Int. J. Inf. Secur. (2007) 6:47–63
DOI 10.1007/s10207-006-0009-3

REGULAR CONTRIBUTION

Efficient generation of secure elliptic curves

Elisavet Konstantinou · Yannis C. Stamatiou · Christos Zaroliagis

Published online: 29 November 2006

Abstract In many cryptographic applications it is necessary to generate elliptic curves (ECs) whose order possesses certain properties. The method that is usually employed for the generation of such ECs is the so-called *Complex Multiplication* method. This method requires 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 generate directly 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. This can be achieved if transformations of their roots to the roots of the corresponding (generated by the same discriminant) Hilbert polynomials are provided. In this paper we present a variant of the Complex Multiplication method that generates ECs of cryptographically strong order. Our variant is based on the computation of Weber polynomials. 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. In addition, we prove a theoretical estimate of the precision required for the computation of Weber polynomials for all values of the discriminant. We present an extensive experimental assessment of the computational efficiency of the Hilbert and Weber polynomials along with their precision requirements for various discriminant values and we compare them with the theoretical estimates. We further investigate the time efficiency of the new Complex Multiplication variant under different implementations of a crucial step of the variant. Our results can serve as useful guidelines to potential implementers of EC cryptosystems involving generation of ECs of a desirable order on resource limited hardware devices or in systems operating under strict timing response constraints.

Keywords Public key cryptography · Elliptic curve cryptosystems · Complex Multiplication · Weber polynomials

This work was partially supported by the IST Programme of EC under contract no. IST-2001-33116 (FLAGS), and by the Action IRAKLITOS (Fellowships for Research in the University of Patras) with matching funds from ESF (European Social Fund) and the Greek Ministry of Education.

E. Konstantinou · Y. C. Stamatiou · C. Zaroliagis
R.A. Computer Technology Institute,
Patras University Campus, N. Kazantzaki Str,
Patras 26500, Greece

E. Konstantinou
Department of Information and Communication Systems Engineering, University of the Aegean, Samos 83200, Greece
e-mail: ekonstantinou@aegean.gr

Y. C. Stamatiou
Department of Mathematics, University of Ioannina,
Ioannina 45110, Greece
e-mail: istamat@cc.uoi.gr

C. Zaroliagis (✉)
Department of Computer Engineering and Informatics,
University of Patras, Patras 26500, Greece
e-mail: zaro@ceid.upatras.gr

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 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 latter is of particular importance, since (under certain circumstances) it involves devices with limited resources such as cell phones, PDAs, and Smartcards.

One of the fundamental issues in ECC is the generation of ECs 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 should possess 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 [6, Sec. V.7] that ensure the robustness of the produced EC against some of the best-known attacks (when considering data encryption or electronic signatures), or it may be necessary that the order has small prime factors (smoothness property, required in primality testing) [2]. 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 [2].

Another application domain that 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 (e.g., in the sense of [15]) in networks of resource limited hardware devices (e.g., microcontroller chips) working on 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 [13]. 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 ECs with predetermined order, possessing certain desirable properties, is the *Complex Multiplication* (CM) method. This method was used by Atkin and Morain [2] for the construction of ECs with good properties in the context of primality proving, while the method was also adapted to give rise to curves with good security properties by Spallek [34] and Lay and Zimmer [22] independently. Furthermore, a number of works appeared that compared variants of the CM method and presented experimental results regarding its construction efficiency; see [3,9,25] for the most representative ones. Briefly, the CM method takes as input a number 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 are perhaps the most crucial steps 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; (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 in order 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 a considerable increase of computing resources. This makes them not appropriate for fast and frequent generation of ECs. To overcome the 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., [29]), or to use Weber polynomials for certain values of D (see e.g., [4,3,18,22,35]) 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 can be 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 [4,3,18,22], while the cases of $D \not\equiv 3 \pmod 8$ and $D \not\equiv 0 \pmod 3$ were treated in [20,35]. 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.

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, we present a variant of the CM that follows the second approach and provide a complete set of transformations of Weber to Hilbert roots that cover *all* possible values of D. We also investigate the theoretical and experimental bit-precision requirement for the construction of Hilbert and Weber polynomials in order to demonstrate the superiority of the latter, and we present a new approximate bound of the bit-precision required for the construction of Weber polynomials for all possible values of D. Our experiments show that this bound is close to the actual precision needed. Combining the theory behind Weber polynomials and our experiments, we can also indicate values of D leading to Weber polynomials with smaller computational requirements compared with other members of this family of polynomials. 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. 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 and implementers of ECC applications.

Besides the construction of the Weber polynomials, another important step of the CM method is the determination of the order p of the underlying prime field and the construction of the order m of the EC. This step is independent of the computation of Hilbert or Weber polynomials. We consider two different methods for implementing this step in our new CM variant. The first method to compute p and m is to use Cornacchia's algorithm [8]. The second method is to generate p and m at random.

Our second contribution in this paper is a comparative experimental study (Sect. 5) regarding these two methods for the computation of p and m. In our experiments, we used a large number of ECs and many different values of discriminant D. Our study revealed that the method based on Cornacchia's algorithm is dramatically slower than the method of generating p and m at random. It is worth mentioning that our findings coincide with those in [3], where a similar comparative experimental study was conducted with the help of the LiDIA library [23].

The rest of the paper is organized as follows. In Sect. 2, we briefly state some basic definitions and results from EC theory and we review some necessary concepts from number theory. In Sect. 3, we present the basic CM method and our variant, while in Sect. 4 we elaborate on the construction of Hilbert and Weber polynomials, the transformation of Weber roots to Hilbert roots and the new approximate bound for the bit-precision requirements of Weber polynomials. In Sect. 5 we discuss our experimental results. We conclude in Sect. 6. Preliminary parts of this work appeared in [20,21].

2 Preliminaries

In this section we review some basic concepts regarding ECs and their definition over finite fields. We also briefly review the theory of quadratic fields and forms, which is necessary for the construction of Hilbert and Weber polynomials. The interested reader may find additional information in [6,14,32,36]. We also assume familiarity with elementary number theory (see e.g., [7]).

2.1 Elliptic curve theory

An *EC* $E(F_p)$ over a finite field F_p, where $p > 3$ and prime, is the set of points $(x, y) \in F_p \times F_p$ (represented by affine coordinates) which satisfy the equation

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

and $a, b \in F_p$ are such that $4a^3 + 27b^2 \neq 0$. The set of solutions (x, y) of Eq. (1) together with a point $\mathcal{O}$, called the *point at infinity*, and a special addition operation define an Abelian group, called the *EC group*. The point $\mathcal{O}$ acts as the identity element (details on how the addition is defined can be found in e.g., [6,14,32,36]).

The *order* m of an EC is the number of the points in $E(F_p)$. The expression $t = p + 1 - m$ (which measures the difference between m and p) is called the *Frobenius trace* t. Hasse's theorem (see e.g., [6,14,32]) states that $|t| \leq 2\sqrt{p}$ which gives 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 is the smallest positive integer n for which $nP = \mathcal{O}$. Application of Langrange's theorem (see e.g., [7]) on $E(F_p)$, gives that the order of a point $P \in E(F_p)$ always divides the order of the EC group, so $mP = \mathcal{O}$ for any point $P \in E(F_p)$, which in turn implies that the order of a point cannot exceed the order of the elliptic curve.

Two important quantities associated with $E(F_p)$ are the *curve discriminant* Δ and the *j-invariant*, defined by

$$\Delta = -16(4a^3 + 27b^2) \tag{3}$$

and

$$j = \frac{-1728(4a)^3}{\Delta}. \tag{4}$$

Given $j_0 \in F_p$ ($j_0 \neq 0, 1728$), two ECs of j-invariant j_0 can be easily constructed. The first EC is of the form defined by Eq. (1) and can be constructed by setting $a = 3k \bmod p$, $b = 2k \bmod p$, where $k = \frac{j_0}{1728 - j_0} \bmod p$. The second EC, called the *twist* of the first, is defined as

$$y^2 = x^3 + ac^2x + bc^3 \tag{5}$$

where c is a quadratic non-residue in F_p. If m_1 is the order of an EC and m_2 is the order of its twist, then $m_1 + m_2 = 2p + 2$, i.e., if one curve has order $p + 1 - t$, then its twist has order $p + 1 + t$, or vice versa [6, Lemma VIII.3].

