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

Sankalp Ghatpande, Johann Großschädl, and Zhe Liu

CSC and SnT, University of Luxembourg
6, Avenue de la Fonte, L–4364 Esch-sur-Alzette, Luxembourg
{sankalp.ghatpande,johann.groszschaedl,zhe.liu}@uni.lu

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. Each of the twisted Edwards curves we introduce in this paper is birationally equivalent to a Montgomery curve 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 such that $(A+2)/4$ is 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 security levels of 80, 96, 112 and 128 bits. In addition, 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

Elliptic Curve Cryptography (ECC), introduced independently by Neal Koblitz [22] and Victor Miller [26] in the mid-1980s, is nowadays widely considered the most viable alternative to RSA and other traditional public-key cryptosystems [10]. The main attraction of ECC is the absence of a subexponential-time algorithm for solving the Discrete Logarithm Problem (DLP) on a general elliptic curve over a finite field [8, 20]. Therefore, elliptic curve cryptosystems can use much smaller groups than their "classical" DLP-based counterparts to achieve a certain level of security. Smaller groups normally implies shorter keys and, in turn, savings in execution time, energy consumption, memory requirements, as well as transmission bandwidth, all of which is important in the embedded and mobile domains. The expansion of the Internet of Things (IoT) in recent years has created a strong demand for lightweight implementations of ECC that can

accommodate the stringent resource constraints of wireless sensor nodes, RFID tags, and various other kinds of smart devices [31]. According to the Ericsson Mobility Report from November 2017 [16], the number of devices connected to the Internet is expected to grow from roughly 19 billion in 2018 to more than 30 billion by the end of 2023. However, only about one third of these 30 billion devices will be classical computers (PCs, laptops, tablets, smart phones), while the remaining two thirds (i.e. around 20 billion devices) will be related to the IoT. Consequently, in a few years, "things" like machines, meters, point-of-sale terminals, consumer electronics, electromechanical sensors, actuators, wearable gadgets, and medical devices will most likely account for far more deployments of ECC than classical computers.

An elliptic curve has to satisfy various security and efficiency requirements to be suitable for cryptographic algorithms [6, 9, 13, 17]. 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 [20]. Therefore, it is common practice to use "standardized" curves that were generated to meet certain security requirements. A multitude of standardization bodies has recommended domain parameters for elliptic curves of different cryptographic strength, in most cases comparable to that of 128, 192, and 256-bit AES. The currently most important and widely-used curves are the ones specified by the US National Institute of Standards and Technology (NIST) [28], which provide security levels in the range of 80 to 256 bits. These so-called NIST curves were allegedly generated by Jerry Solinas in the 1990s, who was an employee of 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

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

where a_4 fixed to -3 for efficiency reasons [20]. However, the Weierstraß form is performance-wise not state-of-the-art anymore since alternative curve models for special families of curves allow for much faster execution times.

Two examples of special elliptic curves with excellent arithmetic properties are (twisted) Edwards curves [3, 15] and Montgomery curves [27]. The addition law of twisted Edwards curves is much more efficient than that of conventional Weierstraß curves and has the further advantage of completeness when certain conditions are met [5]. Also Montgomery curves are attractive for practical use due to an extremely simple, yet very fast, scalar multiplication technique, the so-called Montgomery ladder [27]. In the recent past, a number of new curves in Edwards or Montgomery form, most of them defined over a pseudo-Mersenne prime field, have been published, e.g. [1, 4, 9, 19, 29]. Almost all of these curves target security levels of 128 bits and above, which is somewhat surprising given the rapid proliferation of the IoT along with the fact that many applications in such domains as home automation and consumer electronics do not really have

high security requirements. The only proposals for smaller curves we are aware of came from Aranha et al., who introduced in [1, Sect. A] Montgomery curves over 159 and 191-bit prime fields, as well as Edwards curves over 157, 168, and 191-bit fields, respectively.

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[1] 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 two important aspects. Firstly, 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[2] 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 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.

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

[1] For example, the results in [25] show that a scalar multiplication on a 192-bit elliptic curve (providing about 96 bits of security) takes less than half of the execution time of a scalar multiplication on a 256-bit curve (128 bits of security).

