Some simple ECC tricks

Mike Hamburg
Rambus Cryptography Research

Optimize for

security

simplicity

speed

in that order!

First corollary: use Edwards

Simpler and faster than short Weierstrass

Complete arithmetic – almost always worth it

Easily makes up for the cofactor of 4 or 8

This talk: simple tricks

Example library APIs

Scalar multiplication:

- Signed binary scalars
- Fixed-base precomputed combs

Arithmetic:

- Inverse square root trick

Algorithmic:

- Encoding to an elliptic curve with Elligator 2
- “Decaf”: use quotient groups instead of subgroups

Time permitting:

- STROBE lite accumulator
- Twist rejection
- The 4-isogeny strategy

Library API

Special-purpose library

Support ECDH, Schnorr signatures

- Scalar*Point (ECDH/keygen/sign)
- Scalar*Scalar + Scalar (Schnorr sign)
- Scalar*Point - Scalar2*Base (Sig verify)
- Optional: Scalar*Base (fast keygen/sign)

Operate always on serialized elements.

General-purpose library

	Scalar	Point
Ser/deser	Maybe	√
Add/sub	√	√
Mul by scalar	√	√
Eq test	√	√
Copy/destroy	√	√
Invert	Maybe	
Elligator		Maybe

Maybe also: $s_1P_1+s_2P_2$ protected;
sG protected; $s_1P_1+s_2G$ unprotected

General-purpose library

What operations might be bottlenecks?

	Scalar	Point
Ser/deser	Maybe	√
Add/sub	√	√
Mul by scalar	√	√
Eq test	√	√
Copy/destroy	√	√
Invert	Maybe	
Elligator		Maybe

Only as called from scalarmul

Maybe also: $s_1P_1+s_2P_2$ protected;
sG protected; $s_1P_1+s_2G$ unprotected

Don’t need to optimize anything else!

Cor: no need for eg affine point formats

Questions about API?

Signed binary scalars

Good for simplicity, security and speed!

[GJMRV-2011-CoZ]
[H-2012-FastCompact]

What this trick does

Compute $s, P \longrightarrow sP$

Completely regular double/add algorithm

Doesn't skip 0 bits, doesn't leak bits of scalar

Take advantage of negation map $P \longrightarrow -P$

Within 1% performance of fastest algo available

Idea

Take advantage of negation map

Use digits {-1,1} instead of {0,1}

Downside: All numbers are odd!

Not a problem if group order q is odd

Binary —> signed binary

$$
\begin{aligned}
x &= 100110 \\
\mathrm{sbin}(x) &= 1\bar{1}\bar{1}11\bar{1} \\
\text{want } x \text{ s.t. } \mathrm{sbin}(x) &= \text{some scalar } s \\
\mathrm{sbin}(x) - 2x &= \bar{1}\bar{1}\bar{1}\bar{1}\bar{1}\bar{1} \\
&= -(2^n - 1) \\
\therefore x &= \frac{s + 2^n - 1}{2}
\end{aligned}
$$

Signed binary ladder

Variable base scalarmul: *s*, *P* —> *sP*

Recode *s:* $s = \ldots\ 1\bar{1}\bar{1}\bar{1}\bar{1}1\ \ldots$

$Q = 0$

For $i = n\text{-}1$ down to 0:

if $s_i = 1$: $Q := 2Q + P$

else: $Q := 2Q - P$

Signed binary fixed window

Variable base scalarmul: *s*, *P* —> *sP*

Precompute: $(1\bar{1}\bar{1}, 1\bar{1}1, 11\bar{1}, 111)P = (1, 3, 5, 7)P$

Recode *s*: $s = \ldots\ 1\bar{1}1\ 1\bar{1}1\ \ldots$

Q = 0

For *i* = *n*-*w* down to 0:

  For *j* = 1 to *w*: *Q* := 2*Q*

  Q := *Q* ± table[*s*[*i*..*i*+*w*]]

Questions about signed binary?

Comb algorithms

Fast, secure, relatively simple fixed-base scalarmul

[LimLee-1994-ExpPrecomp]
[HMV-2004-GuideECC]
[HPB-2004-Combs]
[FZZL-2006-MsbComb]
[H-2012-FastCompact]
and several others

What this trick does

Fixed window scalar mul computes $s, P \longrightarrow sP$

Comb algorithm computes $s \longrightarrow sG$

G known in advance

Performance: about 3x as fast as fixed window

State of the art: fastest fixed-base algo available, even with endomorphisms

Comb algorithm

Fixed-base secret scalarmul: s —> sG

Have already precomputed multiples of G

With fixed window table

Eg: $(11\bar{1}\ 1\bar{1}1\ \bar{1}\bar{1}\bar{1}) \cdot G$

$= ((11\bar{1}) \cdot G \cdot 2^3 + (1\bar{1}1) \cdot G) \cdot 2^3 - (111) \cdot G$

Overall 2^{w-1} points, n/w-1 adds, n-w doubles

Comb algorithm

Elements of table have space between digits

Eg:

$$(11\bar{1}11\bar{1}11\bar{1}\bar{1}\bar{1}) \cdot G$$

$$\begin{aligned} = \quad & 1001001 \quad \cdot 2^2 \cdot G \\ + \quad & 100\bar{1}00\bar{1} \quad \cdot 2 \cdot G \\ - \quad & 100\bar{1}001 \cdot G \end{aligned}$$

Overall 2^{w-1} points, n/w-1 adds, n/w-1 doubles

Scaling the table size

Decreasing returns: 2^{w-1} points for $1/w$ work

To avoid cache timing, have to scan entire table

Can't easily reduce #adds in regular algorithm

each add/sub covers at most 1+log(#points) bits

Reduce #doubles?

