Montgomery ladders already compute pairings

Alessandro Sferlazza
joint work with: G. Pope, K. Reijnders, D. Robert, B. Smith
https://eprint.iacr.org/2025/672

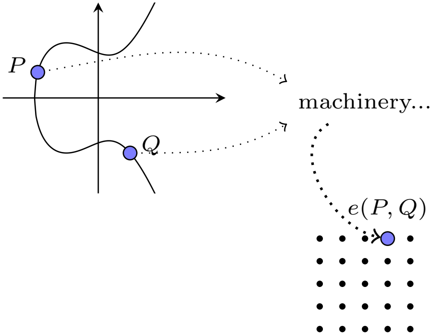
Technical University of Munich

Wednesday 11 June 2025
Aztec Labs, Cryptography seminar

Main character: Pairings on elliptic curves

Pairings are bilinear maps from subgroups/quotients of elliptic curves with nice extra properties

$$e_\ell : \quad G_1 \times G_2 \rightarrow G_T \subseteq k^\times \qquad \ell \in \mathbb{N}$$
$$(P, Q) \mapsto e_\ell(P, Q)$$



- Efficiently computable: e.g. $e_\ell(P,Q) = f_{\ell,P}(Q)^m$ Polynomial in the coordinates of P, Q.
- Destructive use: breaking discrete logs in elliptic curves (MOV reduction)
- Consructive use:
  - Advanced functionalities in encryption, signatures, pseudo-random functions
  - Zero-knowledge proofs
  - tool in Isogeny-based cryptography
  - ...

Motivation: isogeny-based crypto

Pairings are used in different scenarios in cryptography:

- curve-based and pairing-based cryptography:
  $\rightsquigarrow$ freedom to choose highly optimized parameters:
  - field characteristic $p = \operatorname{char} k$ with fast arithmetic
  - P, Q on a fixed curve E with small/nice coefficients
- isogeny-based crypto: no control over specific p, E for fast arithmetic
  - E usually a random supersingular curve over $\mathbb{F}_{p^2}$, with p large
  - p chosen so that $p+1$ has small prime factors $\ell_i \rightsquigarrow$ degree-ℓ_i isogenies are fast to compute

  $\rightsquigarrow$ need *fast* generic pairing.

Cost of generic degree-ℓ pairings per bit of ℓ:

	Tate pairing	Weil pairing
State of the art[1] using Miller's algo	11.3M + 7.7S + 20.7A	2 · Tate pairing
[Rob24][2] $\rightsquigarrow$ our work	9M + 6S + 16A	

[1] Cai, Lin, Zhao, *Pairing Optimizations for Isogeny-based Cryptosystems*, eprint.iacr.org/2024/575

[2] Robert, *Fast pairings via biextensions and cubical arithmetic*, eprint.iacr.org/2024/517

Appendix: divisors

Let $E/\mathbb{F}_q$ be an elliptic curve. A divisor on E is a formal sum

$$D = n_1 \cdot (P_1) + \ldots + n_r \cdot (P_r) \qquad n_i \in \mathbb{Z}, P_i \in E$$

The divisors of degree 0 on E form a group:

$$\mathrm{Div}^0(E) = \{D = n_1(P_1) + \ldots + n_r(P_r) \mid n_1 + \ldots + n_r = 0\}.$$

Given a rational function $f \in \overline{\mathbb{F}}_q(E)$, we attach to it a principal divisor

$$\mathrm{div}\, f = \sum_{P \in E} \mathrm{ord}_P(f) \cdot (P)$$

where $\mathrm{ord}_P(f)$ is the multiplicity of P as a zero of f if > 0, and as pole of f if < 0

Any E elliptic curve is isomorphic to a quotient of $\mathrm{Div}^0(E)$:

$$\begin{array}{ccccc} E & \xrightarrow{\sim} & \mathrm{Pic}^0(E) & = \mathrm{Div}^0(E)/\{\text{principal divisors}\} \\ P & \longmapsto & [(P) - (0_E)] & \end{array}$$

How pairings are computed in practice: Miller's algorithm

Working example: Fix a degree ℓ, a base field $k = \mathbb{F}_q$ containing ℓ-th roots of unity μ_ℓ.
The non-reduced Tate-Lichtenbaum pairing is defined as

$$e_{T,\ell}\colon E[\ell](k) \times E(k)/[\ell]E(k) \to k^\times/(k^\times)^\ell \qquad (P, [Q]) \mapsto f_{\ell,P}(Q)$$

