ELLIPTIC CURVE CRYPTOGRAPHY DAN APLIKASINYA PADA *MOBILE DEVICE*

Hermanto Ong – NIM : 13503069

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if13069@students.if.itb.ac.id

Abstrak

Perkembangan yang pesat dalam penggunaan *mobile device* dan *wireless device* menuntut suatu generasi baru dari skema *Public Key Cryptography* (*PKC*). Generasi baru dari skema *PKC* ini harus dapat mengakomodasi keterbatasan dalam hal kekuatan (*power*) dan *bandwidth*, serta pada saat yang bersamaan menyediakan suatu tingkat kemanan yang cukup baik untuk peralatan-peralatan tersebut. Makalah ini membahas penggunaan *Elliptic Curve Cryptography* (*ECC*) pada lingkungan yang memiliki keterbatasan-keterbatasan tersebut dan mendiskusikan dasar-dasar keamanannya, mengeksplorasi performansinya, serta membahas aplikasi *ECC* yang sekarang ini banyak digunakan dalam kehidupan sehari-hari.

Faktor efisiensi dan keamanan menjadikan *ECC* sebagai suatu alternatif solusi yang sangat baik bagi sistem kripto konvensional, seperti *Rivest Shamir Adleman* (*RSA*) dan *Digital Signature Algorithm* (*DSA*), bukan hanya dalam peralatan-peralatan yang memiliki keterbatasan-keterbatasan, tetapi juga pada komputer dengan spesifikasi yang tinggi. Sebagai perbandingan, sebuah kunci *ECC* 160 bit memiliki tingkat keamanan yang sama dengan sebuah kunci *RSA* 1024 bit. Itulah sebabnya *ECC* sangat cocok diterapkan pada peralatan-peralatan yang memiliki banyak keterbatasan seperti *mobile device* dan *wireless device*.

***Kata kunci**: elliptic curve cryptography*, *public key cryptography*, *mobile device*, *wireless device*, *power*, *bandwidth*, keamanan, kunci.

1. Pendahuluan

Pada tahun 1976, Whitfield Diffie dan Martin Hellman memperkenalkan konsep *public key cryptography* (*PKC*). Sejak saat itu, banyak sekali implementasi dari *PKC* telah diajukan, dan banyak dari aplikasi kriptografi ini mendasarkan keamanannya pada kekuatan permasalahan matematika yang sulit untuk dipecahkan, yaitu *integer factorization problem* (*IFP*) dan *finite field discrete logarithm problem* (*DLP*). Selama bertahun-tahun, algoritma dengan waktu yang subeksponensial dikembangkan untuk memecahkan permasalahan ini. Sebagai hasilnya, ukuran kunci berkembang menjadi lebih dari 1000 bit untuk mencapai suatu tingkat keamanan yang dapat diterima. Dalam lingkungan yang memiliki banyak keterbatasan, di mana kekuatan komputasi, penyimpanan, dan *bandwidth* sangat terbatas, melakukan operasi terhadap ribuan bit menjadi suatu pendekatan yang tidak praktis dalam rangka menyediakan tingkat keamanan yang cukup. Hal ini sangat jelas dalam *hand-held device* seperti telepon genggam, *pager*, dan *PDA* yang memiliki kekuatan pemrosesan dan kekuatan baterai yang sangat terbatas.

Diajukan oleh Neal Koblitz dan Victor Miller pada tahun 1985, *elliptic curve cryptography* (*ECC*) memiliki karakteristik khusus yang sampai sekarang ini diakui sebagai algoritma terbaik yang pernah diketahui untuk memecahkan permasalahan tersebut dalam waktu yang eksponensial. Keamanannya berasal dari *elliptic curve logarithm*, yang merupakan *DLP* dalam suatu kelompok yang didefinisikan oleh titik-titik dalam sebuah kurva elips terhadap *field* yang terbatas. Hal ini menghasilkan penurunan ukuran kunci yang dramatis, yang dibutuhkan untuk mencapai tingkat

keamanan yang sama dengan yang ditawarkan oleh skema *PKC* konvensional.

Makalah ini bertujuan untuk membahas dua buah aspek dari *ECC*, yaitu keamanan dan efisiensinya, untuk memberikan latar belakang penjelasan mengapa *ECC* sangat cocok diterapkan untuk lingkungan-lingkungan yang memiliki keterbatasan. Makalah ini akan dimulai dengan memperkenalkan tiga buah permasalahan matematika dan berbagai algoritma untuk menyelesaikan permasalahan tersebut. Suatu gambaran mengenai metode-metode implementasi dan pertimbangan-pertimbangan akan diberikan, diikuti dengan perbandingan performansi *ECC* dengan aplikasi-aplikasi *PKC* lainnya. Pada bagian akhir, akan dibahas suatu hasil peninjauan terhadap aplikasi-aplikasi *ECC* dalam berbagai *mobile device* sekarang ini.

1.1 Kebutuhan Terhadap Kriptografi Kunci Publik (*Public Key Cryptography*)

Kriptografi kunci privat (*private key cryptography*) banyak digunakan untuk enkripsi data karena kecepatannya. Kriptografi kunci privat yang paling umum digunakan saat ini adalah *Data Encryption Standard* (*DES*). *DES* memiliki kecepatan enkripsi yang sangat tinggi dan ini merupakan suatu kualitas yang sangat menarik dalam hal efisiensi; meskipun demikian, *DES* memiliki kekurangan tertentu yang membuatnya tidak cocok untuk digunakan dalam lingkungan *m-commerce*.

I. Masalah Manajemen Kunci
   Seorang pengguna *wireless* harus dapat melakukan transaksi bisnis dengan tidak hanya satu pihak saja, melainkan dengan banyak pihak yang berbeda-beda. Oleh sebab itu, komunikasi dalam suatu jaringan publik tidak dibatasi satu ke satu, melainkan dapat melayani sejumlah besar pengguna. Untuk sebuah jaringan dengan n orang pengguna, $n(n-1)/2$ buah kunci privat perlu dibangkitkan. Ketika nilai n besar, jumlah kunci menjadi sangat besar sehingga tidak mungkin lagi di-*manage*.

II. Masalah Distribusi Kunci
   Dengan jumlah kunci yang sangat besar yang perlu dibangkitkan dalam sebuah jaringan, pekerjaan untuk membangkitkan kunci dan menemukan saluran yang aman untuk mendistribusikannya menjadi suatu kesusahan tersendiri.

III. Tidak Ada Tanda Tangan Digital yang Memungkinkan
   Sebuah tanda tangan digital adalah suatu analogi elektronik dari sebuah tanda tangan (biasa). Jika Alice mengirimkan sebuah pesan terenkripsi kepada Bob, Bob harus dapat memverifikasi bahwa pesan yang diterima benar-benar berasal dari Alice. Hal ini dapat dilakukan dengan menggunakan tanda tangan Alice; meskipun demikian, kriptografi kunci privat tidak memberikan fitur semacam ini.

Berkebalikan dengan kriptografi kunci privat, kriptografi kunci publik menggunakan dua buah kunci. Masing-masing pengguna dalam sebuah jaringan mengumumkan sebuah kunci enkripsi publik yang dapat digunakan oleh setiap orang untuk mengirimkan pesan, sementara kunci privat dirahasiakan untuk digunakan pada saat dekripsi. Dalam sebuah jaringan yang terdiri dari n orang pengguna, hanya dibutuhkan n buah kunci publik dan n buah kunci privat. Hal ini mereduksi jumlah kunci yang dibutuhkan dari $O(n^2)$ menjadi $O(n)$. Lebih jauh, kriptografi kunci publik memungkinkan penggunaan tanda tangan digital, yang menyediakan aspek nir-penyangkalan. Meskipun demikian, kriptografi kunci publik juga memiliki kekurangan. Dibandingkan dengan kriptografi kunci privat, kriptografi kunci publik berjalan dengan lebih lambat. *RSA* membutuhkan setidaknya kunci 1024 bit, sedangkan *DES* hanya membutuhkan 64 bit. Dalam kenyataannya, kriptografi kunci publik dan kriptografi kunci privat bekerja dengan sangat baik bersama-sama [2]. Kriptografi kunci publik ideal untuk distribusi dan manajemen kunci, menjamin integritas data, menyediakan otentikasi dan nir-penyangkalan, sementara kriptografi kunci privat ideal untuk menjamin

kerahasiaan, seperti untuk mengenkripsi data dan saluran komunikasi. Ini adalah empat buah objektif utama dalam berbagai aplikasi kriptografi.

