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

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

Outline

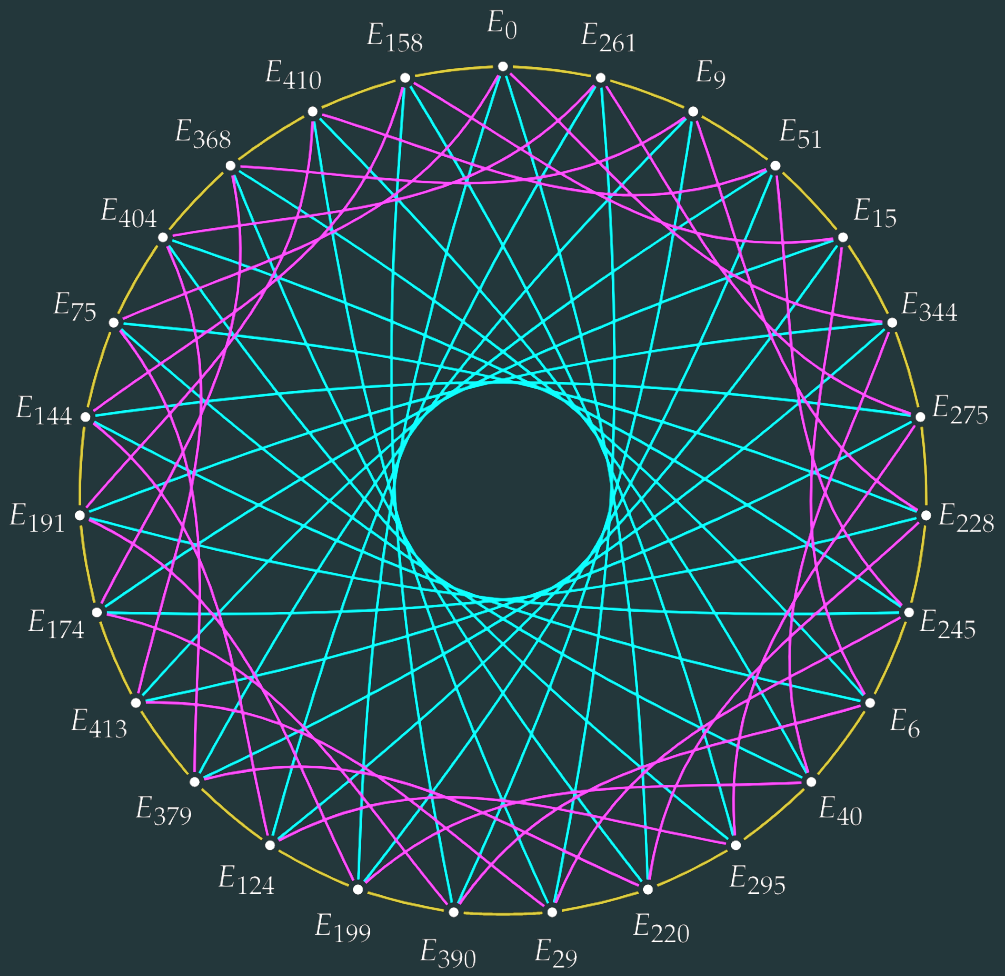
- Preliminaries
- Attacker models & simulation
- Practical experiments
- Countermeasures & performance

Preliminaries

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`

Notions of "constant-time"

- running time, branching, etc. do not depend on secrets, but may vary because of randomness
- execution time is constant

Timing attacks

- number of isogenies **depends on private key**
- effort for multiplication **depends on sign distribution** of private key

Meyer, Campos, Reith (MCR)[1]

- maximal amount of isogenies using dummy isogenies
- exponents in $[0, 2m]$ (instead of $[-m, m]$) avoid timing attacks

Onuki, Aikawa, Yamazaki, Takagi (OYAT)[2]

- two points to evaluate the action (avoid timing attacks)
- keeping exponent range $[-m, m]$
- compared to MCR: speed-up of 27.35%

