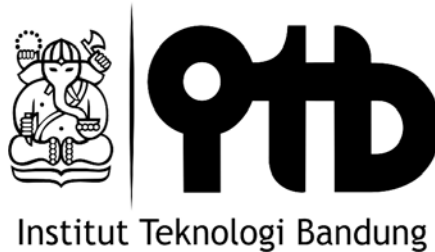


**TUGAS AKHIR
KEAMANAN SISTEM INFORMASI LANJUT
[EC 7030]**

**IMPLEMENTASI *ELLIPTIC CURVES CRYPTOSYSTEM* (ECC)
PADA PROSES PERTUKARAN KUNCI DIFFIE-HELLMAN DAN
SKEMA ENKRIPSI ELGAMAL**

Dosen : Dr.Ir. Budi Rahardjo

Oleh :
Nana Juhana
23202032



**DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI INDUSTRI
PROGRAM PASCASARJANA
INSTITUT TEKNOLOGI BANDUNG
2005**

IMPLEMENTASI *ELLIPTIC CURVES CRYPTOSYSTEM* (ECC) PADA PROSES PERTUKARAN KUNCI DIFFIE-HELLMAN DAN SKEMA ENKRIPSI ELGAMAL

Abstrak

Kriptosistem kurva elips (*elliptic curves cryptosystem*) atau disingkat dengan ECC, merupakan salah satu sistem kriptografi asimetris yang menggunakan persoalan logaritma diskrit (*discrete logarithm problem*). Struktur kurva elips digunakan sebagai grup operasi matematis untuk melangsungkan proses enkripsi dan deskripsinya. Pada tulisan ini diuraikan teknik dasar ECC yang diimplementasikan pada protokol pertukaran kunci publik Diffie-Hellman dan Skema enkripsi ElGamal. Jumlah bit yang digunakan pada parameter-parameter ECC berkisar antara 32 bit sampai dengan 256 bit dengan kenaikan masing-masing sebesar 32 bit.

Pada pertukaran kunci, hasil implementasinya memperlihatkan pertukaran kunci publik antara dua user dan menghitungnya dimasing-masing user yang akan menghasilkan kunci rahasia bersama. Sementara pada skema enkripsi, hasil implementasi telah menunjukkan pesan berupa bilangan integer besar dipetakan dalam titik kurva yang kemudian dienkripsi berhasil dibuka kembali pada proses deskripsinya.

Pada tulisan ini juga dibahas tinjauan keamanan dengan membandingkan kriptosistem kurva elips dengan kriptosistem sejenis yaitu RSA dan DSA. Hasil studi memperlihatkan pada tingkat kewanaman yang sama kriptosistem kurva elips memerlukan jumlah bit kunci yang jauh lebih sedikit dibandingkan dengan RSA dan DSA.

I. Tinjau Umum Kriptografi

Kriptografi (*Cryptography*) merupakan bidang pengetahuan yang menggunakan persamaan matematis untuk melakukan proses enkripsi (*encrypt*) maupun deskripsi (*decrypt*) data. Teknik ini digunakan untuk mengubah data ke dalam kode-kode tertentu, dengan tujuan informasi yang disimpan atau ditransmisikan melalui jaringan yang tidak aman (misalnya saja internet) tidak dapat dibaca oleh siapapun kecuali orang-orang yang berhak.

Pesan atau informasi yang dapat dibaca disebut dengan *plaintext* atau *cleartext*. Proses yang digunakan untuk menyamarkan atau menyembunyikan *plaintext* tersebut disebut dengan enkripsi. Teks yang sudah disamarkan atau disembunyikan pada proses enkripsi berisi informasi yang tidak dapat atau tidak mudah dibaca dan dimengerti dengan jelas. Teks hasil enkripsi ini disebut dengan *chipertext*. Proses kebalikan enkripsi, yaitu mengubah *chipertext* menjadi *plaintext* disebut dengan proses deskripsi.



Gambar 1. Enkripsi dan deskripsi

1.1. Algoritma dan Kunci

Algoritma kriptografi merupakan fungsi matematis yang digunakan untuk proses enkripsi dan deskripsi. Algoritma kriptografi ini bekerja dalam kombinasi dengan menggunakan kunci (*key*) seperti kata, nomor atau frase tertentu.

Bila keamanan algoritma bergantung pada kerahasiaan algoritma yang bekerja, maka algoritma tersebut dikatakan sebagai algoritma terbatas (terbatas kemampuannya). Algoritma terbatas tidak cukup baik untuk diterapkan saat ini. Kerahasiaan algoritmanya menjadi titik kelemahan yang sudah barang tentu tidak mengijinkan adanya kontrol kualitas atau standarisasi.

Kriptografi modern menyelesaikan masalah enkripsi dan deskripsi dengan merahasiakan kunci saja tanpa harus merahasiakan algoritmanya. Kunci ini merupakan nilai yang sangat spesifik dan bekerja dengan algoritma kriptografi untuk menghasilkan teks yang terenkripsi secara spesifik pula. Dengan kunci inilah nantinya kita dapat melakukan proses enkripsi dan deskripsi. Karena keamanan bergantung pada kerahasiaan kuncinya, maka algoritma yang dibentuk dapat dianalisa dan dipublikasikan. Sehingga memungkinkan pengembangan algoritma yang lebih baik.

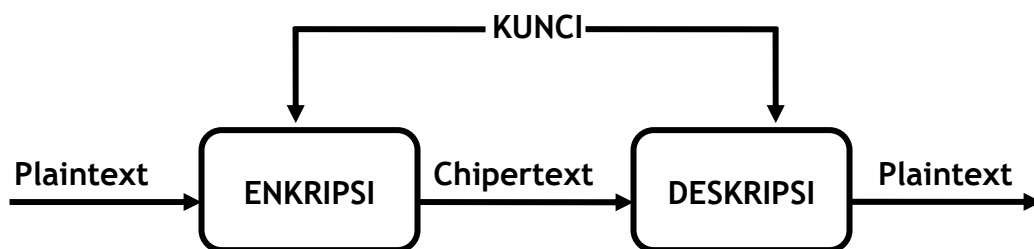
1.2. Pengelompokan Algoritma Kriptografi

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

1. Algoritma Simetris (Algoritma Konvensional)
2. Algoritma Asimetris (Algoritma Kunci Publik)

1.2.1. Algoritma Simetris

Algoritma ini disebut juga dengan algoritma konvensional, yaitu algoritma yang menggunakan kunci yang sama pada proses enkripsi dan deskripsinya. Algoritma ini mengharuskan pengirim dan penerima menyetujui satu kunci tertentu



Gambar 2. Skema enkripsi simetris

Kelompok algoritma simetris adalah OTP, DES, RC2, RC4, RC5, RC6, IDEA, Twofish, Magenta, FEAL, SAFER, LOKI, CAST, Rijndael (AES), Blowfish, GOST, A5 Kasumi dan lain-lain.

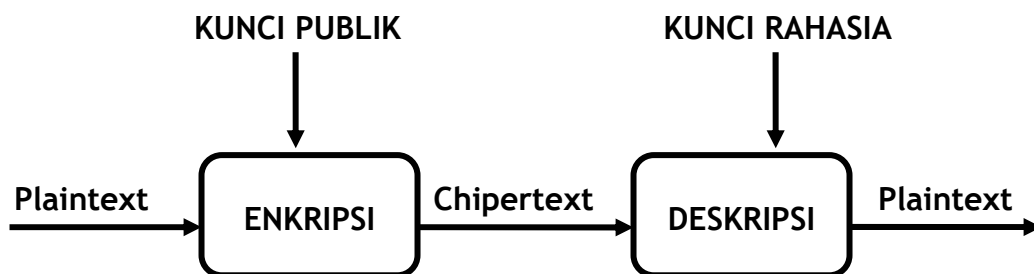
Kelebihan algoritma simetris ini adalah kecepatan proses enkripsi dan deskripsinya yang jauh lebih cepat dibandingkan dengan algoritma asimetris. Sedangkan kelemahan algoritma ini adalah permasalahan distribusi kunci (*key distribution*). Seperti yang telah dibahas, proses enkripsi dan deskripsi menggunakan kunci yang sama. Sehingga muncul persoalan menjaga kerahasiaan kunci, yaitu pada saat pengiriman kunci pada media yang tidak aman seperti internet. Tentunya jika kunci ini sampai hilang atau sudah dapat ditebak oleh orang lain (orang yang tidak berhak), maka kriptosistem ini sudah tidak aman lagi.

Kelemahan lain adalah masalah efisiensi jumlah kunci. Jika terdapat n user, maka diperlukan $n(n-1)/2$ kunci, sehingga untuk jumlah user yang sangat banyak, sistem ini tidak efisien lagi.

1.2.2. Algoritma Asimetris

Masalah distribusi kunci pada algoritma simetris dapat diatasi dengan metode kriptografi asimetris atau disebut juga dengan kriptografi kunci publik (*public key algorithm*). Sebutan asimetris (tidak simetris) memperlihatkan adanya perbedaan kunci yang digunakan antara proses enkripsi dan deskripsi. Kunci publik digunakan untuk proses enkripsi data sedangkan proses deskripsi menggunakan kunci yang biasa disebut dengan kunci rahasia (*private key*).

Berdasarkan konsep ini kunci yang didistribusikan adalah kunci publik yang tidak diperlukan kerahasiannya, sedangkan kunci rahasia tetap disimpan (tidak didistribusikan). Jadi setiap orang yang memiliki kunci publik dapat melakukan proses enkripsi yang hanya bisa dibaca oleh orang yang memiliki kunci rahasia.



Gambar 3. Skema kriptografi asimetris

Keuntungan utama dari algoritma ini adalah memberikan jaminan keamanan kepada siapa saja yang melakukan pertukaran informasi meskipun diantara mereka tidak ada kesepakatan mengenai keamanan data terlebih dahulu maupun saling tidak mengenal satu sama lainnya. Beberapa contoh konsep yang menggunakan algoritma ini adalah skema enkripsi ElGamal, RSA, Diffie – Hellman (DH), DSA (*Digital Signature Algorithm*) dan lain-lain.

Lebih lanjut nantinya akan dibahas implementasi konsep algoritma kunci publik pada protokol pertukaran kunci Diffie-Helman dan skema enkripsi ElGamal menggunakan persamaan kurva elips yang merupakan salah satu jenis *cryptosystem* yang dikenal dengan *Elliptic Curves Cryptosystem* (ECC).

II. Konsep Dasar Matematika Kriptografi

Sebelum membahas lebih jauh mengenai *Elliptic Curves Cryptosystem*, terlebih dahulu akan dipaparkan konsep dasar matematika yang berhubungan dengan persoalan kriptografi.

2.1. Integer

Misalkan diberikan himpunan semua integer yang dinotasikan dengan Z dan N menyatakan himpunan semua bilangan integer positif. Untuk himpunan berhingga A , jumlah elemen pada himpunan A yang dinotasikan dengan $\#A$. Sebuah relasi ekuivalen pada A adalah sebuah relasi biner \sim pada himpunan A , untuk setiap $x, y, z \in A$ terdiri dari :

- Refleksif : $x \sim x$
- Simetris : jika $x \sim y$ maka $y \sim x$
- Transitif : jika $x \sim y$ dan $y \sim z$ maka $x \sim z$

Diberikan \sim adalah sebuah relasi ekuivalen pada A dan $P = \{[a] \mid a \in A\}$ dimana $[a] = \{b \in A \mid a \sim b\}$ adalah sebuah partisi A , yaitu :

1. untuk setiap $P \in P, P \neq \emptyset$
2. jika $S, T \in P$ maka $S = T$ untuk $S \cap T = \emptyset$
3. $\cup_{S \in P} S = A$

Elemen $S \in P$ disebut dengan sebuah kelas ekuivalen dari partisi P .

Properti dasar yang sering digunakan dalam integer adalah Algoritma Pembagian Euclidian (*Euclid's Subdivision Algorithm*), yaitu untuk $a, b \in \mathbb{Z}$, $b \neq 0$, terdapat secara unik $q, r \in \mathbb{Z}$ sedemikian sehingga $a = bq + r$, ($0 \leq r < |b|$).

Jika $r = 0$, kita dapat katakan bahwa b adalah pembagi (*divisor*) a dan dinyatakan sebagai $b|a$. Sebaliknya jika bukan dinyatakan sebagai $b \nmid a$. Untuk $a_1, a_2, \dots, a_k \in \mathbb{Z}$, jika $b|a_i$ ($i=1, 2, \dots, k$) maka b disebut faktor persekutuan (*common divisor*) dari a_1, a_2, \dots, a_k dimana selalu terdapat faktor persekutuan terbesar (*greatest common divisor*) yang dinotasikan dengan $\text{gcd}(a_1, a_2, \dots, a_k)$. $a, b \in \mathbb{Z}$ disebut prima relatif (*coprime*) jika dan hanya jika $\text{gcd}(a, b) = 1$.

