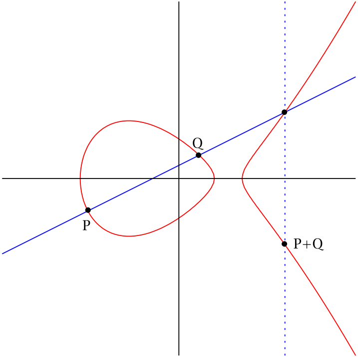
Fast algorithms: from type theory to number theory

Luca De Feo

INRIA Saclay, Projet TANC

October 25, 2010
Séminaire Algorithmes
INRIA Rocquencourt, Le Chesnay

Elliptic curve cryptography



Weierstrass form: $y^2 = x^3 + ax + b$;

Group law: Chord-tangent;

Crypto: Based on discrete log in $E(\mathbb{F}_q)$;

Hasse bound: $|\#E(\mathbb{F}_q) - q - 1| \leqslant 2\sqrt{q}$.

Isogenies

Isogenies are group morphisms of elliptic curves:

$$\mathcal{I} : E \to E'$$

$$\mathcal{I}(x, y) = \left(\frac{g(x)}{h(x)}, cy \left(\frac{g(x)}{h(x)} \right)' \right)$$

What do you do with an isogeny over a finite field?

- Point counting (Schoof 1995);
- Speed up point multiplication (Gallant, Lambert, and Vanstone 2001);
- Reduce a Discrete Logarithm Problem to another (Gaudry, Hess, and Smart 2002; Smith 2009);
- Construct new cryptosystems (Teske 2006; Rostovtsev and Stolbunov 2006);
- Construct hash functions (Charles, Lauter, and Goren 2009).

Isogenies: an example

The GHS attack (Gaudry, Hess, and Smart 2002)

$$E/F_{q^d}$$

- Given an elliptic curve E defined over a composite field $\mathbb{F}_{q^d}$;

Isogenies: an example

The GHS attack (Gaudry, Hess, and Smart 2002)

$$E/F_{q^d} \xrightarrow{\mathcal{I}} H/F_q$$

- Given an elliptic curve E defined over a composite field $\mathbb{F}_{q^d}$;
- Computes an isogeny to an hyperelliptic curve H defined over $\mathbb{F}_q$.
- For certain parameters, the discrete log is easier on H than on E.

Isogenies: an example

The GHS attack (Gaudry, Hess, and Smart 2002)

$$E/F_{q^d} \xrightarrow{\mathcal{J}} H/F_q$$

- Given an elliptic curve E defined over a composite field $\mathbb{F}_{q^d}$;
- Computes an isogeny to an hyperelliptic curve H defined over $\mathbb{F}_q$.
- For certain parameters, the discrete log is easier on H than on E.

A trapdoor cryptosystem (Teske 2006)

Fact: Only a small fraction of the curves over $\mathbb{F}_{q^d}$ is vulnerable to GHS

$$E_{\text{trap}}$$

- Select a curve E_{trap} vulnerable to GHS;

Isogenies: an example

The GHS attack (Gaudry, Hess, and Smart 2002)

$$E/F_{q^d} \xrightarrow{\mathcal{J}} H/F_q$$

- Given an elliptic curve E defined over a composite field $\mathbb{F}_{q^d}$;
- Computes an isogeny to an hyperelliptic curve H defined over $\mathbb{F}_q$.
- For certain parameters, the discrete log is easier on H than on E.

A trapdoor cryptosystem (Teske 2006)

Fact: Only a small fraction of the curves over $\mathbb{F}_{q^d}$ is vulnerable to GHS

$$\begin{tikzcd}
E_{\mathrm{trap}} \arrow[dr] & & \arrow[dr] & \\
& {} \arrow[ur] & & E_{\mathrm{pub}}
\end{tikzcd}$$

- Select a curve E_{trap} vulnerable to GHS;
- Take a random walk through the *isogeny graph*, land on a curve E_{pub} not vulnerable to GHS;

Isogenies: an example

The GHS attack (Gaudry, Hess, and Smart 2002)

$$E/F_{q^d} \xrightarrow{\mathcal{J}} H/F_q$$

- Given an elliptic curve E defined over a composite field $\mathbb{F}_{q^d}$;
- Computes an isogeny to an hyperelliptic curve H defined over $\mathbb{F}_q$.
- For certain parameters, the discrete log is easier on H than on E.

A trapdoor cryptosystem (Teske 2006)

Fact: Only a small fraction of the curves over $\mathbb{F}_{q^d}$ is vulnerable to GHS

$$E_{\mathrm{trap}} \searrow \nearrow \searrow E_{\mathrm{pub}}$$

- Select a curve E_{trap} vulnerable to GHS;
- Take a random walk through the *isogeny graph*, land on a curve E_{pub} not vulnerable to GHS;
- Use E_{pub} for public key cryptography, give E_{trap} to a *trusted authority* for *key escrow*.

Isogenies: a challenge

Let

$$\mathbb{F}_q = \mathbb{F}_2[Z]/(Z^{41} + Z^3 + 1)$$

The following two curves are isogenous:

$$y^2 + xy = x^3 + 1/(Z^{36} + Z^{35} + Z^{34} + Z^{32} + Z^{31} + Z^{30} + Z^{26} + Z^{23} + Z^{22} + Z^{21} + Z^{20} + Z^{18} + Z^{17} + Z^{13} + Z^{12} + Z^{11} + Z^{8} + Z^{7} + Z^{5} + Z^{4} + Z^{2})$$

$$y^2 + xy = x^3 + 1/(Z^{40} + Z^{39} + Z^{38} + Z^{37} + Z^{35} + Z^{34} + Z^{28} + Z^{22} + Z^{15} + Z^{14} + Z^{11} + Z^{10} + Z^{9} + Z^{8} + Z^{7} + Z^{6} + Z^{5} + Z^{4} + Z)$$

- Can you tell of what degree (i.e. size of the kernel)?
- Can you compute the isogeny?

