Selecting Elliptic Curves for Cryptography: an Efficiency and Security Analysis

http://eprint.iacr.org/2014/130.pdf

Craig Costello
ECC2014 – Chennai, India

Joint work with
Joppe Bos (NXP), Patrick Longa (MSR), Michael Naehrig (MSR)

June 2013 – the Snowden leaks

The New York Times

"… the NSA had written the [crypto] standard and could break it."

Post-Snowden responses

- **Bruce Schneier:** *“I no longer trust the constants. I believe the NSA has manipulated them…”*
- **Nigel Smart:** *“Shame on the NSA…”*
- **IACR:** *“The membership of the IACR repudiates mass surveillance and the undermining of cryptographic solutions and standards.”*
- **TLS Working Group:**
  formal request to CFRG for new elliptic curves for usage in TLS!!!
- **NIST:** announces plans to host workshop to discuss new elliptic curves

  http://crypto.2014.rump.cr.yp.to/487f98c1a1a031283925d7affdbdef1c.pdf

Pre-Snowden suspicions re: NIST (and their curves)

- **2013 - Bernstein and Lange:** "*Jerry Solinas at the NSA used this [random method] to generate the NIST curves … or so he says…*"

- **2008 – Koblitz and Menezes:** "*However, in practice the NSA has had the resources and expertise to dominate NIST, and NIST has rarely played a significant independent role.*"

- **2007 – Shumow and Ferguson:** "*We don't know how* $Q = [d]P$ *was chosen, so we don't know if the algorithm designer [NIST] knows [the backdoor]* d*.*"

- **1999 – Scott:** *"So, sigh, why didn't they [NIST] do it that way? Do they want to be distrusted?"*

NIST's CurveP256: one-in-a-million?

Prime characteristic: $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$

Elliptic curve: $E/\boldsymbol{F}_p : y^2 = x^3 - 3x + b$

Curve constant: $b = \sqrt{-\frac{27}{SHA1(s)}}$

Seed: $s =$ c49d360886e704936a6678e1139d26b7819f7e90

Scott '99:

"Consider now the possibility that one in a million of all curves have an exploitable structure that "they" know about, but we don't.. Then "they" simply generate a million random seeds until they find one that generates one of "their" curves…"

Rigidity

- Give reasoning for all parameters and minimize "choices" that could allow room for manipulation

- Hash function needs a seed (digits of e, π, etc), but do choice of seed and choice of hash function themselves introduce more wiggle room?

- **Goal:** Justify all choices with (hopefully) undisputable efficiency arguments

  e.g. choose fast prime field and take smallest curve constant that gives ``optimal'' group order/s [Bernstein'06]

So then, what about these?

Replacement curve	Prime p	Constant b
(NEW) Curve P-256	$2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$	2627
(NEW) Curve P-384	$2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$	14060
(NEW) Curve P-521	$2^{521} - 1$	167884

- Same fields and equations ($E : y^2 = x^3 - 3x + b$) as NIST curves
- BUT smallest constant b (RIGID) such that $\#E$ and $\#E'$ both prime
- So, simply change curve constants, and we're done, right???

(Our) Motivations

1. **Curves that regain confidence**

   - rigid generation / nothing up my sleeves
   - public approval and acceptance

2. **15 years on, we can do so much better than the NIST curves**
   (and this is true regardless of NIST-curve paranoia!)

   - side-channel resistance
   - faster finite fields and modular reduction
   - a whole new world of curve models

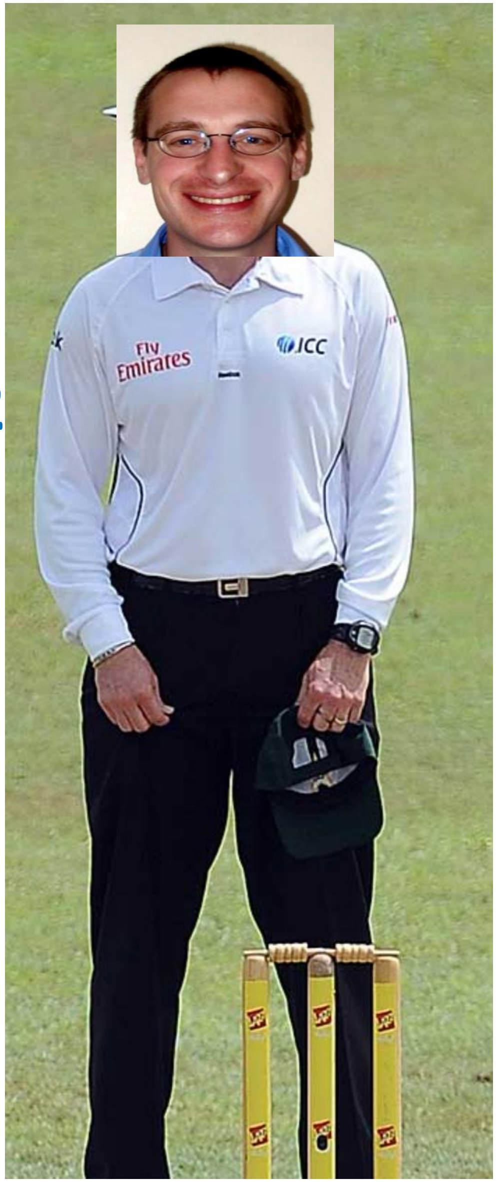
3. **Whether it's cricket or crypto, a proper game needs several players…**

The players

- **Aranha-Barreto-Pereira-Ricardini: M-221, M-383, M-511, E-382,…**
- **Bernstein-Lange: Curve25519, Curve41417, E-521,…**
- **Bos-Costello-Longa-Naehrig: the NUMS curves**
- **Hamburg: Goldilocks448, Ridinghood448,…**
- **ECC Brainpool: brainpoolP256t1, brainpoolP384t1,…**
- **…**
- ***your-name-here?*: *your-curves-here?***

The players

- **Aranha-Barreto-Pereira-Ricardini: M-221, M-383, M,511, E-382**
- **Bernstein-Lange: Curve25519, Curve41417, E-521,…**
- **Bos-Costello-Longa-Naehrig: the NUMS curves**
- **Hamburg: Goldilocks448, Ridinghood448,…**
- **ECC Brainpool: brainpoolP256t1, brainpoolP384t1,…**
- **…**
- ***your-name-here?*: *your-curves-here?***



Umpire Paterson
(CFRG co-chair)

Contents

PART I : CHOOSING CURVES

Speed-records and security hunches

Prime fields and modular reduction

Curve models and killing cofactors

Montgomery ladder and twist-security

Our chosen curves: the NUMS curves

PART II : IMPLEMENTING THEM

Constant-time implementations and recoding scalars

Exception-free algorithms and Weierstrass “completeness”

Performance numbers and practical considerations

Conclusions and recommendations

The last 2 years of "state-of-the-art" speeds

- [LS'12] (*AsiaCrypt*) & [LFS'14] (*JCEN*) ≈90,000 cyc
  4-GLV/GLS using CM curve over quad. ext. field
- [BCHL'13] (*EuroCrypt*) ≈120,000 cyc & [BCLS'14] (AsiaCrypt) ≈90,000 cyc
  Laddering on genus 2 Kummer surface
- [CHS '14] (*EuroCrypt*) ≈140,000 cyc
  2-dimensional Montgomery ladder using Q-curve over quad. ext. field
- [OLAR'13] (*CHES*) ≈115,000 cyc
  GLS on a composite-degree binary extension field

All of the above offer ≈128-bit security against best known attack

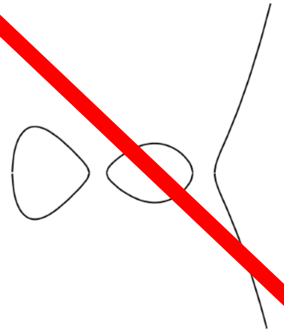
BUT

None of the above have been considered in the search for new curves!!!

