A Compact and Exception-Free Ladder for All Short Weierstrass Elliptic Curves

Ruggero Susella[1](✉) and Sofia Montrasio[2]

[1] STMicroelectronics, Agrate Brianza, Italy
ruggero.susella@st.com
[2] Università degli Studi di Milano, Milan, Italy
sofia.montrasio@gmail.com

Abstract. The field of elliptic curve cryptography has recently experienced a deployment of new models of elliptic curves, such as Montgomery or twisted Edwards. Computations on these curves have been proven to be exception-free and easy to make constant-time. Unfortunately many standards define elliptic curves in the short Weierstrass model, where the above properties are harder to achieve. This is especially true when scalar blinding, a simple but widely deployed side-channel attacks countermeasure, is adopted. In this paper we analyze previously undisclosed exceptional cases of popular scalar multiplication algorithms, highlighting the need for proofs of correctness. Then, with the final goal of providing a compact ECC hardware accelerator for embedded platforms, suitable to offload computations on all elliptic curve models, we present a constant-time adaptation of the Montgomery ladder, leveraging addition formulas by Izu and Takagi, that we *prove* return the correct result for *any* input point, *any* scalar value, on *all* elliptic curves in Weierstrass form defined over $\mathbb{F}_p$ with $p \neq 2, 3$.

Keywords: Elliptic curve cryptography · ECC · Scalar multiplication · Montgomery ladder · Exception-free · Constant-time · Scalar blinding

1 Introduction

The main operation of Elliptic Curve Cryptography (ECC) is the scalar multiplication, it is used for Diffie-Hellman key exchange and for digital signature schemes. Despite being, from a theoretical standpoint, a simple operation consisting in adding a point to itself many times, practical implementations might present undesired properties. In recent times several types of side-channel and fault attacks have been deployed to recover from real devices the secret used in the computation. Side-channel attacks (SCA) work by measuring side-channel leakage such as execution time, power consumption or electromagnetic emissions during a cryptographic computation and thanks to statistical analysis are able to recover the secret key. Fault attacks exploit malicious introduction of errors in the cryptographic computation, for example through laser fault injection, to recover the secret [6,14]. This has manifested the importance of being resilient

K. Lemke-Rust and M. Tunstall (Eds.): CARDIS 2016, LNCS 10146, pp. 156–173, 2017.
DOI: 10.1007/978-3-319-54669-8_10

to such attacks. Among the many proposed countermeasures for SCA, for example in [10,21], the simpler, less expensive, and thus more deployed, are: building constant-time implementation, using scalar blinding and/or randomization of the projective coordinates. To protect against fault attacks, common techniques require checking for invariant values during the computation, such as verifying that input and output points belong to the proper subgroup of the expected elliptic curve [6]. Among the various families of elliptic curve scalar multiplication algorithms, in this work we focused on single base algorithms scanning the scalar one bit at a time. The reason to stick to single base is that those using multiple bases are usually used for signature verification, where no secret is involved, thus not requiring protections against SCA and fault attacks. The reason to avoid the use of windowed ladders, those who scan the scalar in windows of more than one bit at a time, relates to the fact that they are more complex as they require a scalar recoding and precomputation phases which add, apart from greater memory requirements, more target points for attacks. It is in fact common practice for secure embedded implementations, such as those targeting smartcards, to use so called *regular ladders*. Most of these algorithms were deployed targeting prime order elliptic curves in short Weierstrass form, furthermore they usually restrict the set of allowed scalars to those smaller than the order of the curve, complicating the usage of the scalar blinding SCA countermeasure.

In recent times, the field of ECC, has seen a strong push for curves in other forms, such as Montgomery or Twisted Edwards. Among some beneficial properties introduced by these forms is the fact that, if properly chosen, Montgomery curves can rely on a very fast ladder proved to work exception-free [2]. Twisted Edwards curves on the other side can rely on a fast and exception-free addition law [4] which makes it easy to build exception-free and constant time scalar multiplication algorithms. One of the drawback of both forms is that they do not allow prime order curves. The issue of supporting these new forms, at least in secure embedded implementations, is that the designer needs at least one secure scalar multiplication algorithm for each form, thus adding complexity and area/code space requirements. An exception-free secure ECC hardware accelerator as generic as possible would be interesting for the implementor. The generic argument could simply mean to support short Weierstrass form, in this model in fact every Montgomery and (twisted) Edwards curve over prime fields can be represented, as well as any future model that might be proposed. Unfortunately, no scalar multiplication algorithm so far has been proved to work exception-free for short Weierstrass curves of any order and for any scalar.

Our Contributions. In this paper we first analyze commonly deployed scalar multiplication algorithms, like the Double-and-add always of Coron [10], the Montgomery ladder of Brier and Joye [8] and the fast Co-Z ladder of Goundar et al. [16], and show that in some cases, particularly when scalars are allowed to be of any size, they fail to compute the correct result. By finding that such commonly deployed algorithms fail to be exception-free, we want to underline the importance for researchers to look for proofs of correctness.

Then we propose a simple and regular ladder, suitable for secure compact hardware implementations, which works without exceptions for all short Weierstrass elliptic curves and any scalar. We also empirically verified this claim through systematic tests on several curves defined over small finite fields.

Specifically our algorithm is a composition of the Bernstein reformulation [3] of the Montgomery ladder [23] with x-only differential addition and doubling formulas by Izu and Takagi [19] followed by an y-recovery by Ebeid and Lambert [13]. We do not claim novelty for the ladder, even though it was never presented in this form, nor we compare its efficiency with the Montgomery ladder or with window-based ladders, but we prove, using a framework similar to Bernstein [2] that our ladder is exception-free.

We also show how our ladder can be used to compute exception-free scalar multiplications for Montgomery curves, thus providing a viable way to use a single implementation for all elliptic curves.

Related Work. A lot of work has been done in the search for a complete addition formula for elliptic curves which would allow to build exception-free scalar multiplication algorithms. A first attempt to a complete addition formula was made in 2002 by Brier and Joye [8, Sect. 3]. Unfortunately, Izu and Takagi found exceptional cases which they exploited in attacks, even on standardized curves [20]. Complete addition laws have been found for non-Weierstrass models, such as (twisted) Edwards [4], but this fact cannot be used to compute over curves of prime order, like the extensively deployed P-256. The most recent contribution is from 2016, when Renes et al. showed complete addition formulas for elliptic curves with odd order [26]. Their formulas are less efficient than incomplete homogeneous or Jacobian formulas and to compute, for example, on curve25519, they require special handling of the input point and the retrieval of the initial y-coordinate through an expensive square root computation.

To the best of our knowledge the first convincing attempt made to prove completeness of a constant-time scalar multiplication algorithm, instead of an addition formula, was done in 2006 by Bernstein [2] and although successful it is limited to Montgomery curves $y^2 = x^3 + Ax^2 + x$ with non-square $A^2 - 4$. Brier and Joye have presented an adaptation of the Montgomery ladder for short Weierstrass curves [8, Sect. 4], this has unknown exceptions that we will explore in Sect. 3. Another example of ladder which comes with a proof is from Bos et al. who in 2014 presented a complete scalar multiplication algorithm for *prime order* short Weierstrass curves [7, Algorithm 1]. Their method also restricts scalars to those smaller than the order of the curve, thus not allowing scalar blinding.

2 Preliminaries

In this paper we will consider only finite fields $\mathbb{K}$ with $char(\mathbb{K}) \neq 2, 3$. An elliptic curve over $\mathbb{K}$ is a non-singular curve defined by an equation E of the form:

$$E: \; y^2 = x^3 + ax + b \text{ with } a, b \in \mathbb{K} \tag{1}$$

Eq. 1, also called short Weierstrass form, is used to represent the most common curves used in ECC; other curves, using a different definition of elliptic curve, e.g. (twisted) Edwards curves have been shown to be birationally equivalent to short Weierstrass. This relation allows to create an invertible map to compute the scalar multiplication in their short Weierstrass equivalent.

$E(\mathbb{K})$, the set of points $(x, y) \in \mathbb{K} \times \mathbb{K}$ satisfying Eq. 1 along with a point at infinity represented by ∞, has an additive group structure where the neutral element is ∞ and the inverse of $P = (x, y)$ is $-P = (x, -y)$.
If $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ with $P_2 \neq -P_1$, the sum $P_1 + P_2 = (x_3, y_3)$ is:

$$x_3 = \lambda^2 - x_1 - x_2 \text{ and } y_3 = \lambda(x_1 - x_3) - y_1 \tag{2}$$

with $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$ if $P_2 \neq P_1$ and $\lambda = \frac{3x_1^2 + a}{2y_1}$ if $P_2 = P_1$.

