

Isogeny computation in small characteristics

L. De Feo

INRIA Projet TANC & LIX École Polytechnique

March 19, 2010
LORIA, Nancy

What?

Isogenies

(Separable) isogenies: (separable) non-constant regular maps of elliptic curves that are group homomorphism

- Finite kernel,
- onto,
- given by rational fractions.

Multiplication

$$\begin{aligned}[m] : E(\bar{\mathbb{K}}) &\rightarrow E(\bar{\mathbb{K}}) \\ P &\mapsto [m]P\end{aligned}$$

$$\ker \mathcal{I} = E[m].$$

What?

Isogenies

(Separable) isogenies: (separable) non-constant regular maps of elliptic curves that are group homomorphism

- Finite kernel,
- onto,
- given by rational fractions.

Frobenius endomorphism

$$\begin{aligned}\varphi : E(\bar{\mathbb{K}}) &\rightarrow E(\bar{\mathbb{K}}) \\ (X, Y) &\mapsto (X^q, Y^q)\end{aligned}$$

$$\ker \varphi = \{\mathcal{O}\}.$$

What?

Isogenies

(Separable) isogenies: (separable) non-constant regular maps of elliptic curves that are group homomorphism

- Finite kernel,
- onto,
- given by rational fractions.

Separable isogenies, odd degree (simplified Weierstrass model)

$$\mathcal{I}(X, Y) = \left(\frac{g(X)}{h^2(X)}, cY \left(\frac{g(X)}{h^2(X)} \right)' \right)$$

$$\ell = \deg \mathcal{I} = \# \ker \mathcal{I} = 2 \deg h + 1 \text{ odd.}$$

What?

Isogenies

(Separable) isogenies: (separable) non-constant regular maps of elliptic curves that are group homomorphism

- Finite kernel,
- onto,
- given by rational fractions.

Normalised (or strict) isogenies

$$\mathcal{I}(X, Y) = \left(\frac{g(X)}{h^2(X)}, {}^cY \left(\frac{g(X)}{h^2(X)} \right)' \right)$$

$$\ell = \deg \mathcal{I} = \# \ker \mathcal{I} = 2 \deg h + 1 \text{ odd.}$$

What?

Isogenies

(Separable) isogenies: (separable) non-constant regular maps of elliptic curves that are group homomorphism

- Finite kernel,
- onto,
- given by rational fractions.

Normalised (or strict) isogenies

$$\mathcal{I}(X, Y) = \left(\frac{g(X)}{h^2(X)}, Y \left(\frac{g(X)}{h^2(X)} \right)' \right)$$

$$\ell = \deg \mathcal{I} = \# \ker \mathcal{I} = 2 \deg h + 1 \text{ odd.}$$

Why?

Cryptanalysis

- Proving hardness of discrete logarithm ([Jao, Miller, Venkatesan '05]).
- Move discrete logarithms to easier curves ([Gaudry, Hess, Smart '02]).
- Discrete logarithms in genus 3 ([Smith '08]).

Cryptography

- Point counting ([Schoof '95]).
- Speeding up point multiplication ([Gallant, Lambert, Vanstone '01]).
- Hide weak curves behind chains of isogenies ([Teske '06]).
- Define hash functions ([Charles, Lauter, Goren '09]).

Vélu formula

Vélu formula for algebraically closed fields

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

H a subgroup of E , then E/H is an elliptic curve. $\mathcal{I} : E \rightarrow E/H$ given by

$$\mathcal{I}(\mathcal{O}_E) = \mathcal{I}(\mathcal{O}_{E/H})$$

$$\mathcal{I}(P) = \left(x(P) + \sum_{Q \in H - \{\mathcal{O}_E\}} x(P + Q) - x(Q) \quad , \right. \\ \left. y(P) + \sum_{Q \in H - \{\mathcal{O}_E\}} y(P + Q) - y(Q) \right).$$

$E' = E/H$ is recovered through simple formulae. This is a normalised isogeny.

Rational isogenies on non-algebraically closed fields

Knowing $h^2(X) = \prod_{Q \in H - \{\mathcal{O}_E\}} (X - x(Q))$ is enough.

Computing isogenies: which problem?

Modular polynomial

$$\Phi_\ell(j(E), j(E')) = 0 \quad \text{iff } E \text{ } \ell\text{-isogenous to } E'$$

- Bivariate symmetric polynomial, degree ℓ , integer coefficients of $\tilde{O}(\ell)$ bits.
- Computed in $\tilde{O}(\ell^3)$ bit operations (quasi-optimal).

Which problem?

- 1 Given E , find an ℓ -isogenous curve and an ℓ -isogeny.
 - 2 Given E and E' , find an ℓ -isogeny.
 - 3 Given E and E' , find, if it exists, an isogeny of degree up to ℓ .
- Traditional solution to 1: find a curve by factoring $\Phi_\ell(X, j(E))$, then solve 2.
 - In SEA one needs 1, other applications require 2 or 3.
 - We'll focus on 2 and quickly discuss 3.

