A Family of Lightweight Twisted Edwards Curves for the Internet of Things

Sankalp Ghatpande, Johann Grossschädl, Zhe Liu

Lightweight Public-Key Cryptography for the Internet of Things

Sankalp Ghatpande[1], Johann Großschädl[1], and Zhe Liu[2]

[1] Interdisciplinary Centre for Security, Reliability and Trust (SnT),
University of Luxembourg, Esch-sur-Alzette, Luxembourg
{sankalp.ghatpande,johann.groszschaedl}@uni.lu
[2] College of Computer Science and Technology,
Nanjing University of Aeronautics and Astronautics, Nanjing, China
zhe.liu@nuaa.edu.cn

Abstract. We introduce a set of four twisted Edwards curves that satisfy common security requirements and allow for fast implementations of scalar multiplication on 8, 16, and 32-bit processors. Our curves are defined by an equation of the form $-x^2 + y^2 = 1 + dx^2y^2$ over a prime field $\mathbb{F}_p$, where d is a small non-square modulo p. The underlying prime fields are based on "pseudo-Mersenne" primes given by $p = 2^k - c$ and have in common that $p \equiv 5 \bmod 8$, k is a multiple of 32 minus 1, and c is at most eight bits long. Due to these common features, our primes facilitate a parameterized implementation of the low-level arithmetic so that one and the same arithmetic function is able to process operands of different length. The four twisted Edwards curves we present in this paper are all birationally equivalent to Montgomery curves of the form $-(A+2)y^2 = x^3 + Ax^2 + x$ where $4/(A+2)$ is small. Even though this contrasts with the usual practice of choosing $(A+2)/4$ to be small, we show that the Montgomery form of our curves allows for an equally efficient implementation of point doubling as Curve25519. The four curves we put forward roughly match the common symmetric security levels of 80, 96, 112, and 128 bits. Moreover, their Weierstraß representations are isomorphic to curves of the form $y^2 = x^3 - 3x + b$ so as to facilitate inter-operability with TinyECC and other legacy software.

1 Introduction

An elliptic curve E has to satisfy various security and efficiency requirements to be suitable for cryptographic applications [6, 10, 16]. Most importantly, the group of rational points on the curve must contain a (large) subgroup of prime order since this order determines the computational cost of the Elliptic Curve Discrete Logarithm Problem (ECDLP). However, determining whether a curve has a near-prime cardinality requires one to count the number of points on the curve, which is a complicated and computation-intensive endeavor [9]. Therefore, it is common practice to use "standardized" curves that were generated to meet certain security requirements. A multitude of national and international standardization bodies, including the U.S. National Institute of Standards and

Technology (NIST), have published a set of recommended domain parameters for elliptic curves of different cryptographic strength, in most cases comparable to that of 128, 192, and 256-bit AES [14, 25]. The so-called NIST curves were allegedly generated by Jerry Solinas in the late 1990s, who was working for the National Security Agency (NSA) at that time [7]. Five of the NIST curves are defined over prime fields and given by a Weierstraß equation of the form

$$E_W : \; y^2 = x^3 + a_4 x + a_6 \tag{1}$$

with a_4 fixed to -3 for efficiency reasons [20]. However, the Weierstraß form is, in terms of performance, not state-of-the-art anymore since alternative curve models or special families of curves allow for faster execution times. For example, the addition law of twisted Edwards curves is much more efficient than that of normal Weierstraß curves and has the further advantage of completeness if certain conditions are met [3, 21]. On the other hand, the so-called GLV curves feature an efficiently-computable endomorphism, which can be utilized to speed up variable-base scalar multiplication [17, 20].