Multiple combs

Use more than one table to reduce the number of doubles

Eg 2 tables, 3 bit-combs:

$$
\begin{aligned}
&(11\bar{1}11\bar{1}1\bar{1}\bar{1} && 1\bar{1}11\bar{1}11\bar{1}\bar{1}) \cdot G \\
= \quad &(1001001 && \cdot 2^9 G + 1001001 \quad \cdot G) \cdot 2^2 \\
+ \quad &(\ 1001001 && \cdot 2^9 G - \ 1001001 \quad \cdot G) \cdot 2 \\
+ \quad &(-1001001 && \cdot 2^9 G + \quad 1001001 \cdot G)
\end{aligned}
$$

Overall $2^{w-1}t$ points, n/w-1 adds, n/tw-1 doubles

Use a simple script to find the optimal tradeoff point

Comb pseudocode

Given *s*, compute *sG*

Assume we have *t* combs with *w* teeth each spaced *d* apart

Recode *s* in signed binary

Q = 0

For *i* = *d* – 1 down to 0:

    Q = 2*Q*

    For *j* = 0 to *t* – 1:

        index = $\sum_{k=0}^{k<s} 2^k s_{i+d(wj+k)}$

        If index > 0: *Q* += comb[*j*][index]

        Else: *Q* -= comb[*j*][-index]

Questions about combs?

The inverse square root trick

Adds speed at a small cost in complexity

[BDLSY-2011-EdDSA]
[H-2012-FastCompact]

What this trick does

Compute $\sqrt{x/y}$ twice as fast as the obvious way

optionally also compute $1/z$

have to make sure that inputs are nonzero

Eg. Edwards decompression: $x = \pm\sqrt{\dfrac{1-y^2}{1-dy^2}}$

Simplicity: unify division and sqrt at cost of ~1%

Square root of a ratio

$$
\begin{aligned}
\text{If } y &\neq 0 \\
\text{Let } s &= \frac{1}{\sqrt{xy}} \\
\text{Check } (sx)^2 y &= x \\
\text{Then } sx &= \sqrt{\frac{x}{y}}
\end{aligned}
$$

NB: this works for x = 0 if inverse sqrt algorithm returns 0

Inverse from inv sqrt

Need to mind the ±

Simple enough: $$\frac{1}{x} = x \cdot \left(\frac{1}{\pm\sqrt{x^2}} \right)^2$$

Reduce code size by having only one routine

Batch inverse and sqrt

$$
\begin{aligned}
\text{If } x, y, z &\neq 0 \\
\text{Let } s &= \frac{1}{\sqrt{xyz^2}} \\
\text{Check } s^2xyz^2 &= 1 \\
\text{Then } s^2xyz &= \frac{1}{z} \\
\text{And } sxz &= \sqrt{\frac{x}{y}}
\end{aligned}
$$

How to compute $1/\sqrt{x}$

$$\text{If } p \equiv 3 \pmod 4 : \frac{1}{\sqrt{x}} = x^{\frac{p-3}{4}}$$

$$\text{If } p \equiv 5 \pmod 8 : \frac{1}{\sqrt{x}} = x^{\frac{p-5}{8}}$$

$$\text{or } \frac{1}{\sqrt{x}} = x^{\frac{p-5}{8}} \cdot \sqrt{-1}$$

Costs about as much as an inversion with FLT

Questions about invsqrt?

Encoding to an elliptic curve with Elligator 2

A simple explanation

[SvdW-2006-Construction]
[BHKL-2013-Elligator]

What this trick does

Given an input *r*, produce a point (*x*,*y*) on the curve

The map is 2:1 from the field, not quite uniform

Apply twice and add is uniform

Cost: one inverse/square root operation + ~20M

Encoding to EC is useful

Steganography

Password-authenticated key exchange:

- EKE, SPEKE, Dragonfly, SPAKE2-EE

Tight signatures [GJKW-2007-Tight]

Short signatures [BonehBoyen-2004-Short]

Oblivious function evaluation [JareckiLiu-2009-OFE]

Elligator 2

Requires a point of order 2, char(F) > 3

Generically: $Cy^2 = x(x^2 + Ax + B)$

Obvious solution: set $x = r$; while no y, $x := x + 1$

No good: variable time and not uniform

Idea: Given r, choose (x_1, x_2)

Ensure that ratio of their y^2 is not square

—> one will be on the curve and the other not

Elligator 2

We want $\dfrac{y_1^2}{y_2^2} = \dfrac{x_1}{x_2} \cdot \dfrac{x_1^2 + Ax_1 + B}{x_2^2 + Ax_2 + B}$ to be nonsquare

It suffices to set $x_1^2 + Ax_1 + B = x_2^2 + Ax_2 + B$

$$\Leftrightarrow x_1 + x_2 = -A$$

and also $\dfrac{x_1}{x_2} = ur^2$ where *u* is a fixed nonsquare

Solving: $$x_1 = \frac{-Aur^2}{1 + ur^2}, \quad x_2 = \frac{-A}{1 + ur^2}$$

Computing Elligator 2

$$y^2 = \frac{x(x^2+Ax+B)}{C} = \frac{A}{C} \cdot \frac{A^2 ur^2 - B(1+ur^2)^2}{(1+ur^2)^3} \cdot \begin{cases} ur^2 \\ 1 \end{cases}$$

Set u as a 2^nth root of unity (eg, -1 or i)

Square root algo gives you either $\sqrt{\text{ratio}}$ or $\sqrt{u \cdot \text{ratio}}$

If it's the latter, multiply by r

Adjust low bit of y : even if $\sqrt{\text{ratio}}$, odd

Upshot: takes about 1 sqrt operation

Questions about Elligator 2?