In the scalar multiplication a point $Q = (x_Q, y_Q)$ is commonly represented in homogeneous projective coordinates as $(x_Q, y_Q, 1)$. In this way the computation of $n(x_Q, y_Q, 1) = (x, y, z)$ has the advantage of not requiring expensive field inversions, and only at the very end, with just a single field inversion, the affine result $nQ = (x/z, y/z)$ is recovered. The relation between homogeneous and affine coordinates is in fact: $x_Q = x/z$ and $y_Q = y/z$. In the situations where the notation is ambiguous the context will indicate if we are using affine or projective coordinates.

The order of a point divides the cardinality, $\#E(\mathbb{K})$, of the group $E(\mathbb{K})$ which is here the product of a large prime l and a very small *cofactor* h.

Among the countermeasures against Differential Power Analysis (DPA) we highlight here *Scalar blinding* and *Coordinates randomization*. The first one is the substitution of the scalar n with $n' = n + rl$, where l is the order of the input point and $r \in \mathbb{N}$ is a random integer of the form $r = 2^c \cdot o$, where o is odd. The second method uses the equivalence in homogeneous projective coordinates between (x, y, z) and (rx, ry, rz) and before each new scalar multiplication randomizes the coordinates of the input point using the random value $r \in \mathbb{K}^*$.

3 Review of Common Scalar Multiplication Algorithms

In this section we look at some well known and commonly deployed scalar multiplication algorithms: Coron's Double-and-add always, Brier and Joye's Montgomery ladder and Goundar et al. Co-Z Montgomery ladder. We will show that these algorithms are not exception-free. Note that to simplify the description, in this section, we define a point to be ∞ if its z-coordinate is 0 without caring about the other coordinates.

3.1 Double-and-Add Always

Coron in 1999 presented his famous Double-and-add always ladder [10], resistant against simple power analysis, shown here as Algorithm 1. Unfortunately addition laws for short Weierstrass curves have exceptional cases. The main issue of incomplete addition laws is that unexpectedly $P + \infty = \infty$, thus if the accumulating variable of the ladder ($R[0]$) becomes equal to ∞, the final result will be ∞.

Line 3 in fact will produce $R[0] = 2R[0] = \infty$ if $\mathrm{ord}(R[0]) = 2$ (or $R[0] = \infty$). Line 4 will produce $R[1] = R[0] + Q = \infty$ if $R[0] = \pm Q$ (or $R[0] = \infty$). We verified that these exceptions are present with all jacobian and homogeneous mixed additions formulas of [5]. These exceptions explain why Algorithm 1 initializes $R[0] = Q$ in line 1, instead of using ∞, thus a first exception of this ladder happens when $n = 0$ and the ladder incorrectly returns Q. This initialization also makes the execution time dependent on the number of leading zeros in the binary representation of n. It is common practice to solve this last issue by adding l or $2l$ to n, as suggested in [7,9], but we believe that this adds unneeded complexity to the algorithm.

We now consider the exceptions for points of small order. If Q has order 2 the ladder will fail for all odd $n > 1$, because line 3 will produce an $R[0] = \infty$, which is correct by itself, but then line 4 will subsequently fail, causing the error in the result. If, instead, Q has order 3 then line 4 will make $R[1] = \infty$, if in line 5 $R[1]$ is assigned to $R[0]$ the result will be ∞. This is an incorrect result for all n not multiple of 3 with Hamming weight greater than 1.

Finally, there are exceptions even for points of large prime order, although only when $n > l$ as in the case of applying scalar blinding (replacing n with $n' = n + rl$ with a random r writable as $r = 2^c o$), which was presented by Coron in the same paper, without any mention of exceptional cases. Algorithm 1 is exceptional when $n \in \{1, 2, \ldots, 2^c - 1\}$ for $c > 0$ because the most significant part of n' becomes $o \cdot l$ so $R[0]$ becomes ∞. It is also exceptional for $n \in \{2 \cdot 2^c, 2 \cdot 2^c + 1, \ldots, 3 \cdot 2^c - 1\}$ for all c because line 3 will eventually compute $R[0] = 2\left(\frac{o \cdot l+1}{2}Q\right) = Q$ and line 4 will then add Q to it, causing the failure of the incomplete addition.

The only solution to make Algorithm 1 exception-free is to modify the addition to correctly handle exceptions. Unfortunately, since no complete addition formula is known to work for all short Weierstrass curves, possible solutions are to call the doubling when input points are equal or to use *masking* techniques like [7, Algorithms 18, 19]. The first method will break the constant-time property while the second one will introduce dummy operations which are vulnerable to some attacks [28]. It is therefore more secure to avoid both.

Algorithm 1. Double-and-add always ladder

```
Input: $Q \in E(\mathbb{F}_p)$, $n = (n_{t-1}, .., n_0)_2$
Output: $nQ$
1: $R[0] = Q$
2: for $i = t - 2$ down to 0 do
3:     $R[0] \leftarrow 2R[0]$
4:     $R[1] \leftarrow R[0] + Q$
5:     $R[0] \leftarrow R[n_i]$
6: return $R[0]$
```

Algorithm 2. Brier-Joye's ladder

```
Input: $Q \in E(\mathbb{F}_p)$ and $n = (n_{t-1}, ..., n_0)_2$
Output: $X(nQ)$
1: $R[0] = Q; R[1] = 2Q$
2: for $i = t - 2$ down to 0 do
3:     if $(n_i = 0)$ then
4:         $x(R[1]) \leftarrow x(R[0] + R[1])$
5:         $x(R[0]) \leftarrow x(2R[0])$
6:     else $x(R[0]) \leftarrow x(R[0] + R[1])$
7:         $x(R[1]) \leftarrow x(2R[1])$
8: return $x(R[0])$
```

We want to underline that, when scalars are allowed to be larger than l, all Double-and-add algorithms with incomplete addition law suffer from the above issue, as some scalar values will force them to compute the addition (not through doubling) of $Q + Q$.

3.2 Brier and Joye Montgomery Ladder

Brier and Joye in 2002 presented a variant of the Montgomery ladder for short Weierstrass elliptic curve [8], shown here in Algorithm 2. The ladder, like the Double-and-add Always, would leak, through timing, the number of leading zeros of the scalar. But, unlike in the previous case, this ladder allows to be initialized with $R[0] = \infty; R[1] = Q$, so that it can iterate for a constant number of times, thus solving the issue without adding unneeded complexity.

Notice that the authors provide proofs of the validity of their addition law, and that furthermore the ladder is extensively believed to work exception-free, see for example [26, Sect. 1]. Unfortunately, their proof fails to consider the exceptional case when x_Q, the x-coordinate of Q is 0. This can be clearly seen from their formula for the x-coordinate differential addition:

$$x(R[0] + R[1]) = \frac{-4b(x_0 + x_1) + (x_0 x_1 - a)^2}{x_Q(x_0 - x_1)^2} \tag{3}$$

If $x_Q = 0$ the output of the differential addition will always have $z = 0$, causing the ladder to return ∞, for every scalar. In short Weierstrass curves, if b is a non-zero square in the field there are two points with $x = 0$; this happens even for standardized prime order elliptic curves such as NIST curves P-256, P-384 and P-521. The Brier-Joye ladder is therefore exceptional.

3.3 Co-Z Montgomery Ladder

In 2011 Goundar et al. presented several ladders using the Co-Z arithmetic originally proposed by Meloni. We focus our attention on algorithm 15 of [16, Sect. 5.2.1] named “Montgomery ladder with (X,Y)-only co-Z addition formulæ” because it is the asymptotically fastest ladder for short Weierstrass curves known today. As in the case of the Double-and-add Always, this ladder leaks the number of leading zeroes of the scalar, and must be modified in the same way to achieve constant-time execution. Moreover, like in the case of the Brier-Joye ladder, the Co-Z ladder is exceptional if $x_Q = 0$ because the recovered z is defined as $x_Q y(...)$ and in this case becomes automatically 0, giving as output ∞.

If we consider exceptions on scalar values, then in the range of $[0, l]$ the problems are $0, 1$ and $l - 1$. In fact, if $n = 0$ the ladder returns $2Q$, if $n = 1$ it returns $3Q$ and if $n = l - 1$ it returns ∞. The first two cases depend on the definition inside the function $DBLU'$ of the two initial points Q and $2Q$, while in the last one the problem is that $R[0] = (l - 1)Q$ has the same z as $R[1] = lQ = \infty$. The usage of scalar blinding creates additional exceptional cases. The Co-Z ladder in fact returns ∞ when $n \in \{1, 2, \ldots, 2^c - 1\}$ for all $c > 0$ and $n \in \{l - 1, \ldots, l - 2^c\}$ for all c. This ladder has therefore several exceptions.

