Genus 1 point counting in quadratic space and essentially quartic time

Andrew V. Sutherland

Massachusetts Institute of Technology

April 21, 2010

Introduction

A quote from the current world-record holder for genus 1 point counting in large characteristic (8302-bit prime field).

> *“Despite this progress, computing modular polynomial remains the stumbling block for new point counting records. Clearly, to circumvent the memory problems, one would need an algorithm that directly obtains the polynomial specialised in one variable.”*

INRIA Project TANC

Genus 1 point counting in large characteristic

Given an elliptic curve $E/\mathbb{F}_q$, we wish to compute $\#E(\mathbb{F}_q)$.
We assume q is prime and set $n = \log q$.

Algorithm	Time	Space
Schoof's algorithm	$O(n^5 \operatorname{llog} n)$	$O(n^3)$
SEA†	$O(n^4 \log^3 n \operatorname{llog} n)$	$O(n^3 \log n)$
SEA (precomputed Φ_ℓ)	$O(n^4 \operatorname{llog} n)$	$O(n^4)$
Today's talk (GRH)	$O(n^4 \log^2 n \operatorname{llog} n)$	$O(n^2)$
Amortized	$O(n^4 \operatorname{llog} n)$	$O(n^2 \log^2 n)$

†Assumes Φ_ℓ is computed in time $O(\ell^3 \log^4 \ell \operatorname{llog} \ell)$ [Enge '09].

Space and time

In a universe with d dimensions, the amount of data that can be stored within a distance r of the CPU is $O(r^d)$.

An algorithm with space complexity S is at a distance $\Omega(S^{1/d})$ from its data. Access times increase exponentially with $\log S$.

Conversely, reducing space reduces time.

And increases parallelism.

Schoof's algorithm

1. For sufficiently many primes ℓ (up to $\approx n/2$):
   Determine which $t_\ell = 0, 1, \ldots, \ell - 1$ satisfies

$$\pi^2 - [t_\ell]\pi + [q_\ell] \equiv 0 \qquad \bmod f_\ell, E$$

   where $t_\ell = t \bmod \ell$ and $q_\ell = q \bmod \ell$.

2. Use the CRT to uniquely determine $t \in [-2\sqrt{q}, 2\sqrt{q}]$.

The computation of $\pi(x, y) = (x^q, y^q) \bmod f_\ell, E$ dominates.

$$T = \sum_\ell O(n\mathsf{M}(\ell^2 n)) = O(n^5 \operatorname{llog} n)$$

$$S = \max_\ell O(\ell^2 n) = O(n^3)$$

SEA algorithm (Elkies version)

1. For sufficiently many primes ℓ (up to $\approx n$):
   Compute $\Phi_\ell(X, Y)$.
   Evaluate $\phi(Y) = \Phi_\ell\big(\mathrm{j}, Y\big)$, where $\mathrm{j} = \mathrm{j}(E)$.
   If ϕ has a root $\tilde{\mathrm{j}}$ in $\mathbb{F}_q$ then
   Compute a normalized isogeny to $\tilde{E}/\mathbb{F}_q$.
   Compute a factor g_ℓ of f_ℓ.
   Determine which $\lambda_\ell = 0, 1, \ldots, \ell - 1$ satisfies

$$\pi - [\lambda_\ell] \equiv 0 \qquad \mathrm{mod}\ g_\ell, E,$$

   and set $t_\ell = \lambda_\ell + q_\ell/\lambda_\ell \bmod \ell$.
2. Use the CRT to uniquely determine $t \in [-2\sqrt{q}, 2\sqrt{q}]$.

SEA complexity (Elkies version)

Task (for each ℓ)	Time	Space
Compute Φ_ℓ	$O(\ell^3 \log^3 \ell \mathsf{M}(\ell))$	$O(\ell^3 \log \ell)$
Compute ϕ	$O(\ell^2 \mathsf{M}(\ell + n))$	$O(\ell^3 \log \ell)$
Find a root $\tilde{\jmath}$	$O(n\mathsf{M}(\ell n))$	$O(\ell n)$
Construct $\tilde{E}$	$O(\ell^2 \mathsf{M}(n))$	$O(\ell^2 n)$
Compute g_ℓ	$O(\ell^2 \mathsf{M}(n))^\dagger$	$O(\ell n)$
Compute π	$O(n\mathsf{M}(\ell n))$	$O(\ell n)$
Find λ_ℓ (linear)	$O(\ell \mathsf{M}(\ell n))$	$O(\ell n)$
Find λ_ℓ (BSGS)	$O(\sqrt{\ell} \mathsf{M}(\ell n))$	$O(\ell^{3/2} n)$

Applying the CRT takes $O(\mathsf{M}(n) \log n)$ time and $O(n)$ space.