2.2. Grup

Grup adalah sebuah obyek matematika abstrak yang terdiri dari sebuah himpunan G dan sebuah operasi biner $*$ yaitu :

1. $a * (b * c) = (a * b) * c$ untuk $a, b, c \in G$ (asosiatif).
2. terdapat sebuah elemen $e \in G$ sehingga $e * a = a * e = a$ untuk setiap $a \in G$
3. untuk setiap $a \in G$ terdapat sebuah elemen $b \in G$ sehingga $b * a = a * b = e$
 b disebut dengan invers a

Notasi $\langle G, * \rangle$ menyatakan sebuah grup dengan operasi grup $*$. Notasi $\langle G, + \rangle$ disebut dengan grup penjumlahan dan $\langle G, \cdot \rangle$ disebut grup perkalian. Pada grup penjumlahan, elemen netral disimbolkan dengan 0 dan invers dari a dinyatakan sebagai $-a$. Sedangkan grup perkalian elemen netral disimbolkan dengan 1 dan invers a dinyatakan sebagai a^{-1} .

Sebagai contoh integer modulo n , ditulis sebagai $Z_n = \{0, 1, 2, \dots, n-1\}$ merupakan bentuk sebuah grup pada operasi penjumlahan modulo n . Jika p adalah bilangan prima, maka elemen-elemen bukan nol Z_p dapat ditulis sebagai $Z_p^* = \{1, 2, \dots, p-1\}$, merupakan bentuk sebuah grup pada operasi perkalian modulo p . Nilai elemen grup $g \in G$ adalah bilangan integer positif n sehingga $g^n = 1$. Sebagai contoh untuk grup Z_{11}^* dengan elemen $g = 3$ memiliki 5 buah nilai, yaitu :

$$3^1 \equiv 3 \pmod{11},$$

$$3^2 \equiv 9 \pmod{11},$$

$$3^3 \equiv 5 \pmod{11},$$

$$3^4 \equiv 4 \pmod{11},$$

$$3^5 \equiv 1 \pmod{11}.$$

Pada *Discrete Logarithm Problem* (DLP) yang diperkenalkan oleh Diffie dan Hellman mendefinisikan masalah pencarian logaritma dalam grup Z_p^* : diberikan elemen $g \in Z_p^*$ pada n buah nilai dan diberikan $h \in Z_p^*$, temukan sebuah integer x , $0 \leq x \leq n-1$, sedemikian sehingga $g^x \equiv h \pmod{p}$. Nilai x selanjutnya disebut dengan logaritma diskrit dari h pada basis g . Sebagai contoh untuk $p=17$ dan $g=10$ adalah elemen untuk nilai $n=16$ dalam grup Z_{17}^* . Jika $h=11$ maka logaritma diskrit h pada basis g adalah 13 karena $10^{13} \equiv 11 \pmod{17}$.

Konsep logaritma diskrit dapat diperluas menjadi : diberikan G grup dari n dan diberikan α adalah elemen G . DLP dari grup G adalah sebagai berikut :

diketahui α dan $\beta \in G$, temukan sebuah integer x , $0 \leq x \leq n-1$,
sedemikian sehingga $\alpha^x = \beta$.

2.3. Bentuk Polinomial

Sebuah polinomial F dengan derajat n dinyatakan dalam bentuk ekspresi

$$f(x) = \sum_{i=0}^n a_i x^i = a_0 + a_1 x + \dots + a_n x^n \quad (1)$$

dimana n adalah integer positif dan koefisien $a_i \in F (0 \leq i \leq n)$. Untuk mengevaluasi polinomial $f(a)$ untuk beberapa nilai $a \in F$ kita ganti setiap x dalam $f(x)$ dengan a . Diberikan dua bentuk polinomial :

$$f(x) = \sum_{i=0}^n a_i x^i \text{ dan } g(x) = \sum_{i=0}^n b_i x^i \quad (2)$$

penjumlahan $f(x)$ dan $g(x)$ dinyatakan sebagai:

$$f(x) + g(x) = \sum_{i=0}^n (a_i + b_i) x^i \quad (3)$$

Diberikan dua bentuk polinomial :

$$f(x) = \sum_{i=0}^m a_i x^i \text{ dan } g(x) = \sum_{i=0}^n b_i x^i \quad (4)$$

perkalian $f(x)$ dan $g(x)$ dinyatakan sebagai:

$$f(x)g(x) = \sum_{k=0}^{n+m} c_k x^k \quad \text{dimana} \quad \sum_{\substack{i+j=k \\ 0 \leq i \leq n, 0 \leq j \leq m}} a_i b_j \quad (5)$$

III. Kriptosistem Kurva Elips (*Elliptic Curves Cryptosystem*)

Pada tahun 1985, Neil Koblitz dan Viktor Miller secara terpisah memproposalkan kriptosistem kurva elips (*Elliptic Curves Cryptosystem - ECC*) yang menggunakan masalah logaritma diskrit pada titik-titik kurva elips yang disebut dengan ECDLP (*Elliptic Curves Discrete Logarithm Problem*). Kriptosistem kurva elips ini dapat digunakan pada beberapa keperluan seperti :

- Skema enkripsi (ElGamal ECC)
- Tanda tangan digital (ECDSA – *Elliptic Curves Digital Signature*)
- Protokol pertukaran kunci (Diffie Hellman ECC)

Saat ini ada tiga macam sistem kriptografi kunci publik yang aman dan efisien yang dikelompokkan berdasarkan permasalahan matematis, yaitu :

- **Sistem Pemfaktoran Bilangan Integer (*Integer Factorization Systems*)**
Tipe ini menggunakan masalah matematis yang disebut *Integer Factorization Problem* (IFP). Jika diberikan bilangan integer n yang merupakan hasil kali dua buah bilangan prima, maka harus dicari kedua bilangan prima p dan q yang merupakan faktor n , sehingga $n = p * q$. Cara ini tentunya akan menyebabkan kesulitan menghitung faktor integer yang besar.
- **Sistem Logaritma Diskrit (*Discrete Logarithm Systems*)**
Tipe ini menggunakan masalah matematis yang disebut *Discrete Logarithm Problem* (DLP). Taher ElGamal adalah orang pertama mengajukan kriptosistem kunci publik berdasarkan masalah ini. Pada tahun 1991, Claus Schnorr menemukan variasi ElGamal untuk membuat tanda tangan digital yang menawarkan keamanan tambahan dibandingkan dengan sistem aslinya. Pemerintah

Amerika Serikat menggunakan DSA (Digital Signature Algorithm) yang berdasarkan ElGamal ini. DLP merupakan masalah yang didefinisikan pada aritmetika modular serta penggunaan grup perkalian. Jika dipilih bilangan prima p dan diberikan bilangan integer g antara 0 dan $p-1$ serta y merupakan pemangkatan dari g sehingga :

$$y = g^x \pmod{p} \quad (6)$$

untuk beberapa x . Masalah logaritma diskrit pada modulo p adalah untuk mencari x jika diberikan pasangan bilangan g dan y . Cara ini menyebabkan kesulitan menghitung $x = (\log b) \pmod{p}$.

- **Kriptosistem Kurva Elips (*Elliptic Curves Cryptosystem*)**

Pada sistem ini digunakan masalah logaritma diskrit kurva elips dengan menggunakan grup kurva elips. Struktur kurva elips digunakan sebagai grup operasi matematis untuk melangsungkan proses enkripsi dan deskripsi. Cara ini menyebabkan kesulitan menghitung k jika diketahui Q dan P dimana $Q = kP$.

3.1. Kurva Elips

Pada bagian ini akan dibahas teknik dasar kurva elips dalam grup Z_p dimana p adalah bilangan prima yang lebih besar dari 3. Selanjutnya kurva elips secara umum didefinisikan sebagai *field* berhingga (*finite field*).

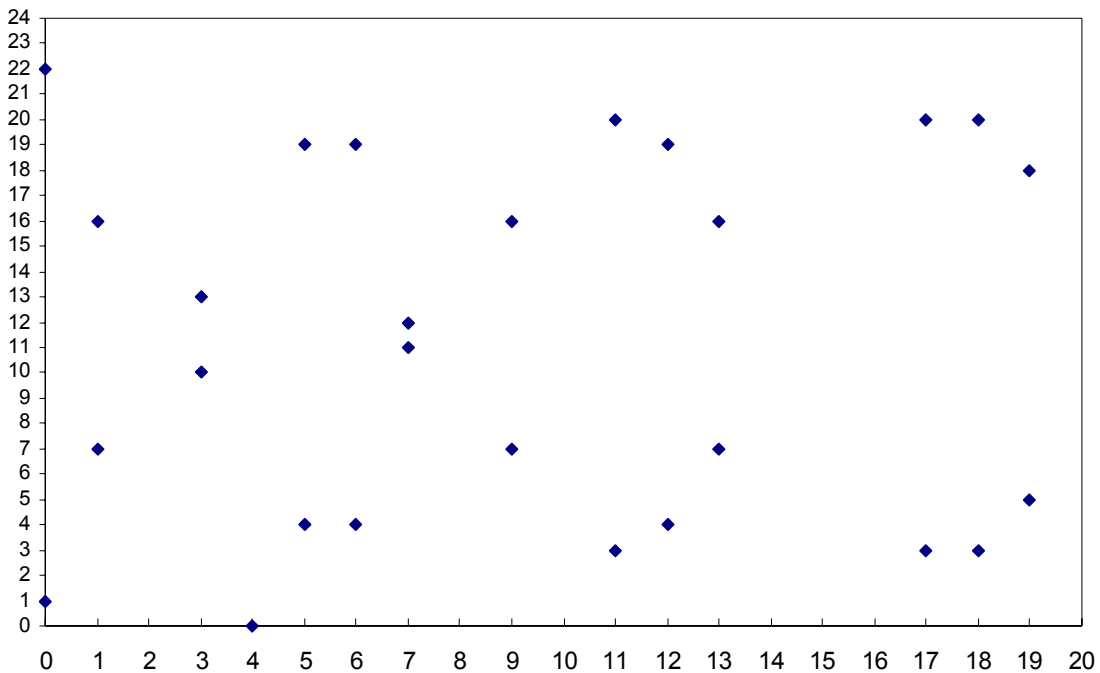
Sebuah kurva elips E pada Z_p didefinisikan dalam persamaan :

$$y^2 = x^3 + ax + b, \quad (7)$$

dimana $a, b \in Z_p$ dan $4a^3 + 27b^2 \neq 0 \pmod{p}$, dan sebuah titik \mathbf{O} yang disebut dengan titik *infinity*. Himpunan $E(Z_p)$ adalah semua titik (x,y) , untuk $x,y \in Z_p$, yang memenuhi persamaan (1) pada titik \mathbf{O} .

Untuk menjelaskan uraian di atas, berikut ini diberikan contoh pencarian himpunan $E(Z_p)$. Diberikan persamaan kurva elips $E: y^2 = x^3 + x + 1$ dengan $p = 23$, yaitu grup Z_{23} (pada persamaan (1) $a = b = 1$). Maka untuk nilai $4a^3 + 27b^2 = 4 + 27 \neq 0$, sehingga E ada dalam kurva elips. Titik-titik dalam $E(Z_{23})$ adalah :

(0,1)	(6,4)	(12,19)
(0,22)	(6,19)	(13,7)
(1,7)	(7,11)	(13,16)
(1,16)	(7,12)	(17,3)
(3,10)	(9,7)	(17,20)
(3,13)	(9,16)	(18,3)
(4,0)	(11,3)	(18,20)
(5,4)	(11,20)	(19,5)
(5,19)	(12,4)	(19,18)



Gambar 4. Sebaran titik-titik pada kurva elips $E(Z_{23})$ untuk $E: y^2 = x^3 + x + 1$

3.2. Aturan Penjumlahan Dua Titik pada Kurva Elips

Untuk membentuk *elliptic curve cryptosystem* (ECC) diperlukan aturan penjumlahan dua titik pada kurva elips $E(Z_p)$ yang menghasilkan titik ke tiga pada kurva elips. Aturan ini dapat dijelaskan secara geometris sebagai berikut :

1. $P + \mathbf{O} = \mathbf{O} + P = P$ untuk setiap $P \in E(Z_p)$.
2. Jika $P = (x,y) \in E(Z_p)$ maka $(x,y) + (x,-y) = \mathbf{O}$. (Titik $(x,-y)$ dinyatakan dengan $-P$, dan disebut negatif P . Tentunya $-P$ merupakan sebuah titik dalam kurva).

