BAB IV
PEMBAHASAN

Tahun 1985, Koblitz dan Miller mengenalkan kriptografi kurva *elliptic* (*elliptic curve cryptography*) yang menggunakan masalah logaritma diskrit pada titik kurva *elliptic*. Kriptografi kurva *elliptic* dapat digunakan untuk beberapa keperluan seperti skema enkripsi (contohnya ElGamal ECC), tanda tangan digital (contohnya ECDSA) dan protokol pertukaran kunci (contohnya Diffie-Hielman ECC). Dalam penulisan skripsi ini, dibahas tentang ElGamal ECC dan hal-hal yang diperlukan atau berkaitan dengan ElGamal ECC. Kemudian dibuat program yang merupakan implementasi dari ElGamal ECC.

4.1. Kriptografi Kurva *Elliptic*

Stallings [13] mendefinisikan kurva *elliptic* sebagai suatu kurva yang dibentuk oleh persamaan kubik dan memiliki persamaan umum

$$y^2 + Axy + By = x^3 + Cx^2 + Dx + E \tag{4.1}$$

dengan *A,B,C,D* dan *E* adalah konstanta bilangan real. Domain *x* dan *y* adalah bilangan real ($\mathbb{R}$). Dalam penulisan skripsi ini, tidak dibahas mengenai persamaan (4.1). Untuk mendukung tujuan penulisan skripsi, penulis akan menjelaskan bentuk kurva yang lebih sederhana dari persamaan (4.1), yaitu

$$y^2 = x^3 + Ax + B \tag{4.2}$$

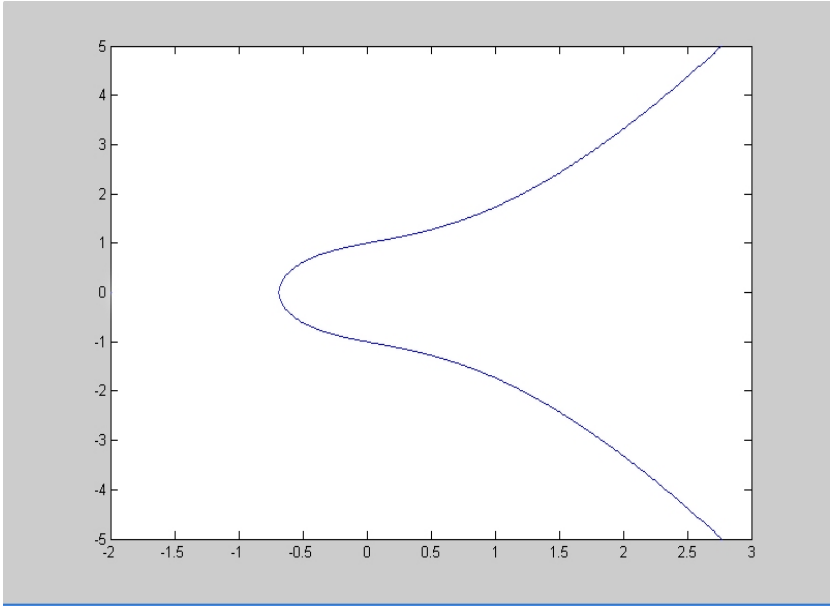
dengan *A* dan *B* dalam $\mathbb{R}$ serta $x, y \in \mathbb{R}$.

Gambar 4.1 merupakan salah satu contoh bentuk geometri dari kurva *elliptic* dengan persamaan $y^2 = x^3 + x + 1$. Setiap kurva *elliptic* akan berbentuk simetris terhadap sumbu *x* atau garis *y*=0. Karena untuk setiap nilai $x \in \mathbb{R}$, terdapat sepasang nilai $y \in \mathbb{R}$ yang memenuhi persamaan (4.2), yaitu

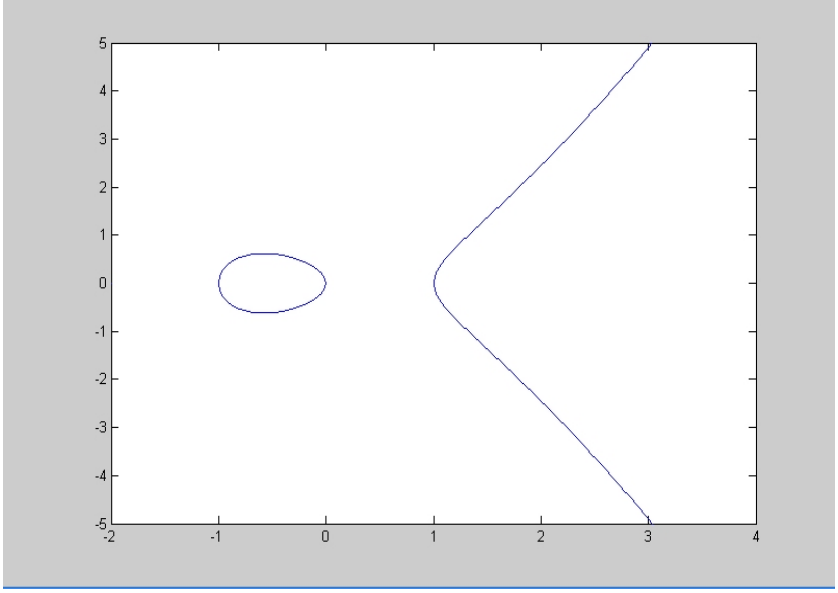
$$y_1 = +\sqrt{x^3 + Ax + B} \text{ dan } y_2 = -\sqrt{x^3 + Ax + B}.$$

Kurva *elliptic* juga dapat dipandang sebagai suatu himpunan yang terdiri dari titik-titik (*x,y*) yang memenuhi persamaan (4.2). Himpunan tersebut dinotasikan dengan *E(A,B).* Untuk setiap nilai *A* dan *B* yang berbeda, dihasilkan

himpunan *E(A,B)* yang berbeda pula. Sebagai contoh, kurva dalam Gambar 4.1 dan Gambar 4.2.



Gambar 4.1. Kurva *Elliptic* $y^2 = x^3 + x + 1$ atau *E(1,1)*



Gambar 4.2. Kurva *Elliptic* $y^2 = x^3 - x$ atau *E(-1,0)*

4.1.1. Kurva Elliptic atas F_p

Ada dua lapangan berhingga yang sering digunakan dalam kriptografi kurva *elliptic*, yaitu lapangan berhingga prima (F_p) dan lapangan karakteristik 2 (F_2^m). Lapangan berhingga F_p lebih efektif untuk implementasi *software* kriptografi kurva *elliptic*. Sedangkan lapangan berhingga F_2^m lebih efektif untuk *hardware* yang sistem kerjanya berdasarkan algoritma kriptografi kurva *elliptic*. Dalam penulisan skripsi ini, hanya dibahas tentang kurva *elliptic E(A,B)* atas F_p.

Berdasarkan Definisi 2.11, lapangan berhingga F_p memiliki p elemen, yaitu {0,1,2,3,...,p-1} dan p adalah bilangan prima. Sebagaimana ditulis oleh Stallings [12], jika p adalah prima maka semua elemen F_p yang tidak nol akan relatif prima terhadap p dan memiliki sebuah invers perkalian modulo p. Sedangkan untuk mencari invers perkalian modulo p dalam F_p, digunakan algoritma *Extended Euclid*.

input : bilangan bulat m dan b
output : B2=b^{-1} (mod m)

1. *(A1, A2, A3)* ⟵ *(1, 0 , m) dan (B1, B2, B3)* ⟵ *(0, 1, b)*
2. *if B3=0 return Tidak memiliki invers. A3=gcd(m,b).*
3. *if B3=1 return B2=b^{-1} (mod m)*
4. $Q = \left\lfloor \frac{A3}{B3} \right\rfloor$
5. *(T1,T2,T3)* ⟵ *(A1-Q*B1, A2-Q*B2, A3-Q.B3)*
6. *(A1,A2,A3)* ⟵ *(B1,B2,B3)*
7. *(B1,B2,B3)* ⟵ *(T1,T2,T3)*

Algoritma 4.1. Algoritma *Extended Euclid*

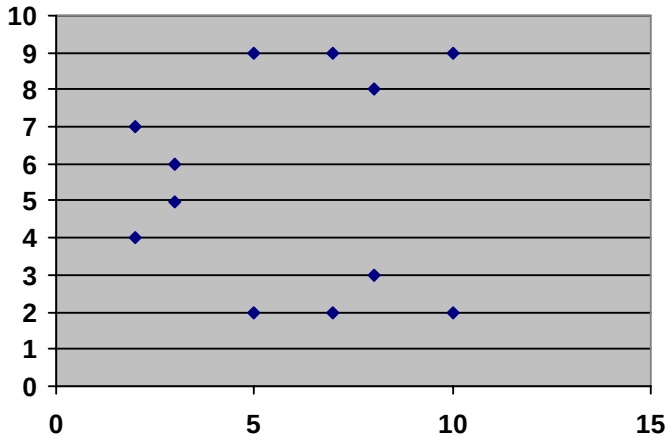
Menurut Menezes *et al* [10], bilangan bulat non negatif d adalah *great common divisor* (gcd) dari bilangan bulat m dan b, dinotasikan dengan $d = \gcd(m,b)$, jika d membagi habis m dan d membagi habis b. Dalam Algoritma 4.1, invers dari b adalah $B2$. Invers tersebut ditemukan saat nilai $B3$=1. Jika selama proses iterasi diperoleh nilai B3=0 maka berarti tidak memiliki invers perkalian modulo p. Contoh eksekusi algoritma *extended euclid* untuk mencari invers 550 dalam F_{1759} terlihat pada Tabel 4.1.