“Decaf” cofactor elimination

For protocols that require prime-order groups

[H-2015-Decaf]

What this trick does

Make a group of order q from a curve of order $4q$

Cost: almost free (i.e. ~20% faster than subgroup)

~10 lines of code

Motivation

Some protocols are easier with prime-order groups

- Can usually be adapted with care
- Most commonly: multiply by cofactor *h*

Previous work: use a subgroup of $\mathbb{G}$

- Effective, but subgroup check is expensive

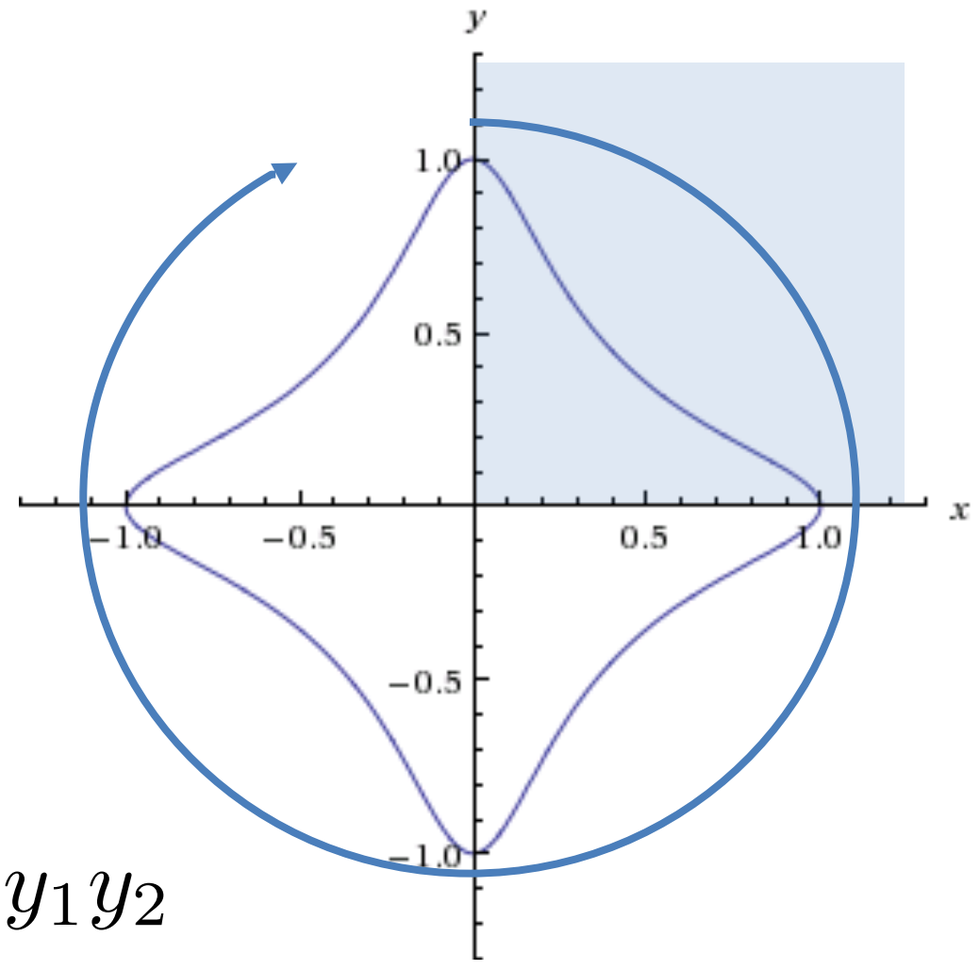
Decaf: use a quotient group

Quotient: $P_1 = P_2$ iff $P_1 - P_2 \in \mathbb{G}[h]$

Let E be an Edwards curve with cofactor $h = 4$

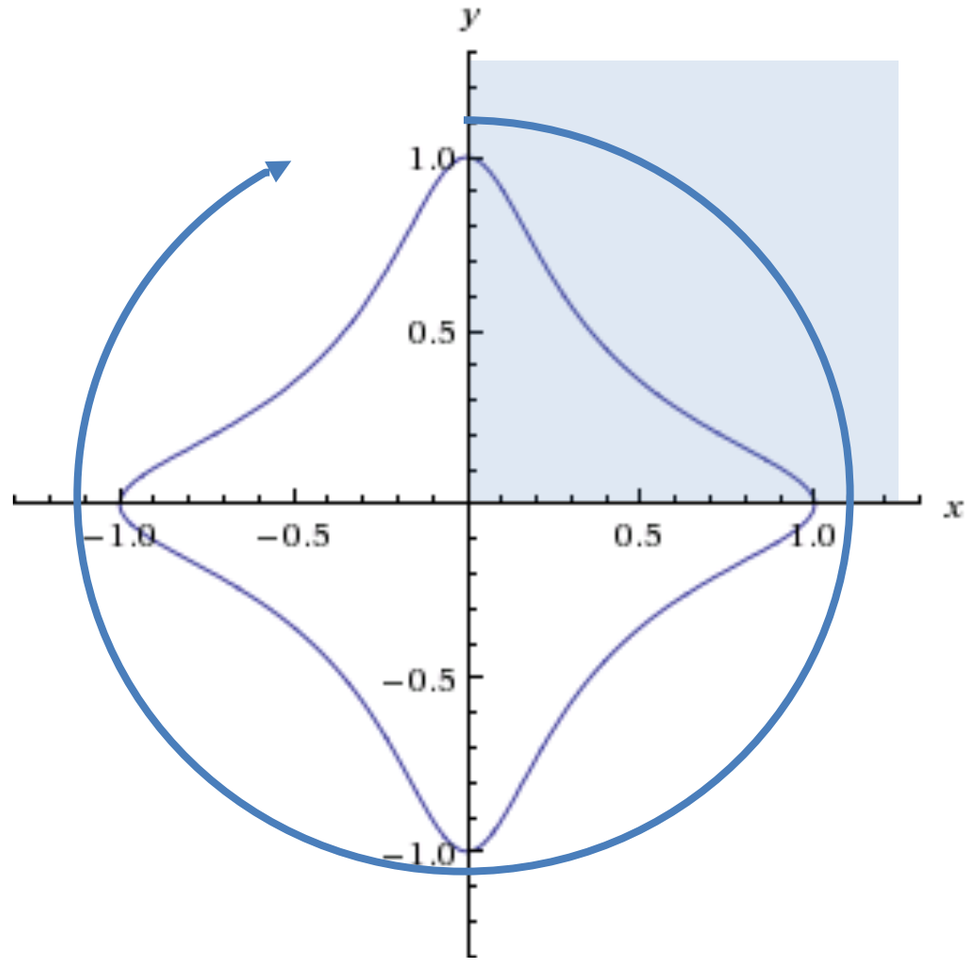
$\mathbb{G}[4]$ is 90° rotations

$P_1 = P_2$ iff $x_1 y_2 = x_2 y_1$ or $x_1 x_2 = -y_1 y_2$



Decaf: serialize

Always write to wire as distinguished point



“First quadrant”

y positive, *x* nonnegative

i.e. *x* and *y* even, $y \neq 0$

Compress: just send *y*

Questions about decaf?

That’s all!

Example library APIs

Scalar multiplication:

- Signed binary scalars
- Fixed-base precomputed combs

Arithmetic:

- Inverse square root trick

Algorithmic:

- Encoding to an elliptic curve with Elligator 2
- “Decaf” cofactor elimination

STROBE lite accumulator

Questions?

References

[AhmadiGranger-2011-IsogenyClasses] Ahmadi and Granger, On isogeny classes of Edwards curves over finite fields
http://eprint.iacr.org/2011/135

[BDPvAvK-2014-Keyak-v1] Bertoni et al., CAESAR submission: Keyak v1
http://competitions.cr.yp.to/round1/keyakv1.pdf

[BDLSY-2011-EdDSA] Bernstein et al., High-speed high-security signatures.
http://ed25519.cr.yp.to/ed25519-20110926.pdf, JCE 2012

[BHKL-2013-Elligator] Bernstein et al., Elligator: Elliptic-curve points indistinguishable from uniform random strings.
ACM-CCS 2013

References

[BonehBoyen-2004-Short] Boneh and Boyen, Short signatures without random oracles.
EUROCRYPT 2004

[FZZL-2006-MsbComb] Feng, Zhu, Zhao, Li, Signed MSB-set comb method for elliptic curve point multiplication.