Computing isogenies in \mathbb{C}

Elliptic functions

$$E \cong \mathbb{C} / (\omega_1 \mathbb{Z} + \omega_2 \mathbb{Z}) \xrightarrow{\mathcal{I}} \mathbb{C} / \left(\frac{\omega_1}{\ell} \mathbb{Z} + \omega_2 \mathbb{Z} \right) \cong E'$$
$$z \longmapsto z$$

Weierstrass functions

$$\wp_E(z) = z^{-2} + \sum_{k=1}^{\infty} c_k z^{2k} \quad \text{with}$$

$$c_1 = -\frac{a}{5}, \quad c_2 = -\frac{b}{7}, \quad c_k = \frac{3}{(k-2)(2k+3)} \sum_{j=1}^{k-2} c_j c_{k-1-j}$$

and they verify

$$\left\{ \begin{array}{l} \wp_E'^2 = 4\wp_E^3 + 4a\wp_E + 4b, \\ \wp_{E'}(z) = \sum_{i=0}^{\ell-1} \wp_E \left(z + i \frac{\omega_1}{\ell} \right) - \wp_E \left(i \frac{\omega_1}{\ell} \right). \end{array} \right.$$

The large characteristic case

Weierstrass functions

$$\wp_E(z) = z^{-2} + \sum_{k=1}^{\infty} c_k z^{2k} \quad \text{with}$$

$$c_1 = -\frac{a}{5}, \quad c_2 = -\frac{b}{7}, \quad c_k = \frac{3}{(k-2)(2k+3)} \sum_{j=1}^{k-2} c_j c_{k-1-j}$$

division by zero when $2k+3 \geq p$.

Large characteristic algorithms

Work with truncated power series with precision $\ll \frac{p}{2}$.

'91 Charlap, Coley, Robbins

$O(\ell^2)$

'92 Elkies

$\tilde{O}(\ell^2)$

'92 Atkin

$\tilde{O}(\ell^2)$

'98 Elkies

$\tilde{O}(\ell^2)$

'08 Bostan, Morain, Salvy, Schost

$\tilde{O}(\ell)$

The large characteristic case

Weierstrass functions

$$\wp_E(z) = z^{-2} + \sum_{k=1}^{\infty} c_k z^{2k} \quad \text{with}$$

$$c_1 = -\frac{a}{5}, \quad c_2 = -\frac{b}{7}, \quad c_k = \frac{3}{(k-2)(2k+3)} \sum_{j=1}^{k-2} c_j c_{k-1-j}$$

division by zero when $2k+3 \geq p$.

Large characteristic algorithms ***Only work for normalised isogenies**

Work with truncated power series with precision $\ll \frac{p}{2}$.

'91 Charlap, Coley, Robbins*

$O(\ell^2)$

'92 Elkies

$\tilde{O}(\ell^2)$

'92 Atkin

$\tilde{O}(\ell^2)$

'98 Elkies

$\tilde{O}(\ell^2)$

'08 Bostan, Morain, Salvy, Schost*

$\tilde{O}(\ell)$

Differential equations

- Set $R_E(z) = \frac{1}{\sqrt{\phi_E(z)}}$, then $R'_E(z)^2 = bR_E(z)^6 + aR_E(z)^4 + 1$.
- Let $\mathcal{I} : E \rightarrow E'$ such that $\mathcal{I}(x, y) = \left(\frac{N(x)}{D(x)}, y \left(\frac{N(x)}{D(x)} \right)' \right)$,
- then $(x^3 + ax + b) \left(\frac{N(x)}{D(x)} \right)'^2 = \left(\frac{N(x)}{D(x)} \right)^3 + a' \frac{N(x)}{D(x)} + b'$.
- Set $S(x) = \sqrt{\frac{D(x^2)}{N(x^2)}}$ so that $R_{E'} = S \circ R_E$,
- then $(bx^6 + ax^4 + 1)S'(x)^2 = 1 + aS(x)^4 + bS(x)^6$

Algorithm

- Compute $S(x)$ using fast computer algebra techniques, deduce T such that $S(x) = xT(x^2)$,
- Compute $U(x) = 1/T(x)^2$,
- Use rational fraction reconstruction on U to deduce $\frac{N(x)}{D(x)} = xU\left(\frac{1}{x}\right)$.

BMSS, non-strict isogenies

Differential equations

- Set $R_E(z) = \frac{1}{\sqrt{\phi_E(z)}}$, then $R'_E(z)^2 = bR_E(z)^6 + aR_E(z)^4 + 1$.
- Let $\mathcal{I} : E \rightarrow E'$ such that $\mathcal{I}(x, y) = \left(\frac{N(x)}{D(x)}, y \left(\frac{N(x)}{D(x)} \right)' \right)$,
- then $(x^3 + ax + b) \left(\frac{N(x)}{D(x)} \right)'^2 = \left(\frac{N(x)}{D(x)} \right)^3 + a' \frac{N(x)}{D(x)} + b'$.
- Set $S(x) = \sqrt{\frac{D(x^2)}{N(x^2)}}$ so that $R_{E'} = S \circ R_E$,
- then $(bx^6 + ax^4 + 1)S'(x)^2 = 1 + aS(x)^4 + bS(x)^6$

Algorithm

- Compute $S(x)$ using fast computer algebra techniques, deduce T such that $S(x) = xT(x^2)$,
- Compute $U(x) = 1/T(x)^2$,
- Use rational fraction reconstruction on U to deduce $\frac{N(x)}{D(x)} = xU\left(\frac{1}{x}\right)$.

BMSS, non-strict isogenies

The strict case

- BMSS only works if $\mathcal{I} : E \rightarrow E'$ is strict,
- suitable for SEA, complexity $\tilde{O}(\ell)$.

