Fault Injection Attacks on Torsion Masking

Valerie Gilchrist[1,4], Yi-Fu Lai[2,4], Michael Meyer[3,4]

[1] Université Libre de Bruxelles, Belgium
[2] Shanghai Jiao Tong University, China
[3] University of Regensburg, Germany
[4] Theoretical Sciences Visiting Program, Okinawa Institute of Science and Technology Graduate University, Japan

Abstract. In 2022, a string of attacks on SIDH was released that made use of the now infamous Kani's Lemma. Since then, several new and exciting isogeny-based protocols have emerged that both avoid the attacks, while at the same time, leverage the power of Kani's Lemma to improve their efficiency. One common technique to do so has been the inclusion of masked torsion points. This is when a protocol publishes information about how a secret isogeny acts on a large torsion subgroup, but masks the exact image points by multiplying them by some secret scalars.
In this work, we present the first analysis of the physical security of the masked torsion point technique. We provide fault injection attacks on the signature scheme PRISM, and the public-key encryption schemes POKÉ and FESTA. Our fault model is standard in the literature, and can be implemented inexpensively. Most notably, three of the five attacks presented only require one or two first-order faults, which is significantly less than what is needed to attack other isogeny-based cryptosystems that do not use torsion masking.

1 Introduction

The field of isogeny-based cryptography has emerged as an interesting option with which to build post-quantum cryptography, in part due to its typically small storage requirements. Only a few years ago, the most popular isogeny-based schemes were the key exchange SIDH and its KEM version SIKE [36], the key exchange CSIDH [20], and the signature scheme SQIsign [1]. SIKE was a promising candidate in NIST's standardization effort for post-quantum cryptographic KEMs, advancing to the final fourth round. However, during this round,

* Authors listed in alphabetical order: see https://www.ams.org/profession/leaders/CultureStatement04.pdf. This research was initiated while visiting the Okinawa Institute of Science and Technology (OIST) through the Theoretical Sciences Visiting Program (TSVP), the Thematic Program on Isogeny-based Cryptography (TP25IC). This work is supported by a FRIA grant by the National Fund for Scientific Research (F.N.R.S.) of Belgium and by the German Federal Ministry of Research, Technology and Space (BMFTR) under the project QUORYPTAN (16KIS2034). Date of this document: 2026-09-09.

a series of attacks [18,43,46] broke SIKE due to its use of public torsion point information. CSIDH did not participate in the NIST standardization process, but was considered very promising due to its non-interactive and commutative structure, allowing for many applications. Despite not being susceptible to the SIKE attacks, the uncertainty regarding its exact quantum security led to increased parameter sizes, and in turn, to worse performance that seems too slow for practical applications in spite of recent optimizations [13,14]. This left SQIsign as the only isogeny-based scheme that could be considered practical.

Though the attacks on SIKE initially appeared to be devastating to the field, they were quickly employed in constructive applications as well. This allowed for the development of a new generation of isogeny-based schemes: the public key encryption schemes M-SIDH [29], FESTA [8], QFESTA [44], and POKÉ [7,11], the hash-and-sign signature scheme PRISM [4,5], and others. These protocols are among the most prominent in the field, and continue to improve upon the commonly criticized efficiency of isogeny-related computations. They also provide important diversification of hardness assumptions to the ongoing development of post-quantum cryptographic schemes. Importantly, a few of the abovementioned schemes have been submitted to the Institute of Commercial Cryptography Standards' NGCC program [35].

An important topic that has not yet been addressed so far for any of these schemes is implementation security, in particular their resistance to physical attacks. In contrast, SIKE and CSIDH were targeted by several works on both passive side-channel attacks (e.g., [27,17,34]) and active fault attacks (e.g., [48,16,3]). Similarly, SQIsign's implementation security has also been studied [40,6]. Due to the increased complexity of its code base, this is an important direction of research that has only recently been initiated. So far none of the SIKE, CSIDH, or SQIsign analyses on physical security affect the new-generation schemes mentioned above.

We address this gap by analyzing fault attacks that target a common design subroutine that all the schemes mentioned share: *torsion point masking*. This shared feature is not coincidental; rather, it is a key ingredient developed after the SIDH attacks either for preventing SIDH-type attacks or for exploiting the underlying theory, Kani's Lemma, as a trapdoor to achieve efficient isogeny evaluation. However, despite its widespread and central use, comparatively little is known about the practical implementation security of this subroutine.

Masked torsion. Recall that in SIDH, Alice computes a secret isogeny $\varphi : E_0 \to E'$, and publishes E' and the image $\varphi(P), \varphi(Q)$ of a certain public torsion basis P, Q, as her public key, where the order N of P, Q is coprime to the degree of φ. Publishing these images of public points under the secret isogeny led to the series of attacks that are able to recover the secret key under certain mild assumptions on the point orders and isogeny degree. The main ingredient for these attacks is Kani's Lemma, see Section 2.2.

Many schemes that were published after the SIDH attacks still require publicly available information about torsion points. A popular technique to achieve

this while avoiding attacks based on Kani's Lemma is torsion point *masking*. Using the same notation as above, we want to avoid publishing the torsion point images $\varphi(P), \varphi(Q)$ under the secret isogeny φ. Instead, we mask these points through the action of an invertible 2×2 matrix M with entries in $\mathbb{Z}_N := \mathbb{Z}/N\mathbb{Z}$. In particular, when writing

$$M = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix}$$

with $m_{ij} \in \mathbb{Z}_N$, this action corresponds to computing

$$\begin{pmatrix} R \\ S \end{pmatrix} = M \begin{pmatrix} \varphi(P) \\ \varphi(Q) \end{pmatrix} = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{pmatrix} \varphi(P) \\ \varphi(Q) \end{pmatrix} = \begin{pmatrix} [m_{11}]\varphi(P) + [m_{12}]\varphi(Q) \\ [m_{21}]\varphi(P) + [m_{22}]\varphi(Q) \end{pmatrix}.$$

Depending on the exact application, we can choose M from a certain group of matrices $\Gamma \leq \mathrm{GL}_2(\mathbb{Z}_N)$. For a general security analysis for different choices of Γ, we refer to [25].

Contributions. We propose several first-order fault attacks that target torsion point masking in different protocols. The general idea is to fault one of the scalar multiplications during the masking instruction. Since the points R and S are published as part of the signature or ciphertext in the schemes we consider, we can analyze the effect of the fault through pairings, allowing to recover the masking coefficients m_{ij}. In the final step, we can unmask the torsion points and use Kani's Lemma to recover an efficient representation for the secret key or to recover plaintexts, respectively.

Our analysis on masked torsion affects some of the most prominent isogeny-based schemes, in particular, those that store the secret masking scalars within their secret keys. Our contributions can be summarized as follows.

- Using only one or two first-order faults, we provide a key-recovery attack on PRISM [4], and message-recovery attacks on POKÉ [7] and FESTA [8]. With access to repeated first-order faults, we describe key-recovery attacks on static-key variants of the latter two as well. These attacks are listed in Table 1. In addition, the attack presented on POKÉ directly applies to the public key encryption schemes PIKE [12] and INKE [38] as well.
- Our attacks are amenable to different fault models ranging in feasibility of implementation, but all of which are considered reasonable and have been used frequently in the fault attack literature. Specifically, our attacks could be run with flexible levels of precision, e.g., with one or multiple bit flips, one or multiple bits being set to 1, or setting a byte to a random value. As such, we leave our analysis modular so that the different fault models can be easily plugged-in. This is seen in Table 1 with the use of the variable $\#\Delta \in [1, 2^8]$, which depends on the specific fault model being used.
- We provide proof-of-concept simulations of the attacks on the reference implementations of PRISM, FESTA and POKE.

Table 1. The table describes the number of faults necessary to implement each of the attacks presented in this work (when implemented targeting $\lambda = 128$ bits of security). These values are independent of the probability of success for the varying fault injection models. The final column lists the new secret space size after the injected faults and post-processing, where $\#\Delta$ varies between 1 and 2^8 depending on the attack models.

	Protocol	Attack	# of faults	Effective secret space size
This work	PRISM	key-recovery (Alg. 2)	2	$4(\#\Delta)^2 \in [2^2, 2^{18}]$
	POKÉ	message-recovery (Alg. 5)	1	$\#\Delta/2 \in [1, 2^7]$
		key-recovery	128	1
	FESTA	message-recovery (Alg. 7)	1	$\#\Delta/2 \in [1, 2^7]$
		key-recovery	632	1

– We discuss countermeasures that can be applied to protect the torsion point masking subroutine against first-order faults in any of these schemes. In addition to the simple classical countermeasures of adding redundancy in the computations, we propose a validation method for masked torsion points that has comparably low cost overhead.

Torsion masking is often employed in protocols as a means of adding randomness. This string of fault attacks serves to remind the community that this randomness frequently has some underlying structure that may leave it vulnerable to attack. Following this, our work shows that in the case of torsion masking, implementations need to apply countermeasures to safeguard against practical fault attacks that recover secret keys or plaintext messages.

Relation to fault attacks on classical ECC. At first glance, this problem setting may seem related to fault attacks on classical elliptic-curve cryptography (ECC): we essentially target scalar multiplication of points on elliptic curves, which is the main operation in ECC. However, the setting in our masked torsion attacks significantly differs from the ECC scenario.

In ECC, we fix an elliptic curve E over $\mathbb{F}_p$ and a public base point P, whose order is known. This point is multiplied by a secret scalar m, and the resulting point $[m]P$ is published. We summarize this as follows, where publicly known data is denoted in blue, and secret data is denoted in red.

$$P \xmapsto{[m]} [m]P$$

In our setting, the points $P, Q \in E$ and their order are known, but only their images under the secret isogeny φ are multiplied by the scalars determined by M. That is, in contrast to the ECC setting, the base points of the scalar multiplications are secret. Furthermore, if M is not a diagonal matrix, only sums of scalar multiples of $\varphi(P)$ and $\varphi(Q)$ are published, which further complicates

the setting. We summarize this below, using the same colouring as above.

$$\begin{pmatrix} P \\ Q \end{pmatrix} \xmapsto{\varphi} \begin{pmatrix} \varphi(P) \\ \varphi(Q) \end{pmatrix} \xmapsto{M} \begin{pmatrix} [m_{11}]\varphi(P) + [m_{12}]\varphi(Q) \\ [m_{21}]\varphi(P) + [m_{22}]\varphi(Q) \end{pmatrix}$$

In summary, our scenario significantly differs from the situation in ECC. Although it is possible to use similar fault injections as for the scalar multiplication in ECC, it is not obvious if this reveals any information on the secret unmasked torsion points $\varphi(P), \varphi(Q)$ or the secret isogeny φ. We answer this in the affirmative by providing a powerful fault attack on different isogeny schemes with torsion point masking that recovers secret keys or plaintexts, while requiring only one or two samples of first-order faults, respectively.

Related work. While there is no prior work on the implementation security of torsion point masking or the schemes we consider, there are several analyses of fault attacks on isogeny-based schemes, with a focus on CSIDH and SIKE. These works are listed in Table 2.

CSIDH. Fault attacks on CSIDH significantly differ from the attacks presented in this work, mainly due to the different structure of the CSIDH algorithm. In most cases, these fault attacks target dummy operations, and obtain an oracle from observing if the output changes after a fault, i.e., if the fault hit a necessary or dummy operation [15,16,41]. These fault attacks usually use a fault model that is easy to achieve in practice, since it is not required to be very precise, and can include instruction skips, faulting memory, or aborting loops. On the other hand, these attacks require many samples of faulty runs, usually at least in the hundreds. Similarly, a fault attack on the point sampling in CSIDH has similar requirements on the fault model, but requires even more faulty samples [3].

SIKE. Fault attacks on SIKE, published prior to the 2022 attacks, include a larger variety of attacks and fault models. For instance, the attack from [2] uses a fault model that injects zeros into a particular curve coefficient; however, zeroing a full $\mathbb{F}_p$ variable is considered difficult in practice.

Other works use a fault model that is more similar to our model: the authors of [16] use memory faults, such as bit flips or random byte assignments, to target the scalar multiplication for computing the secret kernel generator during the key agreement protocol in SIKE. It retrieves the secret scalar one bit at a time, thereby requiring several rounds of fault injections.[5] A similar model is presented in [32], using loop-abort faults during the isogeny computation. This can also be implemented using memory faults, targeting either the loop index in a precise way, or targeting a specific iteration of the loop, requiring some precision.

In [49], the authors investigate fault attacks on SIDH assuming that the victim holds a static public key and secret isogeny φ_A. The adversary is assumed

[5] We show that similar attacks are possible in the context of POKÉ (see Section 5.3) and FESTA (see Section 6.3), although our focus will be attacks that require only one round of fault injection to recover plaintext messages.

to be able to trigger repeated executions of the key generation procedure, and fault injects during the torsion-point evaluation subroutine in key generation. This enables the attacker to obtain the image $\varphi_A(P)$ of a chosen torsion point P, which suffices to recover the secret isogeny. In our work, we do not consider targeting key generation protocols as it is a particularly strong attack model for a fault attack. Thus, [49,33] are excluded from our comparisons in Table 2.

SQIsign. The analysis of the physical security of SQIsign has recently received more attention. In [40] two fault-based key-recovery attacks on SQIsign1.0 [22] are presented and implemented with the algorithms from [26]. Two adversarial capabilities considered are: (i) randomizing the generator used in the commitment-ideal construction, and (ii) inducing an early-abort during the commitment $\ell_i^{e_i}$-isogeny computation for a specified i among a few tens of candidates. It additionally assumes SQIsign to be deterministic. In the first model, a constant number of faults (typically one) can cause the commitment ideal to collapse to $\mathcal{O}_0$, yielding a signature that directly reveals an isogeny between E_0 and E_{pk}, which constitutes a secret key. In the second model, the fault skips the $\ell_i^{e_i}$-isogeny computation. Although the remaining commitment isogeny still has cryptographically large degree, the attack succeeds when SQIsign is assumed to be deterministic: multiple faulty executions together with one correct signature allow the adversary to reconstruct the secret via brute-force matching and divide-and-conquer strategy.

Note that the attacks presented in this work differ greatly from previous work in that the target algorithms being faulted are not the same. Our approach completely avoids the direct recovery of secret isogenies, and only targets the masking subroutine, thereby achieving the attack with fewer faults overall. The fact that our attacks require less faults indicates that isogeny-based cryptosystems relying on torsion masking for their security are more vulnerable to hardware attacks than other systems such as CSIDH, SIKE, and SQIsign.

