# Patient Zero & Patient Six: Zero-Value and Correlation Attacks on CSIDH and SIKE

Fabio Campos[1,3], Michael Meyer[2], Krijn Reijnders[3], and Marc Stöttinger[1]

[1] RheinMain University of Applied Sciences Wiesbaden, Germany
campos@sopmac.de
marc.stoettinger@hs-rm.de
[2] University of Regensburg, Germany
michael@random-oracles.org
[3] Radboud University, Nijmegen, The Netherlands
krijn@cs.ru.nl

**Abstract.** Recent works have started side-channel analysis on SIKE and show the vulnerability of isogeny-based systems to zero-value attacks. In this work, we expand on such attacks by analyzing the behavior of the zero curve $E_0$ and six curve $E_6$ in CSIDH and SIKE. We demonstrate an attack on static-key CSIDH and SIKE implementations that recovers bits of the secret key by observing via zero-value-based resp. exploiting correlation-collision-based side-channel analysis whether secret isogeny walks pass over the zero or six curve. We apply this attack to fully recover secret keys of SIKE and two state-of-the-art CSIDH-based implementations: CTIDH and SQALE. We show the feasibility of exploiting side-channel information for the proposed attacks based on simulations with various realistic noise levels. Additionally, we discuss countermeasures to prevent zero-value and correlation-collision attacks against CSIDH and SIKE in our attacker model.

**Keywords:** post-quantum cryptography · isogeny-based cryptography · CSIDH · SIKE · side-channel analysis · zero-value attacks · correlation attacks · countermeasures

## 1 Introduction

Isogeny-based cryptography is a promising candidate for replacing pre-quantum schemes with practical quantum-resistant alternatives. In general, isogeny-based schemes feature very small key sizes, while suffering from running times that are at least an order of magnitude slower than e.g. lattice- or code-based schemes. Therefore, they present a viable option for applications that prioritize bandwidth over performance. SIKE [29], a key encapsulation mechanism (KEM) based on the key exchange SIDH [30], is the lone isogeny-based participant of

* Author list in alphabetical order; see https://www.ams.org/profession/leaders/CultureStatement04.pdf. This work has been supported by the German Federal Ministry of Education and Research (BMBF) under the project QuantumRISC (ID 16KIS1039). Date of this document: 2023-10-20.

the NIST post-quantum cryptography standardization process, and proceeded to the fourth round. In 2018, only after the NIST standardization process started, the key exchange scheme CSIDH was published [14]. Due to its commutative structure, a unique feature among the known post-quantum schemes, CSIDH allows for a non-interactive key exchange, which gained much attention among the research community. Together with its efficient key validation, which enables a static-static key setting, this makes CSIDH a promising candidate for a drop-in replacement of classical Diffie–Hellman-style schemes.

In this work, we focus on a side-channel attack against CSIDH and SIKE. We follow the main idea of [23], which reconstructs SIKE private keys through *zero-value* attacks. This attack approach tries to force zero values for some intermediate values of computations related to secret key bits. By recognizing these zero values via side-channel analysis (SCA), this allows an attacker to recover bits of the secret key. While *coordinate randomization* is an effective method to mitigate general *Differential Power Analysis* (DPA) and *Correlation Power Analysis* (CPA), it has no effect on zero values, such that forcing their occurrence bypasses this countermeasure, which is incorporated in SIKE [29]. Similar to [23], the recent *Hertzbleed attack* exploits zero values in SIKE [45].

While [23] focuses on forcing values connected to elliptic curve points becoming zero, we discuss the occurrence of zero values as curve parameters. This was first proposed in [31], yet [23] concludes that this idea is unlikely to be applicable in a realistic scenario, since curve representations in SIKE are such that they cannot produce a zero. In spite of this fact, we show that some curves in SIKE and CSIDH, as e.g. the zero curve, have a special correlation in these representations, which admits noticing their occurrence via side-channel analysis.

The secret isogeny computation in SIKE essentially consists of two phases: scalar multiplication and isogeny computation. In general, the first phase is believed to be more vulnerable to physical attacks, since private key bits are directly used there (see [19]). We propose the first passive implementation attack using side-channel analysis that exclusively targets the second phase of the SIKE isogeny computation. Notably, countermeasures like coordinate/coefficient randomization [19] or the *CLN test* [20, 23] do not prevent this attack.

**Our contributions.** In this work, we present zero-value and correlation attacks against state-of-the-art implementations of CSIDH and SIKE. For CSIDH, we use the fact that the zero curve $E_0$, i.e., the Montgomery curve with coefficient $a = 0$, represents a valid curve. Thus, whenever a secret isogeny walk passes over this curve, this can be detected via side-channel analysis. We present a passive adaptive attack that recovers one bit of the secret key per round by forcing the target to walk over the zero curve.

Some implementations, like SQALE and SIKE, represent the zero curve without using zero values. Nevertheless, in such a case there is often (with probability $1/2$ in SQALE and probability 1 in SIKE) a strong correlation between certain variables, which also occurs for the supersingular six curve $E_6$ with coefficient

$a = 6$. Via CPA, we exploit this correlation to detect these curves, and mount a similar adaptive attack.

Using these two approaches, we present a generic attack framework, and apply this attack to the state-of-the-art CSIDH implementations SQALE [16] and CTIDH [4] (Section 3), and to SIKE (Section 4). We explore the practical feasibility of the proposed attacks (Section 5), simulations (Section 6), and different types of countermeasures (Section 7). Our code is available in the public domain:

https://github.com/PaZeZeVaAt/simulation

**Related work.** The analysis of physical attacks on isogeny-based schemes has only recently gained more attention, including both side-channel [23, 26, 31, 45, 46] and fault attacks [1, 9, 10, 25, 32, 42, 43]. Introduced for classical elliptic curve cryptography (ECC) in [3, 27, 28], zero-value attacks were adapted to SIKE in [23], which applies t-tests to determine zero values within power traces [41].

An approach to identify certain structures within traces, similar to the ones occurring in non-zero representations of the zero curve and six curve in our case, are correlation-enhanced power analysis collision attacks [36], such as [5] for ECC. This attack combines the concept of horizontal side-channel analysis [38] with correlation-enhanced power analysis collision attacks to extract leakage from a single trace.

We note that from a constructive perspective, this attack follows the idea of steering isogeny paths over special curves, as proposed for the zero curve in [31]. Furthermore, the attack on SIKE uses the framework of [1] to produce suitable public keys. However, our attack is a *passive* attack that is much easier to perform in practice compared to the elaborate fault injection required for [1].

## 2 Preliminaries

We briefly introduce mathematical background related to isogeny-based cryptography, and the schemes CSIDH [14] and SIKE [29]. For more mathematical details, we refer to [22].

**Mathematical background.** Let $\mathbb{F}_q$ with $q = p^k$ denote the finite field of order $q$, with a prime $p > 3$. Supersingular elliptic curves over $\mathbb{F}_q$ are characterized by the condition $\#E(\mathbb{F}_q) \equiv 1 \mod p$. Throughout this work, we will only encounter group orders that are multiples of 4, and hence elliptic curves $E$ over $\mathbb{F}_q$ with $j(E) \in \mathbb{F}_q$ can be represented in Montgomery form:

$$E_a : y^2 = x^3 + ax^2 + x, \quad a \in \mathbb{F}_q. \tag{1}$$

Given two such elliptic curves $E_a$ and $E_{a'}$, an isogeny is a morphism $\varphi : E_a \to E_{a'}$ such that $\infty_{E_a} \mapsto \infty_{E_{a'}}$ for the neutral elements of $E_a$ and $E_{a'}$. In the context of isogeny-based cryptography, we are only interested in separable isogenies, which are characterized by their kernel (up to isomorphism): A finite

subgroup $G \subset E_a(\overline{\mathbb{F}_q})$ defines a separable isogeny $\varphi : E_a \to E_a/G$ and vice versa. In such a case, the degree of $\varphi$ is equal to the size of its kernel, $|G|$. For any isogeny $\varphi : E_a \to E_{a'}$, there is a unique isogeny $\hat{\varphi} : E_{a'} \to E_a$ such that $\hat{\varphi} \circ \varphi = [\deg(\varphi)]$ is the scalar point multiplication on $E_a$ by $\deg(\varphi)$. We call $\hat{\varphi}$ the dual isogeny. Two elliptic curves $E_a$ and $E_{a'}$ over $\mathbb{F}_q$ are isogenous, i.e., there exists an isogeny between them, if and only if $\#E_a(\mathbb{F}_q) = \#E_{a'}(\mathbb{F}_q)$.

### 2.1 CSIDH

In the context of CSIDH, we choose $p$ of the form $p + 1 = h \cdot \prod_{i=1}^{n} \ell_i$ and work with supersingular elliptic curves over $\mathbb{F}_p$. Each $\ell_i$ is a small odd prime, and $h$ is a suitable cofactor to ensure $p$ is prime, with the additional requirement that $4 \mid h$. Usually, we pick $p$ such that $p \equiv 3 \mod 8$ and work with the set $\mathcal{E}$ of supersingular elliptic curves with minimal endomorphism ring $\mathcal{O} \cong \mathbb{Z}[\sqrt{-p}]$. This ensures that the group order $p+1$ is a multiple of 4, and any such supersingular elliptic curve can be represented uniquely in Montgomery form [14], as given by Equation (1) with $a \in \mathbb{F}_p$.

The main operation in CSIDH is the group action of the ideal class group of $\mathcal{O}$ acting on the set $\mathcal{E}$. We are interested in specific ideals $\mathfrak{l}_i$ of $\mathcal{O}$, whose action $\mathfrak{l}_i * E$ on some curve $E \in \mathcal{E}$ is given by an isogeny of degree $\ell_i$ that is defined by the kernel $G = E[\ell_i] \cap E[\pi - 1]$, where $\pi$ denotes the Frobenius endomorphism, i.e., $\mathbb{F}_p$-rational points that have $\ell_i$-torsion. For $E_a \in \mathcal{E}$ we get that $\#E_a = p + 1$, and $E_a(\mathbb{F}_p) \cong \mathbb{Z}_h \times \prod_{i=1}^{n} \mathbb{Z}_{\ell_i}$. This implies there are $\ell_i$ of such points $P \in E[\ell_i] \cap E[\pi - 1]$, and $\ell_i - 1$ of these (all but the point $\infty_{E_a}$) will generate $G$. The codomain $E_{a'}$ of such an isogeny is again supersingular and so $|E_{a'}(\mathbb{F}_p)| = p + 1$, which implies $\mathfrak{l}_i$ can also be applied to $E_{a'}$. This implies a group action of the ideals $\mathfrak{l}_i$ on the supersingular curves $E_a$ over $\mathbb{F}_p$, which we denote by $[\mathfrak{l}_i] * E_a$. In particular, this group action is commutative: $[\mathfrak{l}_i\mathfrak{l}_j] * E_a \cong [\mathfrak{l}_i] * [\mathfrak{l}_j] * E_a \cong [\mathfrak{l}_j] * [\mathfrak{l}_i] * E_a \cong [\mathfrak{l}_j\mathfrak{l}_i] * E_a$. For each $\mathfrak{l}_i$ there exists an inverse $\mathfrak{l}_i^{-1}$, whose action on $E \in \mathcal{E}$ is given by an $\ell_i$-isogeny that is defined by the kernel $G = E[\ell_i] \cap E[\pi + 1]$.

For reasons of brevity, in the following we will sometimes abuse notation and identify the ideals $\mathfrak{l}_i^{\pm 1}$ with the $\ell_i$-isogenies that their action implies.

**The CSIDH scheme.** The CSIDH scheme is based on the group action as described above: We apply each of the $n$ different $\mathfrak{l}_i^{\pm 1}$ a number of times to a given curve $E_a$, and we denote this number by $e_i$. Hence, the secret key is some vector of $n$ integers $(e_1, \ldots, e_n)$ defining an element $\mathfrak{a} = \prod_{i=1}^{n} \mathfrak{l}_i^{e_i}$ which we can apply to supersingular curves $E_a$ over $\mathbb{F}_p$. There is some variation between different proposals on where $e_i$ is chosen from: The original proposal of CSIDH-512 picks $e_i \in \{-m, \ldots, m\}$ with $m = 5$, but one can also define individual bounds $m_i \in \mathbb{Z}$ per $e_i$. The key space is of size $\prod(2m_i + 1)$. For the original CSIDH-512 proposal with $m_i = 5$ and $n = 74$, this gives roughly size $2^{256}$.

The public key is the supersingular curve $E_a$ corresponding to applying the secret key $\mathfrak{a}$ to the publicly known starting curve $E_0 : y^2 = x^3 + x$:

$$E_a = \mathfrak{a} * E_0 = \mathfrak{l}_1^{e_1} * \cdots * \mathfrak{l}_n^{e_n} * E_0. \tag{2}$$

To derive a shared secret between Alice and Bob with secret keys $\mathfrak{a}$ and $\mathfrak{b}$ and given public keys $E_a = \mathfrak{a} * E_0$ and $E_b = \mathfrak{b} * E_0$, Alice simply computes $E_{ab} = \mathfrak{a} * E_b$ and Bob computes $E_{ba} = \mathfrak{b} * E_a$. From the commutativity of the group action, we get $E_{ab} \cong E_{ba}$.

