Sandy2x: New Curve25519 Speed Records

Tung Chou

Technische Universiteit Eindhoven, The Netherlands

October 13, 2015

X25519 and Ed25519

X25519

- ECDH scheme
- public keys and shared secrets are points on the Montgomery curve

$$y^2 = x^3 + 486662x^2 + x$$

  over $\mathbb{F}_{2^{255}-19}$
- by Bernstein, 2006

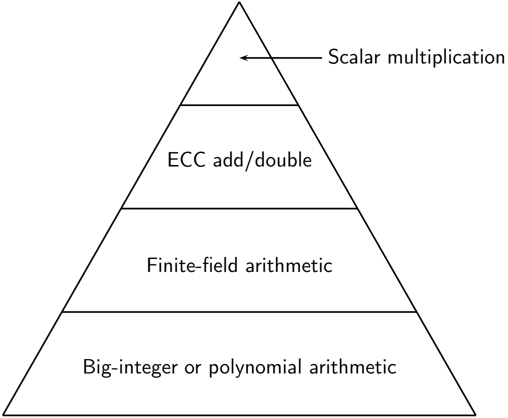
Ed25519

- signature scheme
- public keys and (part of) signatures are points on the twisted Edwards curve

$$-x^2 + y^2 = 1 - 121665/121666x^2y^2$$

  over $\mathbb{F}_{2^{255}-19}$
- by Bernstein, Duif, Lange, Schwabe, and Yang, 2011

The ECC implementation pyramid



(slide credit: Peter Schwabe)

The big multiplier

The big multiplier

- used in all papers about ECC speeds on Intel microarchitectures

The big multiplier

- used in all papers about ECC speeds on Intel microarchitectures
- $64 \times 64 \rightarrow 128$-bit multiplication in one instruction (`mul`)

The big multiplier

- used in all papers about ECC speeds on Intel microarchitectures
- $64 \times 64 \rightarrow 128$-bit multiplication in one instruction (`mul`)
- (This talk focuses on Sandy Bridge/Ivy Bridge)

The radix-2^{51} representation for $\mathbb{F}_{2^{255}-19}$

The radix-2^{51} representation for $\mathbb{F}_{2^{255}-19}$

$$f = \quad f_0 + \quad f_1 2^{51} + \quad f_2 2^{102} + \quad f_3 2^{153} + \quad f_4 2^{204}$$

The radix-2^{51} representation for $\mathbb{F}_{2^{255}-19}$

$$
\begin{aligned}
f &= f_0 + f_1 2^{51} + f_2 2^{102} + f_3 2^{153} + f_4 2^{204} \\
g &= g_0 + g_1 2^{51} + g_2 2^{102} + g_3 2^{153} + g_4 2^{204}
\end{aligned}
$$

The radix-2^{51} representation for $\mathbb{F}_{2^{255}-19}$

$$
\begin{array}{rrrrrr}
f = & f_0 + & f_1 2^{51} + & f_2 2^{102} + & f_3 2^{153} + & f_4 2^{204} \\
g = & g_0 + & g_1 2^{51} + & g_2 2^{102} + & g_3 2^{153} + & g_4 2^{204}
\end{array}
$$

$$
\begin{array}{rrrrrr}
h_0 = & f_0 g_0 + & 19 f_1 g_4 + & 19 f_2 g_3 + & 19 f_3 g_2 + & 19 f_4 g_1 \\
h_1 = & f_0 g_1 + & f_1 g_0 + & 19 f_2 g_4 + & 19 f_3 g_3 + & 19 f_4 g_2 \\
h_2 = & f_0 g_2 + & f_1 g_1 + & f_2 g_0 + & 19 f_3 g_4 + & 19 f_4 g_3 \\
h_3 = & f_0 g_3 + & f_1 g_2 + & f_2 g_1 + & f_3 g_0 + & 19 f_4 g_4 \\
h_4 = & f_0 g_4 + & f_1 g_3 + & f_2 g_2 + & f_3 g_1 + & f_4 g_0
\end{array}
$$