Organization. We begin with an overview of the different isogeny representations that will be relevant to this work in Section 2. We discuss the fault model we will be using and its feasibility in Section 3. In Section 4, we present a key-recovery attack on PRISM; in Section 5 we present both message-recovery and key-recovery attacks on POKÉ; in Section 6 we also present message-recovery and key-recovery attacks on FESTA. Section 7 describes the proof-of-concept simulations of the attacks. In Section 8 we discuss possible countermeasures to our attacks, and make concluding remarks in Section 9.

Acknowledgements. The authors would like to thank the reviewers for their careful feedback, and in particular for suggestions that lead to the addition of Appendix B. They would also like to thank Fabio Campos and Riccardo Invernizzi for helpful discussion on an earlier draft of this work.

Table 2. The table describes the number of faults necessary to implement each attack described in **Related work** (when implemented targeting $\lambda = 128$ bits of security). These values are independent of the probability of success for the varying fault injection models. All of the faults are first-order faults, except when indicated otherwise. The final column lists the new secret space size after the injected faults and post-processing.

	Protocol	Attack	# of faults	Effective secret space size
Related work	CSIDH	key-recovery in [15, Attack 3]	178	$x \in [1, 2^{67}]$
		key-recovery in [3, Sec. 5.1]	512	1
	SIKE	key-recovery in [2, Table 2]	226	1
		key-recovery in [16, Sec. 5.1]	1090	1
		key-recovery in [32, Sec. 4.3]	434	1
		key-recovery in [32, Sec. 4.4]	19	2^{20}
	SQISign	key-recovery in [40, Sec. 4.1]	27	$2^{15.7}$
		key-recovery in [40, Sec. 4.2]	1 (2nd order)	$2^{111.6}$

2 Isogeny Preliminaries

An isogeny can be represented in different ways, be it by its kernel, some interpolation data, or otherwise. In this section we recall different representations for isogenies that appear in the literature.

Notation. For the rest of this work, unless otherwise stated, p, q are primes, and a, b are positive integers. The notation $\hat{\phi}$ refers to the dual isogeny of the isogeny ϕ. The degree of ϕ is denoted by $\deg \phi$, and its kernel by $\ker \phi$.

2.1 Classic representations

Isogenies are non-constant morphisms between elliptic curves that fix the point at infinity. In order to use these maps in cryptographic applications, we require some way of writing them down. In practice, we require an algorithm that can evaluate points efficiently and a representation that is compact to write down. One of the most famous representations of a (separable) isogeny $\varphi : E_1 \to E_2$ uses rational functions f, g in two variables. In this way, given an affine point $P = (x, y) \in E_1$, we may compute $\varphi(P) = (f(x, y), g(x, y))$ explicitly. Unfortunately, as the degree of φ increases, it becomes increasingly difficult to write down the functions f, g, let alone evaluate points using them. Thus, we require other approaches for representing isogenies.

Another very common representation for separable, cyclic isogenies is the *kernel generator representation*. It requires a generating point of the kernel subgroup (or several points). Vélu [50] provides explicit formulae that can evaluate elliptic curve points at an isogeny given the kernel subgroup or kernel generator. It even provides a way of obtaining the rational functions from the previous representation given the kernel generator representation. This representation is

compact so long as the kernel generator is defined over a field extension of small degree, but unfortunately, the evaluation algorithm is efficient only when the degree of the isogeny is smooth, since the complexity is dominated by its largest prime factor.

In recent years, it has become increasingly more common to use the *ideal representation* of isogenies. This representation uses ideals from a quaternion algebra, chosen under the Deuring correspondence. It will not be relevant to this work, however, so we exclude any further description of it. This one and several more representations are described in the survey from Robert [47].

2.2 HD representation

In 2022, a series of attacks [18,43,46] on the isogeny-based key exchange scheme SIDH were released. The attacks leverage Kani's Lemma about isogeny diamonds to recover an isogeny between two elliptic curves, given its degree and knowledge of how it acts on a sufficiently large torsion subgroup. Since then, this style of attack has been used in several constructive applications. Most notably, it provides a new representation for isogenies. We give a brief overview of the attack before describing how it can be used to represent isogenies. For more detailed descriptions, see the original attacks [18,43,46].

Kani attack. Recall, SIDH used a commutative diagram, Kani calls these configurations *isogeny diamonds*, and in [37] gives the following construction.

Theorem 1 (Kani's Lemma (informal)). *Given an isogeny diamond of* $\phi : E_0 \to E_1, \psi : E_0 \to E_2$ *where* $\gcd(\deg\phi, \deg\psi) = 1$*, the isogeny* $F : E_0 \times E_3 \to E_1 \times E_2$*, given by*

$$F\begin{pmatrix} X \\ Y \end{pmatrix} := \begin{pmatrix} \phi & \hat{\psi}' \\ -\psi & \hat{\phi}' \end{pmatrix}\begin{pmatrix} X \\ Y \end{pmatrix} \text{ for } (X, Y) \in E_0 \times E_3$$

has kernel

$$\ker F = \{(\hat{\phi}(P), \psi'(P)) : P \in E_1[\deg\phi + \deg\psi]\}.$$

Thus, by replacing ψ with a strategically chosen, known isogeny, and with knowledge of how ϕ acts on a sufficiently large torsion subgroup, one can construct the 2-dimensional isogeny F, and recover an evaluation algorithm for ϕ from its kernel.

Representing isogenies. With the work summarized in [19], as was pointed out in [47, Thm. 5.1], we can make use of the following theorem to define a new representation of isogenies, namely the *HD representation*.

Theorem 2 (HD representation). *Let* $\varphi : E_1 \to E_2$ *be an* n*-isogeny (with* n *coprime to the characteristic of the base field). Let* G *be a subgroup of* E_1 *of order at least* $4n+1$*, and* $(P_1, \ldots, P_r)$ *generators of* G*. Given a point* $P \in E_1(\mathbb{F}_{q'})$ *and*

the interpolation data $(P_i, \varphi(P_i))$*, there is an algorithm to evaluate* $\varphi(P)$ *in time polynomial in* $\log n, r, \log q$*, the largest prime factor of* n*, the extension degrees of the field of definition of* P *and the points* P_i*, and the extension degree of the points of* $\ell^{\lfloor e/2 \rfloor}$*-torsion for each prime power* $\ell^e \mid \#G$*.*

The efficient evaluation algorithm described in the theorem is achieved by embedding the isogeny φ into a higher dimensional isogeny between abelian varieties, as was done in the Kani attack. The exact runtime of the algorithm depends on the dimension necessary to embed the isogeny (either 2, 4, or 8), the largest prime factor of the torsion order (which will often be smooth in our applications), and the field of definition of the torsion points.

Since all of the attacks described in this work will require relatively few isogeny evaluations in HD representations, we need only be assured that these computations can be performed within a reasonable amount of time. Current implementations of constructive applications give a rough estimate of the runtimes we can also expect for our attack implementations:

- In PRISM [4], the authors provide an implementation that includes an evaluation computation for cryptographically sized isogenies in dimension 2 of non-smooth degree. The runtime was around 500 milliseconds. This implementation will be used in the simulation provided in Section 7.
- In [24], Dartois provides new algorithms for computing dimension-4 isogenies. The timings range between 164-2488 milliseconds, depending on the size of the base field and degree of the isogeny (a power of 3 or 7).

In summary, under this representation, $\varphi : E_1 \to E_2$ is given by

$$\varphi = \big(\deg \varphi, (P_1, \dots P_r), (\varphi(P_1), \dots \varphi(P_r))\big),$$

where the P_i generate a torsion subgroup of order $N > 4 \cdot \deg \varphi + 1$. Further, there exists an efficient evaluation algorithm for the isogeny that runs in polynomial time, and for which there exist practical implementations whose run time is in the order of milliseconds. These runtimes will be more than sufficient for our cryptanalytic purposes.

3 Fault attack model

In this section, we describe the fault model for our attacks. We first outline the aim of our fault injection, and discuss ways to achieve this in practice.

3.1 Aim of our fault injections

The main idea of our fault attacks is to inject a fault in the scalar multiplication to compute the masked torsion points

$$\begin{pmatrix} R \\ S \end{pmatrix} = M \begin{pmatrix} \varphi(P) \\ \varphi(Q) \end{pmatrix} = \begin{pmatrix} [m_{11}]\varphi(P) + [m_{12}]\varphi(Q) \\ [m_{21}]\varphi(P) + [m_{22}]\varphi(Q) \end{pmatrix}.$$

Since these points are published in our applications, we aim to obtain a point $R' \neq R$ and/or $S' \neq S$, which we use for post-processing to obtain secret data, such as secret keys or plaintext messages. The execution of fault attacks depends on the shape of M.

Diagonal matrices M. In the case of a diagonal matrix M, the above equation simplifies to $R = [m_{11}]\varphi(P)$, and similarly for S. In our attacks, we aim to inject a fault that changes the value of the variable m_{11} to some $m'_{11} \neq m_{11}$, where neither of these values needs to be known. However, we require that the number of possible values of $\Delta = m_{11} - m'_{11}$ is bounded by a small number $\#\Delta$, for instance 2^8, since the post-processing of our attacks needs to be able to search through all possible values for Δ.

There are several different options for fault injections to achieve this. Efficient implementations usually use ladder algorithms for scalar multiplication, e.g., the Montgomery ladder [23, Alg. 8], and represent points in x-only coordinates. These ladder algorithms process m_{11} bit by bit, where the i-th step performs a differential addition and doubling, and point swaps depending on the value of the i-th bit of m_{11}. There are various options for a fault injection that results in the output of $[m'_{11}]P$ such that $\#\Delta$ is small. This includes tampering with certain bits or bytes of m_{11} in memory, skipping loop iterations, or loop aborts in the ladder algorithm. We discuss the practicality of these approaches in Section 3.2.

Non-diagonal matrices M. In this case, our aim will be similar: for instance, instead of $R = [m_{11}]\varphi(P) + [m_{12}]\varphi(Q)$, we want to obtain $R' = [m'_{11}]\varphi(P) + [m_{12}]\varphi(Q)$ with $m'_{11} \neq m_{11}$. Then $R - R' = [m_{11} - m'_{11}]\varphi(P)$, and as above we require that the number $\#\Delta$ of possible values for $\Delta = m_{11} - m'_{11}$ is small.

In the case of $m_{11}, m_{12} \neq 0$, efficient implementations usually use a so-called biscalar multiplication—a variant of the ladder algorithm that directly computes $[m_{11}]\varphi(P) + [m_{12}]\varphi(Q)$ without separately computing $[m_{11}]\varphi(P)$ and $[m_{12}]\varphi(Q)$ first [23, Alg. 9]. This means that skipping loop iterations or loop abort faults are not suitable for our purpose, since these attacks would result in some output $[m'_{11}]\varphi(P) + [m'_{12}]\varphi(Q)$ instead of $[m'_{11}]\varphi(P) + [m_{12}]\varphi(Q)$ as required in our attacks. Thus, in this scenario we require using memory faults to alter some bits or a byte of m_{11} in memory before the biscalar ladder is called.

For upper or lower triangular matrices, we can use either of the two approaches above, depending on which scalar multiplication we attempt to fault.

Remark 3. Due to the use of x-arithmetic, there will be added ambiguity in the computation of $R - R'$ since we will only know each of R, R' up to sign. This means there are 4 choices for this point that will be considered in our analysis later on.

3.2 Fault model and practicality

In this work, we assume that for signature schemes, we can trigger multiple signing processes for the same message. Our aim will be to fault during signing and

obtain the secret key from the resulting information. For public-key encryption or key-encapsulation mechanisms, we mostly target the encryption or encapsulation process. Our aim is to recover the message or the encapsulated shared key in plaintext, respectively. Here, we restrict to a single execution of the algorithm to account for the fact that usually ephemeral keys are used in these routines.

As usual in fault attacks, in this work we assume physical access to the device running the algorithms. Our fault model restricts to first-order faults, i.e., the injection of a single fault during the algorithm, making our model more practical than higher-order fault attacks. As discussed above, the main approach of our fault attacks is to tamper with the variables m_{ij} in memory, before the (bi)scalar multiplication algorithm is executed. In particular, the algorithm is supposed to use the value $m'_{ij} \neq m_{ij}$, under the assumption that the number $\#\Delta$ of possible values for $\Delta = m_{ij} - m'_{ij}$ is relatively small, e.g., up to 2^9.

In the subsequent sections, we will only analyze our attacks from a theoretical point of view, focusing on the post-processing of the faulty outputs to recover secret keys or plaintext messages, respectively. Below, we discuss the feasibility of our fault model in practice, using established practical fault scenarios from the literature, in particular from the overview on the practicality of fault attacks in [10]. Based on this, there are several options for the practical deployment of our fault approach.

- In *bit-flip faults*, the attacker can change the value of one or multiple bits of their choice. Due to the required precision, this attack requires a relatively expensive setup, see [10, Table 1]. As we will see, in our scenario this model will mainly be useful if we can flip the least significant bit of the target value. This corresponds to the best case of $\Delta = \pm 1$.
- In *bit-set faults*, the attacker can set the value of one or multiple bits of their choice to 1 or 0, independent of their previous value. This fault approach is easier to execute in practice, and requires a less expensive setup [10, Table 1]. For our purposes, we can assume that it is possible to set a specific byte of a variable m_{ij} to all-1 or all-0. This results in $\#\Delta \leq 2^9$. Again, there are certain trade-offs between the precision of the attack and the value $\#\Delta$, which can be considered for the practical execution. We will assume that we can target the last byte of m_{ij}.
- In *random-byte faults* the attacker can set the value of a chosen byte to a random value, independent of its previous value. This model is more relaxed, and much easier to execute than bit-flip or bit-set faults [10, Table 1]. Since we do not require to know the actual value of the faulted byte, this setting is analogous to setting a byte to all-1 via bit-set for our purposes. In particular, we have $\#\Delta \leq 2^9$. Again we assume that we can target the last byte of m_{ij}.

All options assume that the target value m_{ij} has been loaded into its working registers, and is faulted before the corresponding register value is processed by the (bi)scalar ladder. Note that alternatively, we could replace this approach by injecting a fault while m_{ij} is loaded from memory. Both approaches are transient faults, i.e., m_{ij} is not permanently faulted in memory, but only for the current

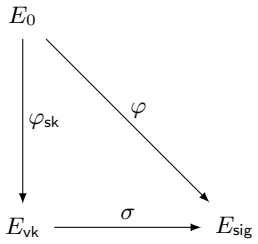

Fig. 1. PRISM identification protocol.

execution. The exact practical execution depends on the ladder implementation and platform, in particular how and when the values m_{ij} are loaded from memory. Nevertheless, we expect the fault model to be practical in either way, since the attacker can adjust the fault injection to the specific behavior in the target implementation.

