COMPUTER TECHNOLOGY INSTITUTE

TECHNICAL REPORT No. TR 2003/06/01

ON THE SELECTION OF CLASS FIELD POLYNOMIALS
IN ELLIPTIC CURVE CRYPTOGRAPHY

Elisavet Konstantinou Yannis C. Stamatiou Christos D. Zaroliagis

6 June 2003

On the Selection of Class Field Polynomials in Elliptic Curve Cryptography*

Elisavet Konstantinou† Yannis C. Stamatiou† Christos Zaroliagis†

6 June 2003

Abstract

In many cryptographic applications ranging from data encryption to primality testing and factoring, it is necessary to generate elliptic curves (ECs) whose order (number of points) possesses certain properties that pertain to the application at hand. Moreover, in certain applications a vast number of such ECs may be required. For example, in the ECPP algorithm for primality proving; or in implementations of EC-based cryptosystems on computing devices with limited resources or on systems operating under strict timing response constraints that are required to frequently change their EC parameters for security reasons. Regardless of the intended use, ECs of a desirable order are usually generated by the so-called Complex Multiplication method. This method eventually entails the use of the roots of certain class field polynomials defined on a specific parameter called the discriminant. The most commonly used polynomials are the Hilbert and Weber ones. The former can be used to directly generate the EC, but they are characterized by high computational demands. The latter have usually much lower computational requirements, but they do not directly construct the desired EC. However, they can be used to generate the EC provided that one can transform their roots to the roots of their corresponding (generated by the same discriminant) Hilbert polynomials. Hence, a natural question that arises is which polynomials to use. Previous research had partly addressed this problem by considering only certain values of the discriminant. 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 values of the discriminant, along with an extensive experimental assessment of their computational efficiency when used for generating ECs of a desirable order with Complex Multiplication. These transformations, along with the experiments we present, may serve as useful guidelines to potential implementers of EC cryptosystems involving generation of ECs of a desirable order.

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

*This work was partially supported by the IST Programme of EU under contracts no. IST-1999-14186 (ALCOM-FT) and no. IST-1999-12554 (ASPIS), and by the Human Potential Programme of EU under contract no. HPRN-CT-1999-00104 (AMORE).

†Computer Technology Institute and Department of Computer Engineering & Informatics, University of Patras, 26500 Patras, Greece. E-mails: {konstane,stamatiu,zaro}@ceid.upatras.gr.

smaller parameters for a given level of cryptographic strength requirements results in 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.

One of the fundamental issues in ECC is the generation of elliptic curves suitable for use in various cryptographic applications ranging from simple data encryption to more advanced uses such as primality testing and factoring. A common requirement of all such applications is that the *order* of the EC (number of elements in the algebraic structure induced by the EC) possesses certain properties, which gives rise to the problem of how such ECs can be generated. For example, it may be required that the order possesses three well-known conditions that ensure the robustness of the produced EC against some of the best known attacks (when considering data encryption or electronic signatures) [2, 4, 16], or it may be necessary that the order has a small prime factor (smoothness property, required in primality testing) [1]. Moreover, in certain 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 is the ECPP algorithm for primality proving [1]. Another application domain that mostly motivates our research concerns implementations of EC-based cryptosystems in computing devices with limited resources, or in systems operating under strict timing response constraints. Two specific scenarios in this framework involve: (i) The development of a proactive cryptosystem approach (e.g., in the sense of [11]) in networks of resource limited hardware devices (e.g., micro-controller chips, smart dust devices) working for some highly critical – with respect to security – task and which for that reason are frequently requested to refresh their security parameters. The requested reconfiguration of EC parameters should be done locally in a periodic manner or upon receipt of a "reconfigure"-signal and not by transmitting the new EC parameters to them. (ii) A wireless and web-based environment in which millions of client devices connect to secure servers [10]. 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 predetermined order, satisfying certain desirable properties, is the *Complex Multiplication* (CM) method [1, 4]. This 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 unboundedly 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; (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.

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. 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., [22]), or to use Weber polynomials for certain values of D (see e.g., [2, 3, 13, 16, 17, 25]) 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 on hardware devices with limited resources. These problems are addressed by the second approach. Despite the space and time efficiency though, 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, 13, 17], while the cases of $D \not\equiv 3 \pmod 8$ and $D \not\equiv 0 \pmod 3$ were treated in [16, 25]. To the best of our knowledge, the other cases for D ($D \equiv 3 \pmod 8$ and $D \equiv 0 \pmod 3$) were not treated.

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 goal of this paper is twofold: (i) to provide a complete set of transformations of Weber to Hilbert roots that cover all possible values of D; and (ii) to provide extensive experimental evidence on which class field polynomials a potential implementer/practitioner should use when considering implementations of EC cryptosystems on resource limited hardware devices or on systems with strict timing response constraints.

Although most of the transformations we present are either known or they can be derived from known properties of Hilbert and Weber polynomials, there still remains the fact that most of them are either scattered in various papers that usually require advance number-theoretic knowledge and are perhaps difficult for a practitioner to follow, or not cover all values of D (e.g., the case $D \equiv 3 \pmod 8$). We believe that our effort to present an exhaustive list of these transformations in a single, unifying, and simple exposition should be useful to designers of ECC applications that require the frequent generation of ECs with a desirable order.

We also present a number of experimental results related to the differences in computational efficiency between Weber and Hilbert polynomials, as well as how these experimental outcomes may be used as guidelines for the selection of the most efficient alternative in practice. We furthermore present a comparison regarding the computational efficiency among the various Weber polynomials. In either case, we feel that our results will be useful for potential implementers of EC cryptosystems.

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 elaborate more on the CM method as well as on another application where the choice of the class field polynomial plays an important role. In Section 4 we give the transformations that map roots (modulo a prime) of Weber polynomials to roots of Hilbert polynomials. In Section 5 we give a number of experimental observation that exhibit properties of Weber and Hilbert polynomials that are of relevance to implementations. We conclude in Section 6.

