$\sqrt{\text{élu's formulas}}$

Faster Evaluation of Isogenies of Large Prime Degree

Luca De Feo

IBM Research Zürich

joint work with D.J. Bernstein, A. Leroux, B. Smith

April 29, 2020, the University of Zürich eSeminar

Why isogenies?

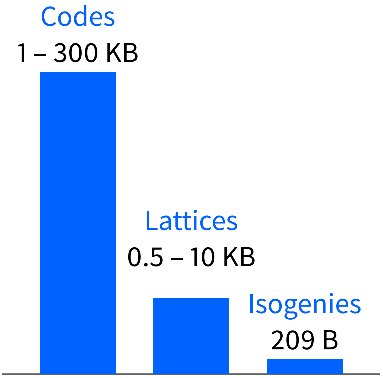
Six families still in NIST post-quantum competition:

Lattices	9 encryption	3 signature
Codes	7 encryption	
Multivariate		4 signature
Isogenies	1 encryption	
Hash-based		1 signature
MPC		1 signature

Why isogenies?

Six families still in NIST post-quantum competition:

Lattices	9 encryption	3 signature
Codes	7 encryption	
Multivariate		4 signature
Isogenies	1 encryption	
Hash-based		1 signature
MPC		1 signature



Public key size

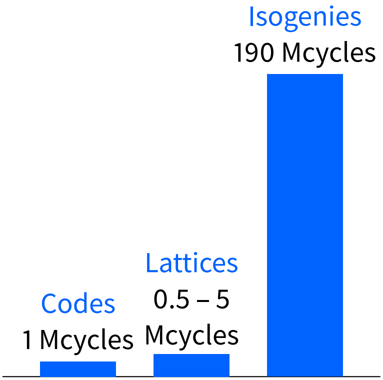
NIST-1 level (AES128)

(not to scale)

Why isogenies?

Six families still in NIST post-quantum competition:

Lattices	9 encryption	3 signature
Codes	7 encryption	
Multivariate		4 signature
Isogenies	1 encryption	
Hash-based		1 signature
MPC		1 signature



Encryption performance

NIST-1 level (AES128)

(not to scale)

Iso-what?!

Keywords

- An isogeny is a map between two elliptic curves;

Iso-what?!

Keywords

- An isogeny is a map between two elliptic curves;
- It is a group morphism:

$$\phi(P+Q)=\phi(P)+\phi(Q);$$

Iso-what?!

Keywords

- An isogeny is a map between two elliptic curves;
- It is a group morphism:

$$\phi(P+Q)=\phi(P)+\phi(Q);$$

- It is an algebraic map:

$$\phi(x,y)=\left(\frac{g(x)}{h(x)},\,y\left(\frac{g(x)}{h(x)}\right)'\right);$$

Iso-what?!

Keywords

- An isogeny is a map between two elliptic curves;
- It is a group morphism:

$$\phi(P+Q) = \phi(P) + \phi(Q);$$

- It is an algebraic map:

$$\phi(x,y) = \left(\frac{g(x)}{h(x)}, y\left(\frac{g(x)}{h(x)}\right)'\right);$$

- It is entirely determined by its kernel (i.e., by a single point);

Iso-what?!

Keywords

- An isogeny is a map between two elliptic curves;
- It is a group morphism:

$$\phi(P+Q)=\phi(P)+\phi(Q);$$

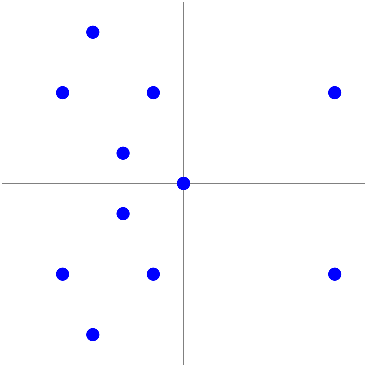
- It is an algebraic map:

$$\phi(x,y)=\left(\frac{g(x)}{h(x)},\, y\left(\frac{g(x)}{h(x)}\right)'\right);$$

