Curve41417: fast, highly secure and implementation-friendly curve

Chitchanok Chuengsatiansup

Technische Universiteit Eindhoven

June 12, 2014

Joint work with Daniel J. Bernstein and Tanja Lange
Existing deployment of Curve41417
What is the goal of new crypto?

- Example of old crypto:
  - OpenSSL secp160r1 (*security level only* $2^{80}$)
    - least secure option supported by OpenSSL
    - ≈ 2.1 million Cortex-A8 cycles (not constant time)

Chitchanok Chuengsatiansup

Curve41417: fast, secure, implementation-friendly
What is the goal of new crypto?

- Example of old crypto:
  - OpenSSL secp160r1 *(security level only $2^{80}$)*
    - least secure option supported by OpenSSL
    - $\approx 2.1$ million Cortex-A8 cycles (not constant time)

- Best speed with acceptable security?
  - Curve25519 *(security level $2^{125}$)*
    - $\approx 0.5$ million Cortex-A8 cycles (constant time)
  - Kummer *(hyperelliptic, security level $2^{125}$)*
    - $\approx 0.3$ million Cortex-A8 cycles (constant time)
What is the goal of new crypto?

Example of old crypto:
- OpenSSL secp160r1 (security level only $2^{80}$)
  - least secure option supported by OpenSSL
  - $\approx 2.1$ million Cortex-A8 cycles (not constant time)

Best speed with acceptable security?
- Curve25519 (security level $2^{125}$)
  - $\approx 0.5$ million Cortex-A8 cycles (constant time)
- Kummer (hyperelliptic, security level $2^{125}$)
  - $\approx 0.3$ million Cortex-A8 cycles (constant time)

Best security with acceptable speed?
- Curve41417 (security level above $2^{200}$)
  - $\approx 1.8$ million Cortex-A8 cycles (constant time)
Design of Curve41417

- High-security elliptic curve (security level above $2^{200}$)
- Defined over prime field $\mathbb{F}_p$ where $p = 2^{414} - 17$
- In Edwards curve form

\[ x^2 + y^2 = 1 + 3617x^2y^2 \]
Design of Curve41417

- High-security elliptic curve (security level above $2^{200}$)
- Defined over prime field $\mathbb{F}_p$ where $p = 2^{414} - 17$
- In Edwards curve form

$$x^2 + y^2 = 1 + 3617x^2y^2$$

- IEEE P1363 criteria (large embedding degree, etc.)
- Large prime-order subgroup (cofactor 8)
- Twist secure (twist cofactor 8)
- 3617 is smallest value satisfying these criteria
Prime $2^{414} - 17$

- Extremely close to a power of 2
- Difference 17 has just two bits set
- $2^{414} \times \mod p$ computed as $16x + x$ with single shift-and-add
- 414 is divisible by 9, 18, 23, 46
- 416 (for 4p) is divisible by 8, 13, 16, 26, 32, 52
- With 32-bit words, wasted bandwidth under 1% ($13 \cdot 32 = 416$) allowing two extra bits for extension e.g., sign bit in a compressed point
Importance of prime choice

- NIST P-384
  - \( p = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1 \)
  - reduction requires 4 additions for radix \( 2^{32} \)
  - for other radix, implementor has a choice:

Note: count subtraction as addition
Importance of prime choice

- NIST P-384
  - $p = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$
  - reduction requires 4 additions for radix $2^{32}$
  - for other radix, implementor has a choice:
    - slower and much more complicated
    - more complicated and much slower

Note: count subtraction as addition
Importance of prime choice

- **NIST P-384**
  
  - $p = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$
  
  - Reduction requires 4 additions for radix $2^{32}$
  
  - For other radix, implementor has a choice:
    
    - Slower and much more complicated
    
    - More complicated and much slower

- **Curve41417**
  
  - $p = 2^{414} - 17$
  
  - Reduction requires 1 shift and 2 additions

**Note:** Count subtraction as addition
### Importance of curve choice

<table>
<thead>
<tr>
<th>Curve</th>
<th>DBL</th>
<th>ADD</th>
<th>mADD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short Weierstrass</td>
<td>8</td>
<td>16</td>
<td>11</td>
</tr>
<tr>
<td>Twisted Hessian</td>
<td>8</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>Twisted Edwards</td>
<td>7</td>
<td>8</td>
<td>7</td>
</tr>
</tbody>
</table>

**Note:** assuming best known coordinates

- mADD = mixed addition
- mDADD = mixed differential addition
### Importance of curve choice

<table>
<thead>
<tr>
<th>Curve</th>
<th>DBL</th>
<th>ADD</th>
<th>mADD</th>
<th>mDADD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short Weierstrass</td>
<td>8</td>
<td>16</td>
<td>11</td>
<td>-</td>
</tr>
<tr>
<td>Twisted Hessian</td>
<td>8</td>
<td>11</td>
<td>9</td>
<td>-</td>
</tr>
<tr>
<td>Twisted Edwards</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>-</td>
</tr>
<tr>
<td>Montgomery</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>5</td>
</tr>
</tbody>
</table>