[2] 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.

[27]. 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 an overall computational cost of about 5ℓ multiplications and 4ℓ squarings for an ℓ-bit scalar, i.e. 5M + 4S per bit [2].

Exactly 20 years after Montgomery's discovery, Harold Edwards introduced a normal form to describe certain elliptic curves, which have since then become known as Edwards curves [15]. Bernstein and Lange [5] showed that curves in Edwards form have good cryptographic properties with respect to performance and protectability against side-channel attacks. *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 $\mathcal{O} = (0, 1)$, which can, under some conditions, be used as an input to the addition formula given in [3]. More concretely, when a is a square and d a non-square in the underlying field $\mathbb{F}_q$, then the addition law from [3] is *complete* and yields the correct sum 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. presented in [21] extended projective coordinates, the currently fastest means of point addition on a curve in TE form. When $a = -1$, then a "mixed" addition $P + Q$, where P is given in extended projective coordinates and Q in extended affine coordinates, requires seven multiplications (7M) in $\mathbb{F}_q$, while the cost of a point doubling amounts to three multiplications (3M) and four squarings (4S) [12, 18].

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]. Specifically, if a, d are distinct and non-zero in $\mathbb{F}_p$, then 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

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

Bernstein et al. demonstrated in [3] that also the converse holds. Namely, 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) where

$$a = \frac{A+2}{B} \quad \text{and} \quad d = \frac{A-2}{B}. \tag{5}$$

This curve always exists since $A \neq \pm 2$ and $B \neq 0$. In some sense, the TE form and Montgomery form complement each other in an optimal way and facilitate so the implementation of elliptic-curve cryptosystems. The Montgomery shape is well suited for scalar multiplication $k \cdot P$ with a variable base point (i.e. P is not known in advance), but little attractive in settings where P is fixed. Fortunately, such fixed-base scalar multiplication is exactly the domain in which the TE shape excels. Various algorithms for scalar multiplication with a fixed base point, such as the comb method or window method [20], are extremely fast on TE-form elliptic curves due to the high efficiency of the addition law [3, 9]. The birational equivalence between these two curve models is particularly useful in ephemeral ECDH key exchange [20], where each involved entity has to perform a fixed-base scalar multiplication (to generate an ephemeral key pair) as well as a variable-base scalar multiplication (to get the shared secret). The former can be efficiently computed on a curve in TE form using e.g. a comb method, while the latter can take advantage of the simple yet fast Montgomery ladder on the birationally-equivalent Montgomery curve (see [25] for details).

3 LiTE Curves

We decided to base our new curves for lightweight ECC on the TE model due to its excellent arithmetic properties that enable fast scalar multiplication and effective protection against (certain kinds of) side-channel attacks. A TE curve over $\mathbb{F}_p$ is fully specified by the prime p and the two coefficients a and d of its defining equation, which is Eq. (3). We fix a to -1 so that implementers can unleash the full performance of the extended coordinates described in [21]. As a consequence, the curve-generation procedure boils down to finding a suitable prime field and second coefficient. For efficiency reasons, it is common practice to use primes of some "special" form that allow one to minimize the cost of the modular reduction operation and to choose the coefficient d to be small since it appears as operand of a multiplication in the addition formulae specified in e.g. [3] (for both projective and inverted coordinates) and [21, Sect. 3.1].

3.1 Selection of Prime Fields

An analysis of recent proposals for new curves shows that the underlying fields are based on three main classes of primes, namely generalized-Mersenne primes [28], pseudo-Mersenne primes, and primes for which Montgomery reduction can be optimized, i.e. "Montgomery-friendly" primes [13]. Pseudo-Mersenne primes seem to be particularly attractive since they were used by the majority of the

recent curve proposals, e.g. [1, 2, 4, 6, 9, 29]. Formally, a pseudo-Mersenne prime can be written as $p = 2^k - c$ where c is small in relation to 2^k. The reduction of a $2n$-bit integer x modulo p requires, in essence, just a multiplication of the upper half of x (i.e. the k most significant bits of x) by c, and then an addition of the product to the lower half of x (see e.g. [2, 9] for further details). Besides excellent arithmetic efficiency, these primes offer the virtue of minimizing the surface for side-channel attacks since the reduction can be easily implemented to have constant (i.e. operand-independent) execution time. Pseudo-Mersenne primes also allow for a parameterized implementation of the modular reduction operation so that one and the same reduction function can be used for primes of different length, which is not possible with the generalized-Mersenne primes of the NIST curves. This combination of desirable features led to our decision to use pseudo-Mersenne prime fields for the LiTE curves.

