Edwards coordinates for elliptic curves, part 2

D. J. Bernstein
University of Illinois at Chicago

(Joint work with Tanja Lange)
Elliptic-curve signatures

Standardize a prime $p = 2^{255} - 19$. Not too small; want hard ECDL! Close to $2^{256}$ for fast arithmetic.

Standardize a “safe” elliptic curve $E$ over $\mathbb{F}_p$: $x^2 + y^2 = 1 + dx^2y^2$ where $d = 1 - 1/121666$.

$\#E(\mathbb{F}_p) = 8q$ where $q$ is prime.

$2(p + 1) - \#E(\mathbb{F}_p) = 4 \cdot$ prime.

(2005 Bernstein “Curve25519: new Diffie-Hellman speed records” as $y^2 = x^3 + 486662x^2 + x$)

Standardize $B \in E(\mathbb{F}_p)$, order $q$.

Standardize a “hash function” $H$. 

Signer has 32-byte secret key $n \in \{0, 1, \ldots, 2^{256} - 1\}$.

Everyone knows signer’s 32-byte public key: compressed $nB$.

To sign a message $m$:

- generate a secret $s$;
- compute $R = sB$;
- compute $t = H(R, m)s + n \mod q$;
- transmit $(m, \text{compressed } R, t)$.

To verify $(m, \text{compressed } R, t)$:

- verify $tB = H(R, m)R + nB$.

(first similar idea: 1985 ElGamal; many generalizations, variations; these choices: 2006 van Duin)
Bottleneck: Several types of elliptic-curve scalar multiplication.

Generating key:
given 256-bit integer $n$,
fixed $B \in E(\mathbb{F}_p)$, compute $nB$.

Generating signature: Same.

Verifying signature:
given 256-bit $t$, 256-bit $h$,
fixed $B$, variable $R$,
compute $tB - hR$.

Similar bottleneck for ECDH:
given 256-bit $n$, variable $R$,
compute $nR$. 
Optimizing scalar multiplication

Crypto 1985, Miller, “Use of elliptic curves in cryptography”:

Using division-polynomial recursions can compute $nP$ given $P$ “in $26 \log_2 n$ multiplications”; but can do better!

“It appears to be best to represent the points on the curve in the following form: Each point is represented by the triple $(x, y, z)$ which corresponds to the point $(x/z^2, y/z^3)$.”
1986 Chudnovsky/Chudnovsky, “Sequences of numbers generated by addition in formal groups and new primality and factorization tests”: “The crucial problem becomes the choice of the model of an algebraic group variety, where computations mod \( p \) are the least time consuming.”
For “traditional” \((X/Z^2, Y/Z^3)\):
Chudnovsky/Chudnovsky state explicit formulas using
\(8\text{M}\) for DBL if \(a_4 = -3\);
\(16\text{M}\) for ADD.

“We suggest to write
addition formulas involving
\((X, Y, Z, Z^2, Z^3)\)”
\(9\text{M}\) DBL if \(a_4 = -3\); \(14\text{M}\) ADD.

Also operation counts for
projective coordinates \((X : Y : Z)\)
representing \((X/Z, Y/Z)\);
Hessian curves; Jacobi quartics;
Jacobi intersections.
Asiacrypt 1998,
Cohen/Miyaji/Ono, “Efficient elliptic curve exponentiation using mixed coordinates”:

1. Faster $X, Y, Z, Z^2, Z^3$ formulas than Chudnovsky/Chudnovsky! Compute $Z^2, Z^3$ only for points that will be added.

2. A new coordinate system; speedups in some cases.


4. The first serious analysis of parameter choices.
“Sliding windows” (1939 Brauer, improved by 1973 Thurber): popular method to compute $nP$ from $P$ using very few additions, subtractions, doublings.

Precompute $2P, 3P, 5P, 7P$.

If $n$ is even, recursively compute $(n/2)P$ and then double.

If $n$ is odd, recursively compute $(n \pm 1)P$ or $(n \pm 3)P$ or $(n \pm 5)P$ or $(n \pm 7)P$, whichever involves the largest power of 2, and then add $\mp P$ or $\mp 3P$ or $\mp 5P$ or $\mp 7P$. 
Why not $2P, 3P, 5P, \ldots, 15P$?
Or $2P, 3P, 5P, \ldots, 31P$?

For $2P, 3P, 5P, \ldots, (2^w - 1)P$: 
$\approx 2^{w-1}$ adds in precomputation; 
on average $\approx 256/(w + 2)$ 
adds in main computation.

Cohen/Miyaji/Ono introduce an option to speed up the adds: 
compute $2P$, convert to affine, 
compute $3P, 4P$, convert, 
compute $5P, 7P, 8P$, convert, etc.
Cohen/Miyaji/Ono analyze adds carefully; account for different types of additions; analyze several different coordinate systems; and identify optimal choices of $\omega$, depending on $I/M$, for 160 bits, 192 bits, 224 bits.

Example of results for 160 bits, assuming $S/M = 0.8$: Cohen/Miyaji/Ono recommend one method using “1610.2M” and one using “$4I + 1488.4M$.”
Subsequent improvements:

1. Faster addition/doubling formulas for old coordinates. Many sources; for survey see Explicit-Formulas Database.

2. Fast new coordinates: e.g. Edwards curves, extended Jacobi quartics, inverted Edwards coordinates.


Asiacrypt 2007, Bernstein/Lange, “Faster addition and doubling on elliptic curves”: fast Edwards computations; comparison to other coordinates for scalar multiplication.