**Security of CSIDH.** The classical security relies mostly on the size of the keyspace $\prod(2m_i + 1)$, but the quantum security of CSIDH is heavily dependent on the size of the group generated by these elements $\mathfrak{l}_i$. It is heuristically assumed that the $\mathfrak{l}_i$ generate a group of size approximately $\sqrt{p}$. While the original CSIDH proposal considered a 512-bit prime $p$ sufficient for NIST security level 1 [14], its exact quantum security is debated [7, 40, 8, 16]. For instance, [16] claims that 4096-bit primes are required for level 1 security. Note that the key space is not required to cover the full group of size roughly $\sqrt{p}$, but can be chosen as a large enough subset, except for particularly bad choices like subgroups. At larger prime sizes, the number $n$ of small primes $\ell_i$ grows, and therefore it becomes natural to pick secret key vectors from $\{-1, 0, 1\}^n$ resp. $\{-1, 1\}^n$ for primes sizes of at least 1792 resp. 2048 bits. This allows for a large enough key space for classical security, while increasing $p$ for sufficient quantum security.

We note that the exact quantum security of CSIDH remains unclear, and thus work on efficient and secure implementations for both smaller and larger parameters continues to appear, e.g. in [4, 16].

**Constant-time implementations.** CSIDH is inherently difficult to implement in constant time, as this requires that the timing of the execution is independent of the respective secret key $(e_1, \ldots, e_n)$. However, picking a secret key vector $(e_1, \ldots, e_n)$ translates to the computation of $|e_i|$ isogenies of degree $\ell_i$, which directly affects the timing of the group action evaluation. One way to mitigate this timing leakage is by using dummy isogenies: We can keep the total number of isogenies per degree constant by computing $m_i$ isogenies of degree $\ell_i$, but discarding the results of $m_i - |e_i|$ of these, effectively making them dummy computations [34, 33]. Several optimizations and different techniques have been proposed in the literature [39, 15, 18].

The latest and currently most efficient variant of constant-time implementations of CSIDH is CTIDH [4]. In contrast to sampling private key vectors such that $e_i \in \{-m_i, \ldots, m_i\}$, CTIDH uses a different key space that exploits the approach of batching the primes $\ell_i$. We define *batches* $B_1, \ldots, B_N$ of consecutive primes of lengths $n_1, \ldots, n_N$, i.e., $B_1 = (\ell_{1,1}, \ldots, \ell_{1,n_1}) = (\ell_1, \ldots, \ell_{n_1})$, $B_2 = (\ell_{2,1}, \ldots, \ell_{2,n_2}) = (\ell_{n_1+1}, \ldots, \ell_{n_1+n_2})$, et cetera. We write $e_{i,j}$ for the (secret) coefficient associated to $\ell_{i,j}$. Instead of defining bounds $m_i$ for each individual $\ell_i$ so that $|e_i| \leq m_i$, CTIDH uses bounds $M_i$ for the batch $B_i$, i.e., we

compute at most $M_i$ isogenies of those degrees that are contained in $B_i$. That is, the key sampling requires $|e_{i,1}| + \cdots + |e_{i,n_i}| \leq M_i$. CTIDH then adapts the CSIDH algorithm such that the distribution of the $M_i$ isogenies among degrees of batch $B_i$ does not leak through the timing channel. Among other techniques, this involves Matryoshka isogenies, first introduced in [7], that perform the exact same sequence of instructions independent of its isogeny degree $\ell_{i,j} \in B_i$.

The main advantage of CTIDH is the ambiguity of the isogeny computations: From a time-channel perspective, a Matryoshka isogeny for $B_i$ could be an $\ell_{i,j}$-isogeny for any $\ell_{i,j} \in B_i$. Thus, in comparison to the previous CSIDH algorithms, CTIDH covers the same key space size in fewer isogenies. For instance, the previously fastest implementation of CSIDH-512 required 431 isogenies in total [2] (including dummies), whereas CTIDH [4] requires only 208 isogenies (including dummies) for the same key space size. This leads to an almost twofold speedup.

**Representation of Montgomery coefficient.** To decrease computational cost by avoiding costly inversions, the curve $E_a$ is almost always represented using *projective* coordinates for $a \in \mathbb{F}_p$. The following two are used most in current CSIDH-based implementations:

- the Montgomery form $(A : C)$, such that $a = A/C$, with $C$ non-zero,
- and the alternative Montgomery form $(A + 2C : 4C)$, such that $a = A/C$, with $C$ non-zero.

The alternative Montgomery form is most common, as it is used in projective scalar point multiplication formulas. Hence, in most state-of-the-art implementations of CSIDH-based systems, the Montgomery coefficient $a$ is mapped to alternative Montgomery form and remains in this form until the end, where it is mapped back to affine form for the public key resp. shared secret (e.g., in SQALE [16]). CTIDH [4] switches between both representations after each isogeny, and maps back to affine $a = A/C$ at the end. For most values of $(A : C)$ and $(A + 2C : 4C)$, $a = A/C$ represents either an ordinary or a supersingular curve. The exceptions are $C = 0$, which represents no algebraic object, and $A = \pm 2C$, which represents the singular curves $E_{\pm 2}$. Specifically the supersingular zero curve $E_0$ is represented as $(0 : C)$ in Montgomery form and $(2C : 4C)$ in alternative Montgomery form, where $C \in \mathbb{F}_p$ can be any non-zero value.

**Isogeny computation in projective form.** When using projective representations to compute isogenies with domain $E_a$ where $a$ is represented as $(A : C)$, most implementations use projectivized versions of Vélu's formulas, described in [44, 35, 6]. To compute the action of $\mathfrak{l}_i^{\pm 1}$ on $E_a$, one finds a point $P$ of order $\ell_i$ on $E_a$ and computes the $x$-coordinates of the points $\{P, [2]P, \ldots, [\frac{\ell_i - 1}{2}]P\}$. Let $(X_k : Z_k)$ denote the $x$-coordinate of $[k]P$ in projective form. Then, the projective Montgomery coefficient $(A' : C')$ of $E_{a'} = \mathfrak{l}_i * E_a$ using Montgomery form

$(A : C)$ is computed by

$$B_z = \prod_{k=1}^{\frac{\ell-1}{2}} Z_k, \quad A' = (A + 2C)^\ell \cdot B_z^8, \tag{3}$$

$$B_x = \prod_{k=1}^{\frac{\ell-1}{2}} X_k, \quad C' = (A - 2C)^\ell \cdot B_x^8, \tag{4}$$

and when using alternative Montgomery form $(\alpha : \beta) = (A + 2C : 4C)$ by

$$B_z = \prod_{k=1}^{\frac{\ell-1}{2}} Z_k, \quad \alpha' = \alpha^\ell \cdot B_z^8, \tag{5}$$

$$B_x = \prod_{k=1}^{\frac{\ell-1}{2}} X_k, \quad \beta' = \alpha' - (\alpha - \beta)^\ell \cdot B_x^8, \tag{6}$$

where $(\alpha' : \beta')$ represents $E_{a'}$ in alternative Montgomery form. Note that the values $(A + 2C)$ in (3), $(A - 2C)$ in (4), $\alpha$ in (5) and $(\alpha - \beta)$ in (6) are never zero: In all cases, this implies $A/C = \pm 2$, i.e., the singular curves $E_{\pm 2}$.

*Remark 1.* So far, we know of no deterministic implementations based on the class group action. This is because in order to perform the isogenies, all current implementations sample a random point $P$ on the curve and compute the scalar multiple of $P$ required to perform isogenies. The projective coordinates $(X_k : Z_k)$ are then non-deterministic, and hence the output of Equations (3) to (6) is non-deterministic. This implies that the representation of $a$ as $(A : C)$ or $(A + 2C : 4C)$ is non-deterministic after the first isogeny. A deterministic approach, e.g. as sketched in [7] using Elligator, ensures a deterministic representation of $a$, but has so far not been put into practice.

### 2.2 SIKE

In SIKE, we pick a prime of the form $p = 2^{e_A} \cdot 3^{e_B} - 1$ such that $2^{e_A} \approx 3^{e_B}$, and work with supersingular elliptic curves over $\mathbb{F}_{p^2}$ in Montgomery form. We choose to work with curves such that $E_a(\mathbb{F}_{p^2}) = (p+1)^2$, and we have $E_a(\mathbb{F}_{p^2}) \cong \mathbb{Z}_{2^{e_A}}^2 \times \mathbb{Z}_{3^{e_B}}^2$ for these curves. Thus, the full $2^{e_A}$- and $3^{e_B}$-torsion subgroups lie in $E_a(\mathbb{F}_{p^2})$. Any point $R_A$ of order $2^{e_A}$ then uniquely (up to isomorphism) determines a $2^{e_A}$-isogeny and codomain curve $E_{a'} = E_a/\langle R_A \rangle$ with kernel $\langle R_A \rangle$. For choosing an appropriate point, the SIKE setup defines basis points $P_A$ and $Q_A$ of the $2^{e_A}$-torsion of the public starting curve. Picking an integer $\mathtt{sk}_A \in [0, 2^{e_A} - 1]$ and computing $R_A = P_A + [\mathtt{sk}_A]Q_A$ then results in choosing such a kernel generator $R_A$ of order $2^{e_A}$.

In practice, such a $2^{e_A}$-isogeny is computed as a sequence of 2-isogenies of length $e_A$. This can be interpreted as a sequence of steps through a graph: For

a prime $\ell$ with $\ell \neq p$, the $\ell$-isogeny graph consists of vertices that represent ($j$-invariants of) elliptic curves, and edges representing $\ell$-isogenies. Due to the existence of dual isogenies, edges are undirected. For supersingular curves, this graph is an $(\ell+1)$-regular expander graph and contains approximately $p/12$ vertices. Hence, a sequence of 2-isogenies of length $e_A$ corresponds to a walk of length $e_A$ through the 2-isogeny graph. An analogous discussion applies to the case of $3^{e_B}$-isogenies. Note that for reasons of efficiency, we often combine two 2-isogeny steps into one 4-isogeny.

The secret keys $\mathtt{sk}_A$, $\mathtt{sk}_B$ can be decomposed as

$$\mathtt{sk}_A = \sum_{i=0}^{e_2-1} \mathtt{sk}_i \cdot 2^i \quad \mathtt{sk}_i \in \{0,1\}, \quad \mathtt{sk}_B = \sum_{i=0}^{e_3-1} \mathtt{sk}_i \cdot 3^i \quad \mathtt{sk}_i \in \{0,1,2\}.$$

We refer to these $\mathtt{sk}_i$ as the *bits* resp. the *trits* of the secret key $\mathtt{sk}_A$ resp. $\mathtt{sk}_B$. For a given $\mathtt{sk}$, we use $\mathtt{sk}_{<k}$ to represent the key up to the $k$-th bit/trit $\mathtt{sk}_{k-1}$.

**The SIKE scheme.** The main idea behind SIDH and SIKE is to use secret isogenies to set up a key exchange scheme resp. key encapsulation mechanism. SIDH fixes $E_6$ as starting curve, and torsion basis points $P_A, Q_A$ and $P_B, Q_B$. It uses the following subroutines:

- $\mathtt{KeyGen}_A$ samples a secret key $\mathtt{sk}_A \in [0, 2^{e_A}-1]$, computes $R_A = P_A + [\mathtt{sk}_A]Q_A$, and the secret isogeny $\phi_A : E_6 \to E_6/\langle R_A \rangle$. It outputs the key pair $(\mathtt{sk}_A, \mathtt{pk}_A)$, where $\mathtt{pk}_A = (\phi_A(P_B), \phi_A(Q_B), \phi_A(Q_B - P_B))$. We write $\mathtt{KeyGen}_A(\mathtt{sk})$ if $\mathtt{KeyGen}_A$ does not sample a secret key, but gets $\mathtt{sk}$ as input.
- $\mathtt{KeyGen}_B$ proceeds analogously with swapped indices $A$ and $B$. The public key is $\mathtt{pk}_B = (\phi_B(P_A), \phi_B(Q_A), \phi_B(Q_A - P_A))$.
- $\mathtt{Derive}_A$ takes as input $(\mathtt{sk}_A, \mathtt{pk}_B) = (S_A, T_A, T_A - S_A)$. It computes the starting curve $E_B$ from the points in $\mathtt{pk}_B$, the secret point $R'_A = S_A + [\mathtt{sk}_A]T_A$, and the isogeny $\phi'_A : E_B \to E_B/\langle R'_A \rangle$.
- $\mathtt{Derive}_B$ proceeds analogously with input $(\mathtt{sk}_B, \mathtt{pk}_A)$, and computes the co-domain curve $E_A/\langle R'_B \rangle$.

When running this key exchange, both parties arrive at a curve (isomorphic to) $E_6/\langle R_A, R_B \rangle$, and (a hash of) its $j$-variant can serve as a shared secret.

