

Elliptic vs. hyperelliptic, part 1

D. J. Bernstein



Elliptic

vs.

Hyper-
elliptic



Part I

Goal: Protect all Internet packets against forgery, eavesdropping.

We aren't anywhere near the goal. Most Internet packets have little or no protection.

Why not deploy cryptography?

Why `http://www.google.com`, not `https://www.google.com`?

Common answer: Cryptography takes too much CPU time.

Obvious response, maybe enough: Faster cryptography!

Streamlining protocols

Often quite easy to save time in cryptographic protocols by recognizing and eliminating wasteful cryptographic structures.

Example #1 of waste:

Sender feeds a message through “public-key encryption” and then “public-key signing.”

Improvement: “Signcryption.”

No need to partition into encryption and signing; combined algorithms are faster.

Example #2: Sender signcrypts two messages for same receiver.

Improvement: Signcrypt one key and use secret-key cryptography to protect both messages.

Example #3: Sender signcrypts randomly generated secret key.

Improvement: Diffie-Hellman, generating unique shared secret for each pair of public keys.

Obtain randomness of secret from randomness of public keys.

No need for extra randomness.

Streamlined structure to protect private communication:

Alice has secret key a ,
long-term public key $G(a)$.

Alice, Bob have long-term
shared secret $G(ab)$.

Alice, Bob use shared secret
to encrypt and authenticate
any number of packets.

(Public communication has a
different streamlined structure.

This talk will focus on
private communication.)

How much does this cost?

Key generation: one evaluation of $a \mapsto G(a)$ for each user.

Shared secrets: one evaluation of $a, G(b) \mapsto G(ab)$ for each pair of communicating users.

Encryption and authentication:
secret-key operations
for each byte communicated.

This talk will focus on applications with many pairs of communicating users and with not much data communicated between each pair.

Bottleneck is $a, G(b) \mapsto G(ab)$.

How fast is this?

Answer depends on CPU, on choice of G , and on choice of method to compute G .

Many parameters.

Many interactions across levels.

Choices are not easy to analyze and optimize.

Elliptic vs. hyperelliptic

PKC 2006: Analyzed wide range of elliptic-curve functions G and methods of computing G .

Obtained new speed records for $a, G(b) \mapsto G(ab)$ on today's most common CPUs.

<http://cr.yp.to/ecdh.html>

The big questions for today:
Can we obtain higher speeds at comparable security levels using genus-2 hyperelliptic curves?
How fast is hyperelliptic-curve scalar multiplication?

Basic advantage of genus 2:
use much smaller field
for same conjectured security.

This talk will focus on a
comfortable security level:
 $> 2^{128}$ bit ops for known attacks.

PKC 2006 genus-1 records
used field size $2^{255} - 19$.
 $\approx 2^{255}$ points on curve.

Jacobian of genus-2 curve
over field of size $2^{127} - 1$
has $\approx 2^{254}$ points.

Much smaller field, so
much faster field mults.

Basic disadvantage of genus 2:
many more field mults.

PKC 2006 genus-1 records
used Montgomery-form curve

$$y^2 = x^3 + 486662x^2 + x,$$

$G(a) = X_0(aP)$, standard P .

10 mults per bit of a .

Culmination of extensive work
on eliminating field mults for
similar $G(a)$ defined by

genus-2 hyperelliptic curve:

25 mults per bit. (2005 Gaudry)

Does the advantage
outweigh the disadvantage?

Superficial analysis: Yes!

Half as many bits in field
means, uhhh, $4\times$ faster? $3\times$?

Anyway, $(3 \times 10)/25 = 1.2$.

That's a 20% gap!

Genus-2 field mults have
finally been reduced enough
to beat genus 1!

This analysis has several flaws.
Let's do a serious analysis.

What are the formulas?

Genus-1 setup: Field k , big char.
Specify elliptic curve $E \subset \mathbf{P}^2$ by
equation $y^2z = x^3 + a_2x^2z + xz^2$.
(Full moduli space *if* $k = \bar{k}$.)

