Curve41417: Karatsuba revisited

Chitchanok Chuengsatiansup

Technische Universiteit Eindhoven

September 25, 2014

Joint work with Daniel J. Bernstein and Tanja Lange

Real-World Application

Performance budget

- This talk focuses on:
  - $n, P \mapsto nP$
  - ARM Cortex-A8

Performance budget

- This talk focuses on:
  - $n, P \mapsto nP$
  - ARM Cortex-A8
- OpenSSL `secp160-r1` (security level only 2^{80})
  - least secure option supported by OpenSSL
  - $\approx$ 2.1 million cycles on FreeScale i.MX515
  - $\approx$ 2.1 million cycles on TI Sitara

Performance budget

- This talk focuses on:
  - $n, P \mapsto nP$
  - ARM Cortex-A8
- OpenSSL `secp160-r1` (security level only 2^{80})
  - least secure option supported by OpenSSL
  - $\approx$ 2.1 million cycles on FreeScale i.MX515
  - $\approx$ 2.1 million cycles on TI Sitara
- Curve41417 (security level above 2^{200})
  - $\approx$ 1.6 million cycles on FreeScale i.MX515
  - $\approx$ 1.8 million cycles on TI Sitara

Design of Curve41417

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

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

- Large prime-order subgroup (cofactor 8)
- IEEE P1363 criteria (large embedding degree, etc.)
- Twist secure, i.e., twist of Curve41417 also secure

Side-Channel Attack

- Prevent software side-channel attack:
  - constant-time
  - no input-dependent branch
  - no input-dependent array index

Side-Channel Attack

- 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]

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

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 $w = 5$
  - precompute $0P, 1P, 2P, \ldots, 16P$
    also multiply $d = 3617$ to T coordinate
  - special first doubling
  - compute T only before addition

ARM Cortex-A8 Vector Unit

- 128-bit vector
- Arithmetic and load/store unit can perform in parallel
- Operate in parallel on vectors of four 32-bit integers or two 64-bit integers
- Each cycle produces:

  four 32-bit integer additions: $a_0+b_0, a_1+b_1, a_2+b_2, a_3+b_3$

  or

  two 64-bit integer additions: c_0+d_0, c_1+d_1

  or

  one multiply-add instruction: $a_0 b_0 + c_0$

  where a_i, b_i are 32- and c_i, d_i are 64-bit integers

Redundant Representation

- Use non-integer radix $2^{414/16} = 2^{25.875}$
- Decompose integer f modulo $2^{414} - 17$ into 16 integer pieces
- Write f as

$$
\begin{array}{llll}
f_0 + & 2^{26} f_1 + & 2^{52} f_2 + & 2^{78} f_3 + \\
2^{104} f_4 + & 2^{130} f_5 + & 2^{156} f_6 + & 2^{182} f_7 + \\
2^{207} f_8 + & 2^{233} f_9 + & 2^{259} f_{10} + & 2^{285} f_{11} + \\
2^{311} f_{12} + & 2^{337} f_{13} + & 2^{363} f_{14} + & 2^{389} f_{15}
\end{array}
$$

Carries

- Goal: Bring each limb down to 26 or 25 bits

Carries

- Goal: Bring each limb down to 26 or 25 bits
- Typical carry chain:
  $m_0 \to m_1 \to m_2 \to \cdots \to m_{14} \to m_{15} \to m_0 \to m_1$

Carries

- Goal: Bring each limb down to 26 or 25 bits
- Typical carry chain:
  $m_0 \to m_1 \to m_2 \to \cdots \to m_{14} \to m_{15} \to m_0 \to m_1$
- Increase throughput:

Carries

- Goal: Bring each limb down to 26 or 25 bits
- Typical carry chain:
  $m_0 \rightarrow m_1 \rightarrow m_2 \rightarrow \cdots \rightarrow m_{14} \rightarrow m_{15} \rightarrow m_0 \rightarrow m_1$
- Increase throughput:
  $m_0 \rightarrow m_1$
  $m_8 \rightarrow m_9$