Plan

1. Transposition principle

2. Artin-Schreier towers

3. Isogenies

The transposition principle

"Let $\mathcal{P}$ be an arbitrary set. To any R*-algebraic algorithm* A *computing a family of linear functions* $(f_p : M \to N)_{p\in\mathcal{P}}$ *corresponds an* R*-algebraic algorithm* A^* *computing the dual family* $(f_p^* : N^* \to M^*)_{p\in\mathcal{P}}$*. The algebraic time and space complexities of* A^* *are bounded by the time complexity of* A*."*

The dual of a diagram

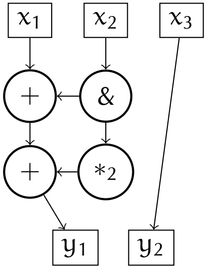
$$
\begin{tikzcd}
A \arrow[r] \arrow[d, "f"'] & B \arrow[drr] & & \\
C \arrow[r, "g"'] & D \arrow[rr, "h"'] & & E
\end{tikzcd}
\quad \Longrightarrow \quad
\begin{tikzcd}
A^* & B^* \arrow[l] & & \\
C^* \arrow[u, "f^*"] & D^* \arrow[l, "g^*"] & & E^* \arrow[ll, "h^*"] \arrow[ull]
\end{tikzcd}
$$

Duality and complexity

- $(f \circ g \circ h)^* = h^* \circ g^* \circ f^*$;
- * is *contravariant*;
- A classical example is transposition of matrices: $(AB)^\top = B^\top A^\top$;
- From an algorithmic point of view, the number of *arrows* is a measure of complexity, and it is preserved under dualization.

Transposition of arithmetic circuits

Arithmetic circuits are like diagrams enriched with a *product*. In particular they can be *transposed*:



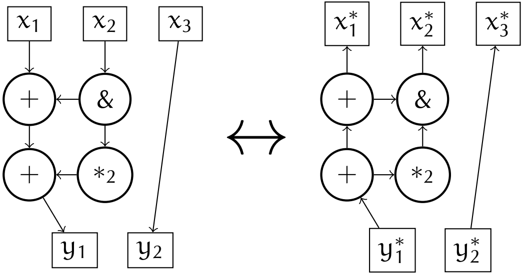
$$y_1 = x_1 + 3x_2$$
$$y_2 = x_3$$

$$\begin{pmatrix} 1 & 3 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Transposition of arithmetic circuits

Arithmetic circuits are like diagrams enriched with a *product*. In particular they can be *transposed*:

$$y_1 = x_1 + 3x_2$$
$$y_2 = x_3$$



$$\begin{pmatrix} 1 & 3 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

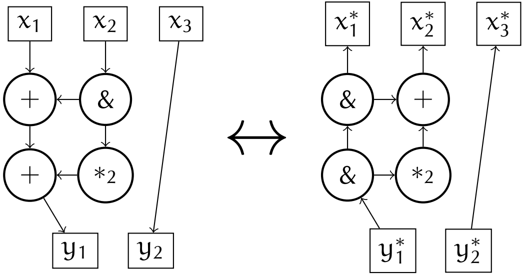
$$\updownarrow$$

$$\begin{pmatrix} 1 & 0 \\ 3 & 0 \\ 0 & 1 \end{pmatrix}$$

Transposition of arithmetic circuits

Arithmetic circuits are like diagrams enriched with a *product*. In particular they can be *transposed*:

$$y_1 = x_1 + 3x_2$$
$$y_2 = x_3$$



$$\begin{pmatrix} 1 & 3 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$
$$\updownarrow$$
$$\begin{pmatrix} 1 & 0 \\ 3 & 0 \\ 0 & 1 \end{pmatrix}$$

This can be made precise using category theory.

Transposition of straight line programs

Straight line programs = Arithmetic circuits

```
a[1] = a[0] + a[1]
a[0] = 0
a[2] = a[1] + a[2]
a[1] = 0
...
a[n-1] = a[n-2] + a[n-1]
a[n-2] = 0
```

$$\begin{pmatrix} 0 & \ldots\ldots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \ldots\ldots & 0 \\ 1 & \ldots\ldots & 1 \end{pmatrix}$$

```
a[n-2] = 0
a[n-2] = a[n-2] + a[n-1]
...
a[1] = 0
a[1] = a[1] + a[2]
a[0] = 0
a[0] = a[0] + a[1]
```

$$\begin{pmatrix} 0 & \ldots & 0 & 1 \\ \vdots & \ldots & \vdots & \vdots \\ 0 & \ldots & 0 & 1 \end{pmatrix}$$

Programs = Families of straight line programs

Automatic transposition?

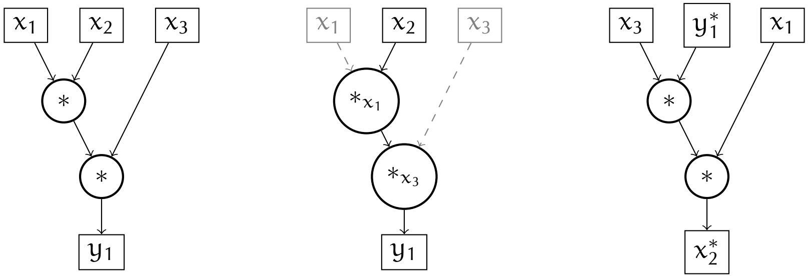
- Algorithms are hard to transpose, transposed algorithms are hard or impossible to understand;
- How to be confident that a transposed algorithm is well implemented if no one understands it?
- When proving programs with a proof assistant, why should we do the work twice?

Previous work

- Originally discovered in *electrical network theory* by Bordewijk 1957 (only works for $\mathbb{C}$); some authors attribute the discovery to Tellegen, Bordewijk's director, but this is debated;
- Fiduccia 1973 and Hopcroft and Musinski 1973: transposition of *bilinear chains*, the most complete formulation (non-commutative rings);
- Special case of *automatic differentiation* Baur and Strassen 1983;
- In *computer algebra*, popularized by Shoup, von zur Gathen, Kaltofen,...
- Bostan, Lecerf, and Schost 2003 improve algorithms for polynomial evaluation and solve an open question on space complexity.

