Efficient ephemeral elliptic curve cryptographic keys

Andrea Miele, Arjen K. Lenstra

EPFL, Lausanne, Switzerland

Abstract. We show how any pair of authenticated users can on-the-fly agree on an elliptic curve group that is unique to their communication session, unpredictable to outside observers, and secure against known attacks. Our proposal is suitable for deployment on constrained devices such as smartphones, allowing them to efficiently generate ephemeral parameters that are unique to any single cryptographic application such as symmetric key agreement. For such applications it thus offers an alternative to long term usage of standardized or otherwise pre-generated elliptic curve parameters, obtaining security against cryptographic attacks aimed at other users, and eliminating the need to trust elliptic curves generated by third parties.
Keywords: Elliptic curve cryptography, complex multiplication method

1 Introduction

Deployment of elliptic curve cryptography (ECC) [32, 40] is becoming more common. A variety of ECC parameters has been proposed or standardized [58, 16, 17, 5, 1, 39, 13, 9], with or without all kinds of properties that are felt to be desirable or undesirable, and as reviewed in Section 2. All these proposals and standards contain a fixed number of possible ECC parameter choices. This implies that many different users will have to share their choice, where either choice implies trust in the party responsible for its construction. Notwithstanding a variety of design methods intended to avoid trust issues (cf. [6]) and despite the fact that parameter sharing is generally accepted for discrete logarithm cryptosystems, recent allegations [52, 28] raise questions. As extensively discussed at the recent Workshop on Elliptic Curve Cryptography Standards [45], currently the main challenge in curve selection lies in re-establishing users' trust in ECC which vacillated after the above allegations were announced, and keep being followed by a continuing string of disconcerting information security related mishaps. Relying on choices made by others, either related to elliptic curve parameter selection or to any other personal choice related to information security, parameter sharing and long term usage of any type of cryptographic key material, may have to be reconsidered. In this paper we consider what can realistically be done if this reconsideration is taken to the extreme and trust in other parties' contributions is reduced to a minimum: never rely on choices made by others, avoid parameter sharing as much as possible, and refresh key material as often as feasible.

Specifically, we consider an approach that is diametrically different from current common practice, namely selection of *personalized*, *short-lived* ECC parameters. By personalized we mean that no party but the party or parties owning or directly involved in the usage of parameters should be responsible for their generation:

- for a certified public key, **only** the owner of the corresponding private key should be responsible for the selection of **all** underlying parameters;
- in the Diffie-Hellman protocol, as there is no a priori reason for the parties to trust each others' public key material other than for mutual authentication, **both** parties, and no other party, should be equally responsible for the construction of the group to be used in the key agreement phase.

Personalization excludes parameter choice interference by third parties with unknown and possibly contrary incentives. It also avoids the threats inherent in parameter sharing.

By short-lived, or *ephemeral*, we mean that parameters are refreshed (and possibly recertified) as often as feasible and permitted by their application; for the Diffie-Hellman protocol it means that a group is generated and used for just a single protocol execution and discarded after completion of the key agreement phase. Ephemeral parameters minimize the attack-window before the parameters are discarded. Attacks after use cannot be avoided for any type of public key system. But the least we can do is to avoid using parameters that may have been exposed to cryptanalysis for an unknown and possibly extended period of time before their usage.

In this paper we discuss existing methods for personalized, short-lived ECC parameter generation. Even with current technology, each end-user can in principle refresh and recertify his or her ECC parameters on a daily basis (cf. Section 2): "in principle" because user-friendly interfaces to the required software are not easily available to regular users. But it allows arbitrary, personalized choices – within the restrictions of ECC of course – in such a way that no other party can control or predict any of the newly selected parameters (including a curve parameterization and a finite field that together define an elliptic curve group, cf. below). Personalization isolates each user from attacks against other users, and using keys for a period of time that is as short as possible reduces the potential attack pay-off. Once personalized, short-lived ECC (public, private) key pairs are adopted at the end-user level, certifying parties may also rethink their sometimes decades-long key validities.

To satisfy the run time requirements of the Diffie-Hellman protocol, it should take at most a fraction of a second (jointly on two consumer-devices) to construct a personalized elliptic curve group suitable for the key agreement phase, that will be used for just that key agreement phase, and that will be discarded right after its usage – never to be used or even met again. In full generality this is not yet possible, as far as we know, and a subject of current research. However, for the moment the method from [34] can be used if one is willing to settle for partially personalized parameters: the finite field and thus the elliptic curve group cardinality are still fully personalized and unpredictable to any third party, but not more than eight choices are available for the Weierstrass equation used for the curve parameterization. Although the resulting parameters are not in compliance with the security criteria adopted by [9] and implied by [39], we point out that there is no indication whatsoever that either of these eight choices offers inadequate security: citing [9] "there is no evidence of serious problems". The choice is between being vulnerable to as yet unknown attacks – as virtually all cryptographic systems are – or being vulnerable to attacks aimed at others by sharing parameters, on top of trusting choices made by others. Given where the uncertainties lie these days, we opt for the former choice.

An issue that we are not paying attention to in this paper is the performance of the elliptic curve cryptosystem itself, once the parameters have been generated, or a comparison between the curves as generated here and the standardized ones. This is not because the issue is not of interest, but mostly because for either type of curve perfectly adequate runtimes can easily be achieved using generally available software. Also, our main point of concern in this paper is not performance optimization but minimization of trust in other parties.

After introductory sections on elliptic curves and their selection for ECC (Section 2) and complex multiplication (Section 3.1) we provide an explanation (in Section 3.2) how the "class number one" Weierstrass equations proposed in [34] were derived and how that same method generalizes to slightly larger class numbers. As a result we expand, also in Section 3.2, the table from [34] with eleven more Weierstrass equations, thereby more than doubling the number of equations available. In Section 3.3 we show how our methods can be further generalized, and why practical application of these ideas may not be worthwhile. In Section 4 we describe a new method for partially personalized ECC parameter generation that is substantially faster than the one from [34] and that also allows generation of *Montgomery friendly* primes and, at non-trivial overhead, of *twist-secure* curves. We demonstrate the effectiveness of our approach with an implementation on an Android Samsung Galaxy S4 smartphone. It generates a unique 128-bit secure elliptic curve group in about 50 milliseconds on average and thus allows efficient generation and ephemeral usage of such groups during Diffie-Hellman key agreement. Security issues (including the one mentioned above) are discussed in Section 5. In the concluding Section 6 we briefly discuss extension of our method to genus 2.

Our source code will be made available. As selecting ECC parameters on the fly adds more complexity to Diffie-Hellman key agreement, users should use the open-source code we will provide, after extensive testing and potential improvement, to minimize the probability of failure. Techniques to make the selection process more robust like additional sanity checks may be applied.

2 Preliminaries

Elliptic curves. We fix notation and recall some well known facts. For a finite field K of characteristic larger than 3, a pair $(a, b) \in K^2$ with $4a^3 + 27b^2 \neq 0$ defines an *elliptic curve* $E_{a,b} = E$ over K, to be thought of as the coefficients of the Weierstrass equation

$$y^2 = x^3 + ax + b.$$

The set of pairs $(x, y) \in K^2$ that satisfy this equation along with a *point at infinity* O is the *set of points* $E(K)$ of E over K. This set has the structure of an abelian group (with O acting as the identity element) and is thus also referred to as the *group of points* of E over K or simply the *elliptic curve group*. Traditionally, this group is written additively. See [53, Chapter III] for a more general and formal introduction to this material, including effective ways to perform the group operation in a constant number of operations in K.

For $g \in E(K)$ the *discrete logarithm* problem with respect to g is the problem to find, given $h \in \langle g \rangle$, an integer n such that $h = ng$. For properly chosen E, the fastest published methods to solve this problem require on the order of $\sqrt{q}$ operations in the group $E(K)$ (and thus in K), where q is the largest prime dividing the order of g. If $k \in \mathbf{Z}$ is such that $2^{k-1} \leq \sqrt{q} < 2^k$, the discrete logarithm problem in $E(K)$ is said to offer *k-bit security*.

With $K = \mathbf{F}_p$ the finite field of cardinality p for a prime $p > 3$, and a randomly chosen elliptic curve E over $\mathbf{F}_p$, the order $\#E(\mathbf{F}_p)$ behaves as a random integer close to $p + 1$ (see [38] for the precise statement) with $|\#E(\mathbf{F}_p) - p - 1| \leq 2\sqrt{p}$. For ECC at k-bit security level it therefore suffices to select a $2k$-bit prime p and an elliptic curve E for which $\#E(\mathbf{F}_p)$ is prime (or *almost prime*, i.e., up to an ℓ-bit factor, at an $\frac{\ell}{2}$-bit security loss, for a small ℓ), and to rely on the alleged hardness of the discrete logarithm with respect to a generator (of a large prime order subgroup) of $E(\mathbf{F}_p)$. How suitable p and E should be constructed is the subject of this paper. For reasons adequately argued elsewhere (cf. [7, Section 4.2]), for cryptographic purposes we explicitly exclude from consideration elliptic curves over extension fields.

Depending on the application, *twist-security* may have to be enforced as well: not just $\#E_{a,b}(\mathbf{F}_p) = p + 1 - t$ must be (almost) prime (where $|t| \leq 2\sqrt{p}$), but also $p + 1 + t$ must be (almost) prime. This number $p + 1 + t$ is the cardinality of the group of points of a *(quadratic) twist* $\widetilde{E} = E_{r^2a,r^3b}$ of $E = E_{a,b}$, where r is any non-square in $\mathbf{F}_p$.

