Subexponential Performance from Generic Group Algorithms

Andrew V. Sutherland

Massachusetts Institute of Technology

April 3, 2008

A familiar example

Binary exponentiation

Given $\alpha \in G$ and $k \in \mathbb{Z}$, $Exp(\alpha, k)$ computes α^k:

1. If $k = 0$ return 1_G.
2. If $k < 0$ return $Exp(\alpha^{-1}, -k)$.
3. Set $\beta \leftarrow Exp(\alpha, \lfloor k/2 \rfloor)$.
4. If k is even return $\beta\beta$, otherwise return $\beta\beta\alpha$.

Generic groups

Computational model for finite groups

- Black-box group operation, inverse, and identity.
- Elements uniquely identified by (opaque) bit strings.
- Uniformly distributed random group elements available.
- Complexity measured by group operations.

Discrete logarithms in a generic group

Easy problem

Given $\alpha \in G$ and $1 \le k \le N = |\alpha|$, compute

$$\beta = \alpha^k.$$

Uses $O(\log N)$ group operations, outputs $\beta \in \langle\alpha\rangle$.

Hard problem

Given $\beta \in \langle\alpha\rangle$, compute the least $k > 0$ such that $\alpha^k = \beta$,

$$k = \log_\alpha \beta.$$

Requires $\Omega(\sqrt{P})$ group operations (Shoup 1997).

DL-based cryptography

Cryptographic requirements

P should be at least 2^{160}, preferably 2^{200} or more.
Ideally, $N = P$ or $N = cP$ for some small cofactor c.

Pohlig-Hellman attack

Suppose $|\alpha| = N = 2 \cdot 3 \cdot 5 \cdots 997 > 2^{1000}$. Let $m = N/\ell$.
To compute $k = \log_\alpha \beta$, modulo ℓ, note that

$$\beta^m = (\alpha^k)^m = (\alpha^m)^k.$$

Therefore $k \equiv \log_{\alpha^m} \beta^m \bmod \ell$.

Hyperelliptic curves

Quick primer

- Projective curve C given by $y^2 = f(x)$ over $\mathbb{F}_p$ (odd char.). $\deg f(x) = 2g + 1$, where g is the genus of C.
- The case $g = 1$ is an elliptic curve: $y^2 = x^3 + ax + b$.
- The Jacobian $J(C)$ is a finite abelian group.
- An element of $J(C)$ corresponds to g points on C. For $g = 1$, we have $J(C) \cong C$.
- $\#J(C) \sim p^g$, specifically $|\#J(C) - p^g| = O(p^{g-1/2})$.

Hyperelliptic curve cryptography

Advantages

- As fast or faster than elliptic curves.
- Small key size, and even smaller field size.
- Well suited to embedded applications: mobile phones, PDAs, stored value cards, secure ID, remote keys, ...

Implementation issues

- Security (and performance) dictate $g = 2$ or 3.
- Field size should be a power of 2, or prime.
  $p = 2^{89} - 1$ and $p = 2^{61} - 1$ work nicely.
- Group order $\#J(C)$ should be prime or near-prime.

The problem

How do we compute $\#J(C)$?

For binary fields, use *p*-adic methods, e.g. Kedlaya's algorithm.
For prime fields, **no good solution is known**.

- A polynomial-time algorithm exists, but is infeasible in practice (Pila 1990).
- Best results in genus 2 take one week to compute $\#J(C) \approx 2^{164}$ (Gaudry and Schost 2004).
- In genus 3, best is $\#J(C) \approx 2^{150}$ (Harvey 2007).
- Both existing methods limited by (effectively) exponential space requirements. Difficult to scale.

How do we compute $|G|$ for abelian G?

The group exponent $\lambda(G)$

$\lambda(G)$ is the least common multiple of $|\alpha|$ over $\alpha \in G$.
A prime p divides $|G|$ if and only if p divides $\lambda(G)$.

Computing the structure of G

Decompose G as a product of cyclic groups:

1. Compute $|\alpha|$ for random $\alpha \in G$ to obtain $\lambda(G)$.
2. Using $\lambda(G)$ to compute in p-Sylow subgroups H_p, compute a basis for each H_p via discrete logarithms.

Given bounds on $|G|$, the expected complexity is dominated by the time to compute the first $|\alpha|$ (PhD thesis, 2007).

How do we compute $|\alpha|$?

Generic method 1: Shanks' baby-step giant-step

Pick B, compute $\beta = \alpha^{-B}$, and then

$$\alpha, \alpha^2, \alpha^3, \ldots, \alpha^B; \qquad \beta, \beta^2, \beta^3, \ldots, \beta^B.$$

Provided $|\alpha| <= B^2$, then some $\alpha^j = \beta^k$ and $\alpha^{j+kB} = 1_G$.

$O(\sqrt{N})$ group operations* (slow!).

Generic method 2: Pollard's $p-1$ method

Let M be the product of all maximal prime powers $q < B$.
If $\alpha^M = 1_G$, then we can compute $|\alpha|$ in $O(B)$ time.

$O(N)$ group operations in the worst case (slower!).

Smooth and semismooth probabilities

Method 2 is fast if $|\alpha|$ is "B-smooth"

Suppose $|\alpha|$ is a random integer in $[1, N]$ and let $B = N^{1/u}$.
With probability $\rho(u) = u^{-u+o(1)}$, compute $|\alpha|$ in time $O(B)$.

