Trouble at the CSIDH: Protecting CSIDH with Dummy-Operations against Fault Injection Attacks

FDTC 2020 - Fault Diagnosis and Tolerance in Cryptography workshop

Fabio Campos\textsuperscript{1}, Matthias J. Kannwischer\textsuperscript{2}, Michael Meyer\textsuperscript{1,3}, Hiroshi Onuki\textsuperscript{4}, Marc Stöttinger\textsuperscript{5}

\textsuperscript{1}RheinMain University of Applied Sciences, Germany
\textsuperscript{2}Radboud University, The Netherlands
\textsuperscript{3}University of Würzburg, Germany
\textsuperscript{4}University of Tokyo, Japan
\textsuperscript{5}Continental AG, Germany
somewhere in the crypto-heaven...

LET US FINALLY DESTROY SOMETHING!

THAT’S BORING!!! LET’S ATTACK SOME FANCY STUFF LIKE ... DUMMY-BASED CONSTANT-TIME CSHIDH!

OK, OK! WHAT ABOUT ... ONLY DEFINING & SIMULATING SOME ATTACKS?

WHAAAAT?!? DO YOU WANT TO ATTACK SOME BLOCKCHAIN AGAIN?

PLEASE DO NOT! WE’RE HAVING ENOUGH TROUBLE IN THE WORLD RIGHT NOW!

SIMULATION, MY A.... AS SOON AS YOU’RE NOT PAYING ATTENTION, I’M GONNA HIT IT SOOO HARD!

WELL ... IF I KNOW SOME WEAK POINTS, I MIGHT BE ABLE TO PROTECT IT BETTER.

Comic art: Lua Campos
Preliminaries
Isogeny

- a map \( (\phi : E \rightarrow E') \) between two elliptic curves
- a group morphism \( \phi(P + Q) = \phi(P) + \phi(Q) \)
- an algebraic map
- entirely determined by its kernel (i.e., by a single point)
let $p = 4\ell_1 \cdots \ell_n - 1$ be prime, where $\ell_1, \ldots, \ell_n$ are small distinct odd primes

let $E_A : y^2 = x^3 + Ax^2 + x$ be a supersingular elliptic curve in Montgomery form over $\mathbb{F}_p$

points of orders $\ell_i$ for all $1 \leq i \leq n$, which can be used as input to compute an isogeny of degree $\ell_i$,

private key $= (e_1, \ldots, e_n)$, where $|e_i|$ = number of isogenies of degree $\ell_i$

sign of $e_i$ determines if order-$\ell_i$ point on the curve or its twist

$e_i$‘s sampled from small interval $[-m, m]$
Union of cycles

- **Nodes:**
  Supersingular curves over $\mathbb{F}_{419}$.

- **Undirected edges:**
  3-, 5-, and 7-isogenies.

Graph mostly "stolen" from Chloe Martindale
https://www.martindale.info/talks/QIT-Bristol.pdf
Timing attacks

- number of isogenies depends on private key
- effort for multiplication depends on sign distribution of private key
Real vs dummy isogenies - different computation blocks

Figure 1: Real isogeny

Figure 2: Dummy isogeny
What about dummy-free constant-time?

Timings for constant-time CSIDH implementations@x86

<table>
<thead>
<tr>
<th>Group action evaluation</th>
<th>Mcycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>not constant-time</td>
<td>103</td>
</tr>
<tr>
<td>Meyer, Campos, Reith (MCR)</td>
<td>298</td>
</tr>
<tr>
<td>Onuki, Aikawa, Yamazaki, Takagi (OYAT)</td>
<td>230</td>
</tr>
<tr>
<td>dummy-free</td>
<td>432</td>
</tr>
</tbody>
</table>

1. optimized versions from https://ia.cr/2020/417
2. almost unoptimized, see https://ia.cr/2018/782
3. see https://ia.cr/2018/1198
4. see https://ia.cr/2019/353
Attacker models & simulation
Setup

- 3 attacker models with **increasing capabilities**
- attacker performs **single** fault injection per run
- **repeatedly evaluation** using same secret key
- injects during computation of **group action**

\[ \varphi \]

\[ E_0 \xrightarrow{\varphi_A} E_A \]

\[ E_B \xrightarrow{\varphi_B} E_{AB} \xrightarrow{\tilde{\varphi}_A} E'_{AB} \]

attacked
1: Shotgun at the CSIDH

- **weakest** adversary model
- **no control** over location of fault injection
- ratio failures $\hat{=} \text{ratio "real" vs. "dummy"}$

---

Photo: Rita Claveau on https://www.pinterest.it/
1: Shotgun at the CSIDH

- **weakest** adversary model
- **no control** over location of fault injection
- ratio failures $\cong$ ratio “real” vs. “dummy”

