IACR Communications in Cryptology
Vol. 3, No. 2, 20 pages.

An analysis of a weakened version of PRISM

Jolijn Cottaar[1,2], Steven D. Galbraith[1,3], Luciano Maino[1,4] and Monika Trimoska[1,2]

[1] Theoretical Sciences Visiting Program (TSVP), Okinawa Institute of Science and Technology, Onna, Japan
[2] Eindhoven University of Technology, Eindhoven, The Netherlands
[3] University of Auckland, Auckland, New Zealand
[4] University of Birmingham, Birmingham, United Kingdom

Abstract. PRISM (PKC25) is a hash-and-sign signature scheme whose security relies on the hardness of computing large-prime-degree isogenies originating from a curve of unknown endomorphism ring. In PRISM, the degree of such isogenies is obtained by hashing messages onto a set of large odd integers that pass a primality test.
In this work, we investigate the impact of the choice of primality test on the security of PRISM. We first show that when a weak primality test is used, the assumption underlying the security proof in the standard model does not hold. We then extend our analysis to the assumption used in the security proof in the (quantum) random oracle model. In this setting, we argue that the Miller-Rabin test suffices and estimate the minimal number of iterations required for PRISM to achieve the desired security level, thus minimising signing costs.

Keywords: isogeny-based signatures · PRISM · primality testing

1 Introduction

Post-quantum cryptography is one of the most prominent areas of research within the field of cryptography. Its objective is to develop cryptosystems that are secure against adversaries with access to both classical and quantum computers. Among the quantum-resistant solutions, *isogeny-based* cryptography has received significant attention thanks to some of its appealing features such as its striking compactness (see, *e.g.*, [AAA+24, §1.1]).

Of particular interest are isogeny-based digital signature schemes. Arguably, the two most studied digital signature schemes are SQIsign [DKL+20] and PRISM [BBC+25]. SQIsign and PRISM build on top of some common building blocks. The key idea that makes both cryptosystems concretely practical is an efficient way to represent isogenies derived from the attacks on SIDH/SIKE [CD23, MMP+23, Rob23].

While SQIsign and PRISM enjoy similar performance, PRISM is conceptually easier. PRISM is a hash-and-sign signature that leverages the following trapdoor mechanism. Given an elliptic curve E, we consider the function $\mathcal{F}$ that, on input an isogeny $\varphi : E \to E'$ of degree a multiple of a large prime, outputs $\deg(\varphi)$. Evaluating $\mathcal{F}$ is efficient for appropriate parameter choices, however, inverting $\mathcal{F}$ is believed to be hard, unless $\mathrm{End}(E)$ is known. Following the standard hash-and-sign paradigm, the authors of PRISM utilise a hash function H that maps onto a set of large primes.

In [BBC+25], the authors argue that PRISM is secure both in the standard model and in the random oracle model (ROM). The security of PRISM in the standard model is

E-mail: jolijncottaar@gmail.com (Jolijn Cottaar), s.galbraith@auckland.ac.nz (Steven D. Galbraith), l.maino@bham.ac.uk (Luciano Maino), m.trimoska@tue.nl (Monika Trimoska)

This work is licensed under a "CC BY 4.0" license.

proven under the assumption that an adversary has access to an oracle, SPEDIO, which, on input a prime $2^{a-1} < q < 2^a$, returns an isogeny of special degree $q(2^a - q)$. In the ROM, the security of PRISM is derived from a milder assumption, which is also used to prove that PRISM can be modified to be secure in the quantum random oracle model (QROM) [BBC+26, §6].

While the proofs seem to be theoretically sound, relatively little attention has been devoted to explaining how to hash onto primes. Although there exists deterministic algorithms to check whether an integer is prime, such tests can incur prohibitive computational costs, especially if several tests must be run every time a message needs to be signed. For this reason, in PRISM, a probabilistic primality test is used.

Contributions

In this paper, we analyse the impact of primality tests on the security of PRISM. In particular, we have the following three results.

1. We construct a mechanism that uses the SPEDIO oracle to efficiently construct isogenies of arbitrary degree $< 2^{a-1}/a^2$. This shows that SPEDIO is significantly more powerful than it may first appear: it is not "just" an oracle to compute a random a-bit prime degree isogeny from a curve E but indeed computes isogenies of any degree $< 2^{a-1}/a^2$. In particular, this construction is oblivious to the primality test used: it is always possible to generate isogenies of arbitrary degree.

2. Equipped with this novel technique to generate isogenies of arbitrary degree, we show that, if a weak primality test is used, there exists a polynomial-time attack against the hardness assumption used in the standard model. This highlights a clear gap between the theoretical proof and the concrete security of PRISM.

3. We extend our analysis to the assumption used to argue security in the (Q)ROM. In this context, an adversary does not have access to SPEDIO. However, if a weak primality test is used, PRISM may fall short from its claimed security. In particular, we concretely assess the number of iterations to use in Miller-Rabin to attain the claimed security. Our analysis provides a useful tool to minimise the signing costs while still being secure.

None of these results yield a concrete attack against PRISM itself; rather, they provide guidance for secure and efficient parameter selection in practical instantiations.

Acknowledgments

This research was conducted while visiting the Okinawa Institute of Science and Technology (OIST) through the Theoretical Sciences Visiting Program (TSVP). We thank Riccardo Invernizzi for useful discussion on the proof-of-concept implementation of PRISM, and Wouter Castryck and Arthur Herlédan Le Merdy for spotting an inaccuracy that appeared in an earlier version of the statement of Theorem 2. We also thank the anonymous reviewers for their comments. This research has been supported by the Dutch government through the NWO XL project OCENW.XL.23.112 (ESCAPE).

2 Preliminaries

In this section, we collect some background knowledge. We first recall some known results about primality tests and primes in Section 2.1. Then, in Section 2.2, we discuss some basic facts about isogenies and elliptic curves. Finally, in Section 2.3, we briefly summarise the digital signature scheme PRISM [BBC+25].

Notation

We define λ to be the security parameter. Additionally, we denote $\log(x)$ to be the natural logarithm. When writing polynomial time, we will mean a quantity which is polynomial in $\log(p)$. Finally, we will use the notation $\widetilde{O}(f(x))$ to refer to $O(f(x)(\log(f(x)))^k)$, for some $k \geq 1$.

2.1 Primality tests and number theory

Deterministic primality tests are very expensive, and thus in practice one tends to use probabilistic primality tests. In this section we will introduce the compositeness/primality tests that we require for our attack. Say we want to figure out if an odd integer n is most likely prime or not.

We start with the Fermat compositeness test, which simply checks for a random base b coprime to n if Fermat's little theorem holds, i.e., if $b^{n-1} \equiv 1 \mod n$. Note that this is not a primality test, since there exists integers n that pass this test for all possible bases a coprime to n. These integers are called *Carmichael numbers*. We will mainly focus on numbers that do not survive the Fermat 2-test, i.e., for it tests if $2^{n-1} \equiv 1 \mod n$. We call these *2-Fermat pseudoprimes*.

Pomerance in [Pom81] has shown the following fact on the number of 2-Fermat pseudoprimes:

Theorem 1 (Upper bound for number of Fermat pseudoprimes [Pom81])**.** *For some upper bound X let $P(X) = \{n | 1 < n < X$, such that $2^{n-1} \equiv 1 \mod n\}$ and $L(X) = \exp(\log(X)\log(\log(\log(X)))/\log(\log(X)))$, then*

$$\exp(\log(X)^{5/14}) \leq P(X) \leq XL(X)^{-1/2}.$$

Pomerance subsequently conjectures that for large X:

$$P(X) = XL(x)^{-1+o(1)}.$$

Note that from this we can conclude that the fraction of 2-Fermat pseudoprimes under some bound X is $\exp(\log(X)^{5/14})/X \leq P(X)/X \leq L(X)^{-1/2}$. Specifically we will assume that $P(X)/X \approx L(X)^{-1}$ for large X.

The Miller-Rabin primality test is a widely-used primality test, introduced in its deterministic version by Miller [Mil76] and made probabilistic by Rabin [Rab80]. This test is based on the property that the only options for a square root of 1 modulo some prime are ± 1. Let $n \geq 1$ be an odd integer, which we can write as $n = 2^e d + 1$ for some integer e and odd integer d. If n is prime, then for any integer b with $1 \leq b < n$ we have either $b^d \equiv 1 \mod n$ or $b^{2^i d} \equiv 1 \mod n$ for some $0 \leq i < e$. The Miller-Rabin test then consists of checking these two conditions for a certain base b and outputting composite if both fail, and probably prime if one of the two hold. If composite is output, the associated b is called a *strong witness*, else b is called a *strong liar* and n is called a *strong pseudoprime*.

