Faster scalar multiplication on elliptic curves

Nicolas Sarkis, 2nd year PhD student
Advisors: Razvan Barbulescu, Damien Robert

Institut de Mathématiques de Bordeaux

October 19, 2023 – Journées C2, Najac

Elliptic curves and Kummer lines

Motivation

On an elliptic curve, we can compute $n \cdot P = P + \cdots + P$.
Leads to elliptic curve cryptography:

- Signature (ECDSA)
- Key exchange (ECDH)

Goal

Compute $n \cdot P$ efficiently.

Elliptic curves (char $k \neq 2, 3$)

- Short Weierstrass (general case):

$$E : y^2 = x^3 + ax + b$$

- Montgomery:

$$E : By^2 = x(x^2 + Ax + 1)$$

- Twisted Edwards (singularities!):

$$E : ax^2 + y^2 = 1 + dx^2y^2$$

Elliptic curves (char $k \neq 2, 3$)

- Short Weierstrass (general case):

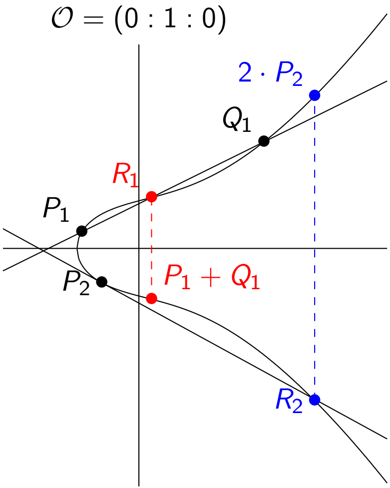
$$E : y^2 = x^3 + ax + b$$

- Montgomery:

$$E : By^2 = x(x^2 + Ax + 1)$$

- Twisted Edwards (singularities!):

$$E : ax^2 + y^2 = 1 + dx^2y^2$$



Kummer lines

We will focus on Montgomery curves:

$$E : By^2 = x(x^2 + Ax + 1)$$

If we know x, we can recover y with a square root up to a sign.
On Kummer lines, we forget about it. This amounts to setting $\mathcal{K} = E/\{\pm 1\}$ because $-P = (x : -y : z)$ if $P = (x : y : z)$:

$$(x : y : z) \mapsto \begin{cases} \infty := (1 : 0) & \text{if } (x : y : z) = (0 : 1 : 0) \\ \frac{x}{z} := (x : z) & \text{else} \end{cases}$$

Notation: $[P] = (x : z)$ if $P = (x : y : z)$.

What do we ~~gain~~ lose?

Problem: no addition law anymore on a Kummer line!

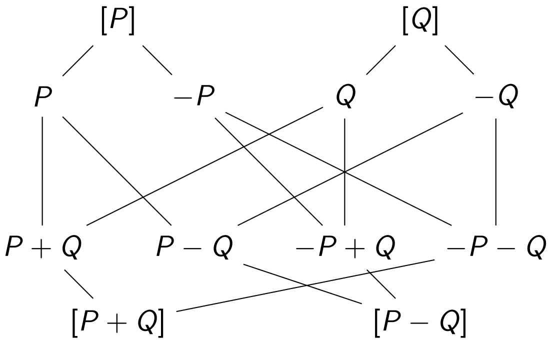


Figure: Two possible choices

What do we ~~gain~~ lose?

Problem: no addition law anymore on a Kummer line!

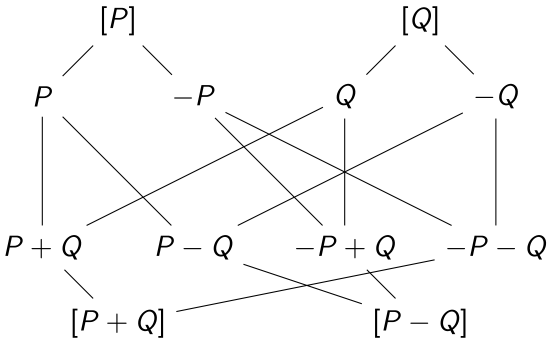


Figure: Two possible choices

However, if you know $[P]$, $[Q]$, $[P-Q]$, you can compute $[P+Q]$

Differential addition and doubling [Mon87]

- M is the cost of a multiplication in k, S the cost of a square in k.
- m_0 is the cost of a multiplication by a small constant in k.