In this paper, we present a set of four twisted Edwards curves over pseudo-Mersenne prime fields that we generated in a transparent and verifiable way to meet common security and efficiency requirements. These four curves, which we call *LiTE curves* (an abbreviation for Lightweight Twisted Edwards), provide security levels of about 80, 96, 112, and 128 bits, respectively, and are suitable for IoT applications running on restricted devices. Using curves that offer less than 128 bits of security allows for large savings in execution time and makes particular sense for applications with low or medium security requirements. The four twisted Edwards curves we present in this paper differ from the Edwards curves introduced by Aranha et al. in [1] in three important aspects. First, we chose the prime fields and generated the curves with the goal of having consistency across security levels, which means they share many basic properties like the group structure. Most notably, all our curves are defined over prime fields with $p = 2^k - c$ elements and have in common that k is a multiple of 32 minus 1 (i.e. $k = 159$, 191, 223, or 255) and c has a length of at most eight bits. This consistency facilitates a parameterized implementation[3] of the field-arithmetic operations, which minimizes the code size when different security levels are to be supported and has some other benefits like reduced development cost. The second difference is that we aimed for curves capable to reach top performance with the twisted Edwards representation *and* the birationally-equivalent Montgomery representation. Aranha et al. [1], on the other hand, specified two sets of curves, namely Montgomery curves with a small parameter A and Edwards curves with a small parameter d; in both cases the rationale was to improve the arithmetic performance. The four twisted Edwards curves we put forward have a small parameter d and a fixed to -1, which implies the parameter A of the

[3] A parameterized implementation of a field-arithmetic operation can support fields of different order (i.e. fields of different bit length), typically in steps of 32 bits. The parameters include besides the operands (or pointers to operands held in RAM) an additional parameter that specifies the length of the operands.

birationally-equivalent Montgomery curves has the property that $4/(A-2)$ is small. While this contrasts with the usual choice of $(A-2)/4$ being small, it is possible to perform a point doubling equally fast as on e.g. Curve25519 thanks to a simple modification of the doubling formula. Finally, the third difference between our curves and those from [1] is that we took potential vulnerabilities to side-channel attacks [18] into account when we chose the base point (i.e. the generator of a prime-order subgroup). In particular, we excluded points whose coordinates have an extraordinary low Hamming weight.

2 Preliminaries

In 1987, Peter Montgomery introduced a new model for elliptic curves and demonstrated its practical use by speeding up algorithms for integer factorization [24]. Formally, a so-called *Montgomery curve* over a non-binary field $\mathbb{F}_q$ can be described through the equation

$$E_M: \; By^2 = x^3 + Ax^2 + x \tag{2}$$

where $A, B \in \mathbb{F}_q$ and $A \neq \pm 2$, $B \neq 0$ (or, equivalently, $B(A^2-4) \neq 0$). Curves of such form allow a full scalar multiplication $k \cdot P$ to be carried out using the x coordinate only, which is clearly more efficient than when both the x and the y coordinate are involved in the point arithmetic. A point $P \in E_M(\mathbb{F}_q)$ given in projective coordinates of the form $(X:Z)$ can be doubled with only three multiplications (3M) and two squarings (2S) in the underlying finite field. On the other hand, a differential addition of two points (i.e. the calculation of the sum $P+Q$ of two points $P, Q \in E_M(\mathbb{F}_q)$ whose difference $P-Q$ is known) requires two multiplications (2M), two squarings (2S), as well as a multiplication by the constant $(A+2)/4$. The so-called Montgomery ladder for scalar multiplication has a total computational cost of roughly $5n$ multiplications and $4n$ squarings for an n-bit scalar, i.e. $5\text{M} + 4\text{S}$ per bit [2].

Exactly 20 years after Montgomery's discovery, Harold Edwards introduced a normal form to describe certain elliptic curves, which have become known as Edwards curves in recent years [15]. Bernstein and Lange showed that curves in Edwards form have good cryptographic properties with respect to performance and protection against side-channel attacks [5]. *Twisted Edwards curves* (in the following abbreviated as "TE curves") were presented in [3] as a generalization of Edwards curves with similarly good implementation properties. A TE curve over a non-binary field $\mathbb{F}_q$ is defined by the equation

$$E_T: \; ax^2 + y^2 = 1 + dx^2y^2 \tag{3}$$

where a and d are distinct elements of $\mathbb{F}_q^*$. The additive group $E_T(\mathbb{F}_q)$ contains a neutral element $\mathcal{O}$, namely the point $(0,1)$, which, under some conditions, can be used as an input to the addition formula specified in [3]. More precisely, the addition law from [3] is complete when a is a square and d a non-square in the underlying field $\mathbb{F}_q$. Here, completeness refers to the property that the addition

