

POKE: A Framework for Efficient PKEs, Split KEMs, and OPRFs from Higher-dimensional Isogenies

Andrea Basso^[0000–0002–3270–1069]

IBM Research Europe, Zürich, Switzerland
University of Bristol, Bristol, United Kingdom
`andrea.basso@ibm.com`

Abstract. We introduce a new framework, POKE, to build cryptographic protocols from irrational isogenies using higher-dimensional representations. The framework enables two parties to manipulate higher-dimensional representations of isogenies to efficiently compute their push-forwards, and ultimately to obtain a shared secret.

We provide three constructions based on POKE: the first is a PKE protocol, which is one of the most compact post-quantum PKEs and possibly the most efficient isogeny-based PKE to date. We then introduce a validation technique to ensure the correctness of uniSIDH public keys: by combining the validation method with a POKE-based construction, we obtain a split KEM, a primitive that generalizes NIKEs and can be used to instantiate a post-quantum version of the Signal’s X3DH protocol. The third construction builds upon the split KEM and its validation method to obtain a round-optimal verifiable OPRF. It is the first such construction that does not require more than λ isogeny computations, and it is significantly more compact and more efficient than all other isogeny-based OPRFs.

We recently discovered that [Assumption 11](#) is not valid: there exists an efficient attack against the validation method proposed in [Section 5.3](#). Thus, the split KEM and OPRF protocols described in [Sections 5](#) and [6](#) are not secure. Note that the attack does not affect the PKE proposed in [Section 4](#). A revised paper is in the works.

1 Introduction

Since its inception nearly two decades ago, isogeny-based cryptography focused on isogenies between elliptic curves. While some protocols based on isogenies between abelian varieties (a generalization of elliptic curves to higher dimensions) had been proposed [[31,39,41](#)], these constructions were often less studied and less efficient than their one-dimensional counterparts. This trend drastically changed in 2022: SIDH [[36,26](#)], possibly the most efficient and well-known isogeny-based PKE, was broken in a series of three breakthrough papers by Castryck and Decru [[14](#)], Maino, Martindale, Panny, Pope and Wesolowski [[42](#)], and Robert [[54](#)].

These attacks, which rely on computing isogenies between abelian varieties, led to a complete key-recovery attack on SIDH. But, more fundamentally, they drastically changed the landscape of hardness assumptions in isogeny-based cryptography: in particular, the attacks showed that, given two curves, the degree of the connecting isogeny, and its action on a sufficiently large torsion basis, it is always possible to recover the connecting isogeny. If the degree is sufficiently smooth, recovering the isogeny means obtaining a kernel generator; this is not possible if the degree is not smooth, but the tools that power the SIDH attacks still enable evaluating the connecting isogeny at any point. Since *knowing* an isogeny really means being able to evaluate it, the techniques used to attack SIDH showed that two sets of curves and torsion points (together with the degree) can provide an alternative representation of an isogeny. This is the first representation that can efficiently describe non-smooth isogenies between any two curves.

Thus, the combination of new techniques to represent isogenies, especially those with non-smooth degree, and improved algorithms [49,22] to efficiently evaluate such isogenies led to a renewed interest in developing cryptographic protocols based on isogenies between abelian surfaces. Existing protocols, such as SQIsign [27] and SCALLOP [25], obtained considerable improvements by relying on higher-dimensional representations [21,19]. New constructions, which could only be built from higher-dimensional representations, also soon appeared, such as FESTA [8] and QFESTA [48]. At the same time, better protocols from isogenies between elliptic curves were also developed: initially, Fouotsa, Moriya, and Petit [32] proposed countermeasures against the SIDH attacks that relied on masked degree and masked torsion points. They obtained MD-SIDH and M-SIDH, which are larger and slower than SIDH, but avoid the existing attacks. More recently, Basso and Fouotsa proposed binSIDH and terSIDH [6], which also avoid the attacks on SIDH while being significantly more efficient than M-SIDH and MD-SIDH. The authors also proposed a mixed approach, where one party computes binSIDH or terSIDH isogenies, while the other rely on SIDH-like isogenies.

However, the two approaches, developing better protocols in dimension one on one side, and using higher dimensional isogenies on the other, have not mixed. While the higher-dimensional protocols mentioned above still compute one-dimensional isogenies, their constructions are fundamentally different from those of SIDH-like protocols. To date, no protocol that builds a commutative diagram à la SIDH works with higher-dimensional isogenies. A possible reason behind this is that SIDH-like protocols rely on rational isogenies (i.e. with their kernel defined over \mathbb{F}_{p^2}) that can be represented by a single curve (its domain, together with a kernel generator); higher-dimensional isogenies, on the other hand, often represent irrational isogenies (i.e. their kernel is not defined over \mathbb{F}_{p^2}) and require a two-curve representation, where both the domain and the codomain are needed.

Contributions. In this work, we introduce the first framework that combines the two approaches. It allows two parties, one computing irrational non-smooth isogenies and the other computing rational smooth isogenies (depending on the

context, this may be isogenies as those computed in SIDH, binSIDH, terSIDH, etc.), to engage in a two-round protocol to establish a commutative diagram and obtain a shared secret. Since one isogeny in the exchange needs both its domain and codomain to be recomputed, all four curves in the commutative diagram need to be public: to obtain a shared secret, both parties reveal the (scaled) action of their secret isogeny on a fixed point and eventually obtain a shared point on a public curve. This approach can be, in some sense, interpreted as a translation of the SiGamal protocol [47] to the setting of SIDH-like isogenies and higher-dimensional representations. Since the two parties end up agreeing on a shared point, we call the framework POKE, for Point-Based Key Exchange.

With the POKE framework, we introduce three new protocols: a public-key encryption scheme, a split key encapsulation mechanism, and a round-optimal verifiable OPRF. The PKE, described in Section 4, uses a secret-degree irrational isogeny as the long-term secret key. To encrypt, the sender computes two parallel SIDH isogenies and reveals enough information for the receiver to complete the commutative diagram. The resulting protocol is extremely compact and efficient: it works with a prime of size 3λ bits, and our unoptimized proof-of-concept implementation in SageMath runs in less than 300 milliseconds.

A split KEM, first introduced in [12], is an intermediate construction between a non-interactive key-exchange (NIKE) and a PKE. Both parties hold a long-term secret key, and one party, the sender, is allowed to send one message. Split KEMs were introduced to achieve a post-quantum version of the Signal protocol [43] since they can replace NIKEs in most instances. To obtain our construction (Section 5), we rely on uniSIDH isogenies (a variant of binSIDH [6, Sec. 4.1]), and we analyze the resistance of FESTA and uniSIDH isogenies against adaptive attacks. For the former, we show it is always easy and efficient to avoid adaptive attacks: in particular, we also solve the open problem posed by [46]. The authors of [46] asked whether it is possible to find an adaptive attack even when the scaling matrix is of the correct form: we show that this is not possible. For uniSIDH isogenies, which are easily susceptible to active attacks, we introduce a novel interactive validation method. This requires multiple repetitions (≈ 12) of the protocol, but it is significantly more efficient than previous methods based on zero-knowledge proofs.

Lastly, we rely on the split-KEM construction and its efficient validation method to obtain an efficient round-optimal verifiable OPRF (Section 6). The resulting protocol is the first isogeny-based verifiable OPRF that does not require more than λ isogeny computations, and it is significantly more compact and more efficient than all other isogeny-based OPRFs. Compared to non-isogeny ones, our protocol is the most compact by at least two orders of magnitude, and our proof-of-concept implementation suggests it might be competitive from an efficiency standpoint, although future work is needed to confirm this.

2 Preliminaries

Notation. We write \mathbb{Z}_n to denote the ring $\mathbb{Z}/n\mathbb{Z}$, while λ denotes the security parameter. We also write $[k]$ for the set $\{1, 2, \dots, k\}$. If $\phi_A : E_0 \rightarrow E_A$ and $\phi_B : E_0 \rightarrow E_B$ are isogenies with the same domain, we write $(\phi_A)_*\phi_B$ for the pushforward of ϕ_B under ϕ_A , i.e. the isogeny $\phi'_B : E_A \rightarrow E_{AB}$ such that $\ker \phi'_B = \phi_A(\ker \phi_B)$.

Elliptic curves and isogenies. For a complete treatment of elliptic curves and isogenies, especially as used in cryptography, we refer to [23]. We call two isogenies ϕ and ϕ' *parallel* with respect to an isogeny ψ (which may be omitted, if explicit from the context) if $\ker \phi' = \psi(\ker \phi)$. We also refer to *rational* isogenies as those whose kernel is defined over \mathbb{F}_{p^2} , and *irrational* isogenies as those whose kernel is defined over a large extension of \mathbb{F}_{p^2} . In this work, we consider three representations of isogenies:

- SIDH isogenies [36]: they are prime-power degree isogenies (in our case, the degree will always be a power of three or five) whose kernel is defined over \mathbb{F}_{p^2} . If their degree is ℓ^e , since a kernel generator can be expressed as a linear combination of any two linearly independent points of order ℓ^e , to compute the pushforward of an SIDH isogeny under a secret isogeny, it is necessary to reveal the action of the secret isogeny on two linearly independent points of order ℓ^e , possibly both scaled by the same random scalar in $\mathbb{Z}_{\ell^e}^*$.
- uniSIDH isogenies [6, Sec 4.1]: they are rational isogenies whose degree divides B , the product of t small primes. Fixed a curve E and a point R of order B , the isogeny is represented by a divisor B' of B : the isogeny is defined as $E/\langle [B/B']R \rangle$. To compute their pushforward under a secret isogeny, it is necessary to reveal the action of the secret isogeny only on R , possibly scaled by a random scalar in \mathbb{Z}_B^* .
- FESTA isogenies [8]: they are isogenies whose degree can be written as $q(2^a - q)$, for some value q ; their kernel is not rational and it is only defined over a large extension field. They can be represented by their domain and codomain, together with their degree and action on the 2^a torsion. We propose a method to compute their pushforward under a secret isogeny in Section 3.

Split KEM. A split KEM [12] is a key encapsulation mechanism with two long-term public keys, one for the sender and one for the receiver. A split KEM protocol comprises of four algorithms: $(\text{sk}_B, \text{pk}_B) \leftarrow \text{sKeyGen}_{enc}()$, $(\text{sk}_A, \text{pk}_A) \leftarrow \text{sKeyGen}_{dec}()$, $(\text{ct}, \text{ss}_B) \leftarrow \text{sEncaps}(\text{pk}_A, \text{sk}_B)$, and $\text{ss}_A \leftarrow \text{sDecaps}(\text{ct}, \text{pk}_B, \text{sk}_A)$.

Security is defined in terms of a lr -IND-CCA, where $\text{lr} \in \{\text{nn}, \text{sn}, \text{mn}, \text{sm}, \text{mm}\}$. The game is a translation of the classic IND-CCA game to the split KEM setting, where the attacker has access to both an encapsulation and decapsulation oracle. The five flavors of the game, defined by the choice of lr , determine whether the attacker has no (n), a single (s), or multiple (m) access(es) to the encapsulation (l) and decapsulation (r) oracles. We refer to [12] for a thorough description of split KEMs and their security properties.

OPRFs. Oblivious pseudorandom functions are a two-party protocol between a client and a server: the protocol allows the client to obliviously evaluate a PRF with a key held by the server, while the server does not learn anything about the client’s input or output. More precisely, an OPRF protocol OPRF is defined by six algorithms: (OPRF.Setup, OPRF.KeyGen, OPRF.ClientReq, OPRF.BlindEval, OPRF.Finalize, OPRF.Eval). OPRF.Setup generates the parameters needed for the protocol, while OPRF.KeyGen is run by the server to generate a pair of public and secret key. The client generates a blinded request of a message m by computing $\text{ClientReq}(m)$; after receiving the request, the server uses its secret key to compute BlindEval , and the client runs Finalize to obtain the PRF value. OPRF.Eval is a deterministic function on the input and the server’s secret key and describes the PRF itself. For correctness, we require the composition of Finalize , BlindEval , and ClientReq to behave exactly as OPRF.Eval. We introduce the security notions of OPRFs that we use in [Appendix A](#).

