On the Use of Weber Polynomials in Elliptic Curve Cryptography*

Elisavet Konstantinou[1,2], Yannis C. Stamatiou[1,3,4], and Christos Zaroliagis[1,2]

[1] Computer Technology Institute, P.O. Box 1122, 26110 Patras, Greece
[2] Dept of Computer Eng. & Informatics, Univ. of Patras, 26500 Patras, Greece
{konstane,zaro}@ceid.upatras.gr
[3] Dept of Mathematics, Univ. of the Aegean, Karlovassi, 83200, Samos, Greece
stamatiu@aegean.gr
[4] Joint Research Group (JRG) on Communications and Information Systems Security (Univ. of the Aegean and Athens Univ. of Economics and Business)

Abstract. In many cryptographic applications it is necessary to generate elliptic curves (ECs) with certain security properties. These curves are commonly constructed using the *Complex Multiplication* method which typically uses the roots of *Hilbert* or *Weber* polynomials. The former generate the EC directly, but have high computational demands, while the latter are faster to construct but they do not lead, directly, to the desired EC. In this paper we present in a simple and unifying manner a complete set of transformations of the roots of a Weber polynomial to the roots of its corresponding Hilbert polynomial for all discriminant values on which they are defined. Moreover, we prove a theoretical estimate of the precision required for the computation of Weber polynomials. Finally, we experimentally assess the computational efficiency of the Weber polynomials along with their precision requirements for various discriminant values and compare the results with the theoretical estimates. Our experimental results may be used as a guide for the selection of the most efficient curves in applications residing in resource limited devices such as smart cards that support secure and efficient Public Key Infrastructure (PKI) services.

1 Introduction

Elliptic curve cryptography (ECC) has gained an increasing popularity over the years, as it emerges as a fundamental and efficient technological alternative for building secure public key cryptosystems. This stems from the fact that elliptic curves (ECs) give rise to algebraic structures that offer a number of distinct advantages (smaller key sizes and highest strength per bit) over more customary algebraic structures used in various cryptographic applications (e.g., RSA). The use of smaller parameters for a given level of cryptographic strength results in

* This work was partially supported by the Action IRAKLITOS (Fellowships for Research in the University of Patras) with matching funds from EC and the Greek Ministry of Education.

faster implementations, less storage space, as well as reduced processing and bandwidth requirements. These characteristics make ECC suitable for software as well as for hardware implementations.

The advantage of EC-based cryptography is particularly apparent in applications supporting *Public Key Infrastructure* (PKI) services. In a typical PKI scenario, all users wishing to communicate with each other securely, or access some other services offered by the PKI (e.g., notary services, user authentication, etc), are given two keys, the *public* key and the *private* key. The former is made available to all users through, e.g., a public directory. The latter should be handled with special care in order to avoid accidental disclosure or illicit interception. Protecting the private key is one of the most important issues within a PKI. To this end, the common practice is to generate the two keys within a physically secured (i.e., tamper resistant) device, which is most often a smart card. The public key is exported to a publicly available directory but the private key never leaves the smart card and, thus, it remains safe. It is at this point where elliptic curves can make a difference compared to more conventional schemes such as RSA: since EC-based cryptosystems can achieve the same level of security using much smaller keys and system parameters, the initialization of an EC-based system (generation of the elliptic curve, generation of private and public keys, etc) can be done more economically in hardware as it requires smaller hardware elements (e.g., registers, arithmetic and logic units, etc). Thus, EC-based cryptography seems a very attractive alternative to conventional public key cryptosystems for the support of PKI services. Moreover, in certain PKI applications, a vast number of such ECs may be required to be generated and this should be done as fast as possible. A typical example concerns a wireless and web-based PKI environment in which millions of client devices (e.g., PDAs) connect to secure servers [8]. Clients may be frequently requested to choose different key sizes and EC parameters depending on vendor preferences, security requirements, and processor capabilities. The large number of client connections/transactions along with the (possibly frequent) change of security parameters by the vendor (e.g., due to evolving market conditions and corporate policies) calls for strict timing response constraints not only on the server, but also on the client side.

A frequently employed method for generating elliptic curves with order satisfying certain desirable (security) properties, is the *Complex Multiplication* (CM) method. This method was used by Atkin and Morain in [1] for the construction of elliptic curves with good properties in the context of primality proving while the method was adapted to give rise to curves with good security properties by Spallek [21] and Lay and Zimmer [13] independently. Furthermore, a number of works appeared that compared variants of the CM method and presented experimental results of the construction efficiency such as the recent works of Enge and Morain [5], Müller and Paulus [16] as well as the PhD thesis of Weng [23] and the PhD thesis of Baier [3]. Briefly, the CM method takes as input a given number (usually a prime or a binary power) representing the order of the finite field upon which the EC will be defined and determines a specific parameter, called the *CM discriminant* D. The EC of the desirable order is generated by constructing certain class field polynomials based on D and finding their roots. The construction and the location of the roots (modulo the finite field's order) of these polynomials is perhaps the most crucial step in the whole process. The most commonly used class field polynomials are the Hilbert (original version of the CM method) and the Weber polynomials. Their main differences are: (i) the coefficients of Hilbert polynomials grow excessively large as the discriminant D increases, while for the same discriminant the Weber polynomials have much smaller coefficients and thus are easier and faster to construct, a thing that is especially desirable in a PKI setting where smart cards are in use; (ii) the roots of the Hilbert polynomial construct directly the EC, while the roots of the Weber polynomial have to be transformed to the roots of its corresponding Hilbert polynomial to construct the EC (a step that is not especially time consuming, however).