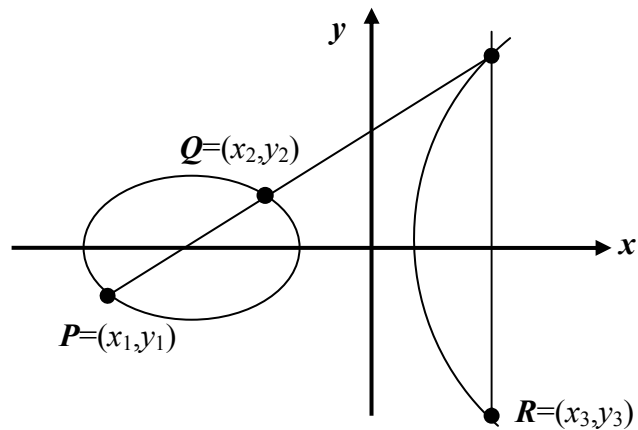
3. Diberikan $P = (x_1, y_1) \in E(Z_p)$ dan $Q = (x_2, y_2) \in E(Z_p)$, dimana $P \neq Q$. Maka $P+Q = (x_3, y_3)$ diperoleh dengan mengambil garis L yang melewati titik P dan Q atau garis singgung L untuk $P=Q$. Misalkan garis $L : y = \lambda x + \beta$ di mana :

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{untuk } P \neq Q \\ \frac{3x_1^2 + a}{2y_1} & \text{untuk } P = Q \end{cases}$$

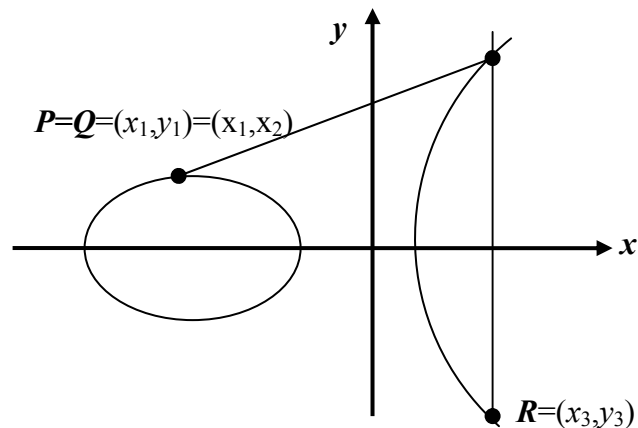
maka diperoleh (x_3, y_3) sebagai berikut :

$$x_3 = \lambda^2 - x_1 - x_2$$

$$y_3 = \lambda(x_1 + x_3) - y_1$$



Gambar 5. Gambaran secara geometri penjumlahan dua titik berbeda



Gambar 6. Gambaran secara geometri penjumlahan dua titik sama

Untuk mengilustrasikan penjelasan di atas, perhatikan contoh berikut ini.

1. Contoh $P \neq Q$, untuk $P=(3,10)$ dan $Q=(9,7)$, maka $P+Q=(x_3,y_3)$ diperoleh melalui :

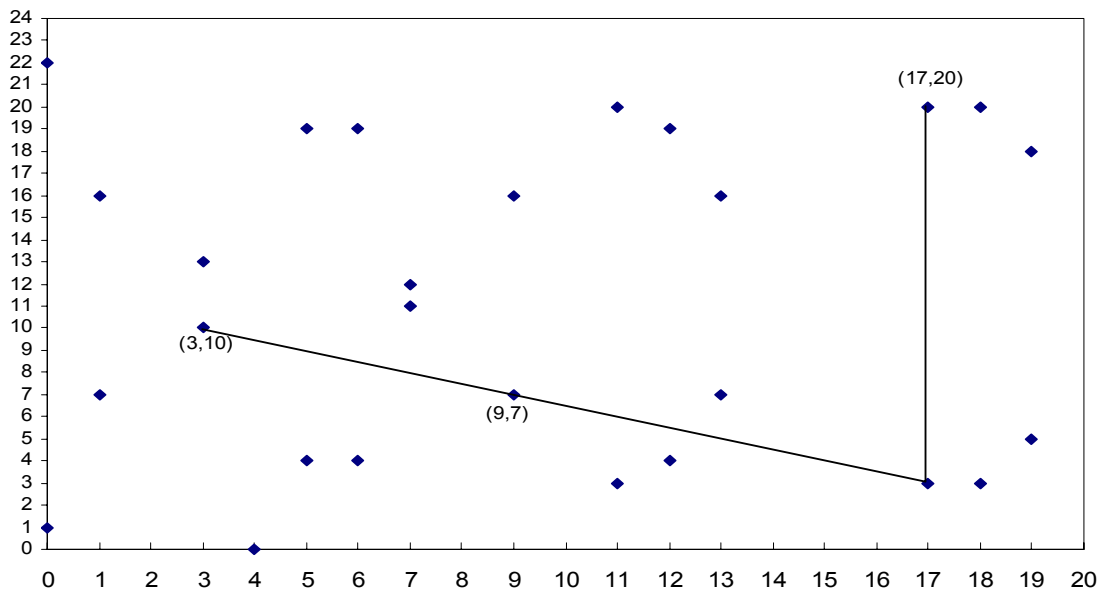
$$\lambda = \frac{7-10}{9-3} = 11 \in Z_{23}$$

$$x_3 = 11^2 - 3 - 9 = -6 \equiv 17 \pmod{23} \text{ dan}$$

$$y_3 = 11(3 - (-6)) - 10 = 89 \equiv 20 \pmod{23}$$

$$\text{Jadi } P+Q = (17,20)$$

Hasil ini diperlihatkan secara geometris dalam gambar di bawah ini.



Gambar 7. Gambaran geometris untuk $(3,10) + (9,7) = (17,20)$

2. Contoh $P=Q$, untuk $P=(3,10)$, maka $2P=P+P=(x_3,y_3)$ diperoleh melalui :

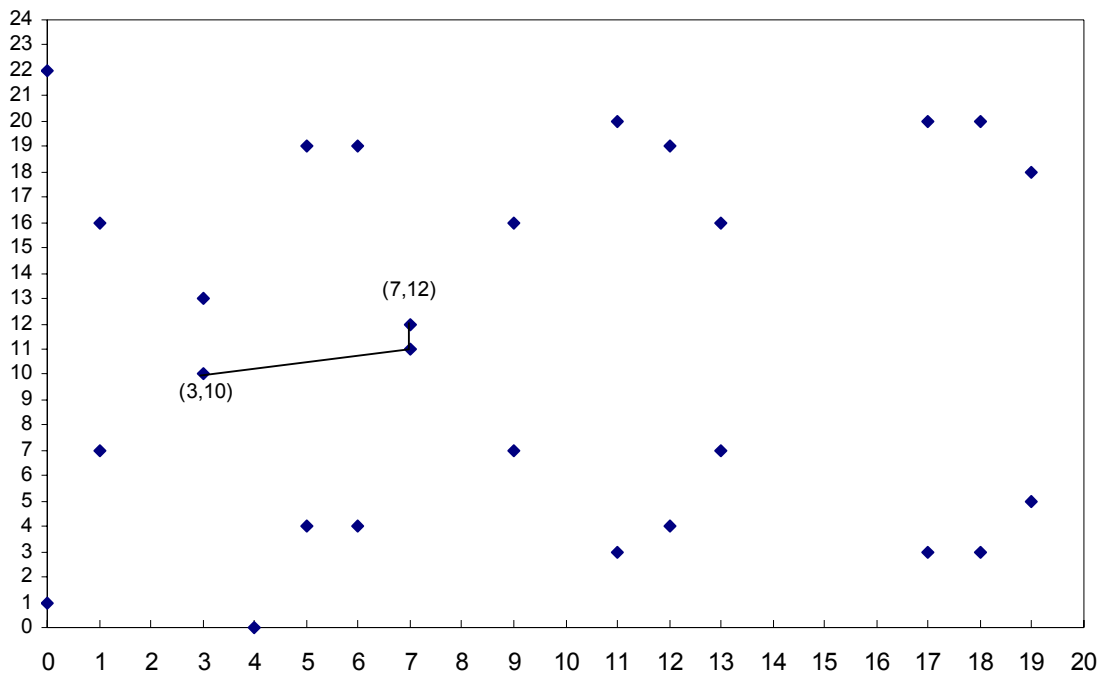
$$\lambda = \frac{3(3^2)+1}{20} = 6 \in Z_{23}$$

$$x_3 = 62 - 6 = 30 \equiv 7 \pmod{23}$$

$$y_3 = 6(3 - 7) - 10 = -11 \equiv 12 \pmod{23}$$

$$\text{Jadi } 2P = (7,12)$$

Hasil ini diperlihatkan secara geometris dalam gambar 8.



Gambar 8. Gambaran geometris untuk $(3,10) + (3,10) = (7,12)$

Berdasarkan penjelasan di atas, operasi grup untuk kurva elips $E(Z_p)$ disebut dengan operasi penjumlahan, dan disebut operasi perkalian pada grup Z_p^* . Untuk lebih jelasnya, tabel 1 memberikan hubungan antara Z_p^* dengan $E(Z_p)$

Tabel 1. Hubungan antara Grup Z_p^* dengan kurva $E(Z_p)$

Label	Grup Z_p^*	Grup Kurva $E(Z_p)$
Elemen	Integer $\{1,2, \dots, p-1\}$	Titik (x,y) pada kurva $E + \mathbf{O}$
Aturan operasi	Perkalian modulo p	Penjumlahan 2 titik
Notasi	Elemen : g, h	Elemen : P, Q
	Perkalian : $g \cdot h$	Perkalian : $P + Q$
	Invers : g^{-1}	Invers : $-P$
	Pembagian : g / h	Pembagian : $P - Q$
	Pemangkatan : g^a	Perkalian : aP
Discrete Logarithm Problem	Diberikan $g \in Z_p^*$ dan $h = g^a \text{ mod } p$, cari a	Diberikan $P \in E(Z_p)$ dan $Q = aP$, cari a

IV. Protokol Pertukaran Kunci (Diffie – Hellman ECC)

Diffie – Hellman pertama kali memperkenalkan algoritma kunci publik pada tahun 1976. Algoritma ini memiliki keamanannya dari kesulitan menghitung logaritma diskrit dalam *finite field*, dibandingkan kemudahan dalam menghitung bentuk eksponensial dalam *finite field* yang sama. Algoritma ini dapat digunakan dalam mendistribusikan kunci publik yang dikenal dengan protokol pertukaran kunci.

Penjelasan protokol pertukaran kunci ini dapat diilustrasikan antara dua orang, misalkan saja **Tera** dan **Jana**, yang keduanya sepakat mengenai bilangan prima yang besar misalkan n dan g dimana g merupakan modulo n . Selanjutnya akan terdapat dua buah integer yang tidak dirahasiakan atau merupakan kunci publik dan dapat didistribusikan dalam saluran bebas. Proses berikutnya dijelaskan dalam tahapan-tahapan di bawah ini :

4.1. Algoritma Pertukaran Kunci Diffie-Hellman

Berikut ini algoritma pertukaran kunci Diffien – Hellman yang diilustrasikan dua orang user Tera dan Jana

1. **Tera** memilih secara acak sebuah bilangan integer x yang besar dan mengirimkannya ke Jana.

$$X = g^x \text{ mod } n$$

2. **Jana** memilih secara acak sebuah bilangan integer y yang besar dan mengirimkannya ke Tera.

$$Y = g^y \text{ mod } n$$

3. **Tera** menghitung nilai $k_1 = Y^x \text{ mod } n$
4. **Jana** menghitung nilai $k_2 = X^y \text{ mod } n$

maka kedua nilai k_1 dan k_2 adalah sama untuk $g^{xy} \text{ mod } n$, sehingga k adalah kunci rahasia Tera dan Jana yang dihitung secara terpisah.

Implementasi *elliptic curve cryptosystem* untuk protokol pertukaran kunci dapat dilakukan dengan menggunakan tahapan pada Diffie-Helman dengan mengubah parameter yang disesuaikan dengan persamaan kurva elips.

4.2. Pertukaran Kunci Diffie Hellman pada *Elliptic Curves Cryptosystem*

Misal diberikan kurva $E : y^2 = x^3 + 2x + 1$ dengan bilangan prima yang dipilih $p=5$, maka himpunan titik-titik pada kurva $E(Z_5) = \{(0,1), (1,3), (3,3), (3,2), (1,2), (0,4)\}$ yang berturut-turut diberi notasi $P_1, P_2, P_3, P_4, P_5, P_6$

1. **Tera** memilih kunci rahasia a , misalkan $a = 2$ dan menghitung kunci publiknya yaitu :

$$P_{\text{Tera}} = a P_1 = 2 P_1 = P_1 + P_1 = P_2 = (1,3)$$

Hasil kunci publik ini dikirm ke **Jana**

2. **Jana** memilih kunci rahasia b , misalkan $b = 3$ dan menghitung kunci publiknya yaitu :

$$P_{\text{Jana}} = a P_1 = 3 P_1 = P_1 + P_1 + P_1 = P_3 = (3,3)$$

Hasil kunci publik ini dikirm ke **Tera**

3. **Tera** dan **Jana** secara terpisah menghitung dari kunci publik yang diterima untuk menghasilkan kunci rahasia yang sama, yaitu :

$$\text{Tera : } S_{\text{Tera_Jana}} = a P_{\text{Jana}} = 2 P_3 = P_6 = (0,4)$$

$$\text{Jana : } S_{\text{Tera_Jana}} = a P_{\text{Tera}} = 3 P_2 = P_6 = (0,4)$$

V. *ElGamal Cryptosystem*

Skema ElGamal merupakan salah satu jenis kriptografi asimetris. Selain digunakan untuk proses enkripsi, skema ini juga dapat digunakan pada tanda tangan digital (*digital signature*). Keamanan pada skema ElGamal diperoleh dari kesulitan menghitung logaritma diskrit dalam *finite field*.