The radix-2^{51} representation for $\mathbb{F}_{2^{255}-19}$

$$
\begin{array}{rrrrrr}
f = & f_0 + & f_1 2^{51} + & f_2 2^{102} + & f_3 2^{153} + & f_4 2^{204} \\
g = & g_0 + & g_1 2^{51} + & g_2 2^{102} + & g_3 2^{153} + & g_4 2^{204}
\end{array}
$$

$$
\begin{array}{rrrrrr}
h_0 = & f_0 g_0 + & 19 f_1 g_4 + & 19 f_2 g_3 + & 19 f_3 g_2 + & 19 f_4 g_1 \\
h_1 = & f_0 g_1 + & f_1 g_0 + & 19 f_2 g_4 + & 19 f_3 g_3 + & 19 f_4 g_2 \\
h_2 = & f_0 g_2 + & f_1 g_1 + & f_2 g_0 + & 19 f_3 g_4 + & 19 f_4 g_3 \\
h_3 = & f_0 g_3 + & f_1 g_2 + & f_2 g_1 + & f_3 g_0 + & 19 f_4 g_4 \\
h_4 = & f_0 g_4 + & f_1 g_3 + & f_2 g_2 + & f_3 g_1 + & f_4 g_0
\end{array}
$$

- 25 multiplication instructions + overhead.

The radix-2^{51} representation for $\mathbb{F}_{2^{255}-19}$

$$
\begin{array}{rrrrrr}
f = & f_0 + & f_1 2^{51} + & f_2 2^{102} + & f_3 2^{153} + & f_4 2^{204} \\
g = & g_0 + & g_1 2^{51} + & g_2 2^{102} + & g_3 2^{153} + & g_4 2^{204}
\end{array}
$$

$$
\begin{array}{rrrrrr}
h_0 = & f_0 g_0 + & 19 f_1 g_4 + & 19 f_2 g_3 + & 19 f_3 g_2 + & 19 f_4 g_1 \\
h_1 = & f_0 g_1 + & f_1 g_0 + & 19 f_2 g_4 + & 19 f_3 g_3 + & 19 f_4 g_2 \\
h_2 = & f_0 g_2 + & f_1 g_1 + & f_2 g_0 + & 19 f_3 g_4 + & 19 f_4 g_3 \\
h_3 = & f_0 g_3 + & f_1 g_2 + & f_2 g_1 + & f_3 g_0 + & 19 f_4 g_4 \\
h_4 = & f_0 g_4 + & f_1 g_3 + & f_2 g_2 + & f_3 g_1 + & f_4 g_0
\end{array}
$$

- 25 multiplication instructions + overhead.
- some *carries* required.

A small multiplier

A small multiplier

- a 2-way vectorized multiplier

A small multiplier

- a 2-way vectorized multiplier
- $32 \times 32 \to 64$-bit multiplications in one instruction (`vpmuludq`)

A small multiplier

- a 2-way vectorized multiplier
- $32 \times 32 \to 64$-bit multiplications in one instruction (`vpmuludq`)
- usage:

$$(a_0 b_0, a_1 b_1) = (a_0, a_1) \times (b_0, b_1)$$

The radix-$2^{25.5}$ representation for $\mathbb{F}_{2^{255}-19}$

The radix-$2^{25.5}$ representation for $\mathbb{F}_{2^{255}-19}$

$$
\begin{aligned}
f &= f_0 + f_1 2^{26} + f_2 2^{51} + f_3 2^{77} + f_4 2^{102} + f_5 2^{128} + f_6 2^{153} + f_7 2^{179} + f_8 2^{204} + f_9 2^{230} \\
g &= g_0 + g_1 2^{26} + g_2 2^{51} + g_3 2^{77} + g_4 2^{102} + g_5 2^{128} + g_6 2^{153} + g_7 2^{179} + g_8 2^{204} + g_9 2^{230}
\end{aligned}
$$

