Hyper-and-elliptic-curve cryptography

(which is not the same as: hyperelliptic-curve cryptography and elliptic-curve cryptography)

Daniel J. Bernstein
University of Illinois at Chicago &
Technische Universiteit Eindhoven

Joint work with:
Tanja Lange
Technische Universiteit Eindhoven

But first some context. . .

ECC security vs. ECDL security

Crypto view of ECDL problem:
Fix finite k, E/k, $P \in E(k)$.
Secret key: random $a \in \mathbf{Z}/\#\mathbf{Z}P$.
Public key: aP.
The ECDL problem: compute secret key from public key.

ECDL solution $\Rightarrow$ ECC attack.

Hyper-and-elliptic-curve cryptography

(which is not the same as: hyperelliptic-curve cryptography and elliptic-curve cryptography)

Daniel J. Bernstein
University of Illinois at Chicago &
Technische Universiteit Eindhoven

Joint work with:
Tanja Lange
Technische Universiteit Eindhoven

But first some context. . .

ECC security vs. ECDL security

Crypto view of ECDL problem:
Fix finite k, E/k, $P \in E(k)$.
Secret key: random $a \in \mathbf{Z}/\#\mathbf{Z}P$.
Public key: aP.
The ECDL problem: compute secret key from public key.

ECDL solution $\Rightarrow$ ECC attack.

ECC attack $\Rightarrow$ ECDL solution?
Not necessarily!

Let's look at some examples.

nd-elliptic-curve
raphy

s not the same as:
ptic-curve cryptography
tic-curve cryptography)

. Bernstein
ty of Illinois at Chicago &
che Universiteit Eindhoven

ork with:
ange
che Universiteit Eindhoven

---

t some context. . .

ECC security vs. ECDL security

Crypto view of ECDL problem:
Fix finite k, E/k, $P \in E(k)$.
Secret key: random $a \in \mathbf{Z}/\#\mathbf{Z}P$.
Public key: aP.
The ECDL problem: compute
secret key from public key.

ECDL solution $\Rightarrow$ ECC attack.

ECC attack $\Rightarrow$ ECDL solution?
Not necessarily!

Let's look at some examples.

Example
(1985 M
receives
uses eas
$X(bA)$; e

-curve

same as:
cryptography
cryptography)

n
is at Chicago &
siteit Eindhoven

siteit Eindhoven

---

ntext. . .

ECC security vs. ECDL security

Crypto view of ECDL problem:
Fix finite k, E/k, $P \in E(k)$.
Secret key: random $a \in \mathbf{Z}/\#\mathbf{Z}P$.
Public key: aP.
The ECDL problem: compute secret key from public key.

ECDL solution $\Rightarrow$ ECC attack.

ECC attack $\Rightarrow$ ECDL solution?
Not necessarily!

Let's look at some examples.

Example 1: **Kumm**
(1985 Miller). Bo
receives $X(A)$ fro
uses easy formulas
$X(bA)$; encrypts u

nphy
phy)

ago &
hoven

hoven

ECC security vs. ECDL security

Crypto view of ECDL problem:
Fix finite k, E/k, $P \in E(k)$.
Secret key: random $a \in \mathbf{Z}/\#\mathbf{Z}P$.
Public key: aP.
The ECDL problem: compute secret key from public key.

ECDL solution $\Rightarrow$ ECC attack.

ECC attack $\Rightarrow$ ECDL solution?
Not necessarily!

Let's look at some examples.

Example 1: **Kummer-line E**
(1985 Miller). Bob has secre
receives $X(A)$ from Alice;
uses easy formulas to compu
$X(bA)$; encrypts using $X(bA$

ECC security vs. ECDL security

Crypto view of ECDL problem:
Fix finite k, E/k, $P \in E(k)$.
Secret key: random $a \in \mathbf{Z}/\#\mathbf{Z}P$.
Public key: aP.
The ECDL problem: compute secret key from public key.

ECDL solution $\Rightarrow$ ECC attack.

ECC attack $\Rightarrow$ ECDL solution?
Not necessarily!

Let's look at some examples.

Example 1: **Kummer-line ECDH** (1985 Miller). Bob has secret b; receives $X(A)$ from Alice; uses easy formulas to compute $X(bA)$; encrypts using $X(bA)$.

ECC security vs. ECDL security

Crypto view of ECDL problem:
Fix finite k, E/k, $P \in E(k)$.
Secret key: random $a \in \mathbf{Z}/\#\mathbf{Z}P$.
Public key: aP.
The ECDL problem: compute secret key from public key.

ECDL solution $\Rightarrow$ ECC attack.

ECC attack $\Rightarrow$ ECDL solution?
Not necessarily!

Let's look at some examples.

Example 1: **Kummer-line ECDH** (1985 Miller). Bob has secret b; receives $X(A)$ from Alice; uses easy formulas to compute $X(bA)$; encrypts using $X(bA)$.

Twist attack: choose $A \in E(\overline{k})$, small $\#\mathbf{Z}A$; learn b mod $\#\mathbf{Z}A$.

ECC security vs. ECDL security

Crypto view of ECDL problem:
Fix finite k, E/k, $P \in E(k)$.
Secret key: random $a \in \mathbf{Z}/\#\mathbf{Z}P$.
Public key: aP.
The ECDL problem: compute secret key from public key.

ECDL solution $\Rightarrow$ ECC attack.

ECC attack $\Rightarrow$ ECDL solution?
Not necessarily!

Let's look at some examples.

Example 1: **Kummer-line ECDH** (1985 Miller). Bob has secret b; receives $X(A)$ from Alice; uses easy formulas to compute $X(bA)$; encrypts using $X(bA)$.

Twist attack: choose $A \in E(\overline{k})$, small $\#\mathbf{Z}A$; learn b mod $\#\mathbf{Z}A$.

Typically Bob checks $X(A) \in k$ but doesn't check $A \in E(k)$.
Formulas also work for $A \in E'(k)$ for appropriate twist E' of E.
Typically $\#E(k)$ is large prime but $\#E'(k)$ has small factors.

curity vs. ECDL security

view of ECDL problem:
e k, E/k, $P \in E(k)$.
key: random $a \in \mathbf{Z}/\#\mathbf{Z}P$.
key: aP.
DL problem: compute
ey from public key.

olution $\Rightarrow$ ECC attack.

ack $\Rightarrow$ ECDL solution?
essarily!

ok at some examples.

Example 1: **Kummer-line ECDH** (1985 Miller). Bob has secret b; receives $X(A)$ from Alice; uses easy formulas to compute $X(bA)$; encrypts using $X(bA)$.

Twist attack: choose $A \in E(\overline{k})$, small $\#\mathbf{Z}A$; learn b mod $\#\mathbf{Z}A$.

Typically Bob checks $X(A) \in k$ but doesn't check $A \in E(k)$. Formulas also work for $A \in E'(k)$ for appropriate twist E' of E. Typically $\#E(k)$ is large prime but $\#E'(k)$ has small factors.

Example
terminat
many el

2004 M
Send $X($

Annoyin

Same ba
random-
(see, e.g
2006 Sc
passwor
exchang
Montag
ID-base

ECDL security

CDL problem:
$P \in E(k)$.
om $a \in \mathbf{Z}/\#\mathbf{Z}P$.

m: compute
ublic key.

ECC attack.

CDL solution?

e examples.

Example 1: **Kummer-line ECDH** (1985 Miller). Bob has secret b; receives $X(A)$ from Alice; uses easy formulas to compute $X(bA)$; encrypts using $X(bA)$.

Twist attack: choose $A \in E(\overline{k})$, small $\#\mathbf{Z}A$; learn b mod $\#\mathbf{Z}A$.

Typically Bob checks $X(A) \in k$ but doesn't check $A \in E(k)$. Formulas also work for $A \in E'(k)$ for appropriate twist E' of E. Typically $\#E(k)$ is large prime but $\#E'(k)$ has small factors.

Example 2: Censo
terminates users w
many elements of

2004 Möller: Fix t
Send $X(aP)$ or X

Annoying: e.g., co

Same basic issue a
random-number ge
(see, e.g., 2006 Gj
2006 Schoenmake
password-authenti
exchange (e.g., 20
Montague–Nguyen
ID-based encryptio

rity

em:

.

$\#\mathbf{Z}P$.

te

ck.

on?

s.

Example 1: **Kummer-line ECDH** (1985 Miller). Bob has secret b; receives $X(A)$ from Alice; uses easy formulas to compute $X(bA)$; encrypts using $X(bA)$.

Twist attack: choose $A \in E(\overline{k})$, small $\#\mathbf{Z}A$; learn b mod $\#\mathbf{Z}A$.

Typically Bob checks $X(A) \in k$ but doesn't check $A \in E(k)$. Formulas also work for $A \in E'(k)$ for appropriate twist E' of E. Typically $\#E(k)$ is large prime but $\#E'(k)$ has small factors.

Example 2: Censor scans ne
terminates users who send
many elements of $X(E(k))$.

2004 Möller: Fix twist-secur
Send $X(aP)$ or $X(a'P')$.

Annoying: e.g., consider EC

Same basic issue arises in
random-number generation
(see, e.g., 2006 Gjøsteen an
2006 Schoenmakers–Sidoren
password-authenticated key
exchange (e.g., 2001 Boyd–
Montague–Nguyen, broken 2
ID-based encryption, etc.

Example 1: **Kummer-line ECDH** (1985 Miller). Bob has secret b; receives $X(A)$ from Alice; uses easy formulas to compute $X(bA)$; encrypts using $X(bA)$.

Twist attack: choose $A \in E(\overline{k})$, small $\#\mathbf{Z}A$; learn b mod $\#\mathbf{Z}A$.

Typically Bob checks $X(A) \in k$ but doesn't check $A \in E(k)$. Formulas also work for $A \in E'(k)$ for appropriate twist E' of E. Typically $\#E(k)$ is large prime but $\#E'(k)$ has small factors.

Example 2: Censor scans network, terminates users who send many elements of $X(E(k))$.

2004 Möller: Fix twist-secure E. Send $X(aP)$ or $X(a'P')$.

Annoying: e.g., consider ECDH.

Same basic issue arises in random-number generation (see, e.g., 2006 Gjøsteen and 2006 Schoenmakers–Sidorenko), password-authenticated key exchange (e.g., 2001 Boyd–Montague–Nguyen, broken 2013), ID-based encryption, etc.

e 1: **Kummer-line ECDH**
lliller). Bob has secret b;
$X(A)$ from Alice;
y formulas to compute
encrypts using $X(bA)$.

ttack: choose $A \in E(\overline{k})$,
$\mathbf{Z}A$; learn b mod $\#\mathbf{Z}A$.

y Bob checks $X(A) \in k$
sn't check $A \in E(k)$.
s also work for $A \in E'(k)$
opriate twist E' of E.
y $\#E(k)$ is large prime
$'(k)$ has small factors.

Example 2: Censor scans network, terminates users who send many elements of $X(E(k))$.

2004 Möller: Fix twist-secure E. Send $X(aP)$ or $X(a'P')$.

Annoying: e.g., consider ECDH.

Same basic issue arises in random-number generation (see, e.g., 2006 Gjøsteen and 2006 Schoenmakers–Sidorenko), password-authenticated key exchange (e.g., 2001 Boyd–Montague–Nguyen, broken 2013), ID-based encryption, etc.

2013 Be
Krasnov
"Elligato
indisting
uniform

Replace
between
e.g., inte

Alice ke
until aP

Two exa
both wit
for reaso

mer-line ECDH

b has secret b;
m Alice;
s to compute
sing $X(bA)$.

oose $A \in E(\overline{k})$,
b mod $\#\mathbf{Z}A$.

cks $X(A) \in k$
$A \in E(k)$.
k for $A \in E'(k)$
ist E' of E.
s large prime
mall factors.

Example 2: Censor scans network, terminates users who send many elements of $X(E(k))$.

2004 Möller: Fix twist-secure E. Send $X(aP)$ or $X(a'P')$.

Annoying: e.g., consider ECDH.

Same basic issue arises in random-number generation (see, e.g., 2006 Gjøsteen and 2006 Schoenmakers–Sidorenko), password-authenticated key exchange (e.g., 2001 Boyd–Montague–Nguyen, broken 2013), ID-based encryption, etc.

2013 Bernstein–Ha
Krasnova–Lange
"Elligator: Elliptic
indistinguishable f
uniform random st

Replace X with fa
between large $S \subseteq$
e.g., interval $\{0, 1$

Alice keeps genera
until $aP \in S$.

Two examples give
both with $\#S \approx 0$
for reasonable cho

ECDH

et b;

ute

$).$

$E(\overline{k})$,

$\mathbb{Z}A$.

$\in k$

.

$E'(k)$

E'.

me

rs.

Example 2: Censor scans network, terminates users who send many elements of $X(E(k))$.

2004 Möller: Fix twist-secure E. Send $X(aP)$ or $X(a'P')$.

Annoying: e.g., consider ECDH.

Same basic issue arises in random-number generation (see, e.g., 2006 Gjøsteen and 2006 Schoenmakers–Sidorenko), password-authenticated key exchange (e.g., 2001 Boyd–Montague–Nguyen, broken 2013), ID-based encryption, etc.

2013 Bernstein–Hamburg–Krasnova–Lange "Elligator: Elliptic-curve poi indistinguishable from uniform random strings":

Replace X with fast bijectio between large $S \subseteq E(k)$ and e.g., interval $\{0, 1, \ldots, 2^b -$

Alice keeps generating a until $aP \in S$.

Two examples given in pape both with $\#S \approx 0.5\#E(k)$ for reasonable choices of k.

Example 2: Censor scans network, terminates users who send many elements of $X(E(k))$.

2004 Möller: Fix twist-secure E. Send $X(aP)$ or $X(a'P')$.

Annoying: e.g., consider ECDH.

Same basic issue arises in random-number generation (see, e.g., 2006 Gjøsteen and 2006 Schoenmakers–Sidorenko), password-authenticated key exchange (e.g., 2001 Boyd–Montague–Nguyen, broken 2013), ID-based encryption, etc.

2013 Bernstein–Hamburg–Krasnova–Lange "Elligator: Elliptic-curve points indistinguishable from uniform random strings":

Replace X with fast bijection between large $S \subseteq E(k)$ and, e.g., interval $\{0, 1, \ldots, 2^b - 1\}$.

Alice keeps generating a until $aP \in S$.

Two examples given in paper, both with $\#S \approx 0.5\#E(k)$ for reasonable choices of k.

e 2: Censor scans network,
tes users who send
ements of $X(E(k))$.

öller: Fix twist-secure E.
(aP) or $X(a'P')$.

g: e.g., consider ECDH.

asic issue arises in
number generation
., 2006 Gjøsteen and
hoenmakers–Sidorenko),
d-authenticated key
e (e.g., 2001 Boyd–
ue–Nguyen, broken 2013),
d encryption, etc.

2013 Bernstein–Hamburg–
Krasnova–Lange
"Elligator: Elliptic-curve points
indistinguishable from
uniform random strings":

Replace X with fast bijection
between large $S \subseteq E(k)$ and,
e.g., interval $\{0, 1, \ldots, 2^b - 1\}$.

Alice keeps generating a
until $aP \in S$.

Two examples given in paper,
both with $\#S \approx 0.5\#E(k)$
for reasonable choices of k.

"Elligato
reinterpr
2013 Fo

Fix prim
$s \in \mathbf{F}_q^*$ w
$(s^2 - 2)$
$c = 2/s^2$
$d = -(c$

Define E
This is a

For ϕ : **F**
on next
of $\phi(t)$ u

r scans network,
who send
$X(E(k))$.

twist-secure E.
$(a'P')$.

onsider ECDH.

arises in
eneration
østeen and
rs–Sidorenko),
cated key
01 Boyd–
n, broken 2013),
on, etc.

2013 Bernstein–Hamburg–
Krasnova–Lange
"Elligator: Elliptic-curve points
indistinguishable from
uniform random strings":

Replace X with fast bijection
between large $S \subseteq E(k)$ and,
e.g., interval $\{0, 1, \ldots, 2^b - 1\}$.

Alice keeps generating a
until $aP \in S$.

Two examples given in paper,
both with $\#S \approx 0.5\#E(k)$
for reasonable choices of k.

"Elligator 1",
reinterpreting and
2013 Fouque–Jou

Fix prime power q
$s \in \mathbf{F}_q^*$ with
$(s^2 - 2)(s^2 + 2) \neq$
$c = 2/s^2$; $r = c +$
$d = -(c + 1)^2/(c$

Define $E : x^2 + y^2$
This is a complete

For $\phi : \mathbf{F}_q \to E(\mathbf{F}_q$
on next slide: the
of $\phi(t)$ under ϕ ar

twork,

re E.

DH.

d
ako),

2013),

2013 Bernstein–Hamburg–
Krasnova–Lange
"Elligator: Elliptic-curve points
indistinguishable from
uniform random strings":

Replace X with fast bijection
between large $S \subseteq E(k)$ and,
e.g., interval $\{0, 1, \ldots, 2^b - 1\}$.

Alice keeps generating a
until $aP \in S$.

Two examples given in paper,
both with $\#S \approx 0.5\#E(k)$
for reasonable choices of k.

"Elligator 1",
reinterpreting and simplifyin
2013 Fouque–Joux–Tibouch

Fix prime power $q \in 3 + 4\mathbf{Z}$
$s \in \mathbf{F}_q^*$ with
$(s^2 - 2)(s^2 + 2) \neq 0$;
$c = 2/s^2$; $r = c + 1/c$;
$d = -(c + 1)^2/(c - 1)^2$.

Define $E : x^2 + y^2 = 1 + dx$
This is a complete Edwards

For $\phi : \mathbf{F}_q \to E(\mathbf{F}_q)$ defined
on next slide: the only preim
of $\phi(t)$ under ϕ are $\{t, -t\}$.

2013 Bernstein–Hamburg–
Krasnova–Lange
"Elligator: Elliptic-curve points
indistinguishable from
uniform random strings":

Replace X with fast bijection
between large $S \subseteq E(k)$ and,
e.g., interval $\{0, 1, \ldots, 2^b - 1\}$.