[To avoid $(k^\times)^\ell$-ambiguity, the reduced Tate pairing $e_{t,\ell}(P,Q) = f_{\ell,P}(Q)^{\frac{q-1}{\ell}}$ is often used.]
where $f_{\ell,P} \in k(E)$ is a Miller function attached to P, i.e. satisfies

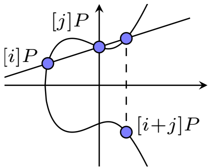
$$\operatorname{div} f_{\ell,P} = (\ell - 1)\,(0_E) + ([\ell]P) - \ell\,(-P) \in \operatorname{Div}^0(E)$$

Other pairings (Weil, (optimal) ate...) are also defined via Miller functions.

These rational functions satisfy

$$f_{i+j,P} = f_{i,P} \cdot f_{j,P} \cdot (l_{[i]P,[j]P}/v_{[j]P})$$

with $l_{R,S}$ = line through R and S, and v_S = vertical line through S.



Miller's algorithm: compute $f_{\ell,P}(Q)$ by:

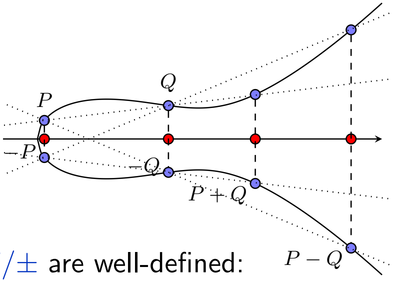
- Fix an addition chain $(1, 2, \ldots, \ell)$
- Step by step compute $(P, f_{1,P}(Q)), ([2]P, f_{2,P}(Q)), \ldots, ([\ell]P, f_{\ell,P}(Q))$

Working with x-only arithmetic

To compute line functions $l_{R,S}$, v_R for Miller's algorithm, we represent points on E as $P = (X_P : Y_P : Z_P)$.
The group law tells us how to add points P, Q together.
What if we forget about Y?

$Y_P = \pm\sqrt{g(X_P, Z_P)}$ $\rightsquigarrow$ sign ambiguity:

$(X_P : Z_P)$ represents $\pm P$



Despite $\pm$, arithmetic is still possible! These operations on $E/\pm$ are well-defined:

$$\mathrm{xDBL}\colon P \mapsto [2]P, \qquad \mathrm{xADD}\colon (P, Q; P-Q) \mapsto P+Q$$

...and quite fast to perform. Montgomery model: only 3 mult, 2 squarings.

$$\mathrm{xDBL}\colon \begin{cases} Q = (X_P + Z_P)^2 \\ R = (X_P - Z_P)^2 \\ S = Q - R \\ [2]P = (QR : S(R + \frac{a+2}{4}S)) \end{cases} \qquad \mathrm{xADD}\colon \begin{cases} U = (X_P - Z_P)(X_Q + Z_Q) \\ V = (X_P + Z_P)(X_Q - Z_Q) \\ X_{P+Q} = Z_{P-Q} \cdot (U+V)^2 \\ Z_{P+Q} = X_{P-Q} \cdot (U-V)^2 \end{cases}$$

Multiplying points by scalars: the Montgomery ladder

Goal: compute scalar multiplication $P \mapsto [\ell]P$
$\rightsquigarrow$ possible using x-only arithmetic!

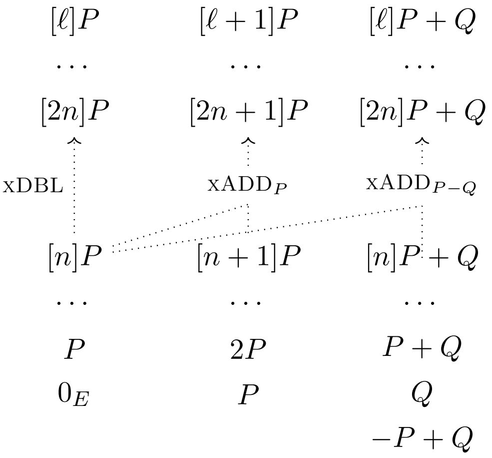
We defined operations on $E/\pm$:

$$\mathrm{xDBL}\colon P \mapsto [2]P$$
$$\mathrm{xADD}\colon (P_1, P_2; P_1 - P_2) \mapsto P_1 + P_2$$

To compute scalar multiplication,
we combine them into a

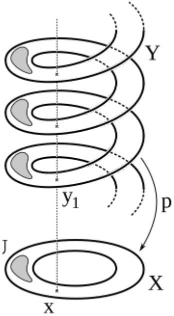
$$\textsc{Ladder}\colon (\ell, P) \mapsto ([\ell]P, [\ell+1]P).$$

Generalization useful later:[3]
$\textsc{3PtLadder}$ with offset Q.
Needs extra input $\pm(P - Q)$.