Multilinearity

Does it make sense to transpose $c := a * b$?



- Most applications require to *linearize* a multi-linear program.
- Can we automatically deduce any possible linearisation of a program?
- Type inference systems can help us

Linearity inference

Suppose given a type `R` implementing a ring. We want to define types `L` (for *linear*) and `S` (for *scalar*) such that the following equations hold

```
plus :: L -> L -> L
plus :: S -> S -> S

times :: L -> S -> L
times :: S -> L -> L
times :: S -> S -> S

zero :: L
zero :: S

one :: S
```

Linearity inference

Suppose given a type `R` implementing a ring. We want to define types `L` (for *linear*) and `S` (for *scalar*) such that the following equations hold

```
plus :: L -> L -> L
plus :: S -> S -> S
```

$$\forall\alpha \in \{L, S\}.\alpha \to \alpha \to \alpha$$

```
times :: L -> S -> L
```
~~`times :: S -> L -> L`~~
```
times :: S -> S -> S
```

$$\forall\alpha \in \{L, S\}.\alpha \to S \to \alpha$$

```
zero :: L
zero :: S
```

$$\forall\alpha \in \{L, S\}.\alpha$$

```
one :: S
```

Linearity inference

The solution in Haskell

```haskell
data L = L R
data S = S R

class Ring r where
  zero :: r
  (<+>) :: r -> r -> r
  neg :: r -> r
  (<*>) :: r -> S -> r

one = S oneR
(S a) == (S b) = a == b
```

To treat `times :: S -> L -> L`, we extend the Hindley-Milner type inference to handle lists of acceptable unifications.

transalpyne[1]

We are implementing

A Python-like ad-hoc language, compiled/interpreted in Python, featuring:

- Algebraic constructs (Rings, Modules, Fields, ...);
- Transposition of multilinear/recursive code;
- Parameterizable linearity inference (including commutative multiplication);
- Algebraic complexity preserving;
- Easily used on top of Computer Algebra Systems that have a Python interface;
- Other Computer Algebra Systems will be able to work with it as we will add more languages to the output of the compiler (OCaml and Haskell look easy, C is somewhat harder).

`http://transalpyne.gforge.inria.fr/`

[1] Luca De Feo and Éric Schost (2010). "transalpyne: a language for automatic transposition." In: *SIGSAM Bulletin* 44.1/2 , pp. 59–71. URL: `http://dx.doi.org/10.1145/1838599.1838624`.

Perspectives

Coding

Integration of automatic transposition in a Computer Algebra System. (Sage? Mathemagix?)

Arithmetic circuits and categorical semantics

Joint work with M. Boespflug:

- We have implemented a Domain Specific Language in Haskell,
- the result is not satisfactory due to Haskell's lack of support for dependent types.

Automated Theorem Provers

We plan to write a library to ease the use of the transposition principle in Automated Theorem Provers. (Coq? Agda? Isabelle?)

Plan

1. Transposition principle

2. Artin-Schreier towers

3. Isogenies

Newton sums

Newton identities

- Given a polynomial $f = \prod_j (X - \alpha_j) \in \mathbb{K}[X]$,
- The Newton sums are the $p_i = \sum_j \alpha_j^i$ for any $i \geqslant 0$

$$\frac{f'}{f} = \sum_{i \geqslant 0} \frac{p_i}{T^{i+1}} \qquad \Leftrightarrow \qquad f = \exp\left(\int \frac{f'}{f}\right) = T^d \exp\left(-\sum_{i \geqslant 1} \frac{p_i}{iT^i}\right).$$

Newton sums

Newton identities

- Given a polynomial $f = \prod_j (X - \alpha_j) \in \mathbb{K}[X]$,
- The Newton sums are the $p_i = \sum_j \alpha_j^i$ for any $i \geqslant 0$

$$\frac{f'}{f} = \sum_{i\geqslant 0} \frac{p_i}{T^{i+1}} \qquad \Leftrightarrow \qquad f = \exp\left(\int \frac{f'}{f}\right) = T^d \exp\left(-\sum_{i\geqslant 1} \frac{p_i}{iT^i}\right).$$

Trace formulas

Let $\mathcal{A} = \mathbb{K}[X]/f(X)$, then

$$p_i = \mathrm{Tr}_{\mathcal{A}/\mathbb{K}} X^i.$$

More generally for any $a, z \in \mathcal{A}$, with z primitive and g its minimal polynomial

$$\sum_{i\geqslant 0} \frac{a \cdot \mathrm{Tr}_{\mathcal{A}/K} z^i}{T^{i+1}} = \sum_{i\geqslant 0} \frac{\mathrm{Tr}_{\mathcal{A}/K} az^i}{T^{i+1}} = \frac{A(T)}{g(T)} \qquad \text{and} \qquad a = \frac{A(z)}{g'(z)}.$$

Shoup's algorithm (Shoup 1995, 1999)

Polynomial evaluation: $k[T] \to \mathbb{K}/k$

$$g \mapsto g(\sigma)$$

Power projection: $(\mathbb{K}/k)^* \to k[T]^*$

$$\ell \mapsto \sum_{i>0} \frac{\ell(\sigma^i)}{T^i}$$

Power projection = transposed polynomial evaluation

Let $\mathcal{A} = \mathbb{K}[X]/f(X)$ and $z \in \mathcal{A}$. Take any algorithm that computes $g \mapsto g(z)$ and transpose it:

- Apply to $\mathrm{Tr}_{\mathcal{A}/\mathbb{K}}$ to compute the characteristic polynomial of z;
- Apply to $a \cdot \mathrm{Tr}_{\mathcal{A}/\mathbb{K}}$ to compute a representation of a as a univariate polynomial in z.

The complexity of the original algorithm is preserved by the transposition principle!

