

# Isogeny computation in small characteristics

L. De Feo

INRIA Projet TANC, LIX École Polytechnique, France

Elliptic Curve Cryptography

August 25, 2009

University of Calgary

# Compute isogenies

## What?

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

- Finite kernel, onto, given by rational fractions.

## Why?

- Point counting.
- Prove hardness of discrete logarithm.
- Move discrete logarithms to easier curves.
- Speed up point multiplication.
- Hide weak curves behind chains of isogenies.
- Define hash functions.

## Multiplication

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

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

# Compute isogenies

## What?

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

- Finite kernel, onto, given by rational fractions.

## Why?

- Point counting.
- Prove hardness of discrete logarithm.
- Move discrete logarithms to easier curves.
- Speed up point multiplication.
- Hide weak curves behind chains of isogenies.
- Define hash functions.

## Frobenius endomorphism

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

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

# Compute isogenies

## What?

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

- Finite kernel, onto, given by rational fractions.

## Why?

- Point counting.
- Prove hardness of discrete logarithm.
- Move discrete logarithms to easier curves.
- Speed up point multiplication.
- Hide weak curves behind chains of isogenies.
- Define hash functions.

## Separable isogenies, odd degree (simplified Weierstrass model)

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

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

# Compute isogenies

## What?

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

- Finite kernel, onto, given by rational fractions.

## Why?

- Point counting.
- Prove hardness of discrete logarithm.
- Move discrete logarithms to easier curves.
- Speed up point multiplication.
- Hide weak curves behind chains of isogenies.
- Define hash functions.

## Normalised (or strict) isogenies

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

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

# Compute isogenies

## What?

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

- Finite kernel, onto, given by rational fractions.

## Why?

- Point counting.
- Prove hardness of discrete logarithm.
- Move discrete logarithms to easier curves.
- Speed up point multiplication.
- Hide weak curves behind chains of isogenies.
- Define hash functions.

## Normalised (or strict) isogenies

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

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

# 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

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

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

$E' = E/H$  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  $\ell$ , integer coefficients of  $\tilde{O}(\ell)$  bits.
- Computed in  $\tilde{O}(\ell^3)$  bit operations (quasi-optimal).

## Which problem?

- 1 Given  $E$ , find an  $\ell$ -isogenous curve and an  $\ell$ -isogeny.
  - 2 Given  $E$  and  $E'$ , find an  $\ell$ -isogeny.
- 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.
  - We'll focus on 2.



# 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\*

$\tilde{O}(\ell)$

# Using $p$ -adics: Lercier-Sirvent

## Weierstrass functions

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

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

## Lercier, Sirvent 2009

Work in the  $p$ -adics to avoid divisions by zero.

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

Works for any  $p$ , complexity  $\tilde{O}(\ell^3)$ , but solves problem 1 directly.

# 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

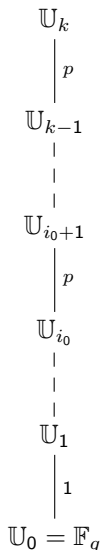
## Other algorithms

'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



## Computing the $p^i$ -torsion

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

## 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

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

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

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

$\vdots$

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

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

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

## 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...



# Solving Artin-Schreier equations in Artin-Schreier towers

$$\begin{array}{ccccc} \mathbb{U}_k & \longrightarrow & \mathbb{L}_k & \longleftarrow & \mathbb{U}'_k \\ | & & | & & | \\ \mathbb{U}_{k-1} & \longrightarrow & \mathbb{L}_{k-1} & \longleftarrow & \mathbb{U}'_{k-1} \\ \vdots & & \vdots & & \vdots \\ \mathbb{U}_1 & \longrightarrow & \mathbb{L}_1 & \longleftarrow & \mathbb{U}'_1 \\ & \searrow & | & \swarrow & \\ & & \mathbb{U}_0 = \mathbb{L}_0 = \mathbb{U}'_0 & & \end{array}$$

# 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 '09])

- 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 '09])

- 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

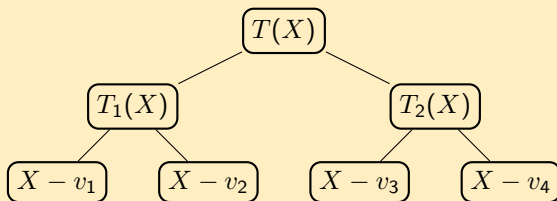
Interpolation of  $v_i \mapsto s_i$  is defined modulo  $T$ , where

