Improving the arithmetic of Kummer lines

DAMIEN ROBERT

Abstract. We explain some improvements to the arithmetic of Kummer lines: doublings, differential additions, scalar multiplications, pairings, isogenies.

Contents

Date: November 2023.

1. Introduction

This is a summery of results that will be presented in a series of articles on the arithmetic of Kummer lines.

- In [BRS23], we focus on the general theory of models of Kummer lines, the conversions between them, and the arithmetic properties of their 2-torsion points (with the relationship between the ramification, the 2-Tate pairing, the 2-theta group and their Galois representation).
- In [RS24], we study isogenies between Kummer lines, and in particular we focus on 2-isogenies. We use the action of the theta group $G(2(0_E))$ rather than Vélu's formula to compute invariant sections, and the fact that the Kummer model is determined by its ramification, to find new and old formulas. This allows us to give a general framework to find equations for 2-isogenies and doublings. We also develop an hybrid arithmetic, combining the best of the (twisted) theta and Montgomery models.
- In [Rob22], we extend the work of [RS24] from doublings to differential additions on a Kummer model (the formula crucially depend on the arithmetic property of the 2-torsion alluded to above). Notably, we explain how to find differential additions formulae which factor through a 2-isogeny. As an application we develop a novel time/memory trade off for the Montgomery ladder.
- In [Rob23c] we develop the arithmetic of the biextension associated to the divisor $2(0_E)$ on some Kummer models. We extend this to the effective computation of the cubical torsor structure. We derive from this efficient pairing formulae.
- In [Rob23f], we use the formula from [Rob23c] to revisit the "Projective coordinates leak" paper [NSS04]. We show that leaking the projectives coordinate in the Montgomery ladder yields a subexponential time recovery of the full secret key (by reduction to the DLP over the base field). The previous attacks only recovered a few bits by leak.

A proof of concept implementation of these algorithms is available in [Rob23e].

2. Summary

In isogeny based cryptography, it is standard to work with the Montgomery model of a Kummer line. In the case where we have an extra point of 2-torsion T_2 along with the standard point of 2-torsion $T_1 = (0 : 1)$ (as happens for supersingular curves), we can use T_2 to speed up the arithmetic.

2.1. **Hybrid arithmetic.** (This is joint work with Nicolas Sarkis):

2.1.1. *Hybrid arithmetic for scalar multiplication.* In the Montgomery ladder for computing $m.P$, we use one doubling and one mixed differential addition by step. In the Montgomery model, doubling is $2M + 2S + 1m_0$ while a mdiffAdd (where we assume our base point $P = (X_P : 1)$ is normalised) is $3M + 2S$, so a ladder step is $5M + 4S + 1m_0$. Here m_0 denotes a multiplication by a curve constant (typically the coefficient A of the Montgomery curve, or rather $(A + 2)/4$). If our starting point $P = (X_P : Z_P)$ is not normalised, we need to add $1M$ by bit to the ladder cost.

When T_2 is rational, we can also use a twisted theta model, where doubling is $4S + 2m_0$, and mdiffAdd is $3M+2S+1m_0$, so a ladder step is $3M+6S+3m_0$. (There is a $1M-1S-1m_0$ tradeoff where a ladder step is $4M + 5S + 2m_0$.)

The two models differ by the translation by T_2 (the doubling $P \mapsto 2.P$ in twisted theta can be interpreted as $P \mapsto 2.P + T_2$ in the Montgomery model and conversely), we can exploit that to combine the best of both worlds: using an hybrid arithmetic where doubling is $4S + 2m_0$ and mdiffAdd is $3M + 2S$. Keeping track of the translation by T_2 we then have a hybrid ladder which cost $3M + 6S + 2m_0$.

2.1.2. *Hybrid arithmetic for 2^n-isogenies.* In the Montgomery model, a 2-isogeny codomain costs $2S$, an image costs $4M$, and doubling cost $2S+2M+2m_0$ (because our curve coefficients are given by a projective point $(A : C)$ and we can no longer assume that $C = 1$). In practice, it is customary to use 4-isogenies instead where the codomain cost $4S$ and images $2S + 6M$.

In the twisted theta model, a 2-isogeny codomain costs $2S$, an image costs $2S + 2M$, and doubling cost $4S + 4m_0$.

Again, in a Montgomery model with full 2-torsion, it is possible to use an hybrid version, with an image costing $2S + 2m_0$ (translated image from the point of view of the Montgomery model) and doubling costing $2S + 2M + 2m_0$ (translated doubling from the point of view of the twisted theta model).

This lines up the cost of two 2-isogeny with the cost of a 4-isogeny (and is actually slightly better). However, for a 2^n-isogeny chain, it is still better to split into 4-isogenies since this gain on the codomain computations.

2.2. **A time/memory trade off for scalar multiplication on Kummer lines.** We have a time/memory trade off for a Montgomery model with full two-torsion, where we precompute some points to speed up scalar multiplications.

We first start with a precomputation depending only on the base point P and which cost of $2S + 1m_0$ by bits (+ the storage of 2 coefficients by bits). Then a scalar multiplication is in $4M + 2S + 1m_0$ by bit (whether the base point is normalised or not).

The total cost, including the precomputation, is $4M + 4S + 2m_0$ which makes it slightly better than the standard Montgomery ladder (and saves $1M$ on non normalised points).

We can do more precomputions by using 1 global inversion and $2S + 1m_0 + 4M$ by bit, then the following scalar multiplications will cost $3M + 2S + 1m_0$ by bit.

A similar algorithm works in higher dimension. For a Kummer surface the precomputation step costs one global inversion and $12M + 4S + 3m_0$ by bits, and then a scalar multiplication with the same base point costs $7M + 4S + 3m_0$; compared to $7M + 12S + 9m_0$ or $10M + 9S + 6m_0$ for the standard ladder.

2.3. **Pairings on Kummer lines.**

2.3.1. *Generic pairings.* Isogeny based cryptography rely on generic pairings, where we cannot assume that one point lives in a smaller field. In [CLN16], the generic cost of the Tate pairing then becomes $5S + 15M$ for doublings, and $4S + 20M$ for additions (see [Rei23]); much more expansive than a simple scalar multiplication. The best generic algorithm in the litterature, in [BELL10], uses $10M + 9S$ for doubling, and $11.5M + 3S$ by addition.

We work out the arithmetic of the biextension associated to the divisor $2(0_E)$ on the Montgomery model of a Kummer line with full rational 2-torsion. We derive from this an efficient ladder like algorithm for pairings computation. Our ladder algorithm costs $7S + 9M$ by bit, which is closer to the cost of a scalar multiplication via the Montgomery ladder. As special cases, when $n = 2^m$ or we compute a self pairing, the cost goes down to $4S + 6M$ by bit.

We also explain how to compute a standard exponentiation (rather than a ladder) in the biextension, this allows to use window-NAF methods. Our algorithm (for now only in the theta model) costs $5S + 6M$ for a doubling, and $6S + 24M$ for an addition.[1] This suggests that the second algorithm will be faster than the first one when using a window $w \geq 5$ (or when computing pairings between points of 2^n-torsion).

2.3.2. *Pairing based cryptography.* For pairing based cryptography on elliptic curves, it is convenient to use the Tate pairing with $P \in \mathbb{G}_1 \subset E(\mathbb{F}_q)$, $Q \in \mathbb{G}_2 \subset E(\mathbb{F}_{q^k})$, and k even to allow for denominator elimination.

Counting only operations involving the big field $\mathbb{F}_{q^k}$, Miller's algorithm cost $1M+1S+1m$ by doubling, and $1M+1m$ by addition. Here $1m$ denotes a multiplication between a coefficient in $\mathbb{F}_q$ and a coefficient in $\mathbb{F}_{q^k}$.

When denominator elimination is not possible (because k is odd or Q is not in $\mathbb{G}_2$), the cost becomes $2M + 2S + 1m$ by doubling, and $2M + 1m$ by addition.

When P is in the small field $E(\mathbb{F}_q)$ and Q is in the big field $E(\mathbb{F}_{q^k})$, our Tate pairing algorithm costs (counting only operations in the big field) $2S + 1M + 2m$ by bits. This is competitive with the standard Miller's algorithm, except when denominator elimination is available.

2.4. **Monodromy leak: Projective coordinates leak revisited.** Assume that we are doing a scalar multiplication via the Montgomery ladder: we start with $P = (x_P : 1)$ and compute $Q = n.P = (X_Q : Z_Q)$. In practice, during the ladder we work with affine coordinates (X_Q, Z_Q) rather than projective coordinates (which would imply one division at each step, or at least scaling by a random scalar). It is only at the end of the computation that a division is computed and the coordinate $x_Q = X_Q/Z_Q$ is returned.

A projective coordinates leak happens whenever an attacker can retrieve (X_Q, Z_Q) directly. It was shown in [NSS04] how to use a projective coordinates leak to retrieve a few bits of the secret scalar n. This was revisited in [AGB20] to adapt it to the Montgomery ladder, still recovering only a few bits.

Instead, we can use the formula from the biextension arithmetic (more precisely, we use the cubical torsor structure, a refinement of the biextension arithmetic) to fully recover the secret n via:

- Solving some DLPs in $\mathbb{F}_q^*$
- Solving a degree 2 equation in $\mathbb{Z}/(q-1)\mathbb{Z}$.

In most cases (except if $q-1$ has a lot of prime divisors) this can be done in subexponential time. The name monodromy leak comes from the fact that the biextension arithmetic and cubical torsor structure gives the monodromy information underlying the Tate and Weil pairing.

3. Models

3.1. **The Montgomery model.** The Montgomery model of E is rational whenever there is a rational cyclic subgroup of order 4 in E, i.e. a point R_1 of order four which is rational in the Kummer line (i.e. $\pi(R_1) = \pm R_1$), i.e. there is a point of order 2 T_1 with trivial self Tate pairing. The Montgomery model is the model where R_1 is sent to $(1 : 1)$, $T_1 = 2R_1$ to $(0 : 1)$ and 0_E to infinity.

The ramification is given by $(0_E) = (1 : 0)$, $T_1 = (0 : 1)$, $T_2 = (A^2 : B^2)$, $T_3 = T_1 + T_2 = (B^2 : A^2)$. Here, T_2, T_3 are not necessarily rational, we denote their coordinates by $(A^2 : B^2)$ to make the link with the theta model more explicit later. Conversely, a Kummer

[1]Standard additions are not available on a Kummer line, but we can compute them over the biextension!

line whose neutral point is at infinity, a ramification point is $(0:1)$, and the remaining two ramifications points are invariant under $(X:Z) \mapsto (Z:X)$ gives a Montgomery model.

Above the canonical two torsion point $T_1 = (0:1)$, we have the canonical four torsion points $R_1 = (1:1) = R_1 + T_1, R_1' = (-1:1) = R_1 + T_2 = R_1 + T_3$.

If T_2 is rational, its coefficients are enough to represent E. The translation by T_2 is given by $(x:z) \mapsto (A^2x - B^2z : B^2x - A^2z)$. From this we can recover the curve coefficient $\mathcal{A}$ of the Montgomery model by $\mathcal{A} = (A:C) = (A^4+B^4 : -A^2B^2)$, $(\mathcal{A}+2)/4 = ((A^2-B^2)^2 : -4A^2B^2)$.

In these notes, we will often focus on the arithmetic of the Kummer line E of a Montgomery model with full rational 2-torsion. In this case, the quotient $E' = E/T_1$ is also a Montgomery model with full rational 2-torsion, so we can exploit the symmetry between E and E' in our arithmetic by factorising through the isogeny $f : E \to E'$.