The runtime of the probabilistic Miller-Rabin test (with naive multiplication) is $O(k\log(n)^3)$ when running k iterations of the test on an integer n. Note that with some improvements one can get closer to a square instead of the cube.

Lemma 1 (Monier-Rabin Bound [Mon80])**.** *Let $n \neq 9$ be odd and composite, let $S(n)$ denote the number of $b \in [1, n-1]$ coprime to n that are strong liars. Then*

$$S(n) \leq \frac{\varphi(n)}{4},$$

where $\varphi(n)$ denotes the Euler totient function.

So the probability of incorrectly declaring a composite n to be probable prime, over the random choice of a coprime to n, is at most $1/4$. Hence, one usually uses $\lambda/2$ iterations for some security parameter λ to get a negligible probability of $2^{-\lambda}$ of having a strong pseudoprime instead of a prime.

Now consider a uniformly random k-bit integer n that passes t Miller-Rabin tests and is declared a probable prime. We are interested in the probability that n is composite (i.e., n is a strong pseudoprime and not prime). This is a subtly different question to the one covered by Lemma 1: the quantity n is not a given integer but we are rather looking at the probability that a uniformly random k-bit integer n that passes t Miller-Rabin tests. Following the work of Damgård, Landrock and Pomerance [DLP93], we denote this probability as $p_{k,t}$. Damgård et al. give the following bounds for $p_{k,t}$, one specifically for $t = 1$ and one for larger t.

Theorem 2 ([DLP93, Theorem 2, 3, 6, Corollary page 191])**.**

- *For $k \geq 2$ we have*
$$p_{k,1} < k^2 4^{2-\sqrt{k}}.$$
- *For k, t such that $k \geq 88, t = 2$ or $k \geq 21$, $3 \leq t \leq k/9$, we have*
$$p_{k,t} < k^{3/2} 2^t t^{-1/2} 4^{2-\sqrt{tk}}.$$
- *For k, t such that $k \geq 21, t \geq k/9$, we have*
$$p_{k,t} < 7/20k2^{-5t} + 1/7k^{15/4}2^{-k/2-2t} + 12k2^{-k/4-3t}.$$
- *For k, t such that $k \geq 21, t \geq k/4$, we have*
$$p_{k,t} < 1/7k^{15/4}2^{-k/2-2t}.$$

Recall that an integer is B-smooth if all its prime factors ℓ satisfy $\ell \leq B$. In our analysis we will need information on the proportion of B-smooth numbers.

Definition 1 (Dickman-de Bruijn function [Dic30])**.** The Dickman-de Bruijn function $\rho(u)$ is a continuous function that satisfies:

$$u\rho'(u) + \rho(u-1) = 0,$$

with initial conditions $\rho(u) = 1$ for $0 \leq u \leq 1$.

Dickman proved the following:

Theorem 3 (Number of B-smooth integers [Dic30])**.** *Let $\Psi(N, B)$ be the number of B-smooth integers below some bound N. Then we can estimate that*

$$\Psi(N, B) \approx N\rho\left(\frac{\log(N)}{\log(B)}\right).$$

We will have a quick look at prime numbers.

Theorem 4 (Prime Number Theorem)**.** *We denote the number of primes less or equal to X as $\pi(X)$. Then $\pi(X) \sim X/\log(X)$.*

Dirichlet then proved the following:

Theorem 5 (Dirichlet's Prime Theorem)**.** *Given two coprime integers a and b, there are infinitely many primes of the form $a + nb$ (i.e. in the arithmetic progression starting at a with difference b).*

He also showed that such arithmetic progressions have the same fraction of primes as the integers themselves.

Finally, we need some information on factorisation algorithms, specifically we are looking at factoring B-smooth numbers where B is very large. Naively we could just try to divide each prime up to B, which gives us a runtime of $O(\pi(B)) = O(B/\log(B))$ for each candidate. Bernstein [Ber04] has given an improved algorithm for the case when many candidates are being tested, but it is impractical for the sizes of B that we will be using. Hence we propose to apply general-purpose factoring methods, namely the Quadratic Sieve [Pom85] and the General Number Field Sieve [LLMP90]. Note that neither specifically focuses on B-smooth numbers and, even though the algorithms can be adapted to take this into account, for this paper we will just use the conjectured running times. The Quadratic Sieve is fastest up to numbers of about 322 bits and its runtime to factor a number X is conjectured at

$$\exp((1+o(1))(\ln(X)\ln(\ln(X)))^{1/2}). \tag{1}$$

The General Number Field Sieve has conjectured runtime of

$$\exp((64/9)^{1/3}(\ln(X)^{1/3}(\ln(\ln(X)))^{2/3})). \tag{2}$$

In the quantum setting, the cost of factoring an integer X lowers down to $\widetilde{O}(\log(X)^2)$ (see *e.g.*, [Sho94]).

2.2 Isogenies

We assume some familiarity with elliptic curves and their isogenies and refer the reader to [De 17, Sil09] for more details. In the rest of the paper, we will assume that $\mathbb{F}_{p^2}$ is our base field; in particular $p = f \cdot 2^e - 1$. Additionally, we fix E_0 to be the curve with equation $y^2 = x^3 + x$. Unless stated otherwise, all isogenies we will consider are always separable.

We first recall some basic facts about isogenies between elliptic curves. An isogeny $\varphi : E \to E'$ is uniquely defined by its kernel $\ker(\varphi)$; the degree of φ, denoted by $\deg(\varphi)$, coincides with the cardinality of $\ker(\varphi)$. When the endomorphism ring of E is known, there exists polynomial-time algorithms to compute isogenies of arbitrary degree (see, *e.g.*, [KLPT14, Wes22]). If this knowledge is not given, the situation drastically changes.

When $\deg(\varphi)$ is composite, it is common to compute the isogeny φ as a chain of prime-degree isogenies. This technique reduces the computational complexity of computing high-degree isogenies. We now review some methods to compute ℓ-isogenies, *i.e.*, isogenies of degree ℓ, where ℓ is a prime, in the case of unknown endomorphism ring; this summary follows the discussion in [BBC+25, §4.4].

The most standard approach is to use Vélu's formulae [Vél71], whose complexity is $O(\ell)$ in the field of definition of its kernel generators. In general, a kernel generator is defined over a field extension of degree ℓ of the base field [EPSV24]. Since performing operations over such a large extension incurs a $\widetilde{O}(\ell)$ overhead, Vélu's formulae require $\widetilde{O}(\ell^2)$ base field operations. Vélu's formulae were later improved in [BDLS20], where an algorithm that runs in $\widetilde{O}(\ell^{1/2})$ field operations is shown. However, taking into account the cost of working with large field extensions, the resulting expected complexity is $\widetilde{O}(\ell^{3/2})$.

For Vélu's formulae and their improvement, we have thus assumed that a kernel generator is already provided. However, if our goal is to compute any isogeny of degree ℓ, we will first have to sample a point of order ℓ. As argued in [BBC+25, §4.4], this task has complexity $\widetilde{O}(\ell^2)$. Hence, even under the improvement provided by [BDLS20], the complexity of computing an ℓ-isogeny is still $\widetilde{O}(\ell^2)$. In [BBC+25, §4.4], some alternative methods to compute ℓ-isogenies are provided. However, they do not outperform the claimed

complexity $\widetilde{O}(\ell^2)$. In the rest of the paper (especially in Section 5), we will assume that, for a random prime ℓ, computing an ℓ-isogeny has complexity $\widetilde{O}(\ell^2)$.

We now briefly recall the notion of *pushforward*. Let $\varphi_1 : E \to E_1$ and $\varphi_2 : E \to E_2$ be two isogenies such that $\gcd(\deg(\varphi_1), \deg(\varphi_2)) = 1$. We denote by $[\varphi_1]_*\varphi_2 : E_1 \to E'$ the pushforward isogeny of φ_2 under φ_1, i.e., the separable isogeny such that $\ker([\varphi_1]_*\varphi_2) = \varphi_1(\ker(\varphi_2))$.

In the rest of the manuscript, we will deal with different ways to "represent" isogenies. We briefly recall the definition of *isogeny representation*; see, for instance [BDD+24, Definition 1].

Definition 2. Let $\mathbb{F}_q$ be a finite field. An *isogeny evaluator* is a pair of polynomial-time algorithms:

- $\texttt{valid}(D)$ taking as input a string $D \in \{0,1\}^*$ and outputting either a symbol $\perp$ or a triple (E, E', d); in the latter case, E and E' are elliptic curves defined over $\mathbb{F}_q$ and there exists an isogeny $\varphi : E \to E'$ of degree d.
- $\texttt{eval}(D, P)$ taking as input a string $D \in \{0,1\}^*$ and a point $P \in E(\mathbb{F}_{q^k})$; if $\texttt{valid}(D) = (E, E', d)$, it outputs the image point $\varphi(P) \in E'(\mathbb{F}_{q^k})$, otherwise the output is undefined.