Rational map $(x : y : z) \mapsto (x : z)$
induces $X : E / \{\pm 1\} \hookrightarrow \mathbf{P}^1$.

Analogous genus-2 setup:
Specify genus-2 curve C by
particular parametrization.

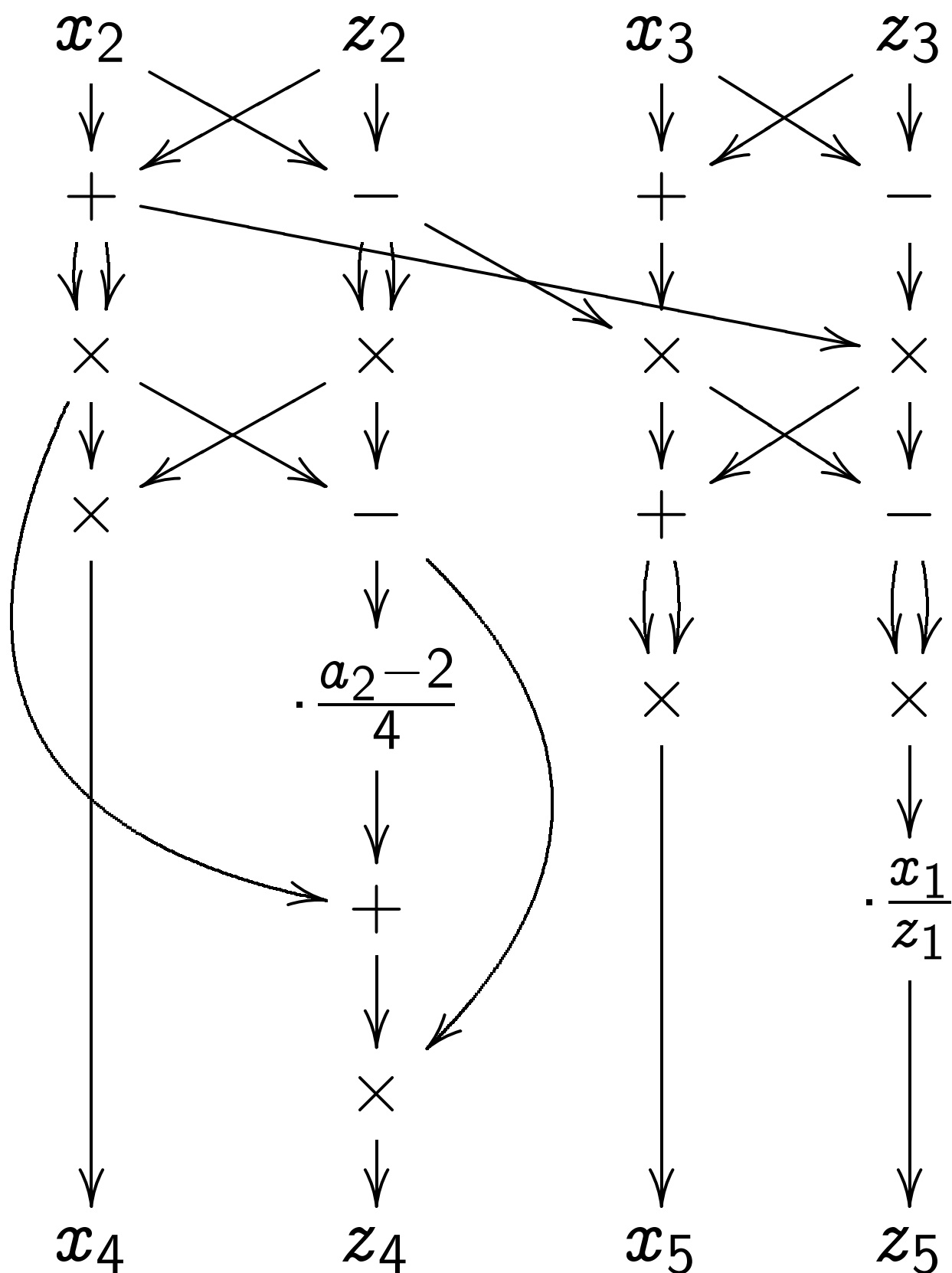
Build “Kummer surface” $K \subset \mathbf{P}^3$
and particular rational map
 $X : (\text{Jac } C) / \{\pm 1\} \hookrightarrow K$.