4 Exception-Free Ladder

In this section we present Algorithm 3, our exception-free (x, y)-ladder, which is the core of this paper. This is divided in two parts: the Montgomery ladder and the y-recovery. The Montgomery ladder uses the additive x-only formulas derived by Izu and Takagi [19] which eliminate the exception $x_Q = 0$ of the formulas in Sect. 3.2 at basically the same cost. Then a procedure based on a formula of Ebeid and Lambert [13] is used to recover the y-coordinate.
Notice five aspects of Algorithm 3:

1. It takes in input and returns (x, y) *and* a flag f. The flag allows to distinguish a valid affine point from ∞. We note that Montgomery ladder on Montgomery curves encodes ∞ into (0,0), which corresponds to a 2-torsion point and avoids to distinguish the two. But this cannot be done on generic curves and we need a flag to make the distinction with valid affine points.
2. It has no conditional branches and the loop is iterated a fixed number of times, independent of the scalar value.
3. In the presented form it has conditional assignments based on the scalar bit. However it is trivial to modify it to use the *cswap* function as done in [3] to make it constant-time even for software implementations.
4. The algorithm, written in this way, takes in input also a random value $r \in \mathbb{F}_p^*$ for coordinates randomization. If the user is not interested in this countermeasure he can spare the two field multiplications in the initialization.
5. Inside the main loop no dummy operation occurs. This avoids some type of safe-error attacks such those in [28]. Dummy operations occur after the main loop but these depend only on the result of the main loop, which is considered known to the attacker, thus no secret can be recovered through safe-error attacks on these dummy operations.

Using the same framework of [2, Appendix B] we state and prove that the Izu and Takagi formulas are exceptional only if both x and z input coordinates are 0. Then we will show in Theorem 3 that our ladder never satisfies such condition and thus provides the correct result for all input points and scalar values. Note that in this paper it x/it z means the quotient of x and z in $\mathbb{F}_p$ if $z \neq 0$; it means ∞ if $x \neq 0$ and $z = 0$; it is undefined if $x = z = 0$.

Theorem 1 (Doubling). *Let p be a prime number with $p \geq 5$. Define E as the non-singular short Weierstrass elliptic curve $y^2 = x^3 + ax + b$ over $\mathbb{F}_p$. Define $X : E(\mathbb{F}_{p^2}) \to \{\infty\} \cup \mathbb{F}_{p^2}$ as follows: $X(\infty) := \infty, X(x, y) := x$. Defining:*

$$x_2 := (x^2 - az^2)^2 - 8bxz^3 \qquad \text{and} \qquad z_2 := 4z(x^3 + axz^2 + bz^3)$$

for fixed $x, z \in \mathbb{F}_p$ with $(x, z) \neq (0, 0)$. Then $X(2Q) = \frac{x_2}{z_2}$ and $(x_2, z_2) \neq (0, 0) \in \mathbb{F}_p \times \mathbb{F}_p$ for all $Q \in E(\mathbb{F}_{p^2})$ s.t. $X(Q) = \frac{x}{z}$.

Algorithm 3. Proposed Montgomery Ladder

Input: $p, a, b, n, x_Q, y_Q, f, r, m$. Where: $p \geq 5$ is a prime of bit-size s.
$a, b \in \mathbb{F}_p$ define $E(\mathbb{F}_p)$. The scalar is $n = (n_{t-1}, ..., n_0)_2 = \sum_{i=0}^{t-1} 2^i n_i \in \mathbb{N}$.
$Q \in E(\mathbb{F}_p)$ is the input base point of the scalar multiplication.
f is a flag $\in \{0,1\}$, s.t. if $f = 1$ then $Q = (x_Q, y_Q)$ with $x_Q, y_Q \in \mathbb{F}_p$ else $Q = \infty$.
r is an optional randomization value $\in \mathbb{F}_p^*$. The algorithm processes $m \in \mathbb{N}$ bits of n, independently of the value of n, with the only requirement of m being $\geq t$.
Output: x, y, f s.t. if $f = 0$ then $nQ = \infty$ else $nQ = (x, y)$

```
                                        ▷ All operations, except lines 11, 16 are performed modulo $p$
 1: function PROPOSED-LADDER($p, a, b, n, x_Q, y_Q, f, r = 1, m = \lceil \log_2(\#E(\mathbb{F}_p)) \rceil$)
 2:     $x_0, z_0, x_1, z_1 \leftarrow 1, 0, x_Q \cdot r, f \cdot r$
 3:     for $i = m-1$ down to 0 do                    ▷ Note: $m$ can be arbitrary big
 4:         $d \leftarrow n_i$                          ▷ $n_i = 0$ for $i \in [m-1, t-1)$
 5:         $x_{\neg d},\ z_{\neg d} \leftarrow 2(x_d z_{\neg d} + x_{\neg d} z_d)(x_d x_{\neg d} + a z_d z_{\neg d}) + 4b z_d^2 z_{\neg d}^2$
                    $- x_Q(x_d z_{\neg d} + x_{\neg d} z_d)^2,\ (x_d z_{\neg d} - x_{\neg d} z_d)^2$
 6:         $x_d, z_d \leftarrow (x_d^2 - a z_d^2)^2 - 8b x_d z_d^3,\ \ 4(x_d z_d(x_d^2 + a z_d^2) + b z_d^4)$
 7:     $y_0 \leftarrow -y_Q z_0$                        ▷ Notice that $y_0/z_0 = -y_Q$
 8:     $x'_0 \leftarrow 2 y_Q x_0 z_1 z_0$             ▷ Use formulas from Lemma 1 for $y$-recovery
 9:     $y'_0 \leftarrow 2b z_1 z_0^2 + z_1(a z_0 + x_Q x_0)(x_Q z_0 + x_0) - x_1(x_Q z_0 - x_0)^2$
10:     $z'_0 \leftarrow 2 y_Q z_1 z_0^2$
11:     $f \leftarrow \mathbb{Z}(2^s - 1 + z_1) \gg s$              ▷ $f = 1 \iff z_1 \neq 0$ else $f = 0$
12:     $x = x_0 - f(x_0 - x'_0)$                     ▷ $x = x'_0 \iff z_1 \neq 0$ else $x = x_0$
13:     $y = y_0 - f(y_0 - y'_0)$                     ▷ $y = y'_0 \iff z_1 \neq 0$ else $y = y_0$
14:     $z = z_0 - f(z_0 - z'_0)$                     ▷ $z = z'_0 \iff z_1 \neq 0$ else $z = z_0$
15:     $x, y \leftarrow x z^{p-2},\ y z^{p-2}$                          ▷ get affine values
16:     $f \leftarrow \mathbb{Z}(2^s - 1 + z) \gg s$       ▷ where $\mathbb{Z}(x)$ means that $x$ is evaluated in $\mathbb{Z}$
17:     return $x, y, f$          ▷ $f = 0 \iff nQ = \infty$ else ($f = 1$ and $nQ = (x, y)$)
```

Proof. As in [2] we analyze all possible cases:

Case 1. $z = 0$. This implies $x \neq 0$ because $(x, z) \neq (0, 0)$. We notice that $x_2 = x^4 \neq 0$ and $z_2 = 0$. Also $X(Q) = \frac{x}{0} = \infty$ so $Q = \infty$ and $2Q = \infty$. Then for definition of X, $X(2Q) = \infty = \frac{x_2}{0} = \frac{x_2}{z_2}$.

Case 2. $z \neq 0$. In this case $Q = (\frac{x}{z}, y)$ for some $y \in \mathbb{F}_{p^2}$ such that $y^2 = (\frac{x}{z})^3 + a(\frac{x}{z}) + b$. Multiplying this equation by $4z^4$, which is nonzero, we obtain $4y^2 z^4 = 4(x^3 z + a x z^3 + b z^4)$ which corresponds to z_2. Now we need to make a distinction between the cases $y^2 = 0$ and $y^2 \neq 0$.

Case 2.1. $z \neq 0$ and $y^2 \neq 0$. Knowing that $z_2 = 4y^2 z^4$ we need to verify if $x_2 = 4y^2 z^4 \cdot X(2Q)$ holds. Replacing $X(2Q)$ with Eq. 2 we get:

$$4y^2 z^4 \cdot X(2Q) = 4y^2 z^4 \cdot \left[\frac{(3(\frac{x}{z})^2 + a)^2 - 8(\frac{x}{z})y^2}{4y^2}\right] = z^4 \left[(3(\frac{x}{z})^2 + a)^2 - 8(\frac{x}{z})y^2\right]$$

Solving and substituting y^2 with $y^2 = (\frac{x}{z})^3 + a(\frac{x}{z}) + b$ we obtain: $(x^2 - a z^2)^2 - 8b x z^3$ which corresponds to our definition of x_2. It is interesting to note that $x = 0$ is not a special case.