Comparison unjustifiably assumed $2P, 3P, 5P, \ldots, 15P$; ignored possibility of inversions.

“This paper is dedicated to Henri Cohen on the occasion of his sixtieth birthday.”

Example of new results for 160-bit scalars:

\[ 11 + 1495.8M \]

for Jacobian coordinates;

\[ 11 + 1434.1M \]

for Jacobian with \( a_4 = -3 \);

\[ 1287.8M \]

for inverted Edwards.
Could also use
“Montgomery coordinates.”
No fast additions, but
fast differential additions
\[ P - Q, P, Q \mapsto P + Q. \]

(1986 Chudnovsky/Chudnovsky; independently 1987 Montgomery with faster formulas)

\[ P \mapsto nP \text{ using } 8.2M \text{ per bit.} \]
Conventional wisdom:
Faster than Jacobian;
therefore the fastest method.
Our prediction: Edwards will be faster than Montgomery for cryptographic applications.

Larger advantage with larger scalars.

Much larger advantage with more scalars: $mP + nQ$.

Need to account carefully for differences between simple multiplication counts and real software speeds.

In progress: implementation.
Double-base chains

Are triplings useful for scalar multiplication?

Can write $nP$ as sum of very few points $c_i 2^{a_i} 3^{b_i} P$
with $c_i = \pm 1$.
But need many doublings, triplings to compute those points.

Asiacrypt 2005,
Dimitrov/Imbert/Mishra: Require $a_0 \geq a_1 \geq \cdots$ and $b_0 \geq b_1 \geq \cdots$.
Only $a_0$ doublings, $b_0$ triplings.
But need more points.
Indocrypt 2006, Doche/Imbert: Use precomputation to expand range of $c_i$’s. Fewer points.

Indocrypt 2007, Bernstein/Birkner/Lange/Peters: Analysis of double-base single-scalar multiplication with various doubling/tripling ratios, various coordinate systems, various addition formulas (including new tripling formulas for Edwards curves), etc.
Basic conclusions: Triplings help Jacobian coordinates, Hessian curves, and tripling-oriented Doche/Icart/Kohel.

But the best resulting speeds are still slower than pure-doubling Edwards.

Analysis assumes 0 inversions. In progress: expand analysis for more inversion strategies. “Grand unified optimization.” And then more scalars...
Many-scalar multiplication

Batch verification of many
\( t_i B - h_i R_i = S_i \): check
\[ \sum_i v_i t_i B - \sum_i v_i h_i R_i - \sum_i v_i S_i = 0 \]
for random 128-bit \( v_i \).
(Naccache et al., Eurocrypt 1994; Bellare et al., Eurocrypt 1998)

Also encounter many scalars
in computing \( nB \) as
\( n_0 B + n_1 2^{16} B + \cdots \)
using precomputed \( 2^{16} B \) etc.
Use subtractive multi-scalar multiplication algorithm:

if $n_1 \geq n_2 \geq \cdots$ then

\[ n_1 P_1 + n_2 P_2 + n_3 P_3 + \cdots = (n_1 - qn_2) P_1 + n_2(qP_1 + P_2) + n_3 P_3 + \cdots \text{ where } q = \left\lfloor \frac{n_1}{n_2} \right\rfloor. \]

(credited to Bos and Coster by de Rooij, Eurocrypt 1994; see also tweaks by Wei Dai, 2007)

Addition speed is critical.

Inverted Edwards coordinates:

$9M + 1S$, speed record.
Elliptic-curve factorization

Bernstein/Birkner/Lange/Peters, in progress: Edwards ECM.

First-stage ECM analysis: similar to ECC analysis. Can use larger scalars, increasing the advantage of Edwards over Montgomery.

Second stage: more complicated. Also some improvements in curve selection.
Elliptic-curve primality proving

Is $n$ prime? Maybe.

Want computation of $kP$ in $E(\mathbb{Z}/n)$

to prove that $kP = 0$ in $E(\mathbb{Z}/p)$
for every prime divisor $p$ of $n$; use this to prove that $n$ is prime.

Proper definition of $E(\mathbb{Z}/n)$ achieves this, but also requires many invertibility tests, each costing at least $1\text{M}$ and extra implementation effort.
For simplicity and speed, current ECPP software omits various tests.

Bernstein question to Morain: “Do the resulting computations actually prove primality?”
For simplicity and speed, current ECPP software omits various tests.

Bernstein question to Morain: “Do the resulting computations actually prove primality?”

Morain answer to Bernstein: “Feel free to look for a non-prime counterexample.”

Disclaimer: There is no evidence that this conversation took place.
Often ECPP uses curves that can be transformed to Montgomery, Edwards, etc. (Chance $\to 1$ as $n \to \infty$?)

With detailed case analysis can eliminate tests for zero from a Montgomery-style ECPP. (2006 Bernstein)

Bernstein/Lange, with Jonas Lindstrøm Jensen, in progress: Aiming for simpler, faster ECPP using Edwards.