Recursively build rational functions F_1, F_2, \dots with $X(nQ) = F_n(X(Q))$ generically.

Recursion uses very fast rational functions $X(nQ) \mapsto X(2nQ)$ and $X(Q), X(nQ), X((n+1)Q) \mapsto X((2n+1)Q)$.

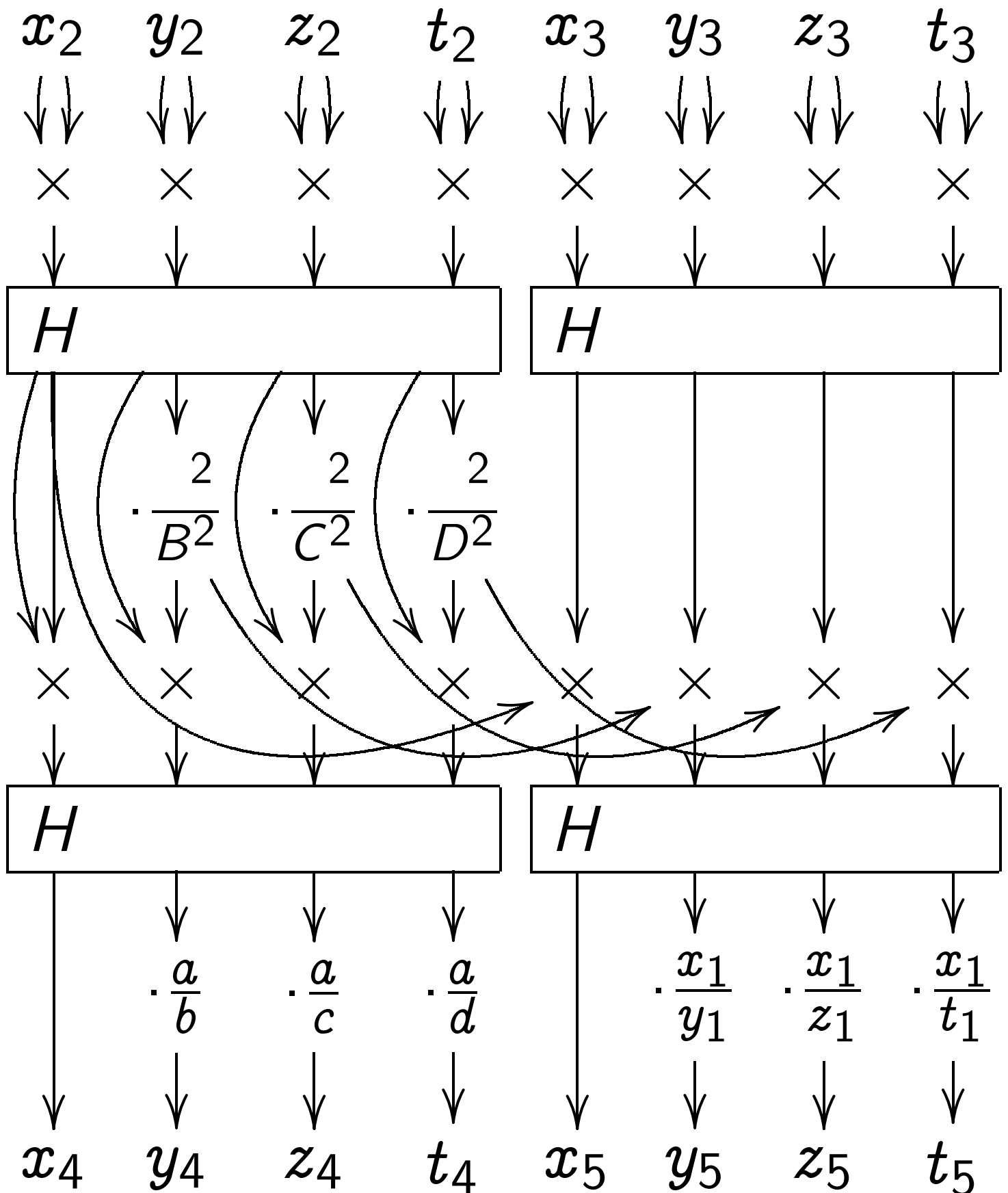
(genus 1: 1986 Chudnovsky, Chudnovsky; independently 1987 Montgomery; 10 mults: 1987 Montgomery; genus 2: 1986 Chudnovsky, Chudnovsky; 25 mults: 2005 Gaudry)