In summary, the literature on fault attacks shows that our fault model is feasible in practice. While the best-case scenario of bit-flip faults requires a relatively sophisticated and expensive setup, it is enough for our attacks to perform random-byte faults, which is much easier and cheaper in practice. This comes at a higher cost for post-processing, but we will show in the following sections that this remains practical for all our attacks.

4 PRISM

In what follows we outline a key-recovery attack on PRISM [4] using the fault model from Section 3. As such, we first give a brief overview of the signing algorithm that we will be attacking prior to presenting the attack itself. Note, we opt to use the deterministic signing algorithm presented in [4], but as we will see later on (see Remark 6), the attack can be adapted to attack a probabilistic version of signing as well.

4.1 Signing in PRISM

PRISM [4] is an isogeny-based identification protocol that is transformed into a signature using the hash-and-sign paradigm. The signature is called $\mathsf{PRISM-sig}$, but we will abuse notation and refer to it as PRISM for brevity. The authors highlight that its security relies on the hardness of computing large prime degree isogenies between elliptic curves when the endomorphism rings of the curves are not known. As we will see, it also relies on the hardness of the core isogeny problem with auxiliary information (masked torsion points).

Let E_0 be a public supersingular starting curve over $\mathbb{F}_{p^2}$ with the torsion group $E_0[2^a]$ defined over $\mathbb{F}_{p^2}$ and a public basis (P_0, Q_0) of $E_0[2^a]$. The identification protocol, depicted in Figure 1, has the prover first select a secret isogeny

Algorithm 1 Signing algorithm from PRISM [4, Alg. 1].

Input: Secret key $\mathsf{sk} = \varphi_{\mathsf{sk}} : E_0 \to E_{\mathsf{vk}}$ of degree N_{sk} with corresponding ideal I_{sk}; public key $\mathsf{vk} = (E_{\mathsf{vk}}, P_{\mathsf{vk}}, Q_{\mathsf{vk}})$; prime number $q \in (2^{a-1}, 2^a)$

Output: A two-dimensional representation of $\sigma : E_{\mathsf{vk}} \to E_{\mathsf{sig}}$ of degree $q(2^a - q)$

Find $m_{11}, m_{12} \in \mathbb{Z}/2^a\mathbb{Z}$ such that $P_{\mathsf{vk}} = [m_{11}]\varphi_{\mathsf{sk}}(P_0) + [m_{12}]\varphi_{\mathsf{sk}}(Q_0)$
Find $m_{21}, m_{22} \in \mathbb{Z}/2^a\mathbb{Z}$ such that $Q_{\mathsf{vk}} = [m_{21}]\varphi_{\mathsf{sk}}(P_0) + [m_{22}]\varphi_{\mathsf{sk}}(Q_0)$
$N \leftarrow q(2^a - q)$
$I_{\mathsf{chall}} \leftarrow \mathsf{RandomFixedNormIdeal}(\mathcal{O}_0, N)$
$I \leftarrow I_{\mathsf{sk}} \cap I_{\mathsf{chall}}$
$\varphi \leftarrow \mathsf{IdealToIsogeny}(I)$
Set E_{sig} as the codomain of σ
$P_{\mathsf{sig}} \leftarrow [m_{11}]\varphi(P_0) + [m_{12}]\varphi(Q_0)$
$Q_{\mathsf{sig}} \leftarrow [m_{21}]\varphi(P_0) + [m_{22}]\varphi(Q_0)$
return $(E_{\mathsf{sig}}, P_{\mathsf{sig}}, Q_{\mathsf{sig}})$

$\varphi_{\mathsf{sk}} : E_0 \to E_{\mathsf{vk}}$ of degree $\deg \varphi_{\mathsf{sk}} = N_{\mathsf{sk}}$ (see Remark 4). They publish the curve E_{vk} along with a deterministic basis $(P_{\mathsf{vk}}, Q_{\mathsf{vk}})$ of $E_{\mathsf{vk}}[2^a]$, and compute the matrix $M \in \mathbb{Z}_{2^a}^{2\times 2}$, such that

$$\begin{pmatrix} P_{\mathsf{vk}} \\ Q_{\mathsf{vk}} \end{pmatrix} = M \cdot \varphi_{\mathsf{sk}} \begin{pmatrix} P_0 \\ Q_0 \end{pmatrix}.$$

This is equivalent to viewing $(P_{\mathsf{vk}}, Q_{\mathsf{vk}})$ as masked torsion points of $(\varphi_{\mathsf{sk}}(P_0), \varphi_{\mathsf{sk}}(Q_0))$ under the action of the matrix M. Note, the degree of φ_{sk} is not made public.

The verifier's challenge is a large prime q, to which the prover responds with a (two-dimensional) representation of an isogeny $\sigma : E_{\mathsf{vk}} \to E_{\mathsf{sig}}$ of degree $q(2^a - q)$. In practice, the prover computes the isogeny $\varphi = \sigma \circ \varphi_{sk}$, and represents σ through its degree and the masked torsion information

$$\begin{pmatrix} P_{\mathsf{sig}} \\ Q_{\mathsf{sig}} \end{pmatrix} = M \cdot \varphi \begin{pmatrix} P_0 \\ Q_0 \end{pmatrix} = M \cdot \sigma \circ \varphi_{sk} \begin{pmatrix} P_0 \\ Q_0 \end{pmatrix} = \sigma \begin{pmatrix} P_{\mathsf{vk}} \\ Q_{\mathsf{vk}} \end{pmatrix}.$$

The signing algorithm is described in Algorithm 1. See [4, Section 3] for details about the subroutines used, and the rest of the PRISM protocol.

Remark 4. In practice, the secret isogeny φ_{sk} is computed by first sampling a uniformly random primitive left $\mathcal{O}_0$-ideal I'_{sk} of large prime norm $D_{\text{mix}} \approx p^2$ to ensure a distribution close to uniform, where $\mathcal{O}_0 \cong \text{End}(E_0)$. An equivalent ideal I_{sk} of smaller prime norm of heuristic size $N_{\mathsf{sk}} \approx 2^{\varepsilon}\sqrt{p}$ is then generated where usually $\varepsilon < 12$ in practice. It is translated to the isogeny φ_{sk} using the $\mathsf{IdealToIsogeny}$ routine (see, e.g., [5]). The ideal I_{sk} is stored as secret key. Note that the reduced size of N_{sk} will be crucial for our attack to work, since the degree of φ_{sk} must be sufficiently small. However, it is not strictly necessary to work with the reduced-norm ideal I_{sk}. Working with I'_{sk} instead would prevent our attack at the cost of slightly worse signing performance for the most recent PRISM implementations [5], larger memory requirements for the secret key, and

increased RAM usage for quaternion arithmetic during signing due to larger integer sizes. We give more details in Appendix B.

4.2 Key-recovery attack on PRISM

For our attack, we assume that we can repeatedly trigger the execution of the signing algorithm with the same input message. We also assume for now, as is described in [4], that the signing algorithm is deterministic (see Remark 6 for a discussion of the probabilistic version). The basic idea is to target the masked torsion points that are output in Algorithm 1. Namely,

$$\begin{pmatrix} P_{\mathsf{sig}} \\ Q_{\mathsf{sig}} \end{pmatrix} = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{pmatrix} \varphi(P_0) \\ \varphi(Q_0) \end{pmatrix} = \begin{pmatrix} [m_{11}]\varphi(P_0) + [m_{12}]\varphi(Q_0) \\ [m_{21}]\varphi(P_0) + [m_{22}]\varphi(Q_0) \end{pmatrix}.$$

By faulting, say, m_{11} to some value m'_{11} during the multiplication in Step 8, we would obtain

$$P'_{\mathsf{sig}} = [m'_{11}]\varphi(P_0) + [m_{12}]\varphi(Q_0)$$

as part of the signature. Thus, by running the algorithm once honestly[6] and once with the fault, we obtain both the correct point P_{sig} and the faulted point P'_{sig}, and we can compute

$$P_{\mathsf{sig}} - P'_{\mathsf{sig}} = [m_{11} - m'_{11}]\varphi(P_0).$$

Following our fault model from Section 3, there is only a small number of possible values of $\Delta = m_{11} - m'_{11}$. Since P_0 is a 2^a-torsion point, we will need Δ^{-1} to recover the information about $\varphi(P_0)$. We can check if Δ is odd (and thus has a well-defined inverse) by checking whether the order of the point $P_{\mathsf{sig}} - P'_{\mathsf{sig}}$ equals 2^a. Remark 5 describes a trade-off between making use of an even Δ at the cost of more post-processing. For conciseness, we will assume that Δ is odd in what follows.

Now, denote the number of remaining candidate solutions of Δ as $\#\Delta$. In addition, if x-only arithmetic is being used in the implementation of Algorithm 1, then there will be 4 options for the value of $P_{\mathsf{sig}} - P'_{\mathsf{sig}}$, since we only know each point up to sign. Overall, this leaves a maximum of $4 \cdot \#\Delta$ candidates. For each candidate, we run the post-processing step of our attack below, and check the correctness of our guess by validating the correctness of the representation of the secret key isogeny φ_{sk} we obtain.

Assuming we have guessed the correct combination of Δ and $P_{\mathsf{sig}} - P'_{\mathsf{sig}}$, we can recover $\varphi(P_0)$ from the knowledge of $[m_{11} - m'_{11}]\varphi(P_0)$. This is done by computing $\varphi(P_0) = [\Delta^{-1}](P_{\mathsf{sig}} - P'_{\mathsf{sig}})$. Recall, we are enforcing Δ to be odd (see Remark 5 for an alternative), which occurs with probability $1/2$. Thus, we obtain $4(\#\Delta/2) = 2(\#\Delta)$ possibilities for $\varphi(P_0)$.

[6] An additional non-faulted query will not be necessary in practice. Details are in Algorithm 2 and its complexity description.

Now, by running a second fault attack that targets, say, m_{12}, we can recover $\varphi(Q_0)$ in the same way. Since we would require the correct combination of $\varphi(P_0), \varphi(Q_0)$, we will have to run the post-processing up to $4(\#\Delta)^2$ times.

Assuming that both guesses for $\varphi(P_0), \varphi(Q_0)$ were correct, we have recovered torsion point information about a torsion subgroup of order 2^{2a}. The degree of the isogeny, however, is $\deg \varphi \approx 2^{3a}$. This is not enough torsion information for the HD representation described in Section 2.2, so instead we will directly target the secret key isogeny φ_{sk} whose degree is $\deg \varphi_{\mathsf{sk}} \approx 2^a$.

Recall that to obtain the HD representation of φ_{sk}, we will need to know both the degree of φ_{sk} and how it acts on torsion points of sufficiently large order. To recover the latter, it is sufficient to recover the values of m_{ij} since $P_{\mathsf{vk}} = [m_{11}]\varphi_{\mathsf{sk}}(P_0) + [m_{12}]\varphi_{\mathsf{sk}}(Q_0)$ and $Q_{\mathsf{vk}} = [m_{21}]\varphi_{\mathsf{sk}}(P_0) + [m_{22}]\varphi_{\mathsf{sk}}(Q_0)$, where $P_{\mathsf{vk}}, Q_{\mathsf{vk}}, P_0, Q_0$ are all publicly known.

Starting with m_{12}, we can compute it using 2 Weil pairings and 1 DLP computation from

$$\begin{aligned} e(P_{\mathsf{sig}}, \varphi(P_0)) &= e([m_{11}]\varphi(P_0) + [m_{12}]\varphi(Q_0), \varphi(P_0)) \\ &= e(\varphi(P_0), \varphi(P_0))^{m_{11}} \cdot e(\varphi(Q_0), \varphi(P_0))^{m_{12}} \\ &= 1 \cdot e(\varphi(Q_0), \varphi(P_0))^{m_{12}}. \end{aligned}$$

Symmetrically, we can recover the remaining m_{ij}. Therefore, we obtain M and compute M^{-1}, from which we can recover

$$\begin{pmatrix} \varphi_{\mathsf{sk}}(P_0) \\ \varphi_{\mathsf{sk}}(Q_0) \end{pmatrix} = M^{-1} \cdot \begin{pmatrix} P_{\mathsf{vk}} \\ Q_{\mathsf{vk}} \end{pmatrix}.$$

Finally, to recover the degree of φ_{sk}, we can also use 2 pairings and 1 DLP computation since

$$e(\varphi_{\mathsf{sk}}(P_0), \varphi_{\mathsf{sk}}(Q_0)) = e(P_0, Q_0)^{\deg \varphi_{\mathsf{sk}}}.$$

The attack is summarized in Algorithm 2 using the following notation: a standard Weil pairing is denoted by $\lambda_{PQ} := e(P, Q)$; $\mathsf{DLP}_n(a, b)$ computes x such that $a = b^x \bmod n$.

Validating correct guesses. In case of the correct guess of the points $\varphi_{\mathsf{sk}}(P_0)$, $\varphi_{\mathsf{sk}}(Q_0)$, Algorithm 2 outputs a valid representation of φ_{sk}. In case of a wrong guess, with overwhelming probability the output will not be the HD representation of an isogeny. Hence, we can efficiently validate our guess for $\varphi_{\mathsf{sk}}(P_0), \varphi_{\mathsf{sk}}(Q_0)$.

In order for the HD representation to be efficient, we would like to construct a $(2^a, 2^a)$-isogeny from our knowledge of how $\varphi_{\mathsf{sk}} : E_0 \to E_{\mathsf{vk}}$ acts on the torsion subgroup. Since the degree $\deg \varphi_{\mathsf{sk}} = d$ is most likely not smooth, the obvious choice would be to decompose the $(2^a, 2^a)$-isogeny into a degree d and degree $d' := 2^a - d$ isogeny. With knowledge of the endomorphism ring of E_0, we can thus compute a degree-d' isogeny $\varphi' : E_0 \to E_1$ (see Fig. 2 for reference).

Algorithm 2 Fault injection key-recovery attack on PRISM.

