
Construction of secure random genus 2 cryptosystems over prime fields

Pierrick Gaudry and Éric Schost

`gaudry@lix.polytechnique.fr`

École polytechnique
France



- Context and previous work
- Generalities on genus 2 curves
- Schoof-like algorithm
- Improvements using symmetric polynomials
- Other algorithms
- Implementation and numerical examples

Cryptographical context

- Cryptosystems based on elliptic curves are now recognized as a reasonable alternative to RSA and to systems based on the classical discrete log.
- **Genus 2 curves** have essentially the same advantages (small key size, small signatures, ...)
- Group law is more complicated as for elliptic curves.
- Recent improvements demonstrated that runtimes are comparables to those obtained with elliptic curves.

The **main difficulty** is now the point-counting problem.

Besides exponential algorithms (Shanks), two classes of algorithms:

• **Schoof-like algorithms:**

- Pila (1990): theoretical algorithm using the Jacobian described as a projective abelian variety. Polynomial time.
- Kampkötter (1991): polynomial time algorithm, specific to genus 2. Gives some division polynomials.
- Adleman-Huang (1996) and Huang-Ierardi (1998): improved on Pila's work (general curves).
- G.-Harley (2000): Genus 2, use Cantor's division polynomials (1994). Reach $\#Jac \approx 10^{38}$.

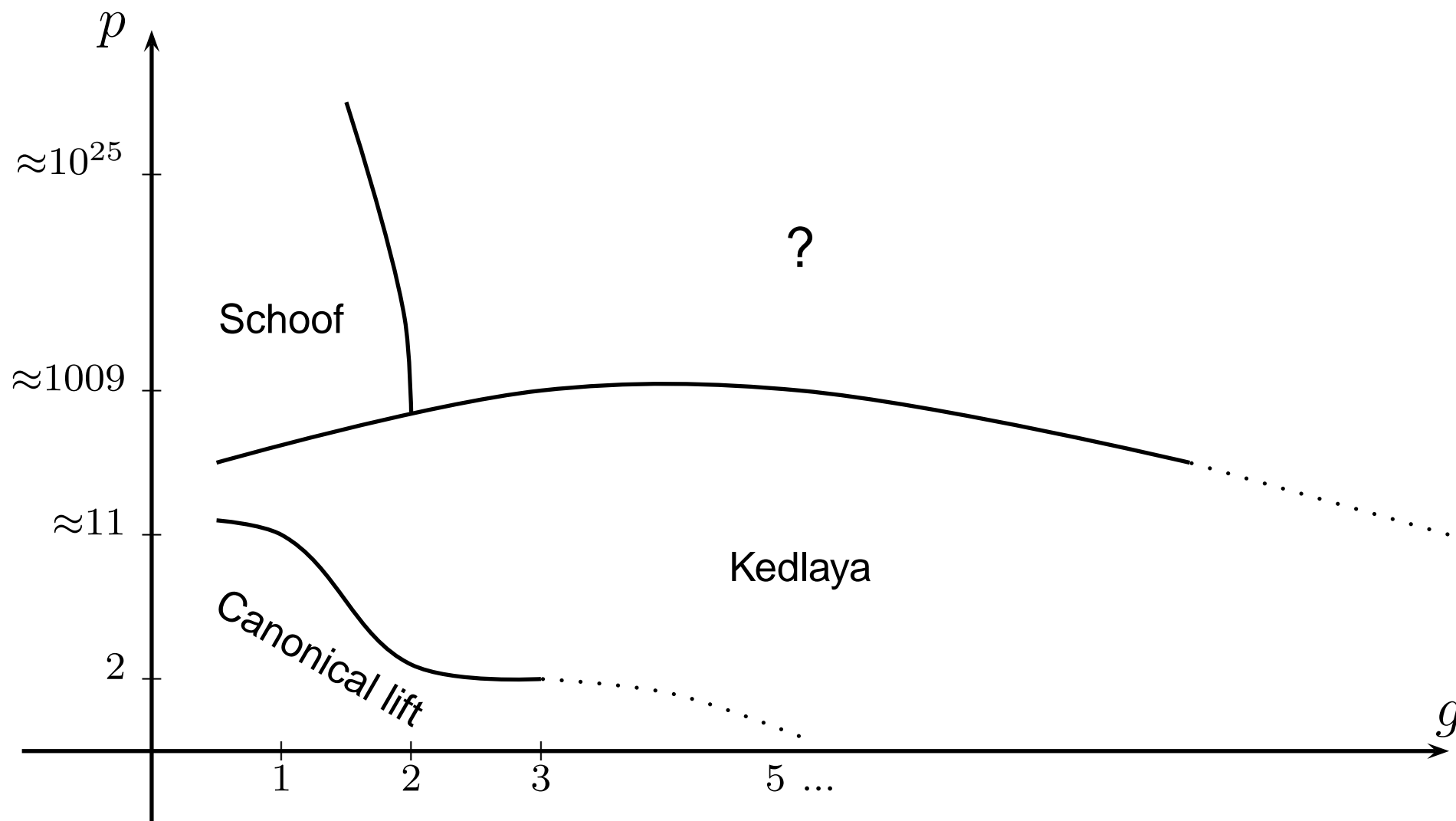
- p -adic algorithms :
 - Satoh (1999): point-counting algorithm for elliptic curves in small characteristic using the canonical p -adic lifting.
 - Many theoretical and practical improvements culminating with Harley's algorithm.

