

**PENGAMANAN *DATABASE FIELD-LEVEL ENCRYPTION* MENGGUNAKAN  
*HYBRID CRYPTOSYSTEM* ALGORITMA *HILL CIPHER* DAN  
ALGORITMA *RABIN P***

**SKRIPSI**

**SASTRA HARAPAN GULO  
201401066**



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

**PENGAMANAN *DATABASE FIELD-LEVEL ENCRYPTION* MENGGUNAKAN  
*HYBRID CRYPTOSYSTEM* ALGORITMA *HILL CIPHER* DAN  
ALGORITMA *RABIN P***

**SKRIPSI**

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

**SASTRA HARAPAN GULO**

**201401066**



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

**LEMBAR PERSETUJUAN**

Judul : PENGAMANAN DATABASE *FIELD-LEVEL ENCRYPTION* MENGGUNAKAN *HYBRID CRYPTOSYSTEM* ALGORITMA *HILL CIPHER* DAN ALGORITMA *RABIN P*

Kategori : SKRIPSI

Nama : SASTRA HARAPAN GULO


Nomor Induk Mahasiswa : 201401066

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

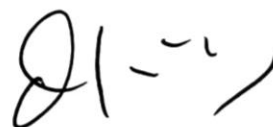
Dosen Pembimbing II



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

NIP. 197401272002122001

Dosen Pembimbing I



Dian Rachmawati S.Si., M.Kom.

NIP. 198307232009122004

Diketahui/Disetujui Oleh  
Ketua Program Studi S-1 Ilmu Komputer



Dr. Amalia S.T., M.T.  
NIP. 19781221 201404 2 001

**PERNYATAAN**

PENGAMANAN *DATABASE FIELD-LEVEL ENCRYPTION* MENGGUNAKAN  
*HYBRID CRYPTOSYSTEM* ALGORITMA *HILL CIPHER* DAN ALGORITMA  
*RABIN P*

**SKRIPSI**

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

Medan, Oktober 2024



Sastra Harapan Gulo

201401066

## **PENGHARGAAN**

Penulis mengucapkan puji syukur kepada Tuhan Yang Maha Esa atas rahmat dan hidayah-Nya, yang memungkinkan penulis untuk menyelesaikan skripsi ini sebagai salah satu syarat untuk meraih gelar Sarjana Komputer di Program Studi S1 Ilmu Komputer, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara.

Penulis ingin menyampaikan rasa hormat dan terima kasih yang sebesar-besarnya kepada:

1. Bapak Prof. Dr. Muryanto Amin, S.Sos, M.Si. selaku Rektor Universitas Sumatera Utara.
2. Ibu Dr. Maya Silvi Lydia, B.Sc., M.Sc. selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara sekaligus selaku Dosen Pembimbing Akademik dan Dosen Pembimbing II yang telah memberikan bimbingan, kritik, motivasi, dan saran kepada penulis dalam menyelesaikan skripsi ini.
3. Ibu Dr. Amalia ST., M.T. selaku Ketua Program Studi S1 Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
4. Ibu Sri Melvani Hardi S.Kom., M.Kom selaku Sekretaris Program Studi S1 Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
5. Ibu Dian Rachmawati S.Si., M.Kom., M.Kom. selaku dosen pembimbing I yang telah memberikan bimbingan, kritik, motivasi, dan saran serta arahan menuju kebaikan kepada penulis dalam menyelesaikan skripsi ini.
6. Bapak Dr. Jos Timanta Tarigan S.Kom., M.Sc. selaku dosen penguji yang telah memberikan arahan dan bimbingan kepada penulis dalam menyelesaikan skripsi ini.
7. Ibu Dr. Ir. Elviawaty Muisa Zamzami S.T., M.T., M.M., IPU. selaku dosen penguji yang telah memberikan arahan dan bimbingan kepada penulis dalam menyelesaikan skripsi ini.
8. Seluruh Bapak dan Ibu Dosen Program Studi S-1 Ilmu Komputer yang telah memberikan waktu dan tenaga untuk mengajar dan membimbing sehingga penulis dapat sampai kepada tahap penyusunan skripsi ini.

9. Orang tua penulis yang selalu memberikan motivasi, dukungan, doa serta saran dan kasih sayang penuh kepada penulis dalam menyelesaikan pendidikan.
10. Saudara/i penulis yang selalu memberikan doa dan dukungan kepada penulis dalam menyelesaikan pendidikan.
11. Sahabat penulis Erick Yudha Pratama Sukku dan Chalil Al Vareel yang telah memberikan dukungan, saran, dan masukan kepada penulis.
12. Teman-teman penulis Anak Yas So Na Wilbert, Wepe, Aldo, Theo, dan sebagainya yang telah memberikan dukungan dan semangat kepada penulis.

Penulis menyadari bahwa skripsi ini masih belum sempurna, oleh karena itu penulis terbuka terhadap kritik dan saran untuk perbaikan lebih lanjut. Semoga skripsi ini dapat memberikan manfaat bagi perkembangan ilmu pengetahuan.

Medan, Oktober 2024

Penulis,

Sastra Harapan Gulo

201401066

PENGAMANAN DATABASE FIELD-LEVEL ENCRYPTION MENGGUNAKAN  
HYBRID CRYPTOSYSTEM ALGORITMA HILL CIPHER DAN  
ALGORITMA RABIN P

**ABSTRAK**

Keamanan data dalam *database* adalah perihal yang sangat penting untuk menghindari pihak yang tidak bertanggungjawab dapat melihat, mengambil, dan memanipulasi data tersebut sehingga diperlukan algoritma kriptografi untuk mengamankan data dalam *database* agar sulit dimengerti. Algoritma *Hill Cipher* merupakan algoritma simetris yang memiliki keunggulan karena cepat dalam proses enkripsi dan dekripsi data namun lemah terhadap *known-plaintext attack*, maka diperlukan *field-level-encryption* agar setiap baris data dalam *database* memiliki kunci yang berbeda dalam proses enkripsi. Namun hal ini menyebabkan sulitnya manajemen kunci. Algoritma *Rabin P* merupakan algoritma asimetris yang memiliki tingkat keamanan yang sangat baik namun *ciphertext* yang dihasilkan beberapa kali lebih panjang dari *plaintext*. *Hybrid Cryptosystem* merupakan penggabungan algoritma simetris dan asimetris dengan memanfaatkan keunggulannya masing-masing. Kunci algoritma *Hill Cipher* untuk enkripsi data dibangkitkan menggunakan *Blum-Blum-Shub*. Pada proses enkripsi dengan *Hybrid Cryptosystem* algoritma *Hill Cipher* dan algoritma *Rabin P* serta pembangkit bilangan acak *Blum-Blum-Shub* berhasil mengamankan data dalam *database* sehingga data yang disimpan menjadi sulit dimengerti dan kunci yang digunakan untuk enkripsi berbeda setiap baris data pada *database* dan terenkripsi. Proses dekripsi dapat mengembalikan data yang telah terenkripsi ke bentuk semula sehingga dapat dimengerti tanpa ada data yang hilang. Waktu yang diperlukan untuk enkripsi dan dekripsi berbanding lurus dengan jumlah karakter. Pembaharuan kunci algoritma *Rabin P* dilakukan dengan mendekripsi data (*ciphertext*) menggunakan kunci yang lama kemudian mengenkripsi ulang data (*plaintext*) menggunakan kunci baru.

Kata kunci : *Blum-Blum-Shub*, *Hill Cipher*, *Hybrid Cryptosystem*, Kriptografi, *Rabin P*

SECURING THE DATABASE WITH FIELD-LEVEL ENCRYPTION USING  
HYBRID CRYPTOSYSTEM HILL CIPHER ALGORITHM AND  
RABIN P ALGORITHM

**ABSTRACT**

Data security in databases is crucial to prevent unauthorized parties from viewing, accessing, and manipulating the data. Cryptography algorithms are required to secure the data in the database to make it incomprehensible. The Hill Cipher algorithm is a symmetric algorithm that is advantageous due to its speed in the encryption and decryption processes but is weak against known-plaintext attacks. Therefore, field-level encryption is necessary to ensure each row of data in the database has a different encryption key, which complicates key management. The Rabin P algorithm, an asymmetric algorithm, offers high security but generates ciphertext several times longer than the plaintext. A Hybrid Cryptosystem combines symmetric and asymmetric algorithms, leveraging their respective strengths. The Hill Cipher algorithm key for data encryption is generated using Blum-Blum-Shub. In the encryption process with the Hybrid Cryptosystem, the Hill Cipher algorithm, the Rabin P algorithm, and the Blum-Blum-Shub random number generator successfully secure the data in the database, making it incomprehensible and ensuring that the encryption keys are different for each row of data and encrypted. The decryption process can return the encrypted data to its original form without any data loss. The time required for encryption and decryption is directly proportional to the number of characters. Key renewal for the Rabin P algorithm is performed by decrypting the data (ciphertext) using the old key and then re-encrypting the data (plaintext) using the new key.

Key Word : Blum-Blum-Shub, Cryptography, Hill Cipher, Hybrid Cryptosystem, Rabin P



## DAFTAR ISI

<b>LEMBAR PERSETUJUAN .....</b>	<b>iii</b>
<b>PERNYATAAN.....</b>	<b>iv</b>
<b>PENGHARGAAN.....</b>	<b>v</b>
<b>ABSTRAK .....</b>	<b>vii</b>
<b>ABSTRACT.....</b>	<b>viii</b>
<b>DAFTAR ISI.....</b>	<b>ix</b>
<b>DAFTAR TABEL .....</b>	<b>xi</b>
<b>DAFTAR GAMBAR.....</b>	<b>xii</b>
<b>BAB 1 PENDAHULUAN .....</b>	<b>1</b>
1.1. Latar Belakang .....	1
1.2. Rumusan Masalah .....	3
1.3. Tujuan Penelitian.....	3
1.4. Manfaat Penelitian.....	3
1.5. Batasan Masalah.....	3
1.6. Metode Penelitian.....	4
1.7. Sistematika Penulisan.....	4
<b>BAB 2 LANDASAN TEORI .....</b>	<b>6</b>
2.1. Kriptografi.....	6
2.2. Database .....	7
2.3. Pembangkit Bilangan Acak .....	8
2.4. Blum-Blum-Shub .....	8
2.5. Hybrid Cryptosystem .....	10
2.6. Hill Cipher.....	10
2.7. Rabin P .....	13
2.8. Penelitian Relevan.....	16
<b>BAB 3 ANALISIS DAN PERANCANGAN .....</b>	<b>18</b>
3.1. Analisis.....	18
3.1.1. Analisis Masalah .....	18
3.1.2. Analisis Kebutuhan.....	19
3.2. Perancangan Sistem.....	20

3.2.1. Diagram Umum .....	20
3.2.2. Use Case Diagram.....	22
3.2.3. Activity Diagram.....	23
3.2.4. Sequence Diagram.....	25
3.2.5. Class Diagram.....	27
3.2.6. Diagram Alir (Flowchart) .....	28
3.2.7. Rancangan Logical Record Structure (LRS) Database.....	36
<b>BAB 4 IMPLEMENTASI DAN PENGUJIAN.....</b>	<b>37</b>
4.1. Implementasi Sistem .....	37
4.1.1. Lingkungan Pengembangan .....	37
4.1.2. Struktur Library.....	37
4.1.3. Implementasi Fungsi Utama.....	39
4.1.4. Dokumentasi Penggunaan Library.....	43
4.1.5. Perhitungan Manual.....	43
4.2. Pengujian Sistem .....	49
4.2.1. Install Library.....	49
4.2.2. Import Fungsi .....	50
4.2.3. Penyimpanan dan Pemanggilan Kunci Rabin P .....	50
4.2.4. Create Data (Enkripsi) .....	50
4.2.5. Show Data (Dekripsi) .....	52
4.2.6. Update Kunci Rabin P .....	54
4.2.7. Pengujian Sistem Terhadap Waktu.....	58
<b>BAB 5 PENUTUP .....</b>	<b>60</b>
5.1. Kesimpulan.....	60
5.2. Saran.....	60
<b>DAFTAR PUSTAKA.....</b>	<b>61</b>
<b>LAMPIRAN A DOKUMENTASI PENGGUNAAN LIBRARY .....</b>	<b>64</b>

**DAFTAR TABEL**

	Halaman
Tabel 2.1 Extended Euclidean .....	15
Tabel 4.1 Extended Euclidean .....	46
Tabel 4.2 Hasil Pengujian Jumlah Karakter Terhadap Waktu Proses .....	58

## DAFTAR GAMBAR

	Halaman
Gambar 2.1 Diagram enkripsi dan dekripsi pesan .....	6
Gambar 2.2 Skema sistem kriptografi simetri .....	7
Gambar 2.3 Skema sistem kriptografi asimetri.....	7
Gambar 3.1 Diagram Ishikawa Masalah Penelitian.....	19
Gambar 3.2 Diagram Umum.....	21
Gambar 3.3 <i>Use Case Diagram</i> .....	22
Gambar 3.4 <i>Activity Diagram</i> Enkripsi .....	24
Gambar 3.5 <i>Activity Diagram</i> Dekripsi .....	25
Gambar 3.6 <i>Sequence Diagram</i> Enkripsi.....	26
Gambar 3.7 <i>Sequence Diagram</i> Dekripsi .....	27
Gambar 3.8 Class Diagram .....	27
Gambar 3.9 <i>Flowchart</i> Enkripsi .....	28
Gambar 3.10 <i>Flowchart</i> Dekripsi .....	29
Gambar 3.11 Flowchart Proses Pembangkitan Bilangan Acak Blum-Blum-Shub.....	30
Gambar 3.12 <i>Flowchart</i> Enkripsi <i>Hill Cipher</i> .....	31
Gambar 3.13 <i>Flowchart</i> Dekripsi <i>Hill Cipher</i> .....	32
Gambar 3.14 <i>Flowchart</i> Pembangkit Kunci <i>Rabin P</i> .....	33
Gambar 3.15 <i>Flowchart</i> Enkripsi <i>Rabin P</i> .....	34
Gambar 3.16 <i>Flowchart</i> Dekripsi <i>Rabin P</i> .....	35
Gambar 3.17 Rancangan LRS Database .....	36
Gambar 4.1 Struktur <i>Library</i> .....	37
Gambar 4.2 Fungsi Enkripsi Data.....	39
Gambar 4.3 Fungsi Dekripsi Data.....	41

Gambar 4.4 <i>Install Library</i> .....	49
Gambar 4.5 Pemanggilan Fungsi Enkripsi dan Dekripsi dari <i>Library</i> .....	50
Gambar 4.6 File '.env' .....	50
Gambar 4.7 Pemanggilan Variabel Kunci <i>Rabin P</i> .....	50
Gambar 4.8 Fungsi <i>Create</i> .....	51
Gambar 4.9 Contoh data soal .....	51
Gambar 4.10 <i>Response</i> Fungsi <i>Create</i> .....	52
Gambar 4.11 Tabel Soal setelah <i>Create</i> .....	52
Gambar 4.12 Fungsi Show Data .....	53
Gambar 4.13 Tampilan id 2 Tabel Soal .....	53
Gambar 4.14 <i>Response</i> Fungsi Show .....	54
Gambar 4.15 Fungsi <i>rabinPKeyUpdate</i> .....	55
Gambar 4.16 Tabel <i>Database</i> Sebelum Update Kunci <i>Rabin P</i> .....	56
Gambar 4.17 Data dekripsi menggunakan kunci <i>Rabin P</i> yang lama .....	56
Gambar 4.18 Tabel <i>Database</i> Setelah Update Kunci <i>Rabin P</i> .....	57
Gambar 4.19 Data dekripsi menggunakan kunci <i>Rabin P</i> yang baru .....	57
Gambar 4.20 Grafik Pengujian Jumlah Karakter Terhadap Waktu Proses.....	59

# BAB 1

## PENDAHULUAN

### 1.1. Latar Belakang

Penyimpanan data dalam *database* pada saat ini menjadi sesuatu hal yang memerlukan perhatian khusus dalam hal keamanannya. Seringnya penyimpanan data dalam *database* dilakukan tanpa adanya pengamanan sehingga pihak yang tidak bertanggungjawab dapat melihat, mengambil dan memanipulasi data tersebut (Rahmadhiyanti, 2019). Terlebih lagi apabila data yang disimpan dalam *database* merupakan informasi yang sensitif atau pribadi maka diperlukan pengamanan yang ekstra. Data yang dimaksud seperti data pribadi (nama, tanggal lahir, alamat, nomor identitas) yang dapat disalahgunakan oleh orang yang tidak bertanggungjawab apabila mendapatkan data tersebut. Sehingga diperlukan suatu teknik untuk menjaga kerahasiaan data di *database*.

Algoritma kriptografi sebagai teknik untuk mengamankan pesan dapat dijadikan suatu opsi untuk pengamanan *database*. Dalam penelitian (Andrico, et al., 2018) menunjukkan bahwa algoritma kriptografi dapat mengamankan data dalam *database* sehingga data lebih aman dari pihak yang tidak bertanggungjawab. Kriptografi algoritma *Hill Cipher* dan algoritma *Rabin p* masih menjadi perhatian dalam penelitian terkait dengan keamanan data.

Penelitian Terdahulu menunjukkan bahwa algoritma *Hill Cipher* dapat mengatasi pencurian (*sniffing*) dan manipulasi (*spoofing*) data dengan metode enkripsi dan dekripsi (Wanto, 2015). *Hill Cipher* sebagai kriptografi simetris menggunakan matriks untuk enkripsi dan dekripsinya serta aritmatika modulo. Dikarenakan pada dasar enkripsi dan dekripsinya menggunakan perkalian matriks, setiap karakter yang sama pada *plaintext* akan diganti menjadi karakter yang berbeda pada *ciphertext* sehingga sulit dipecahkan dengan menggunakan teknik analisis frekuensi (Hidayatuloh, et al., 2021).

Algoritma *Hill Cipher* cepat dalam proses enkripsi dan dekripsinya namun masih dapat diserang dengan *known-plaintext attack* (Munir, 2019). Oleh sebab itu peneliti

membuat *Field-Level Encryption* dimana data akan dienkripsi dengan kunci yang berbeda di setiap barisnya. Kunci yang digunakan akan dibangkitkan memakai algoritma *Blum-Blum-Shub* sebagai pembangkit bilangan acak. Karena kunci yang dipakai pada setiap baris data dalam *database* berbeda maka sulit untuk distribusi dan manajemen kunci sehingga perlu dikembangkan lagi dengan kombinasi dengan algoritma asimetris misalnya algoritma *Rabin p* (Siregar, et al., 2022).

Dalam penelitian (Budiman, et al., 2023) dikatakan bahwa tingkat keamanan algoritma *Rabin p* sangat baik, namun *ciphertext* yang dihasilkan berkali-kali lipat lebih panjang dari *plaintext*. Algoritma *Rabin p* adalah kriptografi asimetris yang memiliki dua buah kunci yakni kunci publik yang bisa disebarluaskan dan kunci privat yang bersifat rahasia. Algoritma *Rabin p* bergantung pada kesulitan dalam pemfaktoran angka yang besar. Semakin jauh selisih kunci  $p$  dan  $q$  yang digunakan semakin susah algoritma *Rabin p* dibobol (Budiman, et al., 2019).

Oleh sebab itu, salah satu solusi untuk keamanan distribusi kunci dan rasio ukuran *ciphertext* adalah *Hybrid Cryptosystem*. *Hybrid Cryptosystem* adalah penggabungan antara algoritma simetris dan algoritma asimetris dengan memanfaatkan keunggulan masing-masing. Data dienkripsi memakai algoritma simetris dan kunci simetris dienkripsi memakai algoritma asimetris (Rachmawati, et al., 2019). Penelitian (Santoso, 2021) menunjukkan bahwa algoritma *Hill Cipher* dan RSA dapat dikombinasi dalam mengamankan pesan. Untuk pengamanan data dalam *database* maka sebelum data disimpan dalam *database* data akan dienkripsi memakai algoritma *Hill Cipher* dengan kunci yang berbeda-beda tiap baris dalam tabel *database* (*Field-Level Encryption*). Kunci yang digunakan dalam enkripsi data dibangkitkan dengan memakai algoritma *blum-blum-shub*. Kunci *Hill Cipher* kemudian dienkripsi menggunakan algoritma asimetris yaitu algoritma *Rabin p* dan disimpan dalam kolom id tiap baris.

