Faulty isogenies: a new kind of leakage

Gora Adj[1], Jesús-Javier Chi-Domínguez[2], Víctor Mateu[2], and Francisco Rodríguez-Henríquez[2]

[1] Departament de Matemàtica, Universitat de Lleida, Spain
gora.adj@gmail.com

[2] Cryptography Research Centre, Technology Innovation Institute, Abu Dhabi, UAE
{jesus.dominguez,victor.mateu,francisco.rodriguez}@tii.ae

Abstract. In SIDH and SIKE protocols, public keys are defined over quadratic extensions of prime fields. We present in this work a projective invariant property characterizing affine Montgomery curves defined over prime fields. We then force a secret 3-isogeny chain to repeatedly pass through a curve defined over a prime field in order to exploit the new property and inject zeros in the A-coefficient of an intermediate curve to successfully recover the isogeny chain one step at a time. Our results introduce a new kind of fault attacks applicable to SIDH and SIKE.

Keywords: isogeny-based cryptography · fault injection attack

1 Introduction

In a seminar held in 1997, Couveignes proposed an isogeny-based scheme for mimicking the Diffie-Hellman key exchange protocol [22]. Couveignes notes were later posted in [7]. The first published isogeny-based cryptographic primitive was by Charles, Lauter and Goren in [3], where they proposed a hash function whose collision resistance was extracted from the problem of path-finding in supersingular isogeny graphs. As early as 2006, Rostovtsev and Stolbunov introduced in [21] isogeny-based cryptographic schemes (this was followed by Stolbunov in [23]) as potential candidates for post-quantum cryptography. In his 2010 paper [23], Stolbunov proposed a Diffie-Hellman-like protocol whose security guarantees were based on the difficulty of finding smooth-degree isogenies between ordinary elliptic curves.

Jao and De Feo [15] proposed in late 2011 a Diffie-Hellman key-exchange scheme, which has as underlying hard problem the difficulty of constructing isogenies between supersingular elliptic curves defined over quadratic extension prime field $\mathbb{F}_{p^2}$. Within the context of the NIST standardization process [17], it was proposed in [1] an isogeny-based key exchange protocol named SIKE, which can be seen as an SIDH variant equipped with a key encapsulation mechanism. SIKE was selected as one of the five third-round alternate KEM candidates of the NIST contest.

As we will see in Section 2.3, SIDH and SIKE public keys include not only the image curve of a secret isogeny, but also the auxiliary images of the other party's

two torsion basis points. It has been long suspected that this extra information can help SIKE cryptanalysis, but until now no efficient passive attack has been found for SIKE, in spite of several potentially promising results [18,19]. [3]

On the contrary, in 2016, Galbraith, Petit, Shani and Ti [11] presented an active attack against SIDH, which exploits the additional torsion-point information included in Alice's public key. The attack in [11] consists of sending to Bob a tuple of manipulated torsion points that appears to be Alice's legitimate public key [4]. Then, the attacker observes if her public key manipulation produces (or not) errors in the protocol and by doing so, starting from the least significant bit, she can guess one bit at a time. This allows recovery of the secret key after a linear number of queries with respect to the size of Bob's exponent e_3 (*cf.* Section 2).

As a countermeasure to their attack, the authors of [11] recommended applying a variant of the Fujisaki-Okamoto transform [10]. Obviously, this mitigation entailed a significant performance cost and SIKE was submitted to the NIST post-quantum cryptography standardization program as the combination of SIDH along with the Hofheinz–Hövelmanns–Kiltz transform [14] (a variant of [10]), for enabling a key encapsulation mechanism.

In 2017, Ti [25] proposed a fault attack that allows to recover Alice's secret isogeny ϕ_A exploiting the fact that if the image of a random point $S \in E[p+1]$ under ϕ_A is revealed, then with high probability, the attack is successful [24, Remark 3]. To accomplish this, the attacker must succeed injecting a fault *before* Alice starts computing her public key. Recall that in a normal execution of Alice's key generation procedure, she must compute the images of fixed basis points of degree three under her secret isogeny ϕ_A. The attacker's goal is to try to perturb any of these points by injecting a fault in a timely manner. Recently, the authors of [24] reported a real implementation of Ti's attack that obtained a small effectiveness (mainly due to the difficulty of producing electromagnetic injections at the right moment).

Since Ti's attack only deals with injecting a fault in the key generation procedure, it works equally well for SIDH and SIKE. However, Ti's attack always produces an error in the SIDH/SIKE shared secret computation. Hence, the recovered secret key can only be useful if the key generation procedure commits the sin of using the same private key, something that should not happen in any proper implementation of SIDH or SIKE.[5] Further, there exist a simple and inexpensive countermeasure to prevent Ti's attack, which is checking the order or correctness of the auxiliary points before to their publication [25,24,4].

---

[3] See [4, §7] for a compelling argument about why is unlikely that these torsion point attacks will ever dent the security provided by SIDH and SIKE.

[4] There is no known efficient approach for validating public keys in SIDH or SIKE. In fact, key validation is so problematic in SIDH or SIKE that if an effective algorithm for validating public keys would ever be found, then such procedure could also be used to efficiently recover secret keys from public keys [22,12]

[5] In [24] the authors suggest multipartite SIDH key exchange as a more plausible scenario for Ti's attack.

Generic fault injection and side-channel attacks against the Fujisaki-Okamoto-based key encapsulation mechanism were presented in [29,26]. In [26, §4.3], the authors applied the attack of [11] to SIKE. To do so, the authors implemented a plaintext-checking oracle by means of a side-channel exploitation. The attacker compares the side-channel leakage output by the SIKE decapsulation block when processing a valid ciphertext with a modified one. The combination of the active attack of [11] with the plaintext-checking oracle, allows to have a full SIKE key recovery by invoking less than 500 oracle accesses (see [26, Table 2]).

In [2], Campos, Krämer and Müller presented a safe-error attack against SIKE and CSIDH protocols. They launched a real attack against the SIKEp434 instantiation of SIKE implemented on a Cortex-m4 processor. The attack achieved almost 100% of full key recovery at the price of five fault injections per single bit for a total of some $1,090$ injections. We stress that the attack in [2] does not exploit any potential vulnerability on the isogeny computations of SIKE.

Recently, De Feo, El Mrabet, Genêt Kaluđerović, Guertechin, Pontié, and Tasso showed SIKE is vulnerable to zero-value attack [9].

Our Contributions. We present a new fault-injection adaptive attack on SIDH, which could apply to SIKE under the same plaintext-checking oracle as in [26]. Our analysis centers on Bob's 3-isogeny chain computation and relies on the following:

– A new $\mathbb{F}_p$-invariant property to characterize Montgomery projective curves defined over $\mathbb{F}_p$.
– Using this $\mathbb{F}_p$-invariant, force Bob's 3-isogeny chain computation to pass through an $\mathbb{F}_p$-curve.

With these results, our fault-injection adaptive attack flow can be summarized as follows:

1. **Send an altered version of Alice's public key to Bob.** By using an appropriate choice of points for Alice's public key, the attacker can force Bob to return to the curve $E\colon y^2 = x^3 + 6x^2 + x$ when processing the ith 3-isogeny.
2. **Inject a fault in the $(i+1)$th 3-isogeny output, such that the imaginary part of the Montgomery A-coefficient is zeroed.** This action allows us to guess the $(i+1)$th trit of Bob's private key.

As a proof of concept, we additionally provide a C-code implementation that simulates the injection and verifies the correctness of the analysis. The code is freely available at https://github.com/FaultyIsogenies/faulty-isogeny-code .