In general

- Use modular polynomial to find $\tilde{E} \cong E'$ such that $\tilde{\mathcal{I}} : E \rightarrow \tilde{E}$ is strict,
- use BMSS, use isomorphism to deduce \mathcal{I} .
- Complexity $O(\ell^3)$, fast in practice.

Generalisation of BMSS

- BMSS fails to solve the d.e. when $\ell \gg p$,
- do computations in \mathbb{Z}_p to avoid divisions by zero,
- only use p -adic precision of $O(\log^2 \ell)$.
- Works for any p , complexity $\tilde{O}(\ell^3)$, but solves problem 1 directly.
- Fast in practice: after evaluation of Φ_ℓ , only $\tilde{O}(\ell)$ operations.

The algorithm

- Lift E to \bar{E} in \mathbb{Q}_q .
- Problem: the lift of E' is not necessarily normalised.
- Lift Φ_ℓ , factor $\bar{\Phi}_\ell(X, j(\bar{E}))$ to obtain a normalised \bar{E}' ,
- use BMMS to compute the lifted isogeny, then reduce.

The small characteristic case

Other algorithms

'94 Couveignes I	$O(\ell^3)$
'96 $p = 2$, Lercier	$\Omega(\ell^3)$?
'96 Couveignes II (+ D.F., Schost)	$\tilde{O}(\ell^2)$

Couveignes I

- Uses formal groups parametrization in place of Weierstrass functions,
- computes all the possible morphisms of formal groups up to a bounded precision,
- reconstructs a rational fraction and tests if it is an isogeny.

The small characteristic case

Other algorithms

'94 Couveignes I	$O(\ell^3)$
'96 $p = 2$, Lercier	$\Omega(\ell^3) ?$
'96 Couveignes II (+ D.F., Schost)	$\tilde{O}(\ell^2)$

Couveignes II

- Exploits the cyclic structure of the p^k -torsion,
- interpolates a polynomial over $E[p^k]$,
- reconstructs a rational fraction and tests if it is an isogeny.
- Uses fast computer algebra techniques.

Structure of the p^k -torsion

$$\begin{array}{c} \mathbb{U}_k \\ \left| \begin{array}{c} p \\ \vdots \\ p \end{array} \right. \\ \mathbb{U}_{k-1} \\ \vdots \\ \mathbb{U}_{i_0+1} \\ \left| \begin{array}{c} p \\ \vdots \\ p \end{array} \right. \\ \mathbb{U}_{i_0} \\ \vdots \\ \mathbb{U}_1 \\ \left| \begin{array}{c} 1 \\ \vdots \\ 1 \end{array} \right. \\ \mathbb{U}_0 = \mathbb{F}_q \end{array}$$

Computing the p^i -torsion

- Iteratively, inverting the map $[p]$,
- Voloch Formula:** $X^p - X = \frac{p\sqrt{y_P\beta(x_P)}}{h}.$

Definition (p^k -torsion tower)

$(\mathbb{F}_q = \mathbb{U}_0, \dots, \mathbb{U}_k)$ is the tower of field extensions of minimal degree s.t. for any i

$$E[p^i] \subset E(\mathbb{U}_i).$$

Theorem (Structure of $(\mathbb{U}_0, \dots, \mathbb{U}_k)$)

There is a $i_0 \geq 1$ s.t. $\mathbb{U}_{i_0} = \mathbb{U}_1$ and for $i \geq i_0$

$$[\mathbb{U}_{i+1} : \mathbb{U}_i] = p.$$

And $[\mathbb{U}_1 : \mathbb{U}_0]$ divides $p - 1$.

Summarizing

Couveignes' algorithm

- ① Compute a p -torsion point of E ,
- ② repeatedly apply Voloch formulae to compute P , a p^k -torsion point of E ,
- ③ do the same to compute P' , a p^k -torsion point of E' ,
- ④ for $i \in [1, \dots, p^k - 1]$, i prime to p
 - ① interpolate the polynomial that sends P over $[i]P'$,
 - ② deduce a rational fraction and check if its denominator is a square.

Informal cost analysis

- To have enough points $\phi(p^k) > 4\ell$, then $[\mathbb{U}_k : \mathbb{F}_q] \sim p^k \sim \ell$.
- Step 1 is easy, step 2 costs $O(p^k)$ operations in the tower.
- Step 3 requires factorisation in \mathbb{U}_k . Cost is $O(p^{3k})$ by linear algebra.
- Steps 4.1 and 4.2 have to be repeated $\phi(p^k)$ times.
- Step 4.1 interpolates a polynomial of degree $\phi(p^k)$ in a field of degree p^{k-1} . That is $O(p^{2k})$ by fast techniques. Step 4.2 is some GCDs in \mathbb{F}_q , costs $O(p^k)$.

Summarizing

Couveignes' algorithm

- ① Compute a p -torsion point of E ,
- ② repeatedly apply Voloch formulae to compute P , a p^k -torsion point of E ,
- ③ do the same to compute P' , a p^k -torsion point of E' ,
- ④ for $i \in [1, \dots, p^k - 1]$, i prime to p
 - ① interpolate the polynomial that sends P over $[i]P'$,
 - ② deduce a rational fraction and check if its denominator is a square.

