An Analysis of High-Performance Primes at High-Security Levels

NIST Workshop on Elliptic Curve Cryptography Standards
June 11-12, 2015, Gaithersburg, USA

Patrick Longa
Microsoft Research

Zhe Liu University of Luxembourg
Hwajeong Seo Pusan National University

Our motivations

1. Curves that regain confidence and get wide acceptance
   – A simple and rigid generation procedure as the foundation
   – Compatibility with existing security levels
   – Design consistency across security levels

2. Curves that are efficient on multiple platforms
   – 8-bit, 32-bit, 64-bit platforms, with and without vectorization support

Performance is important, but should not take priority over goals of security and transparency

In this talk …

- Review a few implementation aspects that are relevant to achieve a robust curve selection.
- Compare the performance of the NUMS curves against the fastest handpicked curves without rigid prime generation.

High-security curves for the analysis

We consider *four* high-security twisted Edwards curves defined by:

$$E/\mathbb{F}_p\colon\ ax^2 + y^2 = 1 + dx^2y^2,$$

with quadratic twist E'.

curve	curve param. (a, d)	prime p	Prime modularity	co-factors (E, E')	bit-security
"NUMS" numsp384t1	$(1, -11556)$	$2^{384} - 317$	$3 (\mathrm{mod}\ 4)$	$(4, 4)$	191
"NUMS" numsp512t1	$(1, -78296)$	$2^{512} - 569$	$3 (\mathrm{mod}\ 4)$	$(4, 4)$	255
Ted37919	$(-1, 143305)$	$2^{379} - 19$	$5 (\mathrm{mod}\ 8)$	$(8, 4)$	188
Ed48817	$(1, 14695)$	$2^{488} - 17$	$3 (\mathrm{mod}\ 4)$	$(4, 4)$	243

High-security curves for the analysis

Curve design considerations:

- “numsp384t1” and “numsp512t1” produced using rigid NUMS generation (curve + prime). *These curves match standard security levels.*
- “Ted37919” and “Ed48817” produced using NUMS curve generation but primes were handpicked for efficiency purposes: *determine upper bound on the performance gap*

- All curves have minimal d in twisted Edwards form and minimal constant $(A+2)/4$ in their isogenous Montgomery form (minimal in absolute value).
- All curves support a complete addition law and are twist-secure.

Implementation aspects: consistency

- Fixing the same curve form across the different security levels can help in reducing security risks, reducing developer/maintenance work, and improving code size.

Example: twisted Edwards supports addition formulas that are incomplete or complete depending on the chosen values for the curve parameters.

The affine addition formula

$$(x_1, y_1) + (x_2, y_2) = \left(\frac{x_1 y_2 + y_1 x_2}{1 + d x_1 y_1 x_2 y_2}, \frac{y_1 y_2 - a x_1 x_2}{1 - d x_1 y_1 x_2 y_2}\right)^{*}$$

is complete if a is square and d is non-square (in $\mathbb{F}_p$),

BUT it's *incomplete* if a is non-square and d is square (in $\mathbb{F}_p$).

* This addition formula is from [Bernstein-Birkner-Joye-Lange-Peters 2008].

Implementation aspects: consistency

- Choosing the same prime form for different security levels can help in reducing developer/maintenance work, and improving code portability and compactness.

Example: all the base field primes used by the NUMS curves have:

- bitlengths with 64-bit alignment
- pseudo-Mersenne form

These design features enable highly-compact and portable field arithmetic, which is desirable in many applications (e.g., IoT).

Implementation aspects: consistency

Portable field multiplication that works for all *six* NUMS curves (Weierstrass and Edwards)

```
void field_mul(digit_t* op1, digit_t* op2, digit_t* res, CurveStruct curve){
    digit_t i, j, rem = 0, mask = (1 << NBITS)-1, t[NLIMBS_MAX] = {0};

    // Integer multiplication
    for (i = 0; i < NLIMBS; i++){
        for (j = 0; j < NLIMBS; j++)
            t[(i+j)-NLIMBS*((i+j)≥ NLIMBS)] += (((i+j)≥ NLIMBS)*(C_CURVE-1) + 1)*op1[i]*op2[j]; }

    // Reduction
    for (j = 0; j ≤ 2; j++){
        t[0] += ((rem+1)*C_CURVE*(j==1)) + ((rem-1) & C_CURVE*(j==2));
        for (i = 0; i < 2*(NLIMBS-2); i++){
            t[i+1] += (t[i] >> NBITS);
            t[i] &= mask; }
        rem = (t[2*(NLIMBS-2)] >> NBITS); }
    for (i = 0; i < NLIMBS; i++) res[i] = t[i];
}
```