3.2. **Twisted theta models.** The Kummer line associated to a theta model $\theta(a:b)$ has for neutral point $0_E = (a:b)$ and ramification $T_1 = (-a:b), T_2 = (b:a), T_3 = (-b:a)$. We have $R_1 = (1:0), R_1' = (0:1)$ two 4-torsion points above $T_1 = (-a:b)$, and $R_2 = (1:1), R_2' = (1:-1)$ above $T_2 = (b:a)$. We denote $(A^2:B^2) = (a^2+b^2 : a^2-b^2)$.

Conversely, a Kummer line with two rational points of 4-torsion R_1, R_2 such that $T_1 = 2R_1 \neq T_2 = 2R_2$ admits a rational theta model. Equivalently, there are two cyclic subgroups of degree 4 on E, K_1, K_2 such that $K_1 \cap K_2 = \{0_E\}$.

We recall that a Montgomery model can be constructed as long as we have a point of 4-torsion on the Kummer. In theta we have two such points: $(1:0)$ above $(-a:b)$ and $(1:1)$ above $(b:a)$, so we have two associated models. Conversion formula are given in Section 3.4.

From a theta model, we explain how to construct several twisted theta models. When we have a theta model $\theta(a:b)$, we can use the dual theta coordinates given by the Hadamard transform, let me denote that by $\theta'(a':b')$ with $(a':b') = (a+b:a-b)$. We can also twist the theta model by looking at the coordinates $(ax:bz)$ instead of $(x:z)$, let me call this $\theta tw(a^2:b^2)$. We can combine the Hadamard transform and the twisted models to obtain four kind of twisted theta models: $\theta tw, \theta tw', \theta' tw, \theta' tw'$.

A theta model on E arises from a (symmetric) isomorphism of the Heisenberg group of level 2 with the theta group $G(2(0_E))$. In a twisted theta model we take an isomorphism from a twist of the Heisenberg group, so we are on the same elliptic curve, it is the theta structure which is twisted. We will use the model $\theta tw'$, the conversion from θ is $(x:z) \mapsto (ax+bz : ax-bz)$; and the twisted model $\theta' tw'$, the conversion from θ is $(x:z) \mapsto (a'x' + b'z' : a'x' - b'z')$ where $(x':z') = (x+z : x-z)$. We will see that $\theta' tw'$ is (up to translation) the Montgomery model associated to the four torsion point $(1:1)$, and the Montgomery model (up to translation) corresponding to the four torsion point $(1:0)$ is given by $\theta tw'$.

In the $\theta tw'$ model, the neutral point becomes $0_E = (A^2:B^2)$, the 2-torsion $T_1 = (B^2 : A^2), T_2 = (1:0), T_3 = (0:1)$, and the 4-torsion is $R_1 = (1:1), R_1' = (-1:1)$, and $R_2 = (a':b') = (a+b:a-b), R_2' = (b', a')$. In particular, the 4-torsion point $R_2 = (a':b')$ above $T_2 = (1:0)$ allows to recover $(a:b)$.

It is convenient to see $\theta' tw'$ coordinates as follow. Let start with a theta model $\theta(a:b)$ on an elliptic curve E_1; we have an isogeny $f : E_1 \to E_2$ whose kernel is given by the two torsion point $(-a:b)$. We also have a "contragredient" isogeny $\widehat{g} : E_1 \to E_0$ whose kernel is given by $(b:a)$.

If E_0 is given by $\theta(a_0:b_0)$, then $g : E_0 \to E_1$ has kernel $(-a_0:b_0)$. From the isogeny formula [Rob23a], we see that a $\theta' tw'$ coordinate $(u:v)$ for $P \in E_1$ can be written as

$(u : v) = (x^2 : z^2)$ where $(x : z)$ is the theta coordinate of $Q \in E_0 \mid g(Q) = P$. In other words: if we use squares of theta coordinates in E_0 to represent points of E_1 we obtain the $\theta' tw'$ model: $\theta^2_{E_0} = \theta' tw'_{E_1}$. We have $(a_0^2 : b_0^2) = (a'^2 + b'^2 : a'^2 - b'^2) = (a^2 + b^2 : 2aa^2 + b^2 : 2ab)$. A similar interpretation holds for the $\theta tw'$ model.

3.3. **Montgomery and theta models.** In the Montgomery model, the neutral point is $0 = (1 : 0)$, the 2-torsion is $T_1 = (0 : 1), T_2 = (A^2 : B^2), T_3 = (B^2 : A^2)$, and the four torsion is $R_1 = (1 : 1), R'_1 = (-1 : 1), R_2 = (a' : b'), R'_2 = (b' : a')$.

In the $\theta tw'$ model, the neutral point becomes $0_E = (A^2 : B^2)$, the 2-torsion $T_1 = (B^2 : A^2), T_2 = (1 : 0), T_3 = (0 : 1)$, and the 4-torsion is $R_1 = (1 : 1), R'_1 = (-1 : 1)$, and $R_2 = (a' : b') = (a + b : a - b), R'_2 = (b', a')$.

The twisted theta model has the same ramification as the Montgomery model, except the neutral point is $(A^2 : B^2)$ which would be a point of 2-torsion T_2 on the Montgomery model, hence why they differ by translation by T_2: $(x : z) \mapsto (A^2 x - B^2 z : B^2 x - A^2 z)$. This sends $(-1 : 1)$ to $(1 : 1)$ and conversely (be careful that due to an unfortunate choice of notations, the R_1 on $\theta tw'$ is sent to R'_1 on Montgomery).

We also have a similar conversion to the Montgomery model on the $\theta' tw'$ model. The two torsion on $\theta' tw'$ is given by $0_E = (a_0^2 : b_0^2)$, $T_2 = (b_0^2 : a_0^2)$, $T_1 = (1 : 0)$ and $T_3 = (0 : 1)$. We also have the four torsion point $(1 : 1)$ above $(b_0^2 : a_0^2)$.

The two torsion gives the ramification on the Kummer line. Now notice how we have exactly the same ramification as the Montgomery model $M : y^2 = x(x - \alpha)(x - 1/\alpha)$ with $\alpha = b_0^2/a_0^2$, except that in our case the neutral point is $(a_0^2 : b_0^2)$ while in Montgomery the neutral point is $(0 : 1)$.

This means that the map $\mathrm{Id} : \theta' tw' \to M$ corresponds to the translation by the two torsion point $T_1 = (1 : 0)$ on the $\theta' tw'$ model and by $T_1 = (a_0^2 : b_0^2)$ on the Montgomery model. Via this translation, the four torsion point $(1 : 1)$ above $(b_0^2 : a_0^2)$ indeed become the four torsion point $(1 : 1)$ above $(0 : 1)$ in the Montgomery model as expected.

This gives the following fact: suitably twisting the theta structure, we obtain conversion formula which are free (ie given by the identity) except that a point P in the twisted theta model will correspond to a point $P + T$ in the Montgomery model for some two torsion point T. If we can get an handle on this translation by T, we can combine the best formula for both models.

This was already used in [BRS23] to construct a hybrid Montgomery ladder combining the best of the theta and Montgomery formula. We now describe a similar approach for isogenies. (It is Nicolas Sarkis who found out that we had a free conversion formula between the two models up to a translation by a point of 2-torsion[2], and I realised we could exploit this for isogenies and the scalar multiplication; the implementation was done by Nicolas.)

3.4. **Conversion formula between the theta model and the Montgomery model in dimension 1.** See also [Rob23a, Appendix A].

Let E/k be an elliptic curve, and $(a : b) = (\theta_0(0_E), \theta_1(0_E))$ be its theta null point. We give formula to convert the theta points $(\theta_0(P) : \theta_1(P))$ into the Montgomery coordinates $(x(P) : z(P))$. The formulas follows by looking at the ramification on the Kummer line on both models, and finding the homography that maps the ramification of one model to the other.

[2]We found out afterwards that this was already done in [HR19]

When the theta null point is rational, the elliptic curve E admits both a rational Montgomery model and a rational Legendre model. They are given by

$$y^2 = x(x-\alpha)(x-1/\alpha) = x(x^2 + \mathcal{A}x + 1)$$

and (up to a quadratic twist, which is harmless because we work on the Kummer line anyway) by

$$y^2 = x(x-1)(x-\lambda).$$

These constants are determined as follows: let $(A : B)$ be the dual coordinates of the canonical 2-isogenous curve (we will only need their square). We have

$$A^2 = a^2 + b^2, B^2 = a^2 - b^2, \tag{1}$$

$$\alpha = A^2/B^2 = (a^2+b^2)/(a^2-b^2), \tag{2}$$

$$\lambda = \alpha^2 = A^4/B^4 = (a^2+b^2)^2/(a^2-b^2)^2, \tag{3}$$

$$\mathcal{A} = -(\alpha + 1/\alpha) = -(\alpha^2+1)/\alpha = -(A^4+B^4)/(A^2B^2) = -2(a^4+b^4)/(a^4-b^4), \tag{4}$$

$$(\mathcal{A}+2)/4 = -b^4/(a^4-b^4). \tag{5}$$

Conversely, from $\mathcal{A}$, we can recover $(a : b)$ via

$$\alpha + 1/\alpha = -\mathcal{A}, \tag{6}$$

$$A^2/B^2 = \alpha, \tag{7}$$

$$a^2 = A^2 + B^2, b^2 = A^2 - B^2, (a^2 : b^2) = (\alpha+1 : \alpha-1). \tag{8}$$

We note that if $(a : b)$ is a solution, then $(a : \zeta b)$ also with $\zeta \in \mu_4$, these correspond to different theta structures.

With these constants defined, we can now explain how to convert the points. If $P = (x : z)$ in Montgomery coordinates, then

$$(\theta_0(P) : \theta_1(P)) = (a(x-z) : b(x+z)). \tag{9}$$

Conversely, if $P = (\theta_0 : \theta_1)$, then in Montgomery coordinates

$$(x(P) : z(P)) = (a\theta_1 + b\theta_0 : a\theta_1 - b\theta_0). \tag{10}$$

On the theta model $0_E = (a : b)$, we have a canonical basis of the 2-torsion given by $T_1 = (a : -b)$ and $T_2 = (b : a)$. We have a canonical basis of the 4-torsion given by $T_1' = (1 : 0)$ above T_1 and $T_2' = (1 : 1)$ above T_2. The map above sends T_1 to $(0 : 1)$ in the Montgomery model, T_1' to $(1 : 1)$, T_2 to $(A^2 : B^2)$, T_2' to $(a+b : a-b)$.

So conversely, given a Montgomery curve, the canonical point $T' = (1 : 1)$ of 4-torsion above the 2-torsion point $T = (0 : 1)$ and a second point $T'' = (r : s)$ above another point of 2-torsion, then the theta null point $(a : b)$ induced by the basis (T', T'') of the 4-torsion is given by $(r+s : r-s)$.

For the case of a general elliptic curve E with a basis (T', T'') of the 4-torsion, we first convert E to a Montgomery model by sending T' to $(1 : 1)$ and $T = 2T'$ to $(0 : 1)$, the map is then $x \mapsto (x - x(T))/(x(T') - x(T))$. Then we apply the above formula to the image of T''.

The alternative Montgomery model. When we have a theta model, we can also introduce the dual theta coordinates

$$(\theta'_0 : \theta'_1) = (\theta_0 + \theta_1 : \theta_0 - \theta_1),$$

in particular the dual theta null point is given by $(a' : b') = (a + b : a - b)$. We can construct another Montgomery model by replacing in the above formula $(a, b, \theta_0, \theta_1)$ by $(a', b', \theta'_0, \theta'_1)$.

