

***ONE TIME PAD (OTP) DENGAN MERSENNE TWISTER DAN  
DEPENDENT RSA 1 (DRSA-1) DALAM SKEMA HYBRID  
CRYPTOSYSTEM***

**SKRIPSI**

**MUHAMMAD ZULHAMUDDIN HARAHAHAP**

**181401012**



**PROGRAM STUDI S1 ILMU KOMPUTER  
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI  
UNIVERSITAS SUMATERA UTARA  
MEDAN  
2024**

*ONE TIME PAD (OTP) DENGAN MERSENNE TWISTER DAN  
DEPENDENT RSA 1 (DRSA-1) DALAM SKEMA HYBRID  
CRYPTOSYSTEM*

SKRIPSI

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

MUHAMMAD ZULHAMUDDIN HARAHAP

181401012



PROGRAM STUDI S1 ILMU KOMPUTER  
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI  
UNIVERSITAS SUMATERA UTARA

MEDAN

2024

## PERSETUJUAN

Judul : *ONE TIME PAD (OTP) DENGAN MERSENNE  
TWISTER DAN DEPENDENT RSA 1 (DRSA-1)  
DALAM SKEMA HYBRID CRYPTOSYSTEM*

Kategori : SKRIPSI

Nama : MUHAMMAD ZULHAMUDDIN HARAHAHAP

Nomor Induk Mahasiswa : 181401012

Program Studi : SARJANA (S-1) ILMU KOMPUTER

Fakultas : ILMU KOMPUTER DAN TEKNOLOGI INFORMASI  
UNIVERSITAS SUMATERA UTARA

Telah diuji dan dinyatakan lulus di Medan, 15 Oktober 2024

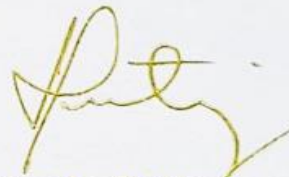
Komisi Pembimbing :

Pembimbing 2

Pembimbing 1



Dr. M. Andri Budiman, S.T., M.Comp.Sc.,  
M.E.M., S.C.J.P.  
NIP. 197510082008011011



Dewi Sartika Br Ginting,  
S.Kom, M.Kom  
NIP. 199005042019032023

Diketahui/disetujui oleh

Program Studi S-1 Ilmu Komputer

Ketua,



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

NIP. 197812212014042001

## **PERNYATAAN**

### ***ONE TIME PAD (OTP) DENGAN MERSENNE TWISTER DAN DEPENDENT RSA 1 (DRSA-1) DALAM SKEMA HYBRID CRYPTOSYSTEM***

## **SKRIPSI**

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

Medan, 15 Oktober 2024



Muhammad Zulhamuddin Harahap

181401012

## **PENGHARGAAN**

Tiada kata yang paling pantas selain dengan mengucapkan rasa syukur kehadirat Allah Subhanahu Wa Ta'ala yang maha pengasih lagi maha penyayang, berkat dan limpahan rahmat-nya sehingga penulis dapat menjalankan perkuliahan sampai kepada menyelesaikan penyusunan skripsi ini.

Oleh karena itu pada kesempatan kali ini penulis ingin mengucapkan terima kasih kepada :

1. Bapak Dr. Muryanto Amin S.Sos., M.Si. selaku Rektor Universitas Sumatera Utara.
2. Ibu Dr. Maya Silvi Lydia B.Sc., M.Sc. selaku Dekan Fasilkom-TI Universitas Sumatera Utara.
3. Ibu Dr. Amalia ST., M.T. selaku Ketua Program Studi S-1 Ilmu Komputer Universitas Sumatera Utara.
4. Ibu Dewi Sartika Br Ginting, S.Kom, M.Kom,. selaku Dosen Pembimbing I yang sudah memberikan banyak arahan kepada penulis, serta memberikan waktu, bimbingan dan saran yang sangat membangun kepada penulis.
5. Bapak Dr. M. Andri Budiman, S.T., M.Comp.Sc., M.E.M., S.C.J.P. selaku Dosen Pembimbing II yang telah memberikan inspirasi luar biasa kepada sang penulis serta memberikan bimbingan, arahan dan masukan kepada penulis.
6. Ibu Dr. Ir. Elviawati Muisa Zamzami, ST, MT. Selaku dosen selaku Dosen Penasihat Akademik selama penulis menempuh pendidikan di Program Studi S-1 Ilmu Komputer.
7. Seluruh Bapak dan Ibu Dosen Program Studi Sarjana Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
8. Seluruh tenaga pengajar dan pegawai di Program Studi S-1 Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
9. Keluarga tercinta, ayahanda Yung Yung dan ibunda Khairani Hasibuan, serta kakak saya Putri Sakinah Harahap dan adik-adik saya Rizki Faurina Harahap, Zukhatika Harahap, dan Hilliatul Aulia Harahap yang selalu memberikan doa dan motivasi yang sangat berarti kepada penulis.

10. Teman teman seperjuangan Khususnya Alvian Ridho Pramudyan, Jeremy Felix Naibaho, Shafwan Hadi Umri Lubis, Amar Daulay, Muhammad Harun Arrasyid dan teman-teman lainnya yang belum dapat penulis sebutkan satu persatu dengan memberikan dukungan dan motivasinya sehingga penulis dapat menyelesaikan penulisan skripsi ini.
11. Teman-teman dari Stambuk 2018 khususnya KOM C yang telah memberikan dukungan doa, motivasi, dan pengalaman bersama selama masa perkuliahan hingga menyelesaikan skripsi ini.
12. Para senior kampus yang telah membantu dan memberi dukungan, doa serta menginspirasi penulis.
13. Dan semua pihak yang telah banyak membantu secara langsung maupun tidak langsung yang tidak bisa disebutkan satu persatu.

Serta semua pihak yang telah membantu, memperhatikan, dan mendukung penulis dalam menyelesaikan skripsi ini hendaknya mendapat rahmat dari Allah SWT. Penulis menyadari masih terdapat kekurangan pada skripsi ini. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang bersifat membangun guna penyempurnaan skripsi ini. Oleh karena itu, kita semua dapat mengambil manfaat darinya.

Medan, 15 Oktober 2024



Muhammad Zulhamuddin Harahap

## ABSTRAK

Perkembangan teknologi sekarang ini semakin pesat dan membuat semakin tingginya ancaman terhadap data dan informasi penting. Kerahasiaan informasi dapat dijaga dengan lebih dari satu cara, salah satunya adalah dengan menyamarkan pesan sebagai kata sandi. Salah satu teknik kriptografi yang digunakan untuk mengamankan data adalah metode *Hybrid Cryptosystem* dengan menggabungkan algoritma simetris dan asimetris untuk mengenkripsi pesan yang akan dikirim. Informasi pesan diperoleh akan dienkripsi menggunakan algoritma *One Time Pad*, kunci OTP dibuat menggunakan algoritma *Mersenne Twister*, dan kunci enkripsi pesan yang telah dihasilkan diperoleh dengan menggunakan algoritma *Dependent RSA 1* (DRSA-1). Untuk meningkatkan keamanan pesan dengan algoritma *One Time Pad* dan algoritma *Dependent RSA 1* (DRSA-1) harus digunakan bersama dengan algoritma yang lebih aman. Proses algoritma *One Time Pad* menjaga privasi data atau informasi yang ingin Anda kirim namun relatif mudah digunakan. algoritma *Dependent RSA 1* (DRSA-1) merupakan variasi dari algoritma RSA. Varian RSA ini telah terbukti aman secara semantik terhadap serangan adaptif pada ciphertext yang dipilih. Tingkat kerumitan algoritma *Dependent RSA 1* (DRSA-1) terletak pada besarnya angka yang dibuat. Semakin besar bilangan prima yang dibuat maka semakin tinggi pula tingkat perhitungan kerumitannya. Waktu rata-rata enkripsi OTP selama 20,6 *millisecond* dan waktu rata-rata enkripsi DRSA1 20,8 *millisecond* sedangkan waktu rata-rata dekripsi OTP 1 *millisecond* dan waktu rata-rata dekripsi DRSA1 22,4 *millisecond*.

Kata kunci : Kriptografi, *One Time Pad*, *Mersenne Twister*, *Dependent RSA 1* (DRSA-1), *Hybrid Cryptosystem*, Kriptanalisis, *Enkripsi*, *Dekripsi*.

# **ONE TIME PAD (OTP) WITH MERSENNE TWISTER AND DEPENDENT RSA 1 (DRSA-1) IN HYBRID CRYPTOSYSTEM SCHEME**

## **ABSTRACT**

The development of technology is now increasingly rapid and makes the threats to important data and information even higher. Information confidentiality can be maintained in more than one way, one of which is by disguising the message as a password. One of the cryptographic techniques used to secure data is the Hybrid Cryptosystem method by combining symmetric and asymmetric algorithms to encrypt the message to be sent. The message information obtained will be encrypted using the One Time Pad algorithm, the OTP key is generated using the Mersenne Twister algorithm, and the message encryption key that has been generated is obtained using the Dependent RSA 1 (DRSA-1) algorithm. To increase the security of the message, the One Time Pad algorithm and the Dependent RSA 1 (DRSA-1) algorithm must be used together with a more secure algorithm. The One Time Pad algorithm process maintains the privacy of the data or information you want to send but is relatively easy to use. Dependent RSA 1 (DRSA-1) algorithm is a variation of the RSA algorithm. This variant of RSA has been proven to be semantically secure against adaptive attacks on selected ciphertexts. The complexity of the Dependent RSA 1 (DRSA-1) algorithm lies in the size of the numbers created. The larger the prime number, the higher the complexity of the calculation. The average time of OTP encryption is 20.6 milliseconds and the average time of DRSA1 encryption is 20.8 milliseconds while the average time of OTP decryption is 1 millisecond and the average time of DRSA1 decryption is 22.4 milliseconds.

**Keywords:** *Cryptography, One Time Pad, Mersenne Twister, Dependent RSA 1 (DRSA-1), Hybrid Cryptosystem, Cryptanalysis, Encryption, Decryption.*



## DAFTAR ISI

PERSETUJUAN .....	i
PERNYATAAN .....	ii
PENGHARGAAN .....	iii
ABSTRAK .....	v
ABSTRACT .....	vi
DAFTAR ISI .....	vii
DAFTAR TABEL.....	x
DAFTAR GAMBAR.....	xi
BAB 1 PENDAHULUAN	
1.1. Latar Belakang.....	1
1.2. Rumusan Masalah.....	2
1.3. Batasan Masalah .....	3
1.4. Tujuan Penelitian .....	3
1.5. Manfaat Penelitian .....	3
1.6. Metode Penelitian .....	3
1.7. Sistematika Penulisan .....	4
BAB 2 LANDASAN TEORI	
2.1 Kriptografi .....	6
2.1.1 Definisi Kriptografi .....	6
2.1.2 Istilah-istilah dalam kriptografi .....	7
2.1.3 Aspek keamanan kriptografi.....	8
2.1.4 Jenis Kriptografi .....	8
2.2 Algoritma One Time Pad (OTP).....	10
2.2.1 Definisi Algoritma One Time Pad .....	11
2.2.2 Aturan enkripsi dan dekripsi algoritma One Time Pad .....	11
2.2.3 Contoh perhitungan algoritma One Time Pad .....	12
2.3 Algoritma Mersenne Twister .....	13
2.3.1 Definisi Algoritma Mersenne Twister .....	13
2.3.2 Parameter dari 32-Bit MT.....	14

2.3.3	Proses Generate Algoritma MT 19937 .....	15
2.4	Algoritma Dependent RSA 1 (DRSA-1) .....	16
2.4.1	Definisi Algoritma Dependent RSA 1 (DRSA-1) .....	16
2.4.2	Landasan Matematika Dependent RSA 1 (DRSA-1) .....	17
2.4.3	Prinsip kerja algoritma Dependent RSA 1 (DRSA-1) .....	22
2.4.4	Contoh perhitungan algoritma Dependent RSA 1 (DRSA-1) .....	23
2.5	Metode Hybrid Cryptosystem .....	25
2.6	Penelitian Relevan .....	26
<b>BAB 3 ANALISIS DAN PERANCANGAN</b>		
3.1	Analisis Sistem .....	28
3.1.1	Analisis Masalah .....	28
3.1.2	Analisis Kebutuhan .....	29
3.1.3	Analisis Proses .....	31
3.2	Diagram Umum Sistem .....	31
3.3	Pemodelan Sistem .....	32
3.3.1	Use-Case Diagram .....	33
3.3.2	Sequence Diagram .....	36
3.3.3	Activity Diagram .....	37
3.4	Flowchart .....	37
3.4.1	Flowchart Sistem .....	38
3.4.2	Flowchart Enkripsi dengan Algoritma One Time Pad .....	39
3.4.3	Flowchart Dekripsi dengan Algoritma One Time Pad .....	40
3.4.4	Flowchart Generate Kunci dengan Dependent RSA 1 (DRSA-1) .....	41
3.4.5	Flowchart Enkripsi dengan Dependent RSA 1 (DRSA-1) .....	42
3.4.6	Flowchart Dekripsi dengan Dependent RSA 1 (DRSA-1) .....	43
3.4.7	Flowchart Generate Kunci dengan Mersenne Twister .....	44
3.5	Perancangan Antarmuka ( <i>Interface</i> ) .....	44
3.5.1	Form Halaman Utama dan Tampilan Awal .....	45
3.5.2	Form Halaman Pembangkit Kunci .....	46
3.5.3	Form Halaman Enkripsi .....	47
3.5.4	Form Halaman Dekripsi .....	48

## BAB 4 IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1	Implementasi Sistem.....	50
4.1.1	Halaman Utama .....	50
4.1.2	Halaman Pembangkit Bilangan Kunci.....	51
4.1.3	Halaman Enkripsi .....	51
4.1.4	Halaman Dekripsi .....	52
4.2	Pengujian Sistem .....	53
4.2.1	Pengujian Pembangkit Kunci .....	54
4.2.2	Pengujian Enkripsi Algoritma <i>One Time Pad</i> .....	56
4.2.3	Pengujian Enkripsi kunci OTP dengan <i>Dependent RSA 1</i> ( <i>DRSA-1</i> ) .....	58
4.2.4	Pengujian Dekripsi Kunci OTP dengan <i>Dependent RSA 1</i> <i>DRSA-1</i> ) .....	61
4.2.5	Pengujian Dekripsi Pesan dengan <i>One Time Pad</i> .....	63
4.3	Waktu Proses ( <i>Real Running Time</i> ).....	64
4.3.1	Pengujian proses enkripsi Pesan dengan <i>One Time Pad</i> .....	65
4.3.2	Pengujian proses enkripsi kunci dengan Algoritma <i>Dependent RSA 1 (DRSA-1)</i> .....	66
4.3.3	Pengujian proses dekripsi cipherkey dengan Algoritma <i>Dependent RSA 1 (DRSA-1)</i> .....	67
4.3.4	Pengujian proses dekripsi pesan dengan <i>One Time Pad</i> .....	68
4.3.5	Perbandingan waktu proses enkripsi dan dekripsi pesan .....	69
4.3.6	Perbandingan waktu proses enkripsi dan dekripsi kunci .....	70

## BAB 5 KESIMPULAN DAN SARAN

5.1.	Kesimpulan .....	71
5.1.	Saran .....	72

DAFTAR PUSTAKA .....	73
----------------------	----

## DAFTAR TABEL

Tabel 2.1	Parameter MT 19937 32 bit .....	14
Tabel 2.2	<i>Invers Modulo</i> .....	18
Tabel 3.1	<i>Narrative Use-Case</i> Enkripsi Pesan.....	34
Tabel 3.2	<i>Narrative Use-Case</i> Dekripsi Pesan .....	35
Tabel 3.3	<i>Narrative Use-Case</i> Bangkitkan Kunci .....	36
Tabel 4.1	Invers Modulo menghitung nilai d.....	56
Tabel 4.2	Enkripsi pesan dengan Algoritma <i>One Time Pad</i> .....	58
Tabel 4.3	Enkripsi Kunci dengan Algoritma <i>Dependent RSA 1 (DRSA-1)</i> .....	60
Tabel 4.4	Dekripsi <i>cipherkey</i> dengan Algoritma <i>Dependent RSA 1 (DRSA-1)</i> ..	62
Tabel 4.5	Dekripsi dengan Algoritma <i>One Time Pad</i> .....	64
Tabel 4.6	Waktu Proses Enkripsi dengan Algoritma <i>One Time Pad</i> .....	65
Tabel 4.7	Waktu Proses Enkripsi kunci dengan Algoritma <i>Dependent RSA 1 (DRSA-1)</i> .....	66
Tabel 4.8	Waktu Proses dekripsi <i>cipherkey</i> menjadi kunci dengan Algoritma <i>Dependent RSA 1 (DRSA-1)</i> .....	67
Tabel 4.9	Waktu Proses Dekripsi dengan Algoritma <i>One Time Pad</i> .....	68
Tabel 4.10	Perbandingan waktu proses enkripsi dan dekripsi pesan .....	69
Tabel 4.11	Perbandingan waktu proses <i>enkripsi</i> dan <i>dekripsi</i> kunci .....	70

## DAFTAR GAMBAR

Gambar 2.1	Proses enkripsi dan dekripsi kriptografi.....	6
Gambar 2.2	Kriptografi kunci simetris .....	9
Gambar 2.3	Kriptografi kunci Asimetris .....	10
Gambar 2.4	Digram block dari MT 19937 .....	13
Gambar 2.5	Mekanisme <i>Hybrid Cryptosystem</i> .....	26
Gambar 3.1	<i>Diagram Ishikawa</i> .....	29
Gambar 3.2	Diagram Umum Sistem.....	32
Gambar 3.3	<i>Diagram Use-Case</i> .....	33
Gambar 3.4	<i>Diagram Sequence</i> .....	37
Gambar 3.5	<i>Activity Diagram</i> .....	37
Gambar 3.6	<i>Flowchart Sistem</i> .....	38
Gambar 3.7	<i>Flowchart Enkripsi dengan Algoritma One Time Pad</i> .....	39
Gambar 3.8	<i>Flowchart Dekripsi dengan Algoritma One Time Pad</i> .....	40
Gambar 3.9	<i>Flowchart Generate Kunci dengan Dependent RSA 1 (DRSA-1)</i> ...41	
Gambar 3.10	<i>Flowchart Enkripsi dengan Dependent RSA 1 (DRSA-1)</i> .....	42
Gambar 3.11	<i>Flowchart Dekripsi dengan Dependent RSA 1 (DRSA-1)</i> .....	43
Gambar 3.12	<i>Flowchart Algoritma Mersenne Twister</i> .....	44
Gambar 3.13	Rancangan Tampilan Halaman Utama dan Tampilan Awal.....	45
Gambar 3.14	Rancangan Tampilan Halaman Pembangkit Kunci .....	46
Gambar 3.15	Rancangan Tampilan Halaman Enkripsi .....	47
Gambar 3.16	Rancangan Tampilan Halaman Dekripsi .....	49
Gambar 4.1	Tampilan Halaman Utama .....	50
Gambar 4.2	Tampilan Halaman Pembangkit Kunci .....	51
Gambar 4.3	Tampilan Halaman Enkripsi .....	52
Gambar 4.4	Tampilan Halaman Dekripsi .....	53
Gambar 4.5	Hasil Pembangkit Kunci <i>Dependent RSA 1 (DRSA-1)</i> .....	54
Gambar 4.6	Hasil Enkripsi pesan dengan <i>One Time Pad</i> .....	57
Gambar 4.7	Hasil Enkripsi Kunci OTP dengan <i>Dependent RSA 1 (DRSA-1)</i> ...58	
Gambar 4.8	Hasil Dekripsi <i>Cipherkey</i> dengan <i>Dependent RSA 1 (DRSA-1)</i> ....	61

Gambar 4.9	Hasil Dekripsi <i>Ciphertext</i> dengan <i>One Time Pad</i> .....	63
Gambar 4.10	Diagram yang menggambarkan hubungan panjang <i>plaintext</i> dan waktu proses enkripsi menggunakan <i>One Time Pad</i> .....	65
Gambar 4.11	Diagram yang menggambarkan hubungan panjang kunci dan waktu proses enkripsi menggunakan <i>Dependent RSA 1</i> (DRSA-1) .....	66
Gambar 4.12	Diagram yang menggambarkan hubungan panjang kunci dan waktu proses dekripsi menggunakan <i>Dependent RSA 1</i> (DRSA-1) .....	67
Gambar 4.13	Diagram yang menggambarkan hubungan panjang <i>plaintext</i> dan waktu proses dekripsi menggunakan <i>One Time Pad</i> .....	68
Gambar 4.14	Diagram yang membandingkan waktu enkripsi dan dekripsi <i>plaintext</i> .....	69
Gambar 4.15	Diagram yang membandingkan waktu enkripsi dan dekripsi kunci .....	70

## BAB 1

### PENDAHULUAN

#### 1.1. Latar Belakang

Seiring kemajuan teknologi setiap hari mungkin terdapat peningkatan jumlah ancaman terhadap keamanan data dan informasi penting. Keamanan data merupakan aspek krusial dalam proses penyampaian informasi melalui internet karena banyaknya permasalahan yang muncul akibat banyaknya kasus pencurian data, baik di situs pemerintah maupun platform media sosial. Ada banyak kesalahan digital yang kita ketahui dalam komunikasi luas dalam mengancam informasi dan keamanan sistem dimana para pelaku ini memanfaatkan celah keamanan yang ada untuk memasuki dan mengontrol informasi. Informasi tersebut mudah dieksploitasi oleh pihak yang tidak berkepentingan jika hal ini diabaikan dengan asumsi bahwa data digunakan untuk tujuan yang tidak pantas, hal ini akan menghambat pengiriman dan penerima data.

Banyak orang belajar bagaimana bagaimana cara untuk mengamankan informasi yang mereka terima atau bagaimana melindungi informasi yang mereka komunikasikan. Untuk mengatasi hal tersebut diperlukan suatu metode yang disebut kriptografi.