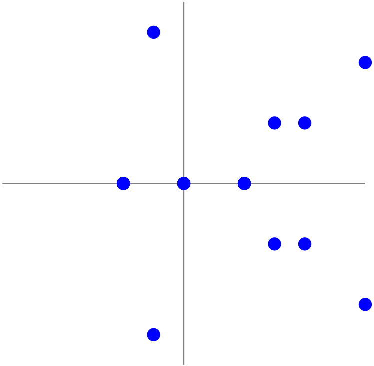
- It is entirely determined by its kernel (i.e., by a single point);
- Isogeny degree $=$ size of the kernel $=$ order of kernel generator $\approx$ size of the polynomials;

Isogenies: an example over $\mathbb{F}_{11}$

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

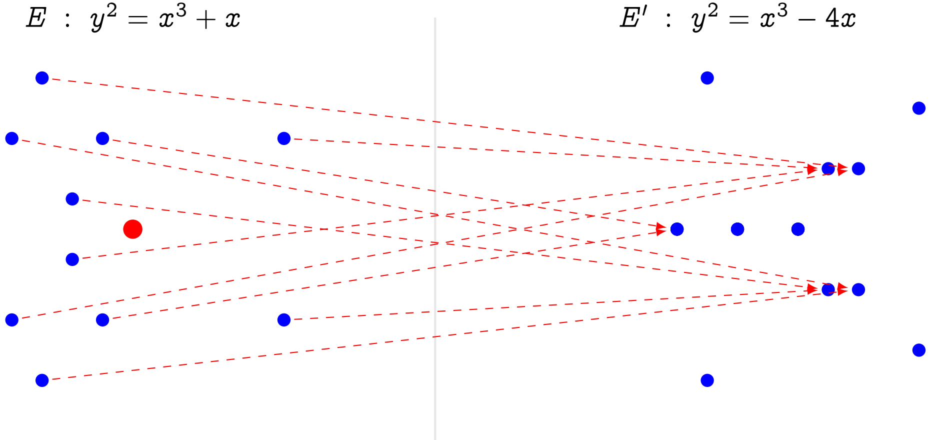


$E' : y^2 = x^3 - 4x$



$$\phi(x, y) = \left(\frac{x^2+1}{x}, \quad y\frac{x^2-1}{x^2}\right)$$

Isogenies: an example over $\mathbb{F}_{11}$



$$\phi(x, y) = \left(\frac{x^2+1}{x}, \quad y\frac{x^2-1}{x^2}\right)$$

- Kernel generator in red.
- This is a degree 2 map.
- Analogous to $x \mapsto x^2$ in $\mathbb{F}_q^*$.

Algebraic complexity

In this talk, k is an arbitrary field.

All complexities are in terms of operations over k.

Anatomy of an isogeny (short Weierstrass form)

$$\phi(x, y) = \left(\frac{x^4 - x^3 + 11x^2 + 9x + 12}{x^3 - x^2 - x + 1},\ y\frac{x^5 - x^4 - 14x^3 - 26x^2 - 67x - 21}{x^5 - x^4 - 2x^3 + 2x^2 + x - 1} \right)$$

Anatomy of an isogeny (short Weierstrass form)

$$\phi(x,y) = \left(\frac{x^4 - x^3 + 11x^2 + 9x + 12}{x^3 - x^2 - x + 1},\ y\frac{x^5 - x^4 - 14x^3 - 26x^2 - 67x - 21}{x^5 - x^4 - 2x^3 + 2x^2 + x - 1} \right)$$

degree (the exponent 4 of the numerator)

degree -1 (the exponent 3 of the denominator)