Plugging in this different model the equations expressing $(a', b', \theta'_0, \theta'_1)$ in terms of $(a, b, \theta_0, \theta_1)$, we obtain alternative formulas:

$$A'^2 = a'^2 + b'^2 = 2(a^2 + b^2), B'^2 = a'^2 - b'^2 = 4ab, \tag{11}$$

$$\alpha' = A'^2/B'^2 = (a^2 + b^2)/(2ab), \lambda' = \alpha'^2, \tag{12}$$

$$\mathcal{A}' = -(\alpha' + 1/\alpha') = -(a^4 + 6a^2b^2 + b^4)/(2(a^3b + ab^3)), \tag{13}$$

$$P = (x : z) \mapsto (\theta_0(P), \theta_1(P)) = (ax - bz : bx - az), \tag{14}$$

$$(\theta_0, \theta_1) \mapsto (x(P) : z(P)) = (a\theta_0 - b\theta_1 : b\theta_0 - a\theta_1). \tag{15}$$

4. Scalar multiplication on Kummer lines

4.1. **Standard arithmetic in the Montgomery and theta models.** Let us first recall the standard formulas in the theta and Montgomery models.

Differential additions in theta coordinates are computed as follow, using [Rob23a, § 5, § 6]. Let $(a : b)$ be the neutral point, and $(A : B)$ as usual: $(A^2 : B^2) = (a^2 + b^2 : a^2 - b^2)$. Let $(XX_P : ZZ_P) = (X_P^2 + Z_P^2 : X_P^2 - Z_P^2)$, $(XX_Q : ZZ_Q) = (X_Q^2 + Z_Q^2 : X_Q^2 - Z_Q^2)$, $(U : V) = (B^2 XX_P XX_Q : A^2 ZZ_P ZZ_Q)$, $(X(P+Q) : Z(P+Q)) = (Z(P-Q)(U+V) : X(P-Q)(U-V))$. Applying this to $P = Q$ gives the doubling.

It is easy to extend this formula to the different twisted variant of the theta model.

In the Montgomery model, the usual differential addition is given as follow: $(U_1 : U_2) = (X_P + Z_P : X_P - Z_P)$, $(U_3 : U_4) = (X_Q + Z_Q : X_Q - Z_Q)$, $(X(P + Q) : Z(P + Q)) = (Z(P - Q)(U_1U_4 + U_2U_3)^2 : X(P - Q)(U_1U_4 - U_2U_3)^2)$.

We cannot apply this for doubling however, because the neutral point in Montgomery is $(1 : 0)$ so we get a division by zero. For doubling we instead use: $(U_1 : U_2) = (X_P + Z_P : X_P - Z_P)$, $U_3 = (U_1^2 - U_2^2)$, $(X(2P) : Z(2P)) = (U_1^2U_2^2 : U_3(U_2^2 + (\mathcal{A} + 2)/4U_3)$. If $\mathcal{A} = (\mathsf{A} : \mathsf{C})$, the last line becomes $(X(2P) : Z(2P)) = (U_1^2U_2^2k_2 : U_3(U_2^2k_2 + k_1U_3)$ with $(k_1 : k_2) = (\mathsf{A} + 2\mathsf{C} : 4\mathsf{C})$.

4.2. **Hybrid arithmetic.** From Section 3.3, if we work in $\theta tw'$ but use the doubling formula in Montgomery to compute $P \to 2P$ instead of the ones of the twisted theta model, we actually obtain $2(P + T_2) + T_2 = 2P + T_2 = 2P + (1 : 0)$ in the $\theta tw'$ model. We can thus compute a translated doubling $2P + T_2$ in $4S + 4m_0$ ($4S + 2m_0$ if we normalize the curve constants), which is interesting if $S < M$ and m_0 is small. We are off by a translation by the point of 2-torsion T_2, but this is easily adjusted to when doing a scalar multiplication by the Montgomery ladder: this does not affect doublings, and for differential additions we just need to track if the base point is P or $P + T_2$.

More generally, given $P, Q, P - Q + T_2$, we can compute $P + Q + T_2$ as follow: $t = (x_P + z_P)(x_Q + z_Q)/(A^2 + B^2)$, $u = (x_P - z_P)(x_Q - z_Q)/(A^2 - B^2)$, $(P + Q + T_2) = ((t + u)^2/x(P - Q + T_2) : (t - u)^2/z(P - Q + T_2))$. This costs $2S + 4M + 2m_0$, $-1m_0$ if the constants are normalised, $-1M$ if the base point $P - Q + T_2$ is normalised.

Applying the formula to $P = Q$, we get the (translated) doubling formula: $t = (x_P + z_P)^2/(A^2 + B^2), u = (x_P - z_P)^2/(A^2 - B^2), (2P + T_2) = ((t + u)^2/A^2 : (t - u)^2/B^2)$, which costs $4S + 4m_0$ ($-2m_0$ if the constants are normalised).

Using these formula, we get the hybrid ladder whose cost is $3M + 6S + 2m_0$.

Likewise, if working in $\theta' tw'$ we use the doubling formula in M to compute $P \to 2P$ instead of the ones of the twisted theta model, we actually obtain $2(P+T_1)+T_1 = 2P+T_1 = 2P + (1 : 0)$ in the $\theta tw'$ model. The doubling formula on M requires $\mathcal{A} = -\alpha - 1/\alpha$ so that $M : y^2 = x(x^2+\mathcal{A}x+1)$; more precisely it requires $(\mathcal{A}+2 : 4) = ((a_0^2-b_0^2)^2 : -(a_0^2+b_0^2)^2)$ so can be computed in $2S + 2a$ from $(a_0^2 : b_0^2)$.

5. 2-isogenies between Kummer lines

5.1. **Standard isogeny formulas.** In the Montgomery model, for an isogeny with kernel $T_2 = (A^2 : B^2) \neq T_1 = (0 : 1)$, the formula is given by [Ren18] $(\mathcal{A}' : 1) = (2(B^4-2A^4) : B^4)$, $(\mathcal{A}' + 2 : 4) = (B^4 - A^4 : B^4)$, and images are given by $(X : Z) \mapsto (X(XA^2 - ZB^2) : Z(XB^2 - ZA^2))$.

In the theta model, the isogeny with kernel $T_1 = (a : -b)$ can be written as follow [Rob23a, § 15.1]. Let $T = (r : s) \in E_0$ be a 8-torsion point above the 4-torsion point $R_1 = (1 : 0)$ which itself is above the 2-torsion point $T_1 = (-a_0 : b_0)$. Then $(A : B) = (r^2 + s^2 : r^2 - s^2)$ so $(a_2 : b_2) = (r^2 : s^2)$. And the isogeny $\theta_{E_1} \to \theta'_{E_2}$ is given by $(x : z) \mapsto (B(x^2 + z^2) : A(x^2 - z^2))$; we need an Hadamard transform to obtain the coordinates in θ_{E_1}.

From these isogeny formula, we can recover isogeny formula on our twisted models and also on the Montgomery model from applying base change. We explain a more general method in Section 5.2.

Let us show how to use the second method, i.e. by base change. To express the isogeny $f : E_1 \to E_2$ in the models $\theta' tw'_{E_1} \to \theta' tw'_{E_2}$ corresponds to writing the isogeny in the models $\theta^2_{E_0} \to \theta^2_{E_1}$. By the above description, the isogeny $g : E_0 \to E_1$ can be written as follow. Let $T = (r : s) \in E_0$ be a 8-torsion point above the 4-torsion point $(1 : 0)$ which itself is above the 2-torsion point $(-a_0 : b_0)$. Then $(a' : b') = (r^2 + s^2 : r^2 - s^2)$ so $(a : b) = (r^2 : s^2)$. And the isogeny $\theta_{E_0} \to \theta'_{E_1}$ is given by $(x : z) \mapsto (b'(x^2 + z^2) : a'(x^2 - z^2))$ so we need an Hadamard transform to obtain the coordinates in θ_{E_1}.

We can use this to describe the isogeny $\theta^2_{E_0} \to \theta^2_{E_1}$. The neutral point on E_1 described by the $\theta^2_{E_0} = \theta' tw'_{E_1}$ is $(a_0^2 : b_0^2)$. The point T above corresponds to a 4-torsion point $T = (r^2 : s^2)$ on $\theta^2_{E_0}$ above the two torsion point $(1 : 0)$ (we can check that in $\theta' tw'_{E_1}$ $(1 : 0)$ corresponds to the 2-torsion point $(-a : b)$ in θ_{E_1}).

Then we can compute $(a : b) = (r^2 : s^2)$, $(a' : b') = (a + b : a - b)$ and the neutral point of E_2 in the $\theta' tw'_{E_2} = \theta^2_{E_1}$ model is $(a^2 : b^2)$ can be computed in $2S$, $2S + 2a$ if we include $(a' : b')$ which will be needed for images.

Let $P = (x^2 : z^2) \in E_1$ in the $\theta^2_{E_0}$ model. The image of P in E_2 in the $\theta^2_{E_1}$ model can be computed as follows: compute $(b'(x^2 + z^2) : a'(x^2 - z^2))$ apply the Hadamard transform and then square the coordinates; this costs $2S + 2M + 4a$.

This gives the isogeny algorithm in the θ^2 model, it is also well known how to compute doublings in this model, see [BRS23] for more details.

A similar approach gives formula in the $\theta tw'$ model. In the $\theta tw'$ model, the isogeny f with kernel by T_1 is given by $(x : z) \mapsto ((x + z)^2/a^2 : (x - z)^2/b^2)$. The neutral point of

E' is then $(a^2 : b^2)$, T_2, T_3 are mapped to $(b^2 : a^2)$, R_1 is mapped to $(1 : 0)$, R'_1 to $(0 : 1)$, R_2, R'_2 to $(1 : 1)$. The dual isogeny $\tilde{f}$ is given by $(x : z) \mapsto ((x + z)^2/A^2 : (x - z)^2/B^2)$.

And the isogeny with kernel T_2 is given by $g : (x : z) \mapsto (((x + z)/a + (x - z)/b)^2 : ((x+z)/a-(x-z)/b)^2)$. We recall that $(a : b)$ can be recovered from R_2. The neutral point is then $g(0) = g(T_2) = (a'^2 : b'^2)$, $g(T_1) = g(T_3) = (b'^2 : a'^2)$, $g(R_1) = g(R'_1) = (1 : 1)$, $g(R_2) = (1 : 0)$, $g(R'_2) = (0 : 1)$.

5.2. **A general framework to derive 2-isogenies between Kummer lines.** Now we want to extend these classical formulas to more general models.

In dimension one, we can work on any model of a Kummer line by specifying its ramification (+ the neutral point). From this data it is easy to recover the action of the theta group $G(20_E)$, and hence compute formula for 2-isogenies between two models. This also allows to obtain doubling formula, and by considering the isogeny: $(P_1, P_2) \mapsto (P_1 + P_2, P_1 - P_2)$ differential addition formula.

If T is a rational two torsion point on our model, we can consider the action g_T of a rational element $g_T \in G(20_E)$ in the theta group above T on the sections $\Gamma(E, 20_E) = \langle X, Z\rangle$. This action is irreducible and faithful, so g_T is completely determined by this action. Then $\lambda = g_T^2 \in \mathbb{G}_m$ is an element, and its class in $k^*/k^{*,2}$ does not depend on the representative, only on T. A small computation shows that this is exactly the (non reduced) Tate pairing $e_{T,2}(T, T)$. The symmetric elements above T are of order exactly 2, so $e_{T,2}(T, T)$ is trivial precisely when these symmetric elements are rational.

We remark that the translation by T is given by a projective homography, which can be determined by the fact that it maps $0 \mapsto T$, $T \mapsto 0$, $T_2 \mapsto T_3$, $T_3 \mapsto T_2$; and we can take for g_T any rational affine lift of this projective translation.

By an homography, we can always send 0_E to $(1 : 0)$ and T to $(0 : 1)$. An element g_T can be given in the form $(X : Z) \mapsto (Z : \lambda X)$, so if $T_2 = (x_2 : z_2)$, $T_3 = (x_3 : z_3)$, so $\lambda = x_3z_3/x_2z_2$, this is well defined in $k^*/k^{*,2}$. Notice that T_2, T_3 are projectively determined only up to an homotety, but this does not change the class of λ.

