A brief discussion on selecting new elliptic curves

Craig Costello, Patrick Longa, and Michael Naehrig

Microsoft Research

Abstract. This position paper summarizes our perspectives on the selection of next-generation elliptic curves for standardization. It also contains generation algorithms intended as a foundation for choosing elliptic curves for cryptography in a simple, consistent and rigid way.

1 Introduction

The discussion on selecting the next generation of elliptic curves for cryptography has taken center stage after doubts about the trustworthiness of the standardized NIST curves emerged. The IETF TLS working group has issued a request to the IETF Crypto Forum Research Group (CFRG) asking for a recommendation of new elliptic curves for the TLS protocol. Discussions in the CFRG have almost exclusively addressed software implementations on a very restricted subset of platforms and have considered only timing attacks. This focus has excluded several relevant scenarios, e.g. implementations on embedded devices, hardware platforms and considerations about other side-channel attacks. Our position is that an independent selection process should consider a wide range of scenarios and platforms, be driven by requirements and be based on a transparent, consistent and deterministic curve generation procedure.

In this position paper, we first give an overview of criteria for the selection of elliptic curves, and then discuss a generation algorithm that can be used to produce secure curves. The algorithm takes the base field prime as input, and thus allows different choices of prime shape candidates depending on specific efficiency and security criteria. We discuss some of the prime shape candidates at the end of the paper, and conclude by presenting an example selection of curves over pseudo-Mersenne primes.

2 Curve models

> *"NIST should generate a new set of elliptic curves for use with ECDSA in FIPS 186... The set of high-quality curves should be described precisely in the standard, and should incorporate the latest knowledge about elliptic curves."*
>
> Edward Felten [17, p. 36]

Let $\mathbb{F}_p$, $p > 3$ denote a large prime field. An elliptic curve $E/\mathbb{F}_p$ can be described by different *models*, i.e. written down using different defining equations, depending on the structure of the Abelian group of $\mathbb{F}_p$-rational points $E(\mathbb{F}_p)$. To date there are essentially three such models of interest in the discussion of standardized curves: the short Weierstrass model, the twisted Edwards model [3], and the Montgomery model [16], written (respectively) as

$$\begin{aligned} E^{\mathrm{W}}_{a,b} &: \quad y^2 = x^3 + ax + b, \\ E^{\mathrm{Ed}}_{a,d} &: \quad ax^2 + y^2 = 1 + dx^2y^2, \quad \text{and} \\ E^{\mathrm{M}}_{A,B} &: \quad By^2 = x^3 + Ax^2 + x, \end{aligned} \tag{1}$$

where the superscripts denote the model and the subscripts denote the $\mathbb{F}_p$-rational *curve constants.*

The short Weierstrass model $E^{\mathrm{W}}_{a,b}$ is general in the sense that it can be used to describe all elliptic curves defined over large prime fields. This includes all such curves in previous standards as well as those currently under discussion for future standards. On the other hand, the twisted Edwards and Montgomery models $E^{\mathrm{Ed}}_{a,d}$ and $E^{\mathrm{M}}_{A,B}$ can only be used for a subset of elliptic curves; namely, they can only describe an elliptic curve E if $4 \mid \#E(\mathbb{F}_p)$, which means that these models cannot be used for prime order curves (i.e., all of the curves defined over prime fields in previous standards). The subset of elliptic curves which have a twisted Edwards model is exactly the same as the subset of curves which have a Montgomery model (because every twisted Edwards curve is birationally equivalent over $\mathbb{F}_p$ to a Montgomery curve, and vice versa [3, Thm. 3.2]).