Tabel 4.1. Invers perkalian dari 550 dalam F_{1759}

A1	*A2*	*A3*	*B1*	*B2*	*B3*	*Q*
1	0	1759	0	1	550	3
0	1	550	1	-3	109	5
1	-3	109	-5	16	5	21
-5	16	5	106	-339	4	1
106	-339	4	-111	**355**	1	

Berdasarkan Definisi 2.12 dan 2.13, kurva *elliptic E(A,B)* atas F_p merupakan himpunann penyelesaian dari persamaan $y^2 = x^3 + Ax + B \pmod p$, termasuk titik khusus ***O***. A,B $\in$ F_p adalah konstan, sehingga memenuhi $4A^3+27B^2 \neq 0 \pmod p$. Domain *x* dan *y* adalah F_p.

Kurva *elliptic E(A,B)* atas F_p tidak memiliki representasi geometrik yang berbentuk kurva seperti kurva *elliptic* dalam $\mathbb{R}$. Tetapi dapat digambarkan titik-titik kurva *elliptic* yang merupakan elemen grup *elliptic* $E_p(A,B)$ atas F_p , seperti terlihat pada Gambar 4.3.



Gambar 4.3. *Scatterplot* dari Grup *Elliptic* $E_{11}(1, 1)$

4.1.2. Aritmetika Kurva Elliptic atas F_p

Berdasarkan Definisi 2.15, grup *elliptic* $E_p(A,B)$ merupakan himpunan titik-titik kurva *elliptic*. Sehingga dapat didefinisikan operasi aritmetika dasar dalam grup *elliptic* tersebut. Menurut Stallings [13], aritmetika grup *elliptic* $E_p(A,B)$ atas F_p adalah

Definisi 4.1 [Stallings, 2003:303], *Misalkan $P(x_P,y_P)$ dan $Q(x_Q,y_Q)$ adalah titik kurva elliptic dalam grup elliptic $E_p(A,B)$.* ***O*** *adalah point at infinity dan persamaan kurva elliptic* $y^2 = x^3 + Ax + B \pmod p$*,dengan p prima. Aritmetika dalam grup elliptic $E_p(A,B)$ atas F_p adalah*

1. $P+\boldsymbol{O} = \boldsymbol{O}+P=P$.
2. *Jika $x_Q = x_P$ dan $y_Q = -y_P$, sehingga $P=(x_P,y_P)$ dan $Q=(x_Q,y_Q)=(x_P,-y_P)= -P$, maka $P+Q = P+ (-P) =$* ***O****. Titik Q adalah negatif dari P atau ditulis –P.*
3. *Jika $Q \neq -P$ maka penjumlahan $P+Q = R = (x_R , y_R)$. Nilai x_R dan y_R adalah*

$$x_R = \Delta^2 - x_P - x_Q \pmod p \text{ dan } y_R = \Delta(x_P - x_R) - y_P \pmod p$$

$$\text{dengan } \Delta = \begin{cases} \dfrac{y_Q - y_P}{x_Q - x_P} & \text{, untuk P} \neq \text{Q} \\ \dfrac{3x_P^2 + A}{2y_P} & \text{, untuk P = Q} \end{cases}$$

Misalkan titik $P(3,10)$ dan $Q(9,7)$ dalam $E_{23}(1,1)$. Maka $P+Q=R(x_R,y_R)$, dengan x_R dan y_R diperoleh dengan menghitung nilai Δ terlebih dahulu.

$$\Delta = \frac{y_Q - y_P}{x_Q - x_P} = \frac{7-10}{9-3} \pmod{23} = \frac{-3}{6} \pmod{23} = \frac{-1}{2} \pmod{23} = -2^{-1} \pmod{23}$$

$\Delta = 11$, sehingga dapat dihitung niali x_R dan y_R , yaitu

$$x_R = \Delta^2 - x_P - x_Q = 11^2 - 3 - 9 \pmod{23} = 109 \pmod{23} = 17$$

$$y_R = \Delta(x_P - x_R) - y_P = 11(3-17) - 10 \pmod{23} = -164 \pmod{23} = -3 \pmod{23}$$

$y_R = 20$.

Jadi $P + Q = R(x_R,y_R) = R(17,20)$.

4. *Operasi perkalian didefinisikan sebagai operasi penjumlahan titik-titik yang berulang. Misalnya diberikan bilangan bulat k dan sebuah titik $P(x_P,y_P)$ dalam $E_p(A,B)$. Perkalian skalar k.P adalah penjumlahan terhadap dirinya sendiri sebanyak k kali.*

$$k.P = \underbrace{P + P + P + ... + P}_{\text{sebanyak } k \text{ kali}}$$

Misalkan titk $P(x_P,y_P) = P(3,10) \in E_{23}(1,1)$. Perkalian skalar $2P$ sama dengan penjumlahan titik P sebanyak 2 kali. Hasil perkalian skalar $2P = P+P$ dihitung dengan cara berikut ini.

$$\Delta = \frac{3x_P^2 + A}{2y_P} = \frac{3.3^2 + 1}{2.10} (\text{mod } 23) = \frac{5}{20} (\text{mod } 23) = \frac{1}{4} (\text{mod } 23) = 4^{-1} (\text{mod } 23) = 6,$$

sehingga dapat dihitung nilai x_{2P} dan y_{2P} sebagai berikut

$$x_{2P} = \Delta^2 - x_P - x_P = 6^2 - 3 - 3 \ (\text{mod } 23) = 30 \ (\text{mod } 23) = 7$$

$$y_{2P} = \Delta(x_P - x_{2P}) - y_P = 6 - 7 - 10 \ (\text{mod } 23) = -34 \ (\text{mod } 23) = -11 \ (\text{mod } 23)$$

$$y_{2P} = 12 .$$

Jadi $2P = P+P = (7,12)$.

Banyaknya titik dalam grup elliptic $E_p(A,B)$ dinotasikan dengan #E dan berada pada interval $\left[p+1-2\sqrt{p} \ , \ p+1+2\sqrt{p} \right]$.

Contoh perhitungan dalam menentukan elemen-elemen grup *elliptic* $E_p(A,B)$ atas F_p. Berdasarkan Definisi 2.12 dan 2.13, persamaan kurva *elliptic* adalah $y^2 = x^3 + Ax + B \ (\text{mod } p)$. Misalkan A=1, B=6 dan p=11, persamaan kurva *elliptic* menjadi $y^2 = x^3 + x + 6 \ (\text{mod} 11)$,sehingga $4A^3+27B^2 = 4.1^3+27.6^2 = 976 \ (\text{mod } 11) = 8 \neq 0 \ (\text{mod } 11)$.

Selanjutnya dicari elemen-elemen grup *elliptic* $E_{11}(1,6)$ atas F_p, dengan F_p={0,1,2,3,4,5,6,7,8,9,10}. Sebelum menentukan elemen-elemen $E_{11}(1,6)$, terlebih dahulu mencari *quadratic residue modulo* 11 (QR_{11}) sesuai dengan Definisi 2.14.

Tabel 4.2. Tabel QR_{11}

F_p	y^2 (mod 11)	QR_{11}
0	0^2 (mod 11)	0
1	1^2 (mod 11)	1
2	2^2 (mod 11)	4
3	3^2 (mod 11)	9
4	4^2 (mod 11)	5
5	5^2 (mod 11)	3
6	6^2 (mod 11)	3
7	7^2 (mod 11)	5
8	8^2 (mod 11)	9
9	9^2 (mod 11)	4
10	10^2 (mod 11)	1

Berdasarkan Tabel 4.2, himpunan *quadratic residue modulo* 11 adalah QR_{11}={0,1,3,4,5,9}. Kemudian menentukan elemen grup *elliptic* $E_{11}(1,6)$ yang merupakan himpunan penyelesaian dari persamaan $y^2 = x^3 + x + 6 \pmod{11}$, untuk $x \in F_{11}$ dan $y^2 \in QR_{11}$.

Tabel 4.3. Tabel Untuk Mencari Elemen $E_{11}(1,6)$

$x \in F_{11}$	$y^2=x^3+x+6 \pmod{11}$	$y^2 \in QR_{11}$ = ?	$(x,y)\ x \in E_{11}(1,6)$
0	6	bukan	-
1	8	bukan	-
2	5	ya	(2,4) dan (2,7)
3	3	ya	(3,5) dan (3,6)
4	8	bukan	-
5	4	ya	(5,2) dan (5,9)
6	8	bukan	-
7	4	ya	(7,2) dan (7,9)
8	9	ya	(8,3) dan (8,8)
9	7	bukan	-
10	4	ya	(10,2) dan (10,9)

Berdasarkan Tabel 4.3, untuk x=2, diperoleh $y^2=2^2+2+6 \pmod{11} = 5$. Sehingga diperoleh nilai $y = 4$ dan $y = 7$. Karena berdasarkan Tabel 4.2, $4^2 \pmod{11}=5$ dan $7^2 \pmod{11}=5$. Perhitungan untuk nilai x dan y yang lain, dilakukan dengan cara yang sama. Sehingga didapatkan elemen-elemen grup *elliptic* modulo 11 atas $F_{11,}$ yaitu $E_{11}(1,6)$={ (2,4), (2,7), (3,5), (3,6), (5,2), (5,9), (7,2), (2,9), (8,3), (8,8), (10,2), (10,9), ***O*** }.