Security hunches killing all the fun

- Best known attacks against the curves on prior page are ≈ the same
- BUT widespread agreement that **random elliptic curves** over **prime fields** are safest hedge for real world deployment
- By "random", I mean huge CM discriminant, huge class number, huge MOV degree… no special structure!
- **Basic recipe:** over fixed prime field, (rigidly) find curve with "optimal" group orders (SEA), then assert above are huge (they will be)

Security hunches killing all the fun

WARNING: ϕ π_p  $< 100{,}000$ cyc

Contents

PART I : CHOOSING CURVES

Speed-records and security hunches

Prime fields and modular reduction

Curve models and killing cofactors

Montgomery ladder and twist-security

Our chosen curves: the NUMS curves

PART II : IMPLEMENTING THEM

Constant-time implementations and recoding scalars

Exception-free algorithms and Weierstrass "completeness"

Performance numbers and practical considerations

Conclusions and recommendations

Two prime forms analyzed

(1) Pseudo-Mersenne primes: $\boldsymbol{p = 2^{\alpha} - \gamma}$
(2) Montgomery-friendly primes: $\boldsymbol{p = 2^{\alpha}(2^{\beta} - \gamma) - 1}$

- For each security level $s \in \{128,192,256\}$, we benchmarked two of both:
  (a) one "full bitlength" prime
  (b) one "relaxed bitlength" prime
- In our case, relaxed meant:
  - drop one bit for pseudo-Mersenne (lazy reduction)
  - drop two bits for Mont-friendly (conditional sub saved in every mul)
- Subject to above, security level **determines** primes
  - α and β determined by s
  - smallest $\gamma > 0$ such that p is prime and $\boldsymbol{p \equiv 3 \bmod 4}$

Some premature performance ratios

Target Security Level	Pseudo-Mers Full	Pseudo-Mers Relaxed	Mont-Friendly Full	Mont-Friendly Relaxed
128	1.00x	0.97x	1.00x	0.84x
192	0.94y	0.90y	1.00y	0.90y
256	0.89z	0.85z	1.00z	0.92z

Cost ratios of variable-base scalar multiplications on twisted Edwards curves at three target security levels

- Relaxed version naturally wins in both cases
- Montgomery-friendly vs. Pseudo-Mersenne not as clear cut
- So what did we end up going for….???

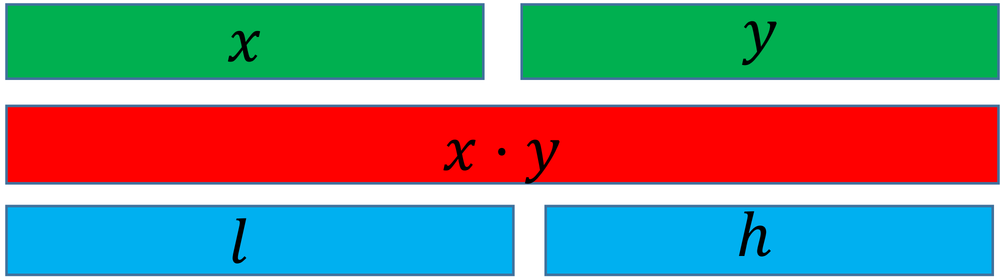
Full length pseudo-Mersenne primes

- We went for **pseudo-Mersenne over Montgomery-friendly**
  - simpler (may depend on who you ask?)
  - take a decent performance hit at 128-bit level
  - closer resemblance to NIST-like arithmetic
- We went for **full-length over relaxed-bitlength**
  - take a performance hit of 2-4%
  - BUT maximizes ECDLP security, maintains 64-bit alignment, & avoids temptation to keep going lower

Security level	Prime
128	$2^{256} - 189$
192	$2^{384} - 317$
256	$2^{512} - 569$

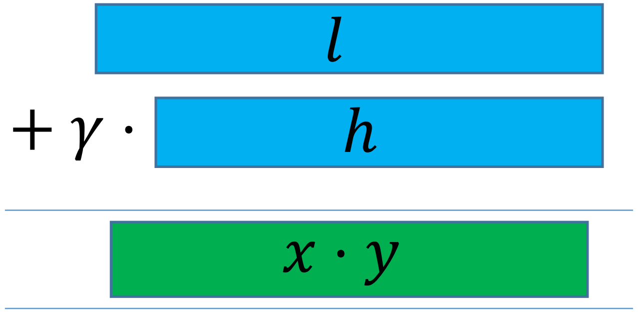
Arithmetic for the pseudo-Mersenne primes

- **Constant time modular multiplication**



input: $0 \leq x, y < 2^{\alpha} - \gamma$

$$x \cdot y \in \mathbf{Z}$$
$$= h \cdot 2^{\alpha} + l$$
$$\equiv h \cdot 2^{\alpha} + l \; - h(2^{\alpha} - \gamma) \bmod (2^{\alpha} - \gamma)$$
$$= l + \gamma \cdot h$$



output: $x \cdot y \;\bmod\; (2^{\alpha} - \gamma)$

(after fixed=worst-case number of reduction rounds)

- **Constant time modular inversion:** $a^{-1} \equiv a^{p-2} \bmod p$
- **Constant time modular square-root:** $\sqrt{a} \equiv a^{(p+1)/4} \bmod p$

What primes do others like?

- **Bernstein and Lange:** **Curve25519, Curve41417, E-521**

$$p = 2^{255} - 19, \qquad p = 2^{414} - 17, \qquad p = 2^{521} - 1$$

- **Hamburg:** **Ed448-Goldilocks, Ed480-Ridinghood**

$$p = 2^{448} - 2^{224} - 1, \quad p = 2^{480} - 2^{240} - 1$$

- **Aranha-Barreto-Pereira-Ricardini:** **M-221, M-383, M-511 , E-382**, etc

$$p = 2^{221} - 3, \qquad p = 2^{383} - 187, \qquad p = 2^{511} - 187, \quad p = 2^{382} - 105$$

- **Brainpool:** **brainpoolP256t1, brainpoolP384t1**, etc

$p =$ 76884956397045344220809746629001649093037950200943055203735601445031516197751

Contents

PART I : CHOOSING CURVES

Speed-records and security hunches

Prime fields and modular reduction

Curve models and killing cofactors

Montgomery ladder and twist-security

Our chosen curves: the NUMS curves

PART II : IMPLEMENTING THEM

Constant-time implementations and recoding scalars

Exception-free algorithms and Weierstrass "completeness"