†Can be made $O(\mathsf{M}(\ell)\mathsf{M}(n))$ using [BGMS 2007].

Computing Φ_ℓ with the CRT

Strategy: compute $\Phi_\ell \bmod p$ for sufficiently many primes p and use the CRT to compute Φ_ℓ (or $\Phi_\ell \bmod q$).

- For "special" primes p we can compute $\Phi_\ell \bmod p$ in time $O(\ell^2 \log^3 p \operatorname{llog} p)$ using isogeny volcanoes [BLS 2010].
- Assuming the GRH, we can efficiently find many special p with $\log p = O(\log \ell)$.

Computing Φ_ℓ takes $O(\ell^3 \log^3 \ell \operatorname{llog} \ell)$ time and $O(\ell^3 \log \ell)$ space.

We can directly compute $\Phi_\ell \bmod q$ using $O(\ell^2(n + \log \ell))$ space. But this is still bigger than we want (or need)...

Computing ϕ with the CRT (version 1)

Strategy: "lift" $\mathrm{j} = \mathrm{j}(E)$ from $\mathbb{F}_q$ to $\mathbb{Z}$ and then compute

$$\phi(Y) = \Phi_\ell(\mathrm{j}, Y) \bmod p$$

for sufficiently many (special) primes p and use the explicit CRT to obtain $\phi \bmod q$.

This uses $O(\ell^2 \log p \operatorname{llog} p)$ time for each p, in $O(\ell \log p)$ space.

However, "sufficiently many" is $O(\ell n)$.
Total time is $O(\ell^3 n \log \ell \operatorname{llog} \ell)$, using $O(\ell n + \ell \log \ell))$ space.

In situations where $n \ll \ell$ this may be useful, but not in SEA.

Computing ϕ with the CRT (version 2)

Strategy: “lift” $\mathrm{j}, \mathrm{j}^2, \ldots, \mathrm{j}^{\ell+1}$ from $\mathbb{F}_q$ to $\mathbb{Z}$, then compute

$$\phi(Y) = \Phi_\ell(\mathrm{j}, Y) \bmod p$$

for sufficiently many (special) primes p and use the explicit CRT to obtain $\phi \bmod q$.

This uses $O(\ell^2 \log^3 p \operatorname{llog} p)$ time per prime p, in $O(\ell^2)$ space.

Now “sufficiently many” is $O(\ell + n)$.
Total time is $O(\ell^3 \log^3 \ell \operatorname{llog} \ell)$, using $O(\ell n + \ell^2)$ space.

This is perfect for SEA, but it isn’t enough...

Modified SEA complexity (in progress)

Task (for each ℓ)	Time	Space
Compute ϕ	$O(\ell^3 \log^3 \ell \operatorname{llog} \ell)$	$O(\ell n + \ell^2)$
Find a root $\tilde{\jmath}$	$O(n\mathsf{M}(\ell n))$	$O(\ell n)$
Construct $\tilde{E}$	$O(\ell^2\mathsf{M}(n))$	$O(\ell^2 n)$
Compute g_ℓ	$O(\ell^2\mathsf{M}(n))$	$O(\ell n)$
Compute π	$O(n\mathsf{M}(\ell n))$	$O(\ell n)$
Find λ_ℓ (linear)	$O(\ell\mathsf{M}(\ell n))$	$O(\ell n)$
Find λ_ℓ (BSGS)	$O(\sqrt{\ell}\mathsf{M}(\ell n))$	$O(\ell^{3/2} n)$

Computing $\tilde{E}$ (and p_1)

To compute g_ℓ we need to correctly normalize the equation

$$y^2 = x^3 + \tilde{a}x + \tilde{b}$$

for the isogenous curve $\tilde{E}$. We also want p_1 (the kernel sum).
To obtain $\tilde{E}$ and p_1 we need to compute:

$$\frac{\tilde{\jmath}'}{\jmath'} = -\frac{\Phi_X(\jmath, \tilde{\jmath})}{\ell \Phi_Y(\jmath, \tilde{\jmath})}$$

$$\frac{\jmath''}{\jmath'} - \ell\frac{\tilde{\jmath}''}{\tilde{\jmath}'} = -\frac{\jmath'^2\Phi_{XX}(\jmath, \tilde{\jmath}) + 2\ell\jmath'\tilde{\jmath}'\Phi_{XY}(\jmath, \tilde{\jmath}) + \ell^2\tilde{\jmath}'^2\Phi_{YY}(\jmath, \tilde{\jmath})}{\jmath'\Phi_X(\jmath, \tilde{\jmath})}$$

This requires us to evaluate various partial derivatives of Φ_ℓ.

