Scalar multiplication in compressed coordinates in the trace-zero subgroup

Giulia Bianco and Elisa Gorla*

Institut de Mathématiques, Université de Neuchâtel
Rue Emile-Argand 11, CH-2000 Neuchâtel, Switzerland

Abstract

We consider trace-zero subgroups of elliptic curves over a degree three field extension. The elements of these groups can be represented in compressed coordinates, i.e. via the two coefficients of the line that passes through the point and its two Frobenius conjugates. In this paper we give the first algorithm to compute scalar multiplication in the degree three trace-zero subgroup using these coordinates.

Introduction

Given an elliptic curve E defined over a finite field $\mathbb{F}_q$, an odd prime n and the group $E(\mathbb{F}_{q^n})$ of $\mathbb{F}_{q^n}$-rational points of E, the trace-zero subgroup T_n of $E(\mathbb{F}_{q^n})$ consists of the $\mathbb{F}_{q^n}$-rational points of E whose trace is zero. Trace-zero subgroups were first proposed for cryptographic applications by Frey in [6], and they turn out to provide good security, efficient computation, and optimal data storage.

It is easy to show that solving the DLP in T_n is as hard as solving the DLP in the entire group $E(\mathbb{F}_{q^n})$ (see e.g. [8, Proposition 1]). Moreover, if E is supersingular, an analogous result holds for the security parameter in the contest of pairing-based cryptography (see [13] and [14]). In particular, the cardinality of $T_3 \subseteq E(\mathbb{F}_{q^3})$ is in the range of q^2 and the complexity of the DLP is $\mathcal{O}(q)$, that is, the square root of the group order (see [1, Section 22.3.4.b]). Hence, from the point of view of security, the degree three trace-zero subgroup of an elliptic curve defined over $\mathbb{F}_q$ is comparable to the group of points of an elliptic curve over a ground field $\mathbb{F}_p$, where p is in the range of q^2.

On the other hand, Weil restriction of scalars allows us to regard $E(\mathbb{F}_{q^n})$ as the set of $\mathbb{F}_q$-rational points of a variety of dimension n defined over $\mathbb{F}_q$, and T_n as the set of $\mathbb{F}_q$-rational points of a subvariety of dimension $n-1$. Hence one would like to be able to represent the elements of T_n via $n-1$ $\mathbb{F}_q$-coordinates, as opposed to the n $\mathbb{F}_q$-coordinates needed to represent an element of $E(\mathbb{F}_{q^n})$. Optimal representations for the degree n trace-zero subgroup of an elliptic curve have been proposed by Naumann in [12] for $n = 3$, Silverberg in [15] and Cesena in [4] for $n = 3, 5$, and Gorla-Masserier in [8] for small values of n and in [9] for any n.

*The research reported in this paper was partially supported by the Swiss National Science Foundation under grant no. 200021_150207.

Optimal coordinates for the degree n trace-zero subgroup of a hyperelliptic curves of genus g were proposed by Lange in [10] for $g = 2$ and $n = 3$, and by Gorla-Massierer in [9] for any $g \geq 1$ and $n \geq 2$.

In order to take full advantage of the optimal representation size for level of security in trace-zero subgroups, one needs efficient algorithms to perform arithmetic on the group elements represented in compressed coordinates. There are two natural ways to perform scalar multiplication in T_n: One can either compute scalar multiplication in $E(\mathbb{F}_{q^n})$ and use compression and decompression algorithms to go back and forth between the usual coordinates in $E(\mathbb{F}_{q^n})$ and the compressed coordinates in T_n, or compute scalar multiplication directly in compressed coordinates in T_n.

The first approach is relatively straightforward: In all previously quoted work dealing with optimal representations in T_n, the authors provide compression and decompression algorithms. There is a wealth of knowledge on how to efficiently perform scalar multiplication on elliptic curves and, in addition, the Frobenius endomorphism φ on the curve allows us to speed up scalar multiplication in $E(\mathbb{F}_{q^n})$, as explained in [1, Sections 15.1 and 15.2]. Following this approach, computing scalar multiplication in T_3 is usually faster than in the group of rational points of a curve over a ground field of prime size in the range of q^2. Observe also that in T_3 scalar multiplications can be further sped up by using the relation $\varphi^2 + \varphi + 1 = 0$ involving the Frobenius endomorphism (see [1, Section 15.3], [2], [3], [10], [12], [16]). Using the same approach, one can also speed up the computation of the Miller function for the Tate pairing, in the context of pairing-based cryptography (see [4]).

The second approach is performing scalar multiplication in T_n in the optimal compressed coordinates. To the extent of our knowledge, no such algorithm has been proposed yet. In this paper, we give an algorithm to perform scalar multiplication in the degree three trace-zero subgroup of an elliptic curve, in the representation proposed in [9]. Namely, let E be an elliptic curve over $\mathbb{F}_q$, whose degree three trace-zero subgroup T_3 is cyclic of prime order p. Our algorithm takes as input an integer m modulo p and the line through $P \in T_3$ and its Frobenius conjugates, and it returns the line through the point mP and its Frobenius conjugates. Our algorithm has interesting similarities with the Montgomery ladder algorithm for computing scalar multiplication for elliptic curves, when the points are represented using their x-coordinate (see [11] and [1, Section 13.2.3.d]). Moreover, our algorithm adapts the above mentioned strategy for exploiting the relation $\varphi^2 + \varphi + 1 = 0$ satisfied by the Frobenius endomorphism. Hence, we can maintain the advantages of such a strategy, even performing the operation directly in compressed coordinates.

The paper is organized as follows. In Section 1 we establish the notations and some preliminaries on the degree three trace-zero subgroup of an elliptic curve. We also present some procedures for computation, that will be used in the subsequent algorithms. In Section 2 we present our algorithm for scalar multiplication. Subsection 2.1 contains a subalgorithm that will be called by the main algorithms, and a lemma which allows us to deal with special cases. In Subsection 2.2 we propose a Montgomery-ladder-style algorithm which computes scalar multiplication in T_3. The algorithm makes use of the subalgorithm of Subsection 2.1. In Subsection 2.3 we exploits the properties of the Frobenius endomorphism to obtain an optimized version of the Montgomery-ladder-style algorithm of Subsection 2.2. The resulting algorithm efficiently computes scalar multiplication in T_3. In the Appendix we give the explicit formulas that we have computed and that we use for computation.

1 Setting, notation, and formulas

1.1 Preliminaries and notation

Let $\mathbb{F}_q$ be a finite field of characteristic different from 2 and 3. Let E be an elliptic curve defined over $\mathbb{F}_q$ by an equation in short Weierstrass form, i.e. E is the zero-locus of a polynomial of the form $y^2 - f(x)$, where $f(x) = x^3 + Ax + B$ has no multiple roots and $A, B \in \mathbb{F}_q$. Denote by $+$ the usual addition between points of E and by P_∞ the neutral element of E. For a field extension $\mathbb{F}_q \subseteq \mathbb{F}_{q^n}$, denote by $E(\mathbb{F}_{q^n})$ the group of $\mathbb{F}_{q^n}$-rational points of E.

Consider the Frobenius endomorphism on the group of $\mathbb{F}_{q^3}$-rational points of E:

$$\varphi : E(\mathbb{F}_{q^3}) \longrightarrow E(\mathbb{F}_{q^3}), \quad (x, y) \mapsto (x^q, y^q),\ P_\infty \mapsto P_\infty.$$

The Frobenius endomorphism induces the trace endomorphism:

$$\mathrm{Tr} : E(\mathbb{F}_{q^3}) \longrightarrow E(\mathbb{F}_q), \quad P \mapsto P + \varphi(P) + \varphi^2(P),$$

whose kernel is the trace zero subgroup T_3 of $E(\mathbb{F}_{q^3})$, i.e.

$$T_3 = \{P \in E(\mathbb{F}_{q^3}) : P + \varphi(P) + \varphi^2(P) = P_\infty\}.$$

Let $P = (x_P, y_P) \in T_3 \setminus \{P_\infty\}$ and denote by h_P the equation of the line through P, $\varphi(P)$, $\varphi^2(P)$. Then

$$h_P = y - (\alpha_1 x + \alpha_0) \tag{1}$$

with $\alpha_1, \alpha_0 \in \mathbb{F}_q$. By [9, Corollary 4.2], h_P of the form (1) exists and is unique. Notice moreover that

$$h_{-P}(x, y) = -h_P(x, -y) = y + (\alpha_1 x + \alpha_0).$$

Following [9], we represent an element $P \in T_3 \setminus \{P_\infty\}$ via the coefficients (α_0, α_1) of h_P. Such a representation is optimal in size, since T_3 is a variety of dimension 2 over $\mathbb{F}_q$. Intuitively, optimality means that the number of coordinates is the least possible, see [9, Definition 2.7] for the formal definition of an optimal representation. In this paper we give an algorithm to compute scalar multiplication in T_3 using the representation from [9]. Scalar multiplication is the operation needed in most applications, e.g. in the Diffie-Hellman key agreement.

Notice that the representation that we use identifies each point with its Frobenius conjugates. As a consequence, addition in compressed coordinates is not well-defined, that is, h_P and h_Q do not determine h_{P+Q}. However, scalar multiplication is well-defined: Given the line $h_P = 0$ and an integer m, the line $h_{mP} = 0$ through mP and its Frobenius conjugates is uniquely determined. Observe the analogy with the representation of points of E via their x-coordinates: m and the x-coordinate of a point $P \in E$ determine the x-coordinate of mP, however the x-coordinates of P and Q do not determine the x-coordinate of the point $P + Q$.

In spite of the fact that one cannot compute h_{P+Q} from h_P and h_Q, one can compute the polynomial $S_{P,Q} \in \mathbb{F}_q[x, y]$ such that

$$\mathrm{div}(S_{P,Q}) = \sum_{0 \le i,j \le 2} (\varphi^i(P) + \varphi^j(Q)) - 9P_\infty.$$

The polynomial $S_{P,Q}$ is unique up to multiplication by a nonzero constant and it is of the form

$$S_{P,Q} = (S_{P,Q})_1 + y(S_{P,Q})_2 = (a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0) + y(b_3x^3 + b_2x^2 + b_1x + b_0).$$

Notice that, if $P+Q, P+\varphi(Q), P+\varphi^2(Q) \neq P_\infty$, then

$$S_{P,Q} = h_{P+Q} h_{P+\varphi(Q)} h_{P+\varphi^2(Q)} \mod y^2 - f(x). \tag{2}$$

From h_P and $S_{P,Q}$ one can compute the polynomials

$$H_P := f - (\alpha_1 x + \alpha_0)^2,\ \Sigma_{P,Q} := f(S_{P,Q})_2^2 - (S_{P,Q})_1^2 \in \mathbb{F}_q[x].$$

In the next lemma we collect a few useful facts.

Lemma 1. *Let $H_P = f - (\alpha_1 x + \alpha_0)^2, \Sigma_{P,Q} = f(S_{P,Q})_2^2 - (S_{P,Q})_1^2$. The following equalities hold, up to a nonzero constant:*

1. $H_P = h_P h_{-P} \mod y^2 - f(x)$,
2. $H_P = (x - x_P)(x - x_P^q)(x - x_P^{q^2})$,
3. $S_{-P,-Q}(x,y) = S_{P,Q}(x,-y)$,
4. $\Sigma_{P,Q} = S_{P,Q} S_{-P,-Q} \mod y^2 - f(x)$,
5. $\Sigma_{P,Q} = \prod_{0 \le i,j \le 2}(x - x_{\varphi^i(P)+\varphi^j(Q)})$.

Moreover, the following are equivalent:

6. $(S_{P,Q})_2 = 0$,
7. $b_3 = 0$,
8. $\varphi^i(P) + \varphi^j(Q) = P_\infty$ *for some* i, j,
9. $\mathrm{div}(S_{P,Q}) = (P - \varphi(P)) + (\varphi(P) - P) + (P - \varphi^2(P)) + (\varphi^2(P) - P) + (\varphi(P) - \varphi^2(P)) + (\varphi^2(P) - \varphi(P)) - 6P_\infty$.