Case 2.2. $z \neq 0$ and $y^2 = 0 \implies x^3 + axz^2 + bz^3 = 0 \implies z_2 = 0$. In this case $Q = (\frac{x}{z}, 0)$ has order two, i.e. $2Q = \infty$. If we show that $x_2 \neq 0$ then $X(2Q) = \infty = \frac{x_2}{0} = \frac{x_2}{z_2}$ and we are done. To prove that $x_2 \neq 0$, we first note that:

$$x^3 + axz^2 + bz^3 = 0 \implies bz^3 = -x^3 - axz^2$$

Replacing bz^3 in $x_2 := (x^2 - az^2)^2 - 8bxz^3$ we get $x_2 = (3x^2 + az^2)^2$ which is 0 iff $az^2 = -3x^2$, substituting az^2 with $-3x^2$ in $x^3 + axz^2 + bz^3 = 0$ we obtain: $\frac{x}{z} = \sqrt[3]{\frac{b}{2}}$. Hence by imposing $x_2 = 0$ and $y^2 = 0$, we have a point of the form $Q = (\sqrt[3]{b/2}, 0)$. Forcing Q to satisfy Eq. 1 we obtain the following relation between a and b: $\frac{b}{2} + a\sqrt[3]{\frac{b}{2}} + b = 0 \implies a = -\frac{3}{2}b\sqrt[3]{\frac{2}{b}} \implies 4a^3 = 4\left(-\frac{27}{8}b^3\frac{2}{b}\right) = -27b^2$. But this contradicts the non-singularity condition $\Delta = -16(4a^3 + 27b^2)$ of the curve, therefore there is no such point Q, so $x_2 \neq 0$.

Note that $x = 0$ presents no issues in this case too. In fact we would obtain $x_2 = a^2z^4$ which is null iff $a = 0$, but this would imply $y^2 = b = 0$, which would again violate the non-singularity condition. □

Theorem 2 (Differential Addition). *Let p be a prime number with $p \geq 5$. Define E as the non-singular short Weierstrass elliptic curve $y^2 = x^3 + ax + b$ over $\mathbb{F}_p$. Define $X : E(\mathbb{F}_{p^2}) \to \{\infty\} \cup \mathbb{F}_{p^2}$ as follows: $X(\infty) := \infty, X(x, y) := x$. Fix $x, z, x', z', x_1, z_1 \in \mathbb{F}_p$ with $(x, z) \neq (0, 0)$,$(x', z') \neq (0, 0)$ and $z_1 \neq 0$. Define:*

$$x_3 := \left[2\left(xz' + x'z\right)\left(xx' + azz'\right) + 4b\left(zz'\right)^2\right] z_1 - x_1\left(xz' - x'z\right)^2$$
$$z_3 := z_1\left(xz' - x'z\right)^2$$

Then $X\left(Q + Q'\right) = \frac{x_3}{z_3}$ for all $Q, Q' \in E(\mathbb{F}_{p^2})$ such that $X\left(Q\right) = \frac{x}{z}$, $X\left(Q'\right) = \frac{x'}{z'}$, $X\left(Q - Q'\right) = \frac{x_1}{z_1}$ and $(x_3, z_3) \neq (0, 0) \in \mathbb{F}_p \times \mathbb{F}_p$.

Proof. Let's again analyze all possible cases:

Case 1. $Q = Q'$. Then $X\left(Q - Q'\right) = X\left(\infty\right) = \infty$, so $z_1 = 0$, contradiction.

Case 2. $Q = \infty$. Then $z = 0$ and $x \neq 0$; also $X\left(Q - Q'\right) = X\left(-Q'\right) = X\left(Q'\right) = \frac{x'}{z'}$, so $\frac{x_1}{z_1} = \frac{x'}{z'}$ and consequently $z' \neq 0$. Under the assumption $z = 0$, $x_3 = 2\left(xz'\right)\left(xx'\right)z_1 - x_1\left(xz'\right)^2$ and $z_3 = z_1\left(xz'\right)^2$, then $\frac{x_3}{z_3} = 2\left(\frac{x'}{z'}\right) - \left(\frac{x_1}{z_1}\right) = \frac{x'}{z'} = X\left(Q'\right) = X\left(Q + Q'\right)$ as wanted.

Case 3. $Q' = \infty$. Then $z' = 0$ and $x' \neq 0$; also $X\left(Q - Q'\right) = X(Q) = \frac{x}{z}$, so $\frac{x_1}{z_1} = \frac{x}{z}$ and consequently $z \neq 0$. Given that $z' = 0$ we have $x_3 = 2\left(x'z\right)\left(xx'\right)z_1 - x_1\left(x'z\right)^2$ and $z_3 = z_1\left(x'z\right)^2$. Then, similarly to the previous case: $\frac{x_3}{z_3} = 2\left(\frac{x}{z}\right) - \left(\frac{x_1}{z_1}\right) = \frac{x}{z} = X\left(Q\right) = X\left(Q + Q'\right)$.

Case 4. $Q = -Q'$. Here $X(Q) = X(-Q') = X(Q')$, so $\frac{x}{z} = \frac{x'}{z'}$ and $xz' = x'z$. Assuming now both $z, z' \neq 0$, then $z_3 = 0$. Suppose that $x_3 = 0$, knowing that

$z_1 \neq 0$ it means $4\left(xz'\right)\left(xx' + azz'\right) + 4b\left(zz'\right)^2 = 0$, which dividing by $4\left(zz'\right)^2$ becomes $\left(\frac{x}{z}\right)^3 + \left(\frac{x}{z}\right)a + b = 0$. This implies $Q = \left(\frac{x}{z}, 0\right) = Q'$ and a contradiction as in Case 1. Thus $x_3 \neq 0$ and $\frac{x_3}{z_3} = \infty = X\left(\infty\right) = X\left(Q + Q'\right)$.

Case 5. $Q \neq \infty$, $Q' \neq \infty$, $Q \neq Q'$ and $Q \neq -Q'$. Then $\frac{x}{z} \neq \frac{x'}{z'}$ so $z_3 \neq 0$. Also $Q = \left(\frac{x}{z}, y\right)$ and $Q' = \left(\frac{x'}{z'}, y'\right)$ for some $y, y' \in \mathbb{F}_{p^2}$ such that $y^2 = (\frac{x}{z})^3 + a(\frac{x}{z}) + b$ and $(y')^2 = (\frac{x'}{z'})^3 + a(\frac{x'}{z'}) + b$. We now define $\alpha := \frac{x'}{z'} - \frac{x}{z} \neq 0$ and $\beta := \frac{x'}{z'} + \frac{x}{z}$. Using Eq. 2 we obtain: $X\left(Q + Q'\right) = \frac{\left(y' - y\right)^2}{\alpha^2} - \beta$ and $X\left(Q - Q'\right) = \frac{\left(-y' - y\right)^2}{\alpha^2} - \beta$. So $X\left(Q + Q'\right) + X\left(Q - Q'\right) = 2\left(\frac{(y')^2 + y^2}{\alpha^2} - \beta\right)$. Replacing $(y')^2$ and y^2 we get: $X\left(Q + Q'\right) + X\left(Q - Q'\right) = \frac{2\left(xz' + x'z\right)\left(xx' + azz'\right) + 4b\left(zz'\right)^2}{(xz' - x'z)^2}$.

The statement follows remembering that $X\left(Q - Q'\right) = \frac{x_1}{z_1}$ □

Before proving the correctness of Algorithm 3, we present the y-recovery procedure used in lines 7–14 of Algorithm 3 which also deals, in constant time, with the exceptional case of z_1 being null at the end of the loop. Note that, although the hypotheses of Lemma 1 exclude points of order two, at the end of the proof of Theorem 3 we will show they will not cause exceptions.

Lemma 1 (y-recovery). *Define X as in previous theorems. Let $Q, R\left[0\right], R\left[1\right] \in E(F_p) \setminus \infty$ s.t. $R\left[0\right] = nQ$, $R\left[1\right] = R[0] + Q$ with $Q = (x_Q, y_Q)$, $X\left(R[0]\right) = \frac{x_0}{z_0}$ and $X\left(R[1]\right) = \frac{x_1}{z_1}$. Defining:*

$$\begin{aligned}
x_0' &:= 2y_Q x_0 z_1 z_0 \\
y_0' &:= 2bz_1 z_0^2 + z_1(az_0 + x_Q x_0)(x_Q z_0 + x_0) - x_1(x_Q z_0 - x_0)^2 \\
z_0' &:= 2y_Q z_1 z_0^2
\end{aligned}$$

then $z_0' \neq 0$ and $R[0] = \left(\frac{x_0'}{z_0'}, \frac{y_0'}{z_0'}\right)$.