2 Preliminaries

In this section we briefly review the necessary concepts from ECs, 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. We assume familiarity with elementary number theory.

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, 24] 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, 24]) 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. 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

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

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[1] 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 [21].

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 $D \equiv 3 \pmod 4$.

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

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

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

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.*

The coefficients of $H_D(x)$ can grow very large. Let $\Lambda_H = \frac{\pi\sqrt{D}}{\ln 2}\sum_{k=1}^{h}\frac{1}{a_k}$. According to [1, 4], the bit precision for the generation of $H_D(x)$ is upper bounded by $v_0 + \binom{h}{\lfloor h/2 \rfloor}\Lambda_H$, where v_0 is a positive constant that handles round-off errors (typically $v_0 = 33$). A more careful elaboration on the precision made in [17] reduces the upper bound to $3.32(\Lambda_H + h/4 + 5)$.

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, 12])

$$\begin{aligned} f(z) &= q^{-1/48}\prod_{m=1}^{\infty}(1 + q^{(m-1)/2}) \qquad f_1(z) = q^{-1/48}\prod_{m=1}^{\infty}(1 - q^{(m-1)/2}) \\ f_2(z) &= \sqrt{2}\ q^{1/24}\prod_{m=1}^{\infty}(1 + q^m) \qquad \text{where } q = e^{2\pi z\sqrt{-1}}. \end{aligned}$$

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 4.

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., [25]). For the case of $D \equiv 7 \pmod 8$ and not divisible by 3, where $h = h'$, a better upper bound of $3.32(1 + (\Lambda_H + h/4 + 5)/47)$ is provided by [17].

Hence, the precision for the construction of the two types of polynomials may differ significantly as the degree of the polynomials and the value of D increases.

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 [6, 7, 23]). A comparison of all possible class invariants for a given D was made in [6] using as criterion the *height*[2] of their minimal polynomials, since it is computationally easier to use invariants that produce polynomials of small height. In particular, it is shown in [6] 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 [6], 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 Applications of Class Field Polynomials in ECC

In this section we elaborate more on two applications where the choice of class field polynomials (Hilbert or Weber) plays a crucial role, namely the CM method for generating ECs of a predetermined order and the primality testing algorithm given by [1].

3.1 Generation of Secure Elliptic Curves

The theory of complex multiplication (CM) of elliptic curves over the rationals can be used to generate elliptic curves of a suitable order m, i.e., the order must satisfy certain conditions which guarantee resistance to known attacks (see [4, Sec. V.7]). This results in the so-called *CM method*. The construction of Hilbert polynomials is the most crucial step in this method. The CM method computes the j-invariants of the EC from which is then easy to construct the EC (cf. Section 2.1).

The method is based on the following idea (for more details see [4, 12]). Hasse's theorem implies that $Z = 4p - (p+1-m)^2$ is positive. This in turn implies that there is a unique factorization $Z = Dv^2$, where D is a square free positive integer. Consequently,

$$4p = u^2 + Dv^2 \tag{6}$$

for some integer u satisfying $m = p + 1 \pm u$. D is called a *CM discriminant for the prime p* and the elliptic curve has a *CM by D*. EC theory dictates that D is congruent to either $0 \pmod 4$ or $3 \pmod 4$. The CM method uses D in order to determine the j-invariant and then constructs an EC of order $p+1-u$ or $p+1+u$.

The CM method requires as input a prime p. It then chooses the smallest D that along with integers u, v satisfy Eq. (6). The next step is to check whether $p+1-u$ and/or $p+1+u$ is suitable. If none of them is suitable, the process is repeated with another prime p as input. Otherwise, the Hilbert polynomial (see Section 2.3) is constructed and its roots (modulo p) are computed. A root of the Hilbert polynomial is the desired j-invariant of the EC. Then, the EC and its twist are constructed as explained in Section 2.1. Since only one of the ECs has the required suitable order, the particular one can be found using Langrange's theorem by picking random points P in each

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

EC until a point is found in some curve for which $mP \neq \mathcal{O}$. Then, the other curve is the one it is sought.

The most time consuming part of the CM method is the construction of the Hilbert polynomial which requires high precision floating point and complex arithmetic. As mentioned before, two alternative approaches have been proposed. The first one [22] presents a variant of the CM method that takes as input a CM discriminant D ($D \equiv 3 \pmod 8$), and subsequently calculates p and m. The advantage is that since Hilbert polynomials depend only on D (and not on p), they can be constructed in a preprocessing phase and stored for later use. Hence, the burden of their construction is excluded from the generation of the EC, but still there may be problems with storing and handling several Hilbert polynomials with huge coefficients on hardware devices with limited resources. The second alternative approach (see e.g., [2, 3, 13, 16, 17, 25]) follows either the original method described above, or a variant of it that takes as input D, computes subsequently p, m, the Weber polynomial and its roots, and then transforms its roots to the roots of the corresponding Hilbert polynomial. The studies in [2, 3, 13, 16, 17, 25] differ in several aspects regarding the implementations of the individual steps of the variant, the values of D considered, and the application domain. It is perhaps worth noting that the methods in [16, 22] seem the most appropriate for implementations on resource-limited hardware devices or in systems with strict timing response requirements.

3.2 Primality testing

Another motivation for the use of the class field polynomials stems from the Elliptic Curve Primality Proving (ECPP) algorithm, discussed in [1]. This algorithm certifies large integers to be prime via an elliptic curve certificate. Its main difference from the Goldwasser-Killian primality test given in [9] is that it avoids the costly computation of the order of the employed elliptic curve.

