4	208.0633	
	5	200.3976	

Gambar 4.13 berikut merupakan penggambaran dari rata-rata waktu enkripsi dengan jumlah karakter.



Gambar 4.13 Grafik Proses Enkripsi *Multi-Factor RSA*

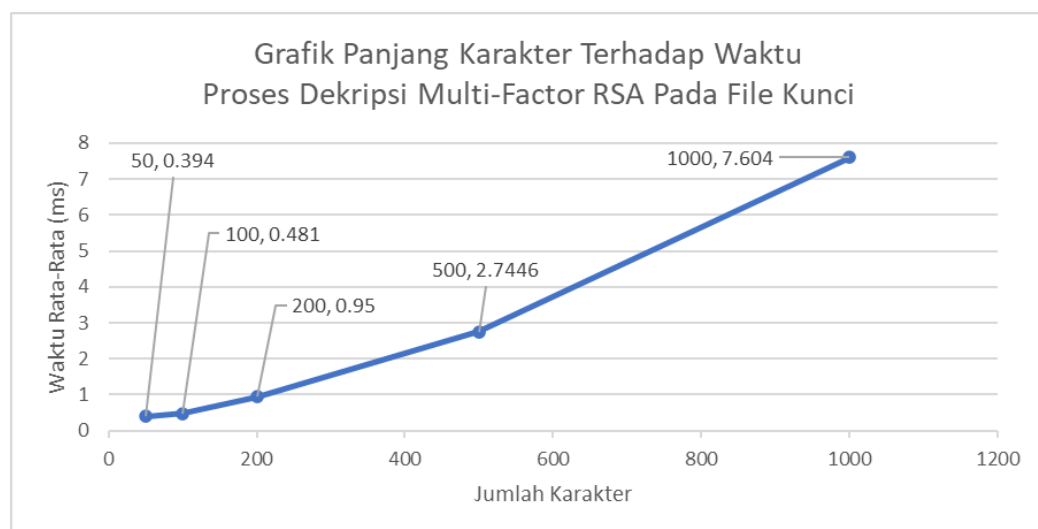
4.3.4 Waktu Proses Dekripsi *File* Kunci

Pada dekripsi *file* kunci dilakukan proses menghitung waktu yang dibutuhkan untuk mendekripsi menggunakan algoritma *Multi-Factor RSA*. Tabel 4.4 menunjukkan hasil dari pengujian waktu proses untuk dekripsi *file* kunci, serta waktu rata-rata untuk setiap jumlah karakter.

Tabel 4.4 Hasil Waktu Uji Coba Dekripsi *File Kunci*

Jumlah Karakter	Uji Coba	Waktu Proses (ms)	Waktu Rata – rata (ms)
50 karakter	1	0.3975	0.394
	2	0.3474	
	3	0.7659	
	4	0.2227	
	5	0.2408	
100 karakter	1	0.3281	0.481
	2	0.5452	
	3	0.5729	
	4	0.4421	
	5	0.5208	
200 karakter	1	0.5571	0.950
	2	1.3137	
	3	0.9178	
	4	1.0416	
	5	0.9238	
500 karakter	1	3.4206	2.7446
	2	2.706	
	3	3.5748	
	4	2.7039	
	5	1.3177	
1000 karakter	1	10.9824	7.604
	2	7.6464	
	3	6.7702	
	4	5.503	
	5	7.1209	

Gambar 4.14 berikut merupakan penggambaran dari rata-rata waktu dekripsi dengan jumlah karakter.

**Gambar 4.14** Grafik Proses Dekripsi *Multi-Factor RSA*

4.4 Kompleksitas Algoritma

Untuk menghitung banyaknya waktu yang dibutuhkan untuk menjalankan algoritma, maka diperlukan penghitungan kompleksitas algoritma dalam pengujian sistem. Penghitungan kompleksitas dilakukan dengan memanfaatkan notasi *Big-Theta* (Big- Θ).