Anatomy of an isogeny (short Weierstrass form)

degree

$$\phi(x,y) = \left(\frac{x^4 - x^3 + 11x^2 + 9x + 12}{x^3 - x^2 - x + 1},\ y\frac{x^5 - x^4 - 14x^3 - 26x^2 - 67x - 21}{x^5 - x^4 - 2x^3 + 2x^2 + x - 1}\right)$$

kernel polynomial

Anatomy of an isogeny (short Weierstrass form)

degree

$$\phi(x,y) = \left(\frac{x^4 - x^3 + 11x^2 + 9x + 12}{(x+1)(x-1)^2},\; y\frac{x^5 - x^4 - 14x^3 - 26x^2 - 67x - 21}{x^5 - x^4 - 2x^3 + 2x^2 + x - 1}\right)$$

kernel polynomial

Anatomy of an isogeny (short Weierstrass form)

degree

$$\phi(x,y) = \left(\frac{x^4 - x^3 + 11x^2 + 9x + 12}{(x+1)(x-1)^2},\ y\frac{x^5 - x^4 - 14x^3 - 26x^2 - 67x - 21}{x^5 - x^4 - 2x^3 + 2x^2 + x - 1} \right)$$

Point of order 2

Two points of order 4

Anatomy of an isogeny (short Weierstrass form)

degree

$$\phi(x,y) = \left(\frac{x^4 - x^3 + 11x^2 + 9x + 12}{h(x)},\; y\frac{x^5 - x^4 - 14x^3 - 26x^2 - 67x - 21}{x^5 - x^4 - 2x^3 + 2x^2 + x - 1}\right)$$

$$h(x) = \prod_{P\in K\setminus\{0\}} \bigl(x - x(P)\bigr)$$

Anatomy of an isogeny (short Weierstrass form)

computed by Vélu–Elkies–Kohel formulas

$$\phi(x,y) = \left(\frac{g(x)}{h(x)},\quad y\frac{x^5 - x^4 - 14x^3 - 26x^2 - 67x - 21}{x^5 - x^4 - 2x^3 + 2x^2 + x - 1} \right)$$

$$h(x) = \prod_{P\in K\setminus\{0\}} \big(x - x(P)\big)$$

Anatomy of an isogeny (short Weierstrass form)

computed by Vélu–Elkies–Kohel formulas

$$\phi(x,y) = \left(\frac{g(x)}{h(x)}, \quad y\left(\frac{g(x)}{h(x)}\right)' \right)$$

$$h(x) = \prod_{P\in K\setminus\{0\}} \big(x - x(P)\big)$$

Anatomy of an isogeny (short Weierstrass form)

computed by Vélu–Elkies–Kohel formulas

$$\phi(x,y) = \left(\frac{g(x)}{h(x)}, \qquad y\left(\frac{g(x)}{h(x)}\right)' \right)$$

$$h(x) = \prod_{P\in K\setminus\{0\}} \big(x - x(P)\big)$$

Input: Finite kernel $K \subset E(k)$ of order n,

Output: Rational fractions $\phi(x,y)$;

Complexity: $\tilde{O}(n)$ operations over k.

Anatomy of an isogeny (short Weierstrass form)

computed by Vélu–Elkies–Kohel formulas

$$\phi(x,y) = \left(\frac{g(x)}{h(x)}, \qquad y\left(\frac{g(x)}{h(x)}\right)' \right)$$

$$h(x) = \prod_{P\in K\setminus\{0\}} \big(x - x(P)\big)$$

Input: Finite kernel $K \subset E(k)$ of order n, a point $Q \in E(k)$;

Output: Rational fractions $\phi(x,y)$;

Complexity: $\tilde{O}(n)$ operations over k.

Anatomy of an isogeny (short Weierstrass form)

computed by Vélu–Elkies–Kohel formulas

$$\phi(x, y) = \left(\frac{g(x)}{h(x)}, \qquad y \left(\frac{g(x)}{h(x)} \right)' \right)$$

$$h(x) = \prod_{P \in K \setminus \{0\}} \big(x - x(P)\big)$$

Input: Finite kernel $K \subset E(k)$ of order n, a point $Q \in E(k)$;

Output: The image point $\phi(Q)$;

Complexity: $\tilde{\mathrm{O}}(n)$ operations over k.