We can also describe the two points of 4-torsion above T (rember that we are on the Kummer, so $[T' + T] = [T']$ by solving the equation $T' + T = T'$.

From all this discussion, it follows that $\lambda = 1$ iff the symmetric elements $\pm g_T$ are rational iff $e_{T,2}(T, T) = 1$, iff the curve is of Montgomery type when T is sent to $(0 : 1)$ and 0_E to infinity, iff (still if these two points are sent likewise) $T' = (1 : 1)$ is a point of 4-torsion above T, iff there are sections X, Z of $2(0_E)$ such that $g_T(X, Z) = (Z, X)$ iff (by Hadamard) there are sections such that $g_T(X, Z) = (-X, Z)$, iff the quotient E/T is of Legendre type. In particular, if T is of Montgomery type and the full 2-torsion is rational, then E/T has full 2-torsion and the generator of the dual isogeny is of Montgomery type, which is a nice symmetric situation.

Anyway, to study the 2-isogeny with kernel T, $f : E \to E' = E/T$, we need to descend the divisor $4(0_E)$ to $2(0_{E'})$. Since the descent is symmetric, it is given by one of the symmetric element $H_T \in G(4(0_E))$ above T, and it is not hard to prove that it is the symmetric element which is given by $H_T = h_T^{\otimes 2}$ where $\pm h_T$ is any of the two symmetric element above T in $G(2(0_E))$. Although h_T may not be rational, if we have g_T of type λ, then by definition of λ, $H_T = g_T^{\otimes 2}/\lambda$, so H_T is always rational.

It follows that the elements of $\Gamma(2(0_{E'}))$ are the sections of $\Gamma(4(0_E))$ invariant by H_T. Now since we only have $(X, Z) \in \Gamma(2(0_E)) = \Gamma(2(0_E))^+$, we can only construct the even elements $(X^2, XZ, Z^2) \in \Gamma(4(0_E)^+)$ (and these span the space of even elements, the rest of $\Gamma(4(0_E))$ is obtained by adding the odd element YZ). Now take X, Z such that $g_T.(X, Z) =$

$(Z, \lambda X)$ as above, then $H_T = g_T^{\otimes 2}/\lambda$ acts on this basis by $H_T.X^2 = Z^2/\lambda, H_T.Z^2 = \lambda X^2, H_T.XZ = \lambda XZ$. Take a space (U, V) of invariants under this action, and compute the ramification of E' by computing (U, V) on T_2, T_3 and T_1'. This give a model of E', then eventually compose by an homography to make it of nice form. All this gives a very general framework to compute formulas for 2-isogenies between different models of Kummer line.

The nicest case is when T is of Montgomery type, ie $\lambda = 1$. If we put T in position $(0:1)$ and 0_E to $(1:0)$, like in the Montgomery model, then g_T is given by $(X, Z) \mapsto (Z, X)$. We apply the Hadamard change of variable: $(X', Z') = (X+Z, X-Z)$. Via this change of variable, the action of the symmetric g_T is given by $(X', Z') \mapsto (X', -Z')$, so we can take $U = X'^2, V = Z'^2$. This explain why on a Montgomery point, the 2-isogeny images is given by two squares (followed by a nice homography to make E' still of Montgomery type).

There are two ways to ensure that E' is still of Montgomery type. The first one is to ask for T_2, T_3 to be rational; this was handled above. The second one is to ask for a rational point of 8-torsion $T"_1$ above $T_1' = (1:1)$, then $f(T"_1)$ gives a 4-torsion point above $f(T_1')$, hence $f(T_1')$ is still of Montgomery type. Following the above strategy, we get the following isogeny formula: $(X:Z) \mapsto (\gamma(X-Z)^2 : 4XZ)$, with $\gamma = (4rs : (r-s)^2)$ where $T"_1 = (r:s)$. (Recall that $4XZ = (X+Z)^2 - (X-Z)^2$). We have $f(0) = f(T_1) = (1:0), f(T_2) = f(T_3) = (1:-\gamma), f(R_1) = (0:1), f(R_2) = (-\gamma:1)$. We recover formulas from [DJP14].

The reader can check that we can recover all formulas from Section 5.1 this way. From these, we can recover the standard doubling addition and differential addition via Montgomery's formula, they cost $2M+2S+2m_0$ and $4M+2S$ respectively. These cost drop to $2M+2S+1m_0$ if we normalize the constants to have $C=1$ and to $3M+2S$ if the base point $P = (X_P : Z_P)$ is normalised to have $Z_P = 1$.

Example 5.1. Let's explain how to compute an isogeny from a theta model to a Montgomery model when we do not have access to a 8-torsion point. From the theta null point $(a:b)$ of E_1, we can compute $(a^2 : b^2)$ the theta null point of E_2 in the $\theta_{E_1}^2 = \theta' tw'_{E_2}$ model, and the isogeny map is $(x:z) \mapsto (x^2 : z^2)$. Translating by $T_2 = (1:0)$ we obtain the coordinates on the Montgomery model of E_2, with $\mathcal{A}_2 = -\alpha_2 - 1/\alpha_2, \alpha_2 = b^2/a^2$.

5.3. Translated isogenies.

There are two ways to obtain translated isogeny formulas. The first one is to remark that computing isogenies in the $\theta tw'$ model (or $\theta' tw'$ model) is slightly faster than in Montgomery, but doubling is faster in Montgomery. Then we can apply the same strategy as in Section 4.2 and work in (say) the Montgomery model but apply the isogeny formula from $\theta tw'$, which from the point of view of the Montgomery model looks like a translated isogeny.

The second way is to apply the method of Section 5.2, find invariant sections, look at the image ramification, but not translate back to send the isogeneous neutral point to infinity. Since we skip the translation, we get faster formula, but if we work in the new codomain as if the point at infinity was our neutral point, we are off by translation by some point. This method is more generic (it applies to all models), but of course give back the same formula as the first model. We will illustrate both. As an aside, translated isogenies allows to recover translated doublings too (by applying the translated dual isogeny). This gives an alternative way to recover the formula from Section 4.2 directly on a Montgomery model, without going through the change of variable to the twisted theta model.

Let's first look at what happens in the Montgomery model where we use the theta formula for images, using the model $\theta' tw'$. As explained above, the point $(a_0^2 : b_0^2)$ corresponding to the neutral point in the $\theta' tw'$ model now corresponds to a 2-torsion point T_1 in M. We can represent M by this 2-torsion point, for doubling we need $(\mathcal{A}+2:4) = ((a_0^2 - b_0^2)^2 :$

$-(a_0^2 + b_0^2)^2)$ which we can recover in $2S$ from T_1. The equation is given by $M : y^2 = x(x-\alpha)(x-1/\alpha) = x(x^2 + \mathcal{A}x + 1)$ with $\alpha = b_0^2/a_0^2$.

We want to compute the isogeny $E_1 \to E_2$ with kernel T_1. Here E_1 is in a Montgomery model where the full 2-torsion is rational, and we quotient by a 2-torsion point which is different from $(0:1)$, so that the four torsion point $(1:1)$ in E_1 is still of four torsion in E_2 and E_2 still has a Montgomery model.

But we want to represent E_2 via a two torsion point like we did for E_1 (more precisely the two torsion point giving the next kernel). So we need to assume that we have a 4-torsion point $T = (r^2 : s^2)$ above T_1. Then $f(T) = T_1' := (r^4 : s^4)$ on E_2 is the two torsion point we use to represent E_2 and is computed in $2S$. This point T_1' will be the kernel of our next isogeny.

From the Montgomery point of view, given a point $P = (x^2 : z^2)$, then computing $(b'(x^2 + z^2) : a'(x^2 - z^2))$ followed by the Hadamard transform then squaring the coordinates corresponds to computing $f(P) + T_1'$ on E_2 and can be done in $2S + 2M + 4a$. Since T_1' is the kernel of the next isogeny, it does not matter that we translate the image, except at the very last step.

In other words: if on a Montgomery curve we have a point of four torsion T which does lies above a two torsion point $T_1 \neq (0:1)$, then if f is the isogeny with kernel T_1 we can use the coordinates of T to compute $P \mapsto f(P) + f(T)$ in $2M + 2S + 4a$ compared to $4M + 4a$ for computing $f(P)$. And $f(T)$ can be computed in $2S$.

For a 2^n-isogeny, if we select T to be the point giving the next kernel, this extra translation in images does not matter (except at the last step).

Now, we reexplain how to get formula by working on the Montgomery model directly, but using the methods of Section 5.2. Let's look at the isogeny $f : E \to E/T_1$, we have seen that invariant sections are given by $U = (X+Z)^2, V = (X-Z)^2$. Let's look at the images of the ramification and 4-torsion points under $G = (U, V)$. First we apply the Hadamard transform: $H(0_E = (1:0)) = (1:1), H(T_1 = (0:1)) = (-1:1), H(T_2 = (A^2 : B^2)) = (a^2 : b^2), H(T_3 = (B^2 : A^2)) = (-a^2 : b^2), H(R_1 = (1:1)) = (1:0), H(R_1' = (-1:1)) = (0:1), H(R_2 = (a' : b')) = (a:b), H(R_2' = (b' : a')) = (-a : b)$.

It follows that $G(0_E) = g(T_1) = (1:1), G(T_2) = g(T_3) = (a^4 : b^4), G(R_1) = (1:0), G(R_1') = g(R_1 + T_2) = (0:1), G(R_2) = g(R_2') = (a^2 : b^2)$. The ramification on the codomain is $(1:1), (a^4 : b^4), (1:0), (0:1)$, given as the image of $0_E, T_2, R_1, R_1'$. We can scale it to be invariant by $(X : Z) \mapsto (Z : X)$ as in the Montgomery form, the scaling is $(X : Z) \mapsto (b^2 X : a^2 Z)$ and the ramification is then $(b^2 : a^2), (a^2 : b^2), (1:0), (0:1)$. However, the point 0_E is sent to $T_2' := (b^2 : a^2)$. So in summary, if H is the Hadamard transform and S the squaring $(X : Z) \mapsto (X^2 : Z^2)$ transform, and C the scaling transform above, we have that $C \circ S \circ H$ is an isogeny with kernel T_1 between our Montgomery curve, and a curve M'' that has the same ramification as a Montgomery curve M' except the neutral point is $T_2' = (b^2 : a^2)$. So the full isogeny, if we want to work on M' rather than M'', is given by translating by T_2'; in other word $C \circ S \circ H : M \to M'$ gives the isogeny translated by $T_2' = f(R_1)$.

In summary, applying the above method, we get the following formulas.

The (translated) isogeny $f : E \to E/T_1$ is given by, if $P = (x : z), f(P + R_1) = ((x+z)^2/a^2 : (x-z)^2/b^2)$, with $(a^2 : b^2) = (A^2 + B^2 : A^2 - B^2)$. Notice that $E' = E/T_1$ is still a Montgomery curve with full rational 2-torsion, so there is a perfect symmetry between E and E'. We have $T_2' = f(R_1) = (b^2 : a^2), f(0) = f(T_1) = (1:0) = 0, f(T_2) = f(T_3) = (0:1) = T_1', f(R_1) = T_2' = (b^2 : a^2), f(R_2) = (a^2 : b^2) = T_3'$.

The (translated) dual isogeny with kernel T_1' is given by $\tilde{f}(P + R_1') = \tilde{f}(P) + T_2 = ((x+z)^2/A^2 : (x-z)^2/B^2)$. Composing $\tilde{f} \circ f$ we recover the (translated) doubling formula $P \mapsto 2P + T_2$ as above. We have $\tilde{f}(0) = \tilde{f}(T_1') = (1 : 0), \tilde{f}(T_2') = \tilde{f}(T_3') = T_1, \tilde{f}(R_1') = T_2 = (A^2 : B^2), \tilde{f}(R_2') = (B^2 : A^2) = T_3$.