SIKE uses the SIDH subroutines $\mathtt{KeyGen}$ and $\mathtt{Derive}$ to construct three algorithms $\mathtt{KeyGen}$, $\mathtt{Encaps}$, and $\mathtt{Decaps}$. Furthermore, we define $h$ and $h'$ to be cryptographic hash functions.

- $\mathtt{KeyGen}$ generates a (static) key pair $(\mathtt{sk}, \mathtt{pk}) \leftarrow \mathtt{KeyGen}_B$.
- $\mathtt{Encaps}$ encapsulates a random value $m$ in the following way:
  - Get an ephemeral key pair $(\mathtt{ek}, c) \leftarrow \mathtt{KeyGen}_A(\mathtt{ek})$ with $\mathtt{ek} = h(\mathtt{pk}, m)$.
  - Compute the shared secret $s \leftarrow \mathtt{Derive}_A(\mathtt{ek}, \mathtt{pk})$.
  - Compute the ciphertext $\mathtt{ct} = (c, h'(s) \oplus m)$.
- $\mathtt{Decaps}$ receives a ciphertext $(c_0, c_1)$, and proceeds as follows:
  - Compute the shared secret $s' \leftarrow \mathtt{Derive}_B(\mathtt{sk}, c_0)$.

- Recover $m' \leftarrow c_1 \oplus h'(s')$.
- Recompute $\mathtt{ek}' = h(\mathtt{pk}, m')$.
- Compute $(\mathtt{ek}', c') \leftarrow \mathtt{KeyGen}_A(\mathtt{ek}')$ and check if $c' = c_0$.

Passing this check guarantees that the ciphertext has been generated honestly, and $m' = m$ can be used to set up a session key.

**Representation of Montgomery coefficients.** As in CSIDH, the curve $E_a$ is almost always represented using projective coordinates, with the caveat that $a \in \mathbb{F}_{p^2}$. The following two representations are used throughout SIKE computations, although in different subroutines.

- The alternative Montgomery form $(A + 2C : 4C)$, such that $a = A/C$ with $C$ non-zero. This representation is used for Alice's computations as it is the most efficient for computing 2-isogenies. It is often written as $(A^+_{24} : C_{24})$ with $A^+_{24} = A + 2C$ and $C_{24} = 4C$ so that $a = 2(2A^+_{24} - C_{24})/C_{24}$.
- The form $(A + 2C : A - 2C)$, such that $a = A/C$, with $C$ non-zero. This representation is used for Bob's computations as it is the most efficient for computing 3-isogenies. It is often written as $(A^+_{24} : A^-_{24})$ with $A^+_{24} = A + 2C$ and $A^-_{24} = A - 2C$ so that $a = 2(A^+_{24} + A^-_{24})/(A^+_{24} - A^-_{24})$.

Note that the values $A, C, A^+_{24}, A^-_{24}$ and $C_{24}$ are in $\mathbb{F}_{p^2}$. When necessary, we write them as $\alpha + \beta i$ with $\alpha, \beta \in \mathbb{F}_p$ and $i^2 = -1$. Equal to CSIDH, both forms represent either an ordinary or a supersingular curve, with the exceptions $C = 0$, which represents no algebraic object, and $A = \pm 2C$, which represents the singular curves $E_{\pm 2}$. For the rest of the paper, we are interested in representations of the supersingular six curve $E_6$. Fortunately, $E_6$ is represented in *both* forms as $(8C : 4C)$, with $C = \alpha + \beta i \in \mathbb{F}_{p^2}$ any non-zero element. For the goal of the paper, this means that the analysis is similar for both forms.

**Isogeny computation in projective form.** SIKE uses the above projective representations to compute the codomain $E_{\tilde{a}}$ of a 3- or 4-isogeny $\phi : E_a \to E_{\tilde{a}}$.

*4-isogeny.* Given a point $P$ of order 4 on $E_a$ with $x$-coordinate $x(P) = (X : Z)$, the codomain $E_{\tilde{a}} = E_a/\langle P \rangle$ with $\tilde{a}$ represented by $(\tilde{A}^+_{24} : \tilde{C}_{24})$ is computed by

$$\tilde{A}^+_{24} = 4 \cdot X^4, \qquad \tilde{C}_{24} = 4 \cdot Z^4. \tag{7}$$

*3-isogeny.* Given a point $P$ of order 3 on $E_a$ with $x$-coordinate $x(P) = (X : Z)$, the codomain $E_{\tilde{a}} = E_a/\langle P \rangle$ with $\tilde{a}$ represented by $(\tilde{A}^+_{24} : \tilde{A}^-_{24})$ is computed by

$$\tilde{A}^+_{24} = (3X + Z)^3 \cdot (X - Z), \quad \tilde{A}^-_{24} = (3X - Z)^3 \cdot (X + Z). \tag{8}$$

## 3 Recovering CSIDH keys with $E_0$ side-channel leakage

In this section, we explore how side-channel information can leak information on secret isogeny walks. As shown in [23], it is possible to detect zero values in isogeny computations using side-channel information. In Section 3.1, we specifically explore how both representations of the zero curve $E_0$, i.e. $(0 : C)$ and $(2C : 4C)$, leak secret information, even though the value $C \in \mathbb{F}_p$ is assumed to be a uniformly random non-zero value. As $E_0$ is always a valid supersingular $\mathbb{F}_p$-curve in CSIDH, we can always construct a walk that potentially passes over $E_0$. This allows us to describe a generic approach to leak a given bit of information of the secret isogeny walk, hence, a general attack on the class group action as introduced in CSIDH. We apply this attack in more detail to the two current state-of-the-art cryptosystems based on this class group action: SQALE in Section 3.2 and CTIDH in Section 3.3. We discuss their practical feasibility in Section 5 and simulate these attacks in Section 6. We note that the proposed attack applies to all variants of CSIDH that we know of, e.g. from [14, 15].

Throughout this work, we assume a static-key setting, i.e., that a long-term secret key $\mathfrak{a}$ is used, and that the attacker can repeatedly trigger key exchange executions on the target device using public key curves of their choice. Formally, this means that we adaptively feed curves $E_{\mathtt{PK}}$ and get side-channel information on the computations $\mathfrak{a} * E_{\mathtt{PK}}$. We exploit this information to reveal $\mathfrak{a}$ bit by bit.

### 3.1 Discovering a bit of information on a secret isogeny walk

**Detecting $E_0$ in Montgomery form.** As described in Remark 1, the representation of the Montgomery coefficient as $(A : C)$ or $(A + 2C : 4C)$ is non-deterministic after the first isogeny, so they effectively contain random $\mathbb{F}_p$-values, representing the affine Montgomery coefficient $a$. This makes it hard to get any information on $E_a$ using side channels. However, in Montgomery form the curve $E_0$ is special: It is simply represented by $(0 : C)$ for some $C \in \mathbb{F}_p$. We define such a representation containing a zero a *zero-value representation.*

**Definition 1.** *Let $E_a$ be an elliptic curve over $\mathbb{F}_p$. A* zero-value representation *is a representation of the Montgomery coefficient $a$ in projective coordinates $(\alpha : \beta)$ such that either $\alpha = 0$ or $\beta = 0$.*

Clearly, a representation of $E_0$ in Montgomery form must be a zero-value representation. As is known for ECC and SIKE, an attacker can observe zero-value representations in several different ways using side-channel analysis [23]. We will expand on this in Section 5 to show that $E_0$ leaks secret information in implementations that use Montgomery form.

**Detecting $E_0$ in alternative Montgomery form.** Using the alternative Montgomery form, no non-singular curve has a zero-value representation, as $(A + 2C : 4C)$ can only be zero for $A = -2C$ corresponding to $a = -2$, which represents the singular curve $E_{-2}$. Thus, the alternative Montgomery form avoids

the side-channel attack described above. Nevertheless, the representation of $E_0$ is still unusual: Whenever $2C$ is smaller than $p/2$, doubling $2C$ does not require a modular reduction, and hence the bit representation of $4C$ is precisely a bit shift of $2C$ by one bit to the left. Such strongly correlated values can be observed in several ways using side-channel analysis, as we detail later in Section 5.

**Definition 2.** *Let $E_a$ be an elliptic curve over $\mathbb{F}_p$. A* strongly-correlated representation *is a representation of the Montgomery coefficient $a$ in projective coordinates $(\alpha : \beta)$ such that the bit representations of $\alpha$ and $\beta$ are bit shifts.*[4]

For $E_0$, for any non-zero value $C$ with $2C \leq p/2$, the representation in alternative Montgomery form by $(2C : 4C)$ is a strongly-correlated representation. As $C$ is effectively random during the computation of the class group action, in roughly 50% of the cases where we pass over $E_0$, the representation is strongly correlated. For random values of $a$, the values of $(A + 2C : 4C)$ are indistinguishable from random $(\gamma : \delta)$, and so an attacker can differentiate $E_0$ from such curves. From this, an attacker only needs a few traces to determine accurately whether a walk passes over $E_0$ or not, as discussed in Section 5.

*Remark 2.* Other curves have strongly-correlated representations too, e.g., the curve $E_6$ requires $A = 6C$ which gives $(8C : 4C)$ with $C \in \mathbb{F}_p$ random and non-zero, and so $E_6$ can be detected in precisely the same way as $E_0$. For simplicity, we focus on the zero curve in the CSIDH attack. We note that analyzing this attack to any curve with strongly-correlated representations is of independent interest for CSIDH and other isogeny-based schemes (such as SIKE).

*Remark 3.* In the case where $2C$ is larger than $p/2$, the modular reduction by $p$ decreases the correlation between $2C$ and $4C$ significantly, which is why we disregard these cases. However, a modular reduction does not affect all bits, and so this correlation remains for unaffected bits. Especially for primes with large cofactor $2^k$ in $p + 1$, or primes close to a power of 2, the correlation between unaffected bits should be exploitable. For the primes used in the CSIDH instances in this work, this effect is negligible. However, the primes used in SIDH and SIKE do have this form and we exploit this in Section 4.

The idea is now to detect $E_0$ in a certain step $k$ of the computation $\mathfrak{a} * E_{\mathtt{PK}}$. In order to ensure that this happens the computation needs to be performed in a known order of isogeny steps $E \to \mathfrak{l}^{(k)} * E$. In general, by the way how isogenies are computed, such a step can fail with a certain probability. The following definition takes this into account.

**Definition 3.** *Let $\mathfrak{a}$ be a secret isogeny walk. An* ordered evaluation *of $\mathfrak{a} * E$ is an evaluation in a fixed order*

$$\mathfrak{l}^{(n)} * \ldots * \mathfrak{l}^{(1)} * E$$

[4] This definition may be expanded to cover other types of correlation, whenever such correlation can be distinguished from random values using side-channel information.

*of $n$ steps, assuming that no step fails. We write $\mathfrak{a}_k * E$ for the first $k$ steps of of such an evaluation,*

$$\mathfrak{l}^{(k)} * \ldots * \mathfrak{l}^{(1)} * E.$$

*We define $p_{\mathfrak{a}}$ resp. $p_{\mathfrak{a}_k}$ as the probability that $\mathfrak{a}$ resp. $\mathfrak{a}_k$ is evaluated without failed steps.*

**Generic approach to discover isogeny walks using $E_0$.** Given the ability to detect $E_0$ in a walk for both the Montgomery form and the alternative Montgomery form, we sketch the following approach to discover bits of a secret isogeny walk $\mathfrak{a}$ that has an ordered evaluation. Assuming we know the first $k-1$ steps $\mathfrak{l}^{(k-1)} * \ldots * \mathfrak{l}^{(1)}$ in the secret isogeny walk $\mathfrak{a}$, denoted by $\mathfrak{a}_{k-1}$, we want to see if the $k$-th step $\mathfrak{l}^{(k)}$ equals $\mathfrak{l}_i$ or $\mathfrak{l}_i^{-1}$ for some $i$. We compute $E_a = \mathfrak{l}_i^{-1} * E_0$ and $E_{a'} = \mathfrak{l}_i^{-1} * E_a$, and as a public key we use $E_{\mathtt{PK}} = \mathfrak{a}_{k-1}^{-1} * E_a$. Then, when applying the secret walk $\mathfrak{a}$ to $E_{\mathtt{PK}}$, the $k$-th step either goes over $E_0$ or over $E_{a'}$. From side-channel information, we observe if the $k$-th step applies $\mathfrak{l}_i^e = \mathfrak{l}_i^1$ or $\mathfrak{l}_i^e = \mathfrak{l}_i^{-1}$, and set $\mathfrak{l}^{(k)} = \mathfrak{l}_i^e$, as shown in Figure 1. Then we repeat with $\mathfrak{a}_k = \mathfrak{l}_i^e \cdot \mathfrak{a}_{k-1}$.

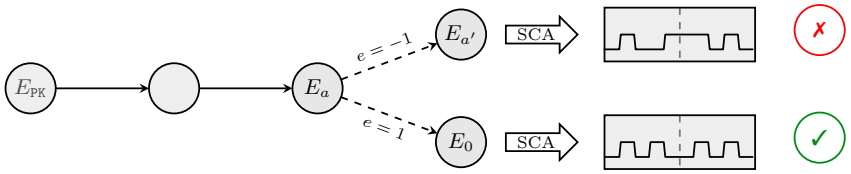



Fig. 1: Generic approach to discover secret bits using side-channel information.