The reminder of this paper is organized as follows. Section 2 presents some basic definitions of elliptic curves and their isogenies. In Section 3, we give a detailed description of the new fault injection attack. In particular, Section 3.1 and Section 3.2 detail the $\mathbb{F}_p$-invariant property and the core of the attack, respectively. Section 4 illustrates an actual proof-of-concept implementation of the attack. We also discuss in Section 4 potential countermeasures to thwart or mitigate the attack. Finally, we draw our concluding remarks in Section 5.

2 Preliminaries

In this section we present some basic mathematical definitions for elliptic curves and isogenies. These definitions are extended and discussed in more detail in [28]. Furthermore, we briefly describe the SIDH key agreement protocol (see [15,8]).

2.1 Supersingular elliptic curves and their isogenies

Let $p > 3$ be a prime number, $\mathbb{F}_p$ the finite field with p elements, and $\mathbb{F}_{p^2}$ its quadratic extension. In this paper, we consider only elliptic curves E defined over $\mathbb{F}_{p^2}$ that are supersingular of order $\#E(\mathbb{F}_{p^2}) = (p+1)^2$ and in Montgomery form given by

$$E\colon By^2 = x^3 + Ax^2 + x, \tag{1}$$

for some $A, B \in \mathbb{F}_{p^2}$. If $A, B \in \mathbb{F}_p$, we say that E is defined over $\mathbb{F}_p$.

The order d of a point $P \in E(\mathbb{F}_{p^2})$ is the smallest positive integer such that

$$[d]P = \underbrace{P + \cdots + P}_{d \text{ times}} = \mathcal{O}.$$

The d-torsion subgroup, denoted by $E[d]$, is the set of points $\{P \in E(\mathbb{F}_{p^2}) \mid [d]P = \mathcal{O}\}$. If $\gcd(p, d) = 1$, then $E[d]$, as a subgroup of E, is isomorphic to $\mathbb{Z}/n\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}$. The j-invariant of an elliptic curve E in Montgomery form is defined as

$$j(E) = \frac{256(A^2 - 2)^3}{A^2 - 4}. \tag{2}$$

Two elliptic curves have the same j-invariant if and only if they are isomorphic over some extension field of $\mathbb{F}_p$. An isogeny $\phi\colon E \to E'$ over $\mathbb{F}_{p^2}$ is a non-zero rational map satisfying $\phi(\mathcal{O}) = \mathcal{O}$. Every isogeny is a surjective group homomorphism with finite kernel. Two elliptic curves E and E' are said to be isogenous over $\mathbb{F}_{p^2}$ if there exists an isogeny $\phi\colon E \to E'$ defined over $\mathbb{F}_{p^2}$, and this happens if and only if $\#E(\mathbb{F}_{p^2}) = \#E'(\mathbb{F}_{p^2})$.

Let ϕ be an isogeny defined over $\mathbb{F}_{p^2}$, then it can be represented as

$$\phi = (r_1(X), r_2(X) \cdot Y),$$

where $r_1, r_2 \in \mathbb{F}_{p^2}(X)$, and is said to be separable if $r_1'(X) \neq 0$, and inseparable otherwise. Now, let $r_1(X) = p_1(X)/q_1(X)$, where $p_1, q_1 \in K[X]$ with $\gcd(p_1, q_1) = 1$. Then the degree of ϕ is $\max(\deg p_1, \deg q_1)$, and ϕ is separable means that $\#\ker \phi = \deg \phi$. All the isogenies considered in this work will be separable. For brevity, we say d-isogeny for a degree-d isogeny.

The dual of a d-isogeny $\phi\colon E \to E'$ is the unique d-isogeny $\hat{\phi}\colon E' \to E$ such that $\hat{\phi} \circ \phi = [d]$ and $\phi \circ \hat{\phi} = [d]$. If ϕ has cyclic kernel $\langle P \rangle$ and $E[d] = \langle P, Q \rangle$, then $\langle \phi(Q) \rangle$ is the kernel of $\hat{\phi}$.

Isogenies of small prime degree d are computed in time $O(d)$ by means of Vélu's formulas. For a degree d^e, one splits a d^e-isogeny as the composition of e small d-isogenies.

2.2   Montgomery x-only point arithmetic

We introduce here the notation $x(P)$ to refer to the x coordinate of an elliptic curve point P. The Montgomery curve model is especially amenable for performing x-only differential point addition as detailed in [6].

Following the SIDH and SIKE notation, we denote:

- $x([2]P) \leftarrow \texttt{xdbl}(x(P), A)$, and $x([2^e]P) \leftarrow \texttt{xdble}(x(P), A, e)$
- $x([3]P) \leftarrow \texttt{xtpl}(x(P), A)$, and $x([3^e]P) \leftarrow \texttt{xtple}(x(P), A, e)$

where A is the constant coefficient of a Montgomery curve E. Given two points P, Q on E and a positive integer k, the x-coordinate $x(P + [k]Q)$ is efficiently computed by means of an $(\log_2 k)$-step Montgomery three point ladder procedure at a per-step cost of one $\texttt{xdbl}$, and one differential point addition.

Computing 2-, 3-, and 4-isogenies can be done efficiently by applying Vélu's formulas [27], but there is a more efficient way. Indeed, the idea of x-only point addition can be extended to x-only isogenies, which are computed from the x-coordinates of the kernel points [5,20]. Let K be a point of order $d \in \{2, 3, 4\}$, and let $\phi\colon E \to E'$ be the isogeny with kernel $\langle K \rangle$. There are two isogeny built-in functions:

- Isogeny construction: A', coeff $\leftarrow \texttt{xisog}(x(K))$
- Isogeny evaluation: $x(\phi(Q)) \leftarrow \texttt{xeval}(x(Q),$ coeff$)$.

The $\texttt{xisog}$ function computes the A-coefficient of the codomain curve E', together with some data coeff that is needed in the $\texttt{xeval}$ function and related to the kernel points. In $\texttt{xeval}$, one pushes the point Q through the isogeny.

From now on, when we take points as inputs and outputs of $\texttt{xisog}$ and $\texttt{xeval}$ in order to alleviate the notation, it should be understood the x-coordinate of these points. As well, for a curve point P, writing $P \in \mathbb{F}_p$ means here $x(P) \in \mathbb{F}_p$. For d-isogenies, we will have $\texttt{xisog}d()$, $\texttt{xeval}d()$.

2.3   The SIDH protocol at a glance

SIDH is a key agreement scheme based on computations of isogenies between supersingular montgomery curves. The protocol consists of 2 algorithms: $\texttt{keygen}$ and $\texttt{derive}$. It also provides the setup information required by the two parties, Alice and Bob, to run the algorithms:

- A prime $p = 2^{e_2} 3^{e_3} - 1$ and $\mathbb{F}_{p^2} = \mathbb{F}_p[i]/(i^2 + 1)$.
- The starting curve $E/\mathbb{F}_{p^2} : y^2 = x^3 + 6x^2 + x$ with $\#E(\mathbb{F}_{p^2}) = (2^{e_2} 3^{e_3})^2$ and j-invariant $j(E) = 287496$.
- The points $P_A, Q_A, D_A = P_A - Q_A$ of order 2^{e_2}.
- The points $P_B, Q_B, D_B = P_B - Q_B$ of order 3^{e_3}.

The first algorithm $\texttt{keygen}$ allows both Alice and Bob to generate their respective public and private keys. Alice:

1. Randomly selects her private key sk_2 from $[\![1 \mathrel{..} 2^{e_2 - 1} - 1]\!]$.

2. Computes $R_A = P_A + [\text{sk}_2]Q_A$.
3. Finds the 2^{e_2}-isogeny ϕ_A generated by R_A.
4. Pushes P_B, Q_B, and D_B through the isogeny ϕ_A to get her public key $\text{pk}_2 = (\phi_A(P_B), \phi_A(Q_B), \phi_A(D_B))$.

