Post-Snowden Elliptic Curve Cryptography

Patrick Longa
Microsoft Research

Joppe Bos       NXP Semiconductors
Craig Costello  Microsoft Research
Michael Naehrig Microsoft Research
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...”

• TLS WG makes formal request to CFRG for new elliptic curves for usage in TLS

• NIST announces plans to host workshop to discuss new elliptic curves
Our motivations

1. **Curves that regain confidence and acceptance from public**
   - simple and rigid generation / “nothing up my sleeves”

2. **Improved performance and security for standard ECC algorithms and protocols**
   - new curve models
   - faster finite fields
   - side-channel resistance

Industry moving to Perfect Forward Secrecy (PFS) modes (e.g., ECDHE)

(e.g., see “Protecting Customer Data from Government Snooping” by Brad Smith, Microsoft General Counsel
“Nothing-Up-My-Sleeve” (NUMS) curve generation

Case with Edwards form, $p = 3 \pmod{4}$

Define the Edwards curve $E_d/\mathbb{F}_p: x^2 + y^2 = 1 + dx^2y^2$ with quadratic twist $E'_d/\mathbb{F}_p: x^2 + y^2 = 1 + (1/d)x^2y^2$.

1. Pick a prime $p$ according to well-defined efficiency/security criteria
2. Find smallest $|d| > 0$, with $d$ non-square in $\mathbb{F}_p$, such that $\#E_d = h \times r$ and $\#E'_d = h' \times r'$, where $r, r'$ are primes and $h = h' = 4$

Note: for both Edwards and twisted Edwards, minimal $d$ corresponds to minimal Montgomery constant $(A + 2)/4$ up to isogeny
“Nothing-Up-My-Sleeve” (NUMS) curve generation

Case with twisted Edwards form, \( p = 1 \pmod{4} \)

Define the twisted Edwards curve \( E_d/\mathbb{F}_p: -x^2 + y^2 = 1 + dx^2y^2 \) with quadratic twist \( E'_d/\mathbb{F}_p: -x^2 + y^2 = 1 + (1/d)x^2y^2 \).

1. Pick a prime \( p \) according to well-defined efficiency/security criteria
2. Find smallest \(|d| > 0\), with \( d \) non-square in \( \mathbb{F}_p \), such that \( \#E_d = h \times r \) and \( \#E'_d = h' \times r' \), where \( r, r' \) are primes and \( \{h, h'\} = \{4,8\} \)

---


2 In addition, care must be taken to ensure MOV degree and CM discriminant requirements.
“Nothing-Up-My-Sleeve” (NUMS) curve generation

- It can be easily adapted to other curve forms.

- There are several alternatives for primes: pseudo-random, pseudo-Mersenne, “Solinas” primes, etc.
  - Our original preference to balance rigidity, consistency and efficiency was to fix \( p = 2^{2s} - c \), where \( c \) is the smallest integer s.t. \( p \equiv 3 \mod 4 \) for \( s \in \{256, 384, 512\} \).
  - Later extended to \( p \equiv 1 \mod 4 \) to enable the use of complete twisted Edwards additions

But if efficiency is the main criteria:

How do we select primes?
Selecting primes: saturated vs. unsaturated arithmetic

**Saturated:**

\[ \text{# limbs} = \text{field bitlength/computer word bitlength} \]
No room for accumulating intermediate values without word spilling

**Unsaturated:**

\[ \text{# limbs} \geq \lceil (\text{field bitlength} + \delta) / \text{computer word bitlength} \rceil, \text{ for some } \delta > 0 \]
Extra room for accumulating intermediate values without word spilling
Selecting primes: saturated vs. unsaturated arithmetic

**Saturated:**
- More efficient when operations with carries are efficient, multiplication is relatively expensive (e.g., AMD, Intel Atom, Intel Quark, ARM w/o NEON, microcontrollers)
- More amenable for “generic” libraries, support for multiple curves
- Cleaner/easier-to-maintain curve arithmetic

**Unsaturated:**
- More efficient when instructions with carries are relatively expensive (e.g., Intel desktop/server)
- More efficient when using vector instructions (e.g., ARM with NEON)
- (When using incomplete reduction) requires specialized curve arithmetic. Bound analysis is required: error prone, errors are more difficult to catch
Comparison of x64 implementations
Unsaturated versus Saturated