[3] De Feo, Jao, Plût, *Towards quantum-secure cryptosystems with isogenies*, `eprint.iacr.org/2011/506`

Core idea: monodromy



Walk on the helix so that the projection below is a loop.

$\Longrightarrow$ Above, we're walking up (or down) one floor!

On the curve: we compute $0_E,\ P,\ [2]P,\ \dots,\ [\ell]P = 0_E$ ...back to the start

By $E \xrightarrow{\sim} \mathrm{Pic}^0(E)$, the torsion relation $[\ell]P = 0$ becomes $\big[\ell(0_E) - \ell(-P)\big] = 0$.

Now walk above: $D = \ell(0_E) - \ell(-P) = \mathrm{div}\, f_{\ell,P} \neq 0$ in $\mathrm{Div}^0(E)$.

Even if $[D] = [0]$, the representative D carries nontrivial information: pairings!

Miller's algorithm computes this monodromy: while walking through $0_E, P, [2]P, \dots, [\ell]P$, accumulates divisor information $f_{\ell,P}(Q) = \prod_j l_{[i_j]P,[i'_j]P}(Q)/v_{[i_j]P}(Q)$.

Monodromy already appears in the Montgomery ladder alone:

- Start with $0_E = (1:0)$ and $P = (X_P : Z_P)$
- Perform $\textsc{Ladder}(P, \ell)$: get $[\ell]P = (X_{\ell P} : 0) = (1:0)$
  $\rightsquigarrow X_{\ell P}$ is a monodromy factor. Projective coordinates carry meaning!!

Montgomery ladders *almost* compute pairings

$$P = (x_P : 1) \in E[\ell], \quad Q = (x_Q : 1), \quad P - Q = (x_{P-Q} : 1)$$

We look at the 3PtLadder where P, Q interact. Observe monodromy factors:

$$\begin{matrix} 0_E = (1,0) \\ Q = (x_Q, 1) \end{matrix} \xrightarrow{3\text{PtLadder}(\ell,P,Q;P-Q)} \begin{matrix} [\ell]P = (X_{\ell P}, 0) & \text{differ by } \lambda_P = X_{\ell P} \\ [\ell]P + Q = (X_{\ell P+Q}, Z_{\ell P+Q}) & \text{differ by } \lambda_Q = Z_{\ell P+Q} \end{matrix}$$

From this we get the Tate pairing! squared, + garbage

$$\lambda_Q/\lambda_P = e_{T,\ell}(P,Q)^2 \cdot \text{STUFF}$$

More precisely, $\text{STUFF} = \dfrac{(4x_P)^{\ell\cdot(\neg\ell+1)}}{(4x_P)^{\ell\cdot\neg\ell}(4x_Q)^{\ell}(4x_{P-Q})^{\neg\ell}}$ depends on [3]

- initial input coordinates
- bit representation of ℓ.

Solution: compute STUFF and divide it out...

or better: edit the Ladder to get rid of STUFF.

[3] notation: $\neg\ell$ = bitwise negation of the bit representation of ℓ

Montgomery ladders compute pairings

Remember $\quad \text{xADD}(P, Q; P - Q) = (X_{P+Q}, Z_{P+Q})$.

Modify into cADD: different projective scaling of the output (X_{P+Q}, Z_{P+Q})

$$\begin{aligned} U, V &= \dots \\ X_{P+Q} &= Z_{P-Q}\,(U+V)^2\,, \\ Z_{P+Q} &= X_{P-Q}\,(U-V)^2\,. \end{aligned} \quad \rightsquigarrow \quad \begin{aligned} U, V &= \dots \\ X_{P+Q} &= (4X_{P-Q})^{-1} \cdot (U+V)^2\,, \\ Z_{P+Q} &= (4Z_{P-Q})^{-1} \cdot (U-V)^2\,. \end{aligned}$$

We call this cubical differential addition.