The use of Hilbert polynomials in the CM method requires high precision in the arithmetic operations involved in their construction, resulting in considerable increase in computing resources and thus make them not appropriate for fast and frequent generation of ECs within resource limited devices such as smart cards and PDAs. To overcome these shortcomings of Hilbert polynomials, two alternatives have been recently proposed: either to compute them off-line and store them for subsequent use (see, e.g., [18]), or to use Weber polynomials for certain values of D (see, e.g., [2,3,10,12,13,22]) and produce the required Hilbert roots from them. Although the former approach tackles adequately the efficient construction of ECs, there may still be problems with storing and handling several Hilbert polynomials with huge coefficients, especially on cryptographic hardware devices with limited resources. These problems are addressed by the second approach. However, the known studies treat only certain values of D; for example, the case of $D \equiv 7 \pmod 8$ and not divisible by 3 is treated in [2,3,10,13], while the cases of $D \not\equiv 3 \pmod 8$ and $D \not\equiv 0 \pmod 3$ were treated in [12,22]. To the best of our knowledge, the other cases of D (i.e., $D \equiv 3 \pmod 8$ and $D \equiv 0 \pmod 3$) have not been treated before explicitly.

Starting from the fact that it is desirable to work with Weber polynomials in various applications that require the fast and frequent generation of ECs, the goals of this paper are the following: (i) to provide a complete set of transformations of Weber to Hilbert roots that cover all possible values of D, (ii) to prove a theoretical estimate of the precision required for the construction of Weber polynomials and (iii) to provide experiments that indicate values of discriminant D leading to Weber polynomials with small computational requirements and compare their actual precision requirements with the theoretical estimate we give. To the best of our knowledge, no general theoretical treatment exists which gives the required root transformations for *all* possible values of the discriminant D. No general treatment exists either that for each value of D provides an estimation for the precision requirements of the Weber polynomials. We believe that our effort to present an exhaustive list of root transformations and an estimate of the precision requirements of Weber polynomials in a simple, unifying exposition will be useful to designers of ECC applications, especially applications that will be placed in smart cards in order to support various PKI services.

The rest of the paper is organized as follows. In Section 2 we review some basic definitions and facts about elliptic curves, number theory, and class field polynomials discussing some of their properties relevant to the generation of ECs. In Section 3 we give the transformations that map roots (modulo a prime) of Weber polynomials to roots of Hilbert polynomials. In Section 4 we provide a theoretical estimate of the precision requirements of Weber polynomials and, finally, in Section 5 we give a number of experimental observations that are of relevance to implementations, especially in resource limited devices.

2 Preliminaries

In this section we briefly review the necessary concepts from EC theory, number theory, and the Hilbert and Weber class field polynomials in order to better explain the importance of polynomials in ECC as well as to facilitate the presentation of the root transformations.

2.1 Elliptic Curves

In this work, we consider ECs defined over prime fields. An *elliptic curve* over a (prime) finite field F_p, $p > 3$ and prime, is denoted by $E(F_p)$ and it is comprised of all the points $(x, y) \in F_p$ (in affine coordinates) such that

$$y^2 = x^3 + ax + b, \tag{1}$$

where $a, b \in F_p$ satisfy $4a^3 + 27b^2 \neq 0$. These points, together with a special point denoted by $\mathcal{O}$ (the *point at infinity*) and a properly defined addition operation form an Abelian group. This is the *Elliptic Curve group* and the point $\mathcal{O}$ is its identity element (see [4,20] for more details on this group).

The *order m of an elliptic curve* is the number of the points in $E(F_p)$. The difference between m and p is measured by the so-called *Frobenius trace* $t = p + 1 - m$ for which Hasse's theorem (see e.g., [4,20]) states that $|t| \leq 2\sqrt{p}$, providing upper and lower bounds for m based on p:

$$p + 1 - 2\sqrt{p} \leq m \leq p + 1 + 2\sqrt{p}. \tag{2}$$

The *order of a point* $P \in E(F_p)$ is the smallest positive integer n for which $nP = \mathcal{O}$. Application of Langrange's theorem on $E(F_p)$ dictates that the order of a point $P \in E(F_p)$ divides the order of the elliptic curve group, and hence $mP = \mathcal{O}$ for any $P \in E(F_p)$. This in turn implies that the order of a point cannot exceed the order of the elliptic curve.

Among the most important quantities defined for an elliptic curve $E(F_p)$ given by Eq. (1) are the *curve discriminant* Δ and the j*-invariant*. These two quantities are given by the equations $\Delta = -16(4a^3+27b^2)$ and $j = -1728(4a)^3/\Delta$.

For a specific j-invariant $j_0 \in F_p$ (where $j_0 \neq 0, 1728$) two ECs can be readily constructed. This j-invariant is actually a root modulo p of a Hilbert polynomial. Let $k = j_0/(1728 - j_0) \bmod p$. One EC is given by Eq. (1) with $a = 3k \bmod p$ and $b = 2k \bmod p$. The other, which is called the *twist* of the first, is given by

$$y^2 = x^3 + ac^2 x + bc^3 \tag{3}$$

with c any quadratic non-residue in F_p. If m_1 and m_2 are the orders of an elliptic curve and its twist respectively, then $m_1 + m_2 = 2p + 2$. This implies that if one of the curves has order $p + 1 - t$, then its twist has order $p + 1 + t$, or vice versa (see [4, Lemma VIII.3]).

2.2 Quadratic Fields and Forms

Let ξ be an algebraic integer (algebraic number satisfying some monic[5] polynomial equation with integral coefficients), and let f and h be polynomials over $\mathbb{Q}$. Then, the collection of all numbers of the form $f(\xi)/h(\xi)$, $h(\xi) \neq 0$, constitutes a field denoted by $\mathbb{Q}(\xi)$ and called *the extension of* $\mathbb{Q}$ *by* ξ. If ξ is a root of an irreducible[1] quadratic polynomial over $\mathbb{Q}$, then $\mathbb{Q}(\xi)$ is called a *quadratic field.* Additional information on algebraic numbers and quadratic fields can be found in [17].