Now, let $g : E \to E_2 = E/T_2$ be the isogeny with kernel T_2. Since we want E_2 to be Montgomery with full rational two torsion, we need a point $S_2 = (a' : b')$ above T_2.

The isogeny g is then given by $g(P + S_2) = (((x+z)/a + (x-z)/b)^2 : ((x+z)/a - (x-z)/b)^2)$, with $(a : b) = (a' + b' : a' - b')$. (Remark that $(A^2 : B^2) = (a^2 + b^2 : a^2 - b^2) = (a'^2 + b'^2 : 2a'b')$). The curve E_2 is represented by its two torsion point $T_2' = g(S_2) = (a'^2 : b'^2)$. The codomain computation costs $2S$, and a translated image $2S + 2m_0$.

We have $g(0) = g(T_2) = 0, g(T_1) = g(T_3) = T_1' = (0 : 1), g(R_1) = g(R_2) = R_2' = (1 : -1), g(R_1 + S_2) = R_1' = (1 : 1), g(S_2) = T_2' = (a'^2 : b'^2), g(S_2 + T_1) = g(S_2 + T_3) = (b'^2 : a'^2) = T_3'$. The dual isogeny $\tilde{g}$ has kernel $T_1' = (0 : 1)$ and is given by $\tilde{g}(P) = (B^2(x+z)^2 : 4A^2xz)$. (Notice that $4xz = (x+z)^2 - (x-z)^2$ so $\tilde{g}$ can be computed in $2S + 2m_0$.) We have $\tilde{g}(0) = \tilde{g}(T_1') = (1 : 0), \tilde{g}(T_2') = \tilde{g}(T_3') = T_2, \tilde{g}(R_1') = T_3, \tilde{g}(R_2') = T_1$. The composition $\tilde{g} \circ g$ gives an alternative formula to compute $P \mapsto 2P + T_2$ in $4S + 2m_0$.

5.4. **Theta versus Montgomery.** To summarize, the complexities for computing isogenies in the theta model are as follows:

(1) $2S + 2a$ for the codomain
(2) $2S + 2M + 4a$ for an image
(3) $4S + 4M + 8a$ for doubling

The input is the theta null point $(a : b)$, which implicitly contains the 2-torsion point $(-a : b)$ used for our kernel; and the images computations needs (some constants computed during) the codomain.

In the Montgomery model, the costs are, using [CLN16; CH17; Ren18]:

(1) $2S + 1a$ for the codomain
(2) $4M + 4a$ for an image (using a precomputation of $2a$)
(3) $2S + 4M + 4a$ for doubling

Here the input is a two torsion point (different from $(0 : 1)$) giving the kernel (and implicitly the curve); the image computation does not needs the codomain.

We see that the theta model is slightly faster then the Montgomery model except for doublings. Using hybrid isogenies allows to combine the best of both models.

To sum up, we can, provided we have a point of 4-torsion T above our kernel $\langle T_2 \rangle$:

(1) Compute a representation of the codomain in $2S$. The representation is given by the 2-torsion point $f(T) = T_2$, which is the kernel of the next isogeny.
    If we need to compute doublings on the codomain, we need to add a $2S + 2a$ precomputation to compute $(\mathcal{A} + 2 : 4)$, and if we need to compute images we need to add a $2a$ precomputation (which is already done if we did the previous $2S + 2a$ precomputation needed for doublings).
(2) Compute "images" in $2M + 2S + 4a$.
(3) Compute "doublings" in $4M + 2S + 4a$.

The words "images" and "doublings" are in quotes because if we consider that we are on a twisted theta models the "doublings" we compute are actually $2P + T_2$, while if we consider that we are in the Montgomery model it is the images that are actually given by $f(P) + T_2$.

The images need some of the constants computed for the codomain. In both cases, this translation is by an element of the next kernel, so does not affect the rest of the computation.

We conclude this with a discussion on 4-isogenies. On the Montgomery model, a 4-isogeny can be computed in [CH17]:

(1) $4S + 5a$ for the codomain
(2) $6M + 2S + 6a$ for images.

Here the input for the codomain is given by the coordinates of a 4 torsion point T, and the input for the images needs some of the constants for the codomain.

If the kernel is given by $K = \langle T \rangle$, we can also look at the cost of decomposing this 4-isogeny as a 2-isogeny where we exploit the 4-torsion point followed by a standard 2-isogeny formula in the Montgomery model:

(1) $4S + 3a$ for the codomain
(2) $6M + 2S + 8a$ for images.

It is probable (but I haven't checked) that we actually obtain essentially the same formula as the 4-isogeny algorithm above, except 2 of the additions needed for images could be moved to a precomputation done in the codomain computation. So the standard 4-isogeny formula can essentially be interpreted as alternating the $2M + 2S$ isogeny formula with the $4M$ isogeny formula. But if we have a 2^n-isogeny to compute, we might as well keep using the $2M + 2S$ formula, except at the very end where we use the $4M$ formula to not be off by translation by a point of 2-torsion (and we might not have a 4-torsion point available anymore anyway).

Remark 5.2. Decomposing a 2^n-isogeny via 2-isogenies or 4-isogenies. While the above formula for 2-isogenies in the Montgomery model are fun to look at, they are not really useful in practice: it is better to decompose a 2^n-isogenies as a sequence of 4-isogenies rather than as a sequence of 2-isogenies. The reason is that the decomposition algorithm is quasi-linear, if we split into blocks of 2^m-isogenies, we gain a bit more than m images and doublings because of the quasi-linearity. Usually this is not interesting because a 2^m-isogeny costs $O(2^m)$ to compute, so is 2^{m-1} more expansive than a 2-isogeny but $2^2 = 4$ hits the sweet spot for an optimal decomposition time. Once an isogeny is decomposed, for a 2^n-image, using the slightly faster 2-isogeny images rather than 4-isogenies would be better however.

6. Time-Memory trade off for the arithmetic

I found these formula (first for the theta model) in December 2022, while working with Barbulescu and Sarkis on models of Kummer lines.

6.1. **Overview.**

Although the Montgomery ladder is very efficient, for fast scalar multiplication the twisted Edward model is often faster because it allows for a time/memory trade off by using the window-NAF method to reduce the amount of additions.

However, when the scalar is a secret, these time/memory trade off are often susceptible to side channel attacks, so although signing on Curve25519 is implemented in Edwards coordinate, the DH key exchange uses the Montgomery ladder.

It might seem that a time/memory trade off is not possible on a Kummer line because standard additions are not available. A way to precompute the Montgomery ladder was presented in [OLHFR18]. When T_2 is rational, we present a novel approach to precompute the ladder that:

(1) does a precomputation of points $P_i = (X_i : Z_i)$ costing $2S + 1m_0$ by bit, and requiring to store two field coefficients by bit.

(2) using this precomputation, the ladder then costs $2S + 1m_0$ for doubling, and $4M$ for a differential addition by bit.

The total cost, including the precomputation, is thus of $4S + 4M + 2m_0$, and further scalar multiples with the same base point then cost $2S + 4M + 1m_0$. Here it does not matter whether $P = (X_P : Z_P)$ is normalised or not.

We stress that the scalar multiplication still uses a ladder approach, with one doubling and one differential addition by bit, thus retaining the same side channel resistance as the standard Montgomery ladder. (We recall that the Montgomery ladder without precomputation costs $5M + 4S + 1m_0$ when the base point P is normalised, and $6M + 4S + 1m_0$ if P is not normalised.)

If we know that P will be used several time (like for the first step of a DH key exchange), we can increase the precomputation to normalise the points $P_i = (X_i/Z_i : 1)$. This costs one field division by bit, and reduces the storage to one field coefficient by bit. We can batch the inversions, to replace the one division by bits by one global inversions and $3 + 1 = 4$ multiplications by bit.

The precomputation is then one global inversion, and $4M + 2S + 1m_0$ by bit (the storage drops to one coefficient by bit). The multiples $m.P$ then cost $2S+3M+1m_0$ by bit, significantly improving on the standard ladder.

Unfortunately, for Curve25519 the point T_2 is not rational. But its 2-isogeneous curve is a Montgomery curve with full rational 2-torsion, so by computing an isogeny at the beginning and the end we can still use our novel time/memory trade off on Curve25519 (however, unlike Curve25519, the curve constant on the isogeneous curve is not small, so we don't gain as much as if we had selected from the beginning a suitable curve with a small m_0).

A similar algorithm works in higher dimension. For a Kummer surface the precomputation step costs one global inversion and $12M + 4S + 3m_0$ by bits, and then a scalar multiplication with the same base point costs $7M + 4S + 3m_0$; compared to $7M + 12S + 9m_0$ or $10M + 9S + 6m_0$ for the standard ladder.

Now let us compare with the results of [OLHFR18]. Their idea is to use a right to left Montgomery ladder rather than the usual left to right ladder. The right to left ladder always involve the points $P_i = 2^i P$ (where P is the base point), so these can be precomputed. (Our approach is related: we remark that we can factor doublings and differential additions through 2-isogenies to get half doublings and half differential additions. We permute the order: rather than at each step, compute an image through a 2-isogeny and then do a half differential additions, we precompute all images and then only do half differential additions for the ladder).

With these precomputations done, the right to left Montgomery ladder then only needs differential additions, except that the difference is not fixed anymore. So a priori we need a full DiffAdd at each step rather than a mDiffAdd. In [OLHFR18], the authors explain how to extract from the P_i a coordinate μ_i (this requires a division) which can be used to get a DiffAdd formula (involving P_i) in $3M + 2S$, exactly like the mDiffAdd.

Their precomputation cost is then the cost of one doubling by bit (to compute the $2^i P$, aka $2M + 2S + 1m_0$ and one division, so batching inversions we get a precomputation cost of one global division and $6M + 2S + 1m_0$ by bit. The scalar multiplication is then like the standard ladder, except all doublings have been removed and only differential addition remains, so it costs $3M + 2S$ by bit. Without the computation of the μ_i, the precomputation would be $2M + 2S + 1m_0$ and the multiplication cost $4M + 2S$ by bit.

We compare two cases. We recall that the standard Montgomery ladder costs $5M + 4S + 1m_0$ by bit when P is normalised.

- We only do a light precomputation without inversions. By bit, our ladder requires a $2S + 1m_0$ precomputation, followed by a $4M + 2S + 1m_0$ for multiplication. The ladder in [OLHFR18] requires a $2M + 2S + 1m_0$ precomputation, followed by $4M + 2S$ for multiplication.
- We allow a global inversion in the precomputations. By bit, our ladder requires a $4M + 2S + 1m_0$ precomputation, followed by a $3M + 2S + 1m_0$ for multiplication. The ladder in [OLHFR18] requires a $6M + 2S + 1m_0$ precomputation, followed by $3M + 2S$ for multiplication.

We see that in both cases, our precomputation is smaller, but we pay for it by needing an extra m_0 in the multiplication step. However, an important point is that our light precomputation is so cheap that even including it we are only at $4M + 4S + 2m_0$, which gains $1M - 1m_0$ compared to the Montgomery formula (and $2M - 1m_0$ if the base point is not normalised).

Furthermore, for Kummer surfaces, our precomputation and multiplication are both faster than an equivalent approach as [OLHFR18] would provide.

6.2. **Explicit formula.** In the theta model, the arithmetic ladder stems from the duplication formula (see [Rob23a]): $\theta_E(P+Q) \star \theta_E(P-Q) = H(\theta'_{E'}(f(P)) \star \theta'_{E'}(f(Q)))$.

The ladder use two steps for the differential addition (doubling is a special case where $P - Q = 0$): compute $f(P)$ via $\theta_E(P) \star \theta_E(P) = H(\theta'_{E'}(f(P)) \star \theta'_{E'}(f(0)))$. This costs $2S + 1m_0$. Do the same for $f(Q)$. Then use $\theta_E(P+Q) \star \theta_E(P-Q) = H(\theta'_{E'}(f(P)) \star \theta'_{E'}(f(Q)))$ to compute $(P+Q) \star (P-Q)$ in $2M$, and then $P+Q$ in again $2M$ (or $1M$ if $P-Q$ is normalised).

