Computing Hilbert class polynomials with the CRT method

Andrew V. Sutherland

Massachusetts Institute of Technology

September 23, 2008

Computing $H_D(x)$

Three algorithms

1. Complex analytic
2. p-adic
3. Chinese Remainder Theorem (CRT)

Computing $H_D(x)$

Three algorithms

1. Complex analytic
2. *p*-adic
3. Chinese Remainder Theorem (CRT)

Comparison

Heuristically, all have complexity $O(|D| \log^{3+\epsilon} |D|)$ [BBEL].

Computing $H_D(x)$

Three algorithms

1. Complex analytic
2. p-adic
3. Chinese Remainder Theorem (CRT)

Comparison

Heuristically, all have complexity $O(|D| \log^{3+\epsilon} |D|)$ [BBEL].

Practically, the complex analytic method is much faster ($\approx 50x$)

Computing $H_D(x)$

Three algorithms

1. Complex analytic
2. *p*-adic
3. Chinese Remainder Theorem (CRT)

Comparison

Heuristically, all have complexity $O(|D| \log^{3+\epsilon} |D|)$ [BBEL].

Practically, the complex analytic method is much faster ($\approx 50x$)

…and it can use much smaller class polynomials ($\approx 30x$).

Constructing elliptic curves of known order

Using complex multiplication (CM method)

Given p and $t \neq 0$, let $D < 0$ be a discriminant satisfying

$$4p = t^2 - v^2 D.$$

We wish to find an elliptic curve $E/\mathbb{F}_p$ with $N = p + 1 \pm t$ points.

Hilbert class polynomials modulo p

Given a root j of $H_D(x)$ over $\mathbb{F}_p$, let $k = j/(1728 - j)$. The curve

$$y^2 = x^3 + 3kx + 2k$$

has trace $\pm t$ (twist to choose the sign).

Not all curves with trace $\pm t$ necessarily have $H_D(j) = 0$.

Hilbert class polynomials

The Hilbert class polynomial $H_D(x)$

$H_D(x) \in \mathbb{Z}[x]$ is the minimal polynomial of the j-invariant of the complex elliptic curve $\mathbb{C}/\mathcal{O}_D$, where $\mathcal{O}_D$ is the imaginary quadratic order with discriminant D.

$H_D(x)$ modulo a (totally) split prime p

The polynomial $H_D(x)$ splits completely over $\mathbb{F}_p$, and its roots are precisely the j-invariants of the elliptic curves E whose endomorphism ring is isomorphic to $\mathcal{O}_D$ ($\mathcal{O}_E = \mathcal{O}_D$).

Practical considerations

We need $|D|$ to be small

Any ordinary elliptic curve can, in principle, be constructed via the CM method. A random curve will have $|D| \approx p$.

We can only handle small $|D|$, say $|D| < 10^{10}$.

Why small $|D|$?

The polynomial $H_D(x)$ is *big*.
We typically need $O(|D| \log |D|)$ bits to represent $H_D(x)$.

If $|D| \approx p$ that might be a lot of bits...

$$H_D(x)$$

Visible
Universe

$\|D\|$	h	$h \lg B$	$\|D\|$	h	$h \lg B$
$10^6 + 3$	105	113KB	$10^6 + 20$	320	909KB
$10^7 + 3$	706	5MB	$10^7 + 4$	1648	26MB
$10^8 + 3$	1702	33MB	$10^8 + 20$	5056	240MB
$10^9 + 3$	3680	184MB	$10^9 + 20$	12672	2GB
$10^{10} + 3$	10538	2GB	$10^{10} + 4$	40944	23GB
$10^{11} + 3$	31057	16GB	$10^{11} + 4$	150192	323GB
$10^{12} + 3$	124568	265GB	$10^{12} + 4$	569376	5TB
$10^{13} + 3$	497056	4TB	$10^{13} + 4$	2100400	71TB
$10^{14} + 3$	1425472	39TB	$10^{14} + 4$	4927264	446TB

Size estimates for $H_D(x)$

$$B = \binom{h}{\lfloor h/2 \rfloor} \exp\left(\pi\sqrt{|D|}\sum_{i=1}^{h}\frac{1}{a_i}\right)$$

More practical considerations

We don't want $|D|$ to be too small

Some security standards require $h(D) \geq 200$.
This is easily accomplished with $|D| \approx 10^6$.