Skema ElGamal memerlukan sepasang kunci yang dibangkitkan dengan memilih sebuah bilangan prima p dan dua buah bilangan random g dan x . Nilai g dan x lebih kecil dari p yang memenuhi persamaan :

$$y = g^x \text{ mod } p \quad (8)$$

Dari persamaan tersebut y, p dan g merupakan kunci publik dan x adalah kunci rahasia. Berikut ini algoritma ElGamal dalam skema enkripsi yang diilustrasikan dengan dua orang user Tera dan Jana.

5.1. Algoritma Enkripsi ElGamal

Berikut ini algoritma enkripsi ElGamal yang diilustrasikan dua orang user Tera dan Jana

1. Diberikan p sebuah bilangan prima dalam Z_p dan α yang merupakan anggota Z_p , dimana p dan α merupakan kunci publik.
2. Setiap user memilih sebuah kunci rahasia a yang merupakan bilangan integer untuk $0 \leq a \leq p-2$.
3. Setiap user menghitung $\beta \equiv \alpha^a \pmod{p}$ yang nilainya akan dikirim.
4. Misalkan Tera akan mengirim pesan $x \in Z_p$, maka dia harus memilih sebuah bilangan k secara random, yaitu $k \in Z_{p-1}^*$ dan mengirimkan pesan terenkripsi ke Jana dengan persamaan :

$$(y_1, y_2) = (\alpha^k \pmod{p}, x\beta^k \pmod{p})$$

5. Untuk melakukan deskripsi, Jana menghitung :

$$y_2(y_1^{a_j})^{-1} \pmod{p}$$

dimana a_j merupakan kunci rahasia Jana.

Implementasi *Elliptic Curves* pada skema enkripsi ElGamal dilakukan dengan menyisipkan *plaintext* pada kurva elips. Hasil penyisipan ini *plaintext* dinyatakan sebagai titik pada kurva E dan komputasi dibentuk pada persamaan kurva E . Selanjutnya dilakukan proses enkripsi seperti dalam algoritma di atas dalam lingkungan parameter kurva elips.

5.2. Skema Enkripsi ElGamal pada *Elliptic Curves Cryptosystem*

Diberikan sebuah bilangan prima pada field Z_p , sebuah kurva elips $E(Z_p)$ dan sebuah titik basis $P \in E$ yang semuanya merupakan konstanta dan merupakan kunci publik.

1. Setiap user memilih acak sebuah bilangan integer a yang akan menjadi kunci rahasianya masing-masing.
2. Setiap *user* mengalikan kunci rahasianya dengan titik basis yang dinyatakan sebagai aP , dimana hasilnya dipublikasikan.

- Misalkan Tera ingin mengirim sebuah pesan m (sebut saja bilangan integer) kepada Jana. Maka pertama yang dia lakukan menyisipkan m ke dalam kurva elips, sehingga dihasilkan sebuah titik P_m pada kurva E .
- Selanjutnya Tera mengenkripsi pesan yang sudah menjadi titik menggunakan persamaan :

$$(C_1, C_2) = (kP, Pm + k(a_J P))$$

dimana k = bilangan integer yang dipilih random oleh Tera

P = titik basis

$a_J P$ = nilai yang diterima dari Jana, yang merupakan hasil perhitungan kunci rahasia Jana a_J dikali dengan titik basis P (lihat no. 2).

Hasil enkripsi di atas menghasilkan pasangan titik C_1, C_2 .

- Proses deskripsi pada *user* Jana dilakukan dengan persamaan :

$$C_2 - a_J(C_1) = Pm + k(a_J P) - a_J(kP) = P_m$$

VI. IMPLEMENTASI DAN HASIL KELUARAN PROGRAM

Implementasi penggunaan ECC pada protokol pertukaran kunci dan skema enkripsi ElGamal dalam program ditulis dalam bahasa C dengan menggunakan compiler Visual C++ versi 6.0.

6.1. Definisi variabel dan konstanta

Variabel dan konstanta untuk keperluan kurva elips didefinisikan dalam file *field2n.h* yang berisi :

```
#define WORDSIZE      (sizeof(int)*8)
#define NUMBITS      160
#define NUMWORD      (NUMBITS/WORDSIZE)
#define UPRSHIFT     (NUMBITS%WORDSIZE)
#define MAXLONG      (NUMWORD+1)
#define MAXBITS      (MAXLONG*WORDSIZE)
#define MAXSHIFT     (WORDSIZE-1)
#define MSB          (1L<<MAXSHIFT)
#define UPRBIT       (1L<<(UPRSIFT-1))
#define UPRMASK      (~(-1L<<UPRSIFT))
#define SUMLOOP(i)   for(i=0; i<MAXLONG; i++)
```

```

typedef short int INDEX;

typedef unsigned long ELEMENT;

/* struktur field2n */
typedef struct {
    ELEMENT      e[MAXLONG];
} FIELD2N;

```

Selain data-data di atas, juga diperlukan sebuah konstanta yang berisi nilai bilangan prima yang dipilih dengan panjang bit tertentu. Bilangan prima ini disusun dalam bentuk polinomial dengan tipe data `FIELD2N`. Konstanta ini ditulis dalam bentuk hexsadesimal. Berikut ini contoh-contoh penulisan bilangan prima yang dipilih :

```

/*FIELD2N poly_prime = {0x00000001,0x00000000,0x00000000,0x00000000,0x00000000,
                        0x00000000,0x00000000,0x00000000,0x00000425}; /*256 bit*/
/*FIELD2N poly_prime = {0x00000001,0x00000000,0x00000000,0x00000000,0x00000000,
                        0x00000000,0x00000000,0x000001B5}; /*224 bit*/
/*FIELD2N poly_prime = {0x00000001,0x00000000,0x00000000,0x00000000,0x00000000,
                        0x00000000,0x00000087}; /*192 bit*/
FIELD2N poly_prime = {0x00000001,0x00000000,0x00000000,0x00000000,0x00000000,
0x0000002D}; /*160 bit*/
/*FIELD2N poly_prime = {0x00000001,0x00000000,0x00000000,0x00000000,0x00000087};
                        /*128 bit*/
/*FIELD2N poly_prime = {0x00000001,0x00000000,0x00000000,0x0000006F}; /*96 bit*/
/*FIELD2N poly_prime = {0x00000001,0x00000000,0x0000001B}; /*64 bit*/
/*FIELD2N poly_prime = {0x00000001,0x0000008D}; /*32 bit*/

```

Perlu diperhatikan bahwa jumlah bit pada konstanta `NUMBITS` harus sama dengan jumlah bit konstanta `poly_prime` yang dipilih. Pengujian bilangan prima dilakukan dengan menggunakan metode Richard Pinch. Idenya dengan menggunakan algoritma **gcd** (*greatest common divisor*) untuk menguji masukan bilangan prima `poly_prime` dalam variabel vektor *v*. Jika faktorisasi `poly_prime` menghasilkan bilangan selain 1 dan bilangan itu sendiri, maka `poly_prime` bukannya bilangan prima (*not irreducible*). Proses selengkapnya dapat dilihat dalam fungsi di bawah ini yang memiliki keluaran 1 untuk prima (*irreducible*) dan 0 untuk bukan prima (*not irreducible*)

```

INDEX irreducible( v)
FIELD2N *v;
{
    FIELD2N    vprm, gcd, x2r, x2rx, temp;
    FIELD2N    sqr_x[NUMBITS+1];
    INDEX      i, r, deg_v, k;

/* uji gcd(v, v') = 1.  jika ya, maka v irreducible. */

    SUMLOOP(i) vprm.e[i] = (v->e[i] >> 1) & DERIVMASK;
    poly_gcd( v, &vprm, &gcd);
    if (gcd.e[NUMWORD] > 1) return(0);
    for (i=0; i<NUMWORD; i++) if (gcd.e[i]) return(0);

/* cari maksimum pangkat */
    deg_v = degreeof(v, NUMWORD);

/* buat tabel vektor x^2k mod v.
   tabel ini digunakan untuk menghitung kuadrat dari x^2^r
*/
    null (&sqr_x[0]);
    sqr_x[0].e[NUMWORD] = 1;
    for (i=1; i<= deg_v; i++)
    {
        mul_x_mod( &sqr_x[i-1], v, &temp);
        mul_x_mod( &temp, v, &sqr_x[i]);
    }

/* uji gcd( x^2^r - x, v) == 1 untuk setiap r <= degreeof(v)/2.
   Set x^2^r = x untuk r = 0.
*/
    null( &x2r);
    x2r.e[NUMWORD] = 2;
    for ( r=1; r <= deg_v/2; r++)
    {
/* kuadratkan x^2^r mod v untuk melihat s^2(x) = s(x^2).
   untuk setiap bit j set dalam x^2^(r-1) jumlahkan dalam x^2j mod v
   untuk menemukan x^2^r.
*/
        null( &x2rx);
        for (i=0; i <= deg_v; i++)
        {
            if(x2r.e[NUMWORD-(i/WORDSIZE)]&(1L<<(i%WORDSIZE)))
                SUMLOOP(k) x2rx.e[k] ^= sqr_x[i].e[k];
        }
/* simpan nilai x^2^r mod v dan hitung x^2^r - x mod v */
        copy( &x2rx, &x2r);
        x2rx.e[NUMWORD] ^= 2;

/* apakah gcd( x^2^r - x, v) == 1?  Jika tidak, keluar dengan hasil
   negatif */
        poly_gcd( &x2rx, v, &gcd);
        if (gcd.e[NUMWORD] > 1) return (0);
        for (i=0; i<NUMWORD; i++) if ( gcd.e[i]) return (0);
    }
    return (1);
}

```

Di bawah ini penggalan program yang digunakan untuk melihat 10 bilangan prima polinom pertama dalam jumlah bit (NUMBITS) yang dipilih. Proses pencariannya dilakukan secara sekuensial pada 32 bit pertama (8 digit heksa) yang dicacah naik 1 angka kemudian diuji menggunakan fungsi `irreducible` di atas.

```

...
random_seed = 0x932b15fe;
count=0;
prime.e[0]=0x00000001;
for (i=1;i<MAXLONG;i++)
    prime.e[i]=0;
printf("Daftar 10 Prima Polinom Pertama Pada %d bit\n",NUMBITS);
puts("-----");
do
{
    prime.e[MAXLONG-1]++;
    if (irreducible(&prime)){print_field("",&prime);
        count++;}
} while (count<10);
...

```

```

E:\POLYECC\DH-ELGAMAL\Debug\DH-ELGAMAL.exe
-----
[1]. Implementasi DH-ECC
[2]. Implementasi ELGAMAL-ECC
[3]. Daftar 10 Prima Polinom Pertama
[0]. Keluar
-----
> 3
Daftar 10 Prima Polinom Pertama Pada 128 bit
-----
1 00000000 00000000 00000000 00000087
1 00000000 00000000 00000000 000000F9
1 00000000 00000000 00000000 0000012F
1 00000000 00000000 00000000 0000013B
1 00000000 00000000 00000000 00000173
1 00000000 00000000 00000000 00000175
1 00000000 00000000 00000000 00000285
1 00000000 00000000 00000000 000002DF
1 00000000 00000000 00000000 000003B1
1 00000000 00000000 00000000 000003F9
-----

```

Gambar 9. Keluaran program 10 prima polinom pertama pada 128 bit

Hasil selengkapnya untuk pencetakan bilangan prima pada jumlah bit 32, 64, 96, 128 dan 160 bit diperlihatkan pada tabel 2. Pada tabel tersebut dicatat hanya 5 buah polinom prima pertama.