Carries

- Goal: Bring each limb down to 26 or 25 bits
- Typical carry chain:
  $m_0 \to m_1 \to m_2 \to \cdots \to m_{14} \to m_{15} \to m_0 \to m_1$
- Increase throughput:
  $m_0 \to m_1 \to m_2$
  $m_8 \to m_9 \to m_{10}$

Carries

- Goal: Bring each limb down to 26 or 25 bits
- Typical carry chain:
  $m_0 \to m_1 \to m_2 \to \cdots \to m_{14} \to m_{15} \to m_0 \to m_1$
- Increase throughput:
  $m_0 \to m_1 \to m_2 \to m_3 \to m_4 \to m_5 \to m_6 \to m_7 \to m_8 \to m_9$
  $m_8 \to m_9 \to m_{10} \to m_{11} \to m_{12} \to m_{13} \to m_{14} \to m_{15} \to m_0 \to m_1$

Carries

- Goal: Bring each limb down to 26 or 25 bits
- Typical carry chain:
  $m_0 \to m_1 \to m_2 \to \cdots \to m_{14} \to m_{15} \to m_0 \to m_1$
- Increase throughput:
  $m_0 \to m_1 \to m_2 \to m_3 \to m_4 \to m_5 \to m_6 \to m_7 \to m_8 \to m_9$
  $m_8 \to m_9 \to m_{10} \to m_{11} \to m_{12} \to m_{13} \to m_{14} \to m_{15} \to m_0 \to m_1$
- Decrease latency:

Carries

- Goal: Bring each limb down to 26 or 25 bits
- Typical carry chain:
  $m_0 \to m_1 \to m_2 \to \cdots \to m_{14} \to m_{15} \to m_0 \to m_1$
- Increase throughput:
  $m_0 \to m_1 \to m_2 \to m_3 \to m_4 \to m_5 \to m_6 \to m_7 \to m_8 \to m_9$
  $m_8 \to m_9 \to m_{10} \to m_{11} \to m_{12} \to m_{13} \to m_{14} \to m_{15} \to m_0 \to m_1$
- Decrease latency:
  $m_0 \to m_1$
  $m_8 \to m_9$

Carries

- Goal: Bring each limb down to 26 or 25 bits
- Typical carry chain:
  $m_0 \to m_1 \to m_2 \to \cdots \to m_{14} \to m_{15} \to m_0 \to m_1$
- Increase throughput:
  $m_0 \to m_1 \to m_2 \to m_3 \to m_4 \to m_5 \to m_6 \to m_7 \to m_8 \to m_9$
  $m_8 \to m_9 \to m_{10} \to m_{11} \to m_{12} \to m_{13} \to m_{14} \to m_{15} \to m_0 \to m_1$
- Decrease latency:
  $m_0 \to m_1$
  $m_8 \to m_9$
  $m_4 \to m_5$
  $m_{12} \to m_{13}$

Carries

- Goal: Bring each limb down to 26 or 25 bits
- Typical carry chain:
  $m_0 \to m_1 \to m_2 \to \cdots \to m_{14} \to m_{15} \to m_0 \to m_1$
- Increase throughput:
  $m_0 \to m_1 \to m_2 \quad \to m_3 \to m_4 \to m_5 \to m_6 \to m_7 \to m_8 \to m_9$
  $m_8 \to m_9 \to m_{10} \to m_{11} \to m_{12} \to m_{13} \to m_{14} \to m_{15} \to m_0 \to m_1$
- Decrease latency:
  $m_0 \to m_1 \to m_2$
  $m_8 \to m_9 \to m_{10}$
  $m_4 \to m_5$
  $m_{12} \to m_{13}$