[1] see `https://ia.cr/2018/1198`
[2] see `https://ia.cr/2019/353`

Real vs dummy isogenies - different computation blocks

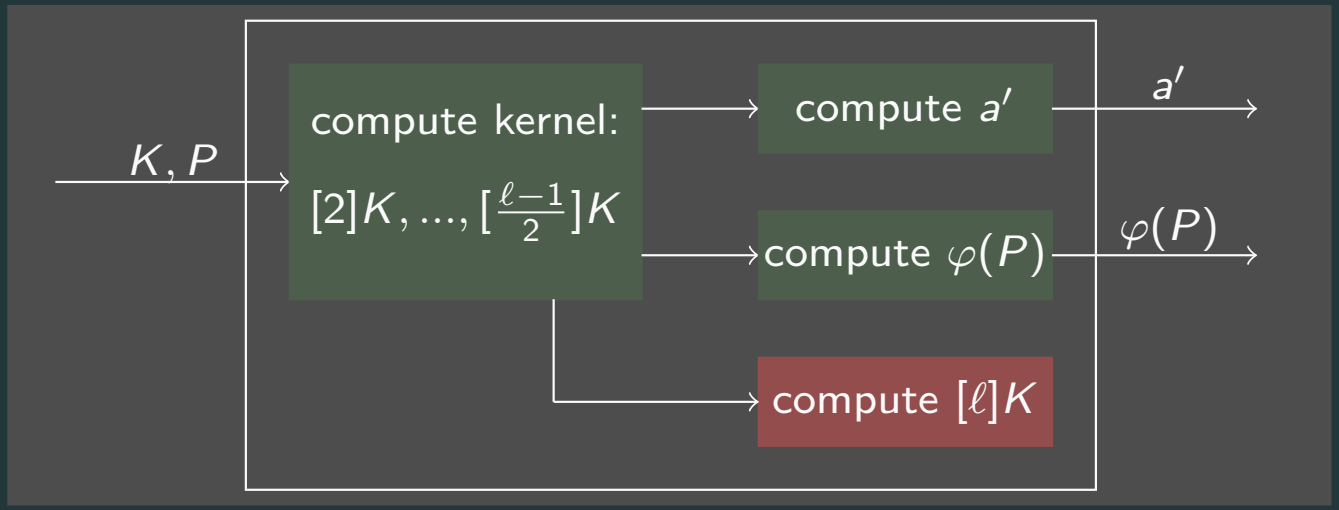


Figure 1: Real isogeny

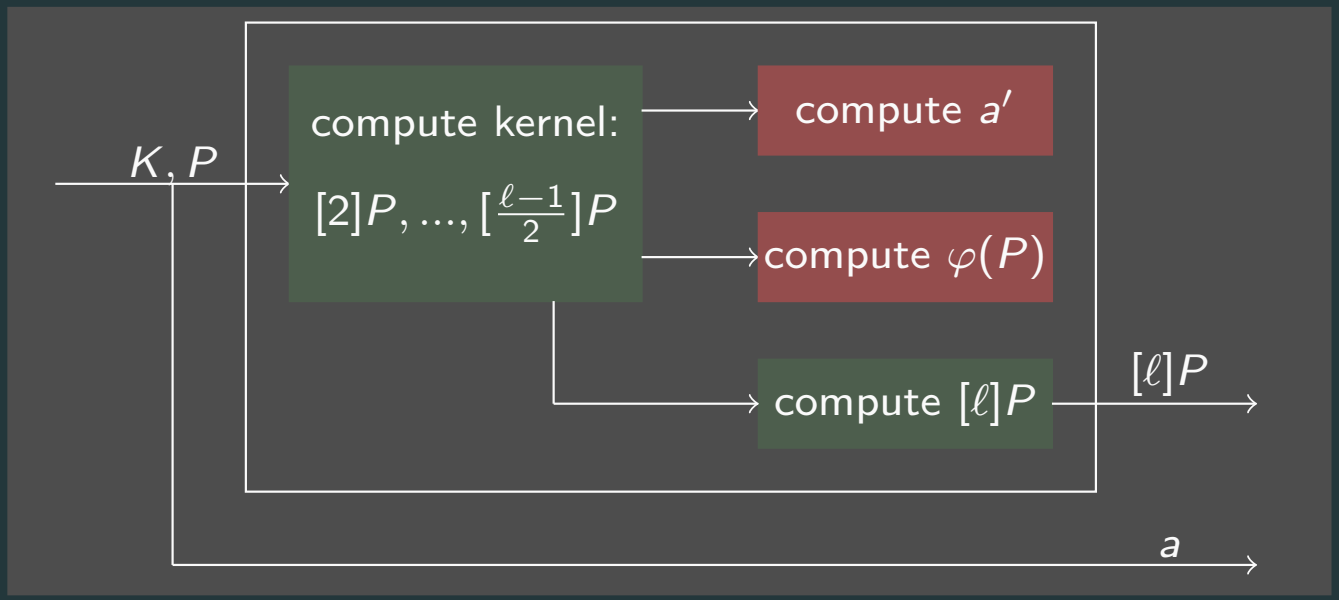


Figure 2: Dummy isogeny

required computation

dummy computation

What about dummy-free constant-time?