Montgomery's recursion for
 genus 1, $X(nQ) = (x_n : z_n)$:



Gaudry's recursion for genus 2,

$$X(nQ) = (x_n : y_n : z_n : t_n):$$



$$H(\alpha, \beta, \gamma, \delta) =$$

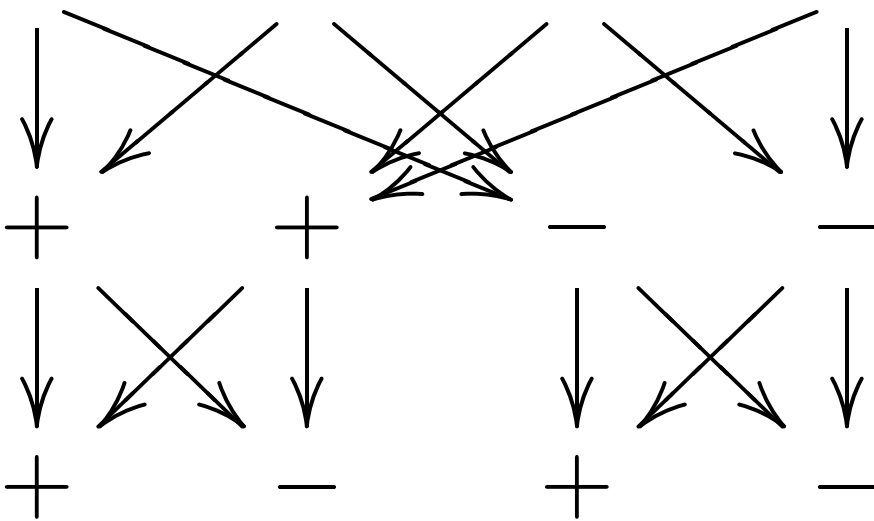
$$(\alpha + \beta + \gamma + \delta,$$

$$\alpha + \beta - \gamma - \delta,$$

$$\alpha - \beta + \gamma - \delta,$$

$$\alpha - \beta - \gamma + \delta).$$

Easy 8-addition chain
 (“fast Hadamard transform”):



Total Gaudry field operations:
 25 mults, 32 adds.

$X(nQ) = F_n(X(Q))$ generically:

“Generically” allows failures.

Maybe trouble for cryptography!

Can detect failures by
testing for zero at each step.

Can we avoid these tests?

For genus 1: Yes,

after replacing X by X_0 .

cr.yp.to/papers.html

#curvezero, Theorem 5.1.

Similar in genus 2?

Looks like painful calculations.

Let me know if you have

ideas for tackling this.

Curve specialization

Montgomery-form curves can be specialized to save time.

For $y^2 = x^3 + 486662x^2 + x$,

1 of the 10 mults is by 121665;
much faster than general mult.

Do Gaudry-form surfaces
allow similar specialization?

Gaudry: Out of 25 mults,

6 “are multiplications by
constants that depend only on the
surface . . . Therefore by choosing
an appropriate surface, a few
multiplications can be saved.”

What's "a few"?

Let's look at the formulas.

Gaudry has params $(a : b : c : d)$.

Also $(\quad : B : C : D)$ satisfying

$$H(\quad^2, B^2, C^2, D^2) = (a^2, b^2, c^2, d^2).$$

Gaudry's 6 mults are by

$$a/b, a/c, a/d,$$

$$(\quad/B)^2, (\quad/C)^2, (\quad/D)^2.$$

Can choose small B, C, D ,

$$\text{small } \quad \in B\mathbf{Z} \cap C\mathbf{Z} \cap D\mathbf{Z}.$$

Then solve for a, b, c, d .

