Genus 1 and 2

Point counting

Division polys

Kernels

Schoof complexity

BSGS

Real multiplication

Split primes

Smaller kernels

New relations

RM Complexity

1 = 2

RM families

Implementation

Cryptographic Jacobians

Too much, too fast

Point Counting for Genus 2 Curves with Real Multiplication

Pierrick Gaudry, David Kohel, Benjamin Smith

Benjamin Smith
INRIA Saclay–Île-de-France
Laboratoire d'Informatique de l'École polytechnique (LIX)

ECC 2011, Nancy, France 21/09/2011

Genus 1 and 2
Point counting
Division polys
Kernels
Schoof complexity
BSGS
Real multiplication
Split primes
Smaller kernels
New relations
RM Complexity
1 = 2
RM families
Implementation
Cryptographic Jacobians
Too much, too fast

Genus 2 cryptosystems have security and efficiency comparable* with elliptic curve cryptosystems...

...but setting up secure genus 2 instances is much harder.

Computing cardinalities over prime fields:

- 256-bit elliptic curve: SEA in seconds
- 256-bit abelian surface: replace seconds with days.

Given $C : y^2 = f(x)$ of genus 2 over $\mathbb{F}_q$
(q odd, J_C ordinary, absolutely irreducible).

We want to compute $\#J_C(\mathbb{F}_q)$. *Equivalently:*
Compute the characteristic polynomial of Frobenius

$$\chi(T) = T^4 - s_1 T^3 + (s_2 + 2q) T^2 - q s_1 T + q^2,$$

which is subject to the Weil bounds

$$|s_1| \leq 4\sqrt{q} \quad \text{and} \quad |s_2| \leq 4q$$

and the Rück bounds

$$s_1^2 - 4s_2 \geq 0 \quad \text{and} \quad s_2 + 4q \geq 2|s_1|.$$

Genus 1 and 2
Point counting
Division polys
Kernels
Schoof complexity
BSGS
Real multiplication
Split primes
Smaller kernels
New relations
RM Complexity
1 = 2
RM families
Implementation
Cryptographic Jacobians
Too much, too fast

Schoof's idea:

characteristic polynomial of Frobenius acting on $J_C[\ell]$ is

$$\chi_\ell(T) := \chi(T) \mod (\ell), \quad \text{so}$$

$$(\pi^2 + [\bar{q}])^2(D) - [\bar{s}_1](\pi^2 + [\bar{q}])\pi(D) + [\bar{s}_2]\pi^2(D) = 0$$

for all D in $J_C[\ell]$ *(here $\bar{\cdot}$ denotes residue mod ℓ)*.

- Compute χ_ℓ for sufficiently many prime (powers) ℓ
- Recover χ via the CRT.

To compute χ_ℓ:

1. compute generic D in $J_C[\ell]$;
2. compute $\pi^2(D)$, $(\pi^2 + [\bar{q}])\pi(D)$, and $(\pi^2 + [\bar{q}])^2(D)$;
3. search for $[\bar{s}_1]$ and $[\bar{s}_2]$ s.t. the relation holds.

Let (u, v) be a generic point of C, and D its image in J_C.

We say $\phi \in \mathrm{End}(J_C)$ is *explicit* if we can compute polynomials $d_0, d_1, d_2, e_0, e_1, e_2$ such that

$$\phi(D) = \left(x^2 + \frac{d_1(u)}{d_2(u)}x + \frac{d_0(u)}{d_2(u)}, y - v\left(\frac{e_1(u)}{e_2(u)}x + \frac{e_0(u)}{e_2(u)}\right)\right).$$

We call the d_i and e_i the *ϕ-division polynomials*.
($=$ Cantor's ℓ-division polys for $\phi = [\ell]$)

We say that ϕ is *efficiently computable* if the ϕ-division polynomials have low degree.
(ie, if evaluating ϕ is in $O(1)$ field ops)

Note: $[\ell]$-division polys have degree in $O(\ell^2)$

Computing generic elements of $\ker \phi \subset J_C$

Let ϕ be an explicit endomorphism,
$(u_1, v_1), (u_2, v_2)$ generic points on C,
D_1, D_2 their images in J_C.

$$D = (x^2 + a_1 x + a_0, y - (b_1 x + b_0)) := D_1 + D_2$$

is a generic point of J_C.