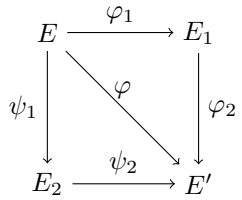
In the case that D is of size polynomial in $\log(q)$ and $\log(d)$ and that $\texttt{valid}(D)$ does not output $\perp$, the string D is called an *efficient representation* of φ.

In [Rob24], Robert shows that any isogeny $\varphi : E \to E'$ can be efficiently represented leveraging its evaluation on a large-enough torsion basis $E[N]$, where N is smooth and the N-torsion is *accessible*, *i.e.* $N = \prod \ell_i^{e_i}$, ℓ_i is polynomial in $\log(p)$ and $E[\ell_i^{e_i}]$ is defined over a field extension of degree $O(\log(p))$ (see also [Rob24, Appendix A]). To be more precise, the data $(E, E', \varphi(E[N]))$ suffices to construct an efficient isogeny representation if $N^2 > 4\deg(\varphi)$, N is smooth and the N-torsion is accessible. Indeed, using $(E, E', E[N])$, we can construct a g-dimensional isogeny Φ that embeds φ, where $g \in \{2, 4, 8\}$. Henceforth, we will refer to this representation as a *high dimensional (HD) representation* of φ.

Example 1. Let $p = f \cdot 2^e - 1$ be a prime, and consider an isogeny $\varphi : E \to E'$ of degree $q(2^a - q)$, for some odd q and $0 < a \leq e$. A direct application of Kani's Lemma [Kan97] shows that φ can be embedded in the 2-dimensional isogeny Φ with kernel $\{([q]P, \varphi(P)) \mid P \in E[2^a]\}$. To be more specific, the isogeny Φ can be represented via the following isogeny matrix

$$\Phi = \begin{pmatrix} \varphi_1 & -\widehat{\varphi}_2 \\ \psi_1 & \widehat{\psi}_2 \end{pmatrix} : E \times E' \to E_1 \times E_2,$$

where the isogenies φ_i and ψ_i fit in the following commutative diagram and $\deg(\psi_1) = \deg(\varphi_2) = 2^n - \deg(\varphi_1) = 2^n - \deg(\psi_2) = 2^n - q$.



This construction provides an HD representation of φ that is crucial in many cryptosystems, *e.g.* [BMP23, NO24, DMPR24, NOC+24, DF24].

Given an HD representation of $\varphi : E \to E'$ such that $\deg(\varphi) = m_1 \cdot m_2$ and $\gcd(m_1, m_2) = 1$, it is possible to extract an efficient representation of $\varphi_1 : E \to E_1$, where $\ker(\varphi_1) = \ker(\varphi)[m_1] = \ker(\varphi) \cap E[m_1]$ [Rob24, Proposition 6.11]. Additionally, given HD representations of $\varphi_1 : E \to E_1$ and $\varphi_2 : E \to E_2$, where $\gcd(\deg(\varphi_1), \deg(\varphi_2)) = 1$, we can obtain an HD representation of $[\varphi_1]_*\varphi_2 : E \to E'$; computing such an HD representation is *efficient*, *i.e.*, it requires a number of base field operations that is polynomial in $\log(\deg(\varphi_1))$, $\log(\deg(\varphi_2))$ and $\log(p)$ [Rob24, Proposition 6.15].

2.3 PRISM

In this section, we recall PRISM [BBC+25]. PRISM builds on top of an identification protocol, which is then transformed into a signature via the hash-and-sign paradigm. For ease of understanding, we will briefly describe the resulting signature scheme, simplifying some of the more complicated isogeny computations and highlighting the role of the hash function.

We start by fixing $0 < a \leq e$; in practice $a \approx 2\lambda - \log(\lambda)$. In PRISM, a secret key consists of an HD representation of an isogeny $\varphi_{\mathsf{sk}} : E_0 \to E_{\mathsf{pk}}$. The corresponding public key is the curve E_{pk} together with a deterministically generated basis $(P_{\mathsf{pk}}, Q_{\mathsf{pk}})$ of $E_{\mathsf{pk}}[2^a]$. To sign a message $\mathsf{msg} \in \{0,1\}^*$, the signer generates an isogeny $\varphi_{\mathsf{sign}} : E_{\mathsf{pk}} \to E_{\mathsf{sign}}$ of degree $q(2^a - q)$, where $2^{a-1} < q < 2^a$ is a prime obtained from msg. To be more precise, the prime q is the output of a hash function $\mathsf{H}_{\mathsf{PRISM}}$, which we will formally define in the remaining part of this section. The signature corresponding to msg is then given by the triple $(E_{\mathsf{sign}}, P_{\mathsf{sign}}, Q_{\mathsf{sign}})$, where $P_{\mathsf{sign}} = \varphi_{\mathsf{sign}}(P_{\mathsf{pk}})$ and $Q_{\mathsf{sign}} = \varphi_{\mathsf{sign}}(Q_{\mathsf{pk}})$. The verification then utilises $(E_{\mathsf{sign}}, P_{\mathsf{sign}}, Q_{\mathsf{sign}})$ and the public information $(E_{\mathsf{pk}}, P_{\mathsf{pk}}, Q_{\mathsf{pk}})$ to recover an HD representation of an isogeny of degree $q(2^a - q)$ connecting E_{pk} and E_{sign}.

We now explain how PRISM determines the degree of the signature isogeny φ_{sign}, *i.e.*, how to obtain q such that $\deg(\varphi_{\mathsf{sign}}) = q(2^a - q)$. In [BBC+25], the authors introduce a hash function $\mathsf{H}_{\mathsf{PRISM}}$ that, on input a supersingular elliptic curve E and a message msg, outputs $q \in \mathsf{Primes}_a := \{q \mid 2^{a-1} < q < 2^a,\ q \text{ is prime}\}$. To be more specific, let $\mathsf{H}_{a-2} : \{0,1\}^* \to [0, 2^{a-2})$ be a hash cryptographic hash function. On input a supersingular elliptic curve E and a message msg, $\mathsf{H}_{\mathsf{PRISM}}$ iteratively computes $q := 2^{a-1} + 2\mathsf{H}_{a-2}(E \| \mathsf{msg} \| \mathsf{counter}) + 1$ for increasing values of $\mathsf{counter}$ until q passes a primality test. The primality test used for $\mathsf{H}_{\mathsf{PRISM}}$ is the focus of this paper.

We give a more formal description of PRISM in Figure 1. This description relies on the following building block; the quantity ℓ_{large} is a large prime, which is a fixed parameter.

> $(\varphi : E \to E') \leftarrow \mathsf{GenIsogeny}(E, \mathrm{End}(E), d)$. On input a supersingular elliptic curve E, together with a description of its endomorphism ring $\mathrm{End}(E)$, and an odd integer d, this algorithm outputs an HD representation of a uniformly distributed isogeny $\varphi : E \to E'$ of degree d. This algorithm was implemented in [BBC+25, Algorithm 1].

2.3.1 The security of PRISM.

In [BBC+25], the authors introduce the following oracle to prove the security of PRISM in the standard model.

Definition 3. A *special degree isogeny oracle* (SPEDIO) is an oracle that, on input a supersingular elliptic curve E and $q \in \mathsf{Primes}_a$, outputs an HD representation of a uniformly distributed isogeny $\varphi : E \to E'$ of degree $q(2^a - q)$.

The authors of PRISM prove the existential unforgeability of the signature scheme in the standard model under the following assumption [BBC+25, Proposition 2].