Tabel 2. Daftar prima polinom (5 urutan pertama)

Jumlah Bit	Nilai Prima Polinom dalam Heksa desimal (5 urutan pertama)					
	h ₄₇ -h ₄₀	h ₃₉ -h ₃₂	h ₃₁ -h ₂₄	h ₂₃ -h ₁₆	h ₁₅ -h ₈	h ₇ -h ₀
32 bit					00000001	0000008D
					00000001	000000AF
					00000001	000000C5
					00000001	000000F5
					00000001	00000125
64 bit				00000001	00000000	0000001B
				00000001	00000000	0000001D
				00000001	00000000	0000008D
				00000001	00000000	000000F5
				00000001	00000000	00000173
96 bit			00000001	00000000	00000000	0000006F
			00000001	00000000	00000000	000000DD
			00000001	00000000	00000000	0000017F
			00000001	00000000	00000000	000001D9
			00000001	00000000	00000000	00000257
128 bit		00000001	00000000	00000000	00000000	00000087
		00000001	00000000	00000000	00000000	000000F9
		00000001	00000000	00000000	00000000	0000012F
		00000001	00000000	00000000	00000000	0000013B
		00000001	00000000	00000000	00000000	00000173
155 bit		80000000	00000000	00000000	00000000	000000B1
		80000000	00000000	00000000	00000000	0000013B
		80000000	00000000	00000000	00000000	0000016D
		80000000	00000000	00000000	00000000	0000024F
		80000000	00000000	00000000	00000000	00000359
160 bit	00000001	00000000	00000000	00000000	00000000	0000002D
	00000001	00000000	00000000	00000000	00000000	00000039
	00000001	00000000	00000000	00000000	00000000	0000008B
	00000001	00000000	00000000	00000000	00000000	000001BF
	00000001	00000000	00000000	00000000	00000000	000001E9

6.2. Implementasi Pertukaran Kunci (Delffie-Hellman Ellitic Curve Crytosystem)

Tahapan-tahapan dalam simulasi pertukaran kunci protokol Diffie-Hellman yaitu sebagai berikut :

1. Pembetulan kurva elips dan pemilihan titik basis secara random.

Public_Curve : $y^2 + xy = x^3 + a_6$: untuk $a_2=0$
atau

Public_Curve : $y^2 + xy = x^3 + a_2x^2 + a_6$: untuk $a_2 \neq 0$
Nilai a_2 dan a_6 diperoleh dari hasil random

2. Pembangunan kunci rahasia di setiap user.

User Tera : **private1**

User Jana : **private2**

Nilai **private1** dan **private2** diperoleh dari hasil random pada kurva

3. Pembangkitan kunci publik di setiap user, dan setiap user saling bertukar kunci publik.

User Tera : **public1** = **private1** * titik basis

User Jana : **public2** = **private2** * titik basis

4. Pembentukan kunci rahasia bersama, yaitu :

User Tera : **key1** = **private1** * **public2**

User Jana : **key2** = **private2** * **public1**

Pada tahapan ini harus dibuktikan bahwa **key1** = **key2**

6.2.1. Pembentukan Kurva Elips dan Pemilihan Titik Secara Random

Persamaan kurva elips dipilih bentuk non supersingular. Bentuk ini menghasilkan banyak kurva dan memungkinkan kurva random. Dalam program diimplementasikan dua buah kondisi a_2 yaitu :

$$a_2 = 0 \rightarrow E : y^2 + xy = x^3 + a_6$$

$$a_2 \neq 0 \rightarrow E : y^2 + xy = x^3 + a_2x^2 + a_6$$

Isi variabel a_2 dan a_6 disimpan dalam tipe data CURVE, yaitu :

```
typedef struct
{
    INDEX    form;
    FIELD2N  a2;
    FIELD2N  a6;
} CURVE;
```

Sedangkan pemilihan bentuk kurva ditulis dalam fungsi di bawah ini :

```
/* bentuk kurva kondisi a2=0*/
void rand_curve_form_0 ( curv)
CURVE *curv;
{
    curv->form = 0;
    random_field( &curv->a6);
    null( &curv->a2);
}

/* bentuk kurva kondisi a2!=0*/
void rand_curve_form_1 ( curv)
CURVE *curv;
{
```

```

    curv->form = 0;
    random_field( &curv->a6);
    random_field( &curv->a2);
}

```

Untuk pemilihan titik kurva basis dapat dilihat pada fungsi di bawah ini :

```

void rand_point( point, curve)
POINT *point;
CURVE *curve;
{
    FIELD2N    rf;
    /* random bilangan prima dalam FIELD2N */
    random_field( &rf);
    /* merangkai hasil dalam variabel-variabel kurva */
    poly_embed( &rf, curve, NUMWORD, rf.e[NUMWORD]&1, point);
}

```

Proses random menggunakan generator *George Marsaglia* yang menghasilkan *pseudo random* 32 bit dengan periode sekitar 2^{250} . Dua buah array disediakan untuk menyimpan nilai *carry* pada alamat pertama dan *random* 16 bit disimpan dalam elemen ke-1 sampai ke-8. Selanjutnya hasil random ini digeser ke posisi 2 sampai 9 dan sebuah nilai *carry* baru dan bilangan yang dihasilkan ditempatkan pada elemen ke-0 dan 1. Proses ini dilakukan sampai dengan 32 bit terisi semua.

6.2.2. Pembentukan Kunci Rahasia Secara Random

Pemilihan kunci rahasia dilakukan secara random dengan memanggil fungsi **random_field** yang disesuaikan dengan jumlah bit yang digunakan. Dalam program disimulasikan pertukaran kunci antara dua orang, sehingga pemilihan kunci rahasia dilakukan sebanyak 2 kali, yaitu :

```

/* Pemilihan random kunci rahasia user Tera */
random_field(&privatel);
print_field("Kunci Rahasia Tera :", &privatel);

/* Pemilihan random kunci rahasia user Jana */
random_field(&private2);
print_field("Kunci Rahasia Jana :", &private2);

```

6.2.3. Pembentukan Kunci Publik

Pada tahapan ini setiap user akan membentuk kunci publik yang hasilnya akan dikirim ke user lain. Variabel yang digunakan untuk menghitung kunci publik adalah :

- variabel **Base** yang merupakan basis titik kurva.
- variabel kunci rahasia disetiap user (**privat1** untuk user Tera dan **privat2** untuk user Jana)
- variabel **e** yang berisi fungsi kurva

Kunci publik R diperoleh dari hasil perkalian $R = k P$ dimana k merupakan bilangan integer (basis 2 yang merupakan variabel **base**) dan P merupakan titik dalam kurva elips (kunci rahasia). Asumsi bahwa k dapat direpresentasikan dalam proyeksi *bit field* yang sama sebagai nilai x, y dan z pada P . Proses selengkapnya dapat dilihat dalam fungsi di bawah ini.

```
void poly_elptic_mul(k, p, r, curv)
FIELD2N      *k;
POINT *p, *r;
CURVE *curv;
{
    char      blncd[NUMBITS+1];
    INDEX     bit_count, i;
    ELEMENT   notzero;
    FIELD2N   number;
    POINT     temp;

/* memastikan parameter input k tidak sama dengan nol dan
mengembalikan titik (jika ada) pada infinity */

    copy( k, &number);
    notzero = 0;
    SUMLOOP (i) notzero |= number.e[i];
    if (!notzero)
    {
        null (&r->x);
        null (&r->y);
        return;
    }

/* mengubah k (bilangan) menjadi representasi non-adjacent */

    bit_count = 0;
    while (notzero)
    {
        /* jika bilangan ganjil, buat 1 atau -1 dari 2 bit terakhir */
        if ( number.e[NUMWORD] & 1 )
        {
            blncd[bit_count] = 2 - (number.e[NUMWORD] & 3);

            if ( blncd[bit_count] < 0 )
```

```

        {
            for (i=NUMWORD; i>=0; i--)
            {
                number.e[i]++;
                if (number.e[i]) break;
            }
        }
    else
        blncd[bit_count] = 0;

/* bagi bilangan dengan 2, naikan bit counter */

    number.e[NUMWORD] &= ~0 << 1;
    rot_right( &number);
    bit_count++;
    notzero = 0;
    SUMLOOP (i) notzero |= number.e[i];
}

bit_count--;
copy_point(p,r);
while (bit_count > 0)
{
    poly_edbl(r, &temp, curv);
    bit_count--;
    switch (blncd[bit_count])
    {
        case 1: poly_esum (p, &temp, r, curv);
                break;
        case -1: poly_esub (&temp, p, r, curv);
                break;
        case 0: copy_point (&temp, r);
    }
}
}

```

Nilai keluaran pada variabel *r* akan dikirim (digunakan) oleh user lain untuk menentukan kunci rahasia baru yang nilai akan sama antara kedua user tersebut.

6.2.4. Menghitung Kunci Rahasia Bersama

Proses terakhir adalah menghitung kunci rahasia baru. Kunci rahasia yang baru ini harus sama satu sama lainnya. Proses ini harus memperlihatkan bahwa hasil perkalian kunci rahasia user Tera dengan kunci publik yang diterimanya harus sama dengan hasil perkalian kunci rahasia user Jana dengan kunci publik yang diterimanya. Untuk lebih jelasnya seluruh variabel yang diperoleh dari hasil proses sebelumnya diperlihatkan dalam tabel di bawah ini :

Tabel 3. Variabel yang terbentuk pada proses pertukaran kunci

Jenis Variabel	Nama Variabel Pada User Tera	Nama Variabel Pada User Jana
Kunci rahasia	private1	private2
Kunci publik	public1	public2
Kunci publik yang diterima	public2	public1
Kunci Rahasia Bersama	key1	key2

Pada tabel 3, maka pada proses ini akan dihitung :

User Tera : $key1 = private1 * public2$

User Jana : $key2 = private2 * public1$

sehingga harus diperlihatkan bahwa $key1 = key2$. Untuk implementasinya diperlihatkan dalam penggalan program di bawah ini

...

```
printf("Hasil akhir, Tera dan Jana memiliki kunci rahasia bersama\n");
printf("-----\n\n");
```

```
DH_key_share( &Base, &rnd_crv, &P2, &private1, &key1);
print_field("Kunci rahasia Ani :", &key1);
DH_key_share( &Base, &rnd_crv, &P1, &private2, &key2);
print_field("Kunci rahasia Ali : ", &key2);P
```

...

Proses perkalian antara kunci rahasia dan kunci publik yang diterima dari user lain dilakukan dengan memanggil fungsi `DH_key_share`, yang di dalam fungsinya terdapat pemanggilan fungsi `poly_elptic_mul`.

```
void DH_key_share(Base_point, E, their_public, my_private,
                 shared_secret)
POINT *Base_point, *their_public;
CURVE *E;
FIELD2N *my_private, *shared_secret;
{
    POINT temp;

    poly_elptic_mul( my_private, their_public, &temp, E);
    copy (&temp.x, shared_secret);
}
```

```

"E:\POLYECC\DH-ELGAMAL\Debug\DH-ELGAMAL.exe"
Implementasi Pertukaran Kunci <Diffie-Hellman Elliptic Curves Cryptosystem>
-----
Jumlah Bit      : 160 bit
Prima Polinom  : 1 00000000 00000000 00000000 00000000 0000002D
-----

Pembentukan Kurva dan Titik Basis
-----
Bentuk Kurva E : y^2 + xy = x^3 + a_6
a6 :           0 63323EAB 10FC68F8 254D4D11 D2D518F2 9979DD24
Titik Basis :
X :           0 5D0BE8BB A913FCDB 91EDEC60 4DA6D486 295D85AD
Y :           0 F12BAFE2 D666C798 3228F1E5 5B63A488 954D7A3E
-----

Pembentukan kunci rahasia di setiap user
-----
Kunci Rahasia Tera : 0 3A0F94F6 E0CAF9A7 2D189F04 8591C5E5 3935D4DC
Kunci Rahasia Jana : 0 741AA073 FEE656E5 B5B92389 1E324560 5C6E42E9
-----

Pembentukan kunci publik di setiap user
-----
Kunci publik Tera :
X :           0 B9CFACE1 7339964D B0952BFE 421092EE 83C4D591
Y :           0 C5519A65 6CAB13C9 4C554B2B CAE44704 506B580C
Kunci publik Jana :
X :           0 375803B2 429EE4EC 45C38A84 6747D256 2FE2C7F2
Y :           0 29D39FC4 FCE70E1D 1BAEF593 70B4EDFF 5E5DCD25
-----

Hasil akhir, user memiliki kunci rahasia yang sama
-----
Kunci rahasia bersama Tera : 0 A91D7DE1 26B96C3C A16478A6 FB7FF2EB 0F564851
Kunci rahasia bersama Jana : 0 A91D7DE1 26B96C3C A16478A6 FB7FF2EB 0F564851
-----

Waktu proses      : 8.2320 detik
Waktu 4 perkalian : 8.0120 detik
-----

```

Gambar 10. Keluaran program pertukaran kunci

Hasil program di atas memperlihatkan antara dua user Tera dan Jana yang memiliki kunci rahasia bersama. Hal ini berarti telah memenuhi persamaan :

$$\text{User Tera : } \text{key1} = \text{private1} * \text{public2}$$

$$\text{User Jana : } \text{key2} = \text{private2} * \text{public1}$$

dimana $\text{key1} = \text{key2}$.