**Note:** assuming best known coordinates

- mADD = mixed addition
- mDADD = mixed differential addition
Diffie–Hellman Key Exchange

\[ a \rightarrow aP \rightarrow a(bP) \leftarrow b \rightarrow bP \leftarrow b(aP) \]
Diffie–Hellman Key Exchange

\[
\begin{align*}
  a &\rightarrow b &\rightarrow c &\rightarrow d \\
  aP &\rightarrow bP &\rightarrow cP &\rightarrow dP \\
  a(bP) &\rightarrow b(aP) &\rightarrow c(dP) &\rightarrow d(cP) \\
  a(cP) &\rightarrow b(dP) &\rightarrow c(aP) &\rightarrow d(bP) \\
  a(dP) &\rightarrow b(cP) &\rightarrow c(bP) &\rightarrow d(aP)
\end{align*}
\]
main DH challenge: make \textit{variable-base} scalar mult as fast as possible
main DH challenge: make variable-base scalar mult as fast as possible
main DH challenge: make \textit{variable-base} scalar mult as fast as possible
main DH challenge: make **variable-base** scalar mult as fast as possible
Prevent software side-channel attack:
- constant-time
- no input-dependent branch
- no input-dependent array index

Constant-time table-lookup:
- read entire table
- select via arithmetic
  if c is 1, select tbl[i]
  if c is 0, ignore tbl[i]

\[
t = (t \cdot (1 - c)) + (\text{tbl}[i] \cdot (c))
\]
\[
t = (t \text{ and } (c - 1)) \text{ xor } (\text{tbl}[i] \text{ and } (-c))
\]
ECC Arithmetic

- Mix coordinate systems:
  - doubling: projective $X, Y, Z$
  - addition: extended $X, Y, Z, T$

  (See https://hyperelliptic.org/EFD/)

- Scalar multiplication:
  - signed fixed windows of width $w = 5$
    Example: $2345 = 10 \ 01001 \ 01001 \ 2$
  - precompute $0P, 1P, 2P, \ldots, 16P$
    also multiply $d = 3617$ to $T$ coordinate
  - compute $T$ only before addition
Example: scaling from 255-bit to 414-bit scalar multiplication
Example: scaling from 255-bit to 414-bit scalar multiplication

Schoolbook field multiplication
expected scalar multiplication scaling \((414/255)^3 \approx 4.3\)
Example: scaling from 255-bit to 414-bit scalar multiplication

Schoolbook field multiplication
expected scalar multiplication scaling \((414/255)^3 \approx 4.3\)

2-level reduced refine Karatsuba
actual performance scaling \((1.8/0.5) \approx 3.6\)
Curve41417

- Very fast
  - \( \approx 1.6 \text{ million cycles on FreeScale i.MX515} \)
  - \( \approx 1.8 \text{ million cycles on TI Sitara} \)

- Very high security (above \(2^{200}\))
  - also twist-secure

- Very flexible radix
  - support different sizes of limbs

- Very easy modular reduction
Curve41417

- Very fast
  - \( \approx 1.6 \text{ million cycles on FreeScale i.MX515} \)
  - \( \approx 1.8 \text{ million cycles on TI Sitara} \)

- Very high security (above \( 2^{200} \))
  - also twist-secure

- Very flexible radix
  - support different sizes of limbs

- Very easy modular reduction

- Real world deployment
  - “Blackphone has been added to the permanent collection at the world-renowned International Spy Museum in the gallery Weapons of Mass Disruption”
## Cost Comparison (Karatsuba)

<table>
<thead>
<tr>
<th>Level</th>
<th>Mult. 64-bit</th>
<th>Add 64-bit</th>
<th>Add 32-bit</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-level</td>
<td>256</td>
<td>15</td>
<td>0</td>
<td>256 + 8 + 0 = 264</td>
</tr>
<tr>
<td>1-level</td>
<td>192</td>
<td>59</td>
<td>16</td>
<td>192 + 30 + 4 = 226</td>
</tr>
<tr>
<td>2-level</td>
<td>144</td>
<td>119</td>
<td>40</td>
<td>144 + 60 + 10 = 214</td>
</tr>
<tr>
<td>3-level</td>
<td>108</td>
<td>191</td>
<td>76</td>
<td>108 + 96 + 19 = 223</td>
</tr>
</tbody>
</table>

Note: use multiply-add instructions

Recall:
- 1 cycle per multiplication
- 0.5 cycle per 64-bit addition
- 0.25 cycle per 32-bit addition
## Cost Comparison (refined Karatsuba)

<table>
<thead>
<tr>
<th>Level</th>
<th>Mult.</th>
<th>Add</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>64-bit</td>
<td>32-bit</td>
</tr>
<tr>
<td>0-level</td>
<td>256</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>1-level</td>
<td>192</td>
<td>52</td>
<td>16</td>
</tr>
<tr>
<td>2-level</td>
<td>144</td>
<td>103</td>
<td>40</td>
</tr>
<tr>
<td>3-level</td>
<td>108</td>
<td>166</td>
<td>76</td>
</tr>
</tbody>
</table>

Note: use multiply-add instructions

Recall:
- 1 cycle per multiplication
- 0.5 cycle per 64-bit addition
- 0.25 cycle per 32-bit addition
Cost Comparison (reduced refined Karatsuba)

<table>
<thead>
<tr>
<th>Level</th>
<th>Mult.</th>
<th>Add</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>64-bit</td>
<td>32-bit</td>
</tr>
<tr>
<td>0-level</td>
<td>256</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>1-level</td>
<td>192</td>
<td>45</td>
<td>16</td>
</tr>
<tr>
<td>2-level</td>
<td>144</td>
<td>96</td>
<td>40</td>
</tr>
<tr>
<td>3-level</td>
<td>108</td>
<td>159</td>
<td>76</td>
</tr>
</tbody>
</table>

Note: use multiply-add instructions

Recall:

1 cycle per multiplication
0.5 cycle per 64-bit addition
0.25 cycle per 32-bit addition