Information Security Practice and Experience 2006

[GJKW-2007-Tight] Goh et al., Efficient Signature Schemes with Tight Reductions to the Diffie-Hellman Problems
Journal of Cryptology, 2007

[GJMRV-2011-CoZ] Goundar, Joye, Miyagi, Rivain, Venelli, Scalar Multiplication on Weierstraß Elliptic Curves from Co-Z Arithmetic
Journal of Cryptographic Engineering, 2011

References

[H-2014-Isogenies] Hamburg, Twisting Edwards curves with isogenies
https://eprint.iacr.org/2014/027

[H-2015-Decaf] Hamburg, Decaf: Eliminating cofactors through point compression
https://eprint.iacr.org/2015/673, CRYPTO 2015

[H-WIP-StrobeLite] Hamburg, STROBE lite sponge framework,
https://github.com/bitwiseshiftleft/strobelite

[H-2012-FastCompact] Hamburg, Fast and compact elliptic-curve cryptography.
http://eprint.iacr.org/2012/309

References

[HMV-2004-GuideECC] Hankerson, Vanstone, Menezes, Guide to Elliptic Curve Cryptography.
Springer-Verlag, 2004

[HPB-2004-Combs] Hedabou, Pinel, Bénéteau, A comb method to render ECC resistant against Side Channel Attacks.
https://eprint.iacr.org/2004/342

[JareckiLiu-2009-OFE] Jarecki and Liu, Efficient oblivious pseudorandom function with applications to adaptive ot and secure computation of set intersection.
TCC 2009

[LimLee-1994-ExpPrecomp] Lim and Lee, More flexible exponentiation with precomputation.
CRYPTO 1994

References

[Saarinen-2013-Blinker] Saarinen, Beyond Modes: Building a Secure Record Protocol from a Cryptographic Sponge Permutation.
CT-RSA 2014; https://eprint.iacr.org/2013/772

[SvdW-2006-Construction] Shallue and van de Woestijne, Construction of rational points on elliptic curves over finite fields.
ANTS 2006

STROBE lite accumulator

Simple and secure but not fast or standard

[Saarinen-2013-Blinker]
[BDPvAvK-2014-Keyak-v1]
[H-WIP-StrobeLite]

What this trick does

Replace all your symmetric crypto with sponges

Good for protocols and noninteractive crypto

Somewhat slow, but very very compact (<2kB code)

The rest of the protocol

ECC for asymmetric. What about symmetric?