Relative cost between Curve25519 amd64-51 (unsaturated) and amd64-64 (saturated). RED indicates amd64-64 is better

Intel Haswell (wintermute): 10%
Intel Ivy Bridge (hydra8): 6%
Intel Sandy Bridge (hydra7): 5%
Intel Atom (h8atom): -36%
AMD Piledriver (hydra9): -39%
AMD Bulldozer (hydra6): -38%
AMD Bobcat (h4e450): -47%

* Source: SUPERCOP, accessed 01/05/2015
A new high-security curve: Ted37919

Ted37919 is defined by the twisted Edwards curve

\[ E: -x^2 + y^2 = 1 + 143305x^2y^2 \]

defined over \( \mathbb{F}_p \) with \( p = 2^{379} - 19 \). \( \#E = 8r \), where \( r = 2^{376} - 212648873052802741983876663836064015775919150954032106379 \).

- Provides \(~188\) bits of security
- Minimal \( d \) in twisted Edwards form
- Minimal constant \((A + 2)/4\) in its isogenous Montgomery form
- Generated with the NUMS curve generation algorithm

- Implementation-friendly to both saturated and unsaturated arithmetic: truly high efficiency \textit{independent} of the platform for the 192-bit level
## Comparison with other high-security curves

Number of limbs for the implementation of different fields (64 and 32-bit CPU)

<table>
<thead>
<tr>
<th>Prime</th>
<th>64-bit limbs or 32-bit limbs</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^{379} - 19$ (Ted37919):</td>
<td>6 64-bit limbs or 12 32-bit limbs</td>
</tr>
<tr>
<td>$2^{389} - 21$ (*):</td>
<td>7 64-bit limbs or 13 32-bit limbs</td>
</tr>
<tr>
<td>$2^{414} - 17$ (Curve41417):</td>
<td>7 64-bit limbs or 13 32-bit limbs</td>
</tr>
<tr>
<td>$2^{448} - 2^{224} - 1$ (Goldilocks):</td>
<td>7 64-bit limbs or 14 32-bit limbs</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Prime</th>
<th>54/55-bit limbs or 25/26-bit limbs</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^{379} - 19$ (Ted37919):</td>
<td>7 54/55-bit limbs or 15 25/26-bit limbs</td>
</tr>
<tr>
<td>$2^{389} - 21$ (*):</td>
<td>7 55/56-bit limbs or 15 25/26-bit limbs</td>
</tr>
<tr>
<td>$2^{414} - 17$ (Curve41417):</td>
<td>8 51/52-bit limbs or 16 25/26-bit limbs</td>
</tr>
<tr>
<td>$2^{448} - 2^{224} - 1$ (Goldilocks):</td>
<td>8 56-bit limbs or 16 28-bit limbs</td>
</tr>
</tbody>
</table>

(*) The use of this prime has been discussed on the CFRG mailing list (e.g., see [http://www.ietf.org/mail-archive/web/cfrg/current/msg05733.html](http://www.ietf.org/mail-archive/web/cfrg/current/msg05733.html))
Comparison with other high-security curves

Cycles to compute scalar multiplication (on “unsaturated-friendly” platforms)

<table>
<thead>
<tr>
<th>Curve</th>
<th>bit security</th>
<th>Intel Sandy Bridge</th>
<th>Intel Haswell</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ted37919, $p = 2^{379} - 19$</td>
<td>187.8</td>
<td>494,000</td>
<td>410,000</td>
</tr>
<tr>
<td>Ed448-Goldilocks, $p = 2^{448} - 2^{224} - 1$ (*)</td>
<td>222.8</td>
<td>658,000</td>
<td>532,000</td>
</tr>
<tr>
<td>E-521, $p = 2^{521} - 1$</td>
<td>259.3</td>
<td>1,030,000</td>
<td>803,000</td>
</tr>
</tbody>
</table>

- Ted37919 implementation is very simple, no use of more complex algorithms such as Karatsuba.
- Pure C versions cost 558,000 and 467,000 cycles on Intel SB and Haswell, respectively.

(*) Source: SUPERCOP, accessed on 01/05/2015
Post-Snowden Elliptic Curve Cryptography

Q&A

Patrick Longa
Microsoft Research
http://research.microsoft.com/en-us/people/plonga/

Joppe Bos
NXP Semiconductors

Craig Costello
Microsoft Research

Michael Naehrig
Microsoft Research