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

<table>
<thead>
<tr>
<th></th>
<th>Scalar</th>
<th>Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ser/deser</td>
<td>Maybe</td>
<td>√</td>
</tr>
<tr>
<td>Add/sub</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Mul by scalar</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Eq test</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Copy/destroy</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Invert</td>
<td>Maybe</td>
<td></td>
</tr>
<tr>
<td>Elligator</td>
<td></td>
<td>Maybe</td>
</tr>
</tbody>
</table>

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?

<table>
<thead>
<tr>
<th></th>
<th>Scalar</th>
<th>Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ser/deser</td>
<td>Maybe</td>
<td>√</td>
</tr>
<tr>
<td>Add/sub</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Mul by scalar</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Eq test</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Copy/destroy</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Invert</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Elligator</td>
<td></td>
<td>Maybe</td>
</tr>
</tbody>
</table>

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 $s, P \rightarrow sP$

Completely regular double/add algorithm

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

Take advantage of negation map $P \rightarrow \neg P$

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 $\rightarrow$ signed binary

\[ x = 100110 \]
\[ \text{sbin}(x) = 1\overline{1}\overline{1}\overline{1}\overline{1}\overline{1} \]

Want \( x \) s.t. \( \text{sbin}(x) = \text{some scalar } s \)

\[ \text{sbin}(x) - 2x = \overline{1}\overline{1}\overline{1}\overline{1}\overline{1}\overline{1} \]
\[ = -(2^n - 1) \]
\[ s + 2^n - 1 \]
\[ \therefore x = \frac{s + 2^n - 1}{2} \]
Signed binary ladder

Variable base scalar multiplication: $s, P \rightarrow sP$

Recode $s$: $s = \ldots 1\overline{1}\overline{1}\overline{1}\overline{1}\overline{1} \ldots$

$Q = 0$

For $i = n-1$ down to 0:

if $s_i = 1$: $Q := 2Q + P$
else: $Q := 2Q - P$
Signed binary fixed window

Variable base scalar multiplication: \( s, P \rightarrow sP \)

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

Recode \( s \): \( s = \ldots \ 1\overline{11} \overline{1}11 \ \ldots \)

\( Q = 0 \)

For \( i = n-w \) down to 0:
  For \( j = 1 \) to \( w \): \( Q := 2Q \)
  \( Q := Q \pm \text{table}[s[i..i+w]] \)
Questions about signed binary?
Comb algorithms

Fast, secure, relatively simple fixed-base scalar multiplication

[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 $s, P \rightarrow sP$

Comb algorithm computes $s \rightarrow sG$

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 scalar mul: \( s \rightarrow sG \)

Have already precomputed multiples of \( G \)

With fixed window table

Eg: \((11\overline{1} 1\overline{1}1 \overline{1}\overline{1}) \cdot G\)

\[ = ((11\overline{1}) \cdot G \cdot 2^3 + (1\overline{1}1) \cdot G) \cdot 2^3 - (111) \cdot G \]

Overall \(2^{w-1}\) points, \(n/w-1\) adds, \(n-w\) doubles
Comb algorithm

Elements of table have space between digits

Eg:

\[
\begin{align*}
(1\overline{1}\overline{1}\overline{1}\overline{1}\overline{1}\overline{1}) \cdot G \\
= & \quad 1001001 \cdot 2^2 \cdot G \\
+ & \quad 100\overline{1}001 \cdot 2 \cdot G \\
- & \quad 100\overline{1}0001 \cdot G
\end{align*}
\]

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 \(1+\log(#\text{points})\) bits

Reduce \#doubles?
Multiple combs

Use more than one table to reduce the number of doubles

Eg 2 tables, 3 bit-combs:

\[
\begin{align*}
&\begin{pmatrix} 1111111111 \quad 1111111111 \end{pmatrix} \cdot G \\
= &\begin{pmatrix} 1001001 \quad 2^9G + 1001001 \quad G \end{pmatrix} \cdot 2^2 \\
+ &\begin{pmatrix} 1001001 \quad 2^9G - 1001001 \quad G \end{pmatrix} \cdot 2 \\
+ &\begin{pmatrix} -1001001 \quad 2^9G + 1001001 \quad G \end{pmatrix}
\end{align*}
\]

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:

$Q = 2Q$

For $j = 0$ to $t - 1$:

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

If index > 0: $Q += \text{comb}[j][\text{index}]$

Else: $Q -= \text{comb}[j][-\text{index}]$
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: $x = \pm \sqrt{\frac{1 - y^2}{1 - dy^2}}$

Simplicity: unify division and sqrt at cost of $\sim 1\%$
Square root of a ratio

If \( y \neq 0 \)

Let \( s = \frac{1}{\sqrt{xy}} \)

Check \( (sx)^2 y = x \)

Then \( sx = \sqrt{\frac{x}{y}} \)

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

If \( x, y, z \neq 0 \)

Let \( s = \frac{1}{\sqrt[\vphantom{2}]{xyz^2}} \)

Check \( s^2xyz^2 = 1 \)

Then \( s^2xyz = \frac{1}{z} \)

And \( sxz = \sqrt{\frac{x}{y}} \)
How to compute $1/\sqrt{x}$

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

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

or 
\[ \frac{1}{\sqrt{x}} = x^{\frac{p-5}{8}} \cdot \sqrt{-1} \]

Costs about as much as an inversion with FLT
Questions about invsqr ?
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 + \( \sim 20M \)
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

\( \rightarrow \) one will be on the curve and the other not
Elligator 2

We want \( \frac{y_1^2}{y_2^2} = \frac{x_1}{x_2} \cdot \frac{x_1^2 + Ax_1 + B}{x_2^2 + Ax_2 + B} \) 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 \( \frac{x_1}{x_2} = ur^2 \) 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

\[ y^2 = \frac{x(x^2 + Ax + B)}{C} = \frac{A}{C} \cdot \frac{A^2 ur^2 - B(1 + ur^2)^2}{(1 + ur^2)^3} \cdot \left\{ \begin{array}{c} \frac{ur^2}{p} \text{ ratio or } \frac{p}{u} \cdot \text{ ratio} \\
1 \end{array} \right. \]

Set \( u \) as a \( 2^n \)th root of unity (e.g., -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
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\degree 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 ≠ 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

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


ACM-CCS 2013
References

   EUROCRYPT 2004

   Information Security Practice and Experience 2006

   Journal of Cryptology, 2007

   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

   http://eprint.iacr.org/2012/309
References


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

tag, op, ...

Prev data  Control word  k_1

Data  k_2

More data  Next control  k

F

F

rate 544+2 b

capacity 256-2 b

frame

t_c

t_c
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 $s^2XZ \equiv 1$

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

For short Weierstrass curves: use invsqrtrt trick instead

See earlier slide for $p \equiv 1 \mod 4$
Questions about twist rejection?
The 4-isogeny strategy: Twisted vs untwisted Edwards curves

Implements 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 $\infty$ on twisted curves

Gain $\sim 10\%$ 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 $p = 1 \mod 4$, models are isomorphic

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

... for operations involving points at $\infty$
The 4-isogeny strategy

Edwards

\[ ax^2 + y^2 = 1 + dx^2y^2 \]

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 \( \phi \)

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

This clears the cofactor
Questions about isogeny strategy?