Anatomy of an isogeny (short Weierstrass form)

computed by Vélu–Elkies–Kohel formulas

$$\phi(x,y) = \left(\frac{g(x)}{h(x)}, \qquad y\left(\frac{g(x)}{h(x)}\right)' \right)$$

$$h(x) = \prod_{P\in K\setminus\{0\}} \big(x - x(P)\big)$$

Input: Finite kernel $K \subset E(k)$ of order n, a point $Q \in E(k)$;

Output: The image point $\phi(Q)$;

Complexity: $O(n)$ operations over k.

Anatomy of an isogeny (short Weierstrass form)

computed by Vélu–Elkies–Kohel formulas

$$\phi(x,y) = \left(\frac{g(x)}{h(x)}, \qquad y\left(\frac{g(x)}{h(x)}\right)' \right)$$

$$h(x) = \prod_{P\in K\setminus\{0\}} \big(x - x(P)\big)$$

Input: Finite kernel $K \subset E(k)$ of order n, a point $Q \in E(k)$;

Output: The image point $\phi(Q)$;

Complexity: $O(n)$ operations over k. Fast?

Anatomy of an isogeny (short Weierstrass form)

computed by Vélu–Elkies–Kohel formulas

$$\phi(x,y) = \left(\frac{g(x)}{h(x)}, \qquad y\left(\frac{g(x)}{h(x)}\right)' \right)$$

$$h(x) = \prod_{P\in K\setminus\{0\}} \big(x - x(P)\big)$$

Input: Finite kernel $K \subset E(k)$ of order n, a point $Q \in E(k)$;

Output: The image point $\phi(Q)$;

Complexity: $O(n)$ operations over k. Fast? YES!

Anatomy of an isogeny (short Weierstrass form)

computed by Vélu–Elkies–Kohel formulas

$$\phi(x,y) = \left(\frac{g(x)}{h(x)},\quad y\left(\frac{g(x)}{h(x)}\right)' \right)$$

$$h(x) = \prod_{P\in K\setminus\{0\}} \big(x - x(P)\big)$$

Input: Finite kernel $K \subset E(k)$ of order n, a point $Q \in E(k)$;

Output: The image point $\phi(Q)$;

Complexity: $\mathrm{O}(n)$ operations over k. Fast? YES! Optimal?

Anatomy of an isogeny (short Weierstrass form)

computed by Vélu–Elkies–Kohel formulas

$$\phi(x,y) = \left(\frac{g(x)}{h(x)},\quad y\left(\frac{g(x)}{h(x)}\right)' \right)$$

$$h(x) = \prod_{P\in K\setminus\{0\}} \big(x - x(P)\big)$$

Input: Kernel generator $P \in E(k)$ of order n, a point $Q \in E(k)$;

Output: The image point $\phi(Q)$;

Complexity: $O(n)$ operations over k. Fast? YES! Optimal?

Anatomy of an isogeny (short Weierstrass form)

computed by Vélu–Elkies–Kohel formulas

$$\phi(x,y) = \left(\frac{g(x)}{h(x)}, \quad y\left(\frac{g(x)}{h(x)}\right)' \right)$$

$$h(x) = \prod_{P\in K\setminus\{0\}} \big(x - x(P)\big)$$

Input: Kernel generator $P \in E(k)$ of order n, a point $Q \in E(k)$;

Output: The image point $\phi(Q)$;

Complexity: $O(n)$ operations over k. Fast? YES! Optimal? NO!

One step back…

$$P(X) = \prod_{i=0}^{n-1} (X - i)$$

One step back…

$$P(\alpha) = \prod_{i=0}^{n-1} (\alpha - i)$$

One step back…

Assume $n = ab$

$$P(\alpha) = \prod_{i=0}^{n-1} (\alpha - i)$$

One step back…

Assume $n = ab$

$$P(\alpha) = \prod_{i=0}^{a-1} \prod_{j=0}^{b-1} (\alpha - i - j \cdot a)$$

One step back...