The security of elliptic curve cryptosystems is based on the difficulty of solving the discrete logarithm problem (DLP) on the EC group. To ensure intractability of solving this problem by all known attacks, the group order m should obey the following conditions (see [6, Chap. V] and [14, Chap. 4]):

1. m must have a sufficiently large prime factor (larger than 2^{160}).
2. m must not be equal to p.
3. For all $1 \leq k \leq 20$, it should hold that $p^k \not\equiv 1 \pmod{m}$.

The first condition excludes the application of types of methods like the Pohlig–Hellman [27] one to solve DLP, the second condition excludes the application of the anomalous attack [28,31,33], while the third condition excludes the MOV attack [24] and the Frey–Rück attack [11]. If the order of an EC group satisfies the above conditions, we call it *suitable*.

We would also like to note that sometimes there exists a fourth security requirement regarding the degree h of the class field polynomial. To the best of our knowledge, such requirement is only posed by the German Information Security Agency, which requires that h should be greater than 200. The reason is that there are few ECs produced from class field polynomials with smaller degrees and which may be amenable to specific attacks. However, no such attacks are known to date and this requirement does not seem to be part of the security requirements in any international security standard [4]. Despite this fact, we have taken into consideration such large values of h in our experimental study.

2.2 Quadratic fields and forms

Let ξ be an algebraic integer (algebraic number satisfying some monic[1] polynomial equation with integer 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 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 [26].

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 algebraic integers in $\mathbb{Q}(\sqrt{-D})$ forms a ring which is denoted by $\mathbb{O}$. A *quadratic form* of $-D$ 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 $-D$, 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 $D \equiv 3 \pmod 4$.

For notational simplicity, we will use (the positive integer) D to refer to the fundamental discriminant throughout the paper.

3 The Complex Multiplication method

The theory of Complex Multiplication (CM) of ECs over the rationals can be used to generate ECs of a suitable order m, resulting in the so-called *CM method*. The CM method computes j-invariants from which, as explained in Sect. 2.1, it is easy to construct the EC. The method is based on the following idea (for more details see [6,16]).

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,

[1] A polynomial is called monic if its leading coefficient is 1.

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

for some integer u satisfying

$$m = p + 1 \pm u. \tag{7}$$

D is actually a fundamental discriminant which is used by the CM method in order to determine a j-invariant and construct an EC of order $p + 1 - u$ or $p + 1 + u$.

The method starts with a prime p and then chooses the smallest D along with an integer u to satisfy Eq. (6). Then, it checks whether $p + 1 - u$ and/or $p + 1 + u$ is suitable. If neither is suitable, the process is repeated. Otherwise, a so-called Hilbert polynomial (see Sect. 4) has to be constructed (based on D) and its roots have to be found. A root of the Hilbert polynomial is the j-invariant we are seeking. The EC and its twist are then constructed as explained in Sect. 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 EC until a point is found in some curve for which $mP \neq \mathcal{O}$. Then, the other curve is the one we are seeking.

A major problem of the CM method is the construction of the Hilbert polynomials which require high precision floating point and complex arithmetic that makes their computation very expensive.

To overcome this problem, a variant of the CM method was proposed in [29]. It takes as input a CM discriminant $D \equiv 3 \pmod 8$, and subsequently calculates p and m, where the only condition posed on m is that it should be a prime. The prime p is found by picking randomly odd u and v of appropriate sizes first, and then checking if $(u^2 + Dv^2)/4$ is prime. An important aspect of the variant concerns the computation of the Hilbert polynomials: since they 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. In the rest of the section, we will describe a variant of the Complex Multiplication method that generates ECs of suitable order, and we will elaborate on a crucial step of this variant.

3.1 A variant of the CM method

In this subsection, we describe an alternative to the variant in [29] with which some similarities are shared: the variant we use takes also as input a CM discriminant D [not necessarily congruent to 3 (mod 8)], and then computes p and m. The differences are that the variant presented in this paper uses Weber instead of Hilbert polynomials, computes u and v using a different approach (for example, it uses Cornacchia's algorithm [8]), and requires m to be suitable, as defined in Sect. 2.1. Actually, the order m of the ECs that we generate is of the form $m = nq$, where n is a small integer and q is a large prime (larger than 2^{160}). A similar variant is also presented in [3] using mainly Hilbert polynomials. However, Weber polynomials are the default choice of our variant, since they require much less precision and, as our experiments show, result in much more efficient computation of ECs. (Hilbert polynomials can be equally used as well.) The polynomials, like in [29], can be constructed in a preprocessing phase.

In the following, we shall give the main steps of this variant. In order to facilitate the discussion of the experiments in Sect. 5, we will also include the choice of Hilbert polynomials in the description.

Preprocessing phase

1. Choose a discriminant D.
2. Construct the Weber (or the Hilbert) polynomial using the discriminant D.

Main phase

3. Specify a prime p such that the Diophantine equation (6) has a solution (u, v), where u, v are integers. The prime number p will be the order of the underlying finite field F_p.
4. Having found a solution (u, v), the possible orders of the elliptic curve are $m = p+1-u$ and $m = p+1+u$. Check if (at least) one of them is suitable. If none is suitable, then return to Step 3. Otherwise, m is the order of the EC that we will generate and proceed to the next step.
5. Compute the roots (modulo p) of the Weber (or Hil-bert) polynomial. This is accomplished using Berle-kamp's algorithm [5]. Transform the roots of the Weber polynomial (if it has been chosen) to the roots of the corresponding Hilbert polynomial (constructed using the same D).
6. Each Hilbert root represents a j-invariant. Construct the two ECs as described in Sect. 2.1 [cf. Eqs. (1) and (5)].
7. Determine which one of the two ECs is of a suitable order: repeatedly pick random points P on each EC, until a point is found for which $mP \neq \mathcal{O}$. Then, we are certain that the other curve is the one we seek.

3.2 Solving the Diophantine equation

A crucial step of our CM variant is Step 3, which concerns the solution of the Diophantine equation $4p = u^2+Dv^2$, where p is a prime number and u, v are integers.

Solving the Diophantine equation is necessary, since D and the polynomial are determined a priori. (The alternative approach of choosing first p and m, and subsequently compute D can be rather slow for constructing the polynomial, because D may be very large.)

In order to solve the Diophantine equation when a suitable order is required, we can use two alternative approaches (analyzed also in [3]). The most straightforward one is to randomly generate pairs (u, v) and then check if the number p that is constructed is a prime number. If it is, then we move to the next step of the CM method checking the two possible orders m for suitability. Otherwise, we generate another pair (u, v) and continue the same process. The second approach is to use Cornacchia's algorithm [8]. This algorithm solves a slightly different equation, namely the equation $p = x^2 + Dy^2$, having as inputs a prime p and the discriminant D. However, it is trivial to convert Eq. (6) into this form. If a solution is found for the Diophantine equation, then we proceed to the next step of the CM method setting $u = 2x$. Otherwise, another prime p is chosen and we again apply Cornacchia's algorithm.

4 Hilbert and Weber polynomials

The most complicated part of the CM method is the construction of the polynomials (Weber or Hilbert). In this section we shall elaborate more on these polynomials and discuss their strengths and limitations. In particular, in the following subsections we describe the construction of the Hilbert and Weber polynomials; we provide the transformations of the Weber roots to the Hilbert roots and we present an approximate bound for the bit-precision requirements of Weber polynomials.

4.1 Construction of polynomials

The CM discriminant D is the only input in the construction of Hilbert and Weber polynomials, denoted by $H_D(x)$ and $W_D(x)$, respectively. 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 the ring of algebraic integers $\mathbb{O} \subset \mathbb{Q}(\sqrt{-D})$, or *Hilbert polynomial*, is defined by

$$H_D(x) = \prod_{k=1}^{h} (x - j(\tau_k)). \tag{8}$$

Since the Hilbert polynomial is the class equation of $\mathbb{O}$, it clearly has integer coefficients. 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 $q = e^{2\pi\tau\sqrt{-1}}$, $\theta(\tau) = \frac{\Delta(2\tau)}{\Delta(\tau)}$, and

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

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.

