International Journal of Security and Its Applications
Vol. 3, No. 3, July, 2009

Elliptic Curve Cryptography on PocketPCs*

Kossi Edoh

Department of Mathematics
NC A&T State University
Greensboro, NC 27411
edoh@acm.org

Abstract

The commercial use of small mobile computer devices by enterprise and government organizations is on the rise as wireless networking is becoming very popular and evolving very fast. Elliptic Curve Cryptography (ECC) seems very useful for providing a high level of security on these devices with small key sizes compared to the traditional public-key cryptographic systems. In this work we implement the National Institute of Standards and Technology (NIST) recommended ECC algorithms on Pocket PCs. The programs are written in Java since a vast array of Internet applications service infrastructure is designed around Java technology. We show that Elliptic Curve Digital Signature Algorithm (ECDSA) can run in a suitable time with sufficient level of security.

***Keywords**: Cryptography, Network security, NIST, Elliptic curves*

1. Introduction

The use of small computer devices with wireless communication abilities is expected to drive the future development of many Internet transactions. The number of people using small devices far exceeds that of Personal Computers. To tap this potential the security issues with applications on these devices has to be addressed. Several approaches to enhance the security are based on cryptographic primitives such as hash functions, message authentication codes, and digital signatures. The limitations of providing a high level of security using these primitives include: the memory needed for generating cryptographic keys; the storage needed for storing the key generating algorithms as well as the keys; the bandwidth necessary to transmit keys; and the CPU to generate keys.

Elliptic Curve Cryptography has emerged as an attractive public-key cryptosystem for use in small wireless environments. Compared to the conventional cryptosystems like RSA, ECC offers a higher level of security with small key sizes, resulting in faster computations, lower power consumption, as well as memory and bandwidth savings. These properties make ECC very useful on small mobile devices and smartcards which are typically limited in terms of their CPU, power and network connectivity. Lenstra and Verheul [9] have shown that 1937-bit key size RSA may provide a similar security as 190-bit key size elliptic curve cryptosystem.

The use of elliptic curve in cryptography was proposed by Miller [14] and Koblitz [5] with an excellent follow-up work by Blake, Seroussi and Smart [2], Koblitz [9, 7], Koblitz, Menezes and Vanstone [8], Menezes [11], and Menezes, van Oorschot and Vanstone [12]. It has been endorsed by the NIST. Among the initial implementations is that of elliptic curves over F_{155} by Agnew et. al. [1] and other related security issues by Edoh et. al. [3]. Devices

*This work is partially supported by NSA Grant BAA-002-06, H98230-06-C-0508

using applications running ECC include Cellular phones, Smart cards, Cisco firewalls, and E-cash systems. Future applications are in ATMs, and Identification systems like passports and driving license.

We implement the NIST recommended elliptic curves over binary fields F_{2^m}, in Java on HP iPAQ h5450 PocketPC with 400MHz Intel Scale processor, 64MB memory and 48MB ROM. We use a 163-bit key which is known to provide security equivalent to 1024-bit RSA [9, 15]. This level of security is sufficient for most electronic commerce applications as long as there are no improvements in solving the elliptic curve discrete-log problem. The use of Java in developing a wide range of Internet applications makes it an appropriate programming language for implementing these programs. We have provided the speeds of some ECC algorithms including the elliptic curve digital signature algorithm (ECDSA). These results are intended to provide a basis of comparison for future algorithms. The enhanced algorithms will then be included in the future versions of the Java Cryptographic Extensions (JCE) a component of the Software Development Kit (SDK). We recommend the creation of a special primitive Java datatype for handling ECC field elements. This will help speed up the computation time.

In the next section we discuss the finite field F_{2^m}, its element representation and arithmetic. In section three is the definition of elliptic curves and algorithms for elliptic point scalar multiplication. Section four is the numerical results using Koblitz curves, a special type of elliptic curves, and in section five is the conclusion.

2. Finite Fields

The use of elliptic curve algorithms depends on the domain parameters: the finite field satisfied by points on the elliptic curve; the field representation; elliptic curve algorithms for field arithmetic; elliptic curve arithmetic; and protocol arithmetic [4]. The optimum selection depends on the security properties under consideration. In this section we define the finite field F_{2^m}, its element representation and 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 its *order*. There exists a finite field of order *q* if and only if q is of a prime power [10]. 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 $q = p^m$ (binary field).