See Appendix B for the proof of Lemma 1.

Theorem 3 (Algorithm 3 is Exception-Free). *Define $XY_1 : E(\mathbb{F}_p) \to \mathbb{F}_p \times \mathbb{F}_p \times \{0, 1\}$ as $XY_1(\infty) := (0, 0, 0)$, $XY_1(x, y) := (x, y, 1)$. Let $Q \in E(\mathbb{F}_p)$ be the input point of the scalar multiplication, let $f \in \{0, 1\}$ be a flag s.t. if $f = 0$ then $Q = \infty$ otherwise $Q = (x_Q, y_Q)$, with $x_Q, y_Q \in \mathbb{F}_p$, then Algorithm 3 correctly returns $XY_1(nQ)$ for every $n \in \mathbb{N}$.*

Proof. It is convenient in the proof to keep the values of intermediate points calculated by the ladder, even if only the x and z coordinates are actually computed. Referring to Algorithm 3 we call $R\left[0\right]_0 = \infty$ and $R\left[1\right]_0 = Q$ the two initial points. Line 5 can be seen as $R\left[\neg d\right] = \left(R\left[d\right] + R\left[\neg d\right]\right)$ with $XZ(R\left[\neg d\right]) := (x_{\neg d}, z_{\neg d})$ where $d \in \{0, 1\}$ and line 6 as $R\left[d\right] = 2R\left[d\right]$ with $XZ(R\left[d\right]) := (x_d, z_d)$.

The loop on the sequence of bits composing the scalar n produces a sequence of couples: $\left(R\left[0\right], R\left[1\right]\right)_1, \left(R\left[0\right], R\left[1\right]\right)_2, ..., \left(R\left[0\right], R\left[1\right]\right)_m$.

We now prove, by induction on j, where $j = m-i$ represents the jth iteration of the loop, that with $f = 1$ the following conditions hold:

$$(x_{0,j}, z_{0,j}) = XZ(R\,[0]_j) = XZ(\textsc{msb}_j(n)Q) \text{ with } (x_{0,j}, z_{0,j}) \neq (0,0)$$
$$\text{and } R\,[1]_j = R\,[0]_j + Q \ \forall j \in 1, 2, ..., m$$

where $\textsc{msb}_j(n)$ indicates the j most significant bits of the scalar n, with $j \leq$ bit-size(n). Let's start by the first iteration, $j = 1$:

Case 1. $d = 0$. We first notice that $R\,[0]_1 = 2(R\,[0]_0) = \infty = 0Q = \textsc{msb}_j(n)Q$ and $(x_{0,1}, z_{0,1}) = XZ(R\,[0]_1) = XZ(2R\,[0]_0) = (1,0)$ due to the definition of x_0, z_0; so $(x_{0,1}, z_{0,1}) \neq (0,0)$. Since $R\,[0]_1 = \infty$ it is trivial to see that $R\,[1]_1 = R\,[0]_1 + Q$ as in the second condition.

Case 2. $d = 1$. Here $R\,[0]_1 = R\,[0]_0 + R\,[1]_0 = Q = \textsc{msb}_j(n)Q$ and by definition $(x_{0,1}, z_{0,1}) = XZ(R\,[0]_1) = (x_Q \cdot r, r) \neq (0,0)$. Then $R\,[1]_1 = 2R\,[1]_0 = 2Q$ so $R\,[1]_1 - R\,[0]_1 = 2Q - Q = Q$ as expected.

Now we suppose this holds for iteration j and prove it for $j+1$. Again we have two cases, depending on the value of bit d in iteration j:

Case 1. $d = 0$. In this case $(x_{0,j+1}, z_{0,j+1}) = XZ(R\,[0]_{j+1}) = XZ(2R\,[0]_j) = XZ(2\textsc{msb}_j(n)Q) = XZ(\textsc{msb}_{j+1}(n)Q) \neq (0,0)$ by Theorem 1.

Using the second condition we also have $R\,[1]_{j+1} = R\,[1]_j + R\,[0]_j = 2R\,[0]_j + Q = R\,[0]_{j+1} + Q$.

Case 2. $d = 1$. Here $(x_{0,j+1}, z_{0,j+1}) = XZ(R\,[0]_{j+1}) = XZ(R\,[0]_j + R\,[1]_j) = XZ(2R\,[0]_j + Q) = XZ((2\textsc{msb}_j(n) + 1)Q) = X(\textsc{msb}_{j+1}(n)Q)$ with $(x_{0,j+1}, z_{0,j+1}) \neq (0,0)$ by Theorem 2, using the fact that the difference between the two points is always Q which has $z_Q \neq 0$.

Also here $R\,[1]_{j+1} = 2R\,[1]_j = 2R\,[0]_j + 2Q = R\,[0]_{j+1} + Q$ as expected.

In the end when $j = m$ (the output case) the conditions become:

$$(x_{0,m}, z_{0,m}) = XZ(R\,[0]_m) = XZ(\textsc{msb}_m(n)Q) = XZ(nQ)$$
$$\text{with } (x_{0,m}, z_{0,m}) \neq (0,0)$$

We now simplify the naming removing the index m which is implied hereafter. In line 7 $y_0 = -y_Q z_0$ is calculated and in lines 8–10 x_0', y_0', z_0' are calculated as per Lemma 1. Then lines 11–14 select, in constant-time, based on the value of z_1 whether x, y, z take the value of x_0, y_0, z_0 (when $z_1 = 0$) or the value of x_0', y_0', z_0' (when $z_1 \neq 0$). After this, the algorithm uses x, y, z to compute $XY_1(nQ)$ where it returns $(0,0,0)$ if $z = 0$ else $(\frac{x}{z}, \frac{y}{z}, 1)$. This last step is done in lines 15–16. We still have to show that $z = 0 \iff nQ = \infty$ and otherwise $(\frac{x}{z}, \frac{y}{z}) = nQ$. We, as the algorithm, need to distinguish the following two cases on z_1:

Case 1. $z_1 \neq 0$ so $x, y, z = x_0', y_0', z_0'$. Here $R\,[1] = R\,[0] + Q \neq \infty$. If $R\,[0] \neq \infty$ also holds, then $(\frac{x}{z}, \frac{y}{z}) = nQ$ with $z \neq 0$ by Lemma 1. Otherwise if $z_0 = 0$ then $nQ = R\,[0] = \infty$ but also $z = z_0' = 0$ as expected.

Case 2. $z_1 = 0$ so $x, y, z = x_0, y_0, z_0$. In this case $z_1 = 0$ implies $R[1] = R[0] + Q = \infty$ and therefore $nQ = R[0] = -Q$, which also implies $R[0] \neq \infty$. Since $y = y_0 = -y_Q z_0 = -y_Q z$ going back to affine coordinates we get $(\frac{x}{z}, \frac{y}{z}) = nQ = -Q$ as expected.

A careful reader could also be worried about a possible point Q with $y_Q = 0$ as this would always make $z'_0 = 0$. This is not an exception. In fact such Q has order 2, so if n is even $R[0] = nQ = \infty$ and $R[1] = Q$, which imply $z_0 = 0, z_1 \neq 0$, thus Case 1 handles it correctly. Otherwise if n is odd $R[0] = nQ = Q$ but then $R[1] = 2Q = \infty$ so $z_0 \neq 0, z_1 = 0$ and Case 2 correctly returns $-Q = Q$.

Until now we supposed $f = 1$ but if $Q = \infty$, $f = 0$ and consequently $z_1 = 0$. Having both $z_0 = 0$ and $z_1 = 0$ for the initial points implies having always $z_{0,j} = 0$ and $z_{1,j} = 0$ in the loop. Therefore $z = z_{0,m} = 0$ and we end again in the situation where $z = 0$. The output will be $(0, 0, 0)$ i.e. ∞ as expected. □

4.1 Algorithm 3 as x-only Ladder and Equivalence with X25519

Algorithm 3 could also be used as an x-only ladder by computing nQ with any $y_Q \neq 0 \in \mathbb{F}_p$ (e.g. $y_Q = 1$) and ignoring the second value of the output. After line 6 the loop returns x_0, z_0, x_1, z_1. At this point if $z_1 = 0$ Algorithm 3 returns $\frac{x}{z} = \frac{x_0}{z_0}$, if on the other hand $z_1 \neq 0$ it returns $\frac{x}{z} = \frac{x'_0}{z'_0} = \frac{2y_Q x_0 z_1 z_0}{2y_Q z_1 z_0^2}$ which remembering that $y_Q \neq 0$ is equal to $\frac{x_0}{z_0}$ if $z_0 \neq 0$. $z_0 = 0$ implies $z = z'_0 = 0$ and so $nQ = \infty$ as expected.