Mencermati hal yang diuraikan di atas, penulis berkeinginan membuat sebuah penelitian dengan judul **“Pengamanan *Database* Field-Level Encryption Menggunakan Hybrid Cryptosystem Algoritma Hill Cipher dan Algoritma Rabin P”**. Dalam penelitian ini akan dikembangkan sebuah *Library Javascript* untuk enkripsi dan dekripsi database sehingga dapat digunakan/diterapkan oleh pengguna.

## 1.2. Rumusan Masalah

Mengacu pada latar belakang penelitian, keamanan data dalam *database* sangat diperlukan mengingat adanya potensi risiko pencurian dan manipulasi data-data penting atau sensitif. Dan pada penelitian ini akan dikaji bagaimana penerapan *Hybrid Cryptosystem* algoritma *Hill Cipher* dan algoritma *Rabin p* dan pembangkit bilangan acak *Blum-Blum-Shub* untuk meningkatkan keamanan dalam penyimpanan data dalam *database Field-Level Encryption*.

## 1.3. Tujuan Penelitian

Penelitian ini dilakukan guna membangun sebuah *Library Javascript* untuk meningkatkan keamanan dalam penyimpanan data dalam *database Field-Level Encryption* dengan menerapkan *Hybrid Cryptosystem* algoritma *Hill Cipher* dan algoritma *Rabin p* dan pembangkit bilangan acak *blum-blum-shub*.

## 1.4. Manfaat Penelitian

Berikut manfaat dari penelitian ini, diantaranya adalah:

1. Meningkatkan keamanan penyimpanan data dalam *database*.
2. Memberikan informasi tentang bagaimana penerapan *Hybrid Cryptosystem* algoritma *Hill Cipher* dan algoritma *Rabin p* dan pembangkit bilangan acak *Blum-Blum-Shub*

## 1.5. Batasan Masalah

Batasan masalah dalam penelitian ini terdiri atas:

1. Penelitian ini hanya membahas tentang teknik enkripsi dan dekripsi memakai algoritma *Hill Cipher* dan *Rabin p* dan pembangkit bilangan acak *blum-blum-shub* tanpa membahas algoritma yang lain.
2. Kode teks yang digunakan yaitu ASCII (*American Standard Code for Information Interchange*).
3. Pembangkit bilangan acak yang digunakan adalah *Blum-Blum-Shub*
4. Yang akan dibangun adalah *Library* dari bahasa pemrograman *Javascript*



5. DBMS yang digunakan adalah *PostgreSQL Relational Database*
6. Bahasa pemrograman yang dipakai dalam penelitian ini adalah *Javascript*.

### 1.6. Metode Penelitian

Penelitian ini dilakukan dengan menggunakan metode penelitian sebagai berikut:

#### 1. Studi Pustaka

Tahap pertama yang dilakukan adalah dengan mencari referensi melalui buku-buku, jurnal, e-book, artikel ilmiah, makalah ataupun situs internet yang berhubungan dengan *Hill Cipher*, *Rabin p*, dan pembangkit bilangan acak *Blum-Blum-Shub*.

#### 2. Analisis dan Perancangan

Penulis menganalisis segala sesuatu yang diperlukan dalam penelitian dan kemudian dibuat dalam diagram alir (*flowchart*), *use case diagram*, *activity diagram*, *sequence diagram* dan diagram ishikawa, *class diagram*.

#### 3. Implementasi

Membuat sebuah *library* dengan menggunakan bahasa pemrograman *Javascript* sesuai dengan hasil analisis dan perancangan.

#### 4. Pengujian

Sistem kemudian diuji coba untuk melakukan pengamanan *database* dengan menggunakan *Hybrid Cryptosystem* algoritma *Hill Cipher* dan *Rabin p* dan pembangkit bilangan acak *Blum-Blum-Shub*.

#### 5. Dokumentasi

Hasil dari penelitian ini dimulai dari tahap analisis hingga tahap pengujian akan didokumentasikan dalam bentuk skripsi.

### 1.7. Sistematika Penulisan

Adapun sistematika penulisan penelitian ini terdiri dari lima bab, yakni:

#### BAB 1 PENDAHULUAN

Berisikan latar belakang pemilihan judul, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, metodologi penelitian, dan sistematika penulisan

**BAB 2      LANDASAN TEORI**

Berisikan tinjauan teoritis yang berkaitan dengan Kriptografi, Algoritma *Hill Cipher*, Algoritma *Rabin P*, serta pembangkit bilangan acak *Blum-Blum-Shub*.

**BAB 3      ANALISIS DAN PERANCANGAN**

Berisikan penjelasan tentang analisis masalah serta bagaimana sistem akan dirancang.

**BAB 4      IMPLEMENTASI DAN PENGUJIAN**

Berisikan tentang implementasi dan pengujian sistem dari analisis dan perancangan yang telah dilakukan.

**BAB 5      PENUTUP**

Berisikan uraian kesimpulan dan saran dari penelitian yang telah dilakukan.

## BAB 2

### LANDASAN TEORI

#### 2.1. Kriptografi

Secara harafiah kriptografi dalam bahasa Yunani berarti “*secret writing*” (tulisan rahasia). Seiring perkembangannya, kriptografi pun dapat diartikan sebagai ilmu dan seni melindungi kerahasiaan pesan dengan mengubah pesan menjadi tidak bermakna (Munir, 2019). Pesan dalam hal ini merupakan pesan yang dikirim dan pesan yang tersimpan (*database*). *Plaintext* adalah teks asli yang dapat dibaca dan bermakna. *Ciphertext* merupakan teks yang sudah tidak bermakna.

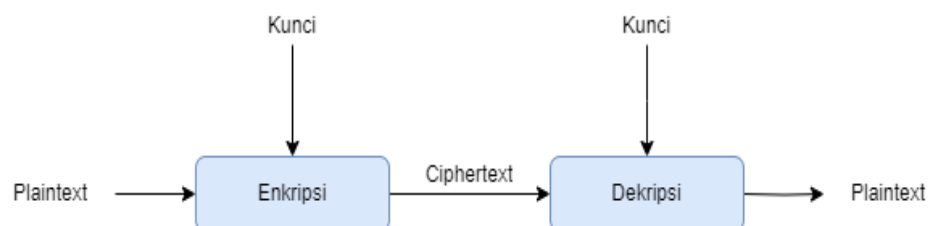
Dalam kriptografi terdapat dua buah proses, yakni:

##### 1. Enkripsi

Enkripsi adalah proses pengubahan suatu kode dari teks biasa (*plaintext*) menjadi kode yang tidak bermakna (*ciphertext*) (Mukhtar, 2018). Proses ini mengubah data yang hendak disimpan kedalam database kebentuk yang sudah tidak bermakna.

##### 2. Dekripsi

Dekripsi adalah proses mengubah kembali *ciphertext* menjadi *plaintext* semula sehingga bisa dimengerti maknanya (Munir, 2019). Proses ini mengubah data yang telah dienkripsi kembali kebentuk aslinya sehingga data dapat dibaca dan dimengerti.

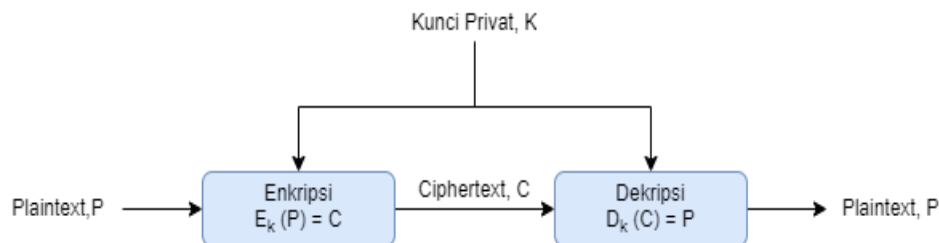


**Gambar 2.1 Diagram enkripsi dan dekripsi pesan**

Kriptografi bisa dibedakan menjadi dua jenis berdasarkan jenis kunci pada proses enkripsi dan dekripsinya, yakni sebagai berikut:

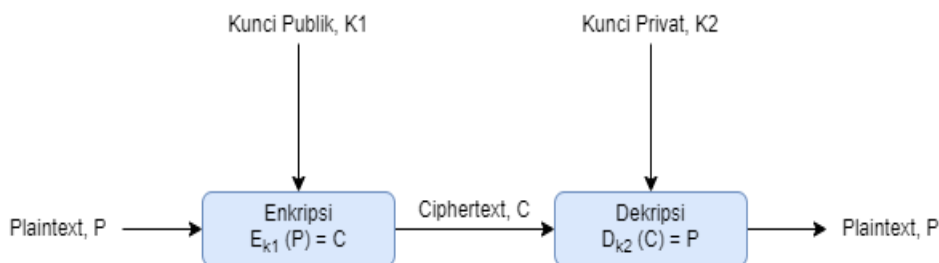
1. Kriptografi kunci simetris (*Symmetric-key Cryptography*) adalah algoritma kriptografi yang metode enkripsi dan dekripsinya menggunakan kunci yang sama.

Kriptografi kunci simetris dapat disebut juga sebagai kriptografi kunci privat. Kriptografi simetris diantaranya adalah *Hill Cipher*, *Caesar Cipher*, *Affine Cipher*, *DES*, *Blowfish*, *AES*, dan lain sebagainya. (Munir, 2019).



**Gambar 2.2 Skema sistem kriptografi simetri**

2. Kriptografi kunci asimetris (*Asymmetric-key Cryptography*) merupakan kriptografi yang metode enkripsi dan dekripsinya memakai kunci yang berbeda. Kriptografi asimetris dikenal juga dengan nama kriptografi kunci publik. Konsep kriptografi asimetris adalah satu kunci digunakan sebagai kunci publik untuk didistribusikan kepada semua orang sebagai kunci enkripsi dan satu kunci privat harus disimpan dan tidak dibagikan sebagai kunci dekripsi. Contoh algoritma asimetris adalah *Rabin p*, *RSA*, *Elgama*, *DSA*, *Diffie-Hellman*, dan lain-lain (Easttom, 2021).



**Gambar 2.3 Skema sistem kriptografi asimetri**

## 2.2. Database

*Database* dapat diartikan sebagai sekumpulan informasi atau data yang terintegrasi yang disimpan secara sistematis dan (Fitri, 2020). Menurut (Jayanti, et al., 2018) database adalah sekumpulan data yang di integrasi dan diorganisasi untuk memenuhi kebutuhan suatu organisasi. Data yang disimpan dalam *database* dapat berupa data-data penting seperti informasi pribadi, data perusahaan, data-data pegawai seperti nama, tanggal lahir, alamat, dan sebagainya. Mengingat pentingnya data dalam organisasi maka sangat penting pengamanan data yang ada dalam *database*.

Menurut (Malik, et al., 2016) beberapa cara untuk mengamankan *database* adalah enkripsi data, kontrol akses, kebijakan inferensi, identifikasi/autentikasi pengguna, akuntabilitas dan audit. Dalam penelitian (Ruswandi, et al., 2022) menunjukkan bahwa enkripsi *database* menggunakan algoritma kriptografi AES-128 dan *Vigenere Cipher* berhasil mengamankan *database* pada aplikasi *helpdesk*. Penelitian (Setiawan, et al., 2021) juga menunjukkan penggunaan metode kriptografi algoritma RC4 berhasil diimplemtasikan untuk mengamankan data penggajian karyawan pada database aplikasi penggajian karyawan PT. Trans Intra Asia.

### 2.3. Pembangkit Bilangan Acak

Bilangan acak adalah bilangan yang kemunculan dan nilainya sulit ditebak. Bilangan acak merupakan aspek penting dalam kriptografi yang dapat diterapkan pada algoritma dan kunci yang memerlukan bilangan acak. Terdapat dua jenis bilangan acak yaitu bilangan acak sejati (*true random numbers*) dan bilangan semi-acak (*pseudorandom numbers*) (Munir, 2019). Pembangkit bilangan semi-acak biasa disebut dengan *pseudorandom number generator* (PRNG).

Pembangkit bilangan acak yang aman sangat diperlukan oleh kriptografi. Aman berarti pihak lawan tidak bisa memprediksi kemunculannya. Pembangkit bilangan acak yang aman disebut *cryptographically secure pseudorandom generator* (CSPRNG). Sebuah CSPRNG wajib memenuhi dua persyaratan berikut:

1. Memiliki karakteristik statistik yang baik.
2. Mampu bertahan dari serangan serius yang bermaksud untuk memperkirakan bilangan acak yang dihasilkan. (Munir, 2019).

### 2.4. Blum-Blum-Shub

*Blum Blum Shub* (BBS) adalah salah satu CSPRNG sederhana dan efisien. BBS dirancang untuk menghasilkan bilangan acak yang sulit diprediksi, sehingga cocok digunakan dalam berbagai aplikasi kriptografi yang membutuhkan keamanan tinggi. Kelebihan utamanya adalah kemampuannya dalam menjaga kualitas bilangan acak yang dihasilkan, sehingga menjadi pilihan yang tepat untuk sistem yang memerlukan keamanan berbasis enkripsi.

Berikut adalah proses pembangkitan bilangan acak dengan *Blum-Blum-Shub* (BBS):

1. Tentukan dua bilang prima sebagai kunci,  $p$  dan  $q$  di mana  $p \equiv q \equiv 3 \pmod{4}$ .

$$p = 19, q = 23$$

2. Kemudian kalikan menjadi  $n = pq$ .

$$n = pq = 19 \cdot 23 = 437$$

3. Tentukan bilangan bulat acak yang lain,  $s$ , yang menjadi umpan sehingga:

- a.  $2 \leq s < n$

- b.  $s$  relatif prima dengan  $n$

$$s = 63$$

4. hitung  $x_0 = s^2 \bmod n$

$$x_0 = 63^2 \bmod 437 = 305$$

5. Barisan bilangan acak akan dihasilkan dari pengulangan perhitungan berikut sepanjang yang diperlukan:

- a. Hitung  $x_i = x_{i-1}^2 \bmod n$

$$x_1 = x_0^2 \bmod n = 305^2 \bmod 437 = 381$$

$$x_2 = x_1^2 \bmod n = 381^2 \bmod 437 = 77$$

$$x_3 = x_2^2 \bmod n = 77^2 \bmod 437 = 248$$

$$x_4 = x_3^2 \bmod n = 248^2 \bmod 437 = 324$$

$$x_5 = x_4^2 \bmod n = 324^2 \bmod 437 = 96$$

$$x_6 = x_5^2 \bmod n = 96^2 \bmod 437 = 39$$

$$x_7 = x_6^2 \bmod n = 39^2 \bmod 437 = 210$$

$$x_8 = x_7^2 \bmod n = 210^2 \bmod 437 = 400$$

$$x_9 = x_8^2 \bmod n = 400^2 \bmod 437 = 58$$

Barisan bilangan acak adalah 381, 77, 248, 324, 96, 39, 210, 400, dan 58.

## 2.5. Hybrid Cryptosystem

*Hybrid Cryptosystem* adalah penggabungan antara kriptografi algoritma simetris dan kriptografi algoritma asimetris dengan memanfaatkan keunggulan masing-masing (Smart, 1999). Data dienkripsi memakai algoritma simetris dan kunci simetris dienkripsi menggunakan algoritma asimetris (Rachmawati, et al., 2019). Penelitian (Jamaludin, et al., 2020) menunjukkan bahwa *Hybrid Cryptosystem* kombinasi algoritma *Vigenere Cipher* dan RSA dapat diimplementasikan untuk menambah keamanan pesan dan mempercepat proses enkripsi pesan. Penelitian (Budiman, et al., 2021) menunjukkan bahwa *Hybrid Cryptosystem* kombinasi algoritma *Rabin P* dan *Affine Cipher* dapat dilakukan untuk enkripsi dan dekripsi data tanpa kehilangan integritas data.

Berikut ini merupakan cara kerja *Hybrid Cryptosystem*:

1. Penerima pesan membuat sepasang kunci asimetris kunci publik dan kunci privat. Kunci publik dikirim ke pengirim pesan sedangkan kunci privat disimpan oleh penerima pesan.
2. Pengirim pesan membuat sebuah kunci simetri dan mengenkripsi pesan yang ingin dikirim dengan menggunakan algoritma simetri.
3. Kunci simetri dienkripsi pengirim menggunakan kunci publik asimetris milik penerima pesan.
4. Pesan dan kunci yang telah dienkripsi dikirim ke penerima pesan.
5. Penerima pesan menggunakan kunci privatnya untuk mendekripsi kunci simetri yang terenkripsi. Kemudian penerima menggunakan kunci tersebut untuk mendeskripsi pesan yang sebelumnya dienkripsi oleh pengirim.

## 2.6. Hill Cipher

Algoritma *Hill Cipher* adalah kriptografi simetris yang diperkenalkan oleh Lester S. Hill pada tahun 1929. Algoritma ini menggunakan matriks sebagai kunci dalam proses enkripsi dan dekripsi serta aritmatika modulo. Untuk melakukan dekripsi diperlukan perhitungan matriks balikan (*invers*) terlebih dahulu, karena matriks *invers* dapat digunakan untuk mengubah *ciphertext* menjadi *plaintext* (Munir, 2019).

Algoritma *Hill Cipher* dipilih karena cepat dan memakai perkalian matriks dalam melakukan enkripsi dan dekripsinya. Jadi setiap karakter yang sama pada *plaintext* akan diubah menjadi karakter yang berbeda pada *ciphertext* sehingga sulit dipecahkan dengan menggunakan teknik analisis frekuensi (Siregar, Faisal, & Handoko, 2022). Algoritma *Hill Cipher* juga dapat mengatasi pencurian (*sniffing*) dan manipulasi (*spoofing*) data memakai metode enkripsi dan dekripsi (Wanto, 2015).

Berikut adalah tahapan-tahapan beserta contoh enkripsi dan dekripsi algoritma *Hill Cipher*:

### 1. Enkripsi

#### a. Menentukan *plaintext*

P = Sastra Gulo

Konversi ke nilai ASCII

S=83, a=97, s=115, t=116, r=114, a=97, spasi=32, G=71, u=117, l=108,  
o=111

Ubah menjadi bentuk matriks dengan panjang kolom sama dengan panjang baris matriks kunci, jika kolom tidak terpenuhi maka ditambahkan dengan kode ASCII “spasi = 32”

$$P = \begin{bmatrix} 83 & 116 & 32 & 108 \\ 97 & 114 & 71 & 111 \\ 115 & 97 & 117 & 32 \end{bmatrix}$$

#### b. Menentukan matriks kunci 3x3 yang digunakan

$$K = \begin{bmatrix} 119 & 246 & 49 \\ 124 & 196 & 213 \\ 82 & 146 & 64 \end{bmatrix}$$

#### c. Melakukan proses enkripsi dengan rumus $C = K.P \bmod 256$

Keterangan:

C = Ciphertext

K = Matriks kunci

P = Plaintext

$$C = \begin{bmatrix} 119 & 246 & 49 \\ 124 & 196 & 213 \\ 82 & 146 & 64 \end{bmatrix} \times \begin{bmatrix} 83 & 116 & 32 & 108 \\ 97 & 114 & 71 & 111 \\ 115 & 97 & 117 & 32 \end{bmatrix} \bmod 256$$

$$C = \begin{bmatrix} 206 & 9 & 127 & 254 \\ 39 & 45 & 53 & 236 \\ 142 & 46 & 232 & 38 \end{bmatrix}$$



## 2. Dekripsi

- a. Menghitung matriks balikan (*invers*) dari matriks kunci dengan mod 256

$$K = \begin{bmatrix} 119 & 246 & 49 \\ 124 & 196 & 213 \\ 82 & 146 & 64 \end{bmatrix}$$

Perhitungan matriks balikan dapat dilakukan dengan rumus berikut:

$$K = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}, K^{-1} = \frac{1}{\det(K)} \begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \end{bmatrix}^T$$

Keterangan:

$$A = (ei - hf) = (1 \times 19 - 4 \times 26) \bmod 256 = -85 \bmod 256 = 171$$

$$B = -(di - fg) = -(20 \times 19 - 2 \times 26) \bmod 256 = -328 \bmod 256 = 184$$

$$C = (dh - eg) = (20 \times 4 - 2 \times 1) \bmod 256 = 78 \bmod 256 = 78$$

$$D = -(bi - ch) = -(33 \times 19 - 4 \times 5) \bmod 256 = -607 \bmod 256 = 161$$

$$E = (ai - cg) = (7 \times 19 - 2 \times 5) \bmod 256 = 123 \bmod 256 = 123$$

$$F = -(ah - bg) = -(7 \times 4 - 2 \times 33) \bmod 256 = 38 \bmod 256 = 38$$

$$G = (bf - ec) = (33 \times 26 - 1 \times 5) \bmod 256 = 853 \bmod 256 = 85$$

$$H = -(af - cd) = -(7 \times 26 - 20 \times 5) \bmod 256 = -82 \bmod 256 = 174$$

$$I = (ae - bd) = (7 \times 1 - 33 \times 20) \bmod 256 = -653 \bmod 256 = 115$$

$$\det(K) = aA + bB + cC = 7 \times (-85) + 33 \times (-328) + 5 \times 78 = -11029 \bmod 256 = 235$$

$$\frac{1}{\det(K)} = \frac{1}{235} = 235^{-1} \bmod 256 = 195$$

Maka nilai matriks balikan dari matriks kunci adalah

$$K^{-1} = 195 \begin{bmatrix} 171 & 161 & 85 \\ 184 & 123 & 174 \\ 78 & 38 & 115 \end{bmatrix} \bmod 256$$

$$K^{-1} = \begin{bmatrix} 65 & 163 & 191 \\ 40 & 177 & 138 \\ 106 & 242 & 153 \end{bmatrix}$$

- b. Melakukan proses dekripsi dengan rumus  $P = K^{-1} \cdot C \bmod 256$

Keterangan:

P = Plaintext

K = Matriks balikan (*invers*) dari matriks kunci

C = Ciphertext

$$P = \begin{bmatrix} 65 & 163 & 191 \\ 40 & 177 & 138 \\ 106 & 242 & 153 \end{bmatrix} \times \begin{bmatrix} 5 & 195 & 80 & 227 \\ 139 & 92 & 169 & 31 \\ 179 & 227 & 11 & 244 \end{bmatrix} \mod 256$$

$$P = \begin{bmatrix} 83 & 116 & 32 & 108 \\ 97 & 114 & 71 & 111 \\ 115 & 97 & 117 & 32 \end{bmatrix}$$

## 2.7. Rabin P

Algoritma kriptografi *Rabin-P* adalah algoritma kriptografi asimetris (kunci-publik) yang diperkenalkan oleh Michael O. Rabin dan Richard M. Karp pada tahun 1987 (Srivastava, et al., 2013). Algoritma Rabin p memiliki dua macam kunci, yaitu kunci publik dan kunci privat. Kunci publik akan disebarluaskan secara umum untuk proses enkripsi, sedangkan kunci privat yang terdiri dari dua bilangan prima dan bersifat rahasia. Semakin jauh selisih kunci  $p$  dan  $q$  yang digunakan semakin sulit bagi pihak-pihak yang tidak bertanggungjawab untuk menembus algoritma *Rabin-p* (Budiman, et al., 2019).

