Curve25519: new Diffie-Hellman speed records

D. J. Bernstein

Thanks to:
University of Illinois at Chicago
Danmarks Tekniske Universitet
Alfred P. Sloan Foundation

Which public-key systems are smallest? Fastest?

Real-world cost measures: Pentium cycles, Athlon cycles, etc. for generating keys, signing, verifying, encrypting, decrypting; key bytes, signed-message bytes, ciphertext bytes, etc.

More useful than simplified cost measures, although harder to analyze.

an speed records

ois at Chicago
ske Universitet
Foundation

Which public-key systems
are smallest? Fastest?

Real-world cost measures:
Pentium cycles, Athlon cycles,
etc. for generating keys, signing,
verifying, encrypting, decrypting;
key bytes, signed-message bytes,
ciphertext bytes, etc.

More useful than
simplified cost measures,
although harder to analyze.

eBATS (ECRYPT
of Asymmetric S
new project to m
time and space c
public-key signat
public-key encryp
public-key secret-

http://ebats.

Which public-key systems are smallest? Fastest?

Real-world cost measures: Pentium cycles, Athlon cycles, etc. for generating keys, signing, verifying, encrypting, decrypting; key bytes, signed-message bytes, ciphertext bytes, etc.

More useful than simplified cost measures, although harder to analyze.

eBATS (ECRYPT Benchmarking of Asymmetric Systems): new project to measure time and space consumed by public-key signature systems, public-key encryption systems, public-key secret-sharing systems.

`http://ebats.cr.yp.to`

y systems
stest?

measures:
Athlon cycles,
ng keys, signing,
ting, decrypting;
-message bytes,
etc.

easures,
to analyze.

eBATS (ECRYPT Benchmarking
of Asymmetric Systems):
new project to measure
time and space consumed by
public-key signature systems,
public-key encryption systems,
public-key secret-sharing systems.

`http://ebats.cr.yp.to`

This talk's scope

Focus on private
ssh, email, purch

Typical setup:
Each communica
has a long-term s
and a long-term

Alice authenticat
encrypts message
using Alice's secr
and Bob's public
Bob verifies and
using Alice's pub
and Bob's secret

eBATS (ECRYPT Benchmarking of Asymmetric Systems): new project to measure time and space consumed by public-key signature systems, public-key encryption systems, public-key secret-sharing systems.

`http://ebats.cr.yp.to`

This talk's scope

Focus on private communications: ssh, email, purchasing, etc.

Typical setup:
Each communicating party has a long-term secret key and a long-term public key.

Alice authenticates and encrypts messages to Bob using Alice's secret key and Bob's public key.
Bob verifies and decrypts using Alice's public key and Bob's secret key.

T Benchmarking
ystems):
easure
onsumed by
ure systems,
ption systems,
-sharing systems.

cr.yp.to

This talk's scope

Focus on private communications: ssh, email, purchasing, etc.

Typical setup:
Each communicating party has a long-term secret key and a long-term public key.

Alice authenticates and encrypts messages to Bob using Alice's secret key and Bob's public key.
Bob verifies and decrypts using Alice's public key and Bob's secret key.

This talk's recom

The "asymmetri
Alice, Bob use C
compute long-te
from secret keys,
Note: minimal a

The "symmetric"
Alice, Bob use s
as key for Poly1
to authenticate+

Curve25519 is th
if there aren't m
This talk focuses

This talk's scope

Focus on private communications: ssh, email, purchasing, etc.

Typical setup:
Each communicating party has a long-term secret key and a long-term public key.

Alice authenticates and encrypts messages to Bob using Alice's secret key and Bob's public key.
Bob verifies and decrypts using Alice's public key and Bob's secret key.

This talk's recommendations

The "asymmetric" part:
Alice, Bob use Curve25519 to compute long-term shared secret from secret keys, public keys.
Note: minimal asymmetric usage!

The "symmetric" part:
Alice, Bob use shared secret as key for Poly1305+Salsa20 to authenticate+encrypt packets.

Curve25519 is the bottleneck *if* there aren't many packets.
This talk focuses on Curve25519.

communications:
asing, etc.

ting party
secret key
public key.

es and
es to Bob
ret key
key.
decrypts
lic key
key.

This talk's recommendations

The "asymmetric" part:
Alice, Bob use Curve25519 to compute long-term shared secret from secret keys, public keys.
Note: minimal asymmetric usage!