3 The POKE framework

We introduce our framework that allows two parties to engage in a two-round protocol to establish a commutative diagram and obtain a shared secret, using different isogeny representations. Nearly all isogeny-based protocols in the literature, such as M-SIDH, MD-SIDH [32], binSIDH, and terSIDH [6], rely on a single type of isogenies: both parties need to compute isogenies with the same representation. A partial exception is the hybrid variant of bin-SIDH and terSIDH [6], which introduces a mixed approach: one party computes binary or ternary isogenies, while the other party computes SIDH-like isogenies. However, such constructions are still constrained to rational isogenies that can be represented by a single curve (both binary/ternary and SIDH-like isogenies can be computed from their domain and a kernel generator on it). The framework we propose enables mixing different representations of isogenies and working with irrational isogenies. In particular, it provides a method to compute the pushforward of isogenies represented by two curves, such as FESTA isogenies. This, however, comes at the cost of additional interaction between the two parties, when compared to SIDH-based constructions.

The POKE construction. The protocol takes place between two parties, Alice and Bob. The description is intentionally general, as it can be instantiated with any isogeny representation. Alice can compute any isogeny, including those that are representable by two curves (say, FESTA isogenies), while Bob is restricted to isogenies representable by a single curve (say, SIDH-like isogenies).

Bob, on the other hand, samples a random *rational* isogeny $\phi_B : E_0 \rightarrow E_B$ whose degree is coprime with that of ϕ_A . Since the isogeny is rational, Bob can compute its pushforward under ϕ_A from the images revealed by Alice: Bob can thus also compute the isogeny $\phi'_B : E_A \rightarrow E_{AB}$. Note that the computation of ϕ_B is independent of Alice’s computations: it thus can be done in parallel and, in some cases, the same ϕ_B can be used with different ϕ_A ’s.

After Bob reveals E_B and E_{AB} , together with the actions of ϕ_B and ϕ'_B on some points on E_0 and E_A , Alice can compute the pushforward of ϕ_A under

ϕ_B and thus recover the isogeny $\phi'_A : E_B \rightarrow E_{AB}$. This allows both parties to construct a commutative diagram; however, it is not sufficient to create a key exchange: similar constructions relied on E_{AB} as their shared secret, but this is no longer possible since it is publicly revealed by Bob to enable Alice's computation of ϕ'_A .

To sidestep the issue, we introduce an additional point X_0 on E_0 of order x , with x coprime to both $\deg \phi_A$ and $\deg \phi_B$. As part of her public key, Alice also reveals the image $X_A = [\alpha]\phi_A(X_0)$, scaled under a random value α . Similarly, Bob reveals the image $X_B = [\beta]\phi_B(X_0)$, scaled under a random value β . Both parties can thus obtain the point $X_{AB} = [\alpha]\phi'_A(X_B) = [\beta]\phi'_B(X_A)$ on E_{AB} , which is then the shared secret of the exchange. The resulting protocol is described in Fig. 1.

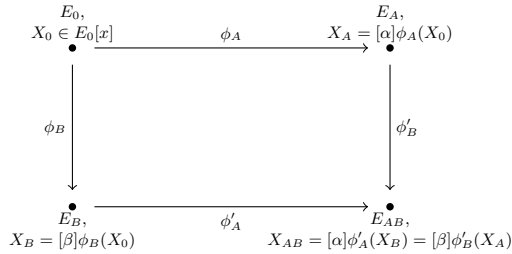


Fig. 1: The POKE framework.

Introducing the point X_0 does not increase the size of the underlying prime: since the order of X_0 does not need to be smooth, the point X_0 can be defined over \mathbb{F}_{p^4} (i.e. with $x \mid p - 1$). Using an x -only representation, it is still efficient to evaluate isogenies on X_0 , while the underlying prime can be smaller. From a security standpoint, POKE requires the parties to reveal the scaled action of their secret isogeny on a public point. We argue this does not affect the security of the resulting protocols: all isogeny representations already reveal the scaled action of a secret isogeny on some points, and thus introducing an additional point can simply be interpreted as slightly increasing the parameters of the existing assumptions¹. Furthermore, the point order x can be chosen to be a large prime (with $\log x \approx \lambda/2$): even revealing the non-scaled action on X_0 would not help an attacker. First, if x is a prime, the reduction by De Feo, Fouotsa, and Panny [30] from one point to two does not apply; second, even if it did, all known algorithms that exploit torsion information would require computing an x -isogeny in dimension four, whose computational cost is $\mathcal{O}(x^4) \approx \mathcal{O}(2^{2\lambda})$. Note, also, that choosing $\log x \approx \lambda/2$ prevents brute-force attacks, since there are about $x^2 \approx 2^\lambda$ possible X_{AB} points.

¹ If the protocol already revealed the action of a secret isogeny ϕ on two points of order N , revealing the image of X_0 is equivalent to revealing the action of ϕ on a point of order $N + x$ and a point of order N .

Remark 1. The POKE protocol presents similarities with the SiGamal PKE [47]. In both cases, the protocols follow a similar flow of information: key generation consists of one isogeny computation, encryption computes two parallel isogenies, and decryption completes the commutative diagram. Moreover, as in SiGamal, the two parties reveal the images of a fixed point under their secret isogenies and eventually both obtain a shared point on the curve E_{AB} . However, POKE differs from SiGamal in two significant aspects: first, the point X_0 is defined over \mathbb{F}_{p^4} , resulting in a smaller prime p and a more compact protocol; second, POKE is not restricted to isogenies over \mathbb{F}_p and instead allows for many different representations of isogenies. This greater flexibility is exploited in the next sections to obtain new and more efficient constructions that would not be possible from SiGamal.

4 An efficient PKE: P(O)KE

We present the first instantiation of the POKE framework, where one party computes irrational isogenies of non-smooth degree, similarly to FESTA [8] and QFESTA [48], and the other computes SIDH-like isogenies. The resulting protocol is a public-key encryption scheme that is one of the most compact post-quantum PKEs and possibly the most efficient isogeny-based PKE to date.

Let p be a prime of the form $p = 2^a 3^b f - 1$, where f is a cofactor needed for primality. Let E_0 be the supersingular elliptic curve defined over \mathbb{F}_{p^2} with j -invariant 1728. Let P_0, Q_0 denote a basis of $E_0[2^a]$, R_0, S_0 denote a basis of $E_0[3^b]$, and X_0 denote a point of order x on E_0 , with x coprime with 2 and 3. Castryck and Vercauteren [16] showed that when the endomorphism ring of the starting curve is known, it is possible to maliciously choose the points P_0, Q_0 to obtain a backdoor in the protocol. To avoid such issues, we select P_0, Q_0 using a nothing-up-my-sleeve approach: we rely on standard algorithms to compute a deterministic basis from a given curve [51]. Since such a basis is uniformly random, the probability of being backdoorable is negligible [16, §5.2].

Let us denote with Bob the party that wants to encrypt a message m , and Alice the party that wants to receive it. To generate a public key, Alice wants to generate an isogeny of degree $q(2^a - q)$, so that it can be easily representable in dimension two (the degree needs to factor into integers whose sum is 2^a). To do so, we adapt the algorithm proposed in QFESTA [48, Algorithm 2]: we first generate an endomorphism θ of degree $q(2^a - q)3^b$, using Algorithm 1 (based on [38]); then, we compute the 3^b -isogeny ψ that factors through θ , and we compute its dual to obtain the codomain of a $q(2^a - q)$ -isogeny ϕ . Evaluating points under ϕ is then equivalent to mapping them under $[1/3^b]\psi \circ \theta$, if their order is coprime with 3; this allows us to map points of order 2^a and obtain a two-dimensional representation of ϕ . This is represented in Algorithm 2.

After Alice has generated a $q(2^a - q)$ -isogeny, the protocol proceeds as described in the POKE framework. To obtain a PKE, Bob attaches $ct' = \text{KDF}(X_{AB}) \oplus m$ to the ciphertext, where KDF is a key derivation function. Alice, following the POKE framework, recomputes X_{AB} and extracts the message m as $m = \text{KDF}(X_{AB}) \oplus ct'$.

Algorithm 1 Endomorphism generation (based on [38])

Input: A degree N , where $N > p$.

Output: A endomorphism of E_0 of degree N .

- 1: Set $u = \lceil \sqrt{N/p} \rceil$
 - 2: **while true do**
 - 3: Sample random integers z, t in \mathbb{Z}_u
 - 4: Set $C = N - p(z^2 + t^2)$
 - 5: **if** C is prime **and** $C = x^2 + y^2$ **then** ▷ x, y computed with Cornacchia's alg.
 - 6: **return** $[x] + [y]\iota + [z]\pi + [t]\iota\pi$
-

The resulting protocol is described in [Protocol 2](#). In [Appendix B](#), we propose a different key generation procedure that enables starting the protocol from a curve E_0 of unknown endomorphism ring.

Protocol 2 (The P(O)KE PKE). Let p be a prime of the form $p = 2^a 3^b f - 1$, where f is a small cofactor needed for primality. Let E_0 be the supersingular elliptic curve defined over \mathbb{F}_{p^2} with j -invariant 1728. Let P_0, Q_0 be a deterministic basis of $E_0[2^a]$, and R_0, S_0 be a deterministic basis of $E_0[3^b]$. Similarly, let X_0 also be a deterministically-generated point on $E_0(\mathbb{F}_{p^4})$ of order x , with $x \mid p - 1$ and coprime with 2 and 3. Let KDF denote a key derivation function.

KeyGen. Alice samples a random prime q in $[0, 2^a]$ such that $q(2^a - q)$ is coprime with 3, and she computes a random isogeny $\phi_A : E_0 \rightarrow E_A$ of degree $q(2^a - q)$ using [Algorithm 2](#). She also samples random integers $\alpha_2, \alpha'_2, \alpha_3, \alpha_x$ in $\mathbb{Z}_{2^a}^* \times \mathbb{Z}_{2^a}^* \times \mathbb{Z}_{3^b}^* \times \mathbb{Z}_x^*$. Alice's public key consists of the curve E_A , and the 2^a -images $P_A = [\alpha_2]\phi_A(P_0), Q_A = [\alpha'_2]\phi_A(Q_0)$, the 3^b -images $R_A = [\alpha_3]\phi_A(R_0), S_A = [\alpha_3]\phi_A(S_0)$, and the point $X_A = [\alpha_x]\phi_A(X_0)$; the secret key includes the scaling values α_2, α'_2, x and the degree q .

Encrypt. To encrypt a message m , Bob samples a random integer β in \mathbb{Z}_{3^b} and computes the parallel isogenies $\phi_B : E_0 \rightarrow E_B$ with kernel $\langle R_0 + [\beta]S_0 \rangle$ and $\phi'_B : E_A \rightarrow E_{AB}$ with kernel $\langle R_A + [\beta]S_A \rangle$. He also samples random integers

Algorithm 2 Generating a $q(2^a - q)$ -isogeny

Input: A degree q , a prime p of the form $p = 2^a B - 1$, where B is smooth.

Output: A representation of an isogeny $\phi : E_0 \rightarrow E_A$ of degree $q(2^a - q)$.