Do we ever need to use larger values of $|D|$?

"Because we need to factor $H_D(x)$, it makes no sense to choose larger class numbers (than 5000) because $\deg(H_D) = h(D)$."

Handbook of Elliptic and Hyperelliptic Curve Cryptography.

Pairing-based cryptography

Pairing-friendly curves

The most desirable curves for pairing-based cryptography have near-prime order and embedding degree k between 6 and 24.

Choosing p and k

We should choose the size of $\mathbb{F}_p$ to balance the difficulty of the discrete logarithm problems in $E/\mathbb{F}_p$ and $\mathbb{F}_{p^k}$. For example

- 80-bit security: $k = 6$ and $170 < \lg p < 192$.
- 110-bit security: $k = 10$ and $220 < \lg p < 256$.

FST, "A taxonomy of pairing-friendly elliptic curves," 2006.

Such curves are very rare...

k	b_0	b_1	$L =$	10^6	10^7	10^8	10^9	10^{10}	10^{11}	10^{12}
6	170	192		0	0	1	11	33	149	493
10	220	256		0	0	0	0	8	29	81

Number of prime-order elliptic curves over $\mathbb{F}_p$ with $b_0 < \lg p < b_1$, embedding degree k, and $|D| < L$.

Karabina and Teske, "On prime-order elliptic curves with embedding degrees $k = 3$, 4, and 6," ANTS VIII (2008).

Freeman, "Constructing pairing-friendly elliptic curves with embedding degree 10," ANTS VII (2006).

Pairing-friendly curves

Bisson-Satoh construction

Given a pairing-friendly curve E with small discriminant D, find a pairing-friendly curve E' with larger discriminant $D' = n^2 D$, while preserving the values of ρ and k.

For example: $D = -3$, $\rho = 1$, and $k = 12$.

Requires large $|D'|$

To make it impractical to compute an isogeny from E' to E, we want prime $n > 10^5$, yielding $|D'| > 10^{10}$.

Bisson and Satoh, "More discriminants with the Brezing-Weng method".

New results

Algorithm to compute $H_D(x) \bmod p$ based on [ALV+BBEL]

- Repairs a technical defect in the algorithm of [BBEL].
- Much better constant factors.
- Heuristic complexity $O(|D| \log^{2+\epsilon} |D|)$ for most D.
- Requires only $O(|D|^{1/2+\epsilon})$ space.
- Faster than the complex analytic method for large D.

Practical achievements

Records to date: $|D| > 10^{12}$ and $h(D) \approx 400,000$.
Constructed many pairing-friendly curves with $|D| > 10^{10}$.

See `http://math.mit.edu/~drew` for examples.

Plus, breaking news (joint work with Andreas Enge).

Basic CRT method (using split primes)

Step 1: Pick split primes

Find $p_1, \ldots, p_n$ of the form $4p_i = u^2 - v^2 D$ with $\prod p_i > B$.

Step 2: Compute $H_D(x)$ mod p_i

Determine the roots $j_1, \ldots, j_h$ of $H_D(x)$ over $\mathbb{F}_{p_i}$.
Compute $H_D(x) = \prod (x - j_k) \bmod p_i$.

Step 3: Apply CRT to compute $H_D(x)$

Compute $H_D(x)$ by applying the CRT to each coefficient.
Better, compute $H_D(x)$ mod P via the *explicit* CRT [MS 1990].