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 [28]. However, choosing the exponents in this way does not necessarily lead to peak performance. Namely, as shown in [2], it can be beneficial to use a prime 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 on bit of "headroom" simplifies the implementation of the field arithmetic when one aims for both high performance and resistance to side-channel attacks via constant (i.e. operand-independent) execution time [9]. Therefore, we decided to fix the values of k to 159, 191, 223, and 255.

The concluding step in the process of selecting a pseudo-Mersenne prime is to determine the constant c, which is typically chosen as the smallest integer so that $p = 2^k - c$ is prime [2]. An additional criterion often taken into account is the congruence class of p modulo 4, whereby the two most common choices are $p \equiv 3 \bmod 4$ and $p \equiv 5 \bmod 8$ (which implies $p \equiv 1 \bmod 4$)[3]. In the former case (i.e. $p \equiv 3 \bmod 4$), it is possible to find a TE curve with the property that the curve and its quadratic twist have both a minimal co-factor of 4 [23]. Unfortunately, -1 is always a non-square modulo such a prime and, consequently, the fast addition for TE curves from [21] is not guaranteed to be complete. On the other hand, if $p \equiv 5 \bmod 8$, then -1 is definitely a square in $\mathbb{F}_p$, but either the TE curve or its quadratic twist will have a co-factor of at least 8. However, we consider having a fast and complete addition law clearly more important than minimal co-factors, and thus we chose the values for c as the smallest integers that yielded primes congruent to 5 mod 8. The four primes we obtained in this way are $2^{159} - 91$, $2^{191} - 19$, $2^{223} - 235$, and $2^{255} - 19$. A TE curve over these

[3] These two choices allow for an efficient computation of square roots in $\mathbb{F}_p$ (which is needed for the decompression of compressed points [8]) through exponentiation.

primes with $a = -1$ can safely use Hişil et al.'s point-arithmetic formulae without compromising completeness [21]. To summarize, 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 mod 8, i.e. -1 is a square in $\mathbb{F}_p$. The second feature guarantees that a reduction modulo each of our primes can be efficiently implemented on 8, 16, and 32-bit microcontrollers since c always fits into a single register.

3.2 Requirements

State-of-the-art curve-generation procedures, in particular the ones described in [9, 17], put a strong emphasis on transparency and reproducibility to help the obtained curves find acceptance and trust in the cryptographic community. An important ingredient of such procedures is a set of well-explained and clearly-specified requirements to convince potential users of the curves that they were generated in a highly systematic and rigid fashion [6]. Our LiTE curves are, in essence, based on four major requirements, namely (i) security, (ii) arithmetic efficiency of operations in both the field and the group, (iii) consistency across security levels, and (iv) inter-operability with "legacy" cryptographic hardware and software that supports only the Weierstraß form.

Security requirements for elliptic curves mainly consist of criteria to ensure the hardness of the ECDLP, but may also take certain implementation aspects into account, e.g. to prevent non-obvious side-channel pitfalls [17]. In our case (i.e. TE curve over a large prime field), the ECDLP is generally assumed to be a hard problem if (i) the group of points on the curve E_T contains a large subgroup of prime order ℓ (or, equivalently, the co-factor $h = \#E_T(\mathbb{F}_p)/\ell$ of E_T is small) and (ii) the curve does not belong to some special class of "weak" curves for which discrete logarithms can be computed in less than the $0.886\sqrt{\ell}$ steps required by Pollard's rho method [6]. Like Montgomery curves, TE curves have a co-factor of $h \geq 4$ [3, 27]. Fortunately, most standards for ECC accept curves with small co-factors (e.g. $h \leq 8$ as in [23, Sect. A.1]), and some standards even tolerate not-so-small co-factors. For example, the NIST permits implementers of ECDSA to use an elliptic curve with a co-factor of up to 2^{10} if ℓ is between 160 and 223 bits long, while h can become as big as 2^{14} for ℓ lying in the range of 224 to 255 bits [28, Table 1]. When generating new TE curves, it is common practice to discard candidates that enable a multiplicative transfer or feature an efficient endomorphism because these properties would allow an attacker to "shortcut" the computation of discrete logarithms [1, 6]. Therefore, one has to check whether a curve candidate has a large embedding degree[4] e and a large Complex-Multiplication (CM) field discriminant D [17]. Some recent proposals for curve generation, e.g. [23, Sect. A], explicitly exclude also curves with trace $t = 0$ (i.e. supersingular curves) and $t = 1$ (i.e. anomalous curves), but this is