Alice keeps generating a
until $aP \in S$.

Two examples given in paper,
both with $\#S \approx 0.5\#E(k)$
for reasonable choices of k.

"Elligator 1",
reinterpreting and simplifying
2013 Fouque–Joux–Tibouchi:

Fix prime power $q \in 3 + 4\mathbf{Z}$;
$s \in \mathbf{F}_q^*$ with
$(s^2 - 2)(s^2 + 2) \neq 0$;
$c = 2/s^2$; $r = c + 1/c$;
$d = -(c + 1)^2/(c - 1)^2$.

Define $E : x^2 + y^2 = 1 + dx^2y^2$.
This is a complete Edwards curve.

For $\phi : \mathbf{F}_q \to E(\mathbf{F}_q)$ defined
on next slide: the only preimages
of $\phi(t)$ under ϕ are $\{t, -t\}$.

rnstein–Hamburg–
a–Lange
or: Elliptic-curve points
guishable from
random strings":

X with fast bijection
large $S \subseteq E(k)$ and,
erval $\{0, 1, \ldots, 2^b - 1\}$.

eps generating a
$\in S$.

mples given in paper,
th $\#S \approx 0.5\#E(k)$
onable choices of k.

"Elligator 1",
reinterpreting and simplifying
2013 Fouque–Joux–Tibouchi:

Fix prime power $q \in 3 + 4\mathbf{Z}$;
$s \in \mathbf{F}_q^*$ with
$(s^2 - 2)(s^2 + 2) \neq 0$;
$c = 2/s^2$; $r = c + 1/c$;
$d = -(c + 1)^2/(c - 1)^2$.

Define $E : x^2 + y^2 = 1 + dx^2y^2$.
This is a complete Edwards curve.

For $\phi : \mathbf{F}_q \to E(\mathbf{F}_q)$ defined
on next slide: the only preimages
of $\phi(t)$ under ϕ are $\{t, -t\}$.

$\phi(\pm 1) =$

Otherwis

$u = (1 -$

$v = u^5 -$

$X = \chi($

$Y =$
$(\chi(v)v)$

$x = (c -$

$y = \frac{rX}{rX}$

amburg–

-curve points
rom
trings":

st bijection
$E(k)$ and,
$\ldots, 2^b - 1\}$.

ting a

en in paper,
$0.5\#E(k)$
ices of k.

"Elligator 1",
reinterpreting and simplifying
2013 Fouque–Joux–Tibouchi:

Fix prime power $q \in 3 + 4\mathbf{Z}$;
$s \in \mathbf{F}_q^*$ with
$(s^2 - 2)(s^2 + 2) \neq 0$;
$c = 2/s^2$; $r = c + 1/c$;
$d = -(c+1)^2/(c-1)^2$.

Define $E : x^2 + y^2 = 1 + dx^2y^2$.
This is a complete Edwards curve.

For $\phi : \mathbf{F}_q \to E(\mathbf{F}_q)$ defined
on next slide: the only preimages
of $\phi(t)$ under ϕ are $\{t, -t\}$.

$\phi(\pm 1) = (0, 1)$.

Otherwise $\phi(t) =$

$u = (1 - t)/(1 +$

$v = u^5 + (r^2 - 2)$

$X = \chi(v)u$,

$Y =$
$(\chi(v)v)^{(q+1)/4}\chi($

$x = (c - 1)sX(1 -$

$y = \dfrac{rX - (1 + X}{rX + (1 + X}$

nts

n
d,
1}.

er,

"Elligator 1",
reinterpreting and simplifying
2013 Fouque–Joux–Tibouchi:

Fix prime power $q \in 3 + 4\mathbf{Z}$;
$s \in \mathbf{F}_q^*$ with
$(s^2 - 2)(s^2 + 2) \neq 0$;
$c = 2/s^2$; $r = c + 1/c$;
$d = -(c+1)^2/(c-1)^2$.

Define $E : x^2 + y^2 = 1 + dx^2y^2$.
This is a complete Edwards curve.

For $\phi : \mathbf{F}_q \to E(\mathbf{F}_q)$ defined
on next slide: the only preimages
of $\phi(t)$ under ϕ are $\{t, -t\}$.

$\phi(\pm 1) = (0, 1)$.

Otherwise $\phi(t) = (x, y)$ whe

$u = (1 - t)/(1 + t)$,

$v = u^5 + (r^2 - 2)u^3 + u$,

$X = \chi(v)u$,

$Y =$
$(\chi(v)v)^{(q+1)/4}\chi(v)\chi\big(u^2 +$

$x = (c - 1)sX(1 + X)/Y$,

$$y = \frac{rX - (1+X)^2}{rX + (1+X)^2}.$$

"Elligator 1",
reinterpreting and simplifying
2013 Fouque–Joux–Tibouchi:

Fix prime power $q \in 3 + 4\mathbf{Z}$;
$s \in \mathbf{F}_q^*$ with
$(s^2 - 2)(s^2 + 2) \neq 0$;
$c = 2/s^2$; $r = c + 1/c$;
$d = -(c+1)^2/(c-1)^2$.

Define $E : x^2 + y^2 = 1 + dx^2y^2$.
This is a complete Edwards curve.

For $\phi : \mathbf{F}_q \to E(\mathbf{F}_q)$ defined
on next slide: the only preimages
of $\phi(t)$ under ϕ are $\{t, -t\}$.

$\phi(\pm 1) = (0, 1)$.

Otherwise $\phi(t) = (x, y)$ where

$$u = (1 - t)/(1 + t),$$

$$v = u^5 + (r^2 - 2)u^3 + u,$$

$$X = \chi(v)u,$$

$$Y = (\chi(v)v)^{(q+1)/4}\chi(v)\chi\left(u^2 + 1/c^2\right),$$

$$x = (c - 1)sX(1 + X)/Y,$$

$$y = \frac{rX - (1 + X)^2}{rX + (1 + X)^2}.$$

or 1",

reting and simplifying

uque–Joux–Tibouchi:

e power $q \in 3 + 4\mathbf{Z}$;

with

$(s^2 + 2) \neq 0$;

^{2}; $r = c + 1/c$;

$+ 1)^2/(c - 1)^2$.

$E : x^2 + y^2 = 1 + dx^2y^2$.

a complete Edwards curve.

$\mathbf{F}_q \to E(\mathbf{F}_q)$ defined

slide: the only preimages

under ϕ are $\{t, -t\}$.

$\phi(\pm 1) = (0, 1)$.

Otherwise $\phi(t) = (x, y)$ where

$$u = (1 - t)/(1 + t),$$

$$v = u^5 + (r^2 - 2)u^3 + u,$$

$$X = \chi(v)u,$$

$$Y = (\chi(v)v)^{(q+1)/4}\chi(v)\chi\left(u^2 + 1/c^2\right),$$

$$x = (c - 1)sX(1 + X)/Y,$$

$$y = \frac{rX - (1 + X)^2}{rX + (1 + X)^2}.$$

"Elligato

Hamburg

(restricte

Fix prim

non-squa

$A, B \in \mathbf{F}$

$A^2 - 4B$

with $\sqrt{a}$

Define E

For ψ : $\mathbf{F}$

on next

of $\psi(t)$

simplifying
k–Tibouchi:

$\in 3 + 4\mathbf{Z}$;

$\neq 0$;
$1/c$;
$-1)^2$.

$^2 = 1 + dx^2y^2$.
e Edwards curve.

$_q$) defined
only preimages
e $\{t, -t\}$.

$\phi(\pm 1) = (0, 1)$.

Otherwise $\phi(t) = (x, y)$ where

$$u = (1 - t)/(1 + t),$$

$$v = u^5 + (r^2 - 2)u^3 + u,$$

$$X = \chi(v)u,$$

$$Y = (\chi(v)v)^{(q+1)/4}\chi(v)\chi\left(u^2 + 1/c^2\right),$$

$$x = (c - 1)sX(1 + X)/Y,$$

$$y = \frac{rX - (1 + X)^2}{rX + (1 + X)^2}.$$

"Elligator 2", 201
Hamburg–Krasnov
(restricted to the e

Fix prime power q
non-square $u \in \mathbf{F}_q$
$A, B \in \mathbf{F}_q^*$ with no
$A^2 - 4B$; $\sqrt{\ } : \mathbf{F}_q^2$
with $\sqrt{a^2} \in \{a, -$

Define $E : y^2 = x$

For $\psi : \mathbf{F}_q \to E(\mathbf{F}$
on next slide: the
of $\psi(t)$ under ψ a

g
i:
;
c^2y^2.
curve.
nages

$\phi(\pm 1) = (0, 1)$.

Otherwise $\phi(t) = (x, y)$ where

$u = (1 - t)/(1 + t)$,

$v = u^5 + (r^2 - 2)u^3 + u$,

$X = \chi(v)u$,

$Y =$
$(\chi(v)v)^{(q+1)/4}\chi(v)\chi\left(u^2 + 1/c^2\right)$,

$x = (c - 1)sX(1 + X)/Y$,

$$y = \frac{rX - (1 + X)^2}{rX + (1 + X)^2}.$$

"Elligator 2", 2013 Bernstei
Hamburg–Krasnova–Lange
(restricted to the easiest cas

Fix prime power $q \in 1 + 4\mathbf{Z}$
non-square $u \in \mathbf{F}_q$;
$A, B \in \mathbf{F}_q^*$ with non-square
$A^2 - 4B$; $\sqrt{\ } : \mathbf{F}_q^2 \to \mathbf{F}_q$
with $\sqrt{a^2} \in \{a, -a\}$.

Define $E : y^2 = x^3 + Ax^2 +$

For $\psi : \mathbf{F}_q \to E(\mathbf{F}_q)$ defined
on next slide: the only preim
of $\psi(t)$ under ψ are $\{t, -t\}$.

$\phi(\pm 1) = (0, 1)$.

Otherwise $\phi(t) = (x, y)$ where

$u = (1 - t)/(1 + t)$,

$v = u^5 + (r^2 - 2)u^3 + u$,

$X = \chi(v)u$,

$Y =$

$(\chi(v)v)^{(q+1)/4}\chi(v)\chi\left(u^2 + 1/c^2\right)$,

$x = (c - 1)sX(1 + X)/Y$,

$$y = \frac{rX - (1 + X)^2}{rX + (1 + X)^2}.$$

"Elligator 2", 2013 Bernstein–Hamburg–Krasnova–Lange (restricted to the easiest case):

Fix prime power $q \in 1 + 4\mathbf{Z}$;
non-square $u \in \mathbf{F}_q$;
$A, B \in \mathbf{F}_q^*$ with non-square $A^2 - 4B$; $\sqrt{\ } : \mathbf{F}_q^2 \to \mathbf{F}_q$ with $\sqrt{a^2} \in \{a, -a\}$.

Define $E : y^2 = x^3 + Ax^2 + Bx$.

For $\psi : \mathbf{F}_q \to E(\mathbf{F}_q)$ defined on next slide: the only preimages of $\psi(t)$ under ψ are $\{t, -t\}$.

$= (0, 1)$.

se $\phi(t) = (x, y)$ where

$- t)/(1 + t)$,

$+ (r^2 - 2)u^3 + u$,

$v)u$,

$^{(q+1)/4}\chi(v)\chi\left(u^2 + 1/c^2\right)$,

$- 1)sX(1 + X)/Y$,

$\frac{- (1 + X)^2}{+ (1 + X)^2}$.

"Elligator 2", 2013 Bernstein–Hamburg–Krasnova–Lange (restricted to the easiest case):

Fix prime power $q \in 1 + 4\mathbf{Z}$;
non-square $u \in \mathbf{F}_q$;
$A, B \in \mathbf{F}_q^*$ with non-square $A^2 - 4B$; $\sqrt{\ } : \mathbf{F}_q^2 \to \mathbf{F}_q$ with $\sqrt{a^2} \in \{a, -a\}$.

Define $E : y^2 = x^3 + Ax^2 + Bx$.

For $\psi : \mathbf{F}_q \to E(\mathbf{F}_q)$ defined on next slide: the only preimages of $\psi(t)$ under ψ are $\{t, -t\}$.

$\psi(0) =$

Otherwi

$v = -A$

$\epsilon = \chi(v$

$x = \epsilon v$ -

$y = -\epsilon\sqrt{}$

Proofs, i

elligat

(x, y) where

$t)$,

$u^3 + u$,

$v)\chi(u^2 + 1/c^2)$,

$+ X)/Y$,

$)^2$
$)^2$.

"Elligator 2", 2013 Bernstein–Hamburg–Krasnova–Lange (restricted to the easiest case):

Fix prime power $q \in 1 + 4\mathbf{Z}$;
non-square $u \in \mathbf{F}_q$;
$A, B \in \mathbf{F}_q^*$ with non-square $A^2 - 4B$; $\sqrt{\ } : \mathbf{F}_q^2 \to \mathbf{F}_q$ with $\sqrt{a^2} \in \{a, -a\}$.

Define $E : y^2 = x^3 + Ax^2 + Bx$.

For $\psi : \mathbf{F}_q \to E(\mathbf{F}_q)$ defined on next slide: the only preimages of $\psi(t)$ under ψ are $\{t, -t\}$.

$\psi(0) = (0, 0)$.

Otherwise $\psi(t) =$

$v = -A/(1 + ut^2)$

$\epsilon = \chi(v^3 + Av^2 +$

$x = \epsilon v - (1 - \epsilon)A$

$y = -\epsilon\sqrt{x^3 + Ax^2}$

Proofs, inverse ma

elligator.cr.yp

ere

$1/c^2)$,

"Elligator 2", 2013 Bernstein–Hamburg–Krasnova–Lange (restricted to the easiest case):

Fix prime power $q \in 1 + 4\mathbf{Z}$;
non-square $u \in \mathbf{F}_q$;
$A, B \in \mathbf{F}_q^*$ with non-square $A^2 - 4B$; $\sqrt{\ } : \mathbf{F}_q^2 \to \mathbf{F}_q$ with $\sqrt{a^2} \in \{a, -a\}$.

Define $E : y^2 = x^3 + Ax^2 + Bx$.

For $\psi : \mathbf{F}_q \to E(\mathbf{F}_q)$ defined on next slide: the only preimages of $\psi(t)$ under ψ are $\{t, -t\}$.

$\psi(0) = (0, 0)$.

Otherwise $\psi(t) = (x, y)$ whe

$v = -A/(1 + ut^2)$,

$\epsilon = \chi(v^3 + Av^2 + Bv)$,

$x = \epsilon v - (1 - \epsilon)A/2$,

$y = -\epsilon\sqrt{x^3 + Ax^2 + Bx}$.

Proofs, inverse maps, etc.:
elligator.cr.yp.to

"Elligator 2", 2013 Bernstein–Hamburg–Krasnova–Lange (restricted to the easiest case):

Fix prime power $q \in 1 + 4\mathbf{Z}$;
non-square $u \in \mathbf{F}_q$;
$A, B \in \mathbf{F}_q^*$ with non-square $A^2 - 4B$; $\sqrt{\ } : \mathbf{F}_q^2 \to \mathbf{F}_q$ with $\sqrt{a^2} \in \{a, -a\}$.

Define $E : y^2 = x^3 + Ax^2 + Bx$.

For $\psi : \mathbf{F}_q \to E(\mathbf{F}_q)$ defined on next slide: the only preimages of $\psi(t)$ under ψ are $\{t, -t\}$.

$\psi(0) = (0, 0)$.

Otherwise $\psi(t) = (x, y)$ where

$$v = -A/(1 + ut^2),$$

$$\epsilon = \chi(v^3 + Av^2 + Bv),$$

$$x = \epsilon v - (1 - \epsilon)A/2,$$

$$y = -\epsilon\sqrt{x^3 + Ax^2 + Bx}.$$

Proofs, inverse maps, etc.:
elligator.cr.yp.to

or 2", 2013 Bernstein–
g–Krasnova–Lange
ed to the easiest case):

e power $q \in 1 + 4\mathbf{Z}$;
are $u \in \mathbf{F}_q$;
$\mathbf{F}_q^*$ with non-square
$\sqrt{\ } : \mathbf{F}_q^2 \to \mathbf{F}_q$
$\sqrt{a^2} \in \{a, -a\}$.

$E : y^2 = x^3 + Ax^2 + Bx$.

$\mathbf{F}_q \to E(\mathbf{F}_q)$ defined
slide: the only preimages
under ψ are $\{t, -t\}$.

$\psi(0) = (0, 0)$.

Otherwise $\psi(t) = (x, y)$ where

$$v = -A/(1 + ut^2),$$

$$\epsilon = \chi(v^3 + Av^2 + Bv),$$

$$x = \epsilon v - (1 - \epsilon)A/2,$$

$$y = -\epsilon\sqrt{x^3 + Ax^2 + Bx}.$$

Proofs, inverse maps, etc.:
elligator.cr.yp.to

Asympto

The orig
Index ca
in subex
ECDL at

3 Bernstein–
va–Lange
easiest case):

$\in 1 + 4\mathbf{Z}$;

;

n-square

$\to \mathbf{F}_q$

$a\}$.

$^3 + Ax^2 + Bx$.

$_q$) defined

only preimages

re $\{t, -t\}$.

$\psi(0) = (0, 0)$.

Otherwise $\psi(t) = (x, y)$ where

$$v = -A/(1 + ut^2),$$

$$\epsilon = \chi(v^3 + Av^2 + Bv),$$

$$x = \epsilon v - (1 - \epsilon)A/2,$$

$$y = -\epsilon\sqrt{x^3 + Ax^2 + Bx}.$$

Proofs, inverse maps, etc.:

elligator.cr.yp.to

Asymptotic ECDL

The original ECC

Index calculus brea

in subexponential

ECDL attack take

n–

se):

;

$+ Bx$.

nages

.

$\psi(0) = (0, 0)$.

Otherwise $\psi(t) = (x, y)$ where

$v = -A/(1 + ut^2)$,

$\epsilon = \chi(v^3 + Av^2 + Bv)$,

$x = \epsilon v - (1 - \epsilon)A/2$,

$y = -\epsilon\sqrt{x^3 + Ax^2 + Bx}$.