4.1.3. Parameter Domain Kurva Elliptic

Dalam subbab ini, dibahas tentang parameter-parameter domain kurva *elliptic* atas F_p . Sebelum mengimplementasikan kriptografi kurva *elliptic*, terlebih dahulu dipersiapkan infrastruktur yang dibutuhkan oleh sistem kriptografi tersebut. Infrastruktur yang dimaksud adalah parameter-parameter domain kurva *elliptic*. Sehingga seluruh pengguna sistem dapat mengetahui beberapa parameter yang akan digunakan bersama. Parameter ini bersifat umum dan boleh diketahui oleh setiap pengguna dalam sistem tersebut.

Definisi 4.2 [Certicom, 200, SEC2:3] *Parameter-parameter domain kurva elliptic atas F_p didefinisikan sebagai six-tuple T.*

$$T = (p, F_p, A, B, G_E, N_G, h)$$

p : *bilangan prima.*

F_p : *lapangan berhingga prima yang memiliki elemen {0,1,2,...,p-1).*

A,B : *koefisien persamaan kurva elliptic* $y^2 = x^3 + Ax + B \pmod p$. $A,B \in F_p$.

G_E : *titik dasar (basic point), yaitu elemen pembangun grup elliptic* $E_p(A,B)$.

N_G : *order dari* G_E, *yaitu bilangan bulat positip terkecil* $\ni N_G . G_E = \boldsymbol{O}$.

h : *kofaktor.* $h = \#E / N_G$, *#E adalah jumlah titik dalam grup elliptic* $E_p(A,B)$.

Kekuatan kriptografi kurva *elliptic* tergantung dari pemilihan parameter-parameter domain yang digunakan. Pemilihan parameter ini dilakukan sedemikian sehingga dapat terhindar dari serangan-serangan terhadap kekuatan algoritma kriptografi kurva *elliptic*. Parameter-parameter tersebut ditentukan secara random menggunakan program yang dibuat sendiri oleh penulis.

Pembaca yang ingin memperoleh parameter-parameter domain kurva *elliptic* tanpa mencarinya terlebih dahulu, dapat menggunakan parameter-parameter yang direkomendasikan oleh *Certicom Research.*

Rekomendasi parameter-parameter domain kurva *elliptic* untuk berbagai ukuran kunci, dapat dilihat secara lengkap pada Certicom [2].

4.2. ElGamal ECC atas F_p

Skema enkripsi ElGamal ECC merupakan pengembangan dari algoritma *generalized ElGamal encryption* yang diterapkan pada aritmetika kurva *elliptic*. Sebelum membahas algoritma ElGamal ECC atas F_p , terlebih dahulu dijelaskan mengenai algoritma *generalized ElGamal encryption.*

4.2.1. Algoritma Generalized ElGamal Encryption atas F_p

Menurut Menezes *et.al* [10], ada 3 algoritma dalam *generalized ElGamal encryption*, yaitu algoritma penentuan kunci, algoritma enkripsi dan algoritma dekripsi.

1. Algoritma penentuan kunci

   Setiap pengguna berhak menentukan *public key* dan *private key* yang akan digunakan bersama. Langkah-langkah yang perlu dilakukan adalah

   a. Menentukan elemen G_α, sedemikian sehingga G_α merupakan elemen pembangun dari grup *G* atas F_p.

   b. Memilih bilangan bulat $V \in [1, n-1]$ secara random, *n* merupakan order dari G_α.

   c. Menghitung $\beta = (G_\alpha)^V$.

   d. β adalah *public key* dan *V* adalah *private key.*

2. Algoritma enkripsi

   Diasumsikan bahwa Bob mengirim *plaintext* yang dienkripsi kepada Iwan dan Bob telah mendapatkan *public key* Iwan, yaitu β. Untuk mengenkripsi *plaintext* diperlukan langkah-langkah sebagai berikut

   a. Merepresentasikan *plaintext* menjadi elemen grup *G*, misalnya $m \in G$.

   b. Memilih bilangan bulat *k* secara random, $k \in [1, n-1]$.

   c. Menghitung $C_1 = (G_\alpha)^k$ dan $C_2 = m.(\beta^k)$.

   d. Mengirim *chipertext* $C_k = (C_1, C_2)$ kepada Iwan.

3. Algoritma dekripsi

   Diasumsikan Iwan menerima *chipertext* $C_k = (C_1, C_2)$ dari Bob. Untuk mendekripsi *chipertext* tersebut, diperlukan langkah-langkah sebagai berikut

   a. Menghitung $(C_1)^{-V}$. *V* adalah *private key* Iwan.

   b. Menghitung $m = C_2 . (C_1)^{-V}$. *m* adalah representasi dari *plaintext.*

   c. Mengkonversi *m* menjadi *plaintext*.

4.2.2. Algoritma ElGamal Elliptic Curve Cryptography (ElGamal ECC)

Berdasarkan Definisi 4.2, parameter-parameter domain kriptografi kurva *elliptic* adalah $T = (p, F_p, A, B, G_E , N_G , h)$, dengan persamaan kurva *elliptic* $y^2 = x^3 + Ax + B \pmod p$. Parameter-parameter tersebut perlu diketahui oleh setiap pengguna dalam suatu *cryptosystem* yang menggunakan algoritma ElGamal

ECC sebagai dasar skema enkripsi. Ada 5 algoritma dalam ElGamal ECC, yaitu

1. Algoritma penentuan kunci

   Setiap pengguna berhak menentukan *public key* dan *private key* yang akan digunakan bersama. Langkah-langkah yang perlu dilakukan adalah

   a. Menentukan bilangan bulat $V \in [1, N_G - 1]$ secara random.

   b. Menghitung $\beta = V. G_E$.

   c. V adalah *private key* dan β adalah *public key*.

2. Algoritma representasi *plaintext* ke titik

   Menurut Cheng [3], algoritma ini diusulkan oleh Koblitz untuk merepresentasikan *plaintext* menjadi titik kurva *elliptic* dalam grup *elliptic* $E_p(A,B)$. Diasumsikan s_j sebagai suatu bilangan bulat dalam F_p dan peluang sebuah bilangan random untuk menjadi bilangan kuadrat adalah ½. Sehingga kemungkinan tidak menemukan sebuah bilangan kuadrat untuk ε percobaan adalah $2^{-\varepsilon}$. Berdasarkan asumsi-asumsi tersebut, langkah-langkah dalam algoritma representasi *plaintext* menjadi titik kurva *elliptic* adalah

   a. Merepresentasikan *plaintext* menjadi bilangan bulat $m > 0 \ni m.\varepsilon < p$.

   b. Asumsikan $x_j = m.\varepsilon + j$, untuk $j \in [0, \varepsilon - 1]$ dan menghitung $s_j = x_j^3 + Ax_j + B$ sampai diperoleh nilai $s_j^{(p-1)/2} = 1 \pmod p$.

   c. Menghitung akar kuadrat dari s_j dan menyimpannya sebagai y_j.

   d. Titik $P_M(x_j , y_j)$ adalah representasi dari *plaintext*.

3. Algoritma enkripsi

   Diasumsikan Bob mengirim *plaintext* ke Iwan. Bob melakukan enkripsi terhadap *plaintext* yang telah direpresentasikan menjadi titik P_M dengan menggunakan *public key* Iwan (β). Langkah-langkah yang perlu dilakukan oleh Bob untuk mengenkripsi *plaintext* adalah

   a. Bob harus mendapatkan *public key* Iwan.

   b. Bob memilih bilangan bulat k secara random, $k \in [1, N_G - 1]$.

   c. Menghitung $P_1 = k.G_E$ dan $P_2 = P_M + k\ \beta$.

   d. Bob mengirim *chipertext pair of points* $P_C = (P_1, P_2)$ kepada Iwan.

4. Algoritma dekripsi

   Diasumsikan Iwan menerima *chipertext pair of points* $P_C = (P_1, P_2)$ dari Bob. Iwan mendekripsi *chipertext* tersebut untuk mendapatkan *plaintext* yang dikirim oleh Bob. Langkah-langkah yang perlu dilakukan adalah

   a. Mengalikan P_1 dengan *private key* Iwan (V) dan menyimpan hasilnya sebagai $M_1 = V.P_1$.
   b. Menghitung $P_2 - M_1$, sehingga diperoleh P_M.
   c. Titik P_M adalah representasi dari *plaintext* yang dikirim oleh Bob.

5. Algoritma representasi titik ke *plaintext*

   Diasumsikan P_M (x_j , y_j) adalah representasi dari *plaintext.* Langkah-langkah untuk mendapatkan *plaintext* tersebut, yaitu

   a. Menghitung $m = \left\lfloor \frac{x_j}{\varepsilon} \right\rfloor$.
   b. Mengubah bilangan bulat *m* menjadi *plaintext*.

