Elliptic Curve Cryptography and Its Applications to Mobile Devices.

Wendy Chou,
University of Maryland, College Park.
Advisor: Dr. Lawrence Washington,
Department of Mathematics

Abstract: The explosive growth in the use of mobile and wireless devices demands a new generation of PKC schemes that has to accommodate limitations on power and bandwidth, at the same time, to provide an adequate level of security for such devices. This paper examines the use of ECC in such constrained environments and discusses the basis of its security, explores its performance and lastly, surveys the use of ECC applications on the market today.

1 Introduction

In 1976, Whitfield Diffie and Martin Hellman introduced the concept of public key cryptography (PKC). Since then, many implementations of it have been proposed, and many of these cryptographic applications base their security on the intractability of hard mathematical problems, namely the integer factorization problem (IFP) and the finite field discrete logarithm problem (DLP). Over the years, sub-exponential time algorithms were developed to solve these problems. As a result, key sizes grew to more than 1000 bits, so as to attain a reasonable level of security. In constrained environments where computing power, storage and bandwidth are limited, carrying out thousand-bit operations becomes an impractical approach to providing adequate security. This is most evident in hand-held devices such as the mobile phones, pagers and PDAs that have very limited processing power and battery life.

Proposed independently by Neal Koblitz and Victor Miller in 1985, elliptic curve cryptography (ECC) has the special characteristic that to date, the best known algorithm that solves it runs in full exponential time. Its security comes from the elliptic curve logarithm, which is the DLP in a group defined by points on an elliptic curve over a finite field. This results in a dramatic decrease in key size needed to achieve the same level of security offered in conventional PKC schemes.

This paper aims to examine two aspects of the ECC, namely its security and efficiency, so as to provide grounds as to why the ECC is most suitable for constrained environments. We begin by introducing the three mathematical problems and the various algorithms that solve them. An overview of implementation methods and considerations will be provided, followed by comparisons in the performance of ECC with other PKC applications. Lastly, there will be a survey of current ECC applications in various mobile devices.

1.1 The Need for Public Key Cryptography

Private key cryptography is widely used for the encryption of data due to its speed. The most commonly used today is the Data Encryption Standard (DES). It has an extremely fast encryption speed and this is a very attractive quality in terms of efficiency; however, it has certain shortcomings that make it unsuitable for use in the m-commerce environment.

I. Key Management Problem

A wireless user should be able to conduct business transactions with not just one party, but with many different ones. Thus, communication on a public network is not restricted to one-on-one, but a large number of users. For a network of n users, $n(n-1)/2$ private keys need to be generated. When n is large, the number of keys becomes unmanageable.

II. Key Distribution Problem

With such a large number of keys that needs to be generated on a network, the job of generating the keys and finding a secure channel to distribute them becomes a burden.

III. No digital signatures possible

A digital signature is an electronic analogue of a handwritten signature. If Alice sends an encrypted message to Bob, Bob should be able to verify that the received message is indeed from Alice. This can be done with Alice's signature; however, private key cryptography does not allow such a feature.

In contrast, public key cryptography uses two keys. Each user on a network publishes a public encryption key that anyone can use to send them messages, while keeping the private key secret for decryption. On a network of n users, it only needs n public and n private keys. This reduces the number of keys needed from $O(n^2)$ to $O(n)$. Furthermore, it allows the use of digital signatures, which ensures non-repudiation. However, public key cryptography does have its drawbacks. Compared to private key cryptography, public key cryptography is orders of magnitude slower. RSA needs at least 1024-bit keys while DES needs only 64 bits. In truth, public and private key cryptography work best together [2]. Public key cryptography is ideal for key distribution and management, ensuring *data integrity*, providing *authentication* and *non-repudiation*, while private key cryptography is ideal for ensuring *confidentiality*, such as encrypting data and communication channels. These are the four main objectives in any cryptographic application.

1.2 Choice of Public Key Cryptosystem

When it comes to choosing which public key cryptosystem to employ in a mobile environment, one has to keep in mind restrictions on bandwidth, memory and battery life. In constrained environments such as mobile phones, wireless pagers or PDAs, these resources are highly limited. Thus, a suitable public key scheme would be one that is efficient in terms of computing costs and key sizes.

To date, the ECC has the highest strength-per-bit compared to other public key cryptosystems. Small key sizes translate into savings in bandwidth, memory and processing power. This makes ECC the obvious choice in this situation. However, there are other aspects that need to be taken into account. In the following section, we will examine the different mathematical problems that underlie the majority of the public key cryptosystems in use today. We will also discuss some of the most efficient algorithms that solve them. This will give us a better understanding of the security on which different types of public key cryptosystems are based.

2. Security of Public Key Cryptosystems

As mentioned before, many of the public key cryptosystems base their security on the difficulty of solving a mathematical problem. Today, there are three problems that are believed to be both secure and practical after years of intensive studying. They are the 1) integer factorization problem, 2) finite field discrete logarithm problem and the 3) elliptic curve discrete logarithm problem. (Although there are other cryptographic systems that are lattice based, they will not be discussed in this paper.) While this by no means *proves* that they are unbreakable, it is highly unlikely that anyone will find an efficient algorithm to solve them in the near future.

The security of a cryptosystem depends on how hard it is to solve the underlying mathematical problem. The difficulty of a problem is determined by the asymptotic runtime of the algorithms that solve the problem.

> **Definition:** An algorithm with input size n is ***sub-exponential*** if there are constants $c > 0$ and $\alpha \in [0,1)$ such that the running time of the algorithm is in
> $$L_n[\alpha, c] = O(\exp((c+o(1))(\ln x)^{\alpha}(\ln \ln x)^{1-\alpha})) \qquad [3].$$

If $\alpha = 0$, then the algorithm runs in polynomial time. If $\alpha = 1$, then the algorithm is fully exponential.

2.1 Integer Factorization Problem (IFP)

2.1.1 Problem definition

The general integer factorization problem is defined as follows.

> Given a positive integer n, write $n = p_1^{e_1} p_2^{e_2} p_3^{e_3} \ldots p_k^{e_k}$ where the p_i are pairwise distinct primes and each $e_i \geq 1$ [4].

Typically, in practical cryptographic applications, only two factors are used for the modulus n. A larger number of factors for n does not seem to offer any additional security in the IFP.