The "symmetric" part:
Alice, Bob use shared secret as key for Poly1305+Salsa20 to authenticate+encrypt packets.

Curve25519 is the bottleneck *if* there aren't many packets.
This talk focuses on Curve25519.

Curve25519 secre
Curve25519 publ
Time to compute
957904 Pentium
624786 Athlon c
plus negligible ha

No data-depende
No data-depende
No known patent
Software is in pu
http://cr.yp.t

Best attack know
more expensive t
128-bit brute-forc

This talk's recommendations

The "asymmetric" part:
Alice, Bob use Curve25519 to compute long-term shared secret from secret keys, public keys.
Note: minimal asymmetric usage!

The "symmetric" part:
Alice, Bob use shared secret as key for Poly1305+Salsa20 to authenticate+encrypt packets.

Curve25519 is the bottleneck *if* there aren't many packets.
This talk focuses on Curve25519.

Curve25519 secret key: 32 bytes.
Curve25519 public key: 32 bytes.
Time to compute shared secret:
957904 Pentium 4 cycles or
624786 Athlon cycles or . . .
plus negligible hashing time.

No data-dependent branches.
No data-dependent indexing.
No known patent problems.
Software is in public domain.
`http://cr.yp.to/ecdh.html`

Best attack known is more expensive than typical 128-bit brute-force search.

mmendations

c” part:
urve25519 to
rm shared secret
public keys.
symmetric usage!

’ part:
hared secret
305+Salsa20
-encrypt packets.

e bottleneck
any packets.
s on Curve25519.

Curve25519 secret key: 32 bytes.
Curve25519 public key: 32 bytes.
Time to compute shared secret:
957904 Pentium 4 cycles or
624786 Athlon cycles or . . .
plus negligible hashing time.

No data-dependent branches.
No data-dependent indexing.
No known patent problems.
Software is in public domain.
`http://cr.yp.to/ecdh.html`

Best attack known is
more expensive than typical
128-bit brute-force search.

Alice’s secret key
integer a; minor

Alice’s public key
power 9^a in Curv

If Bob’s secret ke
Curve25519 uses
as $\{\text{Alice}, \text{Bob}\}$’s

Bob computes sh
with just one exp
and one short ha

Curve25519 secret key: 32 bytes.
Curve25519 public key: 32 bytes.
Time to compute shared secret:
957904 Pentium 4 cycles or
624786 Athlon cycles or . . .
plus negligible hashing time.

No data-dependent branches.
No data-dependent indexing.
No known patent problems.
Software is in public domain.
`http://cr.yp.to/ecdh.html`

Best attack known is
more expensive than typical
128-bit brute-force search.

Alice's secret key is
integer a; minor restrictions.

Alice's public key is
power 9^a in Curve25519 group.

If Bob's secret key is b:
Curve25519 uses hash of 9^{ab}
as $\{\text{Alice}, \text{Bob}\}$'s shared secret.

Bob computes shared secret
with just one exponentiation
and one short hash.

et key: 32 bytes.
ic key: 32 bytes.
e shared secret:
4 cycles or
ycles or . . .
ashing time.

ent branches.
ent indexing.
t problems.
blic domain.
`to/ecdh.html`

vn is
han typical
ce search.

Alice's secret key is integer a; minor restrictions.

Alice's public key is power 9^a in Curve25519 group.

If Bob's secret key is b: Curve25519 uses hash of 9^{ab} as $\{\text{Alice}, \text{Bob}\}$'s shared secret.

Bob computes shared secret with just one exponentiation and one short hash.

Exponentiation m
in the previous li
take more than t
at the Curve2551
(Other secret-sha
even slower.)

Many interacting
in design and im
Hard to find opti

Remainder of thi
some of the choi
in designing and
Curve25519.

Alice's secret key is
integer a; minor restrictions.

Alice's public key is
power 9^a in Curve25519 group.

If Bob's secret key is b:
Curve25519 uses hash of 9^{ab}
as $\{\text{Alice}, \text{Bob}\}$'s shared secret.

Bob computes shared secret
with just one exponentiation
and one short hash.

Exponentiation methods
in the previous literature
take more than twice as long
at the Curve25519 security level.
(Other secret-sharing methods:
even slower.)