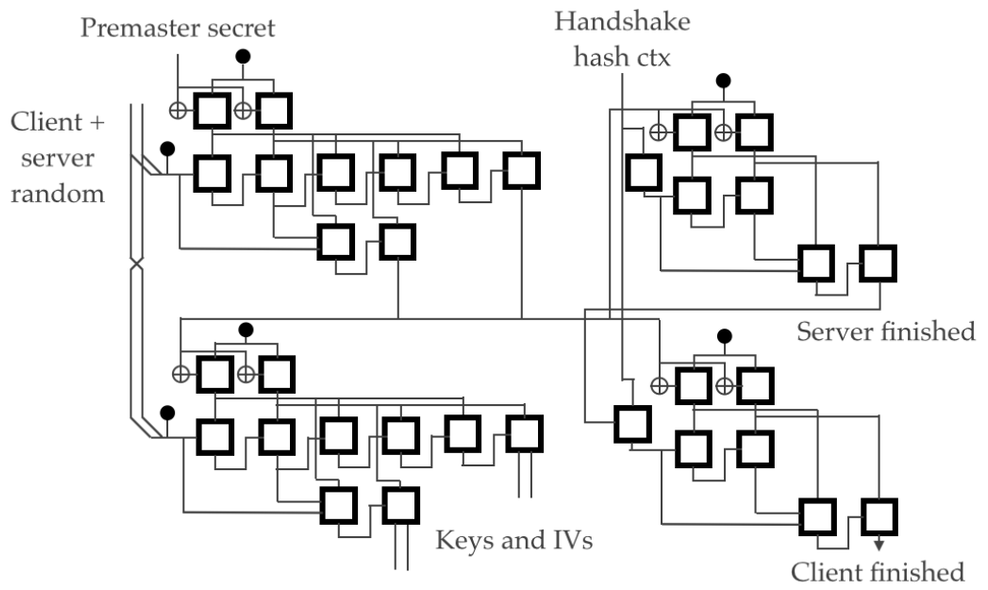
(session key, validators) = hash of handshake msgs?

Parseable, domain separated

Sign hash of handshake msgs?

Encrypted handshake msgs?

Cipher modes? Framing?



STROBE lite

One sponge construction for everything!

Replace hash and cipher

Variant of Markku-Juhani O. Saarinen's BLINKER

Choose your favorite sponge

KeccakF[800] for STROBE lite

< 2kB code (thumb2 C)

<(104,128) or (32,240) bytes (memory,stack)

OK speed: ~200cpb (encrypt 256B) on Cortex-M3

STROBE lite operations

Break down protocol into (tag, operation, data) tuples

Absorb: inject new material into cipher, eg key

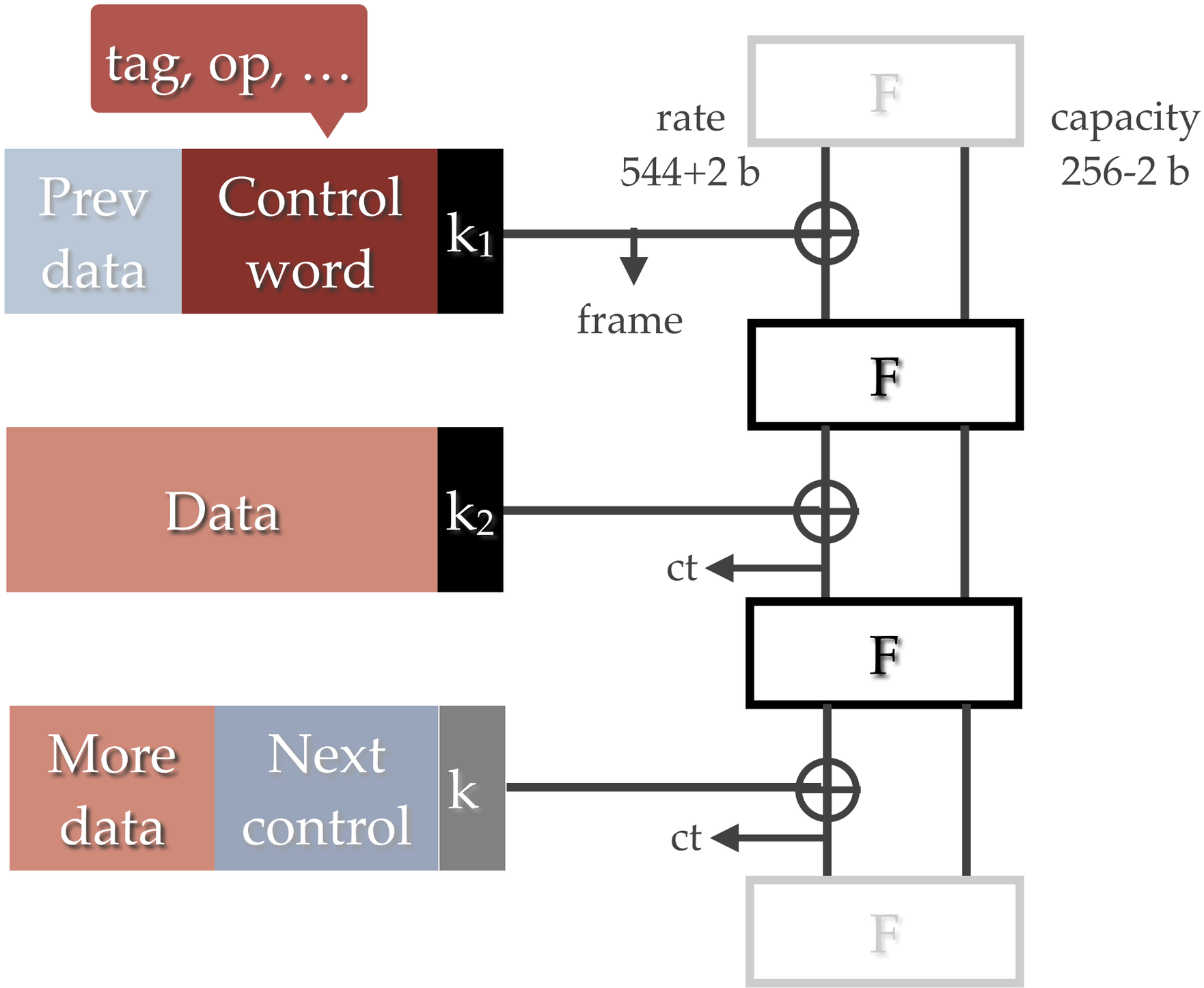
- Plaintext: absorb and also send in the clear