formula produces the correct result for any pair $P, Q \in E_T(\mathbb{F}_q)$, including the corner cases $P = \mathcal{O}$, $Q = \mathcal{O}$, and $P = Q$. Hişil et al. introduced in [21] extended projective coordinates for TE curves, the currently fastest means of performing a (non-differential) point addition on an elliptic curve. When using a TE curve with $a = -1$, two points can be added by executing only seven multiplications (7M) in the underlying field, while the point doubling operation requires three multiplications (3M) and four squarings (4S) [12, 19].

Montgomery curves and TE curves are closely related due to the fortunate fact that every Montgomery curve over $\mathbb{F}_q$ is birationally equivalent over $\mathbb{F}_q$ to a TE curve and vice versa [3]. More concretely, when a, d are distinct non-zero elements of $\mathbb{F}_p$, the TE curve E_T given by Eq. (3) is birationally equivalent over $\mathbb{F}_p$ to the Montgomery curve E_M given by Eq. (2) with the parameters

$$A = \frac{2(a+d)}{a-d} \quad \text{and} \quad B = \frac{4}{a-d}. \tag{4}$$

An affine point (x_t, y_t) on a TE-form elliptic curve E_T can be converted to the corresponding point (x_m, y_m) on the birationally-equivalent Montgomery curve E_M using the following map, which is from [3].

$$\phi : (x_t, y_t) \mapsto (x_m, y_m) = \left(\frac{1+y_t}{1-y_t}, \frac{1+y_t}{(1-y_t)x_t} \right) \tag{5}$$

Bernstein et al. demonstrated in [3] not only that every TE curve is birationally equivalent to a Montgomery curve, but also that the converse holds. In concrete terms, when $A \in \mathbb{F}_p \setminus \{-2, 2\}$ and $B \in \mathbb{F}_p^*$, then the Montgomery curve E_M given by Eq. (2) is birationally equivalent over $\mathbb{F}_p$ to the TE curve given by Eq. (3) with the parameters

$$a = \frac{A+2}{B} \quad \text{and} \quad d = \frac{A-2}{B}. \tag{6}$$

This curve always exists since $A \neq \pm 2$ and $B \neq 0$. Given a point (x_m, y_m) on the Montgomery curve E_M, one can compute the corresponding point on the birationally-equivalent TE curve E_T via the map

$$\psi : (x_m, y_m) \mapsto (x_t, y_t) = \left(\frac{x_m}{y_m}, \frac{x_m - 1}{x_m + 1} \right). \tag{7}$$

3 LiTE Curves

An elliptic curve E over a prime field $\mathbb{F}_p$ is completely specified by the prime p and the two coefficients of its defining equation, which can, depending on the curve model, be e.g. Eq. (1), Eq. (2), or Eq. (3). However, one coefficient is, in practice, often fixed to a specific value for reasons of simplicity or performance [13]. As already mentioned in Sect. 1, the NIST curves (and many other curves in Weierstraß form) use $a_4 = -3$ because this choice allows one to minimize the

computational cost of point doubling in projective coordinates [11, 20]. On the other hand, when generating TE curves, it is common practice to fix a to -1 so that implementers can exploit the full potential of the "extended" coordinates proposed in [21] and perform a mixed addition with only seven multiplications in $\mathbb{F}_p$. Also the four LiTE curves we put forward follow this approach and have the coefficient a set to -1, which means the curve-generation process consists of finding a suitable prime p and coefficient d.

3.1 Selection of Prime Fields

An analysis of recent proposals for new elliptic curves shows that the underlying fields are based on three main categories of primes: generalized-Mersenne primes, pseudo-Mersenne primes, and primes for which Montgomery reduction can be optimized, i.e. "Montgomery-friendly" primes [13]. Taking various efficiency and (side-channel) security aspects to into account, pseudo-Mersenne primes seem to be particularly attractive because they were used in the majority of proposals for new elliptic curves, including [1, 2, 4, 8, 10, 26]. Formally, a pseudo-Mersenne prime has the form $p = 2^k - c$ where c is small in relation to 2^k. The reduction of a $2k$-bit integer x modulo $p = 2^k - c$ requires just a multiplication of the upper half of x (i.e. the k most significant bits of x) by c, followed by an addition of the product to the lower half of x (see e.g. [10] for more details). Besides high arithmetic efficiency, pseudo-Mersenne primes have the virtue of minimizing the surface for side-channel attacks since the reduction can be easily implemented to have constant execution time, irrespective of the actual value of x [2, 10].