Proof. 1. and *2.* follow from [9, Corollary 4.2].
3. Observe that $\mathrm{div}(S_{-P,-Q}) = \sum_{0 \le i,j \le 2}(-\varphi^i(P) - \varphi^j(Q)) - 9P_\infty$, hence

$$S_{-P,-Q}(x,y) = (S_{P,Q})_1(x) - y(S_{P,Q})_2(x) = S_{P,Q}(x,-y)$$

up to a nonzero constant.
4. By 3. $S_{P,Q} S_{-P,-Q} = (S_{P,Q})_1^2 - y^2 (S_{P,Q})_2^2 = \Sigma_{P,Q}$, up to a nonzero constant.
5. By 4.

$$\mathrm{div}(\Sigma_{P,Q}) = \mathrm{div}(S_{P,Q}) + \mathrm{div}(S_{-P,-Q}) = \sum_{0 \le i,j \le 2}(\varphi^i(P) + \varphi^j(Q)) + \sum_{0 \le i,j \le 2}(-\varphi^i(P) - \varphi^j(Q)) - 18P_\infty,$$

hence $\Sigma_{P,Q} = \prod_{0 \le i,j \le 2}(x - x_{\varphi^i(P)+\varphi^j(Q)})$ up to a nonzero constant.
7. $\Rightarrow$ *8.* If $b_3 = 0$, then $\deg(\Sigma_{P,Q}) \le 8$, hence one of the sums $\varphi^i(P) + \varphi^j(Q)$ must be P_∞.
8. $\Rightarrow$ *9.* If $\varphi^i(P) + \varphi^j(Q) = P_\infty$ for some i and j, then $S_{P,Q} = S_{\varphi^i(P),\varphi^j(Q)} = S_{\varphi^i(P),-\varphi^i(P)} = S_{P,-P}$. Hence the zeroes of $S_{P,Q}$ on E are $\pm(P - \varphi(P)), \pm(P - \varphi^2(P)), \pm(\varphi(P) - \varphi^2(P))$ and P_∞, the latter with multiplicity six.
9. $\Rightarrow$ *6.* Since the zeroes of $S_{P,Q}$ on E are $\pm(P - \varphi(P)), \pm(P - \varphi^2(P)), \pm(\varphi(P) - \varphi^2(P))$ and P_∞ with multiplicity six, then $S_{P,Q} = (x - x_{P-\varphi(P)})(x - x_{P-\varphi^2(P)})(x - x_{\varphi(P)-\varphi^2(P)}) \in \mathbb{F}_q[x]$. Hence $(S_{P,Q})_2 = 0$. □

1.2 Procedures for computing doubling and tripling formulas, and the coefficients of $S_{P,Q}$

In this subsection we describe two procedures which allow us to compute doubling and tripling formulas for the equation of a line, and the coefficients of the polynomial $S_{P,Q}$. More precisely:

- Following Procedure 1, we were able to write explicit formulas for the coefficients of $S_{P,Q}$ in terms of the coefficients of h_P and h_Q (see formulas (1) in the appendix) and for the coefficients of h_{2P} in terms of the coefficients of h_P (see formulas (2) in the appendix).

- Following Procedure 2, we wrote explicit formulas for the coefficients of h_{3P} in terms of the coefficients of h_P (see formulas (3) in the appendix).

Moreover, in Proposition 5 we give a procedure to compute the coefficients of h_{P+Q} in terms of the coefficients of H_{P+Q} and $S_{P,Q}$. We assume that $(S_{P,Q})_2 \neq 0, H_{P+Q}$ and that H_{P+Q} is irreducible over $\mathbb{F}_q[x]$ (i.e., that $P+Q \notin E[3](\mathbb{F}_q)$).

Notation 2. For Procedures 1 and 2, we let $\varphi^{i-1}(P) = P_i = (x_{P_i}, y_{P_i})$, respectively $\varphi^{i-1}(Q)) = Q_i = (x_{Q_i}, y_{Q_i})$ for $i \in \{1,2,3\}$. We denote by e_1, e_2, e_3 the symmetric polynomials in $x_{P_1}, x_{P_2}, x_{P_3}$ and by s_1, s_2, s_3 the symmetric polynomials in $x_{Q_1}, x_{Q_2}, x_{Q_3}$.

Procedure 1. Procedure to write formulas for the coefficients of h_{2P} in terms of those of h_P and for the coefficients of $S_{P,Q}$ in terms of those of h_P and h_Q.

---

for $i \in \{1,2,3\}$ $\quad \triangleright\ t_i = 0$ tangent to E in P_i, t_i polynomial in the variables x_{P_i}, y_{P_i}, x, y
  $t_i(x_{P_i}, y_{P_i}, x, y) \leftarrow f'(x_{P_i})x - 2y_{P_i}y + (2y_{P_i}^2 - f'(x_{P_i})x_{P_i})$
  for $j \in \{1,2,3\}$ $\quad \triangleright\ r_{ij} = 0$ line through P_i and Q_j, r_{ij} polynomial in the variables $x_{P_i}, y_{P_i}, x_{Q_j}, y_{Q_j}, x, y$
    $r_{ij}(x_{P_i}, x_{Q_j}, y_{P_i}, y_{Q_j}, x, y) \leftarrow (y_{Q_j} - y_{P_i})x + (x_{P_i} - x_{Q_j})y + ((x_{Q_j} - x_{P_i})y_{P_i} + (y_{P_i} - y_{Q_j})x_{P_i})$
  end for
end for
$T(x_{P_1}, x_{P_2}, x_{P_3}, y_{P_1}, y_{P_2}, y_{P_3}, x, y) \leftarrow \prod_{i=1}^{3} t_i$
$R(x_{P_1}, x_{P_2}, x_{P_3}, y_{P_1}, y_{P_2}, y_{P_3}, x_{Q_1}, x_{Q_2}, x_{Q_3}, y_{Q_1}, y_{Q_2}, y_{Q_3}, x, y) \leftarrow \prod_{1 \leq i,j \leq 3} r_{i,j}$
for $i \in \{1,2,3\}$
  replace y_{P_i} with $(\alpha_1 x_{P_i} + \alpha_0)$ in T and in R
  replace y_{Q_i} with $(\beta_1 x_{Q_i} + \beta_0)$ in R
end for
write $T(x_{P_1}, x_{P_2}, x_{P_3})$, $R(x_{P_1}, x_{P_2}, x_{P_3})$ as polynomials in e_1, e_2, e_3
write $R(x_{Q_1}, x_{Q_2}, x_{Q_3})$ as a polynomial in s_1, s_2, s_3
$E_1 \leftarrow \alpha_1^2$, $E_2 \leftarrow A - 2\alpha_0\alpha_1$, $E_3 \leftarrow \alpha_0^2 - B$
$S_1 \leftarrow \beta_1^2$, $S_2 \leftarrow A - 2\beta_0\beta_1$, $S_3 \leftarrow \beta_0^2 - B$
for $i \in \{1,2,3\}$
  replace e_i with E_i in T, R
  replace s_i with S_i in R
end for
recover h_{2P} via the equality (up to multiplication by a nonzero constant):

$$h_{2P} = T(x, -y)/(h_{-P}^2) \mod y^2 - f(x).$$

22: recover $S_{P,Q}$ via the equality (up to multiplication by a nonzero constant):

$$(S_{P,Q})_1(x) - y(S_{P,Q})_2(x) = R(x,y)/(h_P^3 h_Q^3) \mod y^2 - f(x).$$

Theorem 3. *Procedure 1 is correct.*

Proof. We first prove that the formulas of Procedure 1 are correct when $h_P \neq y$ and $h_Q \neq h_{\pm P}$. We regard $x_{P_1}, x_{P_2}, x_{P_3}, x_{Q_1}, x_{Q_2}, x_{Q_3}$ as variables. Since $h_P \neq y$, one has that $2P_i \neq P_\infty$ for $i \in \{1,2,3\}$, so $t_i(x_{P_i}, y_{P_i}, x, y) = 0$ the equation defining the tangent to E at P_i is of the form given in line 2 and $\operatorname{div}(t_i) = P_i + P_i + (-2P_i) - 3P_\infty$. Since $h_Q \neq h_{\pm P}$, one has that $P_i \pm Q_j \neq P_\infty$ for $i,j \in \{1,2,3\}$. Then $r_{ij}(x_{P_i}, x_{Q_j}, y_{P_i}, y_{Q_j}, x, y) = 0$, the equation of the line through P_i and Q_j, is of the form given in line 4 and $\operatorname{div}(r_{ij}) = P_i + Q_j + (-(P_i+Q_j)) - 3P_\infty$. Let T and R be as in lines 7 and 8 respectively. For $i \in \{1,2,3\}$, one has that $y_{P_i} = \alpha_1 x_{P_i} + \alpha_0$ and $y_{Q_i} = \beta_1 x_{Q_i} + \beta_0$ whence the correctness of lines $9-12$. Moreover, T, R are symmetric polynomials in the variables $x_{P_1}, x_{P_2}, x_{P_3}$, and R is a symmetric polynomial in the variables $x_{Q_1}, x_{Q_2}, x_{Q_3}$. Hence they can be written as polynomial functions of e_1, e_2, e_3 and s_1, s_2, s_3. Correctness of lines 15-20 follows from Lemma 1. Correctness of line 21 follows from observing that

$$\operatorname{div}(T) = \sum_{i=1}^{3} P_i + \sum_{i=1}^{3} P_i + \sum_{i=1}^{3}(-2P_i) - 9P_\infty = 2\operatorname{div}(h_P) + \operatorname{div}(h_{-2P}) = \operatorname{div}(h_P^2 \cdot h_{-2P}),$$

hence $T = h_P^2 \cdot h_{-2P} \mod y^2 - f(x)$ up to multiplication by a nonzero constant. Finally

$$\operatorname{div}(R) = 3\sum_{i=1}^{3} P_i + 3\sum_{j=1}^{3} Q_j + \sum_{1 \leq i,j \leq 3} (-(P_i + Q_j)) - 27P_\infty = \operatorname{div}(h_P^3 h_Q^3 S_{P,Q}(x,-y)),$$

hence $R = h_P^3 h_Q^3 S_{P,Q}(x,-y) \mod y^2 - f(x)$ up to multiplication by a nonzero constant, hence correctness of line 22 follows. To conclude, one can directly check that the formulas computed in this way hold also in the case when $h_P = y$ or $h_Q = h_{\pm P}$. $\square$

Procedure 2. Procedure to write formulas for the coefficients of h_{3P} in terms of those of h_P.

---

for $i \in \{1,2,3\}$ $\triangleright$ doubling formulas for P_i and $\ell_i = 0$ line through P_i, $2P_i$

$\quad\triangleright$ x_{2P_i} written as a rational function in the variables x_{P_i}, y_{P_i}

$\quad x_{2P_i}(x_{P_i}, y_{P_i}) \leftarrow (f'(x_{P_i})/2y_{P_i})^2 - 2x_{P_i}$

$\quad\triangleright$ y_{2P_i} written as a rational function in the variables x_{P_i}, y_{P_i}

$\quad y_{2P_i}(x_{P_i}, y_{P_i}) \leftarrow (f'(x_{P_i})/2y_{P_i})(x_{P_i} - x_{2P_i}) - y_{P_i}$

$\quad\triangleright$ ℓ_i written as a rational function in the variables x_{P_i}, y_{P_i}, x, y

$\quad \ell_i(x_{P_i}, y_{P_i}, x, y) \leftarrow (y_{2P_i} - y_{P_i})x + (x_{P_i} - x_{2P_i})y + ((x_{2P_i} - x_{P_i})y_{P_i} + (y_{P_i} - y_{2P_i})x_{P_i})$

end for

$\quad L(x_{P_1}, x_{P_2}, x_{P_3}, y_{P_1}, y_{P_2}, y_{P_3}, x, y) \leftarrow \prod_{i=1}^{3} \ell_i$

for $i \in \{1,2,3\}$

$\quad$ replace y_{P_i} with $(\alpha_1 x_{P_i} + \alpha_0)$ in L

end for

write $L(x_{P_1}, x_{P_2}, x_{P_3})$ via the elementary symmetric polynomials e_1, e_2, e_3
  $E_1 \leftarrow \alpha_1^2$, $E_2 \leftarrow A - 2\alpha_0\alpha_1$, $E_3 \leftarrow \alpha_0^2 - B$
for $i \in \{1, 2, 3\}$
  replace e_i with E_i in L
end for
Recover h_{3P} using the formulas for h_{2P} found with Procedure 1, together with the equality (up to multiplication by a nonzero constant):

$$(h_{3P}) = L(x, -y)/(h_{-P}h_{-2P}) \mod y^2 - f(x).$$

Theorem 4. *Procedure 2 is correct.*

We omit the proof of Theorem 4, since it is analogous to the proof of correctness for Procedure 1.

We now want to compute h_{P+Q} from H_{P+Q} and $S_{P,Q}$. A straightforward way of doing this is computing the coefficients of h_{P+Q} from those of H_{P+Q} up to sign via the relations $w_2 = -\gamma_1^2$, $w_1 = A - 2\gamma_0\gamma_1$, $w_0 = B - \gamma_0^2$. One can then distinguish $h_{P+Q} = y - (\gamma_0 + \gamma_1 x)$ and $h_{-P-Q} = y + (\gamma_0 + \gamma_1 x)$, since $H_{P+Q} \mid (S_{P,Q})_1 + (\gamma_0 + \gamma_1 x)(S_{P,Q})_2$. This however requires extracting a square root. The next proposition allows us to compute h_{P+Q} from H_{P+Q} and $S_{P,Q}$ more efficiently, by solving a simple linear system.