Rational Univariate Representation

Generalization in many variables (Giusti, Lecerf, and Salvy 2001; Rouillier 1999)

Let $\mathcal{A} = \mathbb{K}[x_1, \ldots, x_n]/I$ and $z \in \mathcal{A}$

$$g(z) = 0,$$
$$x_1 = \frac{g_1(z)}{g(z)},$$
$$\vdots$$
$$x_n = \frac{g_n(z)}{g(z)},$$

Change of basis

These two operations have the same cost, by the transposition principle:

- Going from the univariate basis

  $$\mathbf{Z} = \{1, z, \ldots, z^{d-1}\}$$

  to any basis $\mathbf{B}$ is equivalent to polynomial evaluation in z.
- Going from $\mathbf{B}$ to $\mathbf{Z}$ is equivalent to Rational Univariate Representation.

Application to towers of extension fields

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

$$\Big| \, p$$

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

$$\vdots$$

$$\mathbb{U}_1 = \frac{\mathbb{U}_0[X_1]}{P_0(X_1)}$$

$$\Big| \, p$$

$$\mathbb{U}_0 = \mathbb{F}_{p^d} = \frac{\mathbb{F}_p[X_0]}{Q(X_0)}$$

Change of basis

$\mathbf{Z} = \{1, X_k, X_k^2, \ldots\}$
$\mathbf{B} = \{1, X_{k-1}, X_{k-1}, \ldots, X_k, X_{k-1}X_k, X_{k-1}^2 X_k, \ldots\}$

$$\begin{cases} Q_k(X_k) = 0 \\ X_{k-1} = \frac{R(X_k)}{Q_k'(X_k)} \end{cases} \leftrightarrow \begin{cases} P_{k-1}(X_k, X_{k-1}) = 0 \\ Q_{k-1}(X_{k-1}) = 0 \end{cases}$$

- Multiplication is faster on **Z**;
- Embeddings are faster on **B**;
- A fast algorithm for $\mathbf{Z} \to \mathbf{B}$ implies a fast one for $\mathbf{B} \to \mathbf{Z}$.

Application to Artin-Schreir towers[2]

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

$$\Big|\, p$$

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

$$\vdots$$

$$\mathbb{U}_1 = \frac{\mathbb{U}_0[X_1]}{P_0(X_1)}$$

$$\Big|\, p$$

$$\mathbb{U}_0 = \mathbb{F}_{p^d} = \frac{\mathbb{F}_p[X_0]}{Q(X_0)}$$

Artin-Schreier extension

$\mathbb{L}/\mathbb{K}$ of characteristic p such that

$$\mathbb{L} = \mathbb{K}[X]/(X^p - X - \alpha).$$

Our construction

Let $x_0 = X_0$ such that $\mathrm{Tr}_{\mathbb{U}_0/\mathbb{F}_p}(x_0) \neq 0$, let

$$\begin{aligned} P_0 &= X^p - X - x_0 \\ P_i &= X^p - X - x_i^{2p-1} \end{aligned}$$

with x_{i+1} a root of P_i in $\mathbb{U}_{i+1}$.
This tower is such that x_i generates $\mathbb{U}_i/\mathbb{F}_p$.

[2] Luca De Feo and Éric Schost (2009). "Fast arithmetics in Artin-Schreier towers over finite fields." In: *ISSAC '09: Proceedings of the 2009 international symposium on Symbolic and algebraic computation* . Seoul, Republic of Korea: ACM, pp. 127–134. URL: `http://dx.doi.org/10.1145/1576702.1576722`.

Application to Artin-Schreir towers

The algorithms

All of these operations can be done in quasi-optimal time and space (w.r.t. the size of $\mathbb{U}_k$):

- Minimal polynomials of x_i over $\mathbb{F}_p$ computed iteratively;
- Change $\mathbf{Z} \to \mathbf{B}$ using a p-ary divide-and-conquer;
- Change $\mathbf{B} \to \mathbf{Z}$ by trace formulas + transposed algorithms;
- Fast univariate multiplication via FFT, fast arithmetics (inversion, GCD, ...);
- Traces and *pseudotraces*, Frobenius morphisms;
- Isomorphisms with arbitrary Artin-Schreier towers via Couveignes 2000.

Implementation

- `C++` with `NTL` implementation released under GPL: `http://www.lix.polytechnique.fr/~defeo/FAAST/`
- Port to SAGE one day?

Plan

1. Transposition principle
2. Artin-Schreier towers
3. Isogenies

Isogenies between elliptic curves

$$\mathcal{I} : E \to E'$$

(Separable) isogeny: (separable) non-constant rational morphism preserving the point at infinity.

Properties

- Finite kernel, surjective (in $\bar{\mathbb{K}}$);
- Defined by rational fractions with a pole at infinity;
- $\#E(\mathbb{F}_{q^n}) = \#E'(\mathbb{F}_{q^n})$ for every n,
- *Dual* isogeny: $[m] = \mathcal{I} \circ \hat{\mathcal{I}}$.

Multiplication

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

$\ker \mathcal{I} = E[m], \quad \deg \mathcal{I} = m^2.$

Isogenies between elliptic curves

$$\mathcal{I} : E \to E'$$

(Separable) isogeny: (separable) non-constant rational morphism preserving the point at infinity.

Properties

- Finite kernel, surjective (in $\bar{\mathbb{K}}$);
- Defined by rational fractions with a pole at infinity;
- $\#E(\mathbb{F}_{q^n}) = \#E'(\mathbb{F}_{q^n})$ for every n,
- *Dual* isogeny: $[m] = \mathcal{I} \circ \hat{\mathcal{I}}$.

Frobenius endomorphism

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

$\ker \varphi = \{\mathcal{O}\}$, $\quad \deg \mathcal{I} = q$.

Isogenies between elliptic curves

$$\mathcal{I} : E \to E'$$

(Separable) isogeny: (separable) non-constant rational morphism preserving the point at infinity.

Properties

- Finite kernel, surjective (in $\bar{\mathbb{K}}$);
- Defined by rational fractions with a pole at infinity;
- $\#E(\mathbb{F}_{q^n}) = \#E'(\mathbb{F}_{q^n})$ for every n,
- *Dual* isogeny: $[m] = \mathcal{I} \circ \hat{\mathcal{I}}$.