If $E_0$ is not detected in the above setting, i.e. $e = -1$, we can confirm this by an additional measurement: We compute $\tilde{E_a} = \mathfrak{l} * E_0$ and $\tilde{E_{a'}} = \mathfrak{l} * \tilde{E_a}$, and use $\tilde{E}_{\mathtt{PK}} = \mathfrak{a}_{k-1}^{-1} * \tilde{E_a}$ as public key. If $e = -1$, the isogeny walk now passes over $E_0$, which can be recognized via side-channel analysis. More formally, we get:

**Lemma 1.** *Let $\mathfrak{a}$ be any isogeny walk of the form $\mathfrak{a} = \prod \mathfrak{l}_i^{e_i}$. Assume the evaluation of $\mathfrak{a}$ is an ordered evaluation. Then, there exists a supersingular curve $E_{\mathtt{PK}}$ over $\mathbb{F}_p$ such that $\mathfrak{a} * E_{\mathtt{PK}}$ passes over $E_0$ in the $k$-th step.*

After successfully detecting all steps $\mathfrak{l}^{(k)}$, the private key elements $e_i$ can simply be recovered by counting how often $\mathfrak{l}_i$ resp. $\mathfrak{l}_i^{-1}$ appeared in the evaluation.

This generic approach has a nice advantage: If one detects the $k$-th step to walk over $E_0$, this confirms all previous steps were guessed correctly. In other words, guessing wrongly in a certain step will be noticed in the next step: Denote a wrong guess by $\mathfrak{a}_k^{\text{wrong}} = \mathfrak{l}^{-e} \cdot \mathfrak{a}_{k-1}$. The attacker computes $E_a$ from $E_0$ so that $\mathfrak{l}' * E_a = E_0$ and gives the target $E_{\mathtt{PK}}$ such that $\mathfrak{a}_k^{\text{wrong}} * E_{\mathtt{PK}} = E_a$. Due to the wrong guess, neither $e = 1$ nor $e = -1$ lead to $E_0$, as the *actual* secret walk $\mathfrak{a}$ leads to $E_{a'} = \mathfrak{a}_k * E_{\mathtt{PK}}$, and the case $e = 1$ leads to $E_{\neg 0} = \mathfrak{l}' * E_{a'} = \mathfrak{l}^{-2e} * E_0$, as shown in Figure 2.

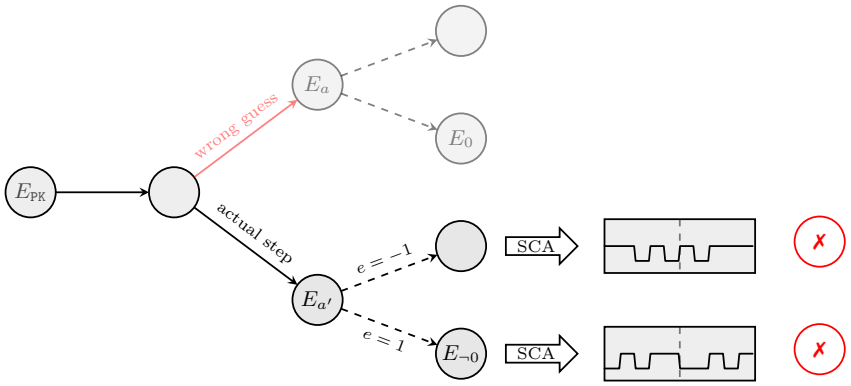



Fig. 2: Due to a wrong guess of the isogeny path $\mathfrak{a}_k$, an attacker miscomputes $E_{\mathrm{PK}}$ and the actual walk does not pass over $E_0$.

*Remark 4.* Note that $E_{\mathrm{PK}}$ given by Lemma 1 is a valid CSIDH public key, so public key validation (see [14]) does not prevent this attack.

**Probability of a walk passing over $E_0$.** Due to the probabilistic nature of the computation of the class group action, not every evaluation $\mathfrak{a} * E_{\mathrm{PK}}$ passes over $E_0$ in the $k$-th step: One of the steps $\mathfrak{l}^{(j)}$ for $1 \leq j \leq k-1$ can fail with probability $1/\ell^{(j)}$, and if so, the $k$-th step passes over a different curve. With $E_{\mathrm{PK}}$ as given by Lemma 1, the probability that an ordered evaluation $\mathfrak{a} * E_{\mathrm{PK}}$ passes over $E_0$ is then described by $p_{\mathfrak{a}_k}$, which we compute in Lemma 2.

**Lemma 2.** *Let $\mathfrak{a}$ be an isogeny walk computed as an ordered evaluation $\mathfrak{l}^{(n)} * \ldots * \mathfrak{l}^{(1)} * E_{\mathrm{PK}}$. Then $p_{\mathfrak{a}_k}$, the probability that the first $k$ isogenies succeed, is*

$$p_{\mathfrak{a}_k} := \prod_{j=1}^{k} \frac{\ell^{(j)} - 1}{\ell^{(j)}}$$

*where $\ell^{(j)}$ is the degree of the isogeny $\mathfrak{l}^{(j)}$ in the $j$-th step.*

As $p_{\mathfrak{a}_k}$ describes the chance that we pass over $E_0$ in the $k$-th step, $1/p_{\mathfrak{a}_k}$ gives us the estimated number of measurements of $\mathfrak{a} * E_{\mathrm{PK}}$ we need in order to pass over $E_0$ in step $k$. We apply this more concretely in Sections 3.2 and 3.3.

*Remark 5.* Instead of learning bit by bit starting from the beginning of the secret isogeny walk, we can also start at the end of the walk. To do so, we use the twist $E_{-t}$ of the target's public key $E_t$, for which $\mathfrak{a} * E_{-t} = E_0$. As for the generic attack, we feed $E_{\mathrm{PK}} = \mathfrak{l}^{-1} * E_{-t}$ and $\tilde{E}_{\mathrm{PK}} = \mathfrak{l} * E_{-t}$. The computation then passes over $E_0$ in the *last* step instead of the *first*. This approach requires the same probability $p_{\mathfrak{a}_k}$ to recover the $k$-th bit, but assumes knowledge of all bits after $k$ instead of before. Hence, we can discover starting and ending bits of $\mathfrak{a}$ in parallel.

### 3.2 Recovering secret keys in SQALE

SQALE [16] is the most recent and most efficient constant-time implementation of CSIDH for large parameters, featuring prime sizes between 1024 and 9216 bit. In this section, we explain how the attack from Section 3.1 can be applied to SQALE, leading to a full key recovery. For concreteness, we focus on SQALE-2048, which uses parameters $n = 231$ and secret exponents $e_i \in \{-1, 1\}$ for $1 \leq i \leq 221$. The $\ell_i$ with $i > 221$ are not used in the group action.

**Algorithmic description of SQALE.** Given a starting curve $E_A$, the SQALE implementation computes the group action in the following way:

- Sample random points $P_+ \in E_A[\pi - 1]$ and $P_- \in E_A[\pi + 1]$, and set $E \leftarrow E_A$.
- Iterate through $i \in \{1, \ldots, n\}$ in ascending order, and attempt to compute $\phi : E \to \mathfrak{l}_i^{e_i} * E$ using $P_+$ resp $P_-$. Push both points through each $\phi$.
- In case of point rejections, sample fresh points and attempt to compute the corresponding isogenies, until all $\mathfrak{l}_i^{e_i}$ have been applied.

In order to speed up computations, SQALE additionally pushes intermediate points through isogenies, which saves computational effort in following steps [18]. However, the exact design of the computational strategy inside CSIDH is not relevant for the proposed attack. Using the above description, we sketch the adaptive attack on SQALE-2048 to recover the secret key bit by bit. In case of no point rejections, the order of steps in which $\mathfrak{a} * E_{\mathtt{PK}}$ is computed in SQALE is deterministic, and thus we can immediately apply Lemmas 1 and 2:

**Corollary 1.** *If no point rejections occur, the computation $\mathfrak{a} * E_{\mathtt{PK}}$ in SQALE is an ordered evaluation with*

$$\mathfrak{l}^{(n)} * \ldots * \mathfrak{l}^{(1)} * E_{\mathtt{PK}} = \mathfrak{l}_{221}^{e_{221}} * \ldots * \mathfrak{l}_1^{e_1} * E_{\mathtt{PK}}.$$

*Hence, $p_{\mathfrak{a}_k} = \prod_{i=1}^{k} \frac{\ell_i - 1}{\ell_i}$.*

SQALE uses coefficients in alternative Montgomery form $(A + 2C : 4C)$, so that passing over the curve $E_0$ can be detected as described in Section 3.1.

**Recovering the $k$-th bit.** Recovering the $k$-th bit of a SQALE secret key works exactly as described in Figure 1, as in a successful run SQALE performs each step $\mathfrak{l}_i^{\pm 1}$ in ascending order. Thus, the $k$-th step, in a run where the first $k$ steps succeed, computes $E \to \mathfrak{l}_k^{\pm 1} * E$. For the attack, we assume knowledge of the first $k-1$ bits of the secret to produce public keys $E_{\mathtt{PK}}$ resp. $\tilde{E}_{\mathtt{PK}}$ that lead the target through $E_0$ via an application of $\mathfrak{l}_k^{-1}$ resp. $\mathfrak{l}_k$, as given by Lemma 1. For one of these cases, with probability $p_{\mathfrak{a}_k}$ (Lemma 2), the target passes over $E_0$ on the $k$-th step, and we learn the $k$-th secret bit $e_k$ from side-channel information.

As $k$ increases, $p_{\mathfrak{a}_k}$ decreases: In order for the target to pass over $E_0$ in one of the two cases, *all* previous isogenies have to succeed, for which Corollary 1 gives the probability $p_{\mathfrak{a}_k}$. Thus, the fact that SQALE first computes small-degree

isogenies is slightly inconvenient for the attack, due to their low success probabilities. Nevertheless, attacking the last round of SQALE-2048 has a success probability of roughly $p_{\mathfrak{a}_{221}} = \prod_{j=1}^{221}(\ell_j - 1)/\ell_j \approx 19.3\%$, so that in about 1 in 5 runs, every isogeny succeeds and we pass over $E_0$ for the 221-th bit, compared to 2 in 3 runs to pass over $E_0$ for the first bit ($p_{\mathfrak{a}_1} = \frac{2}{3}$). This means that we need about three times as many measurements to discover the last bit, than the first bit. Nonetheless the required total number of measurements for all bits is very managable; we get with Lemma 2:

**Corollary 2.** *Assuming a pass over $E_0$ leaks the $k$-th bit when the representation is strongly correlated, the estimated number of measurements to recover a SQALE-2048 key is*

$$4 \cdot \sum_{k=1}^{221} \frac{1}{p_{\mathfrak{a}_k}} = 4 \cdot \sum_{k=1}^{221} \prod_{i=1}^{k} \frac{\ell_i}{\ell_i - 1} \approx 4 \cdot 1020.$$

Here, the factor 4 represents the fact that we need to feed both $E_{\mathtt{PK}}$ and $\tilde{E}_{\mathtt{PK}}$, and that only half the time ($2C : 4C$) is strongly-correlated. In practice, for more certainty, we increase the number of attempts per bit by some constant $\alpha$, giving a total of $\alpha \cdot 4 \cdot 1020$ expected attempts. We detail this in Section 6.

### 3.3 Recovering secret keys in CTIDH

CTIDH [4] is the most efficient constant-time implementation of CSIDH to date, although the work restricts to the CSIDH-512 and CSIDH-1024 parameter sets. We note that techniques from CTIDH can be used to significantly speed up CSIDH for larger parameters too, yet this appears to require some modifications that have not been explored in the literature yet. In this section, we explain how zero-value curve attacks can be mounted on CTIDH, leading to a partial or full key recovery, depending on the number of measurements that is deemed possible. For concreteness, we focus on the CTIDH parameter set with a 220-bit key space, dubbed CTIDH-511 in [4], which uses 15 batches of up to 8 primes. The bounds satisfy $M_i \leq 12$.

**Algorithmic description of CTIDH.** Given a starting curve $E_A$, CTIDH computes the group action by multiple rounds of the following approach:

- Set $E \leftarrow E_A$, sample random points $P_+ \in E[\pi - 1]$ and $P_- \in E[\pi + 1]$.
- Per batch $B_i$, (attempt to) compute $\phi : E \to \mathfrak{l}_{i,j}^{\mathtt{sign}(e_{i,j})} * E$ using $P_+$ resp $P_-$ (or dummy when all $\mathfrak{l}_{i,j}^{e_{i,j}}$ are performed). Push both points through each $\phi$.
- Repeat this process until all $\mathfrak{l}_{i,j}^{e_{i,j}}$ and dummy isogenies have been applied.

Furthermore, the following design choices in CTIDH are especially relevant:

- Per batch $B_i$, CTIDH computes real isogenies first, and (potential) dummy isogenies after, to ensure $M_i$ isogenies are computed, independent of $(e_{i,j})$.