Input: Public parameters $\mathsf{pp} = (a, E_0, E_{\mathsf{vk}}, (P_0, Q_0), (P_{\mathsf{vk}}, Q_{\mathsf{vk}}), q)$, faulted signature $\mathsf{sig}' = (E_{\mathsf{sig}}, P'_{\mathsf{sig}}, Q_{\mathsf{sig}})$, second faulted signature $\mathsf{sig}'' = (E_{\mathsf{sig}}, P_{\mathsf{sig}}, Q'_{\mathsf{sig}})$, (odd) candidate $\Delta_1 = m_{11} - m'_{11}, \Delta_2 = m_{22} - m'_{22}$ values

Output: Secret signing key φ_{sk}

`// Recover secret scalars`

Compute $\varphi(P_0) = [\Delta_1^{-1}](P_{\mathsf{sig}} - P'_{\mathsf{sig}})$ and $\varphi(Q_0) = [\Delta_2^{-1}](Q_{\mathsf{sig}} - Q'_{\mathsf{sig}})$

Compute Weil pairings $\lambda_{P_{\mathsf{sig}}\varphi(P_0)}, \lambda_{P_{\mathsf{sig}}\varphi(Q_0)}, \lambda_{Q_{\mathsf{sig}}\varphi(P_0)}, \lambda_{Q_{\mathsf{sig}}\varphi(Q_0)}, \lambda_{\varphi(P_0)\varphi(Q_0)}, \lambda_{\varphi(Q_0)\varphi(P_0)}$

Compute $m_{11} = \mathsf{DLP}_{2^a}(\lambda_{P_{\mathsf{sig}}\varphi(Q_0)}, \lambda_{\varphi(P_0)\varphi(Q_0)})$, $m_{12} = \mathsf{DLP}_{2^a}(\lambda_{P_{\mathsf{sig}}\varphi(P_0)}, \lambda_{\varphi(Q_0)\varphi(P_0)})$, $m_{21} = \mathsf{DLP}_{2^a}(\lambda_{Q_{\mathsf{sig}}\varphi(Q_0)}, \lambda_{\varphi(P_0)\varphi(Q_0)})$, $m_{22} = \mathsf{DLP}_{2^a}(\lambda_{Q_{\mathsf{sig}}\varphi(P_0)}, \lambda_{\varphi(Q_0)\varphi(P_0)})$

`// Recover HD representation of` φ_{sk}

Compute $\varphi_{\mathsf{sk}}(P_0), \varphi_{\mathsf{sk}}(Q_0)$ such that $P_{\mathsf{vk}} = [m_{11}]\varphi_{\mathsf{sk}}(P_0) + [m_{12}]\varphi_{\mathsf{sk}}(Q_0)$ and $Q_{\mathsf{vk}} = [m_{21}]\varphi_{\mathsf{sk}}(P_0) + [m_{22}]\varphi_{\mathsf{sk}}(Q_0)$

Compute pairings $\lambda_{\varphi_{\mathsf{sk}}(P_0)\varphi_{\mathsf{sk}}(Q_0)}, \lambda_{P_0Q_0}$

Compute $\deg \varphi_{\mathsf{sk}} = \mathsf{DLP}_{2^a}(\lambda_{\varphi_{\mathsf{sk}}(P_0)\varphi_{\mathsf{sk}}(Q_0)}, \lambda_{P_0Q_0})$

return $\varphi_{\mathsf{sk}} = (\deg \varphi_{\mathsf{sk}}, (P_0, Q_0), (\varphi_{\mathsf{sk}}(P_0), \varphi_{\mathsf{sk}}(Q_0)))$

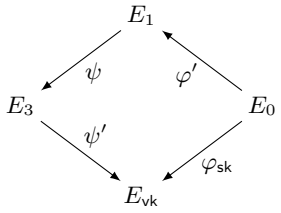

Fig. 2. The Kani isogeny diamond used to validate a candidate guess for the PRISM secret key.

Let $P_1 := \varphi'(P_0), Q_1 := \varphi'(Q_0)$. Following Section 2.2, we have sufficient torsion information so that we can compute the $(2^a, 2^a)$-isogeny

$$\Phi = \begin{bmatrix} \hat{\varphi}' & \hat{\varphi}_{\mathsf{sk}} \\ -\psi & \hat{\psi}' \end{bmatrix} : E_1 \times E_{\mathsf{vk}} \to E_0 \times E_3,$$

and observe whether its codomain splits. Its kernel is $\ker(\Phi) = \langle (P_1, \varphi_{\mathsf{sk}}(P_0)), (Q_1, \varphi_{\mathsf{sk}}(Q_0)) \rangle$.

Note, though it's not necessary for validation, Φ gives a method for evaluating points at φ_{sk}. Namely, with our knowledge of φ', we are able to isolate for the action of $\hat{\varphi}_{\mathsf{sk}}$, and thus recover a representation for φ_{sk} directly.

Complexity. Overall, we have to run Algorithm 2 up to $4(\#\Delta)^2$ times to recover φ_{sk}, requiring a fixed number of Weil pairings, DLP computations over smooth groups, point arithmetic, and a $(2^a, 2^a)$-isogeny. Thus, the final time complexity is dominated by $4(\#\Delta)^2$.

In terms of fault injections, we require only two faulty signing queries, each with a first-order fault. Further, those are the *only* queries required, i.e. no fault-free query is necessary since the two faulted queries already return all of the necessary information, namely $(P'_{\mathsf{sig}}, Q_{\mathsf{sig}})$ and $(P_{\mathsf{sig}}, Q'_{\mathsf{sig}})$.

Probability of success. Let γ denote the probability of success for the type of fault injection being used. Recall that we are assuming that both Δ_1 and Δ_2 are odd integers. This occurs with probability $1/2$ each. Thus the probability of injecting a useful fault each time is $\gamma/2$.

Remark 5. If Δ is even, we are not strictly required to attempt a new fault attack. Instead, if we assume that Δ is even, but $4 \nmid \Delta$, then, e.g., we obtain $[2]\varphi(P)$ instead of $\varphi(P)$. This leaves two possible candidates for $\varphi(P)$, for which we can run our attack. We can proceed similarly if larger powers of 2 divide Δ, thereby increasing the number of necessary attempts, but avoiding additional faulted samples. If the target isogeny degree is sufficiently larger than the available torsion, then an alternative approach would be to scale all of the torsion information by this small power of two. This provides a trade-off between the probability of success and the cost of post-processing. Note that we obtain the 2-valuation of Δ from the point order of $P_{\mathsf{sig}} - P'_{\mathsf{sig}}$.

Remark 6 (Probabilistic Prism). The above attack relies on the fact that the signing algorithm is deterministic, as is stated in [4, Remark 6]. One countermeasure, however, could be to make the signature probabilistic, in which case the points $P_{\mathsf{sig}}, P'_{\mathsf{sig}}$ would lie on different elliptic curves, despite signing the same message. With access to 4 fault injections instead of 2, however, the secret information can still be recovered. The details of this attack are reserved for Appendix A.

5 POKÉ

POKÉ [7] is an isogeny-based public-key encryption scheme. In this section we present attacks for recovering both a single plaintext and the secret key, where the former uses a weaker attack model aligned with Section 3 and the latter requires stronger assumptions.

5.1 Summary of POKÉ

POKÉ [7] first aims to compute a commutative diagram in order to settle on a shared secret, where it achieves commutativity using the addition of masked torsion points. The authors highlight that the security of the scheme relies on the hardness of recovering the secret isogeny given these scaled torsion points but without publishing the degree of the isogeny. Once the shared key is established, it uses a key-derivation function to complete encryption.

We begin by describing the full encryption algorithm, as sketched in Figure 3.

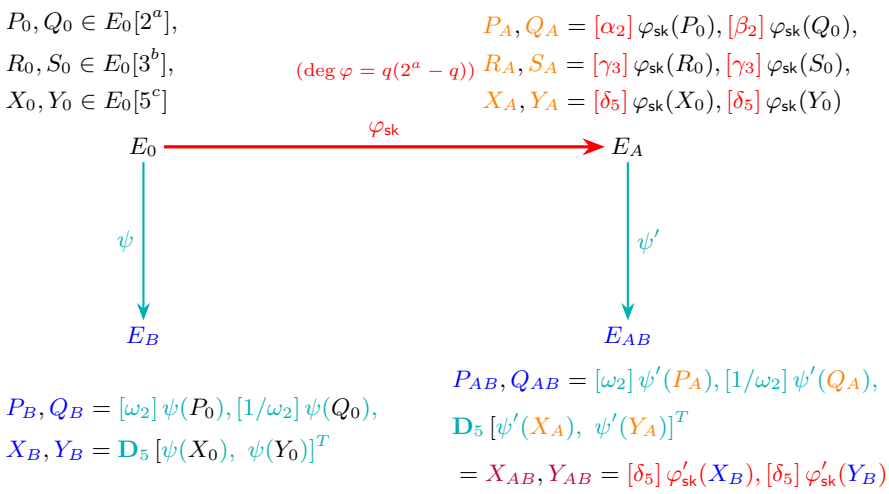

Fig. 3. A sketch of POKÉ's encryption algorithm. Black: pk parameter; red: secret key; orange: public key; lightblue: nonce; blue: ciphertext; purple: the shared key.

Key generation. Let $p = 2^a 3^b 5^c f - 1$ and E_0 be a supersingular elliptic curve of known endomorphism ring. Let $\langle P_0, Q_0 \rangle = E_0[2^a], \langle R_0, S_0 \rangle = E_0[3^b], \langle X_0, Y_0 \rangle = E_0[5^c]$.

The secret key is an isogeny $\varphi_{\mathsf{sk}} : E_0 \to E_A$. The public key is E_A, together with the masked points

$$\begin{pmatrix} P_A \\ Q_A \end{pmatrix} = \varphi_{\mathsf{sk}} \begin{pmatrix} \alpha_2 & 0 \\ 0 & \beta_2 \end{pmatrix} \begin{pmatrix} P_0 \\ Q_0 \end{pmatrix}, \qquad \begin{pmatrix} R_A \\ S_A \end{pmatrix} = [\gamma_3]\varphi_{\mathsf{sk}} \begin{pmatrix} R_0 \\ S_0 \end{pmatrix}, \quad \begin{pmatrix} X_A \\ Y_A \end{pmatrix} = [\delta_5]\varphi_{\mathsf{sk}} \begin{pmatrix} X_0 \\ Y_0 \end{pmatrix},$$

where $\alpha_2, \beta_2 \in \mathbb{Z}_{2^a}^\times, \gamma_3 \in \mathbb{Z}_{3^b}^\times, \delta_5 \in \mathbb{Z}_{5^c}^\times$ are random.

Encryption. Encryption is described in Algorithm 3. To encrypt, the user generates a random isogeny $\psi : E_0 \to E_B$, whose kernel is of the form $\langle R_0 + [r_3]S_0 \rangle$ for a random $r_3 \in \mathbb{Z}_{3^b}^\times$. By using the public key, the user will publish

$$\begin{pmatrix} P_B \\ Q_B \end{pmatrix} = \begin{pmatrix} \omega_2 & 0 \\ 0 & \omega_2^{-1} \end{pmatrix} \psi \begin{pmatrix} P_0 \\ Q_0 \end{pmatrix}, \text{ and } \begin{pmatrix} X_B \\ Y_B \end{pmatrix} = \mathbf{D}_5 \psi \begin{pmatrix} X_0 \\ Y_0 \end{pmatrix},$$

for some random $\omega_2 \in \mathbb{Z}_{2^a}^\times$, and $\mathbf{D}_5 \in SL_2(\mathbb{Z}_{5^c})$. Note, a new ψ is generated for each message, meaning the approach from Section 4 of calling the signing algorithm two distinct times will not work here.

Using the same secret scalars $r_3, \omega_2, \mathbf{D}_5$, and the public key points $P_A, Q_A, R_A, S_A, X_A, Y_A$, the user computes the parallel isogeny $\psi' : E_A \to E_{AB}$ and the

Algorithm 3 POKÉ Encryption [7, Alg. 4].

Input: A message m, a public key $E_A, P_A, Q_A, R_A, S_A, X_A, Y_A$.
Output: A ciphertext ct.

1: Sample a random $r_3 \xleftarrow{\$} \mathbb{Z}_{3^b}$.
2: Compute the isogeny $\psi : E_0 \to E_B$ with kernel $\langle R_0 + [r_3]S_0 \rangle$.
3: Compute the isogeny $\psi' : E_A \to E_{AB}$ with kernel $\langle R_A + [r_3]S_A \rangle$.
4: Sample a random $\omega_2 \xleftarrow{\$} \mathbb{Z}_{2^a}^{\times}$.
5: Compute $P_B = [\omega_2]\psi(P_0)$, $Q_B = [1/\omega_2]\psi(Q_0)$.
6: Compute $P_{AB} = [\omega_2]\psi'(P_A)$, $Q_{AB} = [1/\omega_2]\psi'(Q_A)$.
7: Sample a random $\mathbf{D}_5 \xleftarrow{\$} \mathrm{SL}_2(\mathbb{Z}_{5^c})$.
8: Compute $[X_B, Y_B]^T = \mathbf{D}_5\,[\psi(X_0), \psi(Y_0)]^T$.
9: Compute $[X_{AB}, Y_{AB}]^T = \mathbf{D}_5\,[\psi'(X_A), \psi'(Y_A)]^T$.
10: Compute $\mathsf{ct}' = \mathrm{KDF}(X_{AB}, Y_{AB}) \oplus m$.
11: **return** $\mathsf{ct} = \big((E_B, P_B, Q_B, X_B, Y_B), (E_{AB}, P_{AB}, Q_{AB}), \mathsf{ct}'\big)$.

points

$$\begin{pmatrix} P_{AB} \\ Q_{AB} \end{pmatrix} = \begin{pmatrix} \omega_2 & 0 \\ 0 & \omega_2^{-1} \end{pmatrix} \psi' \begin{pmatrix} P_A \\ Q_A \end{pmatrix}, \text{ and } \begin{pmatrix} X_{AB} \\ Y_{AB} \end{pmatrix} = \mathbf{D}_5 \psi' \begin{pmatrix} X_A \\ Y_A \end{pmatrix}.$$

The user publishes $E_B, (P_B, Q_B), (X_B, Y_B)$ as well as $E_{AB}, (P_{AB}, Q_{AB})$, and the encoded message $ct' = m \oplus \mathsf{KDF}(X_{AB}, Y_{AB})$ (here, $\mathsf{KDF}()$ refers to some agreed upon key-derivation function).

Decryption. The decryption algorithm is recalled in Algorithm 4. The goal of decryption is to recompute the shared secret to obtain X_{AB}, Y_{AB}, from which the message can be recovered as $m = ct' \oplus \mathsf{KDF}(X_{AB}, Y_{AB})$.