Now that the basic form of the primes is fixed to $p = 2^k - c$, the next step is to determine the actual values for the exponent k and constant c. Since we aim for elliptic curves providing security levels of (approximately) 80, 96, 112, and 128 bits, their cardinalities need to contain a large prime factor of magnitude 2^{160}, 2^{192}, 2^{224}, and 2^{256}, respectively, which requires due to Hasse's theorem [20] that the underlying prime fields have about the same order. This suggests to use $k = 160$, 192, 224, and 256, yielding primes whose bit-lengths are a multiple of 32, similar to the NIST primes. However, choosing the values for k in this way does not necessarily lead to the maximum arithmetic performance in software. Namely, as demonstrated in [2], it can be beneficial to use primes with a bit-length that is a tad below the "nominal" bit-length for the targeted security level, e.g. a 255-bit prime instead of a 256-bit prime. Having one bit of "headroom" simplifies the implementation of the field arithmetic when one aims for both high performance and resistance to side-channel attacks through constant (i.e. operand-independent) execution time [10]. Therefore, we decided to fix the exponents to $k = 159$, 191, 223, and 255.

The final step in the process of selecting a pseudo-Mersenne prime is to determine the constant c, which is commonly chosen as the smallest integer so that $p = 2^k - c$ is prime. An additional criterion often taken into consideration when choosing c is the congruence class of p modulo 4, i.e. whether $p \equiv 3 \bmod 4$ or $p \equiv 5 \bmod 8$ (which implies $p \equiv 1 \bmod 4$). In the former case, it is possible to find a TE curve such that both the curve and its quadratic twist have a minimal

co-factor of 4 [22]. Unfortunately, -1 is always a non-square modulo such a prime and, therefore, the fast addition formulae for TE curves specified in [21] are not guaranteed to be complete. On the other hand, if $p \equiv 5 \bmod 8$, then -1 is always a square modulo p and the fast point-addition formulae from [21] are complete (i.e. produce the correct result for any pair of $\mathbb{F}_p$-rational points), provided the curve parameter d is a non-square.

Taking all the above into consideration, we opted to choose the four values for the constant c as the smallest positive integers that yield pseudo-Mersenne primes congruent to 5 modulo 8. The four primes we obtained in this way are $2^{159} - 91$, $2^{191} - 19$, $2^{223} - 235$, and $2^{255} - 19$. As $p \equiv 5 \bmod 8$ always implies $p \equiv 1 \bmod 4$, it is guaranteed that -1 is a square in $\mathbb{F}_p$. A TE curve over these four prime fields can safely use Hisil et al's highly-optimized addition formulae for $a = -1$ without compromising completeness [21]. The four pseudo-Mersenne primes we put forward share the following three basic features, which facilitate a "parameterized" implementation of the field arithmetic: (i) the exponent k is a multiple of 32 minus 1, (ii) the constant c is at most eight bits long, (iii) p is congruent to 5 modulo 8 and, consequently, -1 is a square modulo p.

3.2 Requirements and Formal Definition

In this subsection, we first explain our preference for TE curves rather than Montgomery curves and then discuss the objectives of the curve generation process, namely to obtain curves that are secure, arithmetically efficient, consistent across security levels, and compatible with with various curve models and coordinate systems for point representation. Then, we give a formal definition of LiTE curves and describe the requirements a LiTE curve has to satisfy.

