Elliptic Curve Cryptography: Java Implementation

Kossi D. Edoh
Department of Mathematics
North Carolina A&T State University
Greensboro, NC 27411
+1 973-449-3965
edoh@acm.org

ABSTRACT
The use of Java in developing commercial Internet applications is growing very rapidly. A major requirement for e-commerce applications is the provision of security. In this work we consider Elliptic Curve Cryptography (ECC) because of the high level of security it provides with small key sizes. ECC is ideal for use on constrained environments such as pagers, personal digital assistants, cellular phones and smart cards. We implement the ECC algorithms approved by the National Institute of Standards and Technology (NIST) in Java on the Dell Inspiron 7500. The speeds of the various ECC algorithms are provided.

Categories and Subject Descriptors
E.3. [**Data Encryption**]: Public key cryptography, standards.

General Terms
Algorithms, Measurement, Performance, and Security.

Keywords
Cryptography, Network security, NIST, and Elliptic curves.

1 INTRODUCTION
Elliptic curve cryptography is emerging as an attractive public-key cryptosystem for mobile/wireless environments. When compared to a traditional cryptosystem like RSA, ECC offers an equivalent security with smaller key sizes. The result is faster computations, lower power consumption, as well as memory and bandwidth savings.

The use of elliptic curve in cryptography was proposed by Miller [10] and Koblitz N [5]. Excellent follow-up works on ECC are Blake, Seroussi, and Smart [2], Koblitz [6], Koblitz, Menezes and Vanstone [8], Menezes [11], Menezes, van Oorschot and Vanstone [12] and Menezes, Teske, and Weng [13]. A lot of research has been done on its secure and efficient implementation including the development of software and hardware implementations. Elliptic Curve Cryptography offers more advantages compared to conventional public-key cryptosystem. Lenstra and Verheul [9] has shown that 1937-bit key size RSA [15] may provide a similar security as 190-bit key size elliptic curve cryptosystem.

InfoSecCD Conference'04, October 8, 2004, Kennesaw, GA, USA.

The security of cryptographic schemes with ECC relies on elliptic curve discrete logarithm problem. This problem can only be solved in exponential time. So far there is the lack of sub-exponential attacks on ECC.

The choice of the type of elliptic curve to use is dependent on its domain parameters: the finite field; field representation; elliptic curves algorithms for field arithmetic; elliptic curve arithmetic; and protocol arithmetic [4]. The optimum selection of these parameters depends on the security conditions under consideration. There are several research papers on this subject and covering different areas like hardware and software implementations.

In this paper we consider the Java implementation of the NIST-recommended elliptic curves over prime and binary fields on Dell Inspiron with 600MH Pentium III processor. The use of Java in developing a wide range of Internet applications makes it the appropriate language for writing these programs. Most of the programs can be enhanced and optimized for resource constrained environments. We have provided the speed for some ECC algorithms including elliptic curve digital signature algorithm (ECDSA).

The speeds of the algorithms given in this paper will provide a basis for the creation of better algorithms with efficient Java implementations in the future. The more efficient ECC Java codes will eventually be included in Java SDK. Students will be able to implement the algorithms provided in the paper on their own and the results compared to that using Java Cryptography Extensions of traditional cryptosystems in Java 2 SDK Standard Edition, v 1.4. This will promote the understanding of information security issues. It will also reinforce students' ability to analyze, design and implement object-oriented software in Java.

In the next section we describe two finite fields, their element representation, and the field arithmetic. In section three we define two types of elliptic curves with points on each curve satisfying the property of finite fields. We also describe algorithms for various field arithmetic and elliptic point scalar multiplication in this section. In section four we discuss Koblitz curves, a special type of elliptic curves and in section five is the conclusion.

2 FINITE FIELDS
There are several choices that need to be made before implementing an ECC system. Among them is the elliptic curve underlying field and field representation. In this section we define finite fields and show that the points on elliptic curves satisfy the

properties of finite fields. We also describe the field element representation and the algorithms for the field arithmetic.

A finite field is an algebraic system consisting of a finite set F, with addition and multiplication defined on it, and satisfying the following axioms:

- F is an abelian group with respect to addition,
- $F-0$ (0 the zero element) is an abelian group with respect to multiplication,
- and multiplication is distributive over addition.