Tabel 4.5 Kompleksitas Algoritma Enkripsi MSC

No	Listing Program	C	#	C*#
1	<code>string plaintext = richTextBox1.Text;</code>	C ₁	1	C ₁
2	<code>string key = richTextBox5.Text;</code>	C ₁	1	C ₁
3	<code>richTextBox2.Text = encryptedText;</code>	C ₁	1	C ₁
4	<code>if (key.Length >= targetLength) {</code>	C ₂	n	C ₂ n
5	<code>return key.Substring(0, targetLength);</code> <code>}</code>	C ₂	n	C ₂ n
6	<code>else {</code>	C ₃	n	C ₃ n
7	<code>expandedKey.Append(key);</code>	C ₃	n	C ₃ n
8	<code>int remainingLength = targetLength - key.Length;</code>	C ₃	n	C ₃ n
9	<code>int keyIndex = 0;</code>	C ₃	n	C ₃ n
10	<code>while (remainingLength > 0) {</code>	C ₄	n	C ₄ n
11	<code>expandedKey.Append(key[keyIndex]);</code>	C ₄	n	C ₄ n
12	<code>keyIndex = (keyIndex + 1) % key.Length;</code>	C ₄	n	C ₄ n
13	<code>remainingLength--; }</code>	C ₄	n	C ₄ n
14	<code>return expandedKey.ToString();</code> <code>}</code>	C ₄	n	C ₄ n
15	<code>for (int i = 0; i < plaintext.Length; i++) {</code>	C ₅	n	C ₅ n
16	<code>int P = (int)plaintext[i];</code>	C ₅	n	C ₅ n
17	<code>int K = (int)key[i];</code>	C ₅	n	C ₅ n
18	<code>int A = i + 1;</code>	C ₅	n	C ₅ n
19	<code>int Q = ((K * K * K) + (A * K)) % 256;</code>	C ₅	n	C ₅ n
20	<code>if (Q % 2 == 0) {</code>	C ₆	n	C ₆ n

21	Q += 1; }	C ₆	n	C ₆ n
22	int C = (P * Q) % 256;	C ₆	n	C ₆ n
23	encryptedText.Append(C.ToString());	C ₆	n	C ₆ n
24	if (i != plaintext.Length - 1) {	C ₇	n	C ₇ n
25	encryptedText.Append(" "); } }	C ₇	n	C ₇ n
26	return encryptedText.ToString();	C ₁	1	C ₁

Berdasarkan Tabel 4.5, T(n) algoritma enkripsi MSC adalah :

$$T(n) = (C_1 + C_1 + C_1 + C_1) * n^0 + (C_2 + C_2 + C_3 + C_3 + C_3 + C_3 + C_4 + C_4 + C_4 + C_4 + C_4 + C_5 + C_5 + C_5 + C_5 + C_5 + C_6 + C_6 + C_6 + C_6 + C_7 + C_7) * n$$

$$T(n) = (4 * C_1) * n^0 + (2 * C_2 + 4 * C_3 + 5 * C_4 + 5 * C_5 + 4 * C_6 + 2 * C_7) * n$$

$$T(n) = \theta(n)$$

Pada Tabel 4.6 dapat dilihat hasil penghitungan *Big-Theta* dari algoritma Dekripsi MSC.

Tabel 4.6 Kompleksitas Algoritma Dekripsi MSC

No	Listing Program	C	#	C*#
1	string encryptedText = richTextBox1.Text;	C ₁	1	C ₁
2	string key = textBox1.Text;	C ₁	1	C ₁
3	richTextBox3.Text = decryptedText;	C ₁	1	C ₁
4	if (key.Length >= targetLength) {	C ₂	n	C ₂ n
5	return key.Substring(0, targetLength); }	C ₂	n	C ₂ n
6	else {	C ₃	n	C ₃ n
7	expandedKey.Append(key);	C ₃	n	C ₃ n
8	int remainingLength = targetLength - key.Length;	C ₃	n	C ₃ n
9	int keyIndex = 0;	C ₃	n	C ₃ n
10	while (remainingLength > 0) {	C ₄	n	C ₄ n
11	expandedKey.Append(key[keyIndex]);	C ₄	n	C ₄ n