Likewise, Bob:

1. Randomly selects his private key sk_3 from $[\![1 \ .. \ 3^{e_3-1} - 1]\!]$.
2. Computes $R_B = P_B + [\text{sk}_3]Q_B$.
3. Finds the 3^{e_3}-isogeny ϕ_B generated by R_B.
4. Pushes P_A, Q_A, and D_A through the isogeny ϕ_B to get his public key $\text{pk}_3 = (\phi_B(P_A), \phi_B(Q_A), \phi_B(D_A))$.

The second algorithm `derive` receives, as input, a public key and a private key, and computes, as output, the shared j-invariant. Alice would run `derive`$(\text{pk}_3, \text{sk}_2)$ which would perform the following operations:

1. Compute $\phi_B(R_A)$ as $P'_A + [\text{sk}_2]Q'_A$.
2. Find 2^{e_2}-isogeny ψ_A generated by $\phi_B(R_A)$.
3. Obtain the codomain curve E_{AB} from ψ_A.
4. Compute the j-invariant of E_{AB}.

Similarly, Bob would run `derive`$(\text{pk}_2, \text{sk}_3)$ to obtain the same j-invariant. In this case, the algorithm would:

1. Compute $\phi_A(R_B)$ as $P'_B + [\text{sk}_3]Q'_B$.
2. Find 3^{e_3}-isogeny ψ_B generated by $\phi_A(R_B)$.
3. Obtain the codomain curve E_{BA} from ψ_B.
4. Compute the j-invariant of E_{BA}.

Alice and Bob have now created a shared secret by computing the j-invariant of their respective isomorphic curves E_{AB} and E_{BA}.

3   New leakage on $\mathbb{F}_p$-isogenies

Most of the known active attacks on SIDH and SIKE focus on the public elliptic curve points. For example: following the same notation as Section 2.3, Ti's attack [25] is centered on the output points $\phi(P')$, $\phi(Q')$, and $\phi(D')$ of `keygen`, where it changes the point $\phi(D')$ by a random point $\phi(T)$ on the curve E'.

In contrast, our attack to SIDH injects faults on the curve coefficients by building public trapdoor instances. In this section we show how injecting zeros on the A-coefficient of Montgomery curves allows deciding whether A belongs to $\mathbb{F}_p$ or not. Besides, that, we show how to build inputs that allow us to trigger these fault injections in order to fully recover Bob's private key.

3.1 An $\mathbb{F}_p$-invariant on projective representations

For efficiency reasons, SIDH performs 3-isogenies using a projectivized coefficient $A \in \mathbb{F}_{p^2}$ that can be described as, $(\alpha \colon \beta) = (\tilde{A}+2C \colon \tilde{A}-2C)$ such that, $A = \tilde{A}/C$ for some $\tilde{A}, C \in \mathbb{F}_{p^2}$. This representation allows the computations to minimize the number of divisions. Let us write $\alpha = a+ib$ and $\beta = c+id$ with $i^2 = -1$ and $a, b, c, d \in \mathbb{F}_p$. We want to focus on determining when does the quotient $\tilde{A}/C$ belong to $\mathbb{F}_p$.

If we extend the equations we have

$$
\begin{aligned}
A = \frac{\tilde{A}}{C} &= \frac{2(\alpha + \beta)}{\alpha - \beta} = \frac{2(a + c) + 2(b + d)i}{(a - c) + (b - d)i} \\
&= \frac{\big(2(a + c) + 2(b + d)i\big)\big((a - c) - (b - d)i\big)}{(a - c)^2 + (b - d)^2} \\
&= \frac{\big(2(a^2 - c^2) + 2(b^2 - d^2)\big) + \big(2(b + d)(a - c) - 2(a + c)(b - d)\big)i}{(a - c)^2 + (b - d)^2} \\
&= \frac{\big(2(a^2 + b^2) - 2(c^2 + d^2)\big) + 4\big(ad - bc\big)i}{(a - c)^2 + (b - d)^2}
\end{aligned}
\tag{3}
$$

From Equation 3 we have that A belongs to $\mathbb{F}_p$ if and only if $4\big(ad - bc\big) = 0$

A trivial way to have A in $\mathbb{F}_p$ would be if $b = d = 0$. In that case we would obtain the projective A-coefficient $(a \colon c)$. The following lemma ensures that we can always take $b = d = 0$.

Lemma 1. *For any Montgomery curve E with affine A-coefficient in $\mathbb{F}_p$, if $(a + ib \colon c + id)$, for some $a, b, d, c \in \mathbb{F}_p$, is a projective curve coefficient of E, then also is $(a \colon c)$.*

Proof. From Equation 3 we have the formula to compute the affine coefficient A from the projective coefficient $(a \colon c)$ is:

$$
A = \frac{\tilde{A}}{C} = \frac{2a^2 - 2c^2}{(a - c)^2}.
$$

The same equation applied to the affine coefficient A' from the projective coefficient $(a + ib \colon c + id)$ results in:

$$
A' = \frac{\tilde{A}'}{C'} = \frac{2(a^2 + b^2) - 2(c^2 + d^2)}{(a - c)^2 + (b - d)^2}.
$$

Hence

$$A = A' \iff \frac{2a^2 - 2c^2}{(a-c)^2} = \frac{2(a^2 + b^2) - 2(c^2 + d^2)}{(a-c)^2 + (b-d)^2}$$
$$\iff (2a^2 - 2c^2)((a-c)^2 + (b-d)^2) = (2(a^2 + b^2) - 2(c^2 + d^2))(a-c)^2$$
$$\iff (2a^2 - 2c^2)(b-d)^2 = (2b^2 - 2d^2)(a-c)^2$$
$$\iff 2(a-c)(a+c)(b-d)^2 = 2(b-d)(b+d)(a-c)^2$$
$$\iff (a+c)(b-d) = (b+d)(a-c)$$
$$\iff bc = ad.$$

$\square$

Remark 1. Notice, the $\mathbb{F}_p$-invariant given by Lemma 1 easily extends to projective points $(X \colon Z)$ such that $X, Z \in \mathbb{F}_{p^2}$ and $x(P) = X/Z$ for some point P on E. To be more precise, any projective point $(x_0 \colon z_0)$ describes the same affine point in $\mathbb{F}_p$ as another projective point $(x_0 + ix_1 \colon z_0 + iz_1)$ for all $x_0, x_1, z_0, z_1 \in \mathbb{F}_p$ if and only if $x_0 z_1 = z_0 x_1$.

3.2   Faulting 3-isogenies

At the beginning of a SIDH key agreement phase, Bob receives from Alice her public key $\text{pk}_2 = \{\phi_A(P_B), \phi_A(Q_B), \phi_A(P_B - Q_B)\}$. A malicious Alice may send to Bob another set of image points that may help her to [partially] guess Bob's secret key sk_3. Another possibility is that an active attacker Eve launches a man-in-the-middle attack by intercepting Alice image points and relaying to Bob a different set of points of her choice. Either way, according to the SIDH specifications, Bob has little defense to distinguish legitimate image points from [carefully chosen] fake ones [4].