The number of elements in the field is called the *order* of the finite field. There exists a finite field of order q if and only if q is a prime power [14]. Consider the finite field denoted F_q of order q. If $q = p^m$, where p is prime and m a positive integer, then p is called the *characteristic* and m the extension degree of F_q. In ECC the order of F is set to $q = p$ (prime field) or $q = 2^m$ (binary field).

2.1 Example of Finite Fields

In this section we describe two finite fields which are commonly used in ECC and their corresponding arithmetic.

The finite field F_p. The prime field F_p is the set $F_p = \{0,1,2,\ldots,p-1\}$ with addition and multiplication defined on it as follows: if a, b $\in F_p$, then

$$a + b \equiv r (mod\ p) \quad \text{and} \quad a.b \equiv s (mod\ p).$$

The additive inverse of a is denoted $\overline{a}$ where $a + \overline{a} \equiv 0\ mod\ p$ and the multiplicative inverse a^{-1} where $a * a^{-1} \equiv 1\ mod\ p$. For the NIST recommended elliptic curves, p is chosen such that $\lceil log_2 p \rceil \in \{112,128,160,192,224,256,384,521\}$.

The finite field F_{2^m}. This field is called a characteristic 2 finite field with 2^m elements. Among the several basis representations of elements in F_{2^m} are the polynomial and the normal basis.

Polynomial basis. Let $a \in F_{2^m}$, then a has the form

$$a = \sum_{i=0}^{m-1} a_i x^i, \qquad a_i \in \{0,1\}. \tag{2.1}$$

The set $\{x^0, x^1, x^2, ..., x^{m-1}\}$ forms a basis of F_{2^m} over F_2. The element a is represented by the binary vector $(a_{m-1},\ldots,a_1,a_0)$. Let

$$f(x) = x^m + f_{m-1}x^{m-1} + \ldots + f_1 x + f_0 \tag{2.2}$$

where $f_i \in \{0,1\}$ be an irreducible polynomial of degree m over F_2. The following is the finite field arithmetic using polynomial basis. *Addition*: Let $a = (a_{m-1},\ldots,a_1,a_0)$ and $b = (b_{m-1},\ldots,b_1,b_0)$ then $a + b = r \in F_{2^m}$ with $r = (r_{m-1},\ldots,r_1,r_0)$ where $r_i \in \{0,1\}$ and $r_i \equiv (a_i + b_i) mod\ 2$. *Multiplication*: Let $a.b = s \in F_{2^m}$ with $s = (s_{m-1},\ldots,s_1,s_0)$ where $s_i \in \{0,1\}$ is the remainder when $a.b$ is divided by some irreducible function $f(x)$ with all coefficients performed modulo 2.

The additive identity is $(0,0,\ldots,0)$ and the multiplicative identity $(0,0,\ldots,1)$. For the NIST recommended curves $m \in \{113,131,163,193,233,239,283,409,571\}$ with the following corresponding irreducible functions.

Field	*Irreducible function*
$F_{2^{113}}$	$f(x)=x^{113}+x^9+1$
$F_{2^{131}}$	$f(x)=x^{131}+x^8+x^3+x^2+1$
$F_{2^{163}}$	$f(x)=x^{163}+x^7+x^6+x^3+1$
$F_{2^{193}}$	$f(x)=x^{193}+x^{15}+1$
$F_{2^{233}}$	$f(x)=x^{233}+x^{74}+1$
$F_{2^{239}}$	$f(x)=x^{239}+x^{36}+1$
$F_{2^{283}}$	$f(x)=x^{283}+x^{12}+x^7+x^5+1$
$F_{2^{409}}$	$f(x)=x^{409}+x^{87}+1$
$F_{2^{571}}$	$f(x)=x^{571}+x^{10}+x^5+x^2+1$