Proposition 5. *Suppose that $P+Q \notin E[3](\mathbb{F}_q)$, that Q is not a Frobenius conjugate of $-P$ or $-2P$, and that P is not a Frobenius conjugate of $-2Q$. Write $H_{P+Q} = x^3 + w_2x^2 + w_1x + w_0$ and $h_{P+Q} = y - (\gamma_1 x + \gamma_0)$ with $\gamma_1, \gamma_0, w_2, w_1, w_0 \in \mathbb{F}_q$. Then (γ_1, γ_0) is the unique solution of the linear system whose augmented matrix is*

$$L(H_{P+Q}, S_{P,Q}) = \begin{pmatrix} w_0(w_2 - b_2) & (b_0 - w_0) & w_0a_3 - a_4w_2w_0 - a_0 \\ w_0(w_1 - b_1) & (b_0w_2 - w_0b_2) & w_0a_2 - a_4w_1w_0 - a_0w_2 \\ w_0(w_0 - b_0) & (b_0w_1 - b_1w_0) & w_0a_1 - a_4w_0^2 - a_0w_1 \end{pmatrix}.$$

Proof. Using the fact that $H_{P+Q} | (S_{P,Q})_1 + (\gamma_1 x + \gamma_0)(S_{P,Q})_2$, a simple calculation shows that (γ_1, γ_0) is a solution of the linear system with augmented matrix $L(H_{P+Q}, S_{P,Q})$. Let us prove that the solution is unique. Let (t_1, t_0) be a solution of the linear system with augmented matrix $L(H_{P+Q}, S_{P,Q})$ and let $(x_0, y_0) \in T_3$ be one of the Frobenius conjugates of $P + Q$. Notice that, since $P + Q \notin E[3](\mathbb{F}_q)$, the three Frobenius conjugates are distinct. By construction, $(S_{P,Q})_1(x_0) + (t_1x_0 + t_0)(S_{P,Q})_2(x_0) = 0$. We claim that $(S_{P,Q})_2(x_0) \neq 0$. In fact, if $(S_{P,Q})_2(x_0) = 0$, then $(S_{P,Q})_2 = H_{P+Q}$ and $H_{P+Q} \mid (S_{P,Q})_1$. In particular,

$$0 \leq \operatorname{div}(S_{P,Q}) - \operatorname{div}(H_{P+Q}) = \sum_{0 \leq i,j \leq 2 i \neq j} \varphi^i(P) + \varphi^j(Q) - \sum_{i=0}^{2} \varphi^i(-P - Q),$$

hence $-P - Q = \varphi^i(P) + \varphi^j(Q)$ for some i, j distinct. If $i, j \neq 0$, then $-\varphi^k(P) = P + \varphi^i(P) = -Q - \varphi^j(Q) = \varphi^h(Q)$ for some h, k, hence P and $-Q$ are Frobenius conjugates. Similarly, Q and $-2P$ are Frobenius conjugates if $i = 0$ and $j \neq 0$, and P and $-2Q$ are Frobenius conjugates if $i = 0$ and $j \neq 0$. This concludes the proof of the claim. Since $(S_{P,Q})_2(x_0) \neq 0$, then $y_0 = t_1x_0 + t_0$. Hence the line of equation $y - (t_1x + t_0)$ has three points in common with the line of equation h_{P+Q}. This implies that $t_1 = \gamma_1$ and $t_0 = \gamma_0$. $\square$

Example 6. Let $q = 1021$ and $\mathbb{F}_{q^3} = \mathbb{F}_q[\zeta]/(\zeta^3 - 5)$. Let E be the elliptic curve over $\mathbb{F}_q$ of equation $y^2 = x^3 + 230x + 191$. Let $P = (782\zeta^2 + 802\zeta + 45, 979\zeta^2 + 299\zeta + 133)$, $Q = (466\zeta^2 + 528\zeta + 514, 742\zeta^2 + 1016\zeta + 704) \in T_3$, with $h_P = y - (987x + 642)$, $h_Q = y - (729x + 705)$. Using the formulas in the appendix, we can compute:

$$h_{2P} = y - (1000x + 280), \quad h_{3P} = y - (646x + 693),$$

$$S_{P,Q} = (823x^4 + 948x^3 + 709x^2 + 530x + 741) + y(x^3 + +782x^2 + 636x + 100).$$

The matrix from Proposition 5 is:

$$L(H_{P+Q}, S_{P,Q}) = \begin{pmatrix} 809 & 123 & 843 \\ 568 & 823 & 755 \\ 787 & 382 & 388 \end{pmatrix}.$$

Before we compute L, we compute $H_{P+Q} = x^3 + 880x^2 + 123x + 998$ (in the next section we discuss how to compute H_{P+Q}). Solving the system associated to L we find $h_{P+Q} = y - (65x + 260)$.

2 Scalar multiplication in T_3 using compressed coordinates

Throughout this section we assume that $T_3 = \langle P \rangle$ is cyclic of order p, where p is a prime of cryptographic size. Hence $\varphi(P) = sP$, with $s = (q-1)/(2+q-|E(\mathbb{F}_q)|) \mod p$, (see [1, Section 15.3.1]). Let m be an integer modulo p. In this section we develop an efficient algorithm to compute h_{mP} given m and h_P. In order to do this, in Subsection 2.1 we give a subalgorithm that we use within the main algorithm, as well as a lemma which helps us deal with special cases. In Subsection 2.2 we present a Montgomery-ladder-style algorithm that computes h_{mP} from m and h_P. Finally, in Subsection 2.3 we apply the usual Frobenius endomorphism strategy to speed up our algorithm from Section 2.2. This gives our main algorithm to compute scalar multiplication in T_3 using compressed coordinates.

2.1 Subalgorithm and special cases

Throughout this subsection m is an integer $0 < m < p$. Because of the doubling formulas in the Appendix, we may assume that m is odd.

Notation 7. Let m_1, m_2, n_1, n_2 be integers such that $m_1+m_2 = n_1+n_2 = m$. For $i \in \{0, 1, 2\}$, let $h_i = h_{m_1P+\varphi^i(m_2P)}$, $H_i = H_{m_1P+\varphi^i(m_2P)}$, $k_i = h_{n_1P+\varphi^i(n_2P)}$, $K_i = K_{n_1P+\varphi^i(n_2P)}$.

Let m_1, m_2, n_1, n_2 be positive integers such that $m_1 + m_2 = n_1 + n_2 = m$ and suppose that we are given $h_{m_1P}, h_{m_2P}, h_{n_1P}, h_{n_2P}$. The subalgorithm computes h_{mP} by applying the following strategy: Via the formulas found with Procedure 1, one can compute

$$S_1 := S_{m_1P,m_2P} = S_{1,1} + yS_{1,2}$$

from h_{m_1P}, h_{m_2P} and

$$S_2 := S_{n_1P,n_2P} = S_{2,1} + yS_{2,2}$$

from h_{n_1P}, h_{n_2P}. Up to multiplying by a nonzero constant, $S_1 = \prod_{i=0}^{2} h_i \mod y^2 - f(x)$ and $S_2 = \prod_{i=0}^{2} k_i \mod y^2 - f(x)$, hence S_1, S_2 share the factor $h_0 = k_0 = h_{mP}$. By Lemma 1

$$H_{mP} | G := \gcd\,(fS_{1,2}^2 - S_{1,1}^2, fS_{2,2}^2 - S_{1,2}^2).$$

Moreover, if $m_1P + \varphi(m_2P)$ and $m_1P + \varphi(m_2P)$ are not Frobenius conjugates of $\pm(n_1P + \varphi(n_2P))$ or $\pm(n_1P + \varphi^2(n_2P))$, that is if $h_1, h_2 \notin \{k_1(x,y), k_2(x,y), -k_1(x,-y), -k_2(x,-y)\}$, then $G = H_{mP}$. In this case, one can compute h_{mP} from G and S_1 (or from G and S_2) by solving the linear system of Proposition 5, provided that the assumptions of the proposition are satisfied.

We now give the subalgorithm and we prove its correctness.

Subalgorithm 1.

Input: The polynomials $h_{m_1P}, h_{m_2P}, h_{n_1P}, h_{n_2P}$, such that $h_1, h_2 \notin \{k_1, k_2\}$.
Output : $h_{mP} = y - (\gamma_1 x + \gamma_0)$.

---

if $h_{m_1P} = h_{m_2P}$ **then return** h_{-m_1P} **endif**
if $h_{n_1P} = h_{n_2P}$ **then return** h_{-n_1P} **endif**
compute $S_1 = S_{m_1P,m_2P}$ from h_{m_1P}, h_{m_2P} ▷ formulas (1) in the appendix
compute $S_2 = S_{n_1P,n_2P}$ from h_{n_1P}, h_{n_2P}
if $h_{m_1P}(x,y) = -h_{m_2P}(x,-y)$ **then**
  $W \leftarrow \text{monic}(S_1)$
  $L \leftarrow L(W, S_2)$ ▷ see Proposition 5
  compute $h = y - (\gamma_1 x + \gamma_0)$ by solving the linear system associated to L
  return h
end if
if $h_{n_1P}(x,y) = -h_{n_2P}(x,-y)$ **then**
  $W \leftarrow \text{monic}(S_2)$
  $L \leftarrow L(W, S_1)$ ▷ see Proposition 5
  compute $h = y - (\gamma_1 x + \gamma_0)$ by solving the linear system associated to L
  return h
end if
$G \leftarrow \gcd(fS_{1,2}^2 - S_{1,1}^2, fS_{2,2}^2 - S_{2,1}^2)$
decompose G in irreducible factors in $\mathbb{F}_q[x]$
$W_1, \cdots W_s \leftarrow$ monic distinct irreducible factors of G of degree 3
for $j \in \{1, \cdots s\}$ **do**
  $W \leftarrow W_j$
  if $W \neq S_{1,2}$ **then**
    $L \leftarrow L(W, S_1)$ ▷ see Proposition 5
    compute $h = y - (\gamma_1 x + \gamma_0)$ by solving the linear system associated to L
    if $W | (\gamma_1 x + \gamma_0)S_{2,2} + S_{2,1}$ **then** return h
  end if
  else ▷ $W = S_{1,2}$
    $L \leftarrow L(W, S_2)$ ▷ see Proposition 5
    compute $h = y - (\gamma_1 x + \gamma_0)$ by solving the linear system associated to L

```
30:       return h
31:    end if
32: end for
```

Theorem 8. *Subalgorithm 1 is correct.*

To prove the theorem we use the following.

Remark 9. Since T_3 has prime order $p > 3$, then $T_3 \cap E[3](\mathbb{F}_q) = \{P_\infty\}$. Hence H_Q is irreducible over $\mathbb{F}_q$ for every $Q \in T_3 \backslash \{P_\infty\}$, in particular H_{mP} is irreducible over $\mathbb{F}_q[x]$ for every $0 < m < p$. Moreover, $h_{mP} \neq h_{-mP}$, since, if this were the case, then $mP + \varphi^i(mP) = P_\infty$.

Proof of Theorem 8. If $h_{m_1P} = h_{m_2P}$ as in line 1 of the subalgorithm, then $m_2P = \varphi^i(m_1P)$ for some $i \in \{0, 1, 2\}$. Since we assume that m is odd, then $m_1 \neq m_2$ and $m_1 + m_2 = m < p$, hence $i \neq 0$. Therefore $mP = (m_1+m_2)P = m_1(1+\varphi^i)(P) = -m_1\varphi^j(P)$ where $\{i, j\} = \{1, 2\}$, and $i \neq j$. It follows that $h_{mP} = h_{-m_1P}$ and line 1 is correct. The same argument shows that, if $h_{n_1P} = h_{n_2P}$ as in line 2 of the subalgorithm, then $h_{mP} = h_{-n_1P}$, and line 2 is correct.

Correctness of lines $3, 4$ follows from Theorem 3.

Up to multiplication by a nonzero constant, $S_1 = h_{mP}h_1h_2$ and $S_2 = h_{mP}k_1k_2 \mod y^2 - f(x)$. Moreover, by Lemma 1, $fS_{1,2}^2 - S_{1,1}^2 = H_0H_1H_2$ and $fS_{2,2}^2 - S_{2,1}^2 = H_0K_1K_2$ up to multiplication by a nonzero constant. Suppose first that $h_{m_1P} = h_{-m_2P}$ as in line 5. Then $S_1 = h_{mP}(h_{-mP}) = H_{mP} \mod y^2 - f(x)$ (up to multiplication by a nonzero constant). In addition, if $h_{m_1P} = h_{-m_2P}$, then $h_{n_1P} \neq h_{-n_2P}$. In fact, if $h_{n_1P} = h_{-n_2P}$, then $S_2 = h_{mP}h_{-mP} = H_{mP} = S_1 \mod y^2 - f(x)$ (up to multiplication by a nonzero constant), which is not possible since we are supposing $h_1, h_2 \notin \{k_1, k_2\}$. The inequality $h_{n_1P} \neq h_{-n_2P}$ implies $S_{2,2} \neq 0$ by Lemma 1. Moreover, by Remark 9, H_{mP} is irreducible over $\mathbb{F}_q[x]$. So, in order to apply Proposition 5 with $W = \text{monic}(S_1)$ and S_2, it remains to prove that $H_{mP} \neq S_{2,2}$. Suppose this is not the case. Then $k_i = h_{-mP}$ for some $i \in \{0, 1, 2\}$. Since $h_{mP} \neq h_{-mP}$ by Remark 9, we have that $i \in \{1, 2\}$ and $k_i = h_{-mP} = h_1$, which is not possible because $h_1, h_2 \notin \{k_1, k_2\}$ by assumption. Hence one can apply Proposition 5 to $W = H_{mP} = \text{monic}(S_1)$ and S_2, and correctness of lines $5-10$ follows. The proof of correctness of lines $11-16$ is analogous to that for lines $5 - 10$.

From now on, we may assume that $h_{m_1P} \neq h_{-m_2P}$ and $h_{n_1P} \neq h_{-n_2P}$, which imply $S_{1,2}, S_{2,2} \neq 0$ by Lemma 1. Let $1 \leq s \leq 3$, $W_1, \ldots, W_s$ the monic distinct irreducible factors of degree 3 over $\mathbb{F}_q[x]$ of $G = \gcd(fS_{1,2}^2 - S_{1,1}^2, fS_{2,2}^2 - S_{2,1}^2)$. By Remark 9, $H_0 \in \{W_1, \cdots, W_s\}$. Moreover, for $W \in \{W_1, \ldots, W_s\}$, one has that $W = H_j$ for some $j \in \{0, 1, 2\}$. Then, if $W \neq S_{1,2}$, one recovers $h = h_j$ from W and S_1 by solving the linear system of Proposition 5 (lines 22-24 of the subalgorithm).