Proofs, inverse maps, etc.:

elligator.cr.yp.to

Asymptotic ECDL security

The original ECC advertising

Index calculus breaks RSA e

in subexponential time. Sca

ECDL attack takes exp time

$\psi(0) = (0, 0)$.

Otherwise $\psi(t) = (x, y)$ where

$v = -A/(1 + ut^2)$,

$\epsilon = \chi(v^3 + Av^2 + Bv)$,

$x = \epsilon v - (1 - \epsilon)A/2$,

$y = -\epsilon\sqrt{x^3 + Ax^2 + Bx}$.

Proofs, inverse maps, etc.:

`elligator.cr.yp.to`

Asymptotic ECDL security

The original ECC advertising:
Index calculus breaks RSA etc.
in subexponential time. Scary!
ECDL attack takes exp time.

$\psi(0) = (0, 0)$.

Otherwise $\psi(t) = (x, y)$ where

$v = -A/(1 + ut^2)$,

$\epsilon = \chi(v^3 + Av^2 + Bv)$,

$x = \epsilon v - (1 - \epsilon)A/2$,

$y = -\epsilon\sqrt{x^3 + Ax^2 + Bx}$.

Proofs, inverse maps, etc.:

elligator.cr.yp.to

Asymptotic ECDL security

The original ECC advertising:
Index calculus breaks RSA etc.
in subexponential time. Scary!
ECDL attack takes exp time.

Reasonable conjecture $\Rightarrow$
2012 Petit–Quisquater using F4
solves ECDL_2 in subexp time.
Do we throw away ECC_2?

$\psi(0) = (0, 0)$.

Otherwise $\psi(t) = (x, y)$ where

$v = -A/(1 + ut^2)$,

$\epsilon = \chi(v^3 + Av^2 + Bv)$,

$x = \epsilon v - (1 - \epsilon)A/2$,

$y = -\epsilon\sqrt{x^3 + Ax^2 + Bx}$.

Proofs, inverse maps, etc.:

elligator.cr.yp.to

Asymptotic ECDL security

The original ECC advertising:
Index calculus breaks RSA etc.
in subexponential time. Scary!
ECDL attack takes exp time.

Reasonable conjecture $\Rightarrow$
2012 Petit–Quisquater using F4
solves ECDL_2 in subexp time.
Do we throw away ECC_2?

Replace F4 with XL?
Tung Chou is investigating.

$\psi(0) = (0, 0)$.

Otherwise $\psi(t) = (x, y)$ where

$v = -A/(1 + ut^2)$,

$\epsilon = \chi(v^3 + Av^2 + Bv)$,

$x = \epsilon v - (1 - \epsilon)A/2$,

$y = -\epsilon\sqrt{x^3 + Ax^2 + Bx}$.

Proofs, inverse maps, etc.:
elligator.cr.yp.to

Asymptotic ECDL security

The original ECC advertising:
Index calculus breaks RSA etc.
in subexponential time. Scary!
ECDL attack takes exp time.

Reasonable conjecture ⇒
2012 Petit–Quisquater using F4
solves ECDL_2 in subexp time.
Do we throw away ECC_2?

Replace F4 with XL?
Tung Chou is investigating.

Replace XL with Coppersmith
to generalize ECC_2 to ECC?

$(0, 0)$.

se $\psi(t) = (x, y)$ where

$/(1 + ut^2)$,

$^3 + Av^2 + Bv)$,

$- (1 - \epsilon)A/2$,

$\sqrt{x^3 + Ax^2 + Bx}$.

inverse maps, etc.:

tor.cr.yp.to

Asymptotic ECDL security

The original ECC advertising:
Index calculus breaks RSA etc.
in subexponential time. Scary!
ECDL attack takes exp time.

Reasonable conjecture ⇒
2012 Petit–Quisquater using F4
solves ECDL_2 in subexp time.
Do we throw away ECC_2?

Replace F4 with XL?
Tung Chou is investigating.

Replace XL with Coppersmith
to generalize ECC_2 to ECC?

Concrete

Typical f
the "NIS
$y^2 = x^3$
$q = 2^{256}$
$a_6 = 41$
297800
255548

$E(\mathbf{F}_q)$ h
"NIST g
484395
279791
844080

(x, y) where

),

$+ Bv$),

$A/2$,

$\overline{x^2 + Bx}$.

aps, etc.:

p.to

Asymptotic ECDL security

The original ECC advertising:
Index calculus breaks RSA etc.
in subexponential time. Scary!
ECDL attack takes exp time.

Reasonable conjecture ⇒
2012 Petit–Quisquater using F4
solves ECDL_2 in subexp time.
Do we throw away ECC_2?

Replace F4 with XL?
Tung Chou is investigating.

Replace XL with Coppersmith
to generalize ECC_2 to ECC?

Concrete ECDL se

Typical for real-wo
the "NIST P-256"
$y^2 = x^3 - 3x + a_6$
$q = 2^{256} - 2^{224} +$
$a_6 = 4105836372$
2978004726840
2555483525631

$E(\mathbf{F}_q)$ has prime o
"NIST generator"
4843956129390
2797914202762
8440807170824

ere

Asymptotic ECDL security

The original ECC advertising:
Index calculus breaks RSA etc.
in subexponential time. Scary!
ECDL attack takes exp time.

Reasonable conjecture ⇒
2012 Petit–Quisquater using F4
solves ECDL_2 in subexp time.
Do we throw away ECC_2?

Replace F4 with XL?
Tung Chou is investigating.

Replace XL with Coppersmith
to generalize ECC_2 to ECC?

Concrete ECDL security

Typical for real-world ECC:
the "NIST P-256" curve E :
$y^2 = x^3 - 3x + a_6$ over $\mathbf{F}_q$ w
$q = 2^{256} - 2^{224} + 2^{192} + 2^{9}$
$a_6 = 4105836372515214212$
29780047268409114441015
25554835256314039467401

$E(\mathbf{F}_q)$ has prime order ℓ.
"NIST generator": $P = ($
48439561293906451759052
27979142027629495260417
84408071708240463528 6, . . .

Asymptotic ECDL security

The original ECC advertising:
Index calculus breaks RSA etc.
in subexponential time. Scary!
ECDL attack takes exp time.

Reasonable conjecture $\Rightarrow$
2012 Petit–Quisquater using F4
solves ECDL_2 in subexp time.
Do we throw away ECC_2?

Replace F4 with XL?
Tung Chou is investigating.

Replace XL with Coppersmith
to generalize ECC_2 to ECC?

Concrete ECDL security

Typical for real-world ECC:
the "NIST P-256" curve E :
$y^2 = x^3 - 3x + a_6$ over $\mathbf{F}_q$ where
$q = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$,
$a_6 = 41058363725152142129326129780047268409114441015993725554835256314039467401291$.

$E(\mathbf{F}_q)$ has prime order ℓ.
"NIST generator": $P = (48439561293906451759052585252797914202762949526041747995844080717082404635286, \ldots 9)$.

otic ECDL security

ginal ECC advertising:
lculus breaks RSA etc.
ponential time. Scary!
ttack takes exp time.

ble conjecture ⇒
tit–Quisquater using F4
CDL_2 in subexp time.
hrow away ECC_2?

F4 with XL?
nou is investigating.

XL with Coppersmith
alize ECC_2 to ECC?

Concrete ECDL security

Typical for real-world ECC:
the "NIST P-256" curve E :
$y^2 = x^3 - 3x + a_6$ over $\mathbf{F}_q$ where
$q = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$,
$a_6 = 41058363725152142129326129780047268409114441015993725554835256314039467401291$.

$E(\mathbf{F}_q)$ has prime order ℓ.
"NIST generator": $P = ($
$48439561293906451759052585252797914202762949526041747995844080717082404635286, \ldots 9)$.

Textboo

$\approx\sqrt{\pi\ell/$
compute
Negation
$\approx\sqrt{2}$ for
So $\approx 2^{12}$
to comp

security

advertising:
aks RSA etc.
time. Scary!
s exp time.

ture ⇒
uater using F4
ubexp time.
$_2$ ECC$_2$?

XL?
stigating.

Coppersmith
$_2$ to ECC?

Concrete ECDL security

Typical for real-world ECC:
the "NIST P-256" curve E :
$y^2 = x^3 - 3x + a_6$ over $\mathbf{F}_q$ where
$q = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$,
$a_6 = 41058363725152142129326129780047268409114441015993725554835256314039467401291$.

$E(\mathbf{F}_q)$ has prime order ℓ.
"NIST generator": $P = ($
$48439561293906451759052585252797914202762949526041747995844080717082404635286, \ldots 9)$.

Textbook ECDL c

$\approx\sqrt{\pi\ell/2}$ group op
compute DL in orc
Negation map gai
$\approx\sqrt{2}$ for elliptic cu
So $\approx 2^{128}$ group op
to compute P-256

g:
tc.
ry!
e.

g F4
e.

th
,

Concrete ECDL security

Typical for real-world ECC:
the "NIST P-256" curve E :
$y^2 = x^3 - 3x + a_6$ over $\mathbf{F}_q$ where
$q = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$,
$a_6 = 41058363725152142129326129780047268409114441015993725554835256314039467401291$.

$E(\mathbf{F}_q)$ has prime order ℓ.
"NIST generator": $P = (48439561293906451759052585252797914202762949526041747995844080717082404635286, \ldots 9)$.

Textbook ECDL cost analys

$\approx\sqrt{\pi\ell/2}$ group operations t
compute DL in order-ℓ grou
Negation map gains factor
$\approx\sqrt{2}$ for elliptic curves.
So $\approx 2^{128}$ group operations
to compute P-256 ECDL.

Concrete ECDL security

Typical for real-world ECC:
the "NIST P-256" curve E :
$y^2 = x^3 - 3x + a_6$ over $\mathbf{F}_q$ where
$q = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$,
$a_6 = 41058363725152142129326129780047268409114441015993725554835256314039467401291$.

$E(\mathbf{F}_q)$ has prime order ℓ.
"NIST generator": $P = ($
$48439561293906451759052585252797914202762949526041747995844080717082404635286, \ldots 9)$.

Textbook ECDL cost analysis:

$\approx\sqrt{\pi\ell/2}$ group operations to compute DL in order-ℓ group.
Negation map gains factor $\approx\sqrt{2}$ for elliptic curves.
So $\approx 2^{128}$ group operations to compute P-256 ECDL.

Concrete ECDL security

Typical for real-world ECC:
the "NIST P-256" curve E :
$y^2 = x^3 - 3x + a_6$ over $\mathbf{F}_q$ where
$q = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$,
$a_6 = 41058363725152142129326129780047268409114441015993725554835256314039467401291$.

$E(\mathbf{F}_q)$ has prime order ℓ.
"NIST generator": $P = (48439561293906451759052585252797914202762949526041747995844080717082404635286, \ldots 9)$.

Textbook ECDL cost analysis:

$\approx\sqrt{\pi\ell/2}$ group operations to compute DL in order-ℓ group.
Negation map gains factor $\approx\sqrt{2}$ for elliptic curves.
So $\approx 2^{128}$ group operations to compute P-256 ECDL.

This is the best algorithm that *cryptanalysts have published* for P-256 ECDL.

Concrete ECDL security

Typical for real-world ECC:
the "NIST P-256" curve E :
$y^2 = x^3 - 3x + a_6$ over $\mathbf{F}_q$ where
$q = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$,
$a_6 = 41058363725152142129326129780047268409114441015993725554835256314039467401291$.

$E(\mathbf{F}_q)$ has prime order ℓ.
"NIST generator": $P = (48439561293906451759052585252797914202762949526041747995844080717082404635286, \ldots 9)$.

Textbook ECDL cost analysis:

$\approx\sqrt{\pi\ell/2}$ group operations to compute DL in order-ℓ group.
Negation map gains factor $\approx\sqrt{2}$ for elliptic curves.
So $\approx 2^{128}$ group operations to compute P-256 ECDL.

This is the best algorithm that *cryptanalysts have published* for P-256 ECDL.

But is it the best algorithm that *exists*?

e ECDL security

for real-world ECC:
ST P-256" curve E :
$-3x+a_6$ over $\mathbf{F}_q$ where
$^6-2^{224}+2^{192}+2^{96}-1$,
0583637251521421293261
0472684091144410159937
8352563140394674012 91.

as prime order ℓ.
generator": $P=($
5129390645175905258525
-2027629495260417479 95
71708240463528 6, . . . 9).

Textbook ECDL cost analysis:

$\approx\sqrt{\pi\ell/2}$ group operations to compute DL in order-ℓ group.
Negation map gains factor $\approx\sqrt{2}$ for elliptic curves.
So $\approx 2^{128}$ group operations to compute P-256 ECDL.

This is the best algorithm that *cryptanalysts have published* for P-256 ECDL.

But is it the best algorithm that *exists*?

Standard
minimize
total "ti

Many re
tried and
better E

Standar
For each
each P-2
with suc
takes "ti

ecurity

orld ECC:

curve E :

$_6$ over $\mathbf{F}_q$ where

$2^{192} + 2^{96} - 1$,

5152142129326 1

9114441015993 7

4039467401291.

order ℓ.

: $P = ($

5175905258525

9526041747995

635286, . . . 9).

Textbook ECDL cost analysis:

$\approx\sqrt{\pi\ell/2}$ group operations to compute DL in order-ℓ group.

Negation map gains factor $\approx\sqrt{2}$ for elliptic curves.

So $\approx 2^{128}$ group operations to compute P-256 ECDL.

This is the best algorithm that *cryptanalysts have published* for P-256 ECDL.

But is it the best algorithm that *exists*?

Standard definition

minimize "time" (

total "time" over

Many researchers

tried and failed to

better ECDL algor

Standard conject

For each $p \in [0, 1]$

each P-256 ECDL

with success proba

takes "time" $\geq 2^{12}$

:

where
$^6 - 1$,
293261
59937
01291.

58525
47995
.9).

Textbook ECDL cost analysis:

$\approx\sqrt{\pi\ell/2}$ group operations to compute DL in order-ℓ group.
Negation map gains factor $\approx\sqrt{2}$ for elliptic curves.
So $\approx 2^{128}$ group operations to compute P-256 ECDL.

This is the best algorithm that *cryptanalysts have published* for P-256 ECDL.

But is it the best algorithm that *exists*?

Standard definition of "best"
minimize "time" (e.g., minim
total "time" over all inputs)

Many researchers have tried and failed to find better ECDL algorithms.

Standard conjecture:
For each $p \in [0,1]$,
each P-256 ECDL algorithm
with success probability $\geq p$
takes "time" $\geq 2^{128}p^{1/2}$.

Textbook ECDL cost analysis:

$\approx\sqrt{\pi\ell/2}$ group operations to compute DL in order-ℓ group.
Negation map gains factor $\approx\sqrt{2}$ for elliptic curves.
So $\approx 2^{128}$ group operations to compute P-256 ECDL.

This is the best algorithm that *cryptanalysts have published* for P-256 ECDL.

But is it the best algorithm that *exists*?

Standard definition of "best": minimize "time" (e.g., minimize total "time" over all inputs).

Many researchers have tried and failed to find better ECDL algorithms.

Standard conjecture:
For each $p \in [0,1]$,
each P-256 ECDL algorithm with success probability $\geq p$ takes "time" $\geq 2^{128}p^{1/2}$.

k ECDL cost analysis:

$\overline{2}$ group operations to
e DL in order-ℓ group.
n map gains factor
r elliptic curves.
^{8} group operations
ute P-256 ECDL.

he best algorithm that
alysts have published
6 ECDL.

the best algorithm
sts?

Standard definition of "best":
minimize "time" (e.g., minimize
total "time" over all inputs).

Many researchers have
tried and failed to find
better ECDL algorithms.

Standard conjecture:
For each $p \in [0,1]$,
each P-256 ECDL algorithm
with success probability $\geq p$
takes "time" $\geq 2^{128} p^{1/2}$.

Interlude

How mu
following

```
def pic
  if nO
    if
      i
      r
    if
    ret
if n1
  if
  ret
if n2
retur
```

ost analysis:

perations to
der-ℓ group.
ns factor
urves.
perations
ECDL.

gorithm that
e published

algorithm

Standard definition of "best":
minimize "time" (e.g., minimize
total "time" over all inputs).

Many researchers have
tried and failed to find
better ECDL algorithms.

Standard conjecture:
For each $p \in [0,1]$,
each P-256 ECDL algorithm
with success probability $\geq p$
takes "time" $\geq 2^{128} p^{1/2}$.

Interlude regarding

How much "time"
following algorithm

```
def pidigit(n0
  if n0 == 0:
    if n1 == 0
      if n2 ==
      return
    if n2 == 0
    return
  if n1 == 0:
    if n2 == 0
    return
  if n2 == 0: r
  return
```

is:

to

p.

hat

d

Standard definition of "best":
minimize "time" (e.g., minimize
total "time" over all inputs).

Many researchers have
tried and failed to find
better ECDL algorithms.

Standard conjecture:
For each $p \in [0,1]$,
each P-256 ECDL algorithm
with success probability $\geq p$
takes "time" $\geq 2^{128} p^{1/2}$.

Interlude regarding "time"

How much "time" does the
following algorithm take?

```
def pidigit(n0,n1,n2):
  if n0 == 0:
    if n1 == 0:
      if n2 == 0: retu
      return
    if n2 == 0: return
    return
  if n1 == 0:
    if n2 == 0: return
    return
  if n2 == 0: return
  return
```

Standard definition of "best":
minimize "time" (e.g., minimize
total "time" over all inputs).

Many researchers have
tried and failed to find
better ECDL algorithms.

Standard conjecture:
For each $p \in [0,1]$,
each P-256 ECDL algorithm
with success probability $\geq p$
takes "time" $\geq 2^{128} p^{1/2}$.