Let D be a positive integer which is not divisible by any square of an odd prime and which satisfies $D \equiv 3 \pmod 4$ or $D \equiv 4, 8 \pmod{16}$. The quantity $-D < 0$ is called a *fundamental discriminant.* The subset of integers in $\mathbb{Q}(\sqrt{-D})$ forms a ring which is denoted by $\mathbb{O}$. A *quadratic form* of a discriminant $-D$ is a 3-tuple of integers $[a, b, c]$ such that $b^2 - 4ac = -D$. The form is called *primitive* if $gcd(a, b, c) = 1$, and *reduced* if $|b| \leq a \leq c$ and $b \geq 0$ whenever $c = a$ or $|b| = a$.

There is a natural correspondence between a quadratic form $[a, b, c]$ and the root τ of the quadratic equation $az^2 + bz + c = 0$ with $\mathrm{Im}(\tau) > 0$: $-D$ is the discriminant of τ and $\tau = (-b + \sqrt{-D})/2a$. It can be proved that the set of primitive reduced quadratic forms of discriminant $-D$, denoted by $\mathcal{H}(-\mathcal{D})$, is finite. Moreover, it is possible to define an operation that gives to $\mathcal{H}(-\mathcal{D})$ the structure of an Abelian group whose neutral element is called the *principal form.* The order of $\mathcal{H}(-\mathcal{D})$ is denoted by $h(-D)$, or simply h if $-D$ is clear from the context. The principal form is equal to $[1, 0, D/4]$ if $D \equiv 0 \pmod 4$, and to $[1, -1, (D + 1)/4]$ if $D \equiv 3 \pmod 4$. In this case, the root of the quadratic equation, denoted by τ^*, is $\tau^* = \sqrt{-D}/2$ if $D \equiv 0 \pmod 4$, and $\tau^* = (1 + \sqrt{-D})/2$ if $D \equiv 3 \pmod 4$.

2.3 Class Field Polynomials

Let τ_k be the root corresponding to a reduced positive primitive quadratic form $[a_k, b_k, c_k] \in \mathcal{H}(-D)$. Then the *class equation* of $\mathbb{O} \subset \mathbb{Q}(\sqrt{-D})$, or *Hilbert polynomial,* is defined by

[5] A polynomial is called monic if its leading coefficient is 1, and irreducible if it cannot be written as a product of two polynomials.

$$H_D\left(x\right) = \prod_{k=1}^{h} (x - j(\tau_k)) \tag{4}$$

where the quantity $j(\tau_k)$, for $\tau_k \in \mathbb{O}$, is called a *class invariant* of $\mathbb{O}$. In particular, for any $\tau \in \mathbb{O}$, $j(\tau)$ is defined as

$$j(\tau) = \frac{(256\theta(\tau)+1)^3}{\theta(\tau)},$$

where $z = e^{2\pi\sqrt{-1}\tau}$, $\theta(\tau) = \frac{\Delta(2\tau)}{\Delta(\tau)}$, and

$$\Delta(\tau) = z\left(1 + \sum_{n\geq 1}(-1)^n\left(z^{n(3n-1)/2} + z^{n(3n+1)/2}\right)\right)^{24}.$$

The class invariants $j(\tau_k)$ (which are the roots of $H_D(x)$) generate a field over $\mathbb{Q}(\sqrt{-D})$ called the *Hilbert class field*.

Alternative generators of the class field can be provided by singular values of other functions. Such functions are powers of the Weber functions which generate the Weber polynomials.

The first main theorem of complex multiplication says that the class invariant $j(\tau)$, $\tau \in \mathbb{O}$, generates the Hilbert class field over $\mathbb{Q}(\sqrt{-D})$. Weber considered the explicit construction of the Hilbert class field using other modular functions $g(z)$. When $g(\tau)$ and $j(\tau)$ generate the same field over $\mathbb{Q}(\sqrt{-D})$, $g(\tau)$ is also called a *class invariant* of $\mathbb{O}$, and its minimal polynomial $W_D(x)$ is called the *reduced class equation* or the *Weber polynomial*. Weber polynomials are defined using the Weber functions (see [1,9])

$$f(z) = q^{-1/48}\prod_{m=1}^{\infty}(1 + q^{(m-1)/2}) \qquad f_1(z) = q^{-1/48}\prod_{m=1}^{\infty}(1 - q^{(m-1)/2})$$

$$f_2(z) = \sqrt{2}\ \ q^{1/24}\prod_{m=1}^{\infty}(1 + q^{m}) \qquad \text{where } q = e^{2\pi z\sqrt{-1}}.$$

Then, the Weber polynomial $W_D(x)$ is defined as

$$W_D(x) = \prod_{\ell=1}^{h'} (x - g(\tau_\ell)) \tag{5}$$

where $g(\tau_\ell)$ (a class invariant of $W_D(x)$) is an expression – depending on the value of D – of the Weber functions, $\tau_\ell \in \mathbb{O}$ satisfies the equation $a_\ell z^2 + 2b_\ell z + c_\ell = 0$ for which $4b_\ell^2 - 4a_\ell c_\ell = -4d$, where $d = D/4$ if $D \equiv 0 \pmod 4$, and $d = D$ if $D \equiv 3 \pmod 4$, and h' (the degree of $W_D(x)$) can be either h or $3h$. All class invariants for the different values of D will be defined in Section 3.

We would like to mention that the possible class invariants for a given discriminant D are potentially infinite, giving rise to different class polynomials and consequently to the problem of which one to use (for details see [5,6,19]). A comparison of all possible class invariants for a given D was made in [5] using as criterion the *height*[6] of their minimal polynomials, since it is computationally easier to use invariants that produce polynomials of small height. In particular, it is shown in [5] that Weber polynomials is one of the best choices between all possible polynomials. In addition, an ordering of class invariants (from the best to the worst) is made in [5], where the best invariant is the one with minimum height. According to this ordering, the cases of $D \equiv 0 \pmod 3$ are "worse" than the cases of $D \not\equiv 0 \pmod 3$.