First proposed by Chao, Nakamura, Sobataka, and Tsujii (1998).
Agashe, Lauter, and Venkatesan (2004) suggested explicit CRT.

Running time of the CRT method

Time complexity

As originally proposed, Step 2 tests every element of $\mathbb{F}_p$ to see if it is the j-invariant of a curve with endomorphism ring $\mathcal{O}_D$.

The total complexity is then $\Omega(|D|^{3/2})$. This is not competitive.

Modified Step 2 [BBEL 2008]

Find a single root of $H_D(x)$ in $\mathbb{F}_p$, then enumerate conjugates via the action of $Cl(D)$, using an isogeny walk.

Improved time complexity

The complexity is now $O(|D|^{1+\epsilon})$. This is potentially competitive. However, preliminary results are disappointing.

Space required to compute $H_D(x) \bmod P$

Online version of the explicit CRT

Explicit CRT computes each coefficient c of $H_D(x) \bmod P$ as

$$c = \left(\sum a_i M_i c_i - rM\right) \bmod P$$

where r is the closest integer to $\sum a_i c_i / M_i$. The values a_i, M_i, and M are *the same* for each c.
We can forget c_i once we compute its terms in c and r.

Space required to compute $H_D(x)$ mod P

Online version of the explicit CRT

Explicit CRT computes each coefficient c of $H_D(x)$ mod P as

$$c = \left(\sum a_i M_i c_i - rM\right) \bmod P$$

where r is the closest integer to $\sum a_i c_i / M_i$. The values a_i, M_i, and M are *the same* for each c.
We can forget c_i once we compute its terms in c and r.

Space complexity

The total space is then $O(|D|^{1/2+\epsilon} \log P)$.

This is interesting, but only if the time can be improved.

See Bernstein for more details on the explicit CRT.

CRT algorithm (split primes)

Given a fundamental discriminant $D < -4$ and a prime P with $4P = t^2 - v^2 D$, determine $j(E)$ for all $E/\mathbb{F}_P$ with $\mathcal{O}_E = \mathcal{O}_D$:

1. Compute the norm-minimal rep. S of $Cl(D)$ and $b = \lg B$.
   Pick split primes $p_1, \ldots, p_n$ with $\sum \lg p_i > b + 1$.
   Perform CRT precomputation.
2. Repeat for each p_i:
   a. Find $E/\mathbb{F}_{p_i}$ such that $\mathcal{O}_E = \mathcal{O}_D$.
   b. Compute the orbit $j_1, \ldots, j_h$ of $j(E)$ under $\langle S \rangle$.
   c. Compute $H_D(x) = \prod (x - j_k) \bmod p_i$.
   d. Update CRT sums for each coefficient of $H_D(x) \bmod p_i$.
3. Perform CRT postcomputation to obtain $H_D(x) \bmod P$.
4. Find a root of $H_D(x) \bmod P$ and compute its orbit.

Under GRH: Step 2 is repeated $n = O(|D|^{1/2} \log \log |D|)$ times and every step has complexity $O(|D|^{1/2+\epsilon})$ (assume $\log P = O(\log |D|)$).

Step 2a: Finding a curve with trace $\pm t$

First test

Find E and a random $\alpha \in E$ for which $(p + 1 \pm t)\alpha = 0$.

1. If both signs of t are possible, test whether $(p + 1)\alpha$ and $t\alpha$ have the same x coordinate [BBEL].
2. Don't test random curves. Search a parameterized family [Kubert] with suitable torsion (up to 15x faster).
3. Multiply in parallel using affine coordinates.

Second test

Apply a generic algorithm to compute the group exponent of E (or its twist) using an expected $O(\log^{1+\epsilon} p)$ group operations. For $p > 229$ this determines $\#E$.

Step 2a: Finding a curve with $\mathcal{O}_E = \mathcal{O}_D$

Which curves over $\mathbb{F}_p$ have trace $\pm t$?