Interlude regarding "time"

How much "time" does the
following algorithm take?

```
def pidigit(n0,n1,n2):
  if n0 == 0:
    if n1 == 0:
      if n2 == 0: return 3
      return             1
    if n2 == 0: return   4
    return               1
  if n1 == 0:
    if n2 == 0: return   5
    return               9
  if n2 == 0: return     2
  return                 6
```

d definition of "best":
e "time" (e.g., minimize
me" over all inputs).

searchers have
d failed to find
CDL algorithms.

rd conjecture:
$p \in [0, 1]$,
256 ECDL algorithm
ccess probability $\geq p$
ime" $\geq 2^{128} p^{1/2}$.

Interlude regarding "time"

How much "time" does the following algorithm take?

```
def pidigit(n0,n1,n2):
  if n0 == 0:
    if n1 == 0:
      if n2 == 0: return 3
      return             1
    if n2 == 0: return   4
    return               1
if n1 == 0:
  if n2 == 0: return     5
  return                 9
if n2 == 0: return       2
return                   6
```

Students
learn to
Skipped

This alg

n of "best":
e.g., minimize
all inputs).

have
find
rithms.

cture:

,
algorithm
ability $\geq p$
$^{28}p^{1/2}$.

Interlude regarding "time"

How much "time" does the following algorithm take?

```
def pidigit(n0,n1,n2):
  if n0 == 0:
    if n1 == 0:
      if n2 == 0: return 3
      return             1
    if n2 == 0: return   4
    return             1
  if n1 == 0:
    if n2 == 0: return   5
    return             9
  if n2 == 0: return     2
  return               6
```

Students in algorit
learn to count exe
Skipped branches

This algorithm use

”:
mize
.

Interlude regarding “time”

How much “time” does the following algorithm take?

```
def pidigit(n0,n1,n2):
  if n0 == 0:
    if n1 == 0:
      if n2 == 0: return 3
      return             1
    if n2 == 0: return   4
    return             1
  if n1 == 0:
    if n2 == 0: return   5
    return             9
  if n2 == 0: return     2
  return               6
```

Students in algorithm course
learn to count executed “ste
Skipped branches take 0 “st

This algorithm uses 4 “steps

Interlude regarding "time"

How much "time" does the following algorithm take?

```
def pidigit(n0,n1,n2):
  if n0 == 0:
    if n1 == 0:
      if n2 == 0: return 3
      return             1
    if n2 == 0: return   4
    return             1
if n1 == 0:
  if n2 == 0: return   5
  return             9
if n2 == 0: return     2
return               6
```

Students in algorithm courses learn to count executed "steps". Skipped branches take 0 "steps".

This algorithm uses 4 "steps".

Interlude regarding "time"

How much "time" does the following algorithm take?

```
def pidigit(n0,n1,n2):
  if n0 == 0:
    if n1 == 0:
      if n2 == 0: return 3
      return             1
    if n2 == 0: return   4
    return             1
if n1 == 0:
  if n2 == 0: return   5
  return             9
if n2 == 0: return     2
return               6
```

Students in algorithm courses learn to count executed "steps". Skipped branches take 0 "steps".

This algorithm uses 4 "steps".

Generalization: There exists an algorithm that, given $n < 2^k$, prints the nth digit of π using $k + 1$ "steps".

Interlude regarding "time"

How much "time" does the following algorithm take?

```
def pidigit(n0,n1,n2):
  if n0 == 0:
    if n1 == 0:
      if n2 == 0: return 3
      return             1
    if n2 == 0: return   4
    return               1
if n1 == 0:
  if n2 == 0: return     5
  return                 9
if n2 == 0: return       2
return                   6
```

Students in algorithm courses learn to count executed "steps". Skipped branches take 0 "steps".

This algorithm uses 4 "steps".

Generalization: There exists an algorithm that, given $n < 2^k$, prints the nth digit of π using $k + 1$ "steps".

Variant: There exists a 258-"step" P-256 discrete-log attack (with 100% success probability).

Interlude regarding "time"

How much "time" does the following algorithm take?

```
def pidigit(n0,n1,n2):
  if n0 == 0:
    if n1 == 0:
      if n2 == 0: return 3
      return             1
    if n2 == 0: return   4
    return             1
if n1 == 0:
  if n2 == 0: return   5
  return             9
if n2 == 0: return     2
return               6
```

Students in algorithm courses learn to count executed "steps". Skipped branches take 0 "steps".

This algorithm uses 4 "steps".

Generalization: There exists an algorithm that, given $n < 2^k$, prints the nth digit of π using $k+1$ "steps".

Variant: There exists a 258-"step" P-256 discrete-log attack (with 100% success probability). If "time" means "steps" then the standard conjectures are wrong.

e regarding "time"

ch "time" does the
g algorithm take?

```
digit(n0,n1,n2):
0 == 0:
 n1 == 0:
if n2 == 0: return 3
return                 1
 n2 == 0: return       4
turn                   1
1 == 0:
 n2 == 0: return       5
turn                   9
2 == 0: return         2
rn                     6
```

Students in algorithm courses learn to count executed "steps". Skipped branches take 0 "steps".

This algorithm uses 4 "steps".

Generalization: There exists an algorithm that, given $n < 2^k$, prints the nth digit of π using $k + 1$ "steps".

Variant: There exists a 258-"step" P-256 discrete-log attack (with 100% success probability). If "time" means "steps" then the standard conjectures are wrong.

1994 Be
"*We say*
A is a (*t*
A runs i
makes a

g "time"

does the
n take?

```
,n1,n2):

:
 0: return 3
           1
: return   4
           1

: return   5
           9
return     2
           6
```

Students in algorithm courses
learn to count executed "steps".
Skipped branches take 0 "steps".

This algorithm uses 4 "steps".

Generalization: There exists an
algorithm that, given $n < 2^k$,
prints the nth digit of π
using $k + 1$ "steps".

Variant: There exists a 258-
"step" P-256 discrete-log attack
(with 100% success probability).
If "time" means "steps" then the
standard conjectures are wrong.

1994 Bellare–Kilia
"*We say that*
A is a (t, q)*-advers*
A runs in at most
makes at most q o

```
rn 3
   1
   4
   1
   5
   9
   2
   6
```

Students in algorithm courses
learn to count executed "steps".
Skipped branches take 0 "steps".

This algorithm uses 4 "steps".

Generalization: There exists an
algorithm that, given $n < 2^k$,
prints the nth digit of π
using $k + 1$ "steps".

Variant: There exists a 258-
"step" P-256 discrete-log attack
(with 100% success probability).
If "time" means "steps" then the
standard conjectures are wrong.

1994 Bellare–Kilian–Rogawa

"We say that
A is a (t, q)*-adversary if*
A runs in at most t *steps an*
makes at most q *queries to*

Students in algorithm courses
learn to count executed "steps".
Skipped branches take 0 "steps".

This algorithm uses 4 "steps".

Generalization: There exists an
algorithm that, given $n < 2^k$,
prints the nth digit of π
using $k + 1$ "steps".

Variant: There exists a 258-
"step" P-256 discrete-log attack
(with 100% success probability).
If "time" means "steps" then the
standard conjectures are wrong.

1994 Bellare–Kilian–Rogaway:
"*We say that*
A is a (t, q)*-adversary if*
A runs in at most t *steps and*
makes at most q *queries to* $\mathcal{O}$*.*"

Students in algorithm courses learn to count executed "steps". Skipped branches take 0 "steps".

This algorithm uses 4 "steps".

Generalization: There exists an algorithm that, given $n < 2^k$, prints the nth digit of π using $k + 1$ "steps".

Variant: There exists a 258-"step" P-256 discrete-log attack (with 100% success probability). If "time" means "steps" then the standard conjectures are wrong.

1994 Bellare–Kilian–Rogaway: "*We say that A is a (t, q)-adversary if A runs in at most t steps and makes at most q queries to $\mathcal{O}$.*"

Oops: table-lookup attack has very small t.

Paper conjectured "useful" DES security bounds. Any reasonable interpretation of conjecture was false, given paper's definition. Theorems in paper were vacuous.

s in algorithm courses
count executed "steps".
branches take 0 "steps".

orithm uses 4 "steps".

zation: There exists an
m that, given $n < 2^k$,
he nth digit of π
+ 1 "steps".

There exists a 258-
P-256 discrete-log attack
00% success probability).
" means "steps" then the
l conjectures are wrong.

1994 Bellare–Kilian–Rogaway:
"*We say that*
A is a (t, q)*-adversary if*
A runs in at most t *steps and*
makes at most q *queries to* $\mathcal{O}$*.*"

Oops: table-lookup attack
has very small t.

Paper conjectured "useful" DES
security bounds. Any reasonable
interpretation of conjecture was
false, given paper's definition.
Theorems in paper were vacuous.

2000 Be
"*We fix*
Access M
model o
running
executio
of A's d
conventi
caused [
tables . . .

thm courses
cuted "steps".
take 0 "steps".

es 4 "steps".

here exists an
ven $n < 2^k$,
it of π
".

ists a 258-
rete-log attack
ss probability).
steps" then the
res are wrong.

1994 Bellare–Kilian–Rogaway:
"*We say that*
A is a (t, q)*-adversary if*
A runs in at most t *steps and*
makes at most q *queries to* $\mathcal{O}$*.*"

Oops: table-lookup attack
has very small t.

Paper conjectured "useful" DES
security bounds. Any reasonable
interpretation of conjecture was
false, given paper's definition.
Theorems in paper were vacuous.

2000 Bellare–Kilia
"*We fix some part*
Access Machine (
model of computa
running time [mea
execution time plu
of A's description
convention elimina
caused [by] arbitra
tables . . ."

es
eps".
eps".
s".
an
k",
tack
lity).
en the
ong.

1994 Bellare–Kilian–Rogaway:
"*We say that*
A is a (t, q)*-adversary if*
A runs in at most t *steps and*
makes at most q *queries to* $\mathcal{O}$*.*"

Oops: table-lookup attack
has very small t.

Paper conjectured "useful" DES security bounds. Any reasonable interpretation of conjecture was false, given paper's definition.
Theorems in paper were vacuous.

2000 Bellare–Kilian–Rogawa
"*We fix some particular Ran*
Access Machine (RAM) as a
model of computation. . . .
running time [means] A's ac
execution time plus the leng
of A's description . . . This
convention eliminates patho
caused [by] arbitrarily large
tables . . ."

1994 Bellare–Kilian–Rogaway:
"*We say that*
A is a (t, q)*-adversary if*
A runs in at most t *steps and*
makes at most q *queries to* $\mathcal{O}$*.*"

Oops: table-lookup attack
has very small t.

Paper conjectured "useful" DES
security bounds. Any reasonable
interpretation of conjecture was
false, given paper's definition.
Theorems in paper were vacuous.

2000 Bellare–Kilian–Rogaway:
"*We fix some particular Random*
Access Machine (RAM) as a
model of computation. . . . A's
running time [means] A's actual
execution time plus the length
of A's description . . . This
convention eliminates pathologies
caused [by] arbitrarily large lookup
tables . . ."

llare–Kilian–Rogaway:
y that
t, q)-adversary if
n at most t steps and
t most q queries to $\mathcal{O}$."

able-lookup attack
small t.

onjectured "useful" DES
bounds. Any reasonable
tation of conjecture was
ven paper's definition.
ns in paper were vacuous.

2000 Bellare–Kilian–Rogaway:
"*We fix some particular Random Access Machine (RAM) as a model of computation. . . . A's running time [means] A's actual execution time plus the length of A's description . . . This convention eliminates pathologies caused [by] arbitrarily large lookup tables . . .*"

2012 Be

There ar

Assumin
overwhe
compute

There ex
algorithm
and has

"Time"

Inescapa
standar

n–Rogaway:

sary if
t steps and
queries to O."

p attack

"useful" DES
Any reasonable
onjecture was
s definition.
r were vacuous.

2000 Bellare–Kilian–Rogaway:
"*We fix some particular Random Access Machine (RAM) as a model of computation. . . . A's running time [means] A's actual execution time plus the length of A's description . . . This convention eliminates pathologies caused [by] arbitrarily large lookup tables . . .*"

2012 Bernstein–La

There are more pa

Assuming plausible
overwhelmingly ve
computer experim

There exists a P-2
algorithm that tak
and has success pr

"Time" includes a

Inescapable conclu
standard conject

y:

nd
O."

DES
nable
was
n.
uous.

2000 Bellare–Kilian–Rogaway:
"*We fix some particular Random Access Machine (RAM) as a model of computation. . . . A's running time [means] A's actual execution time plus the length of A's description . . . This convention eliminates pathologies caused [by] arbitrarily large lookup tables . . .*"

2012 Bernstein–Lange:

There are more pathologies!

Assuming plausible heuristic
overwhelmingly verified by
computer experiment:

There exists a P-256 ECDL
algorithm that takes "time"
and has success probability

"Time" includes algorithm l

Inescapable conclusion: **The**
standard conjecture is fals

2000 Bellare–Kilian–Rogaway:
"*We fix some particular Random Access Machine (RAM) as a model of computation. . . . A's running time [means] A's actual execution time plus the length of A's description . . . This convention eliminates pathologies caused [by] arbitrarily large lookup tables . . .*"

2012 Bernstein–Lange:

There are more pathologies!

Assuming plausible heuristics, overwhelmingly verified by computer experiment:

There exists a P-256 ECDL algorithm that takes "time" $\approx 2^{85}$ and has success probability ≈ 1.

"Time" includes algorithm length.

Inescapable conclusion: **The standard conjecture is false.**

llare–Kilian–Rogaway:
some particular Random
Machine (RAM) as a
f computation. . . . A's
time [means] A's actual
n time plus the length
escription . . . This
ion eliminates pathologies
[by] arbitrarily large lookup
."

2012 Bernstein–Lange:

There are more pathologies!

Assuming plausible heuristics, overwhelmingly verified by computer experiment:

There exists a P-256 ECDL algorithm that takes "time" $\approx 2^{85}$ and has success probability ≈ 1.

"Time" includes algorithm length.

Inescapable conclusion: **The standard conjecture is false.**

Our reco
the flawe
conjectu

1. Switc
to circui

(Related
Improved
batch N

n–Rogaway:
ticular Random
RAM) as a
tion. . . . A's
ns] A's actual
us the length
. . . This
ates pathologies
arily large lookup

2012 Bernstein–Lange:

There are more pathologies!

Assuming plausible heuristics, overwhelmingly verified by computer experiment:

There exists a P-256 ECDL algorithm that takes "time" $\approx 2^{85}$ and has success probability ≈ 1.

"Time" includes algorithm length.

Inescapable conclusion: **The standard conjecture is false.**

Our recommendat
the flawed security
conjectures, proofs

1. Switch from "t
to circuit AT.

(Related, online sc
Improved AT expo
batch NFS.)

y:
ndom
a
A's
ctual
rth

logies
lookup

2012 Bernstein–Lange:

There are more pathologies!

Assuming plausible heuristics, overwhelmingly verified by computer experiment:

There exists a P-256 ECDL algorithm that takes "time" $\approx 2^{85}$ and has success probability ≈ 1.

"Time" includes algorithm length.

Inescapable conclusion: **The standard conjecture is false.**

Our recommendations to fix the flawed security definition conjectures, proofs:

1. Switch from "time" to circuit AT.

(Related, online soon: Improved AT exponents for *batch* NFS.)

2012 Bernstein–Lange:

There are more pathologies!

Assuming plausible heuristics, overwhelmingly verified by computer experiment:

There exists a P-256 ECDL algorithm that takes "time" $\approx 2^{85}$ and has success probability ≈ 1.

"Time" includes algorithm length.

Inescapable conclusion: **The standard conjecture is false.**

Our recommendations to fix the flawed security definitions, conjectures, proofs:

1. Switch from "time" to circuit AT.

(Related, online soon: Improved AT exponents for *batch* NFS.)

2012 Bernstein–Lange:

There are more pathologies!

Assuming plausible heuristics, overwhelmingly verified by computer experiment:

There exists a P-256 ECDL algorithm that takes "time" $\approx 2^{85}$ and has success probability ≈ 1.

"Time" includes algorithm length.

Inescapable conclusion: **The standard conjecture is false.**

Our recommendations to fix the flawed security definitions, conjectures, proofs:

1. Switch from "time" to circuit AT.

(Related, online soon: Improved AT exponents for *batch* NFS.)

2. Formalize constructivity.

2012 Bernstein–Lange:

There are more pathologies!

Assuming plausible heuristics, overwhelmingly verified by computer experiment:

There exists a P-256 ECDL algorithm that takes "time" $\approx 2^{85}$ and has success probability ≈ 1.

"Time" includes algorithm length.

Inescapable conclusion: **The standard conjecture is false.**

Our recommendations to fix the flawed security definitions, conjectures, proofs:

1. Switch from "time" to circuit AT.

(Related, online soon: Improved AT exponents for *batch* NFS.)

2. Formalize constructivity.

More details and attacks:

cr.yp.to/nonuniform.html

rnstein–Lange:

re more pathologies!

g plausible heuristics,
lmingly verified by
er experiment:

kists a P-256 ECDL
m that takes "time" $\approx 2^{85}$
success probability ≈ 1.

includes algorithm length.

able conclusion: **The**
d conjecture is false.

Our recommendations to fix
the flawed security definitions,
conjectures, proofs:

1. Switch from "time"
to circuit AT.

(Related, online soon:
Improved AT exponents for
batch NFS.)

2. Formalize constructivity.

More details and attacks:
cr.yp.to/nonuniform.html

DH spee

Sandy B
security
("?" if r

2011 Be
Schwabe
2012 Ha
2012 Lo
2013 Bo
Lauter:
2013 Oli
Rodrígue
2013 Faz
Sánchez
2014 Be
Lange–S

ange:

thologies!

e heuristics,
rified by
ent:

256 ECDL
kes "time" $\approx 2^{85}$
robability ≈ 1.

lgorithm length.

usion: **The**
ure is false.

Our recommendations to fix
the flawed security definitions,
conjectures, proofs:

1. Switch from "time"
to circuit AT.

(Related, online soon:
Improved AT exponents for
batch NFS.)

