# BAB III

# ANALISA DAN PERANCANGAN

## III.1. Analisa Sistem

Analisa masalah yang didapat dari penelitian ini adalah membuat implementasi pengamanan *file* teks dengan menggunakan algoritma *triangle chain* dan *rivest cipher (RC4).* Algoritma *triangle chain* dan RC4 adalah suatu metode pengenkripasian file teks yang akan di buat aplikasinya pada pengerjaan penelitian ini. Aplikasi ini bertujuan untuk melakukan pengenkripsian dan deskripsian data secara khusus pada. Algoritma *triangle chain* dan *RC4* diperkenalkan untuk pertama kalinya pada tahun 1987 oleh Rivest Shamir Adleman. Algoritma *triangle chain* merupakan suatu algoritma dasar yang banyak dikembangkan oleh orang-orang. Misalnya saja IDEA, AES, dan OTP. Algoritma *triangle chain* sendiri sering disebut dengan algoritma rantai segitiga

(Doni Ariyus 2008). Algoritma ini juga disebut algoritma " one time pad" karena melakukan pengenkripsian *file* teks dengan cara membangkitkan kunci secara random dan panjang kunci sepanjang kunci plainteks yang akan dienkripsi. Yang dibutuhkan algoritma kriptografi rantai segitiga pembangkitan kunci-kunci secara otomatis dengan teknik berantai terhadap suatu *file*. Alasan pemilihan algoritma adalah bahwa lgoritma ini memberikan performa yang baik untuk pengamanan *file* teks dibandingkan algoritma-algoritma Simetri dan Asimetri pada pembahasan masalah lebih ditekankan pada analisa hasil algoritma *triangle chain* dan *rivest cipher (RC4).*

### III.1.1. Analisa Input

Dalam aplikasi sistem pengamanan *file* teks yang akan di implementasikan dalam aplikasi adalah menggunakan algloritma *triangle chain* serta *RC4*. Dengan membaca tiap karakter yang

dimasukan dari *file* yang dimasukan lalu diproses hingga membentuk suatu tampilan atau hasil yang tidak dapat dibaca. Dalam proses yang dikembangkan hanya menampilkan proses kerja algoritma *triangle chain* dan algoritma RC4 sebagai media pengamanan *file* teks tersebut agar pengguna dapat mengetahui bagaimana proses yang terjadi dari sebuah algoritma *triangle chain* serta algoritma *RC4.*

### III.1.2. Analisa Proses

Permasalahan yang dibahas adalah membuat suatu pengkodean dengan menggunakan algoritma *triangle chain* dan *RC4*. Masalah enkripsi *file* teks dengan algoritma *RC4* muncul ketika proses enkripsi dalam sistem sedang. Pembahasan masalah lebih ditekankan pada proses indeks kerja algoritma *triangle chain* dan juga algoritma *RC4*.

### III.2.1. Algoritma Triangle Chain

Algoritma kriptografi *triangle chain* atau umumnya dikenal dengan sebutan rantai segitiga merupakan *cipher* yang ide awalnya dari algoritma kriptografi *One Time Pad*, yaitu kunci yang dibangkitkan secara *random* dan

panjang kunci sepanjang plainteks yang akan dienkripsi. Secara matematis pola enkripsi rantai segitiga dapat digambarkan dengan matriks N x N dengan N merupakan panjang plainteks yang akan dienkripsi dan operasi pada alfabet ASCII.

Matriks dilambangkan dengan Mij, dengan $1 \leq i \leq N$ dan $1 \leq j \leq N$, nilai integer kunci dengan K, faktor penggali merupakan tabel integer R. Plainteks dengan P dimana P merupakan tabel plainteks dengan panjangN yaitu P[N].

Berikut operasi matriks untuk proses enkripsi:

1. Matriks enkripsi segitiga pertama

   Untuk baris ke-1 :

   M1j = P[ j ] + (K * R[ 1]) mod 256

   untuk baris ke-2 dan selanjutnya untuk nilai j ≥ i :

   Mij = M(i-1)j +(K * R [ i ]) mod 256

   sehingga nilai *cipher*teks yang diperoleh adalah :

   Mij pada nilai j = (N+i)-N.

2. Matriks enkripsi segitiga kedua

   Nilai P diperoleh dari nilai Mij pada i = j

   Untuk baris ke-1 :

   M1j = P[ j ] + (K*R[1])) mod 256

   untuk baris ke 2 dan selanjutnya untuk nilai j ≤ (N+1) – i :

   Mij = M(i-1)j+(K*R[i])) mod 256

   sehingga nilai *cipher*teks yang diperoleh adalah :

   Mij pada nilai j = (N+1)-i.

   Keterangan :

   P = Plainteks

   N = Jumlah karakter *plain*teks

   M = Matriks penampung hasil penyandian

   K = Kunci

   R = *Row* (baris perkalian faktor pengali dengan kunci)

   i = Indeks faktor pengali

j = Indek karakter p*laint*eks

Sedangkan untuk proses dekripsi kebalikan dari proses enkripsi.

**III.2. Studi Kasus**

Pada studi ini akan dilakukan perhitungan enkripsi dan deskripsi dengan algoritma *triangle Chain* sebagai berikut:

1. Matriks enkripsi segitiga pertama

Plaintext = TARONI

Kunci yang digunakan = 3 (Bilangan Asli ⇨ *Integer*)

Sesuai dengan panjang plaintext N = 6

Faktor pengali (*fp = R*) berdasarkan nilai

N = (deret bilangan asli)⇨(1,2,3,4,5,6,).

Sebelum plaintext dienkripsi, setiap karakter terlebih dahulu dirubah kenilai desimal sesuai dengan nilai ASCII dengan nilai mod 256:

**KARAKTER** ⇨ T A R O N I

**NILAI DESIMAL** ⇨ 84 65 82 79 78 73

Langkah selanjutnya adalah melakukan proses enkripsi segitiga pertama sesuai dengan rumus:

P = TARONI

N = 6

K = 3

R = 1, 2, 3, 4, 5, 6

Rumus untuk baris pertama (i = 1):

**M1j = P[ j ] + (K * R[ 1]) mod 256**

Maka penyelesaian enkripsi pertama :
M11
= ( T + 3 * (1)) mod 256

= ( 84 + 3 ) Mod 256  adalah  87 (huruf "W" dalam karakter ASCII 256)

M12
= ( A + 3 * (1)) Mod 256

= ( 65 + 3 ) Mod 256  adalah  68 (huruf "D" dalam karakter ASCII 256)

M13
= (85 + 3 *(1)) Mod 256

= ( R + 3 * (1)) Mod 256  adalah  85 (huruf "U" dalam karakter  ASCII 256)

M14
= ( O + 3 * (1)) mod 256

= ( 79 + 3 ) Mod 256  adalah  82 (huruf "R" dalam karakter  ASCII 256)

M15
= ( N + 3 * (1)) mod 256

= ( 78 + 3 ) Mod 256  adalah  81 (huruf "Q" dalam karakter ASCII 256)

M16
= ( I + 3 * (1)) mod 256

= ( 73 + 3 ) Mod 256 adalah 76 (huruf "L" dalam karakter ASCII 256)

Hasil dari enkripsi baris pertama (i=1) (tanpa tanda " '') adalah  "**WDURQL**"

**TARONI** (nilai desimal ASCII :  84 65 82 79 78 73)  ⇨ i = 0
**WDURQ**L ( nilai desimal ASCII : **87** 68 85 82 81 76)⇨  i = 0
Hasil enkripsi baris pertama akan di gunakan sebagai plaintext untuk baris
Kedua (i = 2), dimana nilai j ≥ i, sehingga:
Enkripsi⇨ i = 2, j = 2
Rumus untuk baris ke-2 (i = 2) dan seterusnya (i = n):

Mij= M(i-1)j+(K*R[i])) mod 256
Penyelesaian:
M12
= ( A + 3 * (1)) Mod 256

= ( 65 + 3 ) Mod 256 adalah 68 (huruf “D” dalam karakter ASCII 256)

M23
= (U + 6) mod 256

= ( 85 + 6) mod 256 adalah 91 (huruf “[” dalam karakter ASCII 256)

M24
= ( M(1)4+ 3 * (2)) Mod 256

= ( 82 + 6) mod 256 adalah 88 (huruf “X” dalam karakter ASCII 256)

M25
= ( M(1)5+ 3 * (2)) Mod 256

= ( 81 + 6) mod 25 adalah6 87 (huruf “W” dalam karakter ASCII 256)

M26
= ( M(1)6+ 3 * (2)) Mod 256

= ( 76 + 6) mod 256 adalah 82 (huruf “R” dalam karakter ASCII 256)

Hasil enkripsi ke-2 ( i – 2) adalah “ J[XWR “ dengan nilai desimal **74** 91 88 87 82.