Generating elliptic curves for ECC. The direct approach is to first select, for k-bit security, a random $2k$-bit prime p and then to randomly select elliptic curves E over $\mathbf{F}_p$ until $\#E(\mathbf{F}_p)$ is (almost) prime. Because of the random behavior of $\#E(\mathbf{F}_p)$, the expected number of elliptic curves to be selected is linear in k and can be halved by considering $\#\widetilde{E}(\mathbf{F}_p)$ as well (and replacing E by $\widetilde{E}$ if a prime $\#\widetilde{E}(\mathbf{F}_p)$ is found first). Because $\#E(\mathbf{F}_p)$ can be computed in time polynomial in k using the Schoof-Elkies-Atkin algorithm (SEA) [50], the overall expected effort is polynomial in k. This method is referred to as *SEA-based ECC parameter selection*. Generating twist-secure curves in this way is slower by a factor linear in k.

Table 1 lists actual ECC parameter generation times, for $k \in \{80, 112, 128\}$. Using primes p with special properties (such as being *Montgomery friendly*, i.e., $p \equiv \pm 1 \bmod 2^{32}$ or 2^{64}) has little or no influence on the timings. For comparison, key generation times are included for traditional non-ECC asymmetric cryptosystems at approximately the same security levels. The ECC parameter generation timings – in particular the twist-secure ones – may explain why the direct approach to ECC parameter generation is not considered to be a method that is suitable for the general public. Although this may have to be reconsidered and end-users could in principle – given appropriate

Table 1: Timings of cryptographic parameter generation on a single 2.7GHz Intel Core i7-3820QM, averaged over 100 parameter sets, for prime elliptic curve group orders and 80-bit, 112-bit, and 128-bit security. For RSA these security levels correspond, roughly but close enough, to 1024-bit, 2048-bit, and 3072-bit composite moduli, for DSA to 1024-bit, 2048-bit, and 3072-bit prime fields with 160-bit, 224-bit, and 256-bit prime order subgroups of the multiplicative group, respectively. The timings in the last two rows have been obtained using a C implementation of the method we present in this paper.

	80-bit security	112-bit security	128-bit security
RSA	80 milliseconds	0.8 seconds	2.5 seconds
DSA	0.2 seconds	1.8 seconds	8 seconds
random ECC (MAGMA)	12 seconds	47 seconds	120 seconds
same, but twist-secure	6 minutes	37 minutes	83 minutes
low discriminant curves over random prime fields	2 milliseconds	5 milliseconds	6 milliseconds
same, but twist secure	10 milliseconds	27 milliseconds	45 milliseconds

software – (re)generate their ECC parameters and key material on a daily basis, the current state-of-the-art of the direct approach does not allow fast enough on-the-fly ECC parameter generation in the course of the Diffie-Hellman protocol. Table 1 also lists timings obtained using the method presented in this paper.

Pre-selected elliptic curves. We briefly discuss some of the elliptic curves that have been proposed or standardized for ECC. As mentioned above, we do not consider any of the proposals that involve extension fields (most commonly of characteristic two).

With two notable exceptions that focus on $\approx$ 125-bit security, most proposals offer a range of security levels. Although 90-bit security [11] is still adequate, it is unclear why parameters that offer less than 112-bit security (the minimal security level recommended by NIST [43]) should currently still be considered, given that the $\approx$ 125-bit security proposals offer excellent performance. With 128-bit security more than sufficient for the foreseeable future, it is not clear either what purpose is served by higher security levels, other than catering to "TOP SECRET" 192-bit security from [44]. In this context it is interesting to note that 256-bit AES, also prescribed by [44] for "TOP SECRET", was introduced only to still have a 128-bit secure symmetric cipher in the post-quantum world (cf. [55]), and that 192-bit security was merely a side-effect that resulted from the calculation $\frac{128+256}{2}$ (cf. [55]). In that world ECC is obsolete anyhow.

In [16] eleven different primes are given, all of a special form that makes modular arithmetic somewhat easier than for generic primes of the same size, and ranging from 112 to 521 bits. They are used to define fifteen elliptic curves of eight security levels from 56-bit to 260-bit, four with $a = 0$ and b small positive ("Koblitz curves"), the other eleven "verifiably at random" but nine of which with $a = p - 3$, and all except two with prime group order (two with cofactor 4 at security levels 56 and 64). Verifiability means that a standard pseudo random number generator when seeded with a value that is provided, results in the parameters a (if $a \neq p-3$) and b. The arbitrary and non-uniform choice for the seeds, however, does not exclude the possibility that parameters were aimed for that have properties that are unknown to the users. This could easily have been avoided, but maybe this was not a concern at the time when these curves were generated (i.e., before the fall of the year 2000). Neither was twist-security a design criterion back then; indeed some curves have poor twist security (particularly so the 96-bit secure curve), whereas the single 192-bit secure curve is perfectly twist-secure. If one is willing to use pre-selected curves, there does not seem to be a valid argument, at this point in time, to settle for anything less than optimal twist-security: for general applications they are arguably preferable and their only disadvantage is that they are relatively hard to find, but this is done just once and thus no concern. Surprisingly, in the latest (2013) update of the federal information processing standards ("FIPS") for digital signatures (cf. [58]) only two out of five twists of the curves at security level 96 or higher and with $a = p - 3$ (all "recommended for federal government use") satisfy the group-cardinality margins allowed by [58].

The use of special primes was understandable back in 2000, because at that time ECC was relatively slow and any method to boost its performance was welcome, if not crucial, for the survival of ECC. The trend to use special primes persists to the present day, in a seemingly unending competition for the fastest ECC system. However, these days also regular primes without any special form offer more than adequate ECC performance. This is reflected in one of the proposals discussed below.

The proposals [5] and [7] each contain a single twist-secure curve of (approximately) 125-bit security, possibly based on the sensible argument that there is no need to settle for less if the performance is adequate, and no need to require more (cf. above). All choices are deterministic given the design criteria, easily verifiable, and have indeed been verified. For instance, the finite field in [5] is defined by the largest 255-bit prime, where the choice 255 is arguably optimal given the clever field arithmetic. The curve equation is the "first" one given the computationally advantageous curve parameterization and various requirements on the group orders. Another, but similarly rigidly observed, design criterion (beyond the scope of the present paper) underlies the proposal in [7].

The curves from [5] and [7] are perfectly adequate from a security-level and design point of view. If the issue of sharing pre-selected curves is disregarded they should suffice to cater to all conceivable cryptographic applications (with the exception of pairing-based cryptography, cf. below). Nevertheless, their design approach triggered two follow-up papers by others. In [1] they are complemented with their counterparts at approximate security levels 112, 192, and 256. In [13] the scope of [7] is broadened by allowing more curve parameterizations and more types of special primes, while handling exceptions more strictly. This leads to eight new twist-secure curves of (approximately) 128-bit security, in addition to eight and ten twist-secure curves at approximate security levels 192 and 256, respectively.

The seven Brainpool curves [39] at seven security levels from 80-bit to 256-bit revert to the verifiably pseudo random approach from [16], while improving it and thereby making it harder to target specific curve properties (but see [6]). The primes p have no special form (except that they are 3 mod 4) and are deterministically determined as a function of a seed that is chosen in a uniform manner based on the binary expansion of $\pi = 3.14159\ldots$. The curves use $a = p - 3$ and a quadratic non-residue $b \in \mathbf{F}_p$ (deterministically determined as a function of a different seed, similarly generated based on $e = 2.71828\ldots$) for which the orders of the groups of the curve and its twist are both prime. As an additional precaution, curves are required to satisfy $\#E_{a,b}(\mathbf{F}_p) < p$. In [37] it is shown how usage of constants such as π and e can be avoided while still allowing verifiable and trustworthy random parameter generation.

The SafeCurves project [9] specifies a set of criteria to analyze elliptic curve parameters aiming to ensure the security of ECC and not just the security (i.e., the difficulty) of the elliptic curve discrete logarithm problem, and analyzes many proposed parameter choices, including many of those presented above, with respect to those criteria. This effort represents a step forward towards better security for ECC. For this paper it is relevant to mention that the SafeCurves security criteria include the requirement that the complex-multiplication field discriminant (cf. below) must be larger than 2^{100} in absolute value. Aside from the lack of argumentation for the bound, this requirement seems to be unnecessarily severe (and considerably larger than the rough 2^{40} requirement implied by [39]), not just because it is not supported by theoretical evidence, but also because the requirement cannot be met by pairing-based cryptography, considered by many as a legitimate and secure application of elliptic curves. On the other hand, [9] does not express concerns about the trust problem inherent in the usage of (shared) parameters pre-selected by third parties.

Attacking multiple keys. We conclude this section with a brief summary of results concerning the security of multiple instances of the "same" asymmetric cryptographic system. Early successes cannot be expected, or are sufficiently unlikely (third case).

1. *Multiple RSA moduli of the same size.* It is shown in [19, Section 4] that after a costly size-specific precomputation (far exceeding the computation and storage cost of an individual factoring effort), any RSA modulus of the proper size can be factored at cost substantially

less than its individual factoring effort. This is not a consequence of key-sharing (as RSA moduli should not be shared), it is a consequence of the number field sieve method for integer factorization [35].