Weber [37] considered the explicit construction of the Hilbert class field using other modular functions $g(z)$. When $g(\tau_\ell)$ and $j(\tau)$, for $\tau_\ell, \tau \in \mathbb{O}$, generate the same field over $\mathbb{Q}(\sqrt{-D})$, $g(\tau_\ell)$ 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 [2,16])

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

$$f_1(\tau) = q^{-1/48} \prod_{m=1}^{\infty} (1 - q^{m-1/2})$$

$$f_2(\tau) = \sqrt{2}\ q^{1/24} \prod_{m=1}^{\infty} (1 + q^m).$$

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

$$W_D(x) = \prod_{\ell=1}^{h'} (x - g(\tau_\ell)) \tag{9}$$

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 $a_\ell z^2 + 2b_\ell z + c_\ell = 0$ and corresponds to a primitive reduced quadratic form $[a_\ell, 2b_\ell, c_\ell]$ 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 (i) $\gcd(a_\ell, b_\ell, c_\ell) = 1$, (ii) $|2b_\ell| \leq a_\ell \leq c_\ell$, and (iii) if either $a_\ell = |2b_\ell|$ or $a_\ell = c_\ell$, then $b_\ell \geq 0$. In particular, $g(\tau_\ell)$ is constructed using the following equation given in [16]:

$$g(\tau_\ell) = \left[N \exp\left(\frac{-\pi\sqrt{-1}KLb_\ell}{24}\right) 2^{-I/6} \left(f_J(\tau_\ell)\right)^K\right]^G \tag{10}$$

where $J \in \{0, 1, 2\}$, $f_0(\tau_\ell) = f(\tau_\ell)$, $G = \gcd(D, 3)$, $I, K \in [0, 6]$, and L, N are positive integers. The precise values of these parameters depend on certain, rather tedious,

conditions among a_ℓ, c_ℓ and D that encompass the various cases of the mathematical definition of the Weber polynomials; the interested reader can find all the details in [16]. The degree h' of $W_D(x)$ can be either h or $3h$.

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 [9,10,30]). A comparison of many possible class invariants for a given D was made in [9] 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 [9] that Weber polynomials are among the best choices between all possible polynomials.

To get an idea on the size of coefficients of Hilbert and Weber polynomials as well as on their space requirements for storing them off-line, we give an example for $D = 472$.

$$
\begin{aligned}
W_{472}(x) &= x^6 - 12x^5 - 22x^3 - 12x - 1\\
H_{472}(x) &= x^6 - 43837086093832036927866859200 0x^5\\
&\quad + 290243510038159955925726906822209766336 \cdot 10^6 \cdot x^4\\
&\quad - 662197893286495898646518596497687462912 \cdot 10^{10} \cdot x^3\\
&\quad + 896632690216502725937652246573453867048 96 \cdot 10^{12} \cdot x^2\\
&\quad + 778276284755579240866437172085664074956 8 \cdot 10^{15} \cdot x\\
&\quad + (8476837240896000000)^3.
\end{aligned}
$$

As this example shows, the memory required for the storage of Hilbert polynomials is considerably larger than that required by the Weber polynomials.

4.2 Transformation of the 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, the following relations can be readily obtained:

$$f(\tau) f_2\left(\frac{-1+\tau}{2}\right) = \zeta_{48}^{-2} \mathrm{e}^{\pi\sqrt{-1}/24}\sqrt{2} \tag{11}$$

$$j(\tau) = \frac{(A-16)^3}{A}, \tag{12}$$

where $A \in \{f^{24}(\tau), -f_1^{24}(\tau), -f_2^{24}(\tau)\}$ and ζ_{48} is the 48th root of unity.

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

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

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

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	$f^3(\sqrt{-d})/(2\sqrt{2})$

Recall from Sect. 4.1 that the Weber polynomial $W_D(x)$ is generated by its class invariants $g(\tau_\ell)$ which are also 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 Sect. 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.

The fact that the class invariants of Tables 1 and 2 are indeed real roots of the Weber polynomial $W_D(x)$ can be verified using the IEEE Standard P1363 [16] and Eq. (10). In particular, the principal form for the case of Weber polynomials is equal to $[1,0,d]$. Then, the class invariant that corresponds to a real root is equal to $g([1,0,d]) = g(\sqrt{-d})$. For example, for $D \equiv 3 \pmod 8$ and $D \equiv 0 \pmod 3$, following [16] and Eq. (10), we get that $N = 1, J = 0, I = 2$ and $G = 3$. Therefore, the class invariant is $g(\sqrt{-d}) = f^3(\sqrt{-d})/2$. All the other nine cases can be similarly verified.

As we mentioned in Sect. 3, it is necessary for the CM method 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 compute 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 $j(\tau_k)$ of the corresponding Hilbert polynomial, then

the same transformation will hold for the roots of the polynomials modulo p.

We have earlier 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 the class invariant $j(\tau^*)$ of $H_D(x)$ is a real root when τ^* corresponds to a principal form. Hence the goal is to find a transformation $T(\,)$ such that $j(\tau^*) = T(F(\sqrt{-d}))$.

The idea is as follows. From Eq. (12) 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 to 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 straightforward. When $D \equiv 3 \pmod 4$ however, then $\tau^* = \frac{-1+\sqrt{-d}}{2}$, and finding $f^{24}\left(\frac{-1+\sqrt{-d}}{2}\right)$ or $-f_1^{24}\left(\frac{-1+\sqrt{-d}}{2}\right)$ or $-f_2^{24}\left(\frac{-1+\sqrt{-d}}{2}\right)$ from $F(\sqrt{-d})$ is more complicated (actually, we have to use Eq. (11) for these cases).

An interesting case of D is when $D \equiv 3 \pmod 8$ [for both $D \not\equiv 0 \pmod 3$ and $D \equiv 0 \pmod 3$]. For these two cases the degree of the Weber polynomial is three times larger than the degree of the corresponding Hilbert polynomial. 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 transformations that will be presented subsequently, map 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.

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.

1. $D \equiv 7 \pmod 8$ *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 Eq. (11), we get

$$f_2(\tau^*) = f_2\left(\frac{-1+\sqrt{-d}}{2}\right) = \zeta_{48}^{-2} e^{\frac{\pi\sqrt{-1}}{24}} \sqrt{2} f^{-1}(\sqrt{-d}) \Rightarrow$$

$$f_2(\tau^*) = \zeta_{48}^{-2} e^{\frac{\pi\sqrt{-1}}{24}} R_W^{-1} \Rightarrow -f_2^{24}(\tau^*) = R_W^{-24} = A.$$

Thus, from Eq. (12) we obtain

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

2. $D \equiv 3 \pmod 8$ *and* $D \not\equiv 0 \pmod 3$. For this case, the degree of the Weber polynomial is three times larger than the degree of the corresponding 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 Eq. (11) we have

$$f_2(\tau^*) = f_2\left(\frac{-1+\sqrt{-d}}{2}\right) = \zeta_{48}^{-2} e^{\frac{\pi\sqrt{-1}}{24}} \sqrt{2} f^{-1}(\sqrt{-d}) \Rightarrow$$

$$-f_2^{24}(\tau^*) = 2^{12} R_W^{-24} = A.$$

Hence, by Eq. (12)

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

3. $D/4 \equiv 2, 6 \pmod 8$ *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,

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

which by Eq. (12) gives

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

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

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

which by Eq. (12) gives

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

5. $D/4 \equiv 5 \pmod 8$ *and* $D \not\equiv 0 \pmod 3$. For this value of $D = 4d$, we have that $\tau^* = \sqrt{-d}$, and the class invariant is $f^4(\sqrt{-d})/2 = R_W$. Thus,

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

By Eq. (12), we get

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

6. $D \equiv 7 \pmod 8$ *and* $D \equiv 0 \pmod 3$. 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 Eq. (11), we get

$$f_2(\tau^*) = f_2\left(\frac{-1+\sqrt{-d}}{2}\right) = \zeta_{48}^{-2} e^{\frac{\pi\sqrt{-1}}{24}} \sqrt{2} f^{-1}(\sqrt{-d}) \Rightarrow$$
$$-f_2^{24}(\tau^*) = 2^{12} f^{-24}(\sqrt{-d}) = R_W^{-8} = A.$$

Hence, by Eq. (12) we get

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

7. $D \equiv 3 \pmod 8$ *and* $D \equiv 0 \pmod 3$. 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 case $D \equiv 3 \pmod 8$ and $D \not\equiv 0 \pmod 3$, 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 Eq. (11) we get

$$f_2(\tau^*) = f_2\left(\frac{-1+\sqrt{-d}}{2}\right) = \zeta_{48}^{-2} e^{\frac{\pi\sqrt{-1}}{24}} \sqrt{2} f^{-1}(\sqrt{-d}) \Rightarrow$$
$$-f_2^{24}(\tau^*) = 2^{12} f^{-24}(\sqrt{-d}) = 2^4 R_W^{-8} = A$$

which, by Eq. (12), leads to

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

8. $D/4 \equiv 2,6 \pmod 8$ *and* $D \equiv 0 \pmod 3$. 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 Eq. (12)

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

9. $D/4 \equiv 1 \pmod 8$ and $D \equiv 0 \pmod 3$. 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 Eq. (12)

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

10. $D/4 \equiv 5 \pmod 8$ *and* $D \equiv 0 \pmod 3$. 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 Eq. (12)

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

The previous discussion is summarized in the following theorem.

