Implementing supersingular isogeny cryptography

David Jao

Department of Combinatorics & Optimization
Centre for Applied Cryptographic Research

CryptoWorks21 UNIVERSITY OF WATERLOO evolution

November 19, 2018

https://ia.cr/2018/932

18 Seconds to Key Exchange: Limitations of Supersingular Isogeny Diffie-Hellman on Embedded Devices

Philipp Koppermann[1], Eduard Pop[1], Johann Heyszl[1], and Georg Sigl[1,2]

ephemeral key exchange still requires more than 18 seconds on a 32-bit Cortex-M4 and more than 11 minutes on a 16-bit MSP430. Those results show that even with an improvement by a factor of 4, SIDH is in-fact impractical for small embedded devices, regardless of further possible im-

SIDH overview

1. Public parameters: Supersingular elliptic curve E over $\mathbb{F}_{p^2}$.
2. Alice chooses a kernel $A \subset E(\mathbb{F}_{p^2})$ and sends E/A to Bob.
3. Bob chooses a kernel $B \subset E(\mathbb{F}_{p^2})$ and sends E/B to Alice.
4. The shared secret is

$$E/\langle A, B\rangle = (E/A)/\phi_A(B) = (E/B)/\phi_B(A).$$

$$\begin{array}{ccc} E & \xrightarrow{\phi_A} & E/A \\ {\scriptstyle\phi_B}\downarrow & & \downarrow \\ E/B & \longrightarrow & E/\langle A, B\rangle \end{array}$$

The core operation in SIDH is to compute $\phi_A\colon E \to E/A$ given A.

Vélu's formulas for constructing isogenies (1971)

Set $S = (A \setminus \{\infty\})/\pm$ (i.e. "A excluding identity, modulo $\pm$").
Then $\phi_A = (\phi_x, \phi_y)$ where

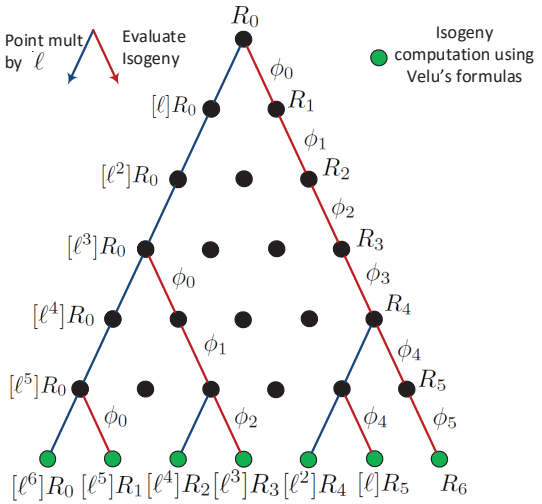
$$\phi_x(x,y) = x + \sum_{Q\in S}\left[\frac{t_Q}{x - x_Q} + \frac{u_Q}{(x-x_Q)^2}\right]$$

$$\phi_y(x,y) = y - \sum_{Q\in S}\left[u_Q\frac{2y}{(x-x_Q)^3} + t_Q\frac{y-y_Q}{(x-x_Q)^2} - \frac{g_Q^x g_Q^y}{(x-x_Q)^2}\right]$$

$$Q = (x_Q, y_Q)$$
$$g_Q^x = 3x_Q^2 + a_4$$
$$g_Q^y = -2y_Q$$
$$t_Q = \begin{cases} g_Q^x & \text{if } Q = -Q \\ 2g_Q^x & \text{if } Q \neq -Q \end{cases}$$
$$u_Q = (g_Q^y)^2$$

SIDH strategies

In SIDH, we use isogenies of degree ℓ^e, where ℓ is a small prime.



Basic constraints:

We can compute point multiplication freely, at will.

However, in order to evaluate an isogeny ϕ_i, we need to compute the point $[\ell^{e-i}]R_i$ first.

Optimization #0: Use smaller parameters

Adj et al., `https://ia.cr/2018/313`:

> *We conclude that using SIDH parameters with $p \approx 2^{448}$ offers CSSI security of at least 128 bits against known classical and quantum attacks, and thus meet the security requirements in NIST's Category 2...*
>
> *SIDH operations are about 4.8 times faster when p_{434} is used instead of p_{751}.*