2. *Multiple discrete logarithms all in the same multiplicative group of a prime field.* Finding a single discrete logarithm in the multiplicative group of a finite field is about as hard as finding any number of discrete logarithms in the same multiplicative group. Sharing a group is common (cf. DSA), but once a single discrete logarithm has been solved, subsequent ones in the same group are relatively easy.
3. *Multiple discrete logarithms all in the same elliptic curve group.* Solving a single discrete logarithm problem takes on the order of $\sqrt{q}$ operations, if the group has prime order q, and solving k discrete logarithm problems takes effort $\sqrt{kq}$. Thus, the average effort is reduced for each subsequent key that uses the same group.
4. *Multiple discrete logarithms in as many distinct, independent groups.* Solving k distinct discrete logarithm problems in k groups that have no relation to each other requires in general solving k independent problems. With the proper choice of groups, no savings can be obtained.

The final two cases most concern us in this paper. In the third case, with k users, an overall attack effort $\sqrt{kq}$ leads to an average attack effort per user of "just" $\sqrt{q/k}$. This may look disconcerting, but if q is properly chosen in such a way that effort $\sqrt{q}$ is infeasible to begin with, there is arguably nothing to be concerned about. Compared to the rather common second case (i.e., shared DSA parameters), the situation is actually quite a bit better. Nevertheless, existing users cannot prevent that new users may considerably affect the attack incentives. In the final case such considerations are of no concern. However, given the figures from Table 1, realizing the final case for ECC with randomly chosen parameters is not feasible yet for all applications. The next best approach that we are aware of is further explored below.

3 Special cases of the complex multiplication method

Our approach is based on and extends [34]. It may be regarded as a special case, or a short-cut, of the well known *complex multiplication* (CM) method of which many variants have been published and which appears under a variety of names in the literature (such as "Atkin-Morain" method). As no explanation is provided in [34], we first sketch one approach to the CM method and describe how it leads to the method from [34]. We then use this description to get a more general method, and indicate how further generalizations can be obtained.

3.1 The CM method

We refer to [3, Chapter 18], [48], and the references therein for all details of the method sketched here. In the SEA-based ECC parameter selection described in Section 2 one selects a prime field $\mathbf{F}_p$ and then keeps selecting elliptic curves over $\mathbf{F}_p$ until the order of the elliptic curve group has a desirable property. Checking the order is relatively cumbersome, making this type of ECC parameter selection a slow process. Roughly speaking, the CM method switches around the order of some of the above steps, making the process much faster at the expense of a much smaller variety of resulting elliptic curves: first primes p are selected until a trivial to compute function of p satisfies a desirable property, and only then an elliptic curve over $\mathbf{F}_p$ is determined that satisfies one's needs.

The standard CM method works as follows. Let $d \neq 1, 3$ be a square-free positive integer and let $H_d(X)$ be the Hilbert class polynomial[1] of the imaginary quadratic field $\mathbf{Q}(\sqrt{-d})$. If $d \equiv 3 \bmod 4$ let $m = 4$ and $s = 1$, else let $m = 1$ and $s = 2$. Find integers u, v such that $u^2 + dv^2$ equals mp for a suitably large prime p and such that $p + 1 \pm su$ satisfies the desired property (such as one of $p + 1 \pm su$ prime, or both prime for perfect twist security). Compute a root j of $H_d(X)$

[1] Obviously, we could have used Weber polynomials instead. Here we explain and generalize the method from [34] and therefore use Hilbert polynomials because those were the ones used in that paper.

modulo p, then the pair $\left(\frac{-27j}{4(j-12^3)}, \frac{27j}{4(j-12^3)}\right) \in \mathbf{F}_p^2$ defines an elliptic curve E over $\mathbf{F}_p$ such that $\#E(\mathbf{F}_p) = p+1 \pm su$ (and $\#\widetilde{E}(\mathbf{F}_p) = p+1 \mp su$). Finally, use scalar multiplications with a random element of $E(\mathbf{F}_p)$ to resolve the ambiguity. For $d \equiv 3 \bmod 4$ the case $u = 1$ should be excluded because it leads to anomalous curves.

The method requires access to a table of Hilbert class polynomials or their on-the-fly computation. Either way, this implies that only relatively small d-values can be used, thereby limiting the resulting elliptic curves to those for which the "complex-multiplication field discriminant" (namely, d) is small. The degree of $H_d(X)$ is the class number h_{-d} of $\mathbf{Q}(\sqrt{-d})$. Because $h_{-d} = 1$ precisely for $d \in \{1, 2, 3, 7, 11, 19, 43, 67, 163\}$ (assuming square-freeness), for those d-values the root computation and derivation of the elliptic curve become a straightforward one-time precomputation that is independent of the p-values that may be used. This is what is exploited in [34], as further explained, and extended to other d-values for which h_{-d} is small, in the remainder of this section.

3.2 The CM method for class numbers at most three

In [34] a further simplification was used to avoid the ambiguity in $p + 1 \pm u$. Here we follow the description from [56, Theorem 1], restricting ourselves to $d > 1$ with $\gcd(d, 6) = 1$, and leaving $d \in \{3, 8\}$ from [34] as special cases. We assume that $d \equiv 3 \bmod 4$ and aim for primes $p \equiv 3 \bmod 4$ to facilitate square root computation in $\mathbf{F}_p$. It follows that $\left(\frac{-1}{p}\right) = -1$.

Let $H_d(X)$ be as in Section 3.1. If $d \equiv 3 \bmod 8$ let $s = 1$, else let $s = -1$. As above, find integers $u > 1, v$ such that $u^2 + dv^2$ equals $4p$ for a (large) prime $p \equiv 3 \bmod 4$ for which the numbers $p + 1 \pm u$ are (almost) prime, and for which

$$a = 27d\sqrt[3]{j} \quad \text{and} \quad b = 54sd\sqrt{d(12^3 - j)}$$

are well-defined in $\mathbf{F}_p$, where j is a root of $H_d(X)$ modulo p. Then for any non-zero $c \in \mathbf{F}_p$, the pair $(c^4a, c^6b) \in \mathbf{F}_p^2$ defines an elliptic curve E over $\mathbf{F}_p$ such that $\#E(\mathbf{F}_p) = p + 1 - \left(\frac{2u}{d}\right)u$ (and $\#\widetilde{E}(\mathbf{F}_p) = p + 1 + \left(\frac{2u}{d}\right)u$).

As an example, let $d = 7$, so $s = -1$. The Hilbert class polynomial $H_7(X)$ of $\mathbf{Q}(\sqrt{-7})$ equals $X + 15^3$, which leads to $j = -15^3$, $a = -3^4 \cdot 5 \cdot 7$, and $b = -54 \cdot 7\sqrt{7(12^3 + 15^3)} = -2 \cdot 3^6 \cdot 7^2$. With $c = \frac{1}{3}$ we find that the pair $(-35, -98)$ defines an elliptic curve E over any prime field $\mathbf{F}_p$ with $4p = u^2 + 7v^2$ and that $\#E(\mathbf{F}_p) = p + 1 - \left(\frac{2u}{7}\right)u$.

Similarly, $H_{11}(X) = X + 2^{15}$ for $d = 11$. With $s = 1$ this leads to $j = -2^{15}$, $a = -2^5 \cdot 2^3 \cdot 11 = -9504$, and $b = 2 \cdot 3^3 \cdot 11\sqrt{11(12^3 + 2^{15})} = 365904$. For any $p \equiv 3 \bmod 4$ the pair $(-9504, 365904)$ defines an elliptic curve E over $\mathbf{F}_p$ for which $\#E(\mathbf{F}_p) = p+1-\left(\frac{2u}{11}\right)u$, where $4p = u^2+11v^2$. This is the twist of the curve for $d = 11$ in [34].

The elliptic curves corresponding to the four d-values with $h_{-d} = 1$ and $d > 11$ are derived in a similar way, and are listed in Table 2. The two remaining cases with $h_{-d} = 1$ listed in Table 2 are dealt with as described in [2, Theorem 8.2] for $d = 3$ and [47] for $d = 8$.

For $d = 91$, the class number h_{-91} of $\mathbf{Q}(\sqrt{-91})$ equals two and $H_{91}(X) = X^2 + 2^{17} \cdot 3^3 \cdot 5 \cdot 227 \cdot 2579X - 2^{30} \cdot 3^6 \cdot 17^3$ has root $j = \left(-2^4 \cdot 3(227 + 3^2 \cdot 7\sqrt{13})\right)^3$. It follows that $a = -2^4 \cdot 3^4 \cdot 7 \cdot 13(227 + 3^2 \cdot 7\sqrt{13})$ and $b = 2^4 \cdot 3^6 \cdot 7^2 \cdot 11 \cdot 13(13 \cdot 71 + 2^8\sqrt{13})$ so that with $c = \frac{1}{3}$ we find that the pair $(-330512 - 91728\sqrt{13}, 103479376 + 28700672\sqrt{13})$ defines an elliptic curve E over any prime field $\mathbf{F}_p$ with $p \equiv 3 \bmod 4$ and $\left(\frac{13}{p}\right) = 1$, and that $\#E(\mathbf{F}_p) = p + 1 - \left(\frac{2u}{91}\right)u$ where $4p = u^2 + 91v^2$.