There are $H(4p - t^2) = H(-v^2 D)$ distinct j-invariants of curves with trace $\pm t$ over $\mathbb{F}_p$ [Duering]. For $D < -4$ we have

$$H(-v^2 D) = \sum_{u|v} h(u^2 D).$$

The term $h(u^2 D)$ counts curves with $D(\mathcal{O}_E) = u^2 D$.

What does this tell us?

If $v = 1$ then E has trace $\pm t$ if and only if $\mathcal{O}_E = \mathcal{O}_D$ (easy).
If $v > 1$ then we have $H(4p - t^2) > h(D)$ (harder).

This is a good thing!

Step 1: Pick your primes with care

The problem

There are only $h(D)$ curves over $\mathbb{F}_p$ with $\mathcal{O}_E = \mathcal{O}_D$.
As p grows, they get harder and harder to find: $O(p/h(D))$.
Especially when $h(D)$ is *small*.

The solution [BBEL]

Use a curve with trace $\pm t$ to find a curve with $\mathcal{O}_E = \mathcal{O}_D$ by climbing isogeny volcanoes.

Improvement

We should pick our primes based on the ratio $p/H(4p - t^2)$.
We want $p/H(4p - t^2) \ll 2\sqrt{p}$. Easy to do when $h(D)$ is big.

Step 2a: Finding a curve with $\mathcal{O}_E = \mathcal{O}_D$

Classical modular polynomials $\Phi_\ell(X, Y)$

There is an ℓ-isogeny between E and E' iff $\Phi_\ell\,(j(E), j(E')) = 0$.
To find ℓ-isogenies from E, factor $\Phi_\ell\,(X, j(E))$.

Isogeny volcanoes [Kohel 1996, Fouquet-Morain 2002]

The isogenies of degree ℓ among curves with trace $\pm t$ form a directed graph consisting of a cycle (the surface) with trees of height k rooted at each surface node ($\ell^k \| v$).

For surface nodes, ℓ^2 does not divide $D(\mathcal{O}_E)$.

How to find a curve with $\mathcal{O}_E = \mathcal{O}_D$

Starting from a curve with trace $\pm t$, climb to the surface of every ℓ-volcano for $\ell | v$.

Step 2b: Computing the orbit of $j(E)$

The group action of $Cl(D)$ on $j(E)$

An ideal α in $\mathcal{O}_E \cong End_{\mathbb{C}}(E)$ defines an ℓ-isogeny

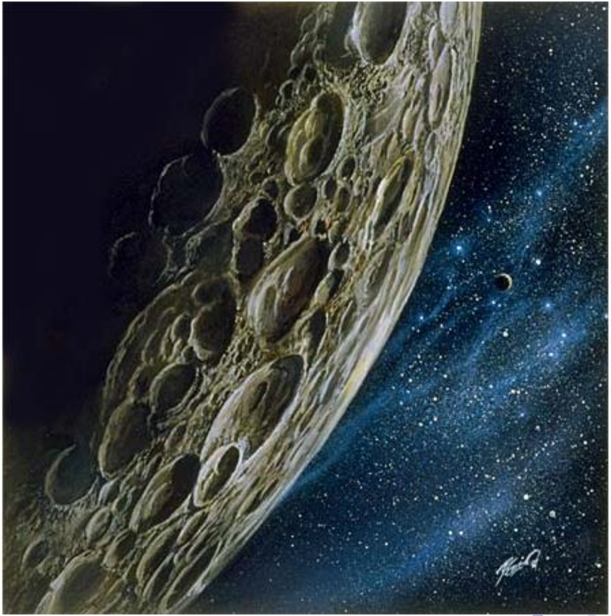
$$E \to E/E[\alpha] = E',$$

with $\mathcal{O}_{E'} = \mathcal{O}_E$ and $\ell = N(\alpha)$. This gives an action on the set $\{j(E) : \mathcal{O}_E = \mathcal{O}_D\}$ which factors through $Cl(D)$ and reduces mod p for split primes (**but ℓ depends on α**).