Assume $n = ab$

$$P(\alpha) = \prod_{i=0}^{a-1} \prod_{j=0}^{b-1} (\alpha - i - j \cdot a)$$

$$G(Y) = \prod_{j=0}^{b-1} (\alpha - Y - j \cdot a)$$

One step back…

Assume $n = ab$

$$P(\alpha) = \prod_{i=0}^{a-1} G(i)$$

$$G(Y) = \prod_{j=0}^{b-1} (\alpha - Y - j \cdot a)$$

One step back…

Assume $n = ab$

$$P(\alpha) = \prod_{i=0}^{a-1} G(i)$$

$$G(Y) = \prod_{j=0}^{b-1} (\alpha - Y - j \cdot a)$$

$$B(Y) = \prod_{i=0}^{a-1} (Y - i)$$

One step back…

Assume $n = ab$

$$P(\alpha) = \mathrm{Res}_Y(G, B)$$

$$G(Y) = \prod_{j=0}^{b-1}(\alpha - Y - j \cdot a)$$

$$B(Y) = \prod_{i=0}^{a-1}(Y - i)$$

One step back…

Assume $n = ab$

$$P(\alpha) = \mathrm{Res}_Y(G, B)$$

$$G(Y) = \prod_{j=0}^{b-1}(\alpha - Y - j \cdot a)$$

$$B(Y) = \prod_{i=0}^{a-1}(Y - i)$$

1. Compute baby steps $B(Y)$
2. Compute giant steps $G(Y)$
3. Compute resultant by multi-point evaluation

One step back…

Assume $n = ab$

$$P(\alpha) = \mathrm{Res}_Y(G, B)$$

$$G(Y) = \prod_{j=0}^{b-1}(\alpha - Y - j \cdot a)$$

$$B(Y) = \prod_{i=0}^{a-1}(Y - i)$$

1. Compute baby steps $B(Y)$ — $\tilde{O}(a)$
2. Compute giant steps $G(Y)$ — $\tilde{O}(b)$
3. Compute resultant by multi-point evaluation — $\tilde{O}(a + b)$

One step back…

Assume $n = ab$

$$P(\alpha) = \mathrm{Res}_Y(G, B)$$

$$G(Y) = \prod_{j=0}^{b-1}(\alpha - Y - j \cdot a)$$

$$B(Y) = \prod_{i=0}^{a-1}(Y - i)$$

1. Compute baby steps $B(Y)$ — $\tilde{O}(a)$
2. Compute giant steps $G(Y)$ — $\tilde{O}(b)$
3. Compute resultant by multi-point evaluation — $\tilde{O}(a + b)$

Pollard '74, Strassen '76: deterministic integer factorization in $O(n^{1/4})$.

Chudnovsky[2] '88: n-th term of a holonomic sequence.

Bostan '20: n-th term of a q-holonomic sequence.

Why did the trick work?

- An arithmetic decomposition of the root set: $i \mapsto i + a \cdot j$;
- Efficient to compute giant steps: $0, a, 2a, \dots, (b-1)a$;
- Only univariate polynomials.

Why did the trick work?

- An arithmetic decomposition of the root set: $i \mapsto i + a \cdot j$;
- Efficient to compute giant steps: $0, a, 2a, \dots, (b-1)a$;
- Only univariate polynomials.

Wait! It works for any algebraic group!

Why did the trick work?

- An arithmetic decomposition of the root set: $g^i \mapsto g^i g^{aj}$;
- Efficient to compute giant steps: $0, a, 2a, \dots, (b-1)a$;
- Only univariate polynomials.

Wait! It works for any algebraic group!

Why did the trick work?

- An arithmetic decomposition of the root set: $g^i \mapsto g^i g^{aj}$;
- Efficient to compute giant steps: $1, (g^a), (g^a)^2, \dots, (g^a)^{b-1}$;
- Only univariate polynomials.

Wait! It works for any algebraic group!