- Field elements are represented as arrays of radix-8 (or radix-16) elements stored in 32-bit (64-bit resp.) signed integer datatypes, depending on the targeted architecture.

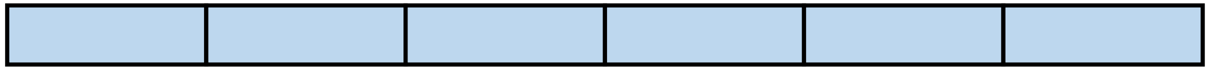
`NLIMBS (curve->NLIMBS)`: number of limbs used to represent a field element.

`NBITS (curve->NBITS)`: number of bits per limb.

`C_CURVE (curve->C_CURVE)`: constant c for a given prime $p = 2^{2s} - c$.

Implementation aspects: representation

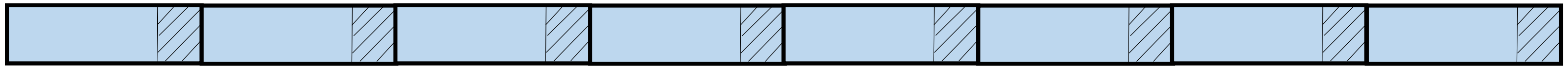
Canonical (or saturated):



limbs = $\lceil$field bitlength/computer word bitlength$\rceil$

No room for accumulating intermediate values without word spilling.

Extended (or unsaturated):



limbs $\geq \lceil$(field bitlength $+ \delta$)/computer word bitlength$\rceil$, for some $\delta > 0$

Extra room for accumulating intermediate values without word spilling.

Implementation aspects: representation

The problem:

Some platforms are more efficient with canonical representations **(e.g., AMD, Intel Atom, Intel Quark, ARM w/o NEON, microcontrollers)**, others are more efficient with extended representations **(e.g., Intel desktop/server, ARM with NEON).**