Salah satu algoritma kriptografi yang masih dianggap aman adalah algoritma *One Time Pad* (OTP). OTP dikenal sebagai sistem enkripsi simetris yang ideal karena menawarkan keamanan teoritis yang sempurna, dimana pesan hanya dapat dienkripsi jika kunci yang digunakan benar-benar acak dan hanya digunakan sekali. Namun salah satu kendala dalam penerapan OTP adalah kebutuhan akan kunci yang sangat panjang dan benar-benar acak, yang sulit untuk diimplementasikan secara praktis.

Untuk mengatasi hal ini pendekatan hybrid dalam kriptografi telah banyak digunakan. Pendekatan ini memadukan teknik enkripsi simetris dan asimetris sehingga memungkinkan memiliki keamanan yang lebih tinggi. Pada skripsi ini penulis menggunakan *Mersenne Twister* sebagai generator bilangan acak pseudorandom untuk menghasilkan kunci OTP. *Mersenne twister* dipilih karena

memiliki kecepatan dan kemampuan untuk menghasilkan bilangan acak sehingga dapat memperbaiki kelemahan OTP dalam hal kebutuhan kunci yang acak.

Selain itu, metode enkripsi asimetris *Dependent RSA 1* (DRSA-1) juga digunakan sebagai bagian dari skema hybrid ini. DRSA-1 menawarkan keamanan tambahan melalui penggunaan prinsip-prinsip RSA yang telah terbukti tangguh dalam dunia kriptografi. Dalam skema hybrid ini, OTP digunakan untuk mengenkripsi pesan, sementara DRSA-1 digunakan untuk mengenkripsi kunci OTP (Riza, 2021).

Penelitian sebelumnya berfokus pada bagaimana mengatasi tantangan dalam penggunaan OTP, terutama dalam pengacakan kunci. Salah satu yang diusulkan adalah penggunaan generator bilangan acak *pseudorandom* untuk menghasilkan kunci yang mendekati acak. *Mersenne Twister* dikenal sebagai salah satu algoritma penghasil bilangan acak *pseudorandom* yang cepat dan berkualitas tinggi dengan periode yang panjang. Berbagai penelitian menunjukkan bahwa *Mersenne Twister* telah digunakan secara luas namun masih perlu diteliti lebih lanjut mengenai penerapannya dalam skema OTP (Siswanto, 2016).

Kombinasi antara One Time Pad (OTP), Mersenne Twister, dan Dependent RSA 1 (DRSA-1) diharapkan mampu menciptakan sebuah skema hybrid cryptosystem yang tidak hanya aman secara teoritis tetapi juga efisien dan dapat diimplementasikan dalam berbagai aplikasi komunikasi data yang membutuhkan tingkat keamanan tinggi. Skripsi ini akan mengeksplorasi bagaimana ketiga komponen tersebut dapat bekerja bersama untuk menciptakan sistem kriptografi yang tangguh, serta menganalisis performa dan keamanan dari skema hybrid yang diusulkan.

## **1.2. Rumusan Masalah**

Perkembangan teknologi yang pesat saat ini meningkatkan ancaman terhadap data dan informasi penting. Oleh karena itu, diperlukan pengembangan metode yang lebih canggih dalam melindungi informasi melalui kriptografi. Untuk mengatasi kelemahan tersebut diperlukan langkah-langkah keamanan yang lebih kuat. Salah satu solusinya adalah dengan menerapkan skema *Hybrid Cryptosystem* yang menggabungkan algoritma *One Time Pad* dengan *Mersenne Twister* serta algoritma



*Dependent RSA 1* (DRSA-1) guna menjamin keamanan pesan dan kunci yang dikirim oleh pengguna.

### **1.3. Batasan Masalah**

Batasan masalah pada penelitian ini sebagai berikut :

1. Menggunakan algoritma kriptografi *One Time Pad (OTP)* dan *Dependent RSA 1 (DRSA-1)* dalam skema *Hybrid Cryptosystem*.
2. Pesan yang diuji adalah pesan *text*.
3. Hanya menggunakan 0 sampai 256 karakter.
4. Hanya menggunakan pembangkit kunci OTP dengan metode *Mersenne Twister*.
5. Sistem yang dibangun dengan bahasa Java, dan aplikasinya berjalan di komputer desktop.

### **1.4. Tujuan Penelitian**

Berdasarkan permasalahan diatas, tujuan dari penelitian ini adalah untuk merancang sebuah sistem yang mampu menjaga keamanan pesan. Sistem dibangun dengan menggunakan algoritma *One Time Pad (OTP)* yang dikombinasikan dengan *Mersenne Twister* dan *Dependent RSA 1 (DRSA-1)* dalam skema *Hybrid Cryptosystem*.

### **1.5. Manfaat Penelitian**

Penulis mampu mengembangkan sistem yang dapat mengamankan pesan dan sebuah aplikasi yang berguna untuk melindungi privasi dan keamanan data yang dikirimkan orang lain.

### **1.6. Metode Penelitian**

Struktur yang dipakai pada penelitian, yaitu :

1. Studi Pustaka

Pada tahap ini, penelitian dimulai dengan melakukan tinjauan terhadap literatur yang meliputi berbagai sumber seperti buku, jurnal, e-book, artikel ilmiah, dan situs web yang membahas topik terkait, seperti *One Time Pad*

(OTP), *Mersenne Twister*, *Dependent RSA 1* (DRSA-1), dan skema *Hybrid Cryptosystem*.

2. Analisa dan Perancangan

Penulis melihat algoritma *One Time Pad* (OTP), *Mersenne Twister*, dan *Dependent RSA 1* (DRSA-1) sebagai bagian dari ruang lingkup penelitian.

3. Implementasi sistem

Pada tahap ini, Pada titik ini penulis mengembangkan sistem yang menggabungkan algoritma *Mersenne Twister* dengan algoritma *One Time Pad* (OTP) dan skema *Hybrid Cryptosystem* dengan *Dependent RSA 1* (DRSA-1).

4. Pengujian

Skema *Hybrid Cryptosystem* digunakan untuk menguji sistem yang dirancang dengan menggabungkan algoritma *Mersenne Twister* dengan algoritma *Dependent RSA 1* (DRSA-1) dan *One Time Pad* (OTP).

5. Dokumentasi

Eksplorasi yang telah dilakukan dicatat pada tahap ini, mulai dari tahap pemeriksaan hingga tahap pengujian, sebagai suatu proposisi.

### 1.7. Sistematika Penulisan

Metode penulisan yang digunakan dalam penelitian, yaitu :

#### BAB 1 PENDAHULUAN

Bagian ini memahami landasan eksplorasi judul proposisi “*One Time Pad* (OTP) dengan *Mersenne Twister* dan *Dependent RSA 1* (DRSA-1) dalam skema hybrid Cryptosystem”, rencana isu, batasan isu, target penelitian, penelitian manfaat, strategi penelitian, dan komposisi yang efisien.

#### BAB 2 LANDASAN TEORI

Menyajikan ringkasan tentang konsep dasar kriptografi, serta skema *Hybrid Cryptosystem*, *One Time Pad* (OTP), *Mersenne Twister*, dan *Dependent RSA 1* (DRSA-1).

### **BAB 3 ANALISIS DAN PERANCANGAN**

Terdiri dari analisis sistem, pemodelan sistem, Flowchart, dan perancangan sistem serta memuat uraian analisis proses kerja *One Time Pad* (OTP) dengan *Mersenne Twister* dan *Dependent RSA 1* (DRSA-1).

### **BAB 4 IMPLEMENTASI DAN PENGUJIAN**

Teknik yang dikembangkan penulis sesuai dengan analisis dan desain. Setelah itu, sistem baru menjalani serangkaian pengujian untuk melihat apakah sesuai dengan cetak biru rancangan aslinya.

### **BAB 5 KESIMPULAN DAN SARAN**

terdiri dari rekomendasi berbasis tes yang diharapkan berguna untuk pengembangan penelitian masa depan secara keseluruhan.

.

## BAB 2

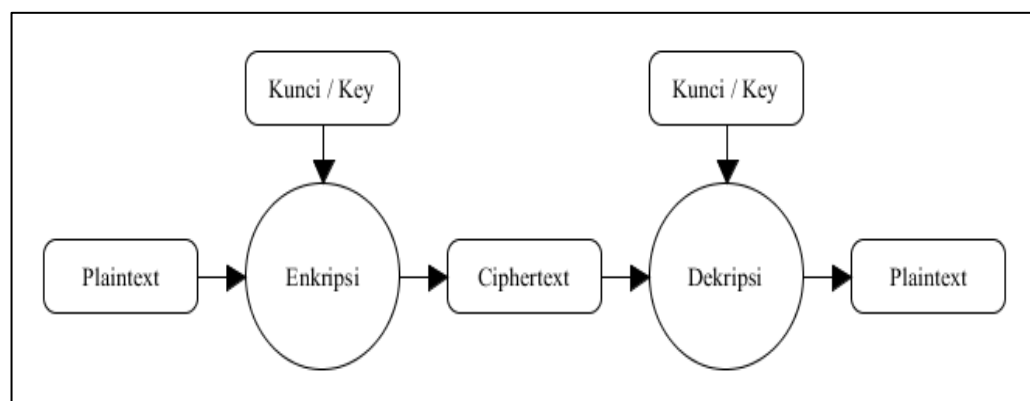
### LANDASAN TEORI

#### 2.1 Kriptografi

Penggunaan alat komunikasi untuk bertukar data melalui internet dapat mengakibatkan data tersebut dibaca oleh pihak lain. Untuk menagatasi hal tersebut diperlukan kerangka keamanan komputer yang dapat menjaga keamanan informasi dari pihak yang tidak berwenang. Jika tidak seseorang yang biasanya menyimpan data penting dalam dokumen tidak terenkripsi akan sangat rentan jika datanya tidak aman. Untuk menjaga keamanan data tersebut pada sebuah software atau aplikasi maka perlu diterapkan sebuah algoritma yang disebut dengan *Cryptography* (Febriansyah, 2012).

##### 2.1.1 Definisi Kriptografi

Kriptografi adalah ilmu sekaligus seni yang berfungsi melindungi kerahasiaan pesan dengan menyembunyikannya di balik kata sandi yang tidak ada artinya, dimana hasil rahasianya disebut *ciphertext* dan pesan rahasianya disebut *plaintext* . Metode yang digunakan untuk mengubah teks biasa menjadi teks tersandi disebut enkripsi, dan metode yang digunakan untuk mengubah teks tersandi menjadi teks biasa disebut dekripsi (Romindo, 2018).



**Gambar 2.1.** Proses enkripsi dan dekripsi kriptografi

### 2.1.2 Istilah-istilah dalam kriptografi

Ada beberapa istilah dalam dunia kriptografi yang penting untuk diketahui, antara lain:

#### 1. Pesan, *Plaintext* , dan *Ciphertext*

Pesan adalah informasi atau data yang dapat dibaca dan dirasakan. Pesan, gambar, atau suara (*audio*) yang terkomputerisasi atau sederhana semuanya dapat menyampaikan pesan. *Plaintext* adalah pesan yang tersusun atau tersusun yang memiliki arti penting dan pesan ini akan ditangani menggunakan perhitungan kriptografi untuk menjadi *ciphertext*. Pesan yang telah dienkripsi disebut *ciphertext*. Karena ditulis dengan karakter yang tidak ada artinya, pesan dalam teks kode tidak dapat terbaca.

#### 2. *Sender* dan *Recipient*

*Sender* adalah pihak yang mengirim pesan antara entitas kepada entitas lainnya. Dan *recipient* adalah pihak yang menerima pesan yang dikirim oleh sender.

#### 3. Enkripsi dan Dekripsi

Proses mengubah suatu pesan yang dapat dipahami (*plaintext* ) menjadi bentuk yang sulit dipahami (*ciphertext*) dikenal dengan istilah enkripsi. Proses mengembalikan ciphertext ke pesan aslinya (*plaintext* ) dikenal dengan istilah dekripsi.

#### 4. *Cipher* dan Kunci

Sandi yang digunakan untuk enkripsi dan dekripsi dimungkinkan untuk menafsirkan sandi sebagai acuan enkripsi dan dekripsi. Kunci adalah prosedur kriptografi yang melakukan proses enkripsi dan dekripsi.

#### 5. Sistem Kriptografi

Kumpulan algoritma untuk enkripsi, dekripsi, kunci, dan semua teks biasa dan teks sandi dikenal sebagai sistem kriptografi (Schneier, 1996).

## 6. Penyadap

Masuk akal untuk berasumsi bahwa orang yang mendengarkan adalah orang yang mencoba mencegat pesan yang sedang dikirim. Alasan penyadapan adalah untuk mendapatkan sebanyak mungkin data tentang kerangka kriptografi yang digunakan untuk memecahkan *ciphertext*.

## 7. Kriptanalisis

Proses memperoleh *plaintext* tanpa memerlukan kunci yang sah secara alami dikenal sebagai kriptanalisis, yang merupakan studi tentang menganalisis kode atau sandi. Ini terjadi ketika teks terenkripsi berhasil diubah kembali menjadi teks asli tanpa menggunakan kunci yang benar.

### 2.1.3 Aspek Keamanan Kriptografi

Ada empat aspek keamanan kriptografi, antara lain (Ariyus, 2008):

1. Kerahasiaan adalah suatu bantuan yang untuk memastikan bahwa pesan tidak dapat diakses atau dibaca orang yang tidak berkepentingan sehingga pesan harus tiba pada penerima yang mempunyai kunci rahasia.
2. Integritas data digunakan untuk memastikan bahwa pesan adalah asli atau belum diubah oleh pihak yang tidak dapat diandalkan.
3. Otentikasi adalah proses untuk memverifikasi identitas yang ingin menggunakan informasi atau data sistem.
4. Anti-penyangkalan adalah layanan untuk mencegah pelaku sistem informasi dan untuk menyangkal tindakan mereka.

### 2.1.4 Jenis Kriptografi

Berdasarkan teknik pengamanannya kriptografi dibagi menjadi dua jenis, yaitu:

#### 1. Kriptografi Kunci Simetris

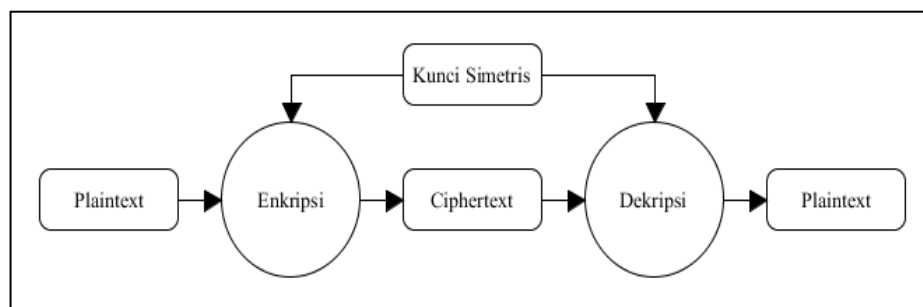
Kriptografi kunci simetris merupakan jenis kunci yang paling umum digunakan, hal ini karena cara membuat pesan yang dikodekan (enkripsi) sama dengan membuka pesan yang dikodekan (dekripsi). Jadi, dalam membuat pesan yang dikodekan, pengirim pesan dan penerima pesan menawarkan kunci serupa sebelum memperdagangkan pesan (Saragih, 2018). Contoh kriptografi kunci simetris adalah *One Time Pad*, *Blowfish*, *Hillcipher*.

Keunggulan dari kunci simetris:

1. Algoritma simetris memiliki kinerja yang lebih cepat dibandingkan dengan algoritma asimetris.
2. Karena kecepatannya lebih tinggi daripada algoritma asimetris, algoritma kunci simetris dapat digunakan dalam sistem real-time.

Kelemahan kunci-simteris

1. Diperlukan kunci yang berbeda untuk mengirim setiap pesan kepada pengguna yang berbeda yang menghasilkan tantangan dalam manajemen kunci.
2. Masalah yang timbul dalam manajemen kunci tersebut sering disebut sebagai "masalah distribusi kunci".



**Gambar 2.2.** Kriptografi kunci simetris

Berdasarkan gambar 2.2, cara kunci simetris adalah dengan terlebih dahulu menginput data sebagai *plaintext* kemudian menggunakan kunci yang sama untuk mengubahnya menjadi *ciphertext* sehingga dapat diubah lagi dengan cara mendekripsi *ciphertext* tersebut hingga menjadi *plaintext* kembali.

## 2. Kriptografi Kunci Asimetris

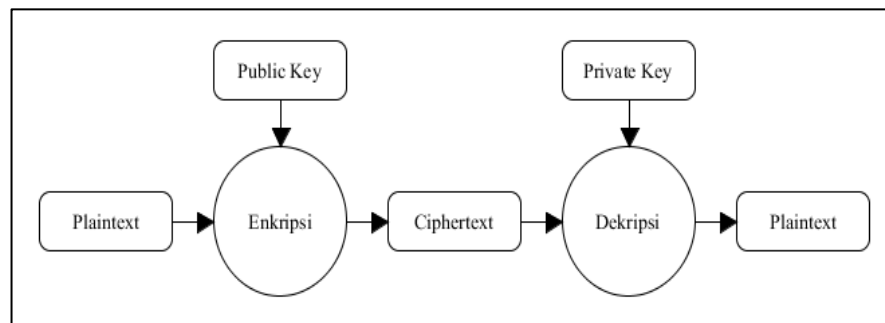
Kriptografi kunci asimetris adalah jenis kriptografi kunci yang digunakan untuk menangani enkripsi dan dekripsi perhitungan dengan dua kunci berbeda, yaitu kunci publik yang tersebar untuk komunikasi enkripsi dan kunci privat didapatkan oleh yang menerima untuk dekripsi pesan. (Sianipar et al., 2019). Contoh kriptografi kunci asimetris adalah RSA, ElGamal, LUC, DSA dan DRSA.

Keunggulan dari kunci asimetris:

1. Masalah keamanan distribusi dapat diatasi lebih mudah dengan kunci asimetris dibandingkan dengan kunci simetris.
2. Dan masalah manajemen kunci juga lebih baik dikarenakan kunci lebih sedikit.

Kelemahan kunci asimetris :

1. Algoritma kunci-asimetris memiliki kecepatan yang lebih lambat dibandingkan dengan algoritma kunci-simetris.
2. Algoritma asimetris menggunakan kunci yang lebih panjang daripada algoritma simetris.



**Gambar 2.3.** Kriptografi kunci asimetris

Gambar 2.2 menjelaskan cara kerja enkripsi dan dekripsi namun berkaitan dengan kunci asimetris yang menggunakan dua kunci berbeda untuk enkripsi dan dekripsi.

## 2.2 Algoritma *One Time Pad* (OTP)

Algoritma *One Time Pad* (OTP) merupakan salah satu algoritma simetris yang melakukan enkripsi dan dekripsi satu karakter setiap kali digunakan. Algoritma ini ditemukan oleh G.Viernam dan Major Joseph Mauborgne pada tahun 1917 sebagai perbaikan dari *Vernam cipher* untuk menghasilkan keamanan yang sempurna. Mauborgne mengusulkan penggunaan pada kertas yang berisi deretan karakter kunci yang dibangkitkan secara acak dan hanya sekali digunakan supaya tidak digunakan orang untuk mengenkripsi pesan lain (Munir, 2011).



### 2.2.1 Definisi Algoritma One Time Pad

Dengan algoritma *One Time Pad* (OTP), algoritma Vernam Cipher telah ditingkatkan untuk menjamin keamanan data secara aman. Algoritma ini masuk dalam golongan kriptografi simetris karena mengenkripsi dan mendekripsi pesan dengan kunci yang sama (Romindo, 2018).

Algoritma *One Time Pad* beroperasi dengan cara di mana penerima pesan memiliki salinan kunci yang identik dengan yang dimiliki oleh pengirim. Kunci tersebut hanya digunakan sekali (hanya sekali) untuk enkripsi, dan setelah penggunaan, kunci tersebut serta bantalan yang digunakan (misalnya, kertas blok catatan) harus segera dihapus. Hal ini dilakukan untuk mencegah penggunaan kembali kunci yang serupa untuk enkripsi dan dekripsi pesan lainnya. *Ciphertext* yang dihasilkan oleh *One Time Pad* memiliki panjang yang serupa pada *plaintext* dan memiliki panjang kunci yang serupa pada *plaintext* dan *ciphertext*. Sehingga tidak adanya pengulangan kunci selama proses enkripsi tersebut.

Algoritma *one time pad* memiliki sifat bahwa panjang pesan (*plaintext*) harus sama panjang dengan kunci, meneliti dengan table ASCII dan panjang key tidak harus sama dengan panjang plainteks. Akan tetapi, kunci yang tidak sama harus mengulang kata sehingga panjang kunci sama dengan panjang pesan (Harahap, 2019).

### 2.2.2 Aturan enkripsi dan dekripsi dalam algoritma One Time Pad

Rumus untuk enkripsi *One Time Pad* :

$$C_i = (P_i + K_i) \bmod 256$$

Rumus untuk dekripsi *One Time Pad* :

$$P_i = (C_i - K_i) \bmod 256$$

Keterangan rumus :

$$C_i = \text{Cipherteks (Ciphertext)}$$

$$P_i = \text{Plainteks (Plaintext)}$$

$$K_i = \text{Kunci (Key)}$$

Dalam situasi ini, panjang teks biasa harus sama dengan panjang kunci, sehingga ada kebutuhan yang kuat untuk mengulang kunci dalam proses enkripsi.

Setelah pengirim mengenkripsi pesan menggunakan kunci, kunci tersebut akan dihancurkan, dan penerima akan mendekripsi pesan menggunakan kunci yang serupa untuk mengembalikan teks biasa.