Although the above models are presented with two parameters, in practice, one of the constants is usually fixed to a specific value and curves are essentially defined by a single curve constant: fixing $a = -3$ in $E^{\mathrm{W}}_{a,b}$ is used for an increased efficiency of curve operations; for the same reasons, the parameter a in $E^{\mathrm{Ed}}_{a,d}$ can be set to $a = 1$ or $a = -1$ depending on the underlying prime field; and, for fixed A, there are (up to isomorphism) only two choices of $B \in \mathbb{F}_p$, those which correspond to the two quadratic twists of $E^{\mathrm{M}}_{A,B}$ (see Section 4). Thus, we use E^{W}_{b}, E^{Ed}_{d} and E^{M}_{A} as shorthand for $E^{\mathrm{W}}_{-3,b}$, $E^{\mathrm{Ed}}_{\pm 1,d}$ and $E^{\mathrm{M}}_{1,A}$ respectively (the sign in $E^{\mathrm{Ed}}_{\pm 1,d}$ will be made clear in context).

There are many different criteria that can be used to differentiate these curve models, and there are numerous pros and cons to each of them. The twisted Edwards curve model has been considered for implementations because it can offer a simple, single group law formula that works for all possible inputs (i.e., is *complete*). The Montgomery model has been considered because it can be used to implement an x-coordinate-only ladder for scalar multiplication. In Table 1 we compare the three models in terms of what we believe are the three main "yes/no" categories; below the table we elaborate on what each of these categories mean in practice. We do not include any performance/speed comparisons in Table 1, since this is very implementation-dependent. It is worth noting however, that roughly speaking, our implementations of variable-base scalar multiplications on 64-bit Intel platforms revealed that the performance of the twisted Edwards and Montgomery models was similar across the three (128-, 192-, and 256-bit) security levels, and that the fastest of these two models was consistently close to a factor 1.20x faster than the Weierstrass model [7, Table 3][1].

Table 1. Practical "yes/no" advantages of different elliptic curve models.

curve model	prime order possible	simple completeness	supports `ADD` function
Weierstrass	✓	✗	✓
twisted Edwards	✗	✓	✓
Montgomery x-only	✗	✓	✗

[1] Over the same finite field. This difference is entirely dependent on the speed of the explicit formulas, so the rough consistency of the performance ratio across security levels is not surprising.

- **Advantages of prime order.** Prime order ($a = -3$ short Weierstrass) curves are backwards compatible with implementations that support the most popular standardized curves. Since there is no co-factor, points that are validated to be on the curve trivially have prime order, so avoiding small subgroup attacks is inherent in point validation.

- **Advantages of completeness.** A single, complete addition law ensures an easier, more compact and exception-free implementation; it makes it easier to write branch-free, constant-time code[2]. In the setting of pseudo-additions using the Montgomery x-coordinate, one can always use the same differential addition formulas which is why we consider this as complete.

- **Advantages of `ADD` function.** Access to a general addition operation ensures cryptographic versatility, e.g. it makes ECDSA signature verification possible. When one can exploit precomputations, an addition function can lead to significant speedups, especially in the ephemeral Diffie-Hellman key setting.

Summary. In light of Table 1, we do not see any significant reason to use the Montgomery model over the twisted Edwards model when working with a cofactor curve. Hence, if cofactor curves are put forward, our opinion is that they should be specified in twisted Edwards form and that twisted Edwards coordinates should be used for simplicity and consistency. To decide on whether the new curves should be prime (short Weierstrass) or composite order (twisted Edwards), the trade-offs in the first two columns of Table 1 and the backwards compatibility offered by prime order short Weierstrass curves should be considered alongside the approximate factor 1.20x performance advantage offered by (twisted) Edwards curves.

3 Security levels

> *"Security first: When NIST issues a standard or guideline whose primary purpose is security, the security of that standard or guideline (e.g. the security of the algorithm, protocol, process, or design that is standardized) should be treated as the top priority. ... This principle also requires that design of security standards and guidelines be conservative with minimal assumptions or issues left to faith or chance."*
>
> Steve Lipner [17, p. 47]

There are two aspects to consider in regards to security levels:

(i) how many security levels (curves) should be put forward?, and
(ii) what is the size of the prime subgroup required to meet a particular security level?