Normal basis. A normal basis of F_{2^m} over F_2 is of the form $\{\alpha, \alpha^2, \ldots, \alpha^{2^{m-1}}\}$ where $\alpha \in F_{2^m}$. Let $a \in F_{2^m}$, then a has the representation

$$a = \sum_{i=0}^{m-1} a_i \alpha^{2^i} \qquad a_i \in \{0,1\}. \tag{2.3}$$

The element a can be represented as a binary vector $(a_{m-1},\ldots,a_1,a_0)$. Let

$$f(x) = x^m + f_{m-1}x^{m-1} + \ldots + f_1 x + f_0 \tag{2.4}$$

where $f_i \in \{0,1\}$ be an irreducible polynomial of degree m over F_2. The following is the field arithmetic defined on F_{2^m}. *Addition*: Let $a = (a_{m-1},\ldots,a_1,a_0)$ and $b = (b_{m-1},\ldots,b_1,b_0)$ then $a + b = r \in F_{2^m}$ with $r = (r_{m-1},\ldots,r_1,r_0)$ and $r_i \equiv (a_i + b_i) mod\ 2$. *Squaring*: $a.a = (a_{m-1}, a_0, a_1, \ldots, a_{m-2})$ is just a cyclic shift in the binary vector representation of the element a. *Multiplication*: This is more difficult. Please see [1] for implementation details.

2.2 Arithmetic in F_{2^m} with Polynomial Basis

In this section we look at the various arithmetic operations in F_{2^m} (2.1) and present some suitable corresponding algorithms. The platform for our implementations is Dell Inspiron with 600MHz Pentium III processor. We use 163-bit key with field elements represented by array of bytes $C[163]$ or as a BigInteger c. The array A is the two's complement representation of the BigInteger a. *Addition*: Let $a, b \in F_{2^m}$, then $a + b \equiv a \oplus b$. *Modula reduction*: Multiplication in F_{2^m} requires the reduction of the product modulo an irreducible polynomial $f(x)$.

Algorithm 1. Modular reduction in F_{2^m}.

Input: $a = (a_{2m-2},\ldots,a_1,a_0)$ and $f = (f_m,\ldots,f_1,f_0)$.

Output: c = a mod f.

for i = 2m-2 to m

for j = 0 to m-1

if $_j \neq 0$ then $a_{i-m+j} \leftarrow a_{i-m+j} + a_i$

return $c \leftarrow (a_{m-1}...a_1a_0)$.

Squaring: The square of $a(x)$ is given by

$$a(x)^2 = (\sum_{i=0}^{m-1} a_i x^i)^2 = \sum_{i=0}^{m-1} a_i x^{2i}. \quad (2.5)$$

Below is the algorithm for squaring an element $a \in F_{2^m}$ based on the information in equation (2.5).

Algorithm 2. Squaring in F_{2^m}.

Input: $a = (a_{m-1},..,a_1,a_0)$ and $f = (f_m,..,f_1,f_0)$.

Output: c = a^2 mod f.

$b \leftarrow \sum_{i=0}^{m-1} a_i^2 x^{2i}$

$c \leftarrow b$ mod f

return c.

Multiplication. The basic algorithm for multiplication is the "shift-and-add" method. Let $a(x) \in F_{2^m}$, then

$xa(x) mod\ f(x) = \sum_{j-1}^{m-1} a_{j-1} x^j$ if $a_{m-1} = 0$ else

$xa(x) mod\ f(x) = \sum_{j-1}^{m-1} a_{j-1+f_j} x^j + f_0$ if $a_{m-1} \neq 0$. The following algorithm performs the multiplication of two elements a and b using the "shift-and-add" method.

Algorithm 3. The 'shit-and-add' method.

Input: $a,b \in F_{2^m}$ and $f = (f_m,..f_1,f_0)$.

Output: c = ab mod f.

$c(x) \leftarrow 0$

for j from m-1 to 0

c(x) $\leftarrow xc(x) mod\ f(x)$

if a $_j \neq 0$ then $c(x) \leftarrow c(x) + b(x)$

return (c).

Inversion. The basic method for inversion is the extended Euclidean algorithm as described below.

Algorithm 4: Extended Euclidean algorithm.

Input: $a \in F_{2^m}, a \neq 0$ and $f = (f_m,..,f_1,f_0)$.

Output: $c = a^{-1} mod\ f$.