### 2.2.3 Contoh perhitungan algoritma One Time Pad

*Plaintext* = AMAN

Langkah awal adalah dengan mengkonversi karakter *plaintext* ke desimal dengan menggunakan kode ASCII

*Plaintext* = 65 77 65 78

Kemudian kunci akan dibangkitkan sepanjang *plaintext* dan dikonversikan kedalam desimal

Kunci = KNCI

Kunci = 75 78 67 73

Proses enkripsi akan dilakukan dengan persamaan :

$$C_i = (P_i + K_i) \bmod 256$$

$$C_i = (65 + 75) \bmod 256, \text{ maka } 140 \bmod 256 = 140$$

$$C_i = (77 + 78) \bmod 256, \text{ maka } 155 \bmod 256 = 155$$

$$C_i = (65 + 67) \bmod 256, \text{ maka } 132 \bmod 256 = 132$$

$$C_i = (78 + 73) \bmod 256, \text{ maka } 151 \bmod 256 = 151$$

Seluruh ciphertext yang masih dalam bentuk desimal dengan dikonversikan kembali kedalam bentuk karakter menjadi :

$$\text{Ciphertext} = \{140, 155, 132, 151\}$$

$$\text{Ciphertext} = \{\text{Æ}, <, ", \text{ù}\}$$

Proses dekripsi dilakukan dengan persamaan :

$$P_i = (C_i - K_i) \bmod 256$$

$$P_i = (140 - 75) \bmod 256, \text{ maka } 65 \bmod 256 = 65$$

$$P_i = (155 - 78) \bmod 256, \text{ maka } 77 \bmod 256 = 77$$

$$P_i = (132 - 67) \bmod 256, \text{ maka } 65 \bmod 256 = 65$$

$$P_i = (151 - 73) \bmod 256, \text{ maka } 78 \bmod 256 = 78$$

$$\text{Plaintext} = (65, 77, 65, 78)$$

$$\text{Plaintext} = \{A, M, A, N\}$$

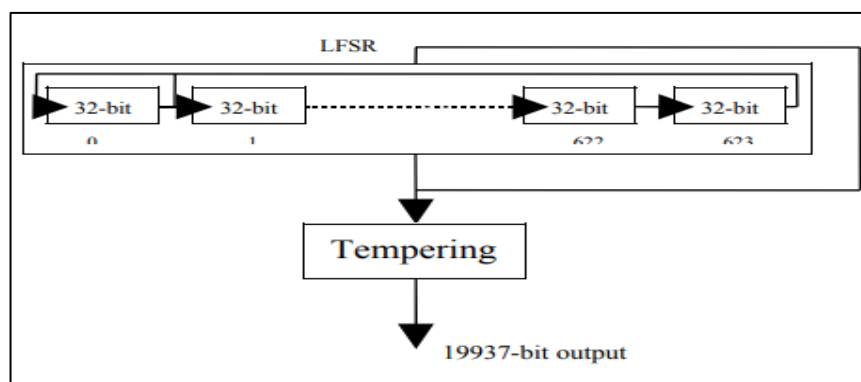
## 2.3 Algoritma *Mersenne Twister*

Algoritma *Mersenne Twister* merupakan algoritma untuk membangkitkan bilangan acak yang sering disebut dengan *Pseudorandom Number Generator* (PRNG) yang dikembangkan oleh Makoto Matsumoto dan Takuji Nishimura pada tahun 1996/1997. Algoritma ini menawarkan generasi yang cepat dari bilangan acak kualitas tinggi telah dirancang secara khusus untuk mengatasi setiap cacat yang ditemukan pada algoritma terdahulu (Siswanto, 2016).

### 2.3.1 Definisi Algoritma *Mersenne Twister*

*Mersenne Twister* adalah varian dari *Twisted Generalized Feedback Shift Register* (TGFSR) dimana algoritma ini peningkatan dari *Generalized Feedback Shift Register* (GFSR). *Mersenne Twister* dimodifikasi untuk mengambil sebuah array yang tidak lengkap kemudian direalisasikan untuk sebuah bilangan prima mersenne sebagai periodenya. Kemudian hasil algoritma *Mersenne Twister* memiliki panjang periode  $2^{19937}-1$  dengan dimensi pemerataan yang tinggi.

Ada dua versi algoritma yang saat ini paling umum digunakan, salah satunya adalah *Mersenne Twister* MT 19937. Walaupun *Mersenne Twister* ini dinyatakan aman dan tidak dapat diprediksi, sebenarnya ada kemungkinan untuk dimodifikasi agar aman digunakan dalam kriptografi. Terdapat beberapa varian dari *Mersenne Twister* termasuk CryptMT, MTGP, TinyMT, dan SFMT.



**Gambar 2.4.** Diagram block dari MT 19937

Berdasarkan diagram block gambar 2.4 menjelaskan bahwa *Mersenne Twister* didasarkan pada persamaan linear berulang atas bidang dari linear sebagai berikut:

$$x_{k+n} = x_{k+m} \oplus (x_k^u | x_{k+1}^l) \text{ A } (k=0,1,\dots).$$

Yang dapat didefinisikan sebagai berikut :

$n$  = Sebagai tingkat pengulangan atau drajat rekursi yakni 624

$k$  = Nilai 0,1,2,3, ....

$x_n$  = Vektor baris ukuran yang digenerate dengan  $k = 0$

$x_0, x_1, \dots, x_{n-1}$  = Inisial dari *seeds*

$m$  = Batas nilai tengah dimana  $1 \leq m \leq n$  yaitu 397

$x_{k+1}^l$  = Bit  $r$  yang paling rendah atau paling kanan dari  $x_{k+1}$

$x_k^u$  = Bit  $w - r$  yang paling tinggi atau kiri dari  $x_k$

$\oplus$  = Notasi dari *bitwise XOR*

$|$  = Operator *concatenate* (penggabungan)

### 2.3.2 Parameter dari 32-bit MT

Berikut ini merupakan parameter-parameter yang dibutuhkan dalam proses pembangkitan bilangan acak semu MT 19937 yang pada dasarnya ini bersifat baku dan sudah memiliki ketetapan yang konsisten.

Tabel 2.1 Parameter MT 19937 32 bit

Parameter	Qty	Description
$n$	624	<i>Degree of recurrence (state size)</i>
$w$	32	<i>Word Size</i>
$r$	1-31	<i>Mask Bits <math>0 &lt; r \leq w-1</math></i>
$m$	397	<i>Middle Word or the number parallel sequence <math>1 \leq m \leq n</math></i>
$a$	9908B0DF	<i>Coefficients of the rational normal from twist matrix (Xor Mask)</i>
$u$	11	<i>Additional MT Tempering Bit Shift</i>
$s$	7	<i>TGFSR (r) MT Tempering Bit Shif</i>

t	15	<i>TGFSR (r) MT Tempering Bit Shift</i>
l	18	<i>Additional MT Tempering Bit Shift</i>
b	9D2C5680	<i>TGFSR (r) MT Tempering Bit Shift</i>
c	EFC60000	<i>TGFSR (r) MT Tempering Bit Shift</i>

### 2.3.3 Proses Generate Algoritma MT 19937

Berikut ini langkah-langkah dari algoritma *Mersenne Twister* seperti yang ditunjukkan dibawah ini :

**Step 0.** Menentukan bitmask dari yang tertinggi dan terendah

$U \leftarrow \underbrace{1 \dots 1}_{w-r} \underbrace{0 \dots 0}_r$  adalah sebuah bitmask tertinggi dari bit-bit  $w-r$

$l \leftarrow \underbrace{0 \dots 0}_{w-r} \underbrace{1 \dots 1}_r$  adalah sebuah bitmask terendah dari bit-bit  $r$

$a \leftarrow a_{w-1} a_{w-2} \dots a_1 a_0, a = a_{w-1}, a_{w-2}, \dots, a_0$  adalah vektor garis bawah dari  $A$

**Step 1.** Menentukan array  $x$  dari nilai yang bukan nol

$x[0], x[1], x[2], \dots, x[n-1]$  nilai awal tidak bernilai nol (hingga  $n-1$ )  
misalnya  $n=8$  maka array  $x$  nya hanya sampai 7.

**Step 2.** Menghitung nilai bit tertinggi dari  $x[k]/x_{w-1}$  dengan penggabungan bit rendah dari  $x[k+1]$  sehingga dari penggabungan bit rendah dan bit tinggi dari  $x[k+1]$  menghasilkan vektor dimensi dengan adanya perkalian  $(x_i^u | x_{k+1}^1)$ . Proses ini disebut dengan *twist*.

**Step 3.** Menghitung nilai twist dengan pengoperasian bit berikut :

$$x_i = \begin{cases} x \gg 1 & \text{if } x_0 = 0 \\ (x \gg 1) \oplus a & \text{if } x_0 = 1 \end{cases}$$

Geser ke kanan sebanyak satu kali jika  $x[i] / x_{w-1} = 0$  dan geser ke kanan sebanyak 1 dan *XOR* dengan  $a$  jika  $= 1$ .

**Step 4.** Ketika  $x[k] / x_w -$  dikalikan dengan  $T$  untuk menghasilkan distribusi yang lebih baik, temper digunakan untuk menganalisis angka acak yang muncul pada putaran berikutnya.

```

y ← x[i]
y ← y ⊕ (y >> u)
y ← y ⊕ ((y << s) & b)
y ← y ⊕ ((y << t) & c)
y ← y ⊕ (y >> I)
output y

```

**Step 5.** Meningkatkan nilai  $i$  sebesar 1.

```
i ← (i+1) mod n
```

**Step 6.** Lakukan kembali proses tersebut secara langsung ke langkah 2.

## 2.4 Algoritma *Dependent RSA 1* (DRSA-1)

RSA adalah sebuah algoritma kriptografi kunci publik yang ditemukan pertama kali oleh Ron Rivest, Adi Shamir, dan Leonard Adleman pada tahun 1977. Nama RSA sendiri diambil dari inisial nama ketiga penemu tersebut (Ginting et al., 2015).

### 2.4.1 Definisi Algoritma *Dependent RSA 1* (DRSA-1)

Algoritma asimetris yang dikenal sebagai Algoritma Rivest (RSA) Shamir Adleman menggunakan kunci berbeda untuk enkripsi dan dekripsi. Dalam algoritma ini, kunci publik dipakai dalam enkripsi pesan sementara kunci pribadi dipakai untuk dekripsi. Keamanan RSA tergantung pada kemampuan untuk memfaktorkan bilangan bulat besar yang merupakan hasil perkalian dua faktor prima. Semakin besar bilangan bulat tersebut, semakin sulit bagi penyerang untuk memecahkan pesan secara tidak sah. Oleh karena itu, keamanan pesan yang dienkripsi menggunakan algoritma RSA tergantung pada kompleksitas faktorisasi bilangan bulat tersebut (Pointcheval, 1999).

Aman secara semantik merupakan istilah yang dikemukakan oleh Goldwasser dan Micali yang bisa didefinisikan bahwa suatu kondisi dimana hanya

sedikit informasi mengenai teks asli yang dapat ditarik oleh musuh dari teks yang tersembunyi (*ciphertext*) yang berarti *ciphertext* tidak membocorkan apapun tentang *plaintext*. Untuk mengatasi hal ini Pointcheval mengusulkan varian baru dari algoritma RSA yaitu dalam skema enkripsi *Dependent RSA 1* (DRSA-1). Algoritma DRSA-1 adalah sebuah algoritma yang terbukti aman secara semantik terhadap penyajian teks sandi yang terpilih terhadap serangan yang tidak bertanggung jawab.

Dalam proses pembangkitan bilangan kunci algoritma *Dependent RSA 1* (DRSA-1) memiliki tahapan-tahapan antara lain :

1. Pilihlah dua bilangan acak yang tidak memiliki faktor bersama  $p$  dan  $q$ , dan  $p$  tidak sama dengan  $q$  dan  $p$  serta  $q$  dijaga kerahasiaannya.
2. Publikasikan nilai  $n$  sebagai kunci publik dengan menggunakan rumus  $n = p \times q$ , dimana  $n$  bukan rahasia.
3. Menghitung  $\Phi(n) = (p - 1)(q - 1)$ .
4. Menentukan nilai  $e$  dengan syarat bahwa  $e$  adalah bilangan relatif prima terhadap  $\Phi(n)$ , dan  $e$  berada dalam rentang lebih besar dari 2 dan lebih kecil dari  $\Phi(n)$ . Nilai  $e$  dan  $n$  akan dipakai sebagai kunci publik.
5. Menghitung nilai  $d$ , dimana  $d = e^{-1} \pmod{\Phi(n)}$ . Nilai  $d$  digunakan sebagai kunci rahasia.
6. Pengirim mempublikasikan nilai  $e$  dan  $n$  sebagai kunci publik dan nilai  $d$  sebagai kunci rahasia.

#### 2.4.2 Landasan Matematika Algoritma *Dependent RSA 1* (DRSA-1)

Algoritma kriptografi dibangun dengan landasan matematika sehingga pemahaman terhadap perhitungan dasar matematika menjadi hal yang wajib dalam mempelajari ilmu kriptografi. Berikut landasan matematika algoritma *Dependent RSA 1* (DRSA-1) :

### 1. Aritmatika Modulo

Aritmatika modulo yaitu sisa suatu integer jika dibagi dengan integer lain atau sisa pembagian dari satu bilangan terhadap bilangan lain. Bentuk umum dari modulo dapat dinyatakan sebagai berikut :

$m \bmod n = r$ , dimana nilai  $r$  merupakan sisa pembagian  $m$  dan  $n$ . demikian pula apabila  $m \bmod n = r$ , maka pasti akan berlaku nilai  $p$  :  $m+p \bmod n = r+p \bmod n$

Contoh :

$$13 \bmod 3 = 1$$

Untuk  $p = 3$ , maka  $13 + 3 \bmod 3 = 1$ , dan  $1 + 3 \bmod 3 = 1$

$$32 \bmod 3 = 2$$

Untuk  $p = 4$ , maka  $32 + 4 \bmod 3 = 0$ , dan  $2 + 4 \bmod 3 = 0$

### 2. Invers Modulo

Apabila  $a^{-1} \times a \equiv 1 \pmod{n}$  maka  $a^{-1}$  adalah invers dari  $a \pmod{n}$ . Syarat agar  $a \pmod{n}$  memiliki invers maka  $a$  harus relatif prima terhadap  $n$  ( $\text{GCD } m, n = 1$ ).

Contoh : Berapa invers dari  $7 \pmod{17}$  ?

Tabel 2.2 Invers Modulo

$m^{-1}$	$1 \times m \pmod{n}$ $1 \times 7 \pmod{17}$
1	$1 \times 7 \pmod{17} = 7$
2	$2 \times 7 \pmod{17} = 14$
3	$3 \times 7 \pmod{17} = 4$
4	$4 \times 7 \pmod{17} = 11$
5	$5 \times 7 \pmod{17} = 1$

Dari penyelesaian contoh invers  $7 \pmod{17}$  dari tabel diatas akan berhenti apabila ketika nilai  $m^{-1} \times 7 \pmod{17} = 1$ , dan pada perhitungan kali ini berhenti pada iterasi yang ke tiga maka  $m^{-1} = 5$ .



### 3. Modulo Eksponensial

Bentuk umum dari modulo eksponensial adalah  $xy \pmod{n}$ , dimana  $x, y, n$  merupakan bilangan bulat. Dalam kriptografi operasi eksponensial yang sering digunakan untuk proses perhitungan enkripsi dan dekripsi.

Contoh : dengan cara iteratif, Berapa  $8^4 \pmod{40}$  ?

Penyelesaian :

$$z = 1$$

$$i = 1, z = 1 \times 8 \pmod{40} = 8$$

$$i = 2, z = 8 \times 8 \pmod{40} = 24$$

$$i = 3, z = 24 \times 8 \pmod{40} = 32$$

$$i = 4, z = 32 \times 8 \pmod{40} = 16$$

dari penyelesaian diatas maka hasil  $8^4 \pmod{40}$  adalah 16.

### 4. Relatif Prima

Bilangan  $a$  dan  $b$  disebut relatif prima jika faktor persekutuan terbesar (GCD) dari kedua bilangan tersebut adalah 1.

Contoh : Apakah 4 dan 9 relatif prima ?

$$\text{GCD}(4, 9)$$

$$4 \pmod{9} = 4$$

$$9 \pmod{4} = 1$$

$$4 \pmod{1} = 0$$

Maka  $\text{GCD}(4, 9) = 1$  adalah relatif prima.

### 5. Algoritma Euclidean

Pembagi Persekutuan Terbesar (GCD), juga dikenal sebagai FPB, dapat ditemukan dengan bantuan algoritma yang disebut algoritma Euclidean. Istilah matematika "faktor persekutuan terbesar" adalah "FPB". Algoritma ini juga berguna untuk memeriksa apakah dua bilangan adalah relatif prima atau tidak.  $\text{GCD}(m, n) = r$  merupakan formulasi umum dari algoritma Euclidean.

Langkah-langkah algoritma euclidean adalah :

1. Bagi nilai  $m$  dengan  $n$  apabila hasilnya  $(r) = 0$ , maka  $n$  merupakan  $\text{GCD}(m, n)$  dan berhenti.

2. Jika  $r$  tidak sama dengan nol, maka nilai  $m$  dan  $n$  diubah, dan nilai  $n$  menjadi  $r$ , kemudian langkah pertama diulang.

Contoh : berapa GCD (123, 277) ?

$$123 \bmod 277 = 123$$

$$277 \bmod 123 = 31$$

$$123 \bmod 31 = 30$$

$$31 \bmod 30 = 1 \text{ (relatif prima setelah 0)}$$

$$30 \bmod 1 = 0 \text{ (berhenti)}$$

Jadi  $\text{GCD}(123, 277) = 1$  maka disebut dengan relatif prima

Berapa GCD (100, 26) ?

$$100 \bmod 26 = 22$$

$$26 \bmod 22 = 4$$

$$22 \bmod 4 = 2 \text{ (tidak relatif prima)}$$

$$4 \bmod 2 = 0 \text{ (berhenti).}$$

## 6. Bilangan Prima

Bilangan prima adalah bilangan bulat positif yang memiliki tepat dua faktor, yaitu 1 dan bilangan itu sendiri. Dengan kata lain, bilangan prima hanya dapat dibagi oleh 1 dan bilangan itu sendiri. Satu-satunya bilangan prima ganjil adalah 2. Bilangan-bilangan seperti 2, 3, 5, 7, 11, 13, 17, 19, 23, dan seterusnya semua termasuk dalam kategori bilangan prima. Yang membedakan bilangan-bilangan tersebut adalah tidak adanya faktor pembagi selain 1 dan bilangan itu sendiri.

## 7. Totient Euler

Totient Euler disimbolkan  $\Phi(n)$ , yang diartikan sebagai jumlah bilangan bulat positif lebih kecil dari  $n$ .

Contoh :  $\Phi(7) = ?$

Bilangan asli yang lebih kecil dari  $7 = \{1, 2, 3, 4, 5, 6\}$  dan yang relatif prima terhadap 7 adalah  $\{1, 5\}$ .

Beberapa ketentuan dari *Totient Euler* sebagai berikut :

1. Apabila  $n$  adalah bilangan prima, maka  $\Phi(n) = n - 1$ .

$$\text{Contoh : } \Phi(7) = \{1, 2, 3, 4, 5, 6\}$$

2. Apabila  $n = p \times q$  dan  $p$  relatif prima terhadap  $q$ , maka  $\Phi(n) = \Phi(p) \times \Phi(q)$ .

Contoh :  $\Phi(7) = ?$

$$n = 30 = 6 \times 5$$

$\text{GCD}(6,5) = 1$ , dan relatif prima

$$\Phi(6) = 2$$

$$\Phi(5) = 4$$

$$\Phi(30) = \Phi(6) \times \Phi(5) = 2 \times 4 = 8, \text{ maka hasil } \Phi(30) = 8.$$

3. Apabila  $n = p \times q$  dimana  $p$  dan  $q$  merupakan bilangan prima, maka  $\Phi(n) = (p - 1)(q - 1)$ .

Contoh :  $\Phi(35) = ?$

$$n = 35 = 7 \times 5$$

$$\Phi(35) = (7 - 1)(5 - 1) = 24.$$

4. Apabila  $n = p_1, p_2, p_3, \dots, p_n$  merupakan bilangan prima yang berbeda, maka  $\Phi(n) = (p_1 - 1)(p_2 - 1)(p_3 - 1)$ .

Contoh :  $\Phi(1155) = ?$

$$n = 1155 = 3 \times 5 \times 7 \times 11$$

$$\Phi(n) = (3 - 1)(5 - 1)(7 - 1)(11 - 1) = 480$$

5. Apabila  $n = p^k$  dan  $p$  adalah bilangan prima, maka  $\Phi(n) = p^k - p^{k-1}$

Contoh :  $\Phi(25) = ?$

$$n = 25 = 5^2$$

$$\Phi(n) = 5^2 - 5^1 = 10$$

8. Metode bangkit bilangan prima *lehmann*

Dengan menggunakan pembangkit bilangan prima Lehmann, langkah-langkah menentukan apakah  $p$  bilangan prima adalah sebagai berikut:

1. pertama adalah memilih nilai secara acak  $a$  sehingga  $a$  kurang dari  $p$ .
2. kedua melibatkan perhitungan  $p \bmod 2$ , jika hasilnya adalah 0, maka dapat disimpulkan bahwa  $p$  bukanlah bilangan prima.
3. ketiga adalah menghitung  $a^{(p-1)/2} \bmod p$ .
4. Jika hasil dari perhitungan  $a^{(p-1)/2} \neq 1$  atau  $-1 \pmod{p}$ , maka dapat dipastikan bahwa  $p$  bukanlah bilangan prima.
5. Apabila hasil dari perhitungan  $a^{(p-1)/2} = 1$  atau  $-1 \pmod{p}$  kemungkinan bahwa  $p$  bukanlah bilangan prima tidak lebih dari 50%.