Timings for constant-time CSIDH implementations@x86

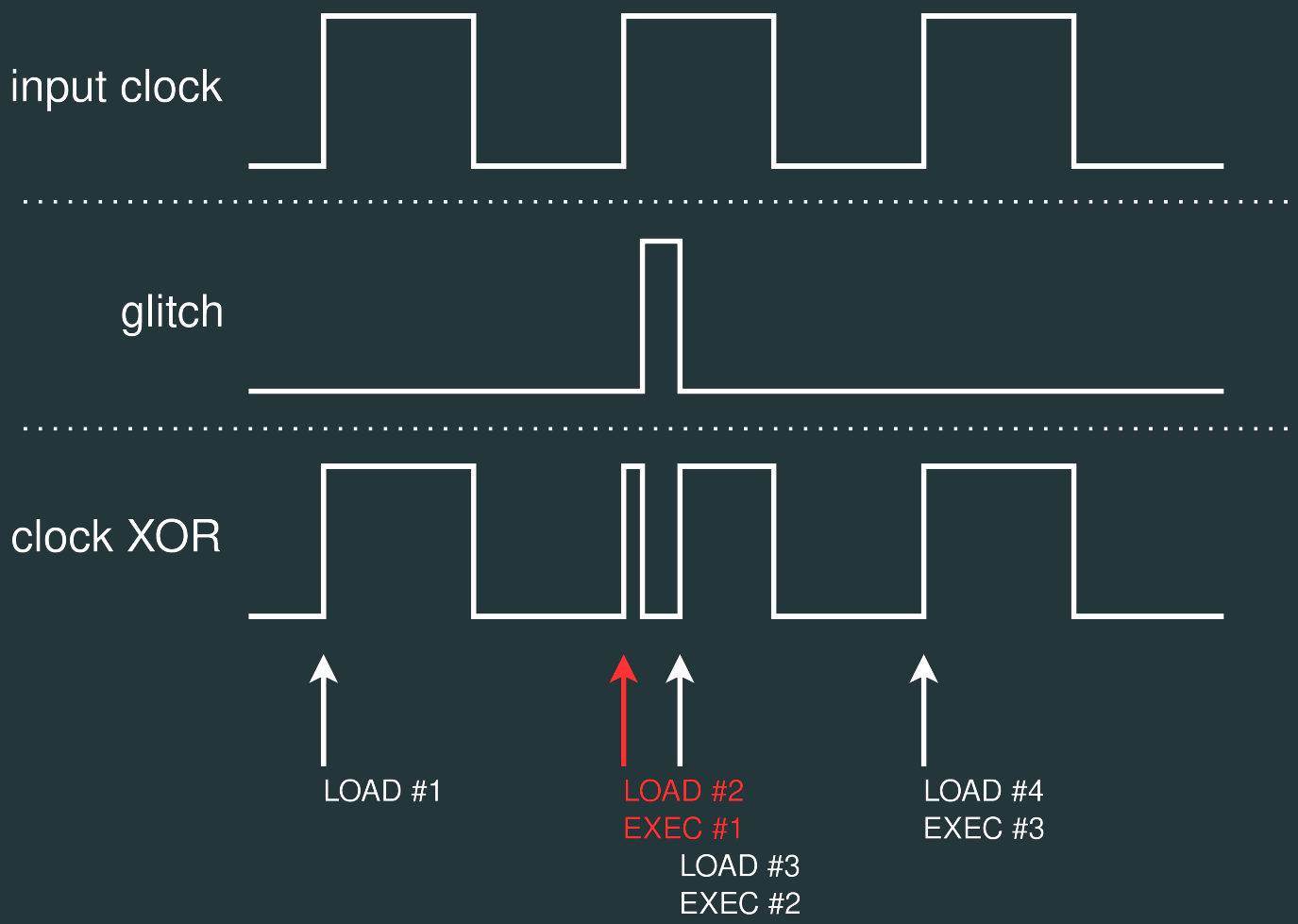
Group action evaluation	Mcycles
not constant-time[3]	103
MCR[4]	298
OYAT[4]	230
dummy-free[4]	432