6.3. Implementasi Enkripsi ElGamal (ElGamal *Elliptic Curves Cryptosystem*)

Skema enkripsi ElGamal disimulasikan dengan pengiriman pesan berupa satu buah bilangan integer yang besar sesuai dengan jumlah bit yang digunakan. Skenario yang digunakan yaitu pengiriman pesan dari user Tera ke user Jana, dan akan diperlihatkan proses enkripsi dan deskripsi datanya.

Tahapan-tahapan dalam simulasi skema enkripsi ElGamal Elliptic Curves Cryptosystem yaitu sebagai berikut :

1. Pembentukan kurva elips dan pemilihan titik basis secara random.

Public_Curve : $y^2 + xy = x^3 + a_6$: untuk $a_2=0$
atau
Public_Curve : $y^2 + xy = x^3 + a_2x^2 + a_6$: untuk $a_2 \neq 0$
Base_Point
Nilai a_2 dan a_6 diperoleh dari hasil random

2. Pemilihan kunci rahasia dan kunci publik di user penerima (Jana)

private2 : kunci rahasia Jana
Their_Public = $private2 * Base_Point$

3. Pembuatan pesan berupa bilangan integer dan titik dalam kurva secara random.

send_data : data yang akan dikirim (plaintext)
Random_Point : titik random dalam kurva

4. Proses enkripsi plaintext menjadi ciphertext (**Hidden_data**) dan hasilnya bersama dengan titik random (**Random_Point**) dikirim ke user lain.
5. Proses deskripsi ciphertext menjadi plaintext. Pada tahapan ini harus dibuktikan data yang diterima harus sama dengan data yang dikirim.

Seperti halnya pada simulasi DH-ECC, persamaan kurva elips dipilih bentuk non supersingular. Bentuk ini menghasilkan banyak kurva dan memungkinkan kurva random dalam bentuk 2 buah persamaan yang dengan kondisi $a_2 = 0$ atau $a_2 \neq 0$.

Pada sisi penerima dipilih kunci rahasia secara acak dan membentuk kunci publik yang akan digunakan oleh si pengirim untuk membentuk sembarang titik pada kurva yang bersama dengan data terekripsi dikirim ke si penerima. Sedangkan untuk keperluan enkripsi dan deskripsinya, disimulasikan dengan pembangkitan data atau pesan berupa bilangan integer besar yang sesuai dengan jumlah bit yang digunakan. Proses-proses ini dapat dilihat dalam penggalan program di bawah ini :

```

...
printf("\n-----\n");
printf("Pembentukan Kunci Rahasia dan Kunci Public di User Jana\n");
printf("-----\n\n");

random_field(&private2);
print_field("Kunci Rahasia Jana :", &private2);

poly_elptic_mul( &private2, &Base_Point, &Their_public, &Public_Curve);
print_point("Kunci Public  Jana :", &Their_public);

printf("\n-----\n");
printf("Pembentukan data integer (pesan) Secara Acak di user Tera\n");
printf("-----\n\n");
random_field( &send_data);
print_field("Pesan yang Dikirim : ", &send_data);
...

```

6.3.1. Proses Enkripsi Data

Proses enkripsi pada elliptic curves cryptosistem dilakukan dengan mengubah pesan menjadi titik yang terdapat dalam kurva. Dalam simulasinya proses enkripsi ini menghasilkan data yang terenkripsi (*chipertext*) dan sebuah titik random yang nantinya akan dikirim ke user lain.

Data *chipertext* yang akan dikirim berupa pasangan titik pada kurva, yaitu (**Random_point**, **Hidden_data**) yang keduanya diperoleh dari :

1. **Random_Point** = **random_value** * **Base_point**
2. **Hidden_data** = **Raw_point** + **random_value** * **Their_public**

dimana :

- random_value** = kunci rahasia pengirim
- Base_point** = titik basis
- Raw_point** = titik data yang akan dikirim
- Their_public** = kunci publik

Proses enkripsi selengkapnya dapat dilihat pada fungsi **send_elgamal** di bawah ini.

```

/* Pengiriman pesan (plaintext) ke user lain menggunakan
protokol ElGamal. Enkripsi pesan menjadi chipertext dan
mengirimkannya beserta titik acak ke user lain*/

void send_elgamal(Base_point, Base_curve, Their_public, raw_data,
                  Hidden_data, Random_point)
FIELD2N *raw_data;

```

```

POINT    *Base_point, *Their_public, *Hidden_data, *Random_point;
CURVE    *Base_curve;
{
    FIELD2N random_value;
    POINT  hidden_point, raw_point;

/* buat titik acak pada kurva */
    random_field (&random_value);
    poly_elptic_mul (&random_value, Base_point, Random_point,
                    Base_curve);

/* mengubah data yang akan dikirim menjadi titik pada kurva */
    poly_embed( raw_data, Base_curve, 0, 0, &raw_point);

/* enkripsi menggunakan kunci publik */
    poly_elptic_mul( &random_value, Their_public, &hidden_point,
                    Base_curve);
/* penjumlahan raw_point dan hidden_point */
    poly_esum( &hidden_point, &raw_point, Hidden_data, Base_curve);
}

```

Fungsi `random_field` dan `poly_eliptic_mul` yang pertama digunakan untuk menghasilkan titik acak pada kurva `Random_point` yang akan diikutkan dalam pengiriman bersama dengan pesan yang terenkripsi. Proses pengubahan data integer menjadi titik dalam kurva dilakukan dengan pemanggilan fungsi `poly_embed` yang menghasilkan variabel `raw_point`. Fungsi `poly_eliptic_mul` ke-2 diperlukan untuk menghasilkan `hidden_point`. Proses akhir dari enkripsi adalah menjumlahkan `raw_point` dengan `hidden_point` dengan menggunakan fungsi `poly_esum`.

6.3.2. Proses Deskripsi Data

Proses deskripsi dilakukan untuk mengubah data chiphertext menjadi data plaintext. pada elliptic curves cryptosistem dilakukan dengan mengubah pesan menjadi titik yang terdapat dalam kurva. Dalam simulasinya proses deskripsi ini mengubah pasangan titik (`Random_point, Hidden_data`) menjadi data integer yang ditampung dalam variabel `get_data`, sehingga harus diperlihatkan bahwa data yang dikirim sama dengan data yang diterima atau `send_data = get_data`. Proses selengkapnya dapat dilihat dalam fungsi `receive_elgama1` di bawah ini.

```

/* Menerima data (chiper text) dari user lain menggunakan protokol
   ElGamal. Deskripsi data menjadi plan text */

void receive_elgamal(Base_point, Base_curve, my_private, Hidden_data,
                    Random_point, raw_data)
FIELD2N *my_private, *raw_data;
POINT *Base_point, *Hidden_data, *Random_point;
CURVE *Base_curve;
{
    POINT hidden_point, raw_point;

/* deskripsi chiper text menggunakan kunci rahasia dan titik acak */

    poly_elptic_mul( my_private, Random_point, &hidden_point,
                    Base_curve);
    poly_esub( Hidden_data, &hidden_point, &raw_point, Base_curve);
    copy(&raw_point.x, raw_data);
}

```

Fungsi `poly_eliptic_mul` digunakan untuk menghasilkan nilai `hidden_point` dengan mengalikan kunci rahasia penerima `my_private` dengan `Random_point`. Selanjutnya nilai `raw_point` diperoleh dengan mengurangi `Hidden_data` dengan `hiddent_point` yang menggunakan fungsi `poly_esub`. Bilangan integer yang diinginkan diperoleh dengan mengambil komponen x pada variabel `row-point` yang menggunakan fungsi `copy`. Nilai inilah yang mengisi variabel `get_data`, sehingga dapat diperlihatkan bahwa `get_data = send_data`.

Pembuktian `get_data = send_data` dapat dilihat dari hasil keluaran program skema enkripsi ElGamal pada gambar 11.

```

"E:\POLYECC\DH-ELGAMAL\Debug\DH-ELGAMAL.exe"
Skenario : Pengiriman Pesan (bilangan integer yang besar) dari Tera ke Jana
-----
Jumlah Bit           : 160 bit
Bilangan Prima Polinom : 1 00000000 00000000 00000000 00000000 0000002D
-----

Pembentukan Kurva dan Titik Basis
-----
Bentuk Kurva E :  $y^2 + xy = x^3 + a_6$ 
a6 :           0 6DA2BBDB ED7BC65C 5F0E19D E1E09E42 A739747B
Titik Basis :
X :           0 11ABE03D 5034328D 4FD88445 27C47B8E BBAC3D20
Y :           0 35EFAB38 470B8BE2 FFB2C223 C5FF7DF2 1F64419
-----

Pembentukan Kunci Rahasia dan Kunci Public di User Jana
-----
Kunci Rahasia Jana : 0 43F530E4 D16DCA14 2C0A25AD 99D557C3 D4FB5BB9
Kunci Public  Jana :
X :           0 ED866458 17D3BEA6 1663D37E 27F2F984 71D16B59
Y :           0 E5F43956 3F1D765E 65377ECC 3990D4A9 2EBF9F10
-----

Pembentukan Pesan (bilangan integer yang besar) Secara Acak di user Tera
-----
Pesan yang Dikirim : 0 F9378CD1 842B7CBF 6B67D05F A3B25CBC 306378A7
-----

Proses Enkripsi Pesan (bentuk titik dalam kurva) dan Mengirimkannya ke Jana
-----
Pesan Terenkripsi :
X :           0 282A6A78 5E461049 3ED90BAF 293BA365 2E5D9ECD
Y :           0 B9E72B3F 8347EBAC 7F44B326 51AA2739 380635D
Titik Acak :
X :           0 C44E2F9E FB47F24F B545F39B EA84FDA8 8379D40D
Y :           0 43E1747 B3BE1BE2 A0A57919 E85C64C2 393E95D1
-----

Proses Deskripsi Pesan di User Jana
-----
Pesan yang Dikirim : 0 F9378CD1 842B7CBF 6B67D05F A3B25CBC 306378A7
Pesan yang Diterima: 0 F9378CD1 842B7CBF 6B67D05F A3B25CBC 306378A7
-----

Waktu proses       : 8.2320 detik
Waktu enkripsi     : 3.8260 detik
Waktu deskripsi    : 1.9930 detik

```

Gambar 11. Keluaran program skema enkripsi ElGamal

6.4. Waktu Komputasi

Secara keseluruhan waktu yang dibutuhkan untuk mensimulasikan proses baik pada pertukaran kunci atau skema enkripsi-deskripsi, lebih banyak dipengaruhi pada perkalian titik polinomial pada kurva elips. Proses ini ditangani oleh fungsi `poly_elptic_mul`.

Sebagai contoh di bawah ini diperlihatkan waktu proses yang dicatat dari hasil simulasi pertukaran kunci Diffie-Hellman :

Jumlah bit : 160 bit

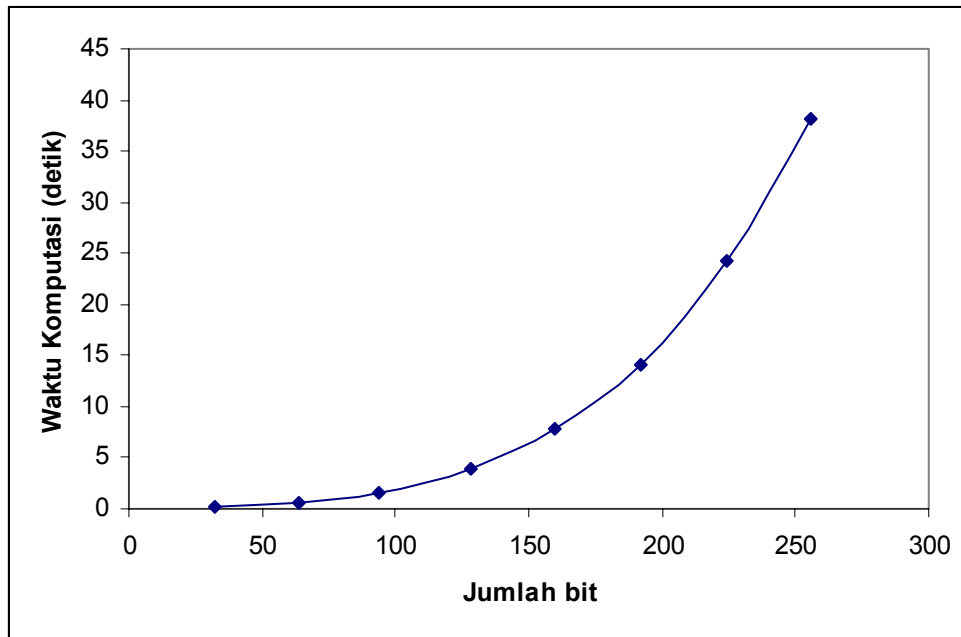
Kurva E : $y^2 + xy = x^3 + a_2x^2 + a_6$

Total waktu (waktu yang dicatat dari awal sampai akhir simulasi) : 7,691 detik

Total waktu perkalian dua titik kurva (4 buah perkalian) : 7,36 detik

dari hasil tersebut menunjukkan hampir 96% dari waktu keseluruhan diperlukan hanya untuk mengalikan titik kurva elips (sebanyak 4 buah perkalian).