2.1 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.

2.1.1. Polynomial basis

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

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

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

$$f(x) = x^m + f_{m-1}x^{m-1} + f_{m-2}x^{m-2} + \cdots + f_1 x + f_0,$$

where $f_i \in \{0,1\}$ is an irreducible polynomial of degree m over F_2. The following is the finite field arithmetic using polynomial basis.

Addition: Let $a = (a_{m-1}, a_{m-2}, \ldots, a_1, a_0)$ and $b = (b_{m-1}, b_{m-2}, \ldots, b_1, b_0)$ then $a + b = r \in F_{2^m}$ with $r = (r_{m-1}, r_{m-2}, \ldots, r_1, r_0)$ 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 $s = (s_{m-1}, s_{m-2}, \ldots, s_1, s_0)$ 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 elliptic curves, the values of m is chosen such that $m \in \{113, 131, 163, 193, 233, 239, 283, 409, 571\}$ with the following corresponding irreducible functions,

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

2.1.2. Normal basis

A normal basis of F_{2^m} over F_2 has the form $\{\alpha, \alpha^2, \alpha^3, \ldots, \alpha^{2^{m-1}}\}$ where $\alpha \in F_{2^m}$. Let $a \in F_{2^m}$, then a is of the form

$$a = \sum_{i=0}^{m-1} a_i \alpha^{2^i}, \quad a_i \in \{0,1\},$$

and can be represented by the binary vector $(a_{m-1}, a_{m-2}, \ldots, a_1, a_0)$.

Let

$$f(x) = x^m + f_{m-1}x^{m-1} + f_{m-2}x^{m-2} + \cdots + f_1 x + f_0$$

where $f_i \in \{0,1\}$ is an irreducible polynomial of degree m over F_2. The following is the field arithmetic defined on F_{2^m}.

Addition: Let $a = (a_{m-1}, a_{m-2}, \ldots, a_1, a_0)$ and $b = (b_{m-1}, b_{m-2}, \ldots, b_1, b_0)$ then $a + b = r \in F_{2^m}$ with $r = (r_{m-1}, r_{m-2}, \ldots, r_1, r_0)$ and $r_i \equiv (a_i + b_i) mod\ 2$.

Squaring: $a \cdot a = (a_{m-2}, \ldots, a_1, a_0, \ldots, a_{m-1})$ is just a cyclic shift in the vector representation of the element a.

Multiplication: See [1] for details.

2.2. Finite field arithmetic in F_{2^m} with polynomial basis

In this section we look at the various arithmetic operations in F_{2^m} and present some suitable corresponding algorithms.
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: $a = (a_{2m-2}, a_{2m-1}, \dots, a_1, a_0)$ and $f = (f_m, f_{m-1}, \dots, f_1, f_0)$
Output: $c = a \bmod f$
for i = 2m-2  to m
        for j = 0 to m -1
                if $f_j \neq 0$ then $a_{i-m+j} \leftarrow a_{i-m+j} + a_i$
return $c \leftarrow (a_{m-1}, a_{m-2}, \dots, a_1, a_0)$
```

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

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

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

Algorithm 2: Squaring in F_{2^m}