3 Transformations of Weber Roots to Hilbert Roots

In this section, we will describe a complete set of transformations of the roots of the Weber polynomials to the roots of the corresponding (generated by the same D) Hilbert polynomials. The transformations will be given for all possible values of discriminant D.

In order to explain the transformations that will follow, we need two relations. From the definition of the Weber functions and the class invariant $j(\tau)$ that generates the Hilbert polynomials, one can readily obtain the following relations:

$$f(\tau)f_2\left(\frac{1+\tau}{2}\right) = e^{\pi\sqrt{-1}/24}\sqrt{2} \tag{6}$$

$$j(\tau) = \frac{(A-16)^3}{A}, \tag{7}$$

where $A \in \{f^{24}(\tau), -f_1^{24}(\tau), -f_2^{24}(\tau)\}$.

Recall from Section 2.3 that the Weber polynomial $W_D(x)$ is generated by its class invariants which are also the roots of the polynomial. The real roots of $W_D(x)$, for all values of D, are given by the class invariants presented in Tables 1 and 2. There are ten cases of discriminant D that define ten different class invariants. Recall from Section 2.2 that D is either $3 \pmod 4$ or $4, 8 \pmod{16}$ and that $d = D/4$ if $D \equiv 0 \pmod 4$, and $d = D$ if $D \equiv 3 \pmod 4$. This in turn implies that $d \equiv 3, 7 \pmod 8$ if $D \equiv 3 \pmod 4$, while $d \equiv 1, 2, 5, 6 \pmod 8$ when $D \equiv 4, 8 \pmod{16}$. The ten class invariants split into two groups of five each, depending on whether $D \not\equiv 0 \pmod 3$ or $D \equiv 0 \pmod 3$. Tables 1 and 2 give the details.

We verified the correctness of the class invariants given in Tables 1 and 2 using the IEEE Standard P1363 [9]. We would like to mention that: (i) the above class invariants have been very recently given in [15], where, however a different invariant is presented for the case of $d \equiv 3 \pmod 8$ in Table 2, which does not agree with the IEEE Standard P1363 [9]; (ii) the class invariants of Table 1 are also given in [1,11,22], where however a different invariant for the case of $d \equiv 5 \pmod 8$ is presented which does not agree with both the IEEE

[6] The logarithm of the largest coefficient of the polynomial.

d mod 8	class invariant
1	$f^2(\sqrt{-d})/\sqrt{2}$
2 or 6	$f_1^2(\sqrt{-d})/\sqrt{2}$
3	$f(\sqrt{-d})$
5	$f^4(\sqrt{-d})/2$
7	$f(\sqrt{-d})/\sqrt{2}$

Table 1. Class invariants for $D \not\equiv 0 \pmod 3$

d mod 8	class invariant
1	$f^6(\sqrt{-d})/(2\sqrt{2})$
2 or 6	$f_1^6(\sqrt{-d})/(2\sqrt{2})$
3	$f^3(\sqrt{-d})/2$
5	$f^{12}(\sqrt{-d})/2^3$
7	$f^3(\sqrt{-d})/(2\sqrt{2})$

Table 2. Class invariants for $D \equiv 0 \pmod 3$

Standard P1363 [9] and the class invariant for the same case ($D \not\equiv 0 \pmod 3$) given in [15].

In both applications that we described, it is necessary to obtain the roots modulo a prime p of the Hilbert polynomial and we want to achieve this by using the roots (modulo p) of Weber polynomials. Hence, there must be a way to retrieve a root modulo p of the Hilbert polynomial $H_D(x)$ from a root modulo p of the corresponding Weber polynomial $W_D(x)$. It can be shown that if we can find a transformation $T(\ )$ of a real root $g(\tau_\ell)$ of the Weber polynomial to a real root $j(\tau_i)$ of the corresponding Hilbert polynomial, then the same transformation will hold for the roots of the polynomials modulo p.

We have already mentioned that the class invariants given in Tables 1 and 2 represent the real roots of the Weber polynomials. We shall refer to all these values by $F(\sqrt{-d})$, where F depends on the value of D. It is also known that when τ^* corresponds to a principal form, then the class invariant $j(\tau^*)$ of $H_D(x)$ is a real root. Hence the goal is to find a transformation $T(\ )$ such that $j(\tau^*) = T(F(\sqrt{-d}))$.

The idea is as follows. From Equation (7) we know that if one of $f^{24}(\tau^*)$, $-f_1^{24}(\tau^*)$, $-f_2^{24}(\tau^*)$ can be calculated, then $j(\tau^*)$ can also be computed. The problem now is reduced in finding one of $f^{24}(\tau^*)$, $-f_1^{24}(\tau^*)$, $-f_2^{24}(\tau^*)$ from $F(\sqrt{-d})$. When $D \equiv 0 \pmod 4$, then $\tau^* = \sqrt{-d}$, and finding $f^{24}(\sqrt{-d})$, or $-f_1^{24}(\sqrt{-d})$, or $-f_2^{24}(\sqrt{-d})$ from $F(\sqrt{-d})$ is rather easy. When $D \equiv 3 \pmod 4$ however, then $\tau^* = \frac{1+\sqrt{-d}}{2}$, and finding $f^{24}(\frac{1+\sqrt{-d}}{2})$, or $-f_1^{24}(\frac{1+\sqrt{-d}}{2})$, or $-f_2^{24}(\frac{1+\sqrt{-d}}{2})$ from $F(\sqrt{-d})$ is a little more complicated (actually, we have to use Equation (6) for these cases).

In the following, we present the details for three different cases of D of the transformation of a real root R_W of a Weber polynomial to a real root R_H of the corresponding Hilbert polynomial. The remaining seven cases can be derived with a similar way and are not described due to lack of space.