- 1: Generate an endomorphism θ of E_0 of degree $q(2^a - q)B$
 - 2: Generate a basis P_0, Q_0 of $E_0[2^a]$
 - 3: Generate a basis R_0, S_0 of $E_0[B]$
 - 4: Compute $P'_0, Q'_0 = \theta(P_0), \theta(Q_0)$
 - 5: Compute $R'_0, S'_0 = \theta(R_0), \theta(S_0)$
 - 6: Find $x, y \in \mathbb{Z}_B$ such that $[x]R'_0 + [y]S'_0 = \mathcal{O}$
 - 7: Compute $\psi : E_0 \rightarrow E_A$ with kernel $\langle [x]R_0 + [y]S_0 \rangle$
 - 8: Compute $P_A, Q_A = [1/B]\psi(P'_0), [1/B]\psi(Q'_0)$
 - 9: Compute Φ with kernel $\langle ([-q]P_0, P_A), ([-q]Q_0, Q_A) \rangle$
 - 10: **return** Φ
-

$\beta_2, \beta'_2, \beta_x \in \mathbb{Z}_{2^a}^* \times \mathbb{Z}_{2^a}^* \times \mathbb{Z}_x^*$ and computes the points $X_B = [\beta_x]\phi_B(X_0)$ and $X_{AB} = [\beta_x]\phi'_B(X_A)$. The ciphertext is then the curve E_B , the images $P_B = [\beta_2]\phi_B(P_0)$, $Q_B = [\beta'_2]\phi_B(Q_0)$ and X_B , together with the curve E_{AB} , the images $P_{AB} = [\beta_2]\phi'_B(P_A)$, $Q_{AB} = [\beta'_2]\phi'_B(Q_A)$, and $ct' = \text{KDF}(X_{AB}) \oplus m$.

Decrypt. To decrypt the ciphertext $(E_B, P_B, Q_B, X_B, E_{AB}, P_{AB}, Q_{AB}, ct')$, Alice scales the points P_{AB}, Q_{AB} by $1/\alpha_2$ and $1/\alpha'_2$ respectively to obtain the images of P_B, Q_B under ϕ'_A . With such information, she can obtain a two-dimensional representation of ϕ'_A , from which she can obtain the point $X_{AB} = [\alpha_x]\phi'_A(X_B)$. The message is then $m = \text{KDF}(X_{AB}) \oplus ct'$.

Security. The security of the protocol informally relies on the hardness of recovering either the secret isogeny ϕ_A from its public key, or the isogenies ϕ_B and ϕ'_B from the ciphertext.

More formally, the security of Alice's secret isogeny relies on the hardness of the following problem:

Problem 3 (Secret-Degree Isogeny (SDI) Problem). Let p be a prime of the form $p = 2^a 3^b f - 1$, with x a prime divisor of $p - 1$. Let E be a supersingular elliptic curve defined over \mathbb{F}_{p^2} , and write P_0, Q_0 for a basis of $E[2^a]$, R_0, S_0 for a basis of $E[3^b]$, and X_0 for a point of order x . Let q be a random prime in $[0, 2^a]$. Let $\phi : E \rightarrow E'$ be a random isogeny of degree $q(2^a - q)$, generated as in [Algorithm 2](#). Given E, E' , the points P_0, Q_0, R_0, S_0, X_0 and the image points $[\alpha_2]\phi(P_0)$, $[\alpha'_2]\phi(Q_0)$, $[\alpha_3]\phi(R_0)$, $[\alpha_3]\phi(S_0)$, $[\alpha_x]\phi(X_0)$, where $\alpha_2, \alpha'_2, \alpha_3, \alpha_x \leftarrow_{\$} \mathbb{Z}_{2^a}^* \times \mathbb{Z}_{2^a}^* \times \mathbb{Z}_{3^b}^* \times \mathbb{Z}_x^*$, recover ϕ .

While the security of the isogenies computed during encryption relies on the hardness of the following problem:

Problem 4 (Double Masked-Torsion Isogeny (DMTI) Problem). Let E_0, E'_0 be two supersingular elliptic curves defined over \mathbb{F}_{p^2} , connected by an isogeny $\psi : E_0 \rightarrow E'_0$. Let P_0, Q_0 span $E_0[2^a]$ and X_0 be a point on E_0 of order x , and let P'_0, Q'_0 span $E'_0[2^a]$. Let $\phi : E_0 \rightarrow E_1$ be a random isogeny of degree 3^b , and let $\phi' : E'_0 \rightarrow E'_1$ be its pushforward $\psi_*\phi$.

Given E_0, E'_0 , the points of $P_0, Q_0, X_0, P'_0, Q'_0$, the curves E_1, E'_1 , the image points $[\beta_2]\phi(P_0)$, $[\beta'_2]\phi(Q_0)$, $[\beta_x]\phi(X_0)$, $[\beta_2]\phi(P'_0)$, $[\beta'_2]\phi(Q'_0)$, where $\beta_2, \beta'_2, \beta_x \leftarrow_{\$} \mathbb{Z}_{2^a}^* \times \mathbb{Z}_{2^a}^* \times \mathbb{Z}_x^*$, recover ϕ .

The hardness of [Problem 3](#) and [Problem 4](#) guarantees security against isogeny-recovering attacks. While it is likely that any attack against the proposed protocol would eventually solve either problem, the security of the protocol relies on a stronger assumption: the hardness of the computational $\text{POKE}^{\text{FESTA}, \text{SIDH}}$ problem, which is the following.

Problem 5 (C-POKE^{FESTA, SIDH}). Let p be a prime of the form $p = 2^a 3^b f - 1$, where f is a small cofactor needed for primality. Let E_0 be a supersingular elliptic curve defined over \mathbb{F}_{p^2} . Let P_0, Q_0 be a basis of $E_0[2^a]$, R_0, S_0 a basis of $E_0[3^b]$, and X_0 a point of order x on $E_0(\mathbb{F}_{p^4})$.

Let $\phi_A : E_0 \rightarrow E_A$ be an isogeny of degree $q(2^a - q)$, for some unknown value q . Write $P_A, Q_A = [\alpha_2]\phi_A(P_0), [\alpha'_2]\phi_A(Q_0)$, $R_A, S_A = [\alpha_3]\phi_A(R_0), [\alpha_3]\phi_A(S_0)$, and $X_A = [\alpha_x]\phi_A(X_0)$, where $\alpha_2, \alpha'_2, \alpha_3, \alpha_x$ are random integers in $\mathbb{Z}_{2^a}^* \times \mathbb{Z}_{2^a}^* \times \mathbb{Z}_{3^b}^* \times \mathbb{Z}_x^*$. Let $\phi_B : E_0 \rightarrow E_B$ be an isogeny of degree 3^b , and write $\phi'_B : E_A \rightarrow E_{AB}$ for the pushforward $(\phi_A)_*(\phi_B)$. Write $P_B, Q_B = [\beta_2]\phi_B(P_0), [\beta'_2]\phi_B(Q_0)$, $P_{AB}, Q_{AB} = [\beta_2]\phi'_B(P_A), [\beta'_2]\phi'_B(Q_A)$, and $X_B = [\beta_x]\phi_B(X_0)$, where $\beta_2, \beta'_2, \beta_x$ are random integers in $\mathbb{Z}_{2^a}^* \times \mathbb{Z}_{2^a}^* \times \mathbb{Z}_x^*$. Lastly, write X_{AB} for the point $\phi'_B(X_A) = \phi'_A(X_B)$.

Given $(E_A, (P_A, Q_A), (R_A, S_A), X_A)$, $(E_B, (P_B, Q_B), X_B)$, and $(E_{AB}, (P_{AB}, Q_{AB}))$, compute X_{AB} .

The IND-CPA security of the protocol is formalized in the following theorem.

Theorem 6. *The $\text{POKE}^{\text{FESTA}, \text{SIDH}}$ protocol is IND-CPA secure in the random oracle model under the assumption that the C- $\text{POKE}^{\text{FESTA}, \text{SIDH}}$ problem is hard.*

Proof. The proof is analagous to the proof of security of the hashed ElGamal PKE, see for instance [37, Theorem 1]. Given an adversary \mathcal{A} that breaks the IND-CPA security of the proposed protocol, it is possible to construct an adversary \mathcal{B} that solves an instance $(E_B, P_B, Q_B, X_B, E_{AB}, P_{AB}, Q_{AB}, ct')$ of the C- $\text{POKE}^{\text{FESTA}, \text{SIDH}}$ problem. Informally, \mathcal{B} simulates the random oracle model and passes $(E_B, P_B, Q_B, X_B, E_{AB}, P_{AB}, Q_{AB}, ct^*)$ to \mathcal{A} , where ct^* is randomly sampled: if \mathcal{A} does not query the ROM with X_{AB} it cannot win the IND-CPA game, and if it does, \mathcal{B} can output a random query. With non-negligible probability (since X_{AB} is guaranteed to be among the queries), \mathcal{B} outputs the solution to the C- $\text{POKE}^{\text{FESTA}, \text{SIDH}}$ problem.

Similarly to the hashed ElGamal PKE and all the analogous constructions, including SiGamal [47], it is also possible to construct an IND-CPA secure PKE in the standard model from the decisional variant of the C- $\text{POKE}^{\text{FESTA}, \text{SIDH}}$ problem. Using standard transformations, such as the Fujisaki-Okamoto transform [33,35], it is also possible to obtain an IND-CCA secure PKE.

Hardness analysis. [Problem 3](#) asks to recover an isogeny from its action on some torsion bases when the degree is unknown. This is assumed to be hard because all isogeny-recovering attacks that exploit the action of the secret isogeny on some torsion points [14,42,54] fundamentally rely on knowing the degree of the isogeny. Thus, the problem appears to reduce to the hardness of recovering the degree $q(2^a - q)$ given the provided information. This problem is expected to be hard since the degree cannot efficiently be brute-forced (there are exponentially many possible degrees) nor recovered through pairing computations. More precisely, the pairings $e_{2^a}(P, Q)$ and $e_{2^a}(P', Q')$ yield the value $q(2^a - q)\alpha_2\alpha'_2$, while the pairings $e_{3^b}(R, S)$ and $e_{3^b}(R', S')$ yields the value $q(2^a - q)\alpha_3^2$. In both cases, the degree is hidden by the unknown scalars α_2, α'_2 , and α_3 .

Relying on secret-degree isogenies was originally proposed in the context of MD-SIDH [32], a secure variant of SIDH. However, in that case, the degree of the isogeny is known to divide a public parameter, whereas in our case no information is known about the degree of the isogeny. Indeed, [Problem 3](#) appears

to provide significantly less information than [32, Problem 5] since part of the torsion information is scaled under a diagonal matrix, and no multiple of the degree is known. While a formal reduction is hard due to the difference in parameters, [Problem 3](#) is expected to be at least as hard as [32, Problem 5]. We further discuss the hardness of [Problem 3](#) in [Appendix C](#).

[Problem 4](#) is a two-instance variant of the more common problem of recovering the isogeny from its scaled action on the torsion points. Such a problem is known in the literature as the Computational Isogeny with Scaled Torsion (CIST) problem, and it was first introduced to argue the security of FESTA [8, Problem 7], but was also used in QFESTA [48], IS-CUBE [45], and binSIDH and terSIDH [6]. The CIST problem is believed to be hard because the revealed images of the torsion points are scaled under different unknown scalars: this reveals significantly less information than what is needed to apply the attacks against SIDH-like isogenies [14,42,54].

While [Problem 4](#) reveals additional information compared to the CIST problem since two related instances are revealed, it appears to be as hard as the original CIST problem. The additional instance does not appear to help solving the problem. A similar argument is proposed in the security analysis of FESTA, which similarly relies on a two-instance version of CIST called CIST². In FESTA, the two isogenies in CIST² are independent and have different degrees, while in our case the two isogenies are the pushforward of one other under the isogeny connecting E and E' . This, however, does not heuristically appear to be a significant difference: even in FESTA, recovering one of the two isogenies is enough to recover both; moreover, it seems hard to exploit the parallelness of the isogenies since the isogeny from E to E' is unknown (in the protocol, this is Alice’s secret isogeny).