Let us first describe the Goldwasser-Killian primality test. Briefly, if p is prime, then Hasses's theorem (see Section 2) states that the order of the elliptic curve defined modulo p lies within the interval $[(\sqrt{p}-1)^2, (\sqrt{p}+1)^2]$ as the elliptic curve coefficients a and b vary. Goldwasser and Killian [9] used this fact to prove the following result on which the test is based: Let p be a positive integer and $E(F_p)$ be an elliptic curve over F_p. Let also m be an integer. Suppose that there exists a prime q that divides m and it is larger than $(p^{\frac{1}{4}}+1)^2$. If there exists a point P on $E(F_p)$ for which (i) $mP = \mathcal{O}$, and (ii) $(m/q)P$ is defined and is not equal to $\mathcal{O}$, then p is prime.

This result can be used for primality testing in the following way [1, 19, 25]. Let p be the input number which has to be tested for primality. Generate an elliptic curve and compute its order m, using the same procedure as in the CM method. Find a factor q of m where $q > (p^{\frac{1}{4}}+1)^2$ and check whether there is a point $P \in E(F_p)$ which satisfies conditions (i) and (ii). If not such q exists and/or the conditions are not satisfied, then find another elliptic curve and repeat the process. Otherwise, recurse with q in place of p until either some known prime is found or the process cannot be continued. In the former case, we know that p is prime by the above result.

There are two crucial steps in using the above primality proving scheme: (a) to generate ECs and compute their order, and (b) to partially factorize the order of an EC. The former step can be achieved using the CM method described in Section 3.1. Since this step needs to be computed repeatedly, it is desirable to have a fast method for generating ECs and compute their order. Weber polynomials serve as an important alternative to Hilbert polynomials that speeds up this step, which again boils down to find an efficient way of transforming Weber roots to Hilbert roots for various values of D. Note that Weber polynomials in primality testing were used in [13], where a transformation is presented for the case $D \equiv 7 \pmod 8$.

4 Transformation of Weber roots to Hilbert roots

In this section, we will elaborate on the transformation 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.

$$f(\tau)f_2\left(\frac{1+\tau}{2}\right) = e^{\pi\sqrt{-1}/24}\sqrt{2} \tag{7}$$

$$j(\tau) = \frac{(A-16)^3}{A}, \tag{8}$$

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 (mod 4) or 4, 8 (mod 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.

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 $D \equiv 0 \pmod 3$

We verified the correctness of the class invariants given in Tables 1 and 2 using the IEEE Standard 1363 [12]. We would like to mention that: (i) the above class invariants have been very recently given in [20], 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 1363 [12]; (ii) the class invariants of Table 1 are also given in [1, 14, 25], where however a different invariant for the case of $d \equiv 5 \pmod 8$ is presented which does not agree with both the IEEE Standard 1363 [12] and the class invariant for the same case ($D \not\equiv 0 \pmod 3$) given in [20].

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 (8) 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 (7) for these cases).

In the following, we present the details for the ten 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.

4.1 The case $D \not\equiv 0 \pmod 3$

(a) $D \equiv 7 \pmod 8$**:** From Table 1, the class invariant in this case is $\frac{f(\sqrt{-d})}{\sqrt{2}} = R_W$. Since $d \equiv 3$ mod 4, we have that $\tau^* = (1+\sqrt{-d})/2$. Using Equation (7), 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 (8) we obtain

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

(b) $D \equiv 3 \pmod 8$**:** 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 (7) 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 (8)

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

(c) $D/4 \equiv 2, 6 \pmod 8$**:** 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 (8) gives

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

(d) $D/4 \equiv 1 \pmod 8$**:** For this value of $D = 4d$, we have that $\tau^* = \sqrt{-d}$, and the class invariant is $f^2(\sqrt{-d})/\sqrt{2} = R_W$ (see Table 1). Thus,

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

which by Equation (8) gives

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

(e) $D/4 \equiv 5 \pmod 8$**:** For this value of $D = 4d$, we have that $\tau^* = \sqrt{-d}$, and the class invariant is $f^4(\sqrt{-d})/2 = R_W$. Thus,

$$f^4(\tau^*) = f^4(\sqrt{-d}) = 2R_W \Rightarrow f^{24}(\tau^*) = 2^6R_W^6 = A$$

By Equation (8), we get

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

4.2 The case $D \equiv 0 \pmod 3$

(a) $D \equiv 7 \pmod 8$**:** As Table 2 dictates, the class invariant in this case is $\frac{f^3(\sqrt{-d})}{2\sqrt{2}} = R_W$. Since $d = D \equiv 3 \bmod 4$, we have that $\tau^* = (1+\sqrt{-d})/2$. From Equation (7), we get

$$f_2(\tau^*) = f_2(\frac{1+\sqrt{-d}}{2}) = e^{\frac{\pi\sqrt{-1}}{24}}\sqrt{2}f^{-1}(\sqrt{-d}) \Rightarrow -f_2^{24}(\tau^*) = 2^{12}f^{-24}(\sqrt{-d}) = R_W^{-8} = A.$$

Hence, by Equation (8) we get

$$R_H = \frac{(A-16)^3}{A} = \frac{(R_W^{-8}-16)^3}{R_W^{-8}}.$$

(b) $D \equiv 3 \pmod 8$**:** Again from Table 2, the class invariant in this case is $\frac{f^3(\sqrt{-d})}{2} = R_W$. The degree of the Weber polynomial, as in the same case for $D \not\equiv 0 \pmod 3$, is three times larger than the degree of the corresponding Hilbert polynomial. For the particular value of D, we have that $\tau^* = (1+\sqrt{-d})/2$. From Equation (7) we get

$$f_2(\tau^*) \quad = \quad f_2(\frac{1+\sqrt{-d}}{2}) = e^{\frac{\pi\sqrt{-1}}{24}}\sqrt{2}f^{-1}(\sqrt{-d}) \Rightarrow -f_2^{24}(\tau^*) = 2^{12}f^{-24}(\sqrt{-d}) = 2^4R_W^{-8} = A$$

which, by Equation (8), leads to

$$R_H = \frac{(A-16)^3}{A} = \frac{(2^4R_W^{-8}-16)^3}{2^4R_W^{-8}}.$$

(c) $D/4 \equiv 2, 6 \pmod 8$**:** From Table 2 the class invariant is $f_1^6(\sqrt{-d})/(2\sqrt{2}) = R_W$, while for the particular D we have that $\tau^* = \sqrt{-d}$. Then,

$$f_1^6(\tau^*) = f_1^6(\sqrt{-d}) = 2\sqrt{2}R_W \Rightarrow -f_1^{24}(\tau^*) = -2^6 R_W^4 = A$$

and by Equation (8)

$$R_H = \frac{(A-16)^3}{A} = \frac{(-2^6 R_W^4 - 16)^3}{-2^6 R_W^4} = \frac{(2^6 R_W^4 + 16)^3}{2^6 R_W^4}.$$