12	keyIndex = (keyIndex + 1) % key.Length;	C ₄	n	C ₄ n
13	remainingLength--;	C ₄	n	C ₄ n
14	} return expandedKey.ToString(); }	C ₃	n	C ₃ n
15	string[] asciiValues = encryptedText.Split(' ');	C ₁	1	C ₁
16	for (int i = 0; i < asciiValues.Length; i++) {	C ₅	n	C ₄ n
17	int C = int.Parse(asciiValues[i]);	C ₅	n	C ₄ n
18	int K = (int)key[i];	C ₅	n	C ₄ n
19	int A = i + 1;	C ₅	n	C ₄ n
20	int Q = ((K * K * K) + (A * K)) % 256;	C ₅	n	C ₄ n
21	if (Q % 2 == 0) {	C ₆	n	C ₅ n
22	Q += 1; }	C ₆	n	C ₅ n
23	int P = (C * D) % 256;	C ₅	n	C ₄ n
24	if (P < 0){	C ₇	n	C ₆ n
25	P += 256; }	C ₇	n	C ₆ n
26	char decryptedChar = (char)P;	C ₅	n	C ₄ n
27	decryptedText.Append(decryptedChar); }	C ₅	n	C ₄ n
28	return decryptedText.ToString();	C ₁	1	C ₁

Berdasarkan Tabel 4.6 T(n) algoritma dekripsi MSC adalah :

$$T(n) = (C_1 + C_1 + C_1 + C_1 + C_1) * n^0 + (C_2 + C_2 + C_3 + C_3 + C_3 + C_3 + C_4 + C_4 + C_4 + C_4 + C_3 + C_5 + C_5 + C_5 + C_5 + C_5 + C_6 + C_6 + C_5 + C_7 + C_7 + C_5 + C_5) * n$$

$$T(n) = (5 * C_1) * n^0 + (2 * C_2 + 5 * C_3 + 4 * C_4 + 8 * C_5 + 2 * C_6 + 2 * C_7) * n$$

$$T(n) = \theta(n)$$

Pada Tabel 4.7 dapat dilihat Hasil penghitungan *Big-Theta* dari algoritma enkripsi *Multi-Factor RSA*.

Tabel 4.7 Kompleksitas Algoritma Enkripsi *Multi-Factor RSA*

No	Listing Program	C	#	C*#
1	<code>string</code> kuncisimetris = richTextBox6.Text;	C ₁	1	C ₁
2	<code>string</code> kuncipublik = richTextBox4.Text;	C ₁	1	C ₁
3	<code>string[]</code> kuncipubliks = kuncipublik.Split('\n');	C ₁	1	C ₁
4	<code>for (int i = 0; i < kuncisimetris.Length; i += 3) {</code>	C ₂	n	C ₂ n
5	<code>string</code> subString = kuncisimetris.Substring(i, Math.Min(3, kuncisimetris.Length - i));	C ₂	n	C ₂ n
6	BigInteger asciiValue = CalculateAsciiValue(subString);	C ₂	n	C ₂ n
7	BigInteger modulo = BigInteger.Parse(kuncipubliks[1]);	C ₂	n	C ₂ n
8	BigInteger pangkat = BigInteger.Parse(kuncipubliks[0]);	C ₂	n	C ₂ n
9	BigInteger hasil = BigInteger.ModPow(asciiValue, pangkat, modulo);	$\log(m)^2$.log (e)	n	$\log(m)^2$.log (e)
10	richTextBox3.Text += hasil.ToString() + " "; }	C ₂	n	C ₂ n

Berdasarkan Tabel 4.7 T(n) algoritma dekripsi *Multi-Factor RSA* adalah :

$$T(n) = (C_1 + C_1 + C_1) * n^0 + (C_2 + C_2 + C_2 + C_2 + C_2$$

$$+ (\log(m))^2 \cdot \log(e) + C_2) * n$$

$$T(n) = (3 * C_1) * n^0 + (6 * C_2 + (\log(m))^2 \cdot \log(e)) * n$$

$$T(n) = \theta(n * (\log(m))^2 \log(e))$$

Pada Tabel 4.8 dapat dilihat hasil penghitungan *Big-Theta* dari algoritma Dekripsi *Multi-Factor RSA*.