The radix-$2^{25.5}$ representation for $\mathbb{F}_{2^{255}-19}$

$$
\begin{aligned}
f &= f_0 + f_1 2^{26} + f_2 2^{51} + f_3 2^{77} + f_4 2^{102} + f_5 2^{128} + f_6 2^{153} + f_7 2^{179} + f_8 2^{204} + f_9 2^{230} \\
g &= g_0 + g_1 2^{26} + g_2 2^{51} + g_3 2^{77} + g_4 2^{102} + g_5 2^{128} + g_6 2^{153} + g_7 2^{179} + g_8 2^{204} + g_9 2^{230}
\end{aligned}
$$

$$
\begin{aligned}
h_0 &= f_0g_0 + 38f_1g_9 + 19f_2g_8 + 38f_3g_7 + 19f_4g_6 + 38f_5g_5 + 19f_6g_4 + 38f_7g_3 + 19f_8g_2 + 38f_9g_1 \\
h_1 &= f_0g_1 + f_1g_0 + 19f_2g_9 + 19f_3g_8 + 19f_4g_7 + 19f_5g_6 + 19f_6g_5 + 19f_7g_4 + 19f_8g_3 + 19f_9g_2 \\
h_2 &= f_0g_2 + 2f_1g_1 + f_2g_0 + 38f_3g_9 + 19f_4g_8 + 38f_5g_7 + 19f_6g_6 + 38f_7g_5 + 19f_8g_4 + 38f_9g_3 \\
h_3 &= f_0g_3 + f_1g_2 + f_2g_1 + f_3g_0 + 19f_4g_9 + 19f_5g_8 + 19f_6g_7 + 19f_7g_6 + 19f_8g_5 + 19f_9g_4 \\
h_4 &= f_0g_4 + 2f_1g_3 + f_2g_2 + 2f_3g_1 + f_4g_0 + 38f_5g_9 + 19f_6g_8 + 38f_7g_7 + 19f_8g_6 + 38f_9g_5 \\
h_5 &= f_0g_5 + f_1g_4 + f_2g_3 + f_3g_2 + f_4g_1 + f_5g_0 + 19f_6g_9 + 19f_7g_8 + 19f_8g_7 + 19f_9g_6 \\
h_6 &= f_0g_6 + 2f_1g_5 + f_2g_4 + 2f_3g_3 + f_4g_2 + 2f_5g_1 + f_6g_0 + 38f_7g_9 + 19f_8g_8 + 38f_9g_7 \\
h_7 &= f_0g_7 + f_1g_6 + f_2g_5 + f_3g_4 + f_4g_3 + f_5g_2 + f_6g_1 + f_7g_0 + 19f_8g_9 + 19f_9g_8 \\
h_8 &= f_0g_8 + 2f_1g_7 + f_2g_6 + 2f_3g_5 + f_4g_4 + 2f_5g_3 + f_6g_2 + 2f_7g_1 + f_8g_0 + 38f_9g_9 \\
h_9 &= f_0g_9 + f_1g_8 + f_2g_7 + f_3g_6 + f_4g_5 + f_5g_4 + f_6g_3 + f_7g_2 + f_8g_1 + f_9g_0
\end{aligned}
$$

The radix-$2^{25.5}$ representation for $\mathbb{F}_{2^{255}-19}$

$$
\begin{aligned}
f &= f_0 + f_1 2^{26} + f_2 2^{51} + f_3 2^{77} + f_4 2^{102} + f_5 2^{128} + f_6 2^{153} + f_7 2^{179} + f_8 2^{204} + f_9 2^{230} \\
g &= g_0 + g_1 2^{26} + g_2 2^{51} + g_3 2^{77} + g_4 2^{102} + g_5 2^{128} + g_6 2^{153} + g_7 2^{179} + g_8 2^{204} + g_9 2^{230}
\end{aligned}
$$