Touring the rim

We compute this action explicitly by walking along the surface of the volcano of ℓ-isogenies. For $\ell \nmid v$, set $j_1 = j(E)$, pick a root j_2 of $\Phi(X, j_1)$, then let j_{k+1} be the root of $\Phi(X, j_k)/(x - j_{k-1})$.

We can handle $\ell | v$, but this is efficient only for very small ℓ.



Step 2b: Computing the orbit of $j(E)$

Walking the entire orbit

Given a basis $\alpha_s, \ldots, \alpha_1$ for $Cl(D) = \langle\alpha_s\rangle \times \cdots \times \langle\alpha_1\rangle$, we compute the orbit of $j = j(E)$ by computing $\beta(j)$ for every $\beta = \alpha_k^{e_k} \cdots \alpha_1^{e_1}$ with $0 \le e_i < |\alpha_i|$ in a lexicographic ordering of $(e_k, \ldots, e_1)$ (one isogeny per step).

Complexity

Each step involves $O(\ell_i^2)$ operations in $\mathbb{F}_p$, where $\ell_i = N(\alpha_i)$. We need the ℓ_i to be small.

But this may not be possible using a basis!

Representation by a sequence of generators

Cyclic composition series

Let $\alpha_1, \ldots, \alpha_s$ generate a finite group G and suppose

$$G = \langle \alpha_1, \ldots, \alpha_s \rangle \longrightarrow \langle \alpha_1, \ldots, \alpha_{s-1} \rangle \longrightarrow \ldots \longrightarrow \langle \alpha_1 \rangle \longrightarrow 1$$

is a cyclic composition series. Let $n_1 = |\alpha_1|$ and define

$$n_i = |\langle \alpha_1, \ldots, \alpha_i \rangle| / |\langle \alpha_1, \ldots, \alpha_{i-1} \rangle|.$$

Each n_i divides (but need not equal) $|\alpha_i|$, and $\prod n_i = |G|$.

Unique representation

Every $\beta \in G$ can be written uniquely as $\beta = \alpha_1^{e_1} \cdots \alpha_s^{e_s}$, with $0 \le e_i < n_i$ (we may omit α_i for which $n_i = 1$).

Step 1: The norm-minimal representation of $Cl(D)$

Generators for $Cl(D)$

Represent $Cl(D)$ with reduced binary quadratic forms $(ax^2 + bxy + cy^2)$. The reduced primeforms of discriminant D generate $Cl(D)$ ($a \le \sqrt{|D|/3}$ or $a \le 6\log^2|D|$ under GRH).

Norm-minimal representation

Let $\alpha_1, \ldots, \alpha_s$ be the sequence of primeforms of discriminant D ordered by a and define $n_1, \ldots, n_s$ as above. The subsequence of α_i with $n_i > 1$ is the norm-minimal representation of $Cl(D)$.

Computing the n_i

We can compute the n_i using either $O(|G|)$ or $O(|G|^{1/2+\epsilon}|S|)$ group operations with a generic group algorithm.

Step 2c: Computing $H_D(x) = \prod(x - j_k) \bmod p_i$

Building a polynomial from its roots

Standard problem with a simple solution: build a product tree. Using *FFT*, complexity is $O(h \log^2 h)$ operations in $\mathbb{F}_{p_i}$.