Why did the trick work?

- An arithmetic decomposition of the root set: $g^i \mapsto g^i g^{aj}$;
- Efficient to compute giant steps: $1, (g^a), (g^a)^2, \dots, (g^a)^{b-1}$;
- Only univariate polynomials???

Wait! It works for any algebraic group! Or does it?

…and two steps forward

$$h(x(Q)) = \prod_{P\in K} (x(Q) - x(P))$$

. . . and two steps forward

$$h(x(Q)) = \prod_{i=1}^{n} (x(Q) - x([i]P))$$

…and two steps forward

$$h(x(Q)) = \prod_{i=1}^{a}\prod_{j=1}^{b}(x(Q) - x([i]P + [a\cdot j]P))$$

…and two steps forward

$$h(x(Q)) = \prod_{i=1}^{a}\prod_{j=1}^{b}(x(Q) - x([i]P + [a\cdot j]P))$$

$$B(Y) = \prod_{i=0}^{a-1}(Y - x([i]P))$$

…and two steps forward

$$h(x(Q)) = \prod_{i=1}^{a}\prod_{j=1}^{b}(x(Q) - x([i]P + [a \cdot j]P))$$

$$G(\mathcal{Y}) = \prod_{j=0}^{b-1}(x(Q) - x(\mathcal{Y} + [a \cdot j]P))$$

$$B(Y) = \prod_{i=0}^{a-1}(Y - x([i]P))$$

…and two steps forward

$$h(x(Q)) = \prod_{i=1}^{a}\prod_{j=1}^{b}(x(Q) - x([i]P + [a\cdot j]P))$$

$$G(\mathcal{Y}) = \prod_{j=0}^{b-1}(x(Q) - x(\mathcal{Y} + [a\cdot j]P))$$

$$B(Y) = \prod_{i=0}^{a-1}(Y - x([i]P))$$

Biquadratic relations (Montgomery model)

Let $E : y^2 = x^3 + Ax^2 + x$, let $P, Q \in E$,

$$\begin{aligned}(X - x(P+Q))(X - x(P-Q)) = {} & X^2 \\ & - 2\frac{(x(P)x(Q)+1)(x(P)+x(Q)) + 2Ax(P)x(Q)}{(x(P)-x(Q))^2}X \\ & + \frac{(x(P)x(Q)-1)^2}{(x(P)-x(Q))^2}\end{aligned}$$

assuming $0 \notin \{P, Q, P+Q, P-Q\}$.

Back to polynomials

$$h(x(Q)) = \prod_{i=1}^{a} \prod_{j=1}^{b} (x(Q) - x([i]P + [a \cdot j]P))$$

$$G(\mathcal{Y}) = \prod_{j=0}^{b-1} (x(Q) - x(\mathcal{Y} + [a \cdot j]P))$$

$$B(Y) = \prod_{i=0}^{a-1} (Y - x([i]P))$$

Back to polynomials

$$h(x(Q)) = \prod_{i\in I}\prod_{j\in J}(x(Q) - x([i]P + [j]P))(x(Q) - x([i]P - [j]P))$$

$$G(\mathcal{Y}) = \prod_{j=0}^{b-1}(x(Q) - x(\mathcal{Y} + [a\cdot j]P))$$

$$B(Y) = \prod_{i\in I}(Y - x([i]P))$$

Back to polynomials

$$h(x(Q)) = \prod_{i\in I}\prod_{j\in J}(x(Q) - x([i]P + [j]P))(x(Q) - x([i]P - [j]P))$$

$$G(\mathcal{Y}) = \prod_{j\in J}(x(Q) - x(\mathcal{Y} + [j]P))(x(Q) - x(\mathcal{Y} - [j]P))$$

$$B(Y) = \prod_{i\in I}(Y - x([i]P))$$

Back to polynomials

$$h(x(Q)) = \prod_{i\in I}\prod_{j\in J}(x(Q) - x([i]P + [j]P))(x(Q) - x([i]P - [j]P))$$