Many interacting parameters
in design and implementation.
Hard to find optimal parameters.

Remainder of this talk discusses
some of the choices made
in designing and implementing
Curve25519.

y is
restrictions.

y is
ve25519 group.

ey is b:
hash of 9^{ab}
shared secret.

hared secret
ponentiation
sh.

Exponentiation methods
in the previous literature
take more than twice as long
at the Curve25519 security level.
(Other secret-sharing methods:
even slower.)

Many interacting parameters
in design and implementation.
Hard to find optimal parameters.

Remainder of this talk discusses
some of the choices made
in designing and implementing
Curve25519.

Curve25519 uses
elliptic-curve gro

"Why not unit g
group T_2 or toru
Why not XTR, u
mults for each ex

Answer: Compar
elliptic curves us
in a smaller field
Overall slightly le

XTR needs large
to protect agains

Exponentiation methods in the previous literature take more than twice as long at the Curve25519 security level. (Other secret-sharing methods: even slower.)

Many interacting parameters in design and implementation. Hard to find optimal parameters.

Remainder of this talk discusses some of the choices made in designing and implementing Curve25519.

Curve25519 uses an elliptic-curve group.

"Why not unit group T_1 or torus group T_2 or torus group T_6? Why not XTR, using only 5.2 mults for each exponent bit?"

Answer: Compared to XTR, elliptic curves use more mults *in a smaller field*. Overall slightly less expensive.

XTR needs larger field to protect against NFS.

nethods
terature
wice as long
19 security level.
aring methods:

; parameters
plementation.
mal parameters.

s talk discusses
ces made
implementing

Curve25519 uses an elliptic-curve group.

"Why not unit group T_1 or torus group T_2 or torus group T_6? Why not XTR, using only 5.2 mults for each exponent bit?"

Answer: Compared to XTR, elliptic curves use more mults *in a smaller field*.
Overall slightly less expensive.

XTR needs larger field to protect against NFS.

Curve25519 com
an elliptic-curve
to a public key x
(Not patented. 1

"But then you ne
an expensive com
Why not also tra

Answer: Transmi
often unacceptab
A square-root co
isn't terribly expe
and is avoided er
Curve25519 com

Curve25519 uses an elliptic-curve group.

"Why not unit group T_1 or torus group T_2 or torus group T_6? Why not XTR, using only 5.2 mults for each exponent bit?"

Answer: Compared to XTR, elliptic curves use more mults *in a smaller field*. Overall slightly less expensive.

XTR needs larger field to protect against NFS.

Curve25519 compresses an elliptic-curve point (x, y) to a public key x. (Not patented. 1986 Miller.)

"But then you need an expensive computation of y! Why not also transmit y?"

Answer: Transmitting y is often unacceptably expensive. A square-root computation isn't terribly expensive— and is avoided entirely in the Curve25519 computation.

an
up.

roup T_1 or torus
s group T_6?
sing only 5.2
xponent bit?"

ed to XTR,
e more mults
.
ess expensive.

r field
st NFS.

Curve25519 compresses
an elliptic-curve point (x, y)
to a public key x.
(Not patented. 1986 Miller.)

"But then you need
an expensive computation of y!
Why not also transmit y?"

Answer: Transmitting y is
often unacceptably expensive.
A square-root computation
isn't terribly expensive—
and is avoided entirely in the
Curve25519 computation.

Curve25519 uses
over a large-char

"Why not char 2
Squaring is almo
Can exploit Frob

Answer: Current
fast floating-poin
for physics simul
Can reuse these
arithmetic in larg
Outweighs the ch

Curve25519 compresses
an elliptic-curve point (x, y)
to a public key x.
(Not patented. 1986 Miller.)

"But then you need
an expensive computation of y!
Why not also transmit y?"

Answer: Transmitting y is
often unacceptably expensive.
A square-root computation
isn't terribly expensive—
and is avoided entirely in the
Curve25519 computation.

Curve25519 uses a curve
over a large-char field.

"Why not char 2?
Squaring is almost for free!
Can exploit Frobenius on curve."

Answer: Current CPUs include
fast floating-point multipliers
for physics simulation etc.
Can reuse these multipliers for
arithmetic in large-char fields.
Outweighs the char-2 advantages.

presses
point (x, y)
?.
1986 Miller.)

eed
nputation of y!
nsmit y?"

itting y is
bly expensive.
mputation
ensive—
ntirely in the
putation.