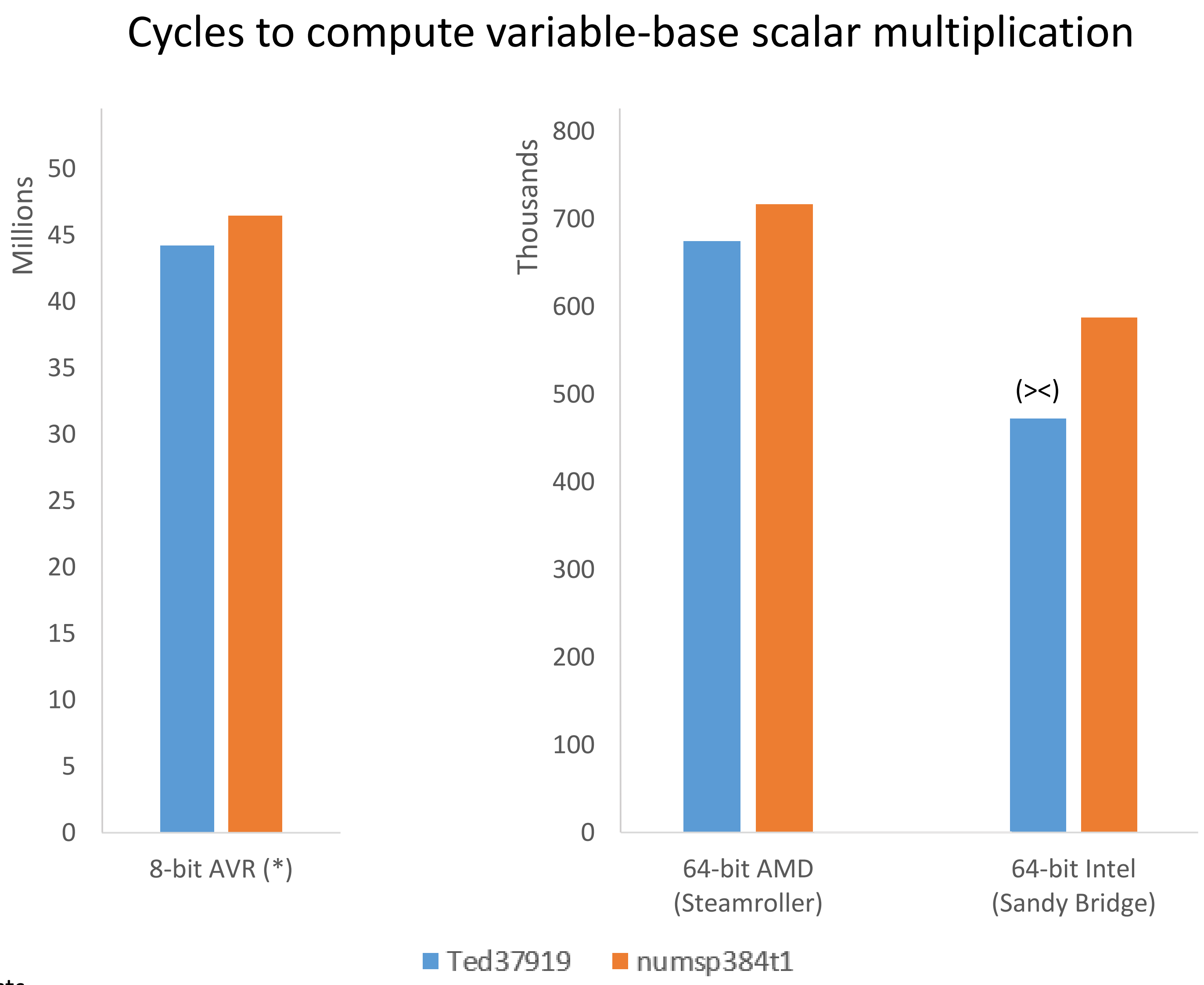
- Primes that are optimal on a certain platform might not be optimal on another platform.

Performance

- We used MSR ECCLib for evaluation of “numsp384t1” and “numsp512t1”.
- We wrote platform-specific implementations for “Ted37919” and “Ed48817”.
- Costs are reported for *variable-base scalar multiplication*.
- All implementations are fully protected against timing and cache attacks.