[4] For a TE curve E_T over $\mathbb{F}_p$ with $\#E(\mathbb{F}_p) = h\ell$, the embedding degree is defined as the smallest positive integer e such that ℓ divides $p^e - 1$.

redundant[5] when targeting TE curves over $\mathbb{F}_p$. An additional requirement often taken into account is *twist security*, which means that not only the TE curve E_T, but also its quadratic twist E'_T, meets the security criteria specified above (i.e. small co-factor and large embedding degree[6]) [6]. Twist security is a useful feature for x-coordinate-only ECDH key exchange based on Montgomery-form curves, such as X25519 [23], because it eliminates the need to check whether an incoming x-coordinate belongs to a point on the curve or on the twist [13]. All LiTE curves are twist-secure since ECDH is one of their main applications.

The second requirement we put on LiTE curves is to enable efficient implementations and facilitate state-of-the-art optimization techniques for both the field and group arithmetic. More precisely, we aim for curves that allow one to reach peak performance not only with the TE model, but also when using the birationally-equivalent Montgomery representation of the curve. A wide range of IoT devices (e.g. wireless sensor nodes) are equipped with small 8-bit microcontrollers whose limited computational capabilities may introduce long delays or high energy consumption when executing scalar multiplications. This makes a good case to take efficiency aspects—at both the field and group level—into account in the curve generation. Our approach of choosing a set of prime fields with good arithmetic properties, even on 8-bit microcontrollers, was explained in Subsect. 3.1. The addition law of TE curves can be fast *and* complete when $a = -1$ is a square in $\mathbb{F}_p$ and d a non-square; ideally, d is a small non-square so that a multiplication by d becomes less costly than an arbitrary multiplication in the field $\mathbb{F}_p$. On the other hand, when generating a Montgomery curve, it is usual practice to fix B to 1 and choose a small A congruent to 2 modulo 4 to ensure a multiplication by $(A+2)/4$ is fast [2, 27]. Unfortunately, a TE curve with "ideal" coefficients (i.e. $a = -1$ and d is small) is birationally-equivalent to a Montgomery curve with coefficients that are far from ideal. Namely, as can be seen from Eq. (4), the coefficient A of the corresponding Montgomery curve is $2(a+d)/(a-d) = 2(1-d)/(d+1)$, which is normally not small. We tackle the problem of non-ideal Montgomery coefficients through a small modification of the (projective) Montgomery doubling to minimize its execution time when $4/(A+2)$ is small instead of $(A+2)/4$ (see Sect. 4 for details).

Our third requirement is consistency across security levels, which means the curves should share certain properties about the structure of the elliptic-curve groups (e.g. co-factor, sign of trace) and the prime fields. Consistency enables a parameterized software implementation of the group arithmetic (i.e. addition and doubling of points) and the scalar multiplication so that one and the same arithmetic function can be used for curves of different order, e.g. ranging from

[5] A TE curve E_T over $\mathbb{F}_p$ can never be anomalous since a co-factor of $h \geq 4$ implies $\#E_T(\mathbb{F}_p) \neq p$ and also $\ell \neq p$. Supersingular curves are implicitly excluded because they do not have a large embedding degree. Concretely, a supersingular TE curve E_T over $\mathbb{F}_p$ has an order of $\#E_T(\mathbb{F}_p) = p+1$, which means its embedding degree is $e = 2$ since $p+1 = h\ell$ divides p^2-1 and, consequently, ℓ divides p^2-1.