Hasil enkripsi pada baris ke-2 (i = 2) akan digunakan sebagai plaintext pada enkripsi baris ke-3 (i = 3), maka enkripsi⇨i = 3, j = 3
M33
= ( M(2)3+ 3 * (3)) Mod 256

= ( 91 + 9) mod 256 adalah 100 (huruf “d” dalam karakter ASCII 256)

M34
= ( M(2)4+ 3 * (3)) Mod 256

= ( 88 + 9) mod 256  adalah  97 (huruf “a” dalam karakter ASCII 256)

M35
= ( M(2)5+ 3 * (3)) Mod 256

= ( 87 + 9) mod 256 adalah  96 (huruf “`” dalam karakter ASCII 256)

M36
= ( M(2)6+ 3 * (3)) Mod 256

= ( 82 + 9) mod 256  adalah  91 (huruf “[” dalam karakter ASCII 256)

Hasil enkripsi ke-3 adalah  “ d a ` [  “ dengan nilai desimal **100** 97 96 91.
Hasil enkripsi baris ke-3 (i = 3) dapat digunakan sebagai plaintext enkripsi baris ke-4 (i = 4) => i = 4, j =
M44
= ( M(3)4+ 3 * (4)) Mod 256

= ( 97 + 12) mod 256 adalah 109 (huruf “m” dalam karakter ASCII 256)

M45
= ( M(3)5+ 3 * (4)) Mod 256

= ( 96 + 12) mod 256  adalah  108 (huruf “l” dalam karakter ASCII 256)

M46
= ( M(3)6+ 3 * (4)) Mod 256

= ( 91 + 12) mod 256  adalah 103 (huruf “g” dalam karakter ASCII 256)
Hasil enkripsi baris ke-4 adalah **“ mlg “** dengan nilai desimal **109** 108 103.
Hasil  enkripsi pada baris ke-4 (i = 4) akan digunakan sebagai plaintext pada enkripsi baris ke-5 (i = 5) i = 5, j = 5
M55
= ( M(4)5+ 3 * (5)) Mod 256

= ( 108 + 15) mod 256  adalah  123 (huruf “{ ” dalam karakter ASCII 256)

M56
= ( M(4)6+ 3 * (5)) Mod 256

= ( 103 + 15) mod 256  adalah 118 (huruf “v” dalam karakter ASCII 256)

Hasil enkripsi ke-5 adalah **“ {v “** dengan bilangan desimal **123** 118.
Hasil enkripsi ke-5 baris (i = 5) akan digunakan sebagai plaintext pada

Enkripsi baris ke-6 (i = 6)

M66
= ( M(5)6+ 3 * (6)) Mod 256

= ( 118 + 18) mod 256  adalah  136 (huruf “ ^ ” dalam karakter ASCII 256)

Hasil enkripsi baris ke-6 adalah **“ ^ “ 136**. Hasil enkripsi keseluruhan segitiga

Pertama sampai baris ke-6  dan nilai enkripsi akan diambil dari nilai setiap

barisnya sebanyak satu karakter yang di mulai dari (i): ke-1, ke-2, ke-3, ke-4,

ke-5, ke-6. Dengan nilai $j = (N + i) – N$, = banyaknya karakter data

(plaintext).

| | | |
|---|---|---|
| TARONI | 84 65 82 79 78 73 | ⇨ i = 0 |
| **W**DURQL | **87** 68 85 82 81 76 | ⇨ i = 1 |
| **J**[XWR | **74** 91 88 87 82 | ⇨ i = 2 |
| **d** a ` [ | **100** 97 87 91 | ⇨ i = 3 |
| **m**lg | **109** 99 103 | ⇨ i = 4 |
| **{**v | **114** 118 | ⇨ i = 5 |
| ^ | **136** | ⇨ i = 6 |

Maka dari hasil proses enkripsi pertama adalah ( **W J d m {** ^ )

2. Matriks enkripsi segitiga ke dua

   Langkah yang dilakukan pada proses enkripsi segitiga kedua hampir sama dengan proses enkripsi segitiga pertama. Untuk faktor pengalih terhadap kunci masih sama. Pada proses enkripsi segitiga kedua ini. yang menjadi plaintext nya adalah hasil nilai enkripsi pada segitiga pertama (tanpa tanda “ “) yaitu “ **WJdm{**^ “ dengan nilai desimal **87 74 100 109 123 136.**
   Rumus enkripsi segitiga kedua ini:
   Untuk baris pertama (*i -= 1*):
   M1j= P[ j ] + (K*R[1])) mod 256

Untuk baris ke-2 dan seterusnya untuk nilai j ≤ *(N + 1) – i*

Mij= M(i-1)j+(K*R[i])) mod 256

Sehingga nilai ciphertext yang diperoleh dari enkripsi segitiga kedua:

Mijpada nilai j = (N+1)-i.

Proses penyelesain:

Plaintext:

| **W** | **J** | **d** | **m** | **{** | **^** |
|---|---|---|---|---|---|
| **87** | **74** | **100** | **109** | **123** | **136.** |

Maka penyelesaian untuk pertama (*i = 1*):

M11
= (W + (3 * 1)) mod 256

= (87 + 3) mod 256 adalah 90 (huruf "Z" dalam karakter ASCII 256)

M12
= (J + (3 * 1)) mod 256

= (74 + 3) mod 256 adalah 77 (huruf "M" dalam karakter ASCII 256)

M13
= (d + (3 * 1)) mod 256

= (100 + 3) mod 256 adalah 103 (huruf "g" dalam karakter ASCII 256)

M14
= (m + (3 * 1)) mod 256

= (109 + 3) mod 256 adalah 112 (huruf "p" dalam karakter ASCII 256)

M15
= ({ + (3 * 1)) mod 256

= (123 + 3) mod 256  adalah 126 (huruf "~" dalam karakter ASCII 256)

M16
= ( ˆ + (3 * 1)) mod 256

= (136 + 3) mod 256  adalah 139 (huruf " ‹ " dalam karakter ASCII 256)

Hasil enkripsi baris ke-1 (i = 1) adalah **" Z M g p ~ ‹ "** dengan nilai desimal 90 77 103 112 126 **139.**
Hasil enkripsi baris (i = 1) akan digunakan sebagai plaintext pada enkripsi ke-2 dimana nilai i = 2; j ≤ (6 + 1) – 2  j ≤ 5
M21
= (Z + (3 * 2)) mod 256
= (90 + 6) mod 256  adalah  96 (huruf " ` " dalam karakter ASCII 256)

M22
= (M + (3 * 2)) mod 256

= (77 + 6) mod 256  adalah  83 (huruf "S" dalam karakter ASCII 256)

M23
= (g + (3 * 2)) mod 256

= (103 + 6) mod 256  adalah 109 (huruf "m" dalam karakter ASCII 256)

M24
= (p + (3 * 2)) mod 256

= (112 + 6) mod 256  adalah 118 (huruf "v" dalam karakter ASCII 256)

M25
= (~ + (3 * 2)) mod 256

= (126 + 6) mod 256  adalah  132 (huruf ",," dalam karakter ASCII 256)

Hasil enkripsi ke-2 (i = 2) (tanpa tanda "  ") adalah " **S m V ,,** " dengan nilai desimal 96 83 109 118 **132.**
Hasil enkripsi  ke-3 (i = 2) akan digunakan sebagai plaintext pada enkripsi

Baris ke-3  i = 3 ; j ≤ (6 + 1) – 3  j ≤ 4
M31
= ( ` + (3 * 3)) mod 256
= (96 + 9) mod 256  adalah 105 (huruf " i " dalam karakter ASCII 256)

M32

= (S + (3 * 3)) mod 256

= (83 + 9) mod 256  adalah  92 (huruf " \ " dalam karakter ASCII 256)

M33

= (m + (3 * 3)) mod 256

= (109 + 9) mod 256  adalah  118 (huruf " v " dalam karakter ASCII 256)

M34

= (V + (3 * 3)) mod 256

= (118 + 9) mod 256  adalah 127 (huruf " □ " dalam karakter ASCII 256 )

Hasil enkripsi baris ke-3 (i = 3) adalah **" i \ v □'** dengan nilai desimal 105 92 118 127.
Hasil enkripsi baris ke-3 (i = 3) akan digunakan sebagai plaintext pada enkripsi baris ke-4 = 4 ; j ≤ (6 + 1) – 4 j ≤ 3
M41

= ( i + (3 * 4)) mod 256

= (105 + 12) mod 256  adalah  117 (huruf " u " dalam karakter ASCII 256)

M42
= ( \ + (3 * 4)) mod 256

= (92 + 12) mod 256  adalah  104 (huruf " h " dalam karakter ASCII 256)

M43

= ( v + (3 * 4)) mod 256

= (118 + 12) mod 256  adalah  130 (huruf " , " dalam karakter ASCII 256)

Hasil enkripsi baris ke-4 adalah " **u h ,** " dengan nilai desimal 117 104 **130.**

Hasil untuk baris ke-4 (i = 4) akan  digunakan sebagai plaintext pada

Enkripsi baris ke-5 i = 5 ; j ≤ (6 + 1) – 5  j ≤ 2
M51

= ( u + (3 * 5)) mod 256

= (117 + 15) mod 256  adalah 132 (huruf " ,, " dalam karakter ASCII 256)

M52

= ( r + (3 * 5)) mod 256

= (104 + 15) mod 256  adalah 119 (huruf " w " dalam karakter ASCII 256)

Hasil enkripsi baris ke-5 (i = 5) adalah " **,, w** " dengan nilai desimal 132 **119**.
Hasil enkripsi baris ke-5 (i = 5) akan digunakan sebagai plaintext pada enkripsi  baris ke-6 i = 6 ; j ≤ (6 + 1) – 6  j ≤ 1
= ( ,, + (3 * 6)) mod 256

= (132 + 18) mod 256  adalah  150 (huruf " – " dalam karakter ASCII 256)

Hasil untuk enkripsi ke-6 (i = 6) adalaha " – " dengan nilai desimal **150.** Hasil enkripsi keseluruhan sampai baris ke-6 dapat dilihat dibawah ini:

| | | |
|---|---|---|
| **W J d m { ^** | 87 74 100 109 123 136 | ⇨ i = 0 |
| **Z M g p ~ (‹)** | 90 77 103 112 126 **139** | ⇨ i = 1 |
| **` S m V (,,)** | 96 83 109 118 **132** | ⇨ i = 2 |
| **i \ v ( ❑)** | 105 92 118 **127** | ⇨ i = 3 |
| **u h (,)** | 117 104 **130** | ⇨ i = 4 |

**„ (w)** 132 **119** ⇨ i = 5

**(-)** **150** ⇨ i = 6

Maka dari hasil proses enkripsi kedua dan ketiga adalah (**– w , □„ ‹**)
Hasil untuk enkripsi kedua, nilai enkripsi yang diambil dari setiap barisnya sebanyak satu karakter yang dimulai dari (i): ke-6. Ke-5, ke4, ke-3, ke-2, ke-1 dengan nilai j ≤ (N + 1) – i.
Ciphertext yang dihasilkan pada proses segitiga 2 merupakan hasil akhir dari Proses enkripsi.