Squeeze: extract pseudorandom data

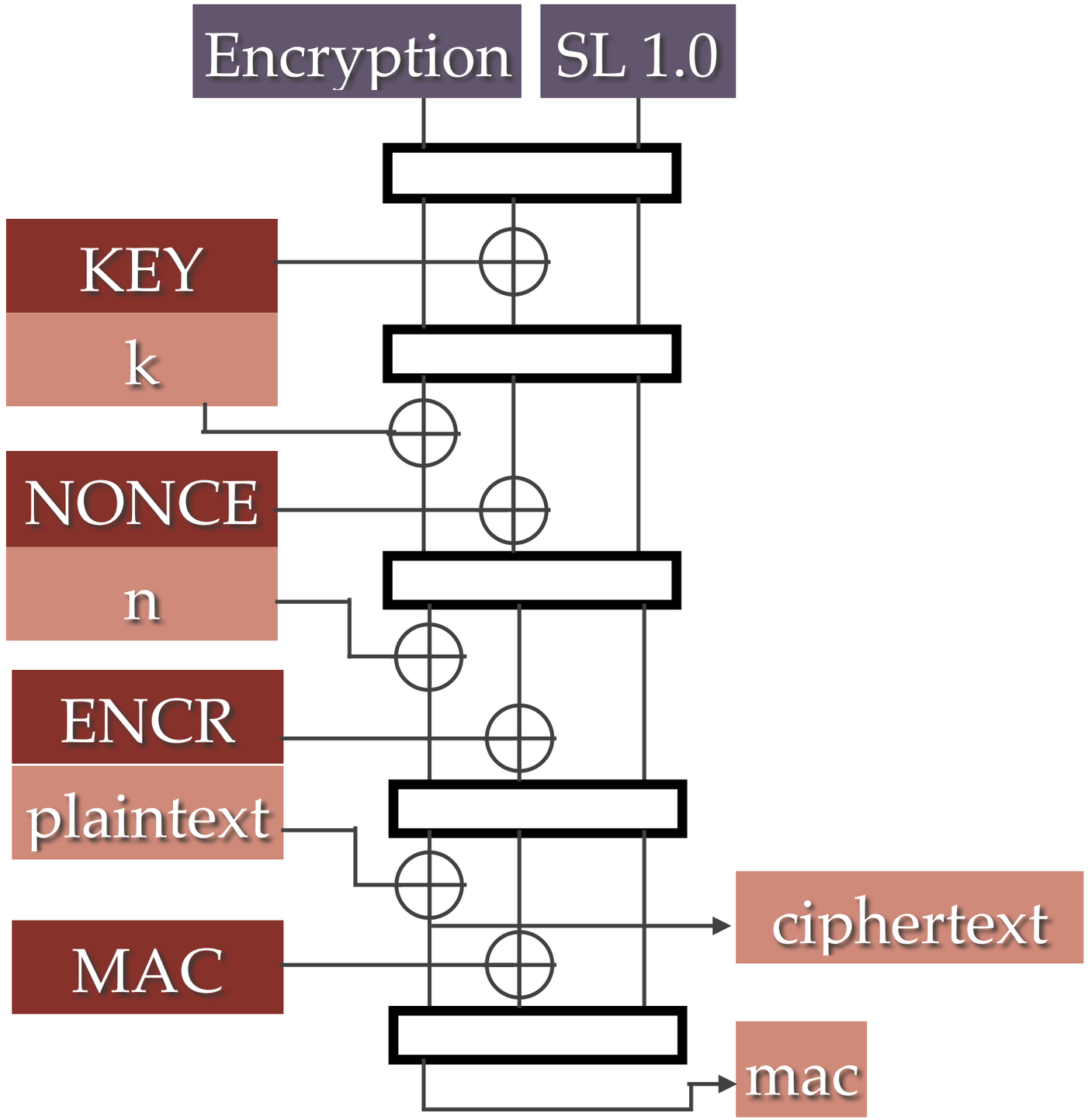
Duplex: encrypt by xoring with squeezed data