New elliptic curves for cryptographic purposes should be generated in an open, transparent, and reproducible way to increase their prospects of finding widespread acceptance in the cryptographic community and general public. The foundation of such a curve-generation process is a set of well-explained and properly-specified requirements that the curves have to meet. Before describing these requirements for our LiTE curves, we outline the objectives we aimed for with our curve-generation process. First and foremost, the curves shall be secure in the sense of not having a weakness that would allow an adversary to compute discrete logarithms in less than the $0.886\sqrt{n}$ steps needed by Pollard's rho method [6]. The second requirement is to enable high-speed implementations and facilitate state-of-the-art optimization techniques for both the field and group arithmetic. In particular, we aim for curves that achieve peak performance not only with the TE model, but also when using the birationally-equivalent Montgomery representation. Our third requirement is consistency among security levels, which means the curves should share certain properties regarding the structure of the elliptic-curve groups (co-factor, twist-security, etc.) and the underlying fields. This consistency enables a "parameterized" software implementation of the group arithmetic (e.g. point addition, point doubling) and scalar multiplication so that one and the same arithmetic function can be used for groups of different order, which significantly reduces the code size compared to an implementation with

separate functions for each curve. Finally, the fourth requirement is compatibility with various other curve models, which includes the flexibility to support different coordinate systems.

Our fourth requirement includes the flexibility to support efficient conversions between different representations of points, not only between TE and Montgomery form, but also between TE or Montgomery form and Weierstraß form. There are several scenarios where the latter conversion can be useful. One such scenario is discussed in [27] and concerns the instantiation of widely-used cryptosystems like ECDSA with a TE or Montgomery curve that is expressed through its Weierstraß form. Many elliptic-curve schemes standardized by international organizations such as the NIST, IEEE, ISO, and ANSI require curve points to be represented in Weierstraß coordinates. However, this does not rule out TE or Montgomery curves since every elliptic curve admits to a Weierstraß equation and most standards also tolerate small co-factors. Therefore, it is possible to instantiate e.g. ECDSA with Wei25519 (a Weierstraß representation of Curve25519, see [27]), which would allow one to use the efficient Edwards addition law for the point arithmetic and scalar multiplication. On the other hand, there exist also situations where using Montgomery or TE coordinates as "wire format" and Weierstraß coordinates for the computation can be necessary. This situation occurs if one wants to implement a state-of-the-art cryptosystem like Curve25519-based key exchange [2] or the EdDSA signature system, but is forced to use a legacy hardware component or software library for the point arithmetic and scalar multiplication that supports only the Weierstraß model. A well-known example for such a legacy software is TinyECC [23], a lightweight ECC library originally developed for wireless sensor networks that continues to be widely used today. However, TinyECC, like most other legacy ECC software, only supports the Weierstraß model for the point arithmetic, which requires conversions between TE/Montgomery and Weierstrass coordinates.

Since we have already chosen the prime fields for the LiTE curves and fixed the coefficient a to -1, the curve-generation process boils down to finding a coefficient d that satisfies all security and efficiency requirements explained above. In fact, all these requirements can be condensed to five conditions on d, which are summarized in the following formal definition of LiTE curves.

Definition 1. *Let $\mathbb{F}_p$ be a prime field with $p \equiv 5 \bmod 8$. A LiTE elliptic curve is a twisted Edwards curve over $\mathbb{F}_p$ given by the equation*

$$E_T : \; -x^2 + y^2 = 1 + dx^2y^2 \tag{8}$$

where d is the smallest element of $\mathbb{F}_p \setminus \{-1, 0\}$ so that the following five conditions are met

1. *d is a non-square in $\mathbb{F}_p$*
2. *E_T has a co-factor of 8 and negative trace (i.e. $\#E_T(\mathbb{F}_p) = 8n > p$), while its quadratic twist E'_T has a co-factor of 4 and positive trace*
3. *E_T and E'_T have a large embedding degree as recommended in [14]*
4. *E_T has a large CM field discriminant as recommended in [6]*

5. *the Weierstraß representation of E_T is isomorphic to a curve defined by an equation of the form $y^3 = x^3 - 3x + b$ where b is a non-square in $\mathbb{F}_p$*

The first condition entails that the TE addition law is complete, which is an efficiency requirement for our curves. On the other hand, the second condition fulfills a security requirement since it guarantees that both the curve and its quadratic twist contain a cyclic subgroup of large order; consequently, the curve is twist-secure. Also the third and fourth condition are linked to security requirements since they ensure that there exist no additive or multiplicative transfers that would allow an adversary to take a "shortcut" when solving the ECDLP. All these security requirements are fairly common and have already been considered in many other curve-generation efforts, most notably [6]. Finally, the fifth condition implies that a LiTE curve can be expressed through a short Weierstraß equation with efficient coefficients as discussed above.