It is important to underline that such a use of Algorithm 3 allows also input points $Q \in E(\mathbb{F}_{p^2}) \setminus \infty$ with $x_Q \in \mathbb{F}_p$, i.e. points on the quadratic twist of E. For a theoretical justification of this fact see Theorem 5 in Appendix A.

The following theorem shows the equivalence between our ladder as x-only ladder and X25519[1] proposed by Bernstein in [2].

Theorem 4 (Equivalence with Bernstein Montgomery Ladder). *Let p be a prime with $p \geq 5$, for all Montgomery curves $E_M : y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_p$ with $A^2 - 4$ non-square in $\mathbb{F}_p$, defining:*

$$x_{djb} = \text{BernsteinLadder}(p, A, x_1, n) \text{ as in [3]}$$

$$x, y, f = \text{Proposed-Ladder}\left(p, \frac{3 - A^2}{3}, \frac{2A^3 - 9A}{27}, n, x_1 + \frac{A}{3}, 1, 1, 1, 256\right)$$

Then $x_{djb} = x - f\frac{A}{3}$ for all $x_1 \in \mathbb{F}_p$ and for all $n \in \mathbb{Z}$.

Proof. $\forall x_1 \in \mathbb{F}_p, \exists\, Q \in E_M(\mathbb{F}_{p^2})$ such that $X_0(Q) = x_1$ and $\forall n \in Z, \exists\, x_{djb} \in \mathbb{F}_p$ such that $x_{djb} = X_0(nQ)$ where $X_0 : E_M(\mathbb{F}_{p^2}) \to \mathbb{F}_{p^2}$ with $X_0(\infty) = 0$ and $X_0(x, y) = x$. E_M is isomorphic to the short Weierstrass elliptic curve $E(\mathbb{F}_{p^2})$ with $a = \frac{3-A^2}{3}$ and $b = \frac{2A^3-9A}{27}$ through the isomorphism:

$$\phi : \begin{cases} \infty \to \infty \\ (x, y) \to (x + A/3, y) \end{cases}$$

[1] Montgomery ladder on the Montgomery curve Curve25519 with $A = 486662$, $B = 1$.

The two points $Q \in E_M(\mathbb{F}_{p^2})$ with $X_0(Q) = x_1$ are mapped through ϕ to $\tilde{Q} = \phi(Q) \in E(\mathbb{F}_{p^2})$. Two cases are now to be considered:

Case 1. $nQ = \infty$, then $X_0(nQ) = X_0(\infty) = 0 = x_{djb}$. Using the isomorphism $\phi(nQ) = \infty = n\phi(Q) = n\tilde{Q}$. In this case Algorithm 3 returns $XY_1(n\tilde{Q}) = (0, 0, 0)$. Therefore the statement holds: $x_{djb} = 0 = x - f\frac{A}{3}$

Case 2. $nQ \neq \infty$, then $X_0(nQ) = x_{djb}$. Algorithm 3 returns $XY_1(\phi(nQ)) = XY_1(n\tilde{Q}) = (x, y, 1)$, so we get through the isomorphism that $x = x_{djb} + \frac{A}{3}$ but then $x_{djb} = x - f\frac{A}{3}$ because $n\tilde{Q} \neq \infty \rightarrow f = 1$. □

4.2 Performance Evaluation

Our formulas, as defined by Algorithms 4 and 5, require the computation and storage of $b4 = 4 \cdot b$ and have a total cost per iteration in the main loop of Algorithm 3 of $10M, 5S, 2Ma, 2Mb4, 13a$,[2] which can be simplified if $a = -3$ to $10M, 5S, 2Mb4, 17a$. This implementation requires 8 auxiliary variables.

Algorithm 4. Doubling

```
Input: $x, z, a, b4 = 4 \cdot b \in \mathbb{F}_p$
Output: $x_2, z_2 \in \mathbb{F}_p$ s.t. $X(2Q) = \frac{x_2}{z_2}$
 1: $r_1 \leftarrow x,\ r_2 \leftarrow z$
 2: $r_7 \leftarrow r_2^2$
 3: $r_5 \leftarrow a \cdot r_7$
 4: $r_8 \leftarrow r_1^2$
 5: $r_6 \leftarrow r_8 + r_5$
 6: $r_5 \leftarrow r_8 - r_5$
 7: $r_5 \leftarrow r_5^2$
 8: $r_6 \leftarrow r_1 \times r_6$
 9: $r_7 \leftarrow b4 \times r_7$
10: $r_7 \leftarrow r_7 \times r_2$
11: $r_8 \leftarrow r_1 \times r_7$
12: $r_8 \leftarrow r_8 + r_8$
13: $r_1 \leftarrow r_5 - r_8$
14: $r_6 \leftarrow r_6 + r_6$
15: $r_6 \leftarrow r_6 + r_6$
16: $r_6 \leftarrow r_6 + r_7$
17: $r_2 \leftarrow r_2 \times r_6$
18: $x_2 \leftarrow r_1$
19: $z_2 \leftarrow r_2$
20: return $x_2, z_2$
```

Algorithm 5. Differential Addition

```
Input: $x, z, x', z', x_1, a, b4 = 4 \cdot b \in \mathbb{F}_p$
Output: $x_3, z_3 \in \mathbb{F}_p$ s.t. $X(Q + Q') = \frac{x_3}{z_3}$
 1: $r_1 \leftarrow x,\ r_2 \leftarrow z,\ r_3 \leftarrow x',\ r_4 \leftarrow z'$
 2: $r_6 \leftarrow r_1 \times r_4$
 3: $r_7 \leftarrow r_1 \times r_3$
 4: $r_5 \leftarrow r_2 \times r_4$
 5: $r_8 \leftarrow r_3 \times r_2$
 6: $r_3 \leftarrow r_6 - r_8$
 7: $r_4 \leftarrow r_3^2$
 8: $r_3 \leftarrow x1 \times r_4$
 9: $r_8 \leftarrow r_8 + r_6$
10: $r_6 \leftarrow a \cdot r_5$
11: $r_7 \leftarrow r_7 + r_6$
12: $r_7 \leftarrow r_7 \times r_8$
13: $r_5 \leftarrow r_5^2$
14: $r_6 \leftarrow b4 \times r_5$
15: $r_7 \leftarrow r_7 + r_7$
16: $r_7 \leftarrow r_7 + r_6$
17: $r_3 \leftarrow r_7 - r_3$
18: $x_3 \leftarrow r_3,\ z_3 \leftarrow r_4$
19: return $x_3, z_3$
```

Alternatively the loop can be rewritten similarly as [15, Algorithm 14] using only 7 auxiliary variables, for a cost per iteration of $8M, 7S, 3Ma, 1Mb4, 24a$ making it more appealing in implementations where squaring is optimized or when $a = -3$ resulting in a cost of $8M, 7S, 1Mb4, 31a$ thus asymptotic to $16M$. Our performances are almost identical to the Brier-Joye ladder. Izu et al. merged the Izu and Takagi differential addition and doubling achieving a total $13M, 4S, 18a$ [18, Appendix A.4] but their formula has exceptions since the speed-up is achieved by mixing the z-coordinates of the two points in the ladder, thus creating effects similar to the Co-Z ladder shown in Sect. 3.3. The recovery of the y-coordinate requires a total of $11M, 1S, 1Ma, 1Mb, 10a$ including the evaluation at line 7 of y_0 but not considering the conditional assignments, which could be implemented in

[2] M:multiplication, S:squaring, Ma, Mb, $Mb4$:multiplication by constants, a:addition.

Table 1. Operations costs per scalar bit for regular ladders without pre-computations except the last one where $w = 4$ indicates a 4-bit window

Ladder	$a \neq -3$						$a = -3$				
	M	S	Ma	Mb	a	**M/loop**	M	S	Mb	a	**M/loop**
Algorithm 3	10	5	2	2	13	20.3	8	7	1	31	19.1
Brier-Joye [8,19]	10	5	2	2	12	20.2	8	7	1	31	19.1
Improved Izu-Takagi [18]	10	4	2	1	18	18.8	10	4	1	23	17.3
(x,y) Co-Z [16, Algorithm 15]	8	6	0	0	43	18.3	8	6	0	43	18.3
x-only Co-Z [17, Algorithm 6]	10	5	0	0	13	16.3	10	5	0	13	16.3
Renes et al. [26] $w = 4$	11	3	3.75	2.5	20.75	22.33	11	3	2.5	28.25	19.33

***M/loop** is calculated as $1S = 1Ma = 1Mb = 1M$ and $1a = 0.1M$

different ways. Table 1 compares our costs with the ones of other scalar multiplication algorithms including a windowed ladder which uses the complete formula for odd order curves by Renes et at. [26]. It presents exceptions although only when used on even order curves (e.g. curve25519).