Set $\text{cDBL} = \text{xDBL}$ and replace cADD into the ladder.

Then $\quad \text{cLadder}(\ell, P, Q; P - Q) \mapsto (\ell P, \ell P + Q) \quad$ in (X, Z)-coordinates:

$$\lambda'_Q / \lambda'_P = Z_{\ell P + Q} / X_{\ell P} = e_{T,\ell}(P, Q)^2 \qquad \text{without extra STUFF!}$$

- The square is not a problem when ℓ is odd ✓ $\quad \ell$ even $\longrightarrow$ small trick to avoid the square
- Just minor tweak needed in the conversion $\text{xADD} \longrightarrow \text{cADD}$
  $\rightsquigarrow$ easy optimized, constant-time implementation. [4]
- Inverses can be pre-computed and batched: only one inversion per pairing

[4] Rust and Sagemath libraries provided at `https://github.com/GiacomoPope/cubical-pairings`

Other pairings

Just seen: from one Montgomery 3-point ladder with edited $\textsc{cADD} \rightsquigarrow$
Non-reduced Tate pairing $e_{T,\ell}(P,Q) = f_{\ell,P}(Q)$ from projective coordinates $(X_{\ell P}, Z_{\ell P+Q})$.

What about other pairings? Also recoverable from ladders & some ratios!

- Reduced Tate pairing: $e_{t,\ell}(P,Q) = f_{\ell,P}(Q)^{\frac{p^k-1}{\ell}}$:
  just exponentiate after finding $e_{T,\ell}$ via $\textsc{cLadder}$.
- Weil pairing
  $$e_{W,\ell} \colon E[\ell] \times E[\ell] \to \mu_\ell \qquad (P,Q) \mapsto f_{\ell,P}(Q)/f_{\ell,Q}(P)$$
  This requires $2 \cdot$ non-reduced Tate pairings $\approx 2 \cdot \textsc{cLadder}$.
- ate pairing
  $$e_{A,\ell} \colon \mathbb{G}_2 \times \mathbb{G}_1 \to \mu_\ell \qquad (P,Q) \mapsto f_{\lambda,P}(Q)^{\frac{q^k-1}{\ell}}$$
  with $\lambda \equiv q \pmod{\ell}$, $\mathbb{G}_1 = E[\ell](\mathbb{F}_q^k)$, and $\mathbb{G}_2 = E[\ell] \cap \ker(\pi_q - [q])$.
  Here, monodromy between one (shorter) $\textsc{cLadder}$ and Frobenius π_q:
  Projectively, $\pi_q(P+Q) = [q]P + Q = \textsc{cLadder}(\lambda, P, Q; P - Q)$.

Possible speedups?

Main idea of the tricks we saw: replace xADD with some cADD
where we change the "affine" scaling λ in of $(\lambda \cdot X_{P+Q}, \lambda \cdot Z_{P+Q})$.

And the Montgomery ladder?

- Good when constant-time is needed, code size is constrained, fast enough
- Otherwise, not the fastest way to scalar-multiply $\ell \cdot P$

Questions:

- Can we replace it with faster differential addition chains?
- Or maybe double-and-add chains?
- Miller loops can be sped up by NAFs/windowing/... Can we do it too?

The answer seems to be no :(
Crucial in cubical ladders: the difference points in $\mathrm{xADD}(P, Q; P - Q)$ are fixed.

- This happens in Montgomery Ladders, doesn't apply to DACs
- workarounds: use full-coordinate (X, Y, Z) additions $\rightsquigarrow$ expensive.

Algebra alert:

Some (high-level) theory behind the result

Cubical arithmetic

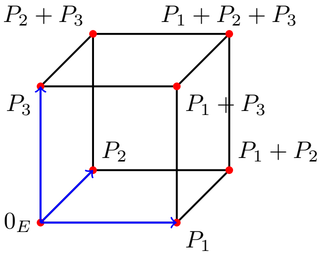
We saw earlier:

- ladder with usual $\mathrm{xADD} \mapsto (X_{P+Q}, Z_{P+Q})$ $\rightsquigarrow Z_{\ell P+Q}/X_{\ell P} = e_{T,\ell}(P,Q)^2 \cdot \mathrm{STUFF}$
- ladder with $\mathrm{cADD} \mapsto (X_{P+Q}/\mu, Z_{P+Q}/\mu)$ $\rightsquigarrow Z_{\ell P+Q}/X_{\ell P} = e_{T,\ell}(P,Q)^2$

There's a preferred projective scaling in the output of xADD. Not a coincidence!