We used the computer algebra system Magma to obtain the coefficient d for each of the four security levels we consider in this paper. More concretely, we wrote a Magma script that essentially consists of a loop that checks in each iteration whether d meets the five conditions defined above and increments d by 1 if it is not the case. This script generated the following coefficients, which define our four LiTE curves:

$$\begin{aligned}
-x^2 + y^2 &= 1 + 49445x^2y^2 \bmod 2^{159} - 91 \\
-x^2 + y^2 &= 1 + 141087x^2y^2 \bmod 2^{191} - 19 \\
-x^2 + y^2 &= 1 + 987514x^2y^2 \bmod 2^{223} - 235 \\
-x^2 + y^2 &= 1 + 4998299x^2y^2 \bmod 2^{255} - 19
\end{aligned}$$

Birationally-Equivalent Montgomery Curves. For a TE curve with $a = -1$, the coefficients A and B of the birationally-equivalent Montgomery curve are as follows

$$A = \frac{2(a+d)}{a-d} = \frac{2(1-d)}{1+d} \tag{9}$$

$$B = \frac{4}{a-d} = -\frac{4}{1+d} = -\frac{2(1-d)+2(1+d)}{1+d} = -(A+2) \tag{10}$$

Unfortunately, these coefficients do not meet the usual efficiency criteria for Montgomery curves since, when d is small, one can not expect that A is small and congruent to 2 modulo 4. However, we found that the the reciprocal of $(A + 2)/4$, namely $4/(A + 2)$ is small when $a = -1$ and d is small. More concretely, due to Eq. (10) we have $4/(1 + d) = A + 2$ and $4/(A + 2) = d + 1$, which means $4/(A + 2)$ is small when d is small.

$$4X_nZ_n = (X_n + Z_n)^2 - (X_n - Z_n)^2 \tag{11}$$

$$X_{2n} = (X_n + Z_n)^2(X_n - Z_n)^2 \tag{12}$$

$$Z_{2n} = (4X_nZ_n)\left[(X_n - Z_n)^2 + ((A+2)/4)\,(4X_nZ_n)\right] \tag{13}$$

Montgomery provided in his seminal paper [24] the above formulae for the doubling of a point given in projective $(X : Z)$ coordinates. The computation

of $4X_nZ_n$ requires two squarings (2S) in $\mathbb{F}_p$, and then the computation of X_{2n} and Z_{2n} takes one multiplication (1M) each, which means the overall cost of the point doubling amounts to $2M + 2S$. Furthermore, a multiplication by the constant $(A + 2)/4$ is required, which can be performed much faster than a conventional multiplication in $\mathbb{F}_p$ when A is small and congruent to 2 modulo 4. Fortunately, these formulae can be easily modified to make them more amenable for the Montgomery representations of our LiTE curves, which have the property that $4/(A + 2)$ is small. Namely, by simply multiplying both X_{2n} and Z_{2n} by $4/(A + 2)$, we get the modified doubling formulae below, which do not contain a multiplication by $(A + 2)/4$ anymore. Note that this modification does not change the affine x-coordinate $x_{2n} = X_{2n}/Z_{2n}$ and, thus, we can safely use these formulae for the computation of a scalar multiplication based on the Montgomery ladder. Similar as with the original doubling formulae, $4X_nZ_n$ is computed first and then the product of $(X_n - Z_n)^2$ and $4/(A+2)$ is formed. This product serves then as input for the computation of X_{2n} and Z_{2n}, respectively, which means the overall cost amounts to $2M + 2S$ and a multiplication by the small constant $4/(A + 2)$. Apart from that, two additions and two subtractions in $\mathbb{F}_p$ have to be carried out, exactly as with the original formulae. In summary, performing a scalar multiplication on the Montgomery curves that are birationally-equivalent to our LiTE curves requires exactly the same number of $\mathbb{F}_p$-operations as when a Montgomery curves with a small coefficient A and $B = 1$ is used, e.g. Curve25519.