A large part of the ladder is hence spent in isogeny images. Let $f_1 = f, f_2 = \tilde{f} \circ f_1, f_3 = f \circ f_2, f_4 = \tilde{f} \circ f_3$ and so on. Assume we had $f_{i+1}(nP), f_{i+1}((n+1))P$. Then from the duplication formula, we could directly find $f_i(2nP), f_i(2(n+1)P), f_i((2n+1)P)$.

The doublings only require the points $f_i(0_E)$ which are given by the two curves E and E'. However the differential addition needs $f_i(P)$. So what we can do is compute $f_i(P), f_i(0_E)$ then apply our duplication formula. This inverse the order: rather than doing two isogeny images and two duplication at each step, we compute all the images first and then do all the duplications. We gain because the images $f_i(0_E)$ are free. We could expect to gain $2S + 1m_0$, but because our points $f_i(P)$ are no longer normalised, we only gain $2S + 1m_0 - M$ compared to the normal ladder with a normalised P.

In summary: we do a precomputation phase with all the $f_i(P)$. This cost $2S + 1m_0$ by bit, along with 2 field coefficients. Then we do our duplication formula: this cost $2S + 1m_0$ for our doublings, and $4M$ for our differential additions (again, because the $f_i(P)$ are not normalised). The final cost including the precomputation is $4M + 4S + 2m_0$. Further multiplication with the same base point P will cost $4M + 2S + 1m_0$. We note that this cost is the same whether P is normalised or not (because even if P is normalised, the $f_i(P)$ won't be).

When we know in advance P will be used (for public key encryption, or the first phase of DH key exchange), it is worth it to normalise the $f_i(P)$ at the cost of $1I$ by bit (the storage is then 1 coeff by bit). Then scalar multiplication will cost $3M + 2S + 1m_0$.

The big advantage compared to other time/memory trade off with elliptic curves (naf, window, ...) is that the scalar multiplication is still a ladder with a double and diff add by bit, hence much less susceptible to side channel attack.

The same principle apply to the twisted theta model $\theta tw'$, by using the linear change of variable from the theta model, but we need some careful translation by $f_i(T_2)$ to gain $1M$ at each step (essentially we use a trick similar to the hybrid ladder): for the differential addition we assume that we have $f_{i+1}(nP), f_{i+1}((n+1)P + T_2)$ (say) and we compute $f_i((2n+1)P + T_2)$. (Doublings are no problem). We obtain the same cost as in the θ model, except the initial translation by the two torsion point; likewise in the Montgomery cap Legendre model.

The formula are as follow (pending typos, I recommend to look at the code in [Rob23e] instead to be sure to use correct formulas...): given $(x_{Pi} : z_{Pi})$, the isogenous point P_{i+1} is given by: $X = (x_{Pi}^2 + z_{Pi}^2)b_i^2 : (x_{Pi}^2 - z_{Pi}^2)a_i^2)$. From P_{i+1} we can compute $2P_i$ via the dual isogeny: $(X+Z)^2 b_{i+1}, (X-Z)^2 a_{i+1})$. The more interesting part is the differential addition, given $P_{i+1} = (xg_P : zg_P), Q_{i+1} + T_{2i+1} = (xg_{Q'} : zg_{Q'}), (P-Q)_i = (x_{PQ} : z_{PQ})$ we recover $(P+Q)_i$ via: $s = (xg_P + zg_P)(xg_{Q'} + zg_{Q'}); t = (xg_P - zg_P)(xg_{Q'} - zg_{Q'})$; $u = s+t; v = s-t; X = u/(x_{PQ}+z_{PQ}); Z = v/(x_{PQ}-z_{PQ}); (P+Q)_i = (X+Z : X-Z)$.

For Curve25519, since the two torsion is not rational, we need to move via a 2-isogeny to the curve above it which is both Montgomery and has full rational two torsion. Unfortunately the constant is large, so the cost of $4M + 4S + 2m_0$ when including the precomputation is essentially the same as with a standard Montgomery ladder: $5M + 4S + 1m_0$ (assuming P is normalised; we gain $1M$ on a non normalised point). Still, with the normalised precomputation, the cost of $3M + 2S + 1m_0$ is still very interesting, even with a large m_0.

The reason we work to work on the Montgomery cap Legendre model, is that if we want the relations $x(P+Q)z(P+Q), x(P-Q)z(P-Q)$ to factor through the isogeny f with kernel a 2-torsion point T, we need T to be of Montgomery type (equivalently the Tate pairing $e(T,T) = 1$, or the symmetric element in the theta group above T is rational). So the curve needs to be Montgomery, but the isogeneous curve should be too (because we go back and forth between the two curves), which is equivalent to the starting curve being in Legendre form.

6.3. **A general framework to find differential additions.** We can extend our general isogeny framework from Section 5.2 to differential additions. In this case we study the isogeny $\xi : (P_1, P_2) \to (P_1+P_2, P_1-P_2)$, the pullback $\xi^*(2(0_{E'}) \star 2(0_{E'}) = 4(0_E) \star 4(0_E)$. The kernel is given by the diagonal embedding of $E[2]$, and the symmetric lift giving our divisor descent is given by $h_T \otimes h_T$ for $T \in E[2]$ and h_T any of the two symmetric lift above T; even if h_T is not rational the tensor product is. We can thus compute the actions on $\Gamma(2(0_E) \star 2(0_E))^{\otimes 2}$, these span the even elements of $\Gamma(4(0_E) \star 4(0_E))$, of dimension 9. It follows that on $E' \times E'$ we can only construct the even elements in $\Gamma(2(0_E) \star 2(0_E))^+$, where the involution is given here by the descent of $(P_1, P_2) \to (P_1, -P_2)$, in other words we can only express elements invariant under the involution $P_1 + P_2 \mapsto P_1 - P_2$: $x(P_1+P_2)x(P_1-P_2)$, $z(P_1+P_2)z(P_1-P_2)$, $x(P_1+P_2)z(P_1-P_2) + z(P_1+P_2)x(P_1-P_2)$.

Now, it is useful to factorize doubling through the isogeny f with kernel T: $[2] = \tilde{f} \circ f$, and we want to do the same with differential additions. In other words, we have $f(P_1), f(P_2)$ and we want to find from this some functions involving $P_1 + P_2, P_1 - P_2$. So consider $F : E \times E \to E' \times E'$ given by the diagonal of f. Then $\ker F = \{(0,0), (0,T), (T,0), (T,T)\}$; notice that it is not included in $\ker \xi$, so ξ does not factorize through F. So we need to consider the pushforward C of F and ξ of kernel $\operatorname{Ker} F + \operatorname{Ker} \xi$. From $f(P_1), f(P_2)$, we can not express all even functions in $P_1 + P_2, P_1 - P_2$, but we can only obtain those that descend to C, ie are invariants by $(0,T), (T,0)$ (or more precisely $1 \otimes h_T$ and $h_T \otimes 1$). It is instructive to look at the case $\lambda_T = 1$, in which case $x(P_1+P_2)x(P_1-P_2), z(P_1+P_2)z(P_1-P_2)$ are invariants. This explain the form the differential addition formula take for a Montgomery point.

As explained in Section 6.2, when doing a Montgomery ladder, we can then use a cycle of f and $\hat{f}$ to interleave the order of isogenies and differential additions: rather than computing images and differential additions (or doublings) at each step, we can compute iterated image, and then compute iterated differential additions (and doublings). The advantage is that in the standard ladder we compute two images at each step, while here we only need to compute the iterated image of our base point, so we gain one image. On the other hand we need memory to store our iterated images, and we cannot assume that these images are normalised (unless we do a normalisation step on our points at the end). So to compute $N.P$ we compute $\log N$ iterated images of P (one by bit), then we do a ladder doing one doubling-through-isogeny, differential-addition through isogeny by step. This gives a general time/memory trade off for the Montgomery ladder.

Since we need to use differential addition formula that factorize through both f and $\tilde{f}$, the best case is thus when both are given by kernels of Montgomery type, ie our starting curve is in Montgomery cap Legendre.

7. Pairings

On a Kummer line, it is useful to interpret pairings as coming from the biextension law [Gro72; Sta08] associated to the divisor $2(0_E)$. It is shown in [Gro72] how the biextension gives rise to the Weil pairing, and [Sta08] extends this to the Tate pairing.

In this section I only give a very brief overview of the algorithm, and refer to the talk [Rob23b] for a bit more details.

For the biextension X associated to the divisor (0_E), an element $g_{P,Q}$ corresponds to a function on $k(E)$ with divisor $(P)+(Q)-(P+Q)-(0_E)$. The biextension partial group laws are given by:

$$(g_{P_1,Q} \star_1 g_{P_2,Q})(R) = g_{P_1,Q}(R) g_{P_2,Q}(R-P_1)$$
$$(g_{P,Q_1} \star_2 g_{P,Q_2})(R) = g_{P,Q_1}(R) g_{P,Q_2}(R) \frac{g_{Q_1,Q_2}(R-P)}{g_{Q_1,Q_2}(R)}$$

Moreover, since the divisor is symmetric, the biextension is symmetric too: $g_{P_1,Q} \star_1 g_{P_2,Q} = g_{Q,P_1} \star_2 g_{Q,P_2}$. This implies: $\mu_{P_1,P_2}(-P_3) = \mu_{P_2,P_3}(-P_1) = \mu_{P_3,P_1}(-P_2)$. A convenient way to represent a biextension element $g_{P,Q}$ is via (P,Q) and its evaluation on some point R. The group law becomes

$$(Q,P_1,c_1) \star_2 (Q,P_2,c_2) = c_1 c_2 \frac{g_{P_1,P_2}(R-Q)}{g_{P_1,P_2}(R)},$$

and in particular we have:

$$g_{Q,P}^{\star_2,\ell} = g_{Q,P}(R)^\ell f_{\ell,P}((R-Q)-(R)),$$

where $\operatorname{div} f_{\ell,P} = \ell P - (\ell P) - (\ell-1)(0_E)$. Thus Miller's algorithm is simply the biextension exponentiation via this representation (and taking $R=0_E$). We also have the following variant, using the symmetry:

$$(Q,P_1,c_1) \star_2 (Q,P_2,c_2) = c_1 c_2 \mu_{P_1,P_2}(-Q) = c_1 c_2 \mu_{P_1,Q}(-P_2),$$

which give the following alternative formula for the Miller addition:

$$f_{m+1,P}(-Q) = f_{m,P}(-Q)\mu_{mP,P}(-Q) = f_{m,P}(-Q)\mu_{P,Q}(-mP).$$

In particular, the biextension arithmetic gives the Tate and Weil pairing. Let $g_{P,Q} \in X(\mathbb{F}_q)$ be any element above (P,Q), $P \in E[\ell]$. since $\ell P = 0$, $g_{P,Q}^{\star_1,\ell}$ is a constant λ_P. If $\mu \in \mathbb{G}_m(\mathbb{F}_q)$ and $g'_{P,Q} = \mu \cdot g_{P,Q}$, then $g'^{\star_1,\ell}_{P,Q} = \mu^\ell \lambda_P$, and the class of λ_P in $\mathbb{F}_q^*/\mathbb{F}_q^{*,\ell}$ is the non reduced Tate pairing. Furthermore, $g_{P,Q}^{\star_1,q-1} = \lambda_P^{(q-1)/\ell}$ is the reduced Tate pairing $e_{T,\ell}(P,Q)$; it does not depends on the choice of $g_{P,Q}$. If $Q \in E[\ell]$, $g_{P,Q}^{\star_2,\ell} = \lambda_Q$; the Weil pairing is given by

$e_{W,\ell}(P,Q) = \lambda_P/\lambda_Q$. We also have similar formulas for the Ate and optimal Ate pairing, see [Rob23b].