Table 3 Transformations for $D \not\equiv 0 \pmod 3$

D	R_H
$D \equiv 7 \pmod 8$	$\frac{\left(R_W^{-24}-16\right)^3}{R_W^{-24}}$
$D \equiv 3 \pmod 8$	$\frac{\left(2^{12} R_W^{-24}-16\right)^3}{2^{12} R_W^{-24}}$
$D/4 \equiv 2,6 \pmod 8$	$\frac{\left(2^6 R_W^{12}+16\right)^3}{2^6 R_W^{12}}$
$D/4 \equiv 1 \pmod 8$	$\frac{\left(2^6 R_W^{12}-16\right)^3}{2^6 R_W^{12}}$
$D/4 \equiv 5 \pmod 8$	$\frac{\left(2^6 R_W^{6}-16\right)^3}{2^6 R_W^{6}}$

Theorem 1 *Suppose R_W is a real root of a Weber polynomial $W_D(x)$ and R_H is a real root of the corresponding Hilbert polynomial $H_D(x)$. Then, R_W can be transformed to R_H using the equations from Tables* 3 *and* 4 *depending on the value of the discriminant D.*

4.3 Precision requirements for the construction of polynomials

In this section we focus on the precision required for the construction of Weber polynomials. For reasons of comparison and completeness, we note that a very accurate estimation of the bit precision of Hilbert polynomials made in [22] gives an upper bound of $3.32(\Lambda_H + h/4 + 5)$,

Table 4 Transformations for $D \equiv 0 \pmod 3$

D	R_H
$D \equiv 7 \pmod 8$	$\frac{\left(R_W^{-8}-16\right)^3}{R_W^{-8}}$
$D \equiv 3 \pmod 8$	$\frac{\left(2^4 R_W^{-8}-16\right)^3}{2^4 R_W^{-8}}$
$D/4 \equiv 2,6 \pmod 8$	$\frac{\left(2^6 R_W^{4}+16\right)^3}{2^6 R_W^{4}}$
$D/4 \equiv 1 \pmod 8$	$\frac{\left(2^6 R_W^{4}-16\right)^3}{2^6 R_W^{4}}$
$D/4 \equiv 5 \pmod 8$	$\frac{\left(2^6 R_W^{2}-16\right)^3}{2^6 R_W^{2}}$

where $\Lambda_H = 0.3(\frac{\pi\sqrt{D}}{\ln 2}\sum_{k=1}^{h}\frac{1}{a_k})$ and a_k is the first integer of the related reduced, primitive quadratic forms. Our experiments showed that this bound is remarkably accurate.

Let $\Lambda_W = \frac{\pi\sqrt{D}}{\ln 2}\sum_{\ell=1}^{h'}\frac{1}{a_\ell}$, where h' is the number of terms of the product in Eq. (9) and a_ℓ is the first integer of the corresponding reduced, primitive form. The bit precision required for the construction of the Weber polynomial is upper bounded by $v_0+\Lambda_W$ (see, e.g., [35]), 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 by 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 in [22]. Moreover, in [3] a more accurate precision bound of $0.015\Lambda_H$ is given for the same values of D. However, these precision estimates cannot be used for other cases of D. The next theorem gives a new precision estimate that covers all values of discriminant D.

Theorem 2 *The bit precision required for the construction of Weber polynomials for various values of the discriminant D is upper bounded by*

$$c_1 h + \frac{\pi\sqrt{d}}{c_2 \ln 2}\sum_{\ell=1}^{c_1 h}\frac{1}{a_\ell},$$

where the sum runs over the same values of ℓ as the product of Eq. (9) *(i.e., $h' = c_1 h$) and the constants c_1 and c_2 are given by*

$$c_1 = \begin{cases} 3 \ \textit{if } D \equiv 3 \pmod 8 \\ 1 \ \textit{if } D \not\equiv 3 \pmod 8 \end{cases} \tag{13}$$

$$c_2 = \begin{cases} 24 \ \textit{if } D \equiv 3,7 \pmod 8 \wedge D \not\equiv 0 \pmod 3 \\ 8 \ \textit{if } D \equiv 3,7 \pmod 8 \wedge D \equiv 0 \pmod 3 \\ 6 \ \textit{if } D/4 \equiv 5 \pmod 8 \wedge D \not\equiv 0 \pmod 3 \\ 2 \ \textit{if } D/4 \equiv 5 \pmod 8 \wedge D \equiv 0 \pmod 3 \\ 12 \ \textit{if } D/4 \equiv 1,2,6 \pmod 8 \wedge D \not\equiv 0 \pmod 3 \\ 4 \ \textit{if } D/4 \equiv 1,2,6 \pmod 8 \wedge D \equiv 0 \pmod 3. \end{cases} \tag{14}$$

Proof Consider the case $D \not\equiv 3 \pmod 8$. From the proof of Proposition (B4.4) in [17], assume that the Weber polynomial is written in the form $W_D(x) = x^h + w_{h-1}x^{h-1} + \cdots + w_1 x + w_0$. It turns out that $|w_i| \le 2^h M$, where $M = \prod_\ell \max(1, |g(\tau_\ell)|)$. This means that the bit precision required for the coefficient w_i 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 $c_1 = 1$. For the case $D \equiv 3 \pmod 8$, the degree of $W_D(x)$ is equal to $3h$. Arguing as above, we conclude that the bit precision is at most $3h + \sum_\ell \log_2(|g(\tau_\ell)|)$, which gives $c_1 = 3$. To conclude the proof, it suffices 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 Tables 1 and 2. We observe that, in general, $g(\tau_\ell)$ is equal to one of these functions raised to a constant power K and multiplied by a small constant, which we can safely 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 $f_2(\tau_\ell)$. Since the latter are related to $j(\tau_\ell)$ through Eq. (12), it turns out that an estimate for the precision of $j(\tau_\ell)$ yields an estimate for the precision of $g(\tau_\ell)$.

Equation (12) implies that the precision needed for $j(\tau_\ell)$ is approximately 48 times the precision needed for $f(\tau_\ell)$, $f_1(\tau_\ell)$ or $f_2(\tau_\ell)$. Consequently, $\log_2|g(\tau_\ell)| \approx \frac{K}{48}\log_2|j(\tau_\ell)|$. Using the expansion of j in terms of its Fourier series [6], we obtain that $|j(\tau_\ell)| \approx |e^{-2\pi\tau_\ell\sqrt{-1}}| = e^{2\pi\sqrt{d}/a_\ell}$. Therefore, the bit precision that is required for the computation of $j(\tau_\ell)$ is $\log_2|j(\tau_\ell)| \approx \frac{2\pi\sqrt{d}}{a_\ell \ln 2}$ and, consequently, the precision required for $g(\tau_\ell)$ is given by $\log_2|g(\tau_\ell)| \approx \frac{K}{48}\log_2|j(\tau_\ell)| = \frac{2K\pi\sqrt{d}}{48 a_\ell \ln 2} = \frac{K\pi\sqrt{d}}{24 a_\ell \ln 2}$. This, in turn, results in the total bit precision requirements for the computation of the Weber polynomial, which is $c_1 h + \frac{K\pi\sqrt{d}}{24\ln 2}\sum_\ell \frac{1}{\alpha_\ell}$.