Berikut adalah tahapan-tahapan beserta contoh enkripsi dan dekripsi algoritma *Rabin p*:

### 1. Menentukan kunci

- a. Bangkitkan 2 buah bilangan prima,  $p$  dan  $q$ , dengan kondisi sebagai berikut:

$$p \neq q \rightarrow p \equiv q \equiv 3 \mod 4$$

Contoh:

$$p = 13$$

→ 13 (bilangan prima)

$$\rightarrow 13 \mod 4 = 3$$

$$q = 19$$

→ 19 (bilangan prima)

$$\rightarrow 19 \mod 4 = 3$$

- b. Hitung nilai  $n$

$$n = p \cdot q$$

$$= 139.191$$

$$= 26549$$

- c. Publis  $n$  sebagai kunci publik dan simpan  $p$  dan  $q$  sebagai kunci privat

## 2. Enkripsi

- a. Dapatkan kunci public

$$n = 26549$$

- b. Menentukan pesan yang ingin dikirim kemudian konversi ke ASCII.

Kemudian konversi lagi ke biner, panjangkan bilangan biner dengan dirinya sendiri, dan konversi bilangan biner ke decimal.

$$m = \text{"7"} = 55_{10}$$

$$m = 55_{10} = 110111_2$$

$$m = 110111|110111 \rightarrow \text{panjangkan}$$

$$m = 3575_{10}$$

- c. Enkripsi dengan rumus

$$C = m^2 \bmod n$$

$$C = 3575^2 \bmod 26549$$

$$C = 10556$$

- d. Kirimkan ciphertext C ke penerima

## 3. Dekripsi

- a. Terima ciphertext C dari pengirim

$$C = 10556$$

- b. Cari nilai  $y_p$  dan  $y_q$  dengan Extended Euclidean GCD

$$y_p \cdot p + y_q \cdot q = \text{GCD}(p, q)$$

$$y_p \cdot p + y_q \cdot q = 1$$

$$139 \cdot y_p + 191 \cdot y_q = 1$$

Temukan nilai  $y_p$  dan  $y_q$  menggunakan Extended Euclidean:

**Tabel 2.1 Extended Euclidean**

$y_p$	$y_q$	<b>D</b>	<b>k</b>
0	1	191	
1	0	139	1
-1	1	52	2
3	-2	35	1
11	-8	1	

$$y_p = 11 \text{ dan } y_q = -8$$

- c. Hitung nilai  $m_p$  dan  $m_q$

$$m_p = C^{\frac{p+1}{4}} \bmod p \qquad m_q = C^{\frac{q+1}{4}} \bmod q$$

$$m_p = 10556^{\frac{139+1}{4}} \bmod 139 \qquad m_p = 10556^{\frac{191+1}{4}} \bmod 191$$

$$m_p = 100 \qquad m_p = 54$$

- d. Hitunglah nilai dari  $r$ ,  $s$ ,  $t$  dan  $u$  memakai *Chinese Remainder Theorem* (CRT).

- Hitunglah nilai variabel  $v$  dan  $w$  terlebih dahulu untuk memudahkan perhitungan.

$$v = y_p \cdot p \cdot m_q = 11 \cdot 139 \cdot 54 = 82566$$

$$w = y_q \cdot q \cdot m_p = -8 \cdot 191 \cdot 100 = -152800$$

- Hitunglah nilai dari  $r$ ,  $s$ ,  $t$  dan  $u$

$$r = (v + w) \bmod n = 9413$$

$$s = (v - w) \bmod n = 22974$$

$$t = (-v + w) \bmod n = 3575$$

$$u = (-v - w) \bmod n = 17136$$

- e. Konversi  $r$ ,  $s$ ,  $t$  dan  $u$  ke biner

$$r = 9413_{10} = 10010011000101_2$$

$$s = 22974_{10} = 101100110111110_2$$

$$t = 3575_{10} = 110111110111_2$$

$$u = 17136_{10} = 100001011110000_2$$

- f. Tentukan plaintext dari nilai  $r$ ,  $s$ ,  $t$  dan  $u$
- $r = 9413_{10} = 1001001|1000101_2$  (kiri  $\neq$  kanan)
  - $s = 22974_{10} = 101100110111110_2$  (Jumlah digit ganjil)
  - $t = 3575_{10} = 110111|110111_2$  (kiri = kanan)
  - $u = 17136_{10} = 100001011110000_2$  (Jumlah digit ganjil)

Pesan ada dalam  $t$

$$m = 110111_2 = 55_{10} = \text{"7"}$$

## 2.8. Penelitian Relevan

1. Penelitian oleh (Andrico, et al., 2018) membahas penggunaan teknik enkripsi untuk mengamankan data informasi akademik. Penelitian ini menerapkan algoritma RC4 dalam aplikasi untuk memudahkan pihak sekolah dalam mengelola data dengan pengamanan yang cukup baik. Hasil penelitian menunjukkan bahwa penggunaan algoritma kriptografi dalam hal ini algoritma RC4 efektif dalam menjaga kerahasiaan dan mengamankan data informasi akademik dari orang-orang yang tidak bertanggungjawab.
2. Penelitian yang dilakukan oleh (Siregar, et al., 2022) membahas penggunaan algoritma *Hill Cipher* dalam pengamanan file. Penelitian ini menerapkan algoritma *Hill Cipher* sebagai pengaman file teks yang berisi informasi penting dari pihak luar. Hasil penelitian ini menunjukkan bahwa algoritma *Hill Cipher* efektif dalam mengamankan file teks dan dapat memproses karakter, symbol, dan angka dalam kode ASCII.
3. Penelitian yang dilakukan oleh (Budiman, et al., 2019) menganalisis algoritma *Rabin P* dengan menggunakan faktorisasi fermat untuk menguji keamanannya. Algoritma *Rabin P* yang keamanannya bergantung pada kesulitan faktorisasi bilangan besar, dibandingkan dengan metode faktorisasi Fermat, yang mencari dua bilangan berbeda kuadratnya yang jumlahnya sama dengan bilangan yang akan difaktorkan. Hasil penelitian ini menunjukkan bahwa meskipun benar bahwa terdapat kecenderungan bahwa semakin besar nilai  $n$  semakin besar waktu pemfaktoran, nilai  $n$  tidak selalu berkorelasi dengan waktu yang dibutuhkan untuk pemfaktoran (pemrosesan) kunci algoritma *rabin p*. Faktor

yang mempengaruhi kecepatan pemfaktoran kunci publik *rabin p* dengan menggunakan metode *Fermat factorization* adalah perbedaan kunci privat ( $p$  dan  $q$ ), semakin jauh perbedaan kunci ( $p$  dan  $q$ ) maka semakin lama waktu yang diperlukan untuk memfaktorkan kunci public *rabin p*.

4. Penelitian oleh (Rachmawati, et al., 2019) membahas penggunaan algoritma kriptografi dalam mengamankan pesan. Penelitian ini menerapkan *Hybrid Cryptosystem* kombinasi algoritma *Hill Cipher* dan *Elgama* sebagai pengaman pesan instan. Dalam penelitian ini pesan di enkripsi memakai algoritma *Hill Cipher*, dan kunci algoritma *Hill Cipher* dienkripsi menggunakan algoritma *Elgama*. Penelitian ini menghasilkan kesimpulan bahwa penggunaan *hybrid cryptosystem* yang menggabungkan algoritma *Hill Cipher 3x3* dan *ElGamal* memiliki potensi untuk memberikan keamanan yang tinggi dalam mengamankan pesan instan, meskipun perlu untuk diperhatikan bahwa kinerja sistem dapat dipengaruhi oleh panjang teks yang dienkripsi dan dekripsi.

## **BAB 3**

### **ANALISIS DAN PERANCANGAN**

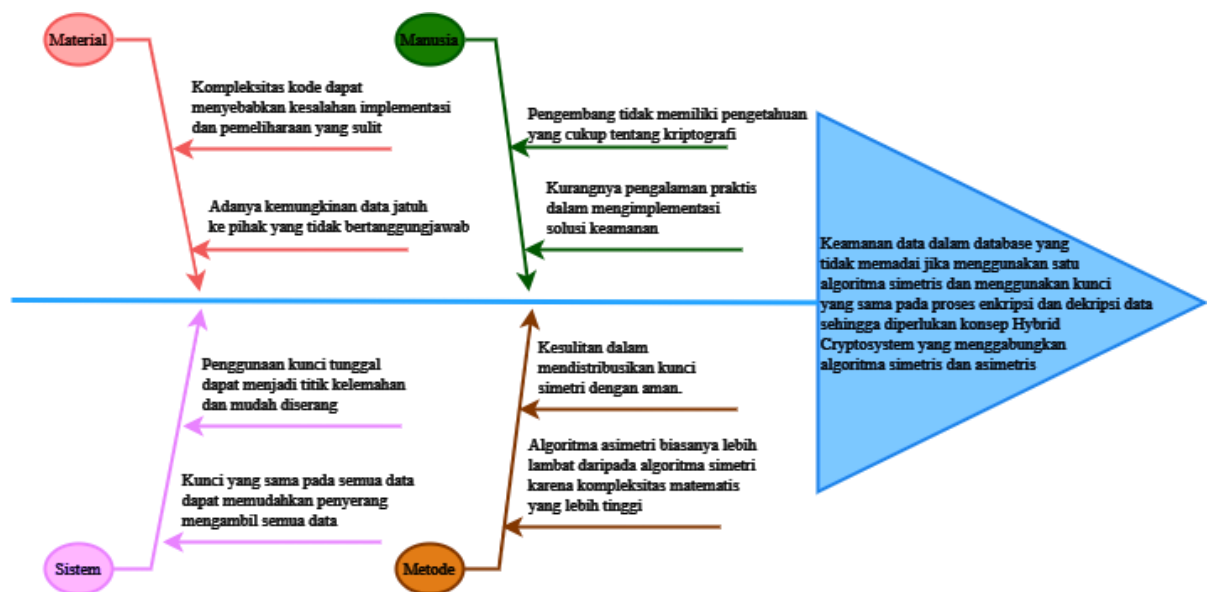
#### **3.1. Analisis**

Analisis digunakan untuk membuat gambaran secara umum pada perangkat lunak. Hasil dari analisis berfungsi untuk menjadi pedoman dasar melakukan perancangan kebutuhan sistem yang akan dibangun pada penelitian ini.

##### **3.1.1. Analisis Masalah**

Masalah keamanan data dalam *database* merupakan suatu aspek penting dalam sebuah sistem informasi. Hal ini dikarenakan oleh kemajuan teknologi yang membuat seseorang yang tidak bertanggungjawab menjadi lebih mudah untuk mengambil, memanipulasi, dan mengambil data-data sensitif dalam *database*. Pada gambar 3.1 adalah diagram masalah umum pada penelitian ini memakai Diagram Ishikawa (*Fishbone Diagram*).

Pada gambar 3.1 bisa dilihat bahwasanya yang menjadi masalah utama yang perlu diselesaikan pada penelitian ini terlihat pada bagian sebelah kanan (kepala ikan) yakni Keamanan data dalam *database* yang tidak memadai jika menggunakan satu algoritma simetris dan memakai kunci yang sama pada proses enkripsi dan dekripsi data. Oleh sebab itu, diperlukan konsep *Hybrid Cryptosystem* yang menggabungkan algoritma simetris dan asimetris. Terdapat empat faktor penyebab dari masalah yang mengarah pada tulang utama yaitu material, manusia, sistem, dan metode.



Gambar 3.1 Diagram Ishikawa Masalah Penelitian

### 3.1.2. Analisis Kebutuhan

#### 1. Analisa Kebutuhan Fungsional

Analisis kebutuhan fungsional adalah langkah penting dalam pengembangan perangkat lunak atau sistem dikarenakan mendefinisikan kebutuhan terkait input, proses, output apa saja yang diperlukan dalam sistem (Mulyani, 2016). Berikut adalah kebutuhan fungsional sistem:

- Menerima input data dan kunci  
Sistem harus dapat menerima data (*plaintext*) dan kunci yang diinput user.
- Mengembalikan output *ciphertext*  
Sistem harus dapat mengembalikan data (*ciphertext*) yang sudah acak dan tidak dimengerti maknanya.
- Membangkitkan kunci *Hill Cipher*  
Sistem (*library*) harus dapat membangkitkan kunci algoritma *Hill Cipher* dengan pembangkit bilangan acak *Blum-Blum-Shub*.
- Enkripsi *Field-Level*  
Sistem (*library*) harus dapat mengenkripsi data menggunakan *Hybrid Cryptosystem* algoritma *Hill Cipher* dan algoritma *Rabin p* pada tingkat field dalam *database*. Setiap field harus dapat dienkripsi dengan kunci simetri yang berbeda.



e. Dekripsi Field-Level

Sistem (library) harus dapat mendekripsi data menggunakan *Hybrid Cryptosystem* algoritma *Hill Cipher* dan algoritma *Rabin p* pada tingkat field dalam database. Dekripsi harus memakai kunci yang sesuai dengan kunci pada saat enkripsi.

f. Dokumentasi

*Library* harus disertai dengan dokumentasi yang jelas dan mudah dipahami, termasuk petunjuk penggunaan dan contoh implementasi.

## 2. **Analisi Kebutuhan Nonfungsional**

Analisis kebutuhan nonfungsional merupakan fungsi tambahan atau pelengkap pada sebuah sistem (Mulyani, 2016).

a. Performa

Sistem harus dapat mengembalikan hasil proses enkripsi yang telah di olah Kembali ke bentuk semula (dekripsi).

b. Efisiensi

Sistem yang dibangun mudah dipahami oleh user dan tidak menyulitkan dalam penggunaannya.

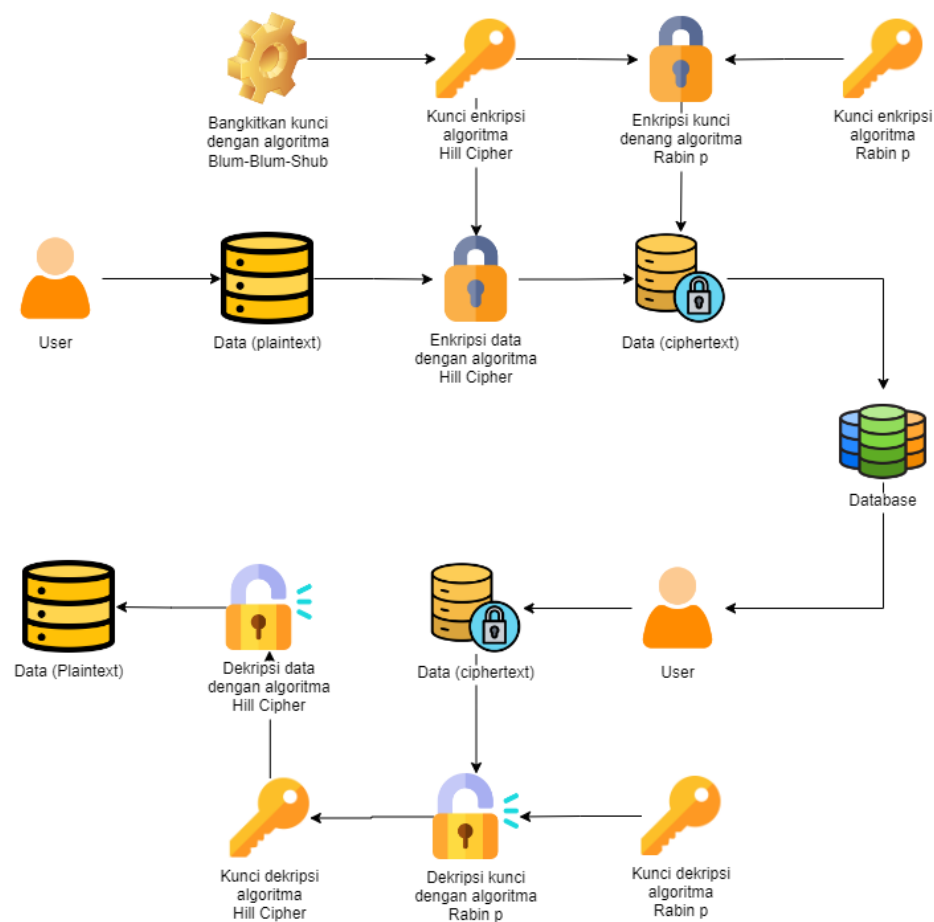
c. Kontrol

Sistem dapat mengembalikan pesan error untuk setiap proses yang tidak sesuai

## 3.2. **Perancangan Sistem**

### 3.2.1. **Diagram Umum**

Diagram umum berfungsi untuk menggambarkan secara keseluruhan sistem yang akan dirancang. Skema ini memberikan gambaran menyeluruh tentang bagaimana setiap komponen dalam sistem akan berinteraksi dan bekerja. Diagram umum penelitian ini bisa dilihat pada gambar 3.2.