2. Formalize constructivity.

More details and attacks:
cr.yp.to/nonuniform.html

DH speed records

Sandy Bridge cycl
security constant-t
("?" if not SUPE

2011 Bernstein–D
Schwabe–Yang:
2012 Hamburg:
2012 Longa–Sica:
2013 Bos–Costello
Lauter:
2013 Oliveira–Lóp
Rodríguez-Henríqu
2013 Faz-Hernánd
Sánchez:
2014 Bernstein–C
Lange–Schwabe:

s,

$\approx 2^{85}$

$\approx 1.$

ength.

e

se.

Our recommendations to fix the flawed security definitions, conjectures, proofs:

1. Switch from "time" to circuit AT.

(Related, online soon: Improved AT exponents for *batch* NFS.)

2. Formalize constructivity.

More details and attacks:

cr.yp.to/nonuniform.html

DH speed records

Sandy Bridge cycles for high security constant-time a, P ("?" if not SUPERCOP-ver

2011 Bernstein–Duif–Lange–
Schwabe–Yang: 19
2012 Hamburg: 15
2012 Longa–Sica: 13
2013 Bos–Costello–Hisil–
Lauter: 12
2013 Oliveira–López–Aranha
Rodríguez-Henríquez: 11
2013 Faz-Hernández–Longa-
Sánchez: 9
2014 Bernstein–Chuengsatia
Lange–Schwabe: 9

Our recommendations to fix the flawed security definitions, conjectures, proofs:

1. Switch from "time" to circuit AT.

(Related, online soon: Improved AT exponents for *batch* NFS.)

2. Formalize constructivity.

More details and attacks:

cr.yp.to/nonuniform.html

DH speed records

Sandy Bridge cycles for high-security constant-time $a, P \mapsto aP$ ("?" if not SUPERCOP-verified):

2011 Bernstein–Duif–Lange–Schwabe–Yang:	194120
2012 Hamburg:	153000?
2012 Longa–Sica:	137000?
2013 Bos–Costello–Hisil–Lauter:	122728
2013 Oliveira–López–Aranha–Rodríguez-Henríquez:	114800?
2013 Faz-Hernández–Longa–Sánchez:	96000?
2014 Bernstein–Chuengsatiansup–Lange–Schwabe:	91460

ommendations to fix
ed security definitions,
res, proofs:

ch from "time"
t AT.

, online soon:
d AT exponents for
FS.)

alize constructivity.

tails and attacks:
to/nonuniform.html

DH speed records

Sandy Bridge cycles for high-security constant-time a, $P \mapsto aP$ ("?" if not SUPERCOP-verified):

2011 Bernstein–Duif–Lange–Schwabe–Yang:	194120
2012 Hamburg:	153000?
2012 Longa–Sica:	137000?
2013 Bos–Costello–Hisil–Lauter:	122728
2013 Oliveira–López–Aranha–Rodríguez-Henríquez:	114800?
2013 Faz-Hernández–Longa–Sánchez:	96000?
2014 Bernstein–Chuengsatiansup–Lange–Schwabe:	91460

Critical f

1986 Ch
tradition
allows fa
14**M** for

2006 Ga
25**M** for
$\mapsto X(2$
6**M** by s

2012 Ga
1000000
found se
surface c

ions to fix
y definitions,
s:

ime"

oon:
onents for

tructivity.

attacks:
iform.html

DH speed records

Sandy Bridge cycles for high-security constant-time $a, P \mapsto aP$ ("?" if not SUPERCOP-verified):

2011 Bernstein–Duif–Lange–Schwabe–Yang:	194120
2012 Hamburg:	153000?
2012 Longa–Sica:	137000?
2013 Bos–Costello–Hisil–Lauter:	122728
2013 Oliveira–López–Aranha–Rodríguez-Henríquez:	114800?
2013 Faz-Hernández–Longa–Sánchez:	96000?
2014 Bernstein–Chuengsatiansup–Lange–Schwabe:	91460

Critical for 122728

1986 Chudnovsky-
traditional Kumme
allows fast scalar
14**M** for $X(P) \mapsto$

2006 Gaudry: even
25**M** for $X(P), X($
$\mapsto X(2P), X(Q +$
6**M** by surface coe

2012 Gaudry–Scho
1000000-CPU-hou
found secure small
surface over $\mathbf{F}_{2^{127}}$

ns,

ml

DH speed records

Sandy Bridge cycles for high-security constant-time a, $P \mapsto aP$ ("?" if not SUPERCOP-verified):

2011 Bernstein–Duif–Lange–Schwabe–Yang:	194120
2012 Hamburg:	153000?
2012 Longa–Sica:	137000?
2013 Bos–Costello–Hisil–Lauter:	122728
2013 Oliveira–López–Aranha–Rodríguez-Henríquez:	114800?
2013 Faz-Hernández–Longa–Sánchez:	96000?
2014 Bernstein–Chuengsatiansup–Lange–Schwabe:	91460

Critical for 122728, 91460:

1986 Chudnovsky–Chudnovs traditional Kummer surface allows fast scalar mult. 14**M** for $X(P) \mapsto X(2P)$.

2006 Gaudry: even faster. 25**M** for $X(P), X(Q), X(Q \mapsto X(2P), X(Q+P)$, inclu 6**M** by surface coefficients.

2012 Gaudry–Schost: 1000000-CPU-hour computa found secure small-coefficien surface over $\mathbf{F}_{2^{127}-1}$.

DH speed records

Sandy Bridge cycles for high-security constant-time $a, P \mapsto aP$ ("?" if not SUPERCOP-verified):

2011 Bernstein–Duif–Lange–Schwabe–Yang:	194120
2012 Hamburg:	153000?
2012 Longa–Sica:	137000?
2013 Bos–Costello–Hisil–Lauter:	122728
2013 Oliveira–López–Aranha–Rodríguez-Henríquez:	114800?
2013 Faz-Hernández–Longa–Sánchez:	96000?
2014 Bernstein–Chuengsatiansup–Lange–Schwabe:	91460

Critical for 122728, 91460:

1986 Chudnovsky–Chudnovsky: traditional Kummer surface allows fast scalar mult. 14**M** for $X(P) \mapsto X(2P)$.

2006 Gaudry: even faster. 25**M** for $X(P), X(Q), X(Q-P) \mapsto X(2P), X(Q+P)$, including 6**M** by surface coefficients.

2012 Gaudry–Schost: 1000000-CPU-hour computation found secure small-coefficient surface over $\mathbf{F}_{2^{127}-1}$.

ed records

Bridge cycles for high-
constant-time a, $P \mapsto aP$
not SUPERCOP-verified):

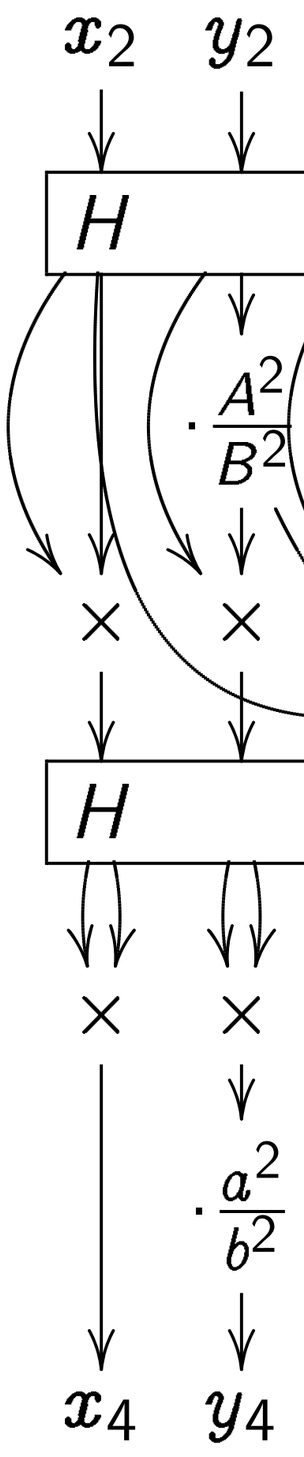
rnstein–Duif–Lange–
e–Yang: 194120
mburg: 153000?
nga–Sica: 137000?
s–Costello–Hisil–
122728
iveira–López–Aranha–
ez-Henríquez: 114800?
z-Hernández–Longa–
: 96000?
rnstein–Chuengsatiansup–
Schwabe: 91460

Critical for 122728, 91460:

1986 Chudnovsky–Chudnovsky: traditional Kummer surface allows fast scalar mult. 14**M** for $X(P) \mapsto X(2P)$.

2006 Gaudry: even faster. 25**M** for $X(P), X(Q), X(Q-P) \mapsto X(2P), X(Q+P)$, including 6**M** by surface coefficients.

2012 Gaudry–Schost: 1000000-CPU-hour computation found secure small-coefficient surface over $\mathbf{F}_{2^{127}-1}$.



es for high-

time a, $P \mapsto aP$

RCOP-verified):

uif–Lange–

194120

153000?

137000?

–Hisil–

122728

ez–Aranha–

uez: 114800?

ez–Longa–

96000?

huengsatiansup–

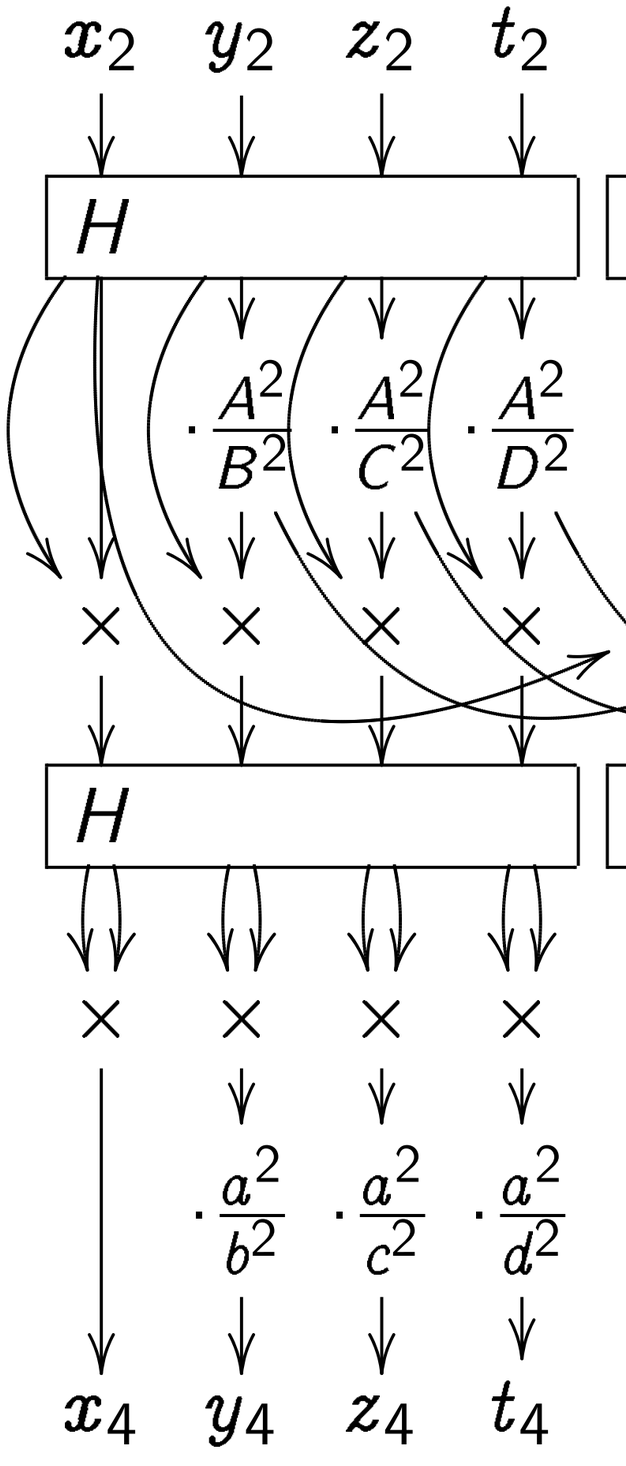
91460

Critical for 122728, 91460:

1986 Chudnovsky–Chudnovsky: traditional Kummer surface allows fast scalar mult. 14**M** for $X(P) \mapsto X(2P)$.

2006 Gaudry: even faster. 25**M** for $X(P), X(Q), X(Q-P) \mapsto X(2P), X(Q+P)$, including 6**M** by surface coefficients.

2012 Gaudry–Schost: 1000000-CPU-hour computation found secure small-coefficient surface over $\mathbf{F}_{2^{127}-1}$.



n-
$\longmapsto aP$
ified):
–
94120
53000?
37000?

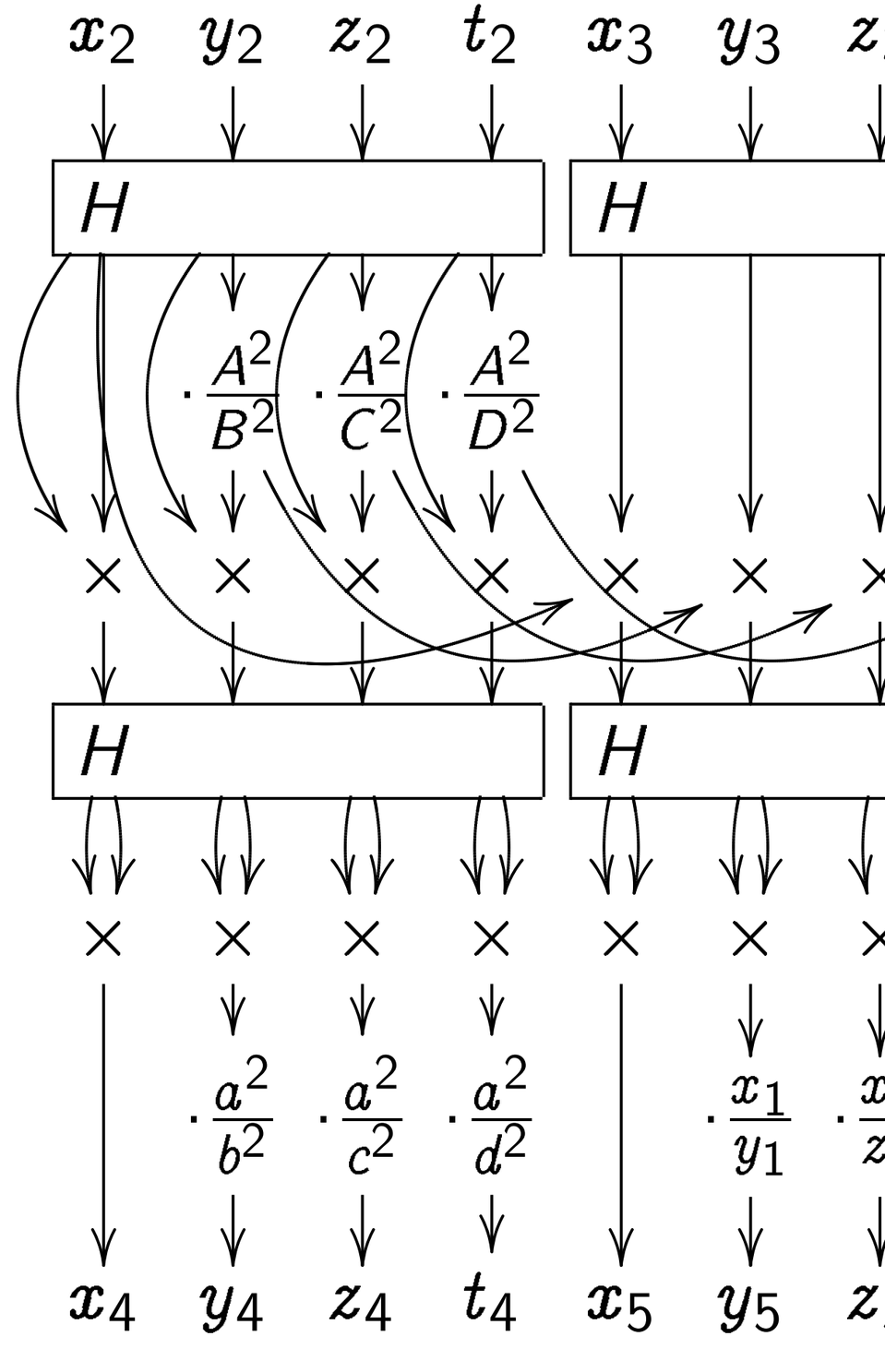
22728
a–
14800?
–
96000?
nsup–
91460

Critical for 122728, 91460:

1986 Chudnovsky–Chudnovsky: traditional Kummer surface allows fast scalar mult. 14**M** for $X(P) \mapsto X(2P)$.

2006 Gaudry: even faster. 25**M** for $X(P), X(Q), X(Q-P) \mapsto X(2P), X(Q+P)$, including 6**M** by surface coefficients.