– Per batch $B_i$, CTIDH computes the actual $\ell_{i,j}$-isogenies in ascending order.
– Per batch $B_i$, CTIDH scales the point rejection probability to the largest value, $1/\ell_{i,1}$. This slightly changes the computation of $p_{\mathfrak{a}_k}$.
– The order in which batches are processed is deterministic.

*Example 1.* Let $B_1 = \{3, 5\}$ with $M_1 = 6$, and let $e_{1,1} = 2$ and $e_{1,2} = -3$. For $B_1$, we first try to compute $E \to \mathfrak{l}_1 * E$, until this succeeds twice. Then, we try to compute $E \to \mathfrak{l}_2^{-1} * E$, until this succeeds three times. After the real isogenies, we try to compute the remaining $B_1$-dummy isogeny. All $B_1$-isogenies, including dummies, have success probability $2/3$. If all six of the $B_1$-isogenies are performed but other $B_i$ are unfinished, we skip $B_1$ in later rounds.

As for SQALE, the above description gives us that the order in which each $\mathfrak{l}$ is applied in CTIDH is deterministic, assuming that none of the steps fail, and so we get with Lemmas 1 and 2 again:

**Corollary 3.** *If no point rejections occur, the computation $\mathfrak{a} * E$ in CTIDH is an ordered evaluation $\mathfrak{l}^{(n)} * \ldots * \mathfrak{l}^{(1)} * E$, with $n = \sum M_i$, including dummy isogenies.*

Hence we can perform the adaptive attack on CTIDH-511 to recover the secret key bit by bit. The CTIDH implementation of [4] uses coefficients in alternative Montgomery form $(A + 2C : 4C)$, but passes over Montgomery form $(A : C)$ after each isogeny. Hence, $E_0$ always has a zero-value representation and we detect $E_0$ as described in Section 3.1. We argue in Section 5 that zero-value representations are easier to detect than strongly-correlated representations.

**Recovering the $k$-th bit.** CTIDH introduces several difficulties for the attack, compared to SQALE. In particular, let $B_i = \{\ell_{i,1}, \ldots, \ell_{i,n_i}\}$ be the batch to be processed at step $k$. Then, since usually $n_i > 1$, we do not get a binary decision at each step as depicted in Figure 1, but a choice between $2 \cdot n_i$ real isogeny steps $\mathfrak{l}_{i,j}^{\pm 1}$, or possibly a dummy isogeny. In practice, with high probability, we do not need to cover all $2 \cdot n_i + 1$ options, as the following example shows.

*Example 2.* As CTIDH progresses through the batch ascendingly from $\ell_{i,1}$ to $\ell_{i,n_i}$, the first step of a batch can often be recovered as in Figure 1, using public keys that are one $\ell_{i,1}$-isogeny away from $E_0$ respectively. If both do not pass over $E_0$, we deduce that $e_{i,1} = 0$, and we repeat this approach using an $\ell_{i,2}$-isogeny. In case of a successful attempt for $\ell_{i,j}$, we learn that the respective key element satisfies $e_{i,j} \leq -1$ resp. $e_{i,j} \geq 1$, depending on which of the binary steps was successful.[5] If we do not succeed in detecting $E_0$ after trying all $\ell_{i,j}^{\pm 1}$ in $B_i$, we learn that the target computes a $B_i$-dummy isogeny, and so all $e_{i,j} = 0$ for $\ell_{i,j} \in B_i$. We can easily confirm dummy isogenies: If the $k$-th step is a dummy isogeny, then using $E_{\mathsf{PK}}$ such that $\mathfrak{a} * E_{\mathsf{PK}}$ passes over $E_0$ in step $k - 1$, we do not move to a different curve in step $k$ and so we observe $E_0$ using side-channel information after steps $k - 1$ *and* $k$.

[5] Note that the $e_{i,j}$ are not limited to $\{-1, 1\}$ in CTIDH, in contrast to $e_i$ in SQALE.

This approach to recover the $k$-th bit in CTIDH-511 only differs slightly from Section 3.2: Given the knowledge of the secret path up to step $k-1$, we recover the $k$-th step by iterating through the target batch $B_i = \{\ell_{i,1}, \ldots, \ell_{i,n_i}\}$, until we detect $E_0$ for a given degree $\ell_{i,j}$, or otherwise assume a dummy isogeny. This iteration becomes easier in later rounds of each batch:

- If a previous round found that some $e_{i,j}$ is positive, we only have to check for positive $\ell_{i,j}$-isogeny steps later on (analogously for negative).
- If a previous round computed an $\ell_{i,j}$-isogeny, we immediately know that the current round cannot compute an $\ell_{i,h}$-isogeny with $h < j$.
- If a previous round detected a dummy isogeny for batch $B_i$, we can skip isogenies for $B_i$ in all later rounds, since only dummy isogenies follow.

Thus, knowledge of the previous isogeny path significantly shrinks the search space for later steps. As in SQALE, the probability $p_{\mathfrak{a}_k}$ decreases the further we get: Batches containing small degrees $\ell_i$ appear multiple times, and steps with small $\ell_i$ have the most impact on $p_{\mathfrak{a}_k}$. For the last step $\mathfrak{l}^{(n)}$, the probability that *all* steps $\mathfrak{l}^{(k)}$ in CTIDH-511 succeed without a single point rejection, is roughly 0.3%. This might seem low at first, but the number of measurements required to make up for this probability does not explode; we are able to recover the full key with a reasonable amount of measurements as shown in Section 6. Furthermore, this probability represents the absolute lower bound, which is essentially the worst-case scenario: It is the probability that for the worst possible key, with no dummy isogenies, all steps must succeed in one run. In reality, almost all keys contain dummy isogenies, and we can relax the requirement that none of the steps fail, as failing dummy isogenies do not impact the curves passed afterwards.

*Example 3.* Let $B_1 = \{3, 5\}$ with $M_1 = 6$ as in CTIDH-511. Say we want to detect some step in the eighth round of some $B_i$ for $i > 1$; it is not relevant in which of the seven former rounds the six $B_1$-isogenies are computed, and thus we can effectively allow for one point rejection in these rounds. This effect becomes more beneficial when dummy isogenies are involved. For example, if three of these six $B_1$-isogenies are dummies, we only need the three actual $B_1$-isogenies to be computed within the first seven rounds. Furthermore, after detecting the first dummy $B_1$-isogeny, we do not need to attack further $B_1$-isogenies as explained above, and therefore save significant attack effort.

*Remark 6.* The generic attack requires that all first $k$ steps succeed. This is not optimal: Assuming that some steps fail increases the probability of success of passing over $E_0$. For example, to attack isogenies in the sixth round and knowing that $e_{1,1} = 5$, it is better to assume that one or two out of these five fail and will be performed after the $\ell_{i,j}$-isogeny we want to detect, than it is to assume that all five of these succeed in the first five rounds. This improves the success probability of passing over $E_0$ per measurement, but makes the analysis of the required number of measurements harder to carry out. Furthermore, this optimal approach highly depends on the respective private key. We therefore do not pursue this approach in our simulations. A concrete practical attack against

a single private key that uses this improved strategy should require a smaller number of measurements.

*Remark 7.* For CTIDH with large parameters, one would expect more large $\ell_i$ and fewer isogenies of low degrees, relative to CTIDH-511. This improves the performance of the attack, as the probability of a full-torsion path increases, and so we expect more measurements to pass over $E_0$. However, the details of such an attack are highly dependent on the implementation of a large-parameter CTIDH scheme. As we know of no such implementation, we do not analyze such a hypothetical implementation in detail.

*Remark 8.* At a certain point, it might be useful to stop the attack, and compute the remaining key elements via a simple meet-in-the-middle search. Especially for later bits, if some dummy isogenies have been detected and most of the key elements $e_{i,j}$ are already known, performing a brute-force attack may be faster than this side-channel attack.

## 4 Recovering SIKE keys with side-channel leakage of $E_6$

We now apply the same strategy from Section 3 to SIKE. In this whole section, we focus on recovering Bob's static key $\mathtt{sk}_B$ by showing side-channel leakage in $\mathtt{Derive}_B$, used in `Decaps`. In general, the idea would apply as well to recover Alice's key $\mathtt{sk}_A$ in static SIDH or SIKE with swapped roles, as we do not use any specific structure of 3-isogenies. One can easily verify that the attack generalizes to SIDH based on $\ell_A$ or $\ell_B$-isogenies for *any* $\ell_A, \ell_B$. We repeat many of the general ideas from Section 3, with some small differences as SIKE operates in isogeny graphs over $\mathbb{F}_{p^2}$ instead of $\mathbb{F}_p$. Fortunately, these differences make the attack *easier*.

**Detecting $E_6$.** As remarked in Section 2, for both representations used in SIDH and SIKE, the curve $E_6$ is represented as $(8C : 4C)$, with $C = \alpha + \beta i \in \mathbb{F}_{p^2}$ non-zero. Similar to the CSIDH situation, whenever $4\alpha$ or $4\beta$ is smaller than $p/2$, doubling $4C$ does not require a modular reduction for these values, and hence the bit representation of $8\alpha$ resp. $8\beta$ of $8C$ is precisely a bit shift of $4\alpha$ resp. $4\beta$ of $4C$ by one bit to the left. Such strongly-correlated values can be observed in several ways using side-channel analysis, as we detail later in Section 5. Different from the CSIDH situation are the following key observations:

- The prime used in SIKE is of the form $p = 2^{e_A} \cdot 3^{e_B} - 1$. As observed in Remark 3, this large cofactor $2^{e_A}$ in $p+1$ implies a modular reduction does *not* affect the lowest $e_A - 1$ bits, except for the shift. Hence, even when $4\alpha$ or $4\beta$ is larger than $p/2$, we see strong correlation between their lowest bits.
- $C$ is now an $\mathbb{F}_{p^2}$ value, so we get strong correlation between $8\alpha$ and $4\alpha$ *and* between $8\beta$ and $4\beta$. This implies at least $2 \cdot (e_A - 1)$ strongly-correlated bits in the worst case (25 %), up to $2 \cdot (\log_2(p) - 1)$ strongly-correlated bits in the best case (25%).

For random curves $E_a$, the representations of $a$ are indistinguishable from random $(\alpha + \beta i : \gamma + \delta i)$, and so an attacker can differentiate $E_6$ from such curves. From this, an attacker only needs a few traces to determine accurately whether a walk passes over $E_6$ or not, as discussed in Section 5.

**General approach to recover the $k$-th trit.** Assuming we know the first $k-1$ trits $\mathtt{sk}_i$ of a secret key $\mathtt{sk}$, i.e. $\mathtt{sk}_{<k-1} = \sum_{i=0}^{k-2} \mathtt{sk}_i \cdot 3^i$, we want to find $\mathtt{sk}_{k-1} \in \{0,1,2\}$. We construct three candidate secret keys, $\mathtt{sk}^{(0)}, \mathtt{sk}^{(1)}, \mathtt{sk}^{(2)}$ as

$$\mathtt{sk}^{(0)} = \mathtt{sk}_{<k-1} + 0 \cdot 3^{k-1}, \quad \mathtt{sk}^{(1)} = \mathtt{sk}_{<k-1} + 1 \cdot 3^{k-1}, \quad \mathtt{sk}^{(2)} = \mathtt{sk}_{<k-1} + 2 \cdot 3^{k-1}.$$

We must have $\mathtt{sk}_{<k} = \mathtt{sk}^{(i)}$ for some $i \in \{0,1,2\}$. Thus, we use these three keys to construct (see Lemma 3) three public keys $\mathtt{pk}^{(0)}, \mathtt{pk}^{(1)}, \mathtt{pk}^{(2)}$ such that $\mathtt{Derive}_B(\mathtt{sk}^{(i)}, \mathtt{pk}^{(i)})$ computes $E_6$. When we feed these three keys to Bob, the computation $\mathtt{Derive}_B(\mathtt{sk}, \mathtt{pk}^{(i)})$ will then pass over $E_6$ in the $k$-th step *if and only if* $\mathtt{sk}_{k-1} = i$. By observing $E_6$ from side-channel information, we find $\mathtt{sk}_{k-1}$.

In this attack scenario, another key observation makes the attack on SIDH and SIKE easier than the attack on CSIDH: The computation $\mathtt{Derive}_B(\mathtt{sk}, \mathtt{pk}^{(i)})$ *always* passes over the same curves, as there are no "steps that can fail" as in CSIDH. We know *with certainty* that Bob will pass over $E_6$ in step $k$ in precisely one of these three computations. Hence, the number of traces required reduces drastically, as we do not need to worry about probabilities, such as $p_{\mathfrak{a}_k}$, that we have for CSIDH.