Table 2 lists nine more d-values for which $h_{-d} = 2$, all with $d \equiv 3 \bmod 4$: for those with $\gcd(d, 6) = 1$ the construction of the elliptic curve goes as above for $d = 91$, the other three (all with $\gcd(d, 6) = 3$) are handled as shown in [30]. The other d-values for which $h_{-d} = 2$ also have $\gcd(d, 6) \neq 1$ and were not considered (but see [30]). The example for $h_{-d} = 3$ in the last row of Table 2 was taken from [30].

Table 2: Elliptic curves for fast ECC parameter selection. Each row contains a value d, the class number h_{-d} of the imaginary quadratic field $\mathbf{Q}(\sqrt{-d})$ with discriminant $-d$, the root used (commonly referred to as the j-invariant), the elliptic curve $E = E_{a,b}$, the constraints on the prime p and the values u and v, the value s such that $\#E(\mathbf{F}_p) = p + 1 - su$, and with γ and $\tilde{\gamma}$ denoting fixed factors of $\#E(\mathbf{F}_p)$ and $\#\widetilde{E}(\mathbf{F}_p)$, respectively.

h_{-d}	d	j-invariant	a, b	$p, u, v \in \mathbf{Z}_{>0}$	s	$\{\gamma\} \cup \{\tilde{\gamma}\}$
1	3	0	$0, 16$	$u^2 + 3v^2 = 4p$, $p \equiv 1 \bmod 3$, $u \equiv 1 \bmod 3$, $v \equiv 0 \bmod 3$	-1	$\{1, 9\}$
	8	20^3	$-270, -1512$	$u^2 + 2v^2 = p$, $u \equiv 1 \bmod 4$ if $p \equiv 3 \bmod 16$, $u \equiv 3 \bmod 4$ if $p \equiv 11 \bmod 16$	2	$\{2\}$
	7	-15^3	$-35, -98$	$u^2 + dv^2 = 4p$, $u > 1$	$\left(\frac{2u}{d}\right)$	$\{8\}$
	11	-32^3	$-9504, 365904$			$\{1, 9\}$
	19	-96^3	$-608, 5776$			$\{1\}$
	43	-960^3	$-13760, 621264$			
	67	-5280^3	$-117920, 15585808$			
	163	-640320^3	$-34790720, 78984748304$			
2	91	$-48^3(227 + 63\sqrt{13})^3$	$-330512 - 91728\sqrt{13}$, $103479376 + 28700672\sqrt{13}$			
	115	$-48^3(785 + 351\sqrt{5})^3$	$-1444400 - 645840\sqrt{5}$, $944794000 + 422522880\sqrt{5}$			
	187	$-240^3(3451 + 837\sqrt{17})^3$	$-51626960 - 12521520\sqrt{17}$, $+201921077072 + 48973056000\sqrt{17}$			
	235	$-528^3(8875 + 3969\sqrt{5})^3$	$-367070000 - 164157840\sqrt{5}$, $3828113058000 + 1711984189440\sqrt{5}$			
	403	$-240^3(2809615 + 779247\sqrt{13})^3$	$-90581987600 - 25122923280\sqrt{13}$, $1399216(10605743499 + 2941504000\sqrt{13})$			
	427	$-5280^3(236674 + 30303\sqrt{61})^3$	$-177865244480 - 22773310560\sqrt{61}$, $1099951(37121542375 + 4752926464\sqrt{61})$			
	51	$-48^3(4 + \sqrt{17})^2(5 + \sqrt{17})^3$	$-245616 - 59568\sqrt{17}$, $66257296 + 16069760\sqrt{17}$			$\{1, 3\}$
	123	$-480^3(32 + 5\sqrt{41})^2(8 + \sqrt{41})^3$	$-580796160 - 90705120\sqrt{41}$, $7619012947280 + 1189889913856\sqrt{41}$			
	267	$-240^3(500 + 53\sqrt{89})^2(625 + 53\sqrt{89})^3$	$-12015034710000 - 1273591132080\sqrt{89}$, $9968(2274273163768531 + 241072473215000\sqrt{89})$			
	35	$-16^3(15 + 7\sqrt{5})^3$	$-226800 - 105840\sqrt{5}$, $60858000 + 27095040\sqrt{5}$			$\{1, 9\}$
3	243	$-160^3(151022371885959 + 104713064226304\sqrt[3]{3} - 72603983653110\sqrt[3]{9})$	$-1560 + 720\sqrt[3]{9}$, $32258 - 11124\sqrt[3]{3} - 7704\sqrt[3]{9}$		$(\frac{-2\alpha}{p})(\frac{2u}{243})$ $\alpha = 2 - \sqrt[3]{9}$	

3.3 The CM method for larger class numbers

In this section we give three examples to illustrate how larger class numbers may be dealt with, still using the approach from Section 3.2. For each applicable d with $h_{-d} < 5$ a straightforward (but possibly cumbersome) one-time precomputation suffices to express one of the roots of $H_d(X)$ in radicals as a function of the coefficients of $H_d(X)$, and to restrict to primes p for which the root exists in $\mathbf{F}_p$. For larger h_{-d} there are in principle two obvious approaches (other possibilities exist, but we do not explore them here). One approach would be to exploit the solvability by radicals of the Hilbert class polynomial [29] for any d, to carry out the corresponding one-time root calculation, and to restrict, as usual, to primes modulo which a root exists. The other approach is to look up $H_d(X)$ for some appropriate d, to search for a prime p such that $H_d(X)$ has a root modulo p, and to determine it. In our application, the precomputation approach leads to relatively lightweight online calculations, which for the last approach quickly become more involved. We give examples for all three cases, with run times obtained on a 2.7GHz Intel Core i7-3820QM.

For $d = 203$ we have $h_{-203} = 4$ and $H_{203}(X) = X^4 + 2^{18} \cdot 3 \cdot 5^3 \cdot 739 \cdot 378577789X^3 - 2^{30} \cdot 5^6 \cdot 17 \cdot 1499 \cdot 194261303X^2 + 2^{54} \cdot 5^9 \cdot 11^6 \cdot 4021X + 2^{66} \cdot 5^{12} \cdot 11^6$ with root $-2^{14} \cdot 5^3 j'$ where

$$j' = 3357227832852 + 623421557759\sqrt{29} + 3367\sqrt{29(68565775894279681 + 12732344942060216\sqrt{29})}.$$

This precomputation takes an insignificant amount of time for any polynomial of degree at most four. With $c = 2^4 \cdot 3^3 \cdot 203$ it follows that the pair $\big(-5c\sqrt[3]{4j'}, c\sqrt{203(3^3 + 2^8 \cdot 5^3 j')}\big)$ defines an elliptic curve E over any prime field $\mathbf{F}_p$ that contains the various roots, and that $\#E(\mathbf{F}_p) = p + 1 - \big(\frac{2u}{203}\big)u$ where $4p = u^2 + 203v^2$. The online calculation can be done very quickly if the choice of p is restricted to primes for which square and cube roots can be computed using exponentiations modulo p.

As an example of the second approach, for $d = 47$ the polynomial $H_{47}(X)$ has degree five and root $25j'$, with the following expression by radicals for j':

$$13^3(7453991996007968795256512 - 2406037696832339815\sqrt{5} + A(4089143609023741 6B - 2809533607727924271200481090552 11\sqrt{5}/B))/(2^{3/5}C)$$

$$-13(5364746311921861372 - 856800988085\sqrt{5} - A(29162309591B - 13500974536508710980159626 4\sqrt{5}))/(2C^2)^{1/5}$$

$$+(3861085845907 - 1237935\sqrt{5})/(2 \cdot 13^3 C^{1/5}) - 18062673 + 13C^{1/5}/2^{2/5},$$

where

$$A = \frac{67206667}{827296299281}, \quad B = \sqrt{47(119957963395745 + 21781710063898\sqrt{5})}$$

and

$$C = -207137462812842515631270898815 29 + 16655517449486339268909175\sqrt{5} - \frac{D}{B}$$

for

$$D = 5^2 \cdot 11^2 \cdot 19 \cdot 23 \cdot 29 \cdot 31 \cdot 41 \cdot 47\big(2069683334124917088 47 - 46149532702509158373845\sqrt{5}\big).$$

This one-time precomputation took 0.005 seconds (using Maple 18). Elliptic curves and group orders follow easily, for properly chosen primes. In principle such root-expressions can be tabulated for any list of d-values one sees fit, but obtaining them, in general and for higher degrees, may be challenging.