**Setup**

- isogeny computations effort about 42%
- **cost-simulation** output transcript of all operations secret
- 100 randomly CSIDH512 keys and 500,000 fault injections

---

Photo: Rita Claveau on https://www.pinterest.it/
1: Shotgun at the CSIDH

- **weakest** adversary model
- **no control** over location of fault injection
- ratio failures $\hat{=} \text{ratio} \ \text{"real" vs. \ "dummy"}$

**Setup**

- isogeny computations effort about **42%**
- **cost-simulation** output transcript of all operations secret
- 100 randomly CSIDH512 keys and 500,000 fault injections

**Impact**

- **correlation not strong** enough
- key space reduction from $2^{256}$ to $\approx 2^{249}$

Photo: Rita Claveau on https://www.pinterest.it/
2: Aiming at isogenies at index $i$

- slightly more powerful
- target $i$-th isogeny computation

---

Photo: Piotr Wilk on https://unsplash.com/

5see https://ia.cr/2020/1006 for randomize order
2: Aiming at isogenies at index $i$

- slightly more powerful
- target $i$-th isogeny computation

Setup

- deterministic computation of $e_i$ : real then dummy\(^5\)
- out of order due to point rejections, point rejection probability $= 1/\ell_i$
- sequence of first 23 isogenies is almost deterministic

\(^5\)see https://ia.cr/2020/1006 for randomize order
2: Aiming at isogenies at index $i$

- slightly more powerful
- target $i$-th isogeny computation

**Setup**

- **deterministic** computation of $e_i$: real then dummy\(^5\)
- **out of order** due to point rejections, point rejection probability $= 1/\ell_i$
- sequence of first 23 isogenies is **almost deterministic**

**Impact**

- best case: key space reduction from $2^{256}$ to $2^{177}$

---

Photo: Piotr Wilk on https://unsplash.com/
\(^5\)see https://ia.cr/2020/1006 for randomize order
3: Aiming at isogeny computations and tracing the order

- **most powerful** attacker model
- able to **trace the order** (SPA) of the attacked isogeny

---

Photo: Alan Belmer on https://freeimages.com/

^6see https://ia.cr/2018/383
3: Aiming at isogeny computations and tracing the order

- **most powerful** attacker model
- able to **trace the order** (SPA) of the attacked isogeny

Setup

- binary search for each individual degree to identify first dummy isogeny

---

Photo: Alan Belmer on https://freeimages.com/

6 see https://ia.cr/2018/383
3: Aiming at isogeny computations and tracing the order

- **most powerful** attacker model
- able to **trace the order** (SPA) of the attacked isogeny

**Setup**

- binary search for each individual degree to identify first dummy isogeny

**Impact**

- on MCR: full key recovery requires 178 injections
- on OAYT: 178 injections $\leadsto$ space reduction to $2^{67.04}$ (average case); further reducible to $\approx 2^{34.5}$ (meet-in-the-middle$^6$)

Photo: Alan Belmer on [https://freeimages.com/](https://freeimages.com/)

Practical experiments
Setup

- **plain C implementation**
- **reduced key space** from $11^{74}$ to $3^2$, secret keys $\in \{-1, 0, 1\}$
- isogenies with **smallest degrees** (3 and 5)
- **ChipWhisperer**-Lite ARM
Setup

- **plain C implementation**
- **reduced key space** from $11^{74}$ to $3^2$, secret keys $\in \{-1, 0, 1\}$
- isogenies with **smallest degrees** (3 and 5)
- **ChipWhisperer-Lite ARM**

\[ E_0 \xrightarrow{\varphi_A} E_A \]
\[ E_B \xrightarrow{\varphi_B} E AB \xrightarrow{\varphi_A} E_{AB} \]
\[ \varphi_B \]

\[ \sim \]
\[ E_{AB} \xrightarrow{\sim \varphi_B} \]
\[ E_{AB} \xrightarrow{\sim \varphi_A} E_{AB} \]

precomputed

attacked
## Accuracy of the results

### Randomized attacks

<table>
<thead>
<tr>
<th>type</th>
<th>key</th>
<th># of trials</th>
<th>faulty shared secret</th>
</tr>
</thead>
<tbody>
<tr>
<td>attack 1</td>
<td>{0,0}</td>
<td>5000</td>
<td>19.8%</td>
</tr>
<tr>
<td></td>
<td>{0,1}</td>
<td>5000</td>
<td>27.3%</td>
</tr>
<tr>
<td></td>
<td>{-1,1}</td>
<td>5000</td>
<td>32.8%</td>
</tr>
<tr>
<td>attack 2</td>
<td>{0,1}</td>
<td>5000</td>
<td>2.1%</td>
</tr>
<tr>
<td></td>
<td>{-1,1}</td>
<td>5000</td>
<td>16.4%</td>
</tr>
</tbody>
</table>