$b_1(x) \leftarrow 1, b_2(x) \leftarrow 0$

$p_1(x) \leftarrow a(x), p_2(x) \leftarrow f(x)$

while $degree(p_1) \neq 0$

if degree (p_1) $< degree(p_2)$ then

exchange p_1, p_2 and b_1, b_2

j $\leftarrow degree(p_1) - degree(p_2)$

$p_1(x) \leftarrow p_1(x) + x^j p_2(x)$

$b_1(x) \leftarrow b_1(x) + x^j b_2(x)$

return $c(x) \leftarrow b_1(x)$.

Timings. The table below shows the running times of the various algorithms discussed in this section. Each operation is performed 1000 times where at each time step different random values are used as input. The running time is then the average.

Table 1: The timings in microseconds

Operation	Time
Multiplication "shit-add Method"	379.45
Inversion	1109.82
Reduction	28.01
Addition	11.45
Squaring	95.62

3 ELLIPTIC CURVES

In this section we give a brief overview of elliptic curves over the finite fields F_{2^m} and F_p. We point out that, the choice of elliptic curve parameters depends on the security considerations, platform's hardware and software constraints, and bandwidth constraints.

3.1 The Elliptic Curves Over F_p and F_{2^m}

The following is the definition of elliptic curve over F_p.

Let p be a prime in F_p and $a,b \in F_p$ such that $4a^3 + 27b^2 \neq 0\ mod\ p$ in F_p, then an elliptic curve $E(F_p)$ is defined as

$$E(F_p) := \{p(x,y), x,y \in F_p\} \quad (3.6)$$

such that $y^2 = x^3 + ax + b\ mod\ p$ together with a point O, called the point at infinity.

Below is the definition of addition of points P and Q on the elliptic curve $E(F_p)$. Let $P(x_1,y_1)$ and $Q(x_2,y_2)$ then

$$P+Q = \begin{cases} O & if\ x_1 = x_2\ and\ y_2 = -y_1, \\ Q = Q + P & if\ P = O, \\ (x_3,y_3) & otherwise, \end{cases}$$

where

$$x_3 = \lambda^2 - x_1 - x_2,$$
$$y_3 = \lambda(x_1 - x_3) - y_1,$$

and

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

The point $p(x,-y)$ is said to be the negation of $p(x,y)$. The following is the algorithm for performing addition of points on elliptic curve $E(F_p)$.

Algorithm 5. Addition of points on $E(F_p)$.

Input: Elliptic curve $E(F_p)$ with the points

$P_1(x_1, y_1)$ and $P_2(x_2, y_2)$.

Output: $Q = P_1 + P_2$.

if $P_1 = O$ *then return* $Q = P_2$

if $P_2 = O$ *then return* $Q = P_1$

if $x_1 = x_2$ then

if $y_1 = y_2$ then $\lambda \leftarrow (3x_1^2 + a)/(2y_1) mod\ p$

else return (Q $\leftarrow O$)

else $\lambda \leftarrow (y_2 - y_1)/(x_2 - x_3)\ mod\ p$

$x_3 \leftarrow \lambda^2 - x_1 - x_2$ mod p

$y_3 \leftarrow \lambda(x_1 - x_3) - y_1$ mod p

return $Q \leftarrow (x_3, y_3)$.

The elliptic curve over F_{2^m} is defined as follows:

Denote the (non-supersingular) elliptic curve over F_{2^m} by $E(F_{2^m})$. If $a, b \in F_{2^m}$ such that $b \neq 0$ then

$$E(F_{2^m}) = \{p(x, y), x, y \in F_{2^m}\} \tag{3.7}$$

such that $y^2 + xy = x^3 + ax^2 + b \in F_{p^m}$ together with a point O, called the point at infinity.

The addition of points on $E(F_{2^m})$ is given as follows: Let $P(x_1, y_1)$ and $Q(x_2, y_2)$ be points on the elliptic curve $E(F_{2^m})$, then

$$P + Q = \begin{cases} O & \text{if } x_1 = x_2 \text{ and } y_2 = -y_1, \\ Q = Q + P & \text{if } P = O, \\ (x_3, y_3) & \text{otherwise.} \end{cases}$$

If $P \neq \pm Q$,

$$\lambda = \frac{y_2 + y_1}{x_2 + x_1},$$
$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a,$$
$$y_3 = \lambda(x_1 + x_3) + x_3 + y_1.$$

If $P = Q$,

$$\lambda = x_1 + \frac{x_1}{y_1},$$
$$x_3 = \lambda^2 + \lambda + a,$$
$$y_3 = \lambda(x_1 + x_3) + x_3 + y_1.$$

The following is the corresponding algorithm for adding two points on the elliptic curve $E(F_{2^m})$.