Differential addition $(2M + 2S + 2m,\ m = M \text{ or } m_0)$

$$u := (x_P + z_P)(x_Q - z_Q);\ v := (x_P - z_P)(x_Q + z_Q);$$
$$w := (u + v)^2;\ t := (u - v)^2;$$
$$x_{P+Q} := z_{P-Q}w;\ z_{P+Q} := x_{P-Q}t;$$

Doubling $(2M + 2S + 1m_0,\ d = \frac{A+2}{4})$

$$u := (x_P + z_P)^2;\ v := (x_P - z_P)^2;\ t := u - v;$$
$$x_{2P} := uv;\ z_{2P} := t(v + dt);$$

Why counting M, S and m_0 differently?

Squaring S (in $\mathbb{F}_{p^2} = \mathbb{F}_p[i]$ with $i^2 = -1$)

M $(a+ib)(c+id) = ac - bd + i(ac + bd - (a-b)(c-d))$ $(3M(\mathbb{F}_p))$

S $(a+ib)^2 = (a-b)(a+b) + 2iab$ $(2M(\mathbb{F}_p))$

So $S/M \approx 2/3$, it is better to have squares.

Why counting M, S and m_0 differently?

Squaring S (in $\mathbb{F}_{p^2} = \mathbb{F}_p[i]$ with $i^2 = -1$)

M $(a+ib)(c+id) = ac - bd + i(ac + bd - (a-b)(c-d))$ $(3M(\mathbb{F}_p))$

S $(a+ib)^2 = (a-b)(a+b) + 2iab$ $(2M(\mathbb{F}_p))$

So $S/M \approx 2/3$, it is better to have squares.

Small constant m_0

Represent numbers as polynomials in $r = 2^{64}$:

$$n, m = P(r), Q(r)$$

Then $nm = PQ(r)$, not linear in general.
But if P is monomial (e.g. n fits in 1 computer word), then nm is linear.

Algorithm 1: Montgomery ladder

Input: $[R] = [m \cdot P]$, $[S] = [(m+1) \cdot P]$, b a bit
Output: $([2 \cdot R], [R+S])$ if $b = 0$ $([R+S], [2 \cdot S])$ if $b = 1$
Data: The point $[P]$

Function `MontgomeryLadder`$([R], [S], b)$:
  if $b = 0$ **then**
    $[S] \leftarrow$ `DiffAdd`$([R], [S], [P])$;
    $[R] \leftarrow$ `Doubling`$([R])$;
  else if $b = 1$ **then**
    $[R] \leftarrow$ `DiffAdd`$([R], [S], [P])$;
    $[S] \leftarrow$ `Doubling`$([S])$;
  end
  return $([R], [S])$;

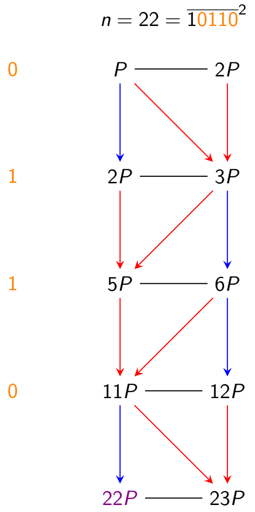


Figure: Chaining ladder

Our results

New assumption and faster formulas

Assume we have complete 2-torsion on the curve.

$$E : By^2 = x(x - \alpha)(x - \alpha^{-1})$$

$[T] = (\alpha : 1)$, we have faster (in some context) formulas for a "quasi-doubling":

Quasi-doubling ($4S + 3m_0$)

$$u := (\alpha - 1)(x_P + z_P)^2; v := (\alpha + 1)(x_P - z_P)^2;$$
$$x_{2.P+T} := (u + v)^2; z_{2.P+T} := \alpha(u - v)^2;$$

We can retrieve $[2 \cdot P + T] = [2 \cdot P - T]$ (because $2 \cdot T = \mathcal{O}$).