To compute $\varphi'_{\mathsf{sk}} : E_B \to E_{AB}$, the receiver computes points $P'_{AB}, Q'_{AB} = [\alpha_2^{-1}]P_{AB}, [\beta_2^{-1}]Q_{AB}$. By commutativity, we get that these points are the image points of P_B, Q_B under φ'_{sk}. Thus, the receiver can use the HD representation of φ'_{sk} to compute X_{AB}, Y_{AB}.

5.2 Message recovery attack on encryption

Similar to the GPST (adaptive) attack on SIDH [31], it was pointed out in [42] that POKÉ may not allow two static keys due to the possibility of a similar attack. Roughly speaking, a dishonest party can adaptively generate malformed public keys by manipulating the torsion points. By observing if the session keys match or not, the adversary can recover 1 bit of information of the secret key iteratively, each session. Hence, in the IND-CPA version, a key is only allowed to be used at most for a few times.[7]

[7] In general, it can be prevented by applying the Fujisaki-Okamoto transform [30] when achieving the IND-CCA security. The transform requires the encrypting party to disclose the randomness in use in the end. In this case, at most one party is allowed to have a static key.

Algorithm 4 POKÉ Decryption [7, Alg. 5].

Input: $\mathsf{sk} = (q, \alpha_2, \beta_2, \delta_5), \mathsf{ct} = ((E_B, P_B, Q_B, X_B, Y_B), (E_{AB}, P_{AB}, Q_{AB}), ct')$
Output: A message m'

1: Compute $P'_{AB} = [\alpha_2^{-1}]P_{AB}, Q'_{AB} = [\beta_2^{-1}]Q_{AB}$
2: Compute Φ' with kernel $\langle([-q]P_B, P'_{AB}), ([-q]Q_B, Q'_{AB})\rangle$ `/* Kani attack */`
3: Evaluate $(X'_B, _) = \Phi'(X_B, \mathcal{O}_{E_{AB}})$
4: Evaluate $(Y'_B, _) = \Phi'(Y_B, \mathcal{O}_{E_{AB}})$
5: Generate a basis U, V of $E_{AB}[5^c]$
6: Evaluate $(U', _) = \Phi'(\mathcal{O}_{E_B}, U)$
7: Evaluate $(V', _) = \Phi'(\mathcal{O}_{E_B}, V)$
8: Find x, y such that $X'_B = [x]U' + [y]V'$
9: Find w, z such that $Y'_B = [w]U' + [z]V'$
10: Compute $X_{AB} = [\delta_5][2^a - q]([x]U + [y]V)$
11: Compute $Y_{AB} = [\delta_5][2^a - q]([w]U + [z]V)$
12: Compute $m' = KDF(X_{AB}, Y_{AB}) \oplus \mathsf{ct}'$
13: **return** m'

Thus, an attack recovering the secret key may only in turn allow an adversary to recover few messages. This means that a key-recovery attack and a plaintext-recovery attack will have similar value from an adversarial viewpoint. Nonetheless, we will present both types of attacks for completeness. Here, we first outline a fault injection attack that recovers the message m by targeting the isogeny ψ.

Following the fault model outlined in Section 3, this attack will fault Algorithm 3, Step 5. It targets the multiplication $P_B = [\omega_2]\psi(P_0)$ by tampering with bits or bytes of ω_2. Call this tampered scalar ω'_2 and the resulting point $P'_B = [\omega'_2]\psi(P_0)$. Thus, P'_B and the untampered $Q_B = [\omega_2^{-1}]\psi(Q_0)$ are published in the ciphertext.

We would like to recover ω_2 and ω'_2 in order to unmask the torsion points and thus obtain an HD representation of ψ. First, we have

$$e(P'_B, Q_B) = e(P_0, Q_0)^{\omega'_2\omega_2^{-1} \deg \psi}$$

with known degree of ψ. Thus, with 2 pairings and a DLP computation, we can recover the product $\omega'_2\omega_2^{-1}$ (modulo 2^a). We know that $\omega'_2 = \omega_2 + \Delta$, where Δ has a small number of possible values depending on the attack model chosen (see Section 3). Thus

$$\omega'_2\omega_2^{-1} = 1 + \Delta\omega_2^{-1}. \tag{1}$$

We know the product $\omega'_2\omega_2^{-1}$, and can guess Δ, so we obtain ω_2^{-1}. Though, again, we require that Δ be odd in order for its inverse to be well defined. By targeting the last byte in ω_2 during the fault injection, we get an odd Δ with probability 1/2. From here, we can recover ω'_2 from Equation 1, and so we can recover $\psi(P_0), \psi(Q_0)$.

With these points, and the known degree 3^b, we have an HD representation of ψ and thus we can compute the message.

The attack is summarized in Algorithm 5 using the following subroutines:

Algorithm 5 Fault injection message recovery attack on POKÉ encryption.

Input: Public parameters $\mathsf{pp} = (p, a, b, c, E_0, E_A, (P_0, Q_0), (R_0, S_0), (X_0, Y_0), (P_A, Q_A), (R_A, S_A), (X_A, Y_A))$, the faulted ciphertext $\mathsf{ct} = \big((E_B, P'_B, Q_B, X_B, Y_B), (E_{AB}, P_{AB}, Q_{AB}), \mathsf{ct}'\big)$, (odd) candidate Δ

Output: Decrypted message m

`// Recover secret scalars`

Compute Weil pairings $\lambda_{P'_BQ_B}, \lambda_{P_0Q_0}$

Compute $\omega'_2\omega_2^{-1} := \mathsf{DLP}_{2^a}(\lambda_{P'_BQ_B}, \lambda_{P_0Q_0})/3^b$

Compute $\omega_2^{-1} = (\omega'_2\omega_2^{-1} - 1)\Delta^{-1}$, and $\omega'_2 = (\omega'_2\omega_2^{-1})\omega_2$

`// Recover HD representations for` ψ, ψ'

Compute the unmasked torsion points $\psi(P_0), \psi(Q_0) = [(\omega'_2)^{-1}]P'_B, [\omega_2]Q_B$ and $\psi'(P_A), \psi'(Q_A) = [\omega_2^{-1}]P_{AB}, [\omega_2]Q_{AB}$

Denote $\psi = (3^b, (P_0, Q_0), (\psi(P_0), \psi(Q_0)))$ and

$\psi' = (3^b, (P_A, Q_A), (\psi'(P_A), \psi'(Q_A)))$

`// Complete message recovery`

Using ψ, compute $\psi(X_0), \psi(Y_0)$

Compute $\mathbf{D}$ such that $X_B, Y_B = \mathbf{D}[\psi(X_0), \psi(Y_0)]^T$

Using ψ', compute $\psi'(X_A), \psi'(Y_A)$

Compute $X_{AB}, Y_{AB} = \mathbf{D}[\psi'(X_A), \psi'(Y_A)]^T$

return $m = \mathsf{KDF}(X_{AB}, Y_{AB}) \oplus \mathsf{ct}'$

- a standard Weil pairing is denoted by $\lambda_{PQ} := e(P, Q)$;
- $\mathsf{DLP}_n(a, b)$ computes x such that $a = b^x \bmod n$;
- the key derivation function $\mathsf{KDF}()$, inherited from POKÉ.

Complexity. As in the previous section, Algorithm 5 requires a fixed number of Weil pairings, DLP computations over smooth groups, and point arithmetic, all of which can be performed efficiently. The bottleneck of the algorithm comes from the HD evaluations of the isogenies ψ, ψ'. The evaluations can be performed similarly to Section 4.2, though it can be made even more efficient since the isogenies have smooth degree already. Denote this complexity as L_{Kani}. Contrary to Section 4, this attack need only be repeated up to $\#\Delta/2$ times since there are no point subtractions and we are enforcing an odd Δ. Similar to the attack on PRISM, we will only obtain a valid HD representation of ψ and ψ' if our guess for Δ was correct. Hence, the correctness of our guess for Δ can easily be verified, and we repeat running Algorithm 5 until we find the correct candidate for Δ. Thus, the final time complexity is dominated by $(\#\Delta/2) \cdot L_{\mathsf{Kani}}$.

In terms of fault injections, we only require a single first-order fault injection during the encryption process in POKÉ.

Probability of success. Let γ be the probability of successfully injecting a fault into the last byte of ω_2. Then, since Δ should be odd, the overall probability of injecting a useful fault is $\gamma/2$.

Note, the same trade-off described in Remark 5 would also be available here. That is, the attack can still be run if Δ is even, but coming at an increased cost to post-processing.

Remark 7. We can also obtain a message-recovery attack using a skip instruction fault. Consider Algorithm 3, Step 5. The attack would skip the instruction to invert ω_2 so that $P_B = [\omega_2]\psi(P_0)$ but also $Q_B = [\omega_2]\psi(Q_0)$. Once these points are made public in the ciphertext, an adversary can employ the attack from [21, Sec. 4]. It specifically targets masked torsion points that have been masked by the same scalar, and when the (co)domain is an elliptic curve that is "close" to its Frobenius conjugate (that is, there exists a small degree isogeny between it and its conjugate). Since POKÉ uses the special curve E_0, whose j-invariant is 1728, as its starting curve, we may apply this attack directly.

Note, instruction skipping is among the least expensive faults to inject. See [10, Table 1] for precise estimates.

Remark 8. The PKEs PIKE [12, Alg. 4, Steps 5, 6] and INKE [38, Alg. 7, Step 6] use the same approach as POKÉ of using both ω_2 and $1/\omega_2$ to mask their public torsion points, and use a similar starting set-up. This means that the attack described in this section applies directly to both of these schemes.

Remark 9. For the proposed POKÉ parameters, the available torsion information falls slightly below the general sufficient bound of Theorem 2 (see Section 7.1). We compensate for this deficit by enumerating backtracking paths of four 3-isogenies from E_B, recovering the remaining information needed to determine the sender's isogeny. As shown by our proof-of-concept implementation, the attack remains effective.

5.3 Key recovery attack on decryption

We now outline a key recovery attack that targets φ_{sk}, but requires a stronger attack model. We assume that the same secret key is being employed for long-term use, for instance, like in a KEM setting.

Knowledge of the secret scalars α_2, β_2 would unmask a torsion subgroup of order 2^{2a} which is of about the same size as $4 \deg \varphi_{\mathsf{sk}} \approx 4 \cdot 2^{2a-2} = 2^{2a}$. Thus, knowledge of α_2, β_2, and $\deg \varphi_{\mathsf{sk}}$ is sufficient to represent φ_{sk} in its HD representation.

The attack will target Algorithm 4, Step 1 using a set-to-1 fault with high precision. The fault will set one bit of α_2 to 1. We can then measure if decryption fails or succeeds using a simple power analysis. If decryption fails, then the bit we had set to 1 was in reality 0. This allows us to recover α_2, β_2 one bit at a time, thereby requiring $\log(\alpha_2) + \log(\beta_2) \approx a$ faults (where for λ bits of security, $a \approx \lambda$ [7, Sec. 7.1]). This attack model requires high precision in the bit tampering, the feasibility of which is described in Section 3.

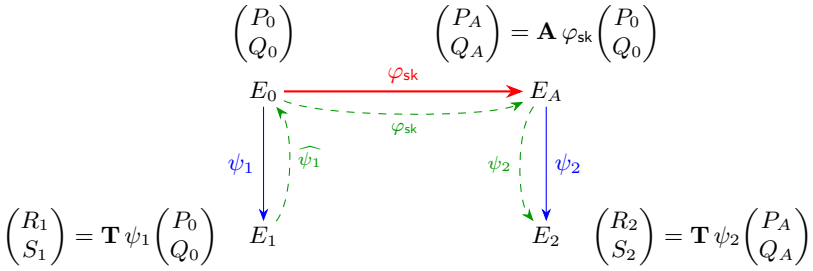

Fig. 4. FESTA Encryption.

This reveals the points $\varphi_{\mathsf{sk}}(P_0), \varphi_{\mathsf{sk}}(Q_0)$, where P_0, Q_0 were already public. Thus, with 2 pairings and 1 DLP computation we can recover the degree from

$$e(\varphi_{\mathsf{sk}}(P_0), \varphi_{\mathsf{sk}}(Q_0)) = e(P_0, Q_0)^{\deg \varphi_{\mathsf{sk}}}.$$

Therefore, we recovered sufficient information for the HD representation of φ_{sk}.

Note, this attack is similar to the recovery of secret scalars in SIKE via fault attacks [16]. Hence, this fault model is already known in the literature.

6 FESTA

FESTA [8] is a public-key encryption scheme built from a trapdoor function. In this section we first recall its procedure and then in Section 6.2 present a message recovery attack using the fault model from Section 3. Section 6.3 presents a secret key recovery attack that requires a stronger attack model, which follows closely that of Section 5.3. Additionally, in Section 6.4 we describe an attack that targets a KEM version of FESTA, which applies analogously to QFESTA.

6.1 Encryption in FESTA

We describe the encryption protocol for FESTA, as sketched in Figure 4. Let $\langle P_0, Q_0 \rangle = E_0[2^b]$.

The secret key is $\varphi_{\mathsf{sk}} : E_0 \to E_A$. The public key is E_A together with the masked torsion points P_A, Q_A under a matrix $\mathbf{A} \in \mathcal{M}_b$, where $\mathcal{M}_b$ is the commutative subset of invertible diagonal matrices over $\mathbb{Z}_{2^b}^{\times}$ modulo $\langle -\mathbf{I}_2 \rangle$ with the 2×2 identity matrix $\mathbf{I}_2$.

To encrypt, the user computes (non-parallel) isogenies $\psi_1 : E_0 \to E_1$, and $\psi_2 : E_A \to E_2$, of known degrees d_1, d_2. For computing the kernels of these isogenies, the user first samples deterministic bases of $E_0[d_1]$ and $E_A[d_2]$ via the TorGen function from [8].

They publish E_1, E_2 along with the masked torsion points

$$\begin{pmatrix} R_1 \\ S_1 \end{pmatrix} = \mathbf{T} \cdot \psi_1 \begin{pmatrix} P_0 \\ Q_0 \end{pmatrix} \text{ and } \begin{pmatrix} R_2 \\ S_2 \end{pmatrix} = \mathbf{T} \cdot \psi_2 \begin{pmatrix} P_A \\ Q_A \end{pmatrix},$$

Algorithm 6 FESTA Encryption [8, Alg. 1+3]

Input: Public key $\mathsf{pk} = (E_A, P_A, Q_A)$; message m.
Output: Ciphertext $\mathsf{ct} = (E_1, R_1, S_1, E_2, R_2, S_2)$.