Contoh :

Diberikan sebuah bilangan  $p=5$ , dipilih  $a = 2,3,4$

$$1. a^{(p-1)/2} \bmod p \equiv 2^{(5-1)/2} \pmod{5}$$

$$\equiv 2^2 \pmod{5}$$

$$\equiv 4 \pmod{5}$$

$$\equiv -1$$

$$2. a^{(p-1)/2} \bmod p \equiv 3^{(5-1)/2} \pmod{5}$$

$$\equiv 3^2 \pmod{5}$$

$$\equiv 9 \pmod{5}$$

$$\equiv 4 \pmod{5}$$

$$\equiv -1$$

$$3. a^{(p-1)/2} \bmod p \equiv 4^{(5-1)/2} \bmod 5$$

$$\equiv 4^2 \bmod 5$$

$$\equiv 16 \bmod 5$$

$$\equiv 1$$

Kemungkinan  $p$  prima = 87,5%

$P$  dikatakan prima karena peluangnya menjadi prima adalah 87,5 %. Generator bilangan prima Lehmann dipilih karena proses pembangkitan bilangan dapat digunakan untuk menentukan apakah suatu bilangan prima atau bukan hanya dengan mengulangnya.

#### 2.4.3 Prinsip kerja algoritma *Dependent RSA 1 (DRSA-1)*

Dalam algoritma RSA terdapat beberapa proses, diantaranya proses pembangkit bilangan kunci, proses enkripsi, proses dekripsi. Berikut ini adalah proses dari algoritma yang dilakukan :

##### 1. Proses pembangkit bilangan kunci

- a. Memilih dua buah bilangan prima  $p$  dan  $q$  yang besar secara acak, dimana  $p \neq q$ . Nilai  $p$  dan  $q$  bersifat rahasia serta  $p$  dan  $q$  akan dibangkitkan dengan menggunakan algoritma lehmann.
- b. Perhitungan  $n = p \times q$
- c. Perhitungan  $\Phi(n) = (p - 1)(q - 1)$

- d. Memilih sebuah bilangan ganjil  $e$ , dimana  $2 < e < \Phi(n)$  dan relatif prima terhadap  $\Phi(n)$
  - e. Menghitung  $d$  yang memenuhi  $d \times e \pmod{\Phi(n)} = 1$ . Dan nilai  $d$  sebagai kunci rahasia.
  - f. Pengirim mempublikasikan nilai  $e$  dan  $n$  sebagai kunci publik dan nilai  $d$  sebagai kunci rahasia.
2. Proses Enkripsi (*sender*)
- a. Pilih pesan yang akan dienkripsi.
  - b. Gunakan kunci publik  $e$  dan  $n$ .
  - c. Memilih angka random  $K$  dimana  $K$  bisa integer antara 1 sampai  $(n-1)$
  - d. Menghitung  $A = K^e \pmod n$
  - e. Menghitung  $B = M (K + 1)^e \pmod n$
  - f. Menghitung  $H = H(M, K)$
  - g. Mengirimkan Ciphertext  $(A, B, H)$  ke penerima
3. Proses Dekripsi
- a. Penerima mendapatkan ciphertext  $C = (A, B, H)$  dari pengirim
  - a. Penerima menghitung  $K = A^d \pmod n$
  - b. Penerima menghitung  $K = B / (K + 1)^e \pmod n$
  - c. Penerima mengecek apakah  $H = H(M, K)$

#### 2.4.4 Contoh perhitungan algoritma *Dependent RSA 1 (DRSA-1)*

Berikut ini adalah ilustrasi langsung Algoritma *Dependent RSA 1 (DRSA-1)* dalam membangkit kunci, enkripsi, dan dekripsi:

1. Proses Pembangkit Kunci
- a. Membangkitkan dua bilangan prima besar yaitu  $p = 3$ ,  $q = 7$ , dengan dibangkitkan algoritma *lehmann*
  - b. Menghitung nilai  $n = p \times q$ 

$$n = 3 \times 7 = 21$$
  - c. Menghitung  $\Phi(n) = (p - 1) (q - 1)$ 

$$\Phi(n) = (3 - 1) (7 - 1)$$

$$= 2 \times 6 = 12$$
  - d. Memilih bilangan  $e$  secara acak  $2 < e < \Phi(n)$  dan relatif prima terhadap  $\Phi(n)$ 

Tes  $e = 7$

$$12 \pmod 7 = 5$$

$$7 \bmod 5 = 2$$

$$5 \bmod 2 = 1$$

$$2 \bmod 1 = 0, \text{ karena } \text{GCD}(12,7) = 1, \text{ maka relatif prima.}$$

- e. Menghitung  $d$  dimana  $d \equiv e^{-1} \pmod{\Phi(n)}$  dan biarkan nilai tersebut.
- f. Mempublikasikan  $e$  dan  $n$  sebagai kunci publik.
- g. Menyimpan nilai  $d$  sebagai kunci rahasia.

## 2. Proses Enkripsi

- a. Pilih pesan yang hendak di enkripsi,  $m$  "14".
- b. Penerima menerima *public key* dari si pengirim.
- c. Pengirim memilih angka random integer antara 1 dan  $n-1$ ,  $k = 9$ .
- d. Pengirim menghitung  $A$

$$A = K^e \bmod n$$

$$A = 9^7 \bmod 21$$

$$A = 9$$

- e. Pengirim menghitung  $B$

$$B = m(k + 1)^e \bmod n$$

$$B = 14(9 + 1)^7 \bmod 21$$

$$B = 14(10) \bmod 21 = 14$$

- f. Pengirim menghitung  $H$

$$H = M^k \bmod 100$$

$$H = 14^9 \bmod 100$$

$$H = 84$$

- g. Pengirim mengirimkan *ciphertext*  $(A,B,H)$  kepada penerima

## 3. Proses Dekripsi

- a. Penerima menerima *ciphertext* dari si pengirim  $(A,B,H) = (9, 14, 84)$
- b. Penerima menghitung  $K$

$$K = A^d \bmod n$$

$$K = 9^7 \bmod 21$$

$$K = 9$$

- c. Penerima menghitung  $M$

$$M = B/(K + 1)^e \bmod n$$

$$M = 14/(9 + 1)^7 \bmod 21$$

$$M = 14/(10)^7 \text{ mod } 21$$

$$M = (14)(10^{-1})^7 \text{ mod } 21$$

$$M = (14)(19)^7 \text{ mod } 21$$

$$M = (14)(19) \text{ mod } 21$$

$$M = 266 \text{ mod } 21$$

$$M = 14 \text{ (terbukti)}$$

d. Penerima mengecek apakah  $H = h(m,k)$

$$H = m^k \text{ mod } 100$$

$$H = 14^9 \text{ mod } 100$$

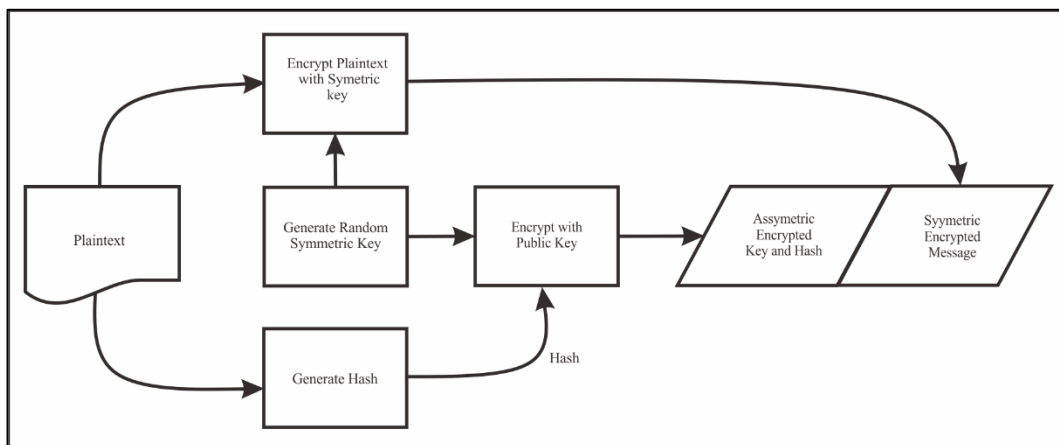
$$H = 84$$

## 2.5 Metode *Hybrid Cryptosystem*

*Hybrid Cryptosystem* adalah sebuah konsep yang digunakan untuk kriptografi dalam meningkatkan keamanan data yang dikirimkan melalui jaringan komputer. Konsep ini menggabungkan dua atau lebih algoritma kriptografi yang berbeda yang berguna untuk melindungi informasi yang terkandung dalam pesan tersebut. Hal ini bertujuan untuk mengatasi kelemahan yang terdapat pada kriptosistem tunggal. Dalam metode *Hybrid Cryptosystem* enkripsi dilakukan dengan menggunakan dua metode enkripsi utama, yaitu enkripsi simetris melibatkan penggunaan kunci yang sama baik dalam proses enkripsi maupun dekripsinya, sementara enkripsi asimetris menggunakan dua kunci yang berbeda, yaitu kunci publik dan kunci privat (Kaur, 2020).

Pesan dienkripsi dan didekripsi menggunakan kriptografi kunci simetris pada skema *Hybrid Cryptosystem*, sedangkan kunci dienkripsi dan didekripsi menggunakan kriptografi kunci asimetris dengan kunci publik. Kunci publik penerima kemudian akan digunakan untuk mengacak kunci pertemuan sebelum mengirimkannya bersama dengan pesan yang disandikan. Penerima mendekripsi pesan menggunakan kunci pribadinya terlebih dahulu sebelum menggunakan kunci sesi untuk mendekripsinya..

Kriptografi *Hybrid Cryptosystem* sering digunakan karena kecepatan algoritma simetris memproses data dan kemudahan algoritma asimetris mentransfer kunci. Hasilnya, kecepatan meningkat tanpa mengorbankan kenyamanan atau keamanan data.



**Gambar 2.5.** Mekanisme Hybrid Cryptosystem  
(Informatika et al., 2023)

## 2.6 Penelitian Relevan

Berikut studi terdahulu yang relevan dengan penelitian saat ini :

1. Menurut artikel yang dipublikasikan dalam jurnal "Penerapan Signcryption dengan Algoritma *Dependent RSA 1* (DRSA-1) dan Algoritma Elgamal Linear Umum untuk Keamanan Data Teks," penerapan metode Signcryption menggunakan algoritma *Dependent RSA 1* (DRSA-1) dan Algoritma Elgamal Linear Umum dalam melindungi data teks dinyatakan menghasilkan kesimpulan bahwa penggunaan algoritma *Dependent RSA 1* akan memberikan tingkat keamanan yang sangat tinggi (Riza, 2021).
2. Menurut kesimpulan yang dibuat dalam jurnal "On Using The First Variant of *Dependent RSA 1* Encryption Scheme to Secure Text : A Tutorial" algoritma Rivest Shamir Adleman (RSA) adalah algoritma asimetris yang menggunakan kunci berbeda untuk enkripsi dan dekripsi. untuk menggunakan kunci publik untuk mengenkripsi pesan dan kunci rahasia atau pribadi untuk dekripsi. Kekuatan faktorbilangan bulat besar, yang merupakan



dua faktor besaran prima, menentukan keamanan algoritma RSA; semakin besar bilangan bulatnya, semakin aman sistem kriptografi RSA (Rachmawati & Budiman, 2020).

3. Penulis artikel yang dimuat dalam jurnal "Implementasi Enkripsi dan Dekripsi Dokumen Menggunakan Algoritma *One Time Pad* (OTP) dengan *Mersenne Twister* sebagai Pembangkit Bilangan Acak" menyimpulkan bahwa pengujian yang dilakukan selama proses enkripsi dokumen dengan menggunakan algoritma *One Time Pad* dan *Mersenne Twister* sangat tergantung pada kunci acak yang digunakan, dengan waktu rata-rata sekitar 0,00322 detik. Selain itu, proses dekripsi dokumen memerlukan waktu sekitar 0,002602619 detik secara normal (Fidelis Asterina Surya Prasetya, 2016).
4. Pada jurnal "*Mersenne Twister* - A Pseudo Random Number Generator and its Variants" yang menyimpulkan bahwa algoritma *Mersenne Twister* secara teoritis terbukti merupakan algoritma PRNG yang baik. Dimana cacat yang ditemukan oleh pengguna telah diperbaiki oleh penemunya. *Mersenne Twister* telah ditingkatkan, agar dapat digunakan dan kompatibel dengan teknologi CPU yang baru muncul seperti SIMD dan saluran paralel dalam versi SFMT-nya. Dan *Mersenne Twister* juga telah ditingkatkan agar aman secara kriptografis yaitu dengan CryptMT (Jagannatham, 1997).

## BAB 3

### ANALISIS DAN PERANCANGAN

#### 3.1 Analisis Sistem

Pendekatan pemecahan persoalan yang dikenal dengan analisis sistem membagi suatu sistem menjadi beberapa subsistem dengan tujuan untuk memeriksa bagaimana masing-masing komponen berfungsi dan berinteraksi satu sama lain untuk mencapai tujuan tertentu. Selain itu, analisis sistem digunakan untuk menganalisis kebutuhan sistem sehingga dapat diperbaiki di masa depan.

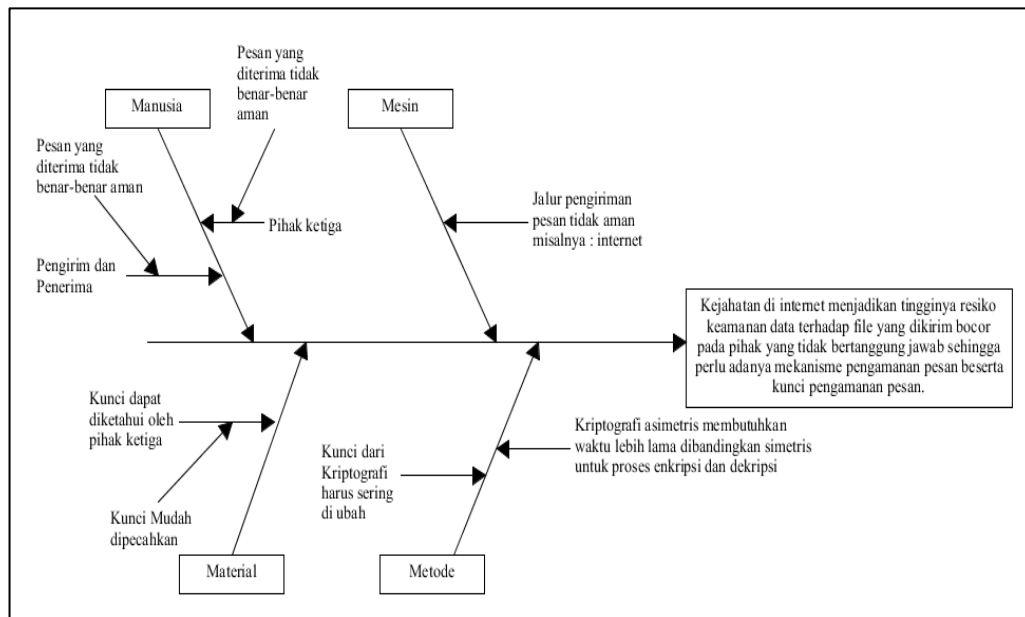
Analisis sistem melibatkan tiga tahap, yakni analisis masalah, analisis kebutuhan, dan analisis proses. Analisis masalah merupakan suatu metode untuk mengenali suatu isu dan menelusuri tujuan di balik isu tersebut. Deskripsi kemampuan dan fungsi sistem merupakan tujuan dari analisis kebutuhan. Mendemonstrasikan cara berperilaku kerangka kerja adalah tujuan penyelidikan siklus.

##### 3.1.1 Analisis Masalah

Seiring kemajuan teknologi setiap hari, mungkin terdapat peningkatan jumlah ancaman terhadap keamanan data dan informasi penting. Selain itu, keingintahuan manusia terhadap akses informasi pun semakin meningkat. Akibatnya seseorang (pihak ketiga) dapat mengakses suatu informasi secara *illegal*. Misalkan pihak ketiga dapat mengetahui pesan informasi yang dikirim oleh pihak pengirim ke penerima. Masalah utama yang dibahas dalam penelitian ini adalah bagaimana Hybrid Cryptosystem dapat digunakan untuk melindungi keamanan dan kerahasiaan teks yang dikirim..

Masalah penelitian ini diidentifikasi dengan menggunakan diagram *Ishikawa*. Diagram *Ishikawa* adalah diagram yang digunakan untuk memahami dan mengidentifikasi serta menggambarkan masalah yang terjadi dan akibat dari masalah yang ditimbulkan.

Secara keseluruhan, permasalahan ini dapat diperinci melalui sebuah diagram *Ishikawa*, yang akan ditampilkan sebagai berikut :



**Gambar 3.1.** Diagram *Ishikawa*

Dilihat dari gambar 3.1, masalah ini berasal dari kepentingan pihak luar sehubungan dengan privasi pesan yang disandikan karena perubahan ciphertext menggunakan kunci yang tidak dapat diandalkan saat mengirim pesan ke pihak lain.

Opsi yang tersedia untuk melakukan pengamanan *plaintext* adalah dengan mengunci *plaintext* tersebut dengan menggunakan kunci simetris dan *ciphertext* tersebut dikunci dengan menggunakan kunci asimetris, pengiriman pesan dengan dilakukan dengan teknik *Hybrid Cryptosystem* antara *One Time Pad (OTP)* dengan *Mersenne Twister* dan *Dependent RSA 1 (DRSA-1)*.

### 3.1.2 Analisis kebutuhan

Selama tahap analisis kebutuhan, terdapat dua jenis kebutuhan yang perlu dipertimbangkan yaitu kebutuhan fungsional dan kebutuhan non-fungsional.

#### 1. Kebutuhan Fungsional

Kebutuhan Fungsional bertujuan untuk memahami proses tindakan yang dapat dilakukan oleh sistem dalam menjalankan administrasi dan berinteraksi dengan informasi serta persyaratan yang perlu dipenuhi oleh sistem :

a. Fungsi Pembangkit Kunci

Pesan dikirim dan diterima oleh penerima dan sistem. Untuk mengerjakan proses enkripsi dan dekripsi, kemampuan membangkit kunci dapat dilakukan dengan mengkoordinasikan informasi atau dibangkit secara acak.

b. Fungsi Enkripsi

Pengguna memiliki kemampuan untuk mengubah pesan dari *plaintext* menjadi ciphertext dengan menggunakan kunci yang telah dihasilkan oleh fungsi pembangkit angka acak.

c. Fungsi Dekripsi

Jika sistem memiliki kunci dekripsi penerima, sistem dapat mendekripsi atau mengubah teks tersandi menjadi teks biasa. Lebih jauh lagi, dalam kemampuan ini kerangka kerja berfungsi sebagai penerima pesan.

d. Fungsi Penguncian Kunci Pesan

Dengan menggunakan kunci publik yang telah diubah menjadi kunci sandi, pengguna dapat mengenkripsi kunci pesan dalam fungsi ini.

2. Kebutuhan Non-Fungsional

Persyaratan non-fungsional mencakup keamanan, mudah dipelajari dan digunakan, struktur penyimpanan, dokumentasi, kontrol, dan ekonomi, di antara fitur, karakteristik, dan batasan lainnya. Ada beberapa persyaratan non-fungsional, antara lain:

a. Keamanan

Sistem harus memastikan bahwa data yang dikirim dan diterima aman dari akses yang tidak sah dengan menggunakan algoritma kriptografi seperti OTP dan DRSA 1 dan lain-lain.

b. Mudah untuk dipelajari dan digunakan

Untuk memahami sistem ini user lama masih memerlukan sekitar 30 menit sedangkan untuk user baru masih memerlukan 1 sampai 2 jam.

c. Struktur Penyimpanan

Informasi teks yang terbentuk, baik dalam bentuk *plaintext* maupun ciphertext, disimpan di lokasi yang telah ditentukan sebelumnya.

d. Dokumentasi

Sistem yang dibangun memiliki panduan penggunaan aplikasi.

e. Kontrol

Jika pengguna tidak memasukkan informasi yang cukup atau tidak memenuhi persyaratan, sistem bawaan akan menampilkan pesan kesalahan (*error*).

f. Ekonomi

Tidak ada kebutuhan akan perangkat atau biaya tambahan untuk menggunakan sistem yang dibangun.

### 3.1.3 Analisis Proses

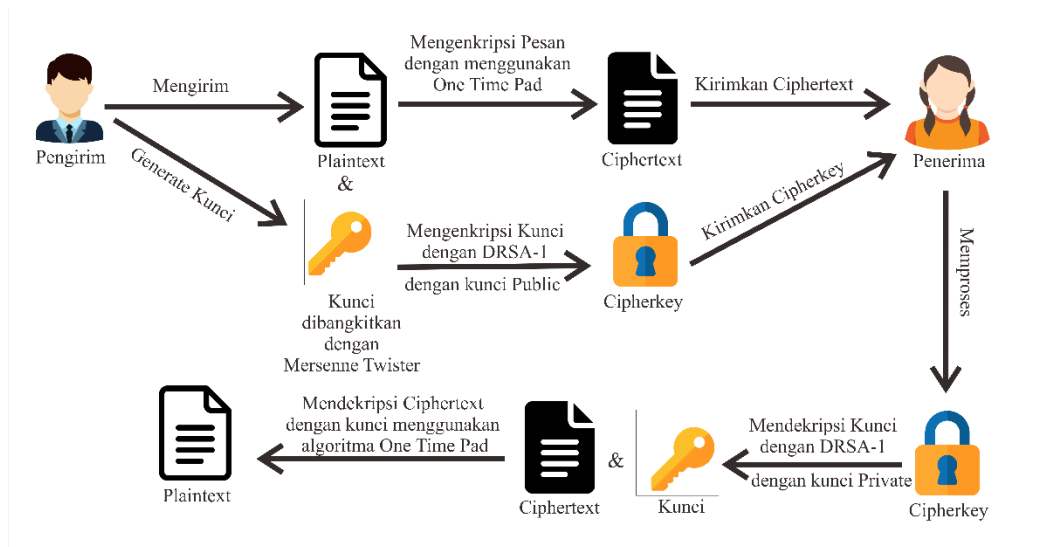
Untuk keperluan enkripsi dan dekripsi pesan *plaintext*, penelitian ini memakai *One Time Pad (OTP)*, dan *Mersenne Twister* untuk pembangkitan angka acak, serta *Dependent RSA 1 (DRSA-1)* untuk dekripsi dan enkripsi pesan kunci yang dihasilkan.