Furthermore, our security model assumes that Eve has the ability of injecting faults during the execution of 1. Let us assume that Bob is executing the $i + 1$ iteration of 1, and that he is in the process of computing the degree-3 isogeny $\phi_i \colon E_i \to E_{i+1}$. Then, by carefully timing her attack, Eve can inject zeroes into the registers $b \mapsto 0$ and $d \mapsto 0$ of A_{i+1} in the i-th iteration of 1. By zeroing the imaginary part of the coefficient A_{i+1}, Eve can infer if it belongs to $\mathbb{F}_p$ or to $\mathbb{F}_{p^2} \setminus \mathbb{F}_p$. In fact, the series of point triplings performed in line 6 could produce degenerated outputs, and from these degenerated outputs arise non-supersingular curves, which are easy to detect. If the supersingularity is preserved, then the attacker can infer that the injection affected one A-coefficient in $\mathbb{F}_p$, otherwise, A_{i+1} cannot possibly live in $\mathbb{F}_p$ as proven in Lemma 1.

In the following lines, we discuss in detail how to take advantage of this simple but powerful observation given the public information from the NIST candidate SIKE.

The current public SIKE parameters corresponding to the third round of the NIST standardization process have $A = 6$ and $B = 1$, implying that the

Algorithm 1 Strategy evaluation for computing 3^{e_3}-isogenies with cyclic kernel generated by a order-3^{e_3} point R (for more details see Algorithms 19-20 of SIKE specifications [1])

Inputs: Point $R \in E$ of order-3^{e_3}. The A-coefficient of E, and a strategy S consisting of $e_3 - 1$ positive integers
Output: Codomain curve $E/\langle R \rangle$ of the 3^{e_3}-isogeny with kernel $\langle R \rangle$

1:   $K \leftarrow []$
2:   $k \leftarrow 0$
3:   **for** $i = 0$ to $e_3 - 2$ **do**
4:      **while** R is not an order-3 point **do**
5:        $K.push(R)$
6:        $R \leftarrow \mathtt{xtple}(R, A, S_k)$
7:        $k \leftarrow k + 1$
8:      **end while**
9:      A, coeff $\leftarrow \mathtt{xisog3}(R)$
10:     **for** $j = 0$ to k **do**
11:       $K_j \leftarrow \mathtt{xeval3}(K_j, \text{coeff})$
12:     **end for**
13:     $R \leftarrow K.pop()$
14: **end for**
15: A, coeff $\leftarrow \mathtt{xisog3}(R)$
16: **return** A

Montgomery coefficient of the initial curve E is purely defined over $\mathbb{F}_p$. In addition, the public order-3^{e_3} points P and Q have x-coordinates over $\mathbb{F}_p$, but $x(D) = x(P - Q)$ lives in $\mathbb{F}_{p^2} \setminus \mathbb{F}_p$ (see SIDH specifications from [1]).

Let Bob's private key sk_3 be represented in radix three as,

$$\mathrm{sk}_3 = \sum_{i=0}^{e_3} s_i 3^i.$$

To recover s_0, we focus on the four order-3 points on $E\colon y^2 = x^3 + 6x^2 + x$, which can be generated by linear combinations of $P_3 = [3^{e_3-1}]P$ and $Q_3 = [3^{e_3-1}]Q$. In other words, any order-3 point is either

$$P_3, \; Q_3, \; P_3 + Q_3, \text{ or } P_3 - Q_3.$$

By construction, P_3 and Q_3 lie on $\mathbb{F}_p$, while $(P_3 + Q_3)$ and $(P_3 - Q_3)$ do not. Given that the secret point of SIDH and SIKE has the form $P + [\mathrm{sk}_3]Q$, it implies that the first 3-isogeny $\phi_1 \colon E \to E_1$ has as kernel generator either P_3, $(P_3 + Q_3)$ or $(P_3 - Q_3)$. Since the coefficient of the curve E_0 lies in $\mathbb{F}_p$, then the coefficient of E_1 will also lie in $\mathbb{F}_p$ if and only if ϕ_1 has an order-3 point generator K such that $x(K) \in \mathbb{F}_p$.

In Table 1 we show how given a successful fault in the first iteration of 1 after line 9, an attacker can obtain the value s_0 by just knowing the information about which points are in $\mathbb{F}_p$ and which ones are not.

As a way of illustration, since $P_3, Q_3 \in \mathbb{F}_p$ and $(P_3+Q_3), (P_3-Q_3) \in \mathbb{F}_{p^2} \setminus \mathbb{F}_p$, we have that P_3 is the one and only kernel living in $\mathbb{F}_p$. So, if the injected fault produces no error it immediately follows that $s_0 = 0$. Otherwise, if the fault produces an error, then it is still unclear whether the secret value of s_0 is one or two. In this case, by repeating this attack but this time sending to Bob the public key

$$\text{pk}_2' = (P + Q, Q, P),$$

the attacker can find out the value of s_0 as follows. Notice that when Bob executes once again 1, depending on the value of s_0, it will produce as kernel of ϕ_1 either $P + [2]Q = P - Q$ or $P + [3]Q = P$. The former point belongs to $\mathbb{F}_{p^2} \setminus \mathbb{F}_p$, whereas the latter point lies in $\mathbb{F}_p$. Hence, if the injected fault produces no error, it immediately follows that $s_0 = 2$. Otherwise, $s_0 = 1$.

Besides the case $P_3 \in \mathbb{F}_p$ and $(P_3 + Q_3), (P_3 - Q_3) \in \mathbb{F}_{p^2} \setminus \mathbb{F}_p$, there are other five possible combinations for these three order-3 points. The analysis of the other five cases is similar to the one given above (corresponding to the first row of Table 1). So although the details are omitted here, we report in Table 1 six cases where it is possible to guess the value of s_0 at the first attempt using the public key $\text{pk}_2 = (P, Q, P - Q)$. Table 1 also shows twelve cases where a second fault is required. If a second attempt is required, the attacker must send to Bob the public key $\text{pk}_2' = (P + Q, Q, P)$.

Remark 2. If the trit s_0 was uniformly sampled, the attacker will successfully learn it after an average of $1 + \frac{2}{3} = \frac{5}{3}$ attempts. More concretely, as shown in Table 1, there are six cases where the attacker can learn the value of s_0 at the first attempt, whereas there exist twelve cases that force the attacker to perform two attempts to fully guess the value of s_0.

$x(Q_3)$		$x(P_3)$	$x(P_3 + Q_3)$	$x(P_3 - Q_3)$	Number of instances		
					trit $s = 0$	trit $s = 1$	trit $s = 2$
		in $\mathbb{F}_p$	not in $\mathbb{F}_p$	not in $\mathbb{F}_p$	1	2	2
in $\mathbb{F}_p$		not in $\mathbb{F}_p$	in $\mathbb{F}_p$	not in $\mathbb{F}_p$	2	1	2
		not in $\mathbb{F}_p$	not in $\mathbb{F}_p$	in $\mathbb{F}_p$	2	2	1
	not in $\mathbb{F}_p$	in $\mathbb{F}_p$	in $\mathbb{F}_p$	1	2	2
not in $\mathbb{F}_p$		in $\mathbb{F}_p$	not in $\mathbb{F}_p$	in $\mathbb{F}_p$	2	1	2
		in $\mathbb{F}_p$	in $\mathbb{F}_p$	not in $\mathbb{F}_p$	2	2	1

Table 1. Number of instances to guess s from two random order-3 points P_3 and Q_3 on $E\colon y^2 = x^3 + 6x^2 + x$ such that $\langle P_3, Q_3 \rangle = E[3]$. The first instance refers to $(P_3, Q_3, P_3 - Q_3)$, while the second instance to $(P_3', Q_3', P_3' - Q_3') = (P_3 + Q_3, Q_3, P_3)$.