(d) $D/4 \equiv 1 \pmod 8$**:** From Table 2 the class invariant is $f^6(\sqrt{-d})/(2\sqrt{2}) = R_W$, while for the particular D we have that $\tau^* = \sqrt{-d}$. Thus,

$$f^6(\tau^*) = f^6(\sqrt{-d}) = 2\sqrt{2}R_W \Rightarrow f^{24}(\tau^*) = 2^6 R_W^4 = A$$

and by Equation (8)

$$R_H = \frac{(A-16)^3}{A} = \frac{(2^6 R_W^4 - 16)^3}{2^6 R_W^4}.$$

(e) $D/4 \equiv 5 \pmod 8$**:** From Table 2, the class invariant for this case is $f^{12}(\sqrt{-d})/2^3 = R_W$. For the particular D, we have that $\tau^* = \sqrt{-d}$. Hence,

$$f^{12}(\tau^*) = f^{12}(\sqrt{-d}) = 2^3 R_W \Rightarrow f_0^{24}(\tau^*) = 2^6 R_W^2 = A$$

and by Equation (8)

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

5 Experimental Results

We have conducted an experimental study regarding the computational efficiency of Hilbert and Weber polynomials. Our implementations for generating the polynomials are actually part of a variant of the CM method that generates ECs of a desirable order. Since one of our main concerns is to give suggestions on potential implementers of ECCs on resource limited hardware devices, all of our implementations have been carried out in ANSI C using the (ANSI C) library GNUMP [8] (for high precision floating point arithmetic and also for the generation and manipulation of integers of unlimited precision), as the vast majority of language tools developed for such devices are based on ANSI C. Our prime goal was not necessarily efficiency, but the production of an EC-generation executable of as small size as possible. 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. It is worth noting that our code does *not* involve any dynamically linked libraries at runtime. The resulting implementation should be contrasted with other implementations regarding generation of ECs (see e.g., [2, 3]) which have been carried out in `C++` and use advanced `C++` libraries (e.g., LiDIA [18]). While these implementations may be faster than ours, they result in large code sizes which makes it difficult to adapt to resource limited hardware devices.

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 concentrated in measuring the time (in secs) for their construction, the bit-precision (number of bits that are required by the GNUMP library), and the number of Taylor terms required by the Taylor series expansion of the functions $\cos(x)$, $sin(x)$ etc. that we used.

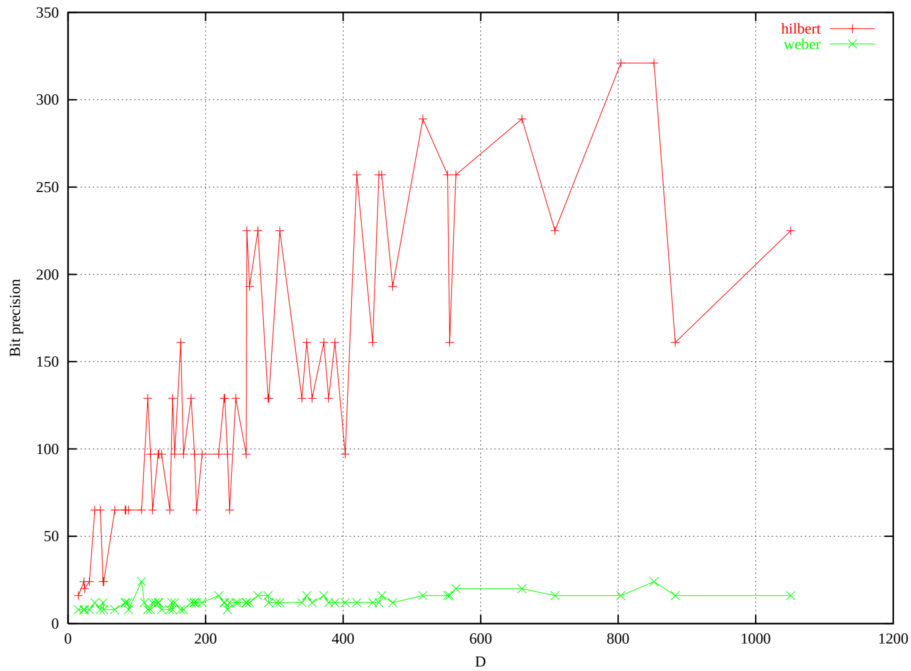

Figure 1: Bit precision for the construction of Hilbert and Weber polynomials

The construction of the Hilbert polynomials requires much larger bit-precision than the construction of the corresponding Weber polynomials for all the cases of discriminant D. This is clearly shown in Figure 1, where the precision needed for Hilbert polynomials is about 15-20 times larger than the precision needed for Weber polynomials, even for moderate values of D with small degree. Detailed values are given in the tables provided in the Appendix. In those tables we have also calculated the expected bit precision for both types of polynomials using the estimations presented in [17] (see Section 2.3). Although the given estimate for Weber polynomials concerns only the case of $D \equiv 7 \pmod 8$ and $D \not\equiv 0 \pmod 3$, we have used that estimate and in the other values of D. We observed that for small values of h the actual bit-precision required for the construction of the Weber polynomials is slightly larger than the expected one, while the opposite happens as h increases (a similar observation has been made in [3] for a different implementation setup). This is partly due to the fact that GNUMP allocates a minimum precision of 8 bits which increases in multiples of 4 up to 32 bits, above which precision is allocated in chunks of 32 bits. The expected bit precision for the Hilbert polynomials were always a little larger than the actual precision. Therefore, our experiments indicate that the estimates for the precision given in [17] for both polynomials are relatively accurate and can be used by designers of EC cryptographic systems to predict the bit precision required for their construction.