$\mathsf{KeyGen}(1^\lambda)$:

1: $\varphi_{\mathsf{sk}} \leftarrow \mathsf{GenIsogeny}(E_0, \mathrm{End}(E_0), \ell_{\mathsf{large}})$
2: Set E_{pk} to be the codomain of φ_{sk}.
3: $\mathsf{sk} \leftarrow \varphi_{\mathsf{sk}}$; $\mathsf{pk} \leftarrow E_{\mathsf{pk}}$
4: **return** $(\mathsf{sk}, \mathsf{pk})$

$\mathsf{Sign}(\mathsf{msg}, \mathsf{sk})$:

1: Using $\varphi_{\mathsf{sk}} : E_0 \to E_{\mathsf{pk}}$ compute $\mathrm{End}(E_{\mathsf{pk}})$.
2: Deterministically compute a basis $(P_{\mathsf{pk}}, Q_{\mathsf{pk}})$ of $E_{\mathsf{pk}}[2^a]$.
3: $q \leftarrow \mathsf{H}_{\mathsf{PRISM}}(E_{\mathsf{pk}}, \mathsf{msg})$
4: $\varphi_{\mathsf{sign}} \leftarrow \mathsf{GenIsogeny}(E_{\mathsf{pk}}, \mathrm{End}(E_{\mathsf{pk}}), q(2^a - q))$
5: Set E_{sign} to be the codomain of φ_{sign}.
6: $P_{\mathsf{sign}}, Q_{\mathsf{sign}} \leftarrow \varphi_{\mathsf{sign}}(P_{\mathsf{pk}}, Q_{\mathsf{pk}})$
7: **return** $E_{\mathsf{sign}}, P_{\mathsf{sign}}, Q_{\mathsf{sign}}$

$\mathsf{Verify}(\mathsf{msg}, E_{\mathsf{sign}}, P_{\mathsf{sign}}, Q_{\mathsf{sign}}, \mathsf{pk})$:

1: Deterministically compute a basis $(P_{\mathsf{pk}}, Q_{\mathsf{pk}})$ of $E_{\mathsf{pk}}[2^a]$.
2: $\tilde{q} \leftarrow \mathsf{H}_{\mathsf{PRISM}}(E_{\mathsf{pk}}, \mathsf{msg})$
3: $\sigma \leftarrow (E_{\mathsf{pk}}, P_{\mathsf{pk}}, Q_{\mathsf{pk}}, E_{\mathsf{sign}}, P_{\mathsf{sign}}, Q_{\mathsf{sign}})$
4: **assert** $\tilde{q} \in \mathsf{Primes}_a$;
5: Check that σ is an HD representation of an isogeny of degree $\tilde{q}(2^a - \tilde{q})$.
6: **return** True

Figure 1: PRISM signature scheme.

Problem 1. *Assuming access to* SPEDIO *and given a random supersingular elliptic curve E, output an isogeny of degree $q'(2^a - q')$ with $q' \in \mathsf{Primes}_a$ different from all degrees formerly generated by* SPEDIO*.*

Additionally, they prove the existential unforgeability of the signature scheme in the random oracle model (ROM) under this weaker assumption [BBC+25, Proposition 3].

Problem 2. *Given a random curve E, a set of N isogenies $\{\varphi_i : E \to E_i\}_{i=1}^N$ of degree $q_i(2^a - q_i)$ for q_i uniformly random in Primes_a and φ_i uniformly random among the isogenies of degree $q_i(2^a - q_i)$, and a prime $\bar{q}$ uniformly random in $\mathsf{Primes}_a \setminus \{q_i\}_{i=1}^N$, give an HD representation of an isogeny $\varphi : E \to E'$ of degree $\bar{q}(2^a - \bar{q})$.*

2.3.2 A variant of PRISM.

In [BBC+26, §6], the authors describe a variant of PRISM, called salted-PRISM, which is existential unforgeable in the quantum random oracle model (QROM) under the assumption that Problem 2 holds. The main differences compared to Section 2.3 are the following:

1. $a = \lambda + 64$;
2. the quantity q is sampled in a different way: rather than using $\mathsf{H}_{\mathsf{PRISM}}$, the quantity q is obtained running $2^{a-1} + 2\mathsf{H}_{a-2}(E \| \mathsf{msg} \| \mathsf{salt}) + 1$ for different $\mathsf{salt} \in \{0,1\}^{\lambda+64}$ until a prime is found.

For the sake of clarity of exposition, we give a more formal description of the digital signature scheme in Figure 2.

3 Generating isogenies of arbitrary degree via SPEDIO

We now describe how to efficiently generate isogenies $\varphi : E \to E'$ of arbitrary degree $m < 2^{a-1}/a^2$ even when $\mathrm{End}(E)$ is unknown. This can be done querying the oracle SPEDIO on a specifically crafted prime q'. This shows that a SPEDIO oracle is more powerful than it may appear. It is not just outputting HD representations of isogenies of degree $q(2^a - q)$, for some prime q, but it can also be leveraged to compute m-isogenies for any m up to $2^{a-1}/a^2$.

The core result of this section is contained in Theorem 6. Its proof relies on several technical lemmata, which we collect in Section 3.1. Since our end goal is to analyse

$\mathsf{KeyGen}(1^\lambda)$:

1: $\varphi_{\mathsf{sk}} \leftarrow \mathsf{GenIsogeny}(E_0, \mathrm{End}(E_0), \ell_{\mathsf{large}})$
2: Set E_{pk} to be the codomain of φ_{sk}.
3: $\mathsf{sk} \leftarrow \varphi_{\mathsf{sk}}$; $\mathsf{pk} \leftarrow E_{\mathsf{pk}}$
4: **return** $(\mathsf{sk}, \mathsf{pk})$

$\mathsf{Sign}(\mathsf{msg}, \mathsf{sk})$:

1: Using $\varphi_{\mathsf{sk}} : E_0 \to E_{\mathsf{pk}}$ compute $\mathrm{End}(E_{\mathsf{pk}})$.
2: Deterministically compute a basis $(P_{\mathsf{pk}}, Q_{\mathsf{pk}})$ of $E_{\mathsf{pk}}[2^a]$.
3: **repeat**
4: $\quad \mathsf{salt} \leftarrow_{\$} \{0,1\}^{\lambda+64}$
5: $\quad q \leftarrow 2^{a-1} + 2\mathsf{H}_{a-2}(E_{\mathsf{pk}} \| \mathsf{msg} \| \mathsf{salt}) + 1$
6: **until** $q \in \mathsf{Primes}_a$
7: $\varphi_{\mathsf{sign}} \leftarrow \mathsf{GenIsogeny}(E_{\mathsf{pk}}, \mathrm{End}(E_{\mathsf{pk}}), q(2^a - q))$
8: Set E_{sign} to be the codomain of φ_{sign}.
9: $P_{\mathsf{sign}}, Q_{\mathsf{sign}} \leftarrow \varphi_{\mathsf{sign}}(P_{\mathsf{pk}}, Q_{\mathsf{pk}})$
10: **return** $E_{\mathsf{sign}}, P_{\mathsf{sign}}, Q_{\mathsf{sign}}, \mathsf{salt}$

$\mathsf{Verify}(\mathsf{msg}, E_{\mathsf{sign}}, P_{\mathsf{sign}}, Q_{\mathsf{sign}}, \mathsf{salt}, \mathsf{pk})$:

1: Deterministically compute a basis $(P_{\mathsf{pk}}, Q_{\mathsf{pk}})$ of $E_{\mathsf{pk}}[2^a]$.
2: $\tilde{q} \leftarrow 2^{a-1} + 2\mathsf{H}_{a-2}(E_{\mathsf{pk}} \| \mathsf{msg} \| \mathsf{salt}) + 1$
3: $\sigma \leftarrow (E_{\mathsf{pk}}, P_{\mathsf{pk}}, Q_{\mathsf{pk}}, E_{\mathsf{sign}}, P_{\mathsf{sign}}, Q_{\mathsf{sign}})$
4: **assert** $\tilde{q} \in \mathsf{Primes}_a$;
5: Check that σ is an HD representation of an isogeny of degree $\tilde{q}(2^a - \tilde{q})$.
6: **return** True

Figure 2: salted-PRISM signature scheme.

the security of PRISM in the standard model (see Section 2.3.1), we will assume that $a \approx 2\lambda - \log(\lambda)$.

Theorem 6. *Let E be a supersingular elliptic curve and let $1 < m < 2^{a-1}/a^2$. Then, we can compute a uniformly random isogeny $\varphi : E \to E'$ of degree m by querying* SPEDIO *only once in polynomial time.*

The high-level idea behind Theorem 6 is as follows: First we compute a prime $q' = 2^a \pmod m$, so that $m \mid (2^a - q')$. We then call the SPEDIO oracle on q' to get a representation of an isogeny of degree $q'(2^a - q')$. It is the isogeny of degree $2^a - q'$ we are interested in. We then use techniques of Robert to extract an isogeny of degree m from this (to efficiently do that we introduce a power of 3 to ensure that $(2^a - q')/m$ is a smooth number).

3.1 Proof of Theorem 6

To prove Theorem 6, we first need to show that for any odd integer $n < 2^{a-1}/a^2$, we can compute a prime q such that $2^a = q \pmod n$.

Proposition 1. *Let n be an odd integer such that $n < 2^{a-1}/a^2$. Then* $\mathsf{PrimesModulo}(n)$ *(Algorithm 1) outputs a result with overwhelming probability. It requires $O(a^2)$ primality tests.*

The proof of Proposition 1 hinges on Lemma 3, which is a consequence of Chernoff's bound [Sho09, Theorem 8.24(ii)].