Computing ϕ_x and ϕ_{xx}

Let $\phi_X(Y) = \Phi_X(j, Y)$ and let $\phi_{XX}(Y) = \Phi_{XX}(j, Y)$.
We can compute ϕ_X and ϕ_{XX} as we compute ϕ (low cost).
We then use:

$$\begin{aligned}
\Phi_X(\mathrm{j}, \tilde{\mathrm{j}}) &= \phi_X(\tilde{\mathrm{j}}) \\
\Phi_Y(\mathrm{j}, \tilde{\mathrm{j}}) &= \phi'(\tilde{\mathrm{j}}) \\
\Phi_{XX}(\mathrm{j}, \tilde{\mathrm{j}}) &= \phi_{XX}(\tilde{\mathrm{j}}) \\
\Phi_{YY}(\mathrm{j}, \tilde{\mathrm{j}}) &= \phi''(\tilde{\mathrm{j}}) \\
\Phi_{XY}(\mathrm{j}, \tilde{\mathrm{j}}) &= \phi_X'(\tilde{\mathrm{j}})
\end{aligned}$$

which allows us to compute $\tilde{E}$, p_1, and g_ℓ.

Modified SEA complexity (in progress)

Task (for each ℓ)	Time	Space
Compute ϕ	$O(\ell^3 \log^3 \ell \operatorname{llog} \ell)$	$O(\ell n + \ell^2 \log \ell)$
Find a root $\tilde{\jmath}$	$O(n\mathsf{M}(\ell n))$	$O(\ell n)$
Construct $\tilde{E}$	$O(\ell \mathsf{M}(n))$	$O(\ell n)$
Compute g_ℓ	$O(\ell^2 \mathsf{M}(n))$	$O(\ell n)$
Compute π	$O(n\mathsf{M}(\ell n))$	$O(\ell n)$
Find λ_ℓ (linear)	$O(\ell \mathsf{M}(\ell n))$	$O(\ell n)$
Find λ_ℓ (BSGS)	$O(\sqrt{\ell}\mathsf{M}(\ell n))$	$O(\ell^{3/2} n)$

Space efficient BSGS

Using a baby-steps giant-steps search to find $[\lambda_\ell] = \pi$ typically involves comparing rational functions of size $O(\ell n)$ with numerators and denominators in the ring $R = \mathbb{F}_q[x, y]/(g_\ell, E)$.

This is a big ring, but we only care about functions that correspond to one of the $\ell - 1$ possible values for λ_ℓ.
With a unique representation, we can use $O(\log \ell)$-bit hashes.

This can be achieved by inverting denominators in R.
Equivalently, compute in $E(R)$ using affine coordinates.

If an inversion fails (unlikely), we find a proper factor of g_ℓ and can reduce the degree of g_ℓ by at least a factor of 2.

Modified SEA complexity (final?)

Task (for each ℓ)	Time	Space
Compute ϕ	$O(\ell^3 \log^3 \ell \operatorname{llog} \ell)$	$O(\ell n + \ell^2)$
Find a root $\tilde{\jmath}$	$O(n\mathsf{M}(\ell n))$	$O(\ell n)$
Construct $\tilde{E}$	$O(\ell \mathsf{M}(n))$	$O(\ell n)$
Compute g_ℓ	$O(\ell^2 \mathsf{M}(n))$	$O(\ell n)$
Compute π	$O(n\mathsf{M}(\ell n))$	$O(\ell n)$
Find λ_ℓ (BSGS)	$O(\sqrt{\ell}\mathsf{M}(\ell n))$	$O(\ell n)$

Total time is $O(n^4 \log^2 n \operatorname{llog} n)$ using $O(n^2)$ space.

We can simultaneously compute $\phi \bmod q$ for $O(\log^2 n)$ curves at essentially no additional cost.

Amortized complexity: $O(n^4 \operatorname{llog} n)$ time using $O(n^2 \log^2 n)$ space.

Alternative modular polynomials

In practice, the modular polynomials Φ_ℓ are not used in SEA. There are alternatives (due to Atkin, Müller, and others) that are smaller by a large constant factor (100x to 1000x is typical).

The isogeny-volcano approach of [BLS 2010] can compute many types of (symmetric) modular polynomials derived from modular functions other than $\mathfrak{j}(z)$, but these do not include the (non-symmetric) polynomials commonly used with SEA.

They do include modular polynomials $\Phi_\ell^{\mathfrak{f}}$ derived from the Weber function $\mathfrak{f}(z)$. These are smaller than Φ_ℓ by a factor of 1728, but they have never(?) been used with SEA before.