Curve25519 uses a curve
over a large-char field.

"Why not char 2?
Squaring is almost for free!
Can exploit Frobenius on curve."

Answer: Current CPUs include
fast floating-point multipliers
for physics simulation etc.
Can reuse these multipliers for
arithmetic in large-char fields.
Outweighs the char-2 advantages.

Curve25519 uses
$y^2 = x^3+Ax^2+x$

"Why not $y^2 = x$
Double (x, y) in
using only 5 field
and 3 extra field

Answer: With y^2
can do projective
doubling *and add*
using 1 field mult
4 field squarings,
5 extra field mult

Curve25519 uses a curve over a large-char field.

"Why not char 2? Squaring is almost for free! Can exploit Frobenius on curve."

Answer: Current CPUs include fast floating-point multipliers for physics simulation etc. Can reuse these multipliers for arithmetic in large-char fields. Outweighs the char-2 advantages.

Curve25519 uses curve shape $y^2 = x^3+Ax^2+x$, tiny $A \in 2+4\mathbf{Z}$.

"Why not $y^2 = x^3 - 3x + a_6$? Double (x, y) in Jacobian coords using only 5 field squarings and 3 extra field mults!"

Answer: With $y^2 = x^3 + Ax^2 + x$, can do projective x-coord doubling *and addition* together using 1 field mult by $(A-2)/4$, 4 field squarings, 5 extra field mults. Never need y.

a curve
field.

?
st for free!
enius on curve."

CPUs include
nt multipliers
ation etc.
multipliers for
ge-char fields.
har-2 advantages.

Curve25519 uses curve shape
$y^2 = x^3 + Ax^2 + x$, tiny $A \in 2 + 4\mathbf{Z}$.

"Why not $y^2 = x^3 - 3x + a_6$?
Double (x, y) in Jacobian coords
using only 5 field squarings
and 3 extra field mults!"

Answer: With $y^2 = x^3 + Ax^2 + x$,
can do projective x-coord
doubling *and addition* together
using 1 field mult by $(A - 2)/4$,
4 field squarings,
5 extra field mults. Never need y.

Curve25519 uses

"Why not an ext
Adapt extension
to CPU's multipl
Avoid carries in a

Answer: Extensio
punishes CPUs w
another multiplie
Maybe tolerable
as CPUs converg
carries are a very

Curve25519 uses curve shape
$y^2 = x^3 + Ax^2 + x$, tiny $A \in 2 + 4\mathbf{Z}$.

"Why not $y^2 = x^3 - 3x + a_6$?
Double (x, y) in Jacobian coords
using only 5 field squarings
and 3 extra field mults!"

Answer: With $y^2 = x^3 + Ax^2 + x$,
can do projective x-coord
doubling *and addition* together
using 1 field mult by $(A - 2)/4$,
4 field squarings,
5 extra field mults. Never need y.

Curve25519 uses a prime field.

"Why not an extension field?
Adapt extension degree
to CPU's multiplier size.
Avoid carries in arithmetic!"

Answer: Extension field
punishes CPUs with
another multiplier size.
Maybe tolerable
as CPUs converge—but
carries are a very small cost.

curve shape
x, tiny $A \in 2+4\mathbf{Z}$.

$x^3 - 3x + a_6$?
Jacobian coords
squarings
mults!"

$^2 = x^3 + Ax^2 + x$,
e x-coord
dition together
t by $(A-2)/4$,

ts. Never need y.

Curve25519 uses a prime field.

"Why not an extension field?
Adapt extension degree
to CPU's multiplier size.
Avoid carries in arithmetic!"

Answer: Extension field
punishes CPUs with
another multiplier size.
Maybe tolerable
as CPUs converge—but
carries are a very small cost.

Curve25519 uses
extremely close t
specifically, 2^{255}

"Why not a word
$2^{256} - 2^{224} + 2^{1}$
Reduce by simple
and subtractions!

Answer: Repeate
are more expensi
a multiplication b
Also, analogous p
to extension field

Curve25519 uses a prime field.

“Why not an extension field? Adapt extension degree to CPU’s multiplier size. Avoid carries in arithmetic!”

Answer: Extension field punishes CPUs with another multiplier size. Maybe tolerable as CPUs converge—but carries are a very small cost.