Concerning (i), some argue that having a single curve at the 128-bit security level is enough. The reasoning here is that if a new (say, subexponential or even polynomial time) attack is discovered that breaks the 128-bit security level, then it is also likely to break higher security levels too. But this disregards the possibility of an improved (but still exponential) attack with better than "square root" complexity. In this case, the security of the lower-security curve(s) may be downgraded to be within cryptanalytic reach, but the higher-security curve(s) could

[2] In [7] we present an efficient implementation technique that addresses the problem of Weierstrass incompleteness, but the presence of one complete addition law remains the most simple solution.

still offer very good security and performance relative to other primitives. Our opinion is that it is worth standardizing curves at multiple security levels and to best prepare ourselves against potential advances in cryptanalysis. Historical hindsight tells us that there is a rich history of underestimating attack improvements and future projections on parameter sizes. Moreover, different entities, such as companies and governments have different requirements and thus different views on the security levels they want to use for different applications.

Assuming that new curves at different security levels are required, then in order to minimize the additional effort, we argue that the curves should be chosen consistently across security levels. This is one of the main requirements behind the selection criteria of the curves in [7] and in the development of the accompanying ECC library [14]: we chose elliptic curves with the same model over prime fields of the same form with 256, 384 and 512 bits – see Section 8. This allowed the reuse of much of the code and algorithms across the security levels and minimized the implementation effort for both the curve and field arithmetic. We do not intend to overstate this overlap since different applications might require the implementation of more aggressive optimizations that would force the use of separate, specialized arithmetic routines. However, we believe that this is an important factor to consider in the next generation of elliptic curves because it would favor deployment in a wide range of settings.

Some have argued in favor of lowering the security goalposts of the next generation of curves to better match the real-world security of the symmetric primitives they are to be paired with, in order to gain efficiency. At present though, there is no consensus on the equivalent strengths between asymmetric (e.g. ECC) and symmetric (e.g. AES) primitives[3], and until a concrete comparison can be made and is agreed on by experts, to us it seems ill-informed to start downgrading the strength of ECC primitives to match conjectural weaknesses in the symmetric world. Others are using ad-hoc “security versus performance” metrics to justify curves at security levels different to the traditional ones. But again, to date there is no well-established metric to measure CPU performance against security. Moreover, based on our experiments with curves at a range of (both traditional and “in-between”) security levels, we are yet to observe any significant performance-based reason to go moving the security goalposts in the next generation of curves.

Summary. Our opinion is that it is advantageous to standardize curves across multiple security levels and that these curves should be chosen consistently so as to reduce the implementation and maintenance effort. Furthermore, at present we do not see any reason significant enough to warrant deviation from the well-established security levels.

4 Twist-security

The notion of “twist-security” was introduced in [2] as a means of avoiding curve pertinence checks in the special case when one is working only with the x-coordinate corresponding to the Montgomery model E_A^{M}. Roughly speaking, this is because around half of the values of $x \in \mathbb{F}_p$ correspond to a point on E_A^{M} and the remaining x-values correspond to a point on the quadratic twist of E_A^{M}. Checking that a particular value of x corresponds to E_A^{M} rather than its twist amounts to checking that $(x^3 + Ax^2 + x)$ is a square in $\mathbb{F}_p$, which costs an exponentiation in $\mathbb{F}_p$. Since this is more expensive than the essentially free point pertinence checks that one performs when both (e.g. Weierstrass or twisted Edwards) coordinates are

[3] For example, see the different recommendations at `http://www.keylength.com/`.

available, one way to avoid it is to make sure that both E_A^{M} and its quadratic twist are secure, i.e. have strong group orders.

The importance of twist-security should not be overstated: the only practical setting where twist-security offers any known advantage is when one is using the Montgomery ladder to perform x-coordinate-only arithmetic. If one uses Weierstrass coordinates or twisted Edwards coordinates then one should always perform the trivial checks to make sure transmitted points are on the right curve. Thus, the absence of this property in some of the NIST curves [22, App. D] and the Brainpool curves [8] should not be used to dismiss any of them.