(a) $D \equiv 7 \pmod 8$ and $D \not\equiv 0 \pmod 3$: From Table 1, the class invariant in this case is $\frac{f(\sqrt{-d})}{\sqrt{2}} = R_W$. Since $d \equiv 3 \bmod 4$, we have that $\tau^* = (1+\sqrt{-d})/2$. Using Equation (6), we get

$$f_2(\tau^*) = f_2(\frac{1+\sqrt{-d}}{2}) = e^{\frac{\pi\sqrt{-1}}{24}}\sqrt{2}f^{-1}(\sqrt{-d}) \Rightarrow f_2(\tau^*) = e^{\frac{\pi\sqrt{-1}}{24}}R_W^{-1}$$

$$\Rightarrow -f_2^{24}(\tau^*) = R_W^{-24} = A.$$

Thus, from Equation (7) we obtain

$$R_H = \frac{(A-16)^3}{A} = \frac{(R_W^{-24}-16)^3}{R_W^{-24}}.$$

(b) $D \equiv 3 \pmod 8$ and $D \not\equiv 0 \pmod 3$: This is a very interesting case because the degree of the Weber polynomial is three times larger than the degree of the corresponding Hilbert polynomial. This case is frequently referred to as problematic, because of the high degree of the resulting Weber polynomials. If the number of distinct roots of the Weber polynomial is exactly three times the number of the roots of the corresponding Hilbert polynomial, the transformation below maps three roots of the Weber polynomial into the same root of the Hilbert polynomial. Obviously, when the number of distinct roots of the Weber polynomial is equal to the number of distinct roots of the Hilbert polynomial, one root of the Weber polynomial is mapped to exactly one root of the Hilbert polynomial.

Since $d = D \equiv 3 \pmod 8$, we have that $\tau^* = \frac{1+\sqrt{-d}}{2}$, and from Table 1 the class invariant is $f(\sqrt{-d}) = R_W$. According to Equation (6) we have

$$f_2(\tau^*) = f_2(\frac{1+\sqrt{-d}}{2}) = e^{\frac{\pi\sqrt{-1}}{24}}\sqrt{2}f^{-1}(\sqrt{-d}) \Rightarrow -f_2^{24}(\tau^*) = 2^{12}R_W^{-24} = A.$$

Hence, by Equation (7)

$$R_H = \frac{(A-16)^3}{A} = \frac{(2^{12}R_W^{-24}-16)^3}{2^{12}R_W^{-24}}.$$

(c) $D/4 \equiv 2, 6 \pmod 8$ and $D \not\equiv 0 \pmod 3$: For this value of $D = 4d$, we have that $\tau^* = \sqrt{-d}$, and the class invariant is $f_1^2(\sqrt{-d})/\sqrt{2} = R_W$ (see Table 1). Then,

$$f_1^2(\tau^*) = f_1^2(\sqrt{-d}) = \sqrt{2}R_W \Rightarrow -f_1^{24}(\tau^*) = -2^6R_W^{12} = A$$

which by Equation (7) gives

$$R_H = \frac{(A-16)^3}{A} = \frac{(-2^6R_W^{12}-16)^3}{-2^6R_W^{12}} = \frac{(2^6R_W^{12}+16)^3}{2^6R_W^{12}}.$$

Finally, we would like to remark that in the IEEE Standard P1363 [9] formulas are given that lead directly from the roots of Weber polynomials to the parameters of the ECs, without involving transformations to roots of Hilbert polynomials. However, we believe that the transformations given in our paper, not only are they not slower than the formulas given in the Standard but they also give a justification as to how the roots of Weber polynomials lead to the construction of ECs.

4 Precision Requirements of Weber Polynomials

In this section we will focus on the precision required for the construction of the Weber polynomials. First, for reasons of comparison and completeness, we note that a very accurate estimation of the bit precision of Hilbert polynomials made in [13] gives an upper bound of $3.32(\Lambda_H + h/4 + 5)$, where $\Lambda_H = \frac{\pi\sqrt{D}}{\ln 10}\sum_{k=1}^{h}\frac{1}{a_k}$. Our experiments showed that this bound is remarkably accurate.

Let $\Lambda_W = \frac{\pi\sqrt{D}}{\ln 2}\sum_{\ell}\frac{1}{a_\ell}$, where the sum runs over the same values of ℓ as the product in Eq. (5). The bit precision required for the construction of the Weber polynomial is upper bounded by $v_0 + \Lambda_W$ (see e.g., [22]), where v_0 is a positive constant that handles round-off errors (typically $v_0 = 33$). This estimate of precision can be, however, much larger than the actual precision required for the Weber polynomials. For the case of $D \equiv 7 \pmod 8$ and not divisible by 3, a better upper bound of $3.32(1 + (\Lambda_H + h/4 + 5)/47)$ is provided by [13]. However, this precision estimate can not be used in other cases of D. For this reason, we provide in the following lemma a new precision estimate that covers all values of discriminant D.

Lemma 1. *The bit precision required for the construction of Weber polynomials for various values of the discriminant D is approximately*

$$c_1 h + \frac{\pi\sqrt{D}}{c_2 \ln 2}\sum_{\ell}\frac{1}{\alpha_\ell},$$

where the sum runs over the same values of ℓ as the product in Eq. (5) and the constants c_1 and c_2 are given by

$$c_1 = \begin{cases} 3 & \text{if } D \equiv 3 \pmod 8 \\ 1 & \text{if } D \not\equiv 3 \pmod 8 \end{cases} \tag{8}$$

$$c_2 = \begin{cases} 24 & \text{if } D \equiv 3,7 \pmod 8 \text{ and } D \not\equiv 0 \pmod 3 \\ 8 & \text{if } D \equiv 3,7 \pmod 8 \text{ and } D \equiv 0 \pmod 3 \\ 6 & \text{if } D/4 \equiv 5 \pmod 8 \text{ and } D \not\equiv 0 \pmod 3 \\ 2 & \text{if } D/4 \equiv 5 \pmod 8 \text{ and } D \equiv 0 \pmod 3 \\ 12 & \text{if } D/4 \equiv 1,2,6 \pmod 8 \text{ and } D \not\equiv 0 \pmod 3 \\ 4 & \text{if } D/4 \equiv 1,2,6 \pmod 8 \text{ and } D \equiv 0 \pmod 3 \end{cases} \tag{9}$$