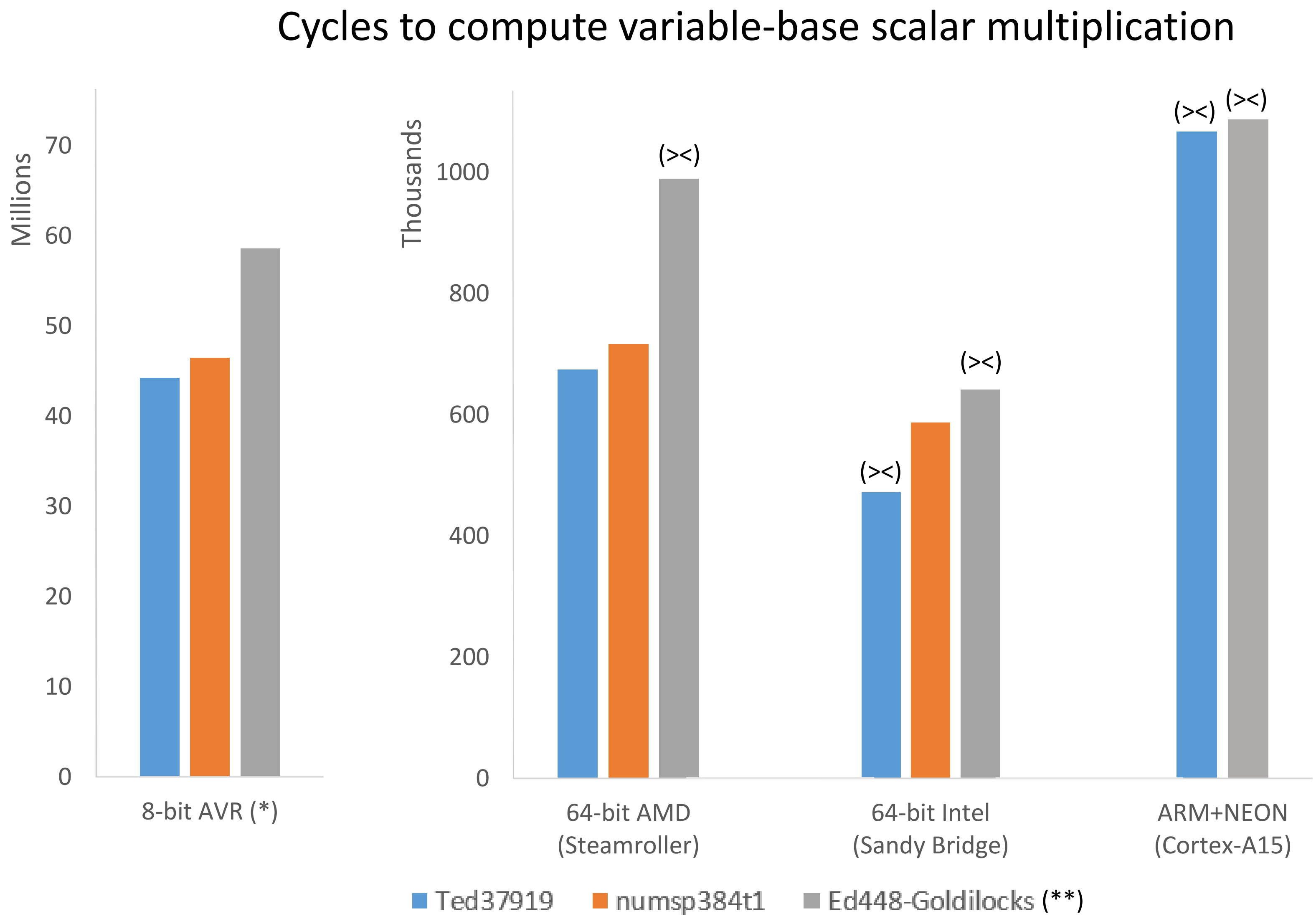
➢ Results for “numsp384t1” and “numsp512t1” from MSR ECCLib are somewhat in disadvantage. MSR ECCLib is a generic and portable library supporting a variety of curves, security levels, operating systems and devices.

Performance: cost of rigidity



(*) Extrapolated from field arithmetic costs.
Canonical representation used except when marked (><) .

Performance across different platforms



(*) Extrapolated from field arithmetic costs.
(**) Ed448-Goldilocks' results were obtained by running SUPERCOP on the 64-bit and ARM platforms.
Canonical representation used except when marked (><) .