We now consider line 25. If $h = h_0 = h_{mP}$, one has that $W|(\gamma_1 x + \gamma_0)S_{2,2} + S_{2,1}$. Else, $h \neq k_s$ for all $s \in \{0, 1, 2\}$, as $h_1, h_2 \notin \{k_1, k_2\}$ by hypothesis. So $W \nmid (\gamma_1 x + \gamma_0)S_{2,2} + S_{2,1}$ by Proposition 5, and line 25 is correct.

Finally, suppose that $W = S_{1,2}$ as in line 26. If $W \neq H_0$, one has that there exists $r \in \{1, 2\}$ such that $h_j = -(h_r(x, -y))$. Moreover, there exists $s \in \{1, 2\}$ such that $h_j = -(k_s(x, -y))$, since $W|G$ and $h_1, h_2 \notin \{k_1, k_2\}$. Then $h_r = k_s$ with r, $s \in \{1, 2\}$, that is not possible as $h_1, h_2 \notin \{k_1, k_2\}$. Hence $W = H_0$ and there exists $r \in \{1, 2\}$ such that $h_{mP} \neq h_{rP} = h_{-mP}$, from which $k_s \neq h_{-mP}$ for all $s \in \{0, 1, 2\}$, since $h_1, h_2 \notin \{k_1, k_2\}$. So $W \neq S_{2,2}$, one recovers $h = h_{mP}$ from W and S_2 by solving the linear system of Proposition 5, and lines 26-30 are correct. $\square$

We use the subalgorithm at each step of our Montgomery-ladder-style algorithm. We have two different types of input lines: The first is used in the general case, and the second for special cases.

(a) **Input lines of type (a)**: The subalgorithm computes h_{mP} from h_P, $h_{(m-1)P}$, $h_{\frac{m-1}{2}P}$ and $h_{\frac{m+1}{2}P}$. The subalgorithm does not apply to a set M of special values for m.

(b) **Input lines of type (b)**: Let $R = \{(-3,-7),(-3,5),(3,-5),(3,7)\}$, $(r_1,r_2) \in R$. The subalgorithm computes h_{mP} for h_{r_iP}, $h_{(m-r_i)P}$ for $i \in \{1,2\}$. The subalgorithm does not apply to a set $M_{(r_1,r_2)}$ of special values for m.

In the next lemma we describe the sets M and $M_{(r_1,r_2)}$. Moreover, we show that $M \cap (\bigcup_{(r_1,r_2)\in R} M_{(r_1,r_2)}) = \emptyset$. Therefore, one can compute h_{mP} using the subalgorithm with input of type (a) if $m \notin M$ and with input of type (b) if $m \in M$.

Lemma 10. *In the setting established above, one has the following:*

1. $h_{P+(m-1)\varphi^i(P)} = h_{\frac{m-1}{2}P+\frac{m+1}{2}\varphi^j(P)}$ *for some* i, $j \in \{1,2\}$ *if and only if* $m \in M$*, where*

$$M = \left\{ \frac{\pm 3}{2s+1}, \frac{s-4}{3s}, \frac{4s-1}{2s+1}, \frac{s+5}{3(s+1)}, \frac{4s+5}{2s+1} \mod p \right\}.$$

Hence Subalgorithm 1 correctly computes h_{mP} *from* h_P, $h_{(m-1)P}$, $h_{\frac{m-1}{2}P}$ *and* $h_{\frac{m+1}{2}P}$ *if* $m \notin M$.

2. *Let* $R = \{(-3,-7),(3,7),(-3,5),(3,-5)\}$, $(r_1,r_2) \in R$. *Then* $h_{r_1P+(m-r_1)\varphi^i(P)} = h_{r_2P+(m-r_2)\varphi^j(P)}$ *for some* $i,j \in \{1,2\}$ *if and only if* $m \in M_{(r_1,r_2)}$*, where*

   - $M_{(3,7)} = \left\{ \frac{17s+4}{2s+1}, \frac{-4s-17}{s-1}, \frac{10s+11}{2s+1}, \frac{10s-1}{2s+1}, \frac{4s-13}{-s-2}, \frac{17s+13}{2s+1} \mod (p) \right\}$,
   - $M_{(-3,-7)} = \{-m \mod (p) \mid m \in M_{(3,7)}\}$,
   - $M_{(-3,5)} = \left\{ \frac{7s+8}{2s+1}, \frac{-8s-7}{s-1}, \frac{2s+13}{2s+1}, \frac{2s-11}{2s+1}, \frac{8s+1}{-s-2}, \frac{7s-1}{2s+1} \mod (p) \right\}$,
   - $M_{(3,-5)} = \{-m \mod (p) \mid m \in M_{(-3,5)}\}$.

Fix $(r_1,r_2) \in R$. *Subalgorithm 1 correctly computes* h_{mP} *from* $h_{r_1P}, h_{r_2P}, h_{(m-r_1)P}, h_{(m-r_2)P}$ *if* $m \notin M_{(r_1,r_2)}$.

3. *One has that* $M \cap (\bigcup_{(r_1,r_2)\in R} M_{(r_1,r_2)}) = \emptyset$. *Hence, if Subalgorithm 1 cannot compute* h_{mP} *with input of type (a), it can compute it with input of type (b).*

Proof. By Theorem 8, and following Notation 7, we have that Subalgorithm 1 correctly computes h_{mP} from the input lines $h_{m_1P} = h_P$, $h_{m_2P} = h_{(m-1)P}$, $h_{n_1P} = h_{\frac{m-1}{2}P}$ and $h_{n_2P} = h_{\frac{m+1}{2}P}$ if $h_1, h_2 \notin \{k_1,k_2\}$, that is, if $h_{P+(m-1)\varphi^i(P)} \neq h_{\frac{m-1}{2}P+\frac{m+1}{2}\varphi^j(P)}$ for all $i,j \in \{1,2\}$. We have that

$$h_{P+(m-1)\varphi^i(P)} = h_{\frac{m-1}{2}P+\frac{m+1}{2}\varphi^j(P)} \text{ for some } i,j \in \{1,2\}$$

if and only if

$$P + (m-1)\varphi^i(P) = \varphi^\ell\left(\frac{m-1}{2}P + \frac{m+1}{2}\varphi^j(P)\right) \text{ for some } i,j \in \{1,2\}, \ell \in \{0,1,2\}.$$

Since $\varphi(P) = sP$ and P is of order p, the last equality is equivalent to

$$1 + (m-1)s^i = s^\ell \left(\frac{m-1}{2} + \frac{m+1}{2} s^j \right) \mod p \text{ for some } i, j \in \{1,2\}, \ell \in \{0,1,2\}. \quad (3)$$

Moreover, $P \in T_3$, so $P + \varphi(P) + \varphi^2(P) = P_\infty$, hence

$$1 + s + s^2 = 0 \mod p, \quad (4)$$

since $\varphi(P) = sP$ and P has order p. From (4) one directly computes that (3) is equivalent to the statement that $m \in M$. Notice that all denominators in M are nonzero modulo p, since (4) holds and $p \neq 2, 3$. We have then proved part 1 of the lemma.

The proof for part 2 is analogous to that of part 1.

We now prove part 3. Suppose that $M \cap (\bigcup_{(r_1,r_2)\in R} M_{(r_1,r_2)}) \neq \emptyset$. One can check by direct computation that $as = b \mod p$ or $as = -b \mod p$ for some a and b such that $0 < a, b \leq 60$ and $a \neq b$. If $as = b \mod p$, then from (4) one obtains that $a^2 + ab + b^2 = 0 \mod p$, which is not possible since $0 < a^2 + ab + b^2 \ll p$. The case $as = -b \mod p$ can be treated similarly. $\square$

Remark 11. Lemma 10 is no longer true for small values of p. Consider e.g. the elliptic curve $y^2 = x^3 + 5x + 4$ over $\mathbb{F}_7$, with $p = 31$ and $s = 25$. We have $M \cap M_{(-3,-7)} = \{7, 11, 13\} \cap \{13, 15\} = \{13\} \neq \emptyset$.

Example 12. Let $q = 1021$ and $\mathbb{F}_{q^3} = \mathbb{F}_q[\zeta]/(\zeta^3 - 5)$. We consider the same E and P as in Example 6, i.e., we let E be the elliptic curve over $\mathbb{F}_q$ of equation $y^2 = x^3 + 230x + 191$ and let $P = (782\zeta^2 + 802\zeta + 45, 979\zeta^2 + 299\zeta + 133)$. Then $p = 1021381$, $s = 161217$, $M = \{161219, 322435, 322437, 465965\}$.

We show how to compute h_{5P} using Subalgorithm 1 with input of type (a). In Example 6 we computed h_{2P} and h_{3P}. Using formulas (1) and (2) in the appendix, we compute $h_{4P} = y-(698x+155)$ from h_{2P}, $S_1 = (524x^4+131x^3+826x^2+631x+160)+y(x^3+243x^2+651x+776)$ from h_P and h_{4P}, $S_2 = (331x^4+653x^3+169x^2+259x+536)+y(x^3+570x^2+680x+578)$ from h_{2P} and h_{3P}. Then we compute $G = \gcd(fS_{1,2}^2 - S_{1,1}^2, fS_{2,2}^2 - S_{2,1}^2) = x^3 + 455x^2 + 81x + 68$, hence $G = H_{5P}$, and $H_{5P} \neq S_{1,2}$. So we obtain $h_{5P} = y - (736x + 804)$ from G and S_1 as in line 24 of Subalgorithm 1.

Similarly one can compute $h_{7P} = y - (112x + 43)$ from h_P, h_{6P}, h_{3P}, h_{4P}.

The next two examples illustrate special cases of Subalgorithm 1.

Example 13. Let E and P be as in the previous example and let $m = 337887$. One can check that

$$P + (m-1)\varphi^2(P) = -\frac{m-1}{2}\varphi^2(P) - \frac{m+1}{2}\varphi(P).$$

If we try to compute h_{mP} using Subalgorithm 1 with input of type (a), we first compute $G = x^6 + 778x^5 + 86x^4 + 778x^3 + 599x^2 + 494x + 658$, which splits over $\mathbb{F}_q$ into two irreducible factors of degree 3, namely $W_1 = x^3 + 11x^2 + 843x + 540$ and $W_2 = x^3 + 767x^2 + 1016x + 5$. From W_1 we recover $h_1 = y - (166x + 727) = 0$ which is the line through $P + (m-1)\varphi^2(P)$, from W_2 we recover $h_2 = y - (423x + 57) = 0$ which is the line through mP. By checking the condition of line 25 of the subalgorithm, we are able to decide that $h_{mP} = h_2$.

Example 14. Let $q = 1021$ and $\mathbb{F}_{q^3} = \mathbb{F}_q[\zeta]/(\zeta^3 - 5)$. Let E be the elliptic curve of equation $y^2 = x^3 + 71x + 529$ defined over $\mathbb{F}_q$. Then T_3 is generated by $P = (853\zeta^2 + 995\zeta + 244, 178\zeta^2 + 927\zeta + 959)$, which has prime order $p = 1009741$. Moreover $s = 325960$ and $M_{(3,-5)} = \{32671, 391027\}$. Let $m = 65339$. One can check that $mP = -3P - (m-3)\varphi^2(P)$. We compute h_{mP} using Subalgorithm 1 with input of type (b), with $(r_1, r_2) = (3, -5)$. We obtain $G = S_{1,2}$, then we can compute $h_{mP} = y - (566x + 37)$ from G and S_2.

2.2 A first algorithm for scalar multiplication

We now present our Montgomery-ladder style algorithm for scalar multiplication in its basic form.

Notation 15. Let m be an integer with $0 < m < p$. Let $m = \sum_{i=0}^{\ell-1} m_i 2^i$ be the binary representation of m, with $m_i \in \{0, 1\}$ for all i, $\ell = \lceil \log_2 m \rceil$ and $m_{\ell-1} = 1$. Let

$$k_i = \sum_{j=i}^{\ell-1} m_j 2^{j-i}$$

for $i \in \{0, \cdots, \ell - 1\}$. Notice that $k_0 = m$. Finally, let

$$M = \left\{ \frac{\pm 3}{2s+1}, \frac{s-4}{3s}, \frac{4s-1}{2s+1}, \frac{s+5}{3(s+1)}, \frac{4s+5}{2s+1} \mod p \right\}$$

and define $\mathcal{M} = M \cap (2\mathbb{Z} + 1)$.

General strategy of the algorithm. Our algorithm takes h_P and m as input, and it returns h_{mP} as output. It adopts the classical double-and-add strategy for scalar multiplication: It computes

$$u_i = h_{k_i P} \text{ and } v_i = h_{(k_i+1)P}$$

for decreasing values of i. At the end of the cycle, it outputs $u_0 = h_{mP}$. In order to compute the polynomials u_i and v_i, the algorithm uses the doubling formulas of the appendix and Subalgorithm 1 with input the polynomials that it has computed in the previous steps.

The proposition below gives recursive definitions for u_i and v_i Our algorithm applies this proposition to construct the polynomials u_i and v_i at each step i.

Notation 16. Write $\mathrm{Subalg}(h_1, h_2, h_3, h_4)$, for the output of Subalgorithm 1 with input h_1, h_2, h_3, h_4. For any $Q \in T_3$, let $D(h_Q) = h_{2Q}$, where h_{2Q} is computed from the coefficients of h_Q via the doubling formulas from the appendix. Then $D^k(h_Q) = h_{2^k Q}$, where $h_{2^k Q}$ is computed from h_Q via iteration of the doubling formulas from the appendix.