The best-known public key cryptosystem that bases its security on the difficulty of the IFP is RSA. Named after its inventors: Ron Rivest, Adi Shamir and Len Adleman who developed it at MIT in 1978, it was the first practical implementation of public key cryptography since the introduction of the concept. Another example is the Rabin-Williams cryptosystem. It is similar to RSA, but it uses an even public exponent [4].

2.1.2 Factoring algorithms

The basic idea behind factoring involves finding two numbers x and y such that (1) $x^2 \equiv y^2 \pmod{n}$ and (2) $x \neq \pm y \pmod{n}$, where n is the number to be factored. Since (1) $\Leftrightarrow (x-y)(x+y) \equiv 0 \pmod{n}$, and (2) implies that n does not divide (x – y) and (x + y), then $\gcd(x-y, n)$ and $\gcd(x+y, n)$ must be nontrivial factors of n. Two of the most extensively used factoring algorithms today are the quadratic sieve and number field sieve. They are both based on the idea of finding a factor base of primes to generate a system of linear equations, whose solution will lead to equation (1) such that (2) holds.

The quadratic sieve (QS) is a general-purpose factoring algorithm because its runtime depends solely on the size of n. An improved variant of the QS, the multiple polynomial QS, achieves a runtime of

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

It gives a better chance of factoring, is well suited for parallel processing, and is the method of choice in practice [4].

Another example of a general-purpose factoring algorithm is the generalized number field sieve (NFS). Initially developed as a special purpose algorithm that factored integers of the form $n = r^e - s$ for small r and $|s|$, it was later extended to work also for general integers [4]. The generalized NFS is considered to be the fastest algorithm for factoring general numbers of at least 120 decimal digits. It achieves a runtime of $L_n[1/3,c]$, for a constant c[1], reducing the exponent in (I) [5]. Originally thought to be slower than the QS for factoring numbers less than 150 digits, recent experiments have shown that the general NFS is substantially faster than the QS, even for numbers in the 115-digit range [4]. Due to this, the generalized NFS is considered to be the most powerful of all general purpose factoring algorithms.

2.1.3 The RSA Challenge

In the past couple of decades, dramatic improvements have been made in the IFP. Below is a table charting the progress of the factorization of RSA numbers. We can see that the factoring of RSA-130 used a new and improved algorithm, which achieved the factorization at only 20% of the computing effort used to factor the smaller RSA-129.

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

Table 2-i (A 1-GHz Pentium is about a 250-MIPS machine.) [6]

A recent idea for attacking the IFP is a special machine called TWINKLE (The Weizmann Institute Key Locating Engine). Its inventor Shamir, who was one of the inventors of the RSA, proposed it in 1999. This sieving device can accelerate the sieving process in the NFS algorithm by two to three degrees of magnitude [7]. The matrix, however, still has to be solved on a conventional computer. The 140-digit RSA number that was broken recently took 200 conventional computers that ran in parallel for 4 weeks to complete the sieving. In contrast, it would only take 6 days on 7 TWINKLE machines. Since

[1] The value of c depends on which version of NFS is used. In the case of special NFS, $c = (32/9)^{1/3} \approx 1.526$, while the generalized NFS has $c = (64/9)^{1/3} \approx 1.923$.

the matrix solution still has to be performed on a conventional computer, this would add 4 more days to the factorization, bringing the total number of days to 10 [7]. The only fact that may bring relief is that TWINKLE, at present, is only a theoretical concept. However, it has been noted that the device is practical and can be manufactured at a cost of only US$5,000 [2] [8].

In light of these events, one might ask, "Is the IFP is no longer a hard problem?" The general consensus is that it is still a hard problem to solve. However, to achieve an adequate level of security, the size of keys must be increased to keep up with the frequent developments in the IFP. This security comes at a hefty price – the price of immense storage requirements, large bandwidth and powerful computing capability, particularly in constrained devices.

2.2 Discrete Logarithm Problem (DLP)

Unlike the IFP, where the size of the problem is the length of the modulus n that must be factored, the input size for the DLP is the number of points N in the group G that we are working with. In the case of the multiplicative group $G = Z^*_p$, where p is a large prime, N is equal to the size of the underlying finite field.

2.2.1 Problem definition

Let $G = Z^*_p$ be a multiplicative group of order $p - 1$, where the group operation is multiplication modulo p. The discrete logarithm problem is defined as follows.

> Given a prime p, a generator α of Z^*_p, and an element $\beta \in Z^*_p$, find the unique integer x, $0 \le x \le p - 2$, such that $\alpha^x = \beta \pmod{p}$ [4].

Cryptographic applications that base their security on the intractability of the DLP include the Diffie-Hellman key agreement scheme, the ElGamal encryption scheme and the digital signature algorithm (DSA).

2.2.2 Discrete logarithm algorithms

The most powerful algorithm known for computing the DLP is the index-calculus method. It is a probabilistic algorithm that applies only to finite fields. Examples of finite fields that are commonly used in practical applications are GF(p) and GF(2^m). The index-calculus method is currently the only known algorithm that solves the DLP in sub-exponential time, making it the champion of all DL algorithms. The index-calculus method works very similarly to NFS and QS of the IFP. It, too, requires the selection of a factor base S of small primes such that a significant portion of elements of G can be efficiently expressed as products of elements in S. It then builds a database of relations, which is used each time the logarithm of a group element is needed. As with the IFP algorithms, index-calculus algorithms can easily be parallelized.

DLP in a prime field is considered to be harder than DLP in fields of characteristic two. The current record for computing discrete logarithms in GF(p) is a 120-digit prime p [9]. This was accomplished by A. Joux et R. Lercier in 2001, using a variation of the index-calculus algorithm called the number field sieve. This algorithm has an expected running time of L_p[1/3,

[2] This is the cost of manufacturing subsequent models of TWINKLE, after building the initial prototype, which could cost hundreds of thousands of dollars.

1.923] [4]. In contrast, the current record for computing logarithms in GF(2^m) is GF(2^{607}). Completed in 2002 by E. Thomé , it was achieved by another variant of the index-calculus method, called the Coppersmith's algorithm [9]. This algorithm has a runtime of $L_2{}^m[1/3,c]$, for $c < 1.587$ [4].