Harvey's experimental znpoly library

Fast polynomial multiplication in $\mathbb{Z}/n\mathbb{Z}$ for $n < 2^{64}$, via multipoint Kronecker substitution. Two to three times faster than NTL for polynomials of degree 10^3 to 10^6.

http://cims.nyu.edu/~harvey/

CRT algorithm (split primes)

Given a fundamental discriminant $D < -4$ and a prime P with $4P = t^2 - v^2 D$, determine $j(E)$ for all $E/\mathbb{F}_P$ with $\mathcal{O}_E = \mathcal{O}_D$:

1. Compute the norm-minimal rep. S of $Cl(D)$ and $b = \lg B$.
   Pick split primes $p_1, \ldots, p_n$ with $\sum \lg p_i > b + 1$.
   Perform CRT precomputation.
2. Repeat for each p_i:
   a. Find $E/\mathbb{F}_{p_i}$ such that $\mathcal{O}_E = \mathcal{O}_D$.
   b. Compute the orbit $j_1, \ldots, j_h$ of $j(E)$ under $\langle S \rangle$.
   c. Compute $H_D(x) = \prod (x - j_k) \bmod p_i$.
   d. Update CRT sums for each coefficient of $H_D(x) \bmod p_i$.
3. Perform CRT postcomputation to obtain $H_D(x) \bmod P$.
4. Find a root of $H_D(x) \bmod P$ and compute its orbit.

Under GRH: Step 2 is repeated $n = O(|D|^{1/2} \log\log |D|)$ times and every step has complexity $O(|D|^{1/2+\epsilon})$ (assume $\log P = O(\log |D|)$).

A back-of-the-envelope complexity discussion

Some useful facts and heuristics

1. $h(D) \approx 0.28|D|^{1/2}$ on average.
2. $\max p_i = O(|D| \log^{1+\epsilon} |D|)$ heuristically ($p_i \ll 2^{64}$).
3. $\max \ell = O(\log^{1+\epsilon} |D|)$ conjecturally, and for most D, $\max \ell = O(\log \log |D|)$ heuristically.

Which step is asymptotically dominant?

If $\mathbb{F}_{p_i}$ adds/mults cost $O(1)$, for most D we expect:

1. Step 2a has complexity $O(|D|^{1/2} \log^{1.5+\epsilon} |D|)$.
2. Step 2b has complexity $O(|D|^{1/2} \log^{1+\epsilon} |D|)$.
3. Step 2c has complexity $O(|D|^{1/2} \log^{2+\epsilon} |D|)$.

For exceptionally bad D, Step 2b is $\Omega(|D|^{1/2} \log^{2} |D|)$.

Summary

Key improvements to [BBEL]

- $O(|D|^{1/2+\epsilon})$ space via online explicit CRT.
- Pick primes and curves carefully!
- Don't be afraid to climb volcanoes.
- Norm-minimal representation of $Cl(D)$.

Key constant factors

- Elliptic curve arithmetic.
- Finding roots of small polynomials.
- Building large polynomials from roots.

$-D$	$12,901,800,539$	$13,977,210,083$	$17,237,858,107$
$h(D)$	54,706	20,944	14,064
$\lceil \lg B \rceil$	5,597,125	2,520,162	1,737,687
ℓ_1	3	3	11
ℓ_2	5		23
$Cl(D)$ time	0.1	0.3	0.2
n	144,301	70,403	50,098
$\lceil \lg(\max p_i) \rceil$	41	38	38
prime time	3.4	1.5	1.0
CRT pre time	2.8	0.9	0.6
CRT post time	0.9	0.9	0.6
(a,b,c) splits	**(61,17,22)**	**(82,8,10)**	**(54,44,2)**
Step 2 time	**98,000**	**34,700**	**59,400**
root time	347	171	67
roots time	220	132	130

CRT method computing H_D mod P (MNT curves, $k = 6$)

(2.8GHz AMD Athlon CPU times in seconds)