3.3   Forcing $E_i\colon y^2 = x^3 + 6x^2 + x$

In this section, we show how to force Bob's strategy evaluation (1) to repeatedly pass at iteration i through the curve $E\colon y^2 = x^3 + 6x^2 + x$. This way, an

attacker would be able to guess the secret trit s_i, by simply determining whether the $(i+1)$th curve is either in $\mathbb{F}_p$ or not. As illustrated in 1, the 3^{e_3}-isogeny procedure is split into e_3 small 3-isogenies, where the only public curves are the first and last ones:

$$E_0 = E \to E_1 \to \cdots \to E_{e-1} \to \tilde{E} = E_{e_3}.$$

Each E_i has its Montgomery coefficient $A_i \in \mathbb{F}_{p^2}$. Note that there exist four different kinds of 3-isogenies with curve domain E_i, but only two of them map to either E_{i+1} or E_{i-1} (this last one goes in the reverse direction and, as mentioned in Section 2.1, corresponds to the dual isogeny). However, we know the successive image points $\phi_1 \circ \phi_2 \circ \cdots \circ \phi_i([3^{e_3-1-i}]Q)$ determine the dual 3-isogenies $\hat{\phi}_i \colon E_i \to E_{i-1}$, and the kernel of the $(i+1)$th 3-isogeny is given by

$$[3^{e_3-1-i}](\phi_1 \circ \phi_2 \circ \cdots \circ \phi_i)(P + [\mathrm{sk}_3]Q).$$

Notice that the trit s_i completely determines the 3-isogeny $\phi_i \colon E_{i-1} \to E_i$. In Section 3.2 we detailed how to obtain the first trit s_0. In the following lines we show how to fully recover Bob's private key by forcing him to return to curve E when processing the ith isogeny in `derive` algorithm.

Once the attacker knows s_0, she can move to E_1 by using the 3-isogeny $\phi_1 \colon E_0 \to E_1$ with kernel generator $[3^{e_3-1}](P + [s_0]Q)$. Next, we look for an order-3^{e_3} point
$$T \in E_1 : [3^{e_3-1}]\phi_1(Q) \neq \pm[3^{e_3-1}]T.$$

Let

  - $P' = \phi_1(Q) + [s_0]T$,
  - $Q' = -T$,
  - $\hat{\phi}_1 \colon E_1 \to E_0$ be the dual 3-isogeny with kernel generator

$$[3^{e_3-1}](P' + [s_0]Q') = [3^{e_3-1}]\phi_1(Q).$$

When Bob runs `derive` algorithm with input $\mathrm{pk}_2 = (P', Q', P' - Q')$, we know that the second iteration of the strategy evaluation will pass through E. Consequently, the attacker can guess s_1, as she did for s_0, by injecting the fault, now in the second iteration, and using Table 1. In case she needs two instances, she could build the second one with input

$$\mathrm{pk}_2' = (P' + [3]Q', Q', P' + [2]Q').$$

Now that the attacker knows s_0 and s_1, she can extend this idea to recover the rest of the trits. As we have shown, the attacker controls the outcome when Bob computes on E. Therefore, to obtain every s_i from sk_3 the attacker needs to provide a proper input public key for Bob such that at iteration i of 1 the curve obtained is E. Figure 1 summarizes this idea.
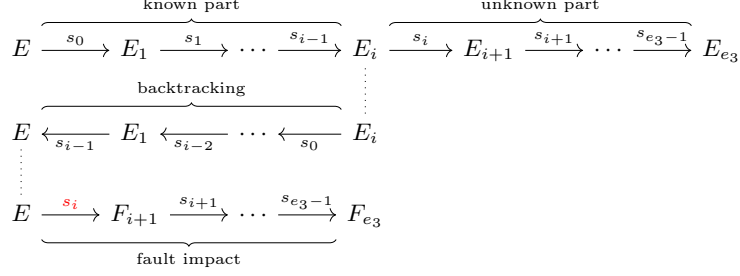
Fig. 1. Attack idea: forcing Bob's strategy evaluation to pass through $E\colon y^2 = x^3 + 6x^2 + x$ at the ith iteration, and guess s_i by knowing whether the $(i+1)$th curve is defined over $\mathbb{F}_p$ or not.

Let us divide the private key sk_3 into two parts

$$\mathrm{sk}_3 = \sum_{j=0}^{i-1} s_j 3^j + \sum_{t=i}^{e_3} s_t 3^t$$

and assume the attacker knows $\mathrm{sk} = \sum_{j=0}^{i-1} s_j 3^j$. With this information, she can easily compute the codomain curve E_i of the 3^i-isogeny $\phi\colon E \to E_i$ with kernel generator

$$[3^{e_3-i}](P + [\mathrm{sk}]Q).$$

The attacker can now forge a public key pk_2 that ensures that the ith iteration of Bob's strategy evaluation passes through $E\colon y^2 = x^3 + 6x^2 + x$ by

1. finding an order-3^{e_3} point $T \in E_i : [3^{e_3-1}]T \neq \pm[3^{e_3-1}]\phi(Q)$,
2. assigning $P' = \phi(Q) + [\mathrm{sk}]T$
3. assigning $Q' = -T$.

Finally, she sends

1. $\mathrm{pk}_2 = (P', Q', P' - Q')$ and,
2. $\mathrm{pk}'_2 = (P' + [3^i]Q', Q', P' + [3^i - 1]Q')$, if needed.

Next, we illustrate how and why the attacker can guess the trit s_i in the following lemma.

Lemma 2. *Let us assume Bob receives the public key* $\mathrm{pk}_2 = (P', Q', P' - Q')$ *or* $\mathrm{pk}'_2 = (P' + [3^i]Q', Q', P' + [3^i - 1]Q')$, *as previously constructed. Then, when Bob performs the* **derive** *procedure with the received public key, his* ith *3-isogenous curve will coincide with* $E\colon y^2 = x^3 + 6x^2 + x$.

Proof. From the nature of pk_2 and pk'_2, the ith 3-isogeny Bob's secret kernel R is

$$[3^{e_3-i}](P' + [3^i]Q' + [\mathrm{sk}_3]Q') = [3^{e_3-i}](P' + [\mathrm{sk}_3]Q').$$

Notice $[3^{e_3-i}]([sk_3]Q') = [3^{e_3-i}]([sk]Q')$ and thus $R = [3^{e_3-i}]\phi(Q)$. By construction, $\phi\colon E \to E_i$ is a 3^i-isogeny with kernel generator $[3^{e_3-i}](P + [sk]Q)$. That is, R is the kernel generator of the dual 3^i-isogeny $\hat{\phi}\colon E_i \to E$ since $\langle P + [sk_3]Q, Q\rangle = E[3^{e_3}]$ holds. Consequently, Bob will pass through the ith codomain curve $E\colon y^2 = x^3 + 6x^2 + x$. On that basis, Bob's $(i+1)$th 3-isogeny will have kernel either P_3, $(P_3 + Q_3)$, or $(P_3 + [2]Q_3) = (P_3 - Q_3)$ as in Equation 4, all of them lying on $E\colon y^2 = x^3 + 6x^2 + x$. Consequently, our attacker has the possibility of correlating the trit s_i by guessing whether Bob passed through $\mathbb{F}_p$ 3-isogeny codomain curves, by applying Lemma 1 and using Table 1.

$$P_3 = [3^{e_3-1-i}]\left(\hat{\phi}(P') + [sk]\hat{\phi}(Q')\right),$$
$$P_3 + Q_3 = [3^{e_3-1-i}]\left(\hat{\phi}(P') + [sk + 3^i]\hat{\phi}(Q')\right), \tag{4}$$
$$P_3 - Q_3 = [3^{e_3-1-i}]\left(\hat{\phi}(P') + [sk + 2 \cdot 3^i]\hat{\phi}(Q')\right).$$

3.4 Proposed attack

Let $\mathbf{O}_{sk_3}(pk_2, i)$ be an oracle taking as input an SIDH/SIKE public key $pk_2 = (P', Q', P' - Q')$ and an integer $0 \le i \le e_3 - 2$. The oracle internally performs a strategy evaluation with Bob's static private key sk_3 and injects a fault at the $(i+1)$th iteration (from line 9 of 1) as detailed in Section 3.3. Then it outputs 1 if supersingularity is preserved for E_{e_3}, and 0 otherwise. Based on Section 3.2 and Section 3.3, and using $\mathbf{O}_{sk_3}(pk_2, i)$, 2 correctly reconstructs Bob's private key sk_3 with about $\frac{5e_3}{3} \approx \frac{5\log_2(p)}{6\log_2(3)} \approx 0.53\log_2(p)$ oracle calls.