Enkripsi dan dekripsi dilakukan dengan menggunakan *One Time Pad (OTP)* untuk mendapatkan hasil berupa *ciphertext* dan setelah di dekripsi akan mendapatkan hasil berupa *plaintext*. Pada algoritma *Mersenne Twister* dipakai untuk menghasilkan kunci yang digunakan untuk enkripsi *plaintext*.

Pada *Dependent RSA 1 (DRSA-1)* kunci yang telah dibangkitkan dengan penggunaan pembangkit bilangan kunci akan di enkripsi kembali dengan membangkitkan kunci dengan memanfaatkan  $p$  dan  $q$ , sehingga  $n$  dan  $e$  menjadi kunci public dan  $d$  menghitung invers  $e$  di kali dengan totient euler dari  $n$  akan menjadi kunci privat.

## 3.2 Diagram Umum Sistem

Diagram Umum adalah perancangan penting yang menggambarkan proses, alur dan interaksi antar komponen dalam suatu sistem. Berikut adalah deskripsi dari tindakan pengguna yang terkait dengan sistem.



**Gambar 3.2.** Diagram Umum Sistem

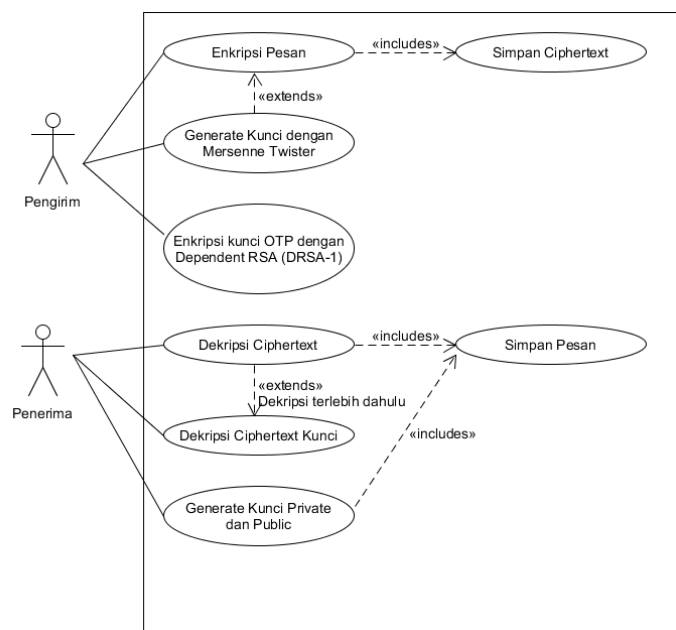
Proses dimulai dengan pengirim membuat *plaintext* terlebih dahulu lalu *plaintext* tersebut akan dienkripsi dengan *One Time Pad (OTP)* serta kunci untuk mengenkripsi *plaintext* dibangkitkan dengan *Mersenne Twister*, lalu kunci tersebut akan *dienkripsi* kembali dengan menggunakan kunci public *Dependent RSA 1 (DRSA-1)* sehingga *ciphertext* dan *cipherkey* akan diterima oleh penerima setelah itu *cipherkey* tersebut akan di dekripsi kembali dengan menggunakan kunci privat *Dependent RSA 1 (DRSA-1)* lalu kunci yang di dekripsi akan memunculkan kunci untuk mendekripsikan *ciphertext* dengan *One Time Pad (OTP)* sehingga muncul *plaintext* yang asli.

### 3.3 Pemodelan Sistem

Dalam pembuatan sistem proses merancang perangkat lunak atau sistem sebelum pelaksanaan dalam bahasa mesin. Pemodelan sistem melibatkan penggunaan diagram Use Case, diagram Sequence, dan diagram Activity.

### 3.3.1 Use Case Diagram

Diagram Use Case adalah narasi atau skenario tentang bagaimana suatu sistem aplikasi digunakan dan berinteraksi dengan pengguna. Identifikasi pengguna sistem sangat penting dalam pembuatan diagram Use-Case. Para pengguna ini disebut sebagai aktor, yang bisa berupa individu atau entitas sistem lainnya yang berinteraksi dengan aplikasi. Seseorang atau sesuatu yang berinteraksi dengan sistem untuk mencapai tujuan tertentu disebut sebagai aktor. Kasus penggunaan dan hubungan antara aktor dijelaskan oleh garis lurus, sementara garis putus-putus dengan informasi *include*, *extend* itu digunakan untuk menggambarkan bagaimana kasus penggunaan dapat dimasukkan atau diperpanjang dengan kasus penggunaan lainnya dalam suatu hubungan.



**Gambar 3.3.** Diagram Use-Case

Dua pengguna bertindak sebagai pengirim dan penerima dalam ilustrasi pada Gambar 3.3. Setelah proses enkripsi pesan menjadi ciphertext dan kunci OTP, kunci yang dihasilkan oleh *One Time Pad* (OTP) dienkripsi kembali menggunakan *Dependent RSA 1* (DRSA-1) menjadi cipherkey. Pengirim awalnya mengenkripsi pesan menggunakan algoritma OTP, dengan perhitungan OTP yang menghasilkan kunci untuk enkripsi pesan. Kemudian, penerima mendekripsi pesan dari ciphertext

ke *plaintext* dengan cara yang sama, yaitu dengan melakukan dekripsi kunci pesan dan menentukan kunci pesan.

Penggunaan *Narrative use-case* untuk enkripsi pesan dan kunci pesan dapat dilihat pada tabel 3.1.

**Tabel 3.1.** *Narrative Use Case* Enkripsi Pesan

Nama Use Case	Enkripsi Pesan
Aktor Utama	Pengguna
Keterlibatan Aktor	Sistem Enkripsi
Pihak-pihak terkait dan kepentingannya	Untuk melindungi pesan, pengirim (Pengguna) dapat mengenkripsinya dengan kunci pesan.
Kondisi Awal	Pengirim (pengguna) dapat memasukkan pesan secara lugas dan mengamankannya dengan algoritma
Minimal Guarantee	Pesan kesalahan ditampilkan oleh sistem jika enkripsi gagal.
Success Guarantee	Ketika enkripsi berhasil, sistem akan menampilkan pesan sukses.
Trigger	Pengguna memiliki opsi untuk memicu proses enkripsi pesan dengan menekan tombol.
Skenario utama keberhasilan	<p>Teks biasa yang akan terenkripsi oleh pengirim.</p> <p>Untuk melakukan enkripsi pada pesan, pengirim menempatkan kunci dan mengklik tombol untuk enkripsi.</p> <p>Sistem menjaga pesan agar tetap aman..</p> <p>Menggunakan algoritma <i>Mersenne Twister</i>, pengirim menghasilkan kunci, sistem akan melakukan generate kunci</p> <p>Pengirim mengenkripsi kunci pesan dengan menggunakan algoritma <i>Dependent RSA 1 (DRSA-1)</i></p> <p>Sistem akan melakukan enkripsi kunci menjadi <i>cipherkey</i>.</p>
Extensions	Kunci enkripsi kosong atau tidak lengkap.



Berikutnya adalah narrative *use-case* untuk dekripsi pesan dan kunci yang ditampilkan pada tabel 3.2

**Tabel 3.2.** Narrative *Use-Case* Dekripsi Pesan

Nama Use Case	Dekripsi Pesan
Aktor Utama	Pengguna
Keterlibatan Aktor	Sistem Dekripsi
Pihak-pihak terkait dan kepentingannya	Yang menerima pesan memiliki kemampuan untuk mengembalikan ciphertext dan cipherkey ke dalam bentuk <i>plaintext</i> melalui proses dekripsi.
Kondisi Awal	Penerima memiliki text <i>ciphertext</i> untuk di dekripsi
Minimal Guarantee	Pesan kesalahan ditampilkan oleh sistem jika enkripsi gagal.
Success Guarantee	Ketika enkripsi berhasil, sistem akan menampilkan pesan sukses.
Trigger	Pengguna memiliki opsi untuk memicu proses dekripsi pesan dengan menekan tombol
Skenario utama keberhasilan	<p>Kunci pribadi DRSA-1 adalah milik pengirim.</p> <p>Kunci pribadi DRSA-1 digunakan oleh pengirim untuk mendekripsi kunci pesan.</p> <p>Ciphertext terenkripsi didekripsi oleh pengirim.</p> <p>Pesan tersebut diterjemahkan oleh sistem.</p> <p>Dekripsi berhasil, seperti yang ditunjukkan oleh sistem.</p> <p><i>Plaintext</i> akan ditampilkan oleh sistem.</p>

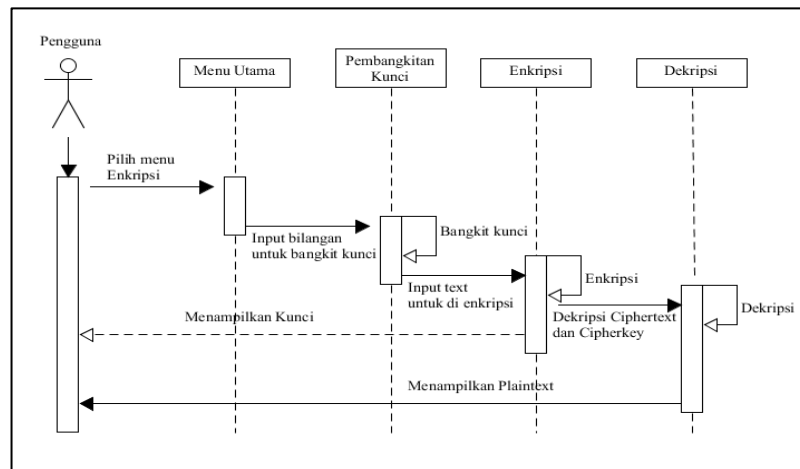
Tabel 3.3 menyajikan kasus penggunaan narrative *use-case* untuk prosedur pembuatan kunci acak.

**Tabel 3.3.** *Narrative Use-Case* Bangkitkan Kunci

Nama Use Case	Pembangkit Kunci Acak
Aktor Utama	Sistem
Keterlibatan Aktor	-
Pihak-pihak terkait dan kepentingannya	Karena akan sulit melakukannya secara manual, pengguna lebih suka membuat kunci secara acak.
Success Guarantee	Nomor acak yang dihasilkan ditampilkan oleh sistem.
Trigger	Tombol hasilkan kunci ( <i>Generate Key</i> ) ditekan oleh pengguna.
Skenario utama keberhasilan	<p>Sistem menerima permintaan untuk menghasilkan kunci acak.</p> <p>Sistem menghasilkan deretan bilangan acak yang dihasilkan oleh algoritma pembangkit kunci acak.</p> <p>Kunci acak disimpan dalam penyimpanan yang sesuai untuk digunakan dalam proses enkripsi.</p> <p>Sistem memberikan konfirmasi bahwa kunci acak telah berhasil dibangkitkan.</p>

### 3.3.2 *Sequence* Diagram

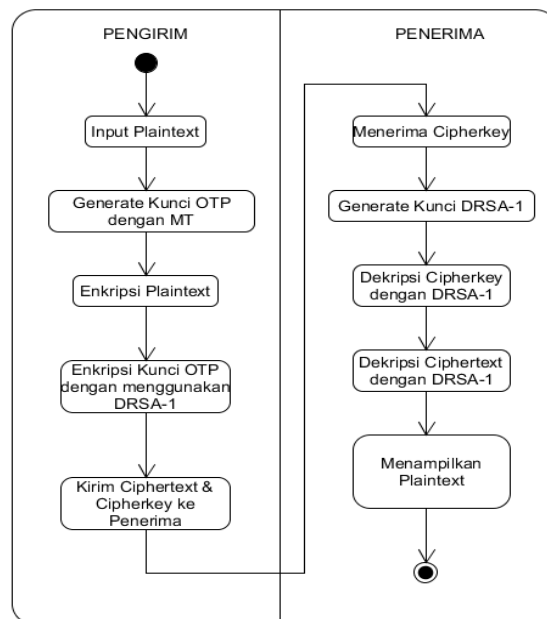
Tujuan pembuatan diagram urutan (*Sequence Diagram*) adalah untuk mengilustrasikan interaksi dan urutan pesan yang akan dikirimkan antar elemen yang terlibat. Diagram urutan tersebut menggambarkan sistem yang sedang dirancang, seperti yang ditampilkan pada gambar 3.4.



**Gambar 3.4.** *Diagram Sequence*

### 3.3.3 Activity Diagram

*Activity Diagram* juga membantu orang memahami prosedur dan menjelaskan bagaimana berbagai use case berinteraksi satu sama lain. Garis besar tindakan direncanakan yang ditampilkan pada gambar 3.5.



**Gambar 3.5.** *Activity Diagram*

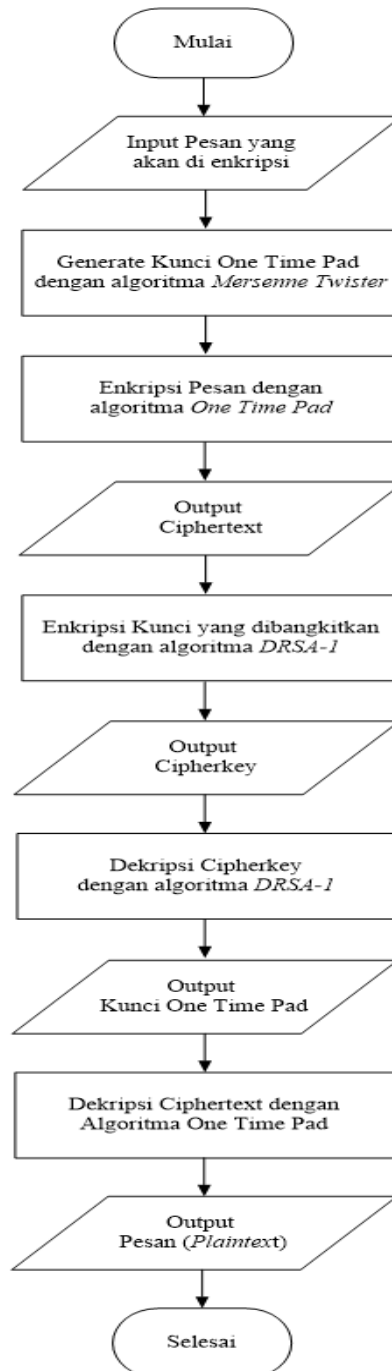
### 3.4 Flowchart

*Flowchart* atau bagan alur adalah yang menunjukkan keputusan dan langkah dalam suatu proses pembuatan program. Dalam bagan alur dimana bagan yang menggambarkan setiap langkah dengan garis atau panah yang menghubungkannya digunakan. Bagan alur memainkan peran penting dalam menentukan langkah

selanjutnya. Selain itu, diagram alur menggunakan diagram alur program, yang lebih jelas, ringkas, dan kecil kemungkinannya untuk disalahartikan. Berikut ini adalah Flowchart dari sistem yang sedang dibuat:

#### 3.4.1 Flowchart Sistem

Di bawah ini diagram alur sistem telah dibuat, dapat ditemukan dalam gambar 3.6.

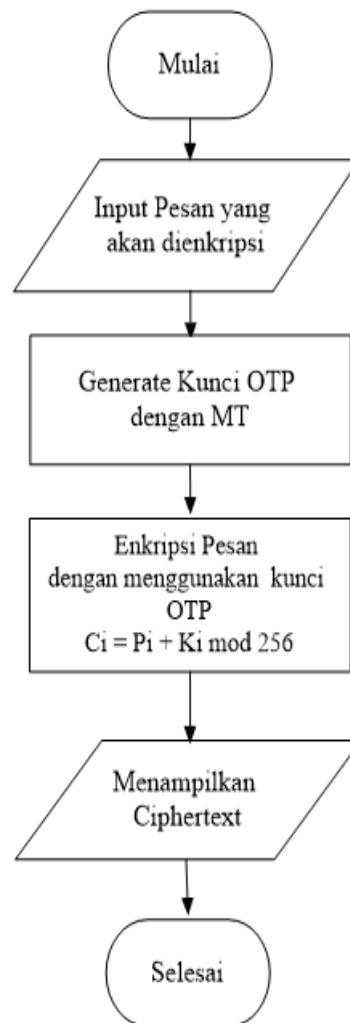


**Gambar 3.6.** Flowchart Sistem

Berdasarkan gambar 3.6, proses pengamanan pesan diawali dengan input pesan yang akan dienkripsi menggunakan kunci yang dihasilkan oleh *Mersenne Twister*. Pesan kemudian akan dienkripsi memakai *One Time Pad* kemudian kuncinya akan dienkripsi menggunakan algoritma *Dependent RSA 1* (DRSA-1) sehingga menghasilkan ciphertext. Setelah itu, cipherkey akan didekripsi sekali lagi menggunakan algoritma *Dependent RSA 1* (DRSA-1) untuk mendapatkan kunci OTP.

#### 3.4.2 Flowchart Enkripsi dengan Algoritma *One Time Pad*

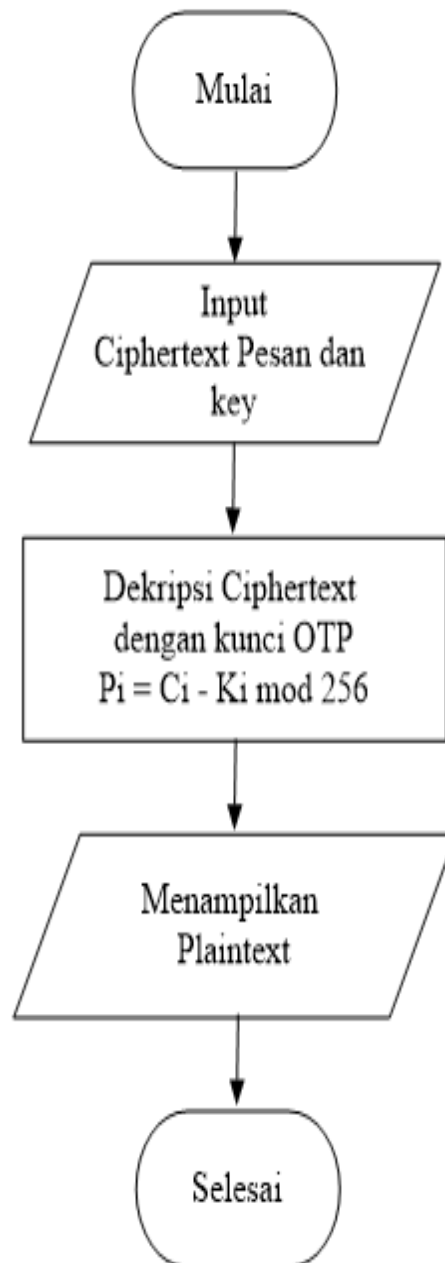
Gambar 3.7 menunjukkan Flowchart yang menjelaskan proses enkripsi dengan menggunakan *One Time Pad* (OTP).



**Gambar 3.7.** Flowchart Enkripsi dengan Algoritma *One Time Pad*

### 3.4.3 Flowchart Dekripsi dengan Algoritma *One Time Pad*

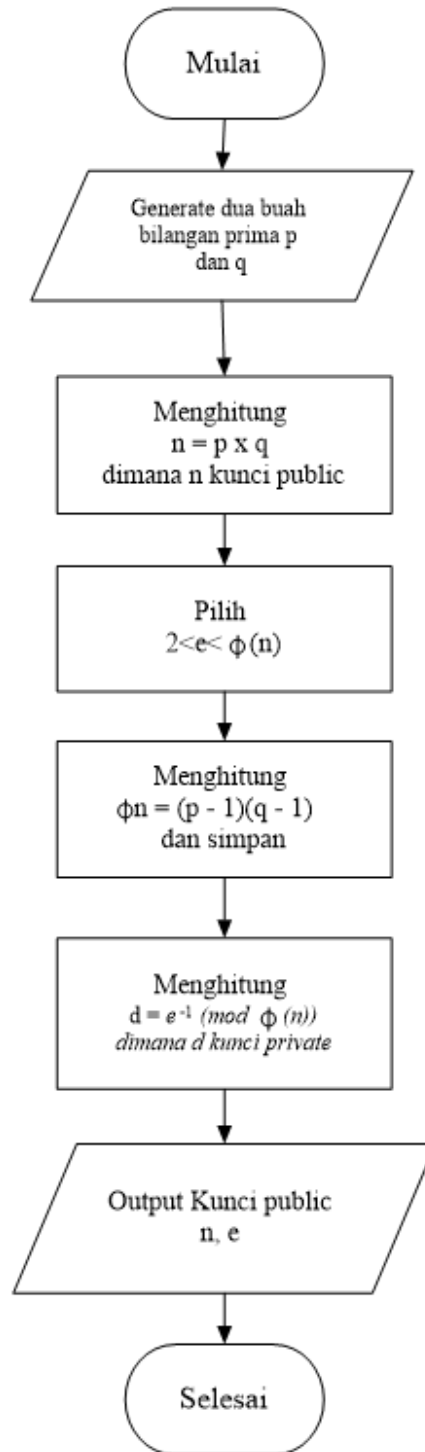
Gambar 3.8 menampilkan Flowchart yang menjelaskan proses dekripsi menggunakan *One Time Pad* (OTP).