Informal cost analysis

- To have enough points $\phi(p^k) > 4\ell$, then $[\mathbb{U}_k : \mathbb{F}_q] \sim p^k \sim \ell$.
- Step 1 is easy, step 2 costs $O(p^k)$ operations in the tower.
- Step 3 requires factorisation in \mathbb{U}_k . Cost is $O(p^{3k})$ by linear algebra.
- Steps 4.1 and 4.2 have to be repeated $\phi(p^k)$ times.
- Step 4.1 interpolates a polynomial of degree $\phi(p^k)$ in a field of degree p^{k-1} . That is $O(p^{2k})$ by fast techniques. Step 4.2 is some GCDs in \mathbb{F}_q , costs $O(p^k)$.
- Total cost is $O(p^{3k}) = O(\ell^3)$.

Improving the isomorphism

Couveignes' algorithm

- ① Compute a p -torsion point of E ,
- ② repeatedly apply Voloch formulae to compute P , a p^k -torsion point of E ,
- ③ do the same to compute P' , a p^k -torsion point of E' ,
- ④ for $i \in [1, \dots, p^k - 1]$, i prime to p
 - ① interpolate the polynomial that sends P over $[i]P'$,
 - ② deduce a rational fraction and check if its denominator is a square.

Informal cost analysis

- Step 3 requires factorisation in \mathbb{U}_k . Cost is $O(p^{3k})$ by linear algebra.
- [Couveignes '00] gives an algorithm with cost $O(p^k)$ operations in the tower.

Artin-Schreier towers

$$\mathbb{U}_k = \frac{\mathbb{U}_{k-1}[X_k]}{P_{k-1}(X_k)}$$

$\left| \begin{array}{c} p \end{array} \right.$

$$\mathbb{U}_{k-1}$$

\vdots

$$\mathbb{U}_{i_0+1} = \frac{\mathbb{U}_0[X_1]}{P_0(X_1)}$$

$\left| \begin{array}{c} p \end{array} \right.$

$$\mathbb{U}_{i_0} = \mathbb{F}_q = \frac{\mathbb{F}_p[X_0]}{Q(X_0)}$$

Artin-Schreier Towers over finite fields

$$P_i = X^p - X - \alpha_i$$

We say that $(\mathbb{U}_0, \dots, \mathbb{U}_k)$ is defined by $(\alpha_0, \dots, \alpha_{k-1})$ over \mathbb{U}_{i_0} .

ANY separable extension of degree p can be expressed this way

Voloch formulae

Remark that Voloch formulae give rise to an Artin-Schreier tower:

$$X^p - X = \frac{\sqrt[p]{y_P \beta(x_p)}}{h}$$

Solving Artin-Schreier equations in Artin-Schreier towers

[Couveignes '00]

- Given $\alpha_i \in \mathbb{U}_i$ solves

$$X^p - X = \alpha_i \in \mathbb{U}_i.$$

- By a change of variables, this is equivalent to solve

$$X^p - X = \beta_i \in \mathbb{U}_{i-1}.$$

- Applies the formula recursively. Complexity is $O(p^i)$.

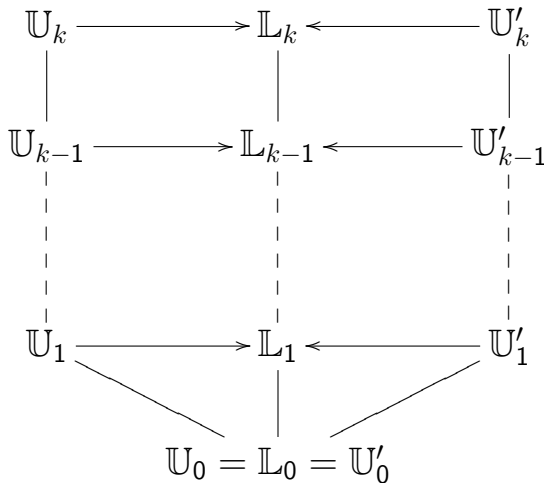
Isomorphisms of Artin-Schreier towers

- Equivalently, the algorithm finds an isomorphism between $(\mathbb{U}_0, \dots, \mathbb{U}_k)$ and the tower defined by $(\alpha_0, \dots, \alpha_{k-1})$.
- If there were a third tower $(\mathbb{L}_0, \dots, \mathbb{L}_k)$ with fast arithmetics...

\mathbb{U}_k
|
 \mathbb{U}_{k-1}
|
...
|
 \mathbb{U}_1
|
 \mathbb{U}_0

\mathbb{U}'_k
|
 \mathbb{U}'_{k-1}
|
...
|
 \mathbb{U}'_1
|
 \mathbb{U}'_0

Solving Artin-Schreier equations in Artin-Schreier towers



Improving the arithmetics

Couveignes' algorithm

- ① Compute a p -torsion point of E ,
- ② repeatedly apply Voloch formulae to compute P , a p^k -torsion point of E ,
- ③ do the same to compute P' , a p^k -torsion point of E' ,
- ④ for $i \in [1, \dots, p^k - 1]$, i prime to p
 - ① interpolate the polynomial that sends P over $[i]P'$,
 - ② deduce a rational fraction and check if its denominator is a square.