We have the following monodromy interpretation of the pairings. The non reduced Tate pairing is then given by $g^{\ell}_{P,Q}$, which can be computed from $\widetilde{\ell P}$, $\widetilde{\ell P+Q}$, which in turn can be computed from a three way affine Montgomery ladder: 1 affine doubling and 2 affine differential addition by step. Equivalently, the non reduced Tate pairing is given by comparing $g^{\ell+1}_{P,Q}$ with $g_{P,Q}$, they differ by a projective factor λ_Q which is precisely the pairing. This λ_Q can be interpreted as a monodromy action: $Q \mapsto (\ell+1)Q$ is trivial at the level of the elliptic curve, but not at the biextension level. Likewise, the Weil pairing is given by the quotient of monodromy λ_Q/λ_P. The reduced Tate pairing is given by comparing $g^{q}_{P,Q}$ with $g_{P,Q}$, since $q-1$ is divisible by ℓ in our pairing situations, this is indeed the same as raising the non reduced Tate pairing to the power $(q-1)/\ell$. From this point of view the reduced Tate pairing is the Weil-Cartier pairing associated to $\pi_q - 1$.

For our efficient formulae, rather than using the Miller representation of the biextension elements, we will use the cubical torsor structure. We refer to [Bre83; Mor85] for cubical torsors.

We can indeed reinterpret the biextension law as follow: the key point is that with a symmetric line bundle, there is a *canonical* isomorphism $t_P^*L \otimes t_Q^*L \otimes t_R^*L \otimes t_S^*L \simeq t_U^*L \otimes t_V^*L \otimes t_W^*L \otimes t_X^*L$ whenever $P+Q+R+S = 2Z$, $U = Z-P$, $V = Z-Q$, $W = Z-R$, $X = Z-S$.

Specialising, we get partial group law on trivialisations of line bundle: $\tilde{0}, \widetilde{P}, \widetilde{Q}, \widetilde{P-Q} \mapsto \widetilde{P+Q}$, $\tilde{0}, \widetilde{P}, \widetilde{Q}, \widetilde{R}, \widetilde{P+Q}, \widetilde{P+R}, \widetilde{Q+R} \mapsto \widetilde{P+Q+R}$. Technically, these relations give the cubical torsor structure, which is a refinement of the arithmetic in the biextension.

(Note: in [Sta08] the biextension appears in the guise of elliptic nets. From our point of view, we can reinterpret elliptic nets as trivialisation of the line bundle $D = (0_E)$ at points P, notably by specifying the value of $Z(P)$ where Z is the section of (0_E). A slight difficulty is that Z has a zero on 0_E, so we need some offset to compute the pairings. The remarkable thing about elliptic nets is that even through we are on level 1 we can still compute the arithmetic of biextension through the linear recurrence of elliptic nets, see [Sta08] for details.

In [LR10; LR15], the biextension is hidden through the guise of the analytic Riemann relations giving the transcendental group law.)

We then represent an element $g_{P,Q}$ of the biextension by the trivialisations $\tilde{x}, \widetilde{x+P}, \widetilde{x+Q}, \widetilde{x+P+Q}$. Changing the trivialisations by $\lambda_x, \lambda_P, \lambda_Q, \lambda_{P+Q}$ give the same element iff $\lambda_x\lambda_{P+Q} = \lambda_P\lambda_Q$ (So our affine lifts represent a cubical torsor structure, and the associated biextension element is an equivalence class under this action).

The affine doublings and affine differential additions are formula lifting the standard projective doublings and projective differential additions. When working on the biextension we have more leeway, but when working on the cubical torsor structure we must be careful to use the correct affine formulas. We refer to the implementation in [Rob23e] for the explicit formula.

In the theta or twisted theta model, using [LR10; LR15] this amount to $7S + 7M + 2m_0$ by bit, assuming our base points are normalised (else add $2M$ by bit). On the Montgomery model, the biextension ladder costs $8M + 6S + 1m_0$ by bit.

By comparison, the best formula I have found for generic pairing computations in the Jacobian model cost $10M + 9S$ for doubling, and $11.5M + 3S$ by addition [BELL10].

In certain cases, we can compute the biextension exponentiation faster:

- In the general case, we compute $g^{\ell}_{P,Q}$ via one affine doubling and two affine differential additions by bits, for a total cost of $8M + 6S + 1m_0$ in the Montgomery model;
- For a self pairing, $P = Q$, we only need one affine doubling and one affine differential addition, for a total cost of $5M + 4S + 1m_0$ by bits. (A word of warning: for a fast self pairing we really need to use the cubical arithmetic rather than just the biextension arithmetic).
- When $n = 2^m$ is a power of 2, we also need only one affine doubling and one affine differential addition, for a total cost of $5M + 4S + 1m_0$ by bits.
- When $n = 2^m$ and $P = Q$, we only need one affine doubling, for a total cost of $2M + 2S + 1m_0$ by bits.

We can also do a standard exponentiation on $g_{P,Q}$ on our biextension, this allows to use the standard NAF and windowing method. We can do additions on the biextension model (at least with our representation), even through we are on the Kummer line on the underlying curve!

I worked out the formula in the theta model, using [LR15; LR16]: doubling cost 1 double and 1 diff add on the underlying curve, for a cost of $4M + 5S + 2m_0$. Addition is more complicated: on the underlying curve this amount to one (projective) compatible addition which cost $27M$ (I am not distinguishing M, S and m_0 here), followed by an affine three way addition which cost $17M$, for a grand total of $44M$. But since our base points are always the same (the ones we computed for our window), we can do some precomputations for these steps, and the compatible addition then cost $17M$, and the three way addition $13M$, for a total of $30M$.

Since doubling is $11M$, this might be competitive with the ladder method (which costs $16M$ by bit) when using a NAF-window with $w \geq 5$.

Remark 7.1. When working on the Kummer line, we are naturally working with the biextension associated to the divisor $2(0_E)$ rather than (0_E), because our coordinates $X, Z \in \Gamma(2(0_E))$. The corresponding biextension monodromy gives thus the square of the usual Tate and Weil pairing; which is no problem when ℓ is odd. This however lose one bit of information when ℓ is even; luckily in this case we can use the natural action of the theta group $G(2(0_E))$ on $\Gamma(2(0_E))$ to recover the Weil and Tate pairings exactly rather than just their squares. Once again we refer to [Rob23e] for the formulas.

7.1. The Tate pairing for pairing based cryptogrpahy.

For pairing based cryptography on elliptic curves, it is convenient to use the Tate pairing with $P \in \mathbb{G}_1 \subset E(\mathbb{F}_q)$, $Q \in \mathbb{G}_2 \subset E(\mathbb{F}_{q^k})$, and k even to allow for denominator elimination.

Counting only operations involving the big field $\mathbb{F}_{q^k}$, Miller's algorithm cost $1M+1S+1m$ by doubling, and $1M+1m$ by addition. Here $1m$ denotes a multiplication between a coefficient in $\mathbb{F}_q$ and a coefficient in $\mathbb{F}_{q^k}$.

When denominator elimination is not possible (because k is odd or Q is not in $\mathbb{G}_2$), the cost becomes $2M + 2S + 1m$ by doubling, and $2M + 1m$ by addition.

Using our arithmetic of biextension on Kummer lines, only counting the operations on the big field, we have $2S + 1M + 2m$ by bit. So better than Miller's algorithm, except when denominator elimination is available.

7.2. Monodromy leak.

It is well known, when $\mu_\ell \subset \mathbb{F}_q$, that the Tate pairing allows to reduce the DLP from an elliptic curve to $\mathbb{F}_q^*$.

From the point of view of étale torsors [Rob23g], the Tate pairing $e_{T,\ell}(P,Q)$ is an isomorphism class of the torsion $f^{-1}(Q)$ where f is the isogeny of kernel $\langle P\rangle$.

When $\mathbb{F}_q$ does not contains μ_ℓ (say ℓ prime for simplicity), a torsor is always trivial, ie $f^{-1}(Q)$ always contains one rational point.

However, we can still recover informations on the DLP if we manage to track explicit isomorphisms between $f^{-1}(Q)$ and $f^{-1}(n.Q)$. The theta group is precisely calibrated to keep track of such isomorphisms. Theta groups and biextensions are closely related (we will explore this topic further in [Rob23d]), and in this section we explore how to use biextensions to attack the DLP. (Very similar ideas were already pursued via elliptic nets in [LS08].)

We assume from now on that we are in this case.

The general idea is as follows: the biextension arithmetic is a juxtaposition of arithmetic on the underlying elliptic curve and in $\mathbb{F}_q^*$. When computing an exponentiation $n \mapsto n.P$, leaking instead a biextension exponentiation $n \mapsto g_P^n$ allows to recover n via a DLP in $\mathbb{F}_q^*$.

It might seem hard to leak such a biextension exponentiation on purpose, but from the pairing formula we see that since we are naturally working on affine coordinates, and the natural affine additions formulas are the ones coming from the biextension arithmetic, we see that on the contrary doing a Montgomery ladder leaks the biextension exponentiation as long as we don't randomize the coordinates (X,Z) by a factor $(\lambda X, \lambda Z)$ or we don't output the division $x = X/Z$.

There are two versions of the projective coordinates leak. The key idea is as follow: from our assumptions there is a unique lift $\widetilde{g_P}$ of P in the biextension that is still of order ℓ. This "canonical lift" can be computed efficiently by a scalar multiplication in the biextension, this scalar being determined by being 0 modulo $p-1$ and 1 modulo ℓ.

We now start with $P = (x_P, 1)$ corresponding to some $g_P = \lambda_1 \widetilde{g_P}$, and with overwhelming probability λ_1 is not trivial (we use g_P as a shortcut for the biextension element associated to $g_{P,P}$). The value of g_P^n is leaked, which gives us $g_P^n = \lambda_2 \widetilde{g_{nP}}$. But since $\widetilde{g_{nP}} = \widetilde{g_P}^n$, we get that $\lambda_2 = \lambda_1^n$. From λ_2, λ_1, we recover n via a DLP in $\mathbb{F}_q^*$. This version requires g_P^n, so a leak of both $\widetilde{nP}$, $(\widetilde{n+1})P$. Furthermore, the biextension arithmetic is slightly different from the way the Montgomery ladder is implemented in practice, so we need to do some slight adjustments (see below).

The stronger version of the projective coordinate leak only requires a leak of $\widetilde{nP}$. This time we need the full power of the cubical torsor structure rather than just of biextension; there is still a unique (using our assumption that ℓ is prime to $q-1$) canonical lift $\widetilde{P}$ which is of ℓ-torsion and can be efficiently computed from P. So we start with $P = (x_P, 1) = \lambda_1 \widetilde{P}$ and we are leaked $n.P = (X,Z) = \lambda_2 \widetilde{nP}$. This time, we have $\lambda_2 = \lambda_1^{n^2}$, so we need a DLP and then solve a square root.

However, taking into account that the actual Montgomery ladder is different from the exact cubical torsor structure arithmetic, we need to correct by some factor, so we actually solve a more general degree two polynomial.

Explicitly, computing $n.P$ via the standard ladder arithmetic rather than via the correct cubical ladder, we are off by a factor $(4x_P)^{n(2^b-n)}$ where b is the bit length of n. Taking a multiplicative generator ζ of F_q^*, we thus need to solve the equation:

$$X^2(\operatorname{dlp}_\zeta(\lambda_1) - \operatorname{dlp}_\zeta(4x_P)) + X2^b \operatorname{dlp}_\zeta(4x_P) - \operatorname{dlp}_\zeta(\lambda_2) = 0. \tag{16}$$

The number of solutions depends to check for afterwards depends on the number of prime factors of $q-1$. In good cases, there are few enough factors to reconstruct the solutions modulo a large enough modulus efficiently. We refer to the code [Rob23e] for more details.