Proof. For the case $D \not\equiv 3 \pmod 8$ and from the proof of Proposition (B4.4) in [11], if the Weber polynomial is written in the form $W_D(x) = x^h + w_{h-1}x^{h-1} + \ldots + w_1 x + w_0$ then $|w_i| \le 2^h M$, where $M = \prod_\ell \max(1, |g(\tau_\ell)|)$. This means that the bit precision required for the coefficient w_i of the polynomial is $\log_2(|w_i|) \le h + \log_2 M \le h + \sum_\ell \log_2(|g(\tau_\ell)|)$. Therefore, the bit precision required for the construction of the whole polynomial (i.e., the construction of its coefficients) is at most $h + \sum_\ell \log_2(|g(\tau_\ell)|)$ and thus $c_1 = 1$. If $D \equiv 3 \pmod 8$, then the degree of $W_D(x)$ is equal to $3h$ and repeating the same steps as above, we conclude that the bit precision is at most $3h + \sum_\ell \log_2(|g(\tau_\ell)|)$ which gives $c_1 = 3$. Thus, we need to estimate the precision requirements for the computation of $g(\tau_\ell)$.

The precision required by each $g(\tau_\ell)$ is related to the precision required by $f(\tau_\ell)$, $f_1(\tau_\ell)$ or $f_2(\tau_\ell)$ as evidenced by Table 1 and 2. We observe that, in general, g is equal to one of these functions raised to a constant power K and multiplied by a small constant which we will ignore in the following computations as it will affect the final estimate only by a very small additive constant. This implies that the precision needed for $g(\tau_\ell)$ is approximately K times the precision needed for $f(\tau_\ell)$, $f_1(\tau_\ell)$ or $f_2(\tau_\ell)$.

In addition, it is known that $j(z) = \frac{(f^{24}(z)-16)^3}{f^{24}(z)} = \frac{(f_1^{24}(z)+16)^3}{f_1^{24}(z)} = \frac{(f_2^{24}(z)+16)^3}{f_2^{24}(z)}$. These equalities imply that the precision needed for $j(\tau_\ell)$ is approximately 48 times the precision needed for $f(\tau_\ell)$, $f_1(\tau_\ell)$ or $f_2(\tau_\ell)$. Using the expansion of j in terms of its Fourier series (see [4]), we obtain that $|j(\tau_\ell)| \approx |e^{-2\pi\sqrt{-1}\ell}| = e^{2\pi\sqrt{D}/\alpha}$. Therefore, the bit precision that is required for the computation of $j(\tau_\ell)$ is $\log_2 |j(\tau_\ell)| \approx \frac{2\pi\sqrt{D}}{\alpha \ln 2}$ and, consequently, the precision required for $g(\tau_\ell)$ is given by $\log_2 |g(\tau_\ell)| \approx \frac{K}{48} \log_2 |j(\tau_\ell)| = \frac{2K\pi\sqrt{D}}{48\alpha \ln 2} = \frac{K\pi\sqrt{D}}{24\alpha \ln 2}$. This, in turn, results in the total bit precision requirements for the computation of the Weber polynomial: $c_1 h + \frac{K\pi\sqrt{D}}{24 \ln 2} \sum_\ell \frac{1}{\alpha_\ell}$.

We will show for a case of D how we can derive the constant c_2: for the case $D \equiv 3 \pmod 8$ and $D \equiv 0 \pmod 3$, the precision required by $g(\tau_\ell)$ is approximately three times (i.e., $K = 3$) the precision required by $f(\tau_\ell)$, $f_1(\tau_\ell)$ or $f_2(\tau_\ell)$ as it is evident from Table (2). Thus we obtain that the bit precision required in this case is given by $3h + \frac{3\pi\sqrt{D}}{24 \ln 2} \sum_\ell \frac{1}{\alpha_\ell}$. This results to $c_2 = 8$. Using the same reasoning we can compute the bit precision requirements for the other cases of D, completing the proof of the lemma. □

We would like to point out that our lemma suggests an ordering (based on the theoretical estimates) of the bit precision requirements for the different values of discriminant D. For example, it seems that the case $D \equiv 3, 7 \pmod 8$ and $D \not\equiv 0 \pmod 3$ requires less precision that the other cases and this can mean that these values of discriminants are better for implementations. This ordering is verified by our experiments described in Section 5 below.

5 Experimental Results

In this section, we present an experimental study regarding the computational efficiency and the bit precision requirements of Weber polynomials and how these requirements are compared with their approximated precision requirements using Lemma 1. Our implementations for generating the polynomials are actually part of a variant of the CM method that generates ECs of a desirable order and uses both Weber and Hilbert polynomials. In contrast with other implementations (see e.g., [2,3]) which have been carried out in `C++` and use advanced `C++` libraries (e.g., LiDIA [14]) our implementations have been carried out in ANSI C using the (ANSI C) library GNUMP [7] (for high precision floating point arithmetic and also for the generation and manipulation of integers of unlimited precision), for maximum portability. Our implementation and experimentation environment was a Pentium III PC (933 MHz) running Linux 2.4.10 and equipped with 256 MB of main memory. The code for the generation of ECs using Weber polynomials had size 53KB, while the code for the generation of ECs using Hilbert polynomials had size 49KB.

The construction of Hilbert and Weber polynomials require high-precision complex and floating point arithmetic with the greater demands placed by Hilbert polynomials. Also, their construction required the implementation of functions such as $\cos(x)$, $\sin(x)$, $\exp(x)$, $\ln(x)$, $\arctan(x)$ and $\sqrt{x}$, which were implemented using their Taylor series expansion. Since the basic complex number algebraic operations (addition, multiplication, exponentiation, and squaring) as well as a high precision floating point implementation of the other involved functions did not exist in GNUMP, we had to implement them from scratch.