Performance numbers and practical considerations

Conclusions and recommendations

A world of curve models

$$\boldsymbol{y^2 = x^3 + ax + b}$$

short Weierstrass curves

$$y^2 = x^4 + 2ax^2 + 1$$

Jacobi quartics

$$ax^3 + y^3 + 1 = dxy$$

(twisted) Hessian curves

$$By^2 = x^3 + Ax^2 + x$$

Montgomery curves

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

(twisted) Edwards curves

$$y^2 = x^3 + ax^2 + 16ax$$

Doubling-oriented DIK curves

$$s^2 + c^2 = 1 \quad \cap \quad as^2 + d^2 = 1$$

Jacobi intersections

See Bernstein and Lange's Explicit-Formulas Database (EFD) and/or Hisil's PhD thesis

The chosen ones

Weierstrass curves

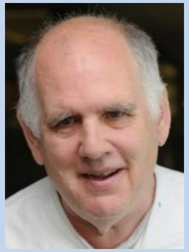
$$y^2 = x^3 + ax + b$$

- Most general form
- Prime order possible
- Exceptions in group law
- NIST and Brainpool curves

Montgomery curves

$$By^2 = x^3 + Ax^2 + x$$

- Subset of curves
- Not prime order
- Fast Montgomery ladder
- $\approx$ Exception free



(twisted) Edwards curves

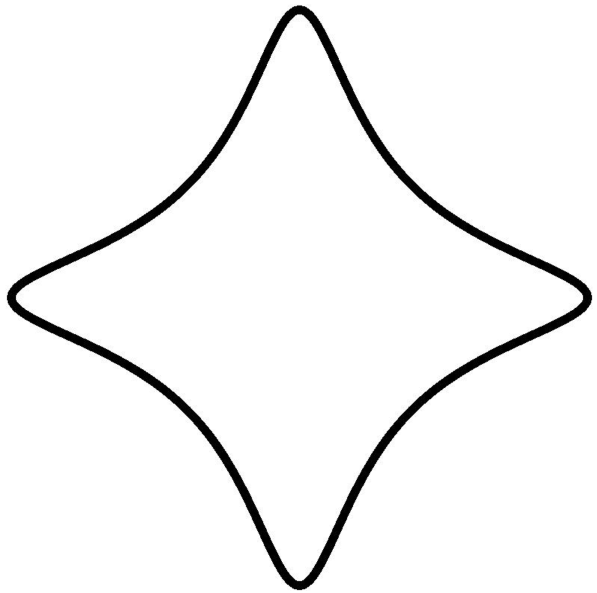
$$ax^2 + y^2 = 1 + dx^2y^2$$

- Subset of curves
- Not prime order
- Fastest addition law
- Some have complete group law

Complete addition on Edwards curves

Let $d \neq \square$ in K and consider Edwards curve

$$E/K:\ x^2 + y^2 = 1 + dx^2y^2$$



For all (!!!) $P_1 = (x_1, y_1), P_2 = (x_2, y_2) \in E(K)$

$$P_1 + P_2 =: P_3 = \left(\frac{x_1y_2 + y_1x_2}{1 + dx_1x_2y_1y_2}, \frac{y_1y_2 - x_1x_2}{1 - dx_1x_2y_1y_2}\right)$$

Denominators never zero, neutral element rational $= (0,1)$*, etc..*

(Bernstein-Lange, AsiaCrypt 2007)

Edwards vs twisted Edwards

General twisted Edwards $\boldsymbol{E_{a,d} : ax^2 + y^2 = 1 + dx^2y^2}$

When $a = 1$ (Edwards!) $E_{1,d} : x^2 + y^2 = 1 + dx^2y^2$

Fastest complete addition (for $d \neq \square$) **9M+1d**

(Bernstein-Lange, AsiaCrypt 2007 and Hisil et al., AsiaCrypt 2008)

When $a = -1$ $E_{-1,d}: -x^2 + y^2 = 1 + dx^2y^2$

Fastest addition **8M**, also (technically) incomplete when $p \equiv 3 \bmod 4$

(Hisil et al., AsiaCrypt 2008)

- **Edwards completeness highly desirable, but so are the fast (twisted Edwards) formulas!**
- **Incomplete formulas still work for any $\boldsymbol{P, Q}$ where $\boldsymbol{P \neq Q}$, and both have odd order…**

Killing cofactors and the fastest formulas

- (Twisted) Edwards curves necessarily have a cofactor of at least 4, so assume $\#E = 4r$ where r is a large prime

- Users will check that $P \in E$, but cannot easily check whether P has order $r, 2r$, or $4r$

- If secret scalars k are in $[1, r)$, then attackers could send P of order $4r$, and on receiving $[k]P$, compute $\boldsymbol{[rk]P = [k \bmod 4]P} \in E(F_p)[4]$ to reveal

  $k \bmod 4$ (i.e. the last two bits of k)

- RECALL: the fastest additions will work for all $P \neq Q$, both of odd order…

Killing cofactors and the fastest formulas

Our approach

- incomplete twisted Edwards curve

$$E_{-1,d} : -x^2 + y^2 = 1 + dx^2y^2$$

- modified set of scalars

$$k \in [1, 2, \dots r - 1] \;\; \leftrightarrow \hat{k} \in [4, 8, 4r - 4]$$

- initial double-double

$$P \in E \mapsto Q \;\; := [4]P \in E[r]$$

- fastest formulas to compute

$$[\hat{k}]P = [k]Q$$

"specified curve" incomplete, but uses fastest formulas and stays on one curve

Killing cofactors and the fastest formulas

Hamburg's approach (http://eprint.iacr.org/2014/027)

- complete Edwards curve

$$E_{1,d} : x^2 + y^2 = 1 + dx^2y^2$$

- use 4-isogeny to incomplete twisted:

$$\phi : E_{1,d} \rightarrow E_{-1,d-1}$$

- fastest formulas to compute:

$[k]P$ on $E_{-1,d-1}$ (since $\mathrm{im}(\phi) = E_{-1,d-1}[r]$)

- use dual to come back to $E_{1,d}$

$$\hat{\phi} : E_{-1,d-1} \rightarrow E_{1,d}$$

"specified curve" complete and uses fastest formulas, but isogeny needed

Killing cofactors and the fastest formulas

Bernstein-Chuengsatiansup-Lange approach (Curve41417)

- complete Edwards curve

$$E_{1,d} : x^2 + y^2 = 1 + dx^2y^2$$

- kill torsion with doublings

$$\hat{k} \in [8, 16, \ldots]$$

- stay on $E_{1,d}$, at the expense of 1M per addition

but compare ≈3727M to ≈3645M $(+\phi + \hat{\phi})$

"specified curve" is complete, stay on it (simple), but slightly slower additions

Contents

PART I : CHOOSING CURVES

Speed-records and security hunches

Prime fields and modular reduction

Curve models and killing cofactors

Montgomery ladder and twist-security

Our chosen curves: the NUMS curves

PART II : IMPLEMENTING THEM

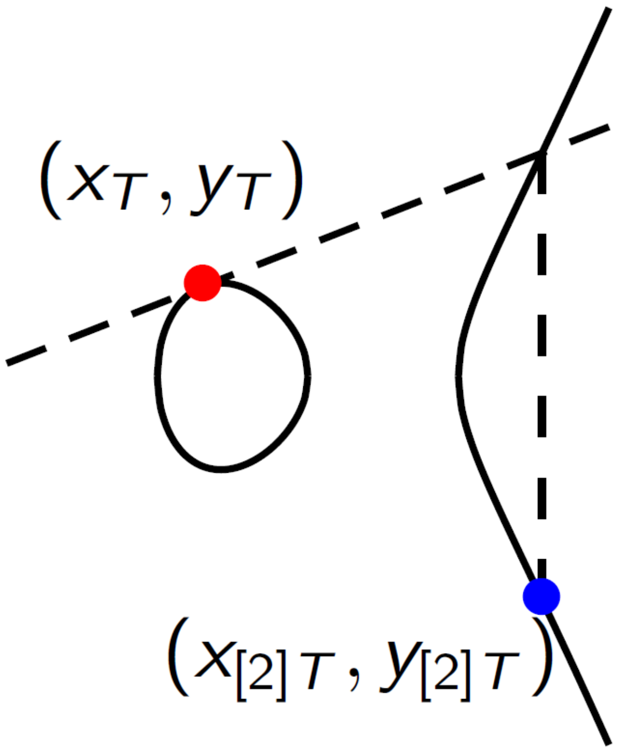
Constant-time implementations and recoding scalars