We turn now to the computation of $c_2 = \frac{24}{K}$. For the case $d = D \equiv 3 \pmod 8$ and $D \equiv 0 \pmod 3$, the precision required by $g(\tau_\ell)$ is approximately three times the precision required by $f(\tau_\ell)$, $f_1(\tau_\ell)$ or $f_2(\tau_\ell)$ as it is evident from Table 2. Consequently, $c_2 = \frac{24}{3} = 8$. The rest of the cases follow similarly, thus completing the proof of the theorem. □

Theorem 2 suggests a natural ranking of the bit precision requirements based on the different values of D. For example, the case $D \equiv 3,7 \pmod 8$ and $D \not\equiv 0 \pmod 3$ requires less precision than the other cases and this implies that these values of D are better for

implementations. This ranking is verified by the experiments described in Sect. 5.

5 Implementation and experimental results

In this section, we discuss some issues regarding the implementation of our variant of the Complex Multiplication method and our experimental results concerning its time and space efficiency. As mentioned in the Sect. 1, one of our main concerns was to investigate the efficiency of implementing CM variants in resource-limited hardware devices (e.g., embedded systems). For that reason and for reasons of proper comparison, we have made all of our implementations in a unified framework using the same language and software libraries. Since the vast majority of language tools developed for such devices are based on ANSI C, we have made all of our implementations in this language using the (ANSI C) GNU Multiple Precision (GNUMP) [12] library for high precision floating point arithmetic and also for generating and manipulating integers of unlimited precision. Our implementation is also part of a software library for ECC that we build [19]. The library is available from http://www.ceid.upatras.gr/faculty/zaro/software/ecc-lib/. Note that there are highly efficient and optimized C++ libraries (e.g., LiDIA [23]), which however result in executables of a few MB, that may be prohibitive for resource-limited hardware devices.

As a first step, we implemented the basic algebraic operations for EC arithmetic. We then turned our attention to the most demanding step of the CM method, which was the construction of the Hilbert and Weber polynomials. They both require high-precision complex and floating point arithmetic with the greater demands placed, of course, by Hilbert polynomials. Also, the operations involved required the implementation of functions such as $\cos(x)$, $\sin(x)$, $\exp(x)$, $\ln(x)$, $\arctan(x)$ and $\sqrt{x}$. Since the basic complex number algebraic operations (addition, multiplication, exponentiation, and squaring) as well as a high precision floating point implementation of the above functions did not exist in GNUMP, we had to implement them from scratch. For the implementation of the particular functions we used their Taylor series expansion. As a starting point for the construction of the Hilbert polynomials, we used the code given in [38] which we considerably modified in order to support high precision floating point arithmetic. For the construction of the Weber polynomials we implemented the functions described in the IEEE Standard P1363 [16], adopting a slightly different way for producing the coefficients α, β, γ described in the standard. For the computation of the roots of polynomials modulo a prime, we used the code given in [38], which we had to modify in order to handle correctly prime numbers of arbitrary precision.

Table 5 Construction of Weber and Hilbert polynomials

D	h	Weber polynomial		Hilbert polynomial	
		Bit precision	Time (s)	Bit precision	Time (s)
228	4	12	0.08	129	0.73
1,384[a]	10	16	0.14	417	7.22
2,487	20	24	0.26	897	69.74
3,092	26	24	0.32	961	110.26
3,967	33	16	0.37	1,217	219.94
5,060	40	24	0.55	1,505	537.71
6,744[a]	44	97	0.95	1,601	699.08
9,924[a]	52	161	1.97	2,049	1,365.29
39,608	100	129	2.74	4513	20,679.00

[a] Coefficients of Hilbert polynomials do not have trailing decimal zeros

Our experiments were carried out on a Pentium III (933 MHz) with 256 MB of main memory, running Linux, and using the ANSI C gcc-2.95.2 compiler (along with the GNUMP library). All reported times are averages over 1,000 ECs per value of the discriminant D. For the size of the field's order, we considered two values, namely 192 and 224 bits. The Weber (resp. Hilbert) version of our code (excluding dynamically called libraries) had size 124 KB (resp. 110 KB) including the code for the generation of the polynomials; exclusion of the latter (i.e., when polynomials are computed off-line) reduces the code size to 72 KB. The small size of our code makes it suitable for memory limited devices. In fact, our code has been successfully ported to Windows CE, which is one of the most commonly used operating systems for PDAs. Details can be found in [1].

5.1 Construction of Hilbert and Weber polynomials

In this section we consider experiments regarding the construction of Hilbert and Weber polynomials. Our experiments are focused on the bit precision and the time requirements needed for the construction of the polynomials.

We have considered various values of D and h and made several experiments. We observed a big difference in favor of Weber polynomials both w.r.t. precision and time. Figure 1 illustrates the approximate (theoretical) estimate of the bit precision required for the construction of Weber and Hilbert polynomials, as discussed in Sect. 4.3, and the *actual precision*, i.e., the minimal precision required for their actual construction during the experiments . In particular, the degree h of the polynomials ranges from 4 to 52 as the discriminant D increases from 228 to 9924.

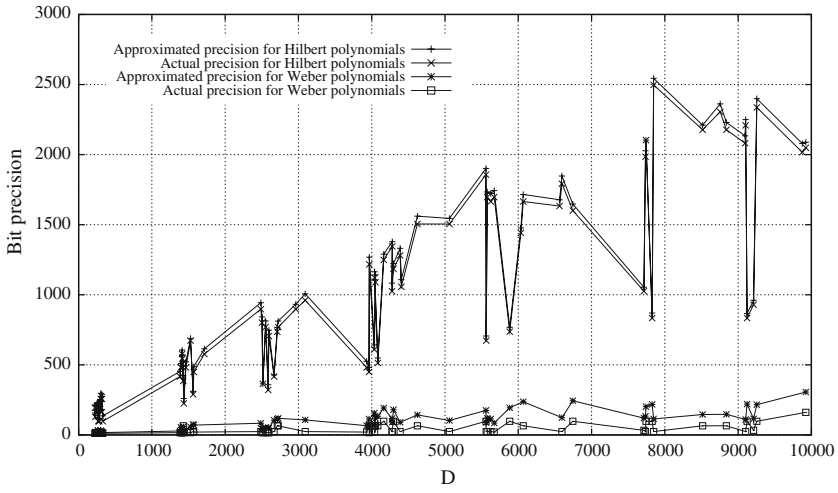

Fig. 1 Actual and approximate bit precision for the construction of Hilbert and Weber polynomials

As it is evident from Fig. 1, there is a large difference in the required precision between the two types of polynomials. We also observe that the approximate precision estimates are very close to the actual precision used in the implementation. For Hilbert polynomials the approximation from [22] was used, while for Weber polynomials that of Theorem 2. The difference is reflected also in the time requirements for the construction of polynomials and grows considerably large for bigger values of D and h as it is indicated in Table 5. For example, when $D = 39,608$ and $h = 100$ the time required for the construction of the Weber polynomial is only 2.74 s, while the time needed for the construction of the corresponding Hilbert polynomial is approximately 5 h and 45 min. In Table 5, we also report on the bit precision requirements for the construction of the two types of polynomials for the different values of D and h considered. In conclusion, the construction of Hilbert polynomials is very inefficient both in space and time.

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. Moreover, it is easier for an implementer to verify that a polynomial is correctly computed if it is known that its coefficients have trailing zeros. We noticed that the trailing zeros appear for polynomials with $D \not\equiv \pm 1 \pmod 5$.

Regarding the precision requirements of Weber polynomials and their theoretical estimates, representative results are reported in Figs. 2 and 3, where we present the approximate versus the actual bit precision for Weber polynomials with even discriminant D, distinguishing between those divisible by 3 (Fig. 2) and those which are not divided by 3 (Fig. 3). The degree h of the polynomials ranges from 4 to 52 and the discriminant D from 228 to 9924. We observe that all the estimates of the precision are larger than the actual precision. However, for all cases of D, the estimate of precision is very close to the actual one for polynomials with small h ($h < 20$, corresponding to $D < 2,500$ approximately). Similar observations hold for odd values of the discriminant D (see Fig. 4). As it was mentioned in Sect. 4.3, a better approximate precision was given in [3] for the case of $D \equiv 7 \pmod 8$ and not divisible by 3.