Carries

- Goal: Bring each limb down to 26 or 25 bits
- Typical carry chain:
  $m_0 \to m_1 \to m_2 \to \cdots \to m_{14} \to m_{15} \to m_0 \to m_1$
- Increase throughput:
  $m_0 \to m_1 \to m_2 \to m_3 \to m_4 \to m_5 \to m_6 \to m_7 \to m_8 \to m_9$
  $m_8 \to m_9 \to m_{10} \to m_{11} \to m_{12} \to m_{13} \to m_{14} \to m_{15} \to m_0 \to m_1$
- Decrease latency:
  $m_0 \to m_1 \to m_2$
  $m_8 \to m_9 \to m_{10}$
  $m_4 \to m_5 \to m_6$
  $m_{12} \to m_{13} \to m_{14}$

Carries

- Goal: Bring each limb down to 26 or 25 bits
- Typical carry chain:
  $m_0 \to m_1 \to m_2 \to \cdots \to m_{14} \to m_{15} \to m_0 \to m_1$
- Increase throughput:
  $m_0 \to m_1 \to m_2 \to m_3 \to m_4 \to m_5 \to m_6 \to m_7 \to m_8 \to m_9$
  $m_8 \to m_9 \to m_{10} \to m_{11} \to m_{12} \to m_{13} \to m_{14} \to m_{15} \to m_0 \to m_1$
- Decrease latency:
  $m_0 \to m_1 \to m_2 \to m_3 \to m_4 \to m_5$
  $m_8 \to m_9 \to m_{10} \to m_{11} \to m_{12} \to m_{13}$
  $m_4 \to m_5 \to m_6 \to m_7 \to m_8 \to m_9$
  $m_{12} \to m_{13} \to m_{14} \to m_{15} \to m_0 \to m_1$

Level of Karatsuba

- Karatsuba splits 1 ($2n \times 2n$) into 3 ($n \times n$)

Level of Karatsuba

- Karatsuba splits 1 $(2n \times 2n)$ into 3 $(n \times n)$
- Zero-level Karatsuba (Schoolbook)
  e.g. for 16 limbs: $16 \times 16 = 256$

Level of Karatsuba

- Karatsuba splits 1 $(2n \times 2n)$ into 3 $(n \times n)$
- Zero-level Karatsuba (Schoolbook)
  e.g. for 16 limbs: $16 \times 16 = 256$
- One-level Karatsuba
  e.g.: $16 \times 16 \rightarrow 3 \cdot (8 \times 8) +$ some additions
  $= 192 +$ some additions

Level of Karatsuba

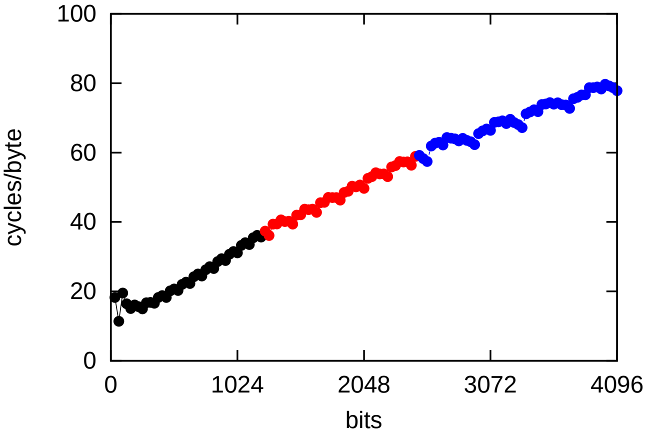
- Karatsuba splits 1 $(2n \times 2n)$ into 3 $(n \times n)$
- Zero-level Karatsuba (Schoolbook)
  e.g. for 16 limbs: $16 \times 16 = 256$
- One-level Karatsuba
  e.g.: $16 \times 16 \rightarrow 3 \cdot (8 \times 8)$ + some additions
  $= 192$ + some additions
- Two-level Karatsuba
  e.g.: $3 \cdot (8 \times 8) \rightarrow 3 \cdot (3 \cdot (4 \times 4))$ + even more additions
  $= 144$ + even more additions