1. Compute $\phi(D_1)$ and $\phi(D_2)$;
2. Solve for (u_1, v_1, u_2, v_2) in $\phi(D_1) = -\phi(D_2)$;
3. Resymmetrizing, compute a triangular ideal I_ϕ of relations in a_1, a_0, b_1, b_0 satisfied when $D \in \ker \phi$.

Suppose degree of ϕ-division polynomials bounded by δ:

- compute I_ϕ in $\widetilde{O}(\delta^3)$ $\mathbb{F}_q$-operations;
- the degree of I_ϕ is in $O(\delta^2)$

Genus 1 and 2
Point counting
Division polys
Kernels
Schoof complexity
BSGS
Real multiplication
Split primes
Smaller kernels
New relations
RM Complexity
1 = 2
RM families
Implementation
Cryptographic Jacobians
Too much, too fast

Genus 1 and 2
Point counting
Division polys
Kernels
Schoof complexity
BSGS
Real multiplication
Split primes
Smaller kernels
New relations
RM Complexity
1 = 2
RM families
Implementation
Cryptographic Jacobians
Too much, too fast

Conventional Schoof–Pila complexity:

- For each prime ℓ:
  1. Compute I_ℓ in $\widetilde{O}(\ell^6)$ field ops
     - $[\ell]$-division polynomials have degree in $O(\ell^2)$
     - triangular I_ℓ has degree in $O(\ell^4)$
  2. compute $\pi^2(D)$, $(\pi^2 + [\bar{q}])\pi(D)$, and $(\pi^2 + [\bar{q}])^2(D)$ in $\widetilde{O}(\ell^4 \log q)$ field ops
  3. Find the $(\bar{s}_1, \bar{s}_2)$ in $(\mathbb{Z}/\ell\mathbb{Z})^2$ such that $(\pi^2 + [\bar{q}])^2(D) - [\bar{s}_1](\pi^2 + [\bar{q}])\pi(D) + [\bar{s}_2]\pi^2(D) = 0$ ...$O(\ell)$ trials, each costing $\widetilde{O}(\ell^4)$ field ops $\implies$ total cost $\widetilde{O}(\ell^5)$ field ops

  $\implies$ Computing χ_ℓ costs $\widetilde{O}(\ell^4(\ell^2 + \log q))$ field ops
- We need χ_ℓ for the $O(\log q)$ primes ℓ in $O(\log q)$
- $\implies$ χ costs $\widetilde{O}(\log^7)$ field ops $= \widetilde{O}(\log^8 q)$ **bit ops**

Genus 1 and 2
Point counting
Division polys
Kernels
Schoof complexity
BSGS
Real multiplication
Split primes
Smaller kernels
New relations
RM Complexity
1 = 2
RM families
Implementation
Cryptographic Jacobians
Too much, too fast

Computing in $J_C[\ell]$ becomes awkward very quickly in genus 2; we're limited to $\ell = O(\text{a handful of bits})$.

This gives us s_1 and s_2 modulo some integer M.

We finish the computation using a generic algorithm such as BSGS, which runs in time

- $\widetilde{O}(q^{3/4}/M)$ when $M < 8\sqrt{q}$, and
- $\widetilde{O}(\sqrt{q/M})$ when $M \geq 8\sqrt{q}$.

This all sounds pretty bad.
Why would we want to use genus 2 again, anyway?

Remember:

Genus 2 is not just a two-dimensional analogue of genus 1 (it's much more fun than that).