Algorithm 6. Addition of points on $E(F_{2^m})$.

Input: Elliptic curve $E(F_{2^m})$ with the points $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$.

Output: $Q = P_1 + P_2$.

if $P_1 = O$ *then return* $Q = P_2$

if $P_2 = O$ *then return* $Q = P_1$

if $x_1 = x_2$ then

if $y_1 = y_2$ then $\lambda \leftarrow (x_1 + y_1/x_1)$

$x_3 \leftarrow \lambda^2 + \lambda + a$

else return (Q $\leftarrow O$)

else $\lambda \leftarrow (y_2 + y_1)/(x_2 + x_1)$

$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a$

$y_3 \leftarrow \lambda(x_1 + x_3) + x_3 + y_1$

return $Q \leftarrow (x_3, y_3)$.

3.2 Elliptic Curve Point Multiplication

Elliptic curve cryptographic schemes are based on elliptic curve point multiplication. Given an integer k and a point P on $E(F_q)$, the multiplication kP is the result of adding P to itself k times. The *order* of a point $P \in E(F_q)$ is the smallest positive integer r such that $rP = O$. Note that if l is an integer, then $kP = lP$ if and only if $k \equiv l\ mod\ r$. The number of points on $E(F_q)$, denoted by $\#E(F_q)$ is called the curve order. This number can be computed in polynomial time by Schoof's algorithm [16].

The security of elliptic curve cryptosystem is based on the apparent intractability of elliptic curve discrete logarithm problem. The following is the *Elliptic Curve Discrete Logarithm Problem*. Given an elliptic curve $E(F_q)$ defined over F_q and two points $P, Q \in E(F_q)$, find an integer $k, 0 \leq k \leq r - 1$ such that $Q = kP$, provided such an integer exists.

The cryptographic domain parameters of an elliptic curve over F_q are the septuple,

$$T = (q, FR, a, b, G, r, h) \tag{3.8}$$

consisting of q specifying $(q = p\ or\ q = 2^m)$, field element representation method FR, constants $a, b \in F_q$ that specify the elliptic curve, $G = (x_G, y_G)$ a base point on $E(F_q)$, r the order of G and h the cofactor given by $h = \#E(F_p)/r$. The cofactor is always selected as small as possible.

The scalar multiplication of a point P by an integer k is defined by

$$Q = kP := P + P + \ldots + P. \tag{3.9}$$

3.2.1 Methods for Scalar Multiplication

Here we consider various methods for computing the scalar multiplication of points on an elliptic curve.

The Binary method. This method is based on the binary representation of k

$$k = \sum_{i=0}^{j-1} k_j 2^j \quad \text{where} \quad k_j \in \{0,1\}, \tag{3.10}$$

and

$$kP = \sum_{i=0}^{j-1} k_j 2^j P. \tag{3.11}$$

A better representation of k known as the **Non-Adjacent Form** (NAF) is given by

$$k = \sum_{i=0}^{j-1} k_j 2^j \quad \text{where} \quad k_j \in \{-1,0,1\} \tag{3.12}$$

such that no two consecutive digits of k_j are nonzero.

A more generalization of the binary method is the window method where a number of pre-computed points may be required. Every positive number k has a unique **width-w nonadjacent form**

$$k = \sum_{i=0}^{j-1} u_j 2^j \quad \text{where } u_i \text{ is odd and } |u_i| < 2^{w-1} \tag{3.13}$$

and among any w consecutive coefficients, at most one is nonzero. The next algorithm computes the $width-wNAF$ is defined as $NAF_w(k) := (u_{l-1}...u_1u_0)$.

Algorithm 7. Computation of $NAF_w(k)$.

Input: An integer k.

Output: $NAF_w(k) = (u_{l-1}...u_1u_0)$.