**Gambar 3.2 Diagram Umum**

Berikut ini adalah diagram umum sistem pada gambar 3.2

1. Enkripsi data

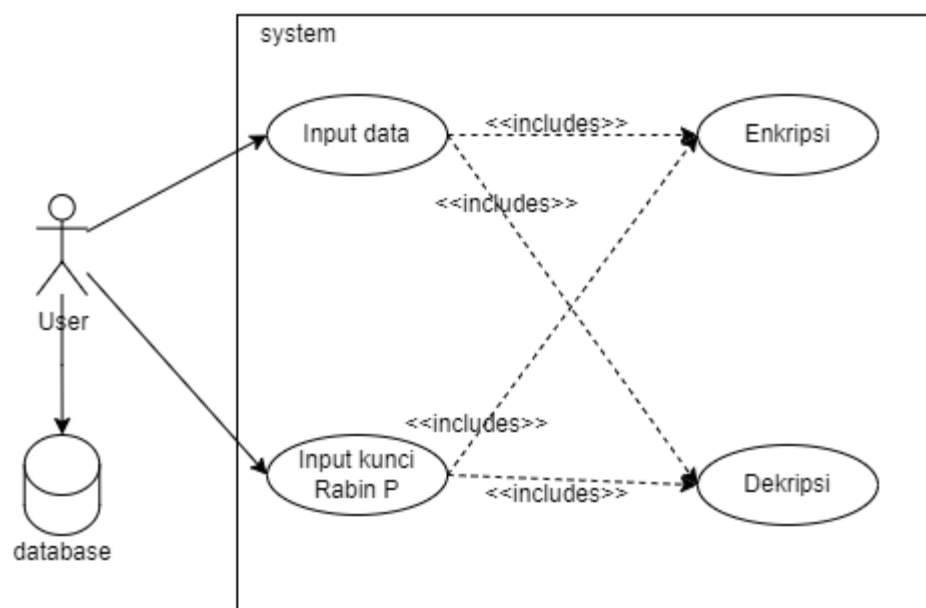
- a. User menyiapkan data yang akan disimpan ke database dan dua buah bilangan prima yang besar sebagai kunci algoritma *Rabin p*.
- b. Data (*plaintext*) kemudian diamankan/dienkripsi memakai algoritma *Hill Cipher* dengan kunci yang dibangkitkan dengan pembangkit bilangan acak *Blum-Blum-Shub* kemudian menghasilkan data yang sudah dienkripsi (*ciphertext*). Kunci yang dipakai untuk enkripsi algoritma *Hill Cipher* dienkripsi menggunakan algoritma *Rabin p* dengan kunci publik dari user kemudian menghasilkan kunci yang sudah dienkripsi (*ciphertext*). Kunci (*ciphertext*) dapat disimpan sebagai id dari data yang telah dienkripsi.
- c. Data (*ciphertext*) dan kunci (*ciphertext*) disimpan ke dalam database.

## 2. Dekripsi

- a. User mengambil data (*ciphertext*) dari database
- b. Id atau kunci (*ciphertext*) dari data (*ciphertext*) didekripsi menggunakan algoritma *Rabin p* dengan kunci privat dari user kemudian menghasilkan kunci yang asli (*plaintext*). Setelah itu, kunci tersebut digunakan untuk dekripsi data (*ciphertext*) menggunakan algoritma *Hill Cipher* sehingga memperoleh data yang asli (*plaintext*).

### 3.2.2. Use Case Diagram

*Use Case Diagram* merupakan visual yang merepresentasikan gambaran interaksi antara pengguna dan sistem (Kurniawan, 2018). Pada gambar 3.3 merupakan sebuah *Use Case Diagram* yang akan dibangun.



**Gambar 3.3 Use Case Diagram**

Pada gambar 3.3 terdapat satu aktor yang digunakan pada sistem yaitu User. User dapat menggunakan *library* untuk melakukan proses enkripsi dan dekripsi data sebelum menyimpan ke *database*. User dapat menjalankan sistem dengan menginputkan data (*plaintext*) dan kunci algoritma Rabin P kemudian melakukan proses enkripsi data. Dimana data akan dienkrpsi memakai algoritma *Hill Cipher* dengan kunci yang dibangkitkan oleh pembangkit bilangan acak *Blum-Blum-Shub* dan kunci *Hill Cipher* kemudian di enkripsi menggunakan algoritma *Rabin P* dengan menggunakan kunci

yang di input pengembang. Data hasil enkripsi dan kunci *Hill Cipher* hasil enkripsi di gabungkan (*ciphertext*). User juga dapat menjalankan sistem dengan menginputkan data (*ciphertext*) dan kunci algoritma *Rabin P* untuk melakukan proses dekripsi data. Dimana data akan di dekripsi memakai algoritma *Hill Cipher* yang menggunakan kunci yang di dekripsikan terlebih dahulu menggunakan algoritma *Rabin P* sehingga menghasilkan data semula (*plaintext*).

### 3.2.3. Activity Diagram

*Activity Diagram* mempresentasikan langkah-langkah aktivitas dalam sistem. *Activity Diagram* dipakai sebagai gambaran urutan dari langkah-langkah atau aktivitas yang terjadi dalam sistem (Bhattacharjee, et al., 2009).

#### 1. Activity Diagram Proses Enkripsi

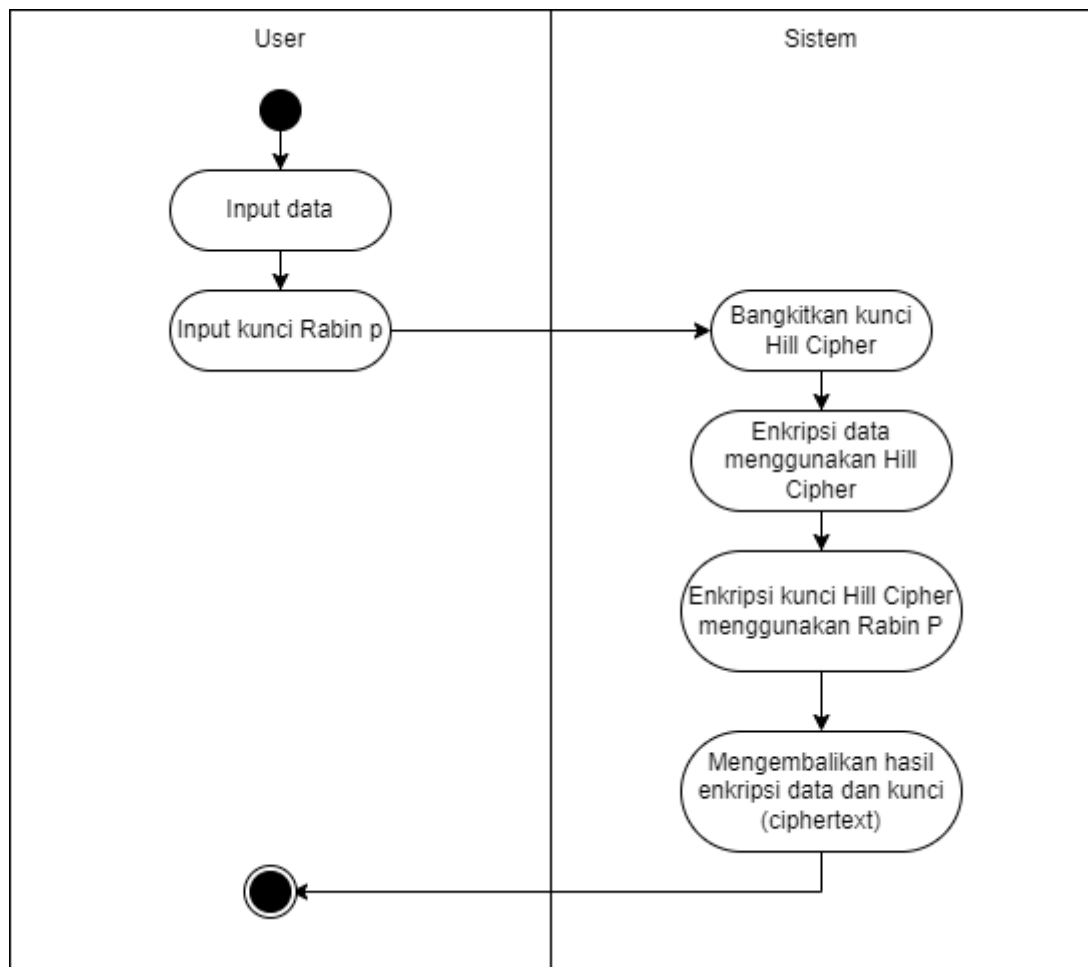
Pada gambar 3.4 menunjukkan *activity diagram* proses enkripsi yang terdapat dalam sistem yang akan dibangun. Terdapat dua buah kotak yaitu:

- a. Bagian sebelah kiri menggambarkan aktivitas yang dilakukan oleh user
- b. Bagian sebelah kanan menggambarkan respon dari sistem terhadap aktivitas yang dilakukan oleh user terhadap sistem.

Deskripsi aktivitas:

- a. User
  - User harus menginputkan data (*plaintext*) dan kunci algoritma *Rabin P*
- b. Sistem
  - Membangkitkan kunci algoritma *Hill Cipher* menggunakan pembangkit bilangan acak *Blum-Blum-Shub*,
  - Menenkripsi data (*plaintext*) menggunakan algoritma *Hill Cipher* dengan kunci yang sudah dibangkitkan.
  - Menenkripsi kunci *Hill Cipher* menggunakan algoritma *Rabin P*.
  - Mengembalikan hasil enkripsi data dan enkripsi kunci (*ciphertext*) kepada user.

Berikut ini dapat dilihat pada gambar 3.4 *Activity Diagram* Enkripsi.



**Gambar 3.4 Activity Diagram Enkripsi**

## 2. Activity Diagram Proses Dekripsi

Pada gambar 3.5 menunjukkan *activity diagram* proses dekripsi yang terdapat dalam sistem yang akan dibangun. Terdapat dua buah kotak yaitu:

- Bagian sebelah kiri menggambarkan aktivitas yang dilakukan oleh user
- Bagian sebelah kanan menggambarkan respon dari sistem terhadap aktivitas yang dilakukan oleh user terhadap sistem.

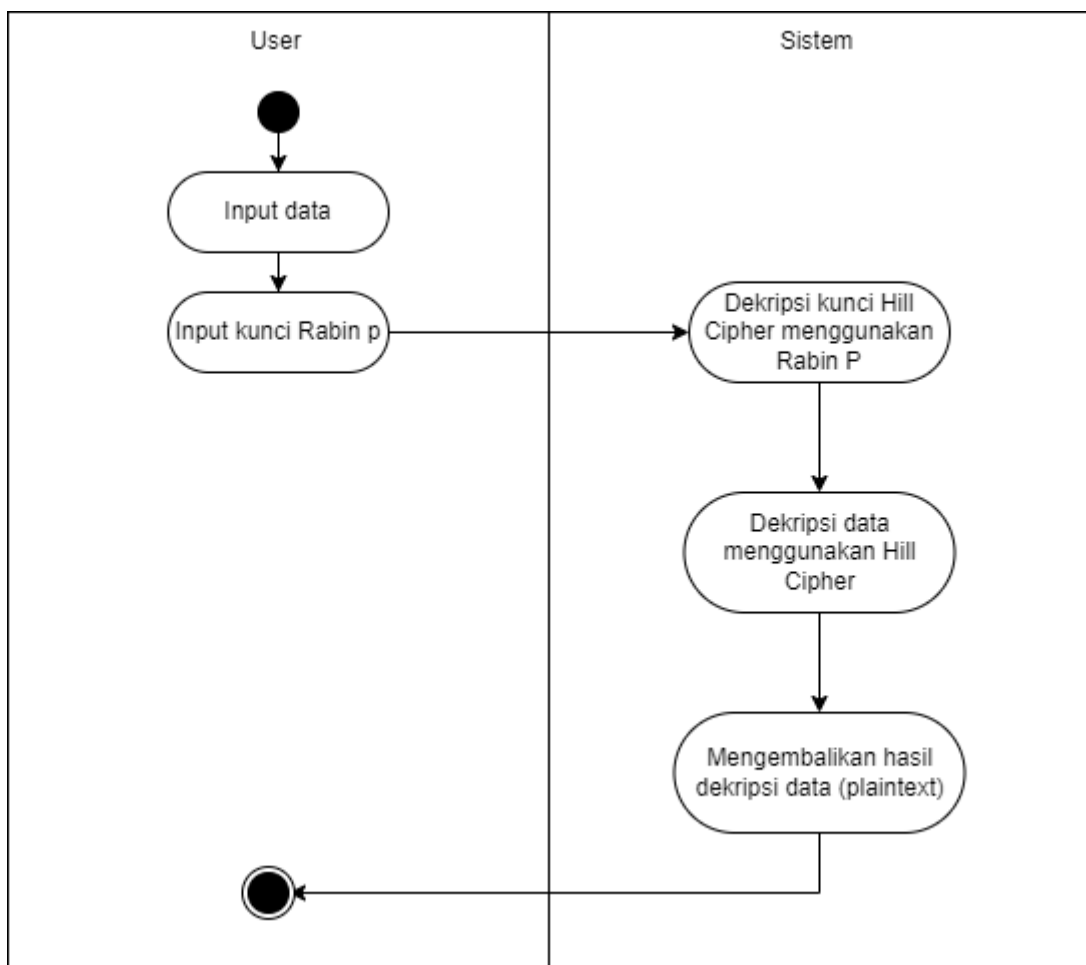
Deskripsi aktivitas:

- User
  - User harus menginputkan data (*ciphertext*) dan kunci algoritma *Rabin P*

b. Sistem

- Mendekripsi kunci *Hill Cipher* menggunakan algoritma *Rabin P*.
- Mendekripsi data (*plaintext*) menggunakan algoritma *Hill Cipher* dengan kunci yang sudah dibangkitkan.
- Mengembalikan hasil dekripsi data (*plaintext*) dan kunci kepada user.

Berikut ini dapat dilihat pada gambar 3.5 *Activity Diagram* Enkripsi.



**Gambar 3.5 Activity Diagram Dekripsi**

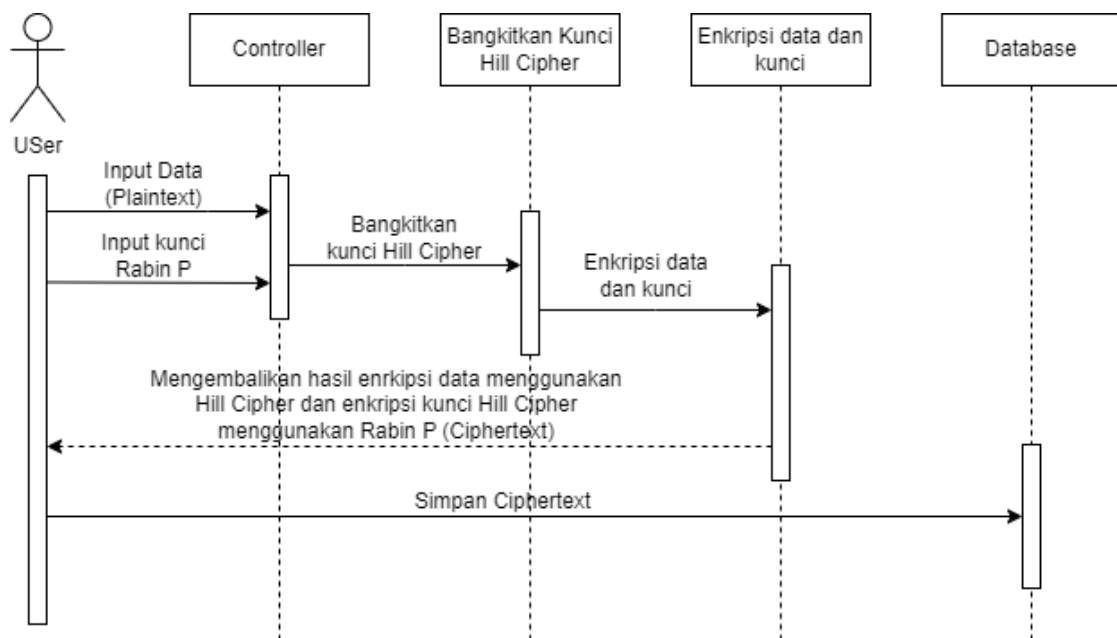
### 3.2.4. Sequence Diagram

*Sequence diagram* merepresentasikan grafis dari interaksi antar objek dalam sistem yang terjadi secara berurutan berdasarkan waktu. *Sequence diagram* menunjukkan bagaimana objek saling berkomunikasi melalui pesan yang dikirimkan dan diterima, dengan setiap objek diwakili oleh garis hidup (*lifeline*) vertikal (Rumbaugh, et al.,

1999). Aktivitas yang dilakukan oleh objek ditunjukkan oleh persegi panjang tipis pada *lifeline*, dan pesan yang dikirimkan digambarkan oleh panah horizontal. *Sequence diagram* membantu memvisualisasikan aliran kontrol dan data dalam sistem, mengidentifikasi tugas dan tanggung jawab setiap objek, serta mendokumentasikan skenario penggunaan untuk perancangan dan implementasi sistem.

### 1. Sequence Diagram Enkripsi

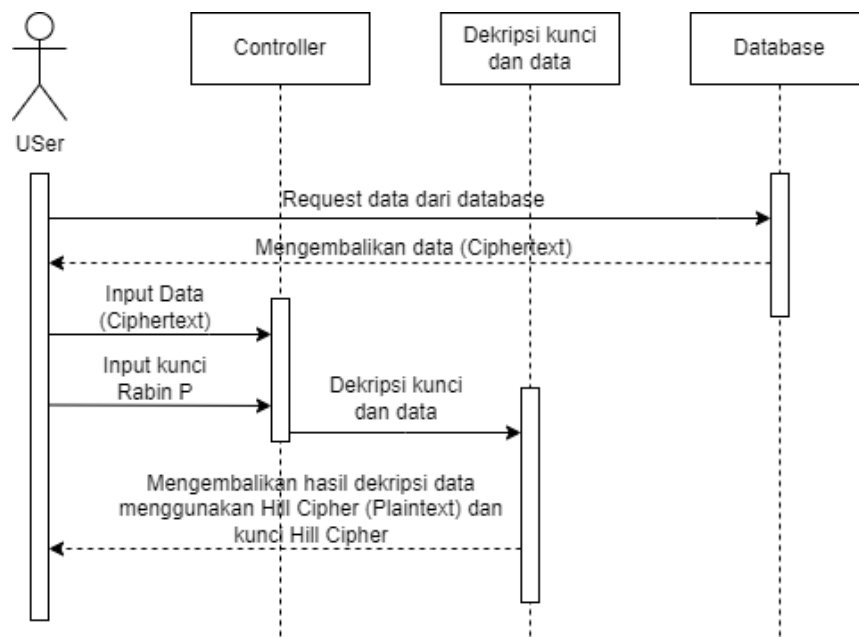
Pada gambar 3.6 bisa dilihat *sequence diagram* yang menunjukkan interaksi antara user dan sistem dengan tahapan enkripsi data.



**Gambar 3.6 Sequence Diagram Enkripsi**

### 2. Sequence Diagram Dekripsi

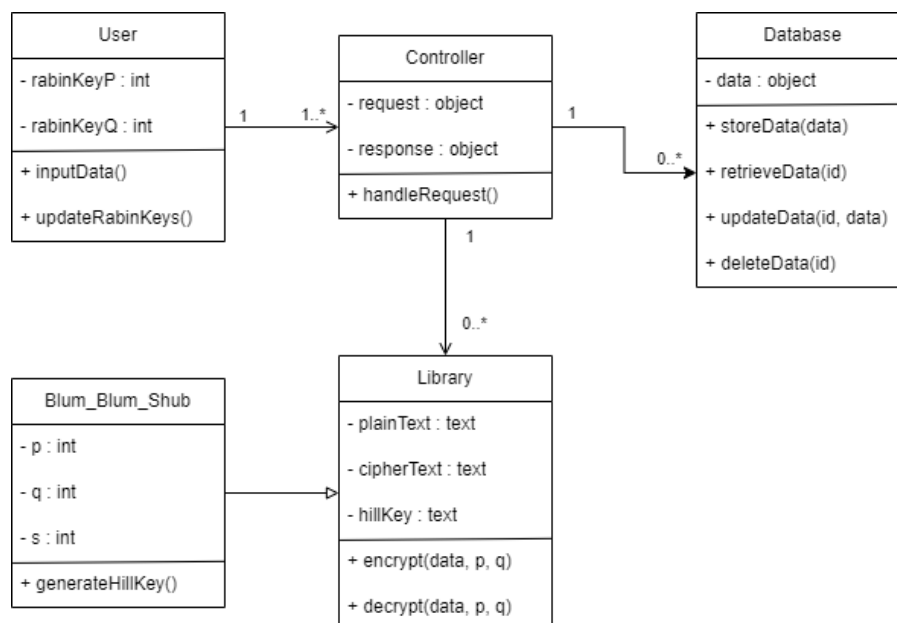
Pada gambar 3.7 dapat dilihat *sequence diagram* yang menunjukkan interaksi antara user dan sistem dengan tahapan dekripsi data.



**Gambar 3.7 Sequence Diagram Dekripsi**

### 3.2.5. Class Diagram

*Class Diagram* adalah gambaran struktur statis dari kelas dalam sistem yang memvisualisasikan atribut, metode, dan hubungan antar kelas (Setiaji, et al., 2021). Pada gambar 3.8 menunjukkan *Class Diagram* pada sistem yang akan dikembangkan didalam penelitian ini.