[3] almost unoptimized, see `https://ia.cr/2018/782`

[4] see `https://ia.cr/2020/417`

Clock glitching

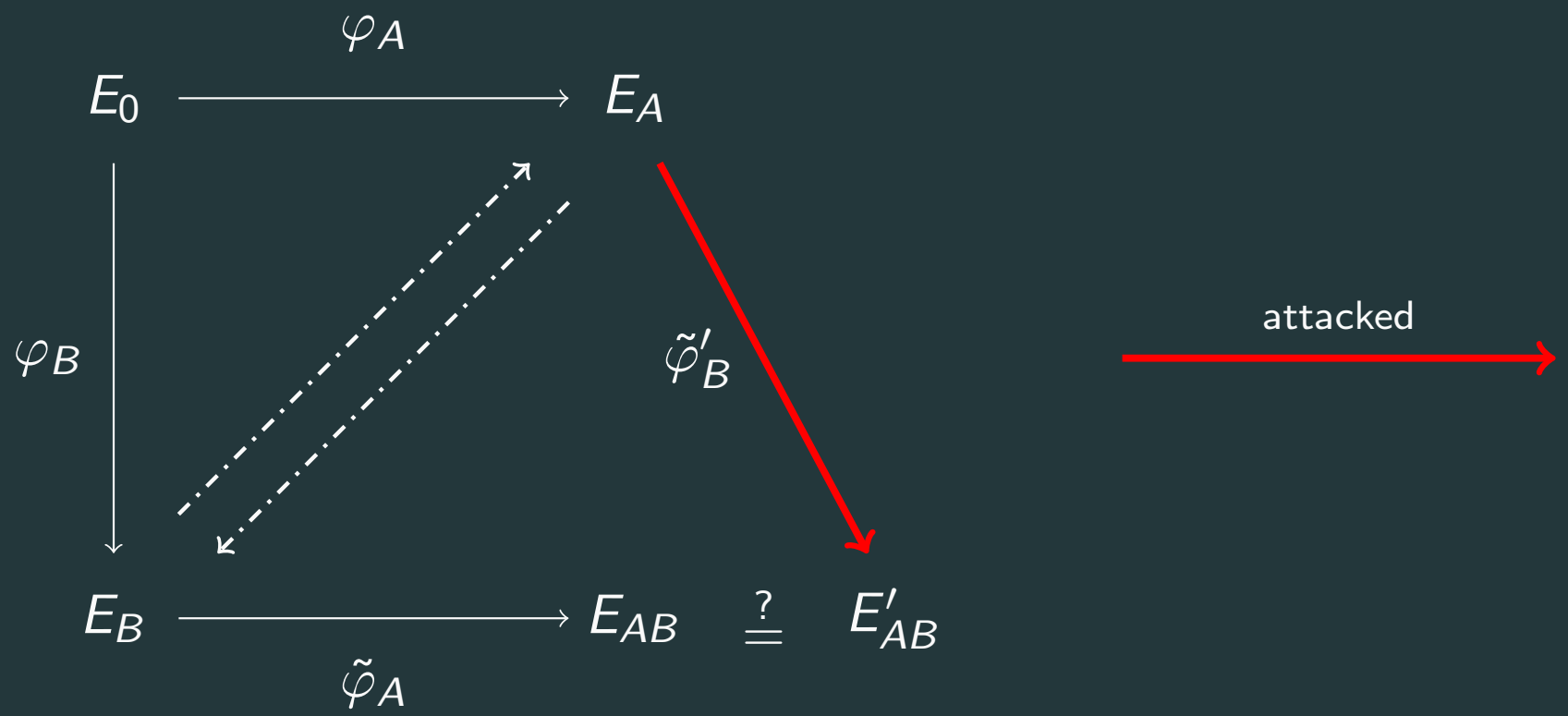
Fault injection caused by clock glitching $\rightsquigarrow$ skip instruction(s)



Attacker models & simulation

- 3 attacker models with **increasing capabilities**
- attacker performs **single** fault injection per run
- **repeatedly evaluation** using same secret key (static-static key exchange)
- injects during computation of **group action**
- check if fault **impacts** shared secret