The Weber modular polynomials $\Phi_\ell^{\mathfrak{f}}$

The Weber $\mathfrak{f}$-function is related to the $\mathfrak{j}$-function via

$$\mathfrak{j} = \Psi(\mathfrak{f}) = \frac{(\mathfrak{f}^{24} - 16)^3}{\mathfrak{f}^{24}}$$

Provided $\mathrm{End}(E)$ has discriminant $D \equiv 1 \bmod 8$ with $3 \nmid D$, the polynomial $\phi^{\mathfrak{f}} = \Phi_\ell^{\mathfrak{f}}(\mathfrak{f}(E), Y)$ parametrizes ℓ-isogenies from E.

This condition is easily checked (without knowing D), and if it fails, powers of $\mathfrak{f}$, or other modular functions may be used.

But we need to know how to compute normalized isogenies!

Using $\Phi_\ell^{\mathfrak{f}}$ to compute normalized isogenies

- Compute $\mathfrak{f} = \mathfrak{f}(E)$ satisfying $\Psi(\mathfrak{f}) = \mathfrak{j}$.
- Compute $\phi^{\mathfrak{f}}$, $\phi_X^{\mathfrak{f}}$, and $\phi_{XX}^{\mathfrak{f}}$ and also

$$\frac{\tilde{\mathfrak{f}}'}{\mathfrak{f}'} = \cdots \qquad \text{and} \qquad \frac{\mathfrak{f}''}{\mathfrak{f}'} - \ell\frac{\tilde{\mathfrak{f}}''}{\tilde{\mathfrak{f}}'} = \cdots$$

- Now apply

$$\frac{\tilde{\mathfrak{j}}'}{\mathfrak{j}'} = \frac{\tilde{\mathfrak{f}}'}{\mathfrak{f}'}\frac{\Psi'(\tilde{\mathfrak{f}})}{\Psi'(\mathfrak{f})}$$

$$\frac{\mathfrak{j}''}{\mathfrak{j}'} - \ell\frac{\tilde{\mathfrak{j}}''}{\tilde{\mathfrak{j}}'} = \frac{\mathfrak{f}''}{\mathfrak{f}'} - \ell\frac{\tilde{\mathfrak{f}}''}{\tilde{\mathfrak{f}}'} + \frac{\Psi''(\mathfrak{f})}{\Psi'(\mathfrak{f})}\mathfrak{f}' - \frac{\Psi''(\tilde{\mathfrak{f}})}{\Psi'(\tilde{\mathfrak{f}})}\ell\tilde{\mathfrak{f}}'$$

and use $\tilde{\mathfrak{j}} = \Psi(\tilde{\mathfrak{f}})$ to construct $\tilde{E}$, p_1, and g_ℓ as before.

Practical results: modular polynomial records

- $\ell = 10079 : 120$ cpu-days (2.4 GHz AMD) to compute a Müller polynomial of size 16GB [Enge 2007].
- $\ell = 10079 : 1$ cpu-hour (3.0 GHz AMD) to compute $\Phi_\ell^{\mathfrak{f}}$ of size 3GB [BLS 2010].
- $\ell = 60013 : 13$ cpu-days (3.0 GHz AMD) to compute $\Phi_\ell^{\mathfrak{f}}$ of size 748GB [BLS 2010].
- $\ell = 100019 : 100$ cpu-days (3.0 GHz AMD) to compute $\Phi_l^{\mathfrak{f}}(\mathfrak{f}(E), Y) \bmod (2^{86243} - 1)$ of size 1GB [S 2010].

For $\ell = 100019$, the size of $\Phi_l^{\mathfrak{f}}$ is over 1TB and Φ_ℓ is over 1PB.

Practical results: point-counting example

$$y^2 = x^3 + 31415926x + 27182818$$
$$q = 10^{3000} + 1027$$

Task ($\ell = 6599$)	Time
Compute $\phi^{\mathfrak{f}}, \phi^{\mathfrak{f}}_X, \phi^{\mathfrak{f}}_{XX}$	1074s
Find a root $\tilde{\mathfrak{f}}$	63983s
Construct $\tilde{E}$	0s
Compute g_ℓ	360s
Compute $\pi^\dagger$	61427s
Find λ_ℓ (119 BSGS steps)	5216s
Total time to compute t_ℓ	132064s

Memory used while computing $\phi^{\mathfrak{f}}$: 60MB.
Memory used for root-finding (NTL): 200MB.

†Can be improved by ≈ 2x using [GM 2006].

Genus 1 point counting in quadratic space and essentially quartic time

Andrew V. Sutherland

Massachusetts Institute of Technology

April 21, 2010