4 Experiments and countermeasures

As a proof-of-concept of our attack, we implemented 2 using the SIDH Library v3.4 (C Edition)[6]. Our software implementation simulates the fault injection at lines 5 and 13 of 2 as follows.

The oracle $\mathbf{O}_{sk}(pk, i)$ accesses the curve $E_i\colon y^2 = x^3 + 6x^2 + x$, injects zeros in the A-coefficient of E_{i+1}, and then performs a supersingularity check by testing at the $(i+2)$th iteration $\texttt{xtpl}(R, A) = \mathcal{O}$ [7]. We ensure that the input public key $pk = (P', Q', P' - Q')$ for $E'\colon y^2 + x^3 + Ax^2 + x$ has been correctly built, by internally verifying $E_i\colon y^2 = x^3 + 6x^2 + x$, using the isogeny $\psi\colon E' \to E_i$ with kernel generator $[3^{e_3-i}](P' + [sk_3 \bmod 3^i]Q') = [3^{e_3-i}]\psi(Q)$ where $E[3^{e_3}] = \langle P, Q\rangle$.

---

[5] E' is the curve determined by P', Q' and $(P' - Q')$. We set for the case $i = 0$, $\psi\colon E \to E$ as the identity map, and $E' = E$.

[6] https://github.com/microsoft/PQCrypto-SIDH

[7] Isogenies preserve the torsion of points, and because R is the kernel of the next 3-isogeny, then verifying R is an order-3 point determines supersingularity.

Algorithm 2 Bob's private key recovery by injecting zeros on the A-coefficients

---

Inputs: Public parameters P, Q, and E from SIDH. The oracle $\mathbf{O}_{\mathrm{sk}_3}()$ as defined
　　in Section 3.4, and $\tilde{E} = E/\langle P + [\mathrm{sk}_3]Q\rangle$

Output: Bob's private key $\mathrm{sk}_3 = \sum_{i=0}^{e_3-1} s_i 3^i$

 1: $\mathrm{sk} \leftarrow 0$
 2: **for** $i = 0$ to $e_3 - 2$ **do**
 3:　　$\mathrm{pk}_2 \leftarrow (P', Q', P' - Q')$ as in Section 3.3
 4:　　Compute P_3, Q_3, and $(P_3 - Q_3)$ following Equation 4
 5:　　$b \leftarrow \mathbf{O}_{\mathrm{sk}_3}(\mathrm{pk}_2, i)$
 6:　　$c_1 \leftarrow$ number of elements in $\{P_3, P_3 + Q_3, P_3 - Q_3\}$ lying on $\mathbb{F}_p$
 7:　　$c_0 \leftarrow$ number of elements in $\{P_3, P_3 + Q_3, P_3 - Q_3\}$ not living in $\mathbb{F}_p$
 8:　　**if** $c_b = 1$ **then**
 9:　　　Guess s_i using Table 1
10:　　　$\mathrm{sk} \leftarrow \mathrm{sk} + s_i 3^i$
11:　　**else**
12:　　　$\mathrm{pk}_2' \leftarrow \left(P' + [3^i]Q', Q', P' + [3^i - 1]Q'\right)$
13:　　　$b' \leftarrow \mathbf{O}_{\mathrm{sk}_3}(\mathrm{pk}_2', i)$
14:　　　**if** $c_{b'} = 1$ **then**
15:　　　　Guess s_i using Table 1
16:　　　　$\mathrm{sk} \leftarrow \mathrm{sk} + s_i 3^i$
17:　　　**else**
18:　　　　Guess s_i by discard from the previous 2 options
19:　　　　$\mathrm{sk} \leftarrow \mathrm{sk} + s_i 3^i$
20:　　　**end if**
21:　　**end if**
22: **end for**
23: Brute force search on $s_{e_3-1} \in \{0, 1, 2\}$ such that $E/\langle P + [\mathrm{sk} + s_{e_3-1}3^{e_3-1}]Q\rangle = \tilde{E}$
24: **return** $\mathrm{sk} + s_{e_3-1}3^{e_3-1}$

　　Furthermore, we verify that P_3, $P_3 + Q_3$, and $P_3 - Q_3$ are different 3-order points on E_i for each instance. Additionally, we set the initial 3^{e_3}-order points P and Q on E: $y^2 = x^3 + 6x^2 + x$ as the public SIDH/SIKE 3^{e_3}-torsion points.

　　Table 2 summarizes the timings and oracle calls simulating the proposed attack, and reports the average running time of $\frac{5e_3}{3} = \frac{5\log_2(p)}{6\log_2(3)} \approx 0.53\log_2(p)$ oracle calls. All of our experiments successfully recover Bob's private key, and were executed on a 2.3 GHz 8-Core Intel Core i9 machine with 16GB of RAM, using gcc version 11.1.0. In particular, we use the optimized configuration from the SIDH Library v3.4 (see footnote 6). Our software implementation is freely available at https://github.com/FaultyIsogenies/faulty-isogeny-code .

　　We emphasize that our software C-code implementation is a proof-of-concept simulator that helps to illustrate and verifying the correctness of the attack. So, omitting each verification step from the code will reduce the attack latency, giving a faster key recovery.

Implications to SIKE. Recently, Ueno et al. [26] analyze a plaintext-checking Oracle-based attack; they focus on the encryption computation from the SIKE decapsulation procedure. However, our attack targets SIDH derive procedure,

Instances	e_3	Clock Cycles	Seconds	Oracle calls
SIDHp434	137	4249	4.62	226
SIDHp503	159	7257	7.76	263
SIDHp610	192	17424	18.64	318
SIDHp751	239	40638	42.77	396

Table 2. All measurements are given in millions of clock cycles, and they correspond to the average of 1000 random instances

but a combination with the plaintext-checking oracle would give a fault-injection attack on the SIKE decapsulation procedure requiring $\frac{5}{3}+1$ queries per iteration and, therefore, $\frac{8e_3}{3} \approx \frac{8\log_2(p)}{6\log_2(3)} \approx 0.84\log_2(p)$ oracle calls. This extra query comes from the nature of our attack that constructs not valid inputs for decapsulating messages (as different from attacking SIDH). Table 3 summarizes the expected running time for a fully private key recovery by attacking `Decaps` procedure.

Instances	e_3	Oracle calls	
		[26]	This work
SIKEp434	137	230	364
SIKEp503	159	266	422
SIKEp610	192	323	512
SIKEp751	239	398	630

Table 3. Expected number of plaintext-checking oracle calls and injected faults required to recovering a fixed Bob's private key when attacking `Decaps`. In the third column we predict that the required number of oracle calls is just $\frac{5}{3}e_3$, which is slightly smaller than the $2e_3$ oracle calls reported in [26, Table 2].