**Constructing $\mathtt{pk}^{(i)}$ from $\mathtt{sk}^{(i)}$ using backtracking.** Whereas in CSIDH it is trivial to compute a curve $E_{\mathtt{PK}}$ such that $\mathfrak{a} * E_{\mathtt{PK}}$ passes over $E_0$ in the $k$-th step (see Lemma 1), in SIDH and SIKE it is not immediatly clear how to construct $\mathtt{pk}^{(i)}$ for $\mathtt{sk}^{(i)}$. We follow [1, § 3.3], using *backtracking* to construct such a $\mathtt{pk}$.[6] The main idea is that any $\mathtt{sk}_{<k}$ corresponds to some *kernel point* $R_B$ of order $3^k$ for some $k$, so to an *isogeny* $\phi^{(k)} : E_6 \to E^{(k)}$. Here, the trits $\mathtt{sk}_i$ determine the steps

$$E_6 = E^{(0)} \xrightarrow{\mathtt{sk}_0} E^{(1)} \xrightarrow{\mathtt{sk}_1} \dots \xrightarrow{\mathtt{sk}_{k-1}} E^{(k)}.$$

The dual isogeny $\hat{\phi}^{(k)} : E^{(k)} \to E_6$ then corresponds to the kernel generator $\phi^{(k)}([3^{e_B-k}]Q_B)$ (see [37]). This leads to [1, Lemma 2].

**Lemma 3 ([1]).** *Let $\mathtt{sk}$ be a secret key, and let $R_k = [3^{e_B-k}](P_B + [\mathtt{sk}_{<k}]Q_B)$ so that $\phi : E_6 \to E^{(k)}$ is the corresponding isogeny for the first $k$ steps. Let $T \in E^{(k)}[3^{e_B}]$ such that $[3^{e_B-k}]T \neq \pm[3^{e_B-k}]\phi(Q_B)$. Then*

$$\mathtt{pk}' = (\phi(Q_B) + [\mathtt{sk}_{<k}]T,\ -T,\ \phi(Q_B) + [\mathtt{sk}_{<k} - 1]T)$$

*is such that $\mathtt{Derive}_B(\mathtt{sk}, \mathtt{pk}')$ passes over $E_6$ in the $k$-th step.*

[6] It is important that such a $\mathtt{pk} = (P, Q, Q-P)$ passes the *CLN test* [20]: $P$ and $Q$ are both of order $3^{e_B}$ and $[3^{e_B-1}]P \neq [\pm 3^{e_B-1}]Q$, so that they generate $E[3^{e_B}]$.

It is necessary that such a $\mathtt{pk}'$ is not rejected by a SIKE implementation.

**Corollary 4 ([1]).** *The points $P'$ and $Q'$ for a $\mathtt{pk}' = (P', Q', Q' - P')$ as constructed in Lemma 3 form a basis for the $3^{e_B}$-torsion of $E^{(k)}$. This implies they are of order $3^{e_B}$ and pass the CLN test.*

Given Lemma 3 and $\mathtt{sk}_{<k-1}$, we can therefore easily compute the $\mathtt{pk}^{(i)}$ corresponding to $\mathtt{sk}^{(i)}$ for $i \in \{0, 1, 2\}$. One of the three attempts $\mathtt{Derive}_B(\mathtt{sk}, \mathtt{pk}^{(i)})$ will then pass over $E_6$ in the $k$-th step, while the other two will not. Only the representation of $E_6$ by $(8C : 4C)$ is then strongly-correlated, and by detecting this representation using side-channel information, we recover $\mathtt{sk}_{k-1}$.

*Remark 9.* A straightforward attack computes $\mathtt{pk}^{(0)}, \mathtt{pk}^{(1)}$ and $\mathtt{pk}^{(2)}$, and feeds all three to Bob, and so requires 3 traces to recover a single trit $\mathtt{sk}_{k-1}$. Clearly, when we already detect $E_6$ in the trace of $\mathtt{Derive}_B(\mathtt{sk}, \mathtt{pk}^{(0)})$, we do not need the traces of $\mathtt{pk}^{(1)}$ and $\mathtt{pk}^{(2)}$, similarly for $\mathtt{Derive}_B(\mathtt{sk}, \mathtt{pk}^{(1)})$. This approach would require on average $\frac{1}{3} \cdot 1 + \frac{1}{3} \cdot 2 + \frac{1}{3} \cdot 3 = 2$ traces per trit. We can do even better: If we do not detect $E_6$ in both $\mathtt{Derive}_B(\mathtt{sk}, \mathtt{pk}^{(0)})$ and $\mathtt{Derive}_B(\mathtt{sk}, \mathtt{pk}^{(1)})$, we do not need a sample for $\mathtt{Derive}_B(\mathtt{sk}, \mathtt{pk}^{(2)})$, as $\mathtt{sk}_{k-1}$ *must* equal 2. This gives $\frac{5}{3}$ samples per trit, giving a total of $\frac{5}{3} \cdot e_B$ traces.

## 5 Feasibility of obtaining the side-channel information

In this section, we discuss the practical feasibility of obtaining the required side-channel information.

**Zero-value representations.** For zero-value representations as in CTIDH, where $E_0$ is represented by $(0 : C)$ in Montgomery form, we exploit side-channel analysis methods to distinguish between the zero curve and others. In particular, as shown in [23], one can apply Welch's t-test [41] to extract the required information from the power consumption of the attacked device. Further, as mentioned in [23], one can use correlation-collision SCA methods to identify zero values using multiple measurements. Therefore, the attack scheme as demonstrated in [23] to SIKE can analogously be applied whenever zero-value representations occur.

**Strongly-correlated representations.** The attacks presented in Sections 3.2 and 4 for implementations using strongly-correlated representations, such as SQALE and SIKE, are more challenging in practice, since no zero values occur. A naïve approach to mount the proposed attack for such instances would be to apply side-channel attacks like CPA or DPA, and estimate or guess the values of intermediate codomain curves. Revealing those intermediate values would require a fitting power model and a sufficiently high signal-to-noise ratio (SNR[7]). By exploiting the pattern similarity in the strongly-correlated representation $(2C : 4C)$ for SQALE or $(8C : 4C)$ for SIKE, as mentioned in Section 3.1

[7] SNR is the ratio between the variance of the signal and the variance of noise. Too small SNR values make information and noise indistinguishable.

and Section 4, we reduce the SNR required to successfully perform the attack. To achieve this, we apply the concept of correlation-collision attacks, so that there is no need to reveal the actual value of $C$ via a sophisticated power model.

We exploit side-channel correlation-collision attacks [36] to find similar values by searching for strongly-correlated patterns versus non-correlated patterns. Instead of measuring *multiple* computations to identify similar or identical patterns, as in [36], we apply the concept of a horizontal side-channel attack as in [38]. That is, we extract the required side-channel information from a *single* segmented power trace. Such a segmented power trace contains the power values of the processed limbs (each limb is 64 bits), required to represent $\mathbb{F}_p$-values, which form a fingerprint characteristic of such a value. These fingerprints then serve as input to calculate the correlation between $2C$ and $4C$ for SQALE, or $4\alpha, 4\beta, 8\alpha$ and $8\beta$ for SIKE, from which we judge their similarity. For strongly-correlated representations of $E_0$ and $E_6$, this gives a higher correlation between the fingerprints than for representations of random curves $E_a$ as either $(A + 2C : 4C)$ or $(A + 2C : A - 2C)$, with $A, C \neq 0$.

For both CSIDH attacks, we assume no point rejections prior to the respective isogeny computation, so that the specific isogeny steps are known in advance. For SIKE, there are no such probabilities involved in the isogeny computation, and so here too the specific isogeny steps are known in advance. This implies that an attacker will know where the values of interest are computed and used within the power trace, and can distinguish the relevant information from the rest of the trace. Thus, in all cases, the points of interest (position of the limbs) within the power trace are known in advance, and segmenting each power trace into vectors of the corresponding processed limbs for mounting the correlation-collision attack is easy.

## 6 Simulating the attacks on SQALE, CTIDH and SIKE

To demonstrate the proposed attacks, we implemented Python (version 3.8.10) simulations for our CTIDH-511[8] and SQALE-2048[9] attacks, and a C simulation of the attack on SIKE.[10] The C code for key generation and collecting the simulated power consumption were compiled with gcc (version 9.4.0). Security-critical spots of the attacked C code remained unchanged in both cases.

For the SQALE and CTIDH attacks, the implemented simulation works as follows: First, we generate the corresponding public keys $E_{\mathtt{PK}}$ and $\tilde{E}_{\mathtt{PK}}$ for the current $k$-th step, as described in Section 3.1. Then we collect the bit values of the resulting codomain curve after the computation of the $k$-th step $E \leftarrow \mathfrak{l}^{(k)} * E$ in the group action $\mathfrak{a} * E_{\mathtt{PK}}$ resp. $\mathfrak{a} * \tilde{E}_{\mathtt{PK}}$ to simulate the power consumption.

We calculate the Hamming weight of these values and add a zero-mean Gaussian standard distribution to simulate noise in the measurement. We picked different values of the standard deviation to mimic realistic power measurements

[8] http://ctidh.isogeny.org/high-ctidh-20210523.tar.gz

[9] https://github.com/JJChiDguez/sqale-csidh-velusqrt, commit `a95812f`

[10] https://github.com/Microsoft/PQCrypto-SIDH, commit `ecf93e9`

with different SNR values. By varying the SNR in such a way, we can determine up to which SNR the attacks are successful, and compare this to known SNR values achieved in physical attacks. For SQALE and CTIDH, we are only interested in power traces passing over $E_0$, and so we need the first $k$ steps to succeed. We therefore take enough samples to ensure high probability that passing over $E_0$ happens multiple times for either $E_{\mathtt{PK}}$ or $\tilde{E_{\mathtt{PK}}}$. Finally, based on the set of collected bit vectors for all these samples, we decide on which of the two cases contains paths over $E_0$, and therefore reveal the $k$-th bit of the secret key.

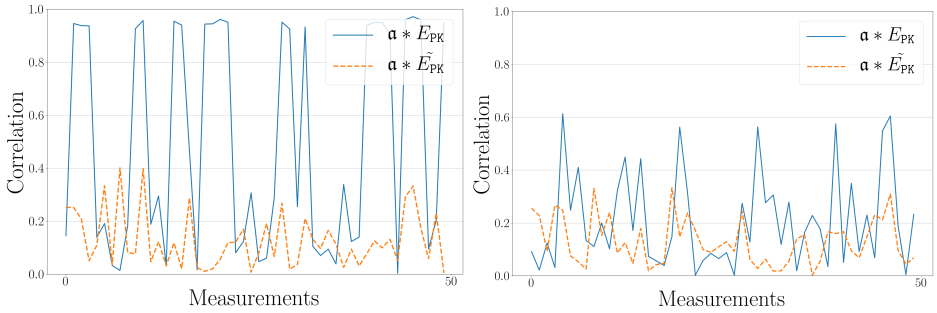
For the SIKE attack, we generate $\mathtt{pk}^{(0)}, \mathtt{pk}^{(1)}$ and $\mathtt{pk}^{(2)}$ for the current $k$-th step, as described in Section 4, and collect the bit values of the resulting codomain curve in the computation of the $k$-th step of $\mathtt{Derive}_B$ in `Decaps`. For simplicity, there is no noise in the simulation, as the results are exactly the same as for the SQALE situation after extracting the bit values. Deciding which sample has strongly-correlated values is easy, as is clear from Figure 3.

As described in previous sections, due to the different representations, the decision step differs between CTIDH and SQALE. For SIKE, the probability to pass over $E_6$ is 100%, and so a single sample per $\mathtt{pk}^{(i)}$ is enough to decide what the $k$-th trit $\mathtt{sk}_{k-1}$ is.

In order to reduce the running time of our simulations for SQALE and CTIDH, we terminate each group action run after returning the required bit values of the $k$-th step. Furthermore, we implemented a threaded version so that we collect several runs in parallel, which speeds up the simulation. All experiments were measured on AMD EPYC 7643 CPU cores.

**Attacking CTIDH-511.** As shown in [23, § 4] a practical differentiation between zero and non-zero values, even with low SNR, is feasible with a single trace containing the zero value. Hence, in CTIDH, where $E_0$ is represented by $(0 : C)$, a single occurrence of $E_0$ leaks enough information for the decision in each step. Thus, the number of required attempts can be calculated as follows: Given $p_{\mathfrak{a}_k}$ from Lemma 2, the success probability of having *at least one* sequence that passes over $E_0$ in the $k$-th step in $t_k$ attempts is $P_{exp}(X \geq 1) = 1 - (1 - p_{\mathfrak{a}_k})^{t_k}$. We can calculate $t_k$ to achieve an expected success rate $P_{exp}$ by $t_k = \log_{(1-p_{\mathfrak{a}_k})}(1 - P_{exp})$. For CTIDH-511, to achieve $P_{exp} \geq 99\%$ for all $k$, we get an estimate of $\sum t_k \approx 130,000$ attempts for full key recovery. In simulations, the required number of attempts for full key recovery was $\approx 85,000$ on average over 100 experiments, due to effects mentioned in Section 3.3. The average execution time was about 35 minutes (single core) or 5 minutes (120 threads). As described in Remark 8, finding the last few key bits by brute force drastically reduces the required measurements, as $p_{\mathfrak{a}_k}$ is low.

**Attacking SQALE-2048.** In this case, we simulate a correlation-collision attack as described in Section 5: We calculate the correlation between the 64-bit limbs that represent the $\mathbb{F}_p$-values, and apply the standard Hamming-weight model with noise drawn from a normal distribution. Even with an SNR as low as 1.40, strongly-correlated representations leak enough information to guess




(a) Correlation results without noise. (b) Correlation results with SNR of 1.40.