Algorithms that solve the DLP for arbitrary groups do exist, however they all run in full exponential time. Examples include the Pollard ρ-method and the baby-step giant-step (BSGS). A downside of BSGS is that it is very memory intensive when the group order becomes large. The Pollard ρ-method, on the other hand, requires little memory and is easily parallelizable. Another algorithm that works for arbitrary groups, but is only efficient for those with orders that are made up of small primes, is the Pohlig-Hellman algorithm. Due to this, group orders are checked to make sure that the Pohlig-Hellman attack cannot be applied. For arbitrary groups, this algorithm also runs in full exponential time.

2.2.3 QS and NFS in DLP and IFP

Both the QS and NFS factoring algorithms are very alike in approach to the index-calculus methods of the DLP. "The two problems are very similar, and all of the modern factoring algorithms can be used to calculate discrete logarithms in the multiplicative group of a finite field" [10]. From this point of view, the amount of work done in computing the logarithm in Z^*_p, where p has k-bits, can be considered to be equal to the amount to work needed to factor a k-bit composite number n. Therefore, if any improvements in algorithms are discovered for either one of the problems, then improvements can also be expected for the other problem [11].

So far, both of the mathematical problems discussed have sub-exponential algorithms that solve them. In the following section, a different type of problem called the elliptic curve discrete logarithm problem will be presented. To date, the best algorithm that computes elliptic curve logarithms runs in full exponential time.

2.3 Elliptic Curve Discrete Logarithm Problem (ECDLP)

2.3.1 Problem definition

Let E be an elliptic curve over some finite field GF(p) and G = $E(F_q)$ be a cyclic additive group, where the group operation is addition modulo p. The elliptic curve discrete logarithm problem is defined as follows.

> Given P $\in$ G and an element Q $\in$ <P>, find the integer m, such that Q = $[m]$P [12].

In 1985, Neal Koblitz and Victor Miller independently proposed the concept of elliptic curve cryptography (ECC). It is based on the DLP in a group defined by points on an elliptic curve over a finite field. Implementations of ECC include elliptic curve analogs of DSA (ECDSA), ElGamal and Diffie-Hellman.

2.3.2 ECDLP algorithms

The most attractive feature of ECC is that at present, the fastest known algorithm that solves it run in full exponential time. Despite the fact that index-calculus methods can compute conventional logarithms in sub-exponential time, they cannot be applied to the case of discrete logarithms over elliptic curves. This is a claim made by Miller in his 1986 paper, which was later backed by theoretical study and computational experiments by J. H. Silverman and Suzuki in their paper published in 1998 [3].

To date, the best-known general-purpose algorithm for solving the ECDLP is the Pollard ρ-method. It can be sped up with special techniques while running on parallel processors to $O(\sqrt{(\pi n)}/(2r))$, where n is the number of elliptic curve additions and r is the number of processors running [11]. When the orders of the curves become sufficiently large, however, methods like Pollard ρ and BSGS become infeasible [12]. In 1997, V. Shoup showed that the running time of any algorithm that solves the DLP for arbitrary groups takes $\Omega(\sqrt{p} \log p)$ steps, where p is the largest prime factor of N [3]. Hence, to significantly improve the efficiency of general purpose algorithms might prove to be a futile task.

2.3.3 Weak curves

There are certain types of elliptic curves in which a successful attack could take place in sub-exponential time. If identified, these curves can easily be tested for and avoided. So far, several classes of curves have been identified and prohibited in all drafted standard specifications for public key cryptography, such as IEEE P1363, ANSI X9.62 and ANSI X9.63 [11]. Such curves are called the supersingular curves and anomalous curves.

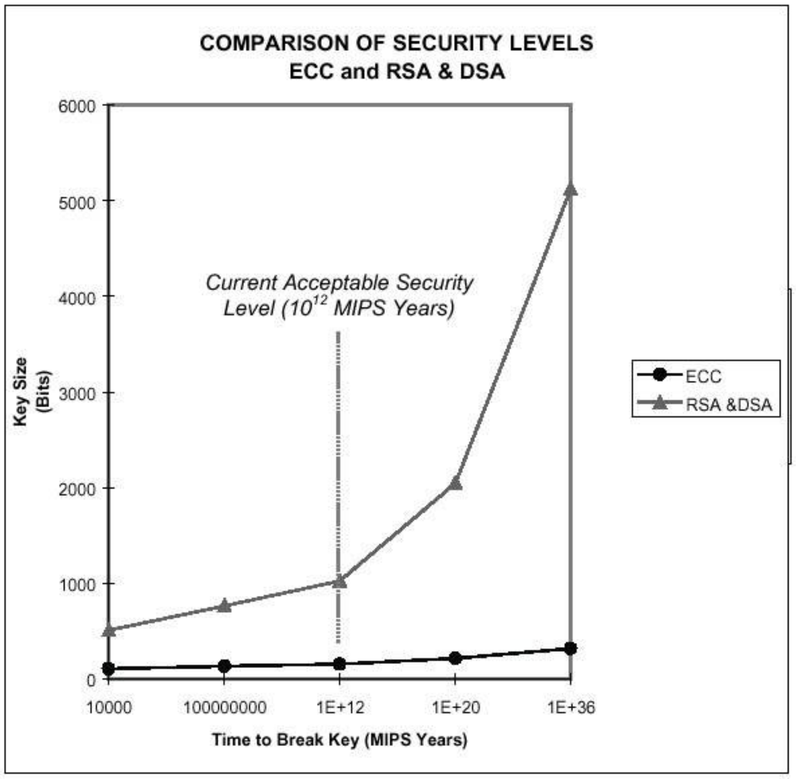
Supersingular curves are a special class of elliptic curves on which the elliptic curve logarithm can be reduced to the case of discrete logarithms in a multiplicative group (classical DLP). When combined with sub-exponential algorithms for solving the classical DLP, this yields a probabilistic sub-exponential running time for computing elliptic curve logarithms on supersingular curves. This was a finding due to Menezes, Okamoto and Vanstone (MOV) in 1991, in which they showed how the ECDLP could be reduced to classical DLP in an extension of a multiplicative group GF(p)[11]. For further reading on the MOV reduction algorithm and proofs, refer to [13].