Recall:

- $\mathrm{End}(J_C) \otimes \mathbb{Q} = \mathbb{Q}(\pi)$ is a quartic CM-field.
- Complex conjugation = Rosati involution $\alpha \mapsto \alpha^\dagger$
- Real quadratic subfield: $\mathbb{Q}(\pi + \pi^\dagger) \cong \mathbb{Q}(\sqrt{\Delta})$ for some $\Delta > 0$.
- We say C has RM by $\mathcal{O}$ if $\mathcal{O}$ is a real quadratic order isomorphic to a subring of $\mathrm{End}(J_C)$
- isomorphism classes with RM by a fixed $\mathcal{O}$ form *Humbert surfaces* in the 3-dimensional moduli space.

Genus 1 and 2
Point counting
Division polys
Kernels
Schoof complexity
BSGS
Real multiplication
Split primes
Smaller kernels
New relations
RM Complexity
$1 = 2$
RM families
Implementation
Cryptographic Jacobians
Too much, too fast

Genus 1 and 2
Point counting
Division polys
Kernels
Schoof complexity
BSGS
Real multiplication
Split primes
Smaller kernels
New relations
RM Complexity
$1 = 2$
RM families
Implementation
Cryptographic Jacobians
Too much, too fast

Elliptic Curves with Schoof–Elkies–Atkin

- $\mathbb{Z}[\pi]$ is an unknown quadratic extension of $\mathbb{Z}$.
- Some primes ℓ split in $\mathbb{Z}[\pi]$.
- $(\ell) = (\alpha)(\bar{\alpha}) \implies E[\ell] = E[\alpha] \oplus E[\bar{\alpha}]$
- For these primes, compute modulo deg$(\ell - 1)/2$ factors of division polynomials (of deg$(\ell^2 - 1)/2$).
- Heuristically (assuming enough split primes), reduces complexity from $\widetilde{O}(\log^5 q)$ to $\widetilde{O}(\log^4 q)$ bit ops.
- **Problem**: we don't know which ℓ split in advance; testing and splitting a given ℓ is complicated...
  - Need to build & factor modular polynomials
  - Extension to genus 2 is problematic

Genus 1 and 2
Point counting
Division polys
Kernels
Schoof complexity
BSGS
Real multiplication
Split primes
Smaller kernels
New relations
RM Complexity
$1 = 2$
RM families
Implementation
Cryptographic Jacobians
Too much, too fast

Our idea:

- $\mathbb{Z} \subset \mathbb{Z}[\phi] \subset \mathbb{Z}[\pi, \pi^\dagger]$; but $\mathbb{Z} \subset \mathbb{Z}[\phi]$ is explicit, so we can split primes ℓ in $\mathbb{Z}[\phi]$ instead of $\mathbb{Z}[\pi, \pi^\dagger]$
- Split $(\ell) = (\alpha_1)(\alpha_2) \implies J_{\mathcal{C}}[\ell] = J_{\mathcal{C}}[\alpha_1] \oplus J_{\mathcal{C}}[\alpha_2]$.
  Efficient $\phi \implies$ explicit $J_{\mathcal{C}}[\alpha_1]$ and $J_{\mathcal{C}}[\alpha_2]$.
- Compute in $J_{\mathcal{C}}[\alpha_1]$ and $J_{\mathcal{C}}[\alpha_2]$ faster than in $J_{\mathcal{C}}[\ell]$.
- Hence, compute χ_ℓ faster for split ℓ.
- The split ℓ are known in advance: $(\Delta/\ell) = 1$; Cebotarev density $\implies$ half the primes ℓ split in $\mathbb{Z}[\phi]$.
- Also, explicit $\mathbb{Z}[\phi] \implies$ a better search space (so we need fewer χ_ℓ to determine χ).
- $\longrightarrow$ a *much* better complexity for computing χ.