Informal cost analysis

- To have enough points $\phi(p^k) > 4\ell$, then $[\mathbb{U}_k : \mathbb{F}_q] \sim p^k \sim \ell$.
- Step 1 is easy, step 2 costs $O(p^k)$ operations in the tower.
- Step 3 requires factorisation in \mathbb{U}_k . Cost is $O(p^k)$ ops by [Couveignes '00].
- Steps 4.1 and 4.2 have to be repeated $\phi(p^k)$ times.
- Step 4.1 interpolates a polynomial of degree $\phi(p^k)$ in a field of degree p^{k-1} . That is $O(p^{2k})$ operations. Step 4.2 is some GCDs in \mathbb{F}_q , costs $O(p^k)$ ops.

Improving the arithmetics

Couveignes' algorithm

- ① Compute a p -torsion point of E ,
- ② repeatedly apply Voloch formulae to compute P , a p^k -torsion point of E ,
- ③ do the same to compute P' , a p^k -torsion point of E' ,
- ④ for $i \in [1, \dots, p^k - 1]$, i prime to p
 - ① interpolate the polynomial that sends P over $[i]P'$,
 - ② deduce a rational fraction and check if its denominator is a square.

Informal cost analysis

- To have enough points $\phi(p^k) > 4\ell$, then $[\mathbb{U}_k : \mathbb{F}_q] \sim p^k \sim \ell$.
- Step 1 is easy, step 2 costs $O(p^k)$ operations in the tower.
- Step 3 requires factorisation in \mathbb{U}_k . Cost is $O(p^k)$ ops by [Couveignes '00].
- Steps 4.1 and 4.2 have to be repeated $\phi(p^k)$ times.
- Step 4.1 interpolates a polynomial of degree $\phi(p^k)$ in a field of degree p^{k-1} . That is $O(p^{2k})$ operations. Step 4.2 is some GCDs in \mathbb{F}_q , costs $O(p^k)$ ops.
- But how much does it cost one operation?

Fast arithmetics in Artin-Schreier towers



Primitive towers ([D.F., Schost '10])

- Find special $(\gamma_0, \dots, \gamma_{k-1})$ that define a tower s.t. $\mathbb{L}_i = \mathbb{F}_p[x_i]$, where $x_i^p - x_i - \gamma_{i-1} = 0$.
- Use univariate representation over \mathbb{F}_p to perform fast arithmetics (FFT multiplication, Newton inversion, etc.).
- Use [Couveignes '00] algorithm to move to (U_0, \dots, U_k) .

Level embedding ([D.F., Schost '10])

- Express the morphisms between the levels to switch back to the multivariate representation.
- Going down is easy: bivariate reduction modulo $X_i^p - X_i - \gamma_{i-1}$.
- Going up much harder: trace formulae, truncated power series arithmetics, transposition principle.

Advertisement: FFAST

Download this C++ library at:

<http://www.lix.polytechnique.fr/Labo/Luca.De-Feo/FFAST>

Improving the interpolation

Couveignes' algorithm

- ① Compute a p -torsion point of E ,
- ② repeatedly apply Voloch formulae to compute P , a p^k -torsion point of E ,
- ③ do the same to compute P' , a p^k -torsion point of E' ,
- ④ for $i \in [1, \dots, p^k - 1]$, i prime to p
 - ① interpolate the polynomial that sends P over $[i]P'$,
 - ② deduce a rational fraction and check if its denominator is a square.

Formal cost analysis

- To have enough points $\phi(p^k) > 4\ell$, then $[\mathbb{U}_k : \mathbb{F}_q] \sim p^k \sim \ell$.
- Step 1 is easy, step 2 costs $O(p^k \log_p q)$ operations.
- Step 3 requires factorisation in \mathbb{U}_k . Cost is $O(p^k \log_p^2 q + \log_p^3 q)$.
- Steps 4.1 and 4.2 have to be repeated $\phi(p^k)$ times.
- Step 4.1 interpolates a polynomial of degree $\phi(p^k)$ in a field of degree p^{k-1} . That is $O(p^{2k} \log_p q)$. Step 4.2 is some GCDs in \mathbb{F}_q , costs $O(p^k \log_p q)$.
- All costs in \mathbb{F}_p -operations.

Improving the interpolation

Couveignes' algorithm

- ① Compute a p -torsion point of E ,
- ② repeatedly apply Voloch formulae to compute P , a p^k -torsion point of E ,
- ③ do the same to compute P' , a p^k -torsion point of E' ,
- ④ for $i \in [1, \dots, p^k - 1]$, i prime to p
 - ① interpolate the polynomial that sends P over $[i]P'$,
 - ② deduce a rational fraction and check if its denominator is a square.

Formal cost analysis

- To have enough points $\phi(p^k) > 4\ell$, then $[\mathbb{U}_k : \mathbb{F}_q] \sim p^k \sim \ell$.
- Step 1 is easy, step 2 costs $O(p^k \log_p q)$ operations.
- Step 3 requires factorisation in \mathbb{U}_k . Cost is $O(p^k \log_p^2 q + \log_p^3 q)$.
- Steps 4.1 and 4.2 have to be repeated $\phi(p^k)$ times.
- Step 4.1 interpolates a polynomial of degree $\phi(p^k)$ in a field of degree p^{k-1} . That is $O(p^{2k} \log_p q)$. Step 4.2 is some GCDs in \mathbb{F}_q , costs $O(p^k \log_p q)$.
- All costs in \mathbb{F}_p -operations.