In our experiments we have mainly focused on measuring the time (in secs) for the construction of Weber polynomials, the bit precision they require (number of bits that are required by the GNUMP library), and the difference between the actual precision and the approximated precision from Lemma 1. In Figure 1 we present the approximated vs. the actual bit precision for Weber polynomials with even discriminant D. The degree h of the polynomials ranges from 4 to 52 as the discriminant D increases from 228 to 9924. We observe that all the estimates of the precision are larger than the actual precision. However, for values of discriminant D that are not divided by 3 the estimation of the precision is close to the actual one, while for the other cases the estimation can be much larger. However, for all the cases of D, the estimate of precision is very close to the actual one for polynomials with small h ($h < 20$, corresponding to $D < 2500$ approximately). Similar observations hold for odd values of the discriminant D.

Figure 1 indicates also that there is an ordering in the precision figures for various values of D. It can be seen, for instance, that the case of $D/4 \equiv 1, 2, 6 \pmod 8$ and $D \not\equiv 0 \pmod 3$ requires less precision than the case $D/4 \equiv 5 \pmod 8$ and $D \not\equiv 0 \pmod 3$, which in turn is better than the case $D/4 \equiv 1, 2, 6 \pmod 8$ and $D \equiv 0 \pmod 3$. The worst case is $D/4 \equiv 5 \pmod 8$ and $D \equiv 0 \pmod 3$. The case of D that requires the least precision among all the possible values is $D \equiv 3, 7 \pmod 8$ and $D \not\equiv 0 \pmod 3$ (it is not included in Figure 1, because in this figure we report only on even values of D). Note also that a similar ordering is implied by the estimates provided by Lemma 1.

The difference in the precision requirements for the various values of discriminant D is reflected in the time requirements for the construction of the polynomials. In Figure 2, we summarize all possible cases of D. The degree h of the polynomials ranges from 50 to 150 and D from 10766 to 69396. It is evident that there is, again an ordering among the different cases of the discriminant. The worst case is $D/4 \equiv 5 \pmod 8 \equiv 0 \pmod 3$, while the best is $D \equiv 3, 7 \pmod 8 \not\equiv 0 \pmod 3$ (for $D = 68383$ and $h = 148$ the time for the construction of the polynomial is only 4.43 seconds). We observe that the three worst cases are those that correspond to values of D that are divided by 3. The ordering

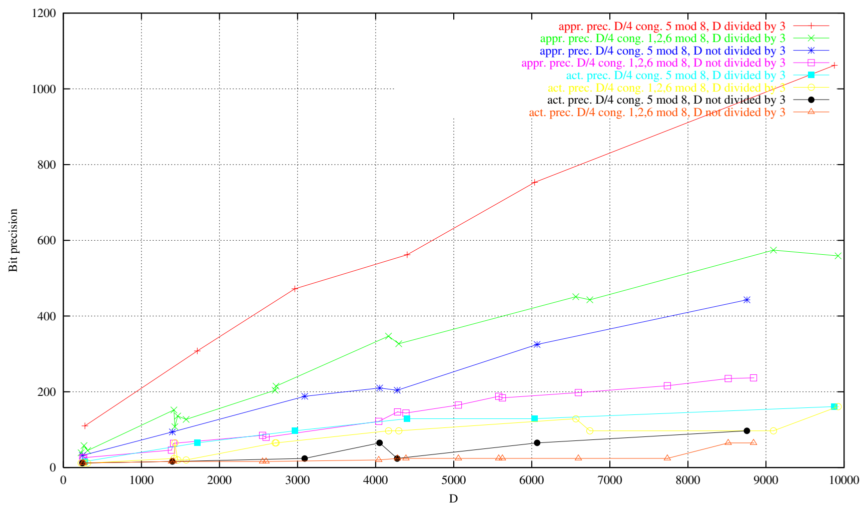

Fig. 1. Actual and approximated bit precision for the construction of Weber polynomials

between the two groups of D (divided or not divided by 3) is the same: the worst case is $D/4 \equiv 5 \pmod 8$, then is the case of $D/4 \equiv 1, 2, 6 \pmod 8$ and the best one is $D \equiv 3, 7 \pmod 8$. We also note that the same ordering is observed in the precision requirements for the various values of D. This apparent ordering may be helpful to the designers of ECCs as it may be used as a guideline for the selection of values of D that lead to Weber polynomials with the least computational requirements.

6 Conclusions

In this paper we have considered the use of Weber polynomials in elliptic curve cryptography implemented in resource limited devices, such as smart cards and PDAs. Weber polynomials are important in EC-based cryptography since their coefficients are considerably smaller than the coefficients of the corresponding Hilbert polynomials and can, thus, be manipulated with ease by resource limited devices that are used within PKIs. The only problem is that the roots of the Weber polynomials do not lead directly to the construction of the EC parameters and should be, first, transformed into equivalent Hilbert polynomial roots. To this end we have presented in this paper, in a unifying and simple manner, all the transformation of roots of Weber polynomials into roots of the corresponding Hilbert polynomials as well as estimates for the precision requirements of Weber polynomials for all possible discriminant values. We have also conducted experiments and compared the construction efficiency of Weber polynomials for various discriminant values. We observed that there is an ordering among the values of D that is defined by its divisibility properties. We believe that our