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

## 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

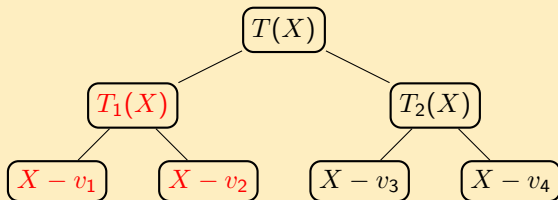
Interpolation of  $v_i \mapsto s_i$  is defined modulo  $T$ , where

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

## 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

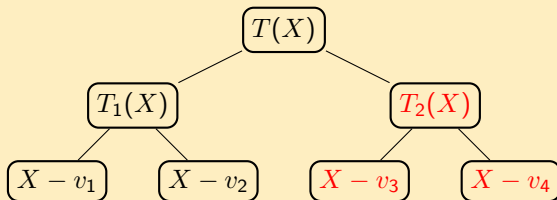
Interpolation of  $v_i \mapsto s_i$  is defined modulo  $T$ , where

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

## 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

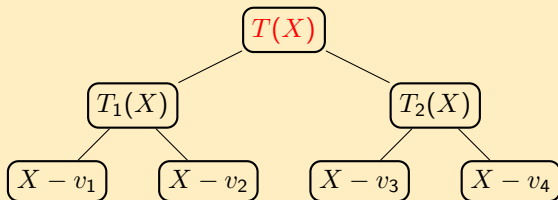
Interpolation of  $v_i \mapsto s_i$  is defined modulo  $T$ , where

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

## 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

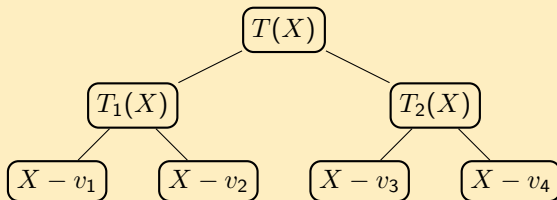
Interpolation of  $v_i \mapsto s_i$  is defined modulo  $T$ , where

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

## Lagrange formula

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

## Interpolation by chinese remaindering





# Faster interpolation using effective Galois groups

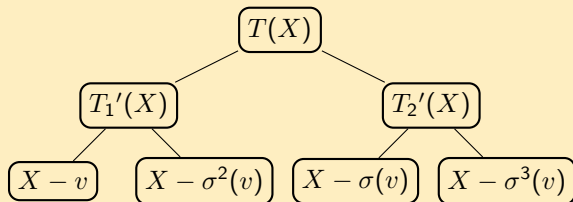
Interpolation of  $v_i \mapsto s_i$  is defined modulo  $T$ , where

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

## 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

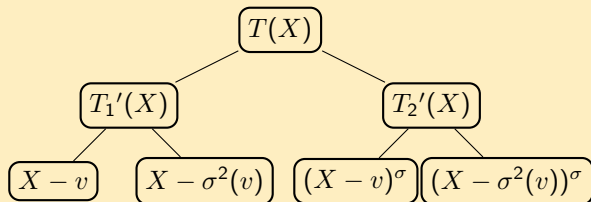
Interpolation of  $v_i \mapsto s_i$  is defined modulo  $T$ , where

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

## Lagrange formula

For a polynomial  $P$  note  $P^\sigma$  the action on the coefficients of  $P$ , then

## Interpolation by chinese remaindering



# Faster interpolation using effective Galois groups

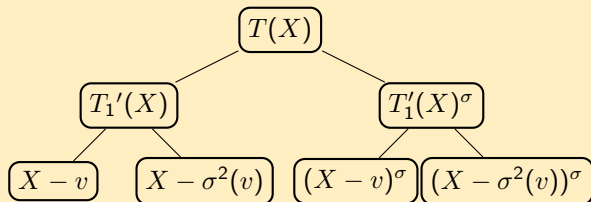
Interpolation of  $v_i \mapsto s_i$  is defined modulo  $T$ , where

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

## Lagrange formula

For a polynomial  $P$  note  $P^\sigma$  the action on the coefficients of  $P$ , then

## Interpolation by chinese remaindering



# Faster interpolation using effective Galois groups

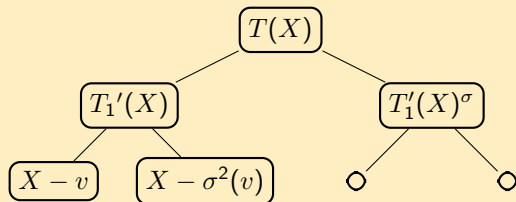
Interpolation of  $v_i \mapsto s_i$  is defined modulo  $T$ , where