The other class of curves, the anomalous curves, allows an even more efficient attack when applicable. Proposed independently in 1998 by Satoh and Araki, Semaev, and the following year by Smart, this type of curves allow the ECDLP to be solved in polynomial time by reducing it to the classical DLP in an additive group GF(p) [3]. Further readings can be found in [14], [15] and [16].

2.3.4 The ECC Advantage

Much like the RSA challenge, the Certicom ECC challenge offers prize money for finding various key sizes of the ECDLP. The current record was set in November 2002 where a 109-bit encryption key was broken with 10,000 computers running 24 hours a day for 549 days [17]. The Certicom ECC challenge website reports that breaking a 163-bit key, which is the standard applied to most commercial ECC applications that Certicom uses, would be a hundred million times harder than breaking the 109-bit key. It is worthy to note that a 160-bit ECC key has about the same level of security as a 1024-bit RSA key.

The most important difference between ECC and other conventional cryptosystems is that for a well-chosen curve, the best method currently known for solving the ECDLP is fully exponential, while sub-exponential algorithms exist for conventional cryptosystems. This difference largely contributes to the huge disparity in their respective running times. It also means that ECC keys have much fewer bits than IFP and DLP based applications. The contrast in key lengths of RSA, DSA and ECC are shown in the graph (Graph 3-i) below. Clearly, ECC keys take much more effort to break compared to RSA and DSA keys. Due to this, many people believe that ECDLP is intrinsically harder than the other two problems. While this deduction might be true, we have no way of *proving* it. We do not know if a fast and efficient elliptic curve DL algorithm that runs in sub-exponential time will be discovered, say, in the next ten years, or if another class of weak curves will be identified that could compromise the security of elliptic curve cryptosystems. But one thing is certain. After years of intensive study, there is currently no faster way to attack the ECDLP other than fully exponential algorithms.



Graph 3-i. [4]

3. Implementation of Elliptic Curve Cryptosystems

There are some important issues that need to be addressed before implementing the elliptic curve cryptosystem. We need to decide whether to use an even or odd characteristic field, and also how to represent the points on the elliptic curve. These choices will not only determine how we implement field arithmetic on elliptic curves, but they will also affect the efficiency of the computations. In section 3.1, we will go over two types of characteristics of fields, including their underlying representations and some field arithmetic operations associated each type of basis representations. Section 3.2 discusses the different representations of points on the elliptic curve. Section 3.3 shows how to set up the ECC and its parameters, and the last section discusses the NIST recommended curves and guidelines.

3.1 Even and Odd Characteristic Fields

There are two types of characteristics of fields, namely even and odd. The prime field GF(p), where p is a large prime, is of odd characteristic. This field has p elements that are represented by integers modulo p. Field arithmetic on GF(p) is implemented in terms of the arithmetic of integers modulo p. The field GF(2^m) is of even characteristic, specifically, of characteristic 2. There are 2^m elements in this field, and they are represented as m-dimensional binary vectors over F_2, i.e. they are bit-strings of length m. Field addition and subtraction are implemented as component-wise XOR, while implementations of multiplication and inversion (division) depend on the choice of basis.

Arithmetic in a prime field is simple; it is just the arithmetic of integers modulo p. For a binary field, the field elements are represented relative to a given basis. There are many choices for a basis. A *polynomial basis* has the form

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

where t is a root of an irreducible polynomial $p(t)$ over F_2. An irreducible polynomial is one that cannot be factored as a product of polynomials of lower degree modulo 2. An element of GF(2^m) $\ni (a_0, a_1, \dots , a_{m-1})$, where $a_i \in \{0,1\}$, w.r.t. a polynomial basis is represented by the polynomial

$$a_0 + a_1t + a_2t^2 \dots + a_{m-1}t^{m-1} \bmod p(t),$$

where $p(t)$ is an irreducible polynomial over F_2. Field arithmetic is performed as polynomial arithmetic modulo $p(t)$ [18].

A *normal basis* is of the form

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

where $\beta \in$ GF(2^m). Elements of GF(2^m) $\ni (a_0, a_1, \dots , a_{m-1})$ w.r.t. a normal basis can be written as

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

where $a_i \in \{0,1\}$. Addition of field elements in normal basis representation is simply bitwise XOR-ing of the vector elements. Squaring can be achieved by a rotation of the vector elements. This is a cheap operation to perform, thus the cost of squaring is often ignored in analyzing runtime complexities. Multiplication is more complicated, but, with optimization, it comes down to a series of m cyclic shifts of the two vector multiplicands. Inversion (division) is the most complex and expensive operation to perform. An example of an inversion algorithm is one proposed by Itoh, Teechai and Tsujii, which can be found at [24]. Here, division is a recursive algorithm that requires $I(m) = \lfloor \log_2(m-1) \rfloor + \omega(m-1) - 1$ field multiplications, where $\omega(m-1)$ is the number of 1's in the binary representation of $m-1$[1].

There are a couple of reasons why fields of characteristic 2 would be preferred over odd characteristic fields. First of all, the field elements in GF(2^m) are represented as bit-strings of length m. In terms of hardware, bit-string representation of integers provides greater ease of implementation than the natural representation of integers. Furthermore, with normal basis representation in GF(2^m), squaring can be achieved by a simple rotation of the vector elements, which means that the cost of squaring is negligible [1]. Another important factor is that normal bases allow for the design of efficient bit-serial multipliers. An example is one that was proposed by Massey and Omura [19].

3.2 Point Representation

In this section, two types of point representation will be discussed – affine and projective coordinates. We will use formulae from point addition in a prime field to illustrate the different costs in performing point arithmetic using the two representations. Refer to Appendix A for formulae on point addition in fields of characteristics 2 and p > 3, using both affine and projective representations.

Affine coordinates (x, y) satisfy the affine equation

$$(i) \qquad E: y^2 = x^3 + ax + b,$$

where $a, b \in GF(p)$. Referring to formula (A-1) in Appendix A, addition of affine coordinates requires 1 inversion, 2 multiplications and 1 squaring (ignoring the cost of field additions and subtractions), when $P_1 \neq P_2$,. While point doubling ($P_1 = P_2$) requires 1 inversion, 2 multiplications and 2 squarings.

