Implementing supersingular isogeny cryptography

David Jao
Department of Combinatorics & Optimization
Centre for Applied Cryptographic Research

November 19, 2018
18 Seconds to Key Exchange: Limitations of Supersingular Isogeny Diffie-Hellman on Embedded Devices

Philipp Koppermann¹, Eduard Pop¹, Johann Heyszl¹, and Georg Sigl¹,²

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-
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

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

The core operation in SIDH is to compute $\phi_A : E \rightarrow E/A$ 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

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

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

$Q = (x_Q, y_Q)$

$g_x^x = 3x_Q^2 + a_4$

$g_y^y = -2y_Q$

$t_Q = \begin{cases} 
\frac{g^x_Q}{2g^x_Q} & \text{if } Q = -Q \\
2g^x_Q & \text{if } Q \neq -Q 
\end{cases}$

$u_Q = (g_y^Q)^2$
In SIDH, we use isogenies of degree $\ell^e$, where $\ell$ is a small prime.

Basic constraints:

We can compute point multiplication freely, at will.

However, in order to evaluate an isogeny $\phi_i$, we need to compute the point $[\ell^{e-i}]R_i$ first.
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 $p_{751}$.***
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

Transposing

Multiplication and Carry Chains

Step 1:

Step 2:

Step 3:

Step 4:
Optimization #2: parallelized isogeny evaluations

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

<table>
<thead>
<tr>
<th>$n = 239$</th>
<th>$K = 2$</th>
<th>$K = 8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>% speedup over Serial</td>
<td>30.26</td>
<td>55.35</td>
</tr>
</tbody>
</table>
Optimization #3: FPGA implementations


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. Essentially, 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

- 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.

<table>
<thead>
<tr>
<th>Work</th>
<th>Prime (bits)</th>
<th># FFs</th>
<th># LUTs</th>
<th># Slices</th>
<th># DSPs</th>
<th># BRAMs</th>
<th>Freq (MHz)</th>
<th>Latency (cc × 10^3)</th>
<th>Total time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Koziel et al. [16]</td>
<td>511</td>
<td>30,031</td>
<td>24,499</td>
<td>10,298</td>
<td>192</td>
<td>27</td>
<td>177</td>
<td>5.967</td>
<td>33.7</td>
</tr>
<tr>
<td>Koziel et al. [17]</td>
<td>503</td>
<td>26,659</td>
<td>19,882</td>
<td>8,918</td>
<td>192</td>
<td>40</td>
<td>181.4</td>
<td>3.80</td>
<td>20.9</td>
</tr>
<tr>
<td>This Work</td>
<td>503</td>
<td>24,908</td>
<td>18,820</td>
<td>7,491</td>
<td>192</td>
<td>43.5</td>
<td>202.1</td>
<td>3.34</td>
<td>16.5</td>
</tr>
<tr>
<td>Improvement over [17]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+8.1%</td>
<td>+10.3%</td>
<td>-13.8%</td>
<td>-27%</td>
</tr>
</tbody>
</table>
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
Baseline platform: Targhetta et al., “The design space of ultra-low energy asymmetric cryptography.”

1. “Pete” — 5-stage pipelined RISC (MIPS) processor, 256kB program ROM, 16kB RAM
2. “Pete_{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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>Cortex-M4</td>
<td>1025</td>
<td>1148</td>
<td>967</td>
<td>1112</td>
</tr>
<tr>
<td>[2]</td>
<td>Pete</td>
<td>4259</td>
<td>4814</td>
<td>4012</td>
<td>4197</td>
</tr>
<tr>
<td></td>
<td>PeteISE</td>
<td>617</td>
<td>679</td>
<td>494</td>
<td>556</td>
</tr>
<tr>
<td></td>
<td>PM32</td>
<td>99</td>
<td>113</td>
<td>85</td>
<td>101</td>
</tr>
<tr>
<td></td>
<td>PM64</td>
<td>33</td>
<td>37</td>
<td>28</td>
<td>34</td>
</tr>
<tr>
<td>[3]</td>
<td>x64</td>
<td>27</td>
<td>31</td>
<td>25</td>
<td>29</td>
</tr>
<tr>
<td>[4]</td>
<td>Virtex-7</td>
<td>1.61</td>
<td>1.74</td>
<td>1.44</td>
<td>1.59</td>
</tr>
</tbody>
</table>

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

2. Calhoun, https://digitalrepository.unm.edu/ece_etds/420
## Size comparison

<table>
<thead>
<tr>
<th></th>
<th>Bytes</th>
<th>FF</th>
<th>LUT</th>
<th>DSP</th>
<th>BRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>NewHope</td>
<td>Artix-7</td>
<td>2178</td>
<td>4452</td>
<td>5142</td>
</tr>
<tr>
<td></td>
<td>PeteISE</td>
<td>Zynq 7Z020</td>
<td>378</td>
<td>3426</td>
<td>5403</td>
</tr>
<tr>
<td>[2]</td>
<td>PM32</td>
<td>Zynq 7Z020</td>
<td>378</td>
<td>3426</td>
<td>5403</td>
</tr>
<tr>
<td></td>
<td>PM64</td>
<td>Zynq 7Z020</td>
<td>378</td>
<td>3426</td>
<td>5403</td>
</tr>
<tr>
<td>[3]</td>
<td>SIDH</td>
<td>Virtex-7</td>
<td>378</td>
<td>24908</td>
<td>18820</td>
</tr>
</tbody>
</table>

**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](https://digitalrepository.unm.edu/ece_etds/420)  

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.