Level of Karatsuba

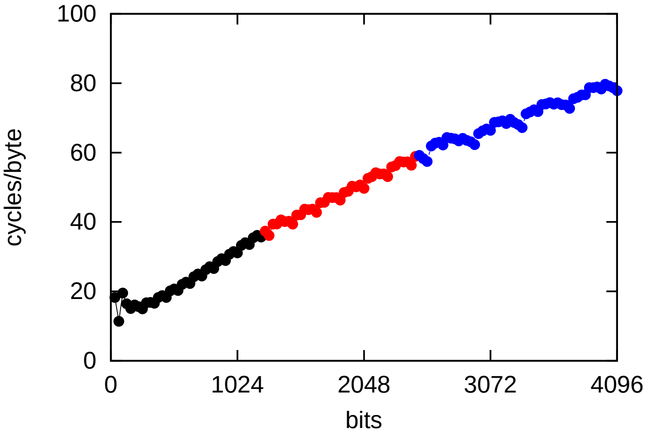
- Karatsuba splits 1 $(2n \times 2n)$ into 3 $(n \times n)$
- Zero-level Karatsuba (Schoolbook)
  e.g. for 16 limbs: $16 \times 16 = 256$
- One-level Karatsuba
  e.g.: $16 \times 16 \to 3 \cdot (8 \times 8)$ + some additions
  $= 192$ + some additions
- Two-level Karatsuba
  e.g.: $3 \cdot (8 \times 8) \to 3 \cdot (3 \cdot (4 \times 4))$ + even more additions
  $= 144$ + even more additions

- What is the zero-level/one-level cutoff for number of limbs?

GMP's cutoffs for Karatsuba

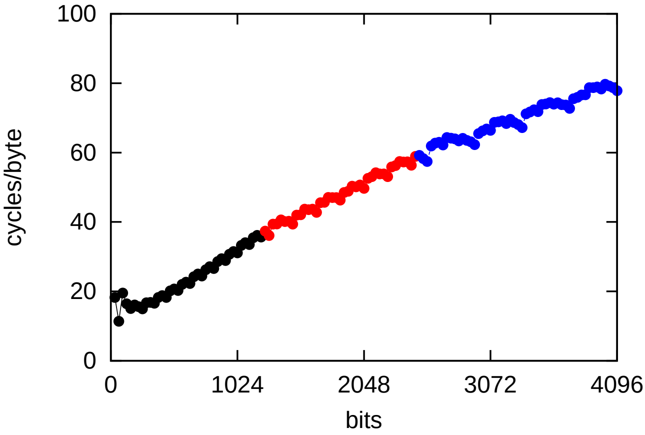


GMP's cutoffs for Karatsuba



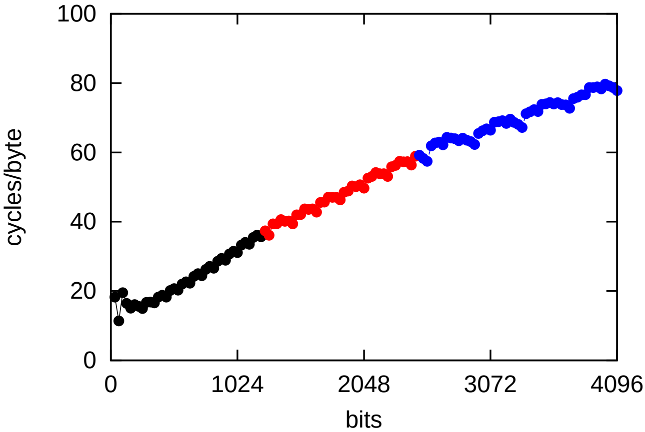
- GMP 6.0.0a library chooses 1248 bits on ARM Cortex-A8

GMP's cutoffs for Karatsuba



- GMP 6.0.0a library chooses 1248 bits on ARM Cortex-A8
- We reduce cutoff via improvements to Karatsuba