Separable isogeny, odd degree (simplified Weierstrass model)

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

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

Vélu formulas

Vélu 1971 (algebraically closed field)

Given the kernel H, computes $\mathcal{I} : E \to E/H$ given by

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

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

For $p \geqslant 3$, given $h(x)$ vanishing on H

$$y^2 = f(x) \qquad t = \sum_{Q \in H^*} f'(Q), \quad u = \sum_{Q \in H^*} 2f(Q), \quad w = u + \sum_{Q \in H^*} x(Q) f'(Q),$$

$$\mathcal{I}(x,y) = \left(\frac{g(x)}{h(x)}, y \left(\frac{g(x)}{h(x)} \right)' \right) \quad \text{avec} \quad \frac{g(x)}{h(x)} = x + t \frac{h'(x)}{h(x)} - u \left(\frac{h'(x)}{h(x)} \right)'$$

Isogeny computation

$$\text{Given } E, E', \ell, \text{ compute } \mathcal{I} : E \to E'$$

By Vélu formulas: $\mathcal{I}(x, y) = \left(\frac{g(x)}{h(x)}, cy\left(\frac{g(x)}{h(x)}\right)'\right)$, hence

$$c^2(x^3 + ax + b)\left(\frac{g(x)}{h(x)}\right)'^2 = \left(\frac{g(x)}{h(x)}\right)^3 + a'\frac{g(x)}{h(x)} + b'$$

BMSS algorithm (Bostan, Morain, Salvy, and Schost 2008)

1. Change variables $S(x) = \sqrt{\frac{h(1/x^2)}{g(1/x^2)}} \quad \Leftrightarrow \quad \frac{g(x)}{h(x)} = \frac{1}{S(1/\sqrt{x})^2}$;
2. Power series solution of $c^2(bx^6 + ax^4 + 1)S'^2 = 1 + a'S^4 + b'S^6$;
3. Inverse the change of variables, reconstruct a rational fraction.

Lercier and Sirvent 2008

When p exceeds the precision, a division by zero happens:

- Lift E and E' in the p-adics while keeping $\Phi_\ell\left(j(\tilde{E}), j(\tilde{E}')\right) = 0$;
- Apply BMSS in $\mathbb{Q}_q$.

Couveignes' algorithms

Idea: Send $E[p^k]$ over $E'[p^k]$

Couveignes 1994

Couveignes 1996

- Compute the extensions $\mathbb{U}_i/\mathbb{F}_q$ such that $E[p^i]$ is defined in $\mathbb{U}_i$;
- Pick k *large enough* ($k \sim \log_p 4\ell$);
- Compute P, a generator of $E[p^k]$;
- Compute P', a generator of $E'[p^k]$;
- Compute the polynomial T vanishing $E[p^k]$;
- Interpolate $A : x(P) \mapsto x(P')$;
- Reconstruct a rational fraction $\frac{g}{h} \equiv A \bmod T$;
- If $\frac{g}{h}$ is an isogeny, done; otherwise pick another P'.

Couveignes' algorithms

Idea: Send $E[p^k]$ over $E'[p^k]$

Couveignes 1994

- Work in the formal group $\mathcal{E}$ of E: a *formal point* is a series in a formal parameter τ;
- Fix a precision *large enough* for $\mathbb{F}_q[[\tau]]$ ($\sim \log_p 4\ell$);
- Compute a morphism $\mathcal{U}(\tau) : \mathcal{E} \to \mathcal{E}'$;
- Reconstruct a rational fraction $\frac{g(X)}{h(X)} = \frac{1}{\mathcal{U}(1/X)}$;
- If $\frac{g}{h}$ is an isogeny, done; otherwise pick another $\mathcal{U}$.

Couveignes 1996

- Compute the extensions $\mathbb{U}_i/\mathbb{F}_q$ such that $E[p^i]$ is defined in $\mathbb{U}_i$;
- Pick k *large enough* ($k \sim \log_p 4\ell$);
- Compute P, a generator of $E[p^k]$;
- Compute P', a generator of $E'[p^k]$;
- Compute the polynomial T vanishing $E[p^k]$;
- Interpolate $A : x(P) \mapsto x(P')$;
- Reconstruct a rational fraction $\frac{g}{h} \equiv A \bmod T$;
- If $\frac{g}{h}$ is an isogeny, done; otherwise pick another P'.

Couveignes' algorithms

Idea: Send $E[p^k]$ over $E'[p^k]$

Couveignes 1994

- Work in the formal group $\mathcal{E}$ of E: a *formal point* is a series in a formal parameter τ;
- Fix a precision *large enough* for $\mathbb{F}_q[[\tau]]$ ($\sim \log_p 4\ell$);
- Compute a morphism $\mathcal{U}(\tau) : \mathcal{E} \to \mathcal{E}'$;
- Reconstruct a rational fraction $\frac{g(X)}{h(X)} = \frac{1}{\mathcal{U}(1/X)}$;
- If $\frac{g}{h}$ is an isogeny, done; otherwise pick another $\mathcal{U}$.
- $\mathcal{U}$ is uniquely determined by it action on $\mathcal{E}[p^k]$ for every k.

Couveignes 1996

- Compute the extensions $\mathbb{U}_i/\mathbb{F}_q$ such that $E[p^i]$ is defined in $\mathbb{U}_i$;
- Pick k *large enough* ($k \sim \log_p 4\ell$);
- Compute P, a generator of $E[p^k]$;
- Compute P', a generator of $E'[p^k]$;
- Compute the polynomial T vanishing $E[p^k]$;
- Interpolate $A : x(P) \mapsto x(P')$;
- Reconstruct a rational fraction $\frac{g}{h} \equiv A \bmod T$;
- If $\frac{g}{h}$ is an isogeny, done; otherwise pick another P'.

Fast Couveignes 1996[3]

- Compute the extensions $\mathbb{U}_i/\mathbb{F}_q$ such that $E[p^i]$ is defined in $\mathbb{U}_i$;
- Pick k *large enough* ($k \sim 4\ell$);
- Compute P, a generator of $E[p^k]$;
- Compute P', a generator of $E'[p^k]$;
- Compute the polynomial T vanishing $E[p^k]$;
- Interpolate $A : x(P) \mapsto x(P')$;
- Reconstruct a rational fraction $\frac{g}{h} \equiv A \bmod T$;
- If $\frac{g}{h}$ is an isogeny, done; otherwise pick another P'.