**Gambar 3.8 Class Diagram**

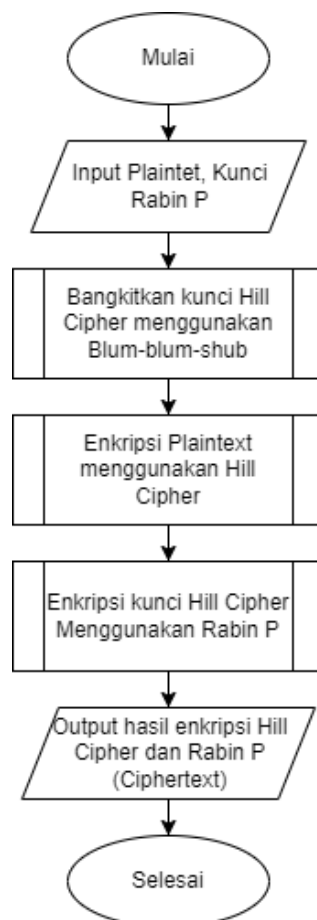


### 3.2.6. Diagram Alir (Flowchart)

Diagram alir (*flowchart*) merupakan gambaran dari urutan langkah-langkah atau proses yang menunjukkan aliran kerja atau operasi dalam suatu sistem. *Flowchart* membantu memvisualisasikan dan memahami proses secara menyeluruh dengan menggunakan simbol-simbol yang mudah untuk dipahami.

#### 1. Flowchart Enkripsi

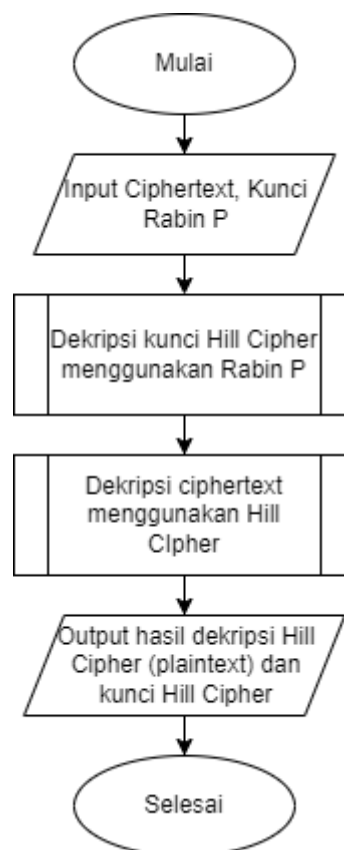
Pada gambar 3.9 menunjukkan proses enkripsi data dimulai dengan user menginput data (*plaintext*) dan kunci algoritma Rabin P. Kemudian kunci algoritma *Hill Cipher* dibangkitkan dengan pembangkit bilangan acak *Blum-Blum-Shub* dan kunci tersebut dipakai sebagai kunci enkripsi data (*plaintext*) memakai algoritma *Hill Cipher*. Setelah itu, kunci dari algoritma *Hill Cipher* di enkripsi memakai algoritma *Rabin P* sehingga menghasilkan data dan kunci yang telah di enkripsi (*ciphertext*).



**Gambar 3.9 Flowchart Enkripsi**

## 2. Flowchart Dekripsi

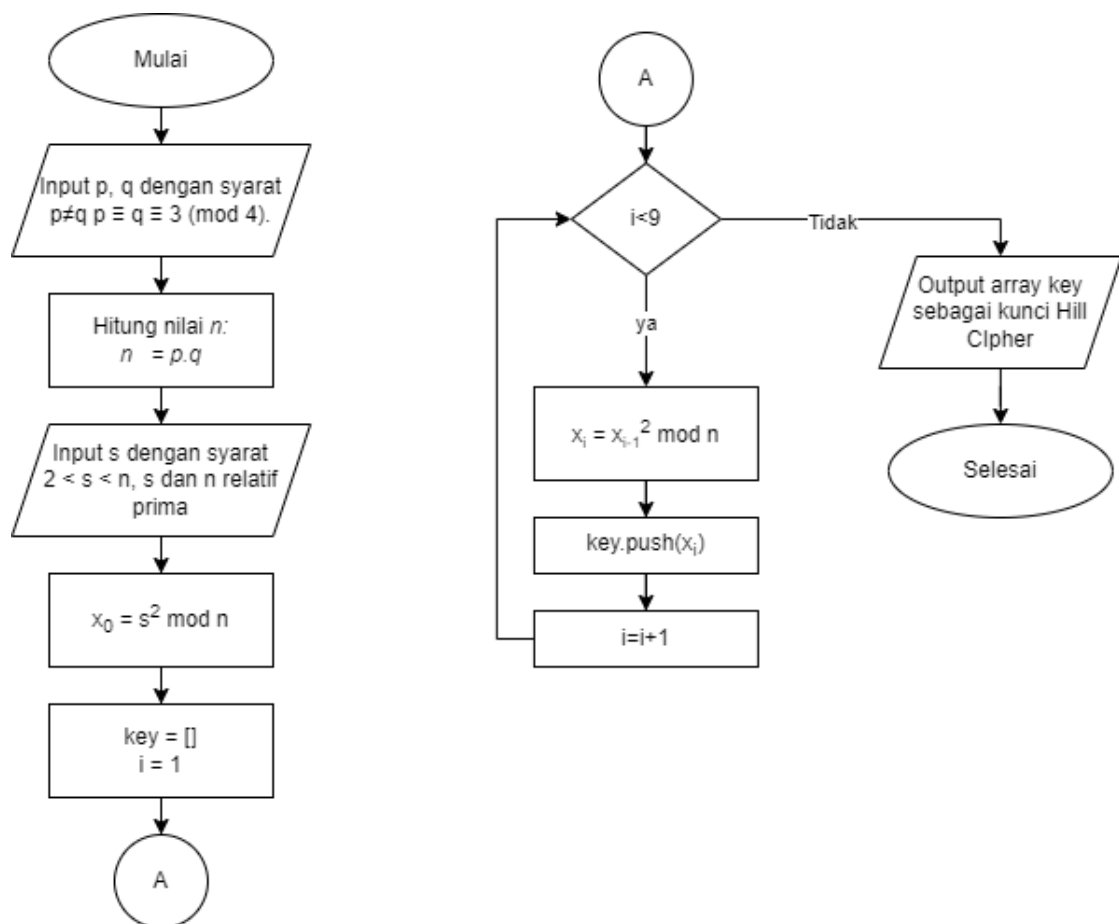
Pada gambar 3.10 menunjukkan proses dekripsi data (*ciphertext*) dimulai dengan user menginputkan data (*ciphertext*) dan kunci algoritma *Rabin P*. Kemudian kunci algoritma *Hill Cipher* didekripsi menggunakan algoritma *Rabin P*. Kunci yang telah di dekripsi digunakan untuk mendekripsi data (*ciphertext*) memakai algoritma *Hill Cipher* sehingga mengembalikan data semula (*plaintext*).



**Gambar 3.10 Flowchart Dekripsi**

### 3. Flowchart Proses Pembangkitan Bilangan Acak Blum-Blum-Shub

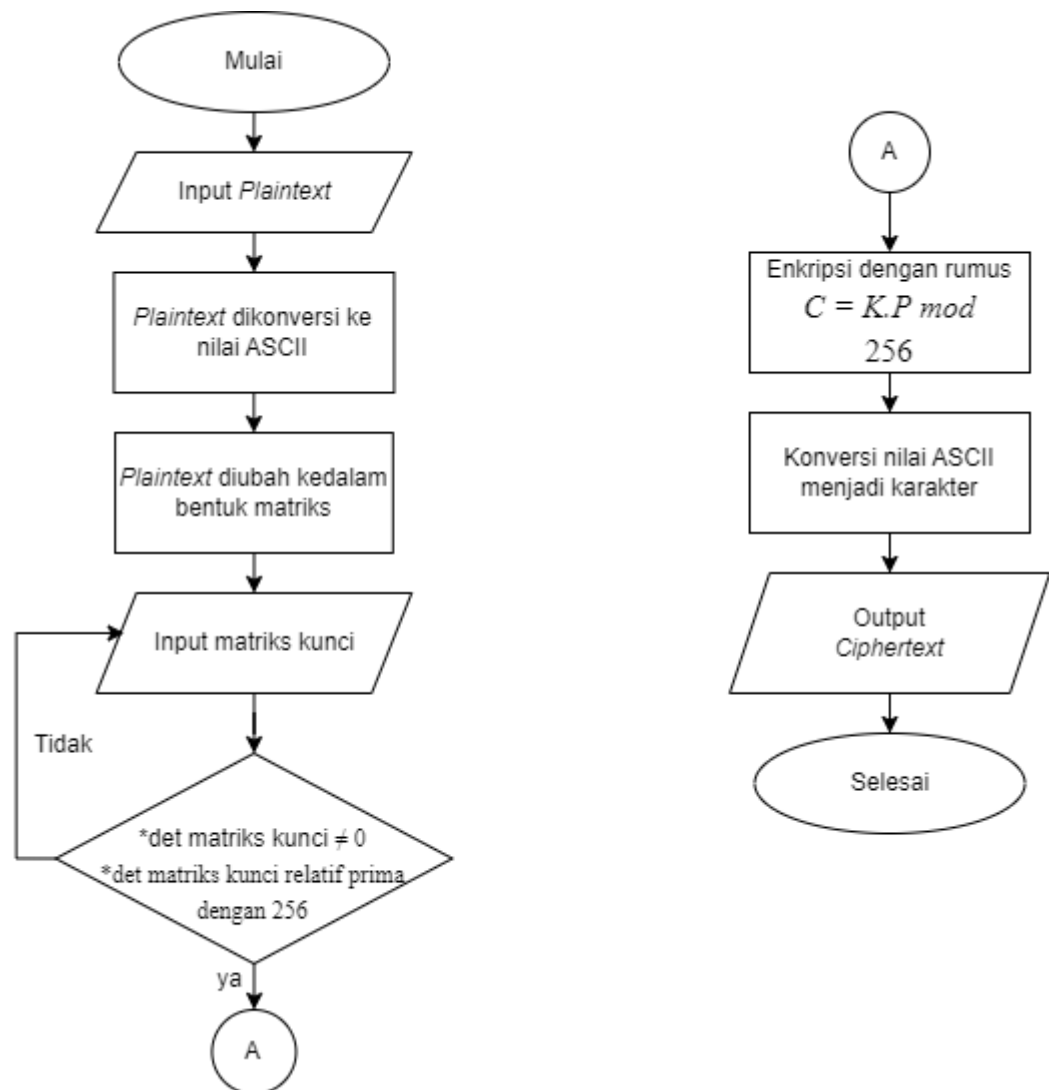
Pada gambar 3.11 menunjukkan proses pembangkitan bilangan acak Blum-Blum-Shub yang dimulai dengan input kunci  $p$  dan  $q$  kemudian menghitung nilai  $n$  dan input bilangan acak  $s$ . Kemudian menghitung inisial nilai dengan rumus  $x_0 = s^2 \bmod n$  dan membuat variabel array  $key$  untuk menampung bilangan acak. Selanjutnya melakukan for loop sebanyak sembilan kali untuk mendapatkan bilangan acak dengan rumus  $x_i = x_{i-1}^2 \bmod n$  sehingga menghasilkan sembilan bilangan acak yang digunakan sebagai kunci algoritma *Hill Cipher*.



Gambar 3.11 Flowchart Proses Pembangkitan Bilangan Acak Blum-Blum-Shub

#### 4. Flowchart Proses Enkripsi Hill Cipher

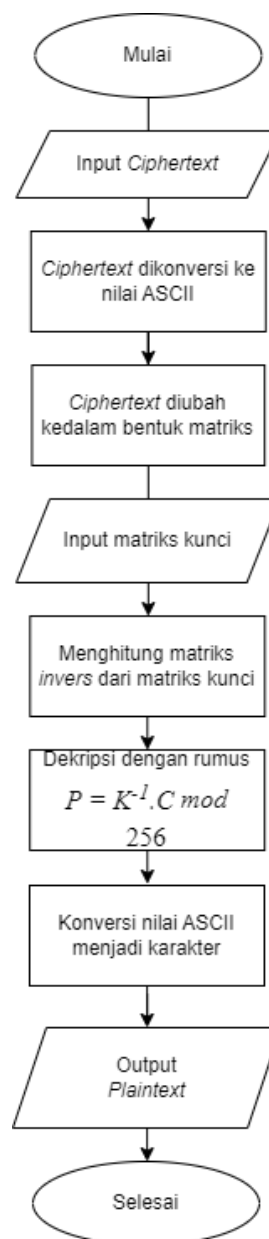
Pada gambar 3.12 menunjukkan proses enkripsi algoritma *Hill Cipher* dimulai dengan memasukan *plaintext* kemudian *plaintext* dikonversi ke nilai kode ASCII dan diubah kedalam bentuk matriks. Selanjutnya adalah memasukan matriks kunci. Jika matriks kunci yang digunakan memiliki determinan sama dengan 0 atau tidak relatif prima dengan 256 maka masukan ulang kunci yang berbeda, jika tidak maka *plaintext* dienkripsi dengan mengalikan matriks kunci dengan matriks *plaintext* dengan modulo 256. Hasil dari perkalian matriks kemudian di konversi dari nilai kode ASCII menjadi karakter dan disusun menghasilkan *chiphertext*.



Gambar 3.12 Flowchart Enkripsi Hill Cipher

## 5. Flowchart Proses Dekripsi Hill Cipher

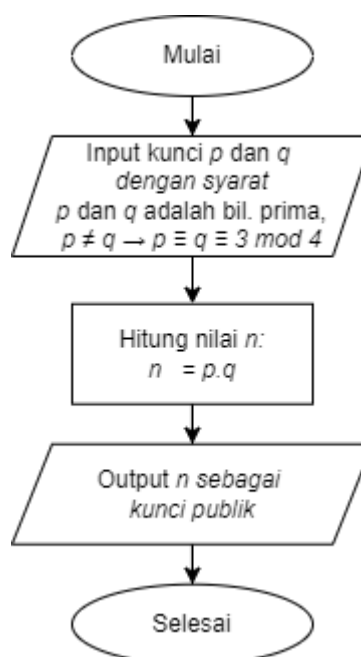
Pada gambar 3.11 menunjukkan proses dekripsi algoritma *Hill Cipher* dimulai dengan memasukan *ciphertext* kemudian dikonversi ke nilai kode ASCII dan diubah kedalam bentuk matriks. Selanjutnya adalah memasukan matriks kunci. Matriks kunci di inverskan kemudian *ciphertext* didekripsi dengan mengalikan matriks balikan dari kunci dengan matriks *ciphertext* dengan modulo 256. Hasil dari perkalian matriks kemudian di konversi dari nilai kode ASCII menjadi karakter dan disusun menghasilkan *plaintext*.



Gambar 3.13 Flowchart Dekripsi Hill Cipher

## 6. Flowchart Pembangkit Kunci Rabin P

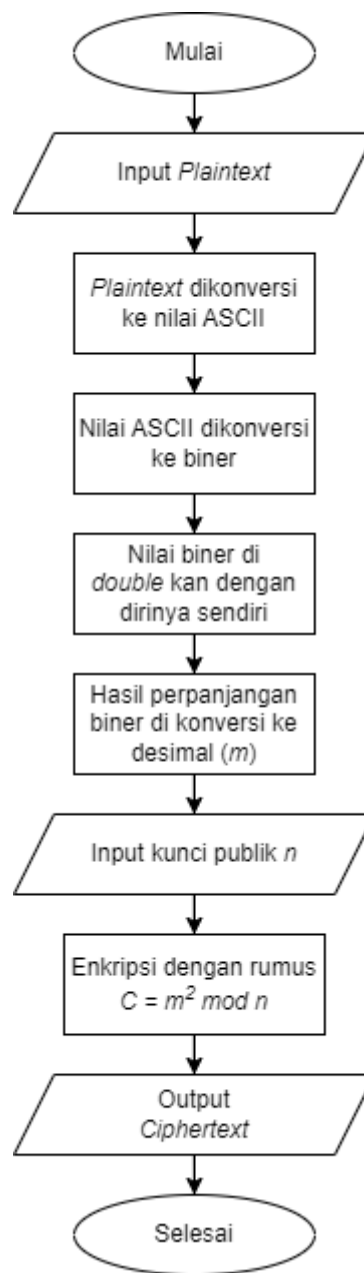
Pada gambar 3.12 menunjukkan proses untuk membangkitkan kunci algoritma *Rabin p* yang dimulai dengan input kunci privat  $p$  dan  $q$ . Jika kunci privat  $p$  dan  $q$  bukan bilangan prima dan tidak ekuivalen dengan  $3 \bmod 4$  maka input kembali  $p$  dan  $q$  yang berbeda, Jika Tidak maka hitung nilai kunci publik  $n$  dengan mengalikan  $p$  dan  $q$  sehingga menghasilkan kunci publik  $n$ .



**Gambar 3.14 Flowchart Pembangkit Kunci Rabin P**

## 7. Flowchart Proses Enkripsi Rabin P

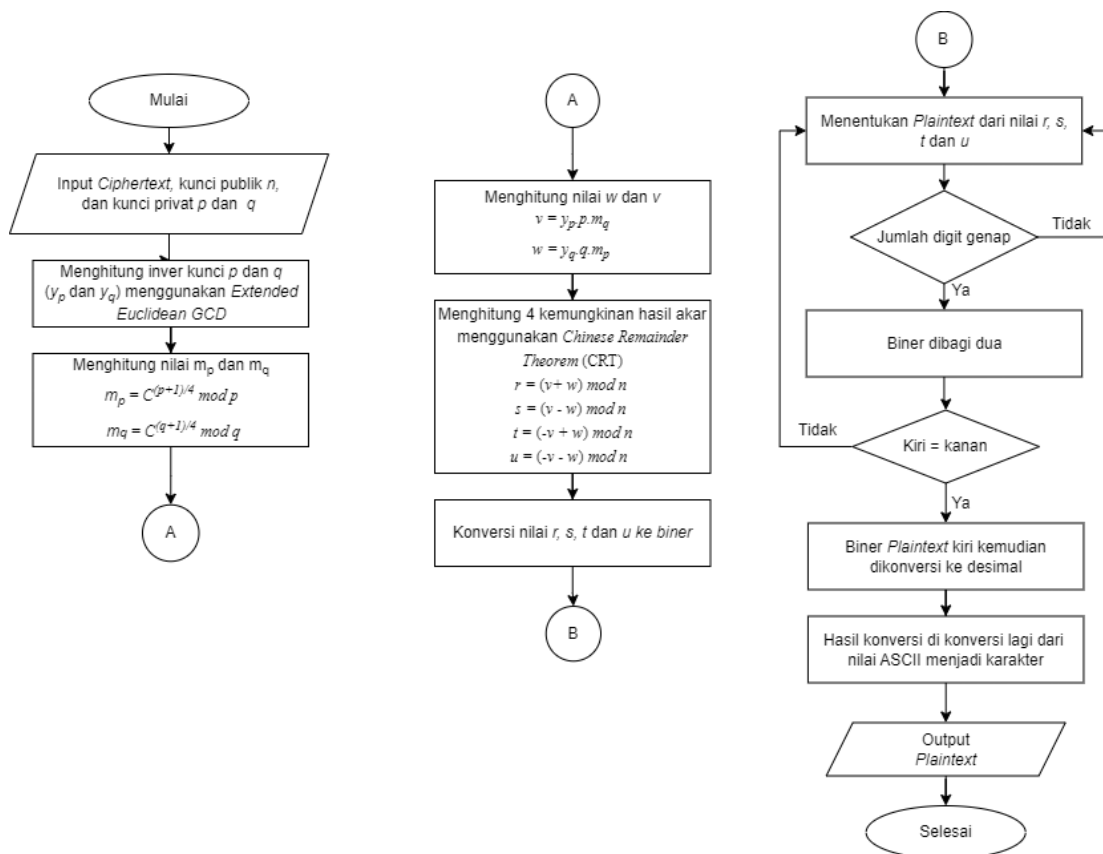
Pada gambar 3.13 menunjukkan proses enkripsi Rabin  $p$  yang dimulai dengan input plaintext dan kunci publik  $n$ . Plaintext kemudian di konversi ke nilai kode ASCII dan dikonversi lagi dari kode ASCII ke biner. Hasil konversi ke biner kemudian di panjangkan dengan dirinya sendiri. Hasil perpanjangan kemudian dikonversi ke desimal kemudian di enkripsi dengan menggunakan rumus  $C = m^2 \bmod n$  yang menghasilkan ciphertext.