GMP's cutoffs for Karatsuba



- GMP 6.0.0a library chooses 1248 bits on ARM Cortex-A8
- We reduce cutoff via improvements to Karatsuba
- We reduce cutoff via redundant representation

Polynomial Multiplication

- Goal: Compute $P = AB$
  given $A = a_0 + a_1 t^n$ and $B = b_0 + b_1 t^n$
- Method 1: schoolbook
  $P = a_0 b_0 + (a_0 b_1 + a_1 b_0) t^n + a_1 b_1 t^{2n}$
- Method 2: Karatsuba ($8n-4$ additions)
  $P = a_0 b_0 + ((a_0+a_1)(b_0+b_1) - a_0 b_0 - a_1 b_1) t^n + a_1 b_1 t^{2n}$
- Method 3: refined Karatsuba ($7n-3$ additions)
  $P = (a_0 b_0 - a_1 b_1 t^n)(1 - t^n) + (a_0 + a_1)(b_0 + b_1) t^n$

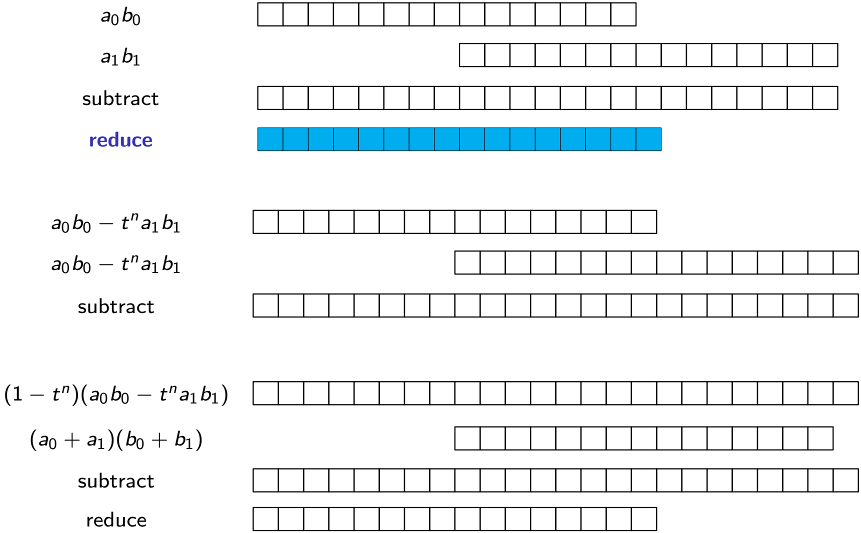
Polynomial Multiplication $\bmod Q$

- Goal: Compute $P = AB \bmod Q$
  given $A = a_0 + a_1 t^n$ and $B = b_0 + b_1 t^n$
- Method 1: schoolbook
  $P = a_0 b_0 + (a_0 b_1 + a_1 b_0) t^n + a_1 b_1 t^{2n} \bmod Q$
- Method 2: Karatsuba ($8n$−4 additions)
  $P = a_0 b_0 + ((a_0 + a_1)(b_0 + b_1) - a_0 b_0 - a_1 b_1) t^n + a_1 b_1 t^{2n} \bmod Q$
- Method 3: refined Karatsuba ($7n$−3 additions)
  $P = (a_0 b_0 - a_1 b_1 t^n)(1 - t^n) + (a_0 + a_1)(b_0 + b_1) t^n \bmod Q$

Polynomial Multiplication $\bmod Q$

- Goal: Compute $P = AB \bmod Q$
  given $A = a_0 + a_1 t^n$ and $B = b_0 + b_1 t^n$
- Method 1: schoolbook
  $P = a_0 b_0 + (a_0 b_1 + a_1 b_0) t^n + a_1 b_1 t^{2n} \bmod Q$
- Method 2: Karatsuba ($8n-4$ additions)
  $P = a_0 b_0 + ((a_0 + a_1)(b_0 + b_1) - a_0 b_0 - a_1 b_1) t^n + a_1 b_1 t^{2n} \bmod Q$
- Method 3: refined Karatsuba ($7n-3$ additions)
  $P = (a_0 b_0 - a_1 b_1 t^n)(1 - t^n) + (a_0 + a_1)(b_0 + b_1) t^n \bmod Q$
- Method 4: reduced refined Karatsuba ($6n-2$ additions) (new)
  $P = (a_0 b_0 - a_1 b_1 t^n \bmod Q)(1 - t^n) + (a_0 + a_1)(b_0 + b_1) t^n \bmod Q$