```
Input: $a = (a_{m-1}, a_{m-2}, \dots, a_1, a_0)$ and $f = (f_m, f_{m-1}, \dots, f_1, f_0)$.
Output: $c = a^2 \bmod f$
$b \leftarrow \sum_{i=0}^{m-1} a_i^2 x^{2i}$
$c \leftarrow b \bmod f$
                    return $c$.
```

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

$$xa(x) \bmod f(x) = \sum_{i=1}^{m-1} a_{i-1} x^i, \ if \quad a_{m-1} = 0, \qquad else$$

$$xa(x) \bmod f(x) = \sum_{i=1}^{m-1} a_{i-1} + f_i x^j + f_0, \ if \ a_{m-1} \neq 0.$$

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 $f = (f_m, f_{m-1}, \dots, f_1, f_0)$.
Output: $c = ab \bmod f$
$c(x) \leftarrow 0$.
for j = from m-1 to 0
    $c(x) \leftarrow xc(x) \bmod f(x)$
    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 $f = (f_m, f_{m-1}, \dots, f_1, f_0)$
Output: $c = a^{-1} \bmod f$
$b_1(x) \leftarrow 1$, $b_2(x) \leftarrow 0$
$p_1(x) \leftarrow a(x)$, $p_2(x) \leftarrow f(x)$.
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 on PocketPC. The description of the numerical implementation is described in section 4.

Table 1. The timings in microseconds

Operations	Time
Multiplication with ‘shift-and-add’ method	148.34
Inversion	1552.66
Reduction	36.90
Squaring	118.82
Addition	14.38

3. Elliptic Curves

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

3.1. The elliptic curve over F_{2^m}

The following is the definition of elliptic curves F_{2^m}.

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

$$E(F_{2^m}) = \{y^2 + xy = x^3 + ax^2 + b,\ x, y \in F_{2^m}\}$$

together with a point O, called the point at infinity.

The addition of points on $E(F_{2^m})$ is defined 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

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

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 $P = Q$,

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

Algorithm 5 is the adding of two points on the elliptic curve $E(F_{2^m})$.

```
Algorithm 5: 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 $\lambda \leftarrow (x_1 + \frac{y_1}{x_1})$
            $x_3 \leftarrow \lambda^2 + \lambda + a$
        else return $(Q \leftarrow O)$
else $\lambda \leftarrow (y_2 + y_1)/(x_2 + x_1)$,
    $x_3 \leftarrow \lambda^2 + \lambda + x_1 + x_2 + a$
$y_3 \leftarrow \lambda(x_1 + x_3) + x_3 + y_1$
return $Q \leftarrow (x_3, y_3)$
```

3.2. Elliptic curve point multiplication

ECC schemes are based on elliptic curve point multiplication. Given an integer k and a point P on $E(F_q)$, the product 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 $k \equiv l \bmod r$. The number of points on $E(F_q)$, denoted by $\#E(F_q)$ is called the *order* of the curve. 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 $k, 0 \le k < r-1$ such that $Q = kP$, provided such an integer exists.

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

$$T = (q, FR, a, b, G, r, h)$$

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 then defined by

$$Q = kP := P + P + \cdots + P.$$

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 integer k

$$k = \sum_{i=0}^{j-1} k_i 2^i$$

where $k_i \in \{0,1\}$, and

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

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

$$k = \sum_{i=0}^{j-1} k_i 2^i$$

where $k_i \in \{-1,0,1\}$, 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**

$$k = \sum_{i=0}^{j-1} u_i 2^i$$

where u_i is odd and $|u_i| < |2^{w-1}|$,
and among any w consecutive coefficients, at most one is nonzero. The next algorithm computes the *width-w NAF* which is defined by $NAF_w(k) := (u_{i-1} \ldots u_1 u_0)$.

Algorithm 6: Computation of $NAF_w(k)$
Input: An integer k

```
Output: $NAF_w(k) := (u_{l-1} \ldots u_1 u_0)$.
$c \leftarrow k, \ \ l \leftarrow 0$.
while $c > 0$
        if c odd then
                $u_l \leftarrow 2 - (c \bmod 2^w)$
                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 $u_l \leftarrow 0$
        $c \leftarrow \frac{c}{2}, \quad l \leftarrow l + 1$
return $NAF_w(k) := (u_{l-1} \ldots u_1 u_0)$
```

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

```
Algorithm 7: The width-w window method
Input: Integers $k$ and $w$, and a point $P(x,y) \in E(F_q)$
Output: $Q = kP(x,y) \in E(F_q)$
$P_0 \leftarrow P, \quad T \leftarrow 2P$
for i = from 1 to $2^{w-2} - 1$.
$P_i \leftarrow P_{i-1} + T$
Compute: $NAF_w(k) := (u_{l-1} \ldots u_1 u_0)$
$Q \leftarrow O$
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. Numerical Results

For our experiments we use the Koblitz curves. These curves were first introduced by Koblitz [9]. The fasted method for computing kP on Koblitz curves was 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

$$(x^4,y^4) + 2P = \mu(x^2,y^2)$$

where $\mu = (-1)^{1-a}$. Using the Frobenius map $\tau(x,y) = (x^2,y^2)$, the preceding equation can be written as $\tau^2 P + 2P = \mu\tau P$. The map can be expressed as a complex number $\tau = (\mu + \sqrt{-7}$)/2 satisfying $\tau^2 + 2 = \mu\tau$. The idea is to replace the coefficient k with a τ-adic number $\bar{k} = \sum_{i=0}^{l-1} \bar{k}_i 2^i$ such that $k \equiv \bar{k}$ and to compute $\bar{k}P$ as $\bar{k}P = \bar{k}_{l-1}\tau^{l-1}P + \cdots + \bar{k}_0 P$.