Lemma 2. *[Chernoff's bound] Let $\{X_i\}_{i=1}^t$ be a sequence of Bernoulli random variables with probability p. Define $\overline{X} := 1/t \sum_{i=1}^t X_i$. Then for every $\epsilon > 0$ we have $Pr[\overline{X} - p \leq -\epsilon] \leq e^{-t\epsilon^2/(2p)}$.*

Lemma 3. *Let $\{X_i\}_{i=1}^t$ be a sequence of Bernoulli random variables with probability p. If $t = \lceil 1/p^2 \rceil$, then $Pr[\sum_{i=1}^t X_i \geq 1] \geq 1 - e \cdot e^{-1/(2p)}$.*

Proof of Lemma 3. Let $\{X_i\}_{i=1}^t$ be a sequence of Bernoulli random variables with probability p. From Lemma 2 we now know that $Pr[1/t \sum_{i=1}^t X_i - p \leq -\epsilon] \leq e^{-t\epsilon^2/(2p)}$.

Algorithm 1 PrimesModulo(n)

Input: An odd integer n, such that $n < 2^{a-1}/a^2$.
Output: A prime $2^{a-1} < q' < 2^a$ such that $q' = 2^a \pmod n$.

1: $b \leftarrow 2^a \pmod n$
2: $q' \leftarrow n\lceil \frac{2^{a-1}+1}{n} \rceil + b$
3: **while** $q' < 2^a$ **do**
4: **if** q' is prime **then**
5: **return** q'
6: $q' \leftarrow q' + n$
7: **return** $\perp$

Let us take the left-hand side and rewrite:

$$Pr\left[1/t\sum_{i=1}^{t} X_i - p \leq -\epsilon\right] = Pr\left[\sum_{i=1}^{t} X_i \leq pt - t\epsilon\right].$$

Now to equate this to $Pr[\sum_{i=1}^{t} X_i \leq 1]$, we need $pt - t\epsilon = 1$, thus $\epsilon = p - 1/t$.

So we have

$$Pr\left[\sum_{i=1}^{t} X_i \leq 1\right] \leq e^{-t(p-1/t)^2/(2p)}$$

and now let $t = 1/p^2$ we get:

$$Pr\left[\sum_{i=1}^{t} X_i \leq 1\right] \leq e^{-1/2p+1-p/2} \leq e \cdot e^{-1/(2p)}.$$

Now it is an easy final step:

$$Pr\left[\sum_{i=1}^{t} X_i \geq 1\right] = 1 - Pr\left[\sum_{i=1}^{t} X_i \leq 1\right] \geq 1 - e \cdot e^{-1/(2p)}.$$

□

Proof of Proposition 1. Since $\gcd(2^a, n) = 1$, the arithmetic progression $q \equiv 2^a \pmod n$ contains primes. The probability that an integer q' defined in Line 6 is a prime is approximately $1/a$ by Theorem 5. Applying Lemma 3 with $p = 1/a$, we can conclude that if we do a^2 iterations of the while loop, we will have found at least one prime with probability $1 - e \cdot e^{-a/2}$. Since $a \approx 2\lambda - \log(\lambda)$ (see Section 2.3), the term $e^{-a/2}$ is negligible. The number of iterations in the loop is at most equal to the number of elements in equivalence classes modulo n in an interval of $2^{a-1} - 1$ integers, so $\lceil \frac{2^{a-1}-1}{n} \rceil > a^2$ by the choice of n. □

Before proving Theorem 6, we need to slightly generalise [Rob24, Proposition 6.11]. The aim of Lemma 4 is to extract an isogeny of degree m_1 from $\varphi : E \to E'$ of degree $m_1 \cdot m_2$ even when $\gcd(m_1, m_2) \neq 1$.

Lemma 4. *Given an HD representation of an isogeny $\varphi : E \to E'$ of degree $m_1 \cdot m_2$, where the largest prime divisor of $d = \gcd(m_1, m_2)$ is polynomial in $\log(p)$, it is possible to extract an HD representation of the isogeny $\varphi_1 : E \to E_1$ where $\ker(\varphi_1) = \ker(\varphi)[m_1]$ in polynomial time in $\log(p)$ and $\log(d)$.*

Proof of Lemma 4. If $d = 1$, we can simply use [Rob24, Proposition 6.11]. Suppose now that $d > 1$. By [Rob24, Proposition 6.11], we can extract an HD representation of the isogeny with kernel $\ker(\varphi)[m_1 d]$. Hence, without loss of generality, we can assume that $m_2 = \gcd(m_1, m_2)$ and that the largest factor of m_2 is polynomial in $\log(p)$.

Let ℓ be a prime factor of m_2. If ℓ^2 divides m_2 and $[\ell]$ divides φ,[1] we redefine φ to be $\varphi/[\ell]$ and m_2 to be m_2/ℓ^2. Here, by redefining φ to be $\varphi/[\ell]$, we mean that we only need to change the HD representation of φ to remove the division by ℓ and adapt the degree of φ. If $\gcd(\ell^2, m_2) = \ell$ and $[\ell]$ divides φ, we redefine φ to be $\psi \circ \varphi/[\ell]$, where $\psi : E' \to E''$ is an isogeny of degree ℓ and m_2 to be m_2/ℓ. Now, up to repeating the same process as above, we assume that $[\ell]$ does not divide φ.

Let K be a generator of $\ker(\widehat{\varphi})[\ell]$; this K can be efficiently computed since ℓ is smooth. We now compute $\psi : E' \to E''$ to be the ℓ-isogeny with kernel $\langle K \rangle$. From the HD representation of φ, we extract an HD representation of $\frac{\psi \circ \varphi}{[\ell]} : E \to E''$. We redefine φ to be $\frac{\psi \circ \varphi}{[\ell]}$ and m_2 to be m_2/ℓ.

We repeat this process for all the prime factors of m_2 until $m_2 = 1$. The complexity of this algorithm is a direct consequence of the fact that $\gcd(\deg(\varphi)/m_1, m_1)$ is smooth by hypothesis. □

Now we have all the tools needed to prove Theorem 6.

Proof of Theorem 6. We first assume that m is odd. We define $n := \max\{t \mid 3^t \cdot m < 2^{a-1}/(a^2)\}$. By Proposition 1, we compute a prime $2^{a-1} < q' < 2^a$ such that $q' = 2^a \bmod 3^n m$ in polynomial time in $\log(p)$. Observe that $\frac{2^a - q'}{3^n m}$ is a smooth quantity since $a \in O(\log(p))$. We now query SPEDIO on q' and obtain an HD representation of a uniformly random isogeny $\psi : E \to E'$ of degree $q'(2^a - q')$. By Lemma 4, we then extract the isogeny $\varphi : E \to E'$ with kernel $\ker(\psi)[m]$. The uniform distribution of φ derives from the fact that ψ is uniformly random and the factoring algorithm in the proof of Lemma 4.

If m is even, we write $m = m' \cdot 2^s$, where m' is odd. Using the technique above, we can compute an HD representation of a uniformly distributed isogeny $\varphi' : E \to E'$ of degree m'. We then obtain an isogeny $\varphi : E \to E''$ of degree m by composing φ' with a random isogeny $\psi : E' \to E''$ of degree 2^s. Such an isogeny ψ can be efficiently computed by chaining s random isogenies of degree 2.

Overall, the complexity of computing such an m-isogeny is a direct consequence of the complexity of all the steps involved. □

4 An attack on the computational assumption in the standard model for a weak version of PRISM

As mentioned in Section 2.3.1, the standard model security proof of PRISM is based on Problem 1. This assumption provides a strong SPEDIO oracle that can be queried on any chosen prime $2^{a-1} < q < 2^a$. It also allows the forgery to be with respect to any chosen prime q. We now consider a weakened version of PRISM where the primality of q is checked via a "compositeness" test (see Definition 4). For example, the compositeness test could be to simply check $2^{q-1} \equiv 1 \pmod q$ (a base-2 Fermat test).

At a high level the approach of this section is the following:

1. Generate an a-bit non-prime q that will pass the weak compositeness test. For example, construct a Carmichael number q in the interval $[2^{a-1}, 2^a]$. Factorise q as $n_1 n_2$ where n_1, n_2 are both sufficiently large (we need $n_1, n_2 < 2^a/a^2$, and hence $n_1, n_2 > a^2$). We do not need n_1 or n_2 to be fully factored.

[1] Checking that $[\ell]$ divides φ is efficient as ℓ is smooth.

2. Factorise $2^a - q$ as $n_3 n_4$ where $n_3, n_4 > a^2$. We do not need n_3 or n_4 to be fully factored.

3. Define $d_{1,2} := \gcd(n_1, n_2)$, $d_{3,4} := \gcd(n_3, n_4)$, $n_1' := n_1/d_{1,2}$, $n_2' := n_2/d_{1,2}$, $n_3' := n_3/d_{3,4}$ and $n_4' := n_4/d_{3,4}$.