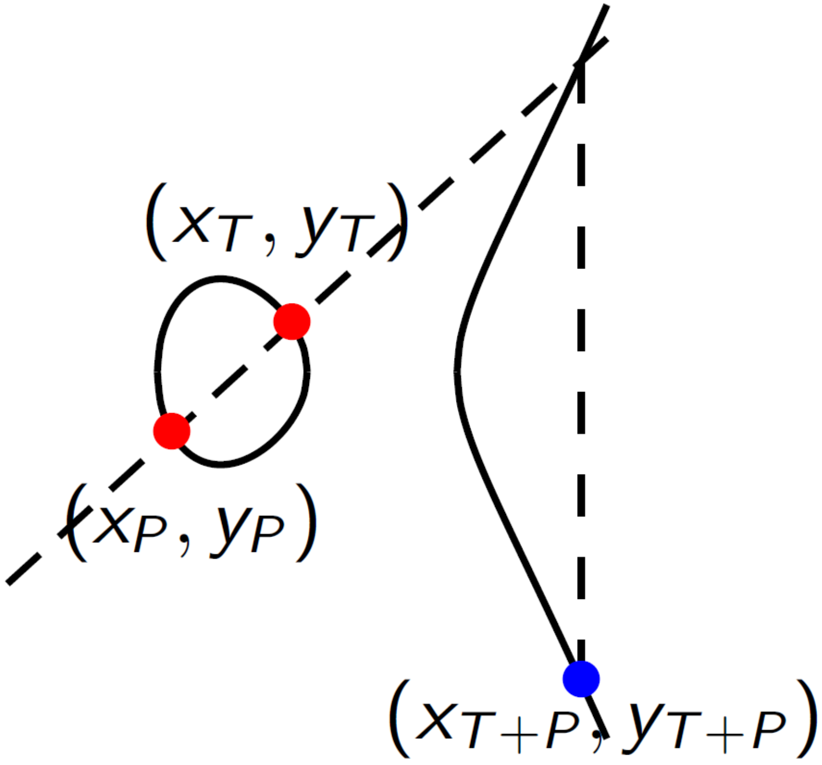
Exception-free algorithms and Weierstrass “completeness”