Tabel 4.8 Kompleksitas Algoritma Dekripsi *Multi-Factor RSA*

No	Listing Program	C	#	C*#
1	<code>string</code> tekscipher = richTextBox5.Text;	C ₁	1	C ₁
2	<code>string</code> kunciprivat = richTextBox4.Text;	C ₁	1	C ₁
3	<code>string[]</code> kunciprivats = kunciprivat.Split('\n');	C ₁	1	C ₁
4	<code>string[]</code> cipherarray = tekscipher.Trim().Split();	C ₁	1	C ₁

5	<code>string hasilstring = "";</code>	C_1	1	C_1
6	<code>foreach (string cipher in cipherarray) {</code>	C_2	n	C_2n
7	<code>BigInteger cipherValue = BigInteger.Parse(cipher);</code>	C_2	n	C_2n
8	<code>BigInteger modulo = BigInteger.Parse(kunciprivats[1]);</code>	C_2	n	C_2n
9	<code>BigInteger pangkat = BigInteger.Parse(kunciprivats[0]);</code>	C_2	n	C_2n
10	<code>BigInteger decryptedValue = BigInteger.ModPow(cipherValue, pangkat, modulo);</code>	$\log(m)^2$ $\cdot \log(e)$	n	$\log(m)^2$ $\cdot \log(e)$
11	<code>hasilstring += decryptedString; }</code>	C_2	n	C_2n
12	<code>richTextBox2.Text = hasilstring;</code>	C_1	1	C_1

Berdasarkan Tabel 4.8 $T(n)$ algoritma dekripsi *Multi-Factor RSA* adalah :

$$T(n) = (C_1 + C_1 + C_1 + C_1 + C_1 + C_1) * n^0 + (C_2 + C_2 + C_2 + C_2$$

$$+ (\log(m))^2 \cdot \log(e) + C_2) * n$$

$$T(n) = (6 * C_1) * n^0 + (5 * C_2 + (\log(m))^2 \cdot \log(e)) * n$$

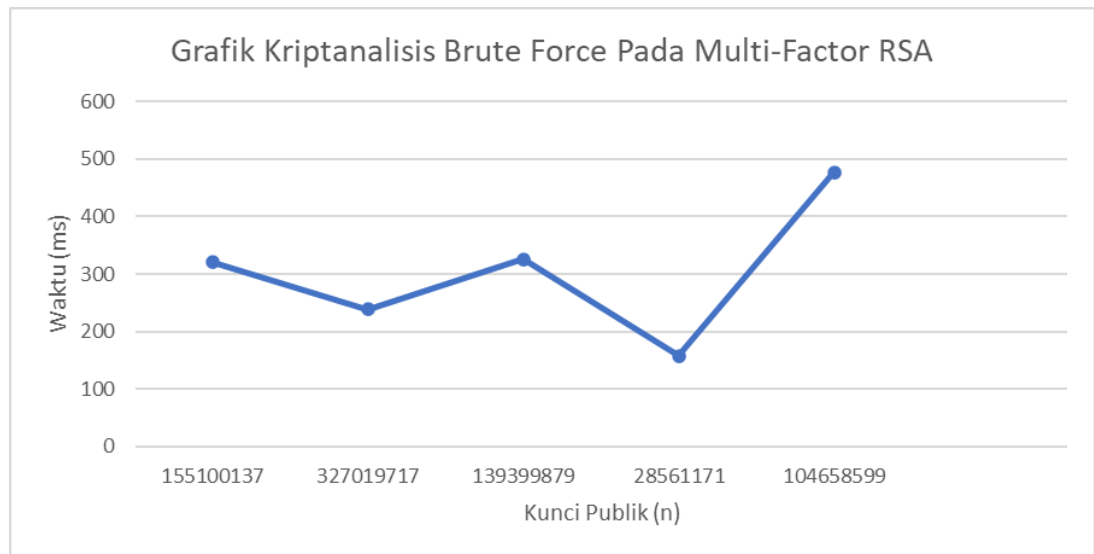
$$T(n) = \Theta(n * (\log(m))^2 \log(e))$$