Reverse duplex: decrypt or forget

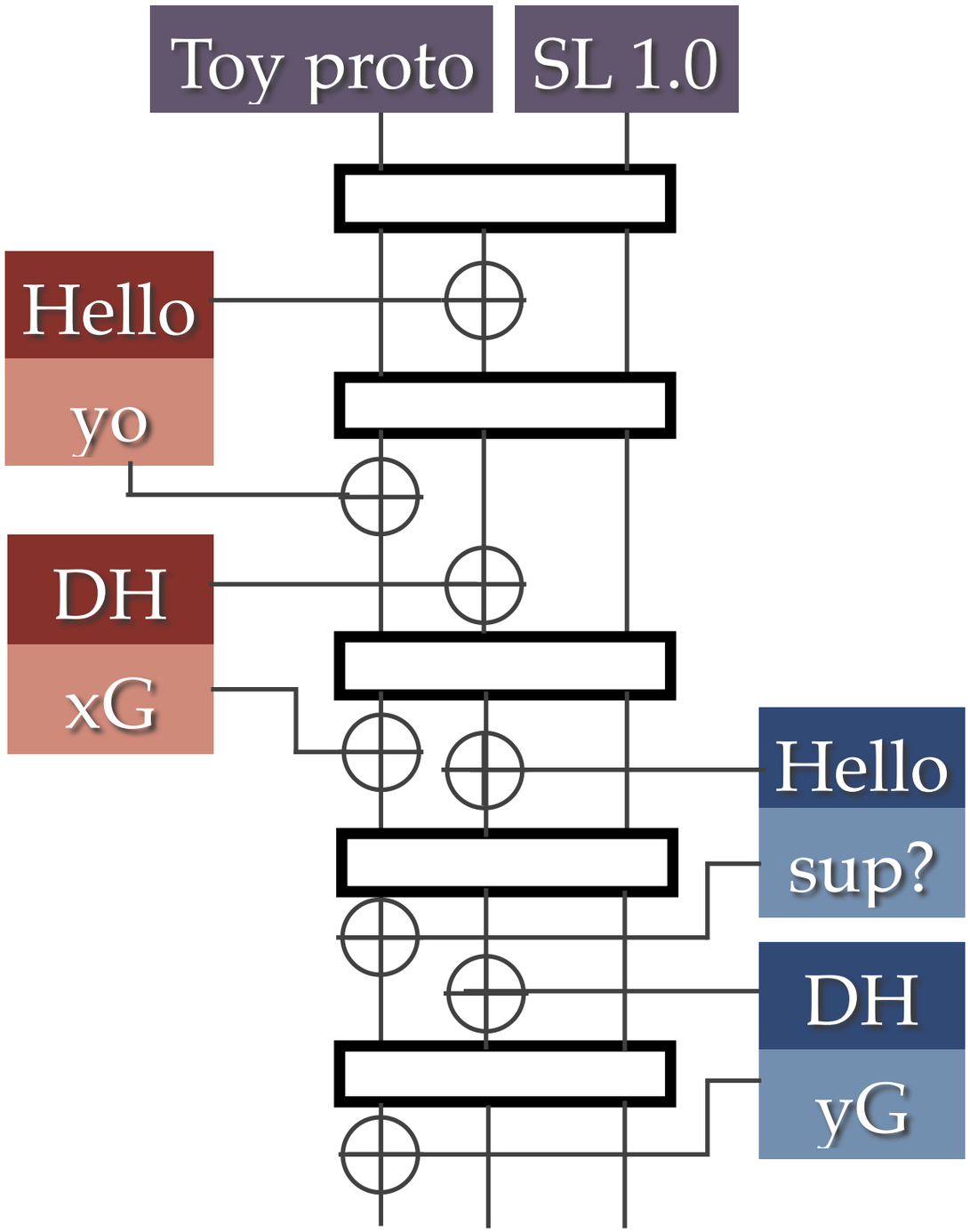
STROBE lite duplex mode

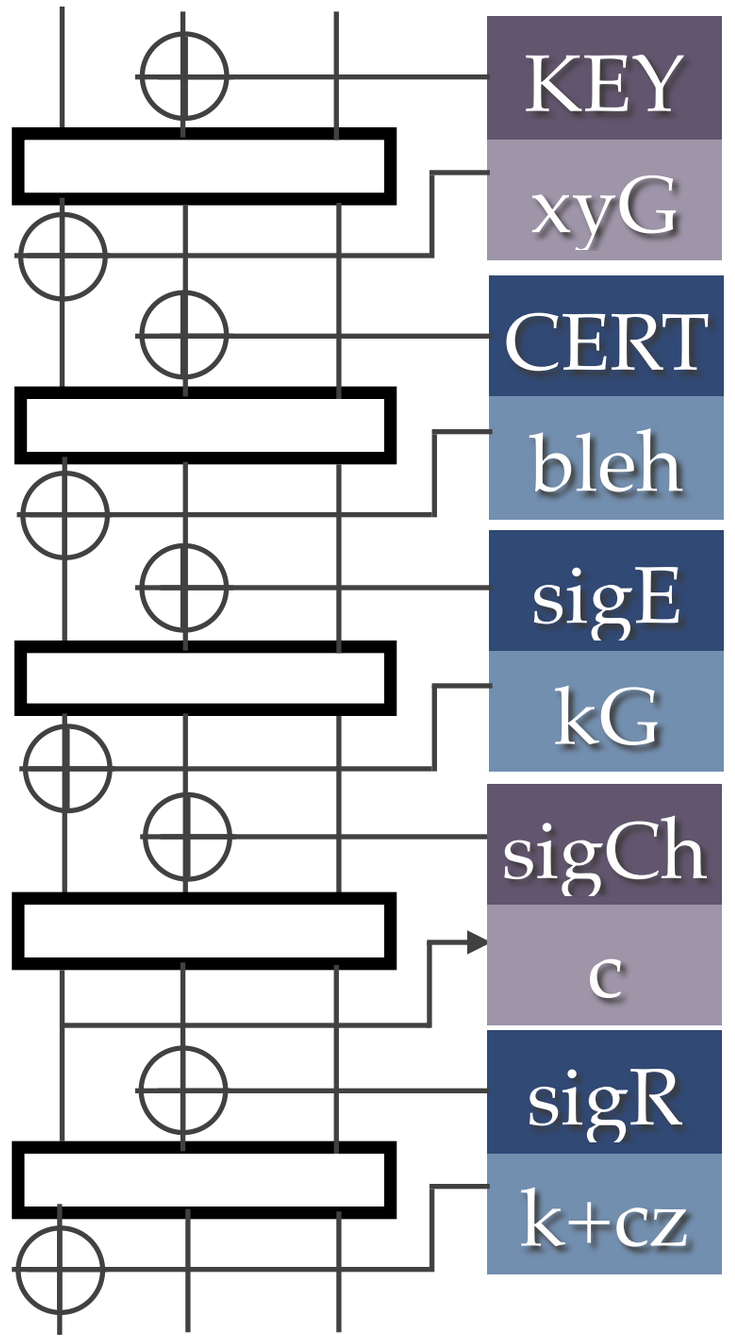


Example: encryption



Example: toy protocol





Questions about STROBE lite?