1. Matriks deskripsi segitiga pertma

Ciphertext = **– w , □„ ‹**
Nilai decimal ciphertext = 150 119 130 127 132 139
Kunci yang digunakan = 2 (Bilangan Asli ⇨nteger)
Sesuai dengan panjang ciphertext *N* = 6
Faktor pengali (*fp = R)* berdasarkan nilai
*N* = (deret bilangan asli)⇨ (1, 2, 3, 4, 5, 6,).

Sesuai ciphertext dedeskripsi, setiap karakter terlebih dahulu dirubah ke nilai Desimal sesuai dengan nilai ASCII Mod 25:

| **KARAKTER** ⇨ | **–** | **w** | **,** | **□** | **„** | **‹** |
|---|---|---|---|---|---|---|
| **NILAI DESIMAL** ⇨ | **150** | **119** | **130** | **127** | **132** | **139** |

Lankah selanjutnya adalah melakukan proses deskripsi segitiga pertama
Sesuai dengan rumus:
Diketahui :
**C = – w , □„ ‹**
**K = 3**
**N = 6**
**R = 1, 2, 3, 4, 5, 6,**
Rumus untuk baris pertama (i = 1):
M1j= C [ j ] - (K * R[ 1 ])) mod 256

Maka penyeleseaian untuk baris pertama (i = 1):
M11
= ( – - (3 * 1)) mod 256

= (150 – 3) mod 256 adalah 147 (huruf “ “ ” dalam karakter ASCII 256)

M12
= ( w – (3 * 1)) mod 256

= (119 – 3) mod 256 adalah 116 (huruf “ t ” dalam karakter ASCII 256)

M13
= ( , – (3 * 1)) mod 256

= (130 – 3) mod 256 adalah 127 (huruf “□” dalam karakter ASCII 256)

M14
= (□– (3 * 1)) mod 256

= (127 – 3) mod 256  adalah 124 (huruf “ | ” dalam karakter ASCII 256)
M15

= ( ,, – (3 * 1)) mod 256

= (132 – 3) mod 256 adalah 129 (huruf “□’ dalam karakter ASCII 256)

M16
= ( ‹ – (3 * 1)) mod 256

= (139 – 3) mod 256 adalah 136 (huruf “ ˆ ” dalam karakter ASCII 256)

Hasil deskripsi baris ke-1 (i = 1) (tanpa tanda “  “) adalah “ **t X | X ˆ.** “

147 116 127 124 129 **136.**

hasil dari dekripsi baris ke-1  (i = 1) akan digunakan sebagai ciphertext  pada baris ke-2
≤ ( 6 + 1 ) – 2 j ≤ 5.
M21
= ( M(1)1– 3 * ( 2 )) mod 256.

= ( “ – 6 ) mod 256

= ( 147 – 6 ) mod 256 adalah 141 (huruf “□” dalam karakter ASCII 256)

M22
= ( t – 6 ) mod 256

= ( 116 – 6 ) mod 256  adalah  110 (huruf “ n ” dalam karakter ASCII 256)

M23

= (□ – 6 ) mod 256

= ( 127 – 6 ) mod 256  adalah 121 (huruf " y " dalam karakter ASCII 256)

M24
= ( | – 6 ) mod 256

= ( 124 – 6 ) mod 256  adalah 118 (huruf " v " dalam karakter ASCII 256)

M25
= (□ – 6 ) mod 256

= ( 129 – 6 ) mod 256 adalah 123 (huruf " { " dalam karakter ASCII 256)

Hasil deskripsi ke-2 adalah " **X n y v {** " dengan nilai desimal 141 110 121 118 **123.**
Hasil deskripsi ke-2 (i = 2) akan digunakan sebagai ciphertext pada deskripsi
baris ke-3, j ≤ ( 6 + 1 ) – 3 j ≤ 4.
M31
= ( □ – 9 ) mod 256

= ( 141 – 9 ) mod 256 adalah 132 (huruf "„ " dalam karakter ASCII 256)

M32= ( M
= ( n – 9 ) mod 256

= ( 110 – 9 ) mod 256 adalah 101 (huruf " e " dalam karakter ASCII 256)

M33
= ( y – 9 ) mod 256

= ( 121 – 9 ) mod 256 adalah 112 (huruf " p " dalam karakter ASCII 256)

M34
=( v – 9 ) mod 256

= ( 118 – 9 ) mod 256 adalah 109 (huruf " m " dalam karakter ASCII 256)

Hasil deskripsi baris ke-3 adalah " **„ e p m** " dengan nilai desimal 32 101 112
**109.**

Hasil deskripsi ke-3 (i = 3) akan digunakan sebagai ciphertext pada deskripsi
Pada deskripsi baris ke-4 j ≤ ( 6 + 1 ) – 4 j ≤ 3.
M41
= ( „ – 12 ) mod 256

= ( 132 – 12 ) mod 256 adalah 120 (huruf “ x ” dalam karakter ASCII 256)

M42
= ( e – 12 ) mod 256

= ( 101 – 12 ) mod 256 adalah 89 (huruf “ Y ” dalam karakter ASCII 256)

M43
= ( p – 12 ) mod 256

= ( 112 – 12 ) mod 256 adalah 100 (huruf “ d ” dalam karakter ASCII 256)

Hasil deskripsibaris ke-4 (i = 4) adalah “ **x Y d** “ dengan nilai desimal 120 89 **100.**

Hasil deskripsi ke-4 (i = 4) akan digunakan sebagai ciphertext pada deskripsi
Pada deskripsi baris ke-5, ≤ ( 6 + 1 ) – 5 j ≤ 2.
M51
= ( x – 15 ) mod 256

= ( 120 – 15 ) mod 256 adalah 105 (huruf “ i ” dalam karakter ASCII 256)

M52
= ( Y – 15 ) mod 256

= ( 89 – 15 ) mod 256 adalah 74 (huruf “ J ” dalam karakter ASCII 256)

Hasil deskripsi ke-5 (i = 5) adalah “ **i J** “ dengan nilai desimal 105 **74.**
Hasil deskripsi ke-5 (i = 5) akan digunakan sebagai ciphertext pada deskripsi
Pada deskripsi baris ke-6. j ≤ ( 6 + 1 ) – 6 j ≤ 1.
M61
= ( i – 18 ) mod 256

= ( 105 – 18 ) mod 256 adalah 87 (huruf “ W ” dalam karakter ASCII 256)

Hasil deskripsi baris ke-6 (i = 6) adalah "**WJdm{ ˆ** " dengan nilai desimal **87** 74 100 109 123 136**.**
Hasil dekripsi sampai pada tahap baris ke enam (i = 6) dapat dilihat di bawah

Ini:

**– w , □„ ‹** 150 119 130 127 132 139 ⇨ i = 0

**" t □| □ˆ** 7 116 127 124 129 **136** ⇨ i = 1

**X n y v {** 141 110 121 118 **123** ⇨ i = 2

**„ e p m** 132 101 112 **109** ⇨ i = 3

**x Y d** 120 89 **100** ⇨ i = 4

**i J** 105 **74** ⇨ i = 5

**W** **87** ⇨ i = 6

sehingga pada proses dekripsi segitiga pertama diperoleh plainteks sesuai

dengan formula Mijpada nilai j = (N+i)-i adalah " **W J d m { ˆ** "

Hasil untuk enkripsi kedua, nilai enkripsi yang diambil dari setiap barisnya sebanyak satu karakter yang dimulai dari (i): ke-6. Ke-5, ke4, ke-3, ke-2, ke-1.
2. Matriks deskripsi segitiga kedua

*Ciphertext* = **W J d m { ˆ**
Nilai desimal ciphertext = **87 74 100 109 123 136**
Kunci yang digunakan = 2 (Bilangan Asli ⇨ Integer)

Rumus dekripsi segitiga kedua:

M1j= C [ j ] – (K * R[ 1 ])) mod 256

Maka penyelesaian untuk baris pertama (i = 1):

M11
= (W – (3 * 1)) mod 256

= (W – (3 * 1)) mod 256

= (87 – 3) mod 256 adalah 84 (huruf “ T ” dalam karakter ASCII 256)

M12
= (J – (3 * 1)) mod 256

= (74 – 3) mod 256 adalah 71 (huruf “ G ” dalam karakter ASCII 256)

M13
= (d – (3 * 1)) mod 256

= (100 – 3) mod 256 adalah 97 (huruf “ a ” dalam karakter ASCII 256)

M14
= (m – (3 * 1)) mod 256

= (109 – 3) mod 256 adalah 106 (huruf “ j ” dalam karakter ASCII 256)

M15
= ( { – (3 * 1)) mod 256

= (123 – 3) mod 256 adalah 120 (huruf “ x ” dalam karakter ASCII 256)