$c \leftarrow k, \quad l \leftarrow 0$

while c > 0

if c is odd

$\bar{u}_l \leftarrow 2 - (c \bmod 2^w)$

if $u_l > 2^{w-1}$ then $u_l \leftarrow u_l - 2^w$

if $u_l < 2^{w-1}$ then $u_l \leftarrow u_l + 2^w$

$c \leftarrow c - u_l$

else $u_l \leftarrow 0$

$c \leftarrow c/2, \quad l \leftarrow l+1$

return $(NAF_w(k) \leftarrow (u_{l-1}...u_1u_0))$.

Using the values of $NAF_w(k)$, the resulting algorithm for computing kP is given as follows:

Algorithm 8. The width-w windows method.

Input: Integers k and w and a point $P(x,y) \in E(F_q)$.

Output: $Q = kP \in E(F_q)$.

$P_0 \leftarrow P, \quad T \leftarrow 2P$

for i from 1 to $2^{w-2} - 1$

$P_i \leftarrow P_{i-1} + T$

Compute $NAF_w(k) = (u_{l-1}..u_1u_0)$

$Q \leftarrow O$

for j from $l-1$ to 0

$Q \leftarrow 2Q$

if $u_j \neq 0$ then

$i \leftarrow (|u_j| - 1)/2$

if $u_j > 0$ then $Q \leftarrow Q + P_i$

else $Q \leftarrow Q - P_i$

return Q.

4. KOBLIZ CURVES

These curves were first introduced by Koblitz [7]. The fasted method for computing kP on Koblitz curves is due to Solinas [17], [18]. Koblitz curves are special elliptic curves over F_{2^m} with coefficient a either 0 or 1 and $b = 1$. The curves have the property that if (x,y) is a point on $E(F_{2^m})$ so is (x^2, y^2). This condition satisfies the relation

$$(x^4, y^4) + 2P = \mu.(x^2, y^2) \tag{4.14}$$

where $\mu = (-1)^{1-a}$. Using the Frobenius map $\tau(x,y) = (x^2, y^2)$, equation 4.14 can be written as $\tau^2 P + 2P = \mu\tau P$. The map can be expressed as a complex number $\tau = \frac{\mu + \sqrt{-7}}{2}$ satisfying $\tau^2 + 2 = \mu\tau$. The idea is to replace the coefficient k by a τ-adic number $\bar{k} = \sum_{i=0}^{l-1} \bar{k}_i 2^i$ where $k \equiv \bar{k}$ and to compute $\bar{k}P$ as

$$\bar{k}P = \bar{k}_{l-1}\tau^{l-1}P + \ldots + \bar{k}_0 P.$$

In this case the calculation of kP is done using Frobenius map in place of point doubling.

In our computations we use the Koblitz curve and the polynomial basis representation with the following parameters.

Parameter	*Value*
Bit size	163
T	4
$f(x)$	$x^{163} + x^7 + x^6 + x^3 + 1$
Curve	$y^2 + xy = x^3 + ax^2 + 1$
a	1
Cofactor f	2
Curve order n	*fr*
Gx	2 *fe*13*c*053 7*bbc*11*ac aa*07*d*793 *de*4*e*6*d*5*e* 5*c*94*eee*8
Gy	2 89070 *fb*0 5*d*38 *ff* 58 321*f* 2*e*80 0536*d*538 *ccdaa*3*d*9
G Order r	5846006549323611672814741753598448348329118574063

The following are the times for the various algorithms for computing the elliptic curve point multiplication. Each method is performed 100 times where at each time different random input values are used. The time is then the mean.

Table 1. The timings in milliseconds

Method	Time
Binary	112.63
Binary NAF	88.83
NAF_4 window	68.41
$\tau - adic$	34.94
$\tau - adic\ width - w(w=4)$	26.67

We have also implemented elliptic curve digital signature algorithm (ecdsa) using both elliptic curves in F_p and F_{2^m} representations. For more accuracy we compute the mean after performing each method 100 times with different input values.

The following are the domain parameters of the elliptic curve F_p used in our computations.

