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

FDTC 2020 - Fault Diagnosis and Tolerance in Cryptography workshop

Fabio Campos[1], Matthias J. Kannwischer[2], Michael Meyer[1,3], Hiroshi Onuki[4], Marc Stöttinger[5]

[1]RheinMain University of Applied Sciences, Germany
[2]Radboud University, The Netherlands
[3]University of Würzburg, Germany
[4]University of Tokyo, Japan
[5]Continental AG, Germany

somewhere in the crypto-heaven ...

Comic art: Lua Campos

Preliminaries

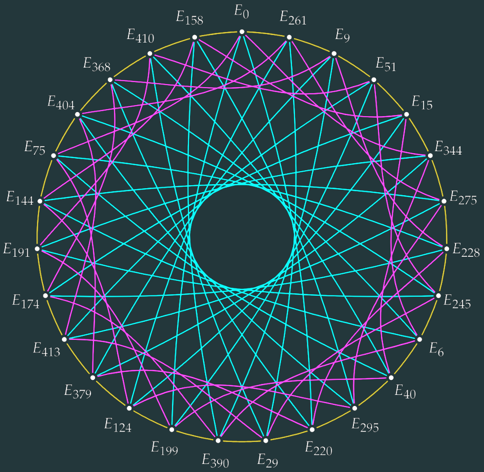
Isogeny

- a map ($\phi : E \to E'$) 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)

CSIDH : algorithmic description

- 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 ℓ_i for all $1 \leq i \leq n$, which can be used as input to compute an isogeny of degree ℓ_i,
- private key $= (e_1, \ldots, e_n)$, where $|e_i| =$ number of isogenies of degree ℓ_i
- sign of e_i determines if order-ℓ_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

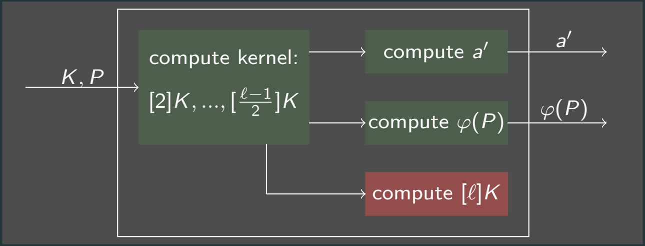


Figure 1: Real isogeny

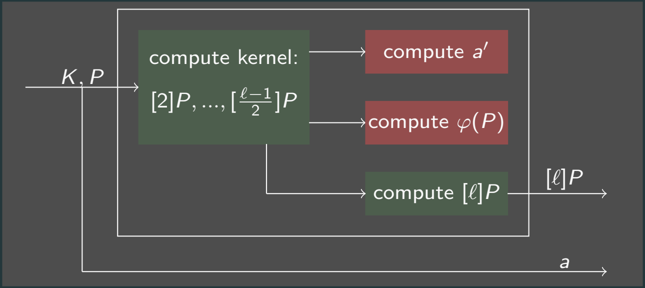


Figure 2: Dummy isogeny

required computation

dummy computation

What about dummy-free constant-time?

Timings for constant-time CSIDH implementations@x86[1]

Group action evaluation	Mcycles
not constant-time[2]	103
Meyer, Campos, Reith (MCR)[3]	298
Onuki, Aikawa, Yamazaki, Takagi (OYAT)[4]	230
dummy-free[1]	432

[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

- 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**

$$
\begin{tikzcd}
E_0 \arrow[r, "\varphi_A"] \arrow[d, "\varphi_B"'] & E_A \arrow[dl, dotted, shift left] \arrow[drr, red, "\tilde{\varphi}'_B"] & & \\
E_B \arrow[r, "\tilde{\varphi}_A"'] \arrow[ur, dotted, shift left] & E_{AB} & \overset{?}{=} & E'_{AB}
\end{tikzcd}
$$

$\xrightarrow{\text{attacked}}$

1: Shotgun at the CSIDH

- **weakest** adversary model
- **no control** over location of fault injection
- ratio failures $\widehat{=}$ **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 $\widehat{=}$ **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 $\widehat{=}$ **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