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

## Lagrange formula

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

## Interpolation by chinese remaindering



# 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  $A_Q(\varphi_q([j]P)) = [j][\lambda]Q$ .

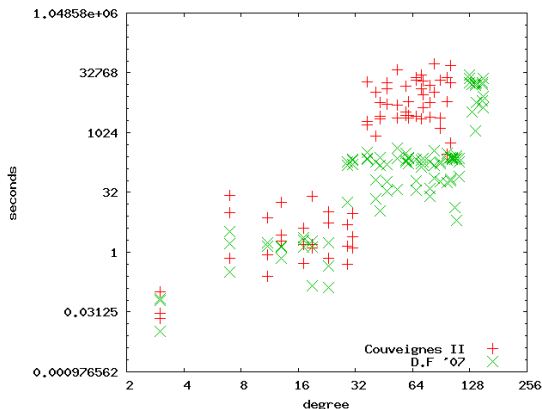
So  $A_Q \circ \varphi_q = A_{[\lambda]Q} \bmod T$ . Solving this is *modular composition*.

## Modular composition

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

# (Old) Timings

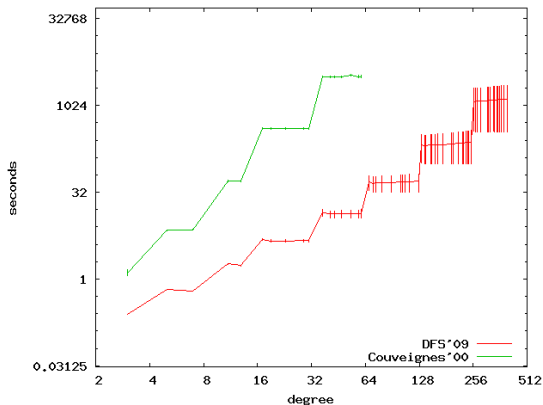
- Magma implementation,
- [Couveignes '96] vs. Fast Interpolation ([D.F. '07]),
- $\mathbb{F}_q = \mathbb{F}_{5^3}$ .



	[Cou'96]	[D.F. '07]
Total time	65951	19864
Compute $E[p^k]$	0,06%	0,5%
Compute $E'[p^k]$	28%	89,5%
Interpolation	71%	9,5%

# Timings

- NTL implementation of [D.F., Schost '09] vs. Magma implementation, of [Couveignes '00]
- $\mathbb{F}_q = \mathbb{F}_{2^{101}}$ .

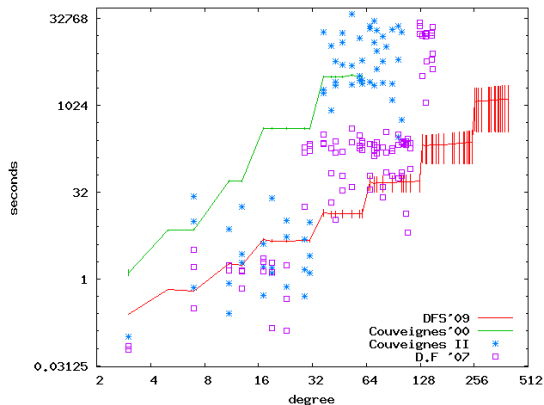


$\ell$	$E[p^k]$	$E'[p^k]$	Interp	Step 6	ModComp	Avg tries	Avg loop time
31	1.3128	1.3128	1.1058	0.00218	0.00218	64	0.279
61	3.5454	3.5464	2.5236	0.00783	0.00900	128	2.154
127	9.2975	9.3026	5.6881	0.03147	0.03634	256	17.359
251	23.7984	23.7984	12.7251	0.12415	0.14519	512	137.902
397	59.7439	59.7579	28.3387	0.36822	0.58027	1024	971.254



# Timings

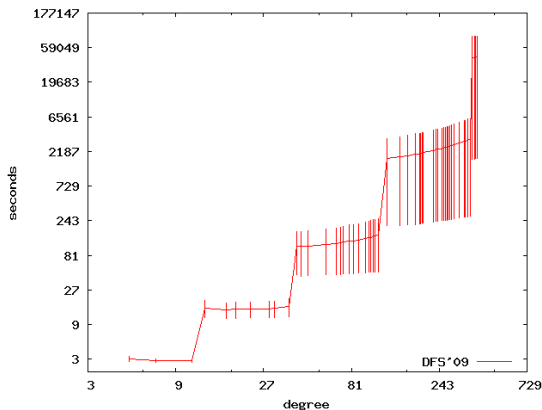
- NTL implementation of [D.F., Schost '09] vs. Magma implementation, of [Couveignes '00]
- $\mathbb{F}_q = \mathbb{F}_{2^{101}}$ .