Can scale formulas to have
multiplications by, e.g., $(BCD)^2$,
 $(CD)^2$, $(BD)^2$, $(BCD)^2$.

Choose any small a, B, C, D .

Can also hope for some of
 a, b, c, d to be small.

More flexibility:

Can choose small a^2, B^2, C^2, D^2 .

e.g. $a^2 = 21, B^2 = 16,$

$C^2 = 8, D^2 = 4, a = 7,$

$b = 5, c = 3, d = 1.$

Scale $1, a/b, a/c, a/d$

to $bcd, acd, abd, abc.$

Apparently “a few” is “all 6”!

Products with a/b , a/c , a/d will be squared before use.

Convenient to change K by squaring coordinates. (as in 1986 Chudnovsky, Chudnovsky)

In data-flow diagram, roll top squarings to bottom and through a, b, c, d layer.

No loss in speed.

(2006 André Augustyniak)

Thus have even more flexibility: small a^2, b^2, c^2, d^2 suffice.

Unfortunately,
these specialized surfaces
have a big security problem:
genus-2 point counting
is too slow to reach 256 bits.

Our only secure genus-2 curves
are from CM. How to locate
a secure *specialized* surface
over, e.g., $\mathbf{Z}/(2^{127} - 1)$?

Maybe can speed up
genus-2 point counting.

Inspiring news: speed records
for Schoof's original algorithm.
(2006 Nikki Pitcher)

Squarings and other operations

For Montgomery-form curves:

4 of the 9 big mults

are squarings;

faster than general mults.

For Gaudry-form surfaces:

9 squarings out of 25 mults.

$4S + 5M$ in big field

comparable to, uhhh,

$12S + 15M$ in small field?

$9S + 16M$ still slightly better,

but gap is only $\approx 5\%$,

depending on S/M ratio.

Gaudry understated benefit of specialized surfaces.

One of Gaudry's speedups:
compute $(a/b)u^2$, $(a/b)uv$
by first computing $(a/b)u$.

$3M$. Total: $9S + 16M$.

Specialized: $2M$.

Specialized total: $9S + 10M$.

Better when a/b is small:

simply undo this speedup.

$S + 3M$. Total: $12S + 16M$.

Specialized: $S + M$.

Specialized total: $12S + 7M$.

The $3\times$, $4\times$ myths

Why do some people say that half as many bits in field means $4\times$ speedup?

Answer: “ n -bit arithmetic takes time n^2 .”

Why do some people say that half as many bits in field means $3\times$ speedup?

Answer: “ n -bit arithmetic takes time $n^{\lg 3}$.”

Reality: Both n^2 and $n^{\lg 3}$ are horribly inaccurate models.

Field speed is CPU-dependent.

Today let's focus on

one common CPU: Pentium M.

Experience says:

Fastest Pentium M arithmetic
uses floating-point operations.

$$\#\{\text{fp ops}\} / \#\{\text{cycles}\} \leq 1;$$

optimized code always close to 1,
very little variation.

PKC 2006 speed records

$$\text{for } y^2 = x^3 + 486662x^2 + x$$

over $\mathbf{Z}/(2^{255} - 19)$:

640838 cycles; 92% fp ops.

Accurately (but not perfectly)
analyze cycles by counting fp ops.

e.g. $\mathbf{Z}/(2^{255} - 19)$ arithmetic
in PKC 2006 records:

10 fp ops for $f, \mapsto f + \dots$

55 fp ops for $f \mapsto 121665f$.

162 fp ops for $f \mapsto f^2$.

243 fp ops for $f, \mapsto f \dots$

Where do these numbers come
from? How do they scale?

Is $\mathbf{Z}/(2^{127} - 1)$ really $4\times$ faster?

Or at least $3\times$ faster?

Element of $\mathbf{Z}/(2^{255} - 19)$ is represented as 10-coeff poly.

Field add is poly add: 10 fp adds.
In context, can skip carries.