M16
= ( ˆ– (3 * 1)) mod 256

= (136 – 3) mod 256 adalah 133 (huruf “... ” dalam karakter ASCII 256)

Hasil deskripsi baris ke-1 ( i = 1) adalah “**T G a j x ...** “ dengan nilai desimal
**84** 71 97 106 120 133.
Hasil deskripsi baris ke-1 (i = 1) akan digunakan sebagai ciphertext pada
Dekripsi baris ke-2, i = 2; j ≥ 2.
M22= (C
= (C[1]2– (3 * 2)) mod 256

= (G – 6) mod 256

= (71 – 6) mod 256 adalaha 65 (huruf “ A ” dalam karakter ASCII 256)

M23

= (a – 6) mod 256

= (97 – 6) mod 256 adalah 91 (huruf “ [ ” dalam karakter ASCII 256)

M24
= ( j – 6) mod 256

= (106 – 6) mod 256 adalah 100 (huruf “ d ” dalam karakter ASCII 256)

M25= (C[2-1]5– (3 * R[ 2 ])) mod 256
= ( x – 6) mod 256

= (120 – 6) mod 256 adalah 114 (huruf “ r ” dalam karakter ASCII 256)

M26
= (... – 6) mod 256

= (133 – 6) mod 256 adalah 127 (huruf “□” dalam karakter ASCII 256)

Hasil deskripsi baris ke-2 (i = 2) adalah “ **A [ d r □**” dengan nilai desimal **65**
91 100 114 127.

Hasil deskripsi baris ke-2 (i = 2) akan digunakan sebagai ciphertext pada

Deskripsi ke-3, i = 3; j ≥ 3:

M33
= (C[2]3– (3 * 3)) mod 256

= ( [ – 9) mod 256

= (91 – 9) mod 256 adalah 82 (huruf “ R ” dalam karakter ASCII 256)

M34

= ( d – 9) mod 256

= (100 – 9) mod 256 adalah 91 (huruf “ [ ” dalam karakter ASCII 256)

M35

= ( r – 9) mod 256

= (114 – 9) mod 256 adalah 105 (huruf “ i ” dalam karakter ASCII 256)

M36
= (□ – 9) mod 256

= (127 – 9) mod 256 adalah 118 (huruf “ v ” dalam karakter ASCII 256)

Hasil deskripsi baris ke-3 (i = 3) adalah “ **R [ i v** “ dengan nilai desimal **82** 91 105 118.

Hasil enkripsi baris ke-3 (i = 3) akan digunakan sebagai ciphertext pada

deskripsi baris ke-4, i = 4; j ≥ 4:

M44
= (C[3]4– (3 * 4)) mod 256

= ( [ – 12) mod 256

= (91 – 12) mod 256 adalah 79 (huruf “ O ” dalam karakter ASCII 256)

M45
= ( i – 12) mod 256

= (105 – 12) mod 256 adalah 93 (huruf “ ] ” dalam karakter ASCII 256)

M46
= ( v – 12) mod 256

= (118 – 12) mod 256 adalah106 (huruf “ j ” dalam karakter ASCII 256)

Hasil deskripsi baris ke-4 (i = 4) adalah “ **O ] j** “ dengan nilai desimal **79** 93 106

Hasil deskripsi baris ke-4 (i = 4) akan digunakan sebagai ciphertext pada
dkripsi baris ke-5, i = 5; j ≥ 5:
M55

= (C[4]5– (3 * 5)) mod 256

= ( ] – 15) mod 256

= (93 – 15) mod 256 adalah 78 (huruf " N " dalam karakter ASCII 256)

M56
= ( j – 15) mod 256

= (106 – 15) mod 256 adalah 91 (huruf " [ " dalam karakter ASCII 256)

Hasil dekripsi baris ke-5 (i = 5) adalah " **N [** " dengan nilai desimal **78** 91.

Hasil dekripsi baris ke-5 (i = 5) akan digunakan sebagai ciphertext pada

dekripsi baris ke-6, i = 6; j ≥ 6;

M56
= (C[5]6– (3 * 6)) mod 256

= ( [ – 18) mod 256

= (91 – 18) mod 256 adalah 73 (huruf " I " dalam karakter ASCII 256)

Hasil untuk dekripsi ke-6 (i = 6) adalah " **TARONi** " dengan nilai desimal **84** 65 82 79 78 73.

Hasil dekripsi keseluruhan dapat dilihat dibawah ini:

**W J d m { ˆ** 87 74 100 109 123 136 ⇨ i = 0

**T G a j x ... 84** 71 97 106 120 133 ⇨ i = 1

**A [ d r □ 65** 91 100 114 127 ⇨ i = 2

**R [ i v 82** 91 105 118 ⇨ i = 3

**O ] j 79** 93 106 ⇨ i = 4

**N[ 78** 91 ⇨ i = 5

**I** **73** ⇨ i = 6

Maka dari hasil proses dekripsi kedua adalah " TARONO " dengan nilai desimal **84** 65 82 79 78 73. Penentuan karakter yang ditetapkan sebagai plainteks (record asli) dilakukan berdasarkan formula Mij pada nilai j = (N+i)-N pada masing-masing baris. Sehingga didapatkan plaintex tadalah TARONI

### III.2.1. Algoritma *RC4*

Algoritma *RC4* mengenkripsi dengan mengombinasikannya dengan *plainteks* dengan menggunakan *bit-wise* Xor (*Exclusive-or*). RC4 menggunakan panjang kunci dari 1 sampai 256 *byte* yang digunakan untuk menginisialisasikan tabel sepanjang 256 *byte*. Tabel ini digunakan untuk generasi yang berikut dari *pseudo random* yang menggunakan XOR dengan *plaintext* untuk menghasilkan *ciphertext*. Masing - masing elemen dalam tabel saling ditukarkan minimal sekali. Proses dekripsinya dilakukan dengan cara yang sama (karena Xor merupakan fungsi simetrik). Untuk menghasilkan *keystream*, *cipher* menggunakan *state* internal yang meliputi dua bagian :

1. Tahap *key scheduling* dimana *state automaton* diberi nilai awal berdasar kan kunci enkripsi. State yang diberi nilai awal berupa *array* yang merepresentasikan suatu permutasi dengan 256 elemen, jadi hasil dari algoritma KSA adalah permutasi awal. *Array* yang mempunyai 256 elemen ini (dengan indeks 0 sampai dengan 255) dinamakan S. Berikut adalah algoritma KSA dalam bentuk *pseudo-code* dimana *key* adalah kunci enkripsi dan *keylength* adalah besar kunci enkripsi dalam *bytes* (untuk kunci 128 bit, *keylength* = 16).

```
for i = 0 to 255
    S [i] := i
    j := 0
for i = 0 to 255
    j := (j + S[i] + key  [I mod keylenght] ) mod 256
    swap (S[i], S[j])
```

2. Tahap *pseudo-random generation* dimana *state automaton* beroperasi dan *output*nya menghasilkan *keystream*. Setiap putaran, bagian *keystream* sebesar 1 *byte* (dengan nilai antara 0 sampai dengan 255) dioutput oleh PRGA berdasarkan *state* S. Berikut adalah algoritma PRGA dalam bentuk *pseudo-code*:

```
i := 0
j := 0
loop
    i :=  ( i + 1 ) mod 256
    j := ( j + S[i] ) mod 256
    swap ( S[i], S[j] )
    output S[ (S[i] + S[j]) mod 256]
```

Setelah terbentuk *keystream*, kemudian *keystream* tersebut dimasukkan dalam operasi XOR dengan *plaintext* yang ada, dengan sebelumnya pesan dipotong-potong terlebih dahulu menjadi *byte-byte*.

1. Inisialisasi Sbox dan Key (Key)

   Algoritma Rc4 sbox array 0....255 mempunyai nilai 0...255 sedangkan key array dalam 0...255mempunyai nilai kode ascii dari setiap karakter string key yang diberikan dan dapat dilihat tabel dibawah ini.

**Tabel III.1. Sbox Dan Key (KSA)**

| inisialisasi sbox dan key(ksa) | | | |
|---|---|---|---|
| iterasi-i | key-char | key[i] | sbox[i] |
| 0 | p | 112 | 0 |
| 1 | o | 111 | 1 |
| 2 | t | 116 | 2 |
| 3 | e | 101 | 3 |
| 4 | n | 110 | 4 |
| 5 | s | 115 | 5 |
| 6 | i | 105 | 6 |
| 7 | p | 112 | 7 |
| 8 | o | 111 | 8 |
| 9 | t | 116 | 9 |
| 10 | e | 101 | 10 |
| 11 | n | 110 | 11 |
| 12 | s | 115 | 12 |
| 13 | i | 105 | 13 |
| 14 | p | 112 | 14 |
| 15 | o | 111 | 15 |
| 16 | t | 116 | 16 |
| 17 | e | 101 | 17 |
| 18 | n | 110 | 18 |
| 19 | s | 115 | 19 |
| 38 | e | 101 | 38 |
| 39 | n | 110 | 39 |
| 40 | s | 115 | 40 |
| 41 | i | 105 | 41 |
| 42 | p | 112 | 42 |
| 43 | o | 111 | 43 |
| 44 | t | 116 | 44 |
| 45 | e | 101 | 45 |
| 46 | n | 110 | 46 |
| 47 | s | 115 | 47 |
| 48 | i | 105 | 48 |
| 49 | p | 112 | 49 |
| 50 | o | 111 | 50 |
| 51 | t | 116 | 51 |
| 52 | e | 101 | 52 |
| 53 | n | 110 | 53 |
| 54 | s | 115 | 54 |
| 55 | i | 105 | 55 |
| 56 | p | 112 | 56 |
| 57 | o | 111 | 57 |