$$
\begin{aligned}
h_0 &= f_0g_0 + 38f_1g_9 + 19f_2g_8 + 38f_3g_7 + 19f_4g_6 + 38f_5g_5 + 19f_6g_4 + 38f_7g_3 + 19f_8g_2 + 38f_9g_1 \\
h_1 &= f_0g_1 + f_1g_0 + 19f_2g_9 + 19f_3g_8 + 19f_4g_7 + 19f_5g_6 + 19f_6g_5 + 19f_7g_4 + 19f_8g_3 + 19f_9g_2 \\
h_2 &= f_0g_2 + 2f_1g_1 + f_2g_0 + 38f_3g_9 + 19f_4g_8 + 38f_5g_7 + 19f_6g_6 + 38f_7g_5 + 19f_8g_4 + 38f_9g_3 \\
h_3 &= f_0g_3 + f_1g_2 + f_2g_1 + f_3g_0 + 19f_4g_9 + 19f_5g_8 + 19f_6g_7 + 19f_7g_6 + 19f_8g_5 + 19f_9g_4 \\
h_4 &= f_0g_4 + 2f_1g_3 + f_2g_2 + 2f_3g_1 + f_4g_0 + 38f_5g_9 + 19f_6g_8 + 38f_7g_7 + 19f_8g_6 + 38f_9g_5 \\
h_5 &= f_0g_5 + f_1g_4 + f_2g_3 + f_3g_2 + f_4g_1 + f_5g_0 + 19f_6g_9 + 19f_7g_8 + 19f_8g_7 + 19f_9g_6 \\
h_6 &= f_0g_6 + 2f_1g_5 + f_2g_4 + 2f_3g_3 + f_4g_2 + 2f_5g_1 + f_6g_0 + 38f_7g_9 + 19f_8g_8 + 38f_9g_7 \\
h_7 &= f_0g_7 + f_1g_6 + f_2g_5 + f_3g_4 + f_4g_3 + f_5g_2 + f_6g_1 + f_7g_0 + 19f_8g_9 + 19f_9g_8 \\
h_8 &= f_0g_8 + 2f_1g_7 + f_2g_6 + 2f_3g_5 + f_4g_4 + 2f_5g_3 + f_6g_2 + 2f_7g_1 + f_8g_0 + 38f_9g_9 \\
h_9 &= f_0g_9 + f_1g_8 + f_2g_7 + f_3g_6 + f_4g_5 + f_5g_4 + f_6g_3 + f_7g_2 + f_8g_1 + f_9g_0
\end{aligned}
$$

- 100 multiplication instructions + overhead; 50 per multiplication.
- some *carries* required.

The radix-$2^{25.5}$ representation for $\mathbb{F}_{2^{255}-19}$

$$
\begin{aligned}
f &= f_0 + f_1 2^{26} + f_2 2^{51} + f_3 2^{77} + f_4 2^{102} + f_5 2^{128} + f_6 2^{153} + f_7 2^{179} + f_8 2^{204} + f_9 2^{230} \\
g &= g_0 + g_1 2^{26} + g_2 2^{51} + g_3 2^{77} + g_4 2^{102} + g_5 2^{128} + g_6 2^{153} + g_7 2^{179} + g_8 2^{204} + g_9 2^{230}
\end{aligned}
$$