Conventional *projective* coordinates (x, y, z) satisfy the homogenous Weierstrass equation

$$E: y^2z = x^3 + axz^2 + bz^3,$$

where $a, b \in F_p$. When $z \neq 0$, the projective point (x, y, z) corresponds to the affine point $(x/z, y/z)$. Projective coordinates are used when field inversions are significantly more expensive than field multiplications. With projective coordinates, the need for performing inversions is replaced with multiplication, thus projective addition can be achieve through only the use of field multiplications. There are other types of projective representations that are more efficient than the convention projective representation. In particular, the *weighted projective* representation (or *Jacobian* representation) results in a more efficient implementation of group operations [12]. Jacobain coordinates (x, y, z) correspond to the affine coordinates $(x/z^2, y/z^3)$, and they satisfy the weighted projective curve equation

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

Referring to formula (A-2) in Appendix A, addition of projective coordinates requires 16 multiplications, while doubling requires 10 multiplications for arbitrary a, and if $a = -3$, doubling requires only 8 multiplications. Evidently, the cost of eliminating inversions is an increased number of multiplications. Thus, the need for using projective coordinates is strongly determined by the ratio $I{:}M$, where I is the number of inversions and M is the number of multiplications [12].

The costs of point addition using both representations are summarized in the table below. Refer to Appendix A for a similar table on point addition over fields of characteristic 2.

Cost of point addition, characteristic p > 3.

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

Table 4-i. (I = inversion, M = multiplication) [12]

3.3 Curve Selection and Setup

The preferred method for generating good curves is to select the curves at random. Randomly generated curves are curves with coefficients that are taken from the output of pseudo-random number generators. Below is an example of an algorithm that selects an appropriate curve over F_p [12]:

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

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

The hardest part of this algorithm is Step 2. For this reason, sometimes E is not chosen at random. A class of curves, known as the Koblitz curves, is particularly favorable because it was shown to be very efficient in computing *ord*(P) for arbitrary P on the curve, which can in turn be used to derive $\#E(F_p)$ quickly. There are various algorithms proposed to compute group orders, readers can refer to Chapter 7 in [12] for a more in depth treatment of this subject. The purpose of Step 5 is to make sure that $\#E(F_p)$ has a large prime factor. This is to guard against a Pohlig-Hellman attack on the 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 $0 < k < q$, $q = ord(P)$.

Table 4-ii. Setting up an elliptic curve cryptosystem.

Above is a table listing the basic computations necessary to set up an elliptic curve cryptosystem. Key pairs are easy to generate; the private key is a randomly chosen integer k, such that the public key P = kG. The basic assumption of ECC is that it is hard to compute the secret key k from the public key P. The elliptic curve parameters can be used for groups of users, where each user in the group has a public and private key pair. Another alternative, which offers more security but additional computation costs, is to use a different curve in the same underlying field for each user. This way, all users require the same hardware implementation for field arithmetic, but the curve can be changed periodically for additional security [1].

3.4 NIST Recommended Fields and Curves

The Federal Information Processing Standards (FIPS) is compilation of standards and guidelines issued by NIST for government use. The revised FIPS 186-2 includes the elliptic curve digital signature algorithm (ECDSA), with recommendations on the selection of finite fields and elliptic curves. These recommended curves have special properties that allow for optimized performance. They are also checked to ensure that none of them belong to the class of supersingular and anomalous curves, which are susceptible to MOV and other known attacks.

FIPS 186-2 recommends a total of 15 elliptic curves over 10 finite fields. For each of the five prime fields and five binary fields, a pseudo-random curve is generated using the SHA-1 method specified in ANSI X9.62 and IEEE P1363 standards [18, p27]. In addition, a Koblitz curve is selected for each of the five binary fields, adding up a total of 15 curves.

The following considerations were made when choosing the finite fields and elliptic curves.

I. Choice of Key Lengths

All curves are chosen to have cofactors 1, 2 or 4. This is done to ensure efficiency in computation. As a result, private and public keys have approximately the same length in bits [18].

II. Choice of Fields

Each field is chosen such that the length of the order in bits is at least twice the key length of common private-key (symmetric-key) block ciphers. This is done because an exhaustive key search of a k-bit block cipher is expected to take about the same time as the solution of an ECDLP, when using the Pollard's ρ algorithm for a suitable elliptic curve over a finite field with an order of length $2k$ [20]. Table 3-i below compares private-key lengths with sizes of various fields.

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

Table 3-i. [20]

III. Choice of p in GF(p) and m in GF(2^m)

For binary fields GF(2^m), m was chosen so that there exists a Koblitz curve of almost prime order over GF(2^m). For prime fields GF(p), p was chosen to be either a Mersenne prime, or a Mersenne-like prime with bitsize being a multiple of 32 [20]. A Mersenne prime is a prime of the form 2^n-1, where n is a prime. Prime fields with p being a Mersenne prime allows for efficient modular reduction. See Appendix B for examples from FIPS 186-2.

IV. Coefficients of curves over prime fields

All selected curves over a prime finite field satisfy the equation $y^2 = x^3 + \mathrm{a}x + \mathrm{b}$, where $\mathrm{a} = -3$. This allows for efficient point doubling when using Jacobian coordinates. For details, refer to IEEE P1363 [20].

4. Performance Analysis

Now, we will turn our attention to the ECC in practical use. In this section, software implementations of the DSA and RSA digital signature schemes will be compared to ECDSA, and the ElGamal encryption scheme will be compared to its elliptic curve counterpart. Experiments were performed on both PCs and mobile devices. Data was collected from various studies conducted by research institutes and individual experiments. All timings are in milliseconds unless stated otherwise. Finite fields recommended by NIST are in italics.

4.1 Digital Signature Schemes

A digital signature is the electronic equivalent of a handwritten signature. When attached to an electronic document, it provides authentication of the signer, date and time of signature and contents of the signed document. Furthermore, the signature must be verifiable to ensure that the signer cannot deny signing the document afterwards. Therefore, a digital signature algorithm needs to be able to generate keys for the signature, sign a document and verify the signature.