Performance numbers and practical considerations

Conclusions and recommendations

Textbook arithmetic on $y^2 = x^3 + ax + b$

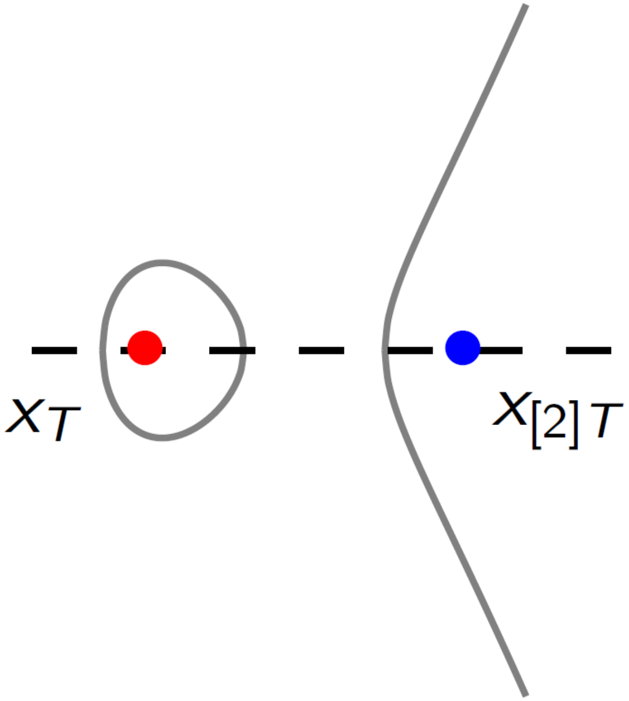


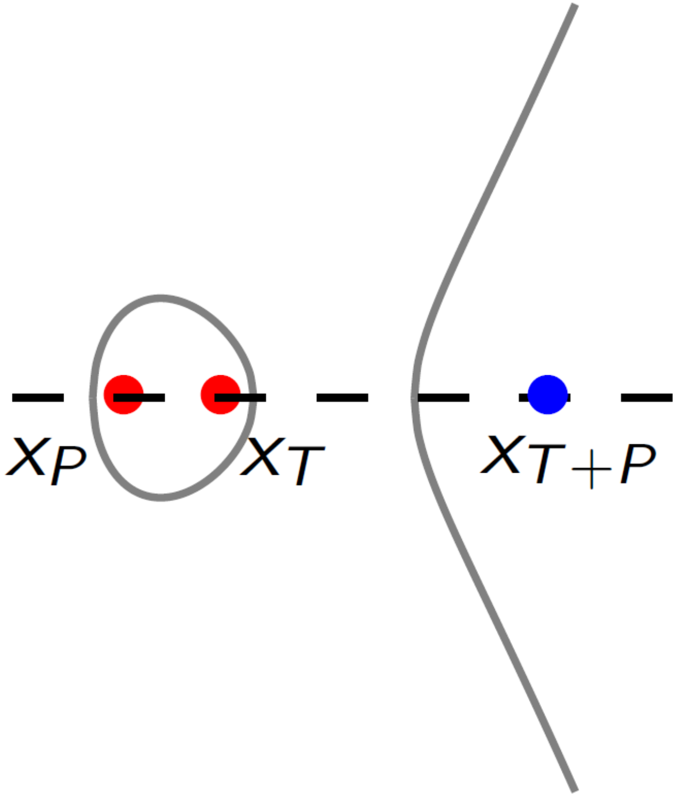


$(x_{[2]T}, y_{[2]T}) = DBL(x_T, y_T)$

$(x_{T+P}, y_{T+P}) = ADD(x_T, y_T, x_P, y_P)$

Montgomery's arithmetic on $By^2 = x^3 + Ax^2 + x$

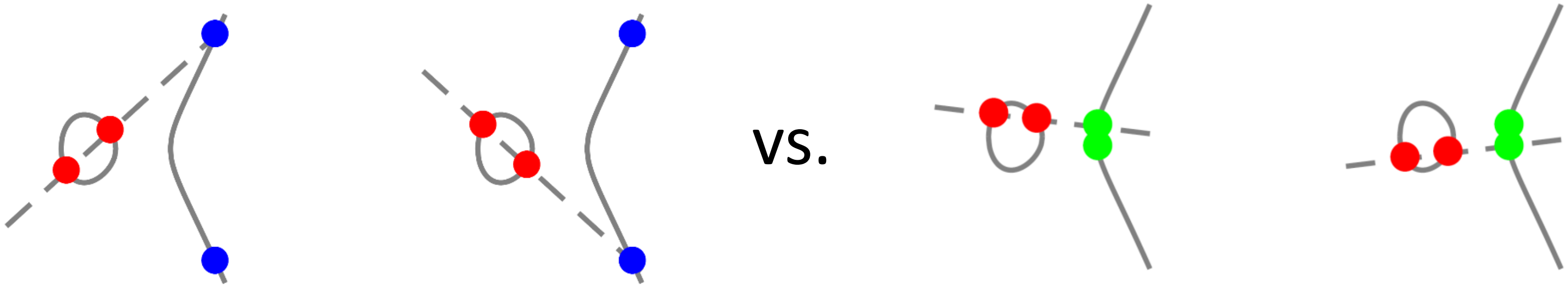




$x_{[2]T} = DBL(x_T)$

$x_{T+P} = DIFFADD(x_T, x_P, x_{T-P})$

Differential additions …

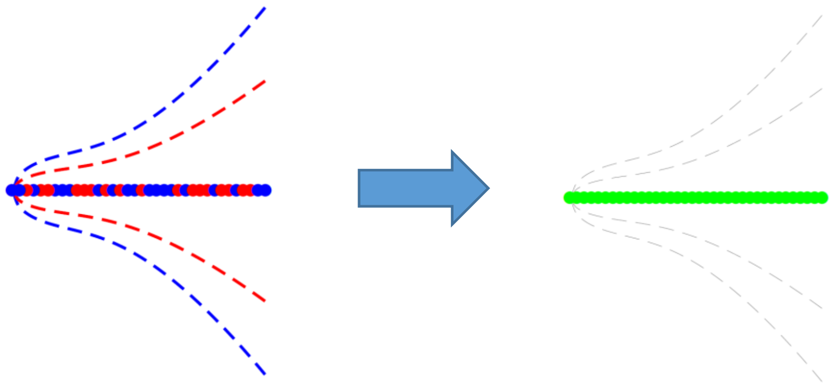


- “Opposite” y’s give different x-coordinate than “same-sign” y’s
- Decide with x-coordinate of difference: $x_{T+P} = DIFFADD(x_T, x_P, x_{T-P})$

… and the Montgomery ladder

- **Invariant:** in $x(P), k \mapsto x([k]P)$, keep this difference fixed as $x(P)$
- **Iteration:** at each intermediate step, we always have $x([m]P), x([m+1]P)$ … so we always add them and double one (depends on binary rep. of k) to preserve the invariant

Twist-security



- Ladder gives scalar multiplications on E: $By^2 = x^3 + Ax^2 + x$ as

$$x([k]P) = LADDER(x(P), k, A)$$

- Does not depend on B, so works on E': $B'y^2 = x^3 + Ax^2 + x$ for any B'
- Up to isomorphism, there are only two possibilities for fixed A:

  E and its quadratic twist E'

- So if E and E' are both secure, no need to check $P \in E$ for any $x(P) \in K$, as $LADDER(x, k, A)$ gives discrete log on E or E' for all $x \in K$
- **Twist-security only really useful when doing x-only computations, but why not have it anyway?**

Contents

PART I : CHOOSING CURVES
Speed-records and security hunches
Prime fields and modular reduction
Curve models and killing cofactors
Montgomery ladder and twist-security
Our chosen curves: the NUMS curves

PART II : IMPLEMENTING THEM
Constant-time implementations and recoding scalars
Exception-free algorithms and Weierstrass "completeness"
Performance numbers and practical considerations
Conclusions and recommendations

The NUMS curves

Security $s =$	Prime $p =$	Weierstrass $b =$	Twisted Edwards $d =$	Montgomery $A =$
128	$2^{256} - 189$	152961	15342 ⟷	-61370
192	$2^{384} - 317$	-34568	333194 ⟷	-1332778
256	$2^{512} - 569$	121243	637608 ⟷	-2550434

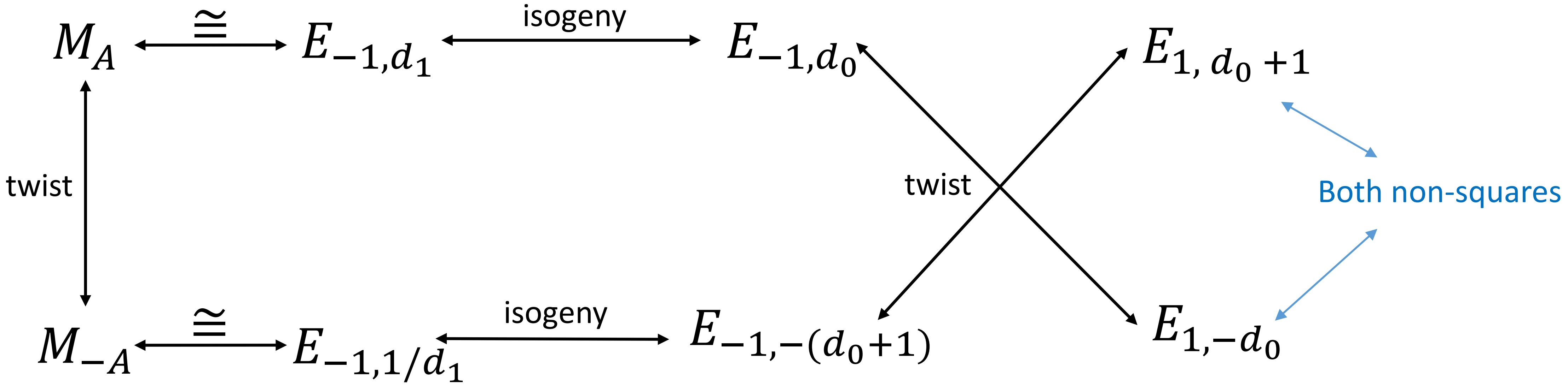
- **Primes:** Largest $p = 2^{2s} - \gamma \equiv 3 \bmod 4$
  (***fun fact:*** *in these cases, largest primes full stop*)
- **Weierstrass:** Smallest $|b|$ such that $\#E$ and $\#E'$ both prime
- **Twisted Edwards:** Smallest $d > 0$ such that $\#E$ and $\#E'$ both 4 times a prime, and $d > 0$ corresponds to $t > 0$.
- **Reminder:** there are 6 "chosen" curves above, but in paper 26 are benchmarked

Small constants all round for $p \equiv 3 \bmod 4$

$M_A: \; y^2 = x^3 + Ax^2 + x$ $\qquad$ $E_{a,d}: ax^2 + y^2 = 1 + dx^2y^2$

Searches minimize $|A|$ with $A \equiv 2 \bmod 4$

$d_1 = -\frac{A-2}{A+2}$ (big) $\qquad$ $d_0 = -\frac{A+2}{4}$ (small)



Upshot: search that minimizes Montgomery constant size also minimizes size of both twisted Edwards and Edwards constants (see Lemmas 1-3)

Contents

PART I : CHOOSING CURVES
Speed-records and security hunches
Prime fields and modular reduction
Curve models and killing cofactors
Montgomery ladder and twist-security
Our chosen curves: the NUMS curves