[3] Luca De Feo (2010). "Fast algorithms for computing isogenies between ordinary elliptic curves in small characteristic." In: *Journal of Number Theory* . URL: `http://dx.doi.org/10.1016/j.jnt.2010.07.003`.

Fast Couveignes 1996[3]

- Compute the extensions $\mathbb{U}_i/\mathbb{F}_q$ such that $E[p^i]$ is defined in $\mathbb{U}_i$;
- Pick k *large enough* ($k \sim 4\ell$);
- Compute P, a generator of $E[p^k]$;
- Compute P', a generator of $E'[p^k]$;
- Compute the polynomial T vanishing $E[p^k]$;
- Interpolate $A : x(P) \mapsto x(P')$;
- Reconstruct a rational fraction $\frac{g}{h} \equiv A \bmod T$;
- If $\frac{g}{h}$ is an isogeny, done; otherwise pick another P'.

An Artin-Schreir tower: $\tilde{O}(\ell)$

[3] Luca De Feo (2010). "Fast algorithms for computing isogenies between ordinary elliptic curves in small characteristic." In: *Journal of Number Theory* . URL: `http://dx.doi.org/10.1016/j.jnt.2010.07.003`.

Fast Couveignes 1996[3]

- Compute the extensions $\mathbb{U}_i/\mathbb{F}_q$ such that $E[p^i]$ is defined in $\mathbb{U}_i$;
- Pick k *large enough* ($k \sim 4\ell$);
- Compute P, a generator of $E[p^k]$;
- Compute P', a generator of $E'[p^k]$;
- Compute the polynomial T vanishing $E[p^k]$;
- Interpolate $A : x(P) \mapsto x(P')$;
- Reconstruct a rational fraction $\frac{g}{h} \equiv A \bmod T$;
- If $\frac{g}{h}$ is an isogeny, done; otherwise pick another P'.

An Artin-Schreir tower: $\tilde{O}(\ell)$

An isomorphism of Artin-Schreier towers: $\tilde{O}(\ell)$

An isomorphism of Artin-Schreier towers: $\tilde{O}(\ell)$

[3] Luca De Feo (2010). "Fast algorithms for computing isogenies between ordinary elliptic curves in small characteristic." In: *Journal of Number Theory* . URL: http://dx.doi.org/10.1016/j.jnt.2010.07.003.

Fast Couveignes 1996[3]

- Compute the extensions $\mathbb{U}_i/\mathbb{F}_q$ such that $E[p^i]$ is defined in $\mathbb{U}_i$;
- Pick k *large enough* ($k \sim 4\ell$);
- Compute P, a generator of $E[p^k]$;
- Compute P', a generator of $E'[p^k]$;
- Compute the polynomial T vanishing $E[p^k]$;
- Interpolate $A : x(P) \mapsto x(P')$;
- Reconstruct a rational fraction $\frac{g}{h} \equiv A \bmod T$;
- If $\frac{g}{h}$ is an isogeny, done; otherwise pick another P'.

An Artin-Schreir tower: $\tilde{O}(\ell)$

An isomorphism of Artin-Schreier towers: $\tilde{O}(\ell)$

An isomorphism of Artin-Schreier towers: $\tilde{O}(\ell)$

Fast interpolation in towers of extensions: $\tilde{O}(\ell)$

[3] Luca De Feo (2010). "Fast algorithms for computing isogenies between ordinary elliptic curves in small characteristic." In: *Journal of Number Theory* . URL: `http://dx.doi.org/10.1016/j.jnt.2010.07.003`.

Fast Couveignes 1996[3]

- Compute the extensions $\mathbb{U}_i/\mathbb{F}_q$ such that $E[p^i]$ is defined in $\mathbb{U}_i$;
- Pick k *large enough* ($k \sim 4\ell$);
- Compute P, a generator of $E[p^k]$;
- Compute P', a generator of $E'[p^k]$;
- Compute the polynomial T vanishing $E[p^k]$;
- Interpolate $A : x(P) \mapsto x(P')$;
- Reconstruct a rational fraction $\frac{g}{h} \equiv A \bmod T$;
- If $\frac{g}{h}$ is an isogeny, done; otherwise pick another P'.

An Artin-Schreir tower: $\tilde{O}(\ell)$

An isomorphism of Artin-Schreier towers: $\tilde{O}(\ell)$

An isomorphism of Artin-Schreier towers: $\tilde{O}(\ell)$

Fast interpolation in towers of extensions: $\tilde{O}(\ell)$

XGCD: $\tilde{O}(\ell)$

[3] Luca De Feo (2010). "Fast algorithms for computing isogenies between ordinary elliptic curves in small characteristic." In: *Journal of Number Theory* . URL: http://dx.doi.org/10.1016/j.jnt.2010.07.003.

Fast Couveignes 1996[3]

- Compute the extensions $\mathbb{U}_i/\mathbb{F}_q$ such that $E[p^i]$ is defined in $\mathbb{U}_i$;
- Pick k *large enough* ($k \sim 4\ell$);
- Compute P, a generator of $E[p^k]$;
- Compute P', a generator of $E'[p^k]$;
- Compute the polynomial T vanishing $E[p^k]$;
- Interpolate $A : x(P) \mapsto x(P')$;
- Reconstruct a rational fraction $\frac{g}{h} \equiv A \bmod T$;
- If $\frac{g}{h}$ is an isogeny, done; otherwise pick another P'.

An Artin-Schreir tower: $\tilde{O}(\ell)$

An isomorphism of Artin-Schreier towers: $\tilde{O}(\ell)$

An isomorphism of Artin-Schreier towers: $\tilde{O}(\ell)$

Fast interpolation in towers of extensions: $\tilde{O}(\ell)$

XGCD: $\tilde{O}(\ell)$

Repeat $O(\ell)$ times

[3] Luca De Feo (2010). "Fast algorithms for computing isogenies between ordinary elliptic curves in small characteristic." In: *Journal of Number Theory* . URL: `http://dx.doi.org/10.1016/j.jnt.2010.07.003`.