There is no clear consensus in the community if new curves should have this property. On the one hand, one could argue that a curve and its quadratic twist are related (after all, they become isomorphic over $\mathbb{F}_{p^2}$), and that one may as well make sure the related object is secure too. On the other hand, some have argued that curves with a strong twist "constitute a distinguished subset of all possible curves" and that "this specialty could lead to future attacks" [12]. We point out, however, that one could apply similar reasoning (without consideration of twists) to say that curves with strong group orders are also a subset of all curves.

Summary. Twist-security is a notion that should be taken into account when working with the x-coordinate-only Montgomery model. When working in other models, such as the short Weierstrass model, the value of this property should not be overstated. There is no clear consensus in the community if new curves should have this property. We think that, as far as generating new curves, having twist-security is a safer bet than not having it, and therefore we include it in our curve generation algorithms.

5 Rigidity and curve constants

> *"NIST should ensure that there are no secret or undocumented components or constants in its cryptographic standards whose origin and effectiveness cannot be explained. "*
>
> Steve Lipner [17, p. 49]

Finding elliptic curves in a transparent and deterministic manner is a way for cryptographers to convince the widespread community that there are no hidden weaknesses in the curve. The desire for this so-called *rigidity* property dates back to Scott's blog post[4] that questioned the process that was used to generate the NIST curves in 1999. Namely, the set of NIST's short Weierstrass curves E_b^{W} of prime order were chosen by passing random seeds through a one-way process f to generate the curve constant $b \in \mathbb{F}_p$, where p was fixed for efficiency reasons. This obviously required searching through many random seeds s_i until $E_b^{\mathrm{W}}(\mathbb{F}_p)$ with $b = f(s_i)$ had prime order. Scott immediately noticed, however, that if the entity generating the curve knew of a subset of curves with hidden weaknesses (among all possible curves $E_b^{\mathrm{W}}/\mathbb{F}_p$), then they could have simply tried as many seeds as they needed to until they found a curve that possessed this weakness and had prime order. Scott's response was to generate curves using a rigid process: one that minimizes the "wiggle-room" a cryptographer (with some conjectured backdoor knowledge) has to try and manipulate a weak curve as a result. This means that, at the very least, users can verify that a curve was the output of a process that is logically explained and deterministic.

[4] See `https://groups.google.com/forum/message/raw?msg=sci.crypt/mFMukSsORmI/FpbHDQ6hM_MJ`.

Since the doubts concerning the origins of the NIST seeds surfaced in the community, most proposals for new curves have done exactly this. For example, in his original post, Scott suggested using the digits of π to kickstart an iteration procedure, while Brainpool used the digits of both π and e to kickstart a search for a secure combination of underlying primes p and curve constants b, respectively. Both of these processes are widely agreed to achieve rigidity (despite the claims to the contrary in [4]).

However, when considering non-Weierstrass models, the situation can change slightly: while the fastest explicit formulas for point operations in Weierstrass form are independent of the curve constant b, the fastest explicit formulas in the twisted Edwards and Montgomery models do make use of multiplications by their respective constants. Naturally then, and in light of the obvious efficiency advantages that are possible, the simplest process to achieve rigidity is to find the *smallest* curve constant such that the curve is secure. While it may appear that there could be a specialized attack that takes advantage of a small constant, experts agree that the size of the constant is irrelevant to the difficulty of attacking the curve.

Summary. Over a fixed prime field $\mathbb{F}_p$, and in light of Section 4, one can select curves over any of the models in a rigid manner by finding the smallest curve constant such that the curve and its twist have an optimal group order. We make this explicit for the cases of interest in the following section.

6 Deterministic algorithms for generating curves

> *"NIST should consider the publication of a standard algorithm and corresponding software to generate additional elliptic curves and should consider to use this tool to also publish some new curves."*
>
> Bart Preneel [17, p. 65]