1: Sample $r \xleftarrow{\$} \mathbb{Z}/d_2\mathbb{Z}$ and $\mathbf{R} \xleftarrow{\$} \mathcal{M}_b$.
2: Write $m' = m \,\|\, 0^k \bmod d_1$ and compute $s = m' + G(r, \mathbf{R})$.
3: Write $(x, \mathbf{X}) = H(s)$ and compute $t = x + r$, $\mathbf{T} = \mathbf{XR}$.
4: Compute the bases $(P_{d_1}, Q_{d_1}) \leftarrow \mathsf{TorGen}(E_0, d_1)$, $(P_{d_2}, Q_{d_2}) \leftarrow \mathsf{TorGen}(E_A, d_2)$.
5: Define $\langle K_1 \rangle := \langle P_{d_1} + [s]Q_{d_1} \rangle \subset E_0[d_1]$ and $\langle K_2 \rangle := \langle P_{d_2} + [t]Q_{d_2} \rangle \subset E_A[d_2]$.
6: Compute the d_1-isogeny $\psi_1 : E_0 \to E_1$ with kernel $\langle K_1 \rangle$.
7: Compute the d_2-isogeny $\psi_2 : E_A \to E_2$ with kernel $\langle K_2 \rangle$.
8: Compute $\begin{pmatrix} R_1 \\ S_1 \end{pmatrix} = \mathbf{T} \begin{pmatrix} \psi_1(P_0) \\ \psi_1(Q_0) \end{pmatrix}$ and $\begin{pmatrix} R_2 \\ S_2 \end{pmatrix} = \mathbf{T} \begin{pmatrix} \psi_2(P_A) \\ \psi_2(Q_A) \end{pmatrix}$.
9: Set $\mathsf{ct} \leftarrow (E_1, R_1, S_1, E_2, R_2, S_2)$.
10: **return** ct.

where $\mathbf{T} \in \mathcal{M}_b$ is a diagonal matrix.

Encryption is summarized in Algorithm 6, where G and H are random oracles with $H : \mathbb{Z}_{d_1} \to \mathbb{Z}_{d_2} \times \mathcal{M}_b$ and $G : \mathbb{Z}_{d_2} \times \mathcal{M}_b \to \mathbb{Z}_{d_1}$.

To decrypt, the public masked torsion points must be unscaled by the matrix $\mathbf{A}$. Since they have already been scaled again by $\mathbf{T}$, unmasking would be simplified if the two secret matrices commuted. For that reason, they were chosen to be diagonal matrices, in which case first applying $\mathbf{A}^{-1}$ to the masked torsion points is fine and sufficient to then recover $\mathbf{T}$.

6.2 Message recovery attack on FESTA

This attack targets Algorithm 6, Step 8. As is suggested in the protocol, let $\mathbf{T}$ be a diagonal matrix

$$\mathbf{T} := \begin{pmatrix} \alpha & 0 \\ 0 & \beta \end{pmatrix}.$$

Then the fault will target only one use of the variable α. Suppose we apply the fault to the computation of R_1 only, meaning $R_1' := [\alpha']\psi_1(P_0)$ while $R_2 = [\alpha]\psi_2(P_A)$ remains honestly computed. Now from

$$e(R_1', S_1) = e(P_0, Q_0)^{\alpha'\beta d_1}, \quad e(R_2, S_2) = e(P_A, Q_A)^{\alpha\beta d_2}$$

we can recover the product $\alpha'\beta$ and $\alpha\beta$ respectively. Denote $\Delta := \alpha' - \alpha$. Then

$$\alpha'\beta = (\alpha + \Delta)\beta = \alpha\beta + \Delta\beta.$$

Since we know the product $\alpha\beta$, and there are few options for the value of Δ (depending on the fault model from Section 3 being used), we can recover β. Note, for this computation, we require Δ to be odd in order for its inverse to be well-defined modulo 2^b. From here, we can compute α from $\alpha\beta$ and thus also α'.

With knowledge of α, α', β we are able to recover the unmasked torsion points $\psi_1(P_0), \psi_1(Q_0), \psi_2(P_A), \psi_2(Q_A)$ [8]. Using the known degrees $d_1 = \deg \psi_1, d_2 = \deg \psi_2$, we obtain HD representations of ψ_1, ψ_2.

This is sufficient information to recover the points K_1, K_2 from Algorithm 6, Step 5, and the scalars s, t. With s, t, and knowledge of α, β, we can recompute $x, \mathbf{X}, r, \mathbf{R}, m'$, and thus m.

The attack is summarized in Algorithm 7 using the following subroutines:

- a Weil pairing is denoted by $\lambda_{PQ} := e(P, Q)$;
- $\mathsf{DLP}_n(a, b)$ computes x such that $a = b^x \bmod n$;
- the $\mathsf{TorGen}()$ deterministic function from FESTA [8, p.4], that takes as input a supersingular elliptic curve E, and positive integer d, and outputs a deterministic basis of the d-torsion of E;
- the $\mathsf{ComputeCanonicalKernel}()$ function from FESTA [8, Alg. 6], takes as input points $\hat{\phi}(P'), \hat{\phi}(Q')$ for a d-isogeny $\phi : E \to E'$ and where $\langle P', Q' \rangle = E'[d]$, and outputs s such that $\ker \phi = \langle P + [s]Q \rangle$ for a canonical basis $(P, Q) = \mathsf{TorGen}(E, d)$;
- the random oracle functions G, H.

Complexity. Similarly to Section 5, Algorithm 7 will need to be run up to $\#\Delta/2$ times, since Δ is assumed to be odd. As in the previous sections, the correctness of a guess for Δ is implicitly validated by observing if we obtain valid HD representations for ψ_1 and ψ_2. This computation can be done similarly to that of Section 4.2. The bottleneck of the algorithm comes from the HD evaluations, which has time complexity L_{Kani}, and the use of the $\mathsf{ComputeCanonicalKernel}()$ function, whose complexity we denote by L_{CanKer}.[9] The final time complexity is thus dominated by $(\#\Delta/2)(2L_{Kani}) + 2L_{\mathsf{CanKer}} = (\#\Delta)(L_{Kani}) + 2L_{\mathsf{CanKer}}$.

Similar to the attack on POKÉ, we only require a single first-order fault injection during encryption.

Probability of success. Let γ denote the probability of successfully injecting a fault into the last byte of α. Then, since we require Δ to be odd, the overall probability of successfully injecting a viable fault is $\gamma/2$.

As for the attacks on PRISM and POKÉ, there exists a trade-off between the probability of success and the cost of post-processing (see Remark 5).

6.3 Key recovery attack on FESTA

The same attack from Section 5.3 can be applied to FESTA decryption. Specifically, we set one bit of the secret scalars used in the matrix $\mathbf{A}$ in [8, Alg. 2, Step

[8] One of α and α' will necessarily be even, thus computing their inverses will require extra work (see Remark 5).

[9] In [8], an implementation of FESTA is provided that includes an implementation of $\mathsf{ComputeCanonicalKernel}()$.

Algorithm 7 Fault injection message recovery attack on FESTA encryption.

Input: Public parameters $\mathsf{pp} = (p, b, E_0, E_A, E_1, E_2, (P_0, Q_0), (P_A, Q_A), d_1, d_2)$, faulted ciphertext $\mathsf{ct}' = (E_1, R_1', S_1, E_2, R_2, S_2)$, (odd) candidate Δ

Output: Decrypted message m'

`// Recover secret scalars`

Compute Weil pairings $\lambda_{R_1'S_1}, \lambda_{R_2S_2}, \lambda_{P_0Q_0}, \lambda_{P_AQ_A}$

Compute $\alpha'\beta = \mathsf{DLP}_{2^b}(\lambda_{R_1'S_1}, \lambda_{P_0Q_0})/d_1$, $\alpha\beta = \mathsf{DLP}_{2^b}(\lambda_{R_2S_2}, \lambda_{P_AQ_A})/d_2$

Compute $\beta = (\alpha'\beta - \alpha\beta)\Delta^{-1}$, $\alpha = (\alpha\beta)\beta^{-1}$, $\alpha' = \alpha + \Delta$

Set $\mathbf{T} = [\alpha\ 0; 0\ \beta]$

`// Recover` ψ_i `in HD representations`

Compute $\psi_1(P_0), \psi_1(Q_0), \psi_2(P_A), \psi_2(Q_A) = [\alpha'^{-1}]R_1', [\beta^{-1}]S_1, [\alpha^{-1}]R_2, [\beta^{-1}]S_2$

Denote representations ψ_1 as $(d_1, (P_0, Q_0), (\psi_1(P_0), \psi_1(Q_0)))$ and ψ_2 as $(d_2, (P_A, Q_A), (\psi_2(P_A), \psi_2(Q_A)))$

`// Recover kernel generators for` ψ_i

Compute canonical bases $(P_1, Q_1) = \mathsf{TorGen}(E_0, d_1), (P_2, Q_2) = \mathsf{TorGen}(E_A, d_2)$, and $(P_i', Q_i') = \mathsf{TorGen}(E_i, d_i), i = 1, 2$

Compute points $\hat{\psi}_i(P_i'), \hat{\psi}_i(Q_i')$ using the HD evaluation, for $i = 1, 2$

Compute $s = \mathsf{ComputeCanonicalKernel}(d_1, \hat{\psi}_1(P_1'), \hat{\psi}_1(Q_1'))$ and $t = \mathsf{ComputeCanonicalKernel}(d_2, \hat{\psi}_2(P_2'), \hat{\psi}_2(Q_2'))$

`// Complete message recovery`

Compute $(x, \mathbf{X}) = H(s), r = t - x, \mathbf{R} = \mathbf{X}^{-1}\mathbf{T}$

Compute $m' = s - G(r, \mathbf{R})$

return m'

1] to 1. If decryption succeeds, then the bit was in fact 1, otherwise, it was 0. This leaks the secret scalars 1 bit at a time, requiring $\log \mathbf{A}_{11} + \log \mathbf{A}_{22} \approx b$ faults. In the interest of efficiency, the parameter b is in general kept as small as possible. For example, in [8, Sec. 7.3], the FESTA-128 parameters suggest $b = 632$.

6.4 KEMs and QFESTA

Recall that a Key Encapsulation Mechanism (KEM) constitutes 3 algorithms: key generation (where a receiver fixes public/private keys); encapsulation (where the sender encapsulates a shared secret using the receiver's public key); and decapsulation (where the receiver recovers the shared secret using their secret key). These are sketched in Figure 5.

Public Key Encryption (PKE) schemes like POKÉ [7] and FESTA [8] can be used to build KEMs by applying the Fujisaki-Okamoto transform [30]. This is what was done in QFESTA [44], where the authors made changes to the original FESTA algorithm in order to improve efficiency and presented both a PKE and a KEM. The main difference between QFESTA and FESTA was that the former uses non-smooth degree isogenies, thus the fault injection attack described in Algorithm 7 remains effective against the QFESTA PKE.

Fault injecting KEMs does not follow directly from an attack on the PKE. For example, suppose that encapsulation were fault injected, similar to the fault

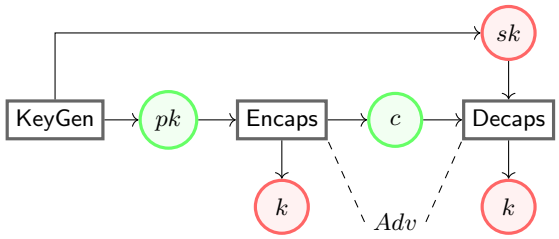

Fig. 5. Fault injection man-in-the-middle attack on a KEM. Red: private information; green: public information. *Adv*: the adversary.

injection on encryption in Section 6.2. While a shared secret may have been recovered from decrypting the ciphertext, this would also mean that the output of encapsulation is corrupted, and so decapsulation would not succeed. At this point, an honest user would abort the entire protocol, and begin again with a new key generation. This renders the recovered shared secret useless.

To avoid this, consider the following, which is sketched in Figure 5. First, a fault injection attack is performed on encapsulation, at which point a man-in-the-middle obtains the shared secret. In the case of QFESTA, Algorithm 7 achieves this. With access to the shared secret and public key of the receiver, they can recompute encapsulation to recover an honestly-generated ciphertext. They provide this corrected output to the honest receiver to decapsulate. In this way, decapsulation passes, and the man-in-the-middle can now read communications between the two users.

While the man-in-the-middle approach is a well studied attack model that is popular in studying the security of web applications, the model is stronger than those used in the previous sections since the adversary must intercept communications twice.

Remark 10. The outlined attack on the QFESTA KEM analogously applies to KEM versions of FESTA and POKÉ based on the Fujisaki-Okamoto transform. Furthermore, the attack targeting decryption described in Sections 5.3 and 6.3 could also be applied to the decapsulation step in KEMs easily.

7 Implementation

In this section we provide experimental evidence of the correctness of the attacks presented in this work.

7.1 Concrete Parameter Analysis in the Reference Implementations

The attacks in the previous sections recover the action of the isogenies on 2-power torsion subgroups. We now verify that the recovered information is sufficient for the concrete parameter sets proposed for PRISM, POKÉ, and FESTA.

Table 3. Concrete comparison between the torsion information recovered by our attacks and the sufficient condition of Theorem 2. For a target n-isogeny and an available subgroup G, the final column lists $\log_2(\#G) - \log_2(4n+1)$.

Scheme	Parameters	Available subgroup G	$\log_2(\#G)$	$\log_2(4n+1)$	Bit difference
PRISM	I ($a = 192$)	$E[2^{194}]$	388	138.86	+249.14
	III ($a = 256$)	$E[2^{258}]$	516	204.56	+311.43
	V ($a = 320$)	$E[2^{322}]$	644	265.92	+378.07
POKÉ	I ($a = 127, b = 162$)	$E[2^{129}]$	258	258.76	−0.76
	III ($a = 190, b = 243$)	$E[2^{192}]$	384	387.16	−3.14
	V ($a = 254, b = 324$)	$E[2^{256}]$	512	515.52	−3.52
FESTA	128 ($b = 632$)	$E[2^{632}]$	1264	261.77	+1002.22
	192 ($b = 992$)	$E[2^{992}]$	1984	389.54	+1594.45
	256 ($b = 1472$)	$E[2^{1472}]$	2944	519.31	+2424.68

Recall that Theorem 2 requires a sufficient condition $\#G \geq 4n+1$ for obtaining an efficient HD representation of an n-isogeny from its action on a subgroup G. Table 3 summarizes the actual parameters used in the reference implementations of each scheme.