4.1 Efficiency

The proposed protocol is one of the most compact post-quantum PKEs and possibly the most efficient isogeny-based PKEs to date.

Parameters. To guarantee the security of the protocol, we need to select $2^a \approx 2^\lambda$ (so that brute-forcing q is hard), $3^b \approx 2^{2\lambda}$ (so that meet-in-the-middle attacks against ϕ_B are exponentially expensive), and $x \approx 2^{\lambda/2}$ (so that brute-forcing X_{AB} is hard). However, x does not need to be smooth and can be chosen among the divisors of $p - 1$. In this way, the points X_i are defined over \mathbb{F}_{p^4} (or equivalently over the twist of the curves considered), and can be efficiently represented over \mathbb{F}_{p^2} using x -only arithmetic. Hence, the underlying prime of the form $p = 2^a 3^b f - 1$ has thus size of about 3λ bits.

The public key consists of one curve and five torsion points. The X_A point is represented by its x -coordinate, whereas the remaining points can be represented as the two points $P_A + R_A$, and $Q_A + S_A$. This means that the public key requires about $\approx 20\lambda$ bits, or—if the points are compressed— $\approx 17\lambda$ bits. Note that, unless X is chosen to have smooth order, it cannot be compressed because solving discrete logarithms (needed for compression) is hard. The ciphertext,

instead, consists of two curves, five torsion points, and a message component. This thus requires 38λ bits uncompressed, or 24λ bits compressed. For $\lambda = 128$, this means that a compressed public key requires about 272 bytes, and a compressed ciphertext about 384 bytes. We remark that this construction is well-suited for a B-SIDH [20] approach: by (partially) moving the kernel of ϕ_B to \mathbb{F}_{p^4} and working over the curve twists, it may be possible to reduce the prime size without significantly affecting the efficiency of the protocol.

Implementation. We implemented the uncompressed version of the protocol in SageMath [56] using the implementation of isogenies between abelian surfaces proposed in [22]. Due to a limitation in the current implementation, the point X_0 has a large prime order, but it is defined over \mathbb{F}_{p^2} . Thus, the implementation works with a characteristic that is about 16% larger than the characteristic if X_0 were defined over \mathbb{F}_{p^4} . Hence, the current timings provide only an upper bound: reducing the characteristic to the expected size should result in a speed-up of about 25%, while an optimized implementation of the same protocol (in a low-level language, with assembly-level optimizations) should result in a speed-up of more than an order of magnitude. The timings, benchmarked on an Apple M3 CPU, are reported in Table 1.

Table 1: Performance of the P(O)KE PKE.

λ	Size (bytes)		Time (ms)		
	$ \mathbf{pk}_{\text{cmp}} $	$ \mathbf{ct}_{\text{cmp}} $	KeyGen	Encrypt	Decrypt
128	272	384	496	110	190
192	408	576	840	201	382
256	544	768	1552	342	657

Comparison with other protocols. We start by considering CSIDH [15]: while the original proposal relied on a 511-bit prime (for $\lambda = 128$), subsequent analysis [50,11,17] of quantum attacks showed that such a prime is not sufficient to guarantee the expected security. Instead, [11] proposed to use a prime of size between 2260 or 5280 bits (depending on the conservativeness of the analysis), while [17] proposes to use a 4096-bit prime. The public key and ciphertexts of CSIDH consists of a single curve defined over \mathbb{F}_p , so they require between 283 (when $\log p = 2260$) and 512 (when $\log p = 4096$) bytes. In all but the most aggressive parameter sets, the public key and ciphertext of our proposed protocol are more compact than SIDH; even in the case where $\log p = 2260$, our protocol is only slightly less compact (about 35% larger). In terms of performance, however, our protocol is significantly more efficient. A recent work [13] reports optimized implementations of CSIDH: encrypting a message (i.e. computing two group action evaluations) takes more than a second with $p = 2048$ and more than several seconds with $p = 4096$, while decryption is only twice as fast. This is significantly longer than the results of our proof-of-concept implementation in

SageMath reported in [Table 1](#), where encryption and decryption take less than 200 and 300 ms.

If we compare our protocol with other isogeny-based construction, the comparison is similarly positive. FESTA [8], as proposed, works with a 1292-bit prime for $\lambda = 128$ and requires 1,122 bytes for its ciphertexts, which are $2.9\times$ larger than our protocol. FESTA, when implemented in SageMath with the same library for isogeny computations [22] and running on the same hardware as [Table 1](#), requires 3.1 seconds for encryption and 2.8 seconds for decryption, hence more than an order of magnitude slower than our proposed protocol. FESTA was later improved in QFESTA [48]; the protocol similarly uses a prime of 3λ bits, but the ciphertext is significantly larger, since in both protocols, the ciphertext mostly consists of two curves and several points, but in QFESTA the order of the torsion points is much larger. Indeed, the QFESTA ciphertexts are more than 100 bytes larger, and their running times (also implemented in SageMath, using the same library for higher-dimensional isogenies) are similarly an order of magnitude larger. Lastly, we compare our protocol with terSIDH-hyb, the most efficient key exchange from [6]. In terSIDH-hyb, the prime characteristic is $\approx 4\times$ larger, public keys are $2.6\times$ larger, and ciphertext (if terSIDH-hyb is used as PKE through a standard transformation) is $1.2\times$ larger. The computation times of encryption is also $30\times$ slower, while decryption is $1.9\times$ faster.

5 A split KEM

In the PKE proposed in the previous section, during encryption Bob computes two isogenies: one from E_0 to E_B , independent of Alice’s public key, and one from E_A to E_{AB} . It may be thus natural to construct a split KEM [12] from the same protocol, where Bob computes the first isogeny and reveals its codomain, together with its action, as a public key. Whenever Bob wants to encrypt a message to Alice, he computes the second isogeny from E_A to E_{AB} , as in the previous construction; to decrypt, Alice proceeds as before, but now also requires Bob’s public key. Such a construction would allow Bob to implicitly authenticate himself to Alice (while the authentication remains deniable to third parties), and it would provide most of the benefits provided by a non-interactive key exchange. In particular, a split KEM could be used to instantiate a post-quantum version of the Signal’s X3DH protocol [43].

In [12], the authors propose different security notions for split KEMs. In the weakest notion, nn-IND-CCA, the attacker is a third party that sees only one encryption. Such a security notion corresponds to the classical IND-CPA security, and it is satisfied by the protocol in the previous section. However, such a security notion is not sufficient for nearly any application: if the protocol in the previous section were used as split KEM, the security of Bob’s secret key is not guaranteed even against honest-but-curious decapsulators (such a setting would be captured by a stronger security notion, mm-IND-CCA).

The attack proceeds as follows: during decapsulation, Alice constructs a representation of the isogeny $\phi'_A : E_B \rightarrow E_{AB}$, which is the pushforward of her

secret isogeny $\phi_A : E_0 \rightarrow E_A$ under Bob's long-term secret isogeny $\phi_B : E_0 \rightarrow E_B$. For any small factor $\ell \mid \deg \phi_A$, Alice can construct a point of order ℓ on E_0 (if ℓ is sufficiently small, $E_0[\ell]$ is defined over a small extension field) and obtain its scaled image on E_B under ϕ_B . If Alice were to repeat the process with multiple honestly-generated public keys, she could obtain the exact images of the ℓ -torsion points (by [7, Lemma 1], combining it with the knowledge of $\deg \phi_B$); repeating this process for multiple small factors ℓ would allow Alice to obtain enough torsion information to apply the SIDH attacks [14,42,54] and recover ϕ_B .

To avoid such an attack, we need the degree of Bob's isogenies to be secret. We thus replace Bob's computation from SIDH-like isogenies to uniSIDH isogenies. This leads us to a new instantiation of the POKE framework with FESTA isogenies for one party and uniSIDH isogenies for the latter, which we call split-POKE. The resulting protocol is secure against passive attacks (i.e. the attacker is not a participant in the protocol) and honest-but-curious participants, but is still vulnerable to active attacks. We present validation methods for both participants, obtaining thus a split KEM that is (presumably) mm-IND-CCA secure, although we leave a formal proof for future work.

5.1 The Split-POKE protocol

We first formalize the Split-POKE protocol, described in its unprotected variant.

Protocol 7 (Split-POKE). Let p be a prime of the form $p = 2^a Bf - 1$, where f is a small cofactor needed for primality. Let E_0 be the supersingular elliptic curve defined over \mathbb{F}_{p^2} with j -invariant 1728. Let P_0, Q_0 be a deterministic basis of $E_0[2^a]$, and R_0 and X_0 be deterministically computed points of order B and x .

sKeyGen_{dec}(\cdot). Alice samples a random prime q in $[0, 2^a]$ such that $2^a - q$ is also prime, and she computes a random isogeny $\phi_A : E_0 \rightarrow E_A$ of degree $q(2^a - q)$ using Algorithm 2. She also samples scaling values $\alpha_2, \alpha'_2, \alpha_B, \alpha_x \in \mathbb{Z}_{2^a}^* \times \mathbb{Z}_{2^a}^* \times \mathbb{Z}_B^* \times \mathbb{Z}_x^*$. Alice's public key consists of the curve E_A , and the points $P_A, Q_A = [\alpha_2]\phi_A(P_0), [\alpha'_2]\phi_A(Q_0)$, $R_A = [\alpha_B]\phi_A(R_0)$, and $X_A = [\alpha_x]\phi_A(X_0)$. The secret key comprises of $(q, \alpha_2, \alpha'_2, \alpha_x)$.

sKeyGen_{enc}(\cdot). Bob samples a random B' dividing B , and computes the isogeny $\phi_B : E_0 \rightarrow E_B$ with kernel $\langle [B/B']R_0 \rangle$. He then samples scaling values $\beta_2, \beta'_2, \beta_x \in \mathbb{Z}_{2^a}^* \times \mathbb{Z}_{2^a}^* \times \mathbb{Z}_x^*$, and he sets $P_B, Q_B = [\beta_2]\phi_B(P_0), [\beta'_2]\phi_B(Q_0)$ and $X_B = [\beta_x]\phi_B(X_0)$. He then publishes its public key $(E_0, E_B, P_B, Q_B, X_B)$, while storing his secret key $(B', \beta_2, \beta'_2, \beta_x)$.

sEncaps(pk_A). Bob computes the isogeny $\phi'_B : E_A \rightarrow E_{AB}$ with kernel $\langle [B/B']R_A \rangle$, and computes $P_{AB}, Q_{AB} = [\beta_2]\phi'_B(P_A), [\beta'_2]\phi'_B(Q_A)$ and $X_{AB} = [\beta'_x]\phi'_B(X_A)$. He then sends the ciphertext $\text{ct} = (E_{AB}, P_{AB}, Q_{AB})$ to Alice, while outputting the shared secret $\text{ss}_B = X_{AB}$.

sDecaps(pk_B, ct). Alice computes the isogeny Φ with kernel $\langle ([-q]P_A, [1/\alpha_2]P_{AB}), ([-q]Q_A, [1/\alpha'_2]Q_{AB}) \rangle$ that represents the isogeny $\phi'_A : E_A \rightarrow E_{AB}$. She outputs the shared secret $\text{ss}_A = \phi'_A(X_B)$.