Algebraic statement: *if* $\Gamma(\mathcal{L}) = \langle X, Z\rangle$, *there's a canonical isomorphism of line bundles*

$$t_{P_1}^*\mathcal{L} \otimes t_{P_2}^*\mathcal{L} \otimes t_{P_3}^*\mathcal{L} \otimes t_{P_1+P_2+P_3}^*\mathcal{L} \cong t_{P_2+P_3}^*\mathcal{L} \otimes t_{P_1+P_3}^*\mathcal{L} \otimes t_{P_1+P_2}^*\mathcal{L} \otimes \mathcal{L}$$



Read as follows: $t_P^*\mathcal{L} \longleftrightarrow$ scaling λ of coordinates X_P, Z_P

Fix scaling of 7 vertices,
isomorphism above $\Longrightarrow$ canonical choice for the 8th

Then, cADD and cDBL are special cases:
Let $(P_1, P_2, P_3) = (P, Q, -Q)$. The vertices

$$(P,\ Q,\ -Q,\ P,\ 0,\ P+Q,\ P-Q,\ 0)$$

Fixing $P, Q, P-Q$ we get $P+Q$ uniquely!

Cubical arithmetic as a way to get Miller functions

Main ingredient for pairings: compute rational fns in $k(E)$ with prescribed divisor:

$$\operatorname{div} f_{\ell,P} = \ell(0_E) - \ell(-P).$$

Projective coordinates X, Z are objects living in a line bundle $\mathcal{L}$.

Even though they're not meromorphic functions (like $x, y, 1$) in $k(E)$, they have a zero locus.
For example, $0_E = (1:0)$: $\rightsquigarrow$ Z has a zero at 0_E (...with multiplicity 2)
$\rightsquigarrow$ $\exists$ reasonable notion of divisor of zeroes:

$$\operatorname{div}_0(Z) = 2(0_E), \qquad \operatorname{div}_0(Z(\cdot + P)) = 2(-P).$$

Idea: compute some ratio $g(\cdot) = \dfrac{Z(\cdot + P_1)\cdots Z(\cdot + P_m)}{Z(\cdot + Q_1)\cdots Z(\cdot + Q_m)}$. Hope: we get

$$g \in k(E), \qquad \operatorname{div} g = 2(-P_1) + \cdots + 2(-P_m) - 2(-Q_1) - \cdots - 2(-Q_m)$$

Generally not well-def: must choose P_i, Q_j carefully, compatible with cubical arithmetic.

Miller fns: $P \in E[\ell]$. Then $f_{\ell,P} : R \mapsto \dfrac{Z(R + \ell P) Z(R)^{\ell - 1}}{Z(P)^{\ell}}$ has divisor $2\big(\ell(0) - \ell(-P)\big)$

End of the theory!

Some applications now

Application: multi-dimensional discrete logarithms

- Consider a torsion basis $\langle P, Q\rangle = E[N]$, with N smooth.
- Let $R \in E[N]$. DLog problem: recover (a, b) s.t. $R = [a]P + [b]Q$.

Exploit the Weil pairing $e_N \colon E[N] \times E[N] \to \mu_N$.

[In isogeny applications, the ($2\times$faster) Tate pairing often shares the same properties:]

- Alternating: $e(P, P) = 1$
- Non-degenerate: if P has order N, there is Q s.t. $e(P, Q)$ has order N.
  $e(P, Q)$ has order $N \iff \langle P, Q\rangle = E[N]$

Some details:

$$\zeta_0 = e_N(P, Q) \qquad \text{has order } N$$

$$h_b = e_N(R, P) = e_N([a]P + [b]Q, P) = \zeta_0^{-b}$$

$$h_a = e_N(R, Q) = e_N([a]P + [b]Q, Q) = \zeta_0^{a}$$

DLog in $E[N]$

$\downarrow$ pairing

DLog in μ_N, much easier

✓ Very useful trick in isogeny protocols. Achieved $\sim$ 40% cost reduction w.r.t. Miller.
e.g. point compression (SIKE †, SQIsign2D): (a, b) is shorter than (X_R, Z_R).