- p -adic algorithms :
 - Satoh (1999): point-counting algorithm for elliptic curves in small characteristic using the canonical p -adic lifting.
 - Many theoretical and practical improvements culminating with Harley's algorithm.
 - Mestre (2000): showed that Richelot isogeny leads to an extension of Satoh's algorithm to genus 2. (Implemented by G.-Harley)

● p -adic algorithms :

- Satoh (1999): point-counting algorithm for elliptic curves in small characteristic using the canonical p -adic lifting.
- Many theoretical and practical improvements culminating with Harley's algorithm.
- Mestre (2000): showed that Richelot isogeny leads to an extension of Satoh's algorithm to genus 2. (Implemented by G.-Harley)
- Kedlaya (2001): proposed another p -adic method based on Monsky-Washnitzer cohomology. Applicable for hyperelliptic curves.
- Lauder-Wan (2001): yet another p -adic algorithm. Polynomial time, for any curves.

Point counting algorithms



Summary: present status for genus 2

- In characteristic **2**, almost as fast as for elliptic curves.
- In **small** characteristic: crypto size is easily feasible.
- In **medium** characteristic: Cartier-Manin's trick helps. Crypto size obtained by Matsuo-Chao-Tsujii (2002).
- For **prime** fields: (this talk) crypto size is reachable in 1 week of computation.

Hyperelliptic curves of genus 2

We consider:

- \mathbb{F}_p : Prime finite field with p elements, $p \neq 2$.
- \mathcal{C} : a curve of genus 2 with equation $y^2 = f(x)$, with $f \in \mathbb{F}_p[x]$ monic squarefree polynomial, $\deg f = 5$.

Group law is defined over the **Jacobian** $\text{Jac}(\mathcal{C})$ of \mathcal{C} :

$$\text{Jac}(\mathcal{C}) = \text{Div}^0(\mathcal{C})/\text{Pr}(\mathcal{C})$$

Consequence of the Riemann-Roch theorem:

In genus 2, every element of $\text{Jac}(\mathcal{C})$ can be uniquely represented by either:

- The null divisor: $D = 0$;
- A divisor of weight 1: $D = P - \infty$, where P is an affine point of \mathcal{C} ;
- A divisor of weight 2: $D = P_1 + P_2 - 2\infty$, where P_1 and P_2 are two non-symmetric affine points.

Mumford's representation

Mumford's representation: every element of $\text{Jac}(\mathcal{C})$ can be uniquely represented by

$$D = \langle u(x), v(x) \rangle, \quad \text{with } u \mid v^2 - f \quad \text{and } \deg v < \deg u \leq 2$$

If $D = \sum P_i - r\infty$, with $P_i = (x_i, y_i)$, then $u(x) = \prod (x - x_i)$ and $v(x_i) = y_i$.

Rem: D is defined over \mathbb{F}_p iff $u(x)$ and $v(x)$ are.

The **generic case** is weight 2 divisor:

$D = \langle x^2 + u_1x + u_0, v_1x + v_0 \rangle$. Representation of elements of a variety of dimension 2 with 4 coordinates.

Group law: Computed with **Cantor's** algorithm; works with Mumford's representation.

Frobenius endomorphism and cardinalities

Let π be the Frobenius automorphism $x \mapsto x^p$, extended to an endomorphism of $\text{Jac}(\mathcal{C})/\overline{\mathbb{F}}_p$.