Pengaruh jumlah bit terhadap waktu komputasi perkalian diperlihatkan dalam grafik di bawah ini.



Gambar 12. Grafik waktu komputasi terhadap kenaikan jumlah bit

Persamaan kurva elips E dalam implementasinya dipilih dua bentuk, yaitu :

1. $E : y^2 + xy = x^3 + a_2 * x^2 + a_6$

2. $E : y^2 + xy = x^3 + a_6$

Pada tabel 4 diperlihatkan pengaruh pemilihan kurva elips terhadap waktu komputasi.

Tabel 4. Perbandingan waktu komputasi terhadap bentuk kurva

Jumlah Bit	Waktu Komputasi pada Kurva (detik)	
	$y^2 + xy = x^3 + a_6$	$y^2 + xy = x^3 + a_2 * x^2 + a_6$
32	0,12	0,13
64	0,511	0,531
94	1,583	1,632
128	3,936	4,005
160	7,881	8,192
192	13,991	14,611
224	24,315	24,505
256	38,134	38,265

Pada tabel di atas, bentuk kurva $E : y^2 + xy = x^3 + a_2 * x^2 + a_6$ relatif memberikan waktu komputasi lebih lama meskipun tidak begitu signifikan. Sedangkan untuk proses enkripsi waktu komputasinya lebih lama dibandingkan waktu deskripsi. Berdasarkan tabel 5, rata-rata waktu komputasi deskripsi 2 kali lebih cepat dibandingkan enkripsi.

Tabel 5. Perbandingan waktu komputasi terhadap bentuk kurva

Jumlah Bit	Waktu Komputasi (detik)		Rasio
	Enkripsi	Deskripsi	
32	0,05	0,02	2,50
64	0,28	0,12	2,33
94	0,871	0,521	1,67
128	1,903	0,961	1,98
160	4,126	2,113	1,95
192	7,751	3,585	2,16
224	12,267	5,999	2,04
256	18,737	9,564	1,96

VII. TINJAUAN KEAMANAN

Pada tabel 6 diperlihatkan perbandingan waktu yang dibutuhkan untuk memecahkan kriptosistem ECC jika dibandingkan dengan algoritma simetris, RSA dan DSA. Nilai dihitung menggunakan satuan MIPS tahun, yaitu waktu perhitungan dalam satu tahun menggunakan satu mesin untuk melaksanakan satu juta instruksi per detik.

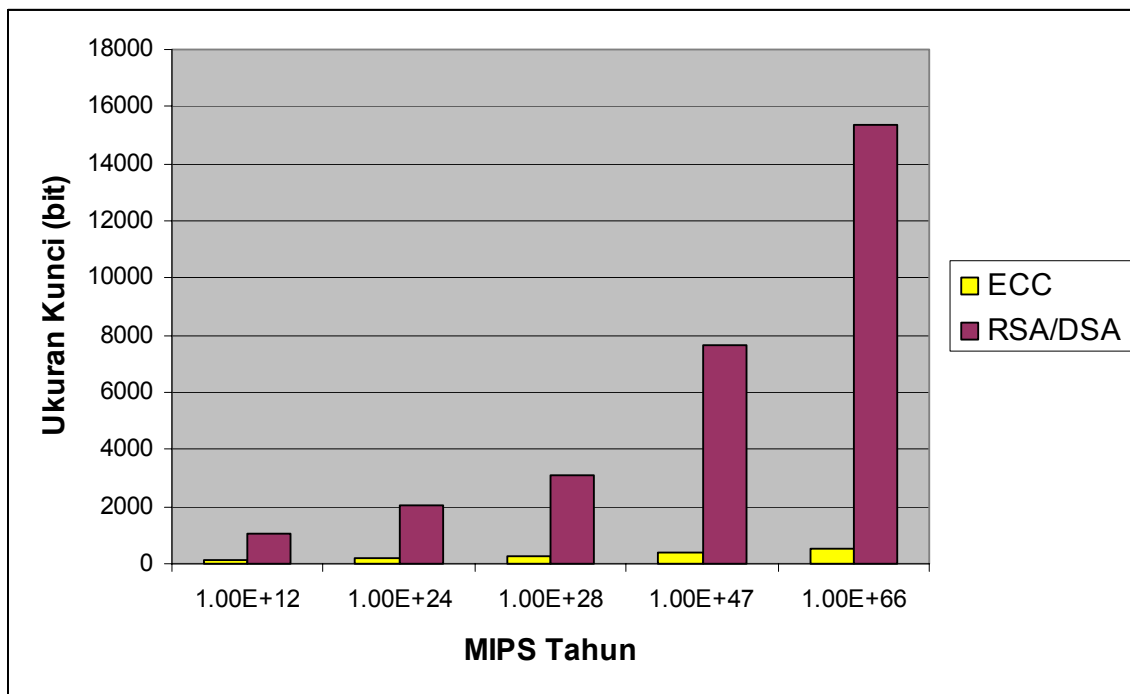
Pada kolom terakhir tabel 6 menunjukkan umur proteksi. Sebagai contoh perbandingan sampai saat ini nilai 10^{12} MIPS tahun merupakan nilai yang masih aman.

Pada nilai waktu tersebut, jumlah bit ECC hanya memerlukan 160 bit yang jauh lebih kecil dibandingkan dengan RSA dan DSA yang memerlukan 1024 bit.

Tabel 6. Perbandingan waktu untuk memecahkan kunci

Jumlah Bit		Waktu Untuk Memecahkan Kunci (MIPS Tahun)	Umur Proteksi
ECC	RSA/DSA		
160	1024	1,00E+12	Sampai 2010
224	2048	1,00E+24	Sampai 2030
256	3072	1,00E+28	Diatas 2031
384	7680	1,00E+47	
512	15360	1,00E+66	

Pada gambar di bawah ini memperlihatkan semakin besar panjang kunci, maka selisih lebar kunci yang digunakan semakin besar. Sebagai contoh untuk nilai keamanan 10^{66} MIPS tahun, ECC hanya membutuhkan 512 bit sedangkan RSA/DSA membutuhkan 15360 bit. Dengan kata lain RSA/DSA membutuhkan 30,5 kali dari jumlah bit kunci ECC untuk nilai 10^{66} MIPS tahun.



Gambar 13. Grafik perbandingan tingkat keamanan ECC dengan RSA/DSA

VIII. Simpulan

Masalah logaritma diskrit kurva elips dapat digunakan sebagai kriptosistem asimetris. Struktur kurva elips digunakan sebagai grup operasi matematis untuk melangsungkan proses enkripsi dan deskripsi. Melalui pembentukan kurva elips pada bilangan prima yang besar, maka setiap pemilihan bilangan prima yang merupakan kunci rahasia dapat dibentuk sebuah titik kurva elips. Titik ini merupakan nilai kunci yang dapat dipublikasikan. Kombinasi kedua nilai tersebut merupakan kunci yang sulit ditebak yang merupakan titik ketiga hasil pemetaan aturan penjumlahan pada kurva elips.

Keunggulan dari kriptosistem kurva elips adalah proses transformasi *plaintext* menjadi titik-titik dalam kurva elips sebelum dilakukan enkripsi. Proses enkripsinya dilakukan dengan menggunakan aturan penjumlahan pada kurva elips. Proses ini tentunya akan memberikan tingkat keamanan yang lebih baik.

Hasil implementasi yang telah dilakukan meskipun sederhana namun cukup memberikan sebuah ilustrasi cara kerja dari kriptosistem kurva elips. Hasilnya telah menunjukkan bahwa hasil pertukaran kunci antara dua user menggunakan protokol Diffie-Hellman memberikan titik ketiga (kunci rahasia bersama) yang sama antara kedua user tersebut. Dalam skema enkripsi ElGamal juga telah diperlihatkan bahwa pesan yang terenkripsi dapat dikembalikan dalam proses deskripsinya.

Waktu komputasi lebih dominan dipengaruhi oleh proses untuk menghitung aturan penjumlahan dua buah titik pada kurva elips. Sementara hasil pada skema enkripsi memperlihatkan proses enkripsi membutuhkan waktu komputasi lebih lama dibandingkan dengan proses deskripsi.

Kriptosistem kurva elips memberikan tingkat keamanan yang lebih baik dibandingkan dengan algoritma asimetris lainnya seperti RSA dan DSA. Hasil tinjauan pustaka memperlihatkan untuk tingkat keamanan yang sama (MIPS tahun yang sama) kriptosistem kurva elips memerlukan jumlah bit kunci yang jauh lebih sedikit dibandingkan dengan RSA atau DSA. Hal ini tentunya kriptosistem kurva elips dapat menjadi pilihan yang baik untuk membangun sistem kriptografi yang memiliki tingkat keamanan yang tinggi.

Daftar Pustaka

- Abdul Kadir, Pemrograman Visual C++, ANDI, Yogyakarta, 2004
- Bruce Schneier, *Applied Cryptography – Second Edition*. John Wiley & Sons, Inc., 1996.
- D.R.Hankerson, D.G.Hoffman, D.A.Leonard, C.C.Lindner, K.T. Phelps, C.A. Rodger, J.R. Wall, *Coding Theory and Chryptography The Essensials – Second Edition, Revised and Expand*, Marcel Dekker, Inc., 2000.
- Mugino Saeki, *Elliptic Curve Cryptosystems*, School of Computer Science McGill University, Montreal, 1997
- Onno W. Purbo, *Mengenal eCommerce*. PT Elex Media Komputindo, Jakarta, 2001.
- Vipul Gupta, Douglas Stebila, *Speeding up Secure Web Transaction Using Elliptic Curve Cryptography*, Sun Microsystem, Inc., 2000

LAMPIRAN SOURCE PROGRAM

```
/*
 *
 *   Program Implemenatasi :
 *   1. Protokol pertukaran kunci Diffie-Hellman
 *      (Diffie-Hellman Elliptic Curves Cryptosystem DH-ECC)
 *
 *   2. Skema Enkripsi ElGamal
 *      (ElGamal Elliptic Curves Cryptosystem ElGamal-ECC)
 *
 */

#include <sys/timeb.h>
#include <stdlib.h>
#include <stdio.h>
#include <dos.h>
#include <time.h>
#include "field2n.h"
#include "poly.h"
#include "eliptic.h"

unsigned long random_seed;
extern  FIELD2N poly_prime;

#include <string.h>

static short induk1[10];
static short induk2[10];
static short mStart=1;

#define m16Long 65536L          /* 2^16 */
#define m16Mask 0xFFFF        /* mask untuk 16 bit lower */
#define m15Mask 0x7FFF        /* mask untuk 15 bit lower */
#define m31Mask 0x7FFFFFFF    /* mask untuk 31 bit */
#define m32Double 4294967295.0 /* 2^32-1 */

/*
 * Pembangkit bilangan acak yang untuk nilai 32 bit dengan periode
 * sekitar 2^250. array induk1 dan induk2 menyimpan nilai carry pada
 * elemen pertama, dan mengacak bilangan 16 bit dalam element 1 - 8.
 * Bilangan acak ini dipindahkan ke elemen 2 - 9. Nilai carry baru dan
 * bilangan dibangkitkan dan ditempatkan pada elemen 0 dan 1. Array
 * induk1 dan induk2 diisi dengan nilan bilangan acak.
 * Keluaran :
 *   Sebuah bilangan acak 32 bit dengan mengkombinasikan keluaran
 *   2 pembangkit dan dikeluarkan dalam pointer *pSeed.
 */

void induk(unsigned long *pSeed)
{
    unsigned long  number,
                  number1,
                  number2;
    short          n,
```

```

        *p;
unsigned short sNumber;

/* Inisialisasi induk ke-i dengan nilai random 9*/
if (mStart) {
    sNumber= *pSeed&m16Mask; /* 16 bit low */
    number= *pSeed&m31Mask; /* 31 bit */

    p=induk1;
    for (n=18;n--;) {
        number=30903*sNumber+(number>>16);

        /* perkalian dengan carry */
        *p+=sNumber=number&m16Mask;
        if (n==9)
            p=induk2;
    }
    /* buat carry 15 bit */
    induk1[0]&=m15Mask;
    induk2[0]&=m15Mask;
    mStart=0;
}

/* pindahkan isi elemen 1 - 8 ke 2 - 9 */
    memmove(induk1+2,induk1+1,8*sizeof(short));
memmove(induk2+2,induk2+1,8*sizeof(short));

/* simpan nilai carry ke number ke-i */
number1=induk1[0];
number2=induk2[0];

/* bentuk kombinasi linear */

number1+=1941*induk1[2]+1860*induk1[3]+1812*induk1[4]+1776*induk1
    [5]+1492*induk1[6]+1215*induk1[7]+1066*induk1[8]+12013*
    induk1[9];

number2+=1111*induk2[2]+2222*induk2[3]+3333*induk2[4]+4444*induk2
    [5]+5555*induk2[6]+6666*induk2[7]+7777*induk2[8]+9272*
    induk2[9];

/* simpan msb number ke-1 sebagai carry baru */
induk1[0]=number1/m16Long;
induk2[0]=number2/m16Long;
/* simpan lsb number ke-i kedalam induk ke-i[1] */
induk1[1]=m16Mask&number1;
induk2[1]=m16Mask&number2;

/* kombinasi 2 bilangan acak 16 bit ke dalam 32 bit */
*pSeed=((long)induk1[1])<<16+(long)induk2[1];
}

/* Pembentukan sebuah pola bit yang disimpan dalam variabel dengan tipe
FIELD2N.Fungsi induk dipanggil sebanyak yang dibutuhkan.
*/