Further applications: torsion bases, supersingularity testing

Weil pairing: $e_{W,N} \colon E[N] \times E[N] \to \mu_N$.

- Non-degenerate $\implies e(P,Q)$ has order N iff P, Q are a torsion basis.

Use cases in CSIDH, key agreement based on group actions on isogenies.

Application #1: Torsion basis generation for very composite $N = \prod_i \ell_i$

- Sample random points P, Q
- Do they form a torsion basis? Test order of $e(P,Q) \in \mu_N$.

  [alternative: trial multiplication $P \mapsto [N/\ell_i]P$. Pairing + order testing is much faster ✓]

Application #2: Supersingularity verification

[In CSIDH, the public key must be a supersingular curve $E/\mathbb{F}_p \rightsquigarrow$ public key validation ✓]

- Let $E/\mathbb{F}_{p^2}$ be a supersingular curve with $E(\mathbb{F}_{p^2}) \cong (\mathbb{Z}/(p+1)\mathbb{Z})^2$.
- Try to generate a $(p+1)$-torsion basis (#1). If SUCCESS, return "E is supersingular".
- Retry few times. FAIL if we find P with $[p+1]P \neq 0$.

  $\rightsquigarrow$ Probability of false negatives: 0. Probability of false positives: negligible.

✓ CSIDH uses even embedding degree $k = 2 \rightsquigarrow$ only $\sim$ 7% cost reduction.

Speedups in pairing-based crypto?

Main motivation of cubical pairings: generic pairings *in isogeny-based crypto*.
Any benefits of the new approach on pairing-friendly curves?

⇝ Parallel paper: [LRZZ25] [4] compares with Miller's algorithm on pairing-friendly curves.

– No denominator elimination in cubical arithmetic,

+ though arithmetic itself is faster if points lie in subfields $\mathbb{F}_q \subset \mathbb{F}_{q^k}$

⇝ in some cases, cubical arithmetic can be faster than Miller's algorithm:

- Odd prime embedding degree k (e.g. BW13, $k = 13$)

[4] Lin, Robert, Zhao, Zheng, *Biextensions in Pairing-based Cryptography*, `eprint.iacr.org/2025/670`

Further directions

The theory of cubical arithmetic applies much more generally:

- Other curve models: Theta, Weierstrass, Edwards, ...
- Higher dimensions: with level-2 theta models, Weil & Tate-Lichtenbaum work similarly
  $\rightsquigarrow$ Cubical pairings already implemented in AVIsogenies (Magma), libraries in Sagemath

Also: in specific contexts, alternative computations to $\textsc{cLadder}$ are competitive (e.g. $\textsc{DoubleAndAdd}$, NAFs, ...).

Thank you for listening! Questions?

Even-degree pairings

Consider an even integer $\ell = 2m$.

$$P \in E[\ell](k), \quad Q \in E(k), \qquad \textsc{cLadder}(\ell, P, Q, P-Q) \mapsto \ell P,\ \ell P + Q$$

We can get the squared Tate pairing: $\lambda_P/\lambda_Q = X_{\ell P}/Z_{\ell P+Q} = e_{T,\ell}(P,Q)^2$

The pairing has order dividing $\ell = 2m \rightsquigarrow$ the square loses one bit of information.

Step 1: only compute ladder of order $m = \ell/2$.

$$\textsc{cLadder}(m, P, Q, P-Q) \mapsto mP,\ mP + Q$$

Step 2: *Linear translations.* $T = mP$ is a point of order 2: on the Kummer line, translation by T induces an involution. It acts linearly on coordinates, for example

$$T = (0:1). \qquad T * (X_P, Z_P) = P + T = (Z_P, X_P)$$

$$T = (A:B) \neq (0:1) \qquad T * (X_P, Z_P) = P + T = (AX_P - BZ_P, AZ_P - BX_P)$$

Step 3: *Monodromy.*

$mP + T$ is projectively $= 0_E$ $\rightsquigarrow$ monodromy factor λ'_P

$(mP + Q) + T$ is projectively $= Q$ $\rightsquigarrow$ monodromy factor λ'_Q

$$\lambda_P/\lambda_Q = X_{mP+T}/Z_{(mP+Q)+T} = e_{T,\ell}(P,Q) \qquad \text{without the square!}$$