[6] There is no need to check the CM field discriminant of E'_T since E_T and E'_T share the same endomorphism ring and, therefore, have the same discriminant.

159 to 255 bits in steps of 32 bits. Similar to efficiency, also consistency affects both the selection of fields (which we already discussed in Subsect. 3.1) and the generation of curves. The four LiTE curves we introduce in this paper have in common that the coefficient d is positive and small enough to fit into a single 32-bit word, which makes it straightforward to write a parameterized function for point addition. Besides point arithmetic also other operations, such as the generation of secret scalars for x-coordinate-only ECDH key exchange, can be implemented in a parameterized way, provided the set of curves meets certain conditions. If, for example, the bitlength of the underlying prime fields differs by a fixed amount (e.g. 32 bits) and each curve has a co-factor of 8 and a negative trace (like Curve25519), then a single parameterized function[7] suffices to generate scalars for all curves. Implementing or using a parameterized software library for the field/group arithmetic (and other operations) in settings where different levels of security need to be supported provides two major advantages compared to a separate implementation for each curve. First, it allows for substantial savings in (binary) code size, which is an important asset in the realm of the IoT. Second, the software development effort is significantly lower since each arithmetic function needs to be written and tested only once [13].

Finally, the fourth requirement is inter-operability with legacy elliptic-curve hardware and software that only supports the standard Weierstraß model given by Eq. (1). For efficiency, the coefficient a_4 of a Weierstraß curve is often fixed to -3, while the second coefficient a_6 is typically chosen to be a non-square in $\mathbb{F}_p$ in order to prevent the existence of points whose x-coordinate is 0. This is necessary because, as noted in [14, Sect. 3], some legacy ECC implementations encode $\mathcal{O}$ as $(0, 0)$, which would cause an ambiguity with one of the two points $(0, \pm\sqrt{a_6})$ when point compression is applied. Our LiTE curves are required to have a Weierstraß-form representation that is isomorphic to a Weierstraß curve with $a_4 = -3$ and a non-square a_6. We clearly prefer an isomorphism over an isogeny to keep the cost of converting points between different representations at a minimum. The need for point conversions between the Montgomery or TE form and the Weierstraß form arises when a state-of-the-art cryptosystem like X25519 [23] or EdDSA has to be implemented on top of some legacy hardware accelerator or software library for scalar multiplication. A well-known example of such legacy software is TinyECC [24], a lightweight ECC library for wireless sensors that supports solely Weierstraß curves with $a_4 = -3$. Another scenario requiring a conversion of points is discussed in [30] and concerns standardized cryptosystems like ECDSA, which use (affine) Weierstraß coordinates as "wire format." Instantiating ECDSA with a TE curve allows an implementer to take advantage of the high performance of the TE addition law for point arithmetic at the (small) expense of a conversion from TE to Weierstraß form during the signature generation, as well as a conversion in the opposite direction (i.e. from Weierstraß to TE form) when verifying a signature.

[7] This parameterized function can follow the approach of Curve25519, which means it first generates an array of (pseudo-)random bytes of the same byte-length as the underlying prime field and then "prunes" the first and last byte as in [23, p. 8].

3.3 Curve Generation

Since we have already chosen the prime fields of our LiTE curves and fixed the coefficient a to -1, the final step of the curve generation consists of finding the smallest coefficient d that satisfies all requirements discussed in Subsect. 3.2. In fact, these requirements can be condensed into five basic conditions, which are specified in the following definition of a LiTE curve.