$$X_{2n} = (X_n + Z_n)^2 (X_n - Z_n)^2 \, (4/(A + 2)) \tag{14}$$

$$\begin{aligned} Z_{2n} &= (4X_nZ_n) \left[(X_n - Z_n)^2 + ((A + 2)/4) \, (4X_nZ_n) \right] (4/(A + 2)) \\ &= (4X_nZ_n) \left[(X_n - Z_n)^2 \, (4/(A + 2)) + (4X_nZ_n) \right] \end{aligned} \tag{15}$$

Base Points. Besides the coefficients of the curve equation and the underlying finite field, domain parameters for ECC also specify a base point $P \in E(\mathbb{F}_p)$ that serves as generator of a cyclic (sub)group. From a theoretical point of view, P has to satisfy only a single requirement, namely to have prime order. It is common practice to choose the point with the smallest x-coordinate as base point; for example the base points of the TE curves specified in [8] were determined in this way. Unfortunately, this practice is problematic from a side-channel perspective, especially when a software implementation of the so-called comb method [20] for fixed-base scalar multiplication is executed on a microcontroller with an early-terminating integer multiplier, e.g. ARM Cortex-M3. Namely, as shown in [18], the so-called early-termination effect (which makes the latency of multiply instructions operand-dependent) can introduce vulnerabilities to timing analysis and Simple Power Analysis (SPA) attacks, even if the field arithmetic has been implemented with the goal of having constant execution time. In order to avoid this "side-channel pitfall," we chose the base points of the LiTE curves in such a way that the early-termination mechanism can not be triggered when a point addition by P (or a small multiple of P) is carried out. This ensures the execution time of the comb method does not leak any information about the scalar.

References

1. D. F. Aranha, P. S. Barreto, G. C. Pereira, and J. E. Ricardini. A note on high-security general-purpose elliptic curves. Cryptology ePrint Archive, Report 2013/647, 2013. Available for download at `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, *Public Key Cryptography — PKC 2006*, volume 3958 of *Lecture Notes in Computer Science*, pages 207–228. Springer Verlag, 2006.
3. D. J. Bernstein, P. Birkner, M. Joye, T. Lange, and C. Peters. Twisted Edwards curves. In S. Vaudenay, editor, *Progress in Cryptology — AFRICACRYPT 2008*, volume 5023 of *Lecture Notes in Computer Science*, pages 389–405. Springer Verlag, 2008.
4. D. J. Bernstein, C. Chuengsatiansup, and T. Lange. Curve41417: Karatsuba revisited. In L. Batina and M. Robshaw, editors, *Cryptographic Hardware and Embedded Systems — CHES 2014*, volume 8731 of *Lecture Notes in Computer Science*, pages 316–334. Springer Verlag, 2014.
5. D. J. Bernstein and T. Lange. Faster addition and doubling on elliptic curves. In K. Kurosawa, editor, *Advances in Cryptology — ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 29–50. Springer Verlag, 2007.
6. D. J. Bernstein and T. Lange. SafeCurves: Choosing safe curves for elliptic-curve cryptography. Avialable online at `http://safecurves.cr.yp.to`, 2013.
7. D. J. Bernstein and T. Lange. Security dangers of the NIST curves. Presentation given at the 3rd Workshop on International View of the State-of-the-Art of Cryptography and Security and its Use in Practice, May 30–31, 2013, Athens, Greece. Slide deck available online at `http://www.hyperelliptic.org/tanja/vortraege/20130531.pdf`, 2013.
8. B. Black, J. W. Bos, C. Costello, P. Longa, and M. Naehrig. Elliptic curve cryptography (ECC) nothing up my sleeve (NUMS) curves and curve generation. Internet Engineering Task Force, Network Working Group, Internet draft draft-black-numscurves-02 (work in progress), Feb. 2015.
9. I. F. Blake, G. Seroussi, and N. P. Smart. *Elliptic Curves in Cryptography*, volume 265 of *London Mathematical Society Lecture Notes Series*. Cambridge University Press, 1999.
10. J. W. Bos, C. Costello, P. Longa, and M. Naehrig. Selecting elliptic curves for cryptography: An efficiency and security analysis. *Journal of Cryptographic Engineering*, 6(4):259–286, Nov. 2016.
11. E. Brier and M. Joye. Fast point multiplication on elliptic curves through isogenies. In M. P. Fossorier, T. Høholdt, and A. Poli, editors, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes — AAECC 2003*, volume 2643 of *Lecture Notes in Computer Science*, pages 43–50. Springer Verlag, 2003.
12. D. Chu, J. Großschädl, Z. Liu, V. Müller, and Y. Zhang. Twisted Edwards-form elliptic curve cryptography for 8-bit AVR-based sensor nodes. In S. Xu and Y. Zhao, editors, *Proceedings of the 1st ACM Workshop on Asia Public-Key Cryptography (AsiaPKC 2013)*, pages 39–44. ACM Press, 2013.
13. C. Costello, P. Longa, and M. Naehrig. A brief discussion on selecting new elliptic curves. Technical Report MSR-TR-2015-46, Microsoft Research, June 2015. Available for download at `http://research.microsoft.com/apps/pubs/default.aspx?id=246915`.