4.5 Kriptanalisis Brute Force

Brute force adalah algoritma kriptanalisis untuk mengetahui lama waktu program dalam mencari faktor-faktor prima dari nilai n dalam satuan *millisecond*. Pengujian dilakukan terhadap algoritma *Multi-Factor RSA* dengan nilai n yang berbeda, dimana setiap nilai diproses sebanyak 5 kali percobaan.

Tabel 4.9 Kriptanalisis *Brute Force*

n	Faktor	Waktu Pemfaktoran (ms)
155100137	349 521 853	320
327019717	521 683 919	238
139399879	409 569 599	326
28561171	211 223 607	158
104658599	173 701 863	477



Gambar 4.15 Grafik Kriptanalisis *Brute Force* Pada *Multi-Factor RSA*

Pada Gambar 4.15 menunjukkan grafik pengujian kriptanalisis *Brute Force* pada *Multi-Factor RSA*. Hasil menunjukkan bahwa kunci yang dibangkitkan sangat cepat untuk diketahui dengan menggunakan metode kriptanalisis *Brute Force*.

BAB 5

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian serta pengujian yang dilakukan terhadap algoritma *MSC* dan algoritma *Multi-Factor RSA* pada *file* yang berekstensi *.txt maka kesimpulan yang didapatkan adalah:

1. Penerapan *Hybrid Cryptosystem* menggunakan algoritma *MSC* dan algoritma *Multi-Factor RSA* untuk mengamankan *file* teks berhasil dilakukan pada tahap pembangkit kunci, enkripsi dan dekripsi.
2. Proses dekripsi dari *ciphertext* maupun *cipherkey* menghasilkan karakter yang sama seperti *plaintext* maupun *plainkey* diawal.
3. Algoritma *MSC* lebih cepat dalam proses enkripsi dan dekripsi *file* text dibandingkan algoritma *Multi-Factor RSA*. Hal ini dikarenakan algoritma *Multi-Factor RSA* memangkatkan bilangan integer yang besar.
4. Kompleksitas algoritma *MSC* menghasilkan $T(n) = \theta(n)$ sedangkan kompleksitas algoritma *Multi-Factor RSA* menghasilkan $T(n) = \theta((\log(e))^2 \cdot \log(m))$.

5.2 Saran

1. Dalam penelitian berikutnya, diharapkan sistem dapat dikembangkan untuk perangkat lainnya, seperti website ataupun platform Android.
2. Pada penelitian selanjutnya diharapkan sistem dapat mengamankan media lainnya, seperti citra atau suara.