Optimization #1: SIMD instructions

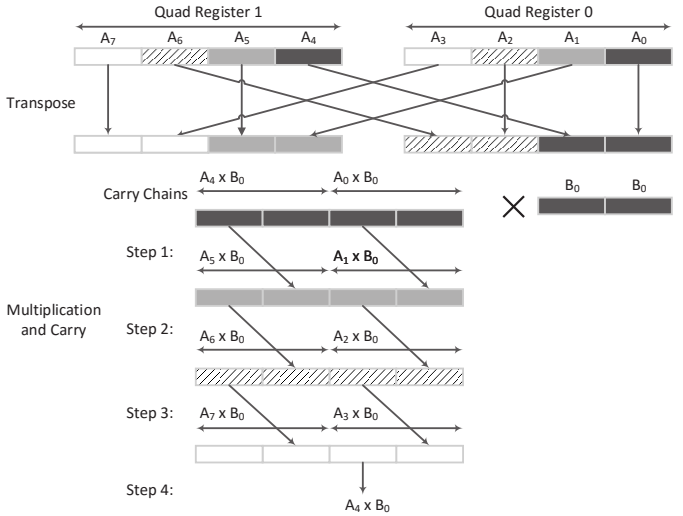
Koziel et al., `https://ia.cr/2016/669`:

- SIMD (Single Instruction Multiple Data) allows multiple operations to be performed in a single cycle.
- The catch: one operand must be constant
  - We can parallelize $a \cdot b$ and $a \cdot c$, but not $a \cdot b$ and $c \cdot d$
- Since our numbers are so large (512 bits and up), we can take advantage of SIMD to parallelize just a single multiplication.

Our approach (on 32-bit ARM):

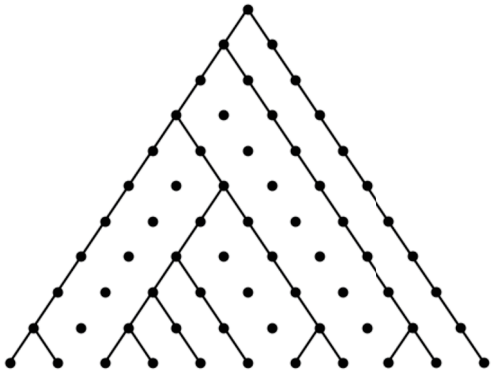
- Use SIMD to multiply 256-bit blocks using parallel operations
- Use Karatsuba multiplication to multiply larger numbers (e.g. 1024-bit numbers) using smaller 256-bit blocks

NEON-SIMD multiplication example: 256×32 bit

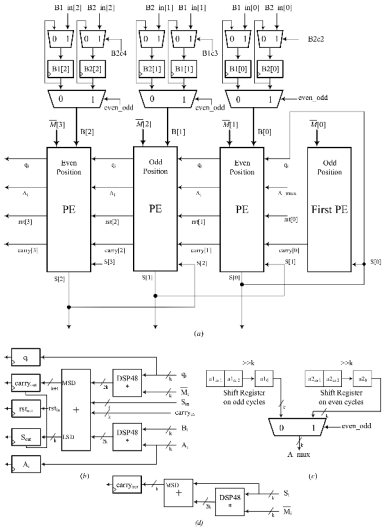


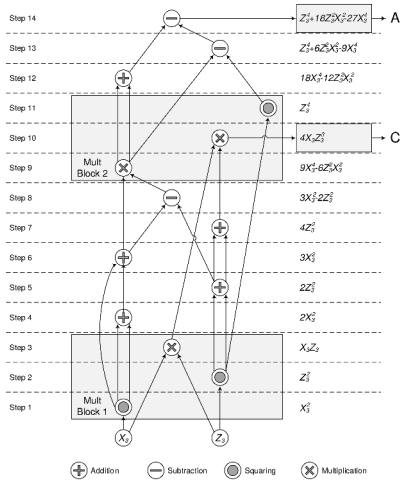
Optimization #2: parallelized isogeny evaluations

Aaron Hutchinson and Koray Karabina, *Constructing canonical strategies for parallel implementation of isogeny based cryptography*, Indocrypt 2018



$n = 239$	$K = 2$	$K = 8$
% speedup over Serial	30.26	55.35





Optimization #3: FPGA implementations

- B. Koziel, R. Azarderakhsh, M. M. Kermani, *Fast hardware architectures for Supersingular Isogeny Diffie-Hellman key exchange on FPGA*, Indocrypt 2016.
- B. Koziel, R. Azarderakhsh, M. M. Kermani, D. Jao, *Post-quantum cryptography on FPGA based on isogenies on elliptic curves*, IEEE TCAS (2017).
- B. Koziel, R. Azarderakhsh, M. M. Kermani, *A high-performance and scalable hardware architecture for isogeny-based cryptography*, IEEE TC **67** (11), 2018.