Thm:(Weil) As an element of $\text{End}(\text{Jac}(\mathcal{C}))$, it has a characteristic polynomial with integer coefficients of the form

$$\chi(t) = t^4 - s_1 t^3 + s_2 t^2 - p s_1 t + p^2,$$

with all the roots of absolute value \sqrt{p} .

Consequence: Bounds: $|s_1| \leq 4\sqrt{p}$ and $|s_2| \leq 6p$.

We have $\#\text{Jac}(\mathcal{C}) = \chi(1) = p^2 + 1 - s_1(p + 1) + s_2$.

Our goal: Compute s_1 and s_2 (equivalent to $\chi(1)$).

- Compute $\chi(t)$ modulo some power of 2;
- Compute $\chi(t)$ modulo some power of 3;
- Compute $\chi(t)$ modulo some small primes $\ell = 5, 7, 11, \dots$
- Deduce $\chi(t)$ using Matsuo-Chao-Tsujii (MCT)'s variant of baby-steps giant-steps.

Thm: If $(\ell, p) = 1$, then the set of ℓ -torsion elements verifies:

$$\text{Jac}(\mathcal{C})[\ell] \cong (\mathbb{Z}/\ell\mathbb{Z})^4 \cong \mathbb{F}_\ell\text{-vector space of dim 4}$$

and π restricted to $\text{Jac}[\ell]$ has characteristic polynomial:
 $\chi(t) \pmod{\ell}$.

Algorithm:

- “Compute” $\text{Jac}[\ell]$,
- Find the action of π on it,
- Deduce $\chi(t) \pmod{\ell}$.

Elliptic case: $E[\ell]$ characterized by the **division polynomial** ψ_ℓ :

$$P = (x, y) \in E[\ell] \Leftrightarrow \psi_\ell(x) = 0.$$

Genus 2: Multivariate analogue:

$$D = \langle x^2 + u_1x + u_0, v_1x + v_0 \rangle \in \text{Jac}[\ell]$$

$$\iff f(u_0, u_1, v_0, v_1) = 0, \forall f \in I_\ell,$$

where I_ℓ is called the **ℓ -division ideal**.

Def: I_ℓ is the ideal of the variety of the weight 2 ℓ -torsion divisors i.e. not of the form $\langle x - x_P, y_P \rangle$.

Rem: I_ℓ is a radical, 0-dimensional ideal in 4 variables.

Goal: Find good generators for I_ℓ that allow us to compute easily ℓ -torsion divisors.

First idea: expand formally $[\ell]D$ with Cantor's algorithm, and compute a Gröbner basis of the resulting system.

Problem: the expansion leads to huge multivariate polynomials, where all the variables are mixed.

Cantor's division polynomials

Let (x_P, y_P) be a point on \mathcal{C} and consider $P = \langle x - x_P, y_P \rangle \in \text{Jac}(\mathcal{C})$, then

$$[\ell]P = \left\langle x^2 + \frac{d_1(x_P)}{d_0(x_P)}x + \frac{d_2(x_P)}{d_0(x_P)}, y_P \left(\frac{e_1(x_P)}{e_0(x_P)}x + \frac{e_2(x_P)}{e_0(x_P)} \right) \right\rangle,$$

where $d_0, d_1, d_2, e_0, e_1, e_2$ are polynomials, degree $O(\ell^2)$, computed by rec. formulae.

Cantor's division polynomials

Let (x_P, y_P) be a point on \mathcal{C} and consider $P = \langle x - x_P, y_P \rangle \in \text{Jac}(\mathcal{C})$, then

$$[\ell]P = \left\langle x^2 + \frac{d_1(x_P)}{d_0(x_P)}x + \frac{d_2(x_P)}{d_0(x_P)}, y_P \left(\frac{e_1(x_P)}{e_0(x_P)}x + \frac{e_2(x_P)}{e_0(x_P)} \right) \right\rangle,$$

where $d_0, d_1, d_2, e_0, e_1, e_2$ are polynomials, degree $O(\ell^2)$, computed by rec. formulae.

Change of variables: represent $D = \langle x^2 + u_1x + u_0, v_1x + v_0 \rangle$ by $D = P_1 + P_2$ with $P_1 = \langle x - x_1, y_1 \rangle$ and $P_2 = \langle x - x_2, y_2 \rangle$.