1.2 Pilihan Sistem Kriptografi Kunci Publik (*Public Key Cryptosystem*)

Ketika memilih sistem kriptografi kunci publik yang akan digunakan dalam suatu lingkungan *mobile*, setiap orang harus memperhatikan keterbatasan-keterbatasan pada *bandwidth*, memori, dan jangka hidup baterai. Dalam lingkungan dengan berbagai keterbatasan seperti telepon genggam, *wireless pager*, atau *PDA*, sumber daya-sumber daya ini sangat terbatas. Oleh sebab itu, skema kunci publik yang cocok adalah skema yang efisien dalam hal biaya komputasi dan ukuran kunci.

Sampai sekarang ini, *ECC* memiliki *strength-per-bit* tertinggi dibandingkan dengan sistem kriptografi kunci publik lainnya. Ukuran kunci yang kecil akan menghemat *bandwidth*, memor, dan kekuatan pemrosesan. Hal ini membuat *ECC* menjadi pilihan yang jelas dalam situasi seperti ini. Meskipun demikian, terdapat aspek-aspek lain yang harus dipertimbangkan. Dalam bagian berikut ini, kita akan membahas permasalahan matematika yang berbeda-beda, yang mendasari sebagian besar sistem kriptografi kunci publik yang digunakan saat ini. Kita juga akan membahas beberapa algoritma yang paling efisien untuk menyelesaikan permasalahan tersebut. Hal ini akan memberikan suatu pemahaman yang lebih baik mengenai keamanan yang mendasari berbagai jenis sistem kriptografi kunci publik.

2. Keamanan Sistem Kriptografi Kunci Publik

Seperti telah disebutkan sebelumnya, banyak sistem kriptografi kunci publik yang mendasarkan keamanannya pada kesulitan menyelesaikan suatu masalah matematika. Sekarang ini, terdapat tiga buah masalah yang dipercaya aman dan praktis setelah dilakukan penelitian bertahun-tahun. Masalah tersebut adalah *integer factorization problem*, *finite field discrete logarithm problem*, dan *elliptic curve discrete logarithm problem*. Ketika masalah-masalah ini telah dibuktikan tidak dapat diselesaikan/dipecahkan, dapat diprediksi bahwa tidak ada seorang pun yang dapat menemukan sebuah algoritma yang efisien untuk menyelesaikan permasalahan ini dalam waktu dekat.

Keamanan dari sebuah sistem kriptografi bergantung pada seberapa sulit untuk menyelesaikan permasalahan matematika yang mendasarinya. Tingkat kesulitan dari sebuah masalah ditentukan oleh waktu eksekusi asimptotik dari algoritma yang dapat menyelesaikan masalah tersebut.

Definisi: Sebuah algoritma dengan ukuran input n adalah **sub-eksponensial** jika terdapat konstanta $c > 0$ dan $\alpha \in [0,1]$ sedemikian sehingga waktu eksekusi dari algoritma adalah dalam

$$L_n[\alpha, c] = O(\exp((c+o(1))(\ln x)^{\alpha}(\ln \ln x)^{1-\alpha})) \text{ [3].}$$

Jika $\alpha = 0$, maka algoritma berjalan dalam waktu yang polinomial. Jika $\alpha = 1$, maka algoritmanya eksponensial penuh.

2.1 *Integer Factorization Problem* (*IFP*)

2.1.1 Definisi Masalah

Integer factorization problem secara umum didefinisikan sebagai berikut:

Diberikan sebuah integer positif n, tuliskan $n = p_1^{e1} p_2^{e2} p_3^{e3} ... p_k^{ek}$ di mana p_i adalah jarak *pairwise* prima (*pairwise distinct primes*) dan setiap $e_i \geq 1$ [4].

Umumnya, dalam aplikasi kriptografi, hanya dua buah faktor yang digunakan untuk modulus n. Sejumlah besar faktor untuk n tampaknya tidak menawarkan keamanan tambahan dalam *IFP*.

Sistem kriptografi kunci publik yang dikenal dengan sangat baik, yang mendasarkan keamanannya pada kesulitan dari *IFP* adalah *RSA*. Diberi nama sesuai dengan nama penemunya: Ron Rivest, Adi Shamir, dan Len Adleman, yang mengembangkannya di MIT pada tahun 1978, *RSA* adalah implementasi praktis pertama dari kriptografi kunci publik sejak pengenalan konsepnya. Contoh lainnya adalah sistem kriptografi Rabin-Williams. Sistem kriptografi ini mirip dengan *RSA*, tetapi

menggunakan sebuah eksponen publik yang genap [4].

2.1.2 Algoritma Pemfaktoran

Ide dasar di balik pemfaktoran mencakup penemuan dua buah bilangan x dan y sedemikian sehingga (1) $x^2 \equiv y^2 \pmod{n}$ dan (2) $x \neq \pm y \pmod{n}$, di mana n adalah bilangan yang akan difaktorkan. Karena (1) <=> $(x - y)(x + y) \equiv 0 \pmod{n}$, dan (2) dinyatakan bahwa n tidak membagi $(x - y)$ dan $(x + y)$, maka $\gcd(x - y, n)$ dan $\gcd(x + y, n)$ harus merupakan faktor nontrivial dari n. Dua algoritma pemfaktoran yang paling banyak digunakan sekarang ini adalah *quadratic sieve* dan *number field sieve*. Keduanya berdasarkan pada ide untuk menemukan sebuah basis faktor prima untuk menghasilkan sebuah sistem persamaan linear, yang solusinya akan menuju ke persamaan (1) sedemikian sehingga (2) tetap berlaku.

Quadratic sieve (*QS*) adalah sebuah algoritma pemfaktoran *general-purpose* karena waktu eksekusinya bergantung semata-mata kepada ukuran n. Sebuah varian pengembangan dari *QS*, *multiple polynomial QS*, mencapai waktu eksekusi sebesar

$$(I) \qquad L_n[1/2,1].$$

Multiple polynomial QS memberi peluang pemfaktoran yang lebih baik, cocok digunakan untuk pemrosesan paralel, dan metode yang banyak dipilih dalam kenyataannya (prakteknya) [4].

Contoh lain dari algoritma pemfaktoran *general-purpose* adalah *generalized number field sieve* (*NFS*). Awalnya dikembangkan sebagai suatu algoritma *special-purpose* yang memfaktorkan integer dalam bentuk $n = r^e - s$ untuk r dan $|s|$ yang kecil, *generalized NFS* selanjutnya dikembangkan agar dapat bekerja juga untuk integer yang umum [4]. *Generalized NFS* dipertimbangkan sebagai algoritma untuk memfaktorkan bilangan umum dengan setidaknya 120 digit desimal yang paling cepat. *Generalized NFS* mencapai waktu eksekusi $L_n[1/3,c]$, untuk konstanta c, mengurangi eksponen dalam (I) [5]. Nilai dari c bergantung pada versi *NFS* yang digunakan. Dalam kasus *NFS* spesial (khusus), $c = (32/9)^{1/3} \approx 1.526$, sementara *generalized NFS* memiliki $c = (64/9)^{1/3} \approx 1.923$. Awalnya diperkirakan lebih lambat daripada *QS* untuk memfaktorkan bilangan dengan digit kurang dari 150, eksperimen akhir-akhir ini telah menunjukkan bahwa *general NFS* pada dasarnya lebih cepat daripada *QS*, bahkan untuk bilangan dalam rentang 115 digit [4]. Karena hal ini, *generalized NFS* dipertimbangkan sebagai algoritma yang paling *powerful* (kuat) dari semua algoritma pemfaktoran *general purpose*.

2.1.3 Tantangan *RSA*

Dalam beberapa dekade terakhir, pengembangan yang dramatis dalam *IFP* telah dibuat. Tabel 1 di bawah adalah sebuah tabel yang menunjukkan kemajuan dari faktorisasi bilangan *RSA*. Kita dapat melihat bahwa pemfaktoran *RSA*-130 menggunakan suatu algoritma baru dan berkembang, yang mencapai faktorisasi dengan hanya 20% dari usaha komputasi yang digunakan untuk memfaktorkan *RSA*-129 yang lebih kecil.