Based on the arguments from the previous section, we now describe simple and deterministic algorithms for generating secure elliptic curves. The algorithms operate without any hidden parameters, reliance on randomness or any other processes offering opportunities for manipulation of the resulting curves. We present algorithms for the Weierstrass and (twisted) Edwards models to generate prime order curves and curves with small co-factors, respectively. In the case of co-factor curves, the selection between curve models is determined by choosing the curve form that supports the fastest (currently known) complete formulas: when $p \equiv 3 \bmod 4$, the fastest complete formulas are in Edwards coordinates on $E^{\mathrm{Ed}}_{1,d}$, but when $p \equiv 1 \bmod 4$, the fastest complete formulas are in twisted Edwards coordinates on $E^{\mathrm{Ed}}_{-1,d}$. The algorithms generate the curves with the smallest curve constant that satisfy the selection criteria. We do not give an algorithm for the Montgomery model since the curve with the smallest absolute value of the constant A corresponds to the (twisted) Edwards curve with the smallest absolute value of the constant d. The correspondence is given via isogenies[5] of degree 4 defined over the base field, which means that the corresponding Montgomery and twisted Edwards curves have the same curve and twist group orders.

The principle of choosing curves with the smallest constants once the base field prime and the selection criteria are fixed, has been used before (with some variations) in several works of different authors (e.g., see [5, 1, 7]).

[5] See `http://cryptosith.org/papers/isogenies_tEd2Mont.pdf` for the technical details.

6.1 Prime order curves

Let t_b denote the trace of Frobenius of the curve $E_b^{\mathrm{W}}/\mathbb{F}_p\colon y^2 = x^3 - 3x + b$, i.e. $t_b = p + 1 - \#E_b^{\mathrm{W}}(\mathbb{F}_p)$. The following algorithm finds the integer b with smallest absolute value such that $E_b^{\mathrm{W}}/\mathbb{F}_p$ is of prime order and has a prime order twist. The algorithm simply tests the values of b in the sequence $1, -1, 3, -3, 4, -4, \ldots$ in that order until a curve-twist pair of prime order curves is found. Note that the values $-2, 2$ for b are left out because these values produce singular curves. In the case that $p \equiv 3 \bmod 4$, the search can be performed twice as fast by ignoring the negative values of b, since E_b^{W} and E_{-b}^{W} are non-trivial quadratic twists of each other; in this case, once a curve-twist pair is found, we can work on the twist whose trace is positive, i.e., the twist with the smaller group order. The notation **for** $b \in [1, -1, 3, -3, 4, -4, \ldots]$ **do** means running through the given sequence of integers in that order.

On input of a prime p, do the following:

```
for $b \in [1, -1, 3, -3, 4, -4, \ldots]$ do
  Compute $t_b$.
  if $p + 1 - t_b$ is prime and $p + 1 + t_b$ is prime then return $b$.
end for
```

6.2 Cofactor curves

Cofactor curves for $p \equiv 1 \bmod 4$. Let t_d denote the trace of Frobenius of the curve $E_{-1,d}^{\mathrm{Ed}}/\mathbb{F}_p\colon -x^2 + y^2 = 1 + dx^2y^2$, i.e. $t_d = p + 1 - \#E_{-1,d}^{\mathrm{Ed}}(\mathbb{F}_p)$. The following algorithm finds the integer d with smallest absolute value such that $E_{-1,d}^{\mathrm{Ed}}/\mathbb{F}_p$ and its quadratic twist have optimal cofactors. This algorithm simply tests the values of d in the sequence $1, 2, -2, 3, \ldots$ until a curve-twist pair is found with optimal cofactors (note that $d \neq -1$).

On input of a prime $p \equiv 1 \bmod 4$, do the following:

```
for $d \in [1, 2, -2, 3, -3, \ldots]$ do
  Compute $t_d$.
  Set $(p + 1 - t_d) = hr$ and $(p + 1 + t_d) = h'r'$, where $h = 2^e$, $h' = 2^{e'}$ and $r$ and $r'$ are odd
  if $((h = 4$ and $h' = 8)$ or $(h = 8$ and $h' = 4))$ and $r$ is prime and $r'$ is prime then
  return $b$.
end for
```

Cofactor curves for $p \equiv 3 \bmod 4$. Let t_d denote the trace of Frobenius of the curve $E_{1,d}^{\mathrm{Ed}}/\mathbb{F}_p\colon x^2 + y^2 = 1 + dx^2y^2$, i.e. $t_d = p + 1 - \#E_{1,d}^{\mathrm{Ed}}(\mathbb{F}_p)$. The following algorithm finds the integer d with smallest absolute value such that $E_{1,d}^{\mathrm{Ed}}/\mathbb{F}_p$ and its quadratic twist have optimal cofactors. This algorithm simply tests the values of d in the sequence $-1, 2, -2, 3, \ldots$ until a curve-twist pair is found with optimal cofactors (note that $d \neq 1$).