| 20 | i | 105 | 20 |
|---|---|---|---|
| 21 | p | 112 | 21 |
| 22 | o | 111 | 22 |
| 23 | t | 116 | 23 |
| 24 | e | 101 | 24 |
| 25 | n | 110 | 25 |
| 26 | s | 115 | 26 |
| 27 | i | 105 | 27 |
| 28 | p | 112 | 28 |
| 29 | o | 111 | 29 |
| 30 | t | 116 | 30 |
| 31 | e | 101 | 31 |
| 32 | n | 110 | 32 |
| 33 | s | 115 | 33 |
| 34 | i | 105 | 34 |
| 35 | p | 112 | 35 |
| 36 | o | 111 | 36 |
| 37 | t | 116 | 37 |
| 76 | i | 105 | 76 |
| 77 | p | 112 | 77 |
| 78 | o | 111 | 78 |
| 79 | t | 116 | 79 |
| 80 | e | 101 | 80 |
| 81 | n | 110 | 81 |
| 82 | s | 115 | 82 |
| 83 | i | 105 | 83 |
| 84 | p | 112 | 84 |
| 85 | o | 111 | 85 |
| 86 | t | 116 | 86 |
| 87 | e | 101 | 87 |
| 88 | n | 110 | 88 |
| 89 | s | 115 | 89 |
| 90 | i | 105 | 90 |
| 91 | p | 112 | 91 |
| 92 | o | 111 | 92 |
| 93 | t | 116 | 93 |
| 94 | e | 101 | 94 |
| 95 | n | 110 | 95 |
| 96 | s | 115 | 96 |
| 97 | i | 105 | 97 |
| 98 | p | 112 | 98 |

| 58 | t | 116 | 58 |
|---|---|---|---|
| 59 | e | 101 | 59 |
| 60 | n | 110 | 60 |
| 61 | s | 115 | 61 |
| 62 | i | 105 | 62 |
| 63 | p | 112 | 63 |
| 64 | o | 111 | 64 |
| 65 | t | 116 | 65 |
| 66 | e | 101 | 66 |
| 67 | n | 110 | 67 |
| 68 | s | 115 | 68 |
| 69 | i | 105 | 69 |
| 70 | p | 112 | 70 |
| 71 | o | 111 | 71 |
| 72 | t | 116 | 72 |
| 73 | e | 101 | 73 |
| 74 | n | 110 | 74 |
| 75 | s | 115 | 75 |
| 117 | s | 115 | 117 |
| 118 | i | 105 | 118 |
| 119 | p | 112 | 119 |
| 120 | o | 111 | 120 |
| 121 | t | 116 | 121 |
| 122 | e | 101 | 122 |
| 123 | n | 110 | 123 |
| 124 | s | 115 | 124 |
| 125 | i | 105 | 125 |
| 126 | p | 112 | 126 |
| 127 | o | 111 | 127 |
| 128 | t | 116 | 128 |
| 129 | e | 101 | 129 |
| 130 | n | 110 | 130 |
| 131 | s | 115 | 131 |
| 132 | i | 105 | 132 |
| 133 | p | 112 | 133 |
| 134 | o | 111 | 134 |
| 135 | t | 116 | 135 |
| 136 | e | 101 | 136 |
| 137 | n | 110 | 137 |
| 138 | s | 115 | 138 |
| 139 | i | 105 | 139 |

| 99 | o | 111 | 99 |
|---|---|---|---|
| 100 | t | 116 | 100 |
| 101 | e | 101 | 101 |
| 102 | n | 110 | 102 |
| 103 | s | 115 | 103 |
| 104 | i | 105 | 104 |
| 105 | p | 112 | 105 |
| 106 | o | 111 | 106 |
| 107 | t | 116 | 107 |
| 108 | e | 101 | 108 |
| 109 | n | 110 | 109 |
| 110 | s | 115 | 110 |
| 111 | i | 105 | 111 |
| 112 | p | 112 | 112 |
| 113 | o | 111 | 113 |
| 114 | t | 116 | 114 |
| 115 | e | 101 | 115 |
| 116 | n | 110 | 116 |
| 158 | n | 110 | 158 |
| 159 | s | 115 | 159 |
| 160 | i | 105 | 160 |
| 161 | p | 112 | 161 |
| 162 | o | 111 | 162 |
| 163 | t | 116 | 163 |
| 164 | e | 101 | 164 |
| 165 | n | 110 | 165 |
| 166 | s | 115 | 166 |
| 167 | i | 105 | 167 |
| 168 | p | 112 | 168 |
| 169 | o | 111 | 169 |
| 170 | t | 116 | 170 |
| 171 | e | 101 | 171 |
| 172 | n | 110 | 172 |
| 173 | s | 115 | 173 |
| 174 | i | 105 | 174 |
| 175 | p | 112 | 175 |
| 176 | o | 111 | 176 |
| 177 | t | 116 | 177 |
| 178 | e | 101 | 178 |
| 179 | n | 110 | 179 |
| 180 | s | 115 | 180 |

| 140 | p | 112 | 140 |
|---|---|---|---|
| 141 | o | 111 | 141 |
| 142 | t | 116 | 142 |
| 143 | e | 101 | 143 |
| 144 | n | 110 | 144 |
| 145 | s | 115 | 145 |
| 146 | i | 105 | 146 |
| 147 | p | 112 | 147 |
| 148 | o | 111 | 148 |
| 149 | t | 116 | 149 |
| 150 | e | 101 | 150 |
| 151 | n | 110 | 151 |
| 152 | s | 115 | 152 |
| 153 | i | 105 | 153 |
| 154 | p | 112 | 154 |
| 155 | o | 111 | 155 |
| 156 | t | 116 | 156 |
| 157 | e | 101 | 157 |
| 199 | e | 101 | 199 |
| 200 | n | 110 | 200 |
| 201 | s | 115 | 201 |
| 202 | i | 105 | 202 |
| 203 | p | 112 | 203 |
| 204 | o | 111 | 204 |
| 205 | t | 116 | 205 |
| 206 | e | 101 | 206 |
| 207 | n | 110 | 207 |
| 208 | s | 115 | 208 |
| 209 | i | 105 | 209 |
| 210 | p | 112 | 210 |
| 211 | o | 111 | 211 |
| 212 | t | 116 | 212 |
| 213 | e | 101 | 213 |
| 214 | n | 110 | 214 |
| 215 | s | 115 | 215 |
| 216 | i | 105 | 216 |
| 217 | p | 112 | 217 |
| 218 | o | 111 | 218 |
| 219 | t | 116 | 219 |
| 220 | e | 101 | 220 |
| 221 | n | 110 | 221 |

| | | | |
|---|---|---|---|
| 181 | i | 105 | 181 |
| 182 | p | 112 | 182 |
| 183 | o | 111 | 183 |
| 184 | t | 116 | 184 |
| 185 | e | 101 | 185 |
| 186 | n | 110 | 186 |
| 187 | s | 115 | 187 |
| 188 | i | 105 | 188 |
| 189 | p | 112 | 189 |
| 190 | o | 111 | 190 |
| 191 | t | 116 | 191 |
| 192 | e | 101 | 192 |
| 193 | n | 110 | 193 |
| 194 | s | 115 | 194 |
| 195 | i | 105 | 195 |
| 196 | p | 112 | 196 |
| 197 | o | 111 | 197 |
| 198 | t | 116 | 198 |
| 240 | t | 116 | 240 |
| 241 | e | 101 | 241 |
| 242 | n | 110 | 242 |
| 243 | s | 115 | 243 |
| 244 | i | 105 | 244 |
| 245 | p | 112 | 245 |
| 246 | o | 111 | 246 |
| 247 | t | 116 | 247 |
| 248 | e | 101 | 248 |
| 249 | n | 110 | 249 |
| 250 | s | 115 | 250 |
| 251 | i | 105 | 251 |
| 252 | p | 112 | 252 |
| 253 | o | 111 | 253 |
| 254 | t | 116 | 254 |
| 255 | e | 101 | 255 |

| | | | |
|---|---|---|---|
| 222 | s | 115 | 222 |
| 223 | i | 105 | 223 |
| 224 | p | 112 | 224 |
| 225 | o | 111 | 225 |
| 226 | t | 116 | 226 |
| 227 | e | 101 | 227 |
| 228 | n | 110 | 228 |
| 229 | s | 115 | 229 |
| 230 | i | 105 | 230 |
| 231 | p | 112 | 231 |
| 232 | o | 111 | 232 |
| 233 | t | 116 | 233 |
| 234 | e | 101 | 234 |
| 235 | n | 110 | 235 |
| 236 | s | 115 | 236 |
| 237 | i | 105 | 237 |
| 238 | p | 112 | 238 |
| 239 | o | 111 | 239 |

2.Permutasi Sbox (KSA)

Permutasi sbox dilakukan sebanyak 256 iterasi dengan mempertukarkan nilai sbox dalam array i (0...255) dengan nilai sbox dalam array j (j=j+sbox[i]+key[i] mod 256). Hasil nya dapat dilihat

dibawah ini:

**Tabel III.2. Sbox Dan Key (KSA)**

| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 112 | 224 | 86 | 190 | 48 | 168 | 23 | 142 | 5 | 130 | 241 | 106 | 233 | 95 | 221 | 91 |
| 223 | 85 | 213 | 91 | 216 | 93 | 266 | 92 | 217 | 96 | 237 | 113 | 253 | 137 | 27 | 159 |
| 45 | 193 | 76 | 223 | 114 | 11 | 150 | 43 | 198 | 88 | 242 | 136 | 40 | 173 | 73 | 235 |
| 88 | 249 | 154 | 65 | 218 | 125 | 38 | 198 | 110 | 22 | 196 | 100 | 14 | 190 | 101 | 20 |
| 195 | 106 | 17 | 194 | 121 | 39 | 221 | 147 | 79 | 226 | 154 | 88 | 227 | 160 | 93 | 25 |
| 206 | 141 | 82 | 14 | 210 | 82 | 200 | 132 | 61 | 9 | 204 | 79 | 196 | 134 | 73 | 196 |
| 80 | 26 | 236 | 190 | 109 | 16 | 228 | 190 | 143 | 104 | 10 | 233 | 186 | 99 | 14 | 230 |
| 86 | 224 | 120 | 80 | 50 | 26 | 249 | 224 | 115 | 43 | 10 | 243 | 226 | 128 | 110 | 92 |
| 5 | 235 | 98 | 88 | 24 | 13 | 202 | 197 | 81 | 220 | 217 | 205 | 201 | 137 | 4 | 209 |
| 207 | 211 | 206 | 133 | 136 | 145 | 28 | 33 | 44 | 46 | 232 | 242 | 2 | 4 | 16 | 162 |
| 88 | 105 | 247 | 14 | 23 | 42 | 67 | 83 | 200 | 224 | 254 | 14 | 40 | 187 | 210 | 241 |
| 16 | 53 | 76 | 109 | 148 | 178 | 216 | 254 | 42 | 72 | 34 | 181 | 218 | 7 | 221 | 16 |
| 53 | 196 | 122 | 35 | 180 | 170 | 85 | 129 | 244 | 243 | 170 | 229 | 174 | 173 | 164 | 162 |
| 229 | 182 | 212 | 212 | 217 | 80 | 148 | 222 | 253 | 254 | 41 | 120 | 250 | 207 | 25 | 165 |
| 190 | 14 | 254 | `133 | 89 | 156 | 116 | 203 | 132 | 99 | 178 | 161 | 118 | 249 | 87 | 181 |
| 25 | 45 | 54 | 53 | 163 | 8 | 109 | 0 | 93 | 229 | 117 | 69 | 106 | 90 | 190 | |

3. PRGA

Tabel sbox diatas digunakan dalam proses PRGA untuk menghasilkan key stream

Yang selanjutnya di XOR kan dengan palintext proses iterasi terjadi sebanyak panjang dari plaintext hasil key stream permutasi tabel sbox sebagai berikut:

i = (i + 1) mod 256

j = (j + sbox[i]) mod 256

selanjutnya dipermutasikan :

swap ( sbox[i], sbox[j])

key = potensi

plaintext = winder

hasil karakter ascii nya dapat dilihat pada tabel dibawah ini:

**Tabel III.3. PRGA**

| Len | Plaintext | Ascii | Cipher | Ascii |
|---|---|---|---|---|

| Text | | Code | | Char |
|---|---|---|---|---|
| 1 | w | 119 | 65 | A |
| 2 | i | 105 | 238 | î |
| 3 | n | 110 | 130 | , |
| 4 | d | 100 | 151 | — |
| 5 | e | 101 | 20 | □ |
| 6 | r | 82 | 156 | œ |

Hasil dari penyadian/enkripsi pada RC4 terlihat pada proses algoritma key dan algoritma RC4 lebih cepat proses enrkripsinya karena berbasis stream cipher yang melakukan enkripsi one byte at atime.

Dari hasil analisa *input* dan analisa proses pada akhirnya akan menghasilkan *output*/hasil keluaran yang diterima pengguna, dari setiap bentuk *file* yang dimasukkan yang telah dienkripsi dengan menggunakan algoritma *RC4* maupun algoritma *triangle chain* akan diubah kedalam bentuk yang tidak dapat dikenali dan hanya akan dapat dilihat jika hasil yang sudah dieknripsi tersebut dikembalikan kebentuk semula dengan proses dekripsi.

## III.2. Strategi Pemecahan Masalah

Untuk membangun aplikasi enkripsi dan dekripsi *file* teks sesuai penggunaan algoritma *triangle chain*serta *RC4* sebagai media implementasi *file* teks. Beberapa strategi pemecahan masalah dalam perancangan adalah sebagai berikut :

1 *Input* dan *Output* merupakan sebuah teks masukan yang dapat diproses oleh aplikasi.

2 Proses enkripsi dan dekripsi pada uji coba hanya dilakukan pada tiap *file* teks dan bukan *file* lainnya.

3 *Interface* memunggunakan tampilan yang disajikan dalam membaca aplikasi yang telah dienkripsi.

### III.3. Perancangan Sistem

Pada perancangan aplikasi menjelaskan mengenai rancangan dan hal-hal yang dikerjakan serta fitur-fitur yang akan dipakai pada aplikasi tersebut. Hal ini bertujuan untuk menjelaskan tahapan-tahapan yang dikerjakan, prosedur penggunaan, disain tampilan, serta spesifikasi sistem dari segi perangkat lunak maupun perangkas keras yang digunakan dalam proses perancangan.

#### III.3.1. Analisa Kebutuhan fungsional

Dalam kebutuhan fungsional adalah jenis kebutuhan yang berisi untuk melengkapi perancangan. Kebutuhan fungsional juga berisi informasi-informasi apa saja yang harus ada dan dihasilkan. Berikut kebutuhan fungsional yang terdapat pada rancangan aplikasi yang dibangun :

1. Mengimplementasikan penggunaan bahasa pemrograman *java* dalam membuat aplikasi media implementasi pengamanan *file* teks menggunakan algoritma *triangle chain* serta algoritma *RC4*.
2. Aplikasi dapat mengubah isi *file* yang telah diinputkan sebagai pengamanan *file*.
3. *Input* dan *output* berupa *file* teks yang dapat diproses dengan algoritma *triangle chain* serta algoritma *RC4*

#### III.3.1. Analisa Kebutuhan Nonfungsional

Dalam perancangan aplikasi pengamanan file teks, beberapa perangkat yang penulis gunakan agar aplikasi berjalan baik, yaitu sebagai berikut :

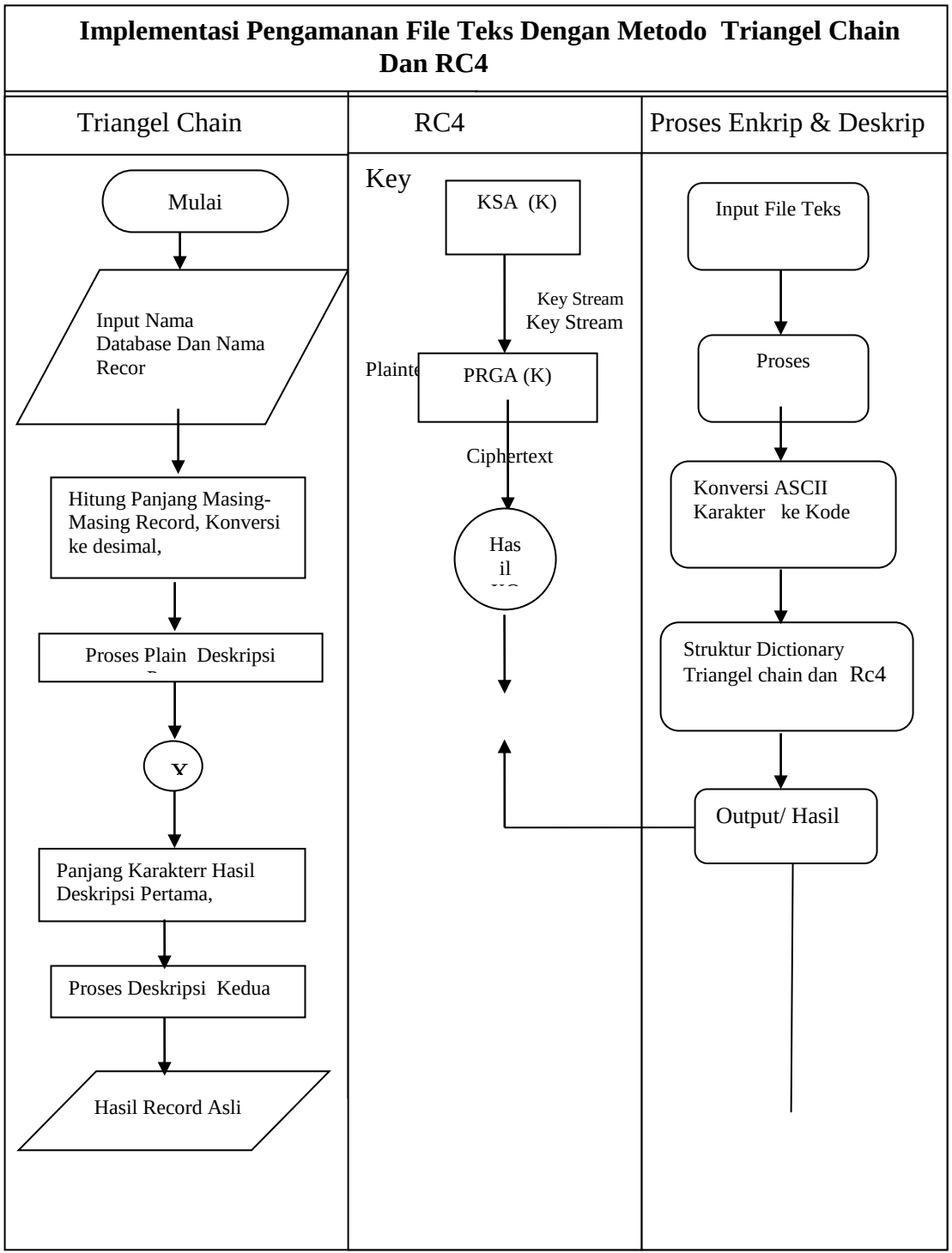
1. Perangkat Lunak (*Software*)
    a. *Operating System Windows Seven*.
    b. *Java* sebagai bahasa program yang digunakan serta *Netbean* sebagi bentuk pengkodean .
2. Perangkat Keras (*Hardware*)

a. Komputer yang setara dengan *Intel pentium Dual Core*.

b. *Mouse, keyboard,* dan *Monitor.*

### III.3.3. *Activity Diagram Blok*

Pada *Activity* Diagram Blok menggambarkan yang menggunakan aplikasi dan perilkau pengguna dapat gambar III.1. berikut.