$[\ell]D = 0 \Leftrightarrow [\ell]P_1 = -[\ell]P_2 \Leftrightarrow$ system of polynomial equations

Resolution of the system

$$\begin{cases} E_1(x_1, x_2) & = & d_1(x_1)d_2(x_2) - d_1(x_2)d_2(x_1) & = & 0, \\ E_2(x_1, x_2) & = & d_0(x_1)d_2(x_2) - d_0(x_2)d_2(x_1) & = & 0, \\ F_1(x_1, x_2, y_1, y_2) & = & y_1e_1(x_1)e_0(x_2) + y_2e_1(x_2)e_0(x_1) & = & 0, \\ F_2(x_1, x_2, y_1, y_2) & = & y_1e_2(x_1)e_0(x_2) + y_2e_2(x_2)e_0(x_1) & = & 0, \end{cases}$$

Advantage: Two equations involve only x_1 and x_2

E_1 and E_2 are divisible by $x_1 - x_2$, therefore we divide them by this factor.

\implies Eliminate x_2 by a resultant computation between $E_1(x_1, x_2)/(x_1 - x_2)$ and $E_2(x_1, x_2)/(x_1 - x_2)$ and get $R(x_1) = 0$ of degree $O(\ell^4)$.

Computation of the resultant

- There is a factor $d_2(x_1)^{2\ell^2-3}$ that appears in $R(x_1)$;
- Evaluate the polynomials for several values of x_1 ; then compute the corresponding univariate resultants. Finally $R(x_1)$ can be interpolated.
- The predicted factor can be taken into account.
- Final degree: $4\ell^4 - 10\ell^2 + 6$.

Deducing an ℓ -torsion divisor

Two strategies:

- **Work modulo the resultant $R(x_1)$.**
Asymptotically, the fastest.
Problem: $R(x_1)$ still contains some parasite factors.
Purging them costs a lot.
Reconstruct the ℓ -torsion ideal as a triangular set.
Apply π to this “element”.
- **Find factors of small degree of $R(x_1)$.**
In practice, this is still the fastest.
In factorization algorithms, the factors of small degrees are found at first: use readily them to produce an ℓ -torsion divisor defined over a small extension.

Compute $\chi(t) \pmod{\ell}$

Given an ℓ -torsion divisor, “plug it” into the characteristic polynomial of Frobenius:

For all pairs $(s_1, s_2) \pmod{\ell}$, compute

$$\pi^4(D) - s_1\pi^3(D) + s_2\pi^2(D) - s_1q\pi(D) + q^2D = 0,$$

and keep only pairs for which this is 0.

NB: use a baby-step, giant-step strategy.

If several pairs remain, build another ℓ -torsion divisor and iterate.

Rem: By this strategy we only compute the minimal polynomial of π . Enough to conclude...

Improvement: re-symmetrisation

To each root x_1 of R corresponds another root x_2 such that the u -polynomial of the corresponding torsion divisor is $(x - x_1)(x - x_2)$. Hence each torsion divisor contributes by two roots in R .

Only the **symmetric functions** of x_1 and x_2 are important.

Instead of computing the minimal polynomial R of x_1 , we are going to compute the minimal polynomial of $u_1 = -(x_1 + x_2)$.

Goal: Divide by two the degree of R .

Problem: How to take advantage of the **compact expression** of E_1 and E_2 .

SLP and symmetric functions

(Work in progress with F. Hivert, É. Schost and N. Thiéry)

Let k a field, and A a rational fraction with coefficients in k , given by an evaluation program.

Let $K = k(s, p)$ be the field of fractions in 2 variables over k . The polynomial $X^2 - sX + p$ is irreducible in $K[X]$, and let \mathfrak{K} be the extension

$$\mathfrak{K} = K[X]/(X^2 - sX + p).$$

The **evaluation** of A at X , the primitive element of \mathfrak{K} can be written:

$$A(X) = A_0(s, p) + A_1(s, p)X,$$

where A_0 and A_1 are rational fractions in s and p .

Let x_1 and x_2 the roots of $X^2 - sX + p$ in an algebraic closure of K . Then

$$A_0(s, p) = \frac{A(x_1) - A(x_2)}{x_1 - x_2}$$

$$A_1(s, p) = \frac{x_1 A(x_2) - x_2 A(x_1)}{x_1 - x_2}$$

Hence the SLP for A is converted in a SLP for computing $\frac{A(x_1) - A(x_2)}{x_1 - x_2}$ and $\frac{x_1 A(x_2) - x_2 A(x_1)}{x_1 - x_2}$ in terms of $s = x_1 + x_2$ and $p = x_1 x_2$.

The number of operations is multiplied by a constant.