4.3. Implementasi ElGamal ECC

Setelah dijelaskan tentang algoritma ElGamal ECC dan berbagai definisi serta dasar-dasar teori yang diperlukan, langkah selanjutnya adalah mengimplementasikannya dalam program komputer. Dalam penulisan skripsi ini, digunakan *software Matlab version 6.1* untuk mengimplementasikan algoritma ElGamal ECC. *Software Matlab* menyediakan berbagai macam *function* untuk melakukan perhitungan atau analisa yang dapat diterapkan dalam berbagai bidang ilmu, termasuk matematika. Seperti *function* yang berkaitan dengan statistik, polynomial, integral, differensial, graf, *artificial intelligence,* simulasi dan lain sebagainya. Selain itu, *programmer* juga dapat membuat program atau *function* sendiri sesuai dengan kebutuhan. *Function* tersebut dapat ditambahkan atau diintegrasikan dalam *Matlab.* Sehingga aplikasi *Matlab* bertambah luas dengan bertambahnya *database* dari *function* yang dibuat oleh *programmer.*

Berdasarkan tujuan penulisan skripsi, penulis akan membuat beberapa *function* yang diperlukan dalam implementasi ElGamal ECC. *Function* tersebut akan ditambahkan dan diintegrasikan langsung dalam *Matlab.* Sehingga *software*

Matlab dapat melakukan operasi aritmetika kurva *elliptic* dan dapat melakukan enkripsi serta dekripsi berdasarkan algoritma ElGamal ECC.

Sebelum mengimplementasikan algoritma ElGamal ECC, perlu ditentukan terlebih dahulu tentang algoritma perkalian skalar kurva *elliptic* yang akan digunakan dalam implementasi tersebut. Karena itu, dalam subbab ini akan dijelaskan tentang algoritma perkalian skalar kurva *elliptic* dan implementasi ElGamal ECC pada *software Matlab.*

4.3.1. Algoritma Perkalian Skalar Kurva Elliptic

Berdasarkan Definisi 4.1, jika diberikan sebuah bilangan bulat k dan titik $P(x_p,y_p)$ dalam $E_p(A,B)$, maka perkalian skalar $k.P$ adalah

$$k.P = \underbrace{P + P + P + ... + P}_{\text{sebanyak } k \text{ kali}}$$

Untuk nilai k yang sangat besar akan membutuhkan waktu yang cukup lama dalam proses perhitungan. Karena itu, diperlukan algoritma perkalian skalar yang lebih efisien dan merupakan representasi dari operasi perkalian skalar kurva *elliptic.*

Sebagaimana dituliskan oleh Doraiswamy, Jainulabudeen dan Kumar [6], algoritma perkalian skalar kurva *elliptic* ada tiga macam.

1. *Binary algorithm*

   Proses perhitungan perkalian skalar kurva *elliptic* ($k.P$) didasarkan pada representasi biner dari k.

   $$k = \sum_{j=0}^{l-1} k_j 2^j$$

   dengan $k_j \in \{0,1\}$ dan l adalah panjang nilai biner k. Sehingga perkalian skalar $k.P$ adalah sebagai berikut

   $$k.P = \sum_{j=0}^{l-1} k_j 2^j P$$

   Contoh untuk nilai k = 11. Nilai biner dari 11 adalah k_j = ’1011’. Berarti l = 4. Sehingga perkalian skalar 11.P adalah

$$11.P = \sum_{j=0}^{l-1} k_j 2^j P = \sum_{j=0}^{3-1} k_j 2^j P = \sum_{j=0}^{2} k_j 2^j P = (1.2^0.P + 1.2^1.P + 0.2^2.P + 1.2^3.P)$$

$$= (1.P + 2.P + 0.P + 8.P) = 11.P\,.$$

2. *Addition-Subtraction algorithm*

   Untuk menghitung perkalian skalar *k.P*, nilai *k* direpresentasikan kedalam bentuk NAF (*Non Adjacent Form*). Metode ini sangat efisien untuk digunakan dalam perhitungan perkalian skalar kurva *elliptic*. Nilai *k* direpresentasikan dalam bentuk sebagai berikut

$$k = \sum_{j=0}^{l-1} u_j 2^j \text{ , dengan } u_j \in \{-1, 0, 1\}\,.$$

   Hasil representasi tersebut akan digunakan untuk perhitungan perkalian skalar kurva *elliptic* dalam *addition-subtraction algorithm.* Algoritma representasi NAF dan *addition-subtraction algorithm* dapat dilihat pada Algoritma 4.2 dan Algoritma 4.3.

   Contoh perhitungan untuk *k* =11, dapat dilihat pada Tabel 4.4 dan Tabel 4.5.

NAF(*k*)

1. u ← 0;
2. c ← k;
3. *l* ← 0;
4. WHILE (c>0)
   - IF (c ganjil)
     - u(*l*) ← 2 – (c mod 4);
     - c ← u(*l*);
   - ELSE u(*l*) ← 0;
   - ENDIF
   - c ← c/2 ;
   - *l* ← *l* + 1 ;
   - ENDWHILE
5. RETURN u;

Algoritma 4.2. Representasi NAF (*Non Adjacent Form*)

ADDITION-SUBTRACTION (k,P*)*

1. u ← NAF(k);
2. R ← ***O***;
3. *l* ← panjang atau banyaknya elemen u;
4. FOR j = *l*-1:-1:0
   R ← R+R;
   IF (u(j) = 1) R ← R + P;
   IF (u(j) = -1) R ← R – P;
   ENDFOR
5. RETURN R;

Algoritma 4.3. *Addition-Subtraction Algorithm*

Tabel 4.4. Representasi NAF dari *k* = 11

Step	Hasil			
1	u = 0			
2	c = 11			
3	l = 0			
4	Iterasi	u	c	*l*
	1	u(0) = -1	(11+1)/2 = 6	1
	2	u(1) = 0	6/2 = 3	2
	3	u(2) = -1	(3+1)/2 = 2	3
	4	u(3) = 0	2/2 = 1	4
	5	u(4) = 1	0/2 = 0	5
5	u = (-1,0,-1,0,1)			

Tabel 4.5. *Addition-Subtraction Algorithm k.P*, untuk *k* = 11

Step	Hasil			
1	u = (-1,0,-1,0,1)			
2	R = ***O***			
3	*l* = 5			
4	Iterasi	j	u	R
	1	4	1	(***O***+***O***) + P
	2	3	0	P+P = 2P
	3	2	-1	(2P+2P) – P = 3P
	4	1	0	3P + 3P = 6P
	5	0	-1	(6P + 6P) – P = 11P
5	R = 11.P			

3. *Repeated-Doubling algorithm*

   Algoritma perkalian skalar ini, khusus digunakan untuk kurva *elliptic* F_{2^m}. Titik kurva *elliptic* $P(x_p,y_p)$ direpresentasikan sebagai $P(x_p, \lambda_p)$.

$$\lambda_p = x_p + \frac{y_p}{x_p}$$

   Berdasarkan batasan masalah, penulisan skripsi ini hanya membahas kurva *elliptic* atas F_p. Bagi pembaca yang tertarik dapat mempelajari *Repeated-Doubling algorithm* yang ditulis oleh Doraiswamy *et.al.* [6].

Menurut Doraiswamy, Jainulabudeen dan Kumar [6], algoritma perkalian skalar kurva *elliptic* atas F_p yang paling efisien adalah *addition-subtraction algorithm.* Dahab dan Lopez [5] juga menyatakan bahwa *addition-subtraction algorithm* memiliki kecepatan 14% di atas *binary algorithm*. Karena itu, algoritma perkalian skalar yang digunakan dalam penulisan skripsi ini adalah *addition-subtraction algorithm.*

4.3.2. Implementasi ElGamal ECC pada Software Matlab

Software yang digunakan untuk implementasi ElGamal ECC adalah *Matlab version 6.1*. Implementasi ini dibagi menjadi 3 bagian program utama, yaitu

1. Program Penentuan Kunci.
2. Program Enkripsi ElGamal ECC.
3. Program Dekripsi ElGamal ECC.

Untuk mencapai tujuan implementasi, diperlukan beberapa *function* yang dapat membangun aplikasi program utama. Sehingga penulisan atau pembuatan ketiga program utama dapat lebih mudah, terstruktur dan hemat memori.

Secara keseluruhan, terdapat 44 *function* yang digunakan dalam implementasi ElGamal ECC, sehingga pembuatan ketiga program utama dapat tercapai. Ada 13 *function* yang telah disediakan dalam *Matlab* dan 31 *function* yang dibuat sendiri oleh penulis. *Function* yang dibuat sendiri oleh penulis, dapat dibagi menjadi 3 kategori, yaitu

1. *Function* aritmetika modulo (7 *function*).
2. *Function* aritmetika kurva *elliptic* (5 *function*).
3. *Function* ElGamal ECC (19 *function*).

Setiap *function* memiliki kegunaan atau fungsi yang berbeda. Kegunaan masing-masing *function* dijelaskan dalam Tabel 4.6, Tabel 4.7, Tabel 4.8 dan Tabel 4.9.