- Targeting critical spots
  - empirically determined with manageable effort
  - accuracy of over 95% (in attack 2 & 3) with single injection
### Accuracy of the results

#### Randomized attacks

<table>
<thead>
<tr>
<th>Type</th>
<th>Key</th>
<th># of trials</th>
<th>Faulty shared secret</th>
</tr>
</thead>
<tbody>
<tr>
<td>attack 1</td>
<td>{0,0}</td>
<td>5000</td>
<td>19.8%</td>
</tr>
<tr>
<td></td>
<td>{0,1}</td>
<td>5000</td>
<td>27.3%</td>
</tr>
<tr>
<td></td>
<td>{-1,1}</td>
<td>5000</td>
<td>32.8%</td>
</tr>
<tr>
<td>attack 2</td>
<td>{0,1}</td>
<td>5000</td>
<td>2.1%</td>
</tr>
<tr>
<td></td>
<td>{-1,1}</td>
<td>5000</td>
<td>16.4%</td>
</tr>
</tbody>
</table>

### Targeting critical spots

- empirically determined with **manageable effort**
- accuracy of **over 95%** (in attack 2 & 3) with single injection
Countermeasures & performance
Basic idea

- detect injections by changing arithmetic operations
Countermeasures

Basic idea

• detect injections by changing arithmetic operations

Objectives

• fault injection $\rightarrow$ output an error
• countermeasures for dummy & real case keeping constant-time
• small overhead
Countermeasures

Basic idea

• detect injections by changing arithmetic operations

Objectives

• fault injection $\leadsto$ output an error
• countermeasures for dummy & real case keeping constant-time
• small overhead

Conditional functions

• $\text{cadd}(x, y, b)$: returns $x + by$
• $\text{cadd2}(x, y, b)$: returns $bx + by$
• $\text{csub}(x, y, b)$: returns $x - by$
• $\text{cverify}(x, y, b)$, checks $x = y$, only outputs result if $b = 1$
Algorithm 1: Toy example

1. Set $\pi_+ \leftarrow 1$, $\pi_- \leftarrow 1$
2. for $i \in \{1, \ldots, (\ell - 1)/2\}$ do
   3. $t_0 \leftarrow \text{cadd}(X_i, Z_i, b)$ // $t_0 = X_i$ \ $t_0 = X_i + Z_i$
   4. $t_1 \leftarrow \text{csub}(X_i, Z_i, b)$ // $t_1 = X_i$ \ $t_1 = X_i - Z_i$
   5. $\pi_+ \leftarrow \pi_+ \cdot t_0$ // $\pi_+ = \prod X_i$ \ $\pi_+ = \prod (X_i + Z_i)$
   6. $\pi_- \leftarrow \pi_- \cdot t_1$ // $\pi_- = \prod X_i$ \ $\pi_- = \prod (X_i - Z_i)$
3. $\text{error} \leftarrow \text{cverify}(\pi_+, \pi_-, \neg b)$ // if $b = 0$: verify that $\pi_+ = \pi_-$

where $b = 0$ if dummy, and $b = 1$ for the real case
Conclusions

- relatively **small overhead** 5% (STM32F407) to 7% (STM32F303)

---

7 core on ChipWhisperer-Lite
Conclusions

• relatively **small overhead** 5% (STM32F407) to 7% (STM32F303\textsuperscript{7})

• some countermeasures applicable to dummy-free variants

\textsuperscript{7}core on ChipWhisperer-Lite
Conclusions

- relatively **small overhead** 5% (STM32F407) to 7% (STM32F303\(^7\))

- some countermeasures **applicable to dummy-free** variants

- CSIDH painfully slow $\leadsto$ experiments with **full scheme** infeasible

---

\(^7\)core on ChipWhisperer-Lite
Conclusions

- relatively **small overhead** 5% (STM32F407) to 7% (STM32F303)$^7$

- some countermeasures **applicable to dummy-free variants**

- CSIDH painfully slow $\leadsto$ experiments with **full scheme infeasible**

- ChipWhisperer: perfectly **adequate**

---

$^7$core on ChipWhisperer-Lite
Thank you for your attention!

OH ALICE ... YOU'RE THE ONE FOR ME.

BUT BOB ... IN A QUANTUM WORLD HOW CAN WE BE SURE? IT'S LOOKING SPOOKIER THAN EVER!

Alice by engin akyurt, Bob by Philipp Lansing on https://unsplash.com/