Fig. 3: Experimental results to discover bit $k = 1$: the correct hypothesis ($\mathfrak{a} * E_{\mathtt{PK}}$) in blue and the wrong hypothesis ($\mathfrak{a} * \tilde{E_{\mathtt{PK}}}$) in orange, for SQALE-2048.

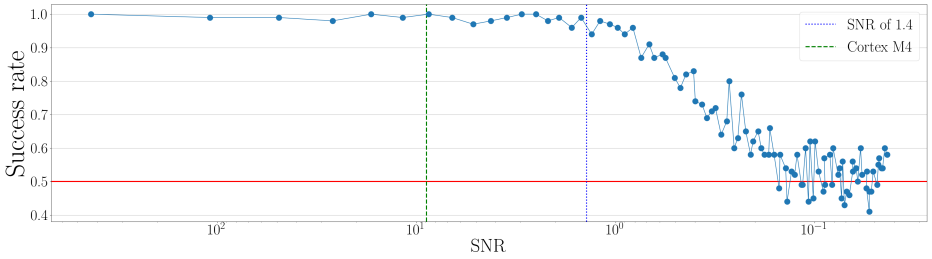



Fig. 4: Relation between SNR and success rate. Rate of 0.5 equals random guess.

the $k$-th bit, as can be seen in Figure 3 for $k = 1$. Both without noise and with SNR 1.40, we are able to determine the right bit in 74% of measurements (where 75% is the theoretical optimum, as $2C \leq \frac{p}{2}$ only half the time). An SNR value of 1.40 is considered *low*: The SNR value of a common embedded device, using a measurement script[11] provided by the ChipWhisperer framework for a ChipWhisperer-Lite board with an ARM Cortex-M4 target, obtains an SNR of 8.90. Figure 4 shows the success rate for different values. We evaluated the following methods for decision-making:

- Decide based on the number of cases with a higher resulting correlation, as exemplified in Figure 3.
- Decide based on the sum of the resulting correlations for each case, to reduce the number of attempts required for a given success rate.

Empirical results show that the sum-based approach reduces the required number of attempts for key recovery by a factor 3 on average (from $\approx 24,819$ to $\approx 8,273$), which leads to an average execution time of 35 minutes (120 threads).

[11] https://github.com/newaetech/chipwhisperer-jupyter/blob/master/archive/PA_Intro_3-Measuring_SNR_of_Target.ipynb, commit 44112f6

**Attacking SIKE.** For SIKE, the analysis after collecting the bit values is similar to that of the SQALE case, and hence the results from Figure 3 apply to these simulated sampels too. Furthermore, for SIKE we have the advantage that **i)** we know that one of the three samples per trit must be an $E_6$-sample, **ii)** we know that even with modular reduction, there is strong correlation between the lowest limbs and **iii)** we can use both $\mathbb{F}_p$-values $\alpha$ and $\beta$ for $C = \alpha + \beta i \in \mathbb{F}_{p^2}$.

As explained in Section 4, we need on average 5/3 samples per trit to find $\mathtt{sk}_i$, for all $e_B$ trits. For SIKEp434, this gives an average of 228 samples to recover $\mathtt{sk}_B$. The average running time over 100 evaluations in each case was $\approx 4$ seconds for SIKEp434, $\approx 8$ seconds for SIKEp503, $\approx 17$ seconds for SIKEp610, and $\approx 42$ seconds for SIKEp751 respectively.

| **Scheme** | SQALE-2048 | CTIDH-511 | SIKEp434 | SIKEp503 | SIKEp610 | SIKEp751 |
|---|---|---|---|---|---|---|
| **Samples** | 8,273 | 85,000 | 228 | 265 | 320 | 398 |

Table 1: Required number of samples to reconstruct secret key in simulations.

## 7 Countermeasures and conclusion

We have shown that both CSIDH and SIKE are vulnerable to leakage of specific curves. For CSIDH, we have shown that both Montgomery form and alternative Montgomery form leak secret information when passing over $E_0$, and for SIKE we have shown that in both forms, the representation of $E_6$ by $(8C : 4C)$ leaks secret information. As described in Section 5, zero-value representations are easiest to detect, and accordingly one should prefer the alternative Montgomery form over the Montgomery form throughout the whole computation for CSIDH variants. However, more effective countermeasures are required to avoid strongly-correlated representations in CSIDH and SIKE.

### 7.1 Public key validation

As mentioned in Section 3, public keys in the proposed attacks on CSIDH variants consist of valid supersingular elliptic curves. Hence, the attack cannot be prevented by public key validation.

For the SIKE attack, the situation is different: Instead of containing valid points $(\phi_A(P_B), \phi_A(Q_B), \phi_A(Q_B - P_B))$ (see Section 2), we construct public keys differently, as described in Lemma 3. However, such public key points are not detected by partial validation methods contained in the current SIKE software, such as the CLN test (see Corollary 4). In general, the full validation of SIDH or SIKE public keys is believed to be as hard as breaking the schemes themselves [24]. It remains an open question if there is an efficient partial validation method to detect the specific public key points generated by our attack.

### 7.2 Avoiding $E_0$ or $E_6$

A straightforward way of mitigating the attacks is to avoid paths that lead over $E_0$ or $E_6$, or any other vulnerable curve. As argued in [1, 23], avoiding them altogether seems difficult. We discuss techniques to achieve this.

**Danger zone.** Similar to the rejection proposal from [1], it may appear intuitive to define a certain *danger zone* around vulnerable curves, e.g. for CSIDH, containing all curves $\mathfrak{l}_i^{\pm 1} * E_0$ for $1 \leq i \leq n$, and abort the execution of the protocol whenever an isogeny path enters this zone. In the SIKE attack, this zone could include the four curves that are 3-isogenous to $E_6$. However, the attacker can simply construct public keys that would or would not pass through this zone, and observe that the protocol aborts or proceeds. This leaks the same information as in the attack targeting only $E_0$ or $E_6$.[12]

**Masking on isogeny level.** One can fully bypass this danger zone by masking by a (small) isogeny before applying secret isogeny walks (see [31, 1]). For CSIDH, for a masking isogeny $\mathfrak{z}$ and a secret $\mathfrak{a}$ we have that by commutativity, $\mathfrak{a} * E = \mathfrak{z}^{-1} * (\mathfrak{a} * (\mathfrak{z} * E))$, so this route avoids the danger zone when $\mathfrak{z}$ is sufficiently large. Drawing $\mathfrak{z}$ from a masking key space of $k$ bits would require the attacker to guess the random ephemeral mask correctly in order to get a successful walk over $E_0$, which happens with probability $2^{-k}$. Thus, a $k$-bit mask increases the number of samples needed by $2^k$. Similarly, as detailed in [1], the secret isogeny in the SIKE attack can be masked by a $2^k$-isogeny, where keeping track of the dual requires some extra cost. Although masking comes at a significant cost if the masking isogeny needs to be large, this appears to be the only known effective countermeasure that fully avoids the proposed attacks.

**Randomization of order (CSIDH).** For CSIDH variants, intuitively, randomizing the order of isogenies, and as proposed in [32] the order of real and dummy isogenies, might seem beneficial to achieve this. However, we can then simply *always* attack the first step of the isogeny path, with a success probability of $1/n$. With enough repetitions, we can therefore statistically guess the secret key, where the exact success probabilities highly depend on the respective CSIDH variant. This countermeasure also significantly impacts performance, making it undesirable.

**Working on the surface (CSIDH).** [13] An interesting approach to avoid vulnerable curves, specific to CSIDH, is to move to the *surface* of the isogeny graph. That is, we use curves $E_A$ with $\mathbb{F}_p$-rational endomorphism ring $\mathbb{Z}[\frac{1+\pi}{2}]$ instead of $\mathbb{Z}[\pi]$, and use a prime $p = 7 \mod 8$. This idea was proposed in [12] and dubbed CSURF. We can still work with elliptic curves in Montgomery form,

[12] Pun aficionados may wish to dub this scenario the *highway to the danger zone*.

[13] We thank the anonymous reviewers of SAC 2022 for this suggestion.

although Montgomery coefficients $a$ are not unique in this setting. However, when following the setup described in [11], we are not aware of any vulnerable curves on the surface, but it seems difficult to prove that vulnerable curves do not exist there. More analysis is necessary to rule out such curves. Nevertheless, working on the surface offers other benefits, and we see no reason to work on the floor with known vulnerabilities, instead of on the surface.

**Precomposition in SIKE.** A potential countermeasure specific to SIKE is precomposing with a random isomorphism, as proposed in [31]. In our attack scenario, the isogeny walk then passes a curve isomorphic to $E_6$ instead of $E_6$, which may eliminate the leakage. However, as discussed in [21], each isomorphism class contains exactly six Montgomery curves, and the isomorphism class of $E_6$ also contains $E_{-6}$, which shares the same vulnerability as $E_6$. Thus, in $1/3$ of cases, leakage still occurs, only moderately increasing the number of required measurements. On the other hand, finding an isomorphism that guarantees the isogeny walk not to pass $E_6$ or $E_{-6}$ only from public key information seems infeasible. Furthermore, the computation of isomorphisms usually contains expensive square root computations.

### 7.3 Avoiding correlations

Another approach to mitigate the attacks is to ensure that vulnerable curves such as $E_0$ and $E_6$ do not leak information when passing over them. This requires adapting the representations of such curves.

**Avoiding correlations in alternative Montgomery form.** As noted for CSIDH variants, the representation $(2C : 4C)$ leaks secret information whenever $2C < \frac{p}{2}$. In order to avoid this, we can try to represent the alternative Montgomery form $(A+2C : 4C)$ differently and use a *flipped* alternative Montgomery form $(A + 2C : -4C)$ instead, which we write as ɐlʇɐɹuɐʇıʌɐ Montgomery form for brevity. In the case of $E_0$, this means that the coefficients $2C$ and $-4C$ are *not* simple shifts of each other for $2C < \frac{p}{2}$, which prevents the correlation attack. In order to still achieve constant-time behavior, we should flip $4C$ for all curves, since otherwise $E_0$ would easily be detectable via side channels. The correctness of computations can be guaranteed by corresponding sign flips in computations that would normally include $4C$. Analogously, we can define a flipped representation of curves in SIKE. Although the ɐlʇɐɹuɐʇıʌɐ Montgomery form is effective in preventing leakage of $E_0$, it creates other vulnerable curves. We discuss this in more detail in Appendix A. It remains an open question to find a representation without both zero-value representations and strongly-correlated representations.

**Masking a single value.** Assuming we are working with the representations $(A_{24}^+ : A_{24}^-)$ or $(A_{24}^+ : C_{24})$ for either CSIDH or SIKE, masking is non-trivial, as it needs to respect the ratio $A/C$ during the computation. However, it is possible

to multiply by some random $\alpha$ during the computation of $A_{24}^+$, and to multiply by $1/\alpha$ in the next computations that use $A_{24}^+$. This requires a careful analysis and implementation, in order to guarantee that no leak of $A_{24}^+$ or some different correlation occurs at a given point in the computation.