Tabel 4.6. *Function* yang Tersedia dalam *Matlab*

No	Nama *Function*	Kegunaan atau Fungsi
1	length(x)	Untuk menghitung panjang parameter ’x’ (baris atau kolom terbesar dari ’x’).
2	size(x)	Untuk menghitung ukuran dari parameter ’x’ (banyaknya baris dan kolom dari ’x’).
3	floor(x)	Untuk menghitung nilai dari ’x’ yang dibulatkan ke bawah
4	ceil(x)	untuk menghitung nilai dari ’x’ yang dibulatkan ke atas.
5	abs(x)	Untuk menghitung nilai mutlak (absolut) dari ’x’.
6	isprime(p)	Untuk mengetahui apakah ’p’ prima atau bukan. Jika ’p’ prima maka akan mengembalikan nilai 1 dan 0 jika ’p’ bukan bilangan prima.
7	mod(x,p)	Untuk menghitung nilai dari ’x mod p’.
8	randint(m,n,[bb ba])	Untuk membangkitkan bilangan bulat secara random dalam interval [bb,ba] sebanyak m *x* n (berbentuk matrik m *x* n).
9	uint8(x)	Untuk mengubah karakter ’x’ (kode ASCII) menjadi bilangan bulat tak bertanda (unsigned integer) berukuran 8 bit.
10	char(x)	untuk mengubah bilangan bulat ’x’ menjadi karakter dalam kode ASCII atau mencari banyaknya karakter dalam string ’x’.
11	isnumeric(x)	Untuk mengetahui apakah ’x’ merupakan nilai numerik atau bukan. Jika benar maka dihasilkan nilai 1 dan jika salah maka dihasilkan nilai 0.
12	num2str(x)	Untuk mengubah nilai numerik ’x’ menjadi string ’x’.
13	str2num(x)	Untuk mengubah string ’x’ menjadi numerik ’x’.

Tabel 4.7. *Function* Aritmetika Modulo

No	Nama File	Nama *Function*	Kegunaan atau Fungsi
1	eccfadd.m	eccfadd(a,b,p)	Untuk menghitung ’(a+b) mod p’.
2	eccnaf.m	eccnaf(k)	Untuk merepresentasikan bilangan bulat ’k’ dalam bentuk NAF (*Non Adjacent Form*).
3	eccfkali.m	eccfkali(k,a,p)	Untuk menghitung ’(ka) mod p’.
4	eccfinv.m	eccfinv(c,p)	Untuk menghitung invers dari ’c’ dalam modulo ’p’ atau c^{-1} mod p.
5	eccfbagi.m	eccfbagi(a,b,p)	Untuk menghitung ’(a/b) mod p’.

Lanjutan Tabel 4.7.

No	Nama File	Nama *Function*	Kegunaan atau Fungsi
6	eccfpangkat.m	eccfpangkat(a,k,p)	Untuk menghitung '(a^k) mod p'.
7	eccfakar.m	eccfakar(z,p)	untuk menghitung akar 'z' dalam modulo 'p' atau '($z^{1/2}$) mod p'.

Tabel 4.8. *Function* Aritmetika Kurva *Elliptic*

No	Nama File	Nama *Function*	Kegunaan atau Fungsi
1	eccfy2.m	eccfy2(p,A,B,x)	Untuk menghitung '(x^3+Ax+B) mod p' .Persamaan kurva *elliptic* $y^2 = (x^3+Ax+B) \bmod p$.
2	eccadd.m	eccadd(p,A,PP,PQ)	Untuk menghitung penjumlahan titik 'PP+PQ'. PP dan PQ adalah titik kurva *elliptic E(A,B)* atas F_p.
3	eccneg.m	eccneg(p,PP)	Untuk menghitung '-PP'. PP adalah titik kurva *elliptic E(A,B)* atas F_p.
4	eccsub.m	eccsub(p,A,PP,PQ)	Untuk menghitung 'PP-PQ'. PP dan PQ adalah titik kurva *elliptic E(A,B)* atas F_p.
5	eccaddsub.m	eccaddsub(p,A,k,PP)	Untuk menghitung perkalian skalar 'k' dengan titik kurva *elliptic* 'PP' menggunakan *addition-subtraction algorithm*. PP adalah titik kurva *elliptic E(A,B)* atas F_p dan 'k' elemen F_p.

Tabel 4.9. *Function* ElGamal ECC

No	Nama File	Nama *Function*	Kegunaan atau Fungsi
1	bin2des.m	bin2des(b)	Untuk mengubah nilai biner 'b' menjadi nilai desimal.
2	des2bin.m	des2bin(d)	Untuk mengubah nilai desimal 'd' menjadi nilai biner.
3	des2bindig.m	des2bindig(d,dig)	Untuk mengubah nilai desimal 'd' menjadi nilai biner 'dig' bit.
4	ecckunci.m	ecckunci(n)	Untuk menghitung batas bawah dan batas atas dari bilangan bulat dengan panjang kunci 'n' bit.
5	eccprima.m	eccprima(bb,ba)	Untuk menentukan atau membangkitkan bilangan prima dalam interval [bb,ba].
6	ecckurv.m	ecckurv(p)	Untuk menentukan koefisien 'A dan B' dari persamaan kurva *elliptic* $y^2 = (x^3+Ax+B) \bmod p$. Sehingga memenuhi syarat $4A^3+27B^2 \neq 0 \pmod p$.

Lanjutan Tabel 4.9.

No	Nama File	Nama *Function*	Kegunaan atau Fungsi
7	eccordgrup.m	eccordgrup(p,A,B)	Untuk mencari order dari grup *elliptic* $E_p(A,B)$ atas F_p dan Persamaan kurva *elliptic*nya adalah $y^2 = (x^3+Ax+B) \bmod p$.
8	eccpoint.m	eccpoint(p,A,B)	Untuk membangkitkan sebuah titik kurva *elliptic* dalam grup *elliptic* $E_p(A,B)$ atas F_p.
9	eccordpoint.m	eccordpoint(p,A,G)	Untuk mencari order dari titik 'G'
10	eccbasic.m	eccbasic(p,A,B,N_E)	Untuk mencari *basic point* dan order dari *basic point*. N_E adalah order dari grup *elliptic* $E_p(A,B)$ atas F_p.
11	eccparameter.m	eccparameter(nkunci)	Untuk menentukan parameter domain kurva *elliptic* (p,A,B,G_E,N_G,h) dengan panjang kunci 'nkunci' bit.
12	eccprivkey.m	eccprivkey(nkunci,N_G)	Untuk menentukan *private key* dengan panjang kunci 'nkunci' bit dan 'N_G'adalah order dari *basic point*.
13	eccpubkey.m	eccpubkey(p,A,V,G_E)	Untuk menghitung *public key* dengan *private key* 'V' dan *basic point* 'G_E'.
14	eccplain2num.m	eccplain2num(s)	Untuk merepresentasikan string (*plaintext*) 's' menjadi nilai numerik.
15	eccnum2titik.m	eccnum2titik(p,A,B,m,e)	Untuk merepresentasikan bilangan bulat 'm' menjadi titik kurva *elliptic*. Banyaknya percobaan representasi titik adalah 'e'.
16	eccenk.m	eccenk(p,A,G_E,N_G,PB,PM)	Untuk mengenkripsi sebuah titik 'PM' dengan *public key* 'PB' menggunakan algoritma ElGamal ECC. Dan parameter-parameter domain kurva *elliptic*nya adalah (p,A,B,G_E,N_G,h).
17	eccdek.m	eccdek(p,A,V,PC)	Untuk mendekripsi sebuah *chipertext pair of points* 'PC' dengan *private key* 'V' menggunakan algoritma ElGamal ECC.
18	ecctitik2num.m	ecctitik2num(PM,e)	Untuk mengubah titik 'PM' menjadi nilai numerik. Banyaknya percobaan representasi titik adalah 'e'.
19	eccnum2plain.m	eccnum2plain(m)	Untuk mengubah nilai numerik 'm' menjadi *plaintext*.

Setelah dijelaskan tentang algoritma perkalian skalar dan *function* yang akan digunakan, selanjutnya dijelaskan tentang program utama dari implementasi ElGamal ECC. Aplikasi program utamanya adalah program penentuan kunci, program enkripsi ElGamal ECC dan program dekripsi ElGamal ECC.

4.3.2.1. Program Penentuan Kunci

Program penentuan kunci ini akan memanggil beberapa *function* yang diperlukan untuk menghasilkan *private key* dan *public key*. Setelah mendapatkan input panjang kunci, program akan memanggil *function* 'eccparameter', sehingga dihasilkan parameter-parameter domain dari ElGamal ECC, yaitu $T=(p,A,B,G_E,N_G,h)$. Selanjutnya, program akan memanggil *function* 'eccprivkey' dan 'eccpubkey' untuk menentukan *private key* dan menghitung *public key*. Ada 3 hasil yang diperoleh dari program ini dan akan digunakan sebagai input dalam program enkripsi dan dekripsi ElGamal ECC. Hasil dari program penentuan kunci adalah parameter-parameter domain ElGamal ECC, *private key* dan *public key*. Untuk memahami cara kerja program penentuan kunci, perhatikan Algoritma 4.4, Algoritma 4.5, Algoritma 4.6 dan Algoritma 4.7.

Function T=eccparameter(nkunci)

1. [bb ba] ← ecckunci(nkunci);
2. pi ← 1;
3. p ← eccprima(bb,ba);
4. ABi ← 1;
5. [A B]← ecckurv(p);
6. N_E ← eccordgrup(p,A,B);
7. NGi ← 1;
8. [N_G G_E] ← eccbasic(p,A,B,N_E);
9. IF (N_G<N_E)
   NGi ← NGi+1; Ulangi Langkah 8;
   ENDIF

Algoritma 4.4. *Function* Untuk Mencari Parameter Domain Kurva *Elliptic*

Lanjutan Algoritma 4.4.