Sebuah ide terbaru untuk menyerang *IFP* adalah suatu mesin khusus yang disebut *TWINKLE* (*The Weizmann Institute Key Locating Engine*). Penemunya Shamir, yang merupakan salah satu penemu dari *RSA*, mengajukannya pada tahun 1999. Alat *sieving* ini dapat mempercepat proses *sieving* dalam algoritma *NFS* dengan besar dua sampai tiga derajat [7]. Meskipun demikian, matriksnya masih harus diselesaikan pada sebuah komputer konvensional. Bilangan *RSA* 140 digit yang telah dipecahkan baru-baru ini memerlukan 200 buah komputer konvensional yang berjalan secara paralel selama 4 minggu untuk menyelesaikan *sieving*. Berkebalikan dengan ini, hanya dibutuhkan 6 hari pada 7 buah mesin *TWINKLE*. Karena solusi matriks masih harus dilakukan pada sebuah komputer konvensional, ini akan menambah 4 hari lebih lama pada faktorisasi, mengakibatkan jumlah hari keseluruhan yang dibutuhkan menjadi 10 [7].

Dengan kejadian ini, seseorang mungkin bertanya, “Apakah *IFP* tidak lagi menjadi suatu permasalahan yang sulit?”. Perjanjian umum adalah bahwa *IFP* masih merupakan

Number of decimal digits	Approximate number of bits	Data achieved	MIPS-years	Algorithm
100	332	April 1991	7	Quadratic sieve
110	365	April 1992	75	Quadratic sieve
120	398	June 1993	830	Quadratic sieve
129	428	April 1994	5000	Quadratic sieve
130	431	April 1996	1000	Generalized number field sieve
140	465	February 1999	2000	Generalized number field sieve
155	512	August 1999	8000	Generalized number field sieve

Tabel 1. Kemajuan Faktorisasi Bilangan *RSA*

suatu masalah yang sulit untuk dipecahkan. Meskipun demikian, untuk mencapai suatu tingkat keamanan yang memadai, ukuran dari kunci harus ditingkatkan untuk mengikuti perkembangan yang terus-menerus dalam *IFP*. Keamanan ini membutuhkan suatu harga yang mahal – harga dari kebutuhan penyimpanan yang sangat besar, *bandwidth* yang besar, dan kemampuan komputasi yang kuat, khususnya dalam peralatan yang memiliki keterbatasan.

2.2 *Discrete Logarithm Problem* (*DLP*)

Tidak seperti *IFP*, di mana yang menjadi masalah adalah panjang dari modulus *n* yang harus difaktorkan, ukuran input untuk *DLP* adalah jumlah titik *N* dalam kelompok *G* yang sedang dikerjakan. Dalam kasus multiplikasi dari kelompok $G = Z^*_p$, di mana *p* adalah suatu bilangan prima yang besar, *N* sama dengan ukuran *finite field* yang mendasarinya.

2.2.1 Definisi Masalah

Misalkan $G = Z^*_p$ adalah suatu kelompok multiplikasi dengan *order* $p - 1$, di mana operasi kelompok adalah multiplikasi modulo *p*. *Discrete logarithm problem* didefinisikan sebagai berikut:

Diberikan sebuah bilangan prima *p*, sebuah *generator* α dari Z^*_p, dan sebuah elemen $\beta \in Z^*_p$, temukan integer unik *x*, $0 \le x \le p-2$, sedemikian sehingga $\alpha^x = \beta \pmod p$ [4].

Aplikasi kriptografi yang mendasarkan keamanannya pada kesulitan dari *DLP* mencakup skema persetujuan kunci Diffie-Hellman, skema enkripsi ElGamal, dan *digital signature algorithm* (*DSA*).

2.2.2 Algoritma *Discrete Logarithm*

Algoritma terbaik yang diketahui untuk menghitung *DLP* adalah metode *index-calculus*. Metode ini merupakan algoritma probabilistik yang diterapkan hanya untuk *finite field*. Contoh *finite field* yang umumnya digunakan dalam aplikasi praktis adalah GF(*p*) dan $GF(2^m)$. Metode *index-calculus* adalah satu-satunya algoritma yang diketahui dapat menyelesaikan *DLP* dalam waktu sub-eksponensial, membuatnya menjadi juara di antara semua algoritma *DL*. Metode *index-calculus* bekerja sangat mirip dengan *NFS* dan *QS* dalam *IFP*. Metode ini juga membutuhkan pemilihan sebuah basis faktor *S* dari bilangan prima yang kecil sedemikian sehingga sebuah porsi elemen dari *G* yang signifikan dapat diekspresikan secara efisien sebagai perkalian dari elemen dalam *S*. Metode ini kemudian membangun sebuah basis data dari relasi-relasi, yang digunakan setiap kali logaritma dari sekelompok elemen dibutuhkan. Seperti halnya algoritma *IFP*, algoritma *index-calculus* juga dapat dengan mudah diparalelkan.

DLP dalam sebuah *field* prima dipertimbangkan lebih sulit daripada *DLP* dalam *field* dari karakteristik dua. Catatan

terbaru untuk menghitung *discrete logarithm* dalam GF(p) adalah sebuah bilangan prima 120 digit p [9]. Hal ini disempurnakan oleh A. Joux et R. Lercier pada tahun 2001, menggunakan suatu variasi dari algoritma *index-calculus* yang disebut *number field sieve*. Algoritma ini memiliki waktu eksekusi yang diharapkan sebesar $L_p[1/3,1.923]$ [4]. Berlawanan dengan hal ini, catatan terbaru untuk menghitung logaritma dalam GF(2^m) adalah GF(2^{607}). Dilengkapi pada tahun 2002 oleh E. Thome, logaritma dalam GF(2^m) dicapai oleh varian lain dari metode *index-calculus*, yang disebut algoritma Coppersmith [9]. Algoritma ini memiliki waktu eksekusi sebesar $L_2^m[1/3,c]$, untuk $c < 1.587$ [4].

Terdapat algoritma yang dapat menyelesaikan *DLP* untuk kelompok yang berubah-ubah, meskipun demikian algoritma-algoritma tersebut berjalan dalam waktu yang eksponensial penuh. Contoh mencakup *Pollard ρ-method* dan *baby-step giant-step* (*BSGS*). Kelebihan *BSGS* adalah sangat intensif dalam hal memori ketika *order* kelompok menjadi besar. Di pihak lain, *Pollard ρ-method* membutuhkan sedikit memori dan dapat dengan mudah diparalelkan. Algoritma lainnya yang bekerja untuk kelompok yang berubah-ubah, tetapi hanya efisien untuk *order* yang disusun oleh bilangan prima yang kecil, adalah algoritma Pohlig-Hellman. Karena itu, *order* kelompok diperiksa untuk memastikan bahwa serangan Pohlig-Hellman tidak dapat diterapkan. Untuk kelompok yang berubah-ubah, algoritma ini juga berjalan dalam waktu yang eksponensial penuh.

2.2.3 *QS* dan *NFS* dalam *DLP* dan *IFP*

Baik algoritma pemfaktoran *QS* maupun *NFS* sangat mirip pendekatannya dengan metode *index-calculus* dari *DLP*. "Kedua masalah tersebut sangat mirip, dan semua algoritma pemfaktoran modern dapat digunakan untuk menghitung *discrete logarithm* dalam kelompok multiplikasi dari suatu *finite field*" [10]. Dari sudut pandang ini, jumlah pekerjaan yang dilakukan dalam menghitung logaritma Z^*_p, di mana p memiliki k bit, dapat dipertimbangkan sama dengan jumlah pekerjaan yang dibutuhkan untuk memfaktorkan suatu k bit bilangan komposit n. Oleh sebab itu, apabila berbagai perkembangan dalam algoritma telah ditemukan untuk salah satu permasalahan tersebut, maka perkembangan tersebut juga dapat diterapkan untuk masalah lainnya [11].

Sejauh ini, kedua masalah matematika yang telah dibahas memiliki algoritma sub-eksponensial yang dapat menyelesaikannya. Pada bagian berikut ini, suatu jenis masalah yang berbeda, yang disebut *elliptic curve discrete logarithm problem*, akan dibahas. Sampai saat ini, algoritma terbaik yang menghitung *elliptic curve logarithm* berjalan dalam waktu yang eksponensial penuh.

2.3 *Elliptic Curve Discrete Logarithm Problem* (*ECDLP*)

2.3.1 Definisi Masalah

Misalkan E adalah suatu kurva elips terhadap beberapa *finite field* GF(p) dan G = $E(F_q)$ adalah sekelompok *cyclic additive*, di mana operasi kelompok adalah penjumlahan modulo p. *Elliptic curve discrete logarithm problem* didefinisikan sebagai berikut:

Diberikan $P \in G$ dan sebuah elemen $Q \in \langle P \rangle$, temukan integer m, sedemikian sehingga $Q = [m]P$ [12].