Countermeasures. A pairing test could work as a countermeasure, but the attacker can manipulate the points to bypass this test. Recall, $P' = \phi(Q) + [\text{sk}]T$ and $Q' = -T$ form a basis, then there is μ such that $e(P', Q') = e(P, Q)^\mu$. So she forces to pass the pairing check by looking for a point T such that $\theta = \mu/2^{e_2}$ is a quadratic residue and $\sqrt{\theta}$ is invertible modulo 3^{e_3}. If that is the case, she sets as basis input $P' \leftarrow [1/\sqrt{\mu}]P'$ and $Q' \leftarrow [1/\sqrt{\mu}]Q'$. If not, we try with another different T.

In [24], Tasso *et al.* implemented Ti's attack and additionally proposed a countermeasure to it. Essentially, they propose to check whether the reconstructed A-coefficient from the computed public key $\text{pk}_2 = (P', Q', R)$ equals the last A-coefficient in `keygen` (output of line 15 of 1). If there is equality then Bob ensures that $R = (P' - Q')$ and returns pk_2; otherwise, an attack is detected.

Nevertheless, the new attack in this paper and the ones from [26] and [29] use valid public keys such that $R = (P' - Q')$ always holds, and thus our attack

is completely immune to Tasso *et al.*'s countermeasure and any other variant of it.

From the mechanism of our attack, each instance given by Equation 4 is on an intermediate curve in Bob's secret 3-isogeny chain. Also, the attack is not limited to use exclusively the curve $E\colon y^2 = x^3 + 6x^2 + x$, we only require a Montgomery curve with affine A-coefficient in $\mathbb{F}_p$.

Naively, to avoid the attack, Bob should check if any of its intermediate secret 3-isogeny computations returns an A-coefficient in $\mathbb{F}_p$, if this happens then Bob would reject Alice's public key. However, such a countermeasure opens the door to another attack:

1. The attacker makes a first guess with $s_0 = 0$
2. Sends the corresponding pk_2 [8] to Bob
3. If Bob rejects pk_2, then she knows her guess was correct;
4. If Bob accepts pk_2, then she make another guess with $s_0 = 1$
5. Sends a different public key pk_2' [8] to Bob
6. If Bob rejects pk_2', then she knows $s_0 = 1$, otherwise, $s_0 = 2$
7. Repeat the same attack flow to recover $s_1, \ldots, s_{e_3 - 2}$.

Remarkably, the attack outlined above does not rely on fault injections, but only on oracle calls. This makes it more dramatic than the attack that Bob tries to avoid in the first place.

Since Lemma 1 characterizes all curves defined over $\mathbb{F}_p$ regardless of whether the affine curve coefficients belong to $\mathbb{F}_p$, randomizing the curve coefficients (assuming multiplying by a random $\mathbb{F}_{p^2}$-element) will not help. The same goes for randomizing points. In other words, all Bob's computations operate with projective points and curves defined over $\mathbb{F}_{p^2} \setminus \mathbb{F}_p$, and then Bob cannot be possibly aware that he is processing $\mathbb{F}_p$-curves. Therefore, the attack is not affecting the constant-time nature of SIDH/SIKE implementations.

A plausible countermeasure consist of using a commutative diagram by pushing forward Bob's 3^{e_3}-isogeny $\psi_B : E_A \to E_{AB}$ through a random isogeny $\rho : E_A \to E_A'$ of degree not divisible by 3 and obtain an isogeny $\psi_B' : E_A' \to E_{AB}'$, then push forward the dual of ρ through ψ_B' to finally obtain $\hat{\rho}' : E_{AB}' \to E_{AB}$.

For instance, Bob can choose ρ to be of degree 2^k-isogenies, for some sufficiently large k. To be more precise, Bob would

1. sample an arbitrary order-2^k point $R \in E_A$;
2. construct $\rho : E_A \to E_A'$ of kernel R;
3. push Alice's public key pk_2 and a point D through ρ, $\rho(D)$ generating $\hat{\rho}$;
4. compute $\psi_B' : E_A' \to E_{AB}'$ using $\rho(\text{pk}_2)$, push $\rho(D)$ through ψ_B' to get D';
5. finally, compute the codomain curve E_{AB} of the isogeny $\hat{\rho}'$ of kernel D'.

This randomized countermeasure increases the number of queries, $0.53 \log_2(p)$, of our attack by a factor of about 2^k, suggesting large (enough) values for k since, asymptotically speaking, it remains the same complexity when k is small compared with e_2. Consequently, a secure countermeasure with, for example, $k = \frac{e_2}{2}$

---

[8] pk_2 and pk_2' are computed by means of Equation 4.

would give a 2x of slowdown of Bob's computations, while the more conservative choice of $k = e_2$ would be 3x slower.

Discussion about the feasibility of our attack. As we have seen, the attack presented in this paper requires injecting faults in a precise step of the SIDH strategy evaluation procedure. More concretely, the attacker must inject zeroes in the imaginary part of the coefficient A_{i+1} at iteration $i + 1$ of line 9 of 1.

But for a real implementation of SIDH and SIKE, how feasible/realistic would be to inject such faults?

In [24] the authors report that electromagnetic injection is considerably difficult to synchronize with the execution of SIKE. Because of this difficulty, the real-scenario attack implemented by the authors achieved a rather modest experimental success rate of just 0.62%, which is relatively inefficient compared with the expected theoretical 50% success rate attributed to Ti's attack [24, Remark 7].

One advantage that Ti's attack offers, is that the perturbations do not need to be placed in a precise step of Key Generation, provided that the faults are injected before the victim starts computing and evaluating isogenies. Moreover, with high probability, it suffices that the torsion points are perturbed by flipping any single bit of the points.

Due to the high timing synchronization required by our attack, its instantiation using electromagnetic injection would appear to be just too difficult for being successfully launched in a real implementation of SIDH.

On the other hand, software oriented fault attacks appear to be more promising for our attack.

For example, the attack presented in [2] was executed on a ChipWhisperer-Lite board equipped with a 32-bit STM32F303 ARM Cortex-M4 processor, using the SIKEp434 Cortex-M4 implementation of [16]. Under the assumption that the attacker knows the exact code locations where the faults must be injected, the authors reports close to 100% of efficiency in their attack. The faults were injected by clock glitching, which forced the ChipWhisperer card to skip an instruction. In [29], the authors report a instruction-skipping generic fault attack that targets all NIST PQC Round 3 KEM candidates including SIKE. The attack was executed on a ChipWhisperer cw308 UFO base-board, which permits fault-injection attacks using clock glitching.

It is conceivable that we can adapt the attacks of [2,29] to our setting, but in our case we should rather focus on perturbing the memory locations where the imaginary parts of the coefficient A are stored.

5   Concluding remarks

In this paper, we presented a new vulnerability based on deciding whether Bob's 3^{e_3}-isogeny computation passes through a curve with A-coefficient in $\mathbb{F}_p$. This way we propose a new fault-injection attack that

- belongs to the same family of [11], known sometimes as *reaction attacks* [19],

- has similar query complexity as the attacks in [29] and [11],
- it is different to mitigate than the one previous SIDH fault attack, known as Ti's attack [25],
- can be combined with the side-channel attack in [26] to achieve full key recovery in SIKE.

From the description of the proposed attack, one can easily see that the best scenarios where it applies are when the individual isogeny degree is as small as possible, namely 2 or 3. We considered 3 in this work because the computed isogenies in SIDH/SIKE are of degree 3 and (almost always) 4. We believe our fault-injection attack can be transformed into a side-channel attack using the techniques of [26] and trying to force working with purely $\mathbb{F}_p$ values, but this appears to be a formidable task that we leave here as future work.