Proposition 17. *For i from $i = \ell - 1$ down to $i = 0$, recursively define u_i and v_i as follows.*

- $u_{\ell-1} = h_P$, $v_{\ell-1} = h_{2P}$.
- $u_{\ell-2} = h_{2P}$ *and* $v_{\ell-2} = h_{3P}$ *if* $m_{\ell-2} = 0$,
  $u_{\ell-2} = h_{3P}$ *and* $v_{\ell-2} = h_{4P}$ *if* $m_{\ell-2} = 1$.
- *For* $0 \le i \le \ell - 3$*:*

– *(General case) if* $k_i, k_i+1 \notin \mathcal{M}$, *let*
$u_i = D(u_{i+1})$ *and* $v_i = \mathrm{Subalg}(h_P, D(u_{i+1}), u_{i+1}, v_{i+1})$ *if* $m_i = 0$,
$u_i = \mathrm{Subalg}(h_P, D(u_{i+1}), u_{i+1}, v_{i+1})$ *and* $v_i = D(v_{i+1})$ *if* $m_i = 1$.

– *(Special cases) if* k_i *or* $k_i + 1 \in \mathcal{M}$:

  ∗ *If* $m_i = 0$, *let*

$$u_i = D(u_{i+1}) \text{ and } v_i = \begin{cases} \mathrm{Subalg}(h_{3P}, D^2(u_{i+2}), h_{7P}, D^3(u_{i+3})) & \text{if} \quad m_{i+1} = m_{i+2} = 1, \\ \mathrm{Subalg}(h_{3P}, D^3(u_{i+3}), h_{-5P}, D^3(v_{i+3})) & \text{if} \quad m_{i+1} = 1, m_{i+2} = 0, \\ \mathrm{Subalg}(h_{-3P}, D^3(v_{i+3}), h_{5P}, D^3(u_{i+3})) & \text{if} \quad m_{i+1} = 0, m_{i+2} = 1, \\ \mathrm{Subalg}(h_{-3P}, D^2(v_{i+2}), h_{-7P}, D^3(v_{i+3})) & \text{if} \quad m_{i+1} = m_{i+2} = 0. \end{cases}$$

  ∗ *If* $m_i = 1$, *let*

$$v_i = D(v_{i+1}) \text{ and } u_i = \begin{cases} \mathrm{Subalg}(h_{3P}, D^2(u_{i+2}), h_{7P}, D^3(u_{i+3})) & \text{if} \quad m_{i+1} = m_{i+2} = 1, \\ \mathrm{Subalg}(h_{3P}, D^3(u_{i+3}), h_{-5P}, D^3(v_{i+3})) & \text{if} \quad m_{i+1} = 1, m_{i+2} = 0, \\ \mathrm{Subalg}(h_{-3P}, D^3(v_{i+3}), h_{5P}, D^3(u_{i+3})) & \text{if} \quad m_{i+1} = 0, m_{i+2} = 1, \\ \mathrm{Subalg}(h_{-3P}, D^2(v_{i+2}), h_{-7P}, D^3(v_{i+3})) & \text{if} \quad m_{i+1} = m_{i+2} = 0. \end{cases}$$

Then $u_i = h_{k_i P}$ *and* $v_i = h_{(k_i+1)P}$, *for all* $i \in \{0, \cdots, \ell - 1\}$.

Proof. We proceed by induction on i. The thesis is easily verified for $i = \ell - 1$ and $i = \ell - 2$. Hence let $0 \le i \le \ell - 3$ and assume that the thesis holds for $j \in \{i+1, \cdots, \ell-1\}$. Suppose first that $k_i, k_i + 1 \notin \mathcal{M}$ and that $m_i = 0$ (the proof for the case $m_i = 1$ is analogous). Then $k_i = 2(k_{i+1})$ and $u_i = D(u_{i+1}) = h_{2k_{i+1}P} = h_{k_i P}$ by induction. Moreover, by induction we get

$$\mathrm{Subalg}(h_P, D(u_{i+1}), u_{i+1}, v_{i+1}) = \mathrm{Subalg}(h_P, h_{2k_{i+1}P}, h_{k_{i+1}P}, h_{(k_{i+1}+1)P}) =$$

$$\mathrm{Subalg}\left(h_P, h_{k_i P}, h_{\frac{k_i}{2}P}, h_{\left(\frac{k_i}{2}+1\right)P}\right).$$

Since $k_i + 1 \notin \mathcal{M}$, Subalgorithm 1 with input of type (a) correctly outputs $v_i = h_{(k_i+1)P}$. Now suppose that k_i or $k_i + 1 \in \mathcal{M}$ and assume that $m_i = 0$, $m_{i+1} = m_{i+2} = 1$ (the proof for the other cases is analogous). If k_i or $k_i + 1 \in \mathcal{M}$, then $i < \ell - 3$, since $5, 7 \notin \mathcal{M}$. Hence we already have computed the polynomials of the three previous steps $i+1$, $i+2$, $i+3$. Since $m_i = 0$, we prove the thesis for u_i as in the general case. On the other hand, $k_i + 1 \in \mathcal{M}$ so we cannot define v_i using Subalgorithm 1 with input of type (a), as we did before. However $k_i + 1 = 3 + 4k_{i+2} = 7 + 8k_{i+3}$, so by induction we get

$$\mathrm{Subalg}(h_{3P}, D^2(u_{i+2}), h_{7P}, D^3(u_{i+3})) = \mathrm{Subalg}(h_{3P}, h_{4(k_{i+2})P}, h_{7P}, h_{8(k_{i+3})P}) =$$

$$\mathrm{Subalg}(h_{3P}, h_{(k_i-2)P}, h_{7P}, h_{(k_i-6)P}).$$

Moreover, since $k_i + 1 \in \mathcal{M}$, then $k_i + 1 \notin M_{(3,7)}$ by Lemma 10, hence Subalgorithm 1 with input of type (b) correctly outputs $v_i = h_{(k_i+1)P}$. □

Remark 18. If $k_i, k_i + 1 \notin \mathcal{M}$, at step i one needs only the polynomials computed in the previous step in order to compute the polynomials u_i, v_i. If k_i or $k_i + 1 \in \mathcal{M}$ one needs the polynomials computed in the steps $i+2$ and $i+3$ in order to compute them. Therefore:

- In our algorithm, the last three pairs of polynomials that have been computed are stored in a vector L, which is updated at each step of the cycle.

- The algorithm looks for the i's for which k_i or $k_i+1 \in \mathcal{M}$ at the start: For each $i \in \{0,\cdots,\ell-2\}$, it computes k_i and k_i+1, and it adds i to the list S if k_i or $k_i+1 \in \mathcal{M}$. Hence, at each step i, we know whether we have to call Subalgorithm 1 with input of type (a) or of type (b), by simply checking if $i \in S$.

Algorithm 1 (Scalar multiplication in T_3)**.**

Input : h_P, m an integer modulo p.
Output : h_{mP}.

$m \leftarrow \sum_{i=0}^{\ell-1} m_i 2^i$ binary expansion of m
▷ collection of the special steps
$S \leftarrow \{i \in \{0,\cdots,\ell-2\} : k_i \leftarrow \sum_{j=i}^{\ell-1} m_j 2^{j-i} \in \mathcal{M}$ or $k_i+1 \in \mathcal{M}\}$
▷ step $i = \ell-1$
$u \leftarrow h_P$, $v \leftarrow h_{2P}$, $L \leftarrow [(u,v)]$ ▷ $L = [(u_{\ell-1}, v_{\ell-1})]$
if $\ell-1=0$ **then** return u **end if**
▷ step $i=\ell-2$
if $m_{\ell-2}=0$ **then** $u \leftarrow h_{2P}$, $v \leftarrow h_{3P}$ **else** $u \leftarrow h_{3P}$, $v \leftarrow h_{4P}$ **end if**
Append (u,v) to L ▷ $L = [(u_{\ell-1}, v_{\ell-1}), (u_{\ell-2}, v_{\ell-2})]$
if $\ell-2=0$ **then** return u **end if**
▷ cycle for: steps from $i=\ell-3$ to $i=0$
for i from $\ell-3$ down to 0 **do**
  ▷ special cases
  if $i \in S$ **then**
    if $m_{i+1}=1$ **then**
      if $m_{i+2}=1$ **then**
        $h_{exc} \leftarrow \text{Subalg}(h_{3P}, D^2(L[2][1]), h_{7P}, D^3(L[1][1]))$
      else ▷ $m_{i+1}=1$, $m_{i+2}=0$
        $h_{exc} \leftarrow \text{Subalg}(h_{3P}, D^3(L[1][1]), h_{-5P}, D^3(L[1][2]))$
      end if
    else ▷ $m_{i+1}=0$
      if $m_{i+2}=1$ **then**
        $h_{exc} \leftarrow \text{Subalg}(h_{-3P}, D^3(L[1][2]), h_{5P}, D^3(L[1][1]))$
      else ▷ $m_{i+1}=0$, $m_{i+2}=0$
        $h_{exc} \leftarrow \text{Subalg}(h_{-3P}, D^2(L[2][2]), h_{-7P}, D^3(L[1][2]))$
      end if
    end if
  if $|L|=3$ **then** remove $L[1]$ from L **end if** ▷ $L = [(u_{i+2}, v_{i+2}), (u_{i+1}, v_{i+1})]$
  ▷ computation of u, v at step i
  if $m_i=0$ **then**
    $u \leftarrow D(L[2][1])$
    if $i \in S$ **then**
      $v \leftarrow h_{exc}$
    else
      $v \leftarrow \text{Subalg}(h_P, D(L[2][1]), L[2][1], L[2][2])$
  else ▷ $m_i = 1$

```
31:     if  i ∈ S then
32:        u ← h_exc
33:     else
34:        u ← Subalg(h_P, D(L[2][1]), L[2][1], L[2][2])
35:     end if
36:     v ← D(L[2][2])
37:   end if
38:   Append (u, v) to L      ▷ L = [(u_{i+2}, v_{i+2}), (u_{i+1}, v_{i+1}), (u_i, v_i)]
39: end for
40: return L[3][1]
```

Theorem 19. *Algorithm 1 is correct.*

Proof. Correctness of lines $3-7$ is easy to check. Notice that, at the beginning of the cycle at line 8, the list L is $L = [(u_{\ell-1}, v_{\ell-1}), (u_{\ell-2}, v_{\ell-2})]$. Moreover, one has that $\ell - 3 \notin S$, since $5, 7 \notin \mathcal{M}$, so we do not need to check whether $\ell - 3 \in S$. Observe now that for each i from $i = \ell - 3$ down to $i = 0$, the list L at line 23 is $L = [(u_{i+2}, v_{i+2}), (u_{i+1}, v_{i+1})]$, while at line 38 the list is $L = [(u_{i+2}, v_{i+2}), (u_{i+1}, v_{i+1}), (u_i, v_i)]$. Hence correctness follows from Proposition 17. □

We now give an example of computation of a multiplication by m for which the algorithm runs into the special cases.

Example 20. Let $q = 1021$ and $\mathbb{F}_{q^3} = \mathbb{F}_q[\zeta]/(\zeta^3 - 5)$. Let E and P be as in Example 6 and Example and 12, i.e., let E be the elliptic curve over $\mathbb{F}_q$ of equation $y^2 = x^3 + 230x + 191$ and let $P = (782\zeta^2 + 802\zeta + 45, 979\zeta^2 + 299\zeta + 133)$. Let $m = 644875$, with binary representation

$$m = 2^{19} + 2^{16} + 2^{15} + 2^{14} + 2^{12} + 2^{10} + 2^9 + 2^8 + 2^3 + 2 + 1.$$

For i from 19 to 0 the pairs $(k_i, k_i + 1)$ are

$$(1,2), (2,3), (4,5), (9,10), (19,20), (39,40), (78,79), (157,158), (314,315), (629,630),$$

$$(1259,1260), (2519,2520), (5038,5039), (10076,10077), (20152,20153), (40304,40305),$$

$$(80609,80610), (161218,161219), (322437,322438), (644875,644876).$$

Hence the set of the special cases is $S = \{2, 1\}$ since $k_2 + 1 = 161219$, $k_1 = 322437 \in \mathcal{M}$. We compute $h_{mP} = y - (105x + 587)$ using Algorithm 1. At step $i = 2$ we compute $v = h_{exc}$ with $m_3 = 1$ and $m_4 = 0$ (line 14 of the algorithm). At step $i = 1$ we compute $u = h_{exc}$ with $m_2 = 0$ and $m_3 = 1$ (line 18 of the algorithm).

2.3 The optimized algorithm for scalar multiplication

In this subsection, we optimize the Montgomery-ladder style algorithm given in the previous subsection and give the conclusive algorithm to perform scalar multiplication in T_3 in optimal coordinates.

Remark 21. Let m be an integer modulo p. If $m > \frac{p-1}{2}$, one can reduce the computation of multiplication by m to the computation of multiplication by $m' = -m \mod p$, with $m' \leq \frac{p-1}{2}$. One does so by using the equality $h_{-P}(x, y) = -h_P(x, -y)$.

Frobenius reduction. We now discuss how the Frobenius endomorphism can be used to increase the efficiency of our Montgomery-ladder-style algorithm for scalar multiplication.