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/`

[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**

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

[5] see `https://ia.cr/2020/1006` for randomize order

- **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

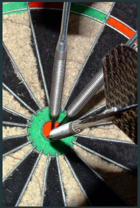
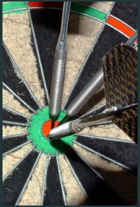


Photo: Alan Belmer on `https://freeimages.com/`

[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

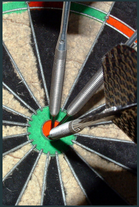
Setup

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

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 $\rightsquigarrow$ 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/`

[6] see `https://ia.cr/2018/383`

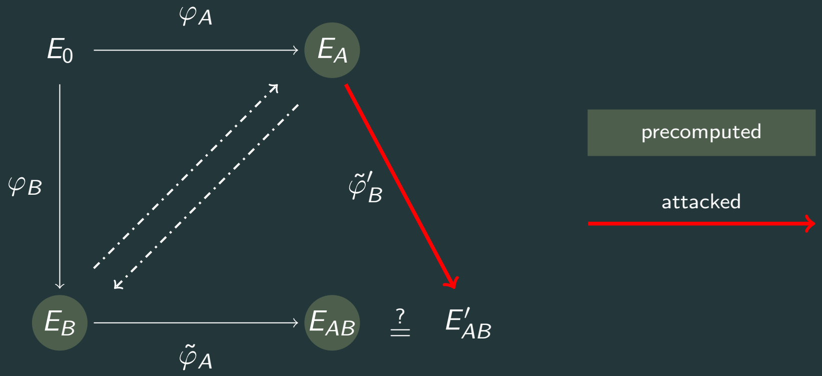
Practical experiments

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



Accuracy of the results

Randomized attacks

type	key	# of trials	faulty shared secret
attack 1	{0,0}	5000	19.8%
	{0,1}	5000	27.3%
	{-1,1}	5000	32.8%
attack 2	{0,1}	5000	2.1%
	{-1,1}	5000	16.4%

Accuracy of the results

Randomized attacks

type	key	# of trials	faulty shared secret
attack 1	{0,0}	5000	19.8%
	{0,1}	5000	27.3%
	{-1,1}	5000	32.8%
attack 2	{0,1}	5000	2.1%
	{-1,1}	5000	16.4%

Targeting critical spots

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

Countermeasures & performance

Countermeasures 1/2

Basic idea

- detect injections by changing arithmetic operations

Countermeasures 1/2

Basic idea

- detect injections by changing arithmetic operations

Objectives

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

Basic idea

- detect injections by changing arithmetic operations

Objectives

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

Conditional functions

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

Countermeasures 2/2

Algorithm 1: Toy example

Set $\pi_+ \leftarrow 1$, $\pi_- \leftarrow 1$

for $i \in \{1, \ldots, (\ell - 1)/2\}$ **do**

  $t_0 \leftarrow \texttt{cadd}(X_i, Z_i, b)$ // $t_0 = X_i \ \mid \ t_0 = X_i + Z_i$

  $t_1 \leftarrow \texttt{csub}(X_i, Z_i, b)$ // $t_1 = X_i \ \mid \ t_1 = X_i - Z_i$

  $\pi_+ \leftarrow \pi_+ \cdot t_0$ // $\pi_+ = \prod X_i \ \mid \ \pi_+ = \prod (X_i + Z_i)$

  $\pi_- \leftarrow \pi_- \cdot t_1$ // $\pi_- = \prod X_i \ \mid \ \pi_- = \prod (X_i - Z_i)$

error $\leftarrow \texttt{cverify}(\pi_+, \pi_-, \neg b)$ // 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])

[7] core on ChipWhisperer-Lite

Conclusions

- relatively **small overhead** 5% (STM32F407) to 7% (STM32F303[7])
- some countermeasures **applicable to dummy-free** variants

[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 $\rightsquigarrow$ 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 $\rightsquigarrow$ experiments with **full scheme infeasible**
- ChipWhisperer: perfectly **adequate**

[7] core on ChipWhisperer-Lite

Paper: https://ia.cr/2020/1005

Code: https://github.com/csidhfi/csidhfi

Thank you for your attention!

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