Function T=eccparameter(nkunci)

10. IF (NGi>N_E)
    ABi ← ABi+1; Ulangi Langkah 5;
    ENDIF
11. IF (ABi > (p-1)*(p-1))
    pi ← pi+1; Ulangi Langkah 3;
    ENDIF
12. IF (pi > (ba-bb))
    nkunci ← input panjang kunci;
    Ulangi Langkah 1;
    ENDIF
13. h ← N_E / N_G ;
14. T ← [p,A,B, G,N_G,h];
15. RETURN T ;

Function V=eccprivkey(nkunci,N_G)

1. [bb ba] ← ecckunci(nkunci);
2. V ← input bilangan dalam interval [1,N_G-1]; atau
   V ← randint(1,1, [1 N_G-1] ;
3. RETURN V;

Algoritma 4.5. *Function* Untuk Menentukan *Private Key*

Function PB=eccpubkey(p,A,V,G_E)

1. PB ← eccaddsub(p,A,V, G_E) ;
2. RETURN PB ;

Algoritma 4.6. *Function* Untuk Menghitung *Public Key*

Genkunci.m

1. nkunci ← input panjang kunci;
2. T ← eccparameter(nkunci);
3. p ← T{1};
4. A ← T{2};
5. B ← T{3};
6. G_E ← T{4};
7. N_G ← T{5};
8. h ← T{6};
9. V ← eccprivkey(nkunci,N_G);
10. PB ← eccpubkey(p,A,V,G_E);
11. RETURN p,A,B,G_E,N_G,h,V,PB;

Algoritma 4.7. Program Penentuan Kunci

4.3.2.2. Program Enkripsi ElGamal ECC

Program enkripsi ini akan membutuhkan input yang dihasilkan dari program penentuan kunci, yaitu parameter-parameter domain ElGamal ECC (*p,A,B,G_E,N_G,h*) dan *public key* serta banyaknya representasi percobaan titik. Kemudian program akan meminta input *plaintext* yang akan dienkripsi. *Plaintext* tersebut akan dipotong untuk setiap $\left\lceil \frac{nkunci}{8} - 1 \right\rceil$ karakter. Setiap potongan *plaintex*t akan direpresentasikan menjadi bilangan bulat dengan memanggil *function* 'eccplain2num'. Kemudian program akan memanggil *function* 'eccnum2titik' untuk merepresentasikan bilangan bulat tersebut menjadi titik kurva *elliptic*. Selanjutnya mengenkripsi titik tersebut menggunakan *function* 'eccenk' sehingga dihasilkan *chipertext pair of points*.

Hasil dari program enkripsi ElGamal ECC adalah *chipertext pair of points* yang akan dikirimkan kepada penerima pesan. Algoritma program enkripsi ElGamal ECC dapat dilihat pada Algoritma 4.11. Selain itu, perlu diperhatikan

juga Algoritma 4.8, Algoritma 4.9 dan Algoritma 4.10 untuk mempermudah dalam memahami cara kerja program enkripsi ElGamal ECC.

```
Function num=eccplain2num(s)
1. FOR j =1:1:length(s)
        s2int ← uint8(s(j));
        s2i ← double(s2int);
        sbin{j} ← des2bindig(s2i , 8);
   ENDFOR
2. c2 ← 0;
3. FOR c=1:1:length(sbin)
        FOR c1=1:1:8
            c2 ← c2+1;
            cbin(c2) ← sbin{c}(c1);
        ENDFOR
   ENDFOR
4. num ← bin2des(cbin);
5. RETURN num ;
```

Algoritma 4.8. *Function* Representasi *Plaintext* Menjadi Nilai Numerik

```
Function PM=eccnum2titik(p,A,B,m,e)
1. IF ( (m*e > p) | (m<0) | (e<1) )
       e ← input Banyaknya percobaan representasi titik;
   ENDIF
2. x ← eccfkali(m,e,p);
3. j ← randint(1,1, [0  e-1] );
4. xj ← eccfadd(x,j,p);
5. sj ← eccfy2(p,A,B,xj);
6. akar=eccfakar(sj,p);
```

Algoritma 4.9. *Function* Representasi Numerik Menjadi Titik

Lanjutan Algoritma 4.9.

Function PM=eccnum2titik(p,A,B,m,e)

7. IF ((akar = []) | (xj = 0)) Ulangi Langkah 3 ;
8. PM ← [xj , akar(1)] ;
9. RETURN PM ;

Function PC=eccenk(p,A,G_E,N_G,PB,PM)

1. K ← input bilangan bulat dalam interval [1 , N_G-1]; Atau
   K ← randint(1,1, [1 N_G-1] ;
2. P1 ← eccaddsub(p,A,K,G_E) ;
3. P21 ← eccaddsub(p,A,K,PB) ;
4. P2 ← eccadd(p,A,PM,P21) ;
5. PC ← [P1(1) , P1(2) , P2(1) , P2(2)] ;
6. RETURN PC ;

Algoritma 4.10. *Function* Enkripsi ElGamal ECC Untuk Satu Titik

Enkripsi.m

1. p ← input bilangan prima;
2. A ← input konstanta A untuk persamaan $y^2 = (x^3 + Ax + b) \bmod p$;
3. B ← input konstanta B untuk persamaan $y^2 = (x^3 + Ax + b) \bmod p$;
4. G_E ← input *basic point;*
5. N_G ← input order dari *basic point;*
6. e ← input banyaknya percobaan representasi titik;
7. PB ← input *public key;*
8. pesan ← input *plaintext;*
9. IF (isnumeric(pesan) = 1) pesan ← num2str(pesan);
10. lpesan ← length(pesan);
11. nkunci ← length(des2bin(p));
12. bpesan ← $\left\lceil \frac{\text{nkunci}}{8} - 1 \right\rceil$;

Algoritma 4.11. Program Enkripsi ElGamal ECC

Lanjutan Algoritma 4.11.

Enkripsi.m

13. ipesan ← $\left\lceil \frac{\text{lpesan}}{\text{bpesan}} \right\rceil$;
14. akhir ← 0;
15. IF (lpesan>bpesan)
    FOR ips=1:1:ipesan
        IF (ips<ipesan)
            awal ← akhir+1;
            akhir ← ips * bpesan;
            plain ← pesan(awal:akhir);
        ELSE
            plain ← pesan(akhir+1:lpesan);
        ENDIF
        p2n ← eccplain2num(plain);
        PM ← eccnum2titik(p,A,B,p2n,e);
        n2t{ips} ← PM;
    ENDFOR
    FOR (nti=1:1:length(n2t)
        PC{nti} ← eccenk(p,A,G_E,N_G,PB,n2t{nti});
    ENDFOR
  ENDIF
16. IF (lpesan<=bpesan)
    p2n ← eccplain2num(pesan);
    n2t ← eccnum2titik(p,A,B,p2n,e);
    PC ← eccenk(p,A,G_E,N_G,PB,n2t);
  ENDIF
17. RETURN e,PC;

4.3.2.3. Program Dekripsi ElGamal ECC

Program ini akan melakukan dekripsi dari *chipertext pair of points*. Selain parameter-parameter domain dari ElGamal ECC, penerima juga memerlukan

private key untuk melakukan dekripsi. Untuk mendekripsi *chipertext pair of points*, program akan memanggil *function* 'eccdek'. Hasilnya akan diubah menjadi *plaintext* dengan memanggil *function* 'ecctitik2num' dan 'eccnum2plain'. Untuk memahami cara kerja program dekripsi ElGamal ECC, perhatikan Algoritma 4.12, Algoritma 4.13, Algoritma 4.14 dan Algoritma 4.15.

Function PM=eccdek(p,A,V,PC)

1. C{1} ← PC(1 : 2) ;
2. C{2} ← PC(3 : 4) ;
3. M1 ← eccaddsub(p,A,C,C{1}) ;
4. PM ← eccsub(p,A,C{2},M1) ;
5. RETURN PM ;

Algoritma 4.12. *Function* Dekripsi ElGamal ECC (Satu *Chipertext Pair of Points*)

Function num=ecctitik2num(PM,e)

1. x ← PM(1) ;
2. num ← $\left\lfloor PM(1) / e \right\rfloor$;
3. RETURN num ;

Algoritma 4.13. *Function* Representasi Titik Menjadi Nilai Numerik

Function psn = eccnum2plain(m)

1. mbin ← des2bin(m) ;
2. mblen ← length(mbin) ;
3. IF (mod(mblen , 8) = 0) mlen ← mblen / 8 ;
4. IF (mod(mblen , 8) ≠ 0) mlen ← $\left\lfloor mblen / 8 \right\rfloor$ + 1 ;
5. i ← 0 ;
6. FOR m1= mlen : -1 :1

   maxps ← mblen – (8 * i) ;

Algoritma 4.14. *Function* Representasi Nilai Numerik Menjadi *Plaintext*

Lanjutan Algoritma 4.14.

Function psn = eccnum2plain(m)

IF (m1 > 1) minps ← maxps – 7 ;
ELSE minps ← 1 ;
ENDIF
psbin ← mbin(minps : maxps) ;
psn{1}(m1) ← bin2des(psbin) ;
i ← i+1 ;
ENDFOR
7. RETURN psn ;

Dekripsi.m

1. p ← input bilangan prima;
2. A ← input konstanta A untuk persamaan $y^2 = (x^3 + Ax + b) \bmod p$;
3. V ← input *private key;*
4. e ← input banyaknya percobaan representasi titik;
5. PC ← input *chipertext pair of points;*
6. PCs ← size(PC);
7. lps ← 1;
8. FOR dek:1:1:PCs(1)
   PM ← eccdek(p,A,V,PC(dek , :);
   t2n ← ecctitik2num(PM,e);
   n2p ← eccnum2plain(t2n);
   pesan (lps : lps + length(char(n2p) – 1)) ← char (n2p);
   lps ← length(pesan);
   ENDFOR
9. RETURN pesan;

Algoritma 4.15. Program Dekripsi ElGamal ECC

4.3.3. Analisa Waktu dan Hasil Implementasi ElGamal ECC

Sebagaimana dijelaskan dalam subbab sebelumnya bahwa implementasi ElGamal ECC menggunakan *software Matlab Student Edition Version 6.1*. Aplikasi tersebut berjalan pada sistem operasi ***Windows XP Professional Version 2002*** dan menggunakan ***Processor*** *Intel Pentium IV* 2.4 GHz, dan ***RAM*** 256 MB *PC2100*. Analisa waktu implementasi dilakukan untuk mengetahui performa waktu setiap proses perhitungan dalam implementasi ElGamal ECC. Analisa waktu tersebut dibagi dalam tiga kategori, yaitu waktu yang dibutuhkan dalam proses aritmetika kurva *elliptic*, proses representasi *plaintext* dan proses enkripsi serta dekripsi ElGamal ECC.