$-D$	$h(D)$	ℓ	$\lceil \lg B \rceil$	time	split
28, 894, 627	724	7	66k	57	(64,35,1)
116, 799, 691	2,112	5	196k	309	(64,32,4)
228, 099, 523	1,296	17	143k	1,300	(32,67,0)
615, 602, 347	5,509	7	514k	2,540	(49,47,4)
1, 218, 951, 379	6,320	5	659k	3,270	(66,29,5)
2, 302, 080, 411	10,152	3/5	1.0m	8,200	(69,25,7)
4, 508, 791, 627	7,867	11	0.9m	16,400	(53,46,1)
9, 177, 974, 187	16,600	3/11	1.8m	46,400	(55,40,5)
17, 237, 858, 107	14,064	11	1.7m	62,900	(57,41,2)
35, 586, 455, 227	18,481	19	2.3m	232,000	(32,67,1)
69, 623, 892, 083	56,760	3	6.8m	212,000	(79,9,12)
137, 472, 195, 531	129,520	3/5	15m	1,170,000	(57,30,12)
275, 022, 600, 899	247,002	3	27m	2,400,000	(58,16,26)
553, 555, 955, 779	122,992	5	16m	1,890,000	(68,24,8)
1, 006, 819, 828, 491	180,616	3	25m	4,430,000	(71,18,11)

CRT method computing H_D mod P (MNT curves, $k = 6$)

(2.8 GHz AMD Athlon CPU seconds)

$-D$	$-D/200,000$	time
$28,894,627$	140	57
$116,799,691$	580	309
$228,099,523$	1,100	1,300
$615,602,347$	3,100	2,540
$1,218,951,379$	6,100	3,270
$2,302,080,411$	11,500	8,200
$4,508,791,627$	22,500	16,400
$9,177,974,187$	45,900	46,400
$17,237,858,107$	86,200	62,900
$35,586,455,227$	178,000	232,000
$69,623,892,083$	348,000	212,000
$137,472,195,531$	687,000	1,170,000
$275,022,600,899$	1,380,000	2,400,000
$553,555,955,779$	2,770,000	1,890,000
$1,006,819,828,491$	5,040,000	4,430,000

CRT method computing H_D mod P (MNT curves, $k = 6$)

(2.8 GHz AMD Athlon CPU seconds)

Scalability

Distributed computation

Large tests were run on 14 PCs in parallel (2 cores each). Elapsed times:

- $D = -1,006,819,828,491$, $h(D) = 181,616$ **1.8 days**
- $D = -905,270,581,331$, $h(D) = 391,652$ **1.1 days***

Minimal space requirements

Largest test used less than 300MB memory (per core). Total disk storage under 1GB.

Plenty of headroom

For $|D|$ in the range 10^8 to 10^{12} the observed running time is essentially linear in $|D|$. Larger computations are feasible.

		Complex Analytic		CRT Method		
$-D$	$h(D)$	bits	time	bits	time	**ratio**
6961631	5000	9.5k	28	269k	190	**0.15**
23512271	10000	20k	210	573k	840	**0.25**
98016239	20000	45k	1,800	1.3m	4,200	**0.43**
357116231	40000	97k	14,000	2.7m	20,000	**0.70**
2093236031	100000	265k	260,000	7.4m	140,000	**1.86**

Complex Analytic (double η quotient) vs.
CRT method (j)

(2.4 GHz AMD Opteron CPU seconds)

Enge, "The complexity of class polynomial computations via floating point approximations" (2008)

What about other class invariants?

Theoretical obstructions [BBEL]

In general, one cannot uniquely determine class invariants other than j over $\mathbb{F}_p$.

What about other class invariants?

Theoretical obstructions [BBEL]

In general, one cannot uniquely determine class invariants other than j over $\mathbb{F}_p$.

Breaking news (joint with Andreas Enge)

The CRT method *can* use other class invariants in many cases. For example:

- If D is not divisible by 3, we achieve a 3x improvement using the invariant γ_2.
- If D is also congruent to 1 mod 8, we achieve up to a 9x improvement using the invariant f^8.

This is work in progress, further improvements are expected. Ideally, we would use f whenever possible (potential 24x).