Setup


φ_A
E_0
E_A
φ_B
$\tilde{\varphi}'_B$
attacked
E_B
E_{AB}
$\stackrel{?}{=}$
E'_{AB}
$\tilde{\varphi}_A$

- **weakest** adversary model
- **no control** over location of fault injection
- **no knowledge of order** of injected isogeny
- ratio failures $\widehat{=}$ **ratio "real" vs. "dummy"**

Photo: Rita Claveau on `https://www.pinterest.it/`

Setup

- isogeny computations effort about **42%**
- **cost-simulation** (python) output transcript of all operations
- **parameterized by relative cost** of operations
- fault into **necessary** operation $\rightsquigarrow$ **wrong** shared secret
- 100 randomly CSIDH512 keys and 500,000 fault injections

Impact

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

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

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

2: Aiming at isogenies at index i

Setup

- **deterministic** computation of e_i : real then dummy[5]
- **out of order** due to point rejections
- first isogenies have large orders ℓ_i
- 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}

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

- **most powerful** attacker model
- additional **side-channel** information exploited
- able to **trace the order** (SPA) of the attacked isogeny

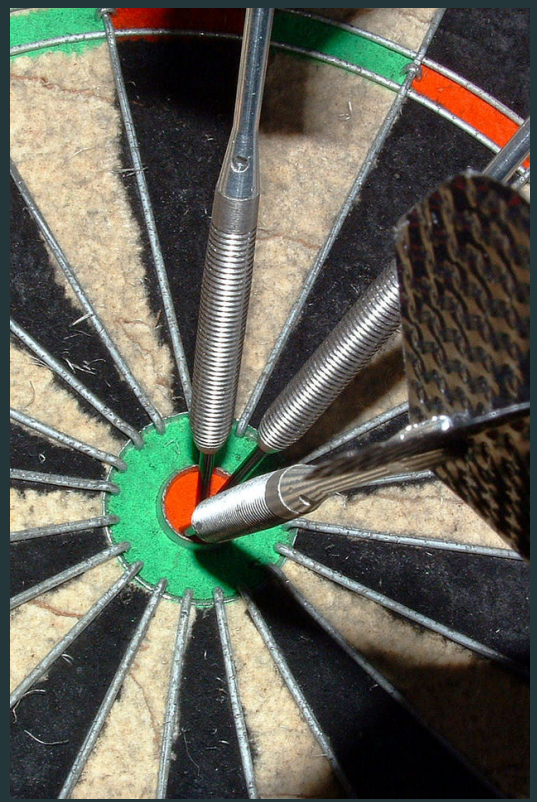


Photo: Alan Belmer on `https://freeimages.com/`

Setup

- secret exponent e_i sampled based on bound vector $\mathtt{m} = (m_1, m_2, \ldots, m_n)$
- binary search for each individual degree until first dummy isogeny is identified

Impact

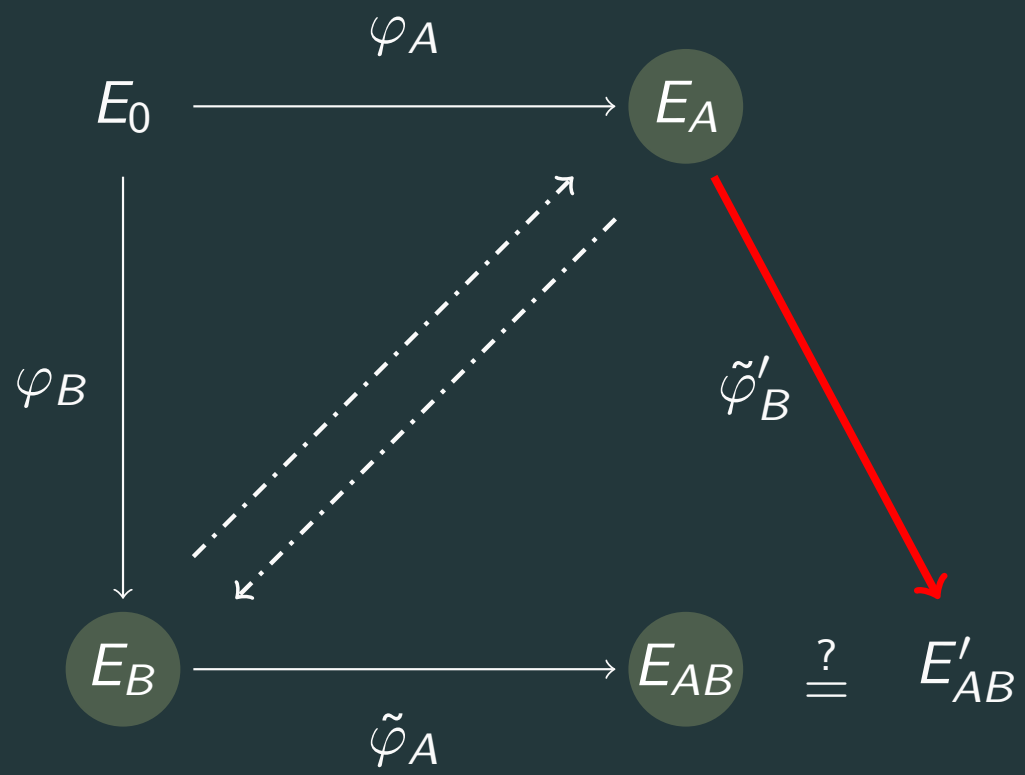
- MCR: full key recovery requires 178 injections
- OAYT: 178 injections $\rightsquigarrow$ space reduction to $2^{67.04}$ (average case);
  further reducible to $\approx 2^{34.5}$ (meet-in-the-middle[6])