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*

$$E_T: \ -x^2 + y^2 = 1 + dx^2y^2$$

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 $h = 8$ and a negative trace (i.e. $\#E_T(\mathbb{F}_p) > p$), while its quadratic twist E'_T has a co-factor of $h' = 4$ and a positive trace*
3. *E_T has an embedding degree of $e \geq (\ell - 1)/100$ and E'_T an embedding degree of $e' \geq (\ell' - 1)/100$*
4. *E_T has a CM field discriminant of $|D| > 2^{100}$*
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 is necessary to ensure that Hişil et al.'s "extended" addition formulae from [21] reach maximum performance *and* are complete, which is an efficiency requirement on our LiTE curves. In contrast, the second condition is related to security and to consistency. It guarantees, on the one hand, a basic prerequisite for the complexity of the ECDLP, namely the existence of a large cyclic subgroup of E_T (and of E'_T). Since we use prime fields with $p \equiv 5 \bmod 8$ (which implies $p \equiv 1 \bmod 4$), it is not possible that both the curve E_T and its quadratic twist E'_T have a minimal co-factor of 4 [23]; either h or h' has to be at least 8. We followed the approach of Curve25519 [2] and opted for $h = 8$ in order to prevent the accidental leakage of a bit of the secret scalar in protocols that involve a co-factor multiplication [23, Sect. A.1]. On the other hand, the second condition contributes to consistency because a negative trace means ℓ is always slightly larger than a power of 2, which enables a parameterized implementation of a function to generate secret scalars as discussed in the previous subsection. The third and fourth condition are linked to security; their purpose is to exclude curves with a transfer or a (fast) endomorphism. Both conditions are not new since they can be found in a similar form in [6, 14, 23]. Finally, the fifth condition guarantees inter-operability with legacy ECC hardware/software that supports only Weierstraß curves with $a_4 = -3$ and ensures the conversion of points through an isomorphism (the conversion of points between isogenous curves would be more complex [11]). An arbitrary Weierstraß curve over $\mathbb{F}_p$ is isomorphic to one governed by the equation $y^3 = x^3 - 3x + b$ when $-3/a_4$ has a fourth root in $\mathbb{F}_p$, which holds in our case for 25% of all values of a_4 [8].

We used the computer algebra system Magma V2.24 to compute the coefficient d according to Def. 1 for each of the four levels of security we consider in this paper. More concretely, we wrote a Magma script that essentially consists of a loop in which d gets incremented in each iteration until all five conditions are satisfied. This script output the coefficients 49445, 141087, 987514, as well as 4998299, which define the four LiTE curves:

$$\begin{aligned}
&\text{LiTE-P159}: -x^2 + y^2 = 1 + 49445x^2y^2 \bmod 2^{159} - 91\\
&\text{LiTE-P191}: -x^2 + y^2 = 1 + 141087x^2y^2 \bmod 2^{191} - 19\\
&\text{LiTE-P223}: -x^2 + y^2 = 1 + 987514x^2y^2 \bmod 2^{223} - 235\\
&\text{LiTE-P255}: -x^2 + y^2 = 1 + 4998299x^2y^2 \bmod 2^{255} - 19
\end{aligned}$$

Our smallest coefficient d (which is the one of the 159-bit curve) is only 16 bits long, whereas the largest coefficient has a length of 23 bits. The execution time of the script on a 2.4 GHz Xeon E5-2407 v2 processor ranged from 11 minutes (for the 159-bit curve LiTE-P159) to roughly 87 hours (LiTE-P255).

4 Birationally-Equivalent Montgomery Curves

For a LiTE curve (or any other TE curve with $a = -1$), the coefficients A and B of the birationally-equivalent Montgomery curve are

$$A = \frac{2(a+d)}{a-d} = \frac{2(1-d)}{1+d}, \tag{6}$$

$$B = \frac{4}{a-d} = -\frac{4}{1+d} = -\frac{2(1-d)+2(1+d)}{1+d} = -(A+2). \tag{7}$$

Consequently, the Montgomery representation of a LiTE curve is given by an equation of the form

$$-(A+2)y^2 = x^3 + Ax^2 + x. \tag{8}$$

The Montgomery-coefficient A obtained via Eq. (6) does not correspond to the common perception of efficiency since it is normally not small (and likely also not congruent to 2 modulo 4). In other words, when generating an efficient TE curve (i.e. a TE curve with $a = -1$ and small d), one can not expect that the birationally-equivalent Montgomery curve is also efficient. This problem exists in the opposite direction as well; for example, the TE curve that is birationally-equivalent to Curve25519 does not have a small coefficient d [30]. One way to deal with this issue is to generate, for each targeted security level, a TE curve with ideal coefficients and a distinct Montgomery curve with ideal coefficients (like in [1]). Unfortunately, this approach is not useful in the case of ephemeral ECDH key exchange, where one typically aims to reach maximum performance with the TE shape and the Montgomery shape of one and the same curve (see Sect. 2 and [25]). Bos et al. [9] approached this problem by exploiting isogenies between elliptic curves; concretely, they generated efficient Montgomery curves that are isogenous to efficient TE curves. In this way, they were able to obtain