PART II : IMPLEMENTING THEM
Constant-time implementations and recoding scalars
Exception-free algorithms and Weierstrass "completeness"
Performance numbers and practical considerations
Conclusions and recommendations

Constant time implementations

- **Constant time:** all computations involving secret data must exhibit regular execution to provide protection against timing and cache attacks
- No data-dependent branches or table lookups depend on scalar k
- Most naïve version: *double-and-add* → *double-and-always-add*

$$k = [-, 0, 0, 1, 0, 1, \ldots]$$

double-and-always-add: *initialize* $Q \leftarrow P$			*[-,*
compute $[2]Q, [2]Q + P$		$Q \leftarrow [2]Q$	*0,*
compute $[2]Q, [2]Q + P$		$Q \leftarrow [2]Q$	*0,*
compute $[2]Q, [2]Q + P$		$Q \leftarrow [2]Q + P$	*1,*
compute $[2]Q, [2]Q + P$		$Q \leftarrow [2]Q$	*0,*
compute $[2]Q, [2]Q + P$		$Q \leftarrow [2]Q + P$	*1, ..*

Fixed-window recoding for variable-base

- "Always-add" obviously brings in solid performance penalty: adding twice as much as usual… **BUT** not when using bigger/optimal windows!!!

$w = 1$ […, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0,…]

$w = 5$ […, 1, 1, 0, 1, 0, | 1, 0, 1, 0, 1, | 0, 0, 0, 1, 0,…]

[…, 26, 21, 2,…]

…5 DBL's → ADD $([26]P)$ **→ 5 DBL's → ADD** $([21]P)$ **→ 5 DBL's → ADD** $([2]P)$**…**

- Basic/naïve: pre-compute and store P,[2]P,…,[30]P, [31]P

- Chances of 5 zeros in a row = 1/32, but we must still **always** add something…

Protected "odd-only" fixed-window recoding algorithm

- Window width w: recodes every odd scalar $k \in [1, r)$ into $(t+1)$ odd values, i.e. $k = (k_t, \ldots, k_0)$, where $t = \left\lceil \left(\frac{\log_2 r}{w}\right) \right\rceil$
- Each recoded value is an integer in $k_i \in \{\pm 1, \pm 3, \pm 5, \ldots, \pm 2^w - 1\}$ (only half the precomputed values needed, and there are no zeros)

- e.g. 256-bit scalars, $w = 5$ optimal for us, 53 windows:
  - precompute table $\{P, [3]P, [5]P, \ldots, [31]P\}$ (1 DBL, 15 ADDS)
  - select first value as $[k_t]P$
  - **5 DBL's**→**ADD**($[k_{t-1}]P$) → … → **5 DBL's** → **ADD** ($[k_0 P]$)

  Total: $52 \times 5 + 1 = 261$ DBL's, $52 + 16 = 68$ ADD's.

- Same total and sequence, whether $k = 1$, $k = r$, or anything in between

Much more to constant-time implementations

- **Identical sequence of operations is just the beginning…**

  e.g: recoding was for odd scalars only: negate every scalar, mask in the odd one, negate every "final" point, mask correct result…

  e.g: recoding the scalars themselves must be constant time

  e.g: must access/load every lookup element, every time, and mask out correct one

  see http://eprint.iacr.org/2014/130.pdf and http://research.microsoft.com/en-us/projects/nums/ for solutions to these problems and more…

- **The recoding is mathematically correct, and facilitates constant-time implementations, BUT only assuming the ECC formulas do their job!**

Contents

PART I : CHOOSING CURVES

Speed-records and security hunches

Prime fields and modular reduction

Curve models and killing cofactors

Montgomery ladder and twist-security

Our chosen curves: the NUMS curves

PART II : IMPLEMENTING THEM

Constant-time implementations and recoding scalars

Exception-free algorithms and Weierstrass "completeness"

Performance numbers and practical considerations

Conclusions and recommendations

Guaranteeing exception-free routines

- The running multiple $Q = [m]P$ of P could be one of the values $P, [3]P, \ldots, [2^w - 1]P$ in the lookup table, or their inverse

- Not a problem if addition formulas are complete, but recall that:

  (i) complete Edwards additions are not the fastest
  (ii) typical Weierstrass additions far from complete

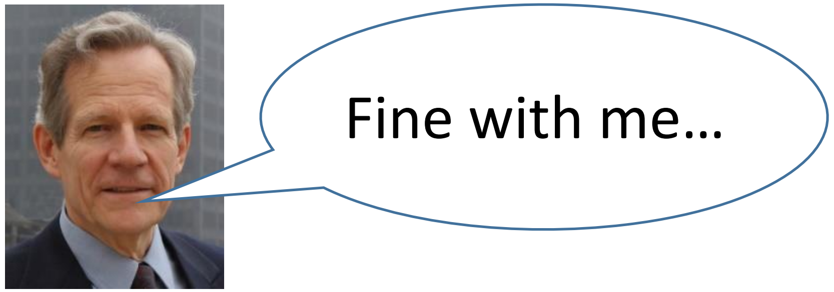
- Not only **variable-base** scenario $[k]P$ for P (as before), but **fixed-base** scenario where P is known (precomps mean larger lookup table – more potential trouble)

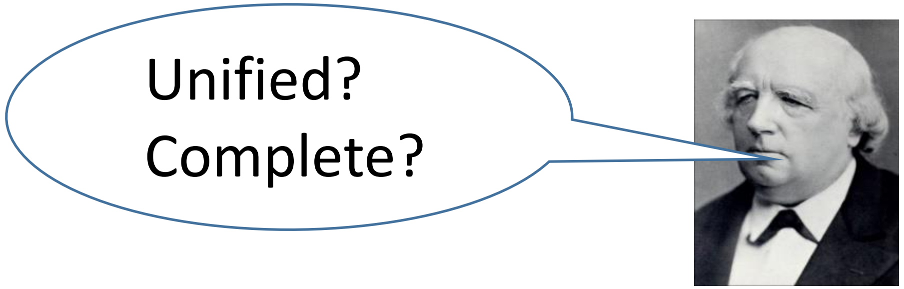
- Can only claim “constant-time” if all combinations of k and P compute $[k]P$ without exception

Guaranteeing exception-free routines

- **Propositions 4,6: (**under prior recoding) Weierstrass and twisted Edwards **variable-base** scalar multiplications will compute without exception if:
  fastest dedicated addition formulas are used throughout, except the final addition, which needs to be unified (for our proof to go through)

- **Propositions 5,7: (**under fixed-base recoding) Weierstrass and twisted Edwards fixed-base scalar multiplications will compute without exception if:
  complete additions are used throughout (for our proof to go through)





Weierstrass completeness

- **Impossibility Theorem (Bosma-Lenstra):** for general elliptic curves, we need to compute **at least two sets of explicit formulae** to guarantee every sum is computed:

  i.e. no f_X, f_Y, f_Z such that

$$X_3 = f_X(X_1, Y_1, Z_1, X_2, Y_2, Z_2)$$
$$Y_3 = f_Y(X_1, Y_1, Z_1, X_2, Y_2, Z_2)$$
$$Z_3 = f_Z(X_1, Y_1, Z_1, X_2, Y_2, Z_2)$$

  computes the correct sum $(X_3 : Y_3 : Z_3) = (X_1 : Y_1 : Z_1) + (X_2 : Y_2 : Z_2)$ for all points on a general curve