4. For $i = 1, \ldots, 4$, compute isogenies $\varphi_i : E \to E_i$ of degree n_i' applying Theorem 6.

5. Compute an HD representation of $\psi_1 := [\varphi_1]_* \varphi_2$ using [Rob24, Proposition 6.15]. This is an isogeny of degree $n_1' n_2' = \frac{q}{d_{1,2}^2}$. Similarly, compute $\psi_2 := [\varphi_3]_* \varphi_4$. This is an isogeny of degree $n_3' n_4' = \frac{2^a - q}{d_{3,4}^2}$.

6. Finally, compute an HD representation of $\varphi = [d_{1,2} \cdot d_{3,4}] \circ [\psi_1]_* \psi_2$ using [Rob24, Proposition 6.15]. This is an isogeny of degree $q(2^a - q)$ as required.

Depending on the weak compositeness test used, there are various known constructions of composites that pass. For an overview see [PSW80].

Definition 4. A *weak compositeness test* is one that correctly accepts primes but, for sufficiently large a, there is an efficient randomised algorithm $\mathsf{Gen}^{\mathsf{comp}}$ to construct polynomially-many composite numbers $q \in (2^{a-1}, 2^a)$ that are accepted by the test and are such that $2^a - q$ is not prime. In addition we require the algorithm $\mathsf{Gen}^{\mathsf{comp}}$ to return the factorisation (or a partial factorisation) of the numbers q, and we require the numbers $2^a - q$ to behave like uniformly distributed integers from the point of view of small prime divisors (namely, for any prime $3 < d \leq a^2$, the probability that $d \mid (2^a - q)$ is $1/d$, over the outputs of $\mathsf{Gen}^{\mathsf{comp}}$).

We now give an example of a weak compositeness test. Suppose the test is the Fermat test to base 2. Our algorithm $\mathsf{Gen}^{\mathsf{comp}}$ will generate an odd Carmichael number, which will always pass this test. Let $x \in \mathbb{N}$ be such that $p_1 = 6x + 1, p_2 = 12x + 1, p_3 = 18x + 1$ are all primes. Then $n = p_1 p_2 p_3$ is a Carmichael number. Note that $n \approx (6x)(12x)(18x) = 2^4 3^4 x^3$. We now explain that there are more than enough x to choose. Since $(2^{a-1}/(2^4 3^4))^{1/3} < x < (2^a/(2^4 3^4))^{1/3}$ it suffices to bound the size of this interval, which is $c2^{a/3} - c2^{(a-1)/3} = c(2^{1/3} - 1)2^{(a-1)/3}$ for $c = 1/(2^4 3^4)^{1/3}$. Hence there are exponentially-many x to choose. Trying random x in this interval until p_1, p_2, p_3 are prime will on average require $O(a^3)$ attempts, and checking primality is polynomial-time. Note that the factorisation of q is known, by construction. Finally, there is not much reason why the numbers $2^a - q$ should behave any differently from random integers of this size from the point of view of divisibility, apart from the known condition $q \equiv 1 \pmod{36}$ and so $2^a - q \equiv 2^a - 1 \pmod{36}$.

Similarly, if $p_1 = 6x + 1, p_2 = 12x + 1, p_3 = 18x + 1, p_4 = 36x + 1, p_5 = 72x + 1$ are all prime then $n = p_1 p_2 p_3 p_4 p_5$ is a Carmichael number. Note that in these cases, the factorisation of q is also known by construction.

We now state our theorem.

Theorem 7. *There is a polynomial-time algorithm for Problem 1 when the verifier uses a weak compositeness test (as in Definition 4).*

Proof. First, run the algorithm $\mathsf{Gen}^{\mathsf{comp}}$ from Definition 4 to generate polynomially-many composite numbers q and their factorisations. For each such candidate q, perform trial division of $2^a - q$ for primes up to a^2 until a factor $n_3 \mid 2^a - q$ is found with $n_3 > a^2$. By the assumptions in Definition 4 we will need fewer than $O(a^2)$ candidates before this is the case.

We now assume we have $q = n_1 n_2$ with $n_1, n_2 < 2^a/a^2$ and such that $2^a - q = n_3 n_4$ with $n_3, n_4 < 2^a/a^2$. Define $d_{1,2} := \gcd(n_1, n_2)$, $d_{3,4} := \gcd(n_3, n_4)$, $n_1' := n_1/d_{1,2}$,

$n_2' := n_2/d_{1,2}$, $n_3' := n_3/d_{3,4}$ and $n_4' := n_4/d_{3,4}$. By Theorem 6, we compute n_i'-isogenies $\varphi_i : E \to E_i$ in polynomial time.

Since $\gcd(n_1', n_2') = 1$, we can compute an HD representation of $\psi_1 := [\varphi_1]_*\varphi_2$ using [Rob24, Proposition 6.15]. This is an isogeny of degree $n_1'n_2'$. Similarly, we also compute $\psi_2 := [\varphi_3]_*\varphi_4$ of degree $n_3'n_4'$. Finally, we return an HD representation of $\varphi = [d_{1,2} \cdot d_{3,4}] \circ [\psi_1]_*\psi_2$ using [Rob24, Proposition 6.15]; in particular $\deg(\varphi) = q(2^a - q)$ as required.

It remains to argue that q is not one of the values that was already queried to the SPEDIO oracle. The only time we query SPEDIO is to generate the n_i'-isogenies via Theorem 6. As explained in the proof of Theorem 6, we query SPEDIO on the outputs of Algorithm 1, which returns "true" primes, whereas q is not a prime. □

Remark 1. While Theorem 7 provides an attack against a weakened version of Problem 1, to the best of our knowledge, it does not pose a threat to PRISM. There are two main reasons.

1. If the real scheme uses a good compositeness test like Miller-Rabin with random bases, then it will be hard for an adversary to construct a composite q that is accepted.

2. In the real scheme the forger does not get to choose the value q in a forgery. In fact, the authors of PRISM argue that the signature scheme is existentially unforgeable in the ROM under the weaker assumption Problem 2. We explore this case in Section 5.

5 An attack in the ROM for a weak version of PRISM

We present an attack on a weak version of PRISM that allows pseudoprimes to be accepted by verifiers. The main idea is that a forger can try to generate a q such that q and $2^a - q$ are smooth enough that it is possible to compute a q-isogeny and a $(2^a - q)$-isogeny without knowing the endomorphism ring. The main issue that we discuss in this section is the complexity of the attack, namely the number of hashes that need to be tried before one finds a suitable q. This depends on the compositeness test used by a verifier. In particular, we analyse the complexity if we use the most basic compositeness test, the Fermat-2 test, and then we analyse the number t of iterations needed in Miller-Rabin to drive up the complexity of this attack.

5.1 The attack

The attack works as follows, where a is the main parameter for the variant of PRISM being attacked (for the original version $a \approx 2\lambda - \log(\lambda)$).

1. Increment the counter/salt in the hash function of PRISM, until a pseudoprime q is found for the specific compositeness test being considered.

2. Test that q and $2^a - q$ are both B-smooth, otherwise return to Step 1.

3. Factorise q and $2^a - q$ as $q = \prod_i \ell_i^{e_i}$ and $2^a - q = \prod_i \ell_i^{f_i}$, where $\max(\ell_i) \leq B$.

4. Compute isogenies $\varphi_1 : E \to E_1'$ of degree q and $\varphi_2 : E_1' \to E_2'$ of degree $2^a - q$.

5. Set $E_{\mathsf{sign}} = E_2'$ and compute the torsion points $P_{\mathsf{sign}}, Q_{\mathsf{sign}}$ by mapping $P_{\mathsf{pk}}, Q_{\mathsf{pk}}$ through $\varphi_2 \circ \varphi_1$.

This attack relies on an attacker being able to rerandomize the computation of the target isogeny degree q. This can be done in both the original and the salted version of PRISM. In the original version, the attacker can modify the hash function $\mathsf{H}_{\mathsf{PRISM}}$ such that the counter is increased until we obtain a pseudoprime of the required characteristics. In this case, we anticipate a practical obstacle, namely the case where the first counter resulting in a prime is smaller than the first counter resulting in a pseudoprime. Then, since this is a deterministic procedure, nothing can be done to sign the target message. However, in practice, the verifier does not do the same computations as the signer for the hashing step. Instead, for efficiency reasons, the counter is given by the signer to the verifier and the verifier does not check that this is indeed the smallest counter that yields a prime. Hence, the rerandomization trick can be used for the attack. In the case of the salted version, the rerandomization is straightforward.

We will now analyse the complexity of this attack. In Step 1 we need to resample until we find a pseudoprime q for whichever compositeness test being considered and where both q and $2^a - q$ are B-smooth. We denote by P_{PP} the probability that one finds a pseudoprime when uniformly hashing into odd numbers. The cost of the hash function is polynomial and thus negligible. The expected cost of this step is $1/P_{\mathsf{PP}}$ since we can consider this a binomial distribution with probability P_{PP}. Next, both q and $2^a - q$ have to be B-smooth. To estimate the likelihood of this event we use results from Dickman (Theorem 3), which state that the fraction of B-smooth numbers is $\rho(\log(2^a)/\log(B))$. So the probability that both q and $2^a - q$ are B-smooth is $1/\rho(\log(2^a)/\log(B))^2$. Note that we are assuming that the smoothness of q and $2^a - q$ are independent events.