## References

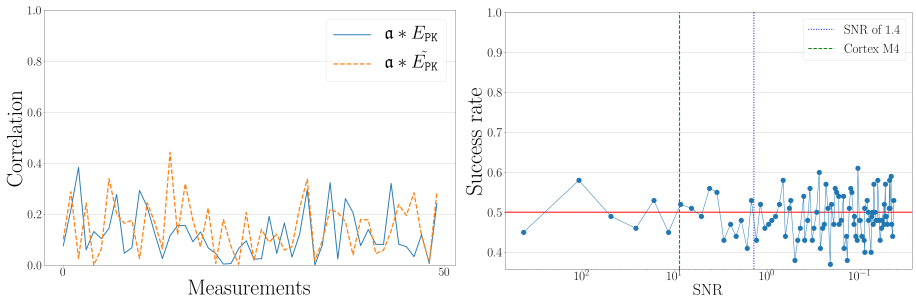

1. Adj, G., Chi-Domínguez, J.J., Mateu, V., Rodríguez-Henríquez, F.: Faulty isogenies: a new kind of leakage. arXiv preprint arXiv:2202.04896 (2022)
2. Adj, G., Chi-Domínguez, J., Rodríguez-Henríquez, F.: Karatsuba-based square-root Vélu's formulas applied to two isogeny-based protocols. Cryptology ePrint Archive, Paper 2020/1109 (2020)
3. Akishita, T., Takagi, T.: Zero-value point attacks on elliptic curve cryptosystem. In: Boyd, C., Mao, W. (eds.) ISC 2003. LNCS, vol. 2851, pp. 218–233. Springer (2003)
4. Banegas, G., Bernstein, D.J., Campos, F., Chou, T., Lange, T., Meyer, M., Smith, B., Sotáková, J.: CTIDH: faster constant-time CSIDH. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2021**(4), 351–387 (2021)
5. Bauer, A., Jaulmes, E., Prouff, E., Reinhard, J.R., Wild, J.: Horizontal collision correlation attack on elliptic curves. Cryptology ePrint Archive, Report 2019/321 (2019)
6. Bernstein, D.J., De Feo, L., Leroux, A., Smith, B.: Faster computation of isogenies of large prime degree. In: Galbraith, S.D. (ed.) ANTS 2020. pp. 39–55. Mathematics Sciences Publishers (2020)
7. Bernstein, D.J., Lange, T., Martindale, C., Panny, L.: Quantum circuits for the CSIDH: optimizing quantum evaluation of isogenies. In: Ishai, Y., Rijmen, V. (eds.) Advances in Cryptology - EUROCRYPT 2019. LNCS, vol. 11477, pp. 409–441. Springer (2019)
8. Bonnetain, X., Schrottenloher, A.: Quantum security analysis of CSIDH. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology - EUROCRYPT 2020. LNCS, vol. 12106, pp. 493–522. Springer (2020)
9. Campos, F., Kannwischer, M.J., Meyer, M., Onuki, H., Stöttinger, M.: Trouble at the CSIDH: protecting CSIDH with dummy-operations against fault injection attacks. In: FDTC 2020. pp. 57–65. IEEE (2020)
10. Campos, F., Krämer, J., Müller, M.: Safe-error attacks on SIKE and CSIDH. In: Batina, L., Picek, S., Mondal, M. (eds.) SPACE 2021. LNCS, vol. 13162, pp. 104–125. Springer (2021)
11. Castryck, W.: CSIDH on the surface (CSURF). Isogeny School 2020 (2021), https://homes.esat.kuleuven.be/~wcastryc/summer_school_csurf.pdf
12. Castryck, W., Decru, T.: CSIDH on the surface. In: International Conference on Post-Quantum Cryptography. pp. 111–129. Springer (2020)
13. Castryck, W., Decru, T., Vercauteren, F.: Radical isogenies. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 493–519. Springer (2020)
14. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: an efficient post-quantum commutative group action. In: Peyrin, T., Galbraith, S.D. (eds.) Advances in Cryptology - ASIACRYPT 2018. LNCS, vol. 11274, pp. 395–427. Springer (2018)

15. Cervantes-Vázquez, D., Chenu, M., Chi-Domínguez, J., De Feo, L., Rodríguez-Henríquez, F., Smith, B.: Stronger and faster side-channel protections for CSIDH. In: Schwabe, P., Thériault, N. (eds.) LATINCRYPT 2019. LNCS, vol. 11774, pp. 173–193. Springer (2019)
16. Chávez-Saab, J., Chi-Domínguez, J., Jaques, S., Rodríguez-Henríquez, F.: The SQALE of CSIDH: square-root Vélu quantum-resistant isogeny action with low exponents. Cryptology ePrint Archive, Paper 2020/1520 (2020)
17. Chi-Domínguez, J.J., Reijnders, K.: Fully projective radical isogenies in constant-time. In: Cryptographers' Track at the RSA Conference. pp. 73–95. Springer (2022)
18. Chi-Domínguez, J.J., Rodríguez-Henríquez, F.: Optimal strategies for CSIDH. Cryptology ePrint Archive, Paper 2020/417 (2020)
19. Costello, C.: The case for SIKE: A decade of the supersingular isogeny problem. Cryptology ePrint Archive, Paper 2021/543 (2021)
20. Costello, C., Longa, P., Naehrig, M.: Efficient algorithms for supersingular isogeny Diffie-Hellman. In: CRYPTO 2016, Part I. pp. 572–601. Springer (2016)
21. Costello, C., Longa, P., Naehrig, M., Renes, J., Virdia, F.: Improved classical cryptanalysis of SIKE in practice. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) Public-Key Cryptography - PKC 2020. LNCS, vol. 12111, pp. 505–534. Springer (2020)
22. De Feo, L.: Mathematics of isogeny based cryptography. CoRR **abs/1711.04062** (2017), http://arxiv.org/abs/1711.04062
23. De Feo, L., El Mrabet, N., Genêt, A., Kaluđerović, N., de Guertechin, N.L., Pontié, S., Tasso, É.: SIKE Channels. Cryptology ePrint Archive, Paper 2022/054 (2022)
24. Galbraith, S.D., Vercauteren, F.: Computational problems in supersingular elliptic curve isogenies. Quantum Inf. Process. **17**(10), 265 (2018)
25. Gélin, A., Wesolowski, B.: Loop-abort faults on supersingular isogeny cryptosystems. In: Lange, T., Takagi, T. (eds.) PQCrypto 2017. LNCS, vol. 10346, pp. 93–106. Springer (2017)
26. Genêt, A., de Guertechin, N.L., Kaluđerović, N.: Full key recovery side-channel attack against ephemeral SIKE on the Cortex-M4. In: Bhasin, S., Santis, F.D. (eds.) COSADE 2021. LNCS, vol. 12910, pp. 228–254. Springer (2021)
27. Goubin, L.: A refined power-analysis attack on elliptic curve cryptosystems. In: Desmedt, Y. (ed.) Public Key Cryptography - PKC 2003. LNCS, vol. 2567, pp. 199–210. Springer (2003)
28. Izu, T., Takagi, T.: Exceptional procedure attack on elliptic curve cryptosystems. In: Desmedt, Y. (ed.) Public Key Cryptography - PKC 2003. LNCS, vol. 2567, pp. 224–239. Springer (2003)
29. Jao, D., Azarderakhsh, R., Campagna, M., Costello, C., De Feo, L., Hess, B., Jalali, A., Koziel, B., LaMacchia, B., Longa, P., Naehrig, M., Renes, J., Soukharev, V., Urbanik, D., Pereira, G.: SIKE–Supersingular Isogeny Key Encapsulation (2017), https://sike.org/
30. Jao, D., De Feo, L.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In: Yang, B. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 19–34. Springer (2011)
31. Koziel, B., Azarderakhsh, R., Jao, D.: Side-channel attacks on quantum-resistant supersingular isogeny diffie-hellman. In: Adams, C., Camenisch, J. (eds.) SAC 2017. pp. 64–81. Springer (2017)
32. LeGrow, J.T., Hutchinson, A.: (short paper) analysis of a strong fault attack on static/ephemeral CSIDH. In: Nakanishi, T., Nojima, R. (eds.) IWSEC 2021. LNCS, vol. 12835, pp. 216–226. Springer (2021)

33. Meyer, M., Campos, F., Reith, S.: On Lions and Elligators: An efficient constant-time implementation of CSIDH. In: Ding, J., Steinwandt, R. (eds.) PQCrypto 2019. LNCS, vol. 11505, pp. 307–325. Springer (2019)
34. Meyer, M., Reith, S.: A faster way to the CSIDH. In: Chakraborty, D., Iwata, T. (eds.) Progress in Cryptology - INDOCRYPT 2018. LNCS, vol. 11356, pp. 137–152. Springer (2018)
35. Moody, D., Shumow, D.: Analogues of Vélu's formulas for isogenies on alternate models of elliptic curves. Mathematics of Computation **85**(300), 1929–1951 (2016)
36. Moradi, A., Mischke, O., Eisenbarth, T.: Correlation-enhanced power analysis collision attack. Cryptology ePrint Archive, Paper 2010/297 (2010)
37. Naehrig, M., Renes, J.: Dual isogenies and their application to public-key compression for isogeny-based cryptography. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology - ASIACRYPT 2019. LNCS, vol. 11922, pp. 243–272. Springer (2019)
38. Nascimento, E., Chmielewski, L.: Horizontal clustering side-channel attacks on embedded ecc implementations (extended version). Cryptology ePrint Archive, Paper 2017/1204 (2017)
39. Onuki, H., Aikawa, Y., Yamazaki, T., Takagi, T.: (Short paper) A faster constant-time algorithm of CSIDH keeping two points. In: Attrapadung, N., Yagi, T. (eds.) IWSEC 2019. LNCS, vol. 11689, pp. 23–33. Springer (2019)
40. Peikert, C.: He Gives C-Sieves on the CSIDH. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology - EUROCRYPT 2020. LNCS, vol. 12106, pp. 463–492. Springer (2020)
41. Schneider, T., Moradi, A.: Leakage assessment methodology - A clear roadmap for side-channel evaluations. In: Güneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 495–513. Springer (2015)
42. Tasso, É., De Feo, L., Mrabet, N.E., Pontié, S.: Resistance of isogeny-based cryptographic implementations to a fault attack. In: Bhasin, S., Santis, F.D. (eds.) COSADE 2021. LNCS, vol. 12910, pp. 255–276. Springer (2021)
43. Ti, Y.B.: Fault attack on supersingular isogeny cryptosystems. In: Lange, T., Takagi, T. (eds.) PQCrypto 2017. LNCS, vol. 10346, pp. 107–122. Springer (2017)
44. Vélu, J.: Isogénies entre courbes elliptiques. Comptes Rendus de l'Académie des Sciences de Paris, Séries A **273**, 238–241 (1971)
45. Wang, Y., Paccagnella, R., He, E.T., Shacham, H., Fletcher, C.W., Kohlbrenner, D.: Hertzbleed: Turning power side-channel attacks into remote timing attacks on x86 (2022)
46. Zhang, F., Yang, B., Dong, X., Guilley, S., Liu, Z., He, W., Zhang, F., Ren, K.: Side-channel analysis and countermeasure design on ARM-based quantum-resistant SIKE. IEEE Trans. Computers **69**(11), 1681–1693 (2020)

## A Flipping $4C$ as a countermeasure.

In this appendix, we discuss the effectiveness of ɐlʇɐɹuɐʇıʌǝ Montgomery form as a countermeasure. As Figure 5 shows, the countermeasure prevents detection of $(2C : 4C)$, and therefore prevents leakage on $E_0$. Similar techniques can also be applied for other strongly-correlated representations, such as those for $E_6$.




(a) Correlation results with countermeasures (without noise).

(b) SNR/Success rate with countermeasures.

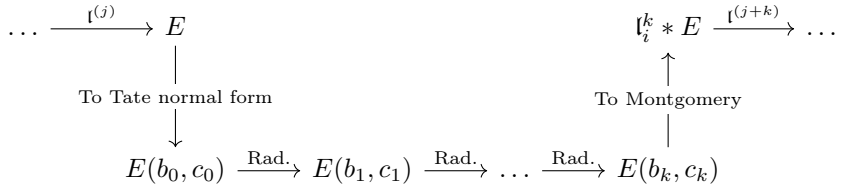
Fig. 5: Correlation values including the countermeasures leak no information.

Nevertheless, ɐlʇɐɹuɐʇıʌǝ Montgomery form creates new problems. In alternative Montgomery form, the curve $E_{-6}$, a valid supersingular curve for most CSIDH-primes, is represented using $(-4C : 4C)$, which is not strongly-correlated. However, by switching to ɐlʇɐɹuɐʇıʌǝ Montgomery form, we get the strongly-correlated representation $(-4C : -4C)$. The attack as described in the paper is then applicable by replacing $E_0$ by $E_{-6}$.

It seems difficult to discover if a curve $E_a$ will leak information before computing the next step: doing so would require knowledge on $a$, and so requires a representation of $a$ in some form. Hence, we cannot decide to use either alternative Montgomery form or ɐlʇɐɹuɐʇıʌǝ Montgomery form before we compute the actual curve.

## B CSIDH implementations using radical isogenies

In [13] an alternative method to compute the action of $\mathfrak{a}$ is proposed, using *radical isogenies*. The evaluation of $\mathfrak{a} * E$ is still an ordered evaluation $\mathfrak{l}_i^{(n)} * \ldots * \mathfrak{l}_i^{(1)} * E$, due to a change in the evaluation algorithm we get chains $\mathfrak{l}_i * \ldots * \mathfrak{l}_i$ of specific degrees $\ell_i \in \{4, 5, 7, 9, 11, 13\}$ in the evaluation. These chains are computed on a different curve form, namely the Tate normal form for that specific degree $\ell_i$, instead of as steps between Montgomery curves.




The Tate normal form for a degree $\ell_i$ in general requires two coefficients $b, c \in \mathbb{F}_p$ instead of the single Montgomery coefficient $a \in \mathbb{F}_p$, and the radical isogeny computes $b', c'$ associated to $\mathfrak{l}_i * E_a$.

In an efficient implementation, both $b$ and $c$ would be represented in projective coordinates. We know of only one such implementation, given in [17]. We sketch two attack approaches to extend the proposed attacks to such an implementation:

1. Find a Tate normal curve of degree $\ell_i$ such that either $b$ or $c$ has a strongly-correlated representation. The generic adaptive attack then works exactly the same.
2. Find the length of the chain by feeding a curve $E_{\mathtt{PK}}$ such that we map back to $E_0$ when we map back to Montgomery form at the end of the chain. This requires feeding several different $E_{\mathtt{PK}j}$ representing several different lengths of chains.

Note that the attack becomes easier when using radical isogenies: these chains are computationally very distinct from ordinary isogeny evaluations, and so we only need to discover the length of the chain. Furthermore, radical isogenies are performed for low degrees (up to 13), which implies that we do not perform these degrees in the rest of the steps $\mathfrak{l}^{(j)}$. This increases $p_{\mathfrak{a}_k}$ substantially.