- Need (f_X, f_Y, f_Z) and $(f_X{'}, f_Y{'}, f_Z{'})$, where at least one set will always do the job…

Weierstrass completeness

- e.g. specialized to $y^2 = x^3 + ax + b$, and in homogeneous space, the sum $(X_1 : Y_1 : Z_1)$+$(X_2 : Y_2 : Z_2)$ will be at least one of $(X_3 : Y_3 : Z_3)$ or $(X_3{}' : Y_3{}' : Z_3{}')$:

$$
\begin{aligned}
X_3 &= (X_1Y_2 - X_2Y_1)(Y_1Z_2 + Y_2Z_1) - (X_1Z_2 - X_2Z_1)(a(X_1Z_2 + X_2Z_1) + 3bZ_1Z_2 - Y_1Y_2);\\
Y_3 &= -(3X_1X_2 + aZ_1Z_2)(X_1Y_2 - X_2Y_1) + (Y_1Z_2 - Y_2Z_1)(a(X_1Z_2 + X_2Z_1) + 3bZ_1Z_2 - Y_1Y_2);\\
Z_3 &= (3X_1X_2 + aZ_1Z_2)(X_1Z_2 - X_2Z_1) - (Y_1Z_2 + Y_2Z_1)(Y_1Z_2 - Y_2Z_1);\\
X_3' &= -(X_1Y_2 + X_2Y_1)(a(X_1Z_2 + X_2Z_1) + 3bZ_1Z_2 - Y_1Y_2) - (Y_1Z_2 + Y_2Z_1)(3b(X_1Z_2 + X_2Z_1) + a(X_1X_2 - aZ_1Z_2));\\
Y_3' &= Y_1^2Y_2^2 + 3aX_1^2X_2^2 - 2a^2X_1X_2Z_1Z_2 - (a^3 + 9b^2)Z_1Z_2^2 + (X_1Z_2 + X_2Z_1)(3b(3X_1X_2 - aZ_1Z_2) - a^2(X_2Z_1 + X_1Z_2));\\
Z_3' &= (3X_1X_2 + aZ_1Z_2)(X_1Y_2 + X_2Y_1) + (Y_1Z_2 + Y_2Z_1)(Y_1Y_2 + 3bZ_1Z_2 + a(X_1Z_2 + X_2Z_1)). \qquad (1)
\end{aligned}
$$

- For our $a = -3$ Weierstrass curves, our first attempt to optimize the above gave $\mathbf{22}\boldsymbol{M} + \mathbf{4}\boldsymbol{M_b}$ (compared to $\approx \mathbf{14}\boldsymbol{M}$ for dedicated projective additions)
- AND the true cost ratio would be far worse than the multiplications indicate

… there’s got to be a better way…

Weierstrass "pseudo-completeness"

- We give a "pseudo-complete'' addition algorithm for general Weierstrass curves
- Exploits similarity in doubling and addition formulas (two main cases)
- Resemblance to Chevallier-Mames, Ciet, and Joye: "Side-channel Atomicity", but they give separate routines – we merge into one with masking

Compare

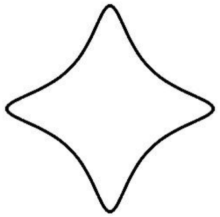
Algorithm 18 Complete (mixed) addition using masking and Jacobian/affine coordinates on prime-order Weierstrass curves E_b.

Input: $P, Q \in E_b(\mathbf{F}_p)$ such that $P = (X_1, Y_1, Z_1)$ is in Jacobian coordinates and $Q = (x_2, y_2)$ is in affine coordinates.

Output: $R = P + Q \in E_b(\mathbf{F}_p)$ in Jacobian coordinates. Computations marked with [∗] are implemented in constant time using masking.

$T[0] = \mathcal{O}$ $\{T[i] = (\tilde{X}_i, \tilde{Y}_i, \tilde{Z}_i)$ for $0 \le i < 4\}$
$T[1] = Q$
$t_2 = Z_1^2$
$t_3 = Z_1 \times t_2$
$t_1 = x_2 \times t_2$
$t_4 = y_2 \times t_3$
$t_1 = t_1 - X_1$
$t_4 = t_4 - Y_1$
index $= 3$
if $t_1 = 0$ **then** [∗]
    index $= 0$ $\{R = \mathcal{O}\}$
    if $t_4 = 0$ **then** index $= 2$ $\{R = 2P\}$ [∗]
if $P = \mathcal{O}$ **then** index $= 1$ $\{R = Q\}$ [∗]
mask $= 0$
if index $= 3$ **then** mask $= 1$ [∗]
$\{$case $P + Q$, else any other case$\}$
$t_3 = X_1 + t_2$
$t_6 = X_1 - t_2$
if mask $= 0$ **then** $t_2 = Y_1$ **else** $t_2 = t_1$ [∗]
$t_5 = t_2^2$
$t_1 = X_1 \times t_5$
$\tilde{Z}_2 = Z_1 \times t_2$
$\tilde{Z}_3 = \tilde{Z}_2$
if mask $\neq 0$ **then** $t_3 = t_2$ [∗]
if mask $\neq 0$ **then** $t_6 = t_5$ [∗]
$t_2 = t_3 \times t_6$
$t_3 = t_2/2$
$t_3 = t_2 + t_3$
if mask $\neq 0$ **then** $t_3 = t_4$ [∗]
$t_4 = t_3^2$
$t_4 = t_4 - t_1$
$\tilde{X}_2 = t_4 - t_1$
$\tilde{X}_3 = \tilde{X}_2 - t_2$
if mask $= 0$ **then** $t_4 = \tilde{X}_2$ **else** $t_4 = \tilde{X}_3$ [∗]
$t_1 = t_1 - t_4$
$t_4 = t_3 \times t_1$
if mask $= 0$ **then** $t_1 = t_5$ **else** $t_1 = Y_1$ [∗]
if mask $= 0$ **then** $t_2 = t_5$ [∗]
$t_3 = t_1 \times t_2$
$\tilde{Y}_2 = t_4 - t_3$
$\tilde{Y}_3 = \tilde{Y}_2$
$R = P[\text{index}]$ $(= (\tilde{X}_{\text{index}}, \tilde{Y}_{\text{index}}, \tilde{Z}_{\text{index}}))$ [∗]
return R

to



$$\left(\frac{x_1y_2+y_1x_2}{1+dx_1x_2y_1y_2}, \frac{y_1y_2-x_1x_2}{1-dx_1x_2y_1y_2}\right)$$

- Edwards elegance unrivalled, but this gets the job done for Weierstrass!
- Jac+aff (dedicated) = **8M+3S**, Jac+aff (complete-masking) = **8M+3S+ϵ** ($\epsilon \approx 20\%$)

Contents

PART I : CHOOSING CURVES

Speed-records and security hunches

Prime fields and modular reduction

Curve models and killing cofactors

Montgomery ladder and twist-security

Our chosen curves: the NUMS curves

PART II : IMPLEMENTING THEM

Constant-time implementations and recoding scalars

Exception-free algorithms and Weierstrass "completeness"

Performance numbers and practical considerations

Conclusions and recommendations

TLS handshake with PFS: ECDH(E)-ECDSA

Three scenarios

- **Variable-base:** $k, P \mapsto [k]P$ (P not known in advance)

  - both sides of static DH
  - half of ephemeral DH(E)
  - constant time (recoding as before, final addition unified)

- **Fixed-base** $k, P \mapsto [k]P$ (P known in advance)
  - other half of ephemeral DH(E)
  - ECDSA signing
  - constant time (fixed-base recoding, all additions complete)

- **Double-scalar** $a, b, P, Q \mapsto [a]P + [b]Q$ (P known in advance, Q not)

  - ECDSA verification
  - constant time unnecessary!