Regarding the number of Taylor terms required by the polynomials, we again have that the Hilbert polynomials demand a significantly larger amount, with one exception though: the case $D \equiv 3 \pmod 8 \equiv 0 \pmod 3$[3]. We noticed that in this case the number of Taylor terms required

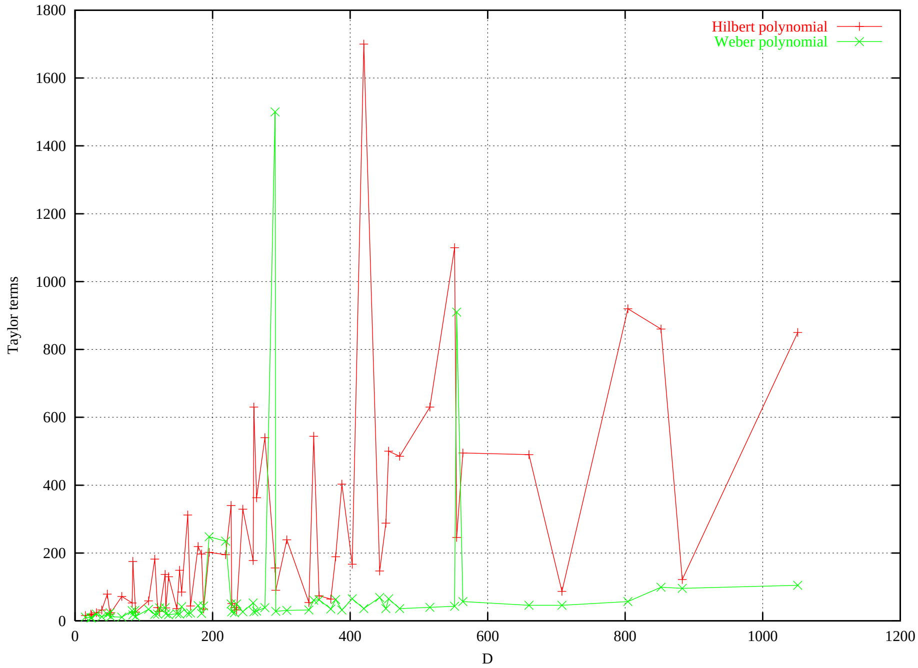

Figure 2: Hilbert vs Weber polynomials concerning the Taylor terms

by Weber polynomials is remarkably large (larger than the Taylor terms for the corresponding Hilbert polynomials) and consequently the construction time requirements, too. However, even in this case the precision requirements are very low for Weber polynomials in comparison with Hilbert polynomials. In Figure 2 the big difference between the number of Taylor terms needed for Hilbert polynomials and the Taylor terms that are needed for Weber polynomials is shown. This difference increases as D becomes larger. The peaks in Figure 2 for Weber polynomials are due to the case $D \equiv 3 \pmod 8 \equiv 0 \pmod 3$. Discarding those values, we observe that the peaks in the Weber polynomials vanish; see Figure 3. In the same figure we can see the clear difference in the Taylor terms between the two polynomials, when those values are discarded. Hence, if one wishes to use Weber polynomials, s/he should perhaps avoid these values of D.

The difference in time efficiency regarding the construction of the Hilbert and Weber polynomials is apparent in almost all cases (see the tables in the Appendix). This is mainly due to the big difference in the Taylor terms between the polynomials. The construction time of almost all Hilbert polynomials takes many hours or days, while the construction of the corresponding Weber polynomials takes few seconds. An exception occurs for the values of $D \equiv 3 \pmod 8 \equiv 0 \pmod 3$, which is the case of peaks of Weber polynomials in Figure 2, and whose construction time is usually larger than that of the corresponding Hilbert polynomials.

For the case $D \equiv 7 \pmod 8$ (either divisible or not by 3), we notice that the construction of Hilbert polynomials required incredibly many Taylor terms. For example the construction of the polynomial with $D = 127$ needed over 2500 Taylor terms and more than 3 days time. Hence, if one wishes to use Hilbert polynomials these values should perhaps be avoided.

We now turn to the bit-precision and the Taylor terms required by Weber polynomials. Both

[3]This notation means that $D \equiv 3 \pmod 8$ and divisible by 3.