Pick $u = \sqrt{2 \log N / \log \log N}$ to obtain $L(1/2, 1/\sqrt{2})$.

Method 2+1 is fast if $|\alpha|$ is "(B^2, B)-semismooth"

With probability $\sigma(u)$, compute $|\alpha|$ in time $O(B)$.
$\sigma(u) = O(\rho(u))$, but about 100 times bigger for moderate u.

A probabilistic paradox

Good news

Let $N \approx 2^{160}$ and $u = 6.4$.
Then $\sigma(u) \approx 1/2640$ and $O(B) \approx 71$ million gops.
For hyperelliptic curves, this is under thirty seconds.

Bad news

Worst case: over one trillion years.
Random case: over ten billion years.

Optimistic/pessimistic strategy

Give up after $O(B)$ gops and try a different C.
Expected time to first success is less than a day.

What good is it?

Apparently useless

The only cases where we can compute $\#J(C)$ are totally unsuitable for cryptographic use.

The group order contains no large prime factors.

But with a slight twist...

$\tilde{C}$ is the curve $y^2 = \tau f(x)$, where τ is not square in $\mathbb{F}_p$.
$\#J(\tilde{C})$ may be prime even though $\#J(C)$ is smooth.

The zeta function of a curve

The zeta function of C over $\mathbb{F}_p$ is given by

$$Z(T) = \exp\left(\sum \frac{N_k}{k} T^k\right) = \frac{P(T)}{(1-T)(1-pT)},$$

where N_k counts points on C over $\mathbb{F}_{p^k}$. In genus 2,

$$P(T) = p^2T^4 + pa_1T^3 + a_2T^2 + a_1T + 1.$$

The polynomial $P(T)$ has the useful property that

$$P(1) = \#J(C); \qquad P(-1) = \#J(\tilde{C}).$$

Using known bounds on the integers a_1 and a_2, we can deduce $P(T)$ from $\#J(C)$ using group operations in $J(\tilde{C})$.

The algorithm

Finding a cryptographically suitable Jacobian

Given N, select a suitable u, and let $B = N^{1/u}$.
For each curve C in a family with $\#J(C) \approx N$:

1. Attempt to compute $\#J(\tilde{C})$ using $O(B)$ gops.
2. When successful, determine $P(T)$ from $P(-1) = \#J(\tilde{C})$. Then compute $\#J(C) = P(1)$.
3. Continue until $\#J(C)$ is prime or near-prime.

Selecting a suitable u

Computing $\sigma(u)$.

The semismooth probability function $G(\alpha, \beta)$ may be used to determine $\sigma(u) = G(1/u, 2/u)$ (Bach-Peralta 1996).

$N = \#J(C)$ is more likely to have small factors than $N \in \mathbb{Z}$.

Let C be a "typical" curve of genus 2 over $\mathbb{F}_p$. If $N = \#J(C)$

$$\Pr[\ell | N] = \frac{1}{\ell} + \frac{1}{\ell^2} + O\left(1/\ell^3\right)$$

for primes $\ell \ll p$ (Achter-Holden 2003).

Performance

Complexity

Expected time is $L(1/2, \sqrt{2})$ group operations.
Space complexity is $L(1/2, 1/\sqrt{2})$, not a limiting factor.

Parellelization

Well suited to distributed computation:

1. Each attempt to compute $\#J(\tilde{C})$ is independent.
2. Minimal coordination required.
3. Fault tolerant.

Examples

Genus 2, $p = 2^{89} - 1$

$y^2 = x^5 + x + 202214$: $\quad \#J(C) =$

$$180 \times 21284660289802222651107604191879163807427101815332O3.$$

Group size is 178 bits, with a 171-bit prime factor.

Genus 3, $p = 2^{61} - 1$

$y^2 = x^7 + 3x^5 + x^4 + 4x^3 + x^2 + 5x + 84538$: $\quad \#J(C) =$

$$14739408 \times 831781325652289358544190241299568732364985371373.$$

Group size is 183 bits, with a 166-bit prime factor.

Trace zero varieties

Definition

Let ϕ denote the Frobenius endomorphism on $J(C/\mathbb{F}_{p^k})$.
The kernel of the trace endomorphism

$$1 + \phi + \phi^2 + \cdots + \phi^{k-1}$$

is a subgroup $T_k(C)$ of $J(C/\mathbb{F}_{p^k})$, the *trace zero variety*.

Provided k does not divide $\#J(C)$, we have

$$\#T_k(C) = \#J(C/\mathbb{F}_{p^k})/\#J(C) \approx p^{(k-1)g}.$$

Trace zero varieties

Implementation

- For cryptographic use, $k = 3$, $g = 2$, and $\#T_3(C) \approx p^4$.
- Must be 20% larger to achieve equivalent security. This still permits much smaller p.
- Performance is often superior to a comparable Jacobian.

Results

- The time required to find $T_3(C)$ with security equivalent to a 200-bit Jacobian is under an hour.
- Several examples have effective security over 300 bits.
- The algorithm can readily find C for which $\#T_3(C)$ and $\#T_3(\tilde{C})$ are both prime.

The bigger picture

Generic subexponential algorithm

Any problem reducible to computing the order of one of a family of generic abelian groups can be solved in subexponential time.

*assuming suitably distributed group orders.

A generic approach to searching for Jacobians, to appear in Math. Comp.

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