14. ECC Brainpool Consortium. ECC Brainpool standard curves and curve generation. Avialable for download at `http://www.ecc-brainpool.org/download/Domain-parameters.pdf`, 2005.
15. H. M. Edwards. A normal form for elliptic curves. *Bulletin of the American Mathematical Society*, 44(3):393–422, July 2007.
16. J.-P. Flori, J. Plût, J.-R. Reinhard, and M. Ekerå. Diversity and transparency for ECC. Cryptology ePrint Archive, Report 2015/659, 2015. Available for download at `http://eprint.iacr.org`.
17. R. P. Gallant, R. J. Lambert, and S. A. Vanstone. Faster point multiplication on elliptic curves with efficient endomorphism. In J. Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 190–200. Springer Verlag, 2001.
18. J. Großschädl, E. Oswald, D. Page, and M. Tunstall. Side-channel analysis of cryptographic software via early-terminating multiplications. In D. Lee and S. Hong, editors, *Information Security and Cryptology — ICISC 2009*, volume 5984 of *Lecture Notes in Computer Science*, pages 176–192. Springer Verlag, 2010.
19. M. Hamburg. Fast and compact elliptic-curve cryptography. Cryptology ePrint Archive, Report 2012/309, 2012. Available for download at `http://eprint.iacr.org`.
20. D. R. Hankerson, A. J. Menezes, and S. A. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer Verlag, 2004.
21. H. Hişil, K. K.-H. Wong, G. Carter, and E. Dawson. Twisted Edwards curves revisited. In J. Pieprzyk, editor, *Advances in Cryptology — ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 326–343. Springer Verlag, 2008.
22. A. Langley, M. Hamburg, and S. Turner. Elliptic curves for security. Internet Engineering Task Force, Internet Research Task Force, RFC 7748, Jan. 2016.
23. A. Liu and P. Ning. TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks. In *Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN 2008)*, pages 245–256. IEEE Computer Society Press, 2008.
24. P. L. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of Computation*, 48(177):243–264, Jan. 1987.
25. National Institute of Standards and Technology (NIST). Recommended Elliptic Curves for Federal Government Use. White paper, available for download at `http://csrc.nist.gov/encryption/dss/ecdsa/NISTReCur.pdf`, July 1999.
26. M. Scott. Ed3363 (HighFive) – An alternative elliptic curve. Cryptology ePrint Archive, Report 2015/991, 2015. Available for download at `http://eprint.iacr.org`.
27. R. Struik. Alternative elliptic curve representations. Internet Engineering Task Force, Light-Weight Implementation Guidance (LWIG) Working Group, Internet draft draft-struik-lwip-curve-representations-00 (work in progress), Oct. 2017.