Algorithm 2: Hybrid ladder

Input: $[R]$, $[S]$ with $[R-S] \in \{[P],[P+T]\}$ and b a bit
Output: $([2 \cdot R+T],[R+S])$ if $b=0$ $([R+S],[2 \cdot S+T])$ if $b=1$
Data: The points $[P]$, $[Q]=[P+T]$

```
Function HybridLadder([R], [S], b):
    [D] ← [R − S];                          // pre-computed
    if b = 0 then
        [S] ← DiffAdd([R], [S], [D]);
        [R] ← QuasiDoubling([R]);
    else if b = 1 then
        [R] ← DiffAdd([R], [S], [D]);
        [S] ← QuasiDoubling([S]);
    end
    return ([R], [S]);
```

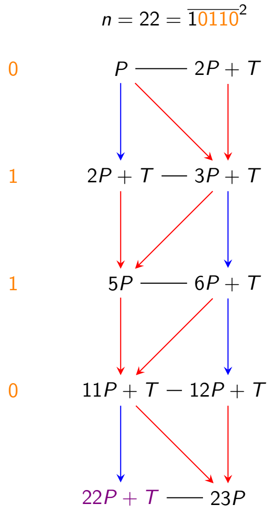


Figure: Chaining hybrid ladder

Complexities

We will consider $S = \frac{2}{3}M$, $m_0 = \frac{1}{10}M$, $m = m_0$ or M.

	Montgomery ladder	Hybrid ladder
Diff. add.	$2M + 2S + 2m$	
(Quasi)-doubling	$2M + 2S + 1m_0$	$4S + 3m_0$
Total	$4M + 4S + 2m + 1m_0$	$2M + 6S + 2m + 3m_0$
$m = M$	$8.77M$	$8.3M$ (5.4%)
$m = m_0$	$6.97M$	$6.5M$ (6.7%)

Table: Comparison between Montgomery and hybrid ladder

Proof of concept

Context

- $\mathbb{F}_{p^{10}} = \mathbb{F}_{p^5}[i]$ and $\mathbb{F}_{p^5} = \mathbb{F}_p[u]$ with $i^2 = -1$, $u^5 = 2$.
- Small multiplications are elements of $\mathbb{F}_p$ times elements of $\mathbb{F}_{p^{10}}$.
- Curve constants: $\alpha = 1 + \mu i$, $d = \frac{2-\alpha-\alpha^{-1}}{4} = \nu + i$, $\mu, \nu \in \mathbb{F}_p$
- $x_P, z_P \in \mathbb{F}_{p^{10}}$, i.e. $m = M$.
- 100 random scalar multiplications, repeated 100 times.

	Montgomery ladder	**Hybrid ladder**
Average (s)	2.502 ± 0.039	2.348 ± 0.017 (**6.2%**)

Table: Timings on Intel Core i5-1145G7 @ 2.60GHz

Conclusion

Future work and research direction

Work in progress:

- Application: Elliptic Curve Method.
- Where does it come from? General framework to compute 2-isogenies between Kummer lines.
- More generally, classification of Kummer lines using their Galois representation and how they relate to different models of elliptic curves.

Research direction:

- Implementation of ECDSA.
- Comparison to `Curve25519`.

[Mon87] Peter L. Montgomery. "Speeding the Pollard and elliptic curve methods of factorization". English. In: *Mathematics of Computation* 48 (1987), pp. 243–264. ISSN: 0025-5718. DOI: 10.2307/2007888.