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

Practical experiments

- **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)
- Bob's, Alice's public key, and Alice's shared secret **precomputed**
- computation of shared secret attacked

Setup



precomputed

attacked

ChipWhisperer

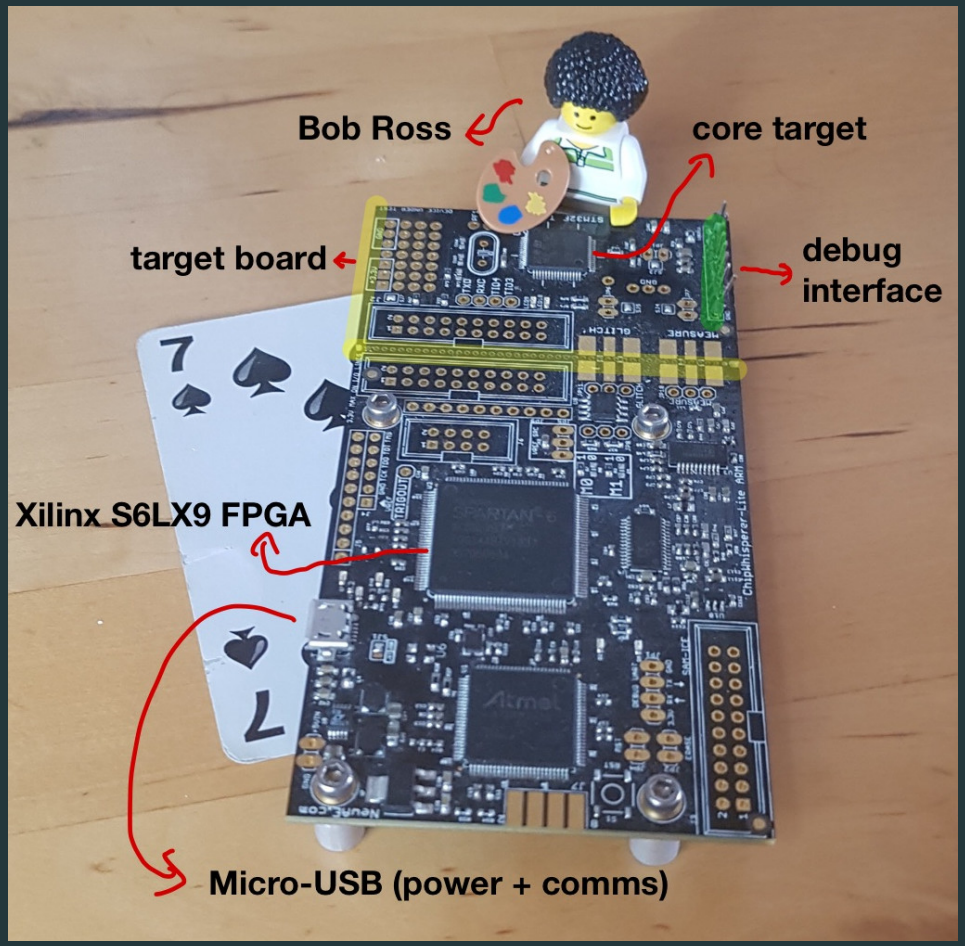


Figure 3: cw1173

- ChipWhisperer-Lite ARM
- 32-bit STM32F303
- open source toolchain
- power analysis
- voltage and clock glitching