$$
\begin{aligned}
h_0 &= f_0g_0 + 38f_1g_9 + 19f_2g_8 + 38f_3g_7 + 19f_4g_6 + 38f_5g_5 + 19f_6g_4 + 38f_7g_3 + 19f_8g_2 + 38f_9g_1 \\
h_1 &= f_0g_1 + f_1g_0 + 19f_2g_9 + 19f_3g_8 + 19f_4g_7 + 19f_5g_6 + 19f_6g_5 + 19f_7g_4 + 19f_8g_3 + 19f_9g_2 \\
h_2 &= f_0g_2 + 2f_1g_1 + f_2g_0 + 38f_3g_9 + 19f_4g_8 + 38f_5g_7 + 19f_6g_6 + 38f_7g_5 + 19f_8g_4 + 38f_9g_3 \\
h_3 &= f_0g_3 + f_1g_2 + f_2g_1 + f_3g_0 + 19f_4g_9 + 19f_5g_8 + 19f_6g_7 + 19f_7g_6 + 19f_8g_5 + 19f_9g_4 \\
h_4 &= f_0g_4 + 2f_1g_3 + f_2g_2 + 2f_3g_1 + f_4g_0 + 38f_5g_9 + 19f_6g_8 + 38f_7g_7 + 19f_8g_6 + 38f_9g_5 \\
h_5 &= f_0g_5 + f_1g_4 + f_2g_3 + f_3g_2 + f_4g_1 + f_5g_0 + 19f_6g_9 + 19f_7g_8 + 19f_8g_7 + 19f_9g_6 \\
h_6 &= f_0g_6 + 2f_1g_5 + f_2g_4 + 2f_3g_3 + f_4g_2 + 2f_5g_1 + f_6g_0 + 38f_7g_9 + 19f_8g_8 + 38f_9g_7 \\
h_7 &= f_0g_7 + f_1g_6 + f_2g_5 + f_3g_4 + f_4g_3 + f_5g_2 + f_6g_1 + f_7g_0 + 19f_8g_9 + 19f_9g_8 \\
h_8 &= f_0g_8 + 2f_1g_7 + f_2g_6 + 2f_3g_5 + f_4g_4 + 2f_5g_3 + f_6g_2 + 2f_7g_1 + f_8g_0 + 38f_9g_9 \\
h_9 &= f_0g_9 + f_1g_8 + f_2g_7 + f_3g_6 + f_4g_5 + f_5g_4 + f_6g_3 + f_7g_2 + f_8g_1 + f_9g_0
\end{aligned}
$$

- 100 multiplication instructions + overhead; 50 per multiplication.
- some *carries* required.

Sandy2x sets new speed records by using the vectorized multiplier.

Performance results

	SB cycles	IB cycles	reference
X25519 public-key generation	54 346	52 169	**Sandy2x**
	61 828	57 612	[A. Moon]
	194 165	182 876	[Ed25519]
X25519 shared secret computation	156 995	159 128	**Sandy2x**
	194 036	182 708	[Ed25519]
Ed25519 public-key generation	57 164	54 901	**Sandy2x**
	63 712	59 332	[A. Moon]
	64 015	61 099	[Ed25519]
Ed25519 sign	63 526	59 949	**Sandy2x**
	67 692	62 624	[A. Moon]
	72 444	67 284	[Ed25519]
Ed25519 verification	205 741	198 406	**Sandy2x**
	227 628	204 376	[A. Moon]
	222 564	209 060	[Ed25519]

- Andrew Moon "floodyberry", https://github.com/floodyberry/ed25519-donna

Why is vectorization better?

Ports

Why is vectorization better?

Ports

- one each SB/IB core there are 6 ports: Port 0,1,5 are for arithmetic.

Why is vectorization better?

Ports

- one each SB/IB core there are 6 ports: Port 0,1,5 are for arithmetic.
- each instruction is decomposed into *microoperations* (μops)

Why is vectorization better?

Ports

- one each SB/IB core there are 6 ports: Port 0,1,5 are for arithmetic.
- each instruction is decomposed into *microoperations* (μops)
    - `mul`: 2 μops, handled by Port 0 and 1.

Why is vectorization better?

Ports

- one each SB/IB core there are 6 ports: Port 0,1,5 are for arithmetic.
- each instruction is decomposed into *microoperations* (μops)
  - `mul`: 2 μops, handled by Port 0 and 1.
  - `vpmuludq`: 1 μop, handled by Port 0.

Why is vectorization better?

Ports

- one each SB/IB core there are 6 ports: Port 0,1,5 are for arithmetic.
- each instruction is decomposed into *microoperations* (μops)
  - `mul`: 2 μops, handled by Port 0 and 1.
  - `vpmuludq`: 1 μop, handled by Port 0.
  - `vpaddq`: 1 μop, handled by either Port 1 or Port 5.

Why is vectorization better?