**Gambar 3.15 Flowchart Enkripsi Rabin P**

## 8. Flowchart Proses Dekripsi Rabin P

Pada gambar 3.14 menunjukkan proses dekripsi algoritma *Rabin p* yang dimulai dengan input *ciphertex*, kunci publik  $n$ , dan kunci privat  $p$  dan  $q$ . Kemudian invers dari kunci privat  $p$  dan  $q$  dihitung menggunakan *Extended Euclidean GCD* setelah itu menghitung nilai  $m_p$  dan  $m_q$  yang dipakai untuk mendapatkan hasil akar kuadrat dari pesan menggunakan *Chinese Remainder Theorem*. Setelah mendapatkan 4 hasil akar maka akan ditentukan *plaintext* yang asli dari ke empat akar tersebut dengan cara mengubahnya ke biner kemudian melihat apakah jumlahnya genap, jika genap maka biner akan dibagi dua jadi biner kiri dan biner kanan untuk membandingkan apakah keduanya sama, jika sama maka kita mendapatkan biner *plaintext*nya. Setelah mendapatkan *plaintext* kemudian *plaintext* di konversi ke desimal kemudian di konversi lagi dari kode ASCII menjadi karakter sehingga mengembalikan pesan *plaintext*.

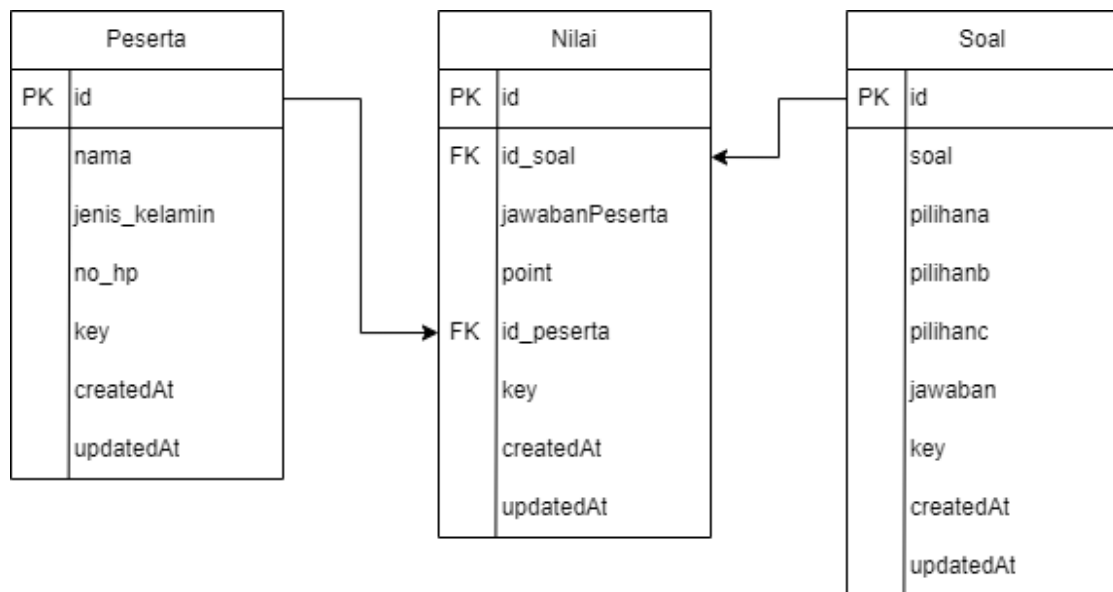


Gambar 3.16 Flowchart Dekripsi Rabin P



### 3.2.7. Rancangan Logical Record Structure (LRS) Database

*Logical Record Structure* (LRS) ialah gambaran dari struktur pada tabel-tabel yang terbentuk dari hubungan antara kumpulan entitas (Nurmalasari, et al., 2019). Pada gambar 3.15 merupakan rancangan *Logical Record Structure* (LRS) *database* yang akan dipakai dalam pengujian sistem yang akan dibuat.



**Gambar 3.17 Rancangan LRS Database**

Pada gambar 3.15 menunjukkan *database* dengan jenis *relational database*. Pada *database* ini terdapat 3 tabel yaitu tabel Peserta, tabel Nilai, dan tabel Soal. Tabel Peserta berisikan 7 kolom yaitu id, nama, jenis\_kelamin, no\_hp, key, createAt, updateAt, dimana kolom id sebagai Primary Key tabel Peserta dan kolom key sebagai tempat untuk kunci algoritma Hill Cipher yang telah dienkrpsi. Tabel Soal berisikan 9 kolom yaitu id, soal, pilihanA, pilihanB, pilihanC, jawaban, key, createAt, updateAt, dimana kolom id sebagai Primary Key tabel Soal dan kolom key sebagai tempat untuk kunci algoritma Hill Cipher yang telah dienkrpsi. Tabel Nilai berisikan 8 kolom yaitu id, id\_soal, jawabanPeserta, point, id\_peserta, key, createAt, updateAt, dimana kolom id sebagai Primary Key tabel Nilai, kolom id\_soal sebagai Foreign Key yang merujuk pada kolom id tabel Soal, kolom ide\_peserta sebaga Foreign Key yang merujuk pada kolom id tabel Peserta, dan kolom key sebagai tempat untuk kunci algoritma *Hill Cipher* yang telah dienkrpsi.

## BAB 4

### IMPLEMENTASI DAN PENGUJIAN

#### 4.1. Implementasi Sistem

Setelah seluruh proses analisis dan perancangan sistem diselesaikan, langkah selanjutnya adalah pengimplementasian sistem ke dalam bahasa pemrograman. Sistem yang telah dirancang akan direalisasikan dengan menerjemahkannya ke dalam kode program yang sesuai. Dalam hal ini, sistem akan dibangun menjadi *Library* bahasa pemrograman *Javascript*.

##### 4.1.1. Lingkungan Pengembangan

Lingkungan pengembangan sistem ini terdiri dari:

- b. Bahasa pemrograman: *Javascript*
- c. Platform: Node.js
- d. Database: PostgreSQL
- e. Perangkat lunak pendukung: Visual Studio Code, Postman

##### 4.1.2. Struktur Library

Struktur file dalam sistem (*library*) ini adalah sebagai berikut:

```
/encryption-library
|-- /src
|   |-- /algorithms
|   |   |-- hillCipher.js
|   |   |-- rabinP.js
|   |-- /utils
|   |   |-- blumBlumShub.js
|   |   |-- chineseRemainderTheorem.js
|   |   |-- ectendedEuclidean.js
|   |   |-- operationMatrix.js
|   |   |-- keyValidator.js
|   |-- index.js
|-- package.json
|-- README.md
```

**Gambar 4.1 Struktur *Library***

Pada gambar 4.1 menunjukkan struktur library Dimana dalam folder *encryption-library* terdapat dua buah folder dan dua buah file.

a. Folder src

Dalam folder src terdapat folder lagi yaitu:

f. Folder algorithms

Berisikan 2 buah file yaitu hillCipher.js yang memiliki dua buah fungsi yaitu fungsi enkripsi dan fungsi dekripsi data memakai algoritma *Hill Cipher*, dan file rabinP.js yang memiliki dua buah fungsi yakni fungsi enkripsi dan fungsi dekripsi kunci *Hill Cipher* menggunakan algoritma *Rabin P*.

g. Foder utils

Berisikan file-file pendukung seperti pembangkit bilangan acak dalam file blumBlumShub.js, chineseRemainderTheorem.js, extendedEuclidean.js, operationMatrix.js, dan keyValidator.js

h. File index.js

Merupakan file utama dari library

b. File package.json

c. README.md

Berisikan panduan penggunaan library

### 4.1.3. Implementasi Fungsi Utama

#### 1. Enkripsi Data

Pada gambar 4.2 menunjukkan fungsi *encrypt* untuk melakukan proses enkripsi pada data (*datas*) menggunakan algoritma *Hybrid Cryptosystem* dengan melibatkan algoritma *Blum-Blum-Shub* (bbs), *Hill Cipher*, dan algoritma *Rabin P*.

```
const generateKeyBBS = require("./utils/blumBlumShub");
const {
  hillCipherEncrypt,
  hillCipherDecrypt,
} = require("./algorithms/hillCipher");
const { rabinPEncrypt, rabinPDecrypt } = require("./algorithms/rabinP");

function encrypt(datas, p, q) {
  const startTime = performance.now();
  const key = generateKeyBBS();

  function splitData(datas) {
    for (const data in datas) {
      if (typeof datas[data] === "object") {
        splitData(datas[data]);
      } else {
        datas[data] = String(datas[data]);
        datas[data] = hillCipherEncrypt(datas[data], key);
      }
    }
  }
  splitData(datas);

  let ciphertext = rabinPEncrypt(key, p, q);
  datas.key = ciphertext.join("/");

  const endTime = performance.now();
  const executionTime3 = endTime - startTime;

  console.log(`Execution time rabin: ${executionTime3} ms`);

  return datas;
}
```

**Gambar 4.2 Fungsi Enkripsi Data**

Parameter fungsi

- **datas**: Data yang akan dienkripsi. Data ini bisa berupa objek yang kompleks.
- **p** dan **q**: Bilangan prima yang digunakan dalam algoritma bbs dan *Rabin P*.

Langkah-langkah Fungsi

#### 1. Pengukuran waktu mulai eksekusi:

```
a. const startTime = performance.now();
```

- b. Mengambil waktu saat fungsi mulai dieksekusi untuk mengukur waktu eksekusi fungsi
2. Pebangkitan kunci:
  - a. `let key = generateKeyBBS();`
  - b. Membuat kunci memakai fungsi algoritma Blum-Blum-Shub.
3. Fungsi Rekursif untuk enkripsi data:
  - a. `function splitData = (datas) => { ... };`
  - b. Mendefinisikan fungsi `splitData` untuk memproses dan mengenkripsi setiap elemen dalam objek `datas`.
4. Enkripsi data:
  - a. `splitData(datas);`
  - b. Memanggil fungsi `splitData` untuk mengenkripsi seluruh elemen dalam objek `datas`.
    - `for (const data in datas) { ... };` Melakukan iterasi pada setiap elemen dalam objek `datas`.
    - `if (typeof datas[data] == "object") { splitData(datas[data]); }` Jika elemen adalah objek, panggil `splitData` secara rekursif.
    - `else { datas[data] = String(datas[data]); datas[data] = hillCipherEncrypt(datas[data], key); }` Jika elemen bukan objek, ubah menjadi string dan enkripsi menggunakan algoritma *Hill Cipher* dengan kunci `key`.
5. Enkripsi kunci *Hill Cipher*:
  - a. `let ciphertextKey = rabinPEncrypt(key, p, q);`
  - b. Mengenkripsi kunci *Hill Cipher* menggunakan algoritma *Rabin P* dengan parameter `p` dan `q`.
6. Penyimpanan kunci terenkripsi:
  - a. `datas.key = ciphertextKey.join();`
  - b. Menyimpan kunci *Hill Cipher* yang telah dienkripsi dalam properti `key` dari objek `datas`.
7. Pengukuran waktu selesai eksekusi:
  - a. `const endTime = performance.now();`
  - b. Mengambil waktu saat fungsi selesai dieksekusi.

## 8. Perhitungan dan percetakan waktu eksekusi:

- a. `const executionTime = endTime - startTime;`
- b. Menghitung waktu eksekusi fungsi.
- c. `console.log(Execution time: ${executionTime} ms) ;`
- d. Mencetak waktu eksekusi fungsi.

## 9. Pengembalian data yang terenkripsi:

- a. `return datas;`
- b. Mengembalikan objek `datas` yang telah dienkripsi.

## 2. Dekripsi Data

Pada gambar 4.3 menunjukkan fungsi `encrypt` untuk melakukan proses enkripsi pada data (`datas`) menggunakan algoritma hybrid cryptosystem dengan melibatkan algoritma *Blum-Blum-Shub* (bbs), *Hill Cipher*, dan algoritma *Rabin P*.

```
const generateKeyBBS = require("./utils/blumBlumShub");
const {
  hillCipherEncrypt,
  hillCipherDecrypt,
} = require("./algorithms/hillCipher");
const { rabinPEncrypt, rabinPDecrypt } = require("./algorithms/rabinP");

function decrypt(datas, p, q) {
  const startTime = performance.now();
  let datasKey = datas.key.split("/");

  let key = rabinPDecrypt(datasKey, p, q);

  function splitData(datas) {
    for (const data in datas) {
      if (data === "key") {
        continue;
      }
      if (typeof datas[data] === "object") {
        splitData(datas[data]);
      } else {
        datas[data] = hillCipherDecrypt(datas[data], key);
      }
    }
  }
  splitData(datas);

  const endTime = performance.now();
  const executionTime2 = endTime - startTime;
  console.log(`Execution time2: ${executionTime2} ms`);

  return datas;
}
```

**Gambar 4.3 Fungsi Dekripsi Data**

### Parameter fungsi

- **datas**: Data yang akan dienkripsi. Data ini bisa berupa objek yang kompleks.
- **p** dan **q**: Bilangan prima yang digunakan dalam algoritma bbs dan *Rabin P*.

### Langkah-langkah Fungsi

#### 1. Pengukuran waktu mulai eksekusi:

- a. `const startTime = performance.now();`
- b. Mengambil waktu saat fungsi mulai dieksekusi untuk mengukur waktu eksekusi fungsi

#### 2. Ekstraksi dan dekripsi kunci terenkripsi:

- a. `let datasKey = datas.key.split("/");`
- b. Mengambil kunci *Hill Cipher* yang terenkripsi dari properti `key` dari objek `ciphertext` dan membaginya menjadi array.
- c. `let key = rabinPDecrypt(ciphertextKey, p, q);`
- a. Mendekripsi kunci *Hill Cipher* menggunakan algoritma *Rabin P* dengan parameter `p` dan `q`.

#### 3. Fungsi Rekursif untuk enkripsi data:

- a. `const splitData = (datas) => { ... };`
- b. Mendefinisikan fungsi `splitData` untuk memproses dan mengenkripsi setiap elemen dalam objek `datas`.

#### 4. Dekripsi data:

- a. `splitData(datas);`
- b. Memanggil fungsi `splitData` untuk mendekripsi seluruh elemen dalam objek `datas`.
  - `for (const data in datas) { ... };` Melakukan iterasi pada setiap elemen dalam objek `datas`.
  - `if (data === "id") { continue; };` Jika elemen adalah properti `id`, lewati elemen tersebut karena itu adalah kunci terenkripsi.
  - `if (typeof ciphertext[data] == "object") { splitData(ciphertext[data]); };` Jika elemen adalah objek, panggil `splitData` secara rekursif.
  - `else { ciphertext[data] = hillCipherDecrypt(ciphertext[data], key); };` Jika elemen

bukan objek, dekripsi elemen tersebut menggunakan algoritma *Hill Cipher* dengan kunci *key*.

5. Pengukuran waktu selesai eksekusi:
  - c. `const endTime = performance.now();`
  - d. Mengambil waktu saat fungsi selesai dieksekusi.
6. Perhitungan dan percetakan waktu eksekusi:
  - a. `const executionTime = endTime - startTime;`
  - b. Menghitung waktu eksekusi fungsi.
  - c. `console.log (Execution time: ${executionTime} ms) ;`
  - d. Mencetak waktu eksekusi fungsi.
7. Pengembalian data yang terenkripsi:
  - c. `return datas;`
  - d. Mengembalikan objek `datas` yang telah didekripsi.

#### 4.1.4. Dokumentasi Penggunaan Library

Dokumentasi penggunaan library dapat dilihat pada lampiran A.

#### 4.1.5. Perhitungan Manual

Pada tahap ini akan dilakukan perhitungan secara manual di mana data (*plaintext*) yang akan digunakan adalah “Kunci enkripsi yang sama dengan proses dekripsi disebut?”. Kunci algoritma Rabin P yang digunakan adalah  $p = 925789379$ ,  $q = 904318643$ . Langkah-langkah perhitungannya dapat dilihat sebagai berikut.

1. Pembangkit Bilangan Acak *Blum-Blum Shub*
  - a. Tentukan dua bilang prima sebagai kunci,  $p$  dan  $q$  di mana  $p \equiv q \equiv 3 \pmod{4}$ .  

$$p = 547, q = 571$$
  - b. Kemudian kalikan menjadi  $n = pq$ .  

$$n = pq = 312337$$
  - c. Tentukan bilangan bulat acak yang lain,  $s$ , yang menjadi umpan sehingga:  

$$2 \leq s < n$$

$$s \text{ dan } n \text{ relatif prima}$$

$$s = 17$$



d. hitung  $x_0 = s^2 \bmod n$

$$x_0 = 17^2 \bmod 312337 = 289$$

e. Barisan bilangan acak akan dihasilkan dari pengulangan perhitungan berikut sepanjang yang diperlukan:

- Hitung  $x_i = x_{i-1}^2 \bmod n$

$$x_1 = x_0^2 \bmod n = 289^2 \bmod 312337 = 83521 \bmod 256 = 65$$

$$x_2 = x_1^2 \bmod n = 83521^2 \bmod 312337 = 22883 \bmod 256 = 99$$

$$x_3 = x_2^2 \bmod n = 22883^2 \bmod 312337 = 154877 \bmod 256 = 253$$

$$x_4 = x_3^2 \bmod n = 154877^2 \bmod 312337 = 28203 \bmod 256 = 43$$

$$x_5 = x_4^2 \bmod n = 28203^2 \bmod 312337 = 199207 \bmod 256 = 39$$

$$x_6 = x_5^2 \bmod n = 199207^2 \bmod 312337 = 75988 \bmod 256 = 212$$

$$x_7 = x_6^2 \bmod n = 75988^2 \bmod 312337 = 2025 \bmod 256 = 233$$

$$x_8 = x_7^2 \bmod n = 2025^2 \bmod 312337 = 40244 \bmod 256 = 52$$

$$x_9 = x_8^2 \bmod n = 40244^2 \bmod 312337 = 112191 \bmod 256 = 63$$

Barisan bilangan acak yang dibangkitkan adalah 65, 99, 253, 43, 39, 212, 233, 52, 63. Digunakan sebagai kunci algoritma *Hill Cipher*.

1. Enkripsi data menggunakan *Hill Cipher*

a. Menentukan *plaintext*

P = 'Kunci enkripsi yang sama dengan proses dekripsi disebut?'