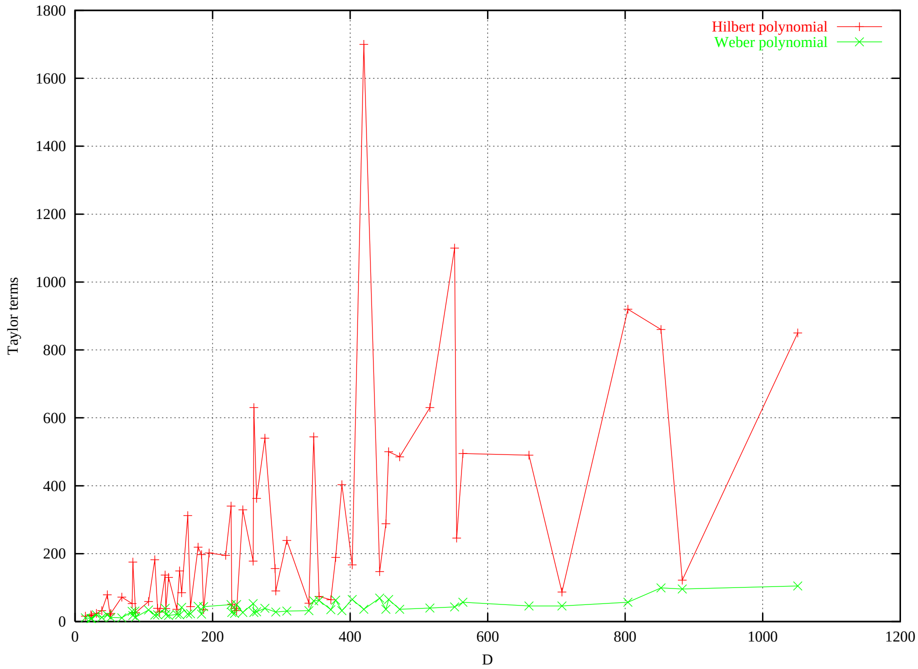

Figure 3: Hilbert vs Weber polynomials concerning the Taylor terms, without the case $D \equiv 3 \pmod 8 \equiv 0 \pmod 3$

seem to be proportional to the discriminant D and not to the degree of the polynomials. More precisely, we noticed that the behavior of Weber polynomials depends on whether D is divisible by 3 or not, even or odd. In Figure 4 there are the four different cases of D and in Figure 5 there are the same four cases, but without the values of $D \equiv 3 \pmod 8 \equiv 0 \pmod 3$. In Figure 5 we can see more closely the differences among the four cases. It is very interesting that for the cases that D is divisible by 3, there are peaks in the Taylor terms, while for the other two cases the Taylor terms increase regularly with D, regardless of the degree of the polynomials. This fact can be exploited by an implementer, since s/he can avoid the cases that D is divisible by 3 and can also have a very accurate estimate of the Taylor terms required by a polynomial when D is not divided by 3. In conclusion, the figures suggest that the best choice among the Weber polynomials are those with even discriminant and not divisible by 3.

On the other hand, Hilbert polynomials exhibit an "aberrant" behavior. Their precision and Taylor terms depend not only on the value of D but also on the degree of the polynomial. We further observed that some Hilbert polynomials have many trailing zeros in their coefficients. These polynomials have an advantage compared to the rest of Hilbert polynomials, because they can be more compactly stored and in many cases they require less Taylor terms and less precision. Moreover, it is easier for an implementer to verify that a polynomial is correctly computed if s/he knows that its coefficients have trailing zeros. We noticed that the trailing zeros appear for polynomials with $D \equiv 0, 2, 3, 5, 7, 8 \pmod{10}$.

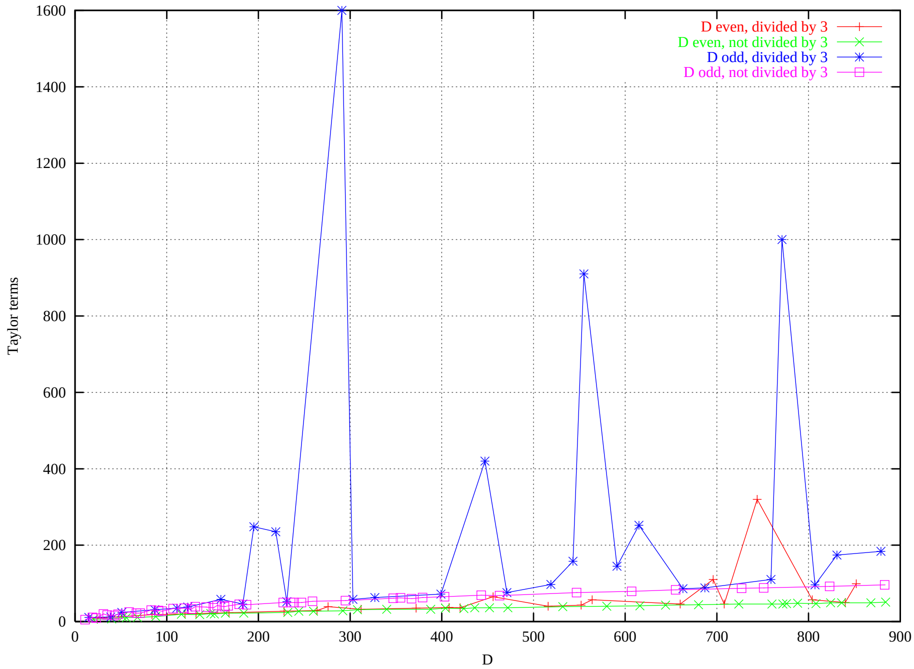

Figure 4: Taylor terms for Weber polynomials

6 Conclusions

In this paper we have considered the use of class field polynomials in elliptic curve cryptography. More specifically we have presented, in a unifying and simple manner, the transformation of roots of Weber polynomials into roots of the corresponding Hilbert polynomials for all possible discriminant values. We also conducted experiments for comparing Hilbert and Weber polynomials, as far as computational requirements are concerned, as well as experiments for comparing Weber polynomials for various discriminant values. We believe that our experimental results can be used as a guideline for the selection of the appropriate class field polynomials when constructing elliptic curves for constrained processing devices.

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.