Ports

- one each SB/IB core there are 6 ports: Port 0,1,5 are for arithmetic.
- each instruction is decomposed into *microoperations* (μops)
  - `mul`: 2 μops, handled by Port 0 and 1.
  - `vpmuludq`: 1 μop, handled by Port 0.
  - `vpaddq`: 1 μop, handled by either Port 1 or Port 5.
- port utilization gives a lower bound of cycle count

Why is vectorization better?

Using the vectorized multiplier

- 109 `vpmuludq` + 95 `vpaddq`

Why is vectorization better?

Using the vectorized multiplier

- 109 `vpmuludq` + 95 `vpaddq`
- lower bound: 109 cycles (dominated by Port 0)

Why is vectorization better?

Using the vectorized multiplier

- 109 `vpmuludq` + 95 `vpaddq`
- lower bound: 109 cycles (dominated by Port 0)
- actual cycle count 112 cycles (56 cycles per multiplication)

Why is vectorization better?

Using the vectorized multiplier

- 109 `vpmuludq` + 95 `vpaddq`
- lower bound: 109 cycles (dominated by Port 0)
- actual cycle count 112 cycles (56 cycles per multiplication)

Using the serial multiplier

- 25 `mul` + 4 `imul` + 20 `add` + 20 `adc`

Why is vectorization better?

Using the vectorized multiplier

- 109 `vpmuludq` + 95 `vpaddq`
- lower bound: 109 cycles (dominated by Port 0)
- actual cycle count 112 cycles (56 cycles per multiplication)

Using the serial multiplier

- 25 `mul` + 4 `imul` + 20 `add` + 20 `adc`
- lower bound: $(25 \cdot 2 + 4 + 20 + 20 \cdot 2)/3 = 38$

Why is vectorization better?

Using the vectorized multiplier

- 109 `vpmuludq` + 95 `vpaddq`
- lower bound: 109 cycles (dominated by Port 0)
- actual cycle count 112 cycles (56 cycles per multiplication)

Using the serial multiplier

- 25 `mul` + 4 `imul` + 20 `add` + 20 `adc`
- lower bound: $(25 \cdot 2 + 4 + 20 + 20 \cdot 2)/3 = 38$
- actual cycle count is much larger: 52 cycles

Why is vectorization better?

Using the vectorized multiplier

- 109 `vpmuludq` + 95 `vpaddq`
- lower bound: 109 cycles (dominated by Port 0)
- actual cycle count 112 cycles (56 cycles per multiplication)

Using the serial multiplier

- 25 `mul` + 4 `imul` + 20 `add` + 20 `adc`
- lower bound: $(25 \cdot 2 + 4 + 20 + 20 \cdot 2)/3 = 38$
- actual cycle count is much larger: 52 cycles
- `perf-stat` shows that the core fails to distribute the μops equally over the ports

Why is vectorization better?

More reasons

Why is vectorization better?

More reasons

- carries take more cycles when using the serial multiplier

	$\mathbf{M}^{-}$	$\mathbf{M}$
serial	52	68
vectorized	56	69.5

Why is vectorization better?

More reasons

- carries take more cycles when using the serial multiplier

	$\mathbf{M}^{-}$	$\mathbf{M}$
serial	52	68
vectorized	56	69.5

- batched squarings are faster

Why is vectorization better?

More reasons

- carries take more cycles when using the serial multiplier

	$\mathbf{M}^-$	$\mathbf{M}$
serial	52	68
vectorized	56	69.5

- batched squarings are faster
- instruction interleaving hides cost for addition/subtraction

Why is vectorization better?

More reasons

- carries take more cycles when using the serial multiplier

	$\mathbf{M}^{-}$	$\mathbf{M}$
serial	52	68
vectorized	56	69.5

- batched squarings are faster
- instruction interleaving hides cost for addition/subtraction
- constant-time table lookups are faster with vector instructions

Ending Remarks

The main messages of this talk:

Ending Remarks

The main messages of this talk:

- Vectorization should be considered on recent Intel microarchitectures.

Ending Remarks

The main messages of this talk:

- Vectorization should be considered on recent Intel microarchitectures.

Code (for X25519 shared secret computation)

- https://sites.google.com/a/crypto.tw/blueprint/