We call this a monodromy leak for the following reason. We'll use the biextension version rather than the cubical version for simplicity. Assume that we are not given the projective coordinate leak of $n.P$, which encodes the information about g_P^n. We can still take any biextension element $g_{P,nP}$ above nP. There is a m such that $g_{P,nP} = g_P^m$, where the value of m is determined modulo $\ell(q-1)$ and is congruent to n modulo ℓ. We have $g_{P,nP} = \lambda_2 \widetilde{g_{P,nP}}$ and $g_P = \lambda_1 \widetilde{g_P}$, so we have the equation $\lambda_2 = \lambda_1^m$ in $\mathbb{F}_q^*$. Unfortunately, this only recover the value of m modulo $q-1$ (at best, if λ_1 is a multiplicative generator), which gives no information on n modulo ℓ since ℓ is prime to $q-1$. The reason the projective coordinate leak above works is that in this case we know that $m \leq \ell$, so is equal to n. Essentially, we know that the value of $g_{P,nP}$ we obtain from the projective coordinate leak is g_P^m with m small enough and not wrapping an unknown number of time around ℓ; which is why we call it a monodromy leak.

Compared to [NSS04], our monodromy leak requires to know the starting coordinates (X_P, Z_P) used in the ladder (usually the point P is normalised so that $z_P = 1$ which is the assumption we have used in the above formulaes; but the general case is not harder, as long as we know the choice of z_P), rather than just the leak of the projective coordinates of $nP = (X_{nP}, Z_{nP})$. On the other hand, it is much more devastating: rather than leaking a few bits of n, we recover it fully via some DLPs in $\mathbb{F}_q^*$, so in subexponential time. (In practice q is of 256 bits, so the DLP is quite effective).

The monodromy leak is thwarted e.g. by doing a constant time division at the end to only send $x_{nP} = X_{nP}/Z_{nP}$. For extra security measure, a supplementary countermeasure is also to mask the projective coordinates of P by a random scalar at the beginning. This protect in case side channels information allows to recover some informations on the intermediate projective coordinates during the ladder. This means that P won't be normalised any longer, so this adds $1M$ by bits in the usual ladder, but luckily the complexity of the time/memory trade off described in Section 6 does not depends on whether P is normalised or not.

Remark 7.2. Fre Vercauteren informed me that the curve NISTp521 uses the prime $p = 2^{521} - 1$ such that $\mathbb{F}_p^*$ has very smooth order (the largest factor has 60 bits). The DLP is very easy in this field.

It is plausible that the monodromy leak described here for the Montgomery ladder extends to more general scalar multiplication, albeit with a more complicated polynomial depending on the exact implementation of the scalar multiplication.

More precisely, what is certainly true is that the biextension and cubical torsor structure exist for all models (see the code [Rob23e]), and that we can efficiently compute "canonical lifts" as above. The only issue is that the scalar multiplication implemented, when interpreted on the affine lifts, won't be the same as the cubical multiplication. For the usual Montgomery ladder, it was easy to keep track of the corrective factor $(4x_P)^{n(2^b-n)}$ above, because the formulas are quite close to the "correct" cubical formulas. In general, it is plausible that there is still a corrective factor that can still be expressed in the exponent as some polynomial in n. This then would give a more complicated polynomial equation than Equation (16).

The main difficulty would be to handle the addition: the cubical arithmetic really needs to use some differential additions (or alternatively would write $(2n+1)P$ as a three way addition $(2n+1)P, P, nP, nP; 0, 2nP, (n+1)P, (n+1)P)$, which Kummer lines arithmetic also uses (maybe with a different constant than the cubical one) but not standard elliptic curve arithmetic.

Anyway, going further into wild speculations, it would not be surprising if the NSA was aware of these kind of “monodromy attacks” or variants. Continuing our wild speculations, we remark that the NIST curves are from 1999, and at that time a 512 bits DLP in $\mathbb{F}_p^*$ was probably quite expensive even for the NSA: even in 2005 the public record for a DLP was for 430 bits. But selecting p such that $p-1$ is smooth would render the DLP in $\mathbb{F}_p^*$ trivial, and at that time [NSS04] was not yet published, so probably not all implementations were protected against projective coordinates leaks…

References

[AGB20] A. C. Aldaya, C. P. García, and B. B. Brumley. “From A to Z: Projective coordinates leakage in the wild”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2020), pp. 428–453 (cit. on p. 4).

[BRS23] R. Barbulescu, D. Robert, and N. Sarkis. “Models of Kummer lines and Galois representations”. June 2023. In preparation. (Cit. on pp. 2, 6, 9).

[BELL10] J. Boxall, N. El Mrabet, F. Laguillaumie, and D.-P. Le. “A variant of miller’s formula and algorithm”. In: *Pairing-Based Cryptography-Pairing 2010: 4th International Conference, Yamanaka Hot Spring, Japan, December 2010. Proceedings 4*. Springer. 2010, pp. 417–434 (cit. on pp. 3, 19).

[Bre83] L. Breen. *Fonctions thêta et théoreme du cube*. Vol. 980. Springer, 1983 (cit. on p. 19).

[CH17] C. Costello and H. Hisil. “A simple and compact algorithm for SIDH with arbitrary degree isogenies”. In: *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II 23*. Springer. 2017, pp. 303–329 (cit. on pp. 13, 14).

[CLN16] C. Costello, P. Longa, and M. Naehrig. “Efficient algorithms for supersingular isogeny Diffie-Hellman”. In: *Advances in Cryptology–CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I 36*. Springer. 2016, pp. 572–601 (cit. on pp. 3, 13).

[DJP14] L. De Feo, D. Jao, and J. Plût. “Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies”. In: *Journal of Mathematical Cryptology* 8.3 (2014), pp. 209–247 (cit. on p. 11).

[Gro72] A. Grothendieck. *Groupes de Monodromie en Géométrie Algébrique: SGA 7*. Springer-Verlag, 1972 (cit. on p. 18).

[HR19] H. Hisil and J. Renes. “On kummer lines with full rational 2-torsion and their usage in cryptography”. In: *ACM Transactions on Mathematical Software (TOMS)* 45.4 (2019), pp. 1–17 (cit. on p. 6).

[LS08] K. E. Lauter and K. E. Stange. “The elliptic curve discrete logarithm problem and equivalent hard problems for elliptic divisibility sequences”. In: *International Workshop on Selected Areas in Cryptography*. Springer. 2008, pp. 309–327 (cit. on p. 21).

[LR10] D. Lubicz and D. Robert. “Efficient pairing computation with theta functions”. In: ed. by G. Hanrot, F. Morain, and E. Thomé. Vol. 6197. Lecture Notes in Comput. Sci. 9th International Symposium, Nancy, France, ANTS-IX, July 19-23, 2010, Proceedings. Springer–Verlag, July 2010. DOI: 10.1007/978-

3-642-14518-6_21. URL: http://www.normalesup.org/~robert/pro/publications/articles/pairings.pdf. Slides: 2010-07-ANTS-Nancy.pdf (30min, International Algorithmic Number Theory Symposium (ANTS-IX), July 2010, Nancy), HAL: hal-00528944. (Cit. on p. 19).

[LR15] D. Lubicz and D. Robert. "A generalisation of Miller's algorithm and applications to pairing computations on abelian varieties". In: *Journal of Symbolic Computation* 67 (Mar. 2015), pp. 68–92. DOI: 10.1016/j.jsc.2014.08.001. URL: http://www.normalesup.org/~robert/pro/publications/articles/optimal.pdf. HAL: hal-00806923, eprint: 2013/192. (Cit. on pp. 19, 20).

[LR16] D. Lubicz and D. Robert. "Arithmetic on Abelian and Kummer Varieties". In: *Finite Fields and Their Applications* 39 (May 2016), pp. 130–158. DOI: 10.1016/j.ffa.2016.01.009. URL: http://www.normalesup.org/~robert/pro/publications/articles/arithmetic.pdf. HAL: hal-01057467, eprint: 2014/493. (Cit. on p. 20).

[Mor85] L. Moret-Bailly. *Pinceaux de variétés abéliennes*. Société mathématique de France, 1985 (cit. on p. 19).

[NSS04] D. Naccache, N. P. Smart, and J. Stern. "Projective coordinates leak". In: *Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23*. Springer. 2004, pp. 257–267 (cit. on pp. 2, 4, 22, 23).

[OLHFR18] T. Oliveira, J. López, H. Hışıl, A. Faz-Hernández, and F. Rodríguez-Henríquez. "How to (pre-) compute a ladder: Improving the performance of X25519 and X448". In: *Selected Areas in Cryptography–SAC 2017: 24th International Conference, Ottawa, ON, Canada, August 16-18, 2017, Revised Selected Papers 24*. Springer. 2018, pp. 172–191 (cit. on pp. 14–16).

[Rei23] K. Reijnders. "Effective Pairings in Isogeny-based Cryptography". In: *Cryptology ePrint Archive* (2023) (cit. on p. 3).

[Ren18] J. Renes. "Computing isogenies between Montgomery curves using the action of (0, 0)". In: *Post-Quantum Cryptography: 9th International Conference, PQCrypto 2018, Fort Lauderdale, FL, USA, April 9-11, 2018, Proceedings*. Springer. 2018, pp. 229–247 (cit. on pp. 9, 13).

[Rob22] D. Robert. "Arithmetic on Kummer lines". Oct. 2022. In preparation. (Cit. on p. 2).

[Rob23a] D. Robert. "A note on optimising 2^n-isogenies in higher dimension". June 2023. URL: http://www.normalesup.org/~robert/pro/publications/notes/2023-06-optimising_isogenies.pdf (cit. on pp. 5, 6, 8, 9, 16).

[Rob23b] D. Robert. "Arithmetic and pairings on Kummer lines". Leuven isogeny days 4, Leuven. Oct. 2023. URL: http://www.normalesup.org/~robert/pro/publications/slides/2023-10-Leuven.pdf (cit. on pp. 18, 19).

[Rob23c] D. Robert. "Biextensions and Pairings on Kummer lines". Aug. 2023. In preparation. (Cit. on p. 2).

[Rob23d] D. Robert. "Canonical liftings to biextensions and theta groups". Aug. 2023. In preparation. (Cit. on p. 21).

[Rob23e] D. Robert. "Kummer Line". Toolbox for computing on Kummer lines. Oct. 2023. URL: https://gitlab.inria.fr/roberdam/kummer-line (cit. on pp. 2, 17, 19, 20, 22).

[Rob23f] D. Robert. “Projective coordinate leaks revisited”. Oct. 2023. In preparation. (Cit. on p. 2).

[Rob23g] D. Robert. “The geometric interpretation of the Tate pairing and its applications”. Feb. 2023. URL: `http://www.normalesup.org/~robert/pro/publications/articles/geometric_tate_pairing.pdf`. eprint: 2023/177, HAL: hal-04295743v1. (Cit. on p. 21).

[RS24] D. Robert and N. Sarkis. “Computing 2-isogenies between Kummer lines”. Jan. 2024. URL: `http://www.normalesup.org/~robert/pro/publications/articles/kummer_isogenies.pdf`. eprint: 2024/037. (Cit. on p. 2).

[Sta08] K. Stange. “Elliptic nets and elliptic curves”. PhD thesis. Brown University, 2008. URL: `https://repository.library.brown.edu/studio/item/bdr:309/PDF/` (cit. on pp. 18, 19).

INRIA Bordeaux–Sud-Ouest, 200 avenue de la Vieille Tour, 33405 Talence Cedex FRANCE
Email address: `damien.robert@inria.fr`
URL: `http://www.normalesup.org/~robert/`

Institut de Mathématiques de Bordeaux, 351 cours de la liberation, 33405 Talence cedex FRANCE