Figures 2, 3 and 4 also indicate that there is a ranking in the precision requirements 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 $D \equiv 0 \pmod 3$. Note also that a similar ranking is implied by the estimates provided by Theorem 2. Moreover, notice from Table 5 that the precision required for the construction of the Weber polynomial $W_{9924}(x)$ is larger than the precision required for the construction of $W_{39608}(x)$, even though the latter has degree approximately two times larger than the former. The reason is that $D = 9,924$ is divisible by 3, while $D = 39,608$ is not.

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 as illustrated in Fig. 5, which summarizes all possible cases of D. The degree h of the polynomials ranges from 50 to 150 and D from 10,766 to 69,396. The ranking among the different values of the discriminant appears in this case, too. The ranking between the two groups of D (divided or not divided by 3) is the same as the one observed for the precision: the worst case is $D/4 \equiv 5 \pmod 8$, followed by the case of $D/4 \equiv 1, 2, 6 \pmod 8$, while

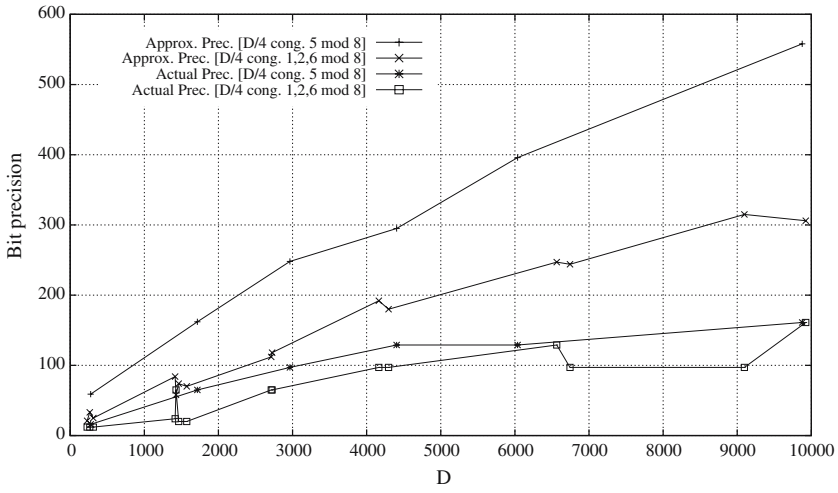

Fig. 2 Actual and approximate bit precision for the construction of Weber polynomials with even D, which is divisible by 3

the best one is $D \equiv 3, 7 \pmod 8$ (for $D = 68,383$ and $h = 148$ the time for the construction of the polynomial is only 4.43 s). This ranking may be helpful for 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.

Concerning the storage requirements of the Weber polynomials, it is clear that their coefficients are much smaller than the coefficients of the corresponding Hilbert polynomials. Moreover, the size of the coefficients of the Weber polynomials for the different cases of D is analogous to the precision required for their construction (see Theorem 2). For example, the coefficients of polynomials with $D \equiv 0 \pmod 3$ are larger than the coefficients of polynomials with $D \not\equiv 0 \pmod 3$, when the discriminants and the degrees of the polynomials are comparable in size. We also noticed that Weber polynomials with even D are more beneficial than Weber polynomials with odd D when it comes to storage requirements. The reason is that the absolute values of their coefficients are symmetric around the central coefficient. For example, $W_{116}(x) = x^6 - 9x^5 + 5x^4 + 2x^3 - 5x^2 - 9x - 1$. Using this property, we can reduce the space required for the storage of the particular Weber polynomials.

5.2 Computation of p and m

We next turn to the efficiency of Steps 3 and 4 of the CM variant (Sect. 3.1) assuming that the polynomials (Weber or Hilbert) have been computed off-line during the preprocessing phase. Firstly, we compared the performance of the Cornacchia's algorithm with the performance of the random method for the solution of the Diophantine equation and for the generation of EC's suitable order m. For the order m we check the three conditions mentioned in Sect. 2.1. The first condition was tested as follows in our implementation. The order must be of the form $m = nq$, where n is an integer and q is a large prime (greater than 2^{160}). The test proceeds by factoring m and demanding that there are at most four small factors (smaller than 20), while one factor should be prime. If this fails, then the particular m is rejected and the process is repeated. It is easy to see that in this

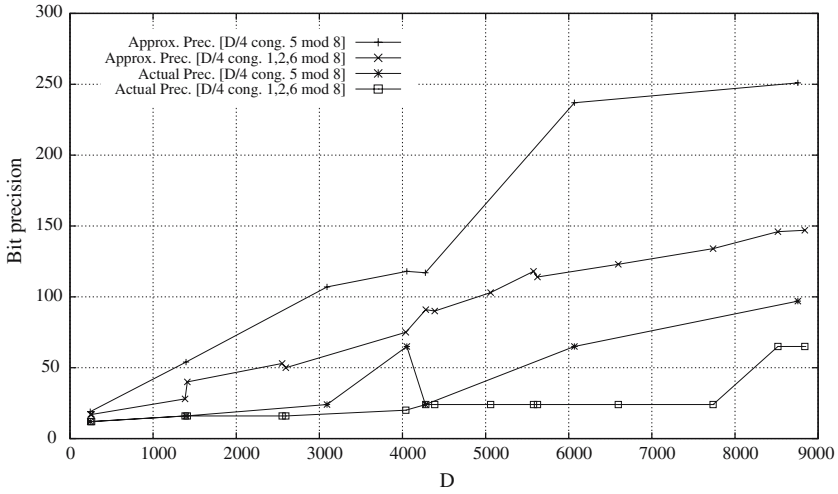

Fig. 3 Actual and approximate bit precision for the construction of Weber polynomials with even D, which is not divisible by 3

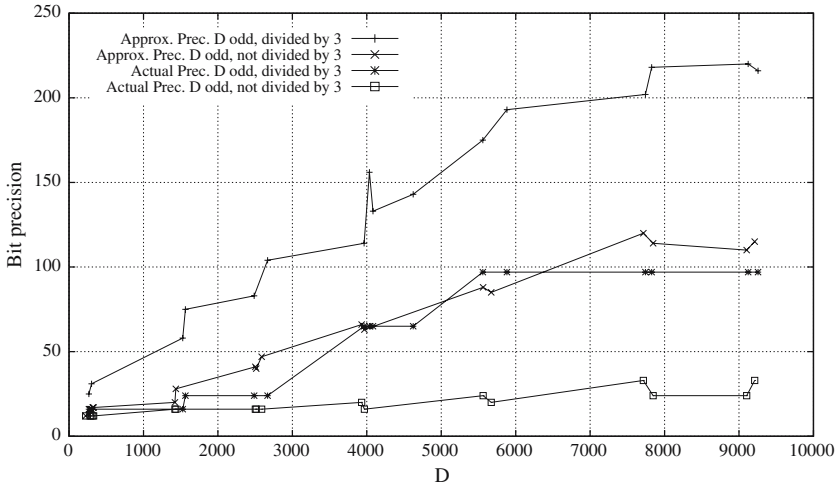

Fig. 4 Actual and approximate bit precision for the construction of Weber polynomials with odd D

Table 6 Timing estimations (in s) for the computation of p and m in the 192-bit finite field

D	h	$\#p$	$\#m$	$T(p,m)$	$\#p_r$	$\#m_r$	$T(p_r,m_r)$
232	2	4	5	0.63	158	5	0.28
568	4	7	6	1.02	101	5	0.27
1,432	6	12	5	1.27	99	5	0.26
3,448	8	15	5	1.34	92	4	0.25
5,272	10	21	5	2.04	95	4	0.25
8,248	12	24	5	2.39	93	4	0.25
9,172	14	28	5	2.80	100	5	0.28
9,640	16	33	6	3.69	182	6	0.29
9,832	18	37	7	4.55	212	7	0.31
19,492	20	42	5	4.78	147	6	0.28
29,908	30	59	5	6.51	109	6	0.28
39,796	50	102	6	11.73	112	5	0.27
39,608	100	195	8	27.42	180	7	0.30

way q is greater than 2^{160} for sizes of 192 or 224 bits for the field's order, since n is at most 20^4. Moreover, when the random method is used for the computation of a prime p in a k-bit finite field, values of p that are not in $[2^{k-1}, 2^k]$ ($k = 192$ or 224 in our experiments) are rejected.