How to recognize an isogeny?

- **Degree**: $\frac{g}{h}$ with $\deg g = \ell$, $\deg h = \ell - 1$; $O(1)$
- **Square factor:** $h = \prod_{Q \in H^*}(X - x(Q)) = f^2$ if ℓ odd; $\tilde{O}(\ell)$
- **Group action:** Test with random points; $O(\ell)$
- **Factor of the ℓ-division polynomial:** Compute $\phi_\ell \bmod h$. $\tilde{O}(\ell)$

How to recognize an isogeny?

$$AU_i + TV_i = R_i \qquad \Leftrightarrow \qquad A \equiv \frac{R_i}{U_i} \bmod T$$

$$\ell = 11$$

How to recognize an isogeny?

$$AU_i + TV_i = R_i \qquad \Leftrightarrow \qquad A \equiv \frac{R_i}{U_i} \bmod T$$

$$\ell = 11$$

$\deg R_i$	$\deg U_i$
31415926535897932384626 43	0

How to recognize an isogeny?

$$AU_i + TV_i = R_i \qquad \Leftrightarrow \qquad A \equiv \frac{R_i}{U_i} \bmod T$$

$$\ell = 11$$

$\deg R_i$	$\deg U_i$
31415926535897932384626433	0
31415926535897932384626422	1

How to recognize an isogeny?

$$AU_i + TV_i = R_i \qquad \Leftrightarrow \qquad A \equiv \frac{R_i}{U_i} \bmod T$$

$$\ell = 11$$

$\deg R_i$	$\deg U_i$
3141592653589793238462643	0
3141592653589793238462642	1
3141592653589793238462641	2

How to recognize an isogeny?

$$AU_i + TV_i = R_i \qquad \Leftrightarrow \qquad A \equiv \frac{R_i}{U_i} \bmod T$$

$$\ell = 11$$

$\deg R_i$	$\deg U_i$
3141592653589793238462643	0
3141592653589793238462642	1
3141592653589793238462641	2
$\vdots$	$\vdots$
3141592653589793238462634	9

How to recognize an isogeny?

$$AU_i + TV_i = R_i \qquad \Leftrightarrow \qquad A \equiv \frac{R_i}{U_i} \bmod T$$

$$\ell = 11$$

$\deg R_i$	$\deg U_i$
3141592653589793238462643	0
3141592653589793238462642	1
3141592653589793238462641	2
$\vdots$	$\vdots$
3141592653589793238462634	9

How to recognize an isogeny?

$$AU_i + TV_i = R_i \qquad \Leftrightarrow \qquad A \equiv \frac{R_i}{U_i} \bmod T$$

$$\ell = 11$$

$\deg R_i$	$\deg U_i$
31415926535897932384626 43	0
31415926535897932384626 42	1
31415926535897932384626 41	2
$\vdots$	$\vdots$
31415926535897932384626 34	9
11	10

How to recognize an isogeny?

$$AU_i + TV_i = R_i \qquad \Leftrightarrow \qquad A \equiv \frac{R_i}{U_i} \bmod T$$

$$\ell = 11$$

$\deg R_i$	$\deg U_i$
3141592653589793238462643	0
3141592653589793238462642	1
3141592653589793238462641	2
$\vdots$	$\vdots$
3141592653589793238462634	9
11	**10**
10	3141592653589793238462633
$\vdots$	$\vdots$

Isogenies of unknown degree

- This pattern is extremely rare.
- This is the only phase of Couveignes' algorithms that depends on ℓ.

Isogenies of unknown degree

- This pattern is extremely rare.
- This is the only phase of Couveignes' algorithms that depends on ℓ.
- Actually, this does not really depend on ℓ, just on the existence of a *gap*.
- If ℓ is not known in advance, it is enough to look for a *gap*.
- Thus, any isogeny of degree $\ll p^k$ can be obtained with one single run of Couveignes' algorithms.

Perspectives

Looking for the quasi-linear complexity

- The Weierstrass model has a canonicity defect: use other parameterizations? Formal groups?
- How to obtain *local* information on the behavior of the isogeny? (for example, its action on $E[p]$)

Isogenies of unknown degree

- This variant of Couveignes 1996 is at the moment the fastest (both in theory and in practice) algorithm for this task.
- We tested two curves over $\mathbb{F}_{2^{161}}$, isogenous of unknown degree, taken from Teske 2006;
- Certified in 258 cpu-hours that no isogeny of degree $2^c\ell$ for any c and $\ell < 2^{11}$ exists;
- Certified in 1195 cpu-hours that no isogeny of degree les then 2^{12} exists.
- The two curves have an isogeny of (very smooth) degree $\sim 2^{1050}$. Proving that no isogeny of smaller degree exists is momentarily out of reach.

Z'en voulez plus?

Fast Algorithms for Towers of Finite Fields and Isogenies

13 décembre, École Polytechnique
heure et amphi à préciser

References I

Fiduccia, Charles M. (1973).
"On the algebraic complexity of matrix multiplication."
PhD thesis. Providence, RI, USA: Brown University.
URL: `http://portal.acm.org/citation.cfm?id=906618`.

Couveignes, Jean-Marc (1994).
"Quelques calculs en théorie des nombres."
PhD thesis. Université de Bordeaux.

Schoof, René (1995).
"Counting points on elliptic curves over finite fields."
In: *Journal de Théorie des Nombres de Bordeaux* 7.1 ,
Pp. 219–254.
URL: `http://www.ams.org/mathscinet-getitem?mr=1413578`.

References II

Gallant, Robert P., Robert J. Lambert, and Scott A. Vanstone (2001).
"Faster Point Multiplication on Elliptic Curves with Efficient Endomorphisms."
In: *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology* .
London, UK: Springer-Verlag,
Pp. 190–200.
URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.32.2004.