Konversi ke nilai ASCII kemudian ubah menjadi bentuk matriks dengan panjang kolom sama dengan panjang baris matriks kunci, jika kolom tidak terpenuhi maka ditambahkan dengan kode ASCII "NUL = 0"

$$P = \begin{bmatrix} 75 & 99 & 101 & 114 & 115 & 121 & 103 & 97 & 32 & 110 & 110 & 114 & 101 & 100 & 114 & 115 & 100 & 101 & 116 \\ 117 & 105 & 110 & 105 & 105 & 97 & 32 & 109 & 100 & 103 & 32 & 111 & 115 & 101 & 105 & 105 & 105 & 98 & 63 \\ 110 & 32 & 107 & 112 & 32 & 110 & 115 & 97 & 101 & 97 & 112 & 115 & 32 & 107 & 112 & 32 & 115 & 117 & 0 \end{bmatrix}$$

b. Menentukan matriks kunci 3x3 yang digunakan

$$K = \begin{bmatrix} 65 & 43 & 233 \\ 99 & 39 & 52 \\ 253 & 212 & 63 \end{bmatrix}$$

- c. Melakukan proses enkripsi dengan rumus  $C = K.P \bmod 256$

Keterangan:

C = Ciphertext

K = Matriks kunci

P = Plaintext

$$C = \begin{bmatrix} 65 & 43 & 233 \\ 99 & 39 & 52 \\ 253 & 212 & 63 \end{bmatrix} \times P \bmod 256$$

$$C = \begin{bmatrix} 208 & 230 & 130 & 133 & 246 & 34 & 50 & 57 & 217 & 132 & 62 & 66 & 22 & 190 & 133 & 246 & 178 & 152 & 9 \\ 44 & 200 & 141 & 213 & 248 & 234 & 17 & 210 & 32 & 239 & 42 & 91 & 20 & 203 & 213 & 248 & 7 & 193 & 117 \\ 21 & 171 & 62 & 46 & 123 & 251 & 152 & 0 & 75 & 225 & 198 & 227 & 237 & 205 & 46 & 123 & 21 & 196 & 208 \end{bmatrix}$$

C =

'208/44/21/230/200/171/130/141/62/133/213/46/246/248/123/34/234/251/50/1  
7/152/57/210/0/217/32/75/132/239/225/62/42/198/66/91/227/22/20/237/190/20  
3/205/133/213/46/246/248/123/178/7/21/152/193/196/9/117/208'

2. Enkripsi kunci *Hill Cipher* menggunakan *Rabin P*

- a. Dapatkan kunci p,q, dan kunci public n

$$p = 925789379$$

$$q = 904318643$$

$$n = 837208594921092697$$

- b. Membagi kunci Hill Cipher menjadi 3 bagian [65,99,253], [43,39,212], [233,52,63]. Kemudian enkripsi setiap bagiannya, disini akan dienkripsi bagian pertama [65,99,253]. Tiap bilangan akan dikonversi ke bentuk biner berukuran 8 bit kemudian digabungkan dan diperpanjang dan dikonversi kembali ke bentuk decimal.

$$m = 65_{10} = 01000001_2$$

$$m = 99_{10} = 01100011_2$$

$$m = 253_{10} = 11111101_2$$

$$m = 010000010110001111111101 | 010000010110001111111101 \rightarrow$$

panjangkan

$$m = 71897706488829_{10}$$

- c. Enkripsi dengan rumus

$$C = m^2 \bmod n$$

$$C = 71897706488829^2 \bmod 837208594921092697$$

$$C = 680140331240547234$$

*Ciphertext* lengkap untuk kunci algoritma *Hill Cipher* adalah

‘680140331240547234/231423279539691610/81131374005234414’

3. Dekripsi kunci *Hill Cipher* memakai algoritma *Rabin P*

- a. Dekripsi kunci *Hill Cipher* yang telah terenkripsi.

$$C = 680140331240547234/231423279539691610/81131374005234414$$

Kunci dibagi menjadi 3 untuk masing-masing didekripsi. Disini akan didekripsi bagian pertama yaitu:

$$C = 680140331240547234$$

- b. Cari nilai  $y_p$  dan  $y_q$  dengan Extended Euclidean GCD

$$y_p \cdot p + y_q \cdot q = \text{GCD}(p, q)$$

$$y_p \cdot p + y_q \cdot q = 1$$

$$925789379 \cdot y_p + 904318643 \cdot y_q = 1$$

Temukan nilai  $y_p$  dan  $y_q$  menggunakan Extended Euclidean:

**Tabel 4.1 Extended Euclidean**

$y_p$	$y_q$	<b>D</b>	<b>k</b>
0	1	925789379	
1	0	904318643	1
-1	1	21470736	42
-438471894	448882289	1	

$$y_p = -438471894 \text{ dan } y_q = 448882289$$

- c. Hitung nilai  $m_p$  dan  $m_q$

$$m_p = C^{\frac{p+1}{4}} \bmod p$$

$$m_p = 680140331240547234^{\frac{925789379+1}{4}} \bmod 925789379$$

$$m_p = 290253715$$

$$m_q = C^{\frac{q+1}{4}} \bmod q$$

$$m_p = 680140331240547234^{\frac{904318643+1}{4}} \bmod 904318643$$

$$m_p = 111782851$$

d. Hitunglah nilai dari  $r$ ,  $s$ ,  $t$  dan  $u$  memakai *Chinese Remainder Theorem* (CRT).

- Hitunglah nilai variabel  $v$  dan  $w$  terlebih dahulu untuk memudahkan perhitungan.

$$v = y_p \cdot p \cdot m_q = -45376305851950421284897926$$

$$w = y_q \cdot q \cdot m_p = -8.191.100 = 117823451707318234406117305$$

- Hitunglah nilai dari  $r$ ,  $s$ ,  $t$  dan  $u$

$$r = (v + w) \bmod n = 256410619819071$$

$$s = (v - w) \bmod n = 451081045521659442$$

$$t = (-v + w) \bmod n = 386127549399433255$$

$$u = (-v - w) \bmod n = 836952184301273626$$

e. Konversi  $r$ ,  $s$ ,  $t$  dan  $u$  ke biner

$$r = 837136697214603868_{10} =$$

$$101110011110000110110110010000001001111010111001001001011100_2$$

$$s = 561252896275404559_{10} =$$

$$11111001001111110001000100100010111101001111001011100001111_2$$

$$t = 275955698645688138_{10} =$$

$$1111010100011001000011111011101111100001010101111101001010_2$$

$$u = 71897706488829_{10} =$$

$$01000001011000111111101010000010110001111111101_2$$

f. Tentukan plaintext dari nilai  $r$ ,  $s$ ,  $t$  dan  $u$

$$r = 837136697214603868_{10} =$$

$$101110011110000110110110010000|001001111010111001001001011100_2$$

(Kiri  $\neq$  kanan)

$$s = 561252896275404559_{10} =$$

$$11111001001111110001000100100010111101001111001011100001111_2$$

(Jumlah digit ganjil)

$$t = 275955698645688138_{10} =$$

$$11110101000110010000111110111|01111100001010101111101001010_2$$

(Kiri  $\neq$  kanan)

$$u = 7189770648882910 =$$

$$01000001011000111111101010000|010110001111111012 \text{ (Kiri = kanan)}$$

Pesan ada dalam  $u$

$$m = 01000001011000111111101_2$$

Pesan dibagi menjadi 3 bagian 8 bit dan dikonversi ke bilangan desimal

$$m_1 = 01000001_2 = 65_{10}$$

$$m_2 = 01100011_2 = 99_{10}$$

$$m_3 = 11111101_2 = 253_{10}$$

Kunci *Hill Cipher* setelah didekripsi adalah [65,99,253], [43,39,212],  
[233,52,63].

#### 4. Dekripsi data menggunakan *Hill Cipher*

- a. Menghitung matriks balikan (*invers*) dari matriks kunci dengan mod 256

$$K = \begin{bmatrix} 65 & 43 & 233 \\ 99 & 39 & 52 \\ 253 & 212 & 63 \end{bmatrix}$$

Perhitungan matriks balikan dapat dilakukan dengan rumus berikut:

$$K = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}, K^{-1} = \frac{1}{\det(K)} \begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \end{bmatrix}^T$$

$$\det(K) = 207$$

$$\frac{1}{\det(K)} = \frac{1}{235} = 207^{-1} \text{ mod } 256 = 47$$

Maka nilai matriks balikan dari matriks kunci adalah

$$K^{-1} = 47 \begin{bmatrix} 65 & 43 & 233 \\ 99 & 39 & 52 \\ 253 & 212 & 63 \end{bmatrix} \text{ mod } 256$$

$$K^{-1} = \begin{bmatrix} 39 & 113 & 51 \\ 73 & 38 & 105 \\ 191 & 101 & 218 \end{bmatrix}$$

b. Matriks data yang akan didekripsi

$$C = \begin{bmatrix} 208 & 230 & 130 & 133 & 246 & 34 & 50 & 57 & 217 & 132 & 62 & 66 & 22 & 190 & 133 & 246 & 178 & 152 & 9 \\ 44 & 200 & 141 & 213 & 248 & 234 & 17 & 210 & 32 & 239 & 42 & 91 & 20 & 203 & 213 & 248 & 7 & 193 & 117 \\ 21 & 171 & 62 & 46 & 123 & 251 & 152 & 0 & 75 & 225 & 198 & 227 & 237 & 205 & 46 & 123 & 21 & 196 & 208 \end{bmatrix}$$

c. Melakukan proses dekripsi dengan rumus  $P = K^{-1} \cdot C \bmod 256$

Keterangan:

P = Plaintext

K = Matriks balikan (*invers*) dari matriks kunci

C = Ciphertext

$$P = \begin{bmatrix} 39 & 113 & 51 \\ 73 & 38 & 105 \\ 191 & 101 & 218 \end{bmatrix} \times C \bmod 256$$

$$P = \begin{bmatrix} 75 & 99 & 101 & 114 & 115 & 121 & 103 & 97 & 32 & 110 & 110 & 114 & 101 & 100 & 114 & 115 & 100 & 101 & 116 \\ 117 & 105 & 110 & 105 & 105 & 97 & 32 & 109 & 100 & 103 & 32 & 111 & 115 & 101 & 105 & 105 & 105 & 98 & 63 \\ 110 & 32 & 107 & 112 & 32 & 110 & 115 & 97 & 101 & 97 & 112 & 115 & 32 & 107 & 112 & 32 & 115 & 117 & 0 \end{bmatrix}$$

Pesan kemudian dikonversi dari kode ASCII menjadi karakter menghasilkan data 'Kunci enkripsi yang sama dengan proses dekripsi disebut?'

## 4.2. Pengujian Sistem

Pengujian sistem bertujuan untuk memastikan bahwa sistem bisa menjalankan proses enkripsi data dan mengembalikannya ke bentuk semula dengan dekripsi data dengan menggunakan algoritma *Hill Cipher*, algoritma *Rabin P*, dan pembangkit bilangan acak *Blum-Blum-Shub*.

### 4.2.1. Install Library

Pada gambar 4.4 menunjukkan bahwa *library* dapat di *install* menggunakan *npm* (*Node Package Manager*) sehingga dapat digunakan oleh user.

```
PS D:\enkripsi dan dekripsi\backend\mvc-boilerplate> npm install hillcipher_rabinp2
added 1 package, and audited 239 packages in 929ms

35 packages are looking for funding
  run `npm fund` for details

3 moderate severity vulnerabilities

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
```

Gambar 4.4 Install Library

#### 4.2.2. Import Fungsi

pada gambar 4.5 menunjukkan proses import fungsi *encrypt* dan *decrypt* dari *library* yang digunakan untuk mengenkripsi dan mendekripsi data.

```
const { encryp, decrypt } = require("hillcipher_rabinp");
```

**Gambar 4.5 Pemanggilan Fungsi Enkripsi dan Dekripsi dari *Library***

#### 4.2.3. Penyimpanan dan Pemanggilan Kunci *Rabin P*

Kunci algoritma *Rabin p* dapat user simpan di file .env dengan nama variabel bebas dari user. Pada gambar 4.7 merupakan contoh isi file .env

```
KEY_RABINP_P = 925789379
KEY_RABINP_Q = 904318643
```

**Gambar 4.6 File '.env'**

```
require("dotenv").config();
const {
  KEY_RABINP_P = process.env.KEY_RABINP_P,
  KEY_RABINP_Q = process.env.KEY_RABINP_Q,
} = process.env;
```

**Gambar 4.7 Pemanggilan Variabel Kunci *Rabin P***

Pada gambar 4.7 menunjukkan pemanggilan untuk variabel kunci *Rabin p* (kunci privat p dan q) dari file .env.

#### 4.2.4. Create Data (Enkripsi)

Pada gambar 4.8 menunjukkan fungsi create yang digunakan untuk menangani permintaan HTTP POST pada server. Fungsi ini menerima data dari request POST, mengenkripsi data menggunakan fungsi enkripsi dari *library*, dan kemudian menyimpannya dalam database.

```

create(req, res) {
  data = {
    soal: req.body.soal,
    pilihana: req.body.pilihana,
    pilihanb: req.body.pilihanb,
    pilihanc: req.body.pilihanc,
    jawaban: req.body.jawaban,
  };
  encrypData = encryp(data, KEY_RABINP_P, KEY_RABINP_Q);
  console.log(encrypData);

  Soal.create(encrypData)
    .then((post) => {
      res.status(201).json({
        status: "OK",
        data: post,
      });
    })
    .catch((err) => {
      res.status(201).json({
        status: "FAIL",
        message: err.message,
      });
    });
},

```

**Gambar 4.8 Fungsi Create**

```

{
  "soal": "Kunci enkripsi yang sama dengan proses dekripsi disebut?",
  "pilihana": "password",
  "pilihanb": "simetri",
  "pilihanc": "asimetri",
  "jawaban": "b"
}

```

**Gambar 4.9 Contoh data soal**

Pada gambar 4.9 menunjukkan contoh data soal dalam bentuk json yang digunakan untuk request POST pada fungsi create.



```

{
  "status": "OK",
  "data": {
    "id": 2,
    "soal": "208/44/21/230/200/171/130/141/62/133/213/46/246/248/123/34/234/251/50/17/152/57/210/0/217/32/75/132/239/225/62/42/198/66/91/227/22/20/237/190/203/205/133/213/46/246/248/123/178/7/21/152/193/196/9/117/208",
    "pilihana": "102/115/81/55/38/132/190/82/122",
    "pilihanb": "11/156/110/227/227/239/169/155/197",
    "pilihanc": "131/92/240/56/26/233/149/21/158",
    "jawaban": "226/230/218",
    "key": "680140331240547234/231423279539691610/81131374005234414",
    "updatedAt": "2024-06-18T17:31:20.550Z",
    "createdAr": "2024-06-18T17:31:20.550Z"
  }
}

```

**Gambar 4.10 Response Fungsi Create**

Query

Query History

1

SELECT \* FROM public."Soals"

2

ORDER BY id ASC

Scratch Pad

X

Data Output

Messages

Notifications

id

[PK] Integer

soal

text

pilihana

text

pilihanb

text

pilihanc

text

jawaban

text

key

text

createdAt

timestamp wi

updatedAt

timestamp wi

1

2

208/44/21/230/200/171/130/141/62/133/213/46/246/248/12...

102/115/...

11/156/1...

131/92/2...

226/230/...

68014033124054...

2024-06-...

2024-06-...

**Gambar 4.11 Tabel Soal setelah Create**

Pada gambar 4.10 menunjukkan response fungsi create yang dikembalikan kepada user dan gambar 4.11 menunjukkan tampilan data soal setelah di enkripsi dan di simpan kedalam tabel Soal pada database PostgreSQL. Data yang telah disimpan kedalam database sudah tidak dapat dibaca dan dimengerti.

#### 4.2.5. Show Data (Dekripsi)

Pada gambar 4.12 menunjukkan fungsi show yang digunakan untuk menangani permintaan HTTP GET pada server. Fungsi ini mengambil data berdasarkan parameter id dari tabel Soal, kemudian jika data ada maka data akan di dekripsi untuk kemudian dikembalikan ke user.

```

async show(req, res) {
  post = await Soal.findByPk(req.params.id);
  if (!post) {
    res.status(404).json({
      status: "FAIL",
      message: "Post not found!",
    });
    return;
  }
  console.log(post);
  data = {
    soal: post.soal,
    pilihana: post.pilihana,
    pilihanb: post.pilihanb,
    pilihanc: post.pilihanc,
    jawaban: post.jawaban,
    key: post.key,
  };
  console.log(data);

  dataDecrypt = decrypt(data, KEY_RABINP_P, KEY_RABINP_Q);

  Object.assign(post, dataDecrypt);

  res.status(200).json({
    status: "OK",
    data: post,
  });
},

```

Gambar 4.12 Fungsi Show Data

Query

Query History

1

SELECT \* FROM public."Soals"

2

ORDER BY id ASC

Scratch Pad

X

Data Output

Messages

Notifications

id

[PK] integer

soal

text

pilihana

text

pilihanb

text

pilihanc

text

jawaban

text

key

text

createdAt

timestamp with time zone

updatedAt

timestamp with time zone

1

2

208/44/21/230/200/171/130/141/62/133/213/46/246/248/12...

102/115/...

11/156/1...

131/92/2...

226/230/...

68014033124054...

2024-06-...

2024-06-...

Gambar 4.13 Tampilan id 2 Tabel Soal

Pada gambar 4.13 menunjukkan tampilan data dengan id 2 di dalam tabel Soal pada database.

```

{
  "status": "OK",
  "data": {
    "id": 2,
    "soal": "Kunci enkripsi yang sama dengan proses dekripsi disebut?",
    "pilihana": "password",
    "pilihanb": "simetri",
    "pilibanc": "asimetri",
    "jawaban": "b",
    "key": "680140331240547234/231423279539691610/81131374005234414",
    "createdAt": "2024-06-18T17:31:20.550Z",
    "updatedAt": "2024-06-18T17:31:20.550Z"
  }
}

```

**Gambar 4.14 Response Fungsi Show**

Pada gambar 4.14 menunjukkan *response* dari fungsi show yang di kembalikan ke user. Data yang dikembalikan sudah dalam bentuk yang bisa dibaca dan dipahami setelah melalui proses dekripsi.

#### **4.2.6. Update Kunci Rabin P**

Untuk melakukan update pada kunci algoritma *Rabin P* yang digunakan, hal yang perlu disiapkan adalah sebagai berikut:

- Kunci algoritma *Rabin P* yang baru dengan syarat  $p \neq q \rightarrow p \equiv q \equiv 3 \text{ mod } 4$
- Tabel database untuk menampung data dari tabel yang kunci algoritma Rabin P nya akan di update, sehingga tabel yang sebelumnya dapat dijadikan backup data dengan kunci algoritma Rabin P yang lama.

```

rabinPKeyUpdate(req, res) {
  Soal.findAll()
    .then((soal) => {
      soal.forEach((soals) => {
        data = {
          soal: soals.soal,
          pilihana: soals.pilihana,
          pilihanb: soals.pilihanb,
          pilihanc: soals.pilihanc,
          jawaban: soals.jawaban,
          key: soals.key,
        };
        dataDecrypt = decrypt(data, KEY_RABINP_P, KEY_RABINP_Q);
        delete dataDecrypt.key;
        console.log(dataDecrypt);
        dataEncrypt = encryp(dataDecrypt, KEY_RABINP_P_NEW, KEY_RABINP_Q_NEW);
        Object.assign(data, dataEncrypt);
        data.id = soals.id;
        console.log(data);
        SoalBaru.create(data)
          .then((post) => {
            soals = data;
          })
          .catch((err) => {
            res.status(201).json({
              status: "FAIL",
              message: err.message,
            });
          });
      });
      res.status(201).json({
        status: "OK",
        data: soal,
      });
    })
    .catch((err) => {
      res.status(400).json({
        status: "FAIL",
        message: err.message,
      });
    });
},

```

**Gambar 4.15 Fungsi *rabinPKeyUpdate***

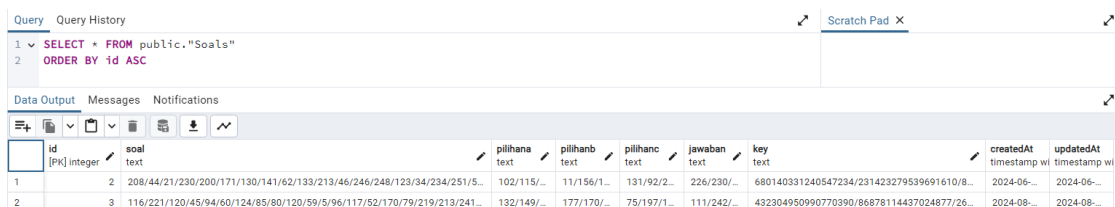
Pada gambar 4.15 menunjukkan proses update kunci algoritma *Rabin P*, langkah-langkahnya sebagai berikut:

- a. Mengambil semua data tabel Soal dari database, hasilnya ditampung dalam soal
- b. Kemudian setiap entri (data) dari soal akan diproses sebagai soals:
  - Menampung nilai dari soals dalam data
  - Dekripsi data menggunakan kunci algoritma *Rabin P* yang lama
  - Menghapus kolom key dari data yang telah di enkripsi karena akan diperbaharui
  - Enkripsi data menggunakan kunci algoritma *Rabin P* yang baru

- Menambahkan kolom id yang lama pada data yang telah dienkripsi untuk memastikan id data pada tabel yang lama tetap sama dengan id data pada tabel yang baru.
- Membuat dan menyimpan data yang di enkripsi dengan kunci algoritma *Rabin P* yang baru kedalam tabel SoalBaru dalam database.

### 1. Tabel dan data sebelum update kunci

Pada gambar 4.16 menunjukkan data pada tabel Soal, dapat dilihat dalam tabel soal terdapat 2 data dengan id 2 dan 3. Data tersebut merupakan data yang dienkripsi menggunakan kunci *Rabin P* yang lama.



The screenshot shows a database query interface with a query window and a results table. The query is: `SELECT * FROM public."Soals" ORDER BY id ASC`. The results table has columns: id, soal, pilihana, pilihanb, pilihanc, jawaban, key, createdAt, and updatedAt. Two rows are displayed, corresponding to id 2 and id 3.

id	soal	pilihana	pilihanb	pilihanc	jawaban	key	createdAt	updatedAt
2	208/44/21/230/200/171/130/141/62/133/213/46/246/248/123/34/234/251/5...	102/115/...	11/156/1...	131/92/2...	226/230/...	680140331240547234/231423279539691610/8...	2024-06-...	2024-06-...
3	116/221/120/45/94/60/124/85/80/120/59/5/96/117/52/170/79/219/213/241...	132/149/...	177/170/...	75/197/1...	111/242/...	432304950990770390/86878114437024877/26...	2024-08-...	2024-08-...