Montgomery ladder with twist rejection

Can improve security at a small cost to simplicity and speed

[H-2012-FastCompact]
with corrections

What this trick does

Reject twisted points in the Montgomery ladder

(Optionally, but as written) reject points of small order

Cost: ~0.1% performance, < 10 lines of code

Motivation

Curve25519's twist is secure for ECDH

Maybe your curve's twist is terrible?

Maybe your protocol doesn't tolerate twist?

Maybe you want to mimic an Edwards impl?

For whatever reason, let's reject twist points.

And small torsion while we're at it…

The doubling formula

$$x_2 = \frac{(x^2-1)^2}{4x(x^2+Ax+1)} = \left(\frac{x^2-1}{2y}\right)^2$$

Even point's x is always square if and only if on curve!

Rejecting twist points

Assumption: clearing a cofactor divisible by 2

Instead of finishing with $X/Z = XZ^{p-2}$

Compute $s := \sqrt{1/XZ} = (XZ)^{(p-3)/4}$

Check $s^2XZ \stackrel{?}{=} 1$

See earlier slide for $p \equiv 1 \bmod 4$

Finally, $X/Z = s^2XX$. Extra cost: ≈+2 field multiplies

For short Weierstrass curves: use invsqrt trick instead

Questions about twist rejection?

The 4-isogeny strategy: Twisted vs untwisted Edwards curves

Improves speed at a small cost to complexity

[AhmadiGranger-2011-IsogenyClasses]
[H-2014-Isogenies]

What this trick does

Translate operation from untwisted Edwards curve to twisted

Avoid problems with points at ∞ on twisted curves

Gain ~10% speed improvement for modest complexity

Within 2% performance of fastest algo available

Twisted vs untwisted

Twisted Edwards $a = -1$:

Slightly simpler

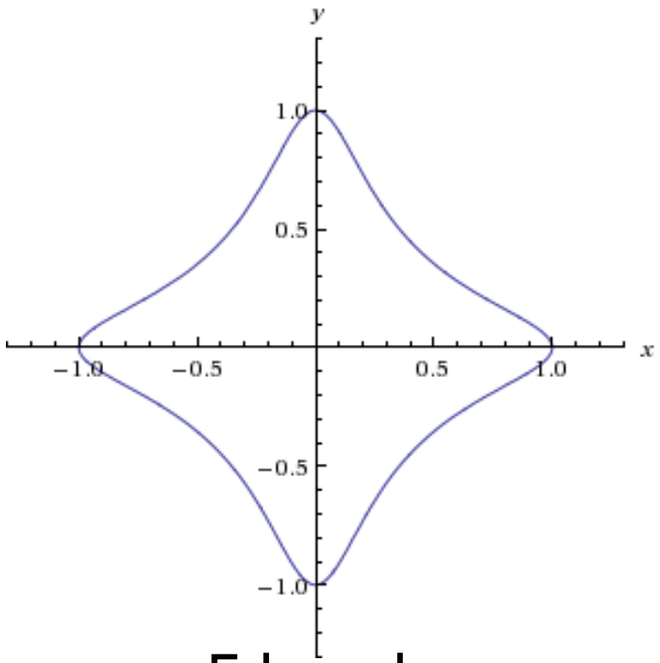
About 10% faster than $a = 1$ (save ~1M)

When $p = 1 \bmod 4$, models are isomorphic

When $p = 3 \bmod 4$, twisted curves are incomplete

… for operations involving points at ∞

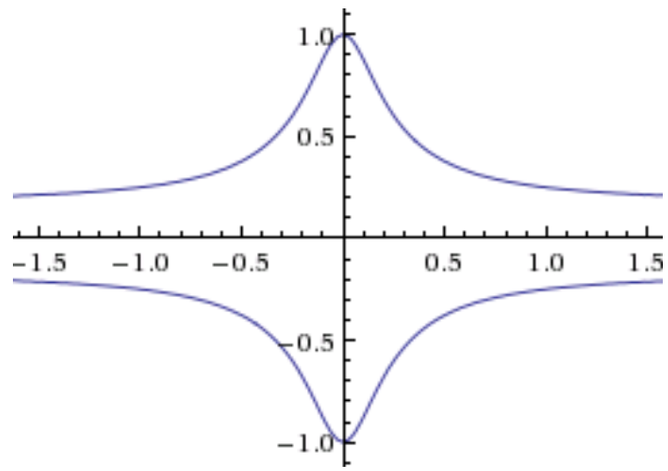
The 4-isogeny strategy



Edwards

$$ax^2 + y^2 = 1 + dx^2y^2$$

(4)



Twisted Edwards

$$-ax^2 + dy^2 = 1 + (d-a)x^2y^2$$

$$\phi_a(x,y) = \left(\frac{2xy}{y^2 - ax^2}, \frac{y^2 + ax^2}{2 - y^2 - ax^2}\right)$$

The 4-isogeny strategy

Compute most things on Edwards curve

Complete addition formulas!

Compute scalarmuls in twisted Edwards

If cofactor = 4, addition laws complete on Im ϕ

Instead of *sP*, compute $\bar{\phi}_a\left(\frac{s}{4} \cdot \phi_a(P)\right)$

This clears the cofactor

Questions about isogeny strategy?