Genus 1 and 2
Point counting
Division polys
Kernels
Schoof complexity
BSGS
Real multiplication
Split primes
Smaller kernels
New relations
RM Complexity
$1 = 2$
RM families
Implementation
Cryptographic Jacobians
Too much, too fast

The details:

Suppose ℓ splits in $\mathbb{Z}[\phi]$.
For our families, the primes over ℓ are principal:

$$(\ell) = (\alpha_1)(\alpha_2) \quad \text{and} \quad J_{\mathcal{C}}[\ell] = J_{\mathcal{C}}[\alpha_1] \oplus J_{\mathcal{C}}[\alpha_2].$$

- We can compute generators $\alpha_i = a_i + b_i\phi$ with a_i, b_i in $O(\sqrt{\ell})$
- The $[a_i]$- and $[b_i]$-division polys have degree in $O(\ell)$
- $\implies$ the α_i-division polys have degree in $O(\ell)$
- $\implies$ kernel ideals I_{α_i} have degrees in $O(\ell^2)$ (& we can compute I_{α_i} in $\widetilde{O}(\ell^3)$ field operations).

Genus 1 and 2
Point counting
Division polys
Kernels
Schoof complexity
BSGS
Real multiplication
Split primes
Smaller kernels
New relations
RM Complexity
1 = 2
RM families
Implementation
Cryptographic Jacobians
Too much, too fast

Suppose $\mathbb{Z}[\pi + \pi^\dagger] \subset \mathbb{Z}[\phi]$, so

$$\pi + \pi^\dagger = m + n\phi$$

for some m and n in $O(\sqrt{q})$. These determine s_1 and s_2:

$$s_1 = \mathrm{Tr}(\pi + \pi^\dagger) = 2m + n\mathrm{Tr}(\phi)$$
$$s_2 = \mathrm{N}(\pi + \pi^\dagger) = \tfrac{1}{4}(s_1^2 - n^2\mathrm{disc}(\mathbb{Z}[\phi])).$$

- $(\pi^2 + [\bar{q}])(D) = [y_i]\pi(D)$ for D in $J_C[a_i + b_i\phi]$, where $y_i = (m - na_i/b_i) \bmod \ell$.
- So we find $\bar{s}_1$ and $\bar{s}_2$ by finding y_1 and y_2: ie $2\times$ one-dimensional DLP in $(\mathbb{Z}/\ell\mathbb{Z})$ (and with fewer costly Frobenius applications).

Genus 1 and 2
Point counting
Division polys
Kernels
Schoof complexity
BSGS
Real multiplication
Split primes
Smaller kernels
New relations
RM Complexity
$1 = 2$
RM families
Implementation
Cryptographic Jacobians
Too much, too fast

RM Schoof–Pila complexity

- For each *split* prime $(\ell) = (\alpha_1)(\alpha_2)$
  1. Compute I_{α_1}, I_{α_2} (deg $O(\ell^2)$) in $\widetilde{O}(\ell^3)$ field ops
  2. Compute $(\pi^2 + [\bar{q}])(D_i)$, $\pi(D_i)$ in $\widetilde{O}(\ell^2 \log q)$ field ops
  3. Recover $\bar{m}, \bar{n}$ from $\bar{y}_1, \bar{y}_2$ in $\mathbb{Z}/\ell\mathbb{Z}$ such that $(\pi^2 + [\bar{q}])(D_i) = [y_1]\pi(D_i)$ ...$O(\sqrt{\ell})$ trials, each costing $\widetilde{O}(\ell^2)$ field ops $\implies$ total cost $\widetilde{O}(\ell^{3/2})$ field ops

  $\implies$ Computing χ_ℓ costs $\widetilde{O}(\ell^2(\ell + \log q))$ field ops *(vs conventional $\widetilde{O}(\ell^4(\ell^2 + \log q))$ field ops)*
- We need χ_ℓ for the $O(\log q)$ split primes in $O(\log q)$
- $\implies$ χ in $\widetilde{O}(\log^4 q)$ field ops $=$ $\widetilde{O}(\log^5 q)$ **bit ops** *(vs conventional $\widetilde{O}(\log^8 q)$ bit ops)*

Genus 1 and 2
Point counting
Division polys
Kernels
Schoof complexity
BSGS
Real multiplication
Split primes
Smaller kernels
New relations
RM Complexity
1 = 2
RM families
Implementation
Cryptographic Jacobians
Too much, too fast

Check it out:

- Schoof for Elliptic Curves / $\mathbb{F}_q$:
  proven $\widetilde{O}(\log^5 q)$ *bit ops*
- Schoof–Elkies–Atkin for Elliptic Curves / $\mathbb{F}_q$:
  heuristic $\widetilde{O}(\log^4 q)$ *bit ops*
- RM Schoof–Pila for genus 2 / $\mathbb{F}_q$:
  proven $\widetilde{O}(\log^5 q)$ *bit ops*

So point counting has the same unconditional complexity for genus 2 explicit-RM curves over $\mathbb{F}_q$ and elliptic curves *over the same* $\mathbb{F}_q$!