The comparison of Cornacchia's algorithm and random method in the 192-bit finite field for various values of D and h is shown in Table 6. Let $\#p$ denote the number of primes that we had to try in order to find a solution (u, v) using Cornacchia's algorithm, and let $\#m$ be the number of orders m that we tried until a suitable one was found. Correspondingly, $\#p_r$ is the number of random pairs (u, v) that we had to try in order to find a prime p in the interval $[2^{191}, 2^{192}]$ with the random method, and $\#m_r$ is the number of orders m that we tried until we found a suitable one. $T(p,m)$ is the time for the computation of p and m using Cornacchia's algorithm, while $T(p_r,m_r)$ is the time needed by the random method. We observe that Cornacchia's algorithm is less efficient than the random method, with performance

Table 7 Timing estimations (in s) for the computation of p and m in the 192-bit finite field

D	h	Cornacchia's algorithm			Random method		
		$\#p$	$\#m$	$T(p,m)$	$\#p_r$	$\#m_r$	$T(p_r,m_r)$
296	10	20	10	3.70	286	7	0.31
724	10	20	9	3.44	122	6	0.27
1,268	10	20	6	2.29	99	5	0.25
3,412	10	20	6	2.20	101	5	0.25
5,272	10	21	5	2.04	95	4	0.25

that becomes worse as the degree h increases. On the other hand, the efficiency of the random method is similar for all values of D and h. The number of trials for order m are approximately the same regardless of the degree of the polynomial and the method used.

According to [6], we have to roughly try $2h$ primes before a solution can be found by Cornacchia's algorithm. This fact was verified by our experiments with surprising accuracy. For the random method many (usually more than 100) pairs of (u, v) must be tried in order to find a prime p for all values of D. This can turn to an advantage of Cornacchia's method for small values of h (even if its performance is still worse than the random method). The advantage is that a user can provide his EC cryptosystem with the primes one wishes to use within a set of "good" primes, like Mersenne primes, when the degree h is relatively small.

Tables 7 and 8 elaborate further on the use of Cornacchia's algorithm and the random method, respectively, for various values of D but for the same $h = 10$ in the 192-bit and 224-bit finite field. The main observation is that the time requirements for the construction of p and m decrease as D increases for both methods, following the number of orders m that we tried until we found a suitable one, in the case of Cornacchia's algorithm, and the number of pairs (u, v) that we had to try in order to find a prime p, in the case of random method.

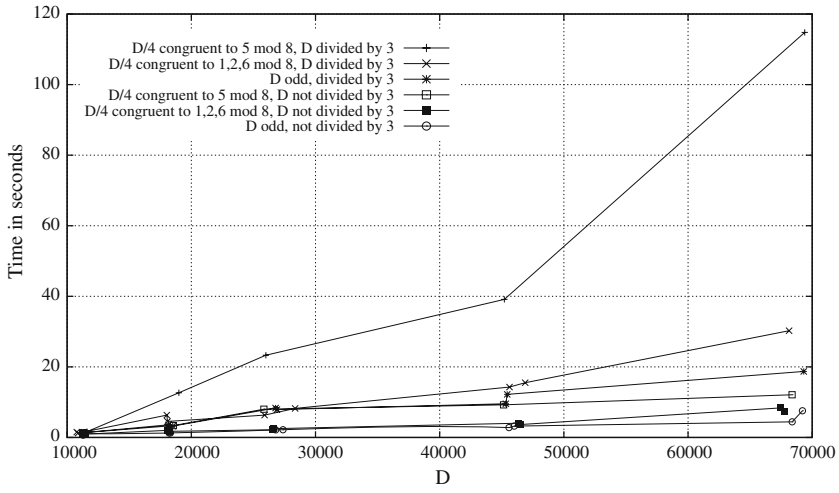

Fig. 5 Time for the construction of Weber polynomials

Table 8 Timing estimations (in s) for the computation of p and m in the 224-bit finite field

D	h	Cornacchia's algorithm			Random method		
		$\#p$	$\#m$	$T(p,m)$	$\#p_r$	$\#m_r$	$T(p_r,m_r)$
296	10	19	13	6.58	348	8	0.47
724	10	20	12	6.51	141	6	0.39
1,268	10	19	9	4.28	115	6	0.39
3,412	10	20	7	3.68	122	6	0.40
5,272	10	20	5	2.84	113	5	0.37

5.3 The other steps of the CM method

We finally turn to the efficiency of the other steps (Steps 5–7) of the CM variant (recall Sect. 3.1) and of our CM implementation in total. In Table 9 we report on the time requirements of all the steps of our CM variant for various values of D and h and we see where exactly the time is spent. We denote by $T[W]$ the time for the construction of the Weber polynomial, by $T(p,m)$ and $T(p_r,m_r)$ the time required to find a prime p and a suitable order m (Steps 3 and 4) using Cornacchia's algorithm and random method, respectively, by T_5 the time required for the computation of a root of the polynomial modulo p (Step 5) and by T_{67} the time required for the construction of the EC (Steps 6 and 7). T_{main} is the total time of the main phase (Steps 3–7) of our variant using Cornacchia's algorithm for Steps 3 and 4, and T^r_{main} is the total time when the random method is used for the computation of p and m. The Weber polynomials have been constructed off-line during the preprocessing phase.

We note that the most time consuming steps of the CM method are Step 5 (computation of a root of the polynomial) and Steps 3–4 when Cornacchia's algorithm is used. As it was expected, T_{67} does not change with D and h.

Table 9 Timing estimations (in s) of our CM variant in the 192-bit finite field

D	h	$T[W]$	$T(p,m)$	$T(p_r,m_r)$	T_5	T_{67}	T_{main}	T^r_{main}
232	2	0.03	0.63	0.28	0.01	0.32	0.96	0.61
568	4	0.05	1.02	0.27	0.04	0.33	1.39	0.64
1,432	6	0.07	1.27	0.26	0.09	0.33	1.69	0.68
3,448	8	0.10	1.34	0.25	0.14	0.35	1.83	0.74
5,272	10	0.13	2.04	0.25	0.21	0.38	2.63	0.84
8,248	12	0.16	2.39	0.25	0.32	0.31	3.02	0.88
9,172	14	0.18	2.80	0.28	0.41	0.33	3.54	1.02
9,640	16	0.19	3.69	0.29	0.51	0.39	4.59	1.18
9,832	18	0.21	4.55	0.31	0.76	0.35	5.66	1.42
19,492	20	0.27	4.78	0.28	1.22	0.30	6.30	1.80
29,908	30	0.61	6.51	0.28	1.77	0.40	8.68	2.45
39,796	50	2.14	11.73	0.27	6.11	0.39	18.23	6.77
39,608	100	2.74	27.42	0.30	23.45	0.35	51.23	24.10

6 Conclusions

We have presented a variant of the Complex Multiplication method for generating secure ECs. The variant uses Weber polynomials that can be used for the construction of ECs of suitable order. We have presented, 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 an experimental study comparing the construction efficiency of Weber and Hilbert polynomials for various discriminant values. We observed that for Weber polynomials there is a ranking among the values of D that is defined by its divisibility properties. We also investigated the time efficiency of the new Complex Multiplication variant under different implementations of a crucial step of the variant. We believe that our experimental results can be used as a guideline for the construction of EC cryptosystems, as the potential designer can have an estimate of the computation time as well as

the precision required before the actual implementation is accomplished.

Acknowledgments We would like to thank the anonymous referees for their valuable comments.