2012 Gaudry–Schost: 1000000-CPU-hour computation found secure small-coefficient surface over $\mathbf{F}_{2^{127}-1}$.

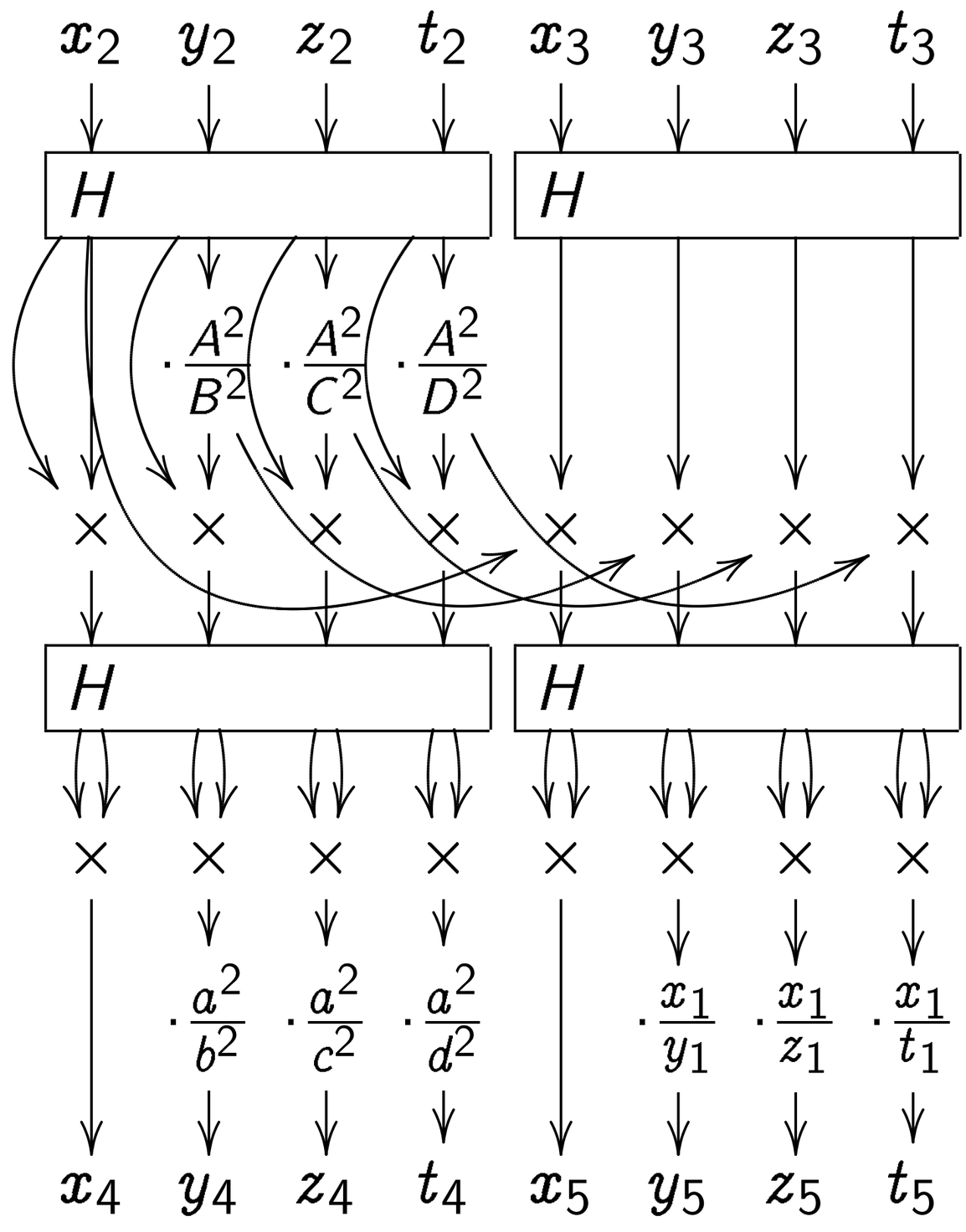


Critical for 122728, 91460:

1986 Chudnovsky–Chudnovsky:
traditional Kummer surface
allows fast scalar mult.
14**M** for $X(P) \mapsto X(2P)$.

2006 Gaudry: even faster.
25**M** for $X(P), X(Q), X(Q-P)$
$\mapsto X(2P), X(Q+P)$, including
6**M** by surface coefficients.

2012 Gaudry–Schost:
1000000-CPU-hour computation
found secure small-coefficient
surface over $\mathbf{F}_{2^{127}-1}$.

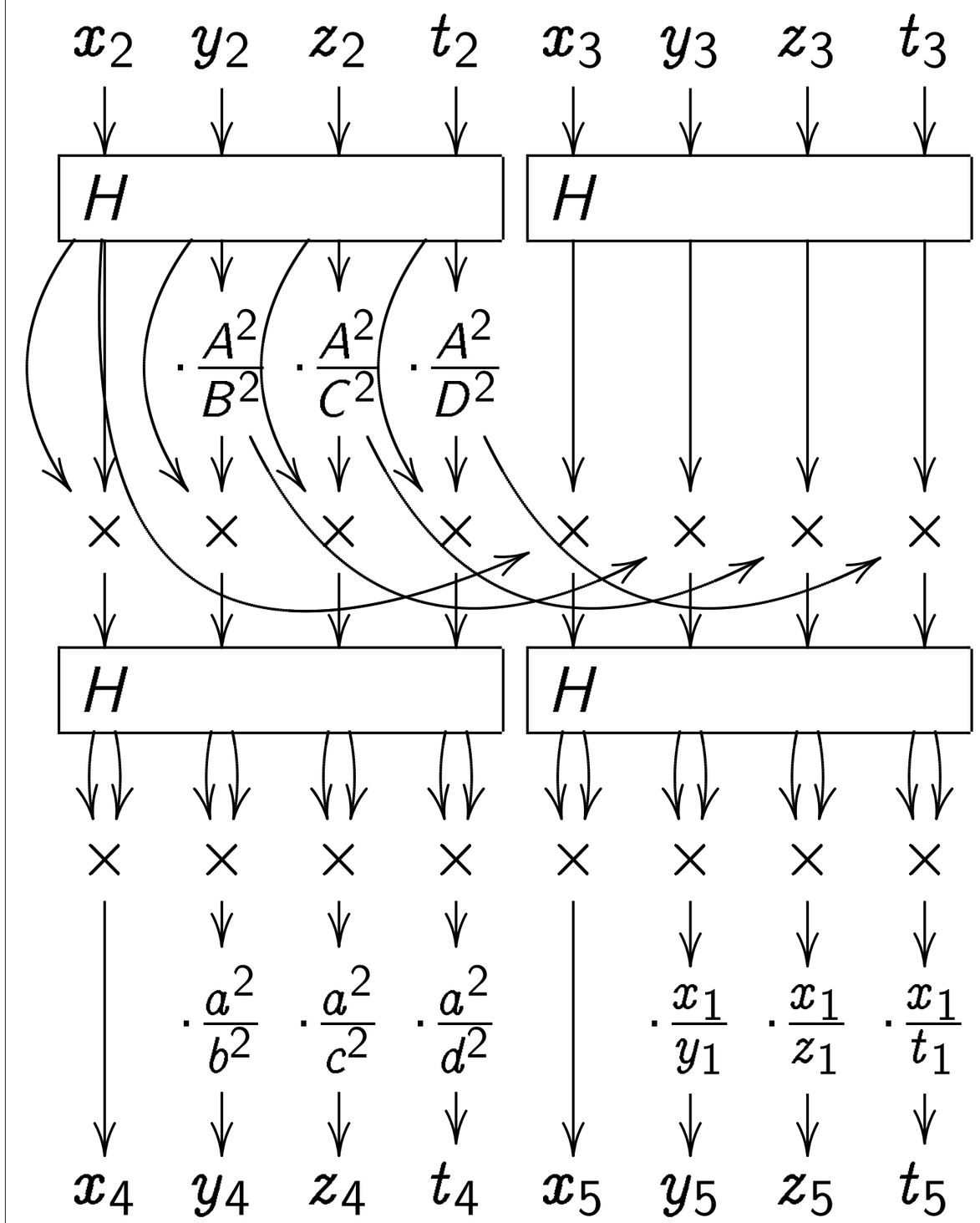


for 122728, 91460:

udnovsky–Chudnovsky:
nal Kummer surface
ast scalar mult.
$X(P) \mapsto X(2P)$.

udry: even faster.
$X(P), X(Q), X(Q-P)$
$P), X(Q+P)$, including
surface coefficients.

udry–Schost:
-CPU-hour computation
ecure small-coefficient
over $\mathbf{F}_{2^{127}-1}$.



Hyper-a

Typical
$y^2 = (z$
$(z$
over $\mathbf{F}_p$
$J =$ Jac
surface
Small K

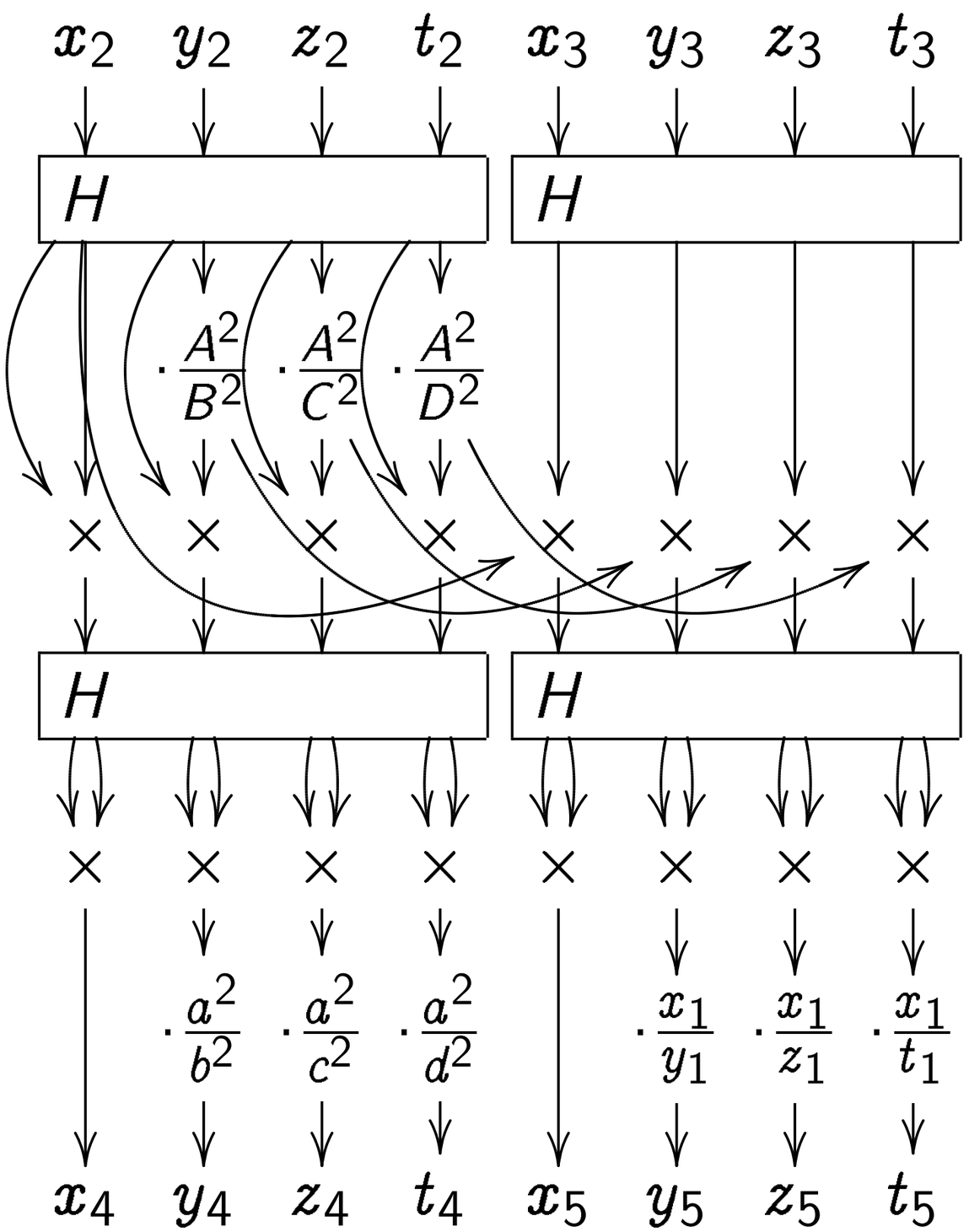
3, 91460:

-Chudnovsky:
er surface
mult.
$X(2P)$.

n faster.
$(Q), X(Q-P)$
$-P)$, including
efficients.

ost:
ur computation
l-coefficient

-1.



Hyper-and-elliptic-

Typical example:
$y^2 = (z-1)(z+$
$(z-1/2)(z+$
over $\mathbf{F}_p$ with $p =$
$J = \operatorname{Jac} H$; traditi
surface K; traditi
Small K coeffs (2

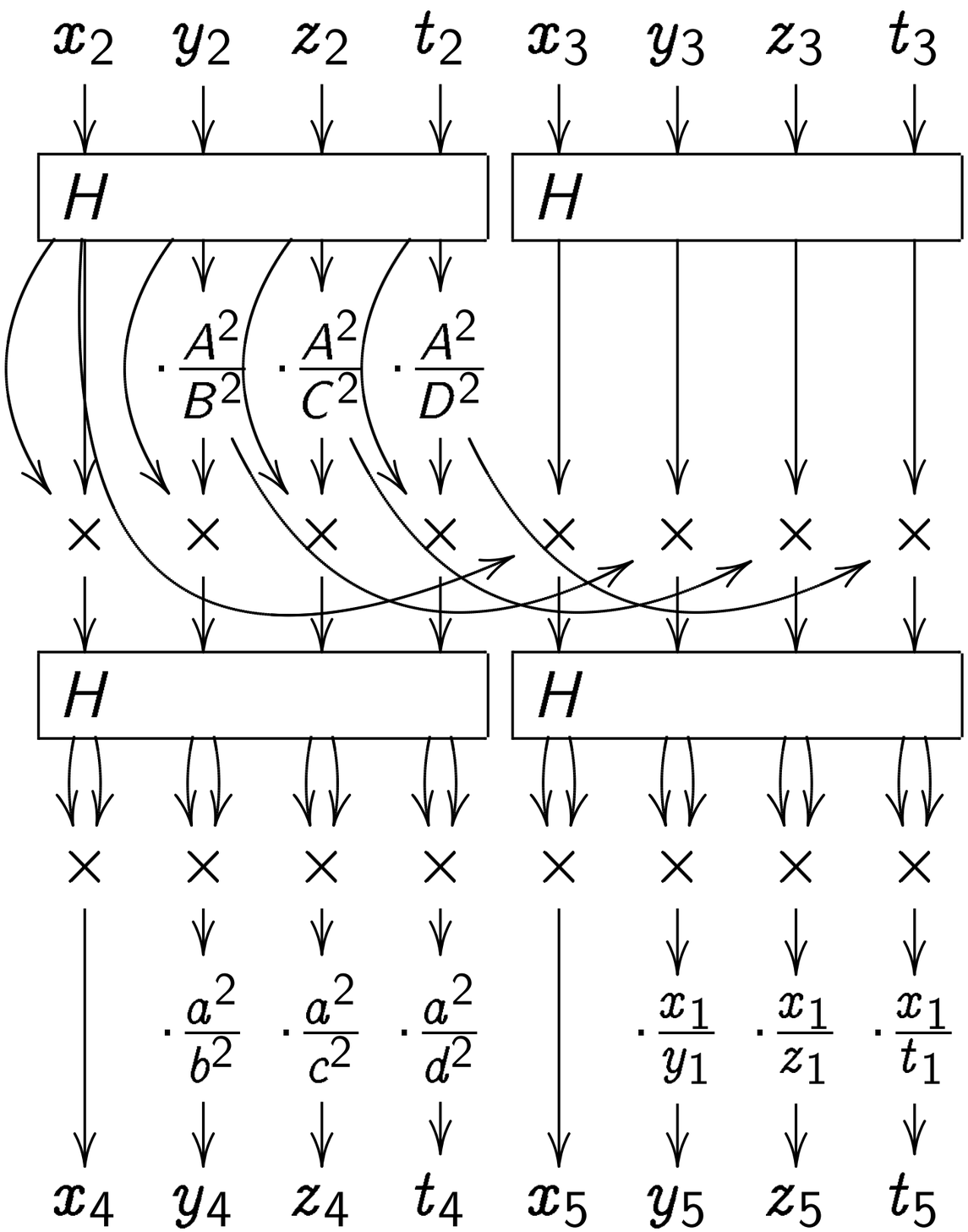
sky:

$-P$)

ding

ation

nt



Hyper-and-elliptic-curve cryp

Typical example: Define H :

$y^2 = (z-1)(z+1)(z+2)$

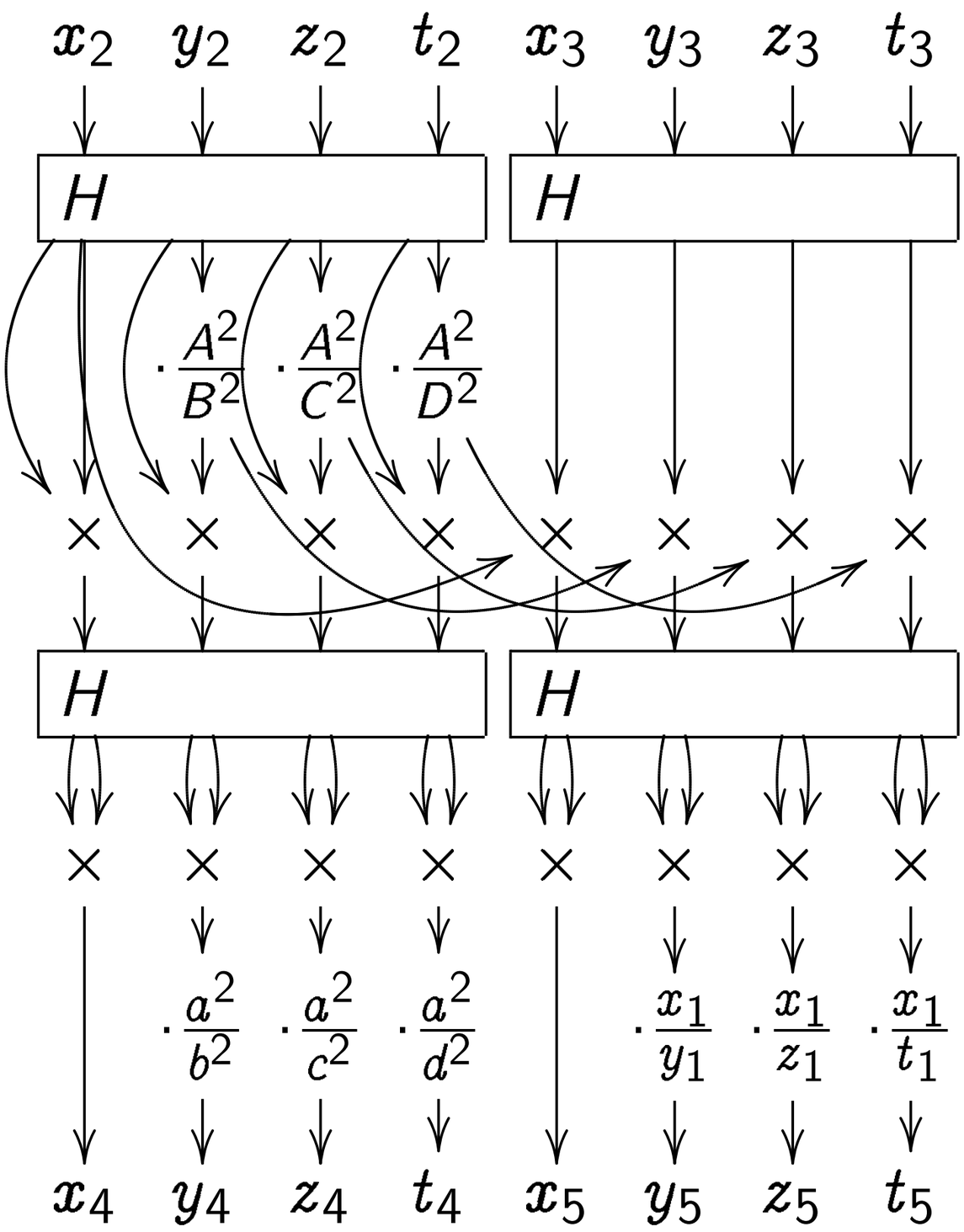
$(z-1/2)(z+3/2)(z-$

over $\mathbf{F}_p$ with $p = 2^{127} - 309$

$J = \operatorname{Jac} H$; traditional Kumm

surface K; traditional $X : J$

Small K coeffs $(20 : 1 : 20 :$



Hyper-and-elliptic-curve crypto

Typical example: Define H :