For PRISM and FESTA, the recovered torsion information exceeds the sufficient bound by a large margin for every proposed parameter set. For POKÉ, the direct bound is missed by less than four bits. As described in Section 5, this small deficit can be handled by enumerating backtracking paths of four 3-isogenies from E_B, which recovers the remaining information required for message recovery. This is also shown in our implementation. Thus, the concrete parameters support the recovery steps used in all three attacks.

7.2 Proof-of-Concept Implementation

As a proof of concept, we implemented a simulation of the key recovery fault attack on the deterministic version of PRISM, and the message recovery attacks on POKÉ and FESTA. Our simulations are based on the SageMath code provided by each respective protocol. The code for our simulation is available at https://github.com/michael-meyer/fault-torsion-masking.

In what follows we take the PRISM simulation as an example for further study. Though, all three attacks make use of similar arithmetic and mathematical operations.

To simulate our fault model, we modified the torsion masking according to the fault approaches described in Section 3. In particular, we choose a masking coefficient m_{ij} and either flip its least significant bit, or set its last three bits or last byte to a random value during the masking multiplications.

In order to find the correct guesses for Δ_1 and Δ_2, we follow Algorithm 2. For simplicity, in case of guesses where $P_{\mathsf{sig}} - P'_{\mathsf{sig}}$ or $Q_{\mathsf{sig}} - Q'_{\mathsf{sig}}$ do not have full order 2^a, we repeat faulty runs until obtaining odd Δ_1, Δ_2 instead of guessing

the correct full-order points as discussed in Remark 5. Additionally, the implementation uses full coordinates, instead of x-only coordinates as usually done for efficiency reasons. Hence, an attack on a real-world implementation would require four times as many post-processing runs in order to guess the correct signs for computing $P_{\mathsf{sig}} - P'_{\mathsf{sig}}$ or $Q_{\mathsf{sig}} - Q'_{\mathsf{sig}}$.

For every guess of (Δ_1, Δ_2), we obtain a guess for $(\varphi_{\mathsf{sk}}(P_0), \varphi_{\mathsf{sk}}(Q_0))$ and $\deg \varphi_{\mathsf{sk}}$. We validate this HD representation of φ_{sk} by computing a $(2^a, 2^a)$-isogeny as described in [4, App. A.2] (see Section 4.2 for details). If the codomain is a product of elliptic curves containing E_0, the choice is likely correct. Experimentally, there is usually one (Δ_1, Δ_2) pair that returns such a codomain, but on occasion two such pairs are obtained of which one is always the correct solution.

Running time. Our non-optimized SageMath implementation took on average 51.8 seconds, over 50 runs when using the set-to-random fault on the last 3 bits of the m_{ij} and targeting $\lambda = 128$ bits of security. This corresponds to the case where the post-processing portion must run through an effective secret space of size 2^7. The average number of faults sufficient to obtain odd Δ_1, Δ_2 values was 4.0. The attack using this fault also ran efficiently for higher security levels.

When running the full random-byte fault targeting $\lambda = 128$ bits of security, the simulation took an average of 2.5 hours over 3 runs. This corresponds to the easiest fault scenario but the most expensive post-processing, with an effective secret space of size 2^{16} over which to search. We stress that this implementation is far from optimized, and simply provides reassurance that the attack is feasible to run in practice.

Remark 11. The SageMath implementation of PRISM differs from Algorithm 1 and [4, Alg. 1] in two ways. First, it is probabilistic by default. We make it deterministic by seeding the PRNG with the message, yielding a unique signature for each message and key pair. This version is considered valid by the authors [4, Remark 6]. Second, the reference implementation uses tm_{ij}, where $t = q^{-1} \bmod 2^a$, instead of m_{ij} in lines 8 and 9 of Algorithm 1 to avoid scalar multiplications during verification (see [5, Alg. 1]). We undo this optimization in order to make the implementation consistent to the original description in Algorithm 1 and [4, Alg. 1]. Nonetheless, we note that this does not affect the attack: in the optimized version, we will recover tm_{ij}, and then obtain m_{ij} since q, and hence t, is public. Thus, both modifications only align the implementation with [4, Alg. 1] and are not part of the fault attack.

8 Countermeasures

In the previous sections we saw that the presented attacks are able to recover secret keys or plaintext messages, respectively, with a very small number of first-order faults. This raises the question of how the technique of torsion point masking can be protected from such faults.

In addition to the already discussed PRISM-specific countermeasure in Remark 4 and Appendix B, we will present two more potential countermeasures

that prevent all of the presented attacks in the fault model we assume. The computational overhead depends on the specific implementation, but is expected to be small in general in comparison to the (higher-dimensional) isogeny computations required in PRISM, POKÉ, and FESTA. Note that, as usual, these protection mechanisms are likely not sufficient against stronger fault models, e.g., allowing higher-order faults.

For this section, assume that there are public torsion points $P, Q \in E$, and a secret isogeny $\varphi : E \to E'$. We want to protect the following masking

$$\begin{pmatrix} R \\ S \end{pmatrix} = M \begin{pmatrix} \varphi(P) \\ \varphi(Q) \end{pmatrix} = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{pmatrix} \varphi(P) \\ \varphi(Q) \end{pmatrix}$$

Redundant computation. A very simple countermeasure against first-order faults is to repeat the vulnerable computation, and verify that both results are equal. This means computing R, S as above, followed by computing R', S' in the same way, making sure that M is freshly fetched from untampered memory (e.g., from the secret key) before the second computation. Thus, a first order fault will result in one of these points being faulty, and hence a failed equality check, so the algorithm aborts.

Validation of the masked torsion points. Similar to the techniques used in the post-processing of our attacks, a user can validate the mask's correctness through pairing computations. Using the notation above, compute

$$\begin{aligned} e(R, S) &= e([m_{11}]\varphi(P) + [m_{12}]\varphi(Q), [m_{21}]\varphi(P) + [m_{22}]\varphi(Q)) \\ &= e(P, Q)^{\deg(\varphi)(m_{11}m_{22} - m_{12}m_{21})}. \end{aligned}$$

Therefore, after computing the masked torsion points R, S the user can validate them at the cost of two Weil pairing computations and an exponentiation. Depending on the application, the pairing $e(P, Q)$ can be precomputed, e.g., if P, Q are public points on a starting curve. In case the above equation does not hold, the algorithm detects a fault injection and can abort.

9 Conclusion

In this work we presented novel fault injection attacks on isogeny-based protocols that employ torsion point masking. We achieved both message-recovery and key-recovery attacks on prominent protocols: PRISM, POKÉ, and (Q)FESTA. We limited our attacks to faults that have already appeared in the fault injection literature such as bit-flip and random-byte faults. These are faults that are generally believed to be reasonable, but random bytes are thought to be less expensive to implement. In any case, all of the attacks we presented could be executed with either fault model.

We also stress that Algorithms 2, 5, and 7 scale extremely well in that the number of faults/search space for Δ does not change for higher parameter sets.

The key-recovery attacks on POKÉ and FESTA, of course scale with the bit length of the secret scalars.

The fault attacks presented in this work serve as a reminder that the way in which torsion point masking is frequently used is not completely random. These attacks were not specific to a particular implementation, but to the core ideas used in the protocol construction. This indicates that this structure may also lead to improved passive attacks, continuing the research direction of [21].

Future work. Our attacks targeted secret scalars being used in the masking process. Critically, all of the protocols we considered stored the masking scalars in the secret key, for use later in the protocol. The fact that all of these protocols either reused the masking scalar in another moment or used only one scalar in their masking of two points is what rendered them vulnerable to our family of attacks. Contrary to this, in M(D)-SIDH [29], the masking scalars are used exactly once and then *deleted*. Thus, M(D)-SIDH remains unaffected by our attacks. We leave exploration into other fault attacks that might affect it as interesting future work.

References

1. Marius A. Aardal, Gora Adj, Diego F. Aranha, Andrea Basso, Isaac Andrés Canales Martínez, Jorge Chávez-Saab, Maria Corte-Real Santos, Pierrick Dartois, Luca De Feo, Max Duparc, Jonathan Komada Eriksen, Tako Boris Fouotsa, Décio Luiz Gazzoni Filho, Basil Hess, David Kohel, Antonin Leroux, Patrick Longa, Luciano Maino, Michael Meyer, Kohei Nakagawa, Hiroshi Onuki, Lorenz Panny, Sikhar Patranabis, Christophe Petit, Giacomo Pope, Krijn Reijnders, Damien Robert, Francisco Rodríguez-Henríquez, Sina Schaeffler, and Benjamin Wesolowski. SQIsign. Technical report, National Institute of Standards and Technology, 2025. available at https://sqisign.org/spec/sqisign-20250205.pdf.
2. Gora Adj, Jesús-Javier Chi-Domínguez, Víctor Mateu, and Francisco Rodríguez-Henríquez. Faulty isogenies: a new kind of leakage. Cryptology ePrint Archive, Paper 2022/153, 2022.
3. Gustavo Banegas, Juliane Krämer, Tanja Lange, Michael Meyer, Lorenz Panny, Krijn Reijnders, Jana Sotáková, and Monika Trimoska. Disorientation faults in CSIDH. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 310–342. Springer, 2023.
4. Andrea Basso, Giacomo Borin, Wouter Castryck, Maria Corte-Real Santos, Riccardo Invernizzi, Antonin Leroux, Luciano Maino, Frederik Vercauteren, and Benjamin Wesolowski. PRISM: simple and compact identification and signatures from large prime degree isogenies. In Tibor Jager and Jiaxin Pan, editors, *Public-Key Cryptography - PKC 2025 - 28th IACR International Conference on Practice and Theory of Public-Key Cryptography, Røros, Norway, May 12-15, 2025, Proceedings, Part III*, volume 15676 of *Lecture Notes in Computer Science*, pages 300–332. Springer, 2025.
5. Andrea Basso, Giacomo Borin, Wouter Castryck, Maria Corte-Real Santos, Riccardo Invernizzi, Antonin Leroux, Luciano Maino, Frederik Vercauteren, and Benjamin Wesolowski. PRISM with a pinch of salt: Simple, efficient and strongly unforgeable signatures from isogenies. Cryptology ePrint Archive, Paper 2026/443, 2026.
6. Andrea Basso, Chenfeng He, David Jacquemin, Fatna Kouider, Péter Kutas, Anisha Mukherjee, Sina Schaeffler, and Sujoy Sinha Roy. Constant-time quaternion algorithms for sqisign. *IACR Cryptol. ePrint Arch.*, page 2192, 2025.
7. Andrea Basso and Luciano Maino. POKÉ: A compact and efficient PKE from higher-dimensional isogenies. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025, Proceedings, Part II*, volume 15602 of *Lecture Notes in Computer Science*, pages 94–123. Springer, 2025.
8. Andrea Basso, Luciano Maino, and Giacomo Pope. FESTA: fast encryption from supersingular torsion attacks. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part VII*, volume 14444 of *Lecture Notes in Computer Science*, pages 98–126. Springer, 2023.

9. Giacomo Borin, Maria Corte-Real Santos, Jonathan Komada Eriksen, Riccardo Invernizzi, Marzio Mula, Sina Schaeffler, and Frederik Vercauteren. Qlapoti: Simple and efficient translation of quaternion ideals to isogenies. In Goichiro Hanaoka and Bo-Yin Yang, editors, *Advances in Cryptology – ASIACRYPT 2025, Part IV*, volume 16248 of *Lecture Notes in Computer Science*, pages 174–205, Melbourne, VIC, Australia, December 8–12, 2025. Springer, Singapore, Singapore.
10. Jakub Breier and Xiaolu Hou. How practical are fault injection attacks, really? *IEEE Access*, 10:113122–113130, 2022.
11. Shiping Cai, Mingjie Chen, Yi-Fu Lai, and Kaizhan Lin. PIKE: Faster isogeny-based public key encryption with pairing-assisted decryption. Cryptology ePrint Archive, Paper 2026/473, 2026.
12. Shiping Cai, Mingjie Chen, Yi-Fu Lai, and Kaizhan Lin. PIKE: Faster isogeny-based public key encryption with pairing-assisted decryption. In *Public-Key Cryptography - PKC 2026 - 29th IACR International Conference on Practice and Theory of Public-Key Cryptography*, Lecture Notes in Computer Science. Springer, 2026.
13. Fabio Campos, Jorge Chávez-Saab, Jesús-Javier Chi-Domínguez, Michael Meyer, Krijn Reijnders, Francisco Rodríguez-Henríquez, Peter Schwabe, and Thom Wiggers. Optimizations and practicality of high-security CSIDH. *IACR Commun. Cryptol.*, 1(1):5, 2024.
14. Fabio Campos, Andreas Hellenbrand, Michael Meyer, and Krijn Reijnders. dc-tidh: Fast & deterministic CTIDH. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2025(3):516–541, 2025.
15. Fabio Campos, Matthias J. Kannwischer, Michael Meyer, Hiroshi Onuki, and Marc Stöttinger. Trouble at the CSIDH: protecting CSIDH with dummy-operations against fault injection attacks. In *17th Workshop on Fault Detection and Tolerance in Cryptography, FDTC 2020, Milan, Italy, September 13, 2020*, pages 57–65. IEEE, 2020.
16. Fabio Campos, Juliane Krämer, and Marcel Müller. Safe-error attacks on SIKE and CSIDH. In Lejla Batina, Stjepan Picek, and Mainack Mondal, editors, *Security, Privacy, and Applied Cryptography Engineering - 11th International Conference, SPACE 2021, Kolkata, India, December 10-13, 2021, Proceedings*, volume 13162 of *Lecture Notes in Computer Science*, pages 104–125. Springer, 2021.
17. Fabio Campos, Michael Meyer, Krijn Reijnders, and Marc Stöttinger. Patient zero & patient six: Zero-value and correlation attacks on CSIDH and SIKE. In Benjamin Smith and Huapeng Wu, editors, *Selected Areas in Cryptography - 29th International Conference, SAC 2022, Windsor, ON, Canada, August 24-26, 2022, Revised Selected Papers*, volume 13742 of *Lecture Notes in Computer Science*, pages 234–262. Springer, 2022.
18. Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023*, volume 14008 of *Lecture Notes in Computer Science*, pages 423–447. Springer, 2023.
19. Wouter Castryck, Thomas Decru, Luciano Maino, Chloe Martindale, Lorenz Panny, Giacomo Pope, Damien Robert, and Benjamin Wesolowski. Isogeny interpolation for elliptic curves, and applications. Presented by Wouter Castryck at the Sixteenth Algorithmic Number Theory Symposium, Massachusetts Institute of Technology, July 15, 2024. https://antsmath.org/ANTSXVI/schedule.html.
20. Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: an efficient post-quantum commutative group action. In Thomas Peyrin and Steven D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018 -*

24th International Conference on the Theory and Application of Cryptology and Information Security, 2018, Proceedings, Part III, volume 11274 of *Lecture Notes in Computer Science*, pages 395–427. Springer, 2018.