On input of a prime $p \equiv 3 \bmod 4$, do the following:

```
for $d \in [-1, 2, -2, 3, -3, ...]$ do
  Compute $t_d$.
  Set $(p + 1 - t_d) = hr$ and $(p + 1 + t_d) = h'r'$, where $h = 2^e$, $h' = 2^{e'}$ and $r$ and $r'$ are odd
  if $h = h' = 4$ and $r$ is prime and $r'$ is prime then return $b$.
end for
```

6.3 Additional security requirements

For the sake of simplicity, the generation algorithms in Section 6.1 and Section 6.2 used the curve-twist group orders as the only termination condition. However, the curve-and-twist outputs of these algorithms should be checked to satisfy additional security properties; these include a large embedding degree (some existing standards state a lower bound on this) to avoid the MOV attack [13, 10], a trace of Frobenius not equal to 1 to avoid the "smart-ASS" attack [19, 18, 20], and a large CM discriminant (again, some existing standards state a lower bound). These conditions can be checked once a curve is found. We do not include them as explicit conditions in the generation algorithms because they are satisfied with high probability for a curve output by one of the algorithms. In the rare case that one of them is not satisfied, the algorithm is simply resumed from the next curve constant.

7 Selection of primes

The rigid curve generation algorithms from Section 6 take as input a prime p such that the generated elliptic curve is defined over $\mathbb{F}_p$. The community has considered both pseudo-random primes and primes of a special shape that accelerate the modular reduction. In this section we provide a short overview of the primes used while highlighting some of their advantages and disadvantages.

7.1 Primes with special form

NIST recommends the use of five prime fields [22], all over *generalized Mersenne primes* which allow fast reduction based on the work by Solinas [21]. More recently, other special primes have gained popularity due to their improved performance. Below, we describe some of the most relevant cases.

In the following, we use the terms *canonical* to refer to representations that use exactly the minimum number of computer words required to represent field elements, and *non-canonical* to refer to redundant representations that use additional words in order to reduce carry handling operations. We remark that preferring one type of representation over the other is highly dependent on the characteristics of a particular platform and application.

1. **(Pseudo-)Mersenne primes.** These primes, which are of the form $2^\alpha - \gamma$ for $\gamma > 0$ relatively small, have gained popularity due to their excellent performance and simplicity[6]. When one uses arithmetic with a *canonical* number of limbs, it is possible to achieve practically the same performance for any $\gamma < 2^w$, where w is the targeted computer wordsize[7]; this allows flexibility to choose primes matching a target security level exactly. On the other hand, it has also been suggested that γ be a very small value (e.g., as small as 5 bits or less) in order to further reduce the cost of modular multiplications when using a *non-canonical* number of limbs. One downside of this restriction is that it may increase the search range when looking for suitable primes, which can have an impact on security or performance[8]. In general, pseudo-Mersenne primes are relatively simple to

[6] When $\gamma = 1$ these are the Mersenne primes which allow even faster modular reduction but, unfortunately, are uncommon in the range of interest for ECC (except for $2^{521} - 1$ which is used by the NIST curve P-521).

[7] If γ is very small and has a low Hamming-weight it is still possible to improve performance further on some platforms.

[8] For example, increasing the size of the prime to meet $\gamma \leq 2^5$ might affect performance in applications or platforms that favor arithmetic using the canonical number of limbs.

implement; they support compact, scalable implementations across security levels, and their performance scales well with the bitlength (as the security level increases).