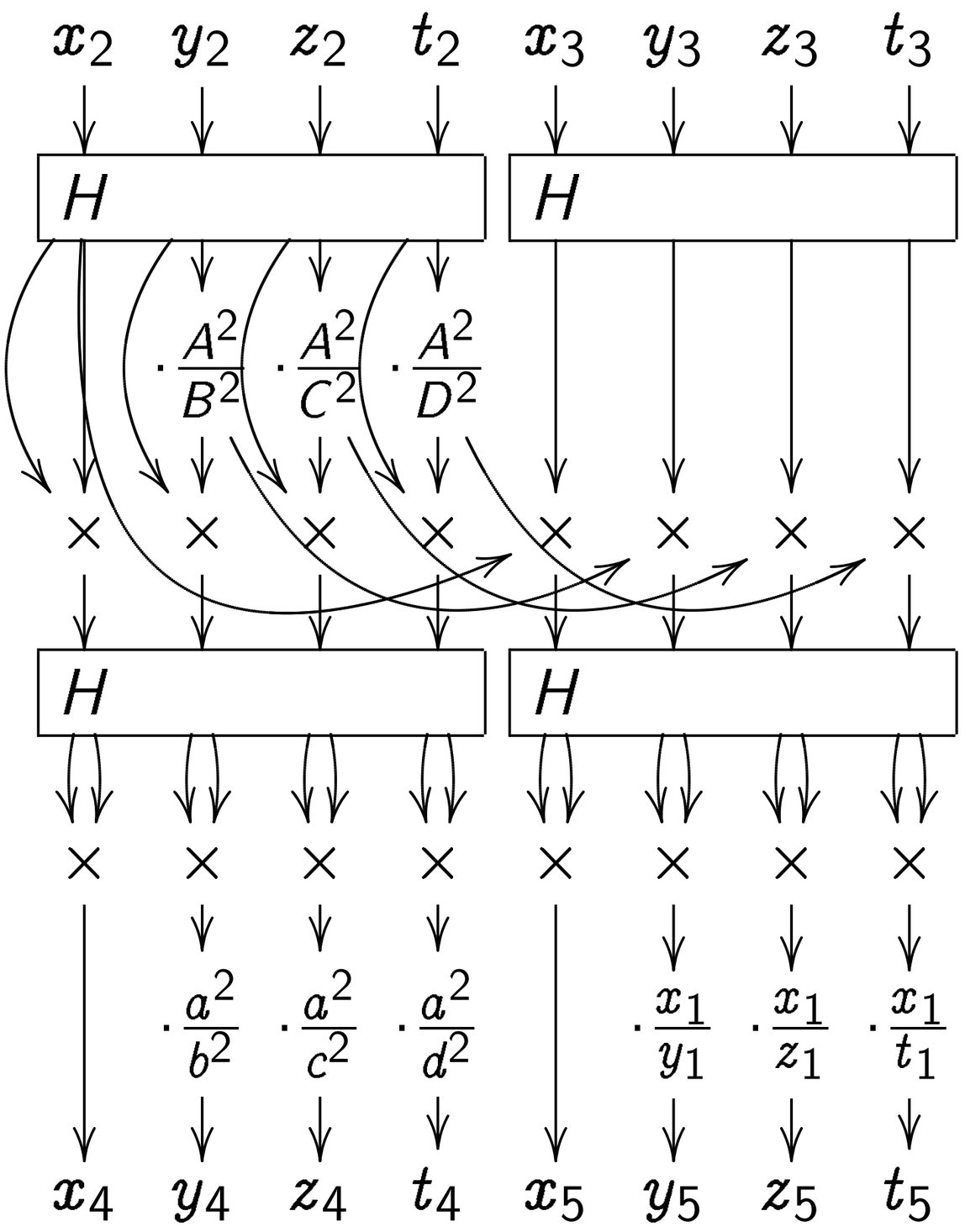
$y^2 = (z-1)(z+1)(z+2)$

$\qquad (z-1/2)(z+3/2)(z-2/3)$

over $\mathbf{F}_p$ with $p = 2^{127} - 309$;

$J = \operatorname{Jac} H$; traditional Kummer surface K; traditional $X : J \to K$.

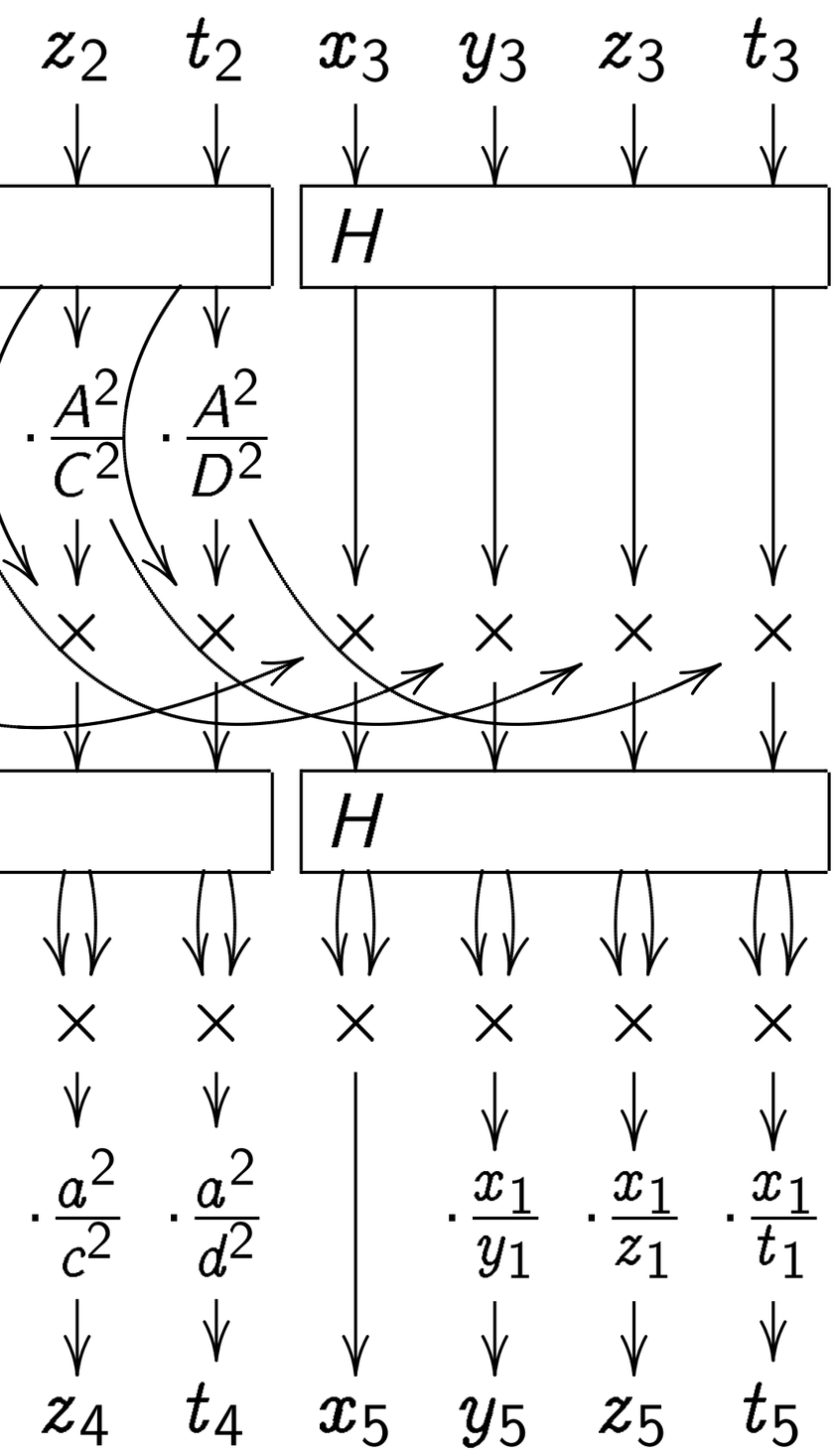
Small K coeffs $(20 : 1 : 20 : 40)$.



Hyper-and-elliptic-curve crypto

Typical example: Define H :
$y^2 = (z-1)(z+1)(z+2)$
$(z-1/2)(z+3/2)(z-2/3)$
over $\mathbf{F}_p$ with $p = 2^{127} - 309$;
$J = \operatorname{Jac} H$; traditional Kummer surface K; traditional $X : J \to K$.
Small K coeffs $(20 : 1 : 20 : 40)$.

Warning: There are errors in the Rosenhain/Mumford/Kummer formulas in 2007 Gaudry, 2010 Cosset, 2013 Bos–Costello–Hisil–Lauter. We have simpler, computer-verified formulas.

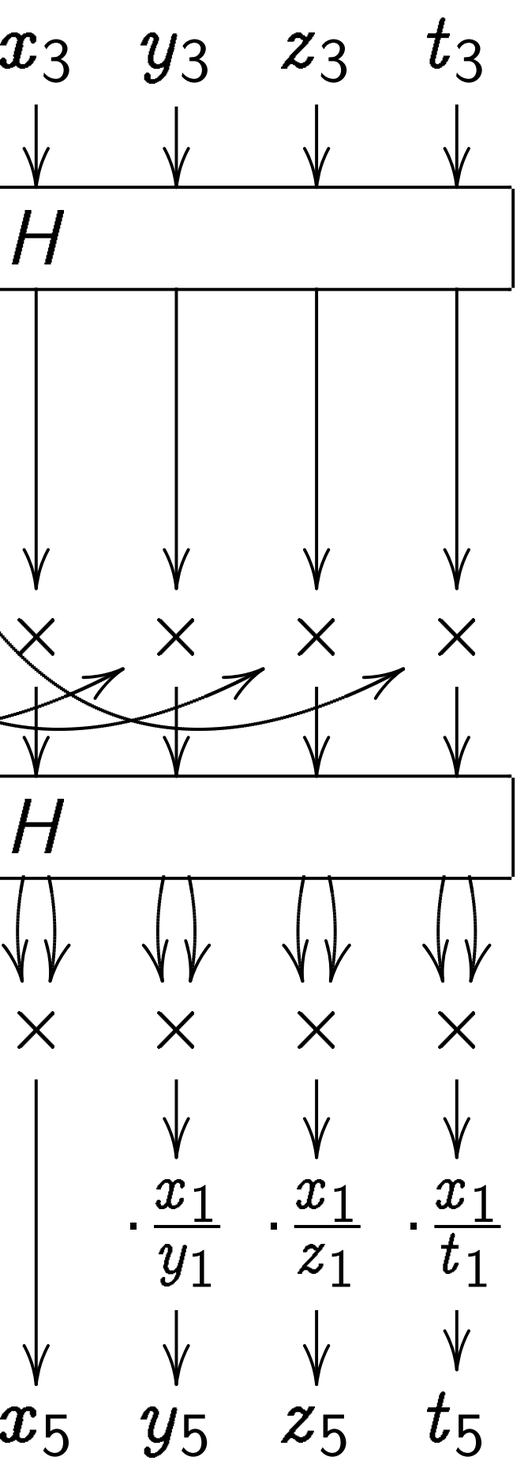


Hyper-and-elliptic-curve crypto

Typical example: Define H :
$y^2 = (z-1)(z+1)(z+2)$
$\quad (z-1/2)(z+3/2)(z-2/3)$
over $\mathbf{F}_p$ with $p = 2^{127} - 309$;
$J = \operatorname{Jac} H$; traditional Kummer surface K; traditional $X : J \to K$.
Small K coeffs $(20 : 1 : 20 : 40)$.

Warning: There are errors in the Rosenhain/Mumford/Kummer formulas in 2007 Gaudry, 2010 Cosset, 2013 Bos–Costello–Hisil–Lauter. We have simpler, computer-verified formulas.

Define **F**
$r = (7 +$
$s = 159$
$C : y^2 =$

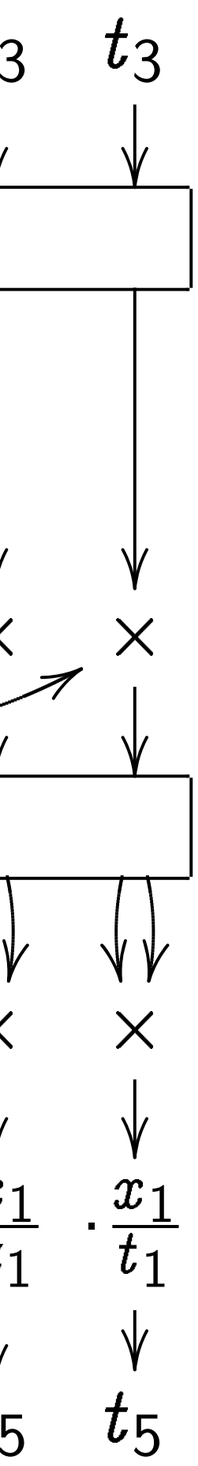


Hyper-and-elliptic-curve crypto

Typical example: Define H :
$y^2 = (z-1)(z+1)(z+2)$
$\quad (z-1/2)(z+3/2)(z-2/3)$
over $\mathbf{F}_p$ with $p = 2^{127} - 309$;
$J = \operatorname{Jac} H$; traditional Kummer surface K; traditional $X : J \to K$.
Small K coeffs $(20 : 1 : 20 : 40)$.

Warning: There are errors in the Rosenhain/Mumford/Kummer formulas in 2007 Gaudry, 2010 Cosset, 2013 Bos–Costello–Hisil–Lauter. We have simpler, computer-verified formulas.

Define $\mathbf{F}_{p^2} = \mathbf{F}_p[i$
$r = (7+4i)^2 = 3$
$s = 159 + 56i$; ω =
$C : y^2 = rx^6 + sx$



Hyper-and-elliptic-curve crypto

Typical example: Define H :
$y^2 = (z-1)(z+1)(z+2)$
$(z-1/2)(z+3/2)(z-2/3)$
over $\mathbf{F}_p$ with $p = 2^{127} - 309$;
$J = \operatorname{Jac} H$; traditional Kummer surface K; traditional $X : J \to K$.
Small K coeffs $(20 : 1 : 20 : 40)$.

Warning: There are errors in the Rosenhain/Mumford/Kummer formulas in 2007 Gaudry, 2010 Cosset, 2013 Bos–Costello–Hisil–Lauter. We have simpler, computer-verified formulas.

Define $\mathbf{F}_{p^2} = \mathbf{F}_p[i]/(i^2+1)$
$r = (7+4i)^2 = 33+56i$;
$s = 159+56i$; $\omega = \sqrt{-384}$
$C : y^2 = rx^6 + sx^4 + \bar{s}x^2 +$

Hyper-and-elliptic-curve crypto

Typical example: Define H :
$y^2 = (z-1)(z+1)(z+2)$
$(z-1/2)(z+3/2)(z-2/3)$
over $\mathbf{F}_p$ with $p = 2^{127} - 309$;
$J = \operatorname{Jac} H$; traditional Kummer surface K; traditional $X : J \to K$.
Small K coeffs $(20 : 1 : 20 : 40)$.

Warning: There are errors in the Rosenhain/Mumford/Kummer formulas in 2007 Gaudry, 2010 Cosset, 2013 Bos–Costello–Hisil–Lauter. We have simpler, computer-verified formulas.

Define $\mathbf{F}_{p^2} = \mathbf{F}_p[i]/(i^2+1)$;
$r = (7+4i)^2 = 33+56i$;
$s = 159+56i$; $\omega = \sqrt{-384}$;
$C : y^2 = rx^6 + sx^4 + \overline{s}x^2 + \overline{r}$.

Hyper-and-elliptic-curve crypto

Typical example: Define H :
$y^2 = (z-1)(z+1)(z+2)$
$(z-1/2)(z+3/2)(z-2/3)$
over $\mathbf{F}_p$ with $p = 2^{127} - 309$;
$J = \operatorname{Jac} H$; traditional Kummer surface K; traditional $X : J \to K$.
Small K coeffs $(20 : 1 : 20 : 40)$.

Warning: There are errors in the Rosenhain/Mumford/Kummer formulas in 2007 Gaudry, 2010 Cosset, 2013 Bos–Costello–Hisil–Lauter. We have simpler, computer-verified formulas.

Define $\mathbf{F}_{p^2} = \mathbf{F}_p[i]/(i^2+1)$;
$r = (7+4i)^2 = 33+56i$;
$s = 159+56i$; $\omega = \sqrt{-384}$;
$C : y^2 = rx^6 + sx^4 + \overline{s}x^2 + \overline{r}$.