**Gambar 3.8** Flowchart Dekripsi *One Time Pad*

#### 3.4.4 Flowchart Generate Kunci dengan Algoritma *Dependent RSA 1 (DRSA-1)*

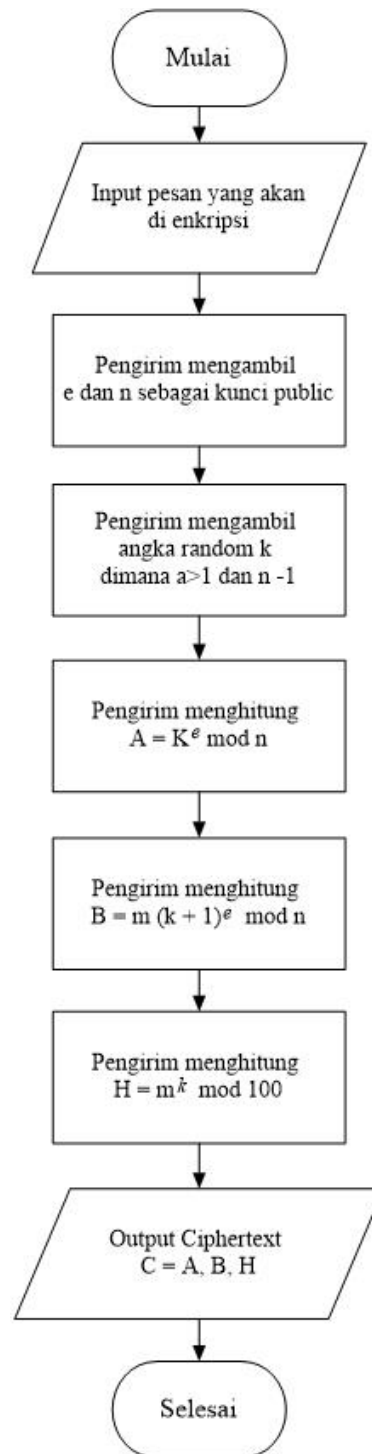
Gambar 3.9 menggambarkan diagram alur yang mengilustrasikan proses pembuatan kunci dengan menggunakan algoritma *Dependent RSA 1 (DRSA-1)*.



**Gambar 3.9.** Flowchart Generate Kunci *Dependent RSA 1 (DRSA-1)*

### 3.4.5 Flowchart Enkripsi dengan Algoritma *Dependent RSA 1 (DRSA-1)*

Gambar 3.10 menampilkan diagram alur yang menggambarkan proses enkripsi dengan menggunakan algoritma *Dependent RSA 1 (DRSA-1)*.

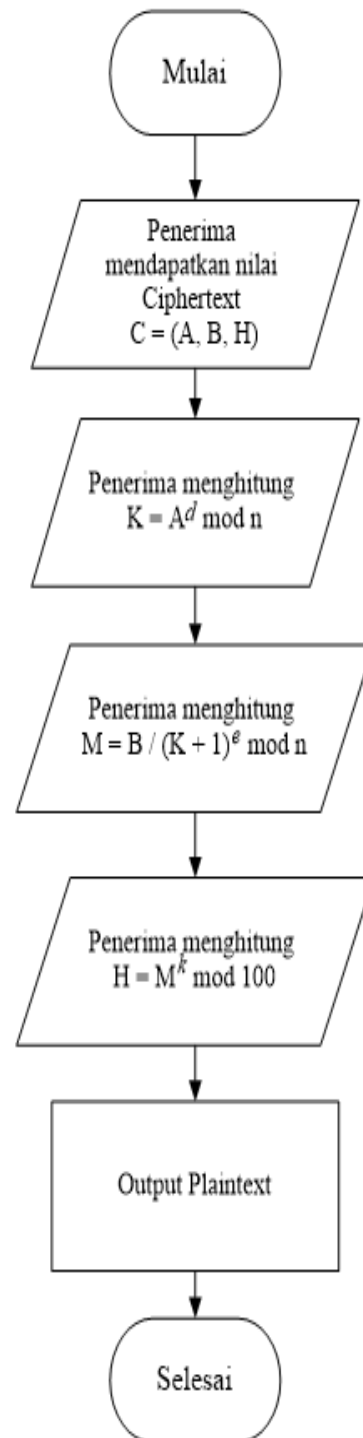


**Gambar 3.10.** Flowchart Enkripsi *Dependent RSA 1 (DRSA-1)*



### 3.4.6 Flowchart Dekripsi dengan Algoritma *Dependent RSA 1 (DRSA-1)*

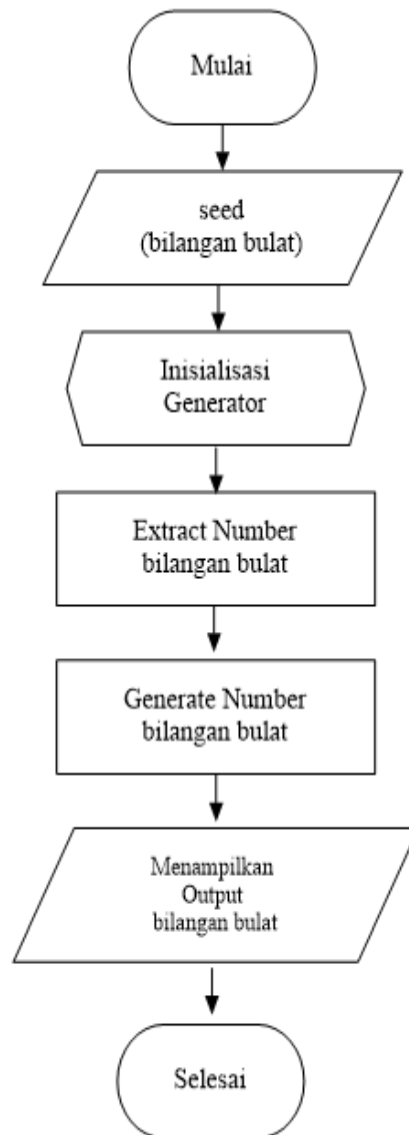
Berikut adalah *Flowchart* dekripsi dengan algoritma *Dependent RSA 1 (DRSA-1)* yang ditampilkan di gambar 3.11.



**Gambar 3.11.** *Flowchart Dekripsi Dependent RSA 1 (DRSA-1)*

### 3.4.7 Flowchart Generate Kunci dengan *Mersenne Twister*

Berikut adalah *Flowchart* algoritma *Mersenne Twister* yang ditampilkan pada gambar 3.12.



**Gambar 3.12.** *Flowchart Algoritma Mersenne Twister*

### 3.5 Perancangan Antarmuka (Interface)

Perancangan antarmuka adalah rencana yang mendasari keberadaan sistem yang akan dibangun. Titik koneksi harus dimasukkan dengan mempertimbangkan faktor kenyamanan dan kemudahan pengguna. Kerangka kerja yang dibangun terdiri dari struktur halaman utama dan presentasi awal, struktur enkripsi dan struktur penguraian.

### 3.5.1 Form Halaman Utama dan Tampilan Awal

Tampilan awal sistem berupa form halaman utama dan tampilan awal. Formulir enkripsi, dekripsi, dan halaman utama ditampilkan pada tampilan ini.

The wireframe shows a main page layout with the following elements and callouts:

- 1**: Button labeled "Halaman Utama" (Main Page).
- 2**: Button labeled "Halaman Pembangkit Kunci" (Key Generation Page).
- 3**: Button labeled "Halaman Enkripsi" (Encryption Page).
- 4**: Button labeled "Halaman Dekripsi" (Decryption Page).
- 5**: Text input field labeled "Judul Skripsi" (Thesis Title).
- 6**: Circular logo area labeled "Logo USU".
- 7**: Text input field labeled "Nama Lengkap NIM" (Full Name NIM).
- 8**: Text area containing "Program Studi Fakultas Universitas Tahun" (Study Program Faculty University Year).
- 9**: Button labeled "Keluar" (Exit).

**Gambar 3.13.** Rancangan Tampilan Halaman Utama dan Tampilan Awal

Keterangan :

1. *Button Halaman Utama*, berfungsi sebagai tombol untuk pergi ke halaman utama atau halaman awal.
2. *Button Halaman Pembangkit Kunci*, berfungsi sebagai tombol untuk pergi ke halaman pembangkit bilangan acak.
3. *Button Halaman enkripsi*, berfungsi sebagai tombol untuk pergi ke halaman enkripsi.
4. *Button Halaman dekripsi*, berfungsi sebagai tombol untuk pergi ke halaman dekripsi.
5. *Label*, berisi informasi judul skripsi.
6. *Label*, yang berisi logo USU.

7. *Label*, berisi informasi nama lengkap dan NIM.
8. *Label*, yang berisi informasi Program Studi, Fakultas, Universitas, dan Tahun Ajaran.
9. *Button keluar*, berfungsi sebagai penghentian sistem.

### 3.5.2 Form Halaman Pembangkit Kunci

Form tampilan *pembangkit kunci* adalah tampilan halaman dimana pengguna dapat melakukan proses pembangkitan kunci. Berikut adalah tampilan sementara dari form pembangkit kunci yang ditampilkan pada gambar 3.14.

The diagram illustrates the layout of the 'HalamanPembangkit Kunci' (Key Generation Page). It features a title bar at the top labeled '1'. Below the title bar, there are five input fields, each preceded by a label: 'p' (2), 'q' (3), 'n' (4), 'e' (5), and 'd' (6). At the bottom of the form, there are five buttons: 'Generate' (7), 'Reset' (8), 'Salin Kunci Public' (9), 'Salin Kunci Private' (10), and 'Kembali' (11).

**Gambar 3.14.** Rancangan Tampilan Halaman Pembangkit Kunci

Keterangan :

1. *Label*, yang berisi informasi halaman pembangkit kunci
2. *Label dan JTextField*, berfungsi untuk menginputkan nilai p sebagai bilangan prima.
3. *Label dan JTextField*, berfungsi untuk menginputkan nilai q sebagai bilangan prima.
4. *Label dan JTextField*, berfungsi untuk menginputkan nilai n.

5. *Label dan JTextField*, berfungsi untuk menginputkan nilai e.
6. *Label dan JTextField*, berfungsi untuk menginputkan nilai d.
7. *Button Generate*, berfungsi untuk mengambil bilangan acak yang telah di generate kemudian ditampilkan.
8. *Button Reset* berfungsi untuk menghilangkan kunci yang telah dibangkitkan sebelumnya.
9. *Button Salin Kunci Public* berfungsi untuk menyalin kunci public.
10. *Button Salin Kunci Privat* berfungsi menyalit kunci privat.
11. *Button kembali*, berfungsi untuk kembali kehalaman utama.

### 3.5.3 Form Halaman Enkripsi

Form tampilan enkripsi adalah tampilan halaman di mana pengguna dan pengirim dapat menyelesaikan metode yang terkait dengan enkripsi pesan (*plaintext* ). Bentuk enkripsi pada gambar 3.15 untuk sementara ditampilkan pada gambar berikut.

**Gambar 3.15.** Rancangan Tampilan Halaman Enkripsi

Keterangan :

1. *Label*, yang berisi informasi halaman enkripsi.
2. *Label dan JScrollPane*, yang berfungsi untuk menginput dan menampilkan pesan yang akan dienkripsi.
3. *Label dan JScrollPane*, yang berfungsi untuk menampilkan pesan yang telah dienkripsi dengan algoritma OTP.
4. *Button Enkripsi*, berfungsi untuk mengenkripsi pesan.
5. *Button Generate Kunci OTP*, berfungsi untuk mengambil bilangan acak yang telah di generate dengan algoritma MT kemudian ditampilkan.
6. *Label dan JScrollPane*, yang berfungsi untuk menginput dan menampilkan kunci OTP yang dibangkitkan dengan MT.
7. *Label dan JScrollPane*, yang berfungsi untuk menampilkan kunci yang telah dienkripsi dengan algoritma DRSA-1.
8. *Label dan JTextField*, yang berfungsi untuk menampilkan kunci public DRSA-1 yaitu  $e$  dan  $n$ .
9. *Button Generate Kunci OTP*, berfungsi untuk mengambil bilangan acak yang telah di generate dengan algoritma MT kemudian ditampilkan.
10. *Button Enkripsi Kunci*, berfungsi untuk mengenkripsi kunci.
11. *Button Salin Ciphertext* berfungsi sebagai untuk menyalin *ciphertext* yang telah dienkripsi.
12. *Button Salin Cipherkey*, berfungsi sebagai untuk menyalin *cipherkey* yang dienkripsi.
13. *Button kembali*, berfungsi untuk kembali kehalaman utama.

#### 3.5.4 Form Halaman Dekripsi

Pengguna dan penerima memiliki kemampuan untuk melakukan proses dekripsi *ciphertext* dan *cipherkey* melalui formulir halaman dekripsi. Bentuk dekripsi, seperti yang ditunjukkan dalam Gambar 3.16, sementara ini dapat dilihat di bawah ini.

The wireframe shows a page titled 'Halaman Dekripsi'. It contains several input fields and buttons, numbered 1 through 9 for reference:

- 1: Title bar 'Halaman Dekripsi'.
- 2: Label 'Kunci Private DRSA-1' above three input fields for 'e', 'd', and 'n'.
- 3: Input field for 'Cipherkey'.
- 4: Input field for 'Kunci OTP'.
- 5: Button labeled 'Dekripsi Kunci'.
- 6: Input field for 'Ciphertext'.
- 7: Input field for 'Plaintext'.
- 8: Button labeled 'Dekripsi'.
- 9: Button labeled 'Kembali' (Return).

**Gambar 3.16.** Rancangan Tampilan Halaman Dekripsi

Keterangan :

1. *Label*, yang berisi informasi halaman dekripsi.
2. *Label dan JTextField*, yang berfungsi untuk menampilkan kunci privat DRSA-1 yaitu e, d, dan n.
3. *Label dan JScrollPane*, yang berfungsi untuk menampilkan cipherkey yang telah dienkripsi dengan algoritma DRSA-1.
4. *Label dan JScrollPane*, yang berfungsi untuk menampilkan kunci OTP yang telah di dekripsi dengan DRSA-1.
5. *Button dekripsi kunci*, berfungsi untuk melakukan proses dekripsi kunci dengan DRSA-1.
6. *Label dan JScrollPane*, yang berfungsi untuk menampilkan pesan yang akan dienkripsi dan setelah di dekripsi dengan kunci OTP.
7. *Label dan JScrollPane*, yang berfungsi untuk menampilkan pesan (*plaintext*) yang akan didekripsi dengan OTP.
8. *Button dekripsi*, berfungsi untuk melakukan proses dekripsi pesan dengan OTP.
9. *Button kembali*, berfungsi untuk kembali kehalaman utama.

## BAB 4

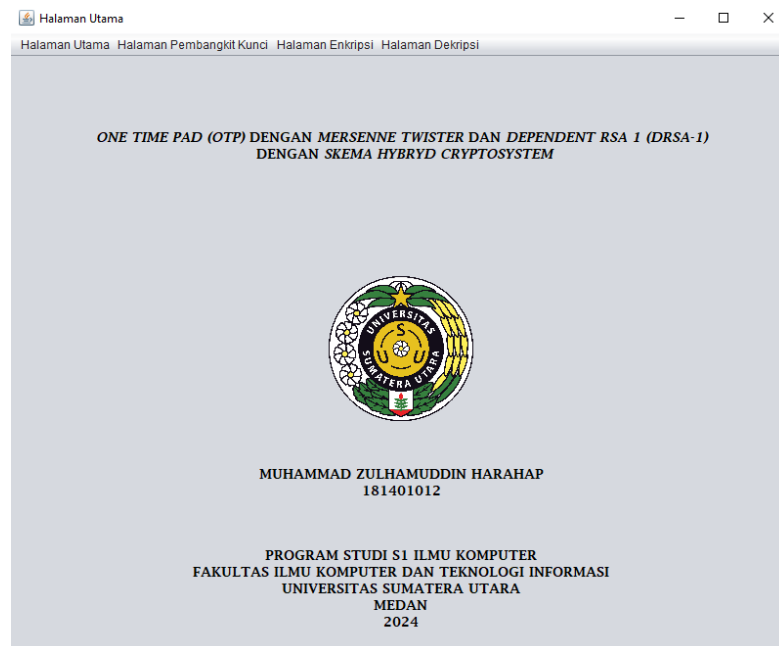
### IMPLEMENTASI DAN PENGUJIAN SISTEM

#### 4.1 Implementasi Sistem

Implementasi sistem adalah tindakan berikutnya dalam rencana kerangka kerja. Proses eksekusi kerangka kerja yang telah disusun akan segera dilaksanakan. Oleh karena itu, algoritma *One Time Pad* dengan *Mersenne Twister* dan Algoritma *Dependent RSA 1* (DRSA-1) dalam skema Hybrid Cryptosystem, Java, dan Apache Netbeans IDE 19 akan digunakan dalam pengembangan sistem dibangun, sesuai dengan tahapan analisis dan desain yang telah diselesaikan. Sistem ini melibatkan empat halaman inti: halaman awal, halaman pembuat kunci, halaman enkripsi, dan tampilan halaman dekripsi.

##### 4.1.1 Halaman Utama

Tampilan utama merupakan tampilan pertama terlihat ketika sistem dijalankan. Di sini, judul sistem, logo universitas, nama dan ID penulis, program studi, fakultas, dan universitas penulis semuanya disajikan. Tampilan halaman utama ini dapat ditemukan dalam Gambar 4.1.



**Gambar 4.1.** Tampilan Halaman Utama



#### 4.1.2 Halaman Pembangkit Bilangan Kunci

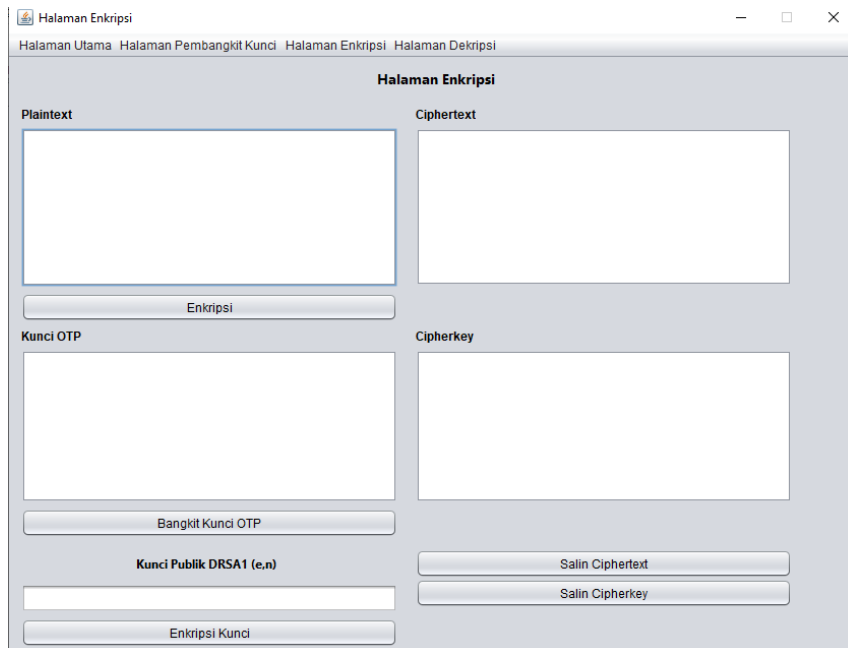
Kunci untuk algoritma *Dependent RSA 1* (DRSA-1) akan dibuat di halaman ini. Pengguna akan mengklik Halaman Penghasil Kunci JmenuBar untuk mengakses halaman pembuat kunci DRSA-1, yang ditampilkan pada gambar 4.2.

**Gambar 4.2.** Tampilan Halaman Pembangkit Kunci

Tampilan ini akan menampilkan beberapa kolom pembangkit kunci yang terdiri dari *textfield* nilai  $p$ ,  $q$ ,  $e$ ,  $n$ , dan  $d$  serta *button* bangkit, reset, salin kunci public dan salin kunci privat. Untuk menghitung nilai prima  $p$ ,  $q$  dan  $e$  serta menghitung nilai  $n$  dan  $d$ . User cukup menekan *button* bangkit maka nilai bilangan  $p$  dan  $q$  serta bilangan  $e$  akan dibangkitkan kemudian perhitungan nilai  $n$  dan  $d$  juga akan diselesaikan.

#### 4.1.3 Halaman Enkripsi

Tampilan ini user hendak melaksanakan proses mengubah pesan asli menjadi pesan terenkripsi, yang ditampilkan pada gambar 4.3.

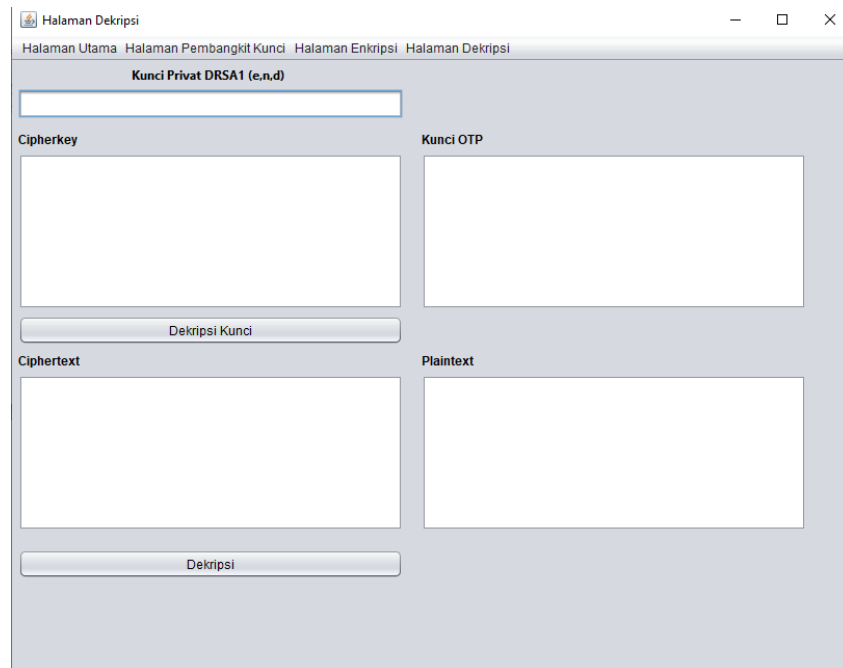


**Gambar 4.3.** Tampilan Halaman Enkripsi

Untuk melakukan proses enkripsi pesan dimana pertama kali *user* harus mengisi *plaintext* (pesan) yang akan di enkripsi, kemudian untuk melakukan bangkit kunci *OTP* dengan menggunakan algoritma *Mersenne Twister*, *user* mengklik *button* bangkit kunci *OTP* akan keluar hasil kunci *OTP* setelah itu untuk mengenkripsi pesan, *user* mengklik *button* “enkripsi” dengan algoritma *One Time Pad* yang akan menghasilkan *ciphertext*. Untuk mengenkripsi kunci, *user* terlebih dahulu membuka halaman pembangkit kunci untuk mendapatkan kunci public *DRSA-1* nilai *e* dan *n* setelah itu *user* mengklik *button* “enkripsi kunci” untuk menghasilkan *cipherkey*. Untuk melakukan salin *ciphertext* dan *cipherkey* dengan mengklik *button* salin *ciphertext* dan salin *cipherkey*.