Curve25519 uses prime extremely close to a power of 2: specifically, $2^{255} - 19$.

“Why not a word-aligned prime, $2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$? Reduce by simple word additions and subtractions!”

Answer: Repeated additions are more expensive than a multiplication by 19. Also, analogous problem to extension fields.

a prime field.

tension field?
degree
ier size.
arithmetic!"

on field
with
er size.

ge—but
small cost.

Curve25519 uses prime extremely close to a power of 2: specifically, $2^{255} - 19$.

"Why not a word-aligned prime, $2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$? Reduce by simple word additions and subtractions!"

Answer: Repeated additions are more expensive than a multiplication by 19. Also, analogous problem to extension fields.

Curve25519 com
largest convenien
with integer w.

Example: With 6
floating-point ma
Curve25519 uses
$\sum_i$ small multipl

"Why not use ra
2^{26}? Doesn't the
to be an integer?

Answer: No, exp
have to be an int
Radix $2^{25.5}$ save
reduction mod 2

Curve25519 uses prime extremely close to a power of 2: specifically, $2^{255} - 19$.

"Why not a word-aligned prime, $2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$? Reduce by simple word additions and subtractions!"

Answer: Repeated additions are more expensive than a multiplication by 19. Also, analogous problem to extension fields.

Curve25519 computation uses largest convenient radix $2^{255/w}$ with integer w.

Example: With 64-bit x86 floating-point mantissas, Curve25519 uses radix $2^{25.5}$, i.e., $\sum_i$ small multiple of $2^{\lceil 25.5i \rceil}$.

"Why not use radix 2^{25}, or radix 2^{26}? Doesn't the exponent have to be an integer?"

Answer: No, exponent doesn't have to be an integer. Radix $2^{25.5}$ saves time in reduction mod $2^{255} - 19$.

prime
o a power of 2:
$-$ 19.

d-aligned prime,
$^{192} + 2^{96} - 1$?
e word additions
!"

ed additions
ve than
by 19.
problem
ls.

Curve25519 computation uses largest convenient radix $2^{255/w}$ with integer w.

Example: With 64-bit x86 floating-point mantissas, Curve25519 uses radix $2^{25.5}$, i.e., $\sum_i$ small multiple of $2^{\lceil 25.5i \rceil}$.

"Why not use radix 2^{25}, or radix 2^{26}? Doesn't the exponent have to be an integer?"

Answer: No, exponent doesn't have to be an integer.
Radix $2^{25.5}$ saves time in reduction mod $2^{255} - 19$.

Curve25519 com
coefficients sligh
than the radix.

"Why not use ca
with minimal co
Smaller coefficie
allow faster arith

Answer: Convers
to canonical form
Making coefficie
is much less exp
than making the
Has most of the

Curve25519 computation uses largest convenient radix $2^{255/w}$ with integer w.

Example: With 64-bit x86 floating-point mantissas, Curve25519 uses radix $2^{25.5}$, i.e., $\sum_i$ small multiple of $2^{\lceil 25.5i \rceil}$.

"Why not use radix 2^{25}, or radix 2^{26}? Doesn't the exponent have to be an integer?"

Answer: No, exponent doesn't have to be an integer. Radix $2^{25.5}$ saves time in reduction mod $2^{255} - 19$.

Curve25519 computation allows coefficients slightly larger than the radix.

"Why not use canonical form, with minimal coefficients? Smaller coefficients allow faster arithmetic!"

Answer: Conversion to canonical form is expensive. Making coefficients *small* is much less expensive than making them *smallest*. Has most of the same benefit.

putation uses
nt radix $2^{255/w}$

64-bit x86
antissas,
radix $2^{25.5}$, i.e.,
e of $2^{\lceil 25.5i \rceil}$.

dix 2^{25}, or radix
e exponent have
?"

onent doesn't
teger.
s time in
$^{255} - 19$.

Curve25519 computation allows
coefficients slightly larger
than the radix.

"Why not use canonical form,
with minimal coefficients?
Smaller coefficients
allow faster arithmetic!"

Answer: Conversion
to canonical form is expensive.
Making coefficients *small*
is much less expensive
than making them *smallest*.
Has most of the same benefit.