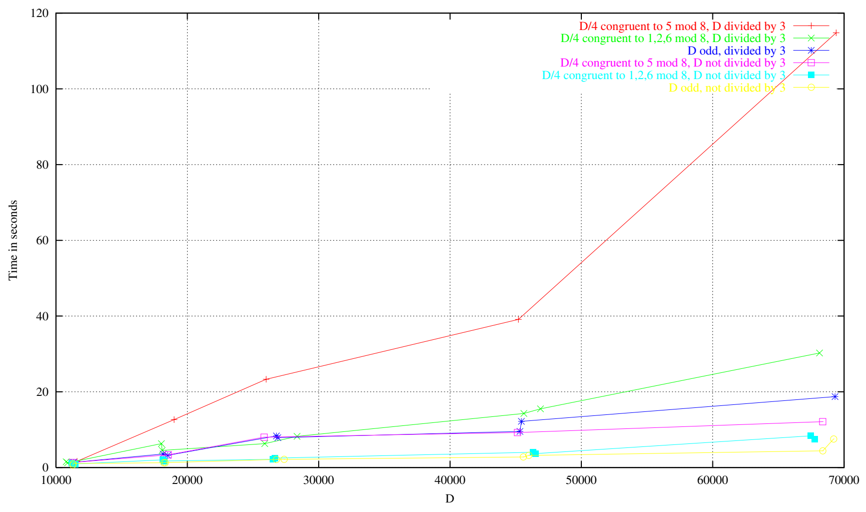

Fig. 2. Time for the construction of Weber polynomials

experimental results can be used as a guideline for the selection of the appropriate class field polynomials when constructing elliptic curves as the potential designer can have an estimate of the computation time as well as the precision required before the actual implementation is accomplished within some devices with limited computing power.

References

1. A.O.L. Atkin and F. Morain, Elliptic curves and primality proving, *Mathematics of Computation* **61**, pp. 29–67, 1993.
2. H. Baier and J. Buchmann, Efficient construction of cryptographically strong elliptic curves, in *Progress in Cryptology* – INDOCRYPT 2000, Lecture Notes in Computer Science Vol. 1977, Springer-Verlag, pp. 191–202, 2000.
3. H. Baier, Efficient Algorithms for Generating Elliptic Curves over Finite Fields Suitable for Use in Cryptography, PhD Thesis, Dept. of Computer Science, Technical Univ. of Darmstadt, May 2002.
4. I. Blake, G. Seroussi, and N. Smart, *Elliptic curves in cryptography* , London Mathematical Society Lecture Note Series 265, Cambridge University Press, 1999.
5. A. Enge and F. Morain, Comparing invariants for class fields of imaginary quadratic fields, in *Algorithmic Number Theory* – ANTS-V, Lecture Notes in Computer Science Vol. 2369, Springer-Verlag, pp. 252–266, 2002.
6. A. Enge and R. Schertz, *Constructing Elliptic Curves from Modular Curves of Positive Genus*, Preprint, March 2003.
7. GNU multiple precision library, edition 3.1.1, September 2000.
   Available at: `http://www.swox.com/gmp`.
8. N. Gura, H. Eberle, and S.C. Shantz, Generic Implementations of Elliptic Curve Cryptography using Partial Reduction, in *Proc. 9th ACM Conf. on Computer and Communications Security* – CCS'02, pp.108-116.

9. IEEE P1363/D13, *Standard Specifications for Public-Key Cryptography*, ballot draft, 1999. `http://grouper.ieee.org/groups/1363/tradPK/draft.html`.
10. E. Kaltofen, T. Valente, and N. Yui, An Improved Las Vegas Primality Test, in *Proc. ACM-SIGSAM 1989 International Symposium on Symbolic and Algebraic Computation*, pp. 26-33, 1989.
11. E. Kaltofen and N. Yui, Explicit construction of the Hilbert class fields of imaginary quadratic fields by integer lattice reduction. Research Report 89-13, Renseelaer Polytechnic Institute, May 1989.
12. E. Konstantinou, Y.C. Stamatiou, and C. Zaroliagis, On the Efficient Generation of Elliptic Curves over Prime Fields, in *Cryptographic Hardware and Embedded Systems* – CHES 2002, Lecture Notes in Computer Science Vol. 2523, Springer-Verlag, pp. 333–348, 2002.
13. G.J. Lay and H. Zimmer, Constructing Elliptic Curves with Given Group Order over Large Finite Fields, in *Algorithmic Number Theory* – ANTS-I, Lecture Notes in Computer Science Vol. 877, Springer-Verlag, pp.250-263, 1994.
14. LiDIA. *A library for computational number theory*, Technical University of Darmstadt. Available from `http://www.informatik.tu-darmstadt.de/TI/LiDIA/Welcome.html`.
15. F. Morain, Computing the cardinality of CM elliptic curves using torsion points, Preprint, October 2002.
16. V. Müller and S. Paulus, On the Generation of Cryptographically Strong Elliptic Curves, preprint, 1997.
17. I. Niven, H.S. Zuckerman, and H.L. Montgomery, *An Introduction to the Theory of Numbers*, John Wiley & Sons, 5th edition, 1991.
18. E. Savaş, T.A. Schmidt, and Ç.K. Koç, Generating Elliptic Curves of Prime Order, in *Cryptographic Hardware and Embedded Systems* – CHES 2001, LNCS Vol. 2162 (Springer-Verlag, 2001), pp. 145-161.
19. R. Schertz, Weber's class invariants revisited, *J. Théor. Nombres Bordeaux* **14:1**, 2002.
20. J. H. Silverman, *The Arithmetic of Elliptic Curves*, Springer-Verlag, GTM 106, 1986.
21. A.-M. Spallek, *Konstruktion einer elliptischen Kurve über einem endli-chen Körper zu gegebener Punktegruppe*, Master Thesis, Universitäat GH Essen, 1992.
22. T. Valente, *A distributed approach to proving large numbers prime*, Rensselaer Polytechnic Institute Troy, New York, PhD Thesis, August 1992.
23. A. Weng, *Konstruktion kryptographisch geeigneter Kurven mit komplexer Multiplikation*, PhD thesis, Institut für Experimentelle Mathematik, Universität GH Essen, 2001.