Step 2 is where we check if the obtained pseudoprime is actually B-smooth. We simply factorise the given numbers, whose cost we denote as C_{F}. Specifically, we define $C_{\mathsf{F}}(a)$ to be the smaller output between Equation (1) and Equation (2) for the corresponding input $X = 2^a$. Doing this means we take quite an optimistic viewpoint on the factoring costs. Note that checking B-smoothness in Step 2 in this way will give us Step 3 for free though.

Finally in Step 4, we have to compute two B-smooth isogenies, whose complexity is $O(B^2)$. Note that Step 5 is negligible in comparison, so we ignore those costs.

The complexity of the overall algorithm is then

$$C(a) = \min_B \frac{1}{P_{\mathsf{PP}}(a) P_{\mathsf{S}}(a, B)^2}((C_{\mathsf{F}}(a) + C_{\mathsf{F}}(a-1))) + B^2,$$

where, recall, P_{PP} denotes the probability of obtaining a pseudoprime (under the specified compositeness test) and depends only on the parameter a, P_{S} denotes the probability that an integer of size at most 2^a is B-smooth, and C_{F} denotes the cost of factoring.

5.2 Running the attack for certain compositeness tests

Now we will analyse this attack for two compositeness tests. We will start with looking at the most simple and cheap option, which is the Fermat-2 test. We will also compute the number of Miller-Rabin iterations needed to make this attack invalid.

5.2.1 Fermat-2 test

We start this analysis by looking at the probability to hash into a Fermat-2 pseudoprime.

Suppose we sample an integer n uniformly in $[1, 2^a]$. Let $P(X) = \{n | 1 < n < X, \text{ such that } 2^{n-1} \equiv 1 \mod n\}$ and $L(X) = \exp(\log(X)\log(\log(\log(X)))/\log(\log(X)))$ as defined in Theorem 1. Then the probability that n is not prime but satisfies $2^{n-1} \equiv 1 \pmod n$ is conjectured at $P(X)/X = L(X)^{-1+o(1)}$, under the assumption that $\mathsf{H}_{\mathsf{PRISM}}$ uniformly maps onto odd integers. The $o(1)$ term will approach to zero as a grows, so we will set it to zero in our calculations, this gives us a conjectured lower bound. We also know the upper bound $P(X)/X \leq L(X)^{-1/2}$. In Section 5.2.1 we show some values

Table 1: Probability of hashing into a base-2 Fermat pseudoprime.

Required security level	Version	a	$\log_2(L(2^a)^{-1/2})$	$\log_2(L(2^a)^{-1})$
128	original	248	-39.4	-78.9
	salted	192	-31.2	-62.3
192	original	376	-57.9	-115.9
	salted	256	-40.6	-81.3
256	original	500	-75.4	-150.9
	salted	320	-50.0	-100.0

Table 2: Total complexity of the attack using the Fermat-2 test plus for all variants of PRISM (both original and salted version). Note that for each attack the lower and the conjectured upper bound is provided.

Required security level	Version	a	Bound	$\log_2(B)$	Complexity in $\log_2$
128	original	248	lower	50	106
			conjectured upper	64	137
	salted	192	lower	41	89
			conjectured upper	52	**113**
192	original	376	lower	86	141
			conjectured upper	91	**188**
	salted	256	lower	51	108
			conjectured upper	67	**140**
256	original	500	lower	83	171
			conjectured upper	112	**234**
	salted	320	lower	60	126
			conjectured upper	80	**166**

corresponding to PRISM parameters for the three NIST security levels for the conjectured lower bound and the upper bound. [2] Note that the actual upper bound is based on a very weak lower bound, and thus using the conjectured upper bound is a safer bet.

Now that we have the costs of hashing into a Fermat-2 pseudoprime we can look at the complexity of the full attack, which can be found in Table 2. Note that a *lower* bound on the probability makes an *upper* bound on the complexity and vice versa. The corresponding column denotes this with reference to the complexity.

If we assume that the actual cost lies closer to the conjectured value, only the first security level of the original version of PRISM meets the security threshold if we use Fermat-2 test. For the other security levels of the original version and for all salted versions of PRISM, the security requirements are not met even with the conjectured upper bound.

[2] Note that we use the set $P(2^a)$ which gives us the pseudoprimes in the interval $[1, 2^a]$, while in practice we consider only those in the interval $[2^{a-1}, 2^a]$. We do not adjust for the difference since it is negligible.

Table 3: Total costs of the attack using the Miller-Rabin test for given t trials for all variants of PRISM (both original and salted version). Note that this is a lower bound for the complexity.

Required security level	Version	a	t	$\log_2(B)$	Complexity in $\log_2$
128	original	248	9	63	130
	salted	192	16	60	129
192	original	376	15	92	192
	salted	256	30	92	194
256	original	500	23	124	256
	salted	320	44	118	258

5.2.2 Miller-Rabin test

Since we have seen that the 2-Fermat test is not valid for all versions and security levels of PRISM, we also analyse what happens if we use a certain number of iterations of Miller-Rabin for our compositeness test. In practice this test is usually implemented to go through $\lambda/2$ iterations, resulting in a probability of $2^{-\lambda}$ of obtaining a strong pseudoprime. This can become quite costly, and thus we aim to calculate exactly how many iterations are sufficient to protect PRISM against the proposed attack. We use the upper bounds of the probability for a composite number to survive t iterations, introduced in Theorem 2, and plug them into the P_{PP} parameter. Note that this is an upper bound for the probability which translates to a lower bound for the complexity of the attack. Thus, as soon as we find a t and a corresponding minimal B such that the complexity of the full attack is greater than the security level, we know that the attack must be at least more expensive than this.

In Table 3 we provide for each of the security levels and for both version of PRISM the minimum t and corresponding B that leads to an attack complexity that exceeds the security level. We see that, for instance, for $\lambda = 128$ we only need 9, respectively 16, iterations for the original and salted versions of PRISM. Comparing this to the 64 iterations that are usually done, this is a substantial speed up. The maximum number of iterations needed across all parameters is only 44 for the salted version of $\lambda = 256$, which is to be compared to having to do 128 iterations.

Since a significant part of the cost is the factorisation cost, we want to see what happens if we have access to a quantum computer. To this end, we consider factoring to be polynomial time, specifically we use $C_{\mathsf{F}}(a) = a^2$. The results can be found in Table 4. Even with a quantum computer for factorisation, we still only need a maximum of 55 iterations for the largest security level.

We refer to https://github.com/JolijnC/attack_weakened_PRISM/tree/main to verify the data of all tables of this section.

6 Concrete implications

We first highlight that the polynomial-time attack in Section 4 does not constitute a concrete threat to PRISM, even if a weak compositeness test is used. The attack only affects the hardness assumption (Problem 1) underlying the proof in the standard model. In practice, an adversary attacking PRISM will not have access to SPEDIO. To the best of our knowledge, it is not possible to instantiate such an oracle by just looking at signatures corresponding to chosen messages.

Table 4: Total costs of the quantum attack using the Miller-Rabin test for given t trials for all variants of PRISM (both original and salted version). Note that this is a lower bound for the complexity.

Required security level	Version	a	t	$\log_2(B)$	Complexity in $\log_2$
128	original	248	16	60	129
	salted	192	22	61	128
192	original	376	27	92	194
	salted	256	39	91	194
256	original	500	38	125	257
	salted	320	55	124	257

A more realistic assumption is the one used for the proof in the (Q)ROM, namely Problem 2. In PRISM, the degrees of the signing isogenies are determined by the output of a hash function and, as a result, should be regarded as unpredictable. Such degrees are odd integers that pass some primality test. In Section 5, we explored under which primality test PRISM can still be considered secure. First, we analysed the case of a simple base-2 Fermat test. As shown in Table 2, if PRISM is deployed using such a simple primality test, it fails to meet the required security requirements. This is particularly evident in the case of salted-PRISM, where the exponent a is chosen to be smaller than the one in PRISM.

After arguing that the base-2 Fermat test is not sufficient to instantiate a secure version of PRISM for almost all parameter sets, we analysed which primality tests ensure that PRISM remains secure. We suggest using a Miller-Rabin test for t random bases, where t is a specific number that can be computed for each parameter set. In particular, we highlight that t is significantly smaller than $\lambda/2$, which is typically considered the number of iterations required to ensure that a composite number passes the test is negligible. This difference in number of iterations comes from the fact that the reference attack requires additional smoothness constraints to be satisfied, while the integers we need to test are "random", and therefore not likely to be very smooth. Hence, we looked at the distribution of integers that pass the Miller-Rabin test t times.