We can construct genus 2 curves with efficient RM using some explicit one/two-parameter families.
(Mestre, Tautz–Top–Verberkmoes, Hashimoto, Brumer...)

Consider the Tautz–Top–Verberkmoes family

$$\mathcal{C} : y^2 = x^5 - 5x^3 + 5x + t.$$

We have an explicit endomorphism ϕ defined by

$$\phi((u, v)) = (x^2 - \tau ux + u^2 + \tau^2 - 4, y - v)$$

where $\tau = \zeta_5 + \zeta_5^{-1}$ (in $\mathbb{F}_q$ if $q \not\equiv \pm 2 \bmod 5$).

We have $\phi^2 + \phi - 1 = 0$, so
$\mathcal{C}$ has efficient RM by $\mathbb{Z}[\phi] \cong \mathbb{Z}[\frac{1+\sqrt{5}}{2}]$.

Genus 1 and 2
Point counting
Division polys
Kernels
Schoof complexity
BSGS
Real multiplication
Split primes
Smaller kernels
New relations
RM Complexity
1 = 2
RM families
Implementation
Cryptographic Jacobians
Too much, too fast

Genus 1 and 2
Point counting
Division polys
Kernels
Schoof complexity
BSGS
Real multiplication
Split primes
Smaller kernels
New relations
RM Complexity
1 = 2
RM families
Implementation
Cryptographic Jacobians
Too much, too fast

A proof-of-concept implementation

Algorithm implemented in C++/NTL
(with Magma for non-critical steps).

- We did *not* use any small prime powers
- We did *not* use BSGS, just accelerated Schoof–Pila

Genus 1 and 2
Point counting
Division polys
Kernels
Schoof complexity
BSGS
Real multiplication
Split primes
Smaller kernels
New relations
RM Complexity
$1 = 2$
RM families
Implementation
Cryptographic Jacobians
Too much, too fast

Cryptographic Jacobians: 256 bits

We searched for a secure genus 2 curve in the family
$$\mathcal{C} : y^2 = x^5 - 5x^3 + 5x + t$$
over $\mathbb{F}_q$ with $q = 2^{128} + 573$.

Computing $\chi(T)$ for a given specialization takes about 3 Core2 core-hours at 2.83GHz; we use the split primes $\ell \leq 131$.

We ran 245 trials, finding 27 prime-order Jacobians.

We found that the Jacobian of the curve at
$$t = 75146620714142230387068843744286456025$$
has prime order, and so does its quadratic twist.

<u>But 256 bits is so two years ago...</u>

...so we computed the order of a kilobit Jacobian (!)

We computed $\chi(T)$ for $C : y^2 = x^5 - 5x^3 + 5x + t$ over $\mathbb{F}_q$ with $q = 2^{512} + 1273$ and

$$t = 2908566633378727243799826112991980174977453300368095776223256986807375270272014471477919882845604269700820270816721532434975921085316560590832659122351278.$$

The computation took about 80 core-days (same setup as before); we use the split primes $\ell \leq 419$.

The cardinality is

$$
\begin{aligned}
N = {} & 17976931348623159077293051907890247336179 \\
& 76978942306572734300811577326758055023757 \\
& 37059489561441845417204171807809294449627 \\
& 63452801227364805323818926258902074851818 \\
& 08988886875773723732892032531588464639346 \\
& 29657544938945248034686681123456817063106 \\
& 48544084486938739666585942218663644225871 \\
& 26841779001051190055 20.
\end{aligned}
$$