4.3 Security Considerations

Due to the regular structure of the Montgomery ladder Algorithm 3 is protected against timing attacks and simple power analysis. To protect against DPA-like attacks our ladder can use both the randomization of projective coordinates and scalar blinding. Note that on a Montgomery ladder the randomization of the projective coordinates usually requires additional costs for each iteration step of the ladder as the formulas are derived assuming $z_Q = 1$. However Okeya et al. showed that randomizing x_1 and z_1 in line 2 is enough [25] and costs only $2M$. We underline the fact that scalar blinding has been claimed to be an ineffective countermeasure, but this happens only when r is chosen too small. If r is chosen accordingly to the form of $\#E(\mathbb{K})$ then scalar blinding is secure as expected [27]. Furthermore the combination of scalar blinding and coordinates randomization has been proved to provide resistance to some *horizontal* attacks, e.g. [12]. Horizontal attacks are more powerful than DPA as the attacker utilizes only a single trace. For some of these attacks ([1,11]) the only known countermeasures are internal to the multiplication algorithm. These are independent of the scalar multiplication structure and are thus out of scope here. In absence of fault attacks it is interesting to note that our proposed ladder does not require any check on the input point if used with a twist secure elliptic curve. When, on the contrary, the curve has a twist with weak order, Algorithm 3 can be modified to check if the output point lies indeed on the proper curve. This, combined with the y-recovery, acts as a fault countermeasure as described by Ebeid and Lambert [13]. We suggest to perform the check just after the y-recovery and also after the conversion to affine coordinates as the cost of point checking is very limited (for both checks the total cost is $5M, 7S$).

To handle with input/output points equal to ∞ Algorithm 3 makes use of a flag f, which might be target of faults. We claim that no additional information on the value of the scalar n could be obtained in this way. In fact f is used in 3 different occasions: (1) for the input point Q, (2) for the point $\frac{x_1}{z_1}$ at the end of the main loop and (3) for the result nQ. In case (1) if $Q \neq \infty$ the result will be ∞, independently of n, while the case where $Q = \infty$ is no different from the unfaulted case where the attacker chooses the input point. In case (2) the faulted result will be $-Q$ (when $nQ \neq -Q$), giving no information about n, or ∞ (when $nQ = -Q$), giving complete knowledge of n *exactly* as in the unfaulted scenario. In case (3) the output coordinates x and y will not change, while the resulting f will change. This might be exploited at protocol level but it will not reveal any information on n.

5 Conclusion

To the best of our knowledge this paper presented the first regular and constant-time scalar multiplication algorithm *proved* to work for *all* points on *all* short Weierstrass elliptic curves with *any* scalar value. This last point, as we showed in Sect. 3, is something that other state-of-the-art algorithms fail to achieve. The presented ladder has performances comparable with other regular ladders for short Weierstrass curves, it supports scalar blinding combined with projective coordinates randomization as countermeasure for side-channel attacks and, thanks to its internal structure of an x-only Montgomery ladder coupled with y-recovery, provides intrinsic protection against fault attacks.

All these properties make it appealing as possible low area hardware accelerator for secure microcontrollers where several ECC operations, on different elliptic curve models, might be required.

A X-only Scalar Multiplication

We now state and prove an analogous version of Theorem 2.1 of [2] which holds for short Weierstrass curves (together with the function X_1). The definition of $X_1 : E(\mathbb{F}_{p^2}) \to \mathbb{F}_{p^2} \times \{0, 1\}$ is $X_1(\infty) = (0, 0), X_1(x, y) = (x, 1)$ and in the following lines we will indicate with δ the smallest non-square integer in $\mathbb{F}_p$. For an exact description of the operations on the extension field $\mathbb{F}_{p^2}$ see [2, Appendix A] but the addition law we will refer to, is the one described in Sect. 2.

First note that the following three sets are subgroups of $E(\mathbb{F}_{p^2})$:

- $\{\infty, \text{points of order two}\}$. Indeed if there is just one point of order two $\infty + \infty = \infty$; $(q_1, 0) + (q_1, 0) = \infty$; and $(q_1, 0) + \infty = (q_1, 0)$. If there are three points we have in addition that $(q_1, 0) + (q_2, 0) = (q_3, 0)$.

– $\{\infty\} \cup (E(\mathbb{F}_{p^2}) \cap (\mathbb{F}_p \times \mathbb{F}_p))$. Indeed, if $x, y, x', y' \in \mathbb{F}_p$ then the quantities λ, x_3, y_3 defined as in Eq. 2 are all in $\mathbb{F}_p$.
– $\{\infty\} \cup (E(\mathbb{F}_{p^2}) \cap (\mathbb{F}_p \times \sqrt{\delta}\mathbb{F}_p))$. This time λ is an element of $\sqrt{\delta}\mathbb{F}_p$ and therefore $x_3 \in \mathbb{F}_p$ while y_3 will be an element of $\sqrt{\delta}\mathbb{F}_p$.

Theorem 5. *Let n be an integer and $(q, f_0) \in \mathbb{F}_p \times \{0,1\}$. Then there exists a unique couple $(s, f_1) \in \mathbb{F}_p \times \{0,1\}$ such that $X_1(nQ) = (s, f_1)$ for all $Q \in E(\mathbb{F}_{p^2})$ such that $X_1(Q) = (q, f_0)$.*

Proof. We first consider $f_0 = 0$. The only Q satisfying $\big\{Q \in \mathbb{F}_{p^2} : X_1(Q) = (0,0)\big\}$ is ∞, so $nQ = \infty$ and $X_1(nQ) = (0,0)$. If $f_0 = 1$ we define, as Bernstein did in [2], $\alpha := q^3 + aq + b$ and check different cases for α.

Case 1. $\alpha = 0$. The only square root of 0 in $\mathbb{F}_{p^2}$ is 0 and therefore we are speaking only of possible points of order two. A curve E could have no points of order two, in this case $\big\{Q \in \mathbb{F}_{p^2} : X_1(Q) = (q,1)\big\} = \{\emptyset\}$. It could have just one single root q_1; then the point $(q_1, 0)$ would be contained in the group $\{\infty, (q_1, 0)\}$. This is a subgroup of $E(\mathbb{F}_{p^2})$ and therefore nQ will lie in it. Depending on the scalar n we will have $X_1(nQ) = (0,0)$ or $X_1(nQ) = (q_1, 1)$. If the polynomial has two roots in $\mathbb{F}_p$ it has the third too and in this last case the set $\{\infty, (q_1, 0), (q_2, 0), (q_3, 0)\}$ is again a subgroup of $E(\mathbb{F}_{p^2})$. The output will be $X_1(nQ) = (0,0)$ or one of the three points returning $X_1(nQ) = (q_i, 1)$. In all these cases $X_1(nQ) \in \mathbb{F}_p \times \{0,1\}$.

Case 2. α nonzero square in $\mathbb{F}_p$. The square roots of α are $\pm r \in \mathbb{F}_p$. Therefore $\big\{Q \in \mathbb{F}_{p^2} : X_1(Q) = (q,1)\big\} = \big\{(q,r), (q,-r)\big\}$. These are contained in the group $\{\infty\} \cup (E(\mathbb{F}_{p^2}) \cap (\mathbb{F}_p \times \mathbb{F}_p))$ and nQ too. Notice that $n(q,-r) = n(-(q,r)) = -n(q,r)$. The function X_1 considers only the x-coordinate and eventually the infinity, so $X_1(n(q,-r)) = X_1(n(q,r))$ which is equal, depending on the scalar n to $(s,1)$ or $(0,0)$. In both cases $X_1(nQ) \in \mathbb{F}_p \times \{0,1\}$.