2. **Solinas primes.** These primes take the form $2^a \pm 2^b \pm 1$, where $a > b$. If one fixes $a = 2b$, then the symmetric alignment makes the use of 2-way Karatsuba especially attractive for suitably chosen primes with large bitlengths, i.e. for high security levels. Nevertheless, the efficiency of this optimization strongly relies on the use of representations that have an even number of words, which may not be possible for canonical representations on many platforms or may force the use of an additional word in non-canonical representations. Moreover, given the scarcity of these primes, the chances are that a and/or b are not computer word-aligned; this can make implementations problematic on platforms that favor the use of a canonical number of limbs. All of these issues make "good" Solinas primes efficient for some platforms but suboptimal for others, and make a consistent selection of Solinas primes across security levels difficult.
3. **Montgomery-friendly primes.** These primes, which are of the form $p = 2^{\alpha}(2^{\beta} - \gamma) - 1$ for positive integers α, β and γ, can be used to speed up the Montgomery multiplication algorithm [15] and can be efficiently implemented on a wide range of platforms including 8-, 32- and 64-bit architectures. Although Montgomery-friendly primes appear to be very efficient for the 128-bit security level, they do not scale well with the bitlength and lose relative efficiency (e.g., compared with pseudo-Mersenne primes) at higher security levels. It is also an open challenge to implement efficient modular arithmetic over these primes using vector instructions.

7.2 Pseudo-random primes

Primes of a special shape are usually selected to achieve better performance in general software environments. However, in hardware or high-assurance software environments, a much broader family of side-channel attacks [11] is taken into account; compared to the "regular" software setting, these implementations need to be secure against different attack scenarios. Using primes of a special form can reduce the availability of countermeasures against certain side-channel attacks. For instance, in [9] it is shown how implementations protected with additive scalar blinding can be attacked when they use the special primes as proposed by NIST. This partially explains why the primes underlying the Brainpool curves [8] are all pseudo-random and have no special shape.

The downside of using pseudo-random primes is that software implementations tend to be more cumbersome and are expected to be roughly two times slower than implementations based on special-form primes.

Summary. When selecting the primes to underly elliptic curves, one can choose either primes which have a special shape or pseudo-random primes. The former result in faster software implementations, but may eliminate certain countermeasures which are important in hardware or high-assurance software environments. In such environments, pseudo-random primes are preferred.

Along the same lines as the summary in Section 3, when selecting curves for different security levels, it is advantageous to select the prime forms consistently. This brings many potential benefits such as scalability, consistency and algorithm/code reuse, and may ease the implementation and maintenance effort. Moreover, having the same prime shape opens up to the possibility of having very compact implementations of the field arithmetic.

8 The NUMS curves

Based on the original research we conducted (with Joppe Bos) in [7], and the arguments presented in the previous sections, we have implemented a high-performance cryptographic library[9] supporting the six curves in Table 2.

security level	prime p	Weierstrass b ($y^2 = x^3 - 3x + b$)	Edwards d ($x^2 + y^2 = 1 + dx^2y^2$)
128	$2^{256} - 189$	152961	-15342
192	$2^{384} - 317$	-34568	-11556
256	$2^{512} - 569$	121243	-78296

Table 2. NUMS curves over pseudo-Mersenne primes at three (standard) security levels.

We note that the Edwards curves in Table 2 are 4-isogenous to the original twisted Edwards curves we presented in the first (ePrint) version [6] of [7]. The subsequent versions of [6] (and also the final version [7]) and [14] employ twisted Edwards curves, some of which have larger curve constants, because of additional implementation considerations we imposed. For example, we wanted all three of the twisted Edwards curve constants in [7] to be positive (for the sake of modularity across security levels) and to correspond to the twist whose trace is positive (i.e., has the smaller group order). Based on extensive feedback and on discussions in the CFRG forum, we decided to remove these additional conditions for the sake of transparency and to revert to the original curves, but this time in ($a = 1$) Edwards form. The reason here relates to the completeness (resp. incompleteness) of the fastest Edwards (resp. $a = -1$ twisted Edwards) addition law for primes that are congruent to 3 modulo 4 – see Sections 2 and 6.