As an example of the final approach mentioned above, for $d = 5923$ the polynomial $H_{5923}(X)$ has degree seven and equals

$$X^7 + 2^{15} \cdot 3^3 \cdot 5^3 \cdot 7 \cdot 31 \cdot 127 \cdot 2429520931 \cdot 13623868977157825621597249025760734749708584156092521957286388166296025747607409463 7X^6$$

$$-2^{30} \cdot 3^7 \cdot 5^6 \cdot 7 \cdot 62983 \cdot 1112240226499 \cdot 19292428007338985647320491911265071 \cdot 17155665707622469968593441685105265307077 7X^5$$

$$+2^{45} \cdot 3^9 \cdot 5^9 \cdot 7 \cdot 53 \cdot 97 \cdot 769 \cdot 259381 \cdot 4437462560116423 \cdot 97604219520630586719251956183 \cdot 27147567165140472264577022190878351X^4$$

$$-2^{60} \cdot 3^{12} \cdot 5^12 \cdot 7 \cdot 31 \cdot 99208777 \cdot 34069172420656302782993334479869 \cdot 21158190059019493731155731639427604962214247 93X^3$$

$$+2^{75} \cdot 3^{16} \cdot 5^{15} \cdot 7 \cdot 11^3 \cdot 10477 \cdot 47581 \cdot 240853 \cdot 104531840353 \cdot 10353927562807 \cdot 3553027351769487927227534889885666212883 1X^2$$

$$-2^{90} \cdot 3^{18} \cdot 5^{18} \cdot 7 \cdot 11^6 \cdot 47^3 \cdot 727 \cdot 7603931 \cdot 88452227997949 \cdot 17493070743470883052636283664191993115899 57X$$

$$+(2^{35} \cdot 3^7 \cdot 5^7 \cdot 11^3 \cdot 17 \cdot 23 \cdot 41 \cdot 47^2 \cdot 71 \cdot 593 \cdot 659 \cdot 1103 \cdot 1109)^3.$$

Given $H_{5923}(X)$ and 128-bit security, we look for 123-bit integers u and v such that $4p = u^2 + 5923v^2$ for a prime p for which $H_{5923}(X)$ has a root j modulo p and such that $\sqrt[3]{j}$ and $\sqrt{j}$ exist in $\mathbf{F}_p$ and can easily be calculated. For the present case it took 0.11 seconds (using Mathematica 9) to find

$$u = 979895489652329742612225722037963658 4,$$

$$v = 679415845702168995816816244342227177 4$$

which leads to the 256-bit prime

$$p = 68376297247017003283970261221870401697343820120616991149309517708508634100051$$

and

$$j = 54243655991109505677097612140273606931478183421749872324499965496758684433 12.$$

Because $p \equiv 2 \bmod 3$ all elements of $\mathbf{F}_p$ have a cube root (in particular $\sqrt[3]{j} = j^{\frac{2p-1}{3}} \bmod p$), $\left(\frac{j}{p}\right) = 1$ and $p \equiv 3 \bmod 4$. The elliptic curve and group order follow in the customary fashion.

From our results and run times it is clear that none of these approaches (one-time root precomputations, or online root calculation) is compatible with the requirements on the class number (at least 10^6 in [39]) or the discriminant (at least 2^{100} in [9]). In the remainder of this paper we focus on the approach from Section 3.2. Our approach thus does not comply with the class number or discriminant requirements from [9, 39], security requirements that are, as far as we know, not supported by published evidence.

4 Ephemeral ECC parameter generation

We describe how to use Table 2 to online generate ephemeral ECC parameters, improving the speed of the search for a prime p and curve E over $\mathbf{F}_p$ compared to the method from [34, Section 3.2], and while allowing an additional security requirement to the ones from [34] (without explicitly mentioning the ones already in place in [34]; refer to Section 5 for details). In the first place, on top of the trivial modifications to handle the extended table and determination of a base point as mentioned in [34, Section 3.6], we introduce the following additional search criteria:

1. *Efficiency considerations.*
   (a) *Montgomery friendly modulus.* The prime p may be chosen as -1 modulo 2^{64} or modulo 2^{32} to allow somewhat faster modular arithmetic.
   (b) *Conversion friendly curve.* A small positive factor f may be prescribed that must divide $\#E(\mathbf{F}_p)$ (such as for instance $f = 4$ to allow conversion to a Montgomery curve).
2. *Twist security.* Writing $\#E(\mathbf{F}_p) = fcq$ and $\#\widetilde{E}(\mathbf{F}_p) = \tilde{c}\tilde{q}$, with $f \in \mathbf{Z}_{>0}$ as above, cofactors $c, \tilde{c} \in \mathbf{Z}_{>0}$, and primes q and $\tilde{q}$, independent upper bounds ℓ and $\tilde{\ell}$ on the total security loss may be specified such that $fc < 2^{\ell}$ and $\tilde{c} < 2^{\tilde{\ell}}$. The roles of E and $\widetilde{E}$ may be reversed to meet these requirements faster (with f always a factor of the "new" $\#E(\mathbf{F}_p)$, which is automatically the case if $p \equiv 3 \bmod 4$ and $f = 4$).

These new requirements still allow a search as in [34, Section 3.2] where, based on external parameters and a random value, an initial pair (u_0, v_0) is chosen and the pairs $(u, v) \in \{(u_0, v_0 + i) : i \in [0, 255]\}$ are inspected on a one-by-one basis for each of the eight rows of [34, Table 1] until a pair is found that corresponds to a satisfactory p and E. If the search is unsuccessful (after trying $256 * 8$ possibilities), the process is repeated with a fresh random value and new initial pair (u_0, v_0). With $m = 1$, $c = 32$, and no restrictions on $\#\widetilde{E}(\mathbf{F}_p)$, it required on average less than ten seconds on a 133MHz Pentium processor to generate a satisfactory ECC parameter set at the 90-bit security level. Though this performance was apparently acceptable at the time [34] was published, it does not bode well for higher security levels and, in particular, when twist security is required as well. This is confirmed by experiments (cf. runtimes reported in Table 3 below).

Sieving-based search. Secondly, we show how the performance of the search can be considerably improved compared to [34]. Because, for a fixed d, the prime p and both group orders are quadratic polynomials in u and v, sieving with a set P of small primes can be used to quickly identify (u, v) pairs that do not correspond to a satisfactory p or E. The remaining pairs, for which the candidates for the prime and for the group order(s) do not have factors in P, can then be subjected to more precise inspection, similar to the search from [34]. We sketch our sieving-based search for ECC parameters as in Table 2 where we assume that $\min(2^\ell - 1, 2^{\tilde{\ell}} - 1) = f$ and $\max(2^\ell - 1, 2^{\tilde{\ell}} - 1) \in \{f, \infty\}$, i.e., we settle for perfect twist security (except for the factor f) or no twist security at all. More liberal choices require a more cumbersome approach to the sieving; we do not elaborate.

Let (u_0, v_0) be chosen as above, but restricted to certain residue classes modulo small primes to satisfy a variety of divisibility criteria depending on the above choices of f, ℓ, and $\tilde{\ell}$, and with respect to Montgomery friendliness. We found it most convenient to fix u_0 and to sieve over regularly spaced $(v_0 + i)$-values, again restricted to certain residue classes for the same reasons (including divisibility of $\#E(\mathbf{F}_p)$ by f in case $f > 1$), but using a much larger range of i-values than in [34]. Fixing u_0, the first at most sixteen compatible d-values from Table 2 are selected; only ten d-values may remain and depending on the parity of u_0 the value $d = 7$ may or may not occur. Let d_0, d_1, ..., d_{k-1} be the selected d-values, with $10 \leq k \leq 16$. With I the set of distinct i-values to be considered, we initialize for all $i \in I$ the sieve-location s_i as $2^k - 1$ (i.e., all "one"-bits in the k bit-positions indexed from 0 to $k - 1$), while leaving the constant difference between consecutive i-values unspecified for the present description. We mostly used difference 16, using difference 4 only for $d = 8$, and using a substantially larger value if the prime p must be Montgomery friendly.

For each d_j and each sieving-prime $\varsigma \in P$ up to six roots $r_{j\varsigma}$ modulo ς of up to three quadratic polynomials are determined (computing square roots using $\frac{\varsigma+1}{4}$-th powering for $\varsigma \equiv 3 \bmod 4$ and using the Tonelli-Shanks algorithm [20, 2.3.8] otherwise); the polynomials follow in a straightforward fashion from Table 2. To sieve for d_j the following is done for all $\varsigma \in P$ and for all roots $r_{j\varsigma}$: all sieve-locations s_i with $i \in (r_{j\varsigma} + \varsigma\mathbf{Z}) \cap I$ are replaced by $s_i \wedge 2^k - 2^j - 1$ (thus setting a possible "one"-bit at bit-position j in s_i to a "zero"-bit, while not changing the bits at the other $k - 1$ bit-positions in s_i).

A "one"-bit at bit-position j in s_i that is still "one" after the sieving (for all indices, all sieving primes, and all roots) indicates that discriminant $-d_j$ and pair $(u_0, v_0 + i)$ warrants closer inspection because all relevant related values are free of factors in P. If the search is unsuccessful (after considering $k|I|$ possibilities), the process is repeated with a new sieve. If for all indices j and all $\varsigma \in P$ all last visited sieve locations are kept (at most $6k|P|$ values), recomputation of the roots can be avoided if the same (u_0, v_0) is re-used with the "next" interval of i-values.

Some savings may be obtained, in particular for small ς values, by combining the sieving for identical roots modulo ς for distinct indices j. Or, one could make just a single sieving pass per ς-value but simultaneously for all indices j and all roots $r_{j\varsigma}$ modulo ς, by gathering (using "$\wedge$"), for that ς, all sieving information (for all indices and all roots) for a block of ς consecutive sieve locations, and using that block for the sieving.

Parameter reconstruction. A successful search results in an index j and value i such that d_j and the prime corresponding to the (u, v)-pair $(u_0, v_0 + i)$ leads to ECC parameters that satisfy the aimed for criteria. Any party that has the information required to construct (u_0, v_0) can use the pair (j, i) to instantaneously reconstruct (using Table 2) those same ECC parameters, without redoing the search. It is straightforward to arrange for an additional value that allows easy (re)construction of a base point.