Pada tahun 1985, Neal Koblitz dan Victor Miller secara terpisah mengajukan konsep *elliptic curve cryptography* (*ECC*). *ECC* berdasarkan pada *DLP* dalam suatu kelompok yang didefinisikan oleh titik-titik pada sebuah kurva elips terhadap suatu *finite field*. Implementasi dari *ECC* mencakup *elliptic curve analogs of DSA* (*ECDSA*), ElGamal, dan Diffie-Hellman.

2.3.2 Algoritma *ECDLP*

Fitur paling menarik dari *ECC* adalah pada saat ini *ECC* merupakan algoritma tercepat yang diketahui, yang dapat menyelesaikan permasalahan tersebut dalam waktu yang eksponensial penuh. Meskipun fakta menunjukkan bahwa metode *index-calculus* dapat menghitung logaritma konvensional dalam waktu sub-eksponensial, metode *index-calculus* tidak dapat diterapkan pada kasus *discrete logarithm* terhadap kurva elips. Ini merupakan sebuah klaim yang dibuat oleh Miller dalam makalahnya pada tahun 1986, yang kemudian didukung oleh

studi teoretis dan eksperimen komputasional yang dilakukan oleh J. H. Silverman dan Suzuki dalam makalah mereka yang diumumkan pada tahun 1998 [3].

Sampai saat ini, algoritma *general-purpose* terbaik yang diketahui untuk menyelesaikan *ECDLP* adalah *Pollard ρ-method*. Metode ini dapat dipercepat dengan teknik khusus ketika berjalan dalam prosesor paralel menjadi $O(\sqrt{(\pi n)}/(2r))$, di mana n adalah jumlah penambahan kurva elips dan r adalah jumlah prosesor yang digunakan [11]. Ketika *order* dari kurva menjadi cukup besar, metode seperti *Pollard ρ* dan *BSGS* menjadi tidak mungkin [12]. Pada tahun 1997, V. Shoup menunjukkan bahwa waktu eksekusi dari berbagai algoritma yang dapat menyelesaikan *DLP* untuk kelompok yang berubah-ubah memerlukan $\Omega(\sqrt{p} \log p)$ langkah, di mana p adalah faktor prima terbesar dari N [3]. Jadi, untuk meningkatkan efisiensi algoritma *general purpose* secara signifikan dapat dikatakan merupakan suatu pekerjaan yang sia-sia.

2.3.3 Kurva Lemah (*Weak Curve*)

Terdapat jenis-jenis kurva elips tertentu di mana serangan yang berhasil dapat terjadi dalam waktu yang sub-eksponensial. Jika diidentifikasi, kurva-kurva ini dapat dengan mudah dites dan dihindari. Sejauh ini, beberapa kelas kurva telah diidentifikasi dan dilarang dalam semua spesifikasi standar untuk kriptografi kunci publik, seperti IEEE P1363, ANSI X9.62, dan ANSI X9.63 [11]. Kurva semacam itu disebut kurva supersingular dan kurva anomalous.

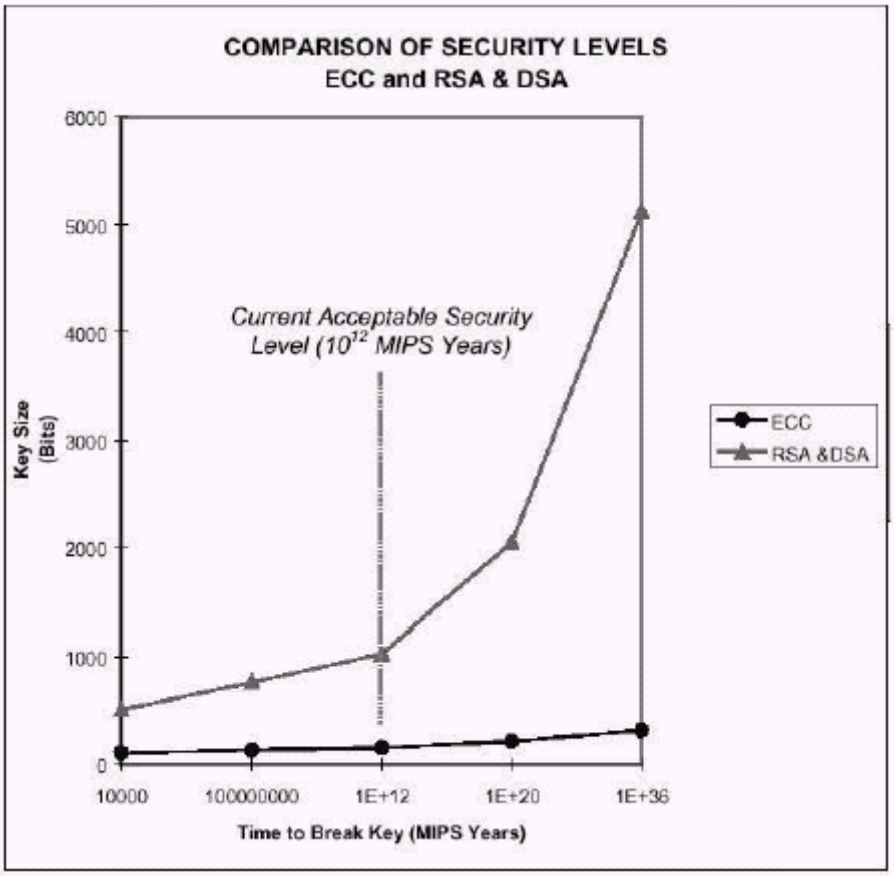
Kurva supersingular adalah sebuah kelas khusus dari kurva elips di mana *elliptic curve logarithm* dapat direduksi menjadi kasus *discrete logarithm* dalam suatu kelompok multiplikasi (*DLP* klasik). Ketika dikombinasikan dengan algoritma sub-eksponensial untuk menyelesaikan *DLP* klasik, dihasilkan sebuah waktu eksekusi probabilistik yang sub-eksponensial untuk menghitung *elliptic curve logarithm* pada kurva supersingular. Ini merupakan sebuah penemuan oleh Menezes, Okamoto, dan Vanstone (MOV) pada tahun 1991, di mana mereka menunjukkan bagaimana *ECDLP* dapat direduksi menjadi *DLP* klasik dalam suatu perluasan dari sebuah kelompok multiplikasi GF(p) [11].

Kelas kurva yang lain, kurva anomalous, memungkinkan suatu serangan yang bahkan lebih efisien ketika dapat dilakukan. Diajukan pada tahun 1998 oleh Satoh dan Araki, Semaev, serta pada tahun berikutnya oleh Smart, jenis kurva ini memungkinkan *ECDLP* untuk diselesaikan dalam waktu polinomial dengan mereduksinya menjadi *DLP* klasik dalam suatu kelompok *additive* GF(p) [3].

2.3.4 Keuntungan *ECC*

Seperti halnya tantangan *RSA*, tantangan Certicom *ECC* menawarkan hadiah uang untuk menemukan berbagai ukuran kunci dari *ECDLP*. Catatan terbaru dibuat pada November 2002, di mana sebuah kunci enkripsi 109 bit dipecahkan dengan 10000 komputer yang bekerja 24 jam per hari selama 549 hari [17]. Situs *web* tantangan Certicom *ECC* melaporkan bahwa memecahkan sebuah kunci 163 bit, yang merupakan standar yang diterapkan terhadap sebagian besar aplikasi *ECC* komersial yang digunakan oleh Certicom, seratus juta kali lebih sulit dibandingkan dengan memecahkan kunci 109 bit. Sebagai catatan, sebuah kunci *ECC* 160 bit kira-kira memiliki tingkat keamanan yang sama dengan sebuah kunci *RSA* 1024 bit.

Perbedaan yang paling penting di antara *ECC* dan sistem kriptografi konvensional lainnya adalah bahwa untuk sebuah kurva yang dipilih dengan baik, metode terbaik yang saat ini dikenal untuk menyelesaikan *ECDLP* adalah eksponensial penuh, sementara itu terdapat algoritma yang sub-eksponensial untuk sistem kriptografi konvensional. Perbedaan ini berkontribusi pada perbedaan dalam waktu eksekusinya yang sangat besar. Ini juga berarti bahwa kunci *ECC* memiliki jumlah bit yang jauh lebih sedikit dibandingkan dengan aplikasi berbasis *IFP* dan *DPL*. Perbedaan panjang kunci yang sangat kontras di antara *RSA*, *DSA*, dan *ECC* ditunjukkan dalam Gambar 1 di bawah. Sangat jelas bahwa kunci *ECC* membutuhkan usaha yang jauh lebih besar untuk memecahkannya dibandingkan dengan kunci *RSA* dan *DSA*. Oleh sebab itu, banyak orang percaya bahwa *ECDLP* secara



Gambar 1. Perbandingan Tingkat Keamanan *ECC* dan *RSA* & *DSA*

intrinsik lebih sulit daripada dua masalah lainnya. Ketika deduksi ini mungkin benar, kita tidak memiliki cara untuk membuktikannya. Kita tidak mengetahui jika suatu algoritma *elliptic curve DL* yang cepat dan efisien, yang berjalan dalam waktu sub-eksponensial akan ditemukan, misalkan, dalam sepuluh tahun yang akan datang, atau jika kelas kurva lemah lainnya akan diidentifikasi yang dapat membahayakan keamanan dari sistem kriptografi kurva elips. Akan tetapi, satu hal yang pasti bahwa setelah pembelajaran intensif selama bertahun-tahun, sekarang ini tidak terdapat cara yang lebih cepat untuk menyerang *ECDLP* selain daripada algoritma eksponensial penuh.