References

1. Argyroudis P.: NTRG ECC-LIB WINCE—a WinCE port of ECC-LIB, available at: http://www.ntrg.cs.tcd.ie/~argp/software/ntrg-ecc-lib-wince.html (2004)
2. Atkin, A.O.L., Morain, F.: Elliptic curves and primality proving. Math. Comput. **61**, 29–67 (1993)
3. Baier, H.: Efficient algorithms for generating elliptic curves over finite fields suitable for use in cryptography. PhD Thesis, Deptartment of Computer Science, Technical University of Darmstadt (2002)
4. Baier, H., Buchmann, J.: Efficient construction of cryptographically strong elliptic curves. In: Progress in Cryptology—INDOCRYPT 2000, LNCS, vol. 1977, pp. 191–202. Springer, Berlin Heidelberg New York (2000)
5. Berlekamp, E.R.: Factoring polynomials over large finite fields. Math Comput **24**, 713–735 (1970)
6. Blake, I., Seroussi, G., Smart, N.: Elliptic curves in cryptography. London Mathematical Society, Lecture Note Series 265. Cambridge University Press, Cambridge (1999)
7. Burton, D.: Elementary Number Theory, 4th edn. McGraw-Hill, New York (1998)
8. Cornacchia, G.: Su di un metodo per la risoluzione in numeri interi dell' equazione $\sum_{h=0}^{n} C_h x^{n-h} y^h = P$. Giornale di Matematiche di Battaglini **46**, 33–90 (1908)
9. Enge, A., Morain, F.: Comparing invariants for class fields of imaginary quadratic fields. In: Algorithmic Number Theory—-ANTS-V. Lecture Notes in Computer Science, vol. 2369, pp. 252–266. Springer, Berlin Heidelberg New York (2002)
10. Enge, A., Schertz, R.: Constructing elliptic curves from modular curves of positive genus. (preprint 2003)
11. Frey, G., Rück, H.G.: A remark concerning m-divisibility and the discrete logarithm problem in the divisor class group of curves. Math Comput **62**, 865–874 (1994)
12. GNU multiple precision library, 3.1.1. edn. Available at: http://www.swox.com/gmp (2000)
13. Gura, N., Eberle, H., Shantz, S.C.: Generic implementations of elliptic curve cryptography using partial reduction. In: Proceedings of the 9th ACM Conference on Computer and Communications Security—CCS'02, pp. 108–116
14. Hankerson, D., Menezes, A., Vanstone, S.: Guide to elliptic curve cryptography. Springer, Berlin Heidelberg New York (2004)
15. Herzberg, A., Jakobsson, M., Jarecki, S., Krawczyk, H., Yung, M.: Proactive public key and signature systems. In: Proceedings of the 4th ACM Conference on Computer and Communications Security—CCS'97, pp. 100–110
16. IEEE P1363/D13: Standard specifications for public-key cryptography, ballot draft. http://www.grouper.ieee.org/groups/1363/tradPK/draft.html (1999)
17. Kaltofen, E., Yui, N.: Explicit construction of the Hilbert class fields of imaginary quadratic fields by integer lattice reduction. Research Report 89-13, Renseelaer Polytechnic Institute (1989)
18. Kaltofen, E., Valente, T., Yui, N.: An improved Las Vegas primality test. In: Proceedings of the ACM-SIGSAM 1989 International Symposium on Symbolic and Algebraic Computation, pp. 26–33 (1989)
19. Konstantinou, E., Stamatiou, Y., Zaroliagis, C.: A software library for elliptic curve cryptography. In: Proceedings of the 10th European Symposium on Algorithms—ESA 2002 (Engineering and Applications Track). Lecture Notes in Computer Science, vol. 2461, pp. 625–637. Springer, Berlin Heidelberg New York (2002)
20. Konstantinou, E., Stamatiou, Y., Zaroliagis, C.: 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, pp. 333–348. Springer, Berlin Heidelberg New York (2002)
21. Konstantinou, E., Stamatiou, Y., Zaroliagis, C.: On the Use of Weber polynomials in elliptic curve cryptography. In: Public key infrastructure—EuroPKI 2004. Lecture Notes in Computer Science, vol. 3093, pp. 335–349. Springer, Berlin Heidelberg New York, (2003)
22. Lay, G.J., Zimmer, H.: Constructing elliptic curves with given group order over large finite fields. In: Algorithmic number theory—ANTS-I. Lecture Notes in Computer Science, vol. 877, pp. 250–263. Springer, Berlin Heidelberg New York (1994)
23. LiDIA: A library for computational number theory, Technical University of Darmstadt. Available from http://www.informatik.tu-darmstadt.de/TI/LiDIA/Welcome.html (2001)
24. Menezes, A.J., Okamoto, T., Vanstone, S.A.: Reducing elliptic curve logarithms to a finite field. IEEE Trans. Info. Theory **39**, 1639–1646 (1993)
25. Müller, V., Paulus, S.: On the generation of cryptographically strong elliptic curves (preprint 1997)
26. Niven, I., Zuckerman, H.S., Montgomery, H.L.: An introduction to the theory of numbers, 5th edn. Wiley, New York (1991)
27. Pohlig, G.C., Hellman, M.E.: An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance. IEEE Trans. Info. Theory **24**, 106–110 (1978)
28. Satoh, T., Araki, K.: Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves. Comm. Math. Univ. Sancti Pauli **47**, 81–91 (1998)
29. Savas, E., Schmidt, T.A., Koc, C.K.: Generating elliptic curves of prime order. In: Cryptographic hardware and embedded systems—CHES 2001. Lecture Notes in Computer Science, vol. 2162, pp. 145–161. Springer, Berlin Heidelberg New York (2001)
30. Schertz, R.: Weber's class invariants revisited. J. Théor. Nombres Bordeaux **14**, 1 (2002)
31. Semaev, I.A.: Evaluation of discrete logarithms on some elliptic curves. Math. Comput. **67**, 353–356 (1998)
32. Silverman, J.H.: The arithmetic of elliptic curves, GTM 106. Springer, Berlin Heidelberg New York (1986)
33. Smart, N.P.: The discrete logarithm problem on elliptic curves of trace one. J. Cryptogr. **12**, 193–196 (1999)
34. Spallek, A.-M.: Konstruktion einer elliptischen Kurve über einem endlichen Körper zu gegebener Punktegruppe. Master Thesis, Universität GH Essen (1992)
35. Valente, T.: A distributed approach to proving large numbers prime. Rensselaer Polytechnic Institute Troy, New York, Thesis (1992)
36. Washington, L.C.: Elliptic curves: number theory and cryptography. Chapman & Hall/CRC, Boca Raton (2003)
37. Weber, H.: Algebra III. Vieweg (1908)
38. Williams, P.: Available at: http://www.mindspring.com/~pate

Author's Biography

Elisavet Konstantinou holds a B.Sc. in Informatics from the University of Ioannina, a M.Sc. in Signal and Image Processing Systems, and a PhD in Theory and Applications of Elliptic Curve Cryptosystems from the University of Patras, Department of Computer Engineering and Informatics. She is currently a Lecturer in the Department of Information and Communication Systems Engineering, University of the Aegean. She was a research fellow in the R.A. Computer Technology Institute and participated in several EC funded projects (such as the IST projects ASPIS and FLAGS), as well as in secveral national projects (such as the HYPERGEN project and the IRAKLITOS Action supported by the Greek Ministry of Education through the EPEAEK II programme). She has published several papers in international conferences and journals. Her research interests include elliptic curves cryptosystems and generation of their parameters, public key cryptosystems, random number generation, algorithm engineering, and algebraic number theory. She is a member of the IEEE Computer Society.

Yannis C. Stamatiou was born in Volos in 1968. He holds a degree of Computer Engineering & Informatics, from the University of Patras and a Ph.D. from the same department. He is currently an Assistant Professor at the University of Ioannina, Mathematics Department, Greece, and a scientific consultant on security and cryptography issues of the Research and Academic Computer Technology Institute (RACTI). His scientific interests lie in the fields of security and cryptography as well as the study of threshold phenomena arising in computationally intractable problems. He is a member of ACM and IEEE.

Christos Zaroliagis received his PhD in computer science from the University of Patras, Greece, in 1991. He is currently an Associate Professor in the Department of Computer Engineering & Informatics, University of Patras, Greece, and a Senior Researcher of the R.A. Computer Technology Institute, Patras, Greece. He held previous positions at the Max-Planck-Institute für Informatik, Saarbrücken, Germany, and in the Department of Computer Science, King's College, University of London, UK. His current research interests include design and analysis of efficient algorithms and data structures, algorithm engineering, theory and applications of parallel and distributed computing, combinatorial optimization, large-scale optimization, robust and online optimization, and cryptography and information security. He has published more than 80 papers in major international journals and conferences. He is editor of the *Journal of Discrete Algorithms* (Elsevier), and he has served in more than 15 Program Committees of leading international computer science conferences. He has participated as a coordinator and/or key researcher in several EC-funded and national projects. He is a member of ACM, IEEE, and EATCS.