Faster interpolation using effective Galois groups

Interpolation of $v_i \mapsto s_i$ is defined modulo T , where

$$T(X) = \prod_i (X - v_i)$$

Lagrange formula

$$A(X) = \sum_{i=1}^n v_i \frac{T(X)}{X - v_i} \prod_{j \neq i} \frac{1}{v_i - v_j}$$

Faster interpolation using effective Galois groups

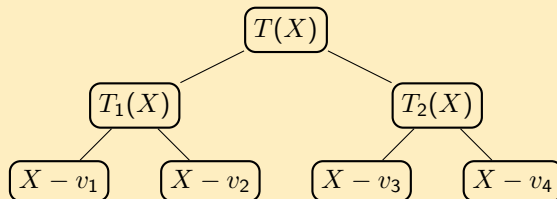
Interpolation of $v_i \mapsto s_i$ is defined modulo T , where

$$T(X) = \prod_i (X - v_i)$$

Lagrange formula

$$A(X) = \sum_{i=1}^n \frac{s_i}{T'(v_i)} \cdot \frac{T(X)}{X - v_i}$$

Interpolation by chinese remaindering



Faster interpolation using effective Galois groups

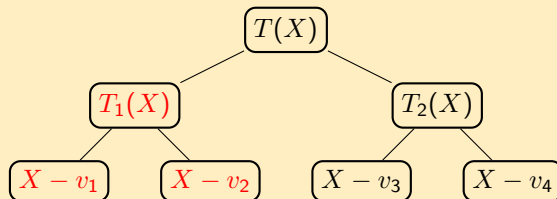
Interpolation of $v_i \mapsto s_i$ is defined modulo T , where

$$T(X) = \prod_i (X - v_i)$$

Lagrange formula

$$A_1(X) = \frac{s_1}{T'(v_1)(X - v_2)} + \frac{s_2}{T'(v_2)(X - v_1)}$$

Interpolation by chinese remaindering



Faster interpolation using effective Galois groups

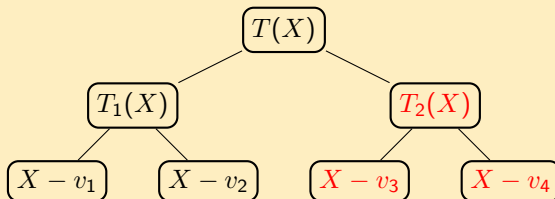
Interpolation of $v_i \mapsto s_i$ is defined modulo T , where

$$T(X) = \prod_i (X - v_i)$$

Lagrange formula

$$A_2(X) = \frac{s_3}{T'(v_3)(X - v_4)} + \frac{s_4}{T'(v_4)(X - v_3)}$$

Interpolation by chinese remaindering



Faster interpolation using effective Galois groups

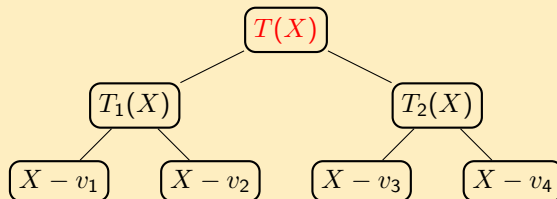
Interpolation of $v_i \mapsto s_i$ is defined modulo T , where

$$T(X) = \prod_i (X - v_i)$$

Lagrange formula

$$A(X) = T_2(X)A_1(X) + T_1(X)A_2(X)$$

Interpolation by chinese remaindering



Faster interpolation using effective Galois groups

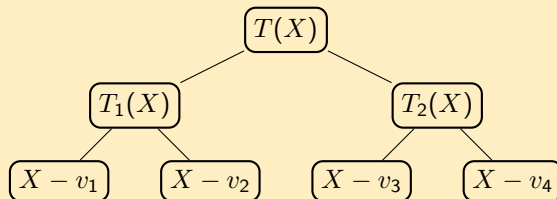
Interpolation of $v_i \mapsto s_i$ is defined modulo T , where

$$T(X) = \prod_i (X - v_i)$$

Lagrange formula

Complexity $\tilde{O}(n)$ operations in the coefficient ring.

Interpolation by chinese remaindering



Faster interpolation using effective Galois groups

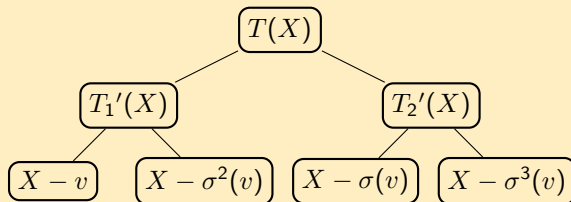
Interpolation of $v_i \mapsto s_i$ is defined modulo T , where

$$T(X) = \prod_i (X - v_i)$$

Lagrange formula

Let now $v \in \mathbb{U}_2$, $\sigma \in \text{Gal}(\mathbb{U}_2/\mathbb{U}_0)$ and $v_i = \sigma^{i-1}(v)$. Rearrange the tree, then $T'_1, T'_2 \in \mathbb{U}_1[X]$ and $T \in \mathbb{U}_0[X]$.