#### 4.1.4 Halaman Dekripsi

Pengguna akan melakukan proses mendekripsi pesan rahasia dan mengubahnya menjadi pesan asli di halaman ini. Untuk masuk ke halaman dekripsi, pengguna akan mengetuk *JmenuBar* dari halaman penguraian. Gambar 4.4 menggambarkan halaman yang berisi deskripsi.



**Gambar 4.4.** Tampilan Halaman Dekripsi

Halaman ini untuk melakukan proses dekripsi dimana *user* harus kehalaman pembangkit kunci untuk mendapatkan kunci privat *DRSA-1* yaitu  $e$ ,  $n$ , dan  $d$ . Setelah itu *user* akan menginput *cipherkey* yang diperoleh dari halaman enkripsi sebelumnya lalu *user* akan melakukan dekripsi kunci untuk mendapatkan kunci *OTP*. Kemudian untuk mendekripsi *ciphertext*, *user* harus menginput *ciphertext* yang didapat dari halaman sebelumnya lalu *user* akan melakukan dekripsi *ciphertext* dengan kunci *OTP* yang di dekripsi untuk menghasilkan pesan yang telah di dekripsi kembali ke pesan semula.

## 4.2 Pengujian Sistem

Pengujian sistem menggunakan algoritma *One Time Pad* (OTP) dengan *Mersenne Twister* dan algoritma *Dependent RSA 1* (DRSA-1) untuk memastikan sistem yang dibangun berfungsi dengan baik dan dapat melakukan semua hal tersebut.

Pada penelitian ini dibuat pengujian pengamanan pesan teks untuk menguji sistem. Pengguna sistem ingin melindungi pesan tersebut, maka proses enkripsi dimulai dengan enkripsi pesan dan dilanjutkan dengan enkripsi kunci pesan. Penerima menerima *ciphertext* dan *cipherkey* dari pengirim pesan pada pengujian

selanjutnya, dan proses selanjutnya adalah dekripsi untuk mengambil pesan dari pengirim.

#### 4.2.1 Pengujian Pembangkit Kunci

Untuk mendapatkan nilai  $p$  dan  $q$  secara acak pengguna harus menekan tombol bangkit selama proses ini. Setelah itu, pastikan nilai kunci publik  $e$ ,  $n$  dan kunci rahasia  $e$ ,  $n$ , dan  $d$ . Gambar 4.5 menunjukkan hasil dari pembangkitan kunci untuk algoritma *Dependent RSA 1* (DRSA-1):

The screenshot shows a window titled 'Halaman Pembangkit Kunci'. Inside, there are five input fields with the following values:  $p = 53$ ,  $q = 41$ ,  $n = 2173$ ,  $e = 69$ , and  $d = 1869$ . Below these fields are four buttons: 'Bangkit', 'Salin Kunci Publik', 'Reset', and 'Salin Kunci Privat'.

**Gambar 4.5.** Hasil Pembangkit Kunci *Dependent RSA 1* (DRSA-1)

Pembuktian dari pembangkitan bilangan prima acak  $p$  dan  $q$  dapat dilihat dari perhitungan manual untuk menguji kebenarannya.

- Bilangan prima  $p = 53$   
 Perulangan pertama,  $a = 23$   
 $a^{(p-1)/2} \bmod p = 23^{(53-1)/2} \bmod 53$   
 $= 23^{26} \bmod 53$   
 $= 52$   
 Perulangan kedua,  $a = 13$   
 $a^{(p-1)/2} \bmod p = 13^{(53-1)/2} \bmod 53$   
 $= 13^{26} \bmod 53$

$$= 1$$

Terbukti bahwa angka 53 adalah bilangan prima

- Bilangan prima  $p = 41$

Perulangan pertama,  $a = 23$

$$\begin{aligned} a^{(p-1)/2} \bmod p &= 23^{(41-1)/2} \bmod 41 \\ &= 23^{20} \bmod 41 \\ &= 1 \end{aligned}$$

Perulangan kedua,  $a = 13$

$$\begin{aligned} a^{(p-1)/2} \bmod p &= 13^{(41-1)/2} \bmod 41 \\ &= 13^{20} \bmod 41 \\ &= 40 \end{aligned}$$

Terbukti bahwa angka **53** dan **41** adalah bilangan prima

- Menghitung nilai kunci public ( $n$ )

$$n = p \times q$$

$$n = 53 \times 41$$

$$n = 2173$$

- Menghitung nilai kunci public ( $e$ )

$$e = 69$$

$$\Phi(n) = (2080)$$

$$\text{GCD}(2080, 69) = 1, \text{ maka hasil benar.}$$

- Menghitung nilai kunci privat ( $d$ )

$$d = e^{-1} \bmod \text{Lcm}(p-1, q-1)$$

$$d = 69^{-1} \bmod \text{Lcm}(53-1, 41-1)$$

$$d = 69^{-1} \bmod \text{Lcm}(52, 40)$$

$$\text{Lcm}(52, 40) = \frac{52 \times 40}{\text{GCD}(52, 40)}$$

$$\text{GCD}(52, 40) = 4$$

$$\text{Lcm}(16, 46) = \frac{52 \times 40}{\text{GCD}(52, 40)}$$

$$= \frac{2080}{4}$$

$$= 520$$

$$d = e^{-1} \bmod Lcm(p-1, q-1)$$

$$e d = 1 \bmod Lcm(p-1, q-1)$$

$$d = 69^{-1} \bmod Lcm(52, 40)$$

$$d = 69^{-1} \bmod 520$$

$$69 d = 1 \bmod 520$$

$$d = 69 \bmod 520$$

$$d = 69$$

Nilai d dapat dihitung menggunakan teknik invers modulo seperti yang ditunjukkan dalam tabel di bawah ini.

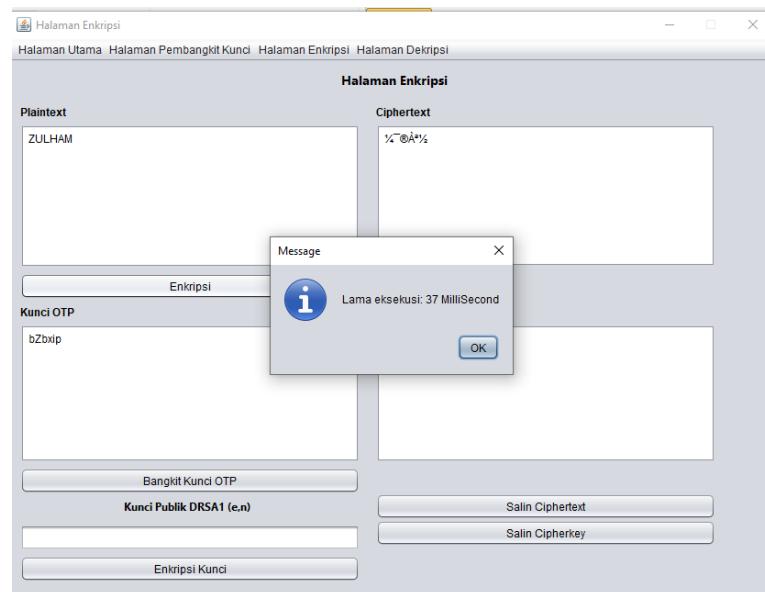
**Tabel 4.1.** Invers Modulo menghitung nilai d

D	$n^{-1} \bmod Lcm(p-1, q-1)$ $69^{-1} \bmod 520$
1	69
2	138
3	207
4	276
.	.
.	.
.	.
1867	383
1868	452
1869	1

Dikarenakan hasil dari  $n^{-1} \bmod Lcm(p-1, q-1) = 1$  pada nilai  $d = 1869$ , maka nilai yang diperoleh sistem adalah benar.

#### 4.2.2 Pengujian Enkripsi algoritma *One Time Pad*

Pada tahap enkripsi, pengguna memasukkan teks biasa yang akan diacak. Di halaman enkripsi, *plaintext* kemudian dienkripsi memakai *One Time Pad* untuk membentuk ciphertext. Penyajian siklus enkripsi yang ditampilkan pada gambar 4.6.



**Gambar 4.6.** Hasil Enkripsi pesan dengan *One Time Pad*

Pada pengujian diatas teks yang digunakan oleh penulis adalah “ZULHAM” dengan jumlah karakter 6. Pembuktian enkripsi di atas dapat diuji dengan melakukan perhitungan manual.

- $P = Z = 90$   
 $K = b = 98$   
 $C = P + K.OTP = 188 (\frac{1}{4})$
- $P = U = 85$   
 $K = Z = 90$   
 $C = P + K.OTP = 175 (\sim)$
- $P = L = 76$   
 $K = b = 98$   
 $C = P + K.OTP = 174 (\textcircled{R})$
- $P = H = 72$   
 $K = x = 120$   
 $C = P + K.OTP = 192 (\text{\AA})$
- $P = A = 65$   
 $K = i = 105$   
 $C = P + K.OTP = 170 (\underline{a})$
- $P = M = 77$

$$K = p = 112$$

$$C = P + K.OTP = 189 (\frac{1}{2})$$

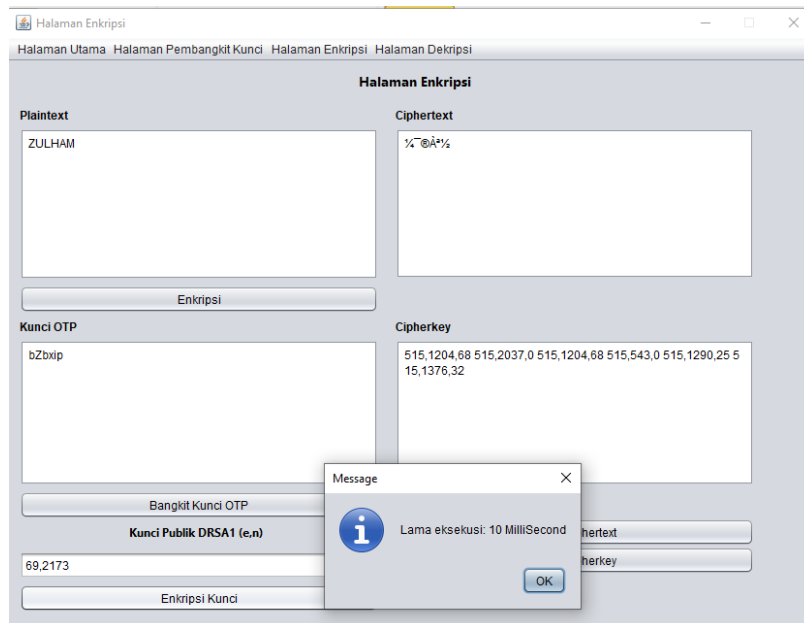
**Tabel 4.2.** Enkripsi pesan dengan Algoritma *One Time Pad*

Karakter	Kode ASCII	Ciphertext
Z	90	188 ( $\frac{1}{4}$ )
U	85	175 ( $\bar{\text{~}}$ )
L	76	174 ( $\textcircled{\text{R}}$ )
H	72	192 ( $\text{\AA}$ )
A	65	170 ( $\underline{\text{a}}$ )
M	77	189 ( $\frac{1}{2}$ )

Dengan membandingkan hasil perhitungan manual dengan output dari sistem, dapat disimpulkan bahwa sistem enkripsi memberikan nilai yang tepat.

#### 4.2.3 Pengujian Enkripsi Kunci OTP dengan *Dependent RSA 1 (DRSA-1)*

Setelah itu, langkah berikutnya adalah mengenkripsi kunci pesan menggunakan kunci publik *Dependent RSA 1 (DRSA-1)*, yaitu  $e$  dan  $n$  dengan hasil enkripsi pesan yang ditampilkan pada gambar 4.7.



**Gambar 4.7.** Hasil Enkripsi Kunci OTP dengan *Dependent RSA 1 (DRSA-1)*



Pada pengujian kunci OTP, kunci yang digunakan adalah “ bZbxip ” dengan dibangkitkan oleh algoritma *Mersenne Twister*. Pembuktian dilakukan dengan melakukan perhitungan manual.

- Huruf b

$$m \text{ (nilai bilangan ASCII)} = 98$$

$$A = K^e \bmod n$$

$$A = 25^{69} \bmod 2173$$

$$A = 515$$

$$B = m (k + 1)^e \bmod n$$

$$B = 98 (25 + 1)^{69} \bmod 2173$$

$$\mathbf{B = 1204}$$

- Huruf Z

$$M \text{ (nilai bilangan ASCII)} = 90$$

$$A = K^e \bmod n$$

$$A = 25^{69} \bmod 2173$$

$$A = 515$$

$$B = m (k + 1)^e \bmod n$$

$$B = 90 (25 + 1)^{69} \bmod 2173$$

$$\mathbf{B = 2037}$$

- Huruf b

$$m \text{ (nilai bilangan ASCII)} = 98$$

$$A = K^e \bmod n$$

$$A = 25^{69} \bmod 2173$$

$$A = 515$$

$$B = m (k + 1)^e \bmod n$$

$$B = 98 (25 + 1)^{69} \bmod 2173$$

$$\mathbf{B = 1204}$$

- Huruf x

$$A = K^e \bmod n$$

$$A = 25^{69} \bmod 2173$$

$$A = 515$$

$$B = m(k + 1)^e \bmod n$$

$$B = 120(25 + 1)^{69} \bmod 2173$$

$$\mathbf{B = 543}$$

- Huruf i

$$m(\text{nilai bilangan ASCII}) = 105$$

$$A = K^e \bmod n$$

$$A = 25^{69} \bmod 2173$$

$$A = 515$$

$$B = m(k + 1)^e \bmod n$$

$$B = 105(25 + 1)^{69} \bmod 2173$$

$$\mathbf{B = 1290}$$

- Huruf p

$$m(\text{nilai bilangan ASCII}) = 112$$

$$A = K^e \bmod n$$

$$A = 25^{69} \bmod 2173$$

$$A = 515$$

$$B = m(k + 1)^e \bmod n$$

$$B = 112(25 + 1)^{69} \bmod 2173$$

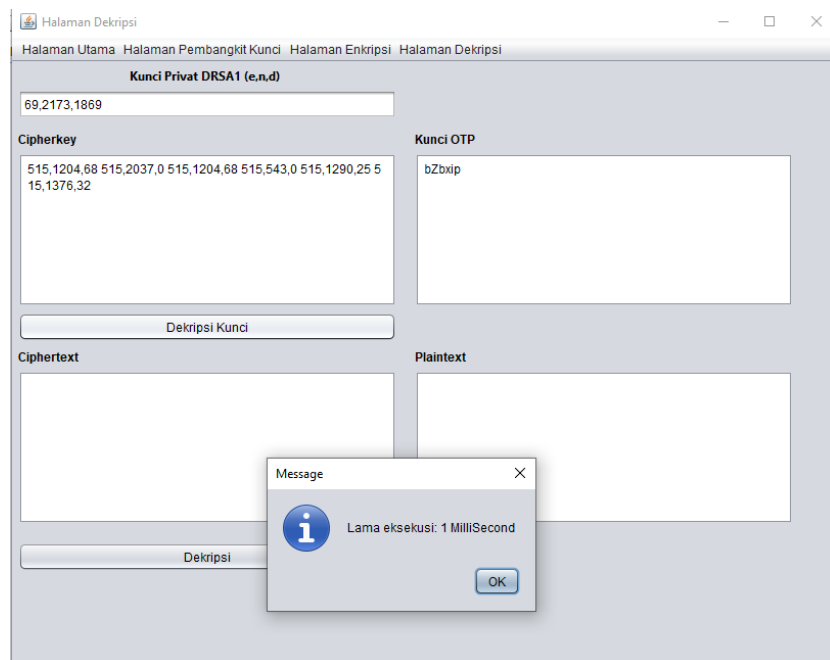
$$\mathbf{B = 1376}$$

**Tabel 4.3.** Enkripsi Kunci dengan Algoritma *Dependent RSA 1 (DRSA-1)*

Karakter	Kode ASCII	Cipherkey
b	98	1204
Z	90	2037
b	98	1204
x	120	543
i	105	1290
p	112	1376

#### 4.2.4 Pengujian Dekripsi Kunci OTP dengan *Dependent RSA 1 (DRSA-1)*

Pada proses dekripsi kunci, *user* menginputkan nilai kunci privat ( $d$ ) serta ( $e$ ), ( $n$ ). Setelah itu *user* akan meyalin *cipherkey* dari form enkripsi untuk di inputkan kedalam dekripsi, kemudian *cipherkey* di dekripsi dengan algoritma *Dependent RSA 1 (DRSA-1)* untuk menghasilkan kunci *OTP*. Gambar 4.8 menunjukkan tampilan proses dekripsi Cipherkey.



**Gambar 4.8.** Hasil Dekripsi *Cipherkey* dengan *Dependent RSA 1 (DRSA-1)*

*Cipherkey* yang dihasilkan 515,1204,68 515,2037,0 515,1204,68 515,543,0 515,1290,25 515,1376,32. Pembuktian hasil dekripsi di atas dapat diuji dengan melakukan perhitungan manual.

- $A = 515$   
 $K = A^d \bmod n$   
 $K = 515^{1869} \bmod 2173$   
 **$K = 25$**
- $M1 = B / (k + 1)^e \bmod n$   
 $M1 = 1204 / (25 + 1)^{69} \bmod 2173$   
 $M1 = (1204) (1588)^{69} \bmod 2173$   
 **$M1 = 98 = b$**

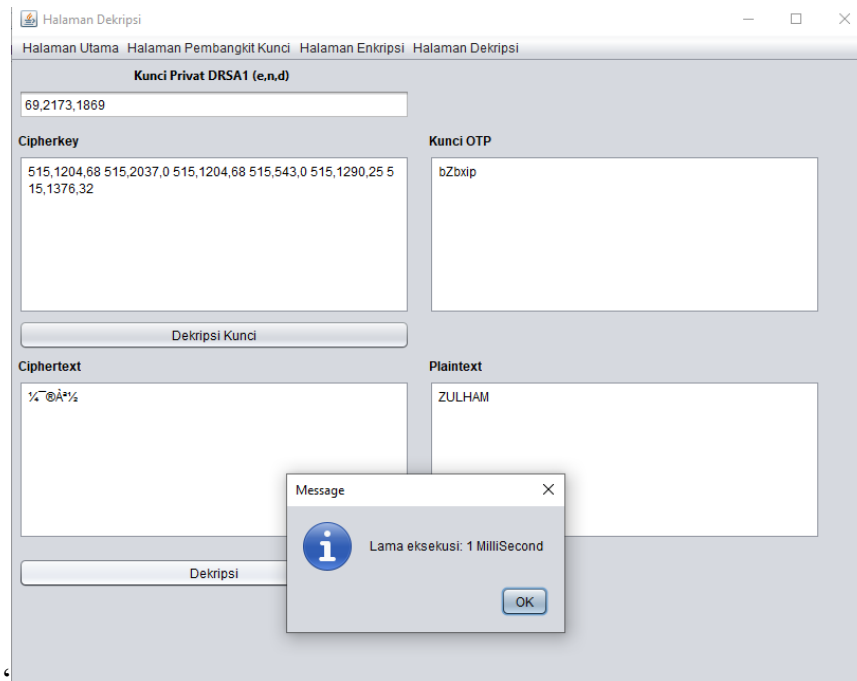
- $M2 = B / (k + 1)^e \bmod n$   
 $M2 = 2037 / (25 + 1)^{69} \bmod 2173$   
 $M2 = (2037) (1588)^{69} \bmod 2173$   
**M2 = 90 = Z**
- $M3 = B / (k + 1)^e \bmod n$   
 $M3 = 1204 / (25 + 1)^{69} \bmod 2173$   
 $M3 = (1204) (1588)^{69} \bmod 2173$   
**M3 = 98 = b**
- $M4 = B / (k + 1)^e \bmod n$   
 $M4 = 543 / (25 + 1)^{69} \bmod 2173$   
 $M4 = (543) (1588)^{69} \bmod 2173$   
**M4 = 120 = x**
- $M5 = B / (k + 1)^e \bmod n$   
 $M5 = 1290 / (25 + 1)^{69} \bmod 2173$   
 $M5 = (1290) (1588)^{69} \bmod 2173$   
**M5 = 105 = i**
- $M6 = B / (k + 1)^e \bmod n$   
 $M6 = 1376 / (25 + 1)^{69} \bmod 2173$   
 $M6 = (1376) (1588)^{69} \bmod 2173$   
**M6 = 112 = p**

**Tabel 4.4.** Dekripsi *cipherkey* dengan Algoritma *Dependent RSA 1 (DRSA-1)*

<i>Cipherkey</i>	<i>Kode ASCII</i>	Karakter
1204	98	b
2037	90	Z
1204	98	b
543	120	x
1290	105	i
1376	112	p

#### 4.2.5 Pengujian Dekripsi pesan dengan *One Time Pad*

Setelah memperoleh hasil dekripsi kunci dengan melakukan dekripsi kunci, langkah berikutnya dalam sistem adalah melakukan dekripsi pesan menggunakan kunci yang telah didekripsi. Gambar 4.9 menampilkan tampilan proses dekripsi pesan.