This strategy was first proposed by Koblitz in [7] for special elliptic curves and it has been applied to the group of $\mathbb{F}_{q^r}$-rational divisor classes of a hyperelliptic curve defined over $\mathbb{F}_q$ for $r > 1$, see [1, Section 15.1]. The idea is splitting the computation of multiplication by m in the computations of several multiplications by smaller scalars. Such computations can be done in parallel, to obtain a faster scalar multiplication algorithm (see [1, Section 15.1.2.d]). In trace-zero subgroups, such a strategy enjoys the benefit of the extra property of the Frobenius on the trace, so that the operation can be further sped up. Hence computation in T_n in the usual coordinates is faster than in the entire group, as shown in [1, Section 15.3], [2], [3], [10], [12], [16].

We now adapt this strategy to our scalar multiplication algorithm. Let m be an integer modulo p. One can write $m = m_0 + sm_1$, with $m_0, m_1 \in \mathcal{O}(q) = \mathcal{O}(\sqrt{p})$, see the discussion in [1, Section 15.3.2]. In order to compute h_{mP} given m and h_P, we call Algorithm 1 three times with input m_0, m_1 and $m_0 + m_1$ respectively, instead of calling Algorithm 1 once with input m. Notice that $m_0, m_1, m_0 + m_1 \in \mathcal{O}(\sqrt{p})$, while $m \in \mathcal{O}(p)$. Hence one reduces computation of the multiplication by m to the computation of at most three multiplications by integers of smaller size. Similarly to what we did in Algorithm 1, one needs to pay attention to the special cases where one cannot apply Subalgorithm 1.

Lemma 22. *Let m, m_0, m_1 be integers modulo p, with $m_0, m_1 \neq 0$. One has the following:*

1. *Subalgorithm 1 with input h_P, h_{mP}, $h_{(m+1)P}$, $h_{(s-1)P}$ correctly outputs $h_{(m+s)P}$ if $m \notin \mathcal{A}_1$, where*
$$\mathcal{A}_1 = \left\{-2, s, \frac{-3(1+s)}{2+s}, \frac{-3}{2+s}, \frac{s+2}{s-1}, \frac{-3}{2s+1} \mod p\right\}.$$

2. *Subalgorithm 1 with input h_{mP}, h_{-mP}, $h_{(m+s)P}$, $h_{-(m+1)P}$ correctly outputs $h_{m(1-s)P}$ if $m \notin \mathcal{A}_2$, where*
$$\mathcal{A}_2 = \left\{1, s, \frac{s+2}{s-1}, \frac{2s+1}{-3}, \frac{1-s}{3s} \mod p\right\}.$$

3. *Subalgorithm 1 with input h_{m_0P}, h_{m_1P}, $h_{(m_0+m_1)P}$, $h_{m_0(1-s)P}$ correctly outputs $h_{(m_0+sm_1)P}$ if $2m_0 + m_1 \neq 0 \mod p$ and $s \notin \mathcal{B}_1$, where*
$$\mathcal{B}_1 = \left\{ \left(\tfrac{3m_0+m_1}{m_1}\right)^{\pm 1}, \left(\tfrac{m_1-m_0}{2m_0+m_1}\right)^{\pm 1}, \tfrac{m_0+2m_1}{-(2m_0+m_1)}, \tfrac{3m_0+2m_1}{-(3m_0+m_1)}, \tfrac{2m_1}{-(3m_0+m_1)} \mod p : \right.$$
$$\left. 3m_0 + m_1, 2m_0 + m_1, m_1 - m_0 \neq 0 \mod p \right\}.$$

4. *Subalgorithm 1 with input h_{m_0P}, h_{m_1P}, $h_{(m_0+m_1)P}$, $h_{m_1(s-1)P}$ correctly outputs $h_{(m_0+sm_1)P}$ if $m_0 + 2m_1 \neq 0 \mod p$ and $s \notin \mathcal{B}_2$, where*
$$\mathcal{B}_2 = \left\{ \left(\tfrac{m_0+3m_1}{-(2m_0+3m_1)}\right)^{\pm 1}, \left(\tfrac{m_0-m_1}{m_0+2m_1}\right)^{\pm 1}, \tfrac{2m_0+3m_1}{-m_0}, \tfrac{m_0+3m_1}{-2m_0} \mod p : \right.$$
$$\left. m_0 + 3m_1, 2m_0 + 3m_1, m_0 + 2m_1, m_1 - m_0 \neq 0 \mod p \right\}.$$

5. *Let* $\mathrm{Poly} = \{t+1, t-1, t+2, t+3, 3t+1, t^2+1, t^2+t+1, t^2+4t+2, 2t^2+t+1,$

$t^2-t-1, 2t^2+4t+1, t^2+4t+1, t^2+2t+2, t^2+3t+1, t^2+t-1, 2t^2+2t+1,$

$t^2+3t+1, t^2-2t-1, t^2+2t-1, 2t^2+3t-1, 2t^2+3t+1\} \subseteq \mathbb{F}_p[t]$

and let $\mathcal{R}$ be the corresponding set of roots in $\mathbb{F}_p$:

$$\mathcal{R} = \{\alpha \in \mathbb{F}_p \mid f(\alpha) = 0 \text{ for some } f \in \mathrm{Poly}\}.$$

Then $s \in \mathcal{B}_1 \cap \mathcal{B}_2$ if and only if $m_0 = \alpha m_1$ for some $\alpha \in \mathcal{R}$.

Proof. Recall that Subalgorithm 1 requires the condition $h_1, h_2 \notin \{k_1, k_2\}$ for the input lines, where we follow Notation 7. The lemma then follows from Theorem 8 by direct computation (the proof is analogous to that of Lemma 10). □

Precomputation. In order to apply Frobenius reduction to scalar multiplication, we need to be able to deal with the special cases of Lemma 22. We chose to solve this problem by using Algorithm 1 to precompute the polynomials of the set

$$\mathcal{L} = \{h_{m(1-s)P} : m \in \mathcal{A}_1 \cup \mathcal{A}_2\} \cup \{h_{(s+\alpha)P} : \alpha \in \mathcal{R}\}. \tag{5}$$

In order to compute the polynomials of the form $h_{m(1-s)P}$, we first compute $h_{(s-1)P} \in \mathcal{L}$, then call Algorithm 1 with input $h_{(1-s)P}$ and m.

We are now ready to present our final algorithm for scalar multiplication in T_3. Recall that at the end of the cycle for in Algorithm 1, one has computed the pair $L[3] = (h_{mP}, h_{(m+1)P})$.

Notation 23. Write $Alg_1(h_P, m)$ for the pair $(h_{mP}, h_{(m+1)P})$, computed with a modified version of Algorithm 1 that outputs the entire pair $L[3]$.

Algorithm 2 (Scalar multiplication in T_3)**.**

Input : h_P, m an integer modulo p.
Output : h_{mP}.

$\mathcal{L} \leftarrow$ set 5 of precomputed lines
if $m > \frac{p-1}{2}$ **then** $\overline{m} \leftarrow -m \mod p$ **else** $\overline{m} \leftarrow m$ **end if**
$\overline{m} \leftarrow m_0 + s m_1$
if $m_0 = 0$ **then** $h \leftarrow Alg_1(h_P, m_1)[1]$
else if $m_1 = 0$ **then** $h \leftarrow Alg_1(h_P, m_0)[1]$
else ▷ $m_0, m_1 \neq 0$
  if $s \in \mathcal{B}_1 \cap \mathcal{B}_2$ **then** ▷ $\overline{m} = m_1(s+\alpha)$ for some $\alpha \in \mathcal{R}$
    $h \leftarrow Alg_1(h_{(s+\alpha)P}, m_1)[1]$ ▷ $h_{(s+\alpha)P} \in \mathcal{L}$
  else ▷ $s \notin \mathcal{B}_1 \cap \mathcal{B}_2$
    $h_{m_0P} \leftarrow Alg_1(h_P, m_0)[1]$
    $h_{m_1P} \leftarrow Alg_1(h_P, m_1)[1]$
    $h_{(m_0+m_1)P} \leftarrow Alg_1(h_P, m_0+m_1)[1]$
    if $s \notin \mathcal{B}_1$ and $2m_0+m_1 \neq 0 \mod p$ **then** ▷ Compute $h_{(m_0+sm_1)P}$ from $h_{m_0P}, h_{m_1P}, h_{(m_0+m_1)P}, h_{m_0(1-s)P}$
      if $m_0 \notin \mathcal{A}_1 \cup \mathcal{A}_2$ **then**

15 : $h_{(m_0+1)P} \leftarrow Alg_1(h_P, m_0)[2]$
16 : $h_{(m_0+s)P} \leftarrow \text{Subalg}(h_P, h_{m_0P}, h_{(m_0+1)P}, h_{(s-1)P})$
17 : $h_{m_0(1-s)P} \leftarrow \text{Subalg}(h_{m_0P}, h_{-m_0P}, h_{-(m_0+1)P}, h_{(m_0+s)P})$
18 : **end if**
19 : $h \leftarrow \text{Subalg}(h_{m_0P}, h_{m_1P}, h_{(m_0+m_1)P}, h_{m_0(1-s)P})$
20 : **else** $\triangleright$ $s \notin \mathcal{B}_2$ and $m_0{+}2m_1 \neq 0 \mod p$: Compute $h_{(m_0+sm_1)P}$ from $h_{m_0P}, h_{m_1P}, h_{(m_0+m_1)P}, h_{m_1(s-1)P}$
21 : **if** $m_1 \notin \mathcal{A}_1 \cup \mathcal{A}_2$ **then**
22 : $h_{(m_1+1)P} \leftarrow Alg_1(h_P, m_1)[2]$
23 : $h_{(m_1+s)P} \leftarrow \text{Subalg}(h_P, h_{m_1P}, h_{(m_1+1)P}, h_{(s-1)P})$
24 : $h_{m_1(1-s)P} \leftarrow \text{Subalg}(h_{m_1P}, h_{-m_1P}, h_{-(m_1+1)P}, h_{(m_1+s)P})$
25 : **end if**
26 : $h \leftarrow \text{Subalg}(h_{m_0P}, h_{m_1P}, h_{(m_0+m_1)P}, h_{m_1(s-1)P})$
27 : **end if**
28 : **if** $m > \frac{p-1}{2}$ **then return** $-h(x,-y)$ **else return** h **end if**

Theorem 24. *Algorithm 2 is correct.*

Proof. Let $\overline{m}$ be as in line 3 of the algorithm. If $m_0 = 0$ as in line 4, or $m_1 = 0$ as in line 5, then $h = h_{\overline{m}P}$ by Theorem 19.

Assume now that $m_0, m_1 \neq 0$, as in line 6. If $s \in \mathcal{B}_1 \cap \mathcal{B}_2$ as in line 7, then by Lemma 22.5 $\overline{m} = m_1(s+\alpha)$ for some $\alpha \in \mathcal{R}$. In addition, $h_{(s+\alpha)P} \in \mathcal{L}$, where $\mathcal{L}$ is the set of precomputed polynomials of line 1, defined in (5). Hence, by Theorem 19, one can compute $h = h_{\overline{m}P}$ as in line 8 of the algorithm.

Now consider the case in which $s \notin \mathcal{B}_1 \cap \mathcal{B}_2$, as in line 9 of the algorithm. Correctness of lines 10, 11 and 12 follows from Theorem 19.

In line 13 we have $s \notin \mathcal{B}_1$ and $2m_0 + m_1 \neq 0 \mod p$. Then, by Lemma 22.3, one can compute $h_{\overline{m}P} = h_{(m_0+sm_1)P}$ using Subalgorithm 1 with input lines h_{m_0P}, h_{m_1P}, $h_{(m_0+m_1)P}$ and $h_{m_0(1-s)P}$. We have already computed h_{m_0P}, h_{m_1P}, $h_{(m_0+m_1)P}$ in lines $10 - 12$. Hence, in order to be able to compute $h_{\overline{m}P}$ with Subalgorithm 1, we still need to compute $h_{m_0(1-s)P}$, see also Lemma 22.3.

If $m_0 \notin \mathcal{A}_1 \cup \mathcal{A}_2$ as in line 14, then one computes $h_{m_0(1-s)P}$ as in lines $15 - 17$ of the Algorithm, by Theorem 8, Theorem 19 and Lemma 22, points 1 and 2. If $m_0 \in \mathcal{A}_1 \cup \mathcal{A}_2$, then we cannot compute the polynomial $h_{m_0(1-s)P}$ as we do in lines $15 - 17$ of the algorithm. Nevertheless, in this case, $h_{m_0(1-s)P}$ belongs to the set $\mathcal{L}$ of precomputed polynomials, by construction of $\mathcal{L}$. Therefore, in both cases Subalgorithm 1 in line 19 correctly computes $h = h_{\overline{m}P}$, by Theorem 8.

Now consider lines $20 - 26$ of the algorithm. We have either $s \in \mathcal{B}_1$ or $2m_0 + m_1 = 0 \mod p$. Suppose first that $s \in \mathcal{B}_1$. Then $s \notin \mathcal{B}_2$, since $s \notin \mathcal{B}_1 \cap \mathcal{B}_2$. Moreover, if $s \in \mathcal{B}_1$ then $m_0 + 2m_1 \neq 0 \mod p$. In fact, one can check by direct computation that $s \in \mathcal{B}_1$ and $m_0 + 2m_1 = 0 \mod p$ implies $s \in \{0, -5^{\pm 1}, -3, -1, -4/5, 2/5 \mod p\}$, since $m_0, m_1 \neq 0 \mod p$, which contradicts the equality $s^2 + s + 1 = 0 \mod p$. Now suppose that $2m_0 + m_1 = 0 \mod p$. By the same arguments as above, one has that $2m_0 + m_1 = 0 \mod p$ implies $s \notin \mathcal{B}_2$ and $m_0 + 2m_1 \neq 0 \mod p$. Hence, in both cases considered in line 20, we have that $s \notin \mathcal{B}_2$ and $m_0 + 2m_1 \neq 0 \mod p$, and one can compute $h_{(m_0+sm_1)P}$ as in line 26 by Lemma 22, 4.