5.2 Validating FESTA-like isogenies

Consider a POKE instantiation where Alice computes FESTA isogenies, similarly to P(O)KE PKE. During the execution of the protocol, she receives two curves E_B and E_{AB} , together with points $P_B, Q_B \in E_B[2^a]$ and $P_{AB}, Q_{AB} \in E_{AB}[2^a]$. To decrypt, she scales P_{AB}, Q_{AB} by $1/\alpha_2$ and $1/\alpha'_2$ respectively, and then attempts to compute the isogeny Φ with kernel $\langle ([-q]P_B, [1/\alpha_2]P_{AB}), ([-q]Q_B, [1/\alpha'_2]Q_{AB}) \rangle$. We identify three possible scenarios: 1. The computation of Φ fails, i.e. the points P_B, Q_B and P_{AB}, Q_{AB} , when scaled appropriately, do not generate the kernel of a two-dimensional isogeny between products of elliptic curves. In this case, Alice aborts. 2. The computation of Φ succeeds, but the points $[1/\alpha_2]P_{AB}$ and $[1/\alpha'_2]Q_{AB}$ are not the images of P_B and Q_B under Φ . The attack against an unprotected variant of FESTA presented in [46] falls into this category. However, this can be easily checked: once Φ is recovered, it is sufficient to map P_B and Q_B under Φ and compare the results with scaled P_{AB} and Q_{AB} . If they do not match, Alice aborts. 3. The computation of Φ succeeds, and the points $[1/\alpha_2]P_{AB}$ and $[1/\alpha'_2]Q_{AB}$ are the images of P_B and Q_B under Φ . In this case, P_{AB} and Q_{AB} must be honestly generated: since α_2 and α'_2 are odd, their inverse modulo 2^a is unique.

Hence, for Alice to protect herself from adaptive attacks, it is sufficient to evaluate the points P_B, Q_B under the recovered isogeny Φ and check them against P_{AB}, Q_{AB} . If the points P_{AB}, Q_{AB} are not the images of P_B, Q_B , scaled by α_2, α'_2 , Alice recognizes the inputs are maliciously generated and aborts. In all other cases, the input must be honestly generated. Note that Alice is already mapping X_B under Φ to recover the message, so the additional cost of this validation is minimal.

5.3 Validating uniSIDH-like isogenies

To compute a ciphertext, Bob considers a public key containing a curve E_A and a point R_A of order $B = \prod_{i=1}^t p_i$ and computes the isogeny ϕ'_B with kernel $\langle [B/B']R_A \rangle$, for some secret B' dividing B . If we write the point R_A as the sum of t points of order p_i , i.e. $R_A = R_{A,1} + R_{A,2} + \dots + R_{A,t}$, where $\text{ord } R_{A,i} = p_i$, the choice of B' is equivalent to choosing a subset of the R_i 's that contribute to the kernel computations. In other words, we have

$$\ker \phi'_B = \langle R_{A,1}^*, R_{A,2}^*, \dots, R_{A,t}^* \rangle \quad \text{where } R_{A,i}^* = \begin{cases} R_{A,i} & \text{if } p_i \mid B', \\ \mathcal{O} & \text{if } p_i \nmid B'. \end{cases}$$

Such a construction is easily vulnerable to adaptive attacks, as noted in [6]: if the point R_i is changed (i.e. is not honestly generated), whether the curve E_{AB} changes as well reveals whether $R_i \in \ker \phi'_B$. Repeating this process for all t points R_i reveals the exact value of B' , and thus the secret isogenies ϕ'_B and ϕ_B .

To protect against such attacks, we introduce a new validation method. We first describe it as an interactive protocol, and later we show how to reduce the number of interactions so that it can be used in the split-KEM setting.

The validation approach relies on the fact that a malicious participant who produces a maliciously generated public key cannot, in most cases, easily decapsulate a ciphertext encrypted with that public key. Our solution to validate a public key is thus to generate several ciphertexts with different ephemeral encapsulation keys, and ask the public key owner to produce the decapsulation output. If the process is repeated for sufficiently many encapsulation keys, the decapsulator can respond with the correct message only if the public key is honestly generated. Note that a similar approach was proposed, in the context of SIDH, in [3] as k -SIDH. In k -SIDH, however, the goal was to build a NIKE: hence, the multiple exchanges were between static keys and thus required a very large number of interactions. In our case, the exchanges are between ephemeral keys (the validation method can be interpreted as a static-ephemeral version of k -SIDH), and thus the number of interactions is much smaller.

Remark 8. A decapsulation public key also contains two points P_A, Q_A of order 2^a , which are mapped by Bob under ϕ'_B and scaled by values β_2, β'_2 . These points do not interact with Bob's secret key in any meaningful way and thus cannot be used to actively attack Bob's secret key.

The interactive validation technique. Let us assume that a malicious Alice has generated a public key (E_A, R_A) , where all but one points are honestly generated (this is the most advantageous scenario for the attacker). In other words, if we write the points R_0 on E_0 and R_A on E_A as the sum

$$R_k = R_{k,1} + R_{k,2} + \dots + R_{k,t}, \text{ where } k \in \{0, A\} \text{ and } \text{ord } R_{k,i} = p_i,$$

we have that for all $i \in [t] \setminus \{j\}$, we have $\phi_A(R_{0,i}) = [\alpha_i]R_{A,i}$, for some $\alpha_i \in \mathbb{Z}_B^*$.

Bob now generates an ephemeral secret isogeny $\tilde{\phi}_B : E_0 \rightarrow \tilde{E}_B$ and produces both an encapsulation public key $(\tilde{E}_B, \tilde{P}_B, \tilde{Q}_B, \tilde{X}_B)$ and a ciphertext $(\tilde{E}_{AB}, \tilde{P}_{AB}, \tilde{Q}_{AB})$, obtained using the ephemeral encapsulation public key and the decapsulation public key to validate.

If the point $R_{0,j}$ is not in the kernel of $\tilde{\phi}_B$, then the maliciously generated point $R_{A,j}$ is not in the kernel of $\tilde{\phi}'_B$, and thus Alice can easily decapsulate the ciphertext. On the other hand, if the point $R_{0,j}$ is in the kernel of $\tilde{\phi}_B$, then the curve \tilde{E}_{BA} (computed by Bob) is not the same as the codomain \tilde{E}_{AB} of $\tilde{\phi}'_A$ (the pushforward of ϕ_A under $\tilde{\phi}_B$), which prevents Alice from computing the point \tilde{X}_{AB} . However, since only $R_{A,j}$ is maliciously generated, the isogeny $\tilde{\phi}'_B$ deviated only by one p_j -isogeny, and hence the curves \tilde{E}_{AB} and \tilde{E}_{BA} are p_j^2 -isogenous. This means that Alice can still recover the point \tilde{X}_{AB} by computing a p_j^2 -isogeny from \tilde{E}_{AB} and mapping \tilde{X}_{AB} under it: with probability $(p_j(p_j + 1))^{-1}$ (i.e. the probability of having guessed \tilde{E}_{BA} correctly), the resulting point will be the correct message.

For reasonably-sized parameters, the probability of Alice successfully decapsulating the ciphertext is much higher than $2^{-\lambda}$. To obtain the desired security level, Bob can repeat the process with multiple ephemeral keys, and ask Alice to decapsulate the resulting ciphertexts. If the process is repeated k times and all k

ephemeral keys use the malicious point $R_{A,j}$, Alice can correctly decapsulate all ciphertexts only if all k guesses are correct (she has no way to check whether a single ciphertext was correctly decapsulated); she thus responds correctly only with probability $(p_i(p_i + 1))^{-k}$.

However, if the k ephemeral keys are generated completely at random, the probability of a key using a given point is only $1/2$, so the success probability of Alice may be much higher. To avoid this, Bob generates the ephemeral keys in such a way that, for all $i \in [t]$, exactly $k/2$ keys use the point $R_{0,i}$ and $k/2$ keys do not.² In this way, the best strategy for a malicious Alice is to guess which $k/2$ public-key/ciphertexts pairs use the malicious point $R_{A,j}$ (the probability of guessing correctly is $1/\binom{k}{k/2}$), and guessing the curves \tilde{E}_{AB} for those $k/2$ cases.

Hence, the probability of Alice successfully decapsulating all k ciphertexts is at most $\binom{k}{k/2}^{-1} \cdot (p_1(p_1 + 1))^{-k/2}$, where p_1 is the smallest prime dividing B . This introduces a trade-off between the number of repetitions and the smallest prime p_1 . If, for instance p_1 is chosen to be $p_1 = 929$, only $k = 12$ repetitions are sufficient to guarantee that Alice can decapsulate all ciphertexts with probability at most 2^{-128} , while if $p_1 = 149$, $k = 16$ repetitions are required. Note that while this validation method still requires to rerun the encapsulation/decapsulation protocol multiple times, the number of repetitions is more than an order of magnitude smaller than the number of repetitions required in proofs of public-key correctness [24,44]. Moreover, our validation method allows us to keep the degree $q(2^a - q)$ secret, which is needed for the security of the split KEM. We summarize the validation protocol in [Protocol 9](#).

Protocol 9. The protocol involves a prover P and a verifier V , and it relies on a repetition parameter k . The protocol defines public parameters $p = 2^a B f - 1$, where $B = \prod_{i=1}^t p_i$ is the product of t small primes and f is a cofactor, and $(E_0, P_0, Q_0, R_0, X_0)$, where E_0 is a supersingular elliptic curve defined over \mathbb{F}_{p^2} , and $\langle P_0, Q_0 \rangle = E_0[2^a]$, $X_0 \in E_0[x]$, and $R_0 \in E_0[B]$ are points of full order. The prover wants to prove that their public key $\mathsf{pk}_{\mathsf{P}} = (E_{\mathsf{P}}, P_{\mathsf{P}}, Q_{\mathsf{P}}, R_{\mathsf{P}}, X_{\mathsf{P}})$ contains honestly generated points R_{P} and X_{P} , i.e. $R_{\mathsf{P}} = [\rho_B] \phi_{\mathsf{P}}(R_0)$ and $X_{\mathsf{P}} = [\rho_x] \phi_{\mathsf{P}}(x_0)$, where $\rho_B, \rho_x \in \mathbb{Z}_{2^a}^* \times \mathbb{Z}_x^*$ and ϕ_{P} is an isogeny from E_0 to E_{P} of degree at most $2^{2a} C$, where $C < 2^{\lambda/k}$.

Challenge. The verifier repeats the following actions for each $i \in [k]$ ³:

1. Sample a random uniSIDH isogeny $\phi_{\mathsf{V}} : E_0 \rightarrow E_{\mathsf{V}}$ with kernel $\langle [B/B']R_0 \rangle$, for a random integer B' dividing B .
2. Compute the parallel isogeny $\phi'_{\mathsf{V}} : E_{\mathsf{P}} \rightarrow E_{\mathsf{P}\mathsf{V}}$ with kernel $\langle [B/B']R_{\mathsf{P}} \rangle$.

² This can be achieved by generating t binary strings of length k with precisely $k/2$ ones and $k/2$ zeros: if the i -th entry in the j -th string is a one, the j -th ephemeral key will use point R_i . This guarantees that any single ephemeral key, if considered individually, is uniformly distributed, while also keeping the desired property.

³ For ease of notation, we drop the i -th marker from each element, but the isogenies $\phi_{\mathsf{V}}, \phi'_{\mathsf{V}}$, the scaling values B', ν_2, ν'_2, ν_x , and all the computed points vary across each iteration.

3. Samples random values $\nu_2, \nu'_2, \nu_x \in \mathbb{Z}_{2^a}^* \times \mathbb{Z}_{2^a}^* \times \mathbb{Z}_x^*$.
4. Compute $P_V, Q_V = [\nu_2]\phi_V(P_0), [\nu'_2]\phi_V(Q_0)$.
5. Compute $P_{PV}, Q_{PV} = [\nu_2]\phi'_V(P_P), [\nu'_2]\phi'_V(Q_P)$.
6. Compute $X_V, X_{PV} = [\nu_x]\phi_V(X_0), [\nu_x]\phi'_V(X_P)$.
7. Sets $\text{chall}_i = ((E_V, P_V, Q_V, X_V), (E_{PV}, P_{PV}, Q_{PV}))$ and $\text{rsp}_i = X_{PV}$.

The verifier sends the sets of challenges $\{\text{chall}_i\}_{i \in [k]}$ to the prover P.