Performance: comparison on x64 processors

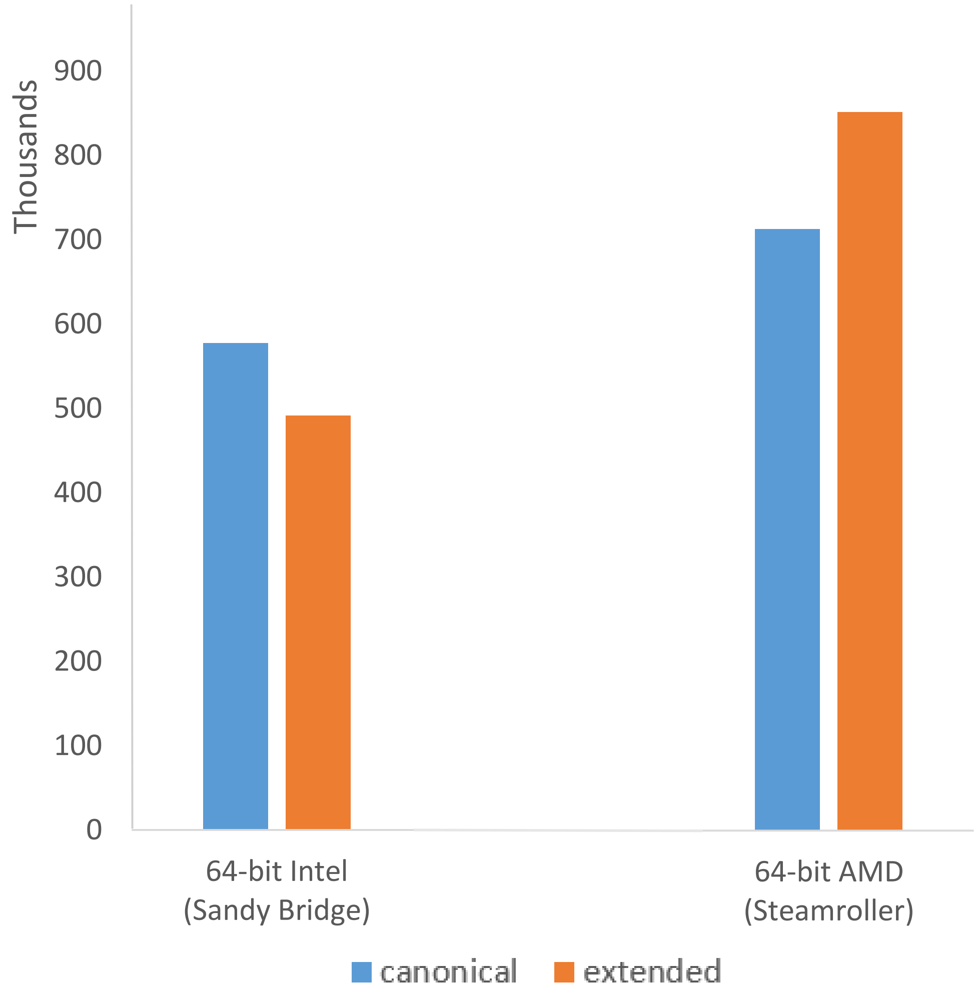
Cycles to compute variable-base scalar multiplication

Curve	prime p	bit security	Intel Sandy Bridge	Intel Haswell	AMD Steamroller
Ted37919	$2^{379} - 19$	188	491,000 (><)	407,000 (><)	675,000
numsp384t1	$\mathbf{2^{384} - 317}$	**191**	**611,000**	**504,000**	**717,000**
Ed448-Goldilocks	$2^{448} - 2^{224} - 1$	223	667,000 (><)	532,000 (><)	990,000 (><)
Ed48817	$2^{488} - 17$	243	1,091,000 (><)	916,000 (><)	1,319,000
numsp512t1	$\mathbf{2^{512} - 569}$	**255**	**1,320,000**	**1,136,000**	**1,523,000**

Canonical representation used except when marked (><) .

Performance: canonical versus extended

Cycles to compute variable-base scalar multiplication, curve “Ted37919”



Conclusions

- Performance should not be the top priority when selecting curves. Performance is technology- and application-dependent, and changes over time.

- We can choose curves that offer superior advantages in terms of rigidity, security, compatibility and consistency, and that also achieve good efficiency across multiple devices.

- Experimental results support selecting curves that follow standard security levels.

Conclusions

- We can innovate our way out of slow-performing implementations to fast performing. We can optimize hardware, etc. What we can't do is to add rigidity back later.

See also …

A brief discussion on selecting new elliptic curves
C. Costello, P. Longa, M. Naehrig, 2015.
http://research.microsoft.com/pubs/246915/NIST.pdf

Selecting Elliptic Curves for Cryptography: An Efficiency and Security Analysis,
J.W. Bos, C. Costello, P. Longa, M. Naehrig,
in Journal of Cryptographic Engineering, 2015.
http://eprint.iacr.org/2014/130

MSR ECCLib, version 2.0
http://research.microsoft.com/en-us/projects/nums/

An Analysis of High-Performance Primes at High-Security Levels

NIST Workshop on Elliptic Curve Cryptography Standards
June 11-12, 2015, Gaithersburg, USA

Patrick Longa
Microsoft Research

Zhe Liu	University of Luxembourg
Hwajeong Seo	Pusan National University