$(x, y) \mapsto (x^2, y)$ takes C to E :
$y^2 = rx^3 + sx^2 + \overline{s}x + \overline{r}$.

Hyper-and-elliptic-curve crypto

Typical example: Define H :
$y^2 = (z-1)(z+1)(z+2)$
$(z-1/2)(z+3/2)(z-2/3)$
over $\mathbf{F}_p$ with $p = 2^{127} - 309$;
$J = \operatorname{Jac} H$; traditional Kummer surface K; traditional $X : J \to K$.
Small K coeffs $(20 : 1 : 20 : 40)$.

Warning: There are errors in the Rosenhain/Mumford/Kummer formulas in 2007 Gaudry, 2010 Cosset, 2013 Bos–Costello–Hisil–Lauter. We have simpler, computer-verified formulas.

Define $\mathbf{F}_{p^2} = \mathbf{F}_p[i]/(i^2+1)$;
$r = (7+4i)^2 = 33+56i$;
$s = 159+56i$; $\omega = \sqrt{-384}$;
$C : y^2 = rx^6 + sx^4 + \overline{s}x^2 + \overline{r}$.

$(x, y) \mapsto (x^2, y)$ takes C to E :
$y^2 = rx^3 + sx^2 + \overline{s}x + \overline{r}$.

$(x, y) \mapsto (1/x^2, y/x^3)$ takes C to
$y^2 = \overline{r}x^3 + \overline{s}x^2 + sx + r$.

Hyper-and-elliptic-curve crypto

Typical example: Define H :
$y^2 = (z-1)(z+1)(z+2)$
$\qquad (z-1/2)(z+3/2)(z-2/3)$
over $\mathbf{F}_p$ with $p = 2^{127} - 309$;
$J = \operatorname{Jac} H$; traditional Kummer surface K; traditional $X : J \to K$.
Small K coeffs $(20 : 1 : 20 : 40)$.

Warning: There are errors in the Rosenhain/Mumford/Kummer formulas in 2007 Gaudry, 2010 Cosset, 2013 Bos–Costello–Hisil–Lauter. We have simpler, computer-verified formulas.

Define $\mathbf{F}_{p^2} = \mathbf{F}_p[i]/(i^2+1)$;
$r = (7+4i)^2 = 33+56i$;
$s = 159+56i$; $\omega = \sqrt{-384}$;
$C : y^2 = rx^6 + sx^4 + \overline{s}x^2 + \overline{r}$.

$(x, y) \mapsto (x^2, y)$ takes C to E :
$y^2 = rx^3 + sx^2 + \overline{s}x + \overline{r}$.

$(x, y) \mapsto (1/x^2, y/x^3)$ takes C to
$y^2 = \overline{r}x^3 + \overline{s}x^2 + sx + r$.

$$(z, y) \mapsto \left(\frac{1+iz}{1-iz}, \frac{\omega y}{(1-iz)^3} \right)$$

takes H over $\mathbf{F}_{p^2}$ to C.

nd-elliptic-curve crypto

example: Define H :
$-1)(z+1)(z+2)$
$-1/2)(z+3/2)(z-2/3)$
with $p = 2^{127} - 309$;
H; traditional Kummer
K; traditional $X : J \to K$.
coeffs $(20 : 1 : 20 : 40)$.

: There are errors in the
in/Mumford/Kummer
s in 2007 Gaudry, 2010
2013 Bos–Costello–
uter. We have simpler,
er-verified formulas.

Define $\mathbf{F}_{p^2} = \mathbf{F}_p[i]/(i^2+1)$;
$r = (7+4i)^2 = 33+56i$;
$s = 159+56i$; $\omega = \sqrt{-384}$;
$C : y^2 = rx^6 + sx^4 + \overline{s}x^2 + \overline{r}$.

$(x, y) \mapsto (x^2, y)$ takes C to E :
$y^2 = rx^3 + sx^2 + \overline{s}x + \overline{r}$.

$(x, y) \mapsto (1/x^2, y/x^3)$ takes C to
$y^2 = \overline{r}x^3 + \overline{s}x^2 + sx + r$.

$$(z, y) \mapsto \left(\frac{1+iz}{1-iz}, \frac{\omega y}{(1-iz)^3} \right)$$

takes H over $\mathbf{F}_{p^2}$ to C.

J is isog
Weil res
computi
Here $\#J$
also reas

-curve crypto

Define H :

$1)(z+2)$

$+3/2)(z-2/3)$

$2^{127}-309$;

onal Kummer

onal $X : J \to K$.

$0:1:20:40)$.

re errors in the

ord/Kummer

Gaudry, 2010

-Costello–

have simpler,

formulas.

Define $\mathbf{F}_{p^2} = \mathbf{F}_p[i]/(i^2+1)$;
$r = (7+4i)^2 = 33+56i$;
$s = 159+56i$; $\omega = \sqrt{-384}$;
$C : y^2 = rx^6 + sx^4 + \overline{s}x^2 + \overline{r}$.

$(x,y) \mapsto (x^2,y)$ takes C to E :
$y^2 = rx^3 + sx^2 + \overline{s}x + \overline{r}$.

$(x,y) \mapsto (1/x^2, y/x^3)$ takes C to
$y^2 = \overline{r}x^3 + \overline{s}x^2 + sx + r$.

$$(z,y) \mapsto \left(\frac{1+iz}{1-iz}, \frac{\omega y}{(1-iz)^3}\right)$$

takes H over $\mathbf{F}_{p^2}$ to C.

J is isogenous to
Weil restriction W
computing $\#J(\mathbf{F}_p$
Here $\#J(\mathbf{F}_p) = 16$
also reasonably tw

pto

- 2/3)
9;
mer
$\to K$.
40).

n the
ner
010

ler,

Define $\mathbf{F}_{p^2} = \mathbf{F}_p[i]/(i^2+1)$;
$r = (7+4i)^2 = 33+56i$;
$s = 159+56i$; $\omega = \sqrt{-384}$;
$C : y^2 = rx^6 + sx^4 + \overline{s}x^2 + \overline{r}$.

$(x,y) \mapsto (x^2,y)$ takes C to E :
$y^2 = rx^3 + sx^2 + \overline{s}x + \overline{r}$.

$(x,y) \mapsto (1/x^2, y/x^3)$ takes C to
$y^2 = \overline{r}x^3 + \overline{s}x^2 + sx + r$.

$$(z,y) \mapsto \left(\frac{1+iz}{1-iz}, \frac{\omega y}{(1-iz)^3}\right)$$
takes H over $\mathbf{F}_{p^2}$ to C.

J is isogenous to
Weil restriction W of E, so
computing $\#J(\mathbf{F}_p)$ is fast.
Here $\#J(\mathbf{F}_p) = 16 \cdot$ prime;
also reasonably twist-secure.

Define $\mathbf{F}_{p^2} = \mathbf{F}_p[i]/(i^2+1)$;
$r = (7+4i)^2 = 33+56i$;
$s = 159+56i$; $\omega = \sqrt{-384}$;
$C : y^2 = rx^6 + sx^4 + \overline{s}x^2 + \overline{r}$.

$(x,y) \mapsto (x^2,y)$ takes C to E :
$y^2 = rx^3 + sx^2 + \overline{s}x + \overline{r}$.

$(x,y) \mapsto (1/x^2, y/x^3)$ takes C to
$y^2 = \overline{r}x^3 + \overline{s}x^2 + sx + r$.

$$(z,y) \mapsto \left(\frac{1+iz}{1-iz}, \frac{\omega y}{(1-iz)^3}\right)$$
takes H over $\mathbf{F}_{p^2}$ to C.

J is isogenous to
Weil restriction W of E, so
computing $\#J(\mathbf{F}_p)$ is fast.
Here $\#J(\mathbf{F}_p) = 16 \cdot \text{prime}$;
also reasonably twist-secure.

Define $\mathbf{F}_{p^2} = \mathbf{F}_p[i]/(i^2+1)$;
$r = (7+4i)^2 = 33+56i$;
$s = 159+56i$; $\omega = \sqrt{-384}$;
$C : y^2 = rx^6 + sx^4 + \overline{s}x^2 + \overline{r}$.

$(x,y) \mapsto (x^2,y)$ takes C to E :
$y^2 = rx^3 + sx^2 + \overline{s}x + \overline{r}$.

$(x,y) \mapsto (1/x^2, y/x^3)$ takes C to
$y^2 = \overline{r}x^3 + \overline{s}x^2 + sx + r$.

$$(z,y) \mapsto \left(\frac{1+iz}{1-iz}, \frac{\omega y}{(1-iz)^3}\right)$$
takes H over $\mathbf{F}_{p^2}$ to C.

J is isogenous to
Weil restriction W of E, so
computing $\#J(\mathbf{F}_p)$ is fast.
Here $\#J(\mathbf{F}_p) = 16 \cdot \text{prime}$;
also reasonably twist-secure.

2003 Scholten:
this strategy for
building genus-2 curves
with fast point-counting.

Define $\mathbf{F}_{p^2} = \mathbf{F}_p[i]/(i^2+1)$;
$r = (7+4i)^2 = 33+56i$;
$s = 159+56i$; $\omega = \sqrt{-384}$;
$C : y^2 = rx^6 + sx^4 + \overline{s}x^2 + \overline{r}$.

$(x,y) \mapsto (x^2,y)$ takes C to E :
$y^2 = rx^3 + sx^2 + \overline{s}x + \overline{r}$.

$(x,y) \mapsto (1/x^2, y/x^3)$ takes C to
$y^2 = \overline{r}x^3 + \overline{s}x^2 + sx + r$.

$$(z,y) \mapsto \left(\frac{1+iz}{1-iz}, \frac{\omega y}{(1-iz)^3}\right)$$

takes H over $\mathbf{F}_{p^2}$ to C.

J is isogenous to
Weil restriction W of E, so
computing $\#J(\mathbf{F}_p)$ is fast.
Here $\#J(\mathbf{F}_p) = 16 \cdot \text{prime}$;
also reasonably twist-secure.

2003 Scholten:
this strategy for
building genus-2 curves
with fast point-counting.

What's new here:

1. Small Kummer coefficients.

Requires lifting Scholten to $\mathbf{Q}$.

$\mathbf{F}_{p^2} = \mathbf{F}_p[i]/(i^2+1)$;
$-4i)^2 = 33+56i$;
$+56i$; $\omega = \sqrt{-384}$;
$= rx^6 + sx^4 + \overline{s}x^2 + \overline{r}$.

$\to (x^2, y)$ takes C to E :
$^3 + sx^2 + \overline{s}x + \overline{r}$.

$\to (1/x^2, y/x^3)$ takes C to
$^3 + \overline{s}x^2 + sx + r$.

$\to \left(\frac{1+iz}{1-iz}, \frac{\omega y}{(1-iz)^3}\right)$
over $\mathbf{F}_{p^2}$ to C.

J is isogenous to
Weil restriction W of E, so
computing $\#J(\mathbf{F}_p)$ is fast.
Here $\#J(\mathbf{F}_p) = 16 \cdot$ prime;
also reasonably twist-secure.

2003 Scholten:
this strategy for
building genus-2 curves
with fast point-counting.

What's new here:

1. Small Kummer coefficients.

Requires lifting Scholten to $\mathbf{Q}$.

2. Expli
$\iota : W \to$
with $\iota \circ$

$]/(i^2+1)$;

$3+56i$;

$=\sqrt{-384}$;

$^4+\overline{s}x^2+\overline{r}$.

akes C to E :

$\overline{s}x+\overline{r}$.

$/x^3)$ takes C to

$sx+r$.

$\left(\frac{\omega y}{(1-iz)^3}\right.$

$\left.\right)$

to C.

J is isogenous to
Weil restriction W of E, so
computing $\#J(\mathbf{F}_p)$ is fast.
Here $\#J(\mathbf{F}_p) = 16 \cdot \text{prime}$;
also reasonably twist-secure.

2003 Scholten:
this strategy for
building genus-2 curves
with fast point-counting.

What's new here:

1. Small Kummer coefficients.
Requires lifting Scholten to $\mathbf{Q}$.

2. Explicit formula
$\iota : W \to J$ and ι'
with $\iota \circ \iota' = 2$.

;

;

$\overline{r}$.

E :

C to

$\left.\right)$

J is isogenous to
Weil restriction W of E, so
computing $\#J(\mathbf{F}_p)$ is fast.
Here $\#J(\mathbf{F}_p) = 16 \cdot$ prime;
also reasonably twist-secure.

2003 Scholten:
this strategy for
building genus-2 curves
with fast point-counting.

What's new here:

1. Small Kummer coefficients.
Requires lifting Scholten to $\mathbf{Q}$.

2. Explicit formulas for isog
$\iota : W \to J$ and $\iota' : J \to W$
with $\iota \circ \iota' = 2$.

J is isogenous to
Weil restriction W of E, so
computing $\#J(\mathbf{F}_p)$ is fast.
Here $\#J(\mathbf{F}_p) = 16 \cdot \text{prime}$;
also reasonably twist-secure.

2003 Scholten:
this strategy for
building genus-2 curves
with fast point-counting.

What's new here:
1. Small Kummer coefficients.
Requires lifting Scholten to $\mathbf{Q}$.

2. Explicit formulas for isogenies
$\iota : W \to J$ and $\iota' : J \to W$
with $\iota \circ \iota' = 2$.

J is isogenous to
Weil restriction W of E, so
computing $\#J(\mathbf{F}_p)$ is fast.
Here $\#J(\mathbf{F}_p) = 16 \cdot \text{prime}$;
also reasonably twist-secure.

2003 Scholten:
this strategy for
building genus-2 curves
with fast point-counting.

What's new here:
1. Small Kummer coefficients.
Requires lifting Scholten to $\mathbf{Q}$.

2. Explicit formulas for isogenies
$\iota : W \to J$ and $\iota' : J \to W$
with $\iota \circ \iota' = 2$.

We took random points
in $H(\mathbf{F}_p) \times H(\mathbf{F}_p)$;
applied $H(\mathbf{F}_p) \to C(\mathbf{F}_{p^2})$
$\to E(\mathbf{F}_{p^2}) = W(\mathbf{F}_p)$;
interpolated formulas for ι'.

Similarly interpolated formulas
for ι; verified composition.

Easy computer calculation.
"Wasting brain power
is bad for the environment."

genous to
triction W of E, so
ng $\#J(\mathbf{F}_p)$ is fast.
$J(\mathbf{F}_p) = 16 \cdot$ prime;
sonably twist-secure.

holten:
tegy for
genus-2 curves
t point-counting.

new here:
l Kummer coefficients.
s lifting Scholten to $\mathbf{Q}$.

2. Explicit formulas for isogenies
$\iota : W \to J$ and $\iota' : J \to W$
with $\iota \circ \iota' = 2$.

We took random points
in $H(\mathbf{F}_p) \times H(\mathbf{F}_p)$;
applied $H(\mathbf{F}_p) \to C(\mathbf{F}_{p^2})$
$\to E(\mathbf{F}_{p^2}) = W(\mathbf{F}_p)$;
interpolated formulas for ι'.

Similarly interpolated formulas
for ι; verified composition.

Easy computer calculation.
"Wasting brain power
is bad for the environment."

3. Using
dynamic
between

e.g. Gen
fast form
Compute
fast form

For more
see our t
Paper co

of E, so
) is fast.
$6 \cdot$ prime;
wist-secure.

urves
unting.

coefficients.
holten to $\mathbf{Q}$.

2. Explicit formulas for isogenies
$\iota : W \to J$ and $\iota' : J \to W$
with $\iota \circ \iota' = 2$.

We took random points
in $H(\mathbf{F}_p) \times H(\mathbf{F}_p)$;
applied $H(\mathbf{F}_p) \to C(\mathbf{F}_{p^2})$
$\to E(\mathbf{F}_{p^2}) = W(\mathbf{F}_p)$;
interpolated formulas for ι'.

Similarly interpolated formulas
for ι; verified composition.

Easy computer calculation.
"Wasting brain power
is bad for the environment."

3. Using isogenies
dynamically move
between $E(\mathbf{F}_{p^2})$ a

e.g. Generate keys
fast formulas for E
Compute shared s
fast formulas for H

For more informat
see our talk at AN
Paper coming soo

ts.
$\mathbf{Q}$.

2. Explicit formulas for isogenies
$\iota : W \to J$ and $\iota' : J \to W$
with $\iota \circ \iota' = 2$.

We took random points
in $H(\mathbf{F}_p) \times H(\mathbf{F}_p)$;
applied $H(\mathbf{F}_p) \to C(\mathbf{F}_{p^2})$
$\to E(\mathbf{F}_{p^2}) = W(\mathbf{F}_p)$;
interpolated formulas for ι'.

Similarly interpolated formulas
for ι; verified composition.

Easy computer calculation.
"Wasting brain power
is bad for the environment."

3. Using isogenies to
dynamically move computat
between $E(\mathbf{F}_{p^2})$ and $J(\mathbf{F}_p)$.

e.g. Generate keys using
fast formulas for E.
Compute shared secrets usin
fast formulas for K.

For more information:
see our talk at ANTS!
Paper coming soon.

2. Explicit formulas for isogenies
$\iota : W \to J$ and $\iota' : J \to W$
with $\iota \circ \iota' = 2$.

We took random points
in $H(\mathbf{F}_p) \times H(\mathbf{F}_p)$;
applied $H(\mathbf{F}_p) \to C(\mathbf{F}_{p^2})$
$\to E(\mathbf{F}_{p^2}) = W(\mathbf{F}_p)$;
interpolated formulas for ι'.

Similarly interpolated formulas
for ι; verified composition.

Easy computer calculation.
"Wasting brain power
is bad for the environment."

3. Using isogenies to
dynamically move computations
between $E(\mathbf{F}_{p^2})$ and $J(\mathbf{F}_p)$.

e.g. Generate keys using
fast formulas for E.
Compute shared secrets using
fast formulas for K.

For more information:
see our talk at ANTS!
Paper coming soon.