Accuracy of the results

Randomized attacks
(w/o knowledge of critical points)

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%** with single injection

Countermeasures & performance

Real vs dummy - different computation blocks

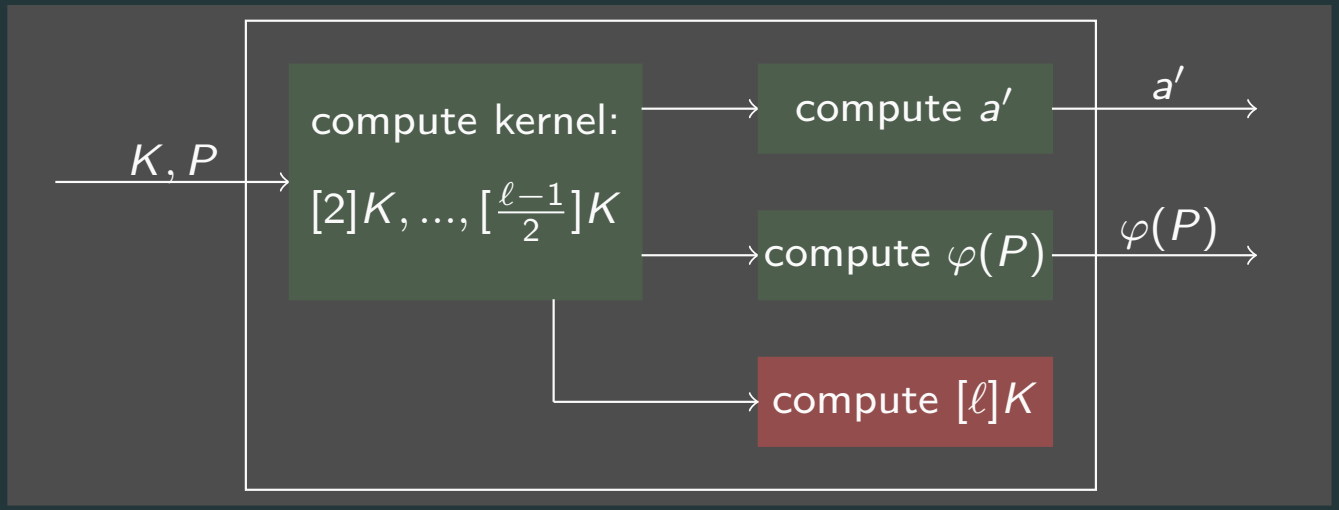


Figure 4: Real isogeny

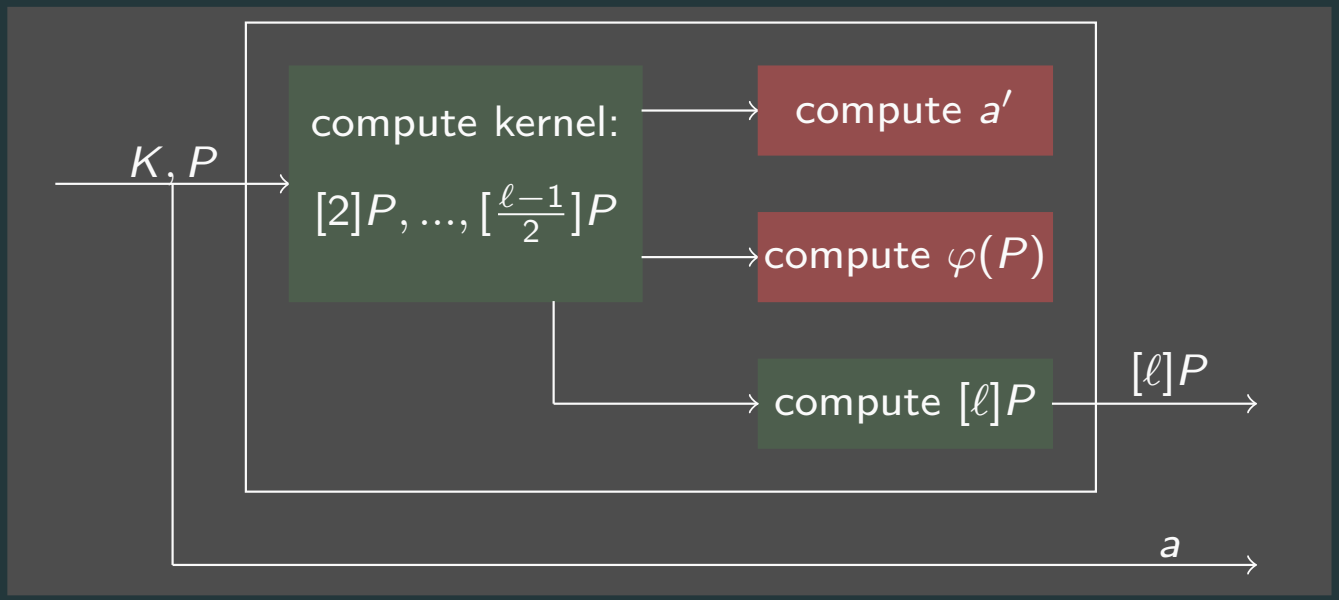


Figure 5: Dummy isogeny

required computation

dummy computation

Countermeasures

Basic idea

- detect injections by changing arithmetic operations

Objectives

- fault injection $\rightsquigarrow$ output an error instead of curve
- countermeasures for both cases to maintain constant-time

Conditional functions

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

Countermeasures: protecting the codomain curve

Input : Curve parameters $A, C \in \mathbb{F}_p$, degree ℓ, kernel points $(X_i : Z_i)$ for $1 \le i \le (\ell-1)/2$, bitmask $b \in \{0,1\}$.
Output: Curve parameters $A', C' \in \mathbb{F}_p$, error variable *error*.

		$b=0$	$b=1$
1	Set $\pi_+ \leftarrow 1$, $\pi_- \leftarrow 1$		
2	**for** $i \in \{1, \ldots, (\ell-1)/2\}$ **do**	(dummy)	(real)
3	$t_0 \leftarrow \texttt{cadd}(X_i, Z_i, b)$	$t_0 = X_i$	$t_0 = X_i + Z_i$
4	$t_1 \leftarrow \texttt{csub}(X_i, Z_i, b)$	$t_1 = X_i$	$t_0 = X_i - Z_i$
5	$\pi_+ \leftarrow \pi_+ \cdot t_0$	$\pi_+ = \prod X_i$	$\pi_+ = \prod (X_i + Z_i)$