Security Level	Prime	Curve	Variable -base	Fixed -base	Double -scalar
128	$p = 2^{256} - 189$	Weierstrass twisted Edwards	270 216	107 82	289 231
192	$p = 2^{384} - 317$	Weierstrass twisted Edwards	714 588	252 201	758 614
256	$p = 2^{512} - 569$	Weierstrass twisted Edwards	1,504 1,242	488 391	1,596 1,308

- Fastest report NIST P-256 (Gueron & Krasnov ‘13): $\approx 400k$ cycles var-based
- Fixed-base may get a fair bit faster in all scenarios, unified/complete adds not necessary?? [*Hamburg, a few days ago, private communication*]
- No assembly above field layer (solid gains possible for our curves)
- Compare Curve25519 $\approx 194{,}000$ to twisted Edwards $\approx 216{,}000$ (sandy)

Contents

PART I : CHOOSING CURVES

Speed-records and security hunches

Prime fields and modular reduction

Curve models and killing cofactors

Montgomery ladder and twist-security

Our chosen curves: the NUMS curves

PART II : IMPLEMENTING THEM

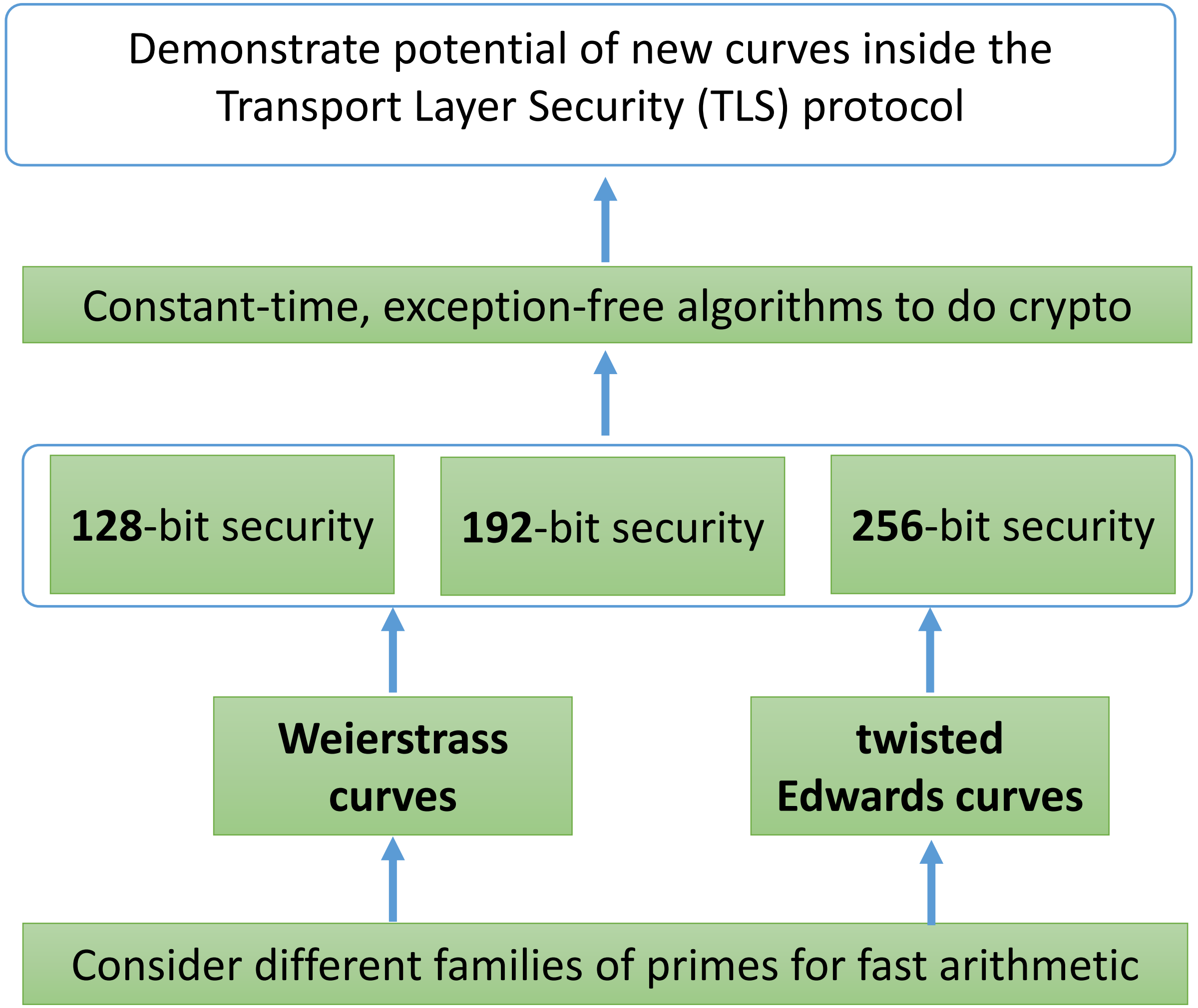
Constant-time implementations and recoding scalars

Exception-free algorithms and Weierstrass "completeness"

Performance numbers and practical considerations

Conclusions and recommendations

Our work (in a nutshell)



The sell: what did we do differently?

- **Modular/consistent implementation across three security levels**
  - twisted Edwards curves generated and implemented the same way
  - same for Weierstrass
- **Also considered/implemented new/better prime-order curves**
  - concrete performance comparison
  - true gauge on pros and cons of shifting to Edwards
- **Two different styles of primes/field arithmetic**
  - Montgomery and Pseudo-Mersenne
  - Stayed fixed on "full-length" Pseudo-Mersenne primes
- **Choose Edwards everywhere over Montgomery ladder**
  - Consistency and no real performance hit
  - More versatile

What could we do differently?

- **Define curves as Edwards, not twisted**
  - Douglas Stebila (8 Aug, 2014) on CFRG mailing list:
    "implementations [should] readily expose both a scalar point multiplication operation and a point addition operation"
  - Perhaps better to define as Edwards equipped with complete add (and optionally use Hamburg's isogeny trick?)
  - Fortunately for $3 \bmod 4$, we get minimal d in either form (just rewrite)
- **Remove $d > 0$ with $t > 0$ restriction**
  - Mike Hamburg (12 Aug, 2014) on CFRG mailing list:
    "If these requirements become final, then surely the complete curves mod the Microsoft primes with a=1 and no restriction on the sign of d (choose the one with q<p) should be in the running".
  - Unrestricted curves in our first preprint, imposed $d > 0$ in v2, go back?

… see also …

- Report:
  http://eprint.iacr.org/2014/130.pdf

- MSR ECC Library:
  http://research.microsoft.com/en-us/projects/nums/

- Specification of curve selection:
  http://research.microsoft.com/apps/pubs/default.aspx?id=219966

- IETF Internet Draft (authored by Benjamin Black)
  http://tools.ietf.org/html/draft-black-numscurves-02