3. Implementasi Sistem Kriptografi Kurva Elips

Terdapat beberapa isu penting yang perlu diperhatikan sebelum mengimplementasikan sistem kriptografi kurva elips. Kita perlu menentukan apakah akan menggunakan suatu *field* dengan karakteristik genap atau ganjil, dan juga bagaimana merepresentasikan titik-titik pada kurva elips. Pilihan ini tidak hanya akan menentukan bagaimana kita mengimplementasikan *field* aritmatika pada kurva elips, tetapi juga akan mempengaruhi efisiensi dari komputasi. Dalam bagian 3.1, kita akan membahas dua jenis karakteristik *field*, termasuk representasi yang mendasarinya dan beberapa operasi *field* aritmatika yang berasosiasi dengan setiap jenis representasi basis. Bagian 3.2 membahas representasi yang berbeda-beda dari titik-titik pada kurva elips. Bagian 3.3 menunjukkan bagaimana untuk membangun *ECC* dan parameter-parameternya, dan bagian terakhir membahas kurva-kurva yang direkomendasikan oleh NIST serta petunjuk-petunjuk lainnya.

3.1 Karakteristik *Field* Genap dan Ganjil

Terdapat dua jenis karakteristik *field*, yaitu genap dan ganjil. *Field* prima GF(p), di mana p adalah sebuah bilangan prima yang besar, merupakan karakteristik ganjil. *Field* ini memiliki p elemen yang direpresentasikan oleh integer modulo p. *Field* aritmatika pada GF(p) diimplementasikan dalam hubungan aritmatika dari integer modulo p. *Field* GF(2^m) merupakan karakteristik genap, secara spesifik, adalah karakteristik 2. Terdapat 2^m elemen dalam *field* ini, dan elemen-elemen tersebut direpresentasikan sebagai vektor biner berdimensi-m di atas F_2, yaitu string bit dengan panjang m. *Field* penjumlahan dan pengurangan diimplementasikan sebagai *component-wise* XOR, sementara implementasi dari perkalian dan inversi (pembagian) bergantung pada pilihan basis.

Aritmatika dalam sebuah *field* prima adalah sederhana; hanya merupakan aritmatika dari integer modulo p. Untuk sebuah *field* biner, elemen-elemen *field* direpresentasikan relatif terhadap basis yang diberikan. Terdapat banyak pilihan untuk sebuah basis. Sebuah basis polinomial memiliki bentuk

$$\{1, t, t^1, ..., t^{m-1}\},$$

di mana t adalah sebuah akar dari suatu polinomial yang tidak dapat direduksi $p(t)$ di atas F_2. Sebuah polinomial yang tidak dapat direduksi adalah polinomial yang tidak dapat difaktorkan sebagai suatu perkalian dari polinomial dengan derajat lebih rendah modulo 2. Sebuah elemen dari GF(2^m) ɜ (a_0, a_1, ..., a_{m-1}), di mana $a_i \in \{0,1\}$, w.r.t sebuah basis polinomial direpresentasikan oleh polinomial

$$a_0 + a_1 t + a_2 t^2 ... + a_{m-1} t^{m-1} \bmod p(t),$$

di mana $p(t)$ adalah sebuah polinomial yang tidak dapat direduksi di atas F_2. *Field* aritmatika dilakukan sebagai aritmatika polinomial modulo $p(t)$ [18].

Sebuah basis normal memiliki bentuk

$$\{\beta, \beta^2, \beta^{2^2}, ..., \beta^{2^{(m-1)}}\},$$

di mana $\beta \in GF(2^m)$. Elemen-elemen dari GF(2^m) ɜ (a_0, a_1, ..., a_{m-1}) w.r.t sebuah basis normal dapat ditulis sebagai

$$a_0\beta + a_1\beta^2 + a_2\beta^{2^2} ... + a_{m-1}\beta^{2^{(m-1)}},$$

di mana $a_i \in \{0,1\}$. Penjumlahan dari elemen-elemen *field* dalam representasi basis normal adalah *bitwise* XOR dari elemen-elemen vektor. Kuadrat dapat dilakukan dengan sebuah rotasi dari elemen-elemen vektor. Ini merupakan operasi yang murah untuk dilakukan, sehingga biaya dari pengkuadratan sering diabaikan dalam menganalisis kompleksitas waktu eksekusi. Perkalian lebih rumit, tetapi dengan optimisasi, operasinya menjadi serangkaian m siklus pergeseran dari dua buah vektor yang akan dikalikan. Inversi (pembagian) adalah operasi yang paling kompleks dan mahal untuk dilakukan. Sebuah contoh dari suatu algoritma inversi adalah algoritma yang diajukan oleh Itoh, Teechai, dan Tsujii, yang dapat ditemukan dalam [24]. Di sini, pembagian adalah suatu algoritma rekursif yang membutuhkan $I(m) = \lfloor \log_2(m-1) \rfloor + \omega(m-1) - 1$ perkalian *field*, di mana $\omega(m-1)$ adalah jumlah bilangan 1 dalam representasi biner dari $m-1$ [1].

Terdapat beberapa alasan mengapa *field* dari karakteristik 2 lebih dipilih dibandingkan dengan *field* karakteristik ganjil. Pertama, elemen-elemen *field* dalam GF(2^m) direpresentasikan sebagai string bit dengan panjang m. Dalam hubungannya dengan perangkat keras, representasi string bit dari integer menyediakan kemudahan implementasi yang jauh lebih besar daripada representasi integer alami. Lebih jauh, dengan representasi basis normal dalam GF(2^m), pengkuadratan dapat dilakukan melalui suatu rotasi sederhana dari elemen-elemen vektor, yang berarti bahwa biaya pengkuadratan dapat diabaikan [1]. Faktor penting lainnya adalah basis normal memungkinkan perancangan perkalian serial bit yang efisien. Salah satu contohnya adalah yang diajukan oleh Massey dan Omura [19].

3.2 Representasi Titik

Pada bagian ini, dua jenis representasi titik akan dibahas – koordinat *affine* dan *projective*. Kita akan menggunakan formula dari penjumlahan titik dalam suatu *field* prima untuk mengilustrasikan perbedaan biaya dalam melakukan aritmatika titik menggunakan kedua representasi tersebut.

Koordinat *affine* (x, y) memenuhi persamaan *affine*

$$E: y^2 = x^3 + ax + b,$$

di mana $a, b \in \text{GF}(p)$.

Koordinat *projective* konvensional (x, y, z) memenuhi persamaan Weierstrass homogen

$$\text{E: } y^2z = x^3 + axz^2 + bz^3,$$

di mana $a, b \in F_p$. Ketika $z \neq 0$, titik *projective* (x, y, z) berkorespondensi dengan titik *affine* $(x/z, y/z)$. Koordinat *projective* digunakan ketika inversi *field* jauh lebih mahal daripada perkalian *field*. Dengan koordinat *projective*, kebutuhan untuk melakukan inversi digantikan dengan perkalian, sehingga penjumlahan *projective* dapat dilakukan melalui penggunaan perkalian *field*. Terdapat jenis lain dari representasi *projective* yang lebih efisien daripada representasi *projective* konvensional. Secara khusus, representasi *weighted projective* (atau representasi *Jacobian*) menghasilkan suatu implementasi yang lebih efisien dari operasi kelompok [12]. Koordinat *Jacobian* (x, y, z) berkorespondensi dengan koordinat *affine* $(x/z^2, y/z^3)$, dan memenuhi persamaan kurva *weighted projective*

$$E: y^2 = x^3 + axz^4 + bz^6.$$

Biaya penjumlahan titik menggunakan kedua representasi tersebut diringkas dalam Tabel 2 di bawah ini.