Implementation results. We implemented the basic search as used in [34] and the sieving based approach sketched above for generic x86 processors and for ARM/Android devices. To make the code easily portable to other platforms as well we used the GMP 6.0 library [24] for multi-precision integer arithmetic after having verified that modular exponentiation (crucial for an efficient search) offers good performance on ARM processors. Making the code substantially faster would require specific ARM processor dependent optimization. We used the Java native interface [46] and the

Android native development kit [26] to allow the part of the application written in Java to call the GMP-based C-routines that underlie the compute intensive core. To avoid making the user interface non-responsive and avoid interruption by the Android run-time environment, a background service (*IntentService* class) [27] is instantiated to run this core independently of the thread that handles the user interface.

Table 3 lists detailed results for the 128-bit security level, using empirically determined (and close to optimal, given the platform) sieving bounds, lengths, etc. The implementations closely followed the description above, but we omit many details that were used to obtain better performance, such as precomputations and extra conditions, and to make sure that a variety of security requirements is met (more on this in Section 5). Table 4 shows average timings in milliseconds for different security levels in two cases: prime order non twist-secure generation and perfect twist security. The x86 platform is an Intel Core i7-3820QM, running at 2.7GHz under OS X 10.9.2 and with 16GB RAM. The ARM device is a Samsung Galaxy S4 smartphone with a Snapdragon 600 (ARM v7) running at 1.9GHz under Android 4.4 with 2GB RAM. Key reconstruction takes around 1.5 (x86) and 10 (ARM) milliseconds.

Table 3: Performance results in milliseconds for parameter generation at the 128-bit security level, with ℓ, $\tilde{\ell}$, f, P, and I as above, the "MF"-column to indicate Montgomery friendliness, and μ the average and σ the standard deviation.

					x86, over 10 000 runs						ARM, over 3000 runs					
					basic		sieving				basic		sieving			
ℓ	$\tilde{\ell}$	$\{\ell\} \cup \{\tilde{\ell}\}$	f	MF	μ	σ	μ	σ	$\lvert P\rvert$	$\lvert I\rvert$	μ	σ	μ	σ	$\lvert P\rvert$	$\lvert I\rvert$
not twist secure:																
		$\{6, \infty\}$			8.2	4.8	7.8	3.6	100	2^{10}	64	47	50	30	150	2^{12}
6					9.6	6.2	8.6	3.8	200	2^{10}	72	58	59	35	250	2^{12}
		$\{6, \infty\}$		✓	8.3	5.0	7.8	3.7	100	2^{10}	64	44	49	29	200	2^{12}
6				✓	9.7	6.4	8.7	3.8	200	2^{10}	71	55	60	33	250	2^{12}
		$\{6, \infty\}$	4		8.4	5.2	7.9	4.0	100	2^{10}	64	49	54	35	200	2^{12}
6			4		9.7	6.4	8.8	4.7	200	2^{10}	71	57	61	36	250	2^{12}
		$\{6, \infty\}$	4	✓	8.6	5.2	7.9	3.8	100	2^{10}	62	48	50	29	200	2^{12}
6			4	✓	9.7	6.4	8.6	3.7	200	2^{10}	72	58	56	35	250	2^{12}
		$\{1, \infty\}$			8.8	5.4	8.0	4.0	100	2^{10}	65	47	53	32	200	2^{12}
1					10.4	7.1	8.9	4.0	200	2^{10}	77	61	58	36	250	2^{12}
		$\{1, \infty\}$		✓	8.8	5.5	8.0	3.9	100	2^{10}	65	50	50	31	200	2^{12}
1				✓	10.4	7.0	8.8	3.9	200	2^{10}	76	62	57	35	250	2^{12}
twist secure:																
		$\{6\}$			148	143	46	33	700	2^{14}	1280	1271	357	304	750	2^{15}
1	6				167	162	55	44	800	2^{14}	1432	1392	410	335	750	2^{15}
		$\{1, 6\}$			160	151	49	34	800	2^{14}	1350	1341	392	326	750	2^{15}
		$\{1\}$			180	177	49	40	800	2^{14}	1433	1372	390	325	750	2^{15}
		$\{6\}$		✓	143	139	50	36	700	2^{14}	1301	1270	390	311	750	2^{15}
1	6			✓	165	161	51	38	800	2^{14}	1428	1321	409	315	750	2^{15}
		$\{1, 6\}$		✓	154	148	49	35	800	2^{14}	1327	1300	380	316	750	2^{15}
		$\{1\}$		✓	172	168	48	36	800	2^{14}	1491	1428	378	326	750	2^{15}
		$\{6\}$	4		162	158	49	34	700	2^{14}	1307	1245	390	319	750	2^{15}
		$\{6\}$	4	✓	165	159	50	38	700	2^{14}	1287	1253	385	318	750	2^{15}

Table 4: Performance results in milliseconds for parameter generation at different security levels: 80-bit, 112-bit, 128-bit, 160-bit, 192-bit and 256-bit. In comparison with Table 3 the Montgomery friendliness option is always disabled and $f = 1$.

		x86					ARM				
		basic	sieving			runs	basic	sieving			runs
k	$\{\ell\} \cup \{\tilde{\ell}\}$	μ	μ	$\lvert P\rvert$	$\lvert I\rvert$		μ	μ	$\lvert P\rvert$	$\lvert I\rvert$	
80	$\{1, \infty\}$	3	3	50	2^9	10000	22	19	100	2^{11}	100
	$\{1\}$	31	10	200	2^{12}	1000	197	61	450	2^{12}	100
112	$\{1, \infty\}$	6	6	100	2^9	10000	47	38	200	2^{10}	100
	$\{1\}$	114	30	800	2^{14}	1000	981	214	650	2^{14}	100
128	$\{1, \infty\}$	9	8	100	2^{10}	10000	65	53	250	2^{12}	3000
	$\{1\}$	180	49	800	2^{14}	10000	1433	390	750	2^{15}	3000
160	$\{1, \infty\}$	19	16	300	2^{11}	1000	143	87	200	2^{10}	100
	$\{1\}$	474	95	800	2^{14}	1000	5425	808	750	2^{15}	100
192	$\{1, \infty\}$	36	25	400	2^{12}	1000	265	169	20	2^{10}	100
	$\{1\}$	1144	222	1200	2^{16}	1000	10785	2231	900	2^{17}	20
256	$\{1, \infty\}$	105	70	400	2^{13}	1000	14543	575	450	2^{11}	100
	$\{1\}$	4635	994	1200	2^{16}	1000	50 sec	10 sec	1200	2^{17}	10

5 Security criteria

In this section we review security requirements that are relevant in the context of ECC. Most are taken from [9], the order and keywords of which we roughly follow for ease of reference, and some are from [22]. We discuss to what extent these requirements are met by the parameters generated by our method. Generally speaking our approach is to focus on existing threats, as dealing with non-existing ones only limits the parameter choice while not serving a published purpose.

ECDLP security. For the security of ECC, the discrete logarithm problem in the group of points of the elliptic curve must be hard. In this first category of security requirements one attempts to make sure that elliptic curve groups are chosen in such a way that this requirement is met.

- **Pollard rho attack** becomes ineffective if the group is chosen in such a way that a sufficiently large prime factor divides its order. This is a straightforward "key-length" issue (cf. [36]). Using a 128-bit prime field cardinality with $\ell \leq 5$, as suggested by Table 3, is more than sufficient.
- **Transfers** refer to the possibility to embed the group into a group where the discrete logarithm problem is easy, as would be the case for "anomalous curves" and for curves with a low "embedding degree". For the former, the elliptic curve group over the finite field $\mathbf{F}_p$ has cardinality p and can be effectively embedded in the additive group $\mathbf{F}_p$, allowing trivial solution of the elliptic curve discrete logarithm problem (cf. [49, 51, 54]). By construction our method avoids these curves.
  For the latter, the group can be embedded in the multiplicative group $\mathbf{F}_{p^k}^{\times}$ of $\mathbf{F}_{p^k}$ for a low embedding degree k. To avoid those curves, we follow the approach from [34] which ties the smallest permissible value for k to the published difficulty of finding discrete logarithms in $\mathbf{F}_{p^k}^{\times}$. It would be trivial, and would have negligible effect on our performance results, to adopt the "overkill" approach favored by [9, 39, 13], but we see no good reason to do so.
- **Complex-multiplication field discriminants** refers to the concern that for small values of the discriminant ($-d$ in our case) there are endomorphism-based speedups for the Pollard rho attack [61, 25]. For instance, the first row of Table 2 leads to groups with the same *automorphism group* [53, Chapter III.10] as the *pairing-friendly* groups proposed in [4] and thereby to an additional speedup of the Pollard rho attack by a factor of $\sqrt{3}$. We refer to [21, 14] for

a discussion of the practical implications and note that such speedups are of no concern for 128-bit prime field cardinalities with $\ell \leq 5$.
Despite the fact that the authors of [9] agree with this observation (cf. their quotation cited in the introduction), and as already mentioned in Section 2, [9] chooses a lower bound of 2^{100} for the absolute value of the complex-multiplication field discriminant while [39] settles for roughly 2^{40}. Neither bound can be satisfied by out method, as amply illustrated in Section 3.3. Until a valid concern is published, we see no reason to abandon our approach.