Daftar Pustaka

- Agustina, E. R., and Kurniati, A. (2009). Pemanfaatan Kriptografi Dalam Mewujudkan Keamanan Informasi Pada e-Voting Di Indonesia. Seminar Nasional Informatika 2009 (semnasIF 2009). 22-28
- Ariyus, Dony. (2006). *Kriptografi Keamanan Data dan Komunikasi*. Yogyakarta: Graha Ilmu.
- Arya, Felix, Paulus, Peter, and Widyarsa, Michael Ivan. (2006). Teknik-Teknik Kriptanalisis Pada RSA.
- Basri. (2016). *Kriptografi Simetris Dan Asimetris Dalam Perspektif Keamanan Data Dan Kompleksitas Komputasi*. Jurnal Ilmiah Ilmu Komputer, 2 (2).
- Boneh, D and Shacham, H. (2007). *Fast Variants of RSA*.
- Budiman, M. A., et al. (2018). Using random search and brute force algorithm in factoring the RSA modulus. *Data Science: Journal of Computing and Applied Informatics*, 2(1), 45-52
- Budiman, M. A., et al. (2021). *A cryptocompression system with Multi-Factor RSA algorithm and Levenstein code algorithm*. 5th International Conference on Computing and Applied Informatics (ICCAI 2020), No.012040.
- Chairiah, F. B. (2021). *VMPC (Variably Modified Permutation Composition) And MultiFactor RSA In Text File Security*. Journal Basic Science and Technology 9 (3).
- Christnatalis., et al. (2016). *Analisa Perbandingan Algoritma Baby Step Giant Step Dan Pohlog-Hellman Untuk Penyelesaian Logaritma Diskrit*. Jurnal Teknik Informatika 9 (2).
- Fauzi, A., et al. (2018). *Analisis Hybrid Cryptosystem Algoritma Algoritma RSA Dan Triple DES*. Jurnal Teknik Informatika Kaputama (JTIK), 1(2), 36-44.
- Humaira, R., Fitriyani, F., and Ikhsan, N. (2015). Kriptanalisis Dengan Metode Brute Force Pada Graphics Processing Unit. *eProceedings of Engineering*, 2(3).
- Menezes, A. Van Oorschot, P. and Vanstone, S. (1996). Handbook of Applied Cryptography. *CRC Press*.
- Mollin, Richard A. (2007). *An Introduction To Cryptography. Discrete Mathematics And Its Application*

- Mulyadi, S. (2015). Analisis Keamanan Akta Elektronik Pada Cyber Notary Sesuai UU Nomor 2 Tahun 2014 Tentang Jabatan Notaris (UUJN).
- Munir, R. (2019). *Kriptografi Edisi Kedua*. Bandung: informatika.
- Nasution, N. W., and Efendi, S. (2020). Analysis of RSA variants in securing message. In *IOP Conference Series: Materials Science and Engineering* (Vol. 725, No.1, p. 012131). IOP Publishing.
- Prayoga, D. D. (2021). *Implementation Of Combination Of Multi-Factor Algorithm RSA And Cipher 4x4 In Hybrid Cryptography Scheme For Security Text Messages On Instant Messaging Application*. Journal Basic Science and Technology 10 (2).
- Rachmawati, D., et al. (2018). *Hybrid Cryptosystem Using Tiny Encryption Algorithm and LUC Algorithm*. 4th International Conference on Operational Research (InteriOR), No. 012042.
- Rachmawati, D., et al. (2019). *A hybrid cryptosystem approach for information security by using RC4 algorithm and LUC algorithm*. IOP Publishing.
- Rajesh, V & Panchami V. (2019). *A Novel Multiplicative Substitution Cryptosystem*. 2019 IEEE International Conference on Electrical, Computer and Communication Technologies.
- Saputro, T. H., Hidayati, N. H., and Ujianto, E. I. H. (2020). Survei Tentang Algoritma Kriptografi Asimetris. *Jurnal Informatika Polinema*, 6(2), 67-72.
- Schneier, Bruce. (2015). *Applied Cryptography : Protocols, Algorithms, and Source Code in C*, 1st Edition Wiley.
- Smart, N. (2016). *Cryptography Made Simple*. Information Security and Cryptography.
- Zega, I., Budiman, M. A., and Efendi, S. (2023). Comparative Analysis of Ciphertext Enlargement on Generalization of the ElGamal and Multi-factor RSA. *Data Science: journal of Computing and Applied Informatics*, 7(1), 44-50.
- Zelvina, A., et al. (2012). *Perancangan Aplikasi Pembelajaran Kriptografi Kunci Publik ElGamal Untuk Mahasiswa*. IOP Publishing. JURNAL DUNIA TEKNOLOGI INFORMASI 1(1).