elliptic curves that *simultaneously* feature a small coefficient A in their Montgomery representation and a small coefficient d in the isogenous TE form. Also Curve448 (an efficient Montgomery curve over a 448-bit prime [19]) is specified in [23] together with an isogenous Edwards curve with a small d. However, the conversion of points between isogenous curves is rather costly; for example, the 4-isogeny maps for Curve448 provided in [23, p. 6] are much more complicated than the birational maps for point conversion from [3].

The approach we take in this paper is to generate "ideal" coefficients for the TE model, but compensate the disadvantage of having a large coefficient A in the birationally-equivalent Montgomery representation by a slight modification of the point doubling for Montgomery curves. As pointed out before, when the coefficient a of a TE curve is set to -1, then the resulting coefficient A of the birationally-equivalent Montgomery curve is $2(1-d)/(d+1)$, which means the constant $(A+2)/4$ is not small. However, we found its reciprocal $4/(A+2)$ to be small when $a=-1$ and d is small. More concretely, due to Eq. (7) we have $4/(A+2) = d+1$, and this implies $4/(A+2)$ is small when d is small.

$$4X_nZ_n = (X_n+Z_n)^2 - (X_n-Z_n)^2 \tag{9}$$

$$X_{2n} = (X_n+Z_n)^2(X_n-Z_n)^2 \tag{10}$$

$$Z_{2n} = (4X_nZ_n)\left[(X_n-Z_n)^2 + ((A+2)/4)\,(4X_nZ_n)\right] \tag{11}$$

Montgomery provided in [27] the above formulae for the doubling of a point in projective $(X:Z)$ coordinates. The computation of $4X_nZ_n$ takes two squarings (2S) in $\mathbb{F}_p$ and, then, the computation of X_{2n} and Z_{2n} requires a multiplication (1M) each, which means the overall cost amounts to $2M+2S$, plus a multiplication by $(A+2)/4$. Fortunately, these formulae can be easily adapted for the Montgomery representations of our LiTE curves, whose A coefficients have the property that $4/(A+2)$ is small. Namely, by simply multiplying both X_{2n} and Z_{2n} by $4/(A+2)$, we obtain the doubling formulae below, which do not contain a multiplication by the constant $(A+2)/4$ anymore.

$$X_{2n} = (X_n+Z_n)^2(X_n-Z_n)^2\,(4/(A+2)) \tag{12}$$

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

This modification does not change the affine x-coordinate $x_{2n} = X_{2n}/Z_{2n}$, and so we can safely use these formulae for the Montgomery ladder. Similar to the original doubling method, $4X_nZ_n$ has to be computed first and, thereafter, the product of $(X_n-Z_n)^2$ and $4/(A+2)$ can be formed. This product serves then as operand for the computation of X_{2n} and Z_{2n}, respectively, which means the total cost amounts to $2M+2S$ and a multiplication by $4/(A+2)$. Apart from that, two additions and two subtractions in $\mathbb{F}_p$ have to be executed, exactly as with the original formulae [27]. In summary, performing a scalar multiplication on the Montgomery curves that are birationally-equivalent to our LiTE curves takes exactly the same number of $\mathbb{F}_p$-operations as when a Montgomery curve with a small coefficient A and $B=1$ is used, e.g. Curve25519.