$\ell$	$E[p^k]$	$E'[p^k]$	Interp	Step 6	ModComp	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

# Timings

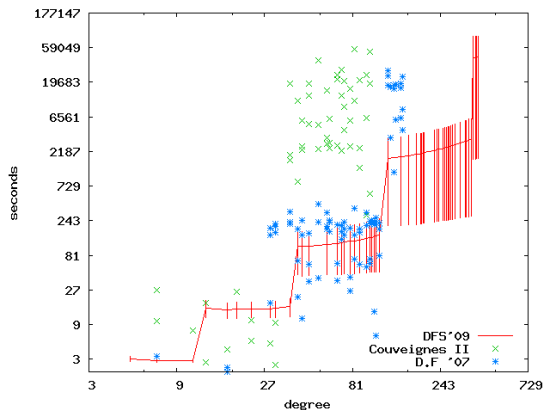
- NTL implementation of [D.F., Schost '09]
- $\mathbb{F}_q = \mathbb{F}_{3^{64}}$ .



$\ell$	$E[p^k]$	$E'[p^k]$	Interp	Step 6	ModComp	Avg tries	Avg loop time
11	0.6109	0.6109	0.4669	0.0194	0.0249	13	0.58
37	2.3946	2.3916	2.1066	0.1988	0.1381	40	13.48
113	9.8045	9.8055	8.5377	1.7712	0.8690	121	319.47
359	38.3292	38.3972	34.7147	17.5004	7.0088	364	8921.35
389	159.8280	159.5690	147.741	45.1558	69.9133	1093	125770.52

# Timings

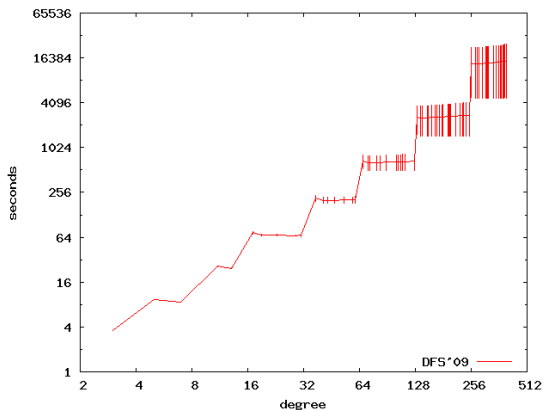
- NTL implementation of [D.F., Schost '09]
- $\mathbb{F}_q = \mathbb{F}_{3^{64}}$ .



$\ell$	$E[p^k]$	$E'[p^k]$	Interp	Step 6	ModComp	Avg tries	Avg loop time
11	0.6109	0.6109	0.4669	0.0194	0.0249	13	0.58
37	2.3946	2.3916	2.1066	0.1988	0.1381	40	13.48
113	9.8045	9.8055	8.5377	1.7712	0.8690	121	319.47
359	38.3292	38.3972	34.7147	17.5004	7.0088	364	8921.35
389	159.8280	159.5690	147.741	45.1558	69.9133	1093	125770.52

# Record Timings!

- NTL implementation of [D.F., Schost '09]
- $\mathbb{F}_q = \mathbb{F}_{2^{1023}}$ .



$\ell$	$E[p^k]$	$E'[p^k]$	Interp	Step 6	ModComp	Avg tries	Avg loop time
31	21.182	21.174	11.597	0.0178	0.02541	64	2.768
61	58.656	58.665	26.826	0.0645	0.10398	128	21.576
127	154.357	154.296	61.202	0.2580	0.41578	256	172.503
251	383.773	383.861	138.428	0.9950	1.66120	512	1360.000
397	931.022	931.610	313.609	3.1819	6.73608	1024	10156.011

# 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  $\ell$ -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 & É. Schost.  
Fast arithmetics in Artin-Schreier towers over finite fields.  
*ISSAC '09*, 2009.
-  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  $\ell$ -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.