$$G(\mathcal{Y}) = \prod_{j\in J}\left(x(Q)^2 + \frac{f_1(x(\mathcal{Y}), x([j]P))}{f_0(x(\mathcal{Y}), x([j]P))}x(Q) + \frac{f_2(x(\mathcal{Y}), x([j]P))}{f_0(x(\mathcal{Y}), x([j]P))}\right)$$

$$B(Y) = \prod_{i\in I}(Y - x([i]P))$$

Back to polynomials

$$h(x(Q)) = \prod_{i \in I} \prod_{j \in J} (x(Q) - x([i]P + [j]P))(x(Q) - x([i]P - [j]P))$$

$$G(Y) = \prod_{j \in J} \left(x(Q)^2 + \frac{f_1(\quad Y \quad, x([j]P))}{f_0(\quad Y \quad, x([j]P))} x(Q) + \frac{f_2(\quad Y \quad, x([j]P))}{f_0(\quad Y \quad, x([j]P))} \right)$$

$$B(Y) = \prod_{i \in I} (Y - x([i]P))$$

Back to polynomials

$$h(x(Q)) = \mathrm{Res}_Y(G, B)$$

$$G(Y) = \prod_{j\in J}\left(x(Q)^2 + \frac{f_1(Y, x([j]P))}{f_0(Y, x([j]P))}x(Q) + \frac{f_2(Y, x([j]P))}{f_0(Y, x([j]P))}\right)$$

$$B(Y) = \prod_{i\in I}(Y - x([i]P))$$

Additional remarks

- Same technique to evaluate the isogeny numerator.
- Similar technique to compute image curve equation.
- Compatible with projective coordinates.
- Similar techniques for y-coordinate (formal derivatives).
- Also applies to kernel points defined over algebraic extensions of k (but in the worst (generic) case it provides no gain)
- Might extend to theta-functions more general than $x(\cdot)$.

Why is this important?

Every efficient isogeny based cryptosystem evaluates tons of isogenies:

SIDH (De Feo, Jao, Plût ’12): only isogenies of degree 2 and 3.

CSIDH (Castryck, Lange, Martindale, Panny, Renes ’18): degrees up to 587.

CSURF (Castryck, Decru ’20): degrees up to 389.

B-SIDH (Costello ’19): degrees in the millions!

others: Galbraith, Petit, Silva ’17,
Delpech de Saint Guilhem, Kutas, Petit, Silva ’19,
…

Why is this important?

Every efficient isogeny based cryptosystem evaluates tons of isogenies:

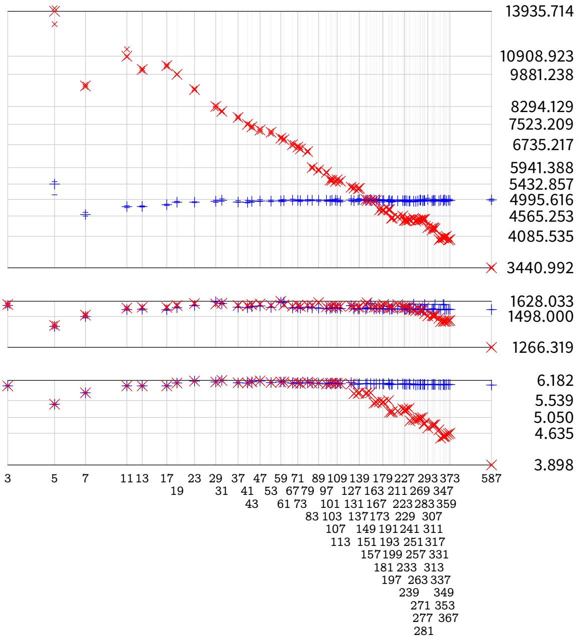
SIDH (De Feo, Jao, Plût ’12): only isogenies of degree 2 and 3. meh

CSIDH (Castryck, Lange, Martindale, Panny, Renes ’18): degrees up to 587. got a chance

CSURF (Castryck, Decru ’20): degrees up to 389. got a chance

B-SIDH (Costello ’19): degrees in the millions! wow!

others: Galbraith, Petit, Silva ’17,
Delpech de Saint Guilhem, Kutas, Petit, Silva ’19, needs work
...