Response. For each $\text{chall}_i = ((E_V, P_V, Q_V, X_V), (E_{PV}, P_{PV}, Q_{PV}))$, the prover uses P_V, Q_V and P_{PV}, Q_{PV} to recover the pushforward $\phi'_P : E_V \rightarrow E_{PV}$ of ϕ_P under the isogeny $\phi_V : E_0 \rightarrow E_V$, and computes $\text{r}\tilde{\text{sp}}_i = [\rho_x]\phi'_P(X_V)$. The responder checks that P_{PV}, Q_{PV} are the images of P_V, Q_V under ϕ'_P : if not, they abort. The prover sends the sets of responses $\{\text{r}\tilde{\text{sp}}_i\}_{i \in [k]}$ to the verifier V.

Verification. The verifier outputs valid if $\text{rsp}_i = \text{r}\tilde{\text{sp}}_i$ for all $i \in [k]$, otherwise they output invalid.

Saving one round of communication. The validation technique presented so far is inherently interactive: Bob sends several ephemeral public keys and ciphertexts to Alice, and she responds with their decapsulation. In the context of a split KEM, this would increase the rounds of interaction from one to three: one for Bob's ephemeral encapsulation, one for Alice's response, and—if she passes verification—one for sending the ciphertext encapsulated with the long-term key. However, Bob verification only consists of an equality check, which we can exploit to reduce the number of rounds. Let ct be the ciphertext encapsulated under Bob's long-term key, and let $(\tilde{ct}_1, \dots, \tilde{ct}_k)$ be the verification ciphertexts encapsulated under ephemeral keys. Let also $\tilde{K}_1, \dots, \tilde{K}_k$ be the shared secrets associated with verification ciphertexts. Bob computes the value $K = \mathcal{H}(\tilde{K}_1 \parallel \dots \parallel \tilde{K}_k)$ (where \mathcal{H} is a hash function) and encrypts the ciphertext ct with an IND-CPA-secure symmetric encryption $(\text{KeyGen}^{\text{sym}}, \text{Enc}^{\text{sym}}, \text{Dec}^{\text{sym}})$ scheme using K as key. He then sends $(\text{Enc}_K^{\text{sym}}(ct), (\tilde{ct}_1, \dots, \tilde{ct}_k))$ to Alice. An honest Alice can compute K and obtain the ciphertext ct , while a malicious one cannot efficiently compute K , which fully hides ct .

When validating the inputs of FESTA-like isogenies, the validation method is perfectly correct: any input that is not honestly generated is detected. In this case, however, the validation offers only computational guarantees. We thus formalize the security of the validation method with the following game and assumption.

Game 10 (uniSIDH validation). Let $p = 2^a Bf - 1$ be a prime, where $B = \prod_{i=1}^t p_i$ is the product of t small primes, and f is a cofactor. Let E_0 be the supersingular elliptic curve defined over \mathbb{F}_{p^2} , and let $\langle P_0, Q_0 \rangle = E_0[2^a]$, $X_0 \in E_0[x]$, and $R_0 \in E_0[B]$ be protocol parameters of $\text{POKE}^{\text{FESTA}, \text{uniSIDH}}$. Lastly, let $(\text{KeyGen}, \text{Enc}, \text{Dec})$ be the PKE protocol P(O)KE instantiated with $\text{POKE}^{\text{FESTA}, \text{uniSIDH}}$.

The game takes place between a challenger and an adversary \mathcal{A} , and it proceeds as follows:

1. The adversary \mathcal{A} outputs a public key $\text{pk}_{\mathcal{A}} = (E_{\mathcal{A}}, P_{\mathcal{A}}, Q_{\mathcal{A}}, R_{\mathcal{A}}, X_{\mathcal{A}})$.

2. The challenger:
 - (a) Samples a random bit $b \leftarrow \{0, 1\}$.
 - (b) Generates k keys $\{\text{pk}_i, \text{sk}_i \leftarrow \text{sKeyGen}_{\text{enc}}()\}_{i \in [k]}$,
 - (c) Computes k encapsulations $\{\text{ct}_i, \text{ss}_i^0 \leftarrow \text{sEncaps}(\text{pk}_i, \text{sk}_i)\}_{i \in [k]}$,
 - (d) Samples k random values $\{\text{ss}_i^1\}_{i \in [k]}$ in the space of shared secrets,
 - (e) Responds $((\text{ct}_1, \dots, \text{ct}_k), (\text{ss}_1^b, \dots, \text{ss}_k^b))$ to \mathcal{A} .
3. The adversary \mathcal{A} outputs a bit b' .

The adversary \mathcal{A} wins if $b = b'$.

Assumption 11 (uniSIDH validation). For any PPT adversary \mathcal{A} that wins the uniSIDH validation game with non-negligible probability, there exists a PPT “extractor” $\bar{\mathcal{A}}$ that, given the same input as \mathcal{A} , outputs $\phi_{\mathcal{A}} : E_0 \rightarrow E_{\mathcal{A}}$ such that $\phi_{\mathcal{A}}(P_0)$, $\phi_{\mathcal{A}}(Q_0)$, $\phi_{\mathcal{A}}(X_0)$, and $\phi_{\mathcal{A}}(R_0)$ are respectively equal to $P_{\mathcal{A}}$, $Q_{\mathcal{A}}$, $R_{\mathcal{A}}$, and $X_{\mathcal{A}}$, up to scalars, and $\deg \phi_{\mathcal{A}} < 2^{2a}C$, where $C < 2^{\lambda/k}$.

The previous assumption relies on an isogeny whose degree is bounded. If $\deg \phi_{\mathcal{A}} < 2^{2a}$, this corresponds to the honest case. However, we have to extend the bound to include a constant C , bounded by $2^{\lambda/k}$. That is because an attacker may compute an isogeny slightly longer than 2^{2a} , say $2^{2a}c$: then, they may still win [Game 10](#) by guessing enough the translation of such a longer isogeny. If they behave honestly elsewhere, this happens with an advantage of $1/c^k$.

More fundamentally, the assumption is stated in terms of an extractor, similarly to the knowledge-of-exponent assumptions [\[9\]](#), to capture the fact the adversary is free to produce any public key and can easily win [Game 10](#) if the public key is honestly generated. To obtain a more limited assumption that does not rely on an extractor, we could assume the attacker is algebraic, i.e. it produces an isogeny from E_0 to $E_{\mathcal{A}}$; in that case, assuming the adversary has negligible advantage in winning [Game 10](#) would be a better assumption. However, no formalization of algebraic adversaries for non-group-action isogenies is reported in the literature, and thus we leave this as future work.

6 A two-round verifiable OPRF: POKE-OPRF

We present a two-round verifiable OPRF protocol based on POKE. The high-level design follows that of the OPRFs presented in [\[10,4\]](#). In all constructions, the client hashes the input m to a curve E_m , computes a blinding isogeny $\phi_b : E_m \rightarrow E_{mb}$, and sends E_{mb} , together with the appropriate torsion points and a proof of their validity, to the server. The server responds by computing an isogeny $\phi_S : E_{mb} \rightarrow E_{mbS}$ associated with its private key, and responds with E_{mbS} , together with the torsion information needed for the client to “undo” the effect of the blinding isogeny and obtain the curve E_{mS} .

Our OPRF, however, crucially differs in two ways from previous protocols: our construction replaces the underlying key exchange from SIDH and M-SIDH to a POKE-based construction where the two parties compute FESTA and

uniSIDH isogenies. This reduces the size of the underlying prime when compared to [4] (by more than 6 times), leading to a much more compact and efficient protocol. More importantly, using a POKE-based construction allows us to use the validation method proposed in Section 5.3 to prevent active attacks on the server’s long-term key. While such a method requires to run k instances of POKE in parallel (with $k \approx 12$, for $\lambda = 128$), it replaces the very expensive proof of isogeny knowledge that was used in [10,4]. These proofs require repeating the underlying key exchange computations between 219 and 486 times (also for $\lambda = 128$); thus, our construction reduces the computations needed (and the communication cost) by more than an order of magnitude.

6.1 The OPRF protocol

The protocol proceeds as follows. Fixed a prime p of the form $p = 2^a S f - 1$, where $S = \prod_{i=1}^t p_i$ is the product of t small primes⁴ and f is a cofactor, denote with E_0 the curve with j -invariant 1728. Let P_0, Q_0 be a deterministically generated basis of $E_0[2^a]$ (for instance, using the algorithms from [51]), and let R_0 and X_0 be a deterministically generated points on E_0 of order S and x , respectively. Since the points are deterministically generated, they can be considered as randomly chosen, and thus the backdoor attack introduced by [16] does not apply.

The server generates a public key by sampling a random uniSIDH isogeny $\phi_S : E_0 \rightarrow E_S$ and revealing its action on the 2^a -torsion, scaled under a diagonal matrix. The client, to evaluate the OPRF on an input m , hashes the input m to a prime degree q such that $2^a - q$ is also prime.⁵ They then proceed to compute an isogeny $\phi_m : E_0 \rightarrow E_m$ of degree q , dependent exclusively on the input m , and a random isogeny $\phi_b : E_m \rightarrow E_{mb}$ of degree $2^a - q$ (see Remark 13 for a discussion on how to compute this step). The client then reveals the curve E_{mb} to the server, together with the action of $\phi_b \circ \phi_m$ on the 2^a -torsion, scaled under a diagonal matrix, and on the points R_0 and X_0 , scaled under independent scalars.

The server computes its response by translating the secret isogeny ϕ_S under $\phi_b \circ \phi_m$ using the image of R_0 to obtain $\phi'_S : E_{mb} \rightarrow E_{mbS}$. Its response then consists of the curve E_{mbS} , together with its action on the 2^a -torsion, scaled under the same diagonal matrix used during key generation. However, the server needs to first check that the client’s query is honest: to do so, it uses the validation method proposed in Protocol 9. In particular, it samples k ephemeral uniSIDH isogenies $\phi_{S,i} : E_0 \rightarrow E_{S,i}$, computes their parallel isogeny $\phi'_{S,i} : E_{mb} \rightarrow E_{mbS,i}$ under $\phi_b \circ \phi_m$, and sends the client the curves $E_{S,i}, E_{mbS,i}$, together with the action of ϕ_S and ϕ'_S on the 2^a -torsion, scaled under the same diagonal matrix, and the scaled action of ϕ_S on X_0 . The server then computes the point $X_{mbS,i}$ as

⁴ The value S is the same as B in the split KEM; however, in the OPRF protocol, it is associated with the degree of the isogenies computed by the server. Thus, for clarity, is renamed S .

⁵ The primality of $2^a - q$ does not seem to be a strict requirement for security, but it adds an additional layer of security (a malicious server cannot guess the last steps of ϕ_b), and it simplifies the security argument, especially for the verifiable version.

the scaled image of X_{mb} (the image of X_0 on E_{mb} revealed by the client); it then encrypts its response under a symmetric encryption scheme with a key derived by all $X_{mbS,i}$, and sends the ciphertext to the client.

If the client has been honest, they can compute the points $X_{mbS,i}$ given the information revealed by the server (see [Protocol 9](#)): with that information, they can decrypt the symmetric ciphertext and obtain the response curve E_{mbS} and the scaled action of ϕ'_S on the 2^a -torsion. Given the images of the 2^a -torsion on E_S and E_{mbS} have been scaled by the same matrix, the client can rely on them to recover a representation of the isogenies $\phi'_m : E_S \rightarrow E_{out}$ and $\phi'_b : E_{out} \rightarrow E_{mbS}$ (see the Decryption algorithm of [Protocol 2](#)), which allows them to undo the effect of the blinding isogeny and obtain the curve E_{out} . The OPRF output is then the hash of the server's public key, the input message, and the j -invariant of E_{out} . The protocol is formalized in [Protocol 12](#).