Interpolation by chinese remaindering



Faster interpolation using effective Galois groups

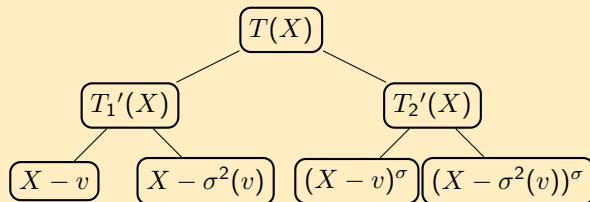
Interpolation of $v_i \mapsto s_i$ is defined modulo T , where

$$T(X) = \prod_i (X - v_i)$$

Lagrange formula

For a polynomial P note P^σ the action on the coefficients of P , then

Interpolation by chinese reminding



Faster interpolation using effective Galois groups

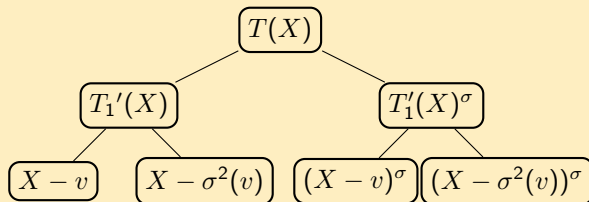
Interpolation of $v_i \mapsto s_i$ is defined modulo T , where

$$T(X) = \prod_i (X - v_i)$$

Lagrange formula

For a polynomial P note P^σ the action on the coefficients of P , then

Interpolation by chinese reminding



Faster interpolation using effective Galois groups

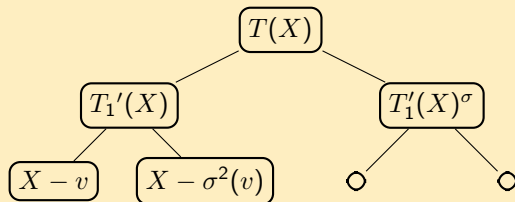
Interpolation of $v_i \mapsto s_i$ is defined modulo T , where

$$T(X) = \prod_i (X - v_i)$$

Lagrange formula

Complexity $\tilde{O}(n)$ operations in \mathbb{U}_0 .

Interpolation by chinese reminding



Summarizing

Couveignes' algorithm

- ① Compute a p -torsion point of E ,
- ② repeatedly apply Voloch formulae to compute P , a p^k -torsion point of E ,
- ③ do the same to compute P' , a p^k -torsion point of E' ,
- ④ for $i \in [1, \dots, p^k - 1]$, i prime to p
 - ① interpolate the polynomial that sends P over $[i]P'$,
 - ② deduce a rational fraction and check if its denominator is a square.

Formal cost analysis

- To have enough points $\phi(p^k) > 4\ell$, then $[\mathbb{U}_k : \mathbb{F}_q] \sim p^k \sim \ell$.
- Step 1 is easy, step 2 costs $O(p^k \log_p q)$ operations.
- Step 3 requires factorisation in \mathbb{U}_k . Cost is $O(p^k \log_p^2 q + \log_p^3 q)$.
- Steps 4.1 and 4.2 have to be repeated $\phi(p^k)$ times.
- Step 4.1 costs $O(p^k \log_p q)$ using the latter algorithm. Step 4.2 is some GCDs in \mathbb{F}_q , costs $O(p^k \log_p q)$.
- Total cost is $O(\ell^2 \log_p q + \ell \log_p^2 q + \log_p^3 q)$.

Reducing the number of interpolations

Couveignes' algorithm

- We need $O(p^k)$ interpolations, each sending $P \in E[p^k]$ over $Q \in E'[p^k]$,
- $Q, R \in E'[p^k]$ are conjugates tied by the relation $Q = \varphi^i(P)$ for some i .

Using modular composition

Let A_Q be the polynomial with coefficients in \mathbb{F}_q sending P over Q , then

$$A_Q([j]P) = [j]Q \text{ for every } j.$$

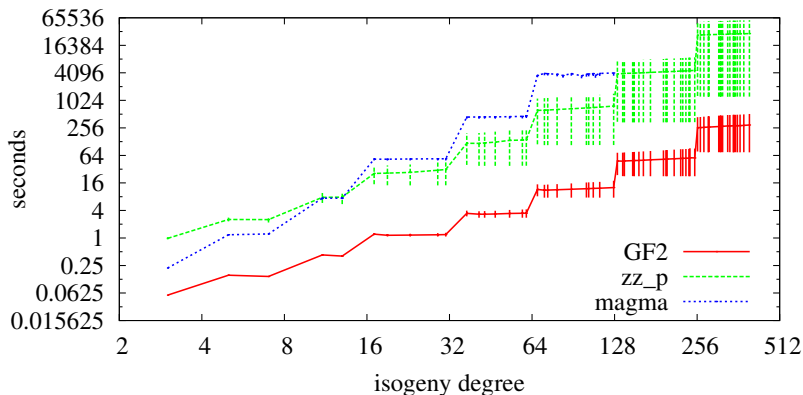
Now let $\varphi_q(Q) = [\lambda]Q$, then $A_Q(\varphi_q([j]P)) = [j][\lambda]Q$.

So $A_Q \circ \varphi_q = A_{[\lambda]Q} \bmod T$. Solving this is *modular composition*.

Modular composition

- Theoretical complexity $O(\ell \log_p q)$, practical complexity $O(\ell^2 \log_p^2 q) \dots$
- ...but still much faster than a single interpolation.