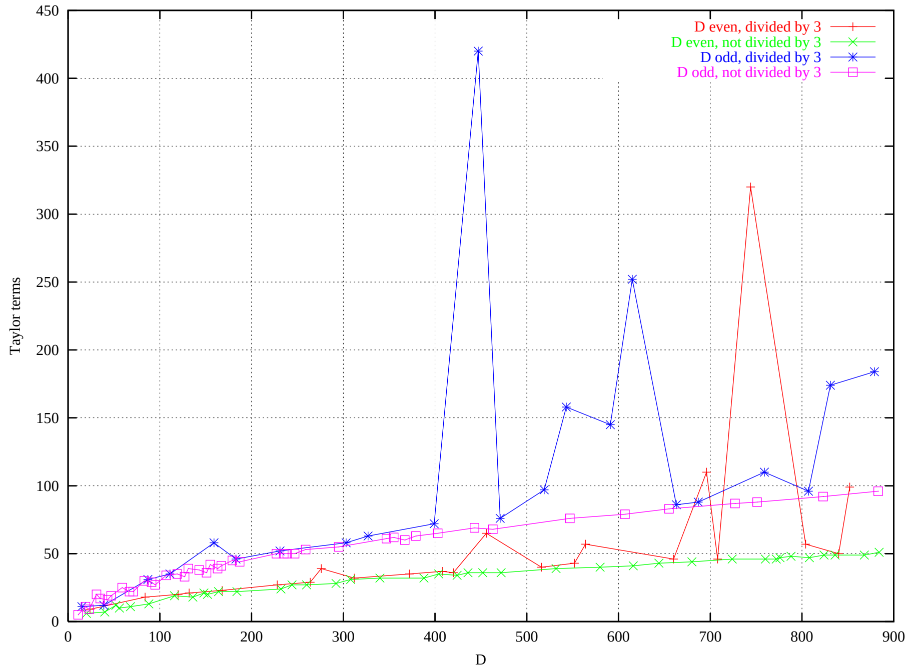

Figure 5: Taylor terms for Weber polynomials, without the case $D \equiv 3 \pmod 8 \equiv 0 \pmod 3$

[4] I. Blake, G. Seroussi, and N. Smart, *Elliptic curves in cryptography* , London Mathematical Society Lecture Note Series 265, Cambridge University Press, 1999.

[5] D. Burton, *Elementary Number Theory*, McGraw Hill, 4th edition, 1998.

[6] 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.

[7] A. Enge and R. Schertz, *Constructing Elliptic Curves from Modular Curves of Positive Genus*, Preprint, March 2003.

[8] GNU multiple precision library, edition 3.1.1, September 2000.
Available at: `http://www.swox.com/gmp`.

[9] S. Goldwasser, J. Kilian, Almost all primes can be quickly certified, *in: Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, pp. 316–329, 1986.

[10] 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.

[11] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung, Proactive Public Key and Signature Systems, in *Proc. 4th ACM Conf. on Computer and Communications Security* – CCS'97, pp.100-110.

[12] IEEE P1363/D13, *Standard Specifications for Public-Key Cryptography*, ballot draft, 1999. `http://grouper.ieee.org/groups/1363/tradPK/draft.html`.

[13] 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.

[14] 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.

[15] D.R. Kohel, Rational Groups of Elliptic Curves Suitable for Cryptography, in *Proceedings of the Workshop on Cryptography and Computational Number Theory* – CCNT 1999, National University of Singapore, 1999.

[16] 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.

[17] 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.

[18] LiDIA. *A library for computational number theory*, Technical University of Darmstadt. Available from `http://www.informatik.tu-darmstadt.de/TI/LiDIA/Welcome.html`.

[19] P. Mihăilesku, Recent developments in primality proving, *Mathematics and Computers in Simulation* **49**, 193–204, 1999.

[20] F. Morain, Computing the cardinality of CM elliptic curves using torsion points, Preprint, October 2002.

[21] I. Niven, H.S. Zuckerman, and H.L. Montgomery, *An Introduction to the Theory of Numbers*, John Wiley & Sons, 5th edition, 1991.

[22] E. Savas, T.A. Schmidt, and C.K. Koc, Generating Elliptic Curves of Prime Order, in *Cryptographic Hardware and Embedded Systems* – CHES 2001, Lecture Notes in Computer Science Vol. 2162, Springer-Verlag, pp. 145–161, 2001.

[23] R. Schertz, Weber's class invariants revisited, *J. Théor. Nombres Bordeaux* bf 14:1, 2002.

[24] J. H. Silverman, *The Arithmetic of Elliptic Curves*, Springer-Verlag, GTM 106, 1986.

[25] T. Valente, *A distributed approach to proving large numbers prime*, Rensselaer Polytechnic Institute Troy, New York, PhD Thesis, August 1992.

APPENDIX

We present here more detailed data regarding our experiments with the construction of Hilbert and Weber polynomials. All times in the tables are in seconds. Similar results hold for other values of D and h, although for the latter we couldn't use larger values as the time construction of the corresponding Hilbert polynomials was prohibitively high.

		Weber polynomial				Hilbert polynomial			
D	h	bit precision	expected precision	Taylor terms	Time	bit precision	expected precision	Taylor terms	Time
23	3	8	5	9	0.09	24	63	20	1.19
31^*	3	8	5	9	0.09	24	70	24	1.58
47	5	8	6	19	0.41	65	104	79	36.70

Table 3: Construction of Weber and Hilbert polynomials for $D \equiv 7 \pmod 8$ and $D \not\equiv 0 \pmod 3$. $(^*)$ Coefficients of Hilbert polynomials do not have trailing decimal zeros.

		Weber polynomial				Hilbert polynomial			
D	h	bit precision	expected precision	Taylor terms	Time	bit precision	expected precision	Taylor terms	Time
131^*	5	12	7	39	4.86	97	128	137	147.47
155	4	12	6	42	4.95	97	126	85	58.97
179^*	5	12	7	45	6.55	129	146	219	490.76
355	4	12	7	62	11.62	129	147	74	62.04
443	5	12	8	69	17.06	161	201	147	257.08
883	3	16	8	96	23.92	161	174	122	119.30
1051^*	5	16	9	105	48.35	225	240	850	20595.00

Table 4: Construction of Weber and Hilbert polynomials for $D \equiv 3 \pmod 8$ and $D \not\equiv 0 \pmod 3$. $(^*)$ Coefficients of Hilbert polynomials do not have trailing decimal zeros.