Application to the torsion of genus 2 curves

The polynomials

$$E_1(x_1, x_2) = \frac{d_1(x_1)d_2(x_2) - d_1(x_2)d_2(x_1)}{x_1 - x_2}$$

and

$$E_2(x_1, x_2) = \frac{d_0(x_1)d_2(x_2) - d_0(x_2)d_2(x_1)}{x_1 - x_2}$$

can be computed in terms of $s = -u_1$ and $p = u_0$ from the formulae of d_0, d_1, d_2 .

Then we can eliminate u_0 by a **resultant** computation.

In practice, again we use a **evaluation/interpolation** strategy, so that we manipulate only univariate polynomials.

The degree of the result is $6\ell^4 - 17\ell^2 + 12$.

Prediction of parasites

Again, we took only two equations into account \implies many roots are **parasites** that do not give a torsion element.

Some of them are predictable: we have a factor

$$\prod_{d_2(x_1)=0} \prod_{d_2(x_2)=0} (X - (x_1 + x_2))$$

This polynomial can be computed by a bivariate resultant computation. There is a faster (recent) algorithm by **Bostan-Flajolet-Salvy-Schost**.

\implies A parasite factor of degree $4\ell^4 - 12\ell^2 + 9$ is eliminated.

The final degree is $2\ell^4 - 5\ell^2 + 3$.

Ex: for $\ell = 19$, degree 258 840 instead of 517 680.

Using the predictable factorization pattern:

The resultant that has to be factored is more or less a **division polynomial**.

As such, its factorization pattern is dictated by the underlying Galois action.

⇒ this can be used to speed-up Shoup's factoring algorithm.

Division by 2

- Computation of a `lex` Gröbner basis corresponding to halving a generic divisor.
 - dimension 0, degree 16, over $\mathbb{F}_p(u_0, u_1, v_0, v_1)$,
 - computed with a Newton method by Schost.
- Factorization of the separating polynomial using the Galois action coming from the group law \implies costs 4 square roots.

Division by 3

- computation of the `lex` Gröbner basis using Magma's implementation of Buchberger's algorithm;
- Naive factorisation

Algorithm of Matsuo-Chao-Tsujii

How to finish the computation when we know $\chi(t) \bmod m$ and bounds on the coefficients?

- Kind of baby steps - giant steps algorithm, adapted to genus 2 case;
- The gain is by a factor of m (instead of \sqrt{m} with the classical algorithm);
- We were able to modify this algorithm using analogues of Pollard's kangaroos, to make it memory efficient and parallelizable.

Key fact: use the fact that we know more than the group order modulo m .

Implemented in Magma and in C++/NTL.

- Magma:
 - high level computations;
 - Gröbner basis for division by 2 and 3;
 - Prediction of factorization patterns of the resultant;
- NTL:
 - Resultant computation;
 - Factorization of large degree polynomials;
 - MCT algorithm;

NB: The C++ programmes are called inside Magma; communication is done via files and (named) pipes.

Let

$$p = 5 \times 10^{24} + 8503491.$$

Early abort strategy:

- pick a random genus 2 curve;
- do the Schoof algorithm; if the group order is divisible by a small prime, start again;
- finish with the other algorithms;

⇒ we found 32 curves with a known Jacobian order $\approx 2^{164}$.

Together with the twists, this gives 64 group orders with no small factors.

Among them, 7 were prime, yielding secure cryptosystems.

Runtime on a Pentium IV at 2 GHz.

ℓ	deg	Resultant	$X^p \bmod R$	ModComp
11	28 680	1 500 s	75 s	325 s
13	56 280	5 100 s	200 s	1 200 s
17	165 600	35 000 s	2 200 s	24 000 s
19	258 840	70 000 s	2 300 s	54 000 s

Division by 2: upto 1024-torsion, **15 h**.

Division by 3: upto 27-torsion **2 h**.

MCT algo: **3 h**.

Total: about **10 days**. (depends heavily on the degree of the smallest factor of the resultant).

Computation is speeded-up by using:

- Algorithms for symmetric polynomials to divide the degree by 2;
- Algorithm by Bostan-Flajolet-Salvy-Schost for parasites;
- Use the predictable factorization pattern of division polynomials to help the factorization;
- Gröbner basis computation for division by 2 and 3;
- Factorization using explicit Galois action on the roots;
- Parallelization whenever it is possible.

Big open question for further improvements:

How to make the **Elkies-Atkin improvements**
(modular equations) efficient in genus 2 ?