21. Wouter Castryck and Frederik Vercauteren. A polynomial time attack on instances of M-SIDH and FESTA. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part VII*, volume 14444 of *Lecture Notes in Computer Science*, pages 127–156. Springer, 2023.
22. Jorge Chavez-Saab, Maria Corte-Real Santos, Luca De Feo, Jonathan Komada Eriksen, Basil Hess, David Kohel, Antonin Leroux, Patrick Longa, Michael Meyer, Lorenz Panny, Sikhar Patranabis, Christophe Petit, Francisco Rodríguez Henríquez, Sina Schaeffler, and Benjamin Wesolowski. SQIsign. Technical report, National Institute of Standards and Technology, 2023. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures.
23. Craig Costello and Benjamin Smith. Montgomery curves and their arithmetic - the case of large characteristic fields. *J. Cryptogr. Eng.*, 8(3):227–240, 2018.
24. Pierrick Dartois. Fast computation of 2-isogenies in dimension 4 and cryptographic applications. *Journal of Algebra*, 683:449–514, 2025.
25. Luca De Feo, Tako Boris Fouotsa, and Lorenz Panny. Isogeny problems with level structure. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part VI*, volume 14656 of *Lecture Notes in Computer Science*, pages 181–204. Springer, 2024.
26. Luca De Feo, Antonin Leroux, Patrick Longa, and Benjamin Wesolowski. New algorithms for the Deuring correspondence - towards practical and secure SQISign signatures. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 659–690, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.
27. Luca De Feo, Nadia El Mrabet, Aymeric Genêt, Novak Kaluderovic, Natacha Linard de Guertechin, Simon Pontié, and Élise Tasso. SIKE channels: Zero-value side-channel attacks on SIKE. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(3):264–289, 2022.
28. Max Duparc, Antonin Leroux, and Sina Schaeffler. Qlapoty: Improved analysis and efficiency for quaternionic ideal to isogeny transformation. Cryptology ePrint Archive, Paper 2026/1700, 2026.
29. Tako Boris Fouotsa, Tomoki Moriya, and Christophe Petit. M-SIDH and MD-SIDH: countering SIDH attacks by masking information. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 282–309. Springer, 2023.
30. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554, Santa Barbara, CA, USA, August 15–19, 1999. Springer Berlin Heidelberg, Germany.
31. Steven D. Galbraith, Christophe Petit, Barak Shani, and Yan Bo Ti. On the security of supersingular isogeny cryptosystems. In Jung Hee Cheon and Tsuyoshi

Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 63–91, Hanoi, Vietnam, December 4–8, 2016. Springer Berlin Heidelberg, Germany.

32. Alexandre Gélin and Benjamin Wesolowski. Loop-abort faults on supersingular isogeny cryptosystems. In Tanja Lange and Tsuyoshi Takagi, editors, *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings*, volume 10346 of *Lecture Notes in Computer Science*, pages 93–106. Springer, 2017.
33. Ariana Goh, Chu-Wee Lim, and Yan Bo Ti. Generalising fault attacks to genus two isogeny cryptosystems. In *Workshop on Fault Detection and Tolerance in Cryptography, FDTC 2022, Virtual Event / Italy, September 16, 2022*, pages 38–49. IEEE, 2022.
34. Dina Hesse, Elisabeth Krahmer, Yi-Fu Lai, and Jonas Meers. A little less secure. In Nadia Heninger and Mike Rosulek, editors, *Advances in Cryptology – CRYPTO 2026*, pages 135–163, Cham, 2026. Springer Nature Switzerland.
35. Institute of Commercial Cryptography Standards, China. Call for Comments on Submission Requirements for Public-Key Cryptographic Algorithms (Draft) and Evaluation Criteria for Public-Key Cryptographic Algorithms (Draft), 2025.
36. David Jao, Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalali, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Joost Renes, Vladimir Soukharev, David Urbanik, Geovandro Pereira, Koray Karabina, and Aaron Hutchinson. SIKE. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions.
37. Ernst Kani. The number of curves of genus two with elliptic differentials. *Journal für die reine und angewandte Mathematik (Crelles Journal)*, 1997:93 – 122, 1997.
38. Hyeonhak Kim, Won Kim, Changmin Lee, Suhri Kim, Seokhie Hong, and Sangjin Lee. INKE: Isogeny-based PKE using intermediate curves. Cryptology ePrint Archive, Paper 2025/1458, 2025.
39. Yi-Fu Lai. Qlapoti+ and more: Optimizing isogeny-based signatures. Cryptology ePrint Archive, Paper 2026/1640, 2026.
40. JeongHwan Lee, Donghoe Heo, Hyeonhak Kim, GyuSang Kim, Suhri Kim, Heeseok Kim, and Seokhie Hong. Fault attack on SQIsign. In Markku-Juhani Saarinen and Daniel Smith-Tone, editors, *Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024, Part II*, pages 54–76, Oxford, UK, June 12–14, 2024. Springer, Cham, Switzerland.
41. Jason T. LeGrow and Aaron Hutchinson. (short paper) analysis of a strong fault attack on static/ephemeral CSIDH. In Toru Nakanishi and Ryo Nojima, editors, *Advances in Information and Computer Security - 16th International Workshop on Security, IWSEC 2021, Virtual Event, September 8-10, 2021, Proceedings*, volume 12835 of *Lecture Notes in Computer Science*, pages 216–226. Springer, 2021.
42. David Lim and Yan Bo Ti. Adaptive attack on static POKÉ keys. Cryptology ePrint Archive, Report 2025/1541, 2025.
43. Luciano Maino, Chloe Martindale, Lorenz Panny, Giacomo Pope, and Benjamin Wesolowski. A direct key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023*, volume 14008 of *Lecture Notes in Computer Science*, pages 448–471. Springer, 2023.
44. Kohei Nakagawa and Hiroshi Onuki. QFESTA: efficient algorithms and parameters for FESTA using quaternion algebras. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024 - 44th Annual International*

Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part V, volume 14924 of *Lecture Notes in Computer Science*, pages 75–106. Springer, 2024.
45. Phong Q Nguyen and Damien Stehlé. An lll algorithm with quadratic complexity. *SIAM Journal on Computing*, 39(3):874–903, 2009.
46. Damien Robert. Breaking SIDH in polynomial time. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023*, volume 14008 of *Lecture Notes in Computer Science*, pages 472–503. Springer, 2023.
47. Damien Robert. On the Efficient Representation of Isogenies - A Survey for NuTMiC 2024. In Andrzej Dabrowski, Josef Pieprzyk, and Jacek Pomykala, editors, *Number-Theoretic Methods in Cryptology - 4th International Conference, NuTMiC 2024, Szczecin, Poland, June 24-26, 2024, Revised Selected Papers*, volume 14966 of *Lecture Notes in Computer Science*, pages 3–84. Springer, 2024.
48. Élise Tasso, Luca De Feo, Nadia El Mrabet, and Simon Pontié. Resistance of isogeny-based cryptographic implementations to a fault attack. In Shivam Bhasin and Fabrizio De Santis, editors, *Constructive Side-Channel Analysis and Secure Design - 12th International Workshop, COSADE 2021, Lugano, Switzerland, October 25-27, 2021, Proceedings*, volume 12910 of *Lecture Notes in Computer Science*, pages 255–276. Springer, 2021.
49. Yan Bo Ti. Fault attack on supersingular isogeny cryptosystems. In Tanja Lange and Tsuyoshi Takagi, editors, *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017*, pages 107–122, Utrecht, The Netherlands, June 26–28, 2017. Springer, Cham, Switzerland.
50. Jacques Vélu. Isogénies entre courbes elliptiques. *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences, Série A*, 273:238–241, 7 1971.

A Probabilistic PRISM

The key recovery attack via fault injections on PRISM described in Section 4 was targeting the deterministic version of PRISM, as described in the original article [4]. Already, in the implementation referenced there, as well as in follow up works, versions of PRISM are presented that are probabilistic. This means that when the attack from Section 4 queries two signatures on the same message, the output points will lie on different elliptic curves, thus making subtraction of the points not possible. This seems to prevent our attack. However, in what follows we outline an alternative attack against the probabilistic PRISM (or salt-PRISM). While the attack requires 4 faults, two more compared to the deterministic version, it remains effective.

Recall the notation and set-up from the description of PRISM in Section 4.1. Let U, V denote

$$\begin{pmatrix} U \\ V \end{pmatrix} = \varphi_{\mathsf{sk}} \begin{pmatrix} P_0 \\ Q_0 \end{pmatrix}.$$

Thus,

$$\begin{pmatrix} P_{\mathsf{vk}} \\ Q_{\mathsf{vk}} \end{pmatrix} = M \cdot \begin{pmatrix} U \\ V \end{pmatrix}.$$

Suppose a signature query has been faulted that targets the value of m_{11} in M. Call this faulted value m'_{11}, and the resulting matrix M'. Then

$$\begin{pmatrix} P'_{\mathsf{sig}} \\ Q_{\mathsf{sig}} \end{pmatrix} := M' \cdot \sigma \begin{pmatrix} U \\ V \end{pmatrix},$$

where σ is the signature isogeny. Note, since P'_{sig} is a faulty point, we do not know σ directly, but only its degree $d := \deg \sigma$, and one non-faulted torsion point Q_{sig}. Let D denote the (unknown) determinant of M, and $\Delta := m_{11} - m'_{11}$ which in also unknown but which has few possibilities depending on the fault model used.

Since $P'_{\mathsf{sig}} = [m'_{11}]\sigma(U) + [m_{12}]\sigma(V)$ and $Q_{\mathsf{sig}} = [m_{21}]\sigma(U) + [m_{22}]\sigma(V)$, due to properties of pairings, we get that

$$e(P'_{\mathsf{sig}}, Q_{\mathsf{sig}}) = e(U, V)^{d(D - \Delta m_{22})} = e(P_{\mathsf{vk}}, Q_{\mathsf{vk}})^{d(D - \Delta m_{22})/D}.$$

Since $e(P'_{\mathsf{sig}}, Q_{\mathsf{sig}})$, $e(P_{\mathsf{vk}}, Q_{\mathsf{vk}})$, and $d = q(2^a - q)$ are known, we can recover $x_{22} := (D - \Delta m_{22})/D \pmod{2^a}$. Thus,

$$\frac{m_{22}}{D} = \frac{1 - x_{22}}{\Delta}.$$

With 3 more fault injections, we can analogously recover

$$\frac{m_{11}}{D}, \frac{m_{12}}{D}, \frac{m_{21}}{D},$$

resulting in the matrix

$$\begin{bmatrix} m_{11}/D & m_{12}/D \\ m_{21}/D & m_{22}/D \end{bmatrix} = \frac{1}{D} M.$$

This matrix will have determinant D^{-1}, meaning we can also recover D and accordingly scale the matrix to obtain M.

Note, we require that the Δ from each fault and the determinant D are invertible modulo 2^a, which is inline with the assumptions made in Section 4.

After recovering M, the rest of the secret information can be recovered as described in Section 4.

Complexity. This attack requires 4 first-order faults. After each successful fault, the post-processing requires only 2 Weil pairings. The dominating cost for post-processing instead comes from the need to validate the correct guess for each of the 4 Δ values. As in the other attacks, this validation is done by computing the corresponding higher dimensional isogeny to determine whether a viable solution is obtained. This means running the Kani computation $(\#\Delta)^4 \in [1, 2^{32}]$ times, depending on the fault model used.

Probability of success. Let γ denote the probability of success for the type of fault injection being used. Recall that we are assuming that all of the Δ are odd integers. This occurs with probability 1/2 each. Thus the probability of injecting a useful fault each time is $\gamma/2$.

B Legacy Issue in the Choice of Secret Ideals

During the preparation of this work, as pointed out by an anonymous reviewers of Asiacrypt, we observed a legacy preference in PRISM implementations for using a small-norm ideal as the secret key. The original PKC implementation of PRISM [4] directly samples a small-norm secret ideal. In contrast, the later salt-PRISM implementation [5] first samples a large-norm secret ideal (for uniformity) and then replaces it with a random equivalent ideal of small norm. Consequently, although the two implementations generate their secret keys differently, the degrees of the secret isogenies actually used by both implementations fall within the range considered in our attacks.

This preference for small-norm representatives does not appear to be accidental. In particular, the reference C implementation of salt-PRISM explicitly remarks that finding a small-norm equivalent ideal is motivated by performance. Our experiments confirm that this consideration is important for the original PKC implementation [4]. At Level 3, replacing its approximately 384-bit secret ideal by an approximately 767-bit secret ideal increases the signing time by roughly 40%. Profiling shows that most of this slowdown occurs in the lattice-reduction step (inside the `find_uv` subroutine) of the norm-solving procedure: the larger secret ideal leads to a larger intermediate ideal, making the corresponding lattice reduction substantially more expensive.

The effect is much milder when using Qlapoti [9], as in the more recent salt-PRISM implementation [5]. In preliminary experiments, the overall signing slowdown caused by using a large-norm secret ideal is less than 5% for all

parameter sets in the implementation of [5]. We attribute this to Qlapoti's different choice of the lattice reduction. Instead of the floating-point LLL as before, Qlapoti adapts a Nguyen–Stehlé-style LLL [45] which is several tens of times faster than the former one. While the lattice reduction using a large-norm secret ideal takes twice as long, it is invoked only once. Hence, the overhead is relatively small in the whole procedure. The follow-up works [39,28] keep the same choice and may have the same implication.

These observations suggest a potential countermeasure to our attack: one may retain a large-norm secret ideal instead of replacing it with a small-norm equivalent representative at mild computational overhead. This increases the degree of the corresponding secret isogeny beyond the range targeted by our attack, at the cost of increased secret-key storage and, depending on the norm-solving algorithm, some additional signing overhead. We note that using a large-norm secret ideal leads to larger integer sizes in the quaternion arithmetic, which might further complicate constant-time implementations of PRISM. This is because they are required to fix a sufficiently large upper bound for the involved integers, where increased size might lead to a larger overhead.