Cost of point addition, characteristic p > 3.

Operation	Coordinates	
	affine	projective
General addition	$1I + 3M$	$16M$
Doubling (arbitrary a)	$1I + 4M$	$10M$
Doubling ($a = -3$)	$1I + 4M$	$8M$

Tabel 2. Biaya Penjumlahan Titik Menggunakan Representasi *Affine* dan *Projective* (I = inversi, M = perkalian) [12]

3.3 Pemilihan dan Aturan Kurva

Metode yang banyak dipilih untuk membangkitkan kurva yang bagus adalah dengan memilih kurva secara acak. Kurva yang dibangkitkan secara acak adalah kurva dengan koefisien yang diambil dari hasil keluaran generator bilangan acak-semu. Di bawah ini adalah contoh dari sebuah algoritma yang memilih suatu kurva yang cocok di atas F_p [12]:

```
Input:  A large finite field Fp, and a small positive
integer s'.
Output: An elliptic curve E over Fp, such that E(Fp) =
        s.r, s ≤ s' and r prime.

   1. Draw E at random, with coefficients in Fp.
   2. Compute the order of E, #E(Fp).
   3. Check the MOV and anomalous conditions. If any
      one of these fails, go to Step 1.
   4. Attempt to factor #E(Fp) in 'reasonable' time.
      If attempt fails, go to Step 1.
   5. If #E(Fp) = s.r, s ≤ s', and r prime, then
      return E. Otherwise, go to Step 1.
```

Gambar 2. Contoh Algoritma Pemilihan Kurva

Bagian tersulit dari algoritma ini adalah langkah 2. Untuk alasan ini, kadang-kadang E tidak dipilih secara acak. Sebuah kelas kurva, yang dikenal sebagai kurva Koblitz, secara khusus lebih disukai karena sangat efisien dalam menghitung *ord*(P) untuk P yang berubah-ubah pada kurva, yang pada gilirannya dapat digunakan untuk menurunkan $\#E(F_p)$ secara cepat. Sedangkan tujuan dari langkah 5 adalah untuk menjamin bahwa $\#E(F_p)$ memiliki suatu faktor prima yang besar. Hal ini untuk menjaga terhadap serangan Pohlig-Hellman pada *ECDLP*.

System parameters	A finite field F, coefficients that define the curve E, a point on E called the generator G, and ord(G).
Public key	A point on curve P = kG, for some secret k.
Secret key	The integer k, where $0 < k < q$, $q = ord$(P).

Tabel 3. Melakukan *Setting* Terhadap Sebuah Sistem Kriptografi Kurva Elips

Tabel di atas adalah tabel yang berisi kebutuhan komputasi dasar untuk membangun sebuah sistem kriptografi kurva elips. Pasangan kunci mudah untuk dibangkitkan; kunci privat adalah suatu integer acak k, sedemikian sehingga kunci publik P = kG. Asumsi dasar dari *ECC* adalah sulitnya menghitung kunci rahasia k dari kunci publik P. Parameter-parameter kurva elips dapat digunakan untuk suatu kelompok pengguna, di mana setiap pengguna dalam kelompok memiliki sebuah pasangan kunci publik dan privat. Alternatif lainnya, yang menawarkan keamanan lebih tetapi membutuhkan biaya komputasi tambahan, adalah dengan menggunakan suatu kurva yang berbeda dalam *field* dasar yang sama untuk setiap pengguna. Dengan cara ini, semua pengguna membutuhkan

implementasi perangkat keras yang sama untuk aritmatika *field*, tetapi kurvanya dapat diubah secara periodik untuk tambahan keamanan [1].

3.4 Rekomendasi *Field* dan Kurva dari NIST

Federal Information Processing Standards (*FIPS*) adalah kompilasi dari standar dan petunjuk yang dikeluarkan oleh NIST untuk digunakan oleh pemerintah. *FIPS* 186-2 yang direvisi mencakup *elliptic curve digital signature algorithm* (*ECDSA*), dengan rekomendasi pada pemilihan *finite field* dan kurva elips. Kurva yang direkomendasikan ini memiliki properti khusus yang memungkinkan optimasi performansi. Di samping itu, kurva tersebut juga diperiksa untuk menjamin bahwa tidak ada kurva yang termasuk ke dalam kelas supersingular dan anomalous, yang rentan terhadap serangan MOV dan serangan lainnya.

FIPS 186-2 merekomendasikan 15 buah kurva elips di atas 10 *finite field*. Untuk setiap lima *field* prima dan lima *field* biner, sebuah kurva acak-semu dibangkitkan dengan menggunakan metode *SHA*-1 yang dispesifikasikan dalam ANSI X9.62 dan standar IEEE P1363 [18, p27]. Sebagai tambahan, sebuah kurva Koblitz dipilih untuk setiap lima *field* biner, menjadikan jumlah keseluruhan kurva 15 buah.

Hal-hal berikut ini harus dipertimbangkan ketika memilih *finite field* dan kurva elips:

I. Pilihan Panjang Kunci
   Semua kurva dipilih untuk memiliki kofaktor 1, 2, atau 4. Ini dilakukan untuk menjamin efisiensi dalam komputasi. Hasilnya, kunci privat dan publik memiliki panjang bit yang kira-kira sama [18].

II. Pilihan *Field*
   Setiap *field* dipilih sedemikian sehingga panjang *order* dalam bit setidaknya dua kali panjang kunci dari kunci privat (kunci simetri) *cipher* blok yang umum. Ini dilakukan karena suatu pencarian *exhaustive* terhadap kunci dari *cipher* blok *k* bit diharapkan memerlukan waktu yang sama dengan solusi dari sebuah *ECDLP*, ketika menggunakan alogritma *Pollard ρ* untuk suatu kurva elips yang cocok di atas sebuah *finite field* dengan suatu *order* yang panjangnya 2*k* [20]. Tabel 4 di bawah ini membandingkan panjang kunci privat dengan ukuran dari berbagai *field*.

Symmetric cipher key length	Example algorithm	Bit-length of p in prime field F_p	Dimension m of binary field F_{2^m}
80	SKIPJACK	192	163
112	Triple-DES	224	233
128	AES Small	256	283
192	AES Medium	384	409
256	AES Large	521	571

Tabel 4. Perbandingan Panjang Kunci Privat dengan Ukuran Berbagai *Field*

III. Pilihan *p* dalam GF(*p*) dan *m* dalam GF(2^m)
   Untuk *field* biner GF(2^m), *m* dipilih sedemikian sehingga terdapat sebuah kurva Koblitz dengan *order* yang hampir prima di atas GF(2^m). Untuk *field* prima GF(*p*), *p* dipilih untuk menjadi sebuah prima Mersenne, atau sebuah prima yang mirip Mersenne dengan ukuran bit kelipatan 32 [20]. Prima Mersenne adalah sebuah bilangan prima dalam bentuk 2^n-1, di mana *n* adalah prima. *Field* prima dengan *p* merupakan sebuah prima Mersenne memungkinkan reduksi modular yang efisien.

IV. Koefisien Kurva di Atas *Field* Prima
   Semua kurva yang dipilih di atas suatu *finite field* prima memenuhi persamaan $y^2 = x^3 + ax + b$, di mana a = -3. Ini memungkinkan penggandaan titik yang efisien ketika menggunakan koordinat *Jacobian*.

4. Analisis Performansi

Dalam bagian ini, implementasi perangkat lunak dari skema tanda-tangan digital *DSA* dan *RSA* akan dibandingkan dengan *ECDSA*, dan skema enkripsi ElGamal akan dibandingkan dengan kurva elips imbangannya. Eksperimen dilakukan baik pada *PC* maupun *mobile device*. Data dikumpulkan dari berbagai studi yang dilakukan oleh institusi penelitian dan eksperimen individual. Seluruh waktu adalah dalam milidetik kecuali jika dinyatakan lain. *Finite field* yang direkomendasikan NIST, dicetak miring.

4.1 Skema Tanda-Tangan Digital

Sebuah tanda-tangan digital ekivalen secara elektronik dengan tanda-tangan biasa. Ketika disertakan pada suatu dokumen elektronik, tanda-tangan digital menyediakan otentikasi dari pemiliknya, tanggal dan waktu tanda-tangan, serta isi dari dokumen tersebut. Lebih jauh, tanda-tangan harus dapat diverifikasi untuk menjamin bahwa pemiliknya tidak dapat menyangkal telah menanda-tangani dokumen setelahnya. Oleh sebab itu, suatu algoritma tanda-tangan digital harus dapat membangkitkan kunci untuk tanda-tangan, memberi tanda-tangan pada suatu dokumen, dan memverifikasi tanda-tangan tersebut.