PCs

Platform: Pentium Pro 200 MHz using C, C++ and assembly
Curves: Random curves over GF(2^{191}) and $F_{p=192}$
(Note that $F_{p=192}$, where p = 192 is a Mersenne-like prime, is one of the NIST recommended finite fields, and its timing is almost half of curves over GF(2^{191}), which is not in FIPS 186-2.[3])

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

Table 5-i. [21]

Platform: Pentium II 400 MHz using C

[3] Refer to Appendix B on examples of using Mersenne-like primes to perform fast modular multiplication.

Curves: Koblitz and random curves over NIST curves *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

Tables 5-ii. [21]

Results from Tables 5-ii clearly show that Koblitz curves produce more efficient computation speeds compared to RSA and DSA, and together with the NIST recommended parameters provided in FIPS 186-2, it proves to be a far superior digital signature scheme in terms of efficiency.

<u>PDAs</u>

Platform: (PalmPilot) Motorola Dragon Ball 15 MHz using C
Curves: Koblitz curves over *GF(2^{163})*
(Keep in mind that the input size of ECDSA- *GF(2^{163})* is almost twice of that in RSA-512.)

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

Table 5-iii. [21]

<u>On Pagers</u>

Platform: (Pager) RIM 10 MHz using C
Curves: Koblitz and random curves over NIST curves *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	-

Tables 5-iv. [21]

From Tables 5-iii and 5-iv, it is clear that the verification process for ECDSA is slower than that of RSA, but the total timing for all three procedures (key generation, signature and verification) of the ECDSA is much faster than RSA and DSA. For an ECDSA key size of 163 over Koblitz curves, it takes a total of 3,588 ms to complete all three procedures, while it takes RSA 597,302 ms, for a key size of 1024. As for DSA, the verification process alone is longer than the total time required for ECDSA.

Comparisons. It is clear from the results of Tables 5-i – 5-iv that NIST recommended finite fields and Koblitz curves offer superior performance compared to random curves and fields not in FIPS186-2. On PCs, ECDSA outperforms RSA and DSA significantly in overall timing. Although its verification process is slightly slower, a 2-3 ms difference is negligible on the PC. On the PalmPilot, using a 1024-bit RSA key was unpractical (since only small amounts of money of about \$10 were involved in the experimental transactions), thus a 512-bit key was chosen instead. Compared to the 163-bit ECDSA key, ECDSA showed better overall performance. The verification process, however, was much slower than the 512-bit RSA. To offset this imbalance, a combination of RSA and ECDSA protocols could be used. For details, refer to [22]. On pagers, the overall timing for ECDSA is much better than RSA and DSA. In conclusion, ECDSA is clearly the most suitable PKC scheme in constrained environments.

4.2 Encryption Schemes

In this section, the ElGamal encryption and decryption algorithms will be compared to its elliptic curve version. Comparisons will be made on cryptosystems with the same level of security. Thus, a 768-bit conventional ElGamal should be compared to a 151-bit ECC ElGamal, while a 1024-bit conventional ElGamal should be compared to a 173-bit ECC-ElGamal. However, for key sizes of 151 and 173 bits on the ECC ElGamal, there does not exist trinomial in polynomial bases (PB) nor optimized normal basis in normal bases (NB) [23], hence 155 and 183-bit key sizes will be used instead. Note, however, that there is slight improvement in security levels in the ECC versions of ElGamal.

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

Tables 5-v. [23]

Comparisons. Elliptic curve operations in NB runs approximately 22% faster than PB for encryption, while decryption in NB is approximately 15% faster than PB. Overall, the performance of EC ElGamal is much better than conventional ElGamal; the improvement in overall speed is over 50% [23].

5. A Survey of Current ECC Applications

When the ECC was first introduced in 1985, there was a lot of skepticism about its security. However, ECC has since come a long way. After nearly a decade of serious study and scrutiny, ECC has yielded highly efficient and secure. Presently, many product vendors have incorporated ECC in their products, and this number has only been on the rise. Uncertainty still exists among some proponents of traditional cryptographic systems, but they are starting to become more accepting of this promising new technology. RSA Security Inc., for example, has long voiced concern regarding the security of ECC since its introduction. In recent years, however, RSA Security has researched on efficient ECC algorithms, and even acquired a patent on a storage-efficient basis conversion algorithm. Moreover, it has also integrated ECC into some of its products, acknowledging the fact that ECC has begun to establish itself as both secure and efficient.

An important factor for this emerging trend is the incorporation of ECDSA in several government and major research institution security standards, including IEEE P1363, ANSI X9.62, ISO 11770-3 and ANSI X9.63. Another factor is the strong promotion of the use of ECC through a Canadian-based Certicom Corporation. Certicom is a company that specializes in information security solutions in a mobile computing environment through providing software and services to its clients. Over the years, Certicom has published numerous papers in support of ECC and has also implemented ECC in all of its commercial products. Its success prompted many other companies to look more closely at the benefits and security of ECC. Now, ECC is becoming the mainstream cryptographic scheme in all mobile and wireless devices.

Below is a short survey of ECC applications seen on the market today. Results of the survey can be broadly divided into four categories: the Internet, smart cards, PDAs and PCs.

<u>Internet</u>

- In September of 2002, SUN Microsystems contributed to the implementation of an ECC cryptographic library and also a common hardware architecture for accelerating ECC (as well as RSA) to be used in openSSL. OpenSSL is a developmental toolkit for the implementation of SSL (Secure Sockets Layer) and TLS (Transport Layer Security) protocols, which are commonly used today in over-the-web transactions and secure document transfers. SUN hopes to promote ECC standardization with SSL, which is the dominant security protocol used on the web today.

- In late 1998, the Treasury Department's Bureau of Engraving and Printing completed a four-month e-commerce pilot program involving the use of smart cards and ECC with SET (Secure Electronic Transaction) specifications. SET is a standard that enables secure credit card transactions over the Internet. The pilot program tested the use of smart cards, embedded with ECC technology, in making online purchases. This program involved a total of nine companies, including MasterCard, Certicom (who supplied the ECC algorithms), Digital Signature Trust Co. (who supplied the MasterCard smart cards) and GlobeSet (a SET vendor), just to name a few. The previous version of SET, version 1.0, supports only RSA Data Security encryption algorithms, but MasterCard hopes to add ECC to the upcoming version of SET.