		Weber polynomial				Hilbert polynomial			
D	h	bit precision	expected precision	Taylor terms	Time	bit precision	expected precision	Taylor terms	Time
88	2	8	6	13	0.15	65	82	26	4.43
136^*	4	8	6	18	0.38	97	121	130	131.50
152	6	8	7	20	0.61	129	162	149	284.84
184^*	4	12	7	22	0.61	97	137	197	328.98
232	2	8	6	24	0.43	97	122	39	12.04
472	6	12	8	36	2.12	193	216	485	5696.00

Table 5: Construction of Weber and Hilbert polynomials for $D/4 \equiv 2, 6 \pmod 8$ and $D \not\equiv 0 \pmod 3$. $(^*)$ Coefficients of Hilbert polynomials do not have trailing decimal zeros.

		Weber polynomial				Hilbert polynomial			
D	h	bit precision	expected precision	Taylor terms	Time	bit precision	expected precision	Taylor terms	Time
68	4	8	6	11	0.18	65	101	72	30.88
164*	8	8	8	22	0.92	161	192	312	2191.00
260	8	12	9	27	1.82	225	229	630	18345.00
292	4	12	7	28	0.98	129	159	90	84.81
388	4	12	8	32	1.27	161	180	403	2479.00
452	8	12	10	36	2.68	257	286	288	2533.00

Table 6: Construction of Weber and Hilbert polynomials for $D/4 \equiv 1 \pmod 8$ and $D \not\equiv 0 \pmod 3$. (*) Coefficients of Hilbert polynomials do not have trailing decimal zeros.

		Weber polynomial				Hilbert polynomial			
D	h	bit precision	expected precision	Taylor terms	Time	bit precision	expected precision	Taylor terms	Time
52	2	8	5	12	0.13	24	68	24	1.58
116*	6	8	7	19	0.55	129	147	182	434.32
148	2	8	6	21	0.34	65	101	36	6.70
244*	6	12	8	27	1.19	129	177	329	1746.00
308	8	12	9	31	2.34	225	243	239	1830.18
340	4	12	7	32	1.65	129	171	54	52.84

Table 7: Construction of Weber and Hilbert polynomials for $D/4 \equiv 5 \pmod 8$ and $D \not\equiv 0 \pmod 3$. (*) Coefficients of Hilbert polynomials do not have trailing decimal zeros.

		Weber polynomial				Hilbert polynomial			
D	h	bit precision	expected precision	Taylor terms	Time	bit precision	expected precision	Taylor terms	Time
15	2	8	5	11	0.11	16	45	16	0.70
39	4	12	6	20	0.53	65	86	32	8.58

Table 8: Construction of Weber and Hilbert polynomials for $D \equiv 7 \pmod 8$ and $D \equiv 0 \pmod 3$. (*) Coefficients of Hilbert polynomials do not have trailing decimal zeros.

		Weber polynomial				Hilbert polynomial			
D	h	bit precision	expected precision	Taylor terms	Time	bit precision	expected precision	Taylor terms	Time
51*	2	12	5	24	0.97	24	62	20	1.19
123	2	12	6	38	2.40	65	86	28	4.80
195	4	12	6	248	490.79	97	124	202	467.18
219*	4	16	7	235	369.62	97	136	195	322.20
291*	4	16	7	1600	103454.00	129	154	156	234.84
555	4	16	8	910	18978.00	161	192	246	1016.54

Table 9: Construction of Weber and Hilbert polynomials for $D \equiv 3 \pmod 8$ and $D \equiv 0 \pmod 3$. (*) Coefficients of Hilbert polynomials do not have trailing decimal zeros.

		Weber polynomial				Hilbert polynomial			
D	h	bit precision	expected precision	Taylor terms	Time	bit precision	expected precision	Taylor terms	Time
24*	2	8	5	9	0.09	20	52	17	0.84
120	4	8	6	20	0.61	97	121	39	24.04
168	4	8	7	23	0.80	97	138	44	28.22
264*	8	12	9	29	2.11	193	221	363	4145.00
456*	8	16	10	65	12.62	257	275	500	10957.00
552	8	16	10	43	5.06	257	291	1150	94049.00

Table 10: Construction of Weber and Hilbert polynomials for $D/4 \equiv 2, 6 \pmod 8$ and $D \equiv 0 \pmod 3$. (*) Coefficients of Hilbert polynomials do not have trailing decimal zeros.

		Weber polynomial				Hilbert polynomial			
D	h	bit precision	expected precision	Taylor terms	Time	bit precision	expected precision	Taylor terms	Time
132	4	12	6	21	0.77	97	124	38	23.37
228	4	12	7	27	1.24	129	157	51	48.84
516*	12	16	11	40	5.89	321	347	630	26864.00
708	4	16	9	46	3.85	225	261	87	219.26
804*	12	16	13	57	12.40	385	416	940	91592.00

Table 11: Construction of Weber and Hilbert polynomials for $D/4 \equiv 1 \pmod 8$ and $D \equiv 0 \pmod 3$. (*) Coefficients of Hilbert polynomials do not have trailing decimal zeros.

		Weber polynomial				Hilbert polynomial			
D	h	bit precision	expected precision	Taylor terms	Time	bit precision	expected precision	Taylor terms	Time
84*	4	12	6	18	0.59	65	105	175	274.88
276*	8	16	9	39	4.14	225	226	540	12281.00
372	4	16	8	35	2.23	161	195	64	90.21
564*	8	20	10	57	9.58	257	303	495	10739.00
660	8	20	10	46	8.06	289	313	490	15278.00
852	8	24	11	99	34.78	321	345	880	50917.00

Table 12: Construction of Weber and Hilbert polynomials for $D/4 \equiv 5 \pmod 8$ and $D \equiv 0 \pmod 3$. (*) Coefficients of Hilbert polynomials do not have trailing decimal zeros.