PC

Platform : Pentium Pro 200 MHz menggunakan C, C++, dan *assembly*

Kurva : Kurva acak di atas GF(2^{191}) dan $F_{p\text{-}192}$

(Perhatikan bahwa $F_{p\text{-}192}$, di mana p = 192 adalah suatu prima yang mirip Mersenne, merupakan salah satu *finite field* yang direkomendasikan oleh NIST, dan waktunya hampir setengah dari kurva di atas GF(2^{191}), yang tidak terdapat dalam *FIPS* 186-2.)

System	Key generation	Signature	Verification	Total time
ECDSA- $F_{2^{191}}$	11.7	11.3	60	83
ECDSA-$F_{p=192}$	5.5	6.3	26	37.8
RSA-1024	1 (sec)	43.3	0.65	1,043.95
DSA-1024	22.7	23.6	28.3	74.6

Tabel 5. Skema Tanda-Tangan Digital Untuk *PC* dengan Platform Pentium Pro 200 MHz dan Kurva Acak di Atas GF(2^{191}) dan $F_{p\text{-}192}$

Platform : Pentium II 400 MHz menggunakan C

Kurva : Kurva Koblitz dan kurva acak di atas kurva NIST *GF(2^{163})*, *GF(2^{233})*, *GF(2^{183})*

Koblitz curves				
System	Key generation	Signature	Verification	Total time
ECDSA- $F_{2^{163}}$	1.47	2.11	4.09	7.67
ECDSA- $F_{2^{233}}$	3.11	4.03	7.87	15.01
ECDSA- $F_{2^{283}}$	4.50	5.64	11.46	21.6
Random Curves				
System	Key generation	Signature	Verification	Total time
ECDSA- $F_{2^{163}}$	2.12	2.64	6.46	11.22
ECDSA- $F_{2^{233}}$	4.58	5.52	14.08	24.18
ECDSA- $F_{2^{283}}$	6.88	8.08	21.15	36.11

RSA				
System	Key generation	Signature	Verification	Total time
RSA-1024	2,740.87	66.56	3.86	2811.29
RSA-2048	26,442.04	440.69	13.45	2,6896.18
DSA				
System	Key generation	Signature	Verification	Total time
DSA-768	14,735	15.55	26.13	1,4776.68
DSA-1024	54,674	24.28	47.23	5,9421.28

Tabel 6. Skema Tanda-Tangan Digital Untuk *PC* dengan Platform Pentium II 400 MHz dan Kurva Koblitz serta Kurva Acak

Hasil dari Tabel 6 menunjukkan dengan jelas bahwa kurva Koblitz menghasilkan kecepatan komputasi yang lebih efisien dibandingkan dengan *RSA* dan *DSA*, serta bersama dengan parameter-parameter yang direkomendasikan oleh NIST yang disediakan dalam *FIPS* 186-2, kurva Koblitz memiliki skema tanda-tangan digital yang jauh lebih baik dalam hal efisiensi.

PDA

Platform : (PalmPilot) Motorola Dragon Ball 15 MHz menggunakan C

Kurva : Kurva Koblitz di atas *GF(2^{163})*

System	Key generation	Signature	Verification	Total time
ECDSA-$F_{2^{163}}$	590	800	2340	3730
RSA-512	360 (sec)	5100	310	3,65410

Tabel 7. Skema Tanda-Tangan Digital Untuk *PDA*

Pager

Platform : (Pager) RIM 10 MHz menggunakan C

Kurva : Kurva Koblitz dan kurva acak di atas kurva NIST *GF(2^{163})*, *GF(2^{233})*, *GF(2^{283})*

Koblitz curves				
System	Key generation	Signature	Verification	Total time
ECDSA- $F_{2^{163}}$	751	1,011	1,826	3,588
ECDSA- $F_{2^{233}}$	1,552	1,910	3,701	7,163
ECDSA- $F_{2^{283}}$	2,369	2,760	5,485	10,614
Random Curves				
System	Key generation	Signature	Verification	Total time
ECDSA- $F_{2^{163}}$	1,085	1,335	3,243	5,663
ECDSA- $F_{2^{233}}$	2,478	3,066	7,321	12,865
ECDSA- $F_{2^{283}}$	3,857	4,264	11,587	19,708

RSA ($e = 2^{16} + 1$)				
System	Key generation	Signature	Verification	Total time
RSA-1024	580,405	15,889	1,008	597,302
RSA-2048	-	111,956	3,608	-
DSA				
System	Key generation	Signature	Verification	Total time
DSA-768	-	6,031	11,594	-
DSA-1024	-	9,529	18,566	-

Tabel 8. Skema Tanda-Tangan Digital Untuk *Pager*

Dari Tabel 7 dan Tabel 8 terlihat jelas bahwa proses verifikasi untuk *ECDSA* lebih lambat daripada *RSA*, tetapi waktu keseluruhan untuk ketiga prosedur (pembangkitan kunci, tanda-tangan, dan verifikasi) dari *ECDSA* jauh lebih cepat daripada *RSA* dan *DSA*. Untuk sebuah ukuran kunci *ECDSA* 163 di atas kurva Koblitz, diperlukan 3,588 ms untuk menyelesaikan ketiga prosedur tersebut, sementara *RSA* memerlukan 597,302 ms untuk sebuah ukuran kunci 1024. Untuk *DSA*, proses verifikasinya saja lebih lama daripada waktu keseluruhan yang diperlukan untuk *ECDSA*.

Perbandingan

Dari hasil pada Tabel 5 – Tabel 8 terlihat jelas bahwa *finite field* dan kurva Koblitz yang direkomendasikan oleh NIST menawarkan performansi yang jauh lebih tinggi dibandingkan dengan kurva dan *field* acak yang tidak terdapat dalam *FIPS* 186-2. Pada *PC*, *ECDSA* mengalahkan *RSA* dan *DSA* secara signifikan dalam waktu keseluruhan. Walaupun proses verifikasinya lebih lambat, perbedaan 2-3 ms dapat diabaikan pada *PC*. Pada PalmPilot, menggunakan sebuah kunci *RSA* 1024 bit tidak praktis, sehingga sebagai gantinya dipilih sebuah kunci 512 bit. Dibandingkan dengan kunci *ECDSA* 163 bit, *ECDSA* menunjukkan performansi keseluruhan yang lebih baik. Meskipun demikian, proses verifikasinya jauh lebih lambat daripada *RSA* 512 bit. Untuk menyelaraskan ketidakseimbangan ini, suatu kombinasi dari protokol *RSA* dan *ECDSA* dapat digunakan. Pada *pager*, waktu keseluruhan untuk *ECDSA* jauh lebih baik daripada *RSA* dan *DSA*. Kesimpulannya, *ECDSA* jelas merupakan skema *PKC* yang paling cocok dalam lingkungan-lingkungan yang memiliki keterbatasan.

4.2 Skema Enkripsi

Pada bagian ini, algoritma enkripsi dan dekripsi ElGamal akan dibandingkan dengan versi kurva elipsnya. Perbandingan akan dibuat pada sistem kriptografi dengan tingkat keamanan yang sama. Oleh sebab itu, ElGamal konvensional 768 bit harus dibandingkan dengan *ECC*-ElGamal 151 bit, sementara ElGamal konvensional 1024 bit harus dibandingkan dengan *ECC*-ElGamal 173 bit. Meskipun demikian, untuk ukuran kunci 151 dan 173 bit pada *ECC*-ElGamal, tidak ada trinomial dalam basis polinomial (PB) maupun basis normal yang dioptimasi dalam basis normal (NB) [23], sehingga sebagai gantinya akan digunakan ukuran kunci 155 dan 183 bit. Perhatikan bahwa terdapat sedikit peningkatan tingkat keamanan dalam ElGamal versi *ECC*.

Platform : Pentium II 175 MHz, Linux OS

System	Encryption	Decryption
ElGamal-768	13,100	6,640
ECC ElGamal-155 (NB)	248	123
ECC ElGamal-155 (PB)	300	139

System	Encryption	Decryption
ElGamal-1024	29,780	15,230
ECC ElGamal-183 (NB)	357	179
ECC ElGamal-183 (PB)	460	212

Tabel 9. Perbandingan Skema Enkripsi

Perbandingan

Operasi kurva elips dalam NB berjalan kira-kira 22% lebih cepat daripada PB untuk enkripsi, sementara dekripsi dalam NB kira-kira 15% lebih cepat daripada PB. Secara keseluruhan, performansi dari *ECC*-ElGamal jauh lebih baik daripada ElGamal konvensional; peningkatan dalam kecepatan keseluruhan adalah melebihi 50% [23].