This means that the NUMS curves in Table 2 are the outputs of the deterministic curve generation processes described in Section 6, when run on input of the three pseudo-Mersenne primes above. We note that these primes are the largest 256-, 384-, and 512-bit primes respectively.

References

1. D. F. Aranha, P. S. L. M. Barreto, G. C. C. F. Pereira, and J. E. Ricardini. A note on high-security general-purpose elliptic curves. Cryptology ePrint Archive, Report 2013/647, 2013. `http://eprint.iacr.org/`.
2. D. J. Bernstein. Curve25519: New diffie-hellman speed records. In M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, editors, *PKC 2006*, volume 3958 of *LNCS*, pages 207–228. Springer, 2006.
3. D. J. Bernstein, P. Birkner, M. Joye, T. Lange, and C. Peters. Twisted Edwards curves. In S. Vaudenay, editor, *AFRICACRYPT 2008*, volume 5023 of *LNCS*, pages 389–405. Springer, 2008.
4. 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/`.
5. D. J. Bernstein, M. Hamburg, A. Krasnova, and T. Lange. Elligator: Elliptic-curve points indistinguishable from uniform random strings. In *ACM Conference on Computer and Communications Security*, 2013.

[9] See `http://research.microsoft.com/en-us/projects/nums/default.aspx`.

6. 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/2014/130`.
7. J. W. Bos, C. Costello, P. Longa, and M. Naehrig. Selecting elliptic curves for cryptography: An efficiency and security analysis. *J. Cryptographic Engineering*, 2015. `http://dx.doi.org/10.1007/s13389-015-0097-y`.
8. ECC Brainpool. ECC Brainpool Standard Curves and Curve Generation. `http://www.ecc-brainpool.org/download/Domain-parameters.pdf`, 2005.
9. B. Feix, M. Roussellet, and A. Venelli. Side-channel analysis on blinded regular scalar multiplications. In W. Meier and D. Mukhopadhyay, editors, *INDOCRYPT 2014*, volume 8885 of *LNCS*, pages 3–20. Springer, 2014.
10. G. Frey, M. Müller, and H. Rück. The tate pairing and the discrete logarithm applied to elliptic curve cryptosystems. *IEEE Transactions on Information Theory*, 45(5):1717–1719, 1999.
11. P. C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In N. Koblitz, editor, *Crypto 1996*, volume 1109 of *LNCS*, pages 104–113. Springer, Heidelberg, 1996.
12. M. Lochter, J. Merkle, J.-M. Schmidt, and T. Schütze. Requirements for standard elliptic curves. Cryptology ePrint Archive, Report 2014/832, 2014. `http://eprint.iacr.org/`.
13. A. Menezes, T. Okamoto, and S. A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5):1639–1646, 1993.
14. Microsoft Research. MSR Elliptic Curve Cryptography Library (MSR ECCLib), 2014. Available at: `http://research.microsoft.com/en-us/projects/nums`.
15. P. L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44(170):519–521, April 1985.
16. P. L. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of computation*, 48(177):243–264, 1987.
17. NIST Cryptographic Standards and Guidelines Development Process. Report and recommendations of the Visiting Committee on Advanced Technology of the National Institute of Standards and Technology. URL: `http://www.nist.gov/public_affairs/releases/upload/VCAT-Report-on-NIST-Cryptographic-Standards-and-Guidelines-Process.pdf`, July 2014.
18. T. Satoh and K. Araki. Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves. *Commentarii Math. Univ. Sancti Pauli*, 47(1):81–92, 1998.
19. I. A. Semaev. Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p. *Math. Comput.*, 67(221):353–356, 1998.
20. N. P. Smart. The discrete logarithm problem on elliptic curves of trace one. *J. Cryptology*, 12(3):193–196, 1999.
21. J. A. Solinas. Generalized Mersenne numbers. Technical Report CORR 99–39, Centre for Applied Cryptographic Research, University of Waterloo, 1999.
22. 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`.