Protocol 12 (POKE-OPRF). The protocol takes place between a client and a server. The protocol relies on three hash functions, dependent on the parameters generated during the setup phase:

- \mathcal{H}_m , from \mathcal{M} , the space of inputs of the OPRF, to the set of isogenies from E_0 of order q , such that q and $2^a - q$ are prime;
- \mathcal{H}_{chall} , from \mathcal{C}^k to $\{0, 1\}^\lambda$, where \mathcal{C} is the space of challenges using in Prot. 9;
- \mathcal{H}_{out} , from $\mathcal{S} \times \mathcal{M} \times \mathbb{F}_{p^2}$ to $\{0, 1\}^\lambda$, where \mathcal{S} is the space of public keys.

Setup: The setup phase determines a prime p of the form $p = 2^a S f - 1$, where $S = \prod_{i=1}^t p_i$ is the product of t small primes, and f is a cofactor. The setup also fixes x to be a large prime dividing $p - 1$, and k to be an integer denoting the number of iterations used for verification. The curve E_0 then denotes the supersingular elliptic curve with j -invariant 1728. The setup also deterministically generates the basis P_0, Q_0 of $E_0[2^a]$ and the points R_0 and X_0 on E_0 of order S and x , respectively.

KeyGen: The server generates a secret key $sk_S = S'$, where S' is a random divisor of S , and computes the isogeny $\phi_S : E_0 \rightarrow E_S$ with kernel $\langle [S/S']R_0 \rangle$. It samples scaling values $\beta_2, \beta'_2 \in \mathbb{Z}_{2^a}^* \times \mathbb{Z}_{2^a}^*$, and it sets $P_S, Q_S = [\beta_2]\phi_S(P_0), [\beta'_2]\phi_S(Q_0)$. It then publishes its public key $pk_S = (E_0, E_S, P_S, Q_S)$.

ClientReq: To evaluate the OPRF on an input $m \in \mathcal{M}$, the client:

1. Generate $(\phi_m : E_0 \rightarrow E_m) \leftarrow \mathcal{H}_m(m)$ and set $q = \deg \phi_m$;
2. Sample a random isogeny $\phi_b : E_m \rightarrow E_{mb}$ of degree $2^a - q$;
3. Sets $\phi_{mb} : E_0 \rightarrow E_{mb}$ as the composition $\phi_b \circ \phi_m$;
4. Samples random values $\alpha_2, \alpha'_2, \alpha_S, \alpha_x \in \mathbb{Z}_{2^a}^* \times \mathbb{Z}_{2^a}^* \times \mathbb{Z}_B^* \times \mathbb{Z}_x^*$;
5. Sets $P_{mb}, Q_{mb} = [\alpha_2]\phi_{mb}(P_0), [\alpha'_2]\phi_{mb}(Q_0)$;
6. Sets $R_{mb} = [\alpha_S]\phi_{mb}(R_0)$ and $X_{mb} = [\alpha_x]\phi_{mb}(X_0)$;
7. Sends $req = (E_{mb}, P_{mb}, Q_{mb}, R_{mb}, X_{mb})$ to the server.

BlindEval: After receiving the client's request $(E_{mb}, P_{mb}, Q_{mb}, R_{mb}, X_{mb})$, the server:

8. Computes the isogeny $\phi'_S : E_{mb} \rightarrow E_{mbS}$ with kernel $\langle [S/S']R_{mb} \rangle$;
9. Computes $P_{mbS}, Q_{mbS} = [\beta_2]\phi'_S(P_{mb}), [\beta'_2]\phi'_S(Q_{mb})$;
10. Sets $\mathbf{rsp} = (E_{mbS}, P_{mbS}, Q_{mbS})$;
11. Computes k pairs $(\mathbf{rsp}_i, \mathbf{chall}_i)$ as in the Challenge algorithm of [Protocol 9](#);
12. Computes $K = \mathcal{H}_{\text{chall}}(\mathbf{rsp}_1 \parallel \mathbf{rsp}_2 \parallel \dots \parallel \mathbf{rsp}_k)$;
13. Encrypts \mathbf{rsp} under K to obtain $\text{CT} = \text{Enc}_K^{\text{sym}}(\mathbf{rsp})$;
14. Sends $(\text{CT}, (\mathbf{chall}_1, \dots, \mathbf{chall}_k))$ to the client.

Finalize: After receiving the server's response $(\text{CT}, (\mathbf{chall}_1, \dots, \mathbf{chall}_k))$, the client:

15. Computes k responses \mathbf{rsp}_i as in the Response algorithm of [Protocol 9](#);
16. Computes $K = \mathcal{H}(\mathbf{rsp}_1 \parallel \mathbf{rsp}_2 \parallel \dots \parallel \mathbf{rsp}_k)$;
17. Decrypts CT with key K to obtain \mathbf{rsp} , i.e. $\mathbf{rsp} = \text{Dec}_K^{\text{sym}}(\text{CT})$;
18. Parses the responses \mathbf{rsp} as $(E_{mbS}, P_{mbS}, Q_{mbS})$;
19. Computes $\Phi : E_S \times E_{mbS} \rightarrow E_{\text{out}}^0 \times E_{\text{out}}^1$ with kernel $\langle ([-q]P_S, [1/\alpha_2]P_{mbS}), ([-q]Q_S, [1/\alpha'_2]Q_{mbS}) \rangle$;
20. If the computation of Φ fails, the client outputs \perp ;
21. Sets E_{out} as the curve in $\{E_{\text{out}}^0, E_{\text{out}}^1\}$ that is $(2^a - q)$ -isogenous to E_{mbS} ;⁶
22. Writes $\phi'_{mb} : E_S \rightarrow E_{mbS}$ as the $q(2^a - q)$ -isogeny represented by Φ ;
23. Checks that $P_{mbS}, Q_{mbS} = \phi'_{mb}(P_S), \phi'_{mb}(Q_S)$, and outputs \perp if not;
24. Outputs $\mathcal{H}_{\text{out}}(\mathbf{pk}_S, m, j(E_{\text{out}}))$;

The OPRF in [Protocol 12](#) is correct: the output produced by Finalize depends only on the input m and the server's secret key S' , and it is independent of the choice of blinding isogeny ϕ_b . This is because the output of Finalize depends on \mathbf{pk}_S , which deterministically depend only on the server's secret key S' , the input m , and E_{out} . By construction, this curve is isomorphic to the codomain of the isogeny $\phi_{mS} : E_0 \rightarrow E_0 / \langle [S/S']R_0, \ker \phi_m \rangle$, so it depends exclusively on S' and m . In other words, if we define $\text{Eval}(m, S')$ to be the function that outputs $\mathcal{H}_{\text{out}}(\mathbf{pk}_S, m, j(E_0 / \langle [S/S']R_0, \ker \phi_m \rangle))$, where ϕ_m is computed as in lines 1-2 of [ClientReq](#), we have that $\text{Finalize}(\text{BlindEval}(\mathbf{sk}_S, \text{ClientReq}(\mathbf{pk}_S, m))) = \text{Eval}(\mathbf{sk}_S, m)$, for any parameters produced by [Setup](#), for any server public and secret key $(\mathbf{pk}_S, \mathbf{sk}_S)$, and for any input $m \in \mathcal{M}$.

Remark 13. To compute their request, the client needs to compute a deterministic (on the input m) isogeny of degree q and a random isogeny of degree $2^a - q$ (Lines 1 and 2 of [Protocol 12](#)). This cannot be computed with an approach similar to [Algorithm 2](#): directly generating an isogeny of degree $q(2^a - q)$ and splitting it into its components would require both isogenies to depend on m or be randomly sampled. Generating two distinct isogenies of degree q and $2^a - q$ from E_0 is possible with [Algorithm 2](#), but there is no simple way to compute the pushforward of one under the other to obtain an isogeny of degree $q(2^a - q)$ from

⁶ Concretely, this is achieved by mapping P_{mbS} and Q_{mbS} under Φ and checking their pairing. Write $P_{\text{out}}^0, P_{\text{out}}^1 = \Phi(\mathcal{O}_{E_S}, P_{mbS})$ and $Q_{\text{out}}^0, Q_{\text{out}}^1 = \Phi(\mathcal{O}_{E_S}, Q_{mbS})$: if $e_{2^a}(P_{\text{out}}^i, Q_{\text{out}}^i) = e_{2^a}(P_{mbS}, Q_{mbS})^q$, the curve E_{out}^i is q -isogenous to E_{mbS} .

E_0 . While we leave as an open problem whether a QFESTA-like approach (as that of [Algorithm 2](#)) can be adapted to work in this instance, a possible solution lies in working explicitly over the quaternion side: the client can generate an ideal I_m of norm q dependent on m and a random ideal I_b of norm $2^a - q$, and compute the desired isogeny of degree $q(2^a - q)$ as the translation from ideal to isogeny (see [\[27, Sec. 8.1\]](#) and [\[28, Sec. 4.1\]](#)) of $I_m \cap I_b$ [\[27, Lemma 3\]](#).

Alternatively, we can modify the OPRF protocol to a message isogeny of degree $q(2^a - q)$. After computing the curve E_m , the client can sample a random rational isogeny $\phi_b : E_m \rightarrow E_{mb}$ of smooth degree (say 3^b , if the prime p is changed such that $3^b \mid p + 1$), and send the curve E_{mb} to the server. The server proceeds as in the protocol, but also reveals the action of its secret isogenies on the 3^b -torsion (which remains secure because the degree of all the server's isogenies is secret). The client can then recover the isogenies parallel to $\phi_b \circ \phi_m$ by first computing the 3^b component (using the 3^b -torsion) and then recovering the $q(2^a - q)$ component by using higher-dimensional representations.

The two constructions are very similar, and the analysis and security arguments can be easily adapted from one to the other. We present and analyze the version of [Protocol 12](#) with message isogeny of degree q and random isogeny of degree $2^a - q$ to simplify the description and analysis.

6.2 Verifiability

The OPRF proposed in [Protocol 12](#), considered as a non-verifiable OPRF, already significantly improves on previous constructions with a similar design [\[10,4\]](#): the expensive proof of isogeny knowledge used to guarantee the security of the server is replaced with the more efficient validation method of [Protocol 9](#). However, previous OPRF constructions also relied on expensive zero-knowledge proofs to achieve verifiability. These proofs guarantee that the server's computation are honest, and they also guarantee that the isogeny is parallel (i.e. it reuses the same secret key) to a previously-committed one. However, these proofs are very computationally demanding: they often require computing several hundreds long isogenies by both parties. For instance, the verifiability proof used in [\[4\]](#), which improves on the proofs used in [\[10\]](#), requires the server to compute 1314 isogenies of large degree and the client 438 such isogenies. Our construction, however, achieves verifiability *for free*: the OPRF of [Protocol 12](#) is already verifiable. To see why, consider that a malicious server has three avenues to cheat:

1. It can send a maliciously generated response $(E_{mbS}, P_{mbS}, Q_{mbS})$ to the client, where the curve E_{mbS} is different than the honestly-generated one. However, in this case, the client's computation of Φ fails. For the server to succeed in fooling the the client, the curve E_{mbS} needs to be $q(2^a - q)$ -isogenous to E_S (remember that q is secret), and the points P_{mbS}, Q_{mbS} need to be the images of P_S, Q_S under the $q(2^a - q)$ -isogeny, after scaling by the secret values α_2, α'_2 , which are only known to the client. This rules out attacks based on computing a $q(2^a - q)$ -isogeny starting from E_S , as the probability of Φ

being computable is the same as guessing the two large ephemeral scalars (α_2, α'_2) .

The server could also attempt to generate an honest response E_{mBS} and guess part of the isogeny that the client recomputes to backtrack it. However, q was chosen such that $2^a - q$ is also prime: hence, the isogeny computed by the client has no small factors, and the server cannot guess any factor isogeny with non-negligible probability.