6	$\pi_- \leftarrow \pi_- \cdot t_1$	$\pi_- = \prod X_i$	$\pi_- = \prod (X_i - Z_i)$
7	$t_0 \leftarrow \texttt{cadd2}(C, C, b)$	$t_0 = 0$	$t_0 = 2C$
8	$t_1 \leftarrow (A - t_0)^\ell \cdot \pi_-^8$	$t_1 = A^\ell \cdot \pi_-^8$	$t_1 = (A - 2C)^\ell \cdot \pi_-^8$
9	$t_0 \leftarrow (A + t_0)^\ell \cdot \pi_+^8$	$t_0 = A^\ell \cdot \pi_+^8$	$t_0 = (A + 2C)^\ell \cdot \pi_+^8$
10	$A' \leftarrow \texttt{cadd}(t_1, t_0, b)$	$A' = t_1$	$A' = t_0 + t_1$
11	$A' \leftarrow \texttt{cadd}(A', A', b)$	$A' = t_1$	$A' = 2(t_0 + t_1)$
12	$C' \leftarrow \texttt{csub}(t_0, t_1, b)$	$C' = t_0$	$C' = t_0 - t_1$
13	$error \leftarrow \texttt{cverify}(A', C', \neg b)$	$A' \stackrel{?}{=} C'$	
14	**return** $A', C', error$		

Other protected spots/scenarios

- point evaluation@dummy
- differential additions@real
- Elligator
- loop-abort faults
- decision bits
- theoretical "twist-attack"

Performance results on ARM

Overhead for one group action CSIDH512 on Cortex-M4

STM32F407	STM32F303[7]
+5%	+7%

[7] core on ChipWhisperer-Lite

Conclusions

- relatively **small overhead** 5% to 7%
- some countermeasures **applicable to dummy-free** variants
- CSIDH painfully slow $\rightsquigarrow$ experiments with **full scheme infeasible**
- ChipWhisperer: perfectly **adequate**

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/