Now, when the individual isogenies have higher (prime) degrees, as in B-SIDH, the attack is unlikely to work as efficiently as 3-isogenies due to the significantly smaller proportion of $\mathbb{F}_p$-isogenies (2 $\mathbb{F}_p$-isogenies in average). However, the number of queries remains a polynomial factor multiplying the isogeny degree. Since in B-SIDH, for instance, the prime isogeny degrees are bounded (by at most 2^{16}) so that the isogenies are efficiently computable, reaching this number of queries should not be a big concern.

To the best of our knowledge, there is no previous attack in which the attacker can inject such amount of zeros in memory (or registers) as we require our attacker to do. We consider finding alternative methods to perform this injection in a controlled way as future work. We would also like to remark that the attack's timing constraints are similar to those in the loop-abort attack presented by Gélin and Wesolowski in 2017 [13]. In addition, we believe our contribution opens a new path of attacks not considered before. It might be later implemented with the appropriate tools as it happened previously in the theoretical attack by Ti in 2017, recently implemented in 2021 by Tasso et al [24].

References

1. Azarderakhsh, R., Campagna, M., Costello, C., Feo, L.D., Hess, B., Jalali, A., Jao, D., Koziel, B., LaMacchia, B., Longa, P., Naehrig, M., Pereira, G., Renes, J., Soukharev, V., Urbanik, D.: Supersingular isogeny key encapsulation. third round candidate of the nist's post-quantum cryptography standardization process (2020), available at: https://sike.org/
2. Campos, F., Krämer, J., Müller, M.: Safe-error attacks on SIKE and CSIDH (2021), https://eprint.iacr.org/2021/1132
3. Charles, D.X., Lauter, K.E., Goren, E.Z.: Cryptographic hash functions from expander graphs. J. Cryptol. **22**(1), 93–113 (2009). https://doi.org/10.1007/s00145-007-9002-x, https://doi.org/10.1007/s00145-007-9002-x

4. Costello, C.: The case for SIKE: A decade of the supersingular isogeny problem. IACR Cryptol. ePrint Arch. p. 543 (2021), https://eprint.iacr.org/2021/543

5. Costello, C., Hisil, H.: A simple and compact algorithm for SIDH with arbitrary degree isogenies. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology - ASIACRYPT 2017 Part II. LNCS, vol. 10625, pp. 303–329. Springer (2017)

6. Costello, C., Smith, B.: Montgomery curves and their arithmetic - the case of large characteristic fields. J. Cryptogr. Eng. **8**(3), 227–240 (2018). https://doi.org/10.1007/s13389-017-0157-6, https://doi.org/10.1007/s13389-017-0157-6

7. Couveignes, J.M.: Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291 (2006), http://eprint.iacr.org/2006/291

8. Feo, L.D., Jao, D., Plût, J.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. J. Mathematical Cryptology **8**(3), 209–247 (2014)

9. Feo, L.D., Mrabet, N.E., Genêt, A., Kaluđerović, N., de Guertechin, N.L., Pontié, S., Élise Tasso: Sike channels. Cryptology ePrint Archive, Report 2022/054 (2022), https://ia.cr/2022/054

10. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Advances in Cryptology - CRYPTO. LNCS, vol. 1666, pp. 537–554. Springer (1999)

11. Galbraith, S.D., Petit, C., Shani, B., Ti, Y.B.: On the security of supersingular isogeny cryptosystems. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology - ASIACRYPT 2016 Part I. LNCS, vol. 10031, pp. 63–91 (2016)

12. Galbraith, S.D., Vercauteren, F.: Computational problems in supersingular elliptic curve isogenies. Quantum Inf. Process. **17**(10), 265 (2018)

13. Gélin, A., Wesolowski, B.: Loop-abort faults on supersingular isogeny cryptosystems. In: Lange, T., Takagi, T. (eds.) Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10346, pp. 93–106. Springer (2017). https://doi.org/10.1007/978-3-319-59879-6_6, https://doi.org/10.1007/978-3-319-59879-6_6

14. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the fujisaki-okamoto transformation. In: Theory of Cryptography - 15th International Conference, TCC. LNCS, vol. 10677, pp. 341–371. Springer (2017)

15. Jao, D., Feo, L.D.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In: Yang, B. (ed.) Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011. LNCS, vol. 7071, pp. 19–34. Springer (2011)

16. Kannwischer, M.J., Rijneveld, J., Schwabe, P., Stoffelen, K.: pqm4: Testing and benchmarking nist pqc on arm cortex-m4. Cryptology ePrint Archive, Report 2019/844 (2019), https://ia.cr/2019/844

17. NIST: NIST Post-Quantum Cryptography Standardization Process. Second Round Candidates (2017), available at: https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions

18. Petit, C.: Faster algorithms for isogeny problems using torsion point images. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology - ASIACRYPT 2017, Proceedings, Part II. LNCS, vol. 10625, pp. 330–353. Springer (2017)

19. de Quehen, V., Kutas, P., Leonardi, C., Martindale, C., Panny, L., Petit, C., Stange, K.E.: Improved torsion-point attacks on SIDH variants. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology - CRYPTO 2021, Proceedings, Part III. LNCS, vol. 12827, pp. 432–470. Springer (2021)

20. Renes, J.: Computing isogenies between montgomery curves using the action of (0, 0). In: Lange, T., Steinwandt, R. (eds.) Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018, Fort Lauderdale, FL, USA, April 9-11, 2018, Proceedings. LNCS, vol. 10786, pp. 229–247. Springer (2018)

21. Rostovtsev, A., Stolbunov, A.: Public-key cryptosystem based on isogenies. IACR Cryptology ePrint Archive **2006**, 145 (2006)

22. Smith, B.: Pre- and post-quantum Diffie–Hellman from groups, actions, and isogenies. In: Budaghyan, L., Rodríguez-Henríquez, F. (eds.) Arithmetic of Finite Fields - WAIFI 2018. LNCS, vol. 11321, pp. 3–40. Springer (2018)

23. Stolbunov, A.: Constructing public-key cryptographic schemes based on class group action on a set of isogenous elliptic curves. Advances in Mathematics of Communications **4**(2), 215–235 (2010)

24. Tasso, É., Feo, L.D., Mrabet, N.E., Pontié, S.: Resistance of isogeny-based cryptographic implementations to a fault attack **12910**, 255–276 (2021). https://doi.org/10.1007/978-3-030-89915-8_12, https://doi.org/10.1007/978-3-030-89915-8_12

25. Ti, Y.B.: Fault attack on supersingular isogeny cryptosystems. In: Lange, T., Takagi, T. (eds.) Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings. LNCS, vol. 10346, pp. 107–122. Springer (2017)

26. Ueno, R., Xagawa, K., Tanaka, Y., Ito, A., Takahashi, J., Homma, N.: Curse of re-encryption: A generic power/em analysis on post-quantum kems. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2022**(1), 296–322 (2022). https://doi.org/10.46586/tches.v2022.i1.296-322, https://doi.org/10.46586/tches.v2022.i1.296-322

27. Vélu, J.: Isogénies entre courbes elliptiques. Comptes Rendus de l'Academie des Sciences de Paris **Serie A 273**, A238–A241 (1971), english translation by A. Ghitza available at https://aghitza.github.io/publication/translation_velu/

28. Washington, L.C.: Elliptic Curves: Number Theory and Cryptography, Second Edition. Chapman & Hall/CRC, 2 edn. (2008)

29. Xagawa, K., Ito, A., Ueno, R., Takahashi, J., Homma, N.: Fault-injection attacks against nist's post-quantum cryptography round 3 KEM candidates **13091**, 33–61 (2021). https://doi.org/10.1007/978-3-030-92075-3_2, https://doi.org/10.1007/978-3-030-92075-3_2