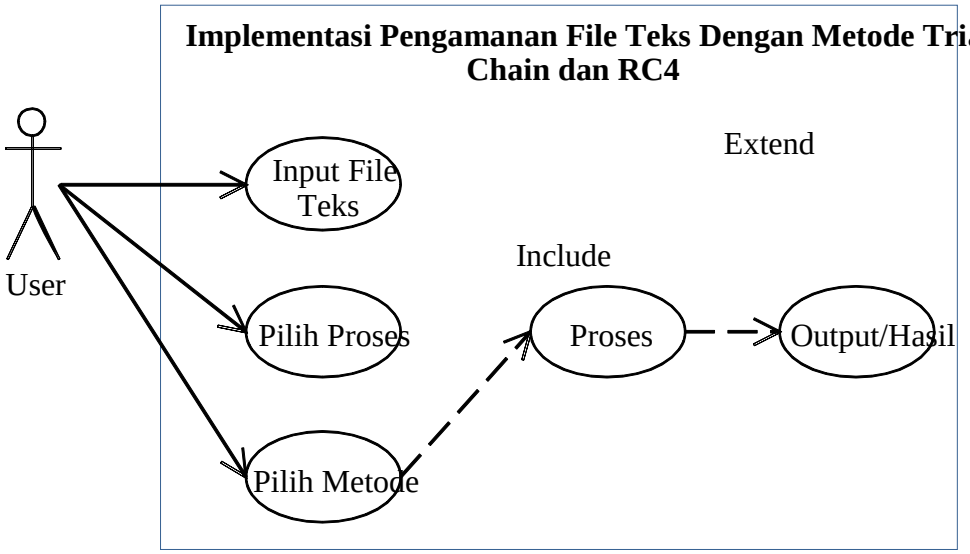


**Gambar III.1. *Activity Diagram Blok***

Kegiatan atau pengguna aplikasi enkripsi file teks, pengguna dapat memahami hasil penelitian ini dan dapat dimanfaatkan khususnya dibidang implemeantasi pengamanan file teks dan melakukan enkripsi terhadap suatu file teks yang sifatnya rahasia, dari hasil algoritma tringel chain dan algoritma RC4.

### III.3.4. *Use Case Diagram*

Pada *Use case* diagram menggambarkan aktor yang menggunakan aplikasi dan perilaku pengguna, dapat dilihat pada gambar III.2 berikut.



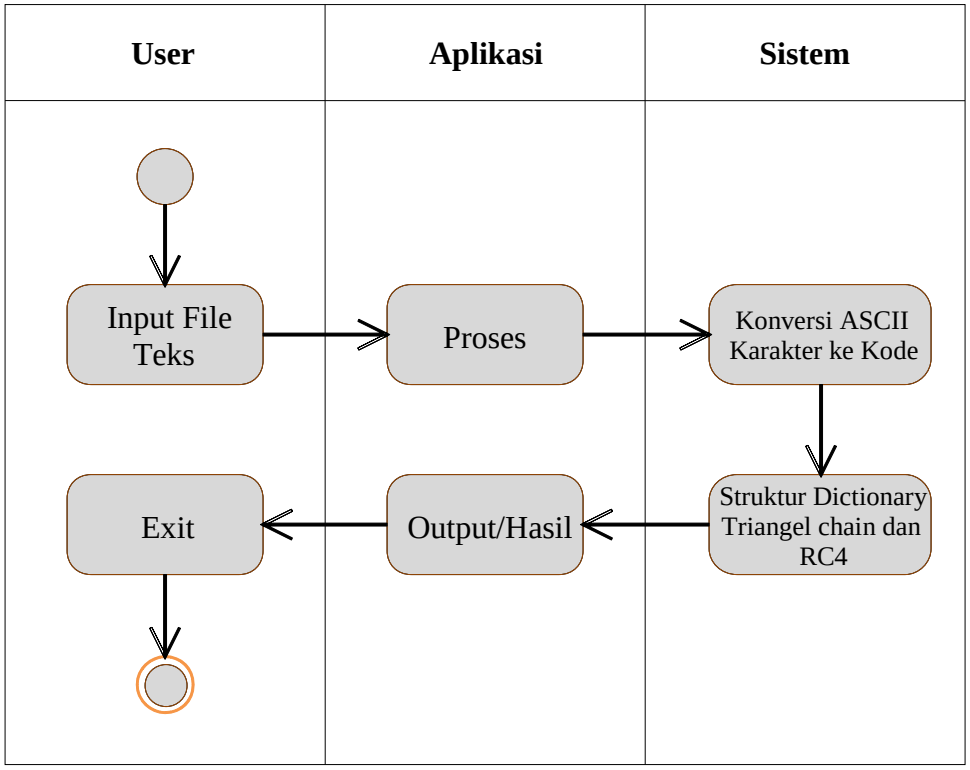

**Gambar III.2. *Use Case Diagram* Pengguna Aplikasi**

Kegiatan aktor atau pengguna pada aplikasi enkripsi *file* teks, pengguna dapat memilih melakukan proses enkripsi *file* teks atau dekripsi *file* teks serta memilih metode yang akan digunakan. Dari proses akan menampilkan media ataupun hasil *output* dari hasil kerja algoritma yang dipilih, baik algoritma *triangle chain* maupun algoritma *RC4.*

### III.3.5. *Activity Diagram* Proses Enkripsi

Pada *Activity diagram* menggambarkan berbagai aliran aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir aktivitas berawal.

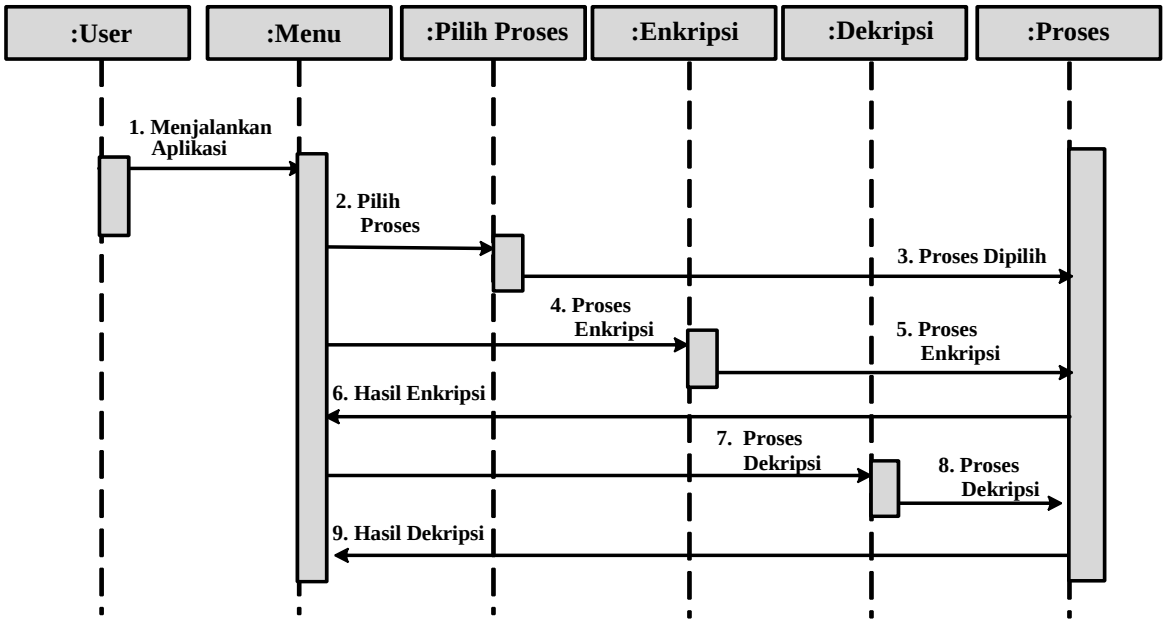
Adapun rancangan diagram aktivitas untuk proses enkripsi dari aplikasi yang dirancang adalah sebagai berikut.




**Gambar III.3. *Activity Diagram* Proses**

### III.3.6. *Sequence* Diagram Proses Enkripsi

Pada *Sequence* diagram proses enkripsi ini menggambarkan kegiatan dari skenario penggunaan aplikasi, *sequence* diagram memilih proses yang terjadi dengan memilih proses enkripsi pada media pengamanan *file* teks dapat dilihat pada gambar III.4 berikut.