Gambar 4.16 Tabel Database Sebelum Update Kunci *Rabin P*



The screenshot shows a JSON response with a status of "OK" and a data object containing an array of two question objects. Each object has fields for id, soal, pilihana, pilihanb, pilihanc, jawaban, key, createdAt, and updatedAt.

```

{
  "status": "OK",
  "data": {
    "soal": [
      {
        "id": 2,
        "soal": "Kunci enkripsi yang sama dengan proses dekripsi disebut?",
        "pilihana": "password",
        "pilihanb": "simetri",
        "pilihanc": "asimetri",
        "jawaban": "b",
        "key": "680140331240547234/231423279539691610/81131374005234414",
        "createdAt": "2024-06-18T17:31:20.550Z",
        "updatedAt": "2024-06-18T17:31:20.550Z"
      },
      {
        "id": 3,
        "soal": "Salah satu komponen komputer yang bersifat fisik, dapat dilihat, dan disentuh adalah..",
        "pilihana": "Software",
        "pilihanb": "Aplikasi",
        "pilihanc": "Hardware",
        "jawaban": "c",
        "key": "432304950990770390/86878114437024877/268356918290423737",
        "createdAt": "2024-08-16T04:00:46.622Z",
        "updatedAt": "2024-08-16T04:00:46.622Z"
      }
    ]
  }
}

```

Gambar 4.17 Data dekripsi menggunakan kunci *Rabin P* yang lama

Pada gambar 4.17 menunjukkan data dari tabel soal ketika di dekripsi menggunakan kunci algoritma *Rabin P* yang lama.

## 2. Tabel dan data setelah update kunci

Pada gambar 4.18 menunjukkan data pada tabel SoalBaru, dapat dilihat dalam tabel soal terdapat 2 data dengan id 2 dan 3. Data tersebut merupakan data yang sama dengan data pada tabel Soal yang dienkripsi dengan kunci algoritma *Rabin P* yang baru.

id	soal	pilihanA	pilihanB	pilihanC	jawaban	key	createdAt	updatedAt
2	254/108/247/40/8/207/53/171/119/66/151/76/136/24/255/58/86/1...	81/253/1...	109/200/...	217/152/...	108/98/6	128156290999833511/90265181333554263/562097058...	2024-08-...	2024-08-...
3	157/87/204/146/109/16/37/111/36/229/118/41/189/143/216/125/2...	81/165/1...	22/161/1...	149/226/...	46/71/208	272596139983447597/151966979459149704/14163764...	2024-08-...	2024-08-...

**Gambar 4.18 Tabel Database Setelah Update Kunci *Rabin P***

```

{
  "status": "OK",
  "data": {
    "soal": [
      {
        "id": 2,
        "soal": "Kunci enkripsi yang sama dengan proses dekripsi disebut?",
        "pilihanA": "password",
        "pilihanB": "simetri",
        "pilihanC": "asimetri",
        "jawaban": "b",
        "key": "128156290999833511/90265181333554263/562097058815136029",
        "createdAt": "2024-08-16T04:18:23.8052",
        "updatedAt": "2024-08-16T04:18:23.8052"
      },
      {
        "id": 3,
        "soal": "Salah satu komponen komputer yang bersifat fisik, dapat dilihat, dan disentuh adalah..",
        "pilihanA": "Software",
        "pilihanB": "Aplikasi",
        "pilihanC": "Hardware",
        "jawaban": "c",
        "key": "272596139983447597/151966979459149704/141637645799927395",
        "createdAt": "2024-08-16T04:18:23.8172",
        "updatedAt": "2024-08-16T04:18:23.8172"
      }
    ]
  }
}

```

**Gambar 4.19 Data dekripsi menggunakan kunci *Rabin P* yang baru**

Pada gambar 4.19 menunjukkan data dari tabel SoalBaru ketika di dekripsi menggunakan kunci algoritma *Rabin P* yang baru.

Pada gambar 4.16 dan gambar 4.18 bisa dilihat bahwa data setelah dienkripsi dengan kunci yang berbeda akan membentuk *chiphertext* yang berbeda untuk disimpan ke dalam *database*. Dari hasil dekripsi seluruh data dalam tabel Soal pada gambar 4.17 dan tabel SoalBaru pada gambar 4.19 bisa dilihat bahwa data pada tabel Soal sama dengan data pada tabel SoalBaru yang menunjukkan bahwa update kunci algoritma *Rabin P* berhasil dilakukan.

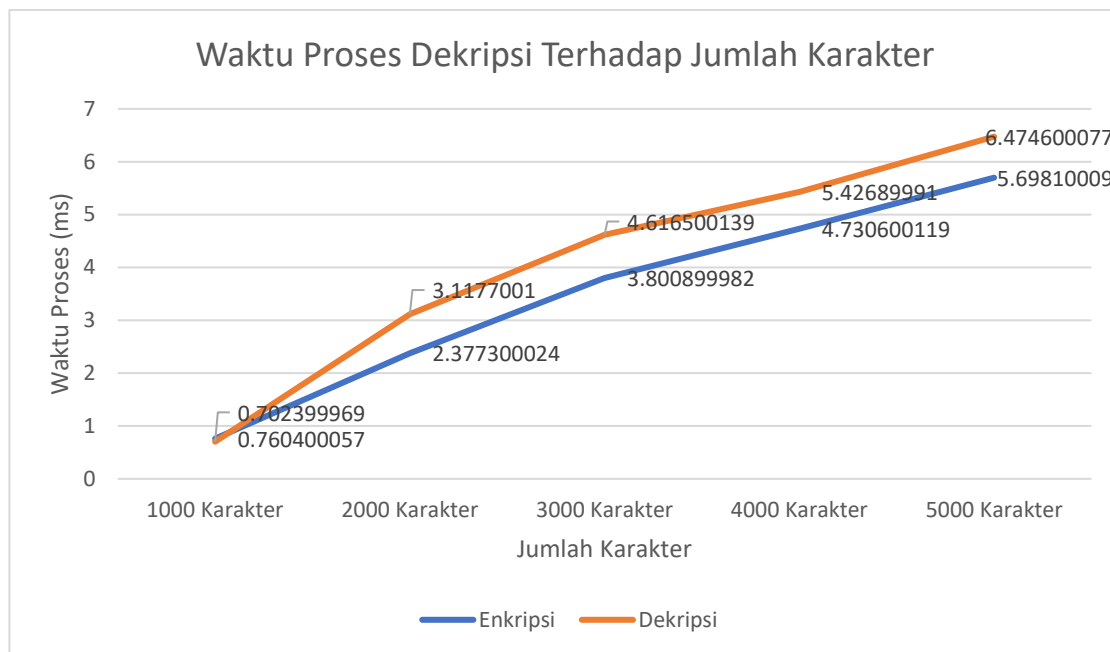
#### 4.2.7. Pengujian Sistem Terhadap Waktu

Pengujian sistem dapat diukur dari waktu yang dibutuhkan sistem pada saat enkripsi dan dekripsi memakai algoritma *Hill Cipher*, dan algoritma *Rabin P*, dan pembangkit bilangan acak *Blum-Blum-Shub* berdasarkan waktu proses yang terdapat pada sistem. Sistem akan diuji dengan jumlah karakter yang berbeda yaitu, 1000 karakter, 2000 karakter, 3000 karakter, 4000 karakter, dan 5000 karakter namun dengan jumlah kolom data yang sama yaitu 1 kolom dan panjang kunci algoritma *Rabin P* yang digunakan adalah 60 bit.

**Tabel 4.2 Hasil Pengujian Jumlah Karakter Terhadap Waktu Proses**

Jumlah karakter	Waktu Proses Hill Cipher (ms)	
	Enkripsi	Dekripsi
1000 karakter	0.7604000568389893	0.7023999691009521
2000 karakter	2.3773000240325928	3.117700099945068
3000 karakter	3.8008999824523926	4.61650013923645
4000 karakter	4.730600118637085	5.4268999099731445
5000 karakter	5.6981000900268555	6.474600076675415

Pada tabel 4.1 menunjukkan bahwa jumlah dari setiap karakter dalam kolom data mempengaruhi waktu proses enkripsi dan dekripsi data. Semakin banyak karakter dalam kolom data maka semakin lama waktu proses yang akan diperoleh.



**Gambar 4.20 Grafik Pengujian Jumlah Karakter Terhadap Waktu Proses**

Pada gambar 4.20 menunjukkan bahwa waktu proses enkripsi dan dekripsi menggunakan *Hill Cipher* dipengaruhi oleh jumlah karakter dalam kolom data. Semakin besar jumlah karakter, maka waktu yang dibutuhkan untuk enkripsi dan dekripsi data semakin lama.

Dari pengujian sistem pada saat enkripsi dan dekripsi dapat dilihat bahwa hubungan jumlah karakter berbanding lurus *linear* terhadap waktu proses enkripsi data dan dekripsi data. Artinya, semakin besar jumlah karakter maka waktu yang dibutuhkan untuk mengeksekusi program juga akan semakin lama.



## **BAB 5**

### **PENUTUP**

#### **5.1. Kesimpulan**

Dengan beralaskan hasil analisis dan pengujian pada sistem pengamanan *database Field-Level Encryption* menggunakan *Hybrid Cryptosystem* algoritma *Hill Cipher* dan algoritma *Rabin P* yang telah dilakukan, maka didapatkan kesimpulan sebagai berikut:

1. Pada proses enkripsi dengan *Hybrid Cryptosystem* algoritma *Hill Cipher* dan algoritma *Rabin P* berhasil dalam pengamanan data dikarenakan data (*ciphertext*) yang disimpan ke database sudah tidak terbaca maknanya.
2. Pada proses dekripsi, *database* terenkripsi berhasil ditampilkan dalam bentuk yang dapat dibaca dan dipahami sesuai dengan data aslinya, tanpa ada kehilangan informasi.
3. Jumlah karakter berbanding lurus *linear* terhadap waktu proses enkripsi data dan dekripsi data. Semakin banyak jumlah karakter pada data maka waktu yang dibutuhkan untuk proses enkripsi dan dekripsinya akan semakin lama.
4. Pembaharuan kunci algoritma *Rabin P* yang diinputkan user dilakukan dengan mendekripsi terlebih dahulu data pada tabel database menggunakan kunci algoritma *Rabin P* yang lama kemudian mengenkripsi ulang data tersebut menggunakan kunci algoritma *Rabin P* yang baru, kemudian disimpan kembali ke tabel *database*. Sistem dapat menangani proses pembaharuan kunci algoritma *Rabin P* tanpa kehilangan data yang lama.

#### **5.2. Saran**

1. Diharapkan pada penelitian selanjutnya dapat menambahkan algoritma enkripsi dan dekripsi lainnya seperti algoritma AES, DES, RC4, dll yang merupakan kriptografi modern sehingga data (*ciphertext*) yang disimpan dalam database akan lebih aman.
2. Diharapkan dalam penelitian selanjutnya dapat menggunakan jenis *database* selain *relational database*.

## DAFTAR PUSTAKA


- Srivastava, A. K., & Mathur, A. (2013). The Rabin Cryptosystem & analysis in measure of. *International Journal of Scientific and Research Publications*, 3(6), 1-4.
- Andrico, & Syafrullah, M. (2018). Aplikasi Enkripsi Database Menggunakan Algoritma Rc4 Berbasis Desktop. 1(3), 1011-1017.
- Bhattacharjee, A., & Shyamasundar, R. (2009). Activity Diagrams : A Formal Framework to Model Business Processes and Code Generation. *Journal of Object Technology*, 8(1), 189-220.
- Budiman, M., & Irvida, T. (2023). Securing text using Rabin-p public key cryptosystem and Spritz algorithm in a hybrid cryptosystem: a tutorial.
- Budiman, M., Handrizal, & Azzahra, S. (2021). An implementation of Rabin-p cryptosystem and affine cipher in a hybrid scheme to secure text. *Journal of Physics: Conference Series*, 1898 012042.
- Budiman, M., Rachmawati, D., & Utami, R. (2019). The cryptanalysis of the Rabin public key algorithm using the Fermat factorization method. *Journal of Physics: Conference Series* 1235.
- Easttom, W. (2021). *Modern Cryptography*. New York: Springer.
- Fitri, R. (2020). *Pemrograman Basis Data Menggunakan MySQL*. Banjarmasin: Poliban Press.
- Hidayatuloh, K., Yustantina, & Kusmadi. (2021). Perbandingan Metode Stream Cipher Dan Hill Cipher Dalam Keamanan Data. *Jurnal Infotronik*, 6(1).
- Jamaludin, & Romindo . (2020). Implementation of Combination Vigenere Cipher and RSA in . *International Journal of Information System & Technology* , 4(1), 471-481.
- Jayanti, N. K., & Sumiari, K. (2018). *Teori Basis Data*. Yogyakarta: ANDI.

- Kurniawan, T. A. (2018). Pemodelan Use Case (Uml): Evaluasi Terhadap Beberapa Kesalahan Dalam Praktik. *Jurnal Teknologi Informasi dan Ilmu Komputer*, 5(1), 77-86.
- Malik, M., & Patel, T. (2016). Database Security - Attacks and Control Methods. 6.
- Mukhtar, H. (2018). *Kriptografi untuk Keamanan Data*. Yogyakarta: Deepublish.
- Mulyani, S. (2016). *Metode Analisis dan Perancangan Sistem*. Bandung: Abdi Sistematika.
- Munir, R. (2019). *Kriptografi Edisi Kedua*. Bandung: Penerbit Informatika.
- Nurmalasari, N., Anna, A., & Arissusandi, R. (2019). Rancang Bangun Sistem Informasi Akuntansi Laporan Laba Rugi Berbasis Web Pada Pt. United Tractors Pontianak. *Evolusi: Jurnal Sains dan Manajemen*, 7(2), 6-14.
- Rachmawati, D., Sharif, A., & Ericko. (2019). Hybrid Cryptosystem Combination Algorithm of Hill Cipher 3x3 and Elgamal To Secure Instant Messaging For Android. *Journal of Physics: Conference Series* 1235.
- Rahmadhiyanti, S. (2019). Implementasi Kriptografi RSA Untuk Peningkatan Keamanan Database E-Commerce. *Jurnal Pelita Informatika*, 8(2), 288-291.
- Rumbaugh, J., Jacobson, I., & Booch, G. (1999). *The unified modeling language reference manual*. Reading, Mass: Addison-Wesley.
- Ruswandi, M., & Windarto. (2022). Enkripsi Database Sistem Informasi Helpdesk Dengan Algoritme Kriptografi AES-128 dan Vigenere Chiper. *SKANIKA: Sistem Komputer dan Teknik Informatika*, 240-254.
- Santoso, Y. S. (2021). Message Security Using a Combination of Hill Cipher and RSA Algorithms. *Jurnal Matematika Dan Ilmu Pengetahuan Alam LLDikti Wilayah 1 (JUMPA)*, 1(1), 20-28.

- Setiaji, & Sastra, R. (2021). Implementasi Diagram UML (Unified Modelling Language) Pada Perancangan . *Jurnal Teknik Komputer AMIK BSI*, 7(1), 106-111.
- Setiawan, A., & Fatimah, T. (2021). Implementasi Algoritma Kriptografi Rc4 Untuk Keamanan Databaseaplikasi Penggajian Karyawan Berbasis Web Pada PT. Trans Intra Asia. *SKANIKA*, 4(1), 66-71.
- Siregar, N., Faisal, I., & Handoko, D. (2022). Menerapkan Algoritma Hill Cipher dan Matriks 2x2 Dalam Mengamankan File Teks Menggunakan Kode ASCII. *Jurnal Ilmu Komputer dan Sistem Informasi (JIRSI)*, Volume: 1, Nomor : 2, 70-83.
- Smart, N. (1999). *Cryptography: An Introduction* (3rd ed.). Bristol City.
- Wanto, A. (2015). Analisis Mengatasi Sniffing Dan Spoofing Menggunakan Metode Enkripsi Dan Dekripsi Algoritma Hill Chiper. *Seminar Nasional Ilmu Komputer (SNIKOM)* , 145-153.

## LAMPIRAN A

### DOKUMENTASI PENGGUNAAN LIBRARY

 README

## PENGAMANAN DATABASE FIELD-LEVEL ENCRYPTION MENGGUNAKAN HYBRID CRYPTOSYSTEM ALGORITMA HILL CIPHER DAN ALGORITMA RABIN P

Sastra Harapan Gulo

201401066

PROGRAM STUDI S1 ILMU KOMPUTER

FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI

UNIVERSITAS SUMATERA UTARA

MEDAN

2024

### Penggunaan Library

#### Install Library

Untuk penggunaan library dalam pengamanan database, lakukan penginstalan library pada proyek node.js anda dengan menggunakan

```
npm intasll hillcipher_rabinp
```

#### Import Fungsi

Setelah instalasi library, anda haru memanggil fungsi yang tersedia dalam library terlebih dahulu yakni {encrypt, decrypt}.

fungsi encrypt digunakan untuk enkripsi data agar sulit dimengerti sbelum disimpan ke databse, fungsi decrypt digunakan untuk dekripsi data dari database agar kembali kebentuk semua.

```
const { encrypt, decrypt } = require("hillcipher_rabinp");
```

#### Kunci Algoritma Rabin P

Siapkanlah kunci privat (p dan q) algoritma Rabin P dengan syarat  $p, q$  adalah bil. prima dan  $p \neq q \rightarrow p \equiv q \equiv 3 \pmod{4}$ .

Kunci dapat disimpan dalam file enviromen variabel `.env` untuk keamanan

#### Contoh Data

```
{
  "soal": "Kunci enkripsi yang sama dengan proses dekripsi disebut?",
  "pilihana": "password",
  "pilihanb": "simetri",
  "pilihanc": "asimetri",
  "jawaban": "b"
}
```

### Creta Data

Berikut adalah contoh penggunaan fungsi encrypt untuk membuat data baru dalam database

```
create(req, res) {
  data = {
    soal: req.body.soal,
    pilihana: req.body.pilihana,
    pilihanb: req.body.pilihanb,
    pilihanc: req.body.pilihanc,
    jawaban: req.body.jawaban,
  };
  encrypData = encryp(data, 925789379n, 904318643n);
  console.log(encrypData);

  Soal.create(encrypData)
    .then((post) => {
      res.status(201).json({
        status: "OK",
        data: post,
      });
    })
    .catch((err) => {
      res.status(201).json({
        status: "FAIL",
        message: err.message,
      });
    });
},
```

### List Data

Berikut adalah contoh penggunaan fungsi decrypt untuk menampilkan data dari database

```
list(req, res) {
  SoalBaru.findAll()
    .then((soal) => {
      soal.forEach((soals) => {
        data = {
          soal: soals.soal,
          pilihana: soals.pilihana,
          pilihanb: soals.pilihanb,
          pilihanc: soals.pilihanc,
          jawaban: soals.jawaban,
          key: soals.key,
        };

        dataDecrypt = decrypt(data, KEY_RABINP_P, KEY_RABINP_Q);

        Object.assign(soals, dataDecrypt);
      });
      res.status(200).json({
        status: "OK",
        data: {
          soal,
        },
      });
    })
    .catch((err) => {
      res.status(400).json({
        status: "FAIL",
        message: err.message,
      });
    });
},
```

## Update Kunci Rabin P

Berikut adalah contoh untuk memperbaharui kunci Rabin P yang digunakan

```
rabinPKeyUpdate(req, res) {  
  Soal.findAll()  
    .then((soal) => {  
      soal.forEach((soals) => {  
        data = {  
          soal: soals.soal,  
          pilihana: soals.pilihana,  
          pilihanb: soals.pilihanb,  
          pilihanc: soals.pilihanc,  
          jawaban: soals.jawaban,  
          key: soals.key,  
        };  
  
        dataDecrypt = decrypt(data, KEY_RABINP_P, KEY_RABINP_Q);  
        delete dataDecrypt.key;  
        dataEncrypt = encrypt(dataDecrypt, KEY_RABINP_P_NEW, KEY_RABINP_Q_NEW);  
        Object.assign(data, dataEncrypt);  
        data.id = soals.id;  
        console.log(data);  
        SoalBaru.create(data)  
          .then((post) => {  
            soals = data;  
          })  
          .catch((err) => {  
            res.status(201).json({  
              status: "FAIL",  
              message: err.message,  
            });  
          });  
      });  
      res.status(201).json({  
        status: "OK",  
        data: soal,  
      });  
    })  
    .catch((err) => {  
      res.status(400).json({  
        status: "FAIL",  
        message: err.message,  
      });  
    });  
  },  
}
```