Curve25519 conv
indexing into arit
given $P[0]$, $P[1]$,
$P[b]$ as $bP[1] + ($

"Why not simply
index? Skip the
$b, 1 - b$ and the a

Answer: This ari
of the Curve2551
Protects against
such as hyperthr
Less expensive th
variable array ind

Curve25519 computation allows coefficients slightly larger than the radix.

"Why not use canonical form, with minimal coefficients? Smaller coefficients allow faster arithmetic!"

Answer: Conversion to canonical form is expensive. Making coefficients *small* is much less expensive than making them *smallest*. Has most of the same benefit.

Curve25519 converts variable indexing into arithmetic: e.g., given $P[0]$, $P[1]$, bit b, compute $P[b]$ as $bP[1] + (1 - b)P[0]$.

"Why not simply use b as an array index? Skip the multiplications by $b, 1 - b$ and the addition!"

Answer: This arithmetic is 6% of the Curve25519 computation. Protects against timing attacks, such as hyperthreading attacks. Less expensive than protecting variable array indexing.

putation allows
tly larger

nonical form,
efficients?
nts
metic!"

sion
n is expensive.
nts *small*
ensive
m *smallest*.
same benefit.

Curve25519 converts variable indexing into arithmetic: e.g., given $P[0]$, $P[1]$, bit b, compute $P[b]$ as $bP[1] + (1-b)P[0]$.

"Why not simply use b as an array index? Skip the multiplications by $b, 1-b$ and the addition!"

Answer: This arithmetic is 6% of the Curve25519 computation. Protects against timing attacks, such as hyperthreading attacks. Less expensive than protecting variable array indexing.

Curve25519 uses
with a secure twi
$y^2 = x^3 + 48666$
Group order $8 \cdot$ p
Twist group orde

"Why worry abo
Why not simply
keys on the twist

Answer: Prohibit
twist means chec
("validating keys
cost by choosing

Curve25519 converts variable indexing into arithmetic: e.g., given $P[0]$, $P[1]$, bit b, compute $P[b]$ as $bP[1] + (1 - b)P[0]$.

"Why not simply use b as an array index? Skip the multiplications by $b, 1 - b$ and the addition!"

Answer: This arithmetic is 6% of the Curve25519 computation. Protects against timing attacks, such as hyperthreading attacks. Less expensive than protecting variable array indexing.

Curve25519 uses a secure curve with a secure twist:
$y^2 = x^3 + 486662x^2 + x$.
Group order $8 \cdot$ prime.
Twist group order $4 \cdot$ prime.

"Why worry about twist order? Why not simply prohibit keys on the twist?"

Answer: Prohibiting keys on the twist means checking for them ("validating keys"). Eliminate this cost by choosing curve carefully.

verts variable
thmetic: e.g.,
bit b, compute
$(1 - b)P[0]$.

use b as an array
multiplications by
addition!"

thmetic is 6%
19 computation.
timing attacks,
eading attacks.
han protecting
lexing.

Curve25519 uses a secure curve with a secure twist:
$y^2 = x^3 + 486662x^2 + x$.
Group order $8 \cdot$ prime.
Twist group order $4 \cdot$ prime.

"Why worry about twist order? Why not simply prohibit keys on the twist?"

Answer: Prohibiting keys on the twist means checking for them ("validating keys"). Eliminate this cost by choosing curve carefully.

What's next?

Culmination of ex
on eliminating fie
genus-2 hyperelli
25 mults per bit.
`eprint.iacr.or`

Half-size prime:
Select curve to m
mults easier, like
this needs faster

Should analyze c
instead of field m
Prediction: this w

Curve25519 uses a secure curve with a secure twist:
$y^2 = x^3 + 486662x^2 + x$.
Group order $8 \cdot$ prime.
Twist group order $4 \cdot$ prime.

"Why worry about twist order? Why not simply prohibit keys on the twist?"

Answer: Prohibiting keys on the twist means checking for them ("validating keys"). Eliminate this cost by choosing curve carefully.

What's next?

Culmination of extensive work on eliminating field mults for genus-2 hyperelliptic curves: 25 mults per bit. Gaudry, `eprint.iacr.org/2005/314`

Half-size prime: e.g., $2^{127} - 1$. Select curve to make some mults easier, like taking tiny A; this needs faster point counting!

Should analyze cycles instead of field mults.
Prediction: this will beat genus 1.