**Gambar 4.9.** Hasil Dekripsi *Ciphertext* dengan *One Time Pad*

Pembuktian hasil dekripsi pesan di atas dapat diuji dengan melakukan perhitungan manual:

- $C = \frac{1}{4} = 108$   
 $C - K.OTP = 188 - 98 = 90$   
 **$P = \text{char}(90) = Z$**
- $C = \sim = 175$   
 $C - K.OTP = 175 - 90 = 85$   
 **$P = \text{char}(85) = U$**
- $C = @ = 174$   
 $C - K.OTP = 174 - 98 = 76$

**P = char (76) = L**

- $C = \text{À} = 192$   
 $C-K.OTP = 192-120 = 72$   
**P = char (72) = H**

- $C = \text{a} = 170$   
 $C-K.OTP = 170-105 = 65$   
**P = char (65) = A**

- $C = \text{½} = 189$   
 $C-K.OTP = 189-112 = 77$   
**P = char (77) = M**

**Tabel 4.5.** Dekripsi dengan Algoritma *One Time Pad*

<i>Ciphertext</i>	<i>Kode ASCII</i>	Karakter
¼	188	Z
-	175	U
®	174	L
À	192	H
a	170	A
½	189	M

### 4.3 Waktu Proses (Real Running Time)

Satu dari tujuan penelitian ini adalah mengevaluasi waktu yang diperlukan untuk setiap langkah dalam sistem, termasuk proses enkripsi dan dekripsi, dengan variasi panjang karakter *plaintext* . Satuan waktu yang digunakan untuk mengukur kecepatan proses adalah milidetik (*millisecond*).

#### 4.3.1 Pengujian proses enkripsi pesan dengan *One Time Pad (OTP)*

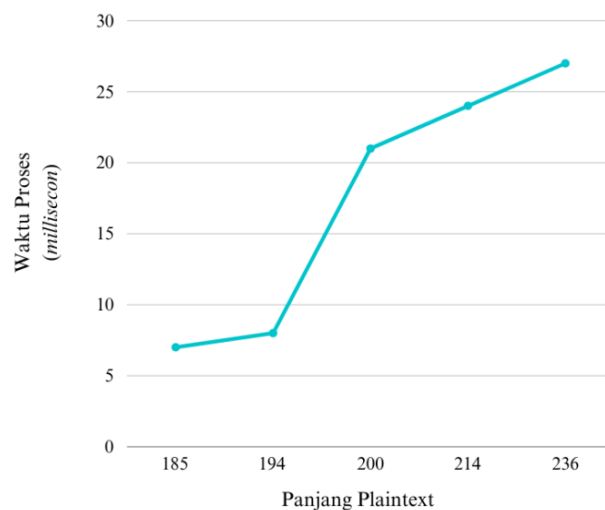
Pengujian ini bertujuan untuk menegaskan dampak panjang *plaintext* dan panjang kunci pada durasi proses enkripsi.

**Tabel 4.6.** Waktu Proses Enkripsi dengan Algoritma *One Time Pad*

No	Panjang <i>plaintext</i> (Karakter)	Waktu Proses (millisecond)
1	185	7 ms
2	194	8 ms
3	200	21 ms
4	214	24 ms
5	236	27 ms

Berikut adalah grafik perbandingan antara panjang *plaintext* dan waktu yang diperlukan untuk proses pemecahan yang tersaji dalam gambar 4.10.

Grafik hubungan panjang plaintext terhadap waktu proses enkripsi *One Time Pad*



**Gambar 4.10.** Diagram yang menggambarkan hubungan panjang *plaintext* dan waktu proses enkripsi menggunakan *One Time Pad*.

#### 4.3.2 Pengujian proses enkripsi kunci dengan Algoritma *Dependent RSA 1* (DRSA-1)

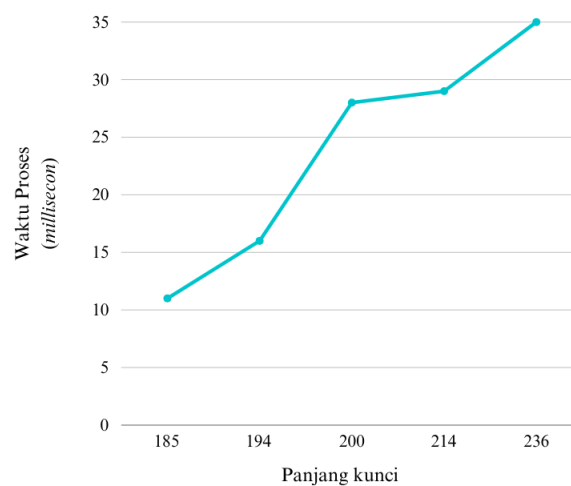
Pada proses enkripsi kunci, dilakukan 5 kali percobaan untuk melihat efisiensi algoritma *Dependent RSA 1* (DRSA-1) di mana waktu proses dihitung berdasarkan panjang *plaintext*. Kunci public yang digunakan dua digit.

**Tabel 4.7.** Waktu Proses Enkripsi kunci dengan Algoritma *Dependent RSA 1* (DRSA-1)

No	Panjang <i>kunci</i> (Karakter)	Waktu Proses (millisecond)
1	185	11 ms
2	194	16 ms
3	200	28 ms
4	214	29 ms
5	236	35 ms

Selanjutnya adalah grafik pemeriksaan antara panjang kunci dengan lama waktu proses sistem yang dapat ditampilkan di gambar 4.11.

Grafik hubungan panjang kunci terhadap waktu proses enkripsi *Dependent RSA* (DRSA-1)



**Gambar 4.11.** Diagram yang menggambarkan hubungan panjang kunci dan waktu proses enkripsi menggunakan *Dependent RSA 1* (DRSA-1).



#### 4.3.3 Pengujian proses dekripsi cipherkey dengan Algoritma *Dependent RSA 1 (DRSA-1)*

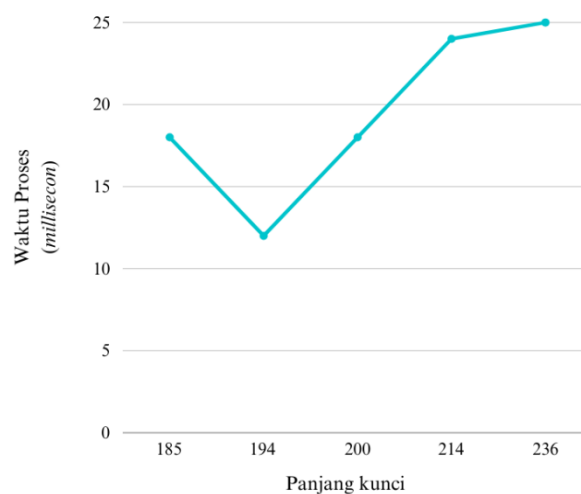
Tujuan pengujian ini adalah untuk menentukan apakah durasi proses enkripsi dan dekripsi dipengaruhi oleh panjang kunci yang diterapkan serta panjangnya itu sendiri.

**Tabel 4.8.** Waktu Proses dekripsi *cipherkey* menjadi kunci dengan Algoritma *Dependent RSA 1 (DRSA-1)*

No	Panjang <i>kunci</i> (Karakter)	Waktu Proses ( <i>millisecond</i> )
1	185	18 ms
2	194	12 ms
3	200	18 ms
4	214	24 ms
5	236	25 ms

Selanjutnya adalah grafik pemeriksaan antara panjang *plaintext* dengan lama waktu proses sistem yang dapat ditampilkan pada gambar 4.12.

Grafik hubungan panjang kunci terhadap waktu proses dekripsi *Dependent RSA (DRSA-1)*



**Gambar 4.12.** Diagram yang menggambarkan hubungan panjang kunci dan waktu proses enkripsi menggunakan *Dependent RSA 1 (DRSA-1)*.

#### 4.3.4 Pengujian proses dekripsi Algoritma *One Time Pad*

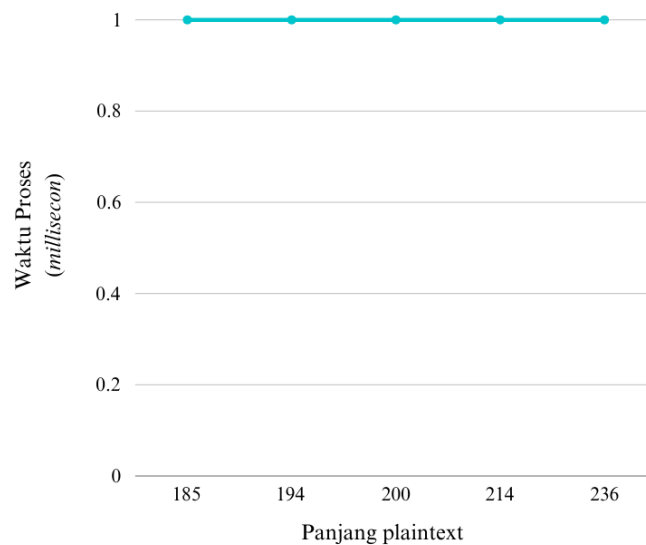
Alasan dilakukannya pengujian ini adalah untuk mengetahui dampak panjang teks biasa dan kunci yang digunakan terhadap jangka waktu sistem dekripsi.

**Tabel 4.9.** Waktu Proses Dekripsi dengan Algoritma *One Time Pad*

No	Panjang <i>plaintext</i> (Karakter)	Waktu Proses ( <i>millisecond</i> )
1	185	1 ms
2	194	1 ms
3	200	1 ms
4	214	1 ms
5	236	1 ms

Selanjutnya adalah grafik pemeriksaan antara panjang *plaintext* dengan lama waktu proses sistem yang dapat ditampilkan pada gambar 4.13.

Grafik hubungan panjang plaintext terhadap waktu proses dekripsi *One Time Pad*



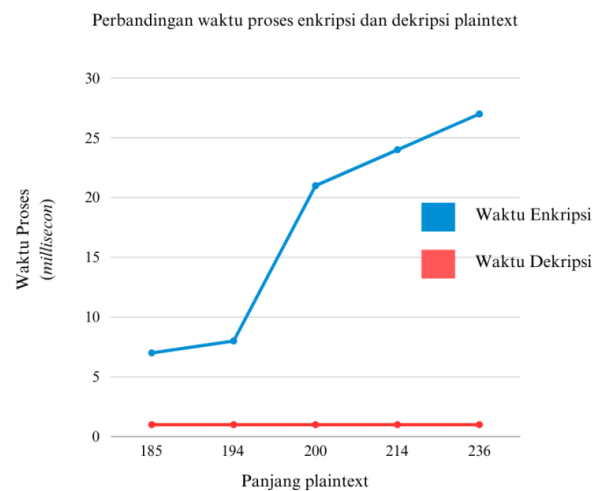
**Gambar 4.13.** Diagram yang menggambarkan hubungan panjang *plaintext* dan waktu proses dekripsi menggunakan *One Time Pad*.

#### 4.3.5 Perbandingan waktu proses enkripsi dan dekripsi pesan

Dibandingkan antara panjang *plaintext* dan waktu yang dibutuhkan untuk proses enkripsi, perbandingannya dapat dilihat dalam tabel 4.10 dan gambar 4.14.

**Tabel 4.10.** Perbandingan waktu proses enkripsi dan dekripsi pesan

No	Panjang <i>plaintext</i> (Karakter)	Waktu Proses enkripsi pesan ( <i>millisecond</i> )	Waktu Proses dekripsi pesan ( <i>millisecond</i> )
1	185	7 ms	1 ms
2	194	8 ms	1 ms
3	200	21 ms	1 ms
4	214	24 ms	1 ms
5	236	27 ms	1 ms



**Gambar 4.14.** Diagram yang membandingkan waktu enkripsi dan dekripsi *plaintext*

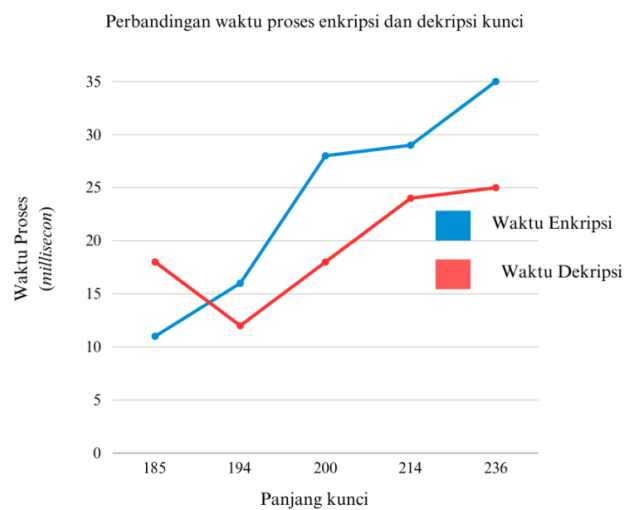
Dari tabel 4.10 dan gambar 4.14 di atas dapat dilihat dengan baik berapa lama waktu yang dibutuhkan untuk proses enkripsi dan decoding tersebut. Berbeda dengan saat pesan didekripsi, kerangka kerja membutuhkan lebih banyak waktu untuk mengenkripsinya.

#### 4.3.6 Perbandingan waktu proses enkripsi dan dekripsi kunci

Grafik perbandingan proses enkripsi dan dekripsi kunci berdasarkan panjang kunci dan waktu proses enkripsi masing-masing ditunjukkan pada Tabel 4.11 dan Gambar 4.15.

**Tabel 4.11.** Perbandingan waktu proses *enkripsi* dan *dekripsi* kunci

No	Panjang <i>kunci</i> (Karakter)	Waktu Proses enkripsi kunci ( <i>millisecond</i> )	Waktu Proses dekripsi kunci ( <i>millisecond</i> )
1	185	11 ms	18 ms
2	194	16 ms	12 ms
3	200	28 ms	18 ms
4	214	29 ms	24 ms
5	236	35 ms	25 ms



**Gambar 4.15.** Diagram yang membandingkan waktu enkripsi dan dekripsi kunci. Proses enkripsi kunci membutuhkan lebih banyak waktu daripada proses dekripsi kunci dalam sistem, seperti yang terlihat pada tabel 4.11 dan gambar 4.15 di atas.

## BAB 5

### KESIMPULAN DAN SARAN

#### 5.1 Kesimpulan

Dari penelitian ini dapat disimpulkan bahwa *One Time Pad* dengan *Mersenne Twister* dan *Dependent RSA 1* (DRSA-1) dalam skema Hybrid Cryptosystem, termasuk :

1. Sistem yang dirancang dapat mengembalikan string yang telah dienkripsi dengan Jumlah nilai karakter setelah dienkripsi akan semakin besar menjadi pesan asli setelah di dekripsi karena dirubah ke dalam bentuk ASCII.
2. Berdasarkan pengujian yang dilakukan sebanyak 5 kali, maka keberhasilan dalam menjalankan proses enkripsi dan dekripsi pesan adalah 100% dimana pesan setelah dienkripsi sama dengan pesan setelah di dekripsi.
3. Algoritma *Mersenne Twister* selain berukuran kecil karena hanya membutuhkan memori yang kecil, juga menghasilkan bilangan acak yang berkualitas tinggi dan memiliki periode yang sangat lama yaitu  $2^{19937} - 1$ .
4. Proses enkripsi One Time Pad (OTP) membutuhkan waktu rata-rata 20,6 milidetik, sedangkan enkripsi DRSA-1 memerlukan waktu rata-rata 20,8 milidetik. Untuk proses dekripsi, One Time Pad (OTP) memerlukan waktu rata-rata 1 milidetik, sementara dekripsi DRSA-1 membutuhkan waktu rata-rata 22,4 milidetik.
5. Metode *hybrid Cryptosystem* memiliki keunggulan yaitu keamanan yang tinggi dan efisiensi komputasi yang baik. Kunci sesi yang dihasilkan secara acak memastikan bahwa setiap sesi enkripsi memiliki kunci yang unik, sementara penggunaan kriptografi asimetris memungkinkan untuk pertukaran kunci yang aman tanpa memerlukan pertukaran langsung kunci sesi.

## 4.2 Saran

Berikut saran dari penulis untuk pengembangan sistem di masa depan, yaitu:

1. Pesan akan dienkripsi dan didekripsi selama penelitian ini dalam bentuk *plaintext*. diharapkan untuk penelitian selanjutnya semua jenis file, termasuk \*.txt, \*.docx, dan \*.otd.
2. Pada penelitian ini, proses enkripsi dan dekripsi hanya dapat memproses karakter string. Kedepannya penulis berharap proses enkripsi dan dekripsi dapat memproses file gambar, audio, atau file simbol lainnya.
3. Pada penelitian ini, sistem dirancang dan dibangun dengan berbasis *desktop*, untuk pengembangan selanjutnya sistem dapat dibangun di perangkat lain seperti berbasis *web* atau *android*.
4. Untuk penelitian selanjutnya diperlukan penelitian lebih lanjut mengeksplorasi optimasi algoritma kriptografi dalam konteks aplikasi nyata, serta potensi integrasi dengan teknologi baru seperti blockchain untuk meningkatkan keamanan dan efisiensi sistem.
5. Untuk penelitian selanjutnya diperlukan penelitian lebih lanjut mengeksplorasi penggunaan algoritma *One Time Pad* yang dapat dikombinasikan dengan *Stream Cipher*, dimana KSA dapat digunakan untuk menghasilkan kunci yang aman. Selain itu *Descendurin Algorithm* sebagai lapisan tambahan untuk menambah kerumitan dan variasi dalam proses enkripsi, sehingga mempersulit upaya dekripsi.

## DAFTAR PUSTAKA

- Febriansyah. (2012). Analisis dan Perancangan Keamanan Data Menggunakan Algoritma Kriptografi DES (Data Encyption Standard). *Skripsi*, XV(November), 1–2.
- Fidelis Asterina Surya Prasetya. (2016). *Dokumen Menggunakan Algoritma One Time Pad Dengan Pembangkit Bilangan Implementation Conventional Encryption Algorithm Using One Time Pad With the Mersenne Twister Random*.
- Ginting, A., Isnanto, R. R., & Windasari, I. P. (2015). Implementasi Algoritma Kriptografi RSA untuk Enkripsi dan Dekripsi Email. *Jurnal Teknologi Dan Sistem Komputer*, 3(2), 253. <https://doi.org/10.14710/jtsiskom.3.2.2015.253-258>.
- Harahap, M. K. (2019). *Analisis Algoritma One Time Pad Dengan Algoritma Cipher Transposisi Sebagai Pengamanan Pesan Teks*. 1(April 2017), 58–62.
- Informatika, S., Teknik, F., & Komputer, I. (2023). *Metode Pengamanan Komunikasi E-Mail pada Perangkat Mobile Berbasis Android dengan Menggunakan Metode Hybrid Cryptosystem*. 7, 12173–12181.
- Jagannatham, A. (1997). *Mersenne Twister – A Pseudo Random Number Generator and its Variants*.
- Munir, R. (2011). Algoritma Enkripsi Citra dengan Pseudo One-Time Pad yang Menggunakan Sistem Chaos. *Konferense Nasional Informatika*, 12–16.
- Pointcheval, D. (1999). New public key cryptosystems based on the dependent-RSA problems. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1592, 239–254. [https://doi.org/10.1007/3-540-48910-X\\_17](https://doi.org/10.1007/3-540-48910-X_17).
- Pratiwi, L. E., Marwati, R., & Yusnitha, I. (2014). Program Aplikasi Kriptografi Penyandian One Time Pad Menggunakan Sandi Vigenere. *Jurnal EurekaMatika*, 2(1), 43–53.
- Rachmawati, D., & Budiman, M. A. (2020). On Using the First Variant of *Dependent RSA 1* Encryption Scheme to Secure Text: A Tutorial. *Journal of Physics:*

- Conference Series*, 1542(1). <https://doi.org/10.1088/1742-6596/1542/1/012024>.
- Riza. (2021). *IMPLEMENTASI METODE SIGNCRYPTION MENGGUNAKAN ALGORITMA DEPENDENT RSA 1 ( DRSA-1 ) DAN ALGORITMA ELGAMAL GENERAL LINEAR DALAM PENGAMANAN DATA TEKS Universitas Sumatera Utara*.
- Romindo. (2018). Analisa Perbandingan Algoritma Monoalphabetic Cipher Dengan Algoritma *One Time Pad* Sebagai Pengamanan Pesan Teks. *Sinkron (Jurnal Dan Penelitian Teknik Informasi*, 2(2), 62–66.
- Saragih, N. E. (2018). Implementasi Algoritma *One Time Pad* pada Pesan. *Jurnal Ilmiah Matrik*, Vol.20 No.(3), 31–40.
- Sentot Kromodimoeljo. (2010). *Teori & Aplikasi Kriptografi*.
- Schneier, B. 1996. *Applied Cryptography: Protocols, Algorithms and Source Code in C*. 2nd Edition. John Wiley & Sons, Inc: New Jersey.
- Sianipar, K. D. R., Siahaan, S. W., Siregar, M., & Gunawan, I. (2019). Pengamanan File Suara Menggunakan Kriptografi Algoritma Rijndael Dengan Proses Enkripsi Dan Dekripsi. *TECHSI - Jurnal Teknik Informatika*, 11(3), 431. <https://doi.org/10.29103/techsi.v11i3.1967>.
- Siswanto, L. A. (2016). *Analisis Keacakan Generator Angka Pseudorandom Mersenne Twister dengan Metode Diehard Test*. 13513024.