Smart Cards

Smart cards are one of the most popular devices for the use of ECC. Many manufacturing companies are producing smart cards that make use of elliptic curve digital signature algorithms. These manufacturing companies include Phillips, Fujitsu, MIPS Technologies and DataKey, while vendors that sell these smart cards include Funge Wireless and Entrust Technologies. Smart cards are very flexible tools and can be used in many situations. For example, smart cards are being used as bank (credit/debit) cards, electronic tickets and personal identification (or registration) cards.

PDAs

PDAs are considered to be a very popular choice for implementing public key cryptosystems because they have more computing power compared to most of the other mobile devices, like cell phones or pagers. However, they still suffer from limited bandwidth and this makes them an ideal choice for using ECC. In the January of 1998, 3Com[4] Corporation teamed up with Certicom to implement ECC in future versions of its PalmPilot organizer series and Palm Computing platform. This new feature will provide protection of confidential information on the hand-held organizers, user authentication in wireless communications and e-commerce transactions, and also ensure data integrity and proof of transactions.

PCs

- Constrained devices have been considered to be the most suitable platforms for implementing the ECC. Recently, several companies have created software products that can be used on PCs to secure data, encrypt e-mail messages and even instant messages with the use of ECC. PC Guardian Technologies is one such company that created the Encryption Plus Hard Disk and Encryption Plus Email software products. The former makes use of both RSA and EC Diffie-Hellman while the latter makes use of a strong 233-bit ECC key to encrypt its private AES keys.

[4] Since the 28 July 2000, Palm Inc. has separated from 3Com, and is now a fully independent company.

- The Top Secret Messenger software was developed by Encryption Software Inc. It encrypts the messages of some of the most popular instant messaging programs today, like ICQ and MSN. It can also be used with e-mail clients such as Microsoft Outlook and Outlook Express to encrypt e-mail messages. This product uses both private and public key cryptosystems, including a 307-bit key for its implementation of the ECC.

6. Conclusion

After examining the security, implementation and performance of ECC applications on various mobile devices, we can conclude that ECC is the most suitable PKC scheme for use in a constrained environment. Its efficiency and security makes it an attractive alternative to conventional cryptosystems, like RSA and DSA, not just in constrained devices, but also on powerful computers. It is, without a doubt, fast being recognized as a powerful cryptographic scheme.

Appendix A – Formulae for Point Addition

Fields of characteristic p > 3.
Affine Coordinates.
Formula (A-1). Let the elliptic curve E be defined as

$$E\colon y^2 = x^3 + ax + b$$

with $a,b \in \mathrm{F}_q$, $q = p^n$, p a prime > 3. Let $\mathrm{P}_1 = (x_1,y_1)$ and $\mathrm{P}_2 = (x_2,y_2)$ be points in $E(\mathrm{F}_q)$ in affine coordinates. Assume that $\mathrm{P}_1, \mathrm{P}_2 \neq O$ and $\mathrm{P}_1 \neq -\mathrm{P}_2$, then the sum $\mathrm{P}_1 + \mathrm{P}_2 = \mathrm{P}_3 = (x_3,y_3)$, where
if $\mathrm{P}_1 \neq \mathrm{P}_2$

$$\lambda = (y_2 - y_1) / (x_2 - x_1)\,,$$
$$x_3 = \lambda^2 - x_1 - x_2\,,$$
$$y_3 = \lambda(x_1 - x_3) - x_3 - y_1.$$

if $\mathrm{P}_1 = \mathrm{P}_2$

$$\lambda = (\mathrm{x}^3{}_1 + a) / 2y_1\,,$$
$$x_3 = \lambda^2 - 2x_1,$$
$$y_3 = \lambda(x_1 - x_3) - x_3 - y_1.$$

Projective Coordinates.
Formula (A-2). Let the elliptic curve E be defined as

$$E\colon y^2 = x^3 + axz + bz^6.$$

Let $\mathrm{P}_1 = (x_1,y_1,z_1)$ and $\mathrm{P}_2 = (x_2,y_2,z_2)$ be points in $E(\mathrm{F}_q)$ in weighted projective coordinates. Assume that $\mathrm{P}_1, \mathrm{P}_2 \neq O$ and $\mathrm{P}_1 \neq -\mathrm{P}_2$, then the sum $\mathrm{P}_1 + \mathrm{P}_2 = \mathrm{P}_3 = (x_3,y_3,z_3)$, where
if $\mathrm{P}_1 \neq \mathrm{P}_2$

$$\lambda_1 = x_1 z_2^2\,,$$
$$\lambda_2 = x_2 z_1^2\,,$$
$$\lambda_3 = \lambda_1 - \lambda_2\,,$$
$$\lambda_4 = y_1 z_2^3\,,$$
$$\lambda_5 = y_2 z_1^3\,,$$
$$\lambda_6 = \lambda_4 - \lambda_5\,,$$
$$\lambda_7 = \lambda_1 + \lambda_2\,,$$
$$\lambda_8 = \lambda_4 + \lambda_5\,,$$
$$z_3 = \lambda_3 z_1 z_2\,,$$
$$x_3 = \lambda_6^2 - \lambda_7 \lambda_3^2\,,$$
$$\lambda_9 = \lambda_7 \lambda_3^2 - 2x_3\,,$$
$$y_3 = (\lambda_9 \lambda_6 - \lambda_8 \lambda_3^3) / 2.$$

if $\mathrm{P}_1 = \mathrm{P}_2$

$$\lambda_1 = 3x_1^2 + az_1^4\,,$$
$$\lambda_2 = 4x_1 y_1^2\,,$$
$$\lambda_3 = 8y_1^4,$$
$$z_3 = 2y_1 z_1\,,$$
$$x_3 = \lambda_1^2 - 2\lambda_2\,,$$
$$y_3 = \lambda_1(\lambda_2 - x_3) - \lambda_3\,.$$