Similar arguments show that lines $21 - 25$ of the algorithm are correct, so Subalgorithm 1 at line 26 correctly outputs $h = h_{\overline{m}P}$. From line 2, we have that $h = h_{\overline{m}P} = h_{-mP}$ if

$m > \frac{p-1}{2}$, and $h = h_{\overline{m}P} = h_{mP}$ otherwise. Hence the algorithm correctly outputs h_{mP} in line 28 by Remark 21. □

Remark 25. The aim of Algorithm 2 is showing how to apply Frobenius reduction in order to speed up our scalar multiplication algorithm. However, further optimizations are possible. For example, one can introduce variations of Subalgorithm 1 in order to reduce the number of precomputed lines.

In conclusion, we give an example of optimized computation following with Algorithm 2.

Example 26. Let $q = 1021$ and $\mathbb{F}_{q^3} = \mathbb{F}_q[\zeta]/(\zeta^3 - 5)$. Let E and P be as in Example 6, Example 12, and Example 20, i.e., let E be the elliptic curve over $\mathbb{F}_q$ of equation $y^2 = x^3 + 230x + 191$ and let $P = (782\zeta^2 + 802\zeta + 45, 979\zeta^2 + 299\zeta + 133)$. Write $m = 483925 = m_0 + sm_1$, where $m_0 = 274$ and $m_1 = 3$.

Algorithm 1 computes h_{mP} by calling Subalgorithm 1 seventeen times with input $h_{m_1P}, h_{m_2P}, h_{n_1P}, h_{n_2P}$ for the following values of (m_1, m_2, n_1, n_2):

$$(1,6,3,4),(1,14,7,8),(1,28,14,15),(1,58,29,30),(1,118,59,60),(1,236,118,119),$$

$$(1,472,236,237),(1,944,472,473),(1,1890,945,946),(1,3780,1890,1891),$$

$$(1,7560,3780,3781),(1,15122,7561,7562),(1,30244,15122,15123),(1,60490,30245,30246),$$

$$(1,120980,60490,60491),(1,241962,120981,120982),(1,483924,241962,241963).$$

Performing the same computation with Algorithm 2, one has that

$$s \notin \mathcal{B}_1 = \{275, 757679, 717376, 508804, 304004, 263701, 527404\},\ 2m_0 + m_1 \neq 0 \mod p$$

and

$$m_0 \notin \mathcal{A}_1 \cup \mathcal{A}_2 = \{1021379, 860162, 161216, 860163, 322435, 161217, 232982, 627181\}.$$

Hence, after computing h_{m_0P}, $h_{(m_0+1)P}$, h_{m_1P}, $h_{(m_0+m_1)P}$, Algorithm 2 calls Subalgorithm 1 three times (in lines 16, 17 and 19) in order to compute h_{mP}. To compute h_{m_0P} and $h_{(m_0+1)P}$, Algorithm 1 calls Subalgorithm 1 with input $h_{m_1P}, h_{m_2P}, h_{n_1P}, h_{n_2P}$ for the following values of (m_1, m_2, n_1, n_2):

$$(1,4,2,3),(1,8,4,5),(1,16,8,9),(1,34,17,18),(1,68,34,35),(1,136,68,69),(1,274,137,138).$$

To compute $h_{(m_0+m_1)P}$, Algorithm 1 calls Subalgorithm 1 with input $h_{m_1P}, h_{m_2P}, h_{n_1P}, h_{n_2P}$ for the following values of (m_1, m_2, n_1, n_2):

$$(1,4,2,3),(1,8,4,5),(1,16,8,9),(1,34,17,18),(1,68,34,35),(1,138,69,70),(1,276,138,139).$$

Hence in total, taking into account overlapping in the computation of h_{m_0P} and $h_{(m_0+m_1)P}$, Algorithm 2 calls Subalgorithm 1 only twelve times.

References

[1] R. M. Avanzi, H. Cohen, C. Doche, G. Frey, T. Lange, K. Nguyen, F. Vercauteren, *Handbook of Elliptic and Hyperelliptic Curve Cryptography*, Discrete Mathematics and Its Applications 34, Chapman & Hall/CRC (2005).

[2] R. M. Avanzi, E. Cesena, *Trace zero varieties over fields of characteristic 2 for cryptographic applications*, Proceedings of the First Symposium on Algebraic Geometry and Its Applications – SAGA '07 (2007), 188-215.

[3] R. M. Avanzi, E. Cesena, T. Lange, *Trace Zero Varieties for Cryptographic Applications*, SPEED-CC, Berlin, October 13th, 2009.

[4] E. Cesena, *Pairing with Supersingular Trace Zero Varieties Revisited*, Available at http: // eprint.iacr.org/2008/404,2008.

[5] C. Diem, J. Scholten, *An attack on a trace-zero cryptosystem*, Available at `http://www.math.uni-leipzig.de/diem/preprints`.

[6] G. Frey, *Applications of Arithmetical Geometry to Cryptographic Constructions*, Proceedings of the 5th International Conference on Finite Fields and Applications, Springer (1999),128-161.

[7] N. Koblitz, *CM-curves with good cryptographic properties*, Advances in Cryptology - Crypto 1991, Lecture Notes in Comput. Sci., vol. 576, Springer-Verlag, Berlin, 1992, 279-287.

[8] E. Gorla, M. Massierer, *Point Compression for the Trace Zero Subgroup over a Small Degree Extension Field*, Designs, Codes and Cryptography 75, no. 2 (2015), 335-357.

[9] E. Gorla, M. Massierer, *An Optimal Representation for the Trace Zero Subgroup*, available at `http://arxiv.org/abs/1405.2733`.

[10] T. Lange, *Trace zero subvarieties of genus* 2 *curves for cryptosystem*, Ramanujan Math. Soc. 19, no. 1 (2004) 15-33.

[11] P. L. Montgomery, *Speeding the Pollard and elliptic curve methods of factorization*, Mathematics of Computation, 48(177):243:264, January 1987.

[12] N. Naumann, *Weil-Restriktion abelscher Varietäten*, Master's thesis (1999), available at `http://web.iem.uni-due.de/ag/numbertheory/dissertationen`.

[13] K. Rubin, A. Silverberg, *Supersingular abelian varieties in cryptology*, Advances in Cryptology: Proocedings of CRYPTO '02 (M. Young, ed), LNCS, vol. 2442, Springer, 2002, pp. 336-353.

[14] K. Rubin, A. Silverberg, *Using abelian varieties to improve pairing-based cryptography*, Journal of Cryptology 22, no. 3 (2009), 330-364.

[15] A. Silverberg, *Compression for Trace Zero Subgroups of Elliptic Curves*, Trends in Mathematics 8 (2005), 93-100.

[16] A. Weimerskirch, *The application of the Mordell-Weil group to cryptographic systems*, Master's thesis, Worcester Polytechnic Institute, Available at http://www.emsec.rub.de/media/crypto/attachments/files/2010/04/ms-weika.pdf, 2001.

A Explicit formulas

(1) Formulas for the coefficients of $S_{P,Q}$ in terms of the coefficients of h_P and h_Q.

$$a_4 = -\alpha_1^3\beta_1\beta_0^2 - 3B\alpha_1^3\beta_1 + 2A\alpha_1^3\beta_0 + 2\alpha_1^2\alpha_0\beta_1^2\beta_0 + A\alpha_1^2\alpha_0\beta_1 - 6B\alpha_1^2\beta_1^2 + 3A\alpha_1^2\beta_1\beta_0 + A^2\alpha_1^2 - \alpha_1\alpha_0^2\beta_1^3 + 6\alpha_1\alpha_0^2\beta_0 + 3A\alpha_1\alpha_0\beta_1^2 + 3\alpha_1\alpha_0\beta_0^2 + 9B\alpha_1\alpha_0 - 3B\alpha_1\beta_1^3 + A\alpha_1\beta_1^2\beta_0 + 2A^2\alpha_1\beta_1 - 3\alpha_1\beta_0^3 + 9B\alpha_1\beta_0 - 3\alpha_0^3\beta_1 + 3\alpha_0^2\beta_1\beta_0 - 3A\alpha_0^2 + 2A\alpha_0\beta_1^3 + 6\alpha_0\beta_1\beta_0^2 + 9B\alpha_0\beta_1 - 6A\alpha_0\beta_0 + A^2\beta_1^2 + 9B\beta_1\beta_0 - 3A\beta_0^2$$

$$a_3 = 4B\alpha_1^3\beta_1^3 - 2A\alpha_1^3\beta_1^2\beta_0 + A^2\alpha_1^3\beta_1 - \alpha_1^3\beta_0^3 + 9B\alpha_1^3\beta_0 - 2A\alpha_1^2\alpha_0\beta_1^3 - \alpha_1^2\alpha_0\beta_1\beta_0^2 + 3B\alpha_1^2\alpha_0\beta_1 - 7A\alpha_1^2\alpha_0\beta_0 + A^2\alpha_1^2\beta_1^2 - 6B\alpha_1^2\beta_1\beta_0 + 3A\alpha_1^2\beta_0^2 + 6AB\alpha_1^2 - \alpha_1\alpha_0^2\beta_1^2\beta_0 + A\alpha_1\alpha_0^2\beta_1 - 6B\alpha_1\alpha_0\beta_1^2 + 12A\alpha_1\alpha_0\beta_1\beta_0 - 8A^2\alpha_1\alpha_0 + A^2\alpha_1\beta_1^3 + 3B\alpha_1\beta_1^2\beta_0 + A\alpha_1\beta_1\beta_0^2 - 6AB\alpha_1\beta_1 + 4A^2\alpha_1\beta_0 - \alpha_0^3\beta_1^3 - 3\alpha_0^3\beta_0 + 3A\alpha_0^2\beta_1^2 + 21\alpha_0^2\beta_0^2 - 18B\alpha_0^2 + 9B\alpha_0\beta_1^3 - 7A\alpha_0\beta_1^2\beta_0 + 4A^2\alpha_0\beta_1 - 3\alpha_0\beta_0^3 + 18B\alpha_0\beta_0 + 6AB\beta_1^2 - 8A^2\beta_1\beta_0 - 18B\beta_0^2 + 4A^3 + 27B^2$$

$$a_2 = -A^2\alpha_1^3\beta_1^3 - 2A\alpha_1^3\beta_1\beta_0^2 - 6AB\alpha_1^3\beta_1 + A^2\alpha_1^3\beta_0 - 2A\alpha_1^2\alpha_0\beta_1^2\beta_0 + 5A^2\alpha_1^2\alpha_0\beta_1 - 3\alpha_1^2\alpha_0\beta_0^3 - 9B\alpha_1^2\alpha_0\beta_0 + 6AB\alpha_1^2\beta_1^2 + 9B\alpha_1^2\beta_0^2 + (2A^3+27B^2)\alpha_1^2 - 2A\alpha_1\alpha_0^2\beta_1^3 - 3\alpha_1\alpha_0^2\beta_1\beta_0^2 + 9B\alpha_1\alpha_0^2\beta_1 + 9A\alpha_1\alpha_0^2\beta_0 + 36B\alpha_1\alpha_0\beta_1\beta_0 - 12A\alpha_1\alpha_0\beta_0^2 - 18AB\alpha_1\alpha_0 - 6AB\alpha_1\beta_1^3 + 5A^2\alpha_1\beta_1^2\beta_0 + 9B\alpha_1\beta_1\beta_0^2 + (4A^3-27B^2)\alpha_1\beta_1 - 3A\alpha_1\beta_0^3 + 36AB\alpha_1\beta_0 - 3\alpha_0^3\beta_1^2\beta_0 - 3A\alpha_0^3\beta_1 + 9B\alpha_0^2\beta_1^2 - 12A\alpha_0^2\beta_1\beta_0 + 6A^2\alpha_0^2 + A^2\alpha_0\beta_1^3 - 9B\alpha_0\beta_1^2\beta_0 + 9A\alpha_0\beta_1\beta_0^2 + 36AB\alpha_0\beta_1 - 24A^2\alpha_0\beta_0 + (2A^3+27B^2)\beta_1^2 - 18AB\beta_1\beta_0 + 6A^2\beta_0^2$$