In the proof-of-concept implementation in [BBC+26], the authors use `mpz_probab_prime_p` with 20 iterations for all parameter sets (see the file src/verification/ref/lvlx/-common.c of the proof-of-concept implementation). The function `mpz_probab_prime_p` first performs some trial divisions and then repeats Miller-Rabin probabilistic primality tests for a given number of iterations. We recommend the authors update the number of iterations according to the results of Table 4.

Overall, our main finding is that PRISM is surprisingly robust against weak primality tests. Since a relatively small number t of iterations suffice, we can optimise the signing and verification procedures in PRISM. This is an important step towards an open problem stated in [BBC+26, §7.4]: optimising signing operations in PRISM. The impact of the primality test is even more evident in the case of high security levels.

References

[AAA+24] 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, 2024. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures.

[BBC+25] 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, *PKC 2025, Part III*, volume 15676 of *LNCS*, pages 300–332. Springer, Cham, May 2025. doi:10.1007/978-3-031-91826-1_10.

[BBC+26] 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. URL: https://eprint.iacr.org/2026/443.

[BDD+24] Andrea Basso, Pierrick Dartois, Luca De Feo, Antonin Leroux, Luciano Maino, Giacomo Pope, Damien Robert, and Benjamin Wesolowski. SQIsign2D-West - the fast, the small, and the safer. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part III*, volume 15486 of *LNCS*, pages 339–370. Springer, Singapore, December 2024. doi:10.1007/978-981-96-0891-1_11.

[BDLS20] Daniel J. Bernstein, Luca De Feo, Antonin Leroux, and Benjamin Smith. Faster computation of isogenies of large prime degree. *Open Book Series*, 4(1):39–55, 2020.

[Ber04] Daniel J. Bernstein. How to find smooth parts of integers. Preprint, 2004. URL: https://cr.yp.to/papers.html#smoothparts.

[BMP23] Andrea Basso, Luciano Maino, and Giacomo Pope. FESTA: Fast encryption from supersingular torsion attacks. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part VII*, volume 14444 of *LNCS*, pages 98–126. Springer, Singapore, December 2023. doi:10.1007/978-981-99-8739-9_4.

[CD23] Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 423–447. Springer, Cham, April 2023. doi:10.1007/978-3-031-30589-4_15.

[De 17] Luca De Feo. Mathematics of isogeny based cryptography. *arXiv preprint arXiv:1711.04062*, 2017.

[DF24] Max Duparc and Tako Boris Fouotsa. SQIPrime: A dimension 2 variant of SQISignHD with non-smooth challenge isogenies. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part III*, volume 15486 of *LNCS*, pages 396–429. Springer, Singapore, December 2024. doi:10.1007/978-981-96-0891-1_13.

[Dic30] Kern W. Dickman. On the frequency of numbers containing prime factors of a certain relative magnitude. *Arkiv för matematik, astronomi och fysik*, 22A(10):1–14, 1930.

[DKL+20] Luca De Feo, David Kohel, Antonin Leroux, Christophe Petit, and Benjamin Wesolowski. SQISign: Compact post-quantum signatures from quaternions and isogenies. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part I*, volume 12491 of *LNCS*, pages 64–93. Springer, Cham, December 2020. doi:10.1007/978-3-030-64837-4_3.

[DLP93] Ivan Damgård, Peter Landrock, and Carl Pomerance. Average case error estimates for the strong probable prime test. *Mathematics of Computation*, 61(203):177–194, 1993. URL: https://math.dartmouth.edu/~carlp/PDF/paper88.pdf.

[DMPR24] Pierrick Dartois, Luciano Maino, Giacomo Pope, and Damien Robert. An algorithmic approach to (2, 2)-isogenies in the theta model and applications to isogeny-based cryptography. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part III*, volume 15486 of *LNCS*, pages 304–338. Springer, Singapore, December 2024. doi:10.1007/978-981-96-0891-1_10.

[EPSV24] Jonathan Komada Eriksen, Lorenz Panny, Jana Sotáková, and Mattia Veroni. Deuring for the people: supersingular elliptic curves with prescribed endomorphism ring in general characteristic. In *LuCaNT: LMFDB, computation, and number theory*, volume 796 of *Contemporary Mathematics*, pages 339–373. American Mathematical Society, 2024. doi:10.1090/conm/796/16008.

[Kan97] Ernst Kani. The existence of curves of genus two with elliptic differentials. *Journal of Number Theory*, 64(1):130–161, 1997. doi:10.1006/jnth.1997.2105.

[KLPT14] David Kohel, Kristin Lauter, Christophe Petit, and Jean-Pierre Tignol. On the quaternion ℓ-isogeny path problem. *LMS Journal of Computation and Mathematics*, 17(A (Special issue)):418–432, 2014. doi:10.1112/S1461157014000151.

[LLMP90] Arjen K. Lenstra, Hendrik W. Lenstra Jr., Mark S. Manasse, and John M. Pollard. The number field sieve. In Harriet Ortiz, editor, *STOC*, pages 564–572. ACM, 1990. doi:10.1145/100216.100295.

[Mil76] Gary L. Miller. Riemann's hypothesis and tests for primality. *Journal of Computer and System Sciences*, 13(3):300–317, 1976. doi:10.1016/S0022-0000(76)80026-8.

[MMP+23] 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, Part V*, volume 14008 of *LNCS*, pages 448–471. Springer, Cham, April 2023. doi:10.1007/978-3-031-30589-4_16.

[Mon80] Louis Monier. Evaluation and comparison of two efficient probabilistic primality testing algorithms. *Theoretical Computer Science*, 12(1):97–108, 1980. URL: https://www.sciencedirect.com/science/article/pii/0304397580900079, doi:10.1016/0304-3975(80)90007-9.

[NO24] Kohei Nakagawa and Hiroshi Onuki. QFESTA: Efficient algorithms and parameters for FESTA using quaternion algebras. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part V*, volume 14924 of *LNCS*, pages 75–106. Springer, Cham, August 2024. doi:10.1007/978-3-031-68388-6_4.

[NOC+24] Kohei Nakagawa, Hiroshi Onuki, Wouter Castryck, Mingjie Chen, Riccardo Invernizzi, Gioella Lorenzon, and Frederik Vercauteren. SQIsign2D-East: A new signature scheme using 2-dimensional isogenies. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part III*, volume 15486 of *LNCS*, pages 272–303. Springer, Singapore, December 2024. doi:10.1007/978-981-96-0891-1_9.

[Pom81] Carl Pomerance. On the distribution of pseudoprimes. *Mathematics of Computation*, 37(156):587–593, 1981. doi:10.1090/S0025-5718-1981-0628717-0.

[Pom85] Carl Pomerance. The quadratic sieve factoring algorithm. In *Proc. of the EUROCRYPT 84 Workshop on Advances in Cryptology: Theory and Application of Cryptographic Techniques*, page 169–182, Berlin, Heidelberg, 1985. Springer-Verlag. doi:10.1007/3-540-39757-4_17.

[PSW80] Carl Pomerance, J. L. Selfridge, and Jr. Wagstaff, Samuel S. The pseudoprimes to $25 \cdot 10^9$. *Mathematics of Computation*, 35(151):1003–1026, 1980. doi:10.2307/2006210.

[Rab80] Michael O. Rabin. Probabilistic algorithm for testing primality. *Journal of Number Theory*, 12(1):128–138, 1980. doi:10.1016/0022-314X(80)90084-0.

[Rob23] Damien Robert. Breaking SIDH in polynomial time. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 472–503. Springer, Cham, April 2023. doi:10.1007/978-3-031-30589-4_17.

[Rob24] Damien Robert. On the efficient representation of isogenies: A survey for nutmic 2024. In *International Conference on Number-Theoretic Methods in Cryptology*, pages 3–84. Springer, 2024. doi:10.1007/978-3-031-82380-0_1.

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pages 124–134. IEEE Computer Society Press, November 1994. doi:10.1109/SFCS.1994.365700.

[Sho09] Victor Shoup. *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, USA, 2 edition, 2009.

[Sil09] Joseph H Silverman. *The arithmetic of elliptic curves*, volume 106. Springer, 2009.

[Vél71] Jacques Vélu. Isogénies entre courbes elliptiques. *Comptes Rendus de l'Académie des Sciences de Paris*, 273:238–241, July 1971. URL: https://gallica.bnf.fr/ark:/12148/bpt6k56191248/f52.item.

[Wes22] Benjamin Wesolowski. The supersingular isogeny path and endomorphism ring problems are equivalent. In *62nd FOCS*, pages 1100–1111. IEEE Computer Society Press, February 2022. doi:10.1109/FOCS52979.2021.00109.