Fields of characteristic 2.
Affine Coordinates.
Formula (A-3). Let the elliptic curve E be defined as

$$E\colon y^2 + xy = x^3 + a_2 x^2 + a_6\,,$$

with $a_2 + a_6 \in \mathrm{F}_q$, $q = 2^n$ and $a_6 \neq 0$. Let $\mathrm{P}_1 = (x_1,y_1)$ and $\mathrm{P}_2 = (x_2,y_2)$ be points in $E(\mathrm{F}_q)$ in affine coordinates. Assume that $\mathrm{P}_1, \mathrm{P}_2 \neq O$ and $\mathrm{P}_1 \neq -\mathrm{P}_2$, then the sum $\mathrm{P}_1 + \mathrm{P}_2 = \mathrm{P}_3 = (x_3,y_3)$, where
if $\mathrm{P}_1 \neq \mathrm{P}_2$

$$\lambda = (y_2 + y_1) / (x_2 + x_1)\,,$$

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a_2 ,$$
$$y_3 = \lambda(x_1 + x_3) + x_3 + y_1.$$

if $P_1 = P_2$

$$\lambda = y_1 / x_1 + x_1 ,$$
$$x_3 = \lambda^2 + \lambda + a_2 ,$$
$$y_3 = \lambda(x_1 + x_3) + x_3 + y_1.$$

Projective Coordinates.
Formula (A-4). Let the elliptic curve E be defined as

$$E: y^2 + xyz = x^3 + a_2x^2z^2 + a_6z^6.$$

Let $P_1 = (x_1,y_1,z_1)$ and $P_2 = (x_2,y_2,z_2)$ be points in $E(F_q)$ in weighted projective coordinates. Assume that $P_1, P_2 \neq O$ and $P_1 \neq -P_2$, then the sum $P_1 + P_2 = P_3 = (x_3,y_3,z_3)$, where
if $P_1 \neq P_2$

$$\lambda_1 = x_1z_2^2 ,$$
$$\lambda_2 = x_2z_1^2 ,$$
$$\lambda_3 = \lambda_1 + \lambda_2 ,$$
$$\lambda_4 = y_1z_2^3 ,$$
$$\lambda_5 = y_2z_1^3 ,$$
$$\lambda_6 = \lambda_4 + \lambda_5 ,$$
$$\lambda_7 = z_1\lambda_3 ,$$
$$\lambda_8 = \lambda_6x_2 + \lambda_7y_2 ,$$
$$z_3 = \lambda_7z_2 ,$$
$$\lambda_9 = \lambda_6 + z_3 ,$$
$$x_3 = a_2z_3^2 + \lambda_6\lambda_9 + \lambda_3^3 ,$$
$$y_3 = \lambda_9x_3 + \lambda_8\lambda_7^2.$$

if $P_1 = P_2$

$$z_3 = x_1z_1^2 ,$$
$$x_3 = (x_1 + a_6z_1^2)^4 ,$$
$$\lambda = z_3 + x_1^2 + y_1z_1 ,$$
$$y_3 = x_1^4z_3 + \lambda x_3 .$$

Cost of Point Addition, characteristic 2.

Operation	Coordinates	
	affine	projective
General addition ($a_2 \neq 0$)	$1I + 2M + 1S$	$15M + 5S$
Doubling ($a_2 \neq 0$)	$1I + 2M + 1S$	$14M + 4S$
Doubling	$1I + 2M + 1S$	$5M + 5S$

Table A-i. (I = inversion, M = multiplication, S = squaring)[12]

Appendix B – Efficient Modular Multiplication using Mersenne Primes

Mersenne primes allow for efficient computation of modular arithmetic. Suppose we need to compute

$$B \equiv A \bmod m,$$

given that $A < m^2$. If m is a generalized Mersenne number, B can be expressed by sums or differences (mod m) of a small number of terms.

(Examples shown below are taken from Appendix 6.1 of FIPS 186-2.)

Example 1

Let Curve P-192 be the NIST recommended curve over a prime field where p = 192. The modulus for this curve can be expressed as $2^{192} – 2^{64} – 1$. Every integer A less than p^2 can thus be written as

$$A = A_5 \times 2^{320} + A_4 \times 2^{256} + A_3 \times 2^{192} + A_2 \times 2^{128} + A_1 \times 2^{64} + A_0,$$

where each A_i is a 64-bit integer. To compute B, evaluate

$$B := T + S_1 + S_2 + S_3 \bmod p;$$

where each 192-bit terms are given by

$$\begin{aligned} T &= A_2 \times 2^{128} + A_1 \times 2^{64} + A_0 \\ S_1 &= A_3 \times 2^{64} + A_3 \\ S_2 &= A_4 \times 2^{128} + A_4 \times 2^{64} \\ S_3 &= A_5 \times 2^{128} + A_5 \times 2^{64} + A_5. \end{aligned}$$

Example 2

Let Curve P-224 be the NIST recommended curve over a prime field where p = 224. The modulus for this curve can be expressed as

$$A = A_{13} \times 2^{416} + A_{12} \times 2^{384} + A_{11} \times 2^{352} + A_{10} \times 2^{320} + A_9 \times 2^{288} + A_8 \times 2^{256} + A_7 \times 2^{224} + A_6 \times 2^{192} + A_5 \times 2^{160} + A_4 \times 2^{128} + A_3 \times 2^{96} + A_2 \times 2^{64} + A_1 \times 2^{32} + A_0,$$

where each A_i is a 32-bit integer. As a concatenation of 32-bit words, this can be denoted by

$$A = (A_{13} \| \ldots \| A_0).$$

To compute B, evaluate

$$B := T + S_1 + S_2 - D_1 - D_2 \bmod p,$$

where the 224-bit terms are given by

$$\begin{aligned} T &= (A_6 \| A_5 \| A_4 \| A_3 \| A_2 \| A_1 \| A_0) \\ S_1 &= (A_{10} \| A_9 \| A_8 \| A_7 \| 0 \| 0 \| 0) \\ S_2 &= (0 \| A_{13} \| A_{12} \| A_{11} \| 0 \| 0 \| 0) \\ D_1 &= (A_{13} \| A_{12} \| A_{11} \| A_{10} \| A_9 \| A_8 \| A_7) \\ D_2 &= (0 \| 0 \| 0 \| 0 \| A_{13} \| A_{12} \| A_{11}) \end{aligned}$$

References

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., and 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 E-Newsletter. November 15, 1999 <http://www.counterpane.com/crypto-gram-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., and Smart, N. *Elliptic Curves in Cryptography*. Cambridge University Press, 1999.
13. Menezes, A., Okamoto, T., and 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. and 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. and 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., and 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. and 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. and 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., and 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., 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.