$$a_1 = -A^2\alpha_1^3\beta_1^2\beta_0 - 4B\alpha_1^3\beta_1\beta_0^2 + (A^3-12B^2)\alpha_1^3\beta_1 + 8AB\alpha_1^3\beta_0 - A^2\alpha_1^2\alpha_0\beta_1^3 - 4B\alpha_1^2\alpha_0\beta_1^2\beta_0 + 16AB\alpha_1^2\alpha_0\beta_1 - 3A^2\alpha_1^2\alpha_0\beta_0 + (-3A^3+12B^2)\alpha_1^2\beta_1^2 - 24AB\alpha_1^2\beta_1\beta_0 + 3A^2\alpha_1^2\beta_0^2 - 2A^2B\alpha_1^2 - 4B\alpha_1\alpha_0^2\beta_1^3 - 3A^2\alpha_1\alpha_0^2\beta_1 - 3\alpha_1\alpha_0^2\beta_0^3 + 15B\alpha_1\alpha_0^2\beta_0 - 24AB\alpha_1\alpha_0\beta_1^2 + 12A^2\alpha_1\alpha_0\beta_1\beta_0 - 6B\alpha_1\alpha_0\beta_0^2 - 18B^2\alpha_1\alpha_0 + (A^3-12B^2)\alpha_1\beta_1^3 + 16AB\alpha_1\beta_1^2\beta_0 - 3A^2\alpha_1\beta_1\beta_0^2 + 14A^2B\alpha_1\beta_1 - 3B\alpha_1\beta_0^3 + 63B^2\alpha_1\beta_0 - 3\alpha_0^3\beta_1\beta_0^2 - 3B\alpha_0^3\beta_1 - 3A\alpha_0^3\beta_0 + 3A^2\alpha_0^2\beta_1^2 - 6B\alpha_0^2\beta_1\beta_0 + 9A\alpha_0^2\beta_0^2 + 6AB\alpha_0^2 + 8AB\alpha_0\beta_1^3 - 3A^2\alpha_0\beta_1^2\beta_0 + 15B\alpha_0\beta_1\beta_0^2 + 63B^2\alpha_0\beta_1 - 3A\alpha_0\beta_0^3 - 42AB\alpha_0\beta_0 - 2A^2B\beta_1^2 - 18B^2\beta_1\beta_0 + 6AB\beta_0^2 + 4A^4 + 27AB^2$$

$$a_0 = 2A^2B\alpha_1^3\beta_1 - A^3\alpha_1^3\beta_0 - A^2\alpha_1^2\alpha_0\beta_1^2\beta_0 - 4B\alpha_1^2\alpha_0\beta_1\beta_0^2 + 12B^2\alpha_1^2\alpha_0\beta_1 - 4AB\alpha_1^2\alpha_0\beta_0 - 5A^2B\alpha_1^2\beta_1^2 + (A^3-24B^2)\alpha_1^2\beta_1\beta_0 + 6AB\alpha_1^2\beta_0^2 + (A^4+6AB^2)\alpha_1^2 - 4B\alpha_1\alpha_0^2\beta_1^2\beta_0 + 2A\alpha_1\alpha_0^2\beta_1\beta_0^2 - 2AB\alpha_1\alpha_0^2\beta_1 - A^2\alpha_1\alpha_0^2\beta_0 + (A^3-24B^2)\alpha_1\alpha_0\beta_1^2 + 24AB\alpha_1\alpha_0\beta_1\beta_0 - 3A^2\alpha_1\alpha_0\beta_0^2 + A^2B\alpha_1\alpha_0 + 2A^2B\alpha_1\beta_1^3 + 12B^2\alpha_1\beta_1^2\beta_0 - 2AB\alpha_1\beta_1\beta_0^2 + (-2A^4-6AB^2)\alpha_1\beta_1 - 5A^2B\alpha_1\beta_0 - \alpha_0^3\beta_0^3 - 3B\alpha_0^3\beta_0 + 6AB\alpha_0^2\beta_1^2 - 3A^2\alpha_0^2\beta_1\beta_0 + 3B\alpha_0^2\beta_0^2 + (A^3+9B^2)\alpha_0^2 - A^3\alpha_0\beta_1^3 - 4AB\alpha_0\beta_1^2\beta_0 - A^2\alpha_0\beta_1\beta_0^2 - 5A^2B\alpha_0\beta_1 - 3B\alpha_0\beta_0^3 + (2A^3-9B^2)\alpha_0\beta_0 + (A^4+6AB^2)\beta_1^2 + A^2B\beta_1\beta_0 + (A^3+9B^2)\beta_0^2 + 4A^3B + 27B^3$$

$$b_3 = \alpha_1^3\beta_0^2 - B\alpha_1^3 - 2\alpha_1^2\alpha_0\beta_1\beta_0 + A\alpha_1^2\alpha_0 + \alpha_1^2\beta_1\beta_0^2 - 3B\alpha_1^2\beta_1 + A\alpha_1^2\beta_0 + \alpha_1\alpha_0^2\beta_1^2 - 2\alpha_1\alpha_0\beta_1^2\beta_0 + 2A\alpha_1\alpha_0\beta_1 - 3B\alpha_1\beta_1^2 + 2A\alpha_1\beta_1\beta_0 + \alpha_0^3 + \alpha_0^2\beta_1^3 + 3\alpha_0^2\beta_0 + A\alpha_0\beta_1^2 + 3\alpha_0\beta_0^2 - B\beta_1^3 + A\beta_1^2\beta_0 + \beta_0^3$$

$$b_2 = A^2\alpha_1^3 + 3\alpha_1^2\alpha_0\beta_0^2 + 9B\alpha_1^2\alpha_0 + 3A^2\alpha_1^2\beta_1 + 3\alpha_1^2\beta_0^3 + 9B\alpha_1^2\beta_0 - 6\alpha_1\alpha_0^2\beta_1\beta_0 - 3A\alpha_1\alpha_0^2 -$$

$6\alpha_1\alpha_0\beta_1\beta_0^2+18B\alpha_1\alpha_0\beta_1-6A\alpha_1\alpha_0\beta_0+3A^2\alpha_1\beta_1^2+18B\alpha_1\beta_1\beta_0-3A\alpha_1\beta_0^2+3\alpha_0^3\beta_1^2+3\alpha_0^2\beta_1^2\beta_0-3A\alpha_0^2\beta_1+9B\alpha_0\beta_1^2-6A\alpha_0\beta_1\beta_0+A^2\beta_1^3+9B\beta_1^2\beta_0-3A\beta_1\beta_0^2$

$b_1=-A^2\alpha_1^3\beta_1^2-2A\alpha_1^3\beta_0^2+2AB\alpha_1^3-12B\alpha_1^2\alpha_0\beta_1^2+4A\alpha_1^2\alpha_0\beta_1\beta_0-3A^2\alpha_1^2\alpha_0-A^2\alpha_1^2\beta_1^3-12B\alpha_1^2\beta_1^2\beta_0+4A\alpha_1^2\beta_1\beta_0^2+A^2\alpha_1^2\beta_0+4A\alpha_1\alpha_0^2\beta_1^2-3\alpha_1\alpha_0^2\beta_0^2-9B\alpha_1\alpha_0^2+4A\alpha_1\alpha_0\beta_1^2\beta_0+2A^2\alpha_1\alpha_0\beta_1+6\alpha_1\alpha_0\beta_0^3+18B\alpha_1\alpha_0\beta_0+2A^2\alpha_1\beta_1\beta_0+9B\alpha_1\beta_0^2+(4A^3+27B^2)\alpha_1+6\alpha_0^3\beta_1\beta_0+A\alpha_0^3-2A\alpha_0^2\beta_1^3-3\alpha_0^2\beta_1\beta_0^2+9B\alpha_0^2\beta_1-9A\alpha_0^2\beta_0+A^2\alpha_0\beta_1^2+18B\alpha_0\beta_1\beta_0-9A\alpha_0\beta_0^2+2AB\beta_1^3-3A^2\beta_1^2\beta_0-9B\beta_1\beta_0^2+(4A^3+27B^2)\beta_1+A\beta_0^3$

$b_0=-2A^2\alpha_1^3\beta_1\beta_0-8B\alpha_1^3\beta_0^2+(A^3+8B^2)\alpha_1^3+A^2\alpha_1^2\alpha_0\beta_1^2-8B\alpha_1^2\alpha_0\beta_1\beta_0+6A\alpha_1^2\alpha_0\beta_0^2-2AB\alpha_1^2\alpha_0+A^2\alpha_1^2\beta_1^2\beta_0+4B\alpha_1^2\beta_1\beta_0^2+(-A^3-12B^2)\alpha_1^2\beta_1+4AB\alpha_1^2\beta_0+4B\alpha_1\alpha_0^2\beta_1^2+A^2\alpha_1\alpha_0^2-2A^2\alpha_1\alpha_0\beta_1^3-8B\alpha_1\alpha_0\beta_1^2\beta_0+8AB\alpha_1\alpha_0\beta_1-6A^2\alpha_1\alpha_0\beta_0+(-A^3-12B^2)\alpha_1\beta_1^2+8AB\alpha_1\beta_1\beta_0-3A^2\alpha_1\beta_0^2+3\alpha_0^3\beta_0^2+B\alpha_0^3-8B\alpha_0^2\beta_1^3+6A\alpha_0^2\beta_1^2\beta_0-3A^2\alpha_0^2\beta_1+3\alpha_0^2\beta_0^3-15B\alpha_0^2\beta_0+4AB\alpha_0\beta_1^2-6A^2\alpha_0\beta_1\beta_0-15B\alpha_0\beta_0^2+(4A^3+27B^2)\alpha_0+(A^3+8B^2)\beta_1^3-2AB\beta_1^2\beta_0+A^2\beta_1\beta_0^2+B\beta_0^3+(4A^3+27B^2)\beta_0$

(2) Doubling formulas for h_P. Write $h_{2P}=cy-(u_0+u_1x)$, then:

$u_1=4B\alpha_1^4-4A\alpha_1^3\alpha_0+4A^2\alpha_1^2-4\alpha_1\alpha_0^3+36B\alpha_1\alpha_0-12A\alpha_0^2$

$u_0=-A^2\alpha_1^4-8B\alpha_1^3\alpha_0+2A\alpha_1^2\alpha_0^2+6AB\alpha_1^2-8A^2\alpha_1\alpha_0-\alpha_0^4-18B\alpha_0^2+4A^3+27B^2$

$c=8B\alpha_1^3-8A\alpha_1^2\alpha_0-8\alpha_0^3$

(3) Tripling formulas for h_P. Write $h_{3P}=dy-(v_0+v_1x)$, then:

$v_1=1/3A^4\alpha_1^9+8A^2B\alpha_1^8\alpha_0+(-4A^3+48B^2)\alpha_1^7\alpha_0^2+(16A^3B+144B^3)\alpha_1^7-48AB\alpha_1^6\alpha_0^3+(-16A^4-240AB^2)\alpha_1^6\alpha_0+10A^2\alpha_1^5\alpha_0^4+192A^2B\alpha_1^5\alpha_0^2+(8A^5+54A^2B^2)\alpha_1^5-24B\alpha_1^4\alpha_0^5+(-112A^3+144B^2)\alpha_1^4\alpha_0^3+(96A^3B+648B^3)\alpha_1^4\alpha_0+12A\alpha_1^3\alpha_0^6-240AB\alpha_1^3\alpha_0^4+(-48A^4-324AB^2)\alpha_1^3\alpha_0^2+(-32A^4B-216AB^3)\alpha_1^3-48A^2\alpha_1^2\alpha_0^5+(64A^5+432A^2B^2)\alpha_1^2\alpha_0+3\alpha_1\alpha_0^8-288B\alpha_1\alpha_0^6+(-24A^3-162B^2)\alpha_1\alpha_0^4+(288A^3B+1944B^3)\alpha_1\alpha_0^2+(-16A^6-216A^3B^2-729B^4)\alpha_1+48A\alpha_0^7+(-64A^4-432AB^2)\alpha_0^3$

$v_0=(-8/3A^3B-64/3B^3)\alpha_1^9+(3A^4+32AB^2)\alpha_1^8\alpha_0-16A^2B\alpha_1^7\alpha_0^2-8A^2B^2\alpha_1^7+(12A^3+16B^2)\alpha_1^6\alpha_0^3+(8A^3B-144B^3)\alpha_1^6\alpha_0+8AB\alpha_1^5\alpha_0^4+288AB^2\alpha_1^5\alpha_0^2+(32A^4B+216AB^3)\alpha_1^5+10A^2\alpha_1^4\alpha_0^5-200A^2B\alpha_1^4\alpha_0^3+(-24A^5-162A^2B^2)\alpha_1^4\alpha_0+32B\alpha_1^3\alpha_0^6+(64A^3+72B^2)\alpha_1^3\alpha_0^4+(192A^3B+1296B^3)\alpha_1^3\alpha_0^2+(96A^3B^2+648B^4)\alpha_1^3-4A\alpha_1^2\alpha_0^7-72AB\alpha_1^2\alpha_0^5+(-176A^4-1188AB^2)\alpha_1^2\alpha_0^3+(-192A^4B-1296AB^3)\alpha_1^2\alpha_0+64A^2\alpha_1\alpha_0^6+(128A^5+864A^2B^2)\alpha_1\alpha_0^2+1/3\alpha_0^9+72B\alpha_0^7+(-120A^3-810B^2)\alpha_0^5+(192A^3B+1296B^3)\alpha_0^3+(-16A^6-216A^3B^2-729B^4)\alpha_0$

$d=A^4\alpha_1^8+24A^2B\alpha_1^7\alpha_0+(-12A^3+144B^2)\alpha_1^6\alpha_0^2+(-24A^3B-144B^3)\alpha_1^6-144AB\alpha_1^5\alpha_0^3+(32A^4+144AB^2)\alpha_1^5\alpha_0+30A^2\alpha_1^4\alpha_0^4+120A^2B\alpha_1^4\alpha_0^2+(-8A^5-54A^2B^2)\alpha_1^4-72B\alpha_1^3\alpha_0^5+720B^2\alpha_1^3\alpha_0^3+(-96A^3B-648B^3)\alpha_1^3\alpha_0+36A\alpha_1^2\alpha_0^6-360AB\alpha_1^2\alpha_0^4+(48A^4+324AB^2)\alpha_1^2\alpha_0^2+96A^2\alpha_1\alpha_0^5+9\alpha_0^8+72B\alpha_0^6+(24A^3+162B^2)\alpha_0^4-16/3A^6-72A^3B^2-243B^4$