```

```

void random_field( value)
FIELD2N *value;
{
    INDEX i;

    SUMLOOP(i)
    {
        induk( &random_seed);
        value->e[i] = random_seed;
    }
    value->e[0] &= UPRMASK;
}

/* Pembentukan kurva elips untuk persamaan :
    $y^2 + xy = x^3 + a_6$  untuk  $a_2=0$ .
*/

void rand_curve_form_0(curv)
CURVE *curv;
{
    curv->form = 0;
    random_field( &curv->a6);
    null( &curv->a2);
}

/* Pembentukan kurva elips untuk persamaan :
    $y^2 + xy = x^3 + a_2x^2 + a_6$ 
*/

void rand_curve_form_1(curv)
CURVE *curv;
{
    curv->form = 1;
    random_field( &curv->a6);
    random_field( &curv->a2);
}

/* Pembangkitan titik pada kurva elips secara acak.
   Masukan terdiri dari variabel kurva dan variabel
   untuk menyimpan hasil.
*/

void rand_point( point, curve)
POINT *point;
CURVE *curve;
{
    FIELD2N    rf;

    random_field( &rf);
    poly_embed( &rf, curve, NUMWORD, rf.e[NUMWORD]&1, point);
}

/* Pertukaran kunci Diffie-Helman, yang digunakan untuk menghitung
   kunci publik pengirim. Variabel input yang digunakan titik basis
   dan kurva, dengan keluaran titik kunci publik, yaitu

```

```

    My_public = my_private*Base_point yang akan dikirim ke user lain.
*/

void DH_gen_send_key( Base_point, E, my_private, My_public)
POINT *Base_point, *My_public;
CURVE *E;
FIELD2N *my_private;
{
    poly_elptic_mul( my_private, Base_point, My_public, E);
}

/* Menghitung kunci rahasia bersama yang hasilnya harus sama antara
user pengirim dengan penerima. Variabel yang diperlukan titik
basis, kurva, kunci rahasia dan kunci publik yang diterima dari user
lain. Keluaran berupa kunci rahasia bersama, yaitu  $x = kP$ 
*/

void DH_key_share(Base_point, E, their_public, my_private,
shared_secret)
POINT *Base_point, *their_public;
CURVE *E;
FIELD2N *my_private, *shared_secret;
{
    POINT temp;

    poly_elptic_mul( my_private, their_public, &temp, E);
    copy (&temp.x, shared_secret);
}

/* Pengiriman pesan (plaintext) ke user lain menggunakan protokol
ElGamal. Enkripsi pesan menjadi ciphertext dan mengirimkannya
berserta titik acak ke user lain
*/

void send_elgamal(
    Base_point, Base_curve,
    Their_public, raw_data,
    Hidden_data, Random_point)
FIELD2N *raw_data;
POINT *Base_point, *Their_public, *Hidden_data, *Random_point;
CURVE *Base_curve;
{
    FIELD2N random_value;
    POINT hidden_point, raw_point;

/* buat titik acak pada kurva */

    random_field (&random_value);
    poly_elptic_mul (&random_value, Base_point, Random_point,
                    Base_curve);

/* memasukan baris data pada kurva */

    poly_embed( raw_data, Base_curve, 0, 0, &raw_point);

/* enkripsi menggunakan kunci publik */

```

```

        poly_elptic_mul( &random_value, Their_public, &hidden_point,
                        Base_curve);
        poly_esum( &hidden_point, &raw_point, Hidden_data, Base_curve);
    }

/* Menerima data (chiper text) dari user lain menggunakan protokol
   ElGamal. Deskripsi data menjadi plan text */

void receive_elgamal(
    Base_point, Base_curve,
    my_private, Hidden_data, Random_point,
    raw_data)
FIELD2N *my_private, *raw_data;
POINT *Base_point, *Hidden_data, *Random_point;
CURVE *Base_curve;
{
    POINT hidden_point, raw_point;

/* deskripsi chiper text menggunakan kunci rahasia dan titik acak */

    poly_elptic_mul( my_private, Random_point, &hidden_point,
                    Base_curve);
    poly_esub( Hidden_data, &hidden_point, &raw_point, Base_curve);
    copy(&raw_point.x, raw_data);
}

main()

{
    struct _timeb t1,t2,t3,t4,t5,t6;
    FIELD2N    privatel, private2, key1,
    key2, send_data, get_data, prime;
    CURVE rnd_crv, Public_Curve;
        POINT Base, P1, P2, Hidden_data, Random_point,
    Base_Point, Their_public;
    INDEX error;
    long mt1,mt2,mt3,mt4,mt5,mt6,i,count;
    short ch;

do {
    puts("-----");
    puts("[1]. Implementasi DH-ECC");
    puts("[2]. Implementasi ELGAMAL-ECC");
    puts("[3]. Daftar 10 Prima Polinom Pertama");
    puts("[0]. Keluar");
    puts("-----");
    printf("> ");scanf("%d",&ch);

    if (ch==3) {
        random_seed = 0x932b15fe;
        count=0;
        prime.e[0]=0x00000001;
        for (i=1;i<MAXLONG;i++)
            prime.e[i]=0;
        printf("Daftar 10 Prima Polinom Pertama Pada %d bit\n",NUMBITS);
    }
}

```

```

puts("-----");
do
{
    prime.e[MAXLONG-1]++;
    if (irreducible(&prime)){
        print_field("", &prime);
        count++;}
} while (count<10);

}
else

if (ch==1) {
    _ftime(&t1);
    mt1 = t1.time*1000 + t1.millitm;
    random_seed = 0x932b15fe;

if (!irreducible(&poly_prime))
{ printf("poly_prime bukan prima polinom, proses tidak dapat
    dilanjutkan\n");
    return(0);}
    printf("Implementasi Pertukaran Kunci (Diffie-Hellman Elliptic
        Curves Cryptosystem)\n");
printf("-----\n");
printf("Jumlah Bit : %d bit\n", NUMBITS);
print_field("Prima Polinom : ", &poly_prime);
printf("-----\n\n");

if (error = init_poly_math())
{
    printf("Matriks S tidak dapat diinisialisasikan, baris =
        %d\n", error);
    return(-1);
}

printf("-----\n");
printf("Pembentukan Kurva dan Titik Basis\n");
printf("-----\n\n");

/*bentuk 0 kurva*/
rand_curve_form_0(&rnd_crv);
print_curve("E : y^2 + xy = x^3 + a_6", &rnd_crv);

/*bentuk 1 kurva
rand_curve_form_1(&rnd_crv);
print_curve("E : y^2 + xy = x^3 + a_2*x^2 + a_6", &rnd_crv);*/

rand_point(&Base, &rnd_crv);
print_point("Titik Basis :", &Base);

printf("\n-----\n");
printf("Pembentukan kunci rahasia di setiap user\n");
printf("-----\n\n");

random_field(&privatel);

```

```

print_field("Kunci Rahasia Tera : ", &privatel);
random_field(&private2);
print_field("Kunci Rahasia Jana : ", &private2);

printf("\n-----\n");
printf("Pembentukan kunci publik di setiap user\n");
printf("-----\n\n");

_ftime(&t3);
mt3 = t3.time*1000 + t3.millitm;
DH_gen_send_key( &Base, &rnd_crv, &privatel, &P1);
print_point("Kunci publik Tera : ", &P1);
printf("\n");
DH_gen_send_key( &Base, &rnd_crv, &private2, &P2);
print_point("Kunci publik Jana : ", &P2);

printf("\n-----\n");
printf("Hasil akhir, user memiliki kunci rahasia yang sama\n");
printf("-----\n\n");

DH_key_share( &Base, &rnd_crv, &P2, &privatel, &key1);
print_field("Kunci rahasia bersama Tera : ", &key1);
DH_key_share( &Base, &rnd_crv, &P1, &private2, &key2);
print_field("Kunci rahasia bersama Jana : ", &key2);
_ftime(&t4);
mt4 = t4.time*1000 + t4.millitm;

_ftime(&t2);
mt2 = t2.time*1000 + t2.millitm;
printf("\n-----\n");
printf("Waktu proses      : %5.4f detik\n", (float)((mt2-
mt1)/1000.0));
printf("Waktu 4 perkalian : %5.4f detik\n", (float)((mt4-
mt3)/1000.0));
printf("-----\n\n");
}

else
if (ch==2) {
_ftime(&t1);
mt1 = t1.time*1000 + t1.millitm;
random_seed = 0x932b15fe;

if (!irreducible(&poly_prime))
{ printf("Prima polinom bukan bilangan prima, proses tidak dapat
dilanjutkan\n");
return(0);}

printf("Skenario : Pengiriman Pesan (bilangan integer yang besar)
dari Tera ke Jana\n");
printf("-----\n");
printf("Jumlah Bit      : %d bit\n", NUMBITS);
print_field("Bilangan Prima Polinom : ", &poly_prime);
printf("-----\n\n");

if (error = init_poly_math())
{

```

```

        printf("Matriks S tidak dapat diinisialisasikan, baris =
                %d\n", error);
        return(-1);
    }

    printf("-----\n");
    printf("Pembentukan Kurva dan Titik Basis\n");
    printf("-----\n\n");

    rand_curve_form_0(&Public_Curve);
    print_curve("E :  $y^2 + xy = x^3 + a_6$ ", &Public_Curve);

    /*rand_curve_form_1(&Public_Curve);
    print_curve("E :  $y^2 + xy = x^3 + a_2*x^2 + a_6$ ",
                &Public_Curve);*/

    rand_point(&Base_Point, &Public_Curve);
    print_point("Titik Basis :", &Base_Point);

    printf("\n-----\n");
    printf("Pembentukan Kunci Rahasia dan Kunci Public di User
            Jana\n");
    printf("-----\n\n");

    random_field(&private2);
    print_field("Kunci Rahasia Jana :", &private2);

    poly_elptic_mul( &private2, &Base_Point, &Their_public,
                    &Public_Curve);
    print_point("Kunci Public  Jana :", &Their_public);

    printf("\n-----\n");
    printf("Pembentukan Pesan (bilangan integer yang besar) Secara
            Acak di user Tera\n");
    printf("-----\n\n");
    random_field( &send_data);
    print_field("Pesan yang Dikirim : ", &send_data);

    printf("\n-----\n");
    printf("Proses Enkripsi Pesan (bentuk titik dalam kurva) dan
            Mengirimkannya ke Jana\n");
    printf("-----\n\n");
    _ftime(&t3);
    mt3 = t3.time*1000 + t3.millitm;
    send_elgamal( &Base_Point, &Public_Curve, &Their_public,
                &send_data, &Hidden_data, &Random_point);
    _ftime(&t4);
    mt4 = t4.time*1000 + t4.millitm;
    print_point("Pesan Terenkripsi : ", &Hidden_data);
    print_point("Titik Acak          : ", &Random_point);

    printf("\n-----\n");
    printf("Proses Deskripsi Pesan di User Jana\n");
    printf("-----\n\n");
    _ftime(&t5);

```

```

mt5 = t5.time*1000 + t5.millitm;
receive_elgamal( &Base_Point, &Public_Curve, &private2,
                &Hidden_data, &Random_point, &get_data);
_ftime(&t6);
mt6 = t6.time*1000 + t6.millitm;
print_field("Pesan yang Dikirim : ", &send_data);
print_field("Pesan yang Diterima: ", &get_data);
printf("\n");
_ftime(&t2);
mt2 = t2.time*1000 + t2.millitm;
printf("-----\n");
printf("Waktu proses      : %5.4f detik\n", (float)((mt2-
mt1)/1000.0));
printf("Waktu enkripsi   : %5.4f detik\n", (float)((mt4-
mt3)/1000.0));
printf("Waktu deskripsi  : %5.4f detik\n", (float)((mt6-
mt5)/1000.0));
printf("-----\n\n");
}

} while (ch!=0);
}

```