5. Sebuah Peninjauan Terhadap Aplikasi-Aplikasi *ECC* Saat Ini

Ketika *ECC* pertama kali diperkenalkan pada tahun 1985, terdapat banyak keraguraguan mengenai keamanannya. Setelah studi dan penelitian yang serius dilakukan hampir satu dekade, *ECC* telah menghasilkan efisiensi dan keamanan yang tinggi. Akhir-akhir ini, banyak *vendor-vendor* produk telah memasukkan *ECC* dalam produk-produk mereka, dan jumlah ini masih terus berkembang. Ketidakpastian masih terdapat di antara beberapa pendukung dari sistem kriptografi tradisional, tetapi mereka mulai dapat menerima teknologi baru yang menjanjikan ini. Sebagai contoh, RSA Security Inc., telah sejak lama menyuarakan keamanan dari *ECC* sejak pertama kali diperkenalkan.

Meskipun demikian, dalam beberapa tahun terakhir ini, RSA Security telah melakukan penelitian pada algoritma *ECC* yang efisien, dan bahkan memperoleh sebuah paten pada algoritma konversi berbasiskan penyimpanan yang efisien. Selain itu, RSA Security juga telah menggabungkan *ECC* ke dalam beberapa produknya, mengakui fakta bahwa *ECC* telah membuktikan keamanan dan efisiensinya.

Sebuah faktor penting untuk tren yang mulai muncul ini adalah penyertaan *ECDSA* dalam beberapa standar keamanan pemerintah dan institusi penelitian, termasuk IEEE P1363, ANSI X9.62, ISO 11770-3, dan ANSI X9.63. Faktor lainnya adalah promosi yang kuat dari penggunaan *ECC* melalui suatu perusahaan Certicom yang berbasis di Kanada. Certicom adalah sebuah perusahaan yang melakukan spesialisasi dalam solusi keamanan informasi dalam suatu lingkungan komputasi *mobile* melalui penyediaan perangkat lunak dan jasa kepada para pelanggannya. Selama bertahun-tahun, Certicom telah menerbitkan banyak makalah dalam mendukung *ECC* dan juga telah mengimplementasikan *ECC* dalam seluruh produk komersialnya. Kesuksesannya telah mendorong banyak perusahaan lainnya untuk melihat lebih dekat pada keuntungan dan keamanan dari *ECC*. Saat ini, *ECC* menjadi skema kriptografi utama dalam seluruh *mobile* dan *wireless device*.

Suatu hasil peninjauan singkat dari aplikasi-aplikasi *ECC* yang terdapat di pasar sekarang ini secara umum dapat dibagi ke dalam empat kategori: Internet, *smart card*, *PDA*, dan *PC*.

6. Kesimpulan

Setelah memeriksa (membahas) keamanan, implementasi, dan performansi dari aplikasi-aplikasi *ECC* pada berbagai *mobile device*, dapat disimpulkan bahwa *ECC* merupakan skema *PKC* yang paling cocok untuk digunakan dalam suatu lingkungan dengan berbagai keterbatasan. Efisiensi dan keamanan membuat *ECC* menjadi suatu alternatif yang sangat menarik bagi sistem kriptografi konvensional, seperti *RSA* dan *DSA*, tidak hanya dalam peralatan-peralatan yang memiliki keterbatasan, tetapi juga pada komputer yang kuat. *ECC* tidak diragukan lagi, telah dikenal dengan sangat cepat sebagai skema kriptografi yang sangat kuat (baik).

DAFTAR PUSTAKA

[1] Menezes, A. J. *Elliptic Curve Public Key Cryptosystems*. Kluwer Academic Publishers, 1993.

[2] Schneier, B. *Applied Cryptography*. John Wiley & Sons, Inc., 1994.

[3] Enge, A. *Elliptic Curves and Their Applications to Cryptography*. Kluwer Academic Publishers, 1999.

[4] Menezes, A., Oorschot, P., dan Vanstone, S. *Handbook of Applied Cryptography*. CRC Press, 1997.

[5] Weisstein, E. W. "Number Field Sieve". Wolfram Research, Inc. <http://mathworld.wolfram.com/NumberFieldSieve.html>

[6] Stallings, W. *Cryptography and Network Security*. Prentice Hall, 2003.

[7] Silverman, R. D. "An Analysis of Shamir's Factoring Device". RSA Security. May 3, 1999. <http://www.rsasecurity.com/rsalabs/bulletins/twinkle.html>

[8] Shamir, A. "Factoring Large Numbers with the TWINKLE Device". In proceedings of *Cryptographic Hardware and Embedded Systems: First International Workshop, CHES'99*. Lecture Notes in Computer Science, vol.1717. Springer-Verlag Heidelberg, January 1999: p 2 – 12.

[9] Lercier, R. Homepage. <http://www.medicis.polytechnique.fr/~lercier/english/index.html>

[10] Schneier, B. "Elliptic Curve Public Key Cryptography". Cryptogram ENewsletter. November 15, 1999. <http://www.counterpane.com/cryptogram-9911.html#EllipticCurvePublic-KeyCryptography>

[11] "Remarks on the Security of the Elliptic Curve Cryptosystem". Certicom, whitepaper. September 1997. <http://www.certicom.com/research/wecc3.html>

[12] Blake, I., Seroussi, G., dan Smart, N. *Elliptic Curves in Cryptography*. Cambridge University Press, 1999.

[13] Menezes, A., Okamoto, T., dan Vanstone, S. "Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field". *Proceedings of the twenty-third annual ACM symposium on Theory of computing*. *Annual ACM Symposium on Theory of Computing*. ACM Press, 1991: p 80 – 89.

[14] Satoh, T. dan Araki, K. "Fermat Quotients and the Polynomial Time Discrete Log Algorithm for Anomalous Elliptic Curves". Commentarii Mathematici Universitatis Sancti Pauli 47, 1998: p 81 – 92.

[15] Semaev, I. A. "Evaluation of Discrete Logarithms in a Group of p-Torsion Points of an Elliptic Curve in Characteristic p". Mathematics of Computation 67, 1998: p 353 – 356.

[16] Smart, N. "The Discrete Logarithm Problem on Elliptic Curves of Trace One". *Journal of Cryptography*, vol. 12 no. 3. Springer-Verlag New York, October 1999: p 193 – 196.

[17] Certicom Press Release. "Certicom Announces Elliptic Curve Cryptosystem (ECC) Challenge Winner". November 6, 2002. <http://www.certicom.com/about/pr/02/021106_ecc_winner.html>

[18] National Institute of Standards and Technology (NIST). *Digital Signature Standard*. Federal Information Processing Standards Publication (FIPS) 186-2, January 27 2000.

[19] Omura, J. dan Massey, J. *Computational Method and Apparatus for Finite Field Arithmetic*. U.S. Patent number 4,587,627, May 1986.

[20] Brown, M., Hankerson, D., Lopez, J., dan Menezes, A. "Software Implementation of the NIST Elliptic Curves over Prime Fields". In proceedings of *Cryptographer's Track at RSA Conference 2001 San Francisco*. Lecture Notes in Computer Science, vol. 2020. Springer-Verlag Heidelberg, January 2001: 250 – 265.

[21] Lopez, J. dan Dahab, R. "Performance of Elliptic Curve Cryptosystems". Technical report IC-00-08, May 2000. Available at <http://www.ic.unicamp.br/reltec-ftp/2000/Titles.html>

[22] Boneh, D. dan Daswani, N. "Experimenting with Electronic Commerce on the PalmPilot". In proceedings of *Financial Cryptography '99*. Lecture Notes in Computer Science, vol. 1648. Springer-Verlag Heidelberg, 1999: p 1 – 16.

[23] Li, Z., Higgins, J., dan Clement, M. "Performance of Finite Field Arithmetic in an Elliptic Curve Cryptosystem". *Ninth Symposium in Modeling, Analysis and Simulation of Computer and Telecommunication Systems*. IEEE Computer Society, 2001: p 249 – 258.

[24] Itoh, T., Teecha, O., dan Tsujii, S. "A Fast Algorithm for Computing Multiplicative Inverses in GF(2^m) using Normal Basis". *Information and Computation*, vol. 79. Elvisor Academic Press, 1988: p 171 – 177.

[25] Chou, Wendy. "Elliptic Curve Cryptography and Its Applications to Mobile Devices". University of Maryland, College Park.