Case 3. α non-square in $\mathbb{F}_p$. By definition of δ, $\frac{\alpha}{\delta}$ is a nonzero square in $\mathbb{F}_p$ with roots $\pm r \in \mathbb{F}_p$; then the only square roots of α are $\pm r\sqrt{\delta} \in \mathbb{F}_{p^2}$. The two points $(q, r\sqrt{\delta}), (q, -r\sqrt{\delta})$ are contained in the group $\{\infty\} \cup (E(\mathbb{F}_{p^2}) \cap (\mathbb{F}_p \times \sqrt{\delta}\mathbb{F}_p)$ so it contains also $n(q, r\sqrt{\delta})$ and $n(q, -r\sqrt{\delta})$ which again are opposite and have the same $X_1(nQ) = (s,1)/(0,0)$ depending on n with s still in $\mathbb{F}_p$ as shown above. □

B Proof of Lemma 1

The hypothesis $R[0], R[1] \neq \infty$ implies $y_Q \neq 0$. In fact if $y_Q = 0$ then Q would have had order 2 causing either $R[0]$ or $R[1]$ $(= R[0] + Q)$ to be ∞. Since $y_Q, z_0, z_1 \neq 0$, then $z_0' \neq 0$. It is easy to see that $\frac{x_0'}{z_0'} = \frac{2y_Q x_0 z_1 z_0}{2y_Q z_1 z_0^2} = \frac{x_0}{z_0} = X\,(R[0])$. For the y-coordinate of $R[0]$, hereafter referred to as $y_{R[0]}$, we have two cases.

Case 1. $R\,[0] \neq Q$. Using Eq. 2 we have $\frac{x_1}{z_1} = \left(\frac{y_{R[0]} - y_Q}{x_0/z_0 - x_Q}\right)^2 - x_Q - \frac{x_0}{z_0} = \frac{-2y_Q y_{R[0]} + 2b + (a + x_Q\frac{x_0}{z_0})(x_Q + \frac{x_0}{z_0})}{(x_0/z_0 - x_Q)^2}$. Multiplying by $(x_0/z_0 - x_Q)^2 \neq 0$ and dividing by $2y_Q$ we obtain $y_{R[0]} = \frac{2bz_1z_0^2 + z_1(az_0 + x_Q x_0)(x_Q z_0 + x_0) - x_1(x_0 - x_Q z_0)^2}{2y_Q z_1 z_0^2} = \frac{y_0'}{z_0'}$.

Case 2. $R[0] = Q$ (i.e. $n \equiv 1 \mod \text{ord}(Q)$). Applying the equivalence $x_0 = z_0 x_Q$ we get $\frac{y'_0}{z'_0} = \frac{2bz_0^2+2x_Q z_0(az_0+x_Q^2 z_0)}{2y_Q z_0^2} = \frac{x_Q^3+ax_Q+b}{y_Q} = \frac{y_Q^2}{y_Q} = y_Q$ as expected. $\square$

References

1. Bauer, A., Jaulmes, É., Prouff, E., Reinhard, J.R., Wild, J.: Horizontal collision correlation attack on elliptic curves. Cryptogr. Commun. **7**(1), 91–119 (2015)
2. Bernstein, D.J.: Curve25519: new Diffie-Hellman speed records. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 207–228. Springer, Heidelberg (2006). doi:10.1007/11745853_14
3. Bernstein, D.J.: Does the Curve25519 Montgomery ladder always work? CFRG Mailing List. https://www.ietf.org/mail-archive/web/cfrg/current/msg05004.html (2014). Accessed 22 Jan 2016
4. Bernstein, D.J., Lange, T.: Faster addition and doubling on elliptic curves. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 29–50. Springer, Heidelberg (2007). doi:10.1007/978-3-540-76900-2_3
5. Bernstein, D.J., Lange, T.: Explicit-Formulas Database (2016). http://hyperelliptic.org/EFD
6. Biehl, I., Meyer, B., Müller, V.: Differential fault attacks on elliptic curve cryptosystems. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 131–146. Springer, Heidelberg (2000). doi:10.1007/3-540-44598-6_8
7. Bos, J.W., Costello, C., Longa, P., Naehrig, M.: Selecting elliptic curves for cryptography: an efficiency and security analysis. J. Cryptographic Eng. **6**(4), 259–286 (2016). doi:10.1007/s13389-015-0097-y
8. Brier, E., Joye, M.: Weierstraß elliptic curves and side-channel attacks. In: Naccache and Paillier [24], pp. 335–345
9. Brumley, B.B., Tuveri, N.: Remote timing attacks are still practical. In: Atluri, V., Diaz, C. (eds.) ESORICS 2011. LNCS, vol. 6879, pp. 355–371. Springer, Heidelberg (2011). doi:10.1007/978-3-642-23822-2_20
10. Coron, J.-S.: Resistance against differential power analysis for elliptic curve cryptosystems. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 292–302. Springer, Heidelberg (1999). doi:10.1007/3-540-48059-5_25
11. Danger, J.-L., Guilley, S., Hoogvorst, P., Murdica, C., Naccache, D.: Improving the Big Mac attack on elliptic curve cryptography. In: Ryan, P.Y.A., Naccache, D., Quisquater, J.-J. (eds.) The New Codebreakers. LNCS, vol. 9100, pp. 374–386. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49301-4_23
12. Dugardin, M., Papachristodoulou, L., Najm, Z., Batina, L., Danger, J.-L., Guilley, S.: Dismantling real-world ECC with horizontal and vertical template attacks. In: Standaert, F.-X., Oswald, E. (eds.) COSADE 2016. LNCS, vol. 9689, pp. 88–108. Springer, Heidelberg (2016). doi:10.1007/978-3-319-43283-0_6
13. Ebeid, N.M., Lambert, R.: Securing the elliptic curve montgomery ladder against fault attacks. In: Breveglieri, L., Koren, I., Naccache, D., Oswald, E., Seifert, J. (eds.) Sixth International Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2009, Lausanne, Switzerland, 6 September 2009, pp. 46–50. IEEE Computer Society (2009)
14. Fouque, P., Lercier, R., Réal, D., Valette, F.: Fault attack on elliptic curve Montgomery ladder implementation. In: Breveglieri, L., Gueron, S., Koren, I., Naccache, D., Seifert, J. (eds.) Fifth International Workshop on Fault Diagnosis and Tolerance in Cryptography, 2008, FDTC 2008, Washington, DC, USA, 10 August 2008, pp. 92–98. IEEE Computer Society (2008)

15. Goundar, R.R., Joye, M., Miyaji, A.: Co-Z addition formulæ and binary ladders on elliptic curves. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 65–79. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15031-9_5
16. Goundar, R.R., Joye, M., Miyaji, A., Rivain, M., Venelli, A.: Scalar multiplication on Weierstraß elliptic curves from Co-Z arithmetic. J. Cryptogr. Eng. **1**(2), 161–176 (2011)
17. Hutter, M., Joye, M., Sierra, Y.: Memory-constrained implementations of elliptic curve cryptography in Co-Z coordinate representation. In: Nitaj, A., Pointcheval, D. (eds.) AFRICACRYPT 2011. LNCS, vol. 6737, pp. 170–187. Springer, Heidelberg (2011). doi:10.1007/978-3-642-21969-6_11
18. Izu, T., Möller, B., Takagi, T.: Improved elliptic curve multiplication methods resistant against side channel attacks. In: Menezes, A., Sarkar, P. (eds.) INDOCRYPT 2002. LNCS, vol. 2551, pp. 296–313. Springer, Heidelberg (2002). doi:10.1007/3-540-36231-2_24
19. Izu, T., Takagi, T.: A fast parallel elliptic curve multiplication resistant against side channel attacks. In: Naccache and Paillier [24], pp. 280–296
20. Izu, T., Takagi, T.: Exceptional procedure attack on elliptic curve cryptosystems. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 224–239. Springer, Heidelberg (2003). doi:10.1007/3-540-36288-6_17
21. Joye, M., Tymen, C.: Protections against differential analysis for elliptic curve cryptography — an algebraic approach —. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 377–390. Springer, Heidelberg (2001). doi:10.1007/3-540-44709-1_31
22. Kim, K. (ed.): ICISC 2001. LNCS, vol. 2288. Springer, Heidelberg (2002)
23. Montgomery, P.L.: Speeding the Pollard and elliptic curve methods of factorization. Math. Comput. **48**(177), 243–264 (1987)
24. Naccache, D., Paillier, P. (eds.): PKC 2002. LNCS, vol. 2274. Springer, Heidelberg (2002)
25. Okeya, K., Miyazaki, K., Sakurai, K.: A fast scalar multiplication method with randomized projective coordinates on a montgomery-form elliptic curve secure against side channel attacks. In: Kim [22], pp. 428–439
26. Renes, J., Costello, C., Batina, L.: Complete addition formulas for prime order elliptic curves. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9665, pp. 403–428. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49890-3_16
27. Schindler, W., Wiemers, A.: Efficient Side-Channel Attacks on Scalar Blinding on Elliptic Curves with Special Structure. NIST Workshop on ECC Standards (2015)
28. Yen, S., Kim, S., Lim, S., Moon, S.: A countermeasure against one physical cryptanalysis may benefit another attack. In: Kim [22], pp. 414–427