Reduced Refined Karatsuba



Cost Comparison (Karatsuba)

Level	Mult.	Add		Cost
		64-bit	32-bit	
0-level	256	15	0	$256+\ 8+\ 0=264$
1-level	192	59	16	$192+30+\ 4=226$
2-level	144	119	40	$144+60+10=214$
3-level	108	191	76	$108+96+19=223$

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 (Karatsuba)

Level	Mult.	Add 64-bit	Add 32-bit	Cost
0-level	256	15	0	$256+8+0=264$
1-level	192	59	16	$192+30+4=226$
2-level	144	119	40	$144+60+10=214$
3-level	108	191	76	$108+96+19=223$

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)

Level	Mult.	Add		Cost
		64-bit	32-bit	
0-level	256	15	0	$256+\ 8+\ 0 = 264$
1-level	192	52	16	$192+26+\ 4 = 222$
2-level	144	103	40	$144+52+10 = 206$
3-level	108	166	76	$108+83+19 = 210$

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)

Level	Mult.	Add 64-bit	Add 32-bit	Cost
0-level	256	15	0	$256 + 8 + 0 = 264$
1-level	192	52	16	$192 + 26 + 4 = 222$
2-level	144	103	40	$144 + 52 + 10 = 206$
3-level	108	166	76	$108 + 83 + 19 = 210$

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)

Level	Mult.	Add 64-bit	Add 32-bit	Cost
0-level	256	15	0	$256+\ 8+\ 0=264$
1-level	192	45	16	$192+23+\ 4=219$
2-level	144	96	40	$144+48+10=202$
3-level	108	159	76	$108+80+19=207$

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)

Level	Mult.	Add		Cost
		64-bit	32-bit	
0-level	256	15	0	$256+\ 8+\ 0 = 264$
1-level	192	45	16	$192+23+\ 4 = 219$
2-level	144	96	40	$144+48+10 = 202$
3-level	108	159	76	$108+80+19 = 207$

Note: use multiply-add instructions

Recall:

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

Performance Comparison

- OpenSSL

curve	# cycle on i.MX515	# cycle on Sitara
secp160r1	$\approx$ 2.1 million	$\approx$ 2.1 million
nistp192	$\approx$ 2.9 million	$\approx$ 2.8 million
nistp224	$\approx$ 4.0 million	$\approx$ 3.9 million
nistp256	$\approx$ 4.0 million	$\approx$ 3.9 million
nistp384	$\approx$ 13.3 million	$\approx$ 13.2 million
nistp521	$\approx$ 29.7 million	$\approx$ 29.7 million

- Curve41417 (security level above 2^{200})
  - $\approx$ 1.6 million cycles on FreeScale i.MX515
  - $\approx$ 1.8 million cycles on TI Sitara

Performance Comparison

- OpenSSL

curve	# cycle on i.MX515	# cycle on Sitara
secp160r1	≈ 2.1 million	≈ 2.1 million
nistp192	≈ 2.9 million	≈ 2.8 million
nistp224	≈ 4.0 million	≈ 3.9 million
nistp256	≈ 4.0 million	≈ 3.9 million
nistp384	≈ 13.3 million	≈ 13.2 million
nistp521	≈ 29.7 million	≈ 29.7 million

- Curve41417 (security level above 2^{200})
  - ≈ 1.6 million cycles on FreeScale i.MX515
  - ≈ 1.8 million cycles on TI Sitara

- Coming soon: Intel Haswell implementation with Sebastian Verschoor