In this case the calculation of kP is done using Frobenius map rather than point doubling to speed up the computations.

In our computations we use the polynomial basis representation of Koblitz curves and with the following NIST 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
$Cofactor\quad f$	2
$Curve\ order\ n$	fr
G_x	2 fe13c053 7bbc11ac aa07d793 de4e6d5e 5c94eee8
G_y	2 89070fb0 5d38ff58 321f2e80 0536d538 ccdaa3d9
$G\ Order\ r$	5846006549323611672814741753598448348329118574063

We use a 163-bit key and field elements that are represented by array of bytes $C[163]$ or as a Java BigInteger c. The array C is the two's complement binary representation of the BigInteger c. We compile the programs using Java Platform Micro Edition - Java ME Software Development Kit 3.0 which provides a complete micro application development environment for small mobile devices. We run the application or the executables on PocketPCs with and commercial Personal Java JVM Insignia Jeode.

In the experiments each operation is performed 1000 times where for each time different random values are used as input. The running time is then computed as the average. The results are shown in section 2 table 1.

In table 2 are the computation times for the various algorithms for computing the elliptic curve point multiplication. For each method the computation is performed 100 times and each time different random input values are used.

Table 2. The timings in milliseconds

Method	Time
Binary	163.31
Binary NAF	124.36
NAF_4 window	103.56
$\tau - adic$	47.17
$\tau - adic\ width - w(w = 4)$	38.19

Table 3. The timing for ECDSA algorithm in milliseconds

Method	F_{2^m}
Signature	68.71
Verification	331.43

In table 3 are the results for the elliptic curve digital signature algorithm (ECDSA) using elliptic curves in F_{2^m}. For more accurate results we compute the mean after running the program for 100 times and each time using different input values.

5. Conclusion

We provide an overview of elliptic curve cryptography and the running times of some resulting algorithms when implemented in Java on a PocketPCs. We show that elliptic curve algorithms can run in a suitable on small devices. Future work include the investigation and creation of a Java data type for handling ECC field elements so as to speed up computation time.

References

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

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

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

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

[5] N. Koblitz, ``Elliptic curve cryptosystem'', Mathematics of Computation, Vol. 48, 1987, pp. 203-209.

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

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

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

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

[10] R.J McEliece, ``Finite Fields for Computer Scientists and Engineers'', Kluwer Academic Publishers, 1987.

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

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

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

[14] V. Miller, ``Use of elliptic curves in cryptography''. Advances in Cryptography-Crypto '85, Springer-Verlag New York, LNCS 218, 1986, pp. 417-426.

[15] R. Rivest, A. Schamir, and Adleman, ``A method for obtaining digital signatures and public-key cryptosystems'', Communications of the ACM, Vol. 21, 1987, pp. 120-126.

[16] R. Schoof, ``Elliptic curves over finite fields and the computation of square roots mod p'', Mathematics of Computation, Vol. 44, 1985, pp. 483-494.

[17] J. Solinas, ``An improved algorithm for arithmetic on a family of elliptic curves'', Advances in Cryptology, Proc. Crypto '97, Springer-Verlag, LNCS 1294, 1997, pp. 357- 371.

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

Authors

Kossi Edoh received his B.Sc. degree in Mathematics from University of Cape Coast, Ghana, in 1986, and his M.Sc. and Ph.D. degrees from Simon Fraser University, Canada, in 1991 and 1995 respectively.

He has taught Computer Science at Elizabeth City State University, and Montclair State University in the US for eight years as an assistant professor. He is currently an associate professor of Mathematics at North Carolina Agricultural and Technical State University, USA. His research interests include information security and numerical dynamical systems.