Field mult is poly mult
and reduction mod $2^{255} - 19$
and carrying:

10^2 fp mults for poly,

$(10 - 1)^2$ fp adds for poly,

$10 - 1$ fp mults for reduce,

$10 - 1$ fp adds for reduce,

$4 \cdot 10 + 4$ fp adds for carry.

Squaring: save $(10 - 1)^2$ ops.

Element of $\mathbf{Z}/(2^{127} - 1)$ is represented as 5-coeff poly.

Field add is 5 fp ops; $2\times$ faster.

Poly mult is $5^2 + (5 - 1)^2$

but reduce is $(5 - 1) + (5 - 1)$

and carry is $4 \cdot 5 + 4$.

73 fp ops; $3.329\times$ faster.

Squaring saves $(5 - 1)^2$ ops.

57 fp ops; $2.842\times$ faster.

Surprisingly small ratios,
even *without* Karatsuba.

Heavy optimization of mults
makes linear effects more visible.

Montgomery uses 8 adds,

1 mult by 121665,

4 squarings, 5 mults:

$$8 \cdot 10 + 1 \cdot 55 + 4 \cdot 162 + 5 \cdot 243 = 1998.$$

Gaudry uses 32 adds,

9 squarings, 16 mults:

$$32 \cdot 5 + 9 \cdot 57 + 16 \cdot 73 = 1841.$$

Gaudry loses in adds,

wins in squarings,

wins in other mults.

Specialized Gaudry: [1355, 1659]

depending on exact coeff size.

Far fewer than 1998 ops!

Reciprocals

What about divisions?

At end of computation,

$$(x : y : z : t) \mapsto (x/t, y/t, z/t)$$

for transmission.

Three multiplications and
one reciprocal in $\mathbf{Z}/(2^{127} - 1)$.

Montgomery needs division
in $\mathbf{Z}/(2^{255} - 19)$;

more than twice as slow.

Not big part of computation
but still a disadvantage.

Space disadvantage for Gaudry:
 ≈ 384 bits in $(x/t, y/t, z/t)$.

Standard 512-bit alternative:
blinding. Choose random r ,
send $(xr : yr : zr : tr)$.

Negligible computation cost.

Also negligible for Montgomery.

Standard 256-bit alternative:
point compression.

Transmit, e.g., $(x/t, y/t)$.

Then have to solve quartic.

Disadvantage for Gaudry.

Open: Compression method
allowing faster decompression?

Extra Gaudry division problem:
recall multiplications by
 $x_1/y_1, x_1/z_1, x_1/t_1$.

Even if we're given $t_1 = 1$,
have to divide by y_1, z_1 .

How to avoid extra division?

Can't merge with final division.

Scaling $(1 : x_1/y_1 : x_1/z_1 : x_1)$
is bad: extra mult for each bit.

Easy solution: Don't send

$(x/t, y/t, z/t)$. Instead send

$(t/x, t/y, t/z)$ or $(x/y, x/z, x/t)$.

Sender can merge divisions.

Software speed measurements

Using qhasm tools, wrote
Pentium M implementation
of scalar multiplication
(with no input-dependent
branches, indices, etc.)
on a Gaudry-form surface.

$n, P \mapsto nP$. Coords

$(x/y, x/z, x/t)$ for P, nP .

Arbitrary params a, b, c, d .

Recall the competition,
speed record from PKC 2006:
640838 cycles for genus 1.

Genus 2: 582363 cycles.

New Diffie-Hellman speed record!

Try the software yourself:

cr.yp.to/hecdh.html

Standardize genus-2 curve
for cryptography? Use CM
to generate secure a, b, c, d ?

I think that's premature.

Very small choices of a, b, c, d
will provide a big speedup.

Let's wait for point counting,
then standardize.

Halftime advertising, part 1

Part 1 was brought to you by . . .

“SPEED: Software
Performance Enhancement
for Encryption and Decryption”

A workshop on software speeds
for secret-key cryptography
and public-key cryptography.

Amsterdam, June 11–12, 2007

<http://>

www.hyperelliptic.org/SPEED