Gaudry, Pierrick, Florian Hess, and Niegel Smart (2002).
"Constructive and destructive facets of Weil descent on elliptic curves."
In: *Journal of Cryptology* 15.1 ,
Pp. 19–46–46.
URL: http://dx.doi.org/10.1007/s00145-001-0011-x.

References III

Smith, Benjamin (2009).
"Isogenies and the Discrete Logarithm Problem in Jacobians of Genus 3 Hyperelliptic Curves."
In: *Journal of Cryptology* 22.4 ,
Pp. 505–529–529.
URL: http://dx.doi.org/10.1007/s00145-009-9038-1.

Teske, Edlyn (2006).
"An Elliptic Curve Trapdoor System."
In: *Journal of Cryptology* 19.1 ,
Pp. 115–133.
URL: http://dx.doi.org/10.1007/s00145-004-0328-3.

Rostovtsev, Alexander and Anton Stolbunov (2006).
Public-key Cryptosystem Based On Isogenies .
URL: http://eprint.iacr.org/2006/145.

References IV

Charles, Denis, Kristin Lauter, and Eyal Goren (2009).
"Cryptographic Hash Functions from Expander Graphs."
In: *Journal of Cryptology* 22.1 ,
Pp. 93–113.
URL: `http://dx.doi.org/10.1007/s00145-007-9002-x`.

Bordewijk, J. (1957).
"Inter-reciprocity applied to electrical networks."
In: *Applied Scientific Research, Section B* 6.1 ,
Pp. 1–74.
URL: `http://dx.doi.org/10.1007/BF02920362`.

References V

Hopcroft, John E. and Jean Musinski (1973).
"Duality applied to the complexity of matrix multiplications and other bilinear forms."
In: *STOC '73: Proceedings of the fifth annual ACM symposium on Theory of computing* .
Austin, Texas, United States: ACM,
Pp. 73–87.
URL: http://dx.doi.org/10.1145/800125.804038.

Baur, Walter and Volker Strassen (1983).
"The complexity of partial derivatives."
In: *Theoretical Computer Science* 22.3 ,
Pp. 317–330.
URL: http://dx.doi.org/10.1016/0304-3975(83)90110-X.

References VI

Bostan, Alin, Grégoire Lecerf, and Éric Schost (2003).
"Tellegen's principle into practice."
In: *ISSAC '03: Proceedings of the 2003 international symposium on Symbolic and algebraic computation* .
Philadelphia, PA, USA: ACM,
Pp. 37–44.
URL: http://dx.doi.org/10.1145/860854.860870.

De Feo, Luca and Éric Schost (2010).
"transalpyne: a language for automatic transposition."
In: *SIGSAM Bulletin* 44.1/2 ,
Pp. 59–71.
URL: http://dx.doi.org/10.1145/1838599.1838624.

Shoup, Victor (1995).
"A new polynomial factorization algorithm and its implementation."
In: *Journal of Symbolic Computation* 20.4 ,
Pp. 363–397.
URL: http://dx.doi.org/10.1006/jsco.1995.1055.

References VII

Shoup, Victor (1999).
"Efficient computation of minimal polynomials in algebraic extensions of finite fields."
In: *ISSAC '99: Proceedings of the 1999 international symposium on Symbolic and algebraic computation* .
Vancouver, British Columbia, Canada: ACM,
Pp. 53–58.
URL: `http://dx.doi.org/10.1145/309831.309859`.

Giusti, Marc, Grégoire Lecerf, and Bruno Salvy (2001).
"A Gröbner free alternative for polynomial system solving."
In: *Journal of Complexity* 17.1 ,
Pp. 154–211.
URL: `http://dx.doi.org/10.1006/jcom.2000.0571`.

References VIII

Rouillier, Fabrice (1999).
"Solving Zero-Dimensional Systems Through the Rational Univariate Representation."
In: *Applicable Algebra in Engineering, Communication and Computing* 9.5
,
Pp. 433–461.
URL: `http://dx.doi.org/10.1007/s002000050114`.

De Feo, Luca and Éric Schost (2009).
"Fast arithmetics in Artin-Schreier towers over finite fields."
In: *ISSAC '09: Proceedings of the 2009 international symposium on Symbolic and algebraic computation* .
Seoul, Republic of Korea: ACM,
Pp. 127–134.
URL: `http://dx.doi.org/10.1145/1576702.1576722`.

References IX

Couveignes, Jean-Marc (2000).
"Isomorphisms between Artin-Schreier towers."
In: *Mathematics of Computation* 69.232 ,
Pp. 1625–1631.
URL: `http://dx.doi.org/10.1090/S0025-5718-00-01193-5`.

Vélu, Jean (1971).
"Isogénies entre courbes elliptiques."
In: *Comptes Rendus de l'Académie des Sciences de Paris* 273 ,
Pp. 238–241.

Bostan, Alin, François Morain, Bruno Salvy, and Éric Schost (2008).
"Fast algorithms for computing isogenies between elliptic curves."
In: *Mathematics of Computation* 77 ,
Pp. 1755–1778.
URL: `http://dx.doi.org/10.1090/S0025-5718-08-02066-8`.

References X

Lercier, Reynald and Thomas Sirvent (2008).
"On Elkies subgroups of ℓ-torsion points in elliptic curves defined over a finite field."
In: *Journal de théorie des nombres de Bordeaux* 20.3 ,
Pp. 783–797.
URL:
http://perso.univ-rennes1.fr/reynald.lercier/file/LS08.pdf.

Couveignes, Jean-Marc (1996).
"Computing l-Isogenies Using the p-Torsion."
In: *ANTS-II: Proceedings of the Second International Symposium on Algorithmic Number Theory* .
London, UK: Springer-Verlag,
Pp. 59–65.
URL: http://portal.acm.org/citation.cfm?id=749581.

References XI

De Feo, Luca (2010).
"Fast algorithms for computing isogenies between ordinary elliptic curves in small characteristic."
In: *Journal of Number Theory* .
URL: http://dx.doi.org/10.1016/j.jnt.2010.07.003.