Alternative class invariants with the CRT method

The class invariants: f, j, and γ_2 [Weber]

Define the complex function $f(z)$ by

$$f(z) = e^{-\pi i/24}\frac{\eta((z+1)/2)}{\eta(z)}$$

where $\eta(z)$ is the Dedekind η-function. We then have

$$j(z) = \frac{(f^{24}(z)-16)^3}{f^{24}(z)}; \qquad \gamma_2(z) = \frac{f^{24}(z)-16}{f^8(z)}.$$

Note that $j = (\gamma_2)^3$.

Alternative class invariants with the CRT method

Modified CRT method using γ_2

Provided that D is not divisible by 3:

- Reduce height estimate by a factor of 3.
- Restrict to $p_i \equiv 2 \bmod 3$ so that cube roots are unique.
- Compute $\gamma_2 = \sqrt[3]{j}$ for each j enumerated in Step 2b.
- Form $W_{\gamma_2}(x) = \prod(x - \gamma_2)$ instead of $H_D(x)$ in Step 2c.
- Cube a root of $W_{\gamma_2}(x)$ mod P to get desired j at the end.

Further Improvement

Using suitable modular polynomials, enumerate γ_2 values directly rather than taking the cube root of each j.

$-D$	$12,901,800,539$	$13,977,210,083$	$17,237,858,107$
$h(D)$	54,706	20,944	14,064
ℓ_1	3	3	11
ℓ_2	5		23
$\lceil \lg B \rceil$	5,597,125	2,520,162	1,737,687
n	144,301	70,403	50,098
(a,b,c) splits	(61,17,22)	(82,8,10)	(54,44,2)
Step 2 time	98,000	34,700	59,400
$\lceil \lg B \rceil$	**1,814,367**	**883,076**	**574,545**
n	**49,122**	**24,279**	**17,196**
(a,b,c) splits	**(59,13,28)**	**(78,7,14)**	**(55,43,2)**
Step 2 time	**28,400**	**9,100**	**20,400**

CRT method j vs. γ_2 (MNT curves, $k = 6$)

(2.8GHz AMD Athlon CPU times in seconds)

$-D$	$h(D)$	time (j)	**time** (γ_2)
$28,894,627$	724	57	**21**
$116,799,691$	2,112	309	**94**
$228,099,523$	1,296	1300	**404**
$615,602,347$	5,509	2,540	**895**
$1,218,951,379$	6,320	3,270	**1,000**
$4,508,791,627$	7,867	16,400	**5,400**
$17,237,858,107$	14,064	62,900	**20,400**
$35,586,455,227$	18,481	232,000	**74,600**
$69,623,892,083$	56,760	212,000	**55,600**
$275,022,600,899$	247,002	2,400,000	**690,000**
$553,555,955,779$	122,992	1,890,000	**480,000**
$905,270,581,331$	391,652	7,860,000	**2,200,000**

CRT method j vs. γ_2 (MNT curves, $k = 6$)

(2.8 GHz AMD Athlon CPU seconds)

		Complex Analytic		CRT Method		
$-D$	$h(D)$	bits	time	bits	time	**ratio**
6961631	5000	9.5k	28	30k	34	**0.82**
23512271	10000	20k	210	64k	150	**1.4**
98016239	20000	45k	1,800	141k	710	**2.5**
357116231	40000	97k	14,000	302k	3,200	**4.4**
2093236031	100000	265k	260,000	827k	22,000	**12**

Complex Analytic (double η quotient) vs.
CRT method (f^8)

(2.4 GHz AMD Opteron CPU seconds)

Areas for future work

To do list

- Continue to improve constant factors.
- Expand and refine the use of other class invariants.
- Post more pairing-friendly curves at

  `http://math.mit.edu/~drew`

  Requests welcome.
- Source code will be available under GPL.

Open question

Is there an $O(p^{1/2+\epsilon})$ algorithm to compute $H_D(x) \bmod p$ for an inert prime p?