– **Rigidity** is the security requirement that the entire parameter generation process must be transparent and exclude the possibility that malicious choices are targeted. Assuming a transparent process to generate the initial pair (u_0, v_0) (for instance by following the approach described in [34]) the process proposed here is fully deterministic, fully explained, and leaves no room for trickery. Note also that a third party is excluded and that the affected parties (the public key owner or the two communicating parties engaging in the Diffie-Hellman protocol) are the only ones involved in the parameter generation process.

ECC security. Properly chosen groups can still be used in insecure ways. Here we discuss a number of precautions that may be taken to avoid some attacks that are aimed at exploiting the way ECC may be used.

– **Constant-time single-coordinate scalar multiplication** ("Ladders" in [9]) makes it harder to exploit timing differences during the most important operation in ECC, the multiplication of a group element by a scalar that usually needs to be kept secret, as such differences may reveal information about the scalar (where it should be noted that the "single-coordinate" part is just for efficiency and ease of implementation). For all Weierstrass curve parameterizations used here constant-time single-coordinate scalar multiplication can be achieved using the method from [15]. If efficiency is a bigger concern than freedom of choice, one may impose the requirement that the group order is divisible by four ("$f = 4$" in Table 3) as it allows conversion to Montgomery form [42] and thereby a more efficient constant-time single-coordinate scalar multiplication [41].
– **Invalid-point attacks** ("Twists" in [9]) refer to attempts to exploit a user's omission to verify properties of alleged group elements received. They are of no concern if the proper tests are consistently performed (at the cost of some performance loss) or if a closed software environment can be relied upon. Some are also thwarted if the curve's twist satisfies the same ECDLP security requirements as the curve itself, an approach that thus avoids implementation assumptions while replacing recurring verification costs by one-time but more costly parameter generation: for one-time parameter usage one-time verification is less costly (than relatively expensive generation of twist secure parameters), for possibly repeated usage (as in certified keys) twist secure parameters may be preferred. Our parameter selection method includes the twist security option and thus caters to either scenario. Below we elaborate on the various attack possibilities.
  Small-subgroup attacks. If the group order is not prime but has a relatively small factor h, an attacker may send a group element of order h (as opposed to large prime order), learn the residue class modulo h of the victim's secret key, and thus obtain a speedup of the Pollard rho attack by a factor of $\sqrt{h}$. It suffices to ascertain that group elements received do not have order dividing h, or to generate the parameters such that the group order is prime (one of our options).
  Invalid-curve attacks. An attacker may send elements of different small prime orders belonging to different appropriately selected elliptic curve groups, all distinct from the proper group. Each time the targeted victim fails to check proper group membership of elements received the attacker learns the residue class modulo a new small prime of the victim's secret key, ultimately enabling the attacker to use the Chinese remainder theorem to recover the key [10]. This attack cannot be avoided at the parameter selection level, but is avoided by checking that each element received belongs to the right group (at negligible cost). Also, using parameters just once renders the attack ineffective.

Twist attacks against single-coordinate scalar multiplication. Usage of single coordinates goes a long way to counter the above invalid-curve attacks, because each element that does not belong to the group of the curve automatically belongs to the group of the twist of the curve. Effective attacks can thus be avoided either by checking membership of the proper group (i.e., not of the group of the twist) or by making sure that the group of the twist of the curve satisfies the same security requirements as the group of the curve itself (at a one-time twist secure parameter generation cost, avoiding the possibly recurring membership test). As mentioned above, it depends on the usage scenario which method is preferred; for each scenario our method offers a compatible option.

- **Exceptions in scalar multiplication** ("Completeness" in [9]). Depending on the curve parameterization, the implementation of the group law may distinguish between adding two distinct points and doubling a point. Using addition where doubling should have been used may be leveraged by an attacker to learn information about the secret key [31]. Either a check must be included (while maintaining constant-time execution, as in [13]) or a "complete" addition formula must be used, i.e., one that works even if the two points are not distinct. This leads to a somewhat slower group law for our Weierstrass curve parameterizations, but if they are used along with $f = 4$ in Table 3 the parameterization can be converted to Edwards or Montgomery form, which are both endowed with fast complete formulae for the group law [8], [5].
- **Indistinguishability** of group elements and uniform random strings is important for ECC applications such as censorship-circumvention protocols [9], but we are not aware of its importance for the applications targeted in this paper. We refer to [7, 23] for ways to achieve indistinguishability using families of curves in Montgomery, Edwards or Hessian form and to [57] for a solution that applies to the Weierstrass curve parameterization (which, however, doubles the lengths of the strings involved). Either way, our methods can be made to deal with this issue as well.
- **Strong Diffie-Hellman problem** (not mentioned in [9]). In [18] it is shown that for protocols relying on the ECC version of the strong Diffie-Hellman problem the large prime q dividing the group order must be chosen such that $q - 1$ and $q + 1$ both have a large prime factor. Although several arguments are presented in [17, Section B.1] why this attack is "unlikely to be feasible", [17] nevertheless continues with "as a precautionary measure, one may want to choose elliptic curve domain parameters that resist Cheon's attack by arranging that $q-1$ and $q+1$ have very large prime factors". Taking this precaution, however, would add considerable overhead to the parameter generation process. Our methods can in principle be adapted to take this additional requirement into account, but doing so will cause the parameter generation timings to skyrocket. The attack is not considered in [9], and none of the standardized parameter choices that we inspected take the precaution recommended in [17].

Side-channel attacks are physical attacks on the device executing the parameter generation process or the cryptographic protocols. Most of these attacks require multiple runs of the ECC protocol with the same private key (cf. [22, Table 1]) and are thus of no concern in an ephemeral key agreement application. There are three attacks for which a single protocol execution suffices:

Simple power analysis (SPA) attacks are avoided when using a scalar multiplication algorithm ensuring that the sequence of operations performed is independent of the scalar.

Fault induced invalid curve attacks can be expected to require several trials before a weak parameter choice is hit, and can be prevented by enforcing more sanity checks in the scalar multiplication [22].

Template attacks may recover a small number of bits of the secret key and can be avoided using one of the randomization techniques mentioned in [22].

6 Conclusions and future work

We showed how communicating parties can efficiently generate fresh ECC parameters every time they need to agree on a session key, generalizing and improving the method from [34]. Our major

modifications consist of the use of sieving to speed up the generation process, a greater variety of security and efficiency options, and the inclusion of eleven more curve equations. Furthermore, we explained how to further generalize our method and showed that doing so may have limited practical value. We demonstrated the practical potential of our method on constrained devices, presented performance figures of an implementation on an ARM/Android platform, and discussed relevant security issues.

Future work could include further efficiency enhancements by targeting specific ARM processors, direct inclusion of Montgomery and Edwards forms, extension to genus 2 hyperelliptic curves and, much more challenging and important, improving elliptic curve point counting methods to allow on-the-fly generation of ephemeral random elliptic curves over prime fields. Unfortunately, we do not know yet how to approach the latter problem, but genus 2 extension of our methods seems to be quite within reach. We conclude with a few remarks on this issue.

Extension to genus 2 hyperelliptic curves. Jacobians of hyperelliptic curves of genus 2 allow cryptographic applications similar to elliptic curves [33] and, as recently shown in [12], offer comparable or even better performance. Genus 2 hyperelliptic curves may thus be a worthwhile alternative to elliptic curves and, in particular given the lack of a reasonable variety of standardized genus 2 curves, generalization of our methods to the genus 2 case may have practical appeal. In [60] it is described how this could work. The imaginary quadratic fields are replaced by *quartic* CM fields and the j-invariant (a root of the Hilbert class polynomial) is replaced by three j-invariants which are usually referred to as Igusa's invariants. In [59] a table is given listing equations with integer coefficients of genus 2 hyperelliptic curves having complex multiplication by class number one quartic CM fields and class number two quartic CM fields. The three algorithms presented at the beginning of [60, Section 8] can then be used to easily compute the orders of the Jacobians of these curves over suitably chosen prime fields. The main remaining problem seems to be to resolve the ambiguity between the order of the Jacobian of the hyperelliptic curve and of its quadratic twist other than by using scalar multiplication. We leave the solution of this problem – and implementation of the resulting genus 2 parameter selection method – as future work.

Acknowledgement. Thanks to Adrian Antipa for bringing the strong Diffie-Hellman security requirement and additional precaution from [17, Section B.1] to our attention, and to René Schoof for inspiring this paper by providing the original table in [34].

References

1. D. F. Aranha, P. S. L. M. Barreto, C. C. F. P. Geovandro, and J. E. Ricardini. A note on high-security general-purpose elliptic curves. *IACR Cryptology ePrint Archive*, 2013:647, 2013.
2. A. O. L. Atkin and F. Morain. Elliptic curves and primality proving. *Math. Comp*, 61:29–68, 1993.
3. R. M. Avanzi, H. Cohen, C. Doche, G. Frey, T. Lange, K. Nguyen, and F. Vercauteren. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. Chapman & Hall/CRC, 2006.
4. P. S. L. M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In B. Preneel and S. E. Tavares, editors, *Selected Areas in Cryptography*, volume 3897 of *LNCS*, pages 319–331. Springer, 2005.
5. D. J. Bernstein. Curve25519: New Diffie-Hellman speed records. In M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, editors, *Public Key Cryptography – PKC 2006*, volume 3958 of *Lecture Notes in Computer Science*, pages 207–228. Springer, Heidelberg, 2006.
6. D. J. Bernstein, T. Chou, C. Chuengsatiansup, A. Hülsing, T. Lange, R. Niederhagen, and C. van Vredendaal. How to manipulate curve standards: a white paper for the black hat. Cryptology ePrint Archive, Report 2014/571, 2014. `http://eprint.iacr.org/2014/571`.
7. D. J. Bernstein, M. Hamburg, A. Krasnova, and T. Lange. Elligator: elliptic-curve points indistinguishable from uniform random strings. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, CCS '13, pages 967–980, New York, NY, USA, 2013. ACM.
8. D. J. Bernstein and T. Lange. Faster addition and doubling on elliptic curves. In K. Kurosawa, editor, *Asiacrypt*, volume 4833 of *Lecture Notes in Computer Science*, pages 29–50. Springer, Heidelberg, 2007.
9. D. J. Bernstein and T. L. Lange. Safecurves: choosing safe curves for elliptic-curve cryptography.
10. I. Biehl, B. Meyer, V. Müller, U. K. D. Wacana, and J. D. Wahidin. Differential fault attacks on elliptic curve cryptosystems. pages 131–146. Springer-Verlag, 2000.