Parallelism in FPGA implementations

- $\mathbb{F}_{p^2}$ multiplication $\rightarrow$ 3 $\mathbb{F}_p$ multiplications
- Perform isogeny evaluations in parallel (as in Hutchinson and Karabina, Indocrypt 2018)

> to a new curve. In software implementations such as [6], [7], [15], these isogeny evaluations are computed serially. As a contrast, we emphasize that our hardware architecture can compute each of these isogeny evaluations in parallel, as there are no data dependencies between pivot points. Essen-
>
> ...
>
> the total time in SIDH. By parallelizing the isogeny evaluations, we reduced the total time of Bob's first round from 2.9 million cycles to 1.9 million cycles for this example, a speed improvement of 1.53. The only downside to including

Results

Work	Prime (bits)	Area # FFs	# LUTs	# Slices	# DSPs	# BRAMs	Time Freq. (MHz)	Latency ($cc \times 10^6$)	Total time (ms)
Koziel et al. [16]	511	30,031	24,499	10,298	192	27	177	5.967	33.7
Koziel et al. [17]	503	26,659	19,882	8,918	192	40	181.4	3.80	20.9
This Work	503	24,908	18,820	7,491	192	43.5	202.1	3.34	16.5
Improvement over [17]	-	-7.0%	-5.6%	-19%	-	+8.1%	+10.3%	-13.8%	-27%

- 4x more area, 10x less speed, 6x smaller keys (vs. NewHope)
- Finite field exponentiation (needed for constant-time field inversion) remains a bottleneck, as it is difficult to parallelize

Optimization #4: ECC hardware acceleration

J. D. Calhoun, "Optimization of supersingular isogeny cryptography for deeply embedded systems," `https://digitalrepository.unm.edu/ece_etds/420`

- 6.3-7.5x speed improvement using instruction set extensions for finite field arithmetic
- (Further) 6.0-6.1x speed improvement using an existing finite field arithmetic coprocessor design with a 32-bit datapath
- (Further) 2.6-2.9x speed improvement using a slightly modified finite field coprocessor with a 64-bit datapath

Target platforms

Baseline platform: Targhetta et al., "The design space of ultra-low energy asymmetric cryptography." `http://ieeexplore.ieee.org/document/6844461/`

1. "Pete" — 5-stage pipelined RISC (MIPS) processor, 256kB program ROM, 16kB RAM
2. "Pete$_{\mathsf{ISE}}$" — Pete with instruction set extensions for prime fields
3. "PM32" — Pete with "Monte" GF(p) arithmetic accelerator
4. "PM64" — Modified Monte accelerator with 64-bit word size

All (except PM64) were originally designed for ECC (not SIDH).

Results

Work	Platform	Key gen. Alice	Key gen. Bob	Secr. gen. Alice	Secr. gen. Bob
[1]	Cortex-M4	1025	1148	967	1112
[2]	Pete	4259	4814	4012	4197
	Pete$_{\mathsf{ISE}}$	617	679	494	556
	PM32	99	113	85	101
	PM64	33	37	28	34
[3]	x64	27	31	25	29
[4]	Virtex-7	1.61	1.74	1.44	1.59

Table: Clock cycle count [$\times 10^6$] for SIDH on p_{751}

1. Koppermann et al., `https://ia.cr/2018/932`
2. Calhoun, `https://digitalrepository.unm.edu/ece_etds/420`
3. Faz-Hernández et al., `https://ia.cr/2017/1015`
4. Koziel et al., IEEE TC **67** (11), 2018.

Size comparison

			Bytes	FF	LUT	DSP	BRAM
[1]	NewHope	Artix-7	2178	4452	5142	2	4
[2]	Pete_{ISE}	Zynq 7Z020	378	2944	4658	1	33.5
	PM32			3426	5403	5	44.5
	PM64			3700	6074	17	47.5
[3]	SIDH	Virtex-7	378	24908	18820	192	43.5

FF — flip flops

LUT — lookup tables

DSP — digital signal processing slices

BRAM — Block RAM

1. Oder and Güneysu, LatinCrypt 2017
2. Calhoun, `https://digitalrepository.unm.edu/ece_etds/420`
3. Koziel et al., IEEE TC **67** (11), 2018.

Note: The design in [2] is software-configurable for any field size.

Conclusions

- SIDH with NewHope-like hardware acceleration resources is clock-for-clock comparable to SIDH on x64 in speed.
- SIDH on IoT likely requires hardware acceleration.
- Taking into account cost of communication, SIDH may be of interest to IoT implementors.
- Future work: Authenticated key exchange, signatures, CSIDH.