Timings



Comparison of Couveignes II implementations

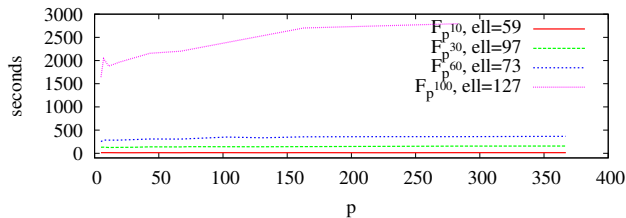
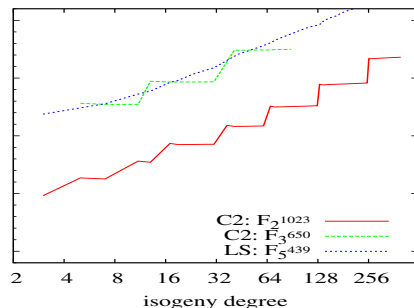
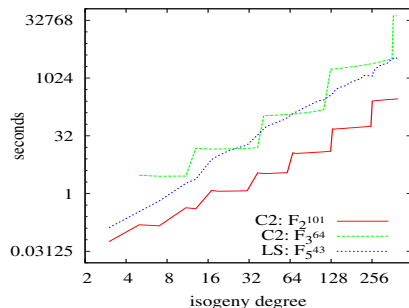
- NTL + gf2x,
- NTL with zz_p,
- Magma (no fast Artin-Schreier).

Timings

ℓ	$E[p^k]$	$E'[p^k]$	FI	RFR	MC	Avg tries	Avg loop time
31	1.3128	1.3128	1.1058	0.00218	0.00218	64	0.279
61	3.5454	3.5464	2.5236	0.00783	0.00900	128	2.154
127	9.2975	9.3026	5.6881	0.03147	0.03634	256	17.359
251	23.7984	23.7984	12.7251	0.12415	0.14519	512	137.902
397	59.7439	59.7579	28.3387	0.36822	0.58027	1024	971.254

Table: Comparative timings for the phases of C2-AS-FI-MC for curves over $\mathbb{F}_{2^{101}}$ using NTL + gf2x.

Comparison with Lercier-Sirvent



Ongoing work

Implementation (with F. Morain and E. Schost)





- SAGE porting of FFAST,
- SAGE porting of SEA + Lercier + Couveignes II,
- comparison with Lercier,
- comparison with Lercier-Sirvent.

Theory

- Try a p -adic version of Couveignes II + BMSS08 to reduce the number of tries in the final loop,
- Improve Lercier-Sirvent and make it the best algorithm for this problem.

Thanks

Bibliography

-  I. Blake, G. Seroussi & N. Smart
Elliptic Curves in Cryptography
LMS 265, Cambridge University Press, 1999
-  (edited by) I. Blake, G. Seroussi & N. Smart
Advances in Elliptic Curve Cryptography
LMS 317, Cambridge University Press, 2005
-  J.S. Milne.
Elliptic curves.
BookSurge Publishers, ISBN 1-4196-5257-5, 2006.
-  J.H. Silverman
The Arithmetic of Elliptic Curves
GTM 106, Springer-Verlag, 1986

Bibliography



A. Bostan, F. Morain, B. Salvy, É. Schost.

Fast algorithms for computing isogenies between elliptic curves.

Math. Comp. 77, 263, 1755-1778, 2008.



D.X. Charles, K.E. Lauter & E.Z. Goren.

Cryptographic Hash Functions from Expander Graphs.

J. Cryptology 22:93–113, 2009.



J.-M. Couveignes.

Computing ℓ -isogenies with the p -torsion.

Lecture Notes in Computer Science vol. 1122, pages 59–65, Springer-Verlag, 1996.



J.-M. Couveignes.

Isomorphisms between Artin-Schreier tower.

Math. Comp. 69(232): 1625–1631, 2000.



L. De Feo.

Calcul d'isogénies.

Master thesis. <http://www.lix.polytechnique.fr/Labo/Luca.De-Feo/>

Bibliography



L. De Feo.

Fast algorithms for computing isogenies between ordinary elliptic curves in small characteristic.

Preprint, 2010.



L. De Feo & É. Schost.

Fast arithmetics in Artin-Schreier towers over finite fields.

Preprint, 2010.



R.P. Gallant, R.J. Lambert & S.A. Vanstone.

Faster Point Multiplication on Elliptic Curves with Efficient Endomorphisms.

CRYPTO '01, LNCS, pages 190–200, Springer, 2001.



P. Gaudry, F. Hess, N. P. Smart.

Constructive and destructive facets of Weil descent on elliptic curves.

J. Cryptology 15:19-46, 2002.



D. Jao, S.D. Miller & R. Venkatesan,

Do All Elliptic Curves of the Same Order Have the Same Difficulty of Discrete Log?

ASIACRYPT '05, LNCS, pages 21–40, Springer, 2005.

Bibliography



A. Joux, R. Lercier.

Counting points on elliptic curves in medium characteristic.

Cryptology ePrint Archive 2006/176, 2006.



R. Lercier, T. Sirvent.

On Elkies subgroups of ℓ -torsion points in curves defined over a finite field.

To appear *J. de Théorie des Nombres de Bordeaux*.



R. Schoof.

Counting points on elliptic curves over finite fields.

J. de Théorie des Nombres de Bordeaux, 7:219-254, 1995.



B. Smith.

Isogenies and the Discrete Logarithm Problem in Jacobians of genus 3 hyperelliptic curves.

In *EUROCRYPT '08*, LNCS, Springer, 2008.



E. Teske.

Elliptic curve trapdoor system.

J. Cryptology 19:115-133, 2006.