4.3.3.1. Analisa Waktu Aritmetika Kurva Elliptic

Pengujian dilakukan untuk melihat waktu yang dibutuhkan masing-masing operasi aritmetika kurva *elliptic*. Operasi tersebut meliputi operasi penjumlahan, operasi pengurangan dan operasi perkalian skalar kurva *elliptic*. Hasil implementasi terhadap waktu yang dibutuhkan masing-masing operasi aritmetika kurva *elliptic* ditunjukkan dalam Tabel 4.10.

Tabel 4.10. Waktu Untuk Operasi Aritmetika Kurva *Elliptic*

Operasi	Kunci	Perulangan	Waktu
Penjumlahan	32 bit	1000	4 – 5 detik
Pengurangan	32 bit	1000	4 – 5 detik
Perkalian	32 bit	1000	178 – 180 detik
	24 bit	1000	109 – 110 detik
	16 bit	1000	54 – 55 detik
	8 bit	1000	10 – 11 detik

Hasil pengujian pada Tabel 4.10 menunjukkan bahwa operasi perkalian merupakan operasi yang membutuhkan waktu paling banyak dibandingkan operasi aritmetika kurva *elliptic* yang lain. Operasi perkalian skalar kurva *elliptic* membutuhkan waktu sekitar 36 kali lebih banyak dibandingkan dengan operasi penjumlahan dan pengurangan. Sedangkan waktu yang dibutuhkan untuk operasi penjumlahan dan pengurangan kurva *elliptic* tidak jauh berbeda.

Berdasarkan Tabel 4.10, dalam interval waktu [4,5] detik, program mampu

melakukan pehitungan operasi penjumlahan atau pengurangan aritmetika kurva *elliptic* 32 bit sebanyak 1000 kali. Sehingga untuk 1 kali operasi penjumlahan atau pengurangan dengan panjang kunci 32 bit hanya dibutuhkan waktu sekitar 0.004 sampai 0.005 detik. Sebagaimana dituliskan dalam subbab sebelumnya bahwa algoritma perkalian skalar yang digunakan adalah *addition-subtraction algorithm.* Hasil pengujian pada Tabel 4.10 menunjukkan bahwa dalam interval [178,180] detik, program mampu melakukan operasi perkalian skalar kurva *elliptic* 32 bit sebanyak 1000 kali. Sehingga untuk 1 kali proses perkalian skalar 32 bit hanya dibutuhkan waktu sekitar 0.178 sampai 0.18 detik. Panjang kunci 32 bit berarti nilai skalarnya berada dalam interval [2147483648 , 4294967295].

4.3.3.2. Analisa Waktu Representasi Plaintext

Untuk melakukan enkripsi menggunakan algoritma ElGamal ECC, setiap *plaintext* akan direpresentasikan menjadi nilai numerik dan selanjutnya direpresentasikan menjadi titik kurva *elliptic*. Sedangkan proses pengembalian representasi titik kurva *elliptic* menjadi *plaintext* diperlukan dalam proses dekripsi yang menggunakan algoritma ElGamal ECC. Waktu yang dibutuhkan untuk representasi *plaintext* diberikan dalam Tabel 4.11 dan Tabel 4.12.

Tabel 4.11. Waktu Untuk Representasi *Plaintext* $\rightleftarrows$ Numerik

Representasi	Kunci	Perulangan	Waktu
Plaintext – Numerik	32 bit	1000	8 – 9 detik
Numerik – *Plaintext*	32 bit	1000	8 – 9 detik

Tabel 4.12. Waktu Untuk Representasi Numerik $\rightleftarrows$ Titik

Representasi	Kunci	Perulangan	Percobaan Titik	Waktu
Numerik – Titik	32 bit	100	100	16 – 17 detik
Titik – Numerik	32 bit	100	100	0.001 – 0.002 detik

Hasil pengujian pada Tabel 4.11 menunjukkan bahwa interval waktu yang dibutuhkan untuk representasi *plaintext* menjadi nilai numerik sama dengan proses pengembalian numerik menjadi *plaintext.* Waktu yang dibutuhkan untuk 1000 kali proses representasi *plaintext* menjadi numerik sekitar 8 sampai 9 detik,

dengan panjang kunci 32 bit. Sehingga untuk 1 kali proses representasi *plaintext* menjadi numerik hanya dibutuhkan waktu sekitar 0.008 sampai 0.009 detik. Waktu yang dibutuhkan untuk mengembalikan nilai numerik menjadi *plaintext* memiliki interval waktu yang sama dengan proses representasi *plaintext* menjadi numerik. Untuk panjang kunci 32 bit, jika dalam *plaintext* terdapat 3000 karakter maka waktu yang dibutuhkan untuk representasi *plaintext* menjadi numerik sekitar 8 sampai 9 detik. Karena untuk panjang kunci 32 bit, pemotongan pesan dilakukan untuk setiap 3 karakter.

Berdasarkan hasil pengujian pada Tabel 4.12, terlihat bahwa waktu yang dibutuhkan untuk representasi numerik menjadi titik kurva *elliptic* jauh lebih besar dibandingkan dengan waktu untuk mengembalikan titik menjadi nilai numerik. Karena dalam algoritma representasi numerik menjadi titik membutuhkan perhitungan dan iterasi yang lebih banyak, seperti perhitungan persamaan kurva *elliptic*, mencari akar modulo dan lain sebagainya. Kedua algoritma tersebut dapat dilihat pada subbab 4.3.2.2 dan 4.3.2.3, yaitu Algoritma 4.9 dan Algoritma 4.13.

4.3.3.3. Analisa Waktu Enkripsi dan Dekripsi ElGamal ECC

Pengujian dilakukan untuk mengetahui waktu yang dibutuhkan dalam proses enkripsi dan dekripsi menggunakan algoritma ElGamal ECC. Waktu yang dibutuhkan untuk kedua implementasi tersebut diberikan pada Tabel 4.13.

Tabel 4.13. Waktu Untuk Enkripsi dan Dekripsi ElGamal ECC

Proses	Kunci	Karakter	Perulangan	Total Karakter	Waktu
Enkripsi	32 bit	3	10	30	3 – 5 detik
Dekripsi	32 bit	3	10	30	1 – 2 detik

Berdasarkan hasil pengujian pada Tabel 4.13, terlihat bahwa waktu yang dibutuhkan untuk proses enkripsi lebih besar dibandingkan proses dekripsi ElGamal ECC. Operasi enkripsi membutuhkan waktu yang lebih banyak dibandingkan dengan operasi dekripsi, karena dalam proses enkripsi diperlukan 2 kali proses perkalian sedangkan proses dekripsi hanya membutuhkan 1 kali proses

perkalian. Sebagaimana dijelaskan sebelumnya, proses perkalian skalar kurva *elliptic* merupakan operasi aritmetika yang membutuhkan waktu paling lama dibandingkan operasi aritmetika kurva *elliptic* yang lain. Akibatnya waktu yang dibutuhkan untuk proses enkripsi menjadi lebih lama , sekitar 2 kali waktu yang dibutuhkan dalam proses dekripsi ElGamal ECC.

4.3.3.4. Hasil Implementasi ElGamal ECC

Hasil implementasi ElGamal ECC merupakan hasil dari *running* program yang dibuat oleh penulis. Ada tiga program utama yang dibuat penulis, yaitu

1. Program penentuan kunci
2. Program enkripsi ElGamal ECC
3. Program dekripsi ElGamal ECC

Hasil dari ketiga program utama ini merupakan salah satu contoh hasil implementasi ElGamal ECC. Jika *input* yang diberikan berbeda maka *output* yang dihasilkan program utama juga akan berbeda. Karena itu, diberikan salah satu contoh *output* dari ketiga program utama yang merupakan hasil implementasi ElGamal ECC. Hasil *running* ketiga program utama tersebut diberikan dalam Tabel 4.14, Tabel 4.15 dan Tabel 4.16. Penulisan hasil implementasi dalam bentuk tabel bertujuan untuk mempermudah dalam membedakan *input/output* hasil *running* program. Selain itu, dapat membantu dan mempermudah pembaca dalam memahami hasil implementasi ElGamal ECC.