11. M. Blaze, W. Diffie, R. L. Rivest, B. Schneier, T. Shimomura, E. Thompson, and M. Wiener. Minimal key lengths for symmetric ciphers to provide adequate commercial security. `http://www.schneier.com/paper-keylength.pdf`, January 1996.
12. J. W. Bos, C. Costello, H. Hisil, and K. Lauter. Fast cryptography in genus 2. In T. Johansson and P. Q. Nguyen, editors, *EUROCRYPT*, volume 7881 of *LNCS*, pages 194–210. Springer, 2013.
13. J. W. Bos, C. Costello, P. Longa, and M. Naehrig. Selecting elliptic curves for cryptography: An efficiency and security analysis. Cryptology ePrint Archive, Report 2014/130, 2014. `http://eprint.iacr.org/`.
14. J. W. Bos, C. Costello, and A. Miele. Elliptic and hyperelliptic curves: A practical security analysis. In H. Krawczyk, editor, *Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 203–220. Springer, 2014.
15. E. Brier and M. Joye. Weierstraß elliptic curves and side-channel attacks. In D. Naccache and P. Paillier, editors, *Public Key Cryptography – PKC 2002*, volume 2274 of *Lecture Notes in Computer Science*, pages 335–345. Springer, Heidelberg, 2002.
16. Certicom Research. Standards for efficient cryptography 2: Recommended elliptic curve domain parameters. Standard SEC2, Certicom, 2000.
17. Certicom Research. Standards for efficient cryptography 1: Elliptic curve cryptography (version 2.0). Standard SEC1, Certicom, 2009.
18. J. H. Cheon. Security analysis of the strong Diffie-Hellman problem. In S. Vaudenay, editor, *Eurocrypt 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 1–11. Springer, Heidelberg, 2006.
19. D. Coppersmith. Modifications to the number field sieve. *Journal of Cryptology*, 6(3):169–180, 1993.
20. R. Crandall and C. Pomerance. *Prime Numbers: A Computational Perspective (second edition)*. Lecture notes in statistics. Springer, 2005.
21. I. M. Duursma, P. Gaudry, and F. Morain. Speeding up the discrete log computation on curves with automorphisms. In K.-Y. Lam, E. Okamoto, and C. Xing, editors, *Asiacrypt 1999*, volume 1716 of *Lecture Notes in Computer Science*, pages 103–121. Springer, Heidelberg, 1999.
22. J. Fan and I. Verbauwhede. An updated survey on secure ecc implementations: Attacks, countermeasures and cost. In D. Naccache, editor, *Cryptography and Security: From Theory to Applications*, volume 6805 of *Lecture Notes in Computer Science*, pages 265–282. Springer Berlin Heidelberg, 2012.
23. P.-A. Fouque, A. Joux, and M. Tibouchi. Injective encodings to elliptic curves. In C. Boyd and L. Simpson, editors, *Information Security and Privacy*, volume 7959 of *Lecture Notes in Computer Science*, pages 203–218. Springer Berlin Heidelberg, 2013.
24. Free Software Foundation, Inc. *GMP: The GNU Multiple Precision Arithmetic Library*, 2014. Available at `http://www.gmplib.org/`.
25. R. P. Gallant, R. J. Lambert, and S. A. Vanstone. Improving the parallelized Pollard lambda search on anomalous binary curves. *Mathematics of Computation*, 69(232):1699–1705, 2000.
26. Google. Android NDK. `https://developer.android.com/tools/sdk/ndk/index.html`.
27. Google. Android SDK guide. `http://developer.android.com/guide/index.html`.
28. T. C. Hales. The NSA Back Door to NIST. *Notices of the AMS*, 61(2):190–192, 2013.
29. G. Hanrot and F. Morain. Solvability by radicals from an algorithmic point of view. In *Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation*, ISSAC '01, pages 175–182, New York, NY, USA, 2001. ACM.
30. N. Ishii. Trace of frobenius endomorphism of an elliptic curve with complex multiplication. *Bulletin of the Australian Mathematical Society*, 70:125–142, 8 2004.
31. T. Izu and T. Takagi. Exceptional procedure attack on elliptic curve cryptosystems. In Y. Desmedt, editor, *Public Key Cryptography — PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 224–239. Springer Berlin Heidelberg, 2002.
32. N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.
33. N. Koblitz. Hyperelliptic cryptosystems. *Journal of cryptology*, 1(3):139–150, 1989.
34. A. K. Lenstra. Efficient identity based parameter selection for elliptic curve cryptosystems. In *Proceedings of the 4th Australasian Conference on Information Security and Privacy*, ACISP '99, pages 294–302, London, UK, UK, 1999. Springer-Verlag.
35. A. K. Lenstra and H. W. Lenstra, Jr. *The Development of the Number Field Sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer-Verslag, 1993.
36. A. K. Lenstra and E. R. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology*, 14(4):255–293, 2001.
37. A. K. Lenstra and B. Wesolowski. A random zoo: sloth, unicorn, and trx. Cryptology ePrint Archive, Report 2015/366, 2015. `http://eprint.iacr.org/2015/366`.

38. H. W. Lenstra Jr. Factoring integers with elliptic curves. *Annals of Mathematics*, 126(3):649–673, 1987.
39. M. Lochter and J. Merkle. Elliptic curve cryptography (ECC) brainpool standard curves and curve generation. RFC 5639, 2010.
40. V. S. Miller. Use of elliptic curves in cryptography. In H. C. Williams, editor, *Crypto 1985*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426. Springer, Heidelberg, 1986.
41. P. L. Montgomery. Speeding the Pollard and Elliptic Curve Methods of Factorization. *Mathematics of Computation*, 48(177):243–264, 1987.
42. F. Morain. Edwards curves and cm curves. Technical report, 2009.
43. National Institute of Standards and Technology. Special publication 800-57: Recommendation for key management part 1: General (revised). `http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2_Mar08-2007.pdf`.
44. National Security Agency. Fact sheet NSA Suite B Cryptography. `http://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml`, 2009.
45. NIST. Workshop on Elliptic Curve Cryptography Standards 2015. `http://www.nist.gov/itl/csd/ct/ecc-workshop.cfm`, June 2015.
46. Oracle. Java native interface.
47. A. R. Rajwade. Certain classical congruences via elliptic curves. *J. London Math. Soc. (2)*, 8:60–62, 1974.
48. K. Rubin and A. Silverberg. Choosing the correct elliptic curve in the cm method. *Mathematics of Computation*, 79(269):545–561, 2010.
49. T. Satoh and K. Araki. Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves. *Commentarii Mathematici Universitatis Sancti Pauli*, 47(1):81–92, 1998.
50. R. Schoof and P. R. E. Schoof. Counting points on elliptic curves over finite fields, 1995.
51. I. A. Semaev. Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p. *Mathematics of Computation*, 1998.
52. D. Shumov and N. Ferguson. On the Possibility of a Back Door in the NIST SP800-90 Dual EC PRNG. 2007.
53. J. H. Silverman. *The Arithmetic of Elliptic Curves*, volume 106 of *Gradute Texts in Mathematics*. Springer-Verlag, 1986.
54. N. P. Smart. The discrete logarithm problem on elliptic curves of trace one. *J. Cryptology*, 12(3):193–196, 1999.
55. B. Snow, June 2014. Private communication.
56. H. Stark. Counting points on *cm* elliptic curves. *Rocky Mountain Journal of Mathematics*, 26(3):1115–1138, 09 1996.
57. M. Tibouchi. Elligator squared: Uniform points on elliptic curves of prime order as uniform random strings. *IACR Cryptology ePrint Archive*, 2014:43, 2014.
58. U.S. Department of Commerce/National Institute of Standards and Technology. Digital Signature Standard (DSS). FIPS-186-4, 2013. `http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf`.
59. P. B. van Wamelen. Examples of genus two cm curves defined over the rationals. *Math. Comput.*, 68(225):307–320, 1999.
60. A. Weng. Constructing hyperelliptic curves of genus 2 suitable for cryptography. *Math. Comput.*, 72(241):435–458, 2003.
61. M. J. Wiener and R. J. Zuccherato. Faster attacks on elliptic curve cryptosystems. In S. Tavares and H. Meijer, editors, *Selected Areas in Cryptography – (SAC) 1998*, volume 1556 of *Lecture Notes in Computer Science*, pages 190–200. Springer New York, 1999.