Parameter	*Value*
Bit size	192
Curve	$y^2 = x^3 - 3x + b(mod)p$
b	64210519*e*59*c*80*e*70 *fa*7*e*9*ab*72243049 *feb*8*deecc*146*b*9*b*1
Output c of	3099*d*2*bbbfcb*2538542*dcd*5*f*
SHA−1 *algo.*	*b*078*b*6*ef* 5*f* 3*d*6 *fe*2*c*745*de*65
Seed s to	3045*ae*6 *fc*8422 *f* 64*ed*579
SHA−1 *algo.*	528*d*38120*eae*12196*d*5
Cofactor f	1
$\#E(F_p)$ *n*	*fr*
Gx	188*da*80*eb*03090 *f* 67*cbf* 20*eb*43*a*18800 *f* 4 *ff* 0*afd*82 *ff*1012
Gy	07192*b*95 *ffc*8*da*78631011*ed*6*b*24*cdd*573 *f* 977*a*11*e*794811
G Order r	6277101735386680763835789423176059013767194773182842284081
Prime p	6277101735386680763835789423207666416083908700390324961279

Table 3. The timing for ecdsa algorithm in milliseconds

Method	ECDSA F_p	ECDSA F_{2^m}
Signature	84	47
Verification	460	245

5. CONCLUSION

We provide a brief overview of elliptic curve cryptography and obtain the run-times of the resulting algorithms when implemented in Java. We are working on improving the run-times of the algorithms for possible inclusion in the future versions of Java JCE. We want to build ECC software for providing socket layer security (similar to OpenSSL [19]) on small mobile devices and also for use in multicast security [3].

6. ACKNOWLEDGMENTS

This work is partially supported by the Separated Budgeted Research Grant from Montclair State University.

7. REFERENCES

[1] Agnew, G.B., Mullin, R. C., and Vanstone, S. A. An Implementation of elliptic curve cryptosystem over F_{155}. *IEEE journal on selected areas in communication*, Vol 11, No 5, 1993 804-813.

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

[3] Edoh K. D., and Wahab H. Multicast security: Issues and new schemes for key management. *Proceedings of 2003 Information Resource Management Association International Conference*, 2003, 123-126.

[4] Hankerson, D., Hernandez, L. J., and Menezes A. Software implementation of elliptic curve cryptography over binary fields, *CHESS* 2000, LNCS 1965, 2000, 1-24.

[5] Koblitz N. Elliptic curve cryptosystem. *Mathematics in Computation.* (48), 1987, 203-209.

[6] Koblitz N. *A Course in Number Theory and Crytography*, Springer Verlag. 2nd Ed, 1994.

[7] Koblitz N. CM-curves with good cryptographic properties. *Advances in Cryptology-CRYPTO '91, LNCS 576, Springer-Verlag*, 1992, 279-287.

[8] Koblitz N., Menezes A.J., and Vanstone S.A. The state of elliptic curve cryptography. *Design, Codes, and Cryptography.* Vol 19, Issue 2-3, 2000, 173-193.

[9] Lenstra A., and Verheul E. *Selecting cryptographic key sizes.* Third International Workshop on Practice and Theory in Public Key Cryptography-PKC 2000. LNCS 1751, 2000.

[10] Miller V. Use of elliptic curves in cryptography. *Advances in Cryptography-Crypto '85*. LNCS 218, Springer Verlag, 1986, 417-426.

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

[12] Menezes A.J., van Oorschot P.C., and Vanstone S.A. *Handbook of Applied Cryptography.* CRC Press, 1997.

[13] Menezes A.J., Teske E., and Weng A. *Weak fields for ECC.* CORR 2003-15, Technical Report, University of Waterloo, 2003.

[14] McEliece R.J. *Finite Fields for Computer Scientists and Engineers*. Kluwer Academic Publishers, 1987.

[15] Rivest R., Schamir A. and Adleman. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21, 1978, 120-126.

[16] Schoof R. Elliptic curves over finite fields and the computation of square roots mod p. *Math. Comp.*, 44, 1985, 483-494.

[17] Solinas J. An improved algorithm for arithmetic on a family of elliptic curves. *Advances in Cryptology, Proc. Crypto '9.* LNCS 1294 Kaliski B., Ed., Springer-Verlag, 1997, 357- 371.

[18] Solinas J. Efficient arithmetic on Koblitz curves. *Design, Codes and Cryptography, 19*, 2000, 195-249.

[19] *The openSSL project*, http://www.openSSL.org. 1998-2003.