Performance of new (red) against old (blue) algorithm, in three different implementations.

Time to evaluate an isogeny with base field CSIDH-512. x-axis: isogeny degree, y-axis:

Top: Cycle counts of pure C implementation based on Flint.

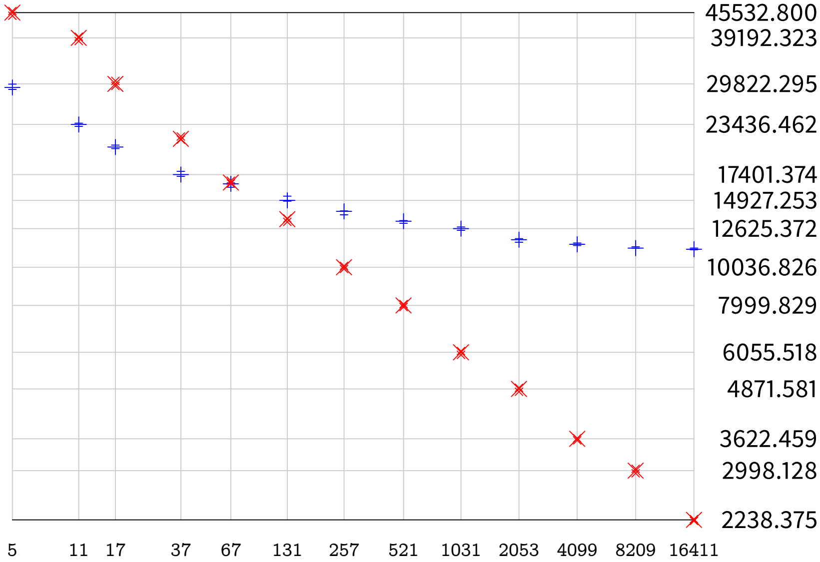
Middle: Cycle counts of assembly optimized implementation based on original CSIDH-512.

Bottom: $\mathbb{F}_p$ multiplication counts of the assembly optimized implementation.

CSIDH results

- Assembly implementation improves CSIDH-512 by 1%, CSIDH-1024 by 8%.
- Slightly smaller gain expected on CSURF-512.
- Not constant-time.
- We only applied algebraic algorithms for polynomial multiplication.
  There is still some room for improvement.

Very large degrees



Performance comparison of new (red) against old (blue) algorithm in a Julia/Nemo implementation on a 256-bits base field.

x-axis: isogeny degree.

y-axis: cycle counts.

B-SIDH results

B-SIDH: a variation on SIDH with optimal field size

- First NIST level 1 secure instantiation of B-SIDH (256 bits base field).
- High level, unoptimized implementation in Julia/Nemo.
- Largest isogeny degree: $6548911 \approx 2^{23}$.
- Alice completes one round of key exchange in 0.56s, against 2s for old algorithm.
- Bob completes one round of key exchange in 10s, against 10 minutes for old algorithm!

Implementation details

Code (5 different implementations!) at: `https://velusqrt.isogeny.org`

- Lots of exploitable symmetries due to the Montgomery form,
- Interesting challenge: fast polynomial multiplication for small degree polynomials over moderately sized fields:
  - Naive, Karatsuba,
  - Middle product algorithms.
  - Toom-Cook? For what sizes?
  - We did not explore Kronecker substitution at all!
- Scaled remainder trees to reduce number of inversions in multi-point evaluation.

Open questions

- Constant time implementation.
- Impact CSIDH? On isogeny action crypto in general?
- What about memory-constrained architectures?
- Lower bounds? See also VDFs...

https://velusqrt.isogeny.org

Thank you

https://defeo.lu/

@luca_defeo