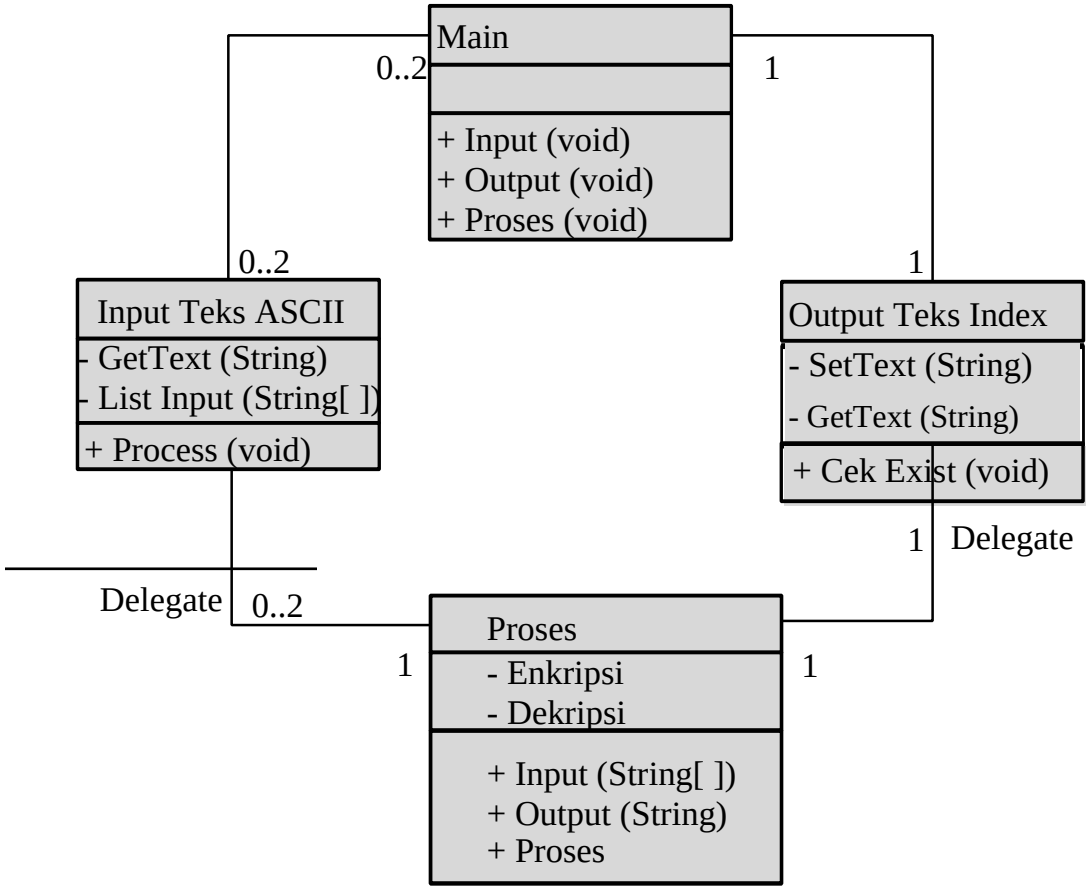



**Gambar III.4. *Sequence Diagram* Proses Enkripsi**

Untuk lebih jelasnya gambar *sequence* diagram proses enkripsi yang terdapat pada gambar III.4 diatas menjelaskan bahwa setelah *user* atau pengguna memulai menjalankan aplikasi sehingga terdapat menu utama dari sistem dengan lanjut berinteraksi melalui pilihan proses yang ada pada menu utama, pengguna memilih proses yang disediakan yaitu proses enkripsi dan serta proses dekripsi. Setelah pilihan proses ditentukan oleh pengguna kembali pada menu utama. Proses enkripsi maupun dekripsi hanya dapat dilakukan setelah pengguna memasukan *file input* yang ingin diproses. Selanjutnya proses menggunakan algoritma *triangle chain* ataupun algoritma *RC4* dapat dilakukan dengan menghasilkan *output* dari algoritma yang ditentukan sebelumnya.

### III.3.7. *Class* Diagram

Pada *Class* diagram perancangan aplikasi ini, dapat dilihat pada gambar III.5 berikut.




**Gambar III.5. *Class* Diagram Sistem Perancangan Aplikasi**

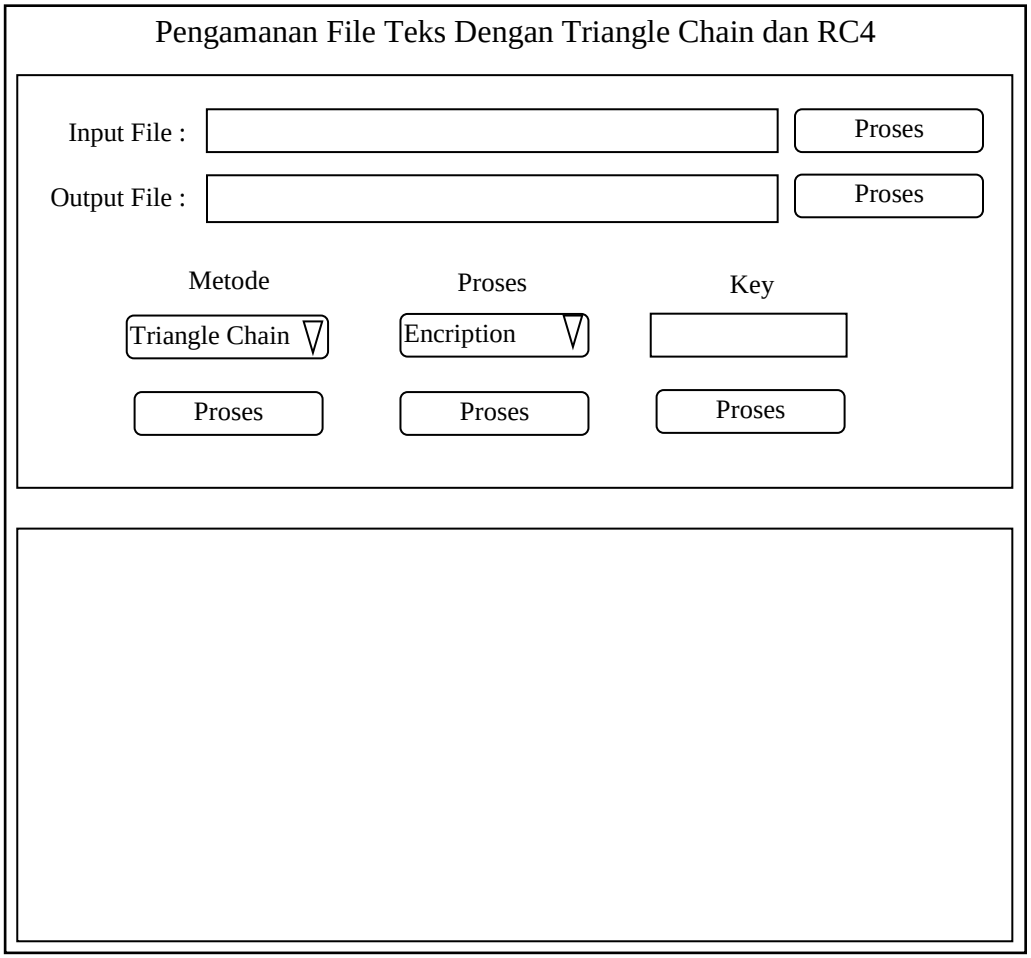
*Class* diagram adalah sebuah *class* yang menggambarkan struktur dan penjelasan *class*, paket, dan objek serta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. *Class* diagram juga menjelaskan hubungan antar *class* dalam sebuah sistem yang sedang dibuat dan bagaimana caranya agar mereka saling berkolaborasi untuk mencapai sebuah tujuan.

## III.4. Perancangan Tampilan

Pada perancangan tampilan aplikasi yang akan dibangun nantinya akan memiliki tampilan yang direncanakan. Adapun rancangan tampilan masing-masing halaman *form* tersebut dapat dijelaskan pada gambar berikut.

### III.4.1. Tampilan *Form* Utama

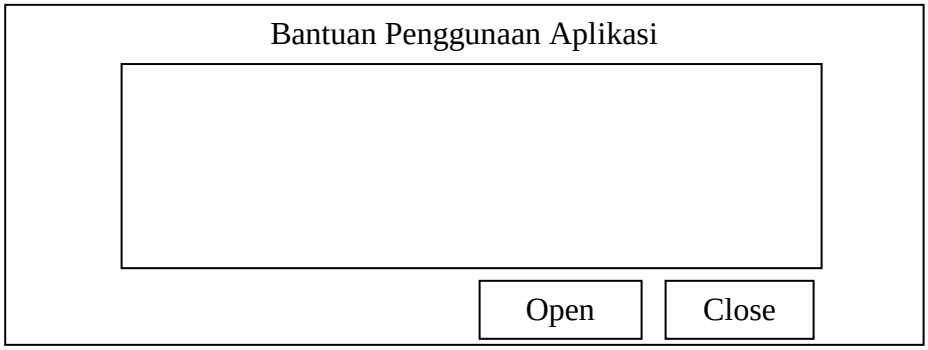
Tampilan *form* utama merupakan tampilan *form* yang fungsi sebagai media proses, didalamnya terdapat *field-field input* dan media tampilan algoritma *RC4*. Adapun rancangan tampilan *form* utama dapat dilihat pada gambar III.6.




**Gambar III.6. Tampilan *Form* Utama**

### III.4.2. Rancangan *Interface Help*

Pada desain tampilan *form help* berfungsi untuk memberikan informasi bantuan penggunaan untuk pengguna. Dapat dilihat pada gambar III.6.




**Gambar III.7. Rancangan *Interface Help***