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 (eds.), *Public Key Cryptography — PKC 2006*, vol. 3958 of *Lecture Notes in Computer Science*, pp. 207–228. Springer Verlag, 2006.
3. D. J. Bernstein, P. Birkner, M. Joye, T. Lange, and C. Peters. Twisted Edwards curves. In S. Vaudenay (ed.), *Progress in Cryptology — AFRICACRYPT 2008*, vol. 5023 of *Lecture Notes in Computer Science*, pp. 389–405. Springer Verlag, 2008.
4. D. J. Bernstein, C. Chuengsatiansup, and T. Lange. Curve41417: Karatsuba revisited. In L. Batina and M. Robshaw (eds.), *Cryptographic Hardware and Embedded Systems — CHES 2014*, vol. 8731 of *Lecture Notes in Computer Science*, pp. 316–334. Springer Verlag, 2014.
5. D. J. Bernstein and T. Lange. Faster addition and doubling on elliptic curves. In K. Kurosawa (ed.), *Advances in Cryptology — ASIACRYPT 2007*, vol. 4833 of *Lecture Notes in Computer Science*, pp. 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. I. F. Blake, G. Seroussi, and N. P. Smart. *Elliptic Curves in Cryptography*, vol. 265 of *London Mathematical Society Lecture Notes Series*. Cambridge University Press, 1999.
9. 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.
10. J. W. Bos, J. A. Halderman, N. Heninger, J. Moore, M. Naehrig, and E. Wustrow. Elliptic curve cryptography in practice. In N. Christin and R. Safavi-Naini (eds.), *Financial Cryptography and Data Security — FC 2014*, vol. 8437 of *Lecture Notes in Computer Science*, pp. 157–175. Springer Verlag, 2014.
11. E. Brier and M. Joye. Fast point multiplication on elliptic curves through isogenies. In M. P. Fossorier, T. Høholdt, and A. Poli (eds.), *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes — AAECC 2003*, vol. 2643 of *Lecture Notes in Computer Science*, pp. 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 (eds.), *Proceedings of the 1st ACM Workshop on Asia Public-Key Cryptography (AsiaPKC 2013)*, pp. 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. Ericsson. Ericsson Mobility Report November 2017. Available for download at `http://www.ericsson.com/assets/local/mobility-report/documents/2017/ericsson-mobility-report-november-2017.pdf`, 2017.
17. 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`.
18. M. Hamburg. Fast and compact elliptic-curve cryptography. Cryptology ePrint Archive, Report 2012/309, 2012. Available for download at `http://eprint.iacr.org`.
19. M. Hamburg. Ed448-Goldilocks, a new elliptic curve. Cryptology ePrint Archive, Report 2015/625, 2015. 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 (ed.), *Advances in Cryptology — ASIACRYPT 2008*, vol. 5350 of *Lecture Notes in Computer Science*, pp. 326–343. Springer Verlag, 2008.
22. N. I. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, Jan. 1987.
23. A. Langley, M. Hamburg, and S. Turner. Elliptic curves for security. Internet Engineering Task Force, Internet Research Task Force, RFC 7748, Jan. 2016.
24. 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)*, pp. 245–256. IEEE Computer Society Press, 2008.
25. Z. Liu, E. Wenger, and J. Großschädl. MoTE-ECC: Energy-scalable elliptic curve cryptography for wireless sensor networks. In I. Boureanu, P. Owezarski, and S. Vaudenay (eds.), *Applied Cryptography and Network Security — ACNS 2014*, vol. 8479 of *Lecture Notes in Computer Science*, pp. 361–379. Springer Verlag, 2014.
26. V. S. Miller. Use of elliptic curves in cryptography. In H. C. Williams (ed.), *Advances in Cryptology — CRYPTO '85*, vol. 218 of *Lecture Notes in Computer Science*, pp. 417–426. Springer Verlag, 1986.
27. P. L. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of Computation*, 48(177):243–264, Jan. 1987.
28. National Institute of Standards and Technology (NIST). Digital Signature Standard (DSS). FIPS Publication 186-4, available for download at `http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf`, July 2013.
29. M. Scott. Ed3363 (HighFive) – An alternative elliptic curve. Cryptology ePrint Archive, Report 2015/991, 2015. Available for download at `http://eprint.iacr.org`.
30. R. Struik. Alternative elliptic curve representations. Internet Engineering Task Force, Light-Weight Implementation Guidance (LWIG) Working Group, Internet draft draft-struik-lwip-curve-representations-02 (work in progress), July 2018.
31. L. Yan, Y. Zhang, L. T. Yang, and H. Ning. *The Internet of Things: From RFID to the Next-Generation Pervasive Networked Systems*. Auerbach Publications, 2008.