Tabel 4.14. Hasil Implememtasi Program Penentuan Kunci

INPUT		
Panjang Kunci (nkunci)		32
OUTPUT		
Bilangan Prima (p)		3946183951
Koefisien Persamaan Kurva *Elliptic* $y^2 = x^3 + Ax + B$	A	537680305
	B	1059676324
Basic Point (G_E)		(1152222263 , 3133703258)
Order *Basic Point* (N_G)		3946206427
Kofaktor (h)		1
Private Key (V)		2759936539
Public Key (β)		(3539395206 , 1802765602)

Tabel 4.15. Hasil Implementasi Program Enkripsi ElGamal ECC

INPUT		
Bilangan Prima (p)		3946183951
Koefisien Persamaan Kurva *Elliptic* $y^2 = x^3 + Ax + B$	A	537680305
	B	1059676324
Basic Point (G_E)		(1152222263 , 3133703258)
Order *Basic Point* (N_G)		3946206427
Banyaknya Percobaan Representasi Titik (ε)		100
Public Key (β)		(3539395206 , 1802765602)
Plaintext (pesan)	Implementasi ElGamal ECC menggunakan software Matlab Student Edition Version 6.1. Aplikasi tersebut berjalan pada sistem operasi Windows XP Professional Version 2002 dan menggunakan Processor Intel Pentium IV 2.4 GHz, dan RAM 256 MB PC2100.	
OUTPUT		
Chipertext	3713176816 2667860958 3467254065 1203046706 508327490 890698402 3880542999 873172883 2197816276 309805134 1484573791 363148350 704240343 1535806795 3914281142 1766729068 1235067086 560748724 2530896756 2135768648 3072966783 100851621 2568304029 2769187438 1182138370 1145256211 1629905884 529284477 2428427195 3738917079 2170161065 442104761 256457278 825010047 1236728713 3740806696 266074242 1612464806 3273689647 525537648 108080924 1559985589 1270470940 2671986191 531891046 2271726192 3717379793 674625714 768894394 2053587908 2083632209 1668824858 302032790 2760307404 962633767 840085651 2600733957 2377987824 3697949061 1695141776 2460990159 608509636 1745245151 3075123480 310804293 3685703054 1291552889 1138875364 2992447074 764521653 3882484657 708108359 2855396488 1317574337 2210618093 1100886891 3393669928 1219181878 2611407279 3586887319 3326671246 198358419 2083095301 1358834964 1986218232 693798216 570539317 2104952608 1919858957 650213148 3123593025 2354744366 560200324 128146259 1860821161 2535874192 1315159882 2590426541 1393335596 3279232077 1444185791 570600507 1938727432 1346524390 1548452328 1569331598 1528943981 1743901677 403978666 1743151203 2891856331 1535353811 1649679535 1391002387 3779361632 2738026484	

Lanjutan Tabel 4.15.

OUTPUT	
Chipertext	2463402192 3841665691 3838133235 3109625239 2948260648 238917396 1794117580 1595571738 1731927804 1806324376 675722714 558282816 3522343176 3768943661 1068917444 485332569 627990649 3174212475 3857162970 3706349583 854080812 1501446975 1414077883 3913949773 1442353281 1832656599 991853592 2222616390 546332926 3754974245 1286681145 3710904556 311368984 290855217 2523339369 2070130582 2814844264 1759311489 1426020261 1816749920 2032863193 3724559391 218388061 2743743526 1058728867 1576951702 1613533622 74219845 2329408125 889197180 819777474 1878890738 675773578 3858293081 1127560903 1137685875 1207502493 253713092 1282778741 1578557009 1847595746 736205944 667446970 1041347175 3823490192 491032894 875128413 1405440539 3367362343 3489449443 2302488696 1114991611 1174710755 624614015 3655577877 2354502151 1249861415 811144839 3861619686 3895797311 2190132504 2614325672 1741453040 1326412455 2096787798 1458045176 562019906 2420257363 436466836 3623507937 52774750 447550477 2691396445 3351456532 3461520297 919239738 2886128772 2905182765 282207974 1775261495 3011267267 126331242 2855955232 252153145 2812441287 700113735 1840356952 2074914500 2816402247 477956744 3383253902 1396750526 2977335974 3388022847 3537032124 2852007468 1249386766 456789013 2440976544 1957554547 1598792044 504910382 1113531833 1969361013 249683634 2857194281 1473082688 494577984 3182243308 3375051560 642400005 937592937 1153328997 1692216513 3460164133 144536605 2867906237 976047845 2574613192 3620761934 2520891451 2819721483 1354891914 1626116767 2875871838 554402698 464499768 294104367 916332389 3021508591 91843070 2910576496 2030336028 3319867796 3045613965 3836105262 1153697313 587353502 1885832408 3061981662 3077311284 67305656 151488218 260143866 3265130908 1328378617 1621732129 768311542 1752208165 355610663 1700584479 1940692813

Lanjutan Tabel 4.15.

OUTPUT	
Chipertext	1373417655 278363494 2474561009 2593347791 1234198938 2335608737 136790476 3042940263 3710935248 141022157 2637250214 1338556478 2994367837 3411566687 3561531451 246770444 3762448962 3588009397 1853103490 1483240088 3616014993 515524358 212182721 125222518 936385716 120270585 1816178236 1457118280 3529310492 1504942177 250086000 160160491

Tabel 4.16. Hasil Implementasi Program Dekripsi ElGamal ECC

INPUT		
Bilangan Prima (p)		3946183951
Koefisien Persamaan Kurva *Elliptic* $y^2 = x^3 + Ax + B$	A	537680305
Private Key (V)		2759936539
Banyaknya Percobaan Representasi Titik (ε)		100
Chipertext	3713176816 2667860958 3467254065 1203046706 508327490 890698402 3880542999 873172883 2197816276 309805134 1484573791 363148350 704240343 1535806795 3914281142 1766729068 1235067086 560748724 2530896756 2135768648 3072966783 100851621 2568304029 2769187438 1182138370 1145256211 1629905884 529284477 2428427195 3738917079 2170161065 442104761 256457278 825010047 1236728713 3740806696 266074242 1612464806 3273689647 525537648 108080924 1559985589 1270470940 2671986191 531891046 2271726192 3717379793 674625714 768894394 2053587908 2083632209 1668824858 302032790 2760307404 962633767 840085651 2600733957 2377987824 3697949061 1695141776 2460990159 608509636 1745245151 3075123480 310804293 3685703054 1291552889 1138875364 2992447074 764521653 3882484657 708108359 2855396488 1317574337 2210618093 1100886891 3393669928 1219181878 2611407279 3586887319 3326671246 198358419 2083095301 1358834964 1986218232 693798216 570539317 2104952608 1919858957 650213148 3123593025 2354744366 560200324 128146259 1860821161 2535874192 1315159882 2590426541 1393335596 3279232077

Lanjutan Tabel 4.16.

INPUT	
Chipertext	1444185791 570600507 1938727432 1346524390 1548452328 1569331598 1528943981 1743901677 403978666 1743151203 2891856331 1535353811 1649679535 1391002387 3779361632 2738026484 2463402192 3841665691 3838133235 3109625239 2948260648 238917396 1794117580 1595571738 1731927804 1806324376 675722714 558282816 3522343176 3768943661 1068917444 485332569 627990649 3174212475 3857162970 3706349583 854080812 1501446975 1414077883 3913949773 1442353281 1832656599 991853592 2222616390 546332926 3754974245 1286681145 3710904556 311368984 290855217 2523339369 2070130582 2814844264 1759311489 1426020261 1816749920 2032863193 3724559391 218388061 2743743526 1058728867 1576951702 1613533622 74219845 2329408125 889197180 819777474 1878890738 675773578 3858293081 1127560903 1137685875 1207502493 253713092 1282778741 1578557009 1847595746 736205944 667446970 1041347175 3823490192 491032894 875128413 1405440539 3367362343 3489449443 2302488696 1114991611 1174710755 624614015 3655577877 2354502151 1249861415 811144839 3861619686 3895797311 2190132504 2614325672 1741453040 1326412455 2096787798 1458045176 562019906 2420257363 436466836 3623507937 52774750 447550477 2691396445 3351456532 3461520297 919239738 2886128772 2905182765 282207974 1775261495 3011267267 126331242 2855955232 252153145 2812441287 700113735 1840356952 2074914500 2816402247 477956744 3383253902 1396750526 2977335974 3388022847 3537032124 2852007468 1249386766 456789013 2440976544 1957554547 1598792044 504910382 1113531833 1969361013 249683634 2857194281 1473082688 494577984 3182243308 3375051560 642400005 937592937 1153328997 1692216513 3460164133 144536605 2867906237 976047845 2574613192 3620761934 2520891451 2819721483 1354891914 1626116767 2875871838 554402698 464499768 294104367 916332389 3021508591 91843070 2910576496 2030336028 3319867796 3045613965 3836105262

Lanjutan Tabel 4.16

INPUT	
Chipertext	1153697313 587353502 1885832408 3061981662 3077311284 67305656 151488218 260143866 3265130908 1328378617 1621732129 768311542 1752208165 355610663 1700584479 1940692813 1373417655 278363494 2474561009 2593347791 1234198938 2335608737 136790476 3042940263 3710935248 141022157 2637250214 1338556478 2994367837 3411566687 3561531451 246770444 3762448962 3588009397 1853103490 1483240088 3616014993 515524358 212182721 125222518 936385716 120270585 1816178236 1457118280 3529310492 1504942177 250086000 160160491
OUTPUT	
Plaintext (pesan)	Implementasi ElGamal ECC menggunakan software Matlab Student Edition Version 6.1. Aplikasi tersebut berjalan pada sistem operasi Windows XP Professional Version 2002 dan menggunakan Processor Intel Pentium IV 2.4 GHz, dan RAM 256 MB PC2100.