Note that this also rules out those attacks where the server act ‘honestly’, but use a different secret key instead. This approach, however, also fails: if the isogenies ϕ_S and ϕ'_S are not parallel, the curves E_S and E_{mbS} are not $q(2^a - q)$ -isogenous and thus the computation of Φ fails. In some sense, our OPRF replaces the expensive proofs of parallel isogeny with an implicit proof: since the client’s isogeny is recomputed from its domain and codomain (and the its action on the 2^a -torsion), it provides a proof that the server’s isogeny ϕ'_S is parallel to the isogeny ϕ_S computed during key generation.

2. The server could generate an honestly computed curve E_{mbS} , but manipulate the points P_{mbS}, Q_{mbS} . However, as argued in [Section 5.3](#), checking that P_{mbS}, Q_{mbS} are the correct images of P_S, Q_S , as done in Line 24 of [Protocol 12](#) is guaranteed to detect malicious points. We note that, even if the server were able to use malicious points for an adaptive attack, the server would only learn some partial information on the ephemeral values α_2 and α'_2 . Unless such information allowed the server to recompute the entire values after only iteration, such an attack would not be useful in extracting information on the message m .
3. Lastly, the server could attempt to extract some information on the message by providing maliciously generated challenges during the verification phase. However, the analysis of the previous cases also applies here, and together with the analysis of [Section 5.3](#), it shows that if the server maliciously generates some of the challenges, the client would detect it and abort.

Hence, the proposed OPRF achieves verifiability without requiring expensive zero-knowledge proofs or a high number ($>\lambda$) of long isogenies computations. To the best of our knowledge, this is the first isogeny-based OPRF to do so.

6.3 Security

We first prove pseudorandomness. To do so, we need to introduce the following game and assumption, based [\[4, Problem 4\]](#).

Game 14 (One-more unpredictability). Let E_0 denote the supersingular curve with j -invariant 1728. The game takes place between a challenger and an adversary \mathcal{A} , and it proceeds as follows:

1. The challenger generates $(p, k) = \text{Setup}()$.
2. The challenger generates a secret/public key pair $(S', \text{pk}_S) = \text{KeyGen}()$.
3. The challenger deterministically generates $R_0 \in E_0[S]$.
4. The challenger computes $\phi_S : E_0 \rightarrow E_S$ with kernel $\langle [S/S']R_0 \rangle$

5. The adversary is given access to the oracle $\text{QUERY}(\phi)$, which acts as follow:
 - (a) The oracle checks that the domain of ϕ is E_0 . If not, return \perp .
 - (b) The oracle checks that $\deg \phi < 2^{2a+\lambda/k}$. If not, return \perp .
 - (c) The oracle returns the pushforward $(\phi_S)_*\phi$
6. The adversary is also given access to the oracle $\text{CHALL}()$, which returns a random isogeny ϕ from E_0 of degree q such that $2^a - q$ is also prime.
7. The adversary outputs a list of pairs (ϕ, E_ϕ) .

We call a pair (ϕ_i, E_i) *correct* if ϕ_i is the output of a CHALL query, and E_i is the output of $\text{QUERY}(\phi_i)$. The adversary wins the game if the number of correct pairs it produces is greater than the number of oracle queries.

Assumption 15 (One-more unpredictability). Every PPT adversary wins [Game 14](#) with probability negligible in λ .

This game is a translation of [4, Problem 4] from the M-SIDH setting to the setting of FESTA isogenies, with one notable exception: the oracle, rather than return just the curve $E_0/\langle [S/S']R_0, \ker \phi \rangle$, returns the entire isogeny pushforward of the isogeny ϕ_m under the isogeny ϕ_S . However, this change does not fundamentally affect the hardness of the game: if the isogeny ϕ_m had smooth degree, there is a subexponential reduction between the two cases (the reduction corresponds to the second attack presented in [7]). Moreover, the attacks proposed in [7] do not apply here for two reasons: the isogeny ϕ_m is irrational, which prevents the attack because it would require the attacker to work with points that are defined over an exponentially large field; more fundamentally, each input m corresponds to a different degree of ϕ_m . This means that, even if the attacker were able to generate a torsion basis on E_S of a certain order, this would only help them evaluate one message isogeny. For this reason, we believe that the hardness of [Game 14](#) is significantly greater than the hardness of [4, Problem 4]. However, the two games also differ in the choice of starting curve E_0 : in [Game 14](#), E_0 has known endomorphism ring. This, however, does not seem to give a significant advantage to the adversary. Note that the recent attack on pSIDH [18], which require knowledge of the endomorphism ring, does not apply here since the isogeny ϕ_S has unknown degree.

Theorem 16. *The OPFR protocol ([Protocol 12](#)) satisfies the POPFR property ([Definition 20](#)), with $\mathcal{H}_m, \mathcal{H}_{\text{chall}}, \mathcal{H}_{\text{out}}$ modeled as random oracles, if*

- *The one-more unpredictability assumption ([Assumption 15](#)) holds,*
- *The uniSIDH validation assumption ([Assumption 11](#)) holds,*
- *$\Pi = (\text{KeyGen}^{\text{sym}}, \text{Enc}^{\text{sym}}, \text{Dec}^{\text{sym}})$ is IND-CPA secure.*

Due to space restrictions, we report the proof of [Theorem 16](#) in [Appendix D](#).

Client’s privacy. We now formally prove the OPRF has client’s privacy when the server is honest. To do so, we rely on the following game.

Game 17 (Decisional Multi-Instance Scaled Torsion Game). The game takes place between a challenger and an adversary \mathcal{A} , and it proceeds as follows:

1. The challenger generates $(p, k) = \text{Setup}()$.
2. The challenger samples a random bit $b \leftarrow \{0, 1\}$.
3. The adversary commits to a curve E .
4. The challenger chooses a basis P, Q of $E[2^a]$.
5. The adversary is given access to the oracle $\mathcal{O}()$, which:
 - (a) Samples two primes q_0, q_1 such that $2^a - q_0$ and $2^a - q_1$ are also prime.
 - (b) Samples two random isogenies $\phi_i : E \rightarrow E_i$ of order q_i , for $i \in \{0, 1\}$.
 - (c) Samples two random isogenies $\psi_i : E_i \rightarrow E_i^*$ of order $2^a - q_i$, for $i \in \{0, 1\}$.
 - (d) Computes $P_0^*, Q_0^* = [\beta_0]\psi_0\phi_0(P), [\beta'_0]\psi_0\phi_0(Q)$, for $\beta_0, \beta'_0 \leftarrow_{\$} (\mathbb{Z}_{2^a}^*)^2$.
 - (e) Computes $P_1^*, Q_1^* = [\beta_1]\psi_1\phi_1(P), [\beta'_1]\psi_1\phi_1(Q)$, for $\beta_1, \beta'_1 \leftarrow_{\$} (\mathbb{Z}_{2^a}^*)^2$.
 - (f) Deterministically generates points R_i of order S on E_i .
 - (g) Computes $R_0^*, R_1^* = [\alpha_0]\psi_0(R_0), [\alpha_1]\psi_1(R_1)$, for $\alpha_0, \alpha_1 \leftarrow_{\$} (\mathbb{Z}_S^*)^2$.
 - (h) Returns $((\phi_0, \phi_1), (E_b^*, P_b^*, Q_b^*, R_b^*), (E_{1-b}^*, P_{1-b}^*, Q_{1-b}^*, R_{1-b}^*))$.
6. The adversary outputs a bit \tilde{b} .

The adversary wins the game if $b = \tilde{b}$.

Assumption 18 (Decisional Scaled Torsion). Every PPT adversary wins [Game 17](#) with probability negligible in λ .

Theorem 19. *The OPFR protocol ([Protocol 12](#)) satisfies the $\text{OPRIV1}_{\text{OPRF}, \text{RO}}^{A, b}$ property ([Definition 21](#)), with $\mathcal{H}_0, \mathcal{H}_1, \mathcal{H}_{\text{out}}$ modeled as random oracles, if the Decisional Scaled-Torsion assumption ([Assumption 18](#)) holds.*

Proof. Assume that there exists an adversary \mathcal{A} that wins the $\text{OPRIV1}_{\text{OPRF}, \text{RO}}^{A, b}$ game with non-negligible advantage. We show we can use \mathcal{A} to build an adversary \mathcal{B} that wins [Game 17](#) with the same advantage. The adversary \mathcal{B} runs \mathcal{A} as a subroutine, and it simulates the random oracle for \mathcal{A} . \mathcal{B} starts by generating public parameters and an OPRF server's secret and public key sk, pk and passes everything to the adversary \mathcal{A} . Whenever \mathcal{A} queries the RUN oracle on m_0, m_1 in the $\text{OPRIV1}_{\text{OPRF}, \text{RO}}^{A, b}$ game, \mathcal{B} queries the \mathcal{O} oracle in [Game 17](#) to obtain $((\phi_0, \phi_1), (E_0^*, P_0^*, Q_0^*, R_0^*), (E_1^*, P_1^*, Q_1^*, R_1^*))$. It then programs the random oracle such that $\mathcal{H}_m(m_i) = \phi_i$. These values are exactly two pairs of blinded and unblinded OPRF requests: since \mathcal{B} knows the secret sk , for both requests it can compute both the response rsp_i to blinded requests and the response to the unblinded request, which is the same as the output of $\text{OPRF.Finalize}(\text{rsp}_i)$. \mathcal{B} then sends these values to \mathcal{A} : if \mathcal{A} wins the $\text{OPRIV1}_{\text{OPRF}, \text{RO}}^{A, b}$ game with non-negligible advantage, the same output b wins [Game 10](#) with the same advantage.

To show the OPRF has client's privacy when the server is malicious, we need to show that whenever the server deviates from honest computations, the client can detect it and abort. While the analysis of [Section 6.2](#) provides some justification, we leave formalizing the necessary assumptions and proving the corresponding theorem as future work.

6.4 Results

For $\lambda = 128$, we choose $a = 136$. Since we choose q to be a random prime of a bits such that $2^a - q$ is also prime, choosing a slightly larger than λ guarantees enough entropy in the choice of q . We select S to be the product of $t = \lambda$ consecutive primes, starting from $p_1 = 929$. This corresponds to a repetition parameter $k = 12$ and a prime p of about 1541 bits. We remark that changing p_1 (and consequently k) introduces a trade-off between the computation time of the client and the computation time of the server: the larger p_1 is, the longer the server takes to compute its secret isogenies, but larger p_1 's also mean smaller k 's (hence fewer repetitions), which speeds up the client's computation. However, larger p_1 also increases the size of the underlying prime, slowing down all computations and increasing the communication cost of the protocol. The optimal trade-off thus depends on the specific use case and depends on an accurate timing of all the operations involved in the protocol. The server's public key and the client's request are a single curve and points on the curve, which can be efficiently represented (without the more expensive compression) in $6 \log p$ bits, or 1156 bytes with the proposed parameters. The server's response includes $2k$ curves and points for the verification and one curve and point for the response. In total, this amounts to 28.9 kB.

We also developed a proof-of-concept implementation of all the building blocks⁷ of the OPRF protocol in SageMath [56], relying on the implementation of two-dimensional isogenies proposed in [22]. The implementation, for simplicity, uses a rational point X for verification, does not implement the symmetric-key encryption protocol, but also avoids several optimizations. The timings should thus be interpreted as an upper bound, and we expect that a more optimized implementation would be significantly faster. In particular, we expect that switching from SageMath to a low-level implementation would provide a speed-up of more than an order of magnitude. With the parameters proposed above, the implementation, when running on an Apple M3 CPU, takes 5.3 seconds for the server's key generation, 17.7 seconds for the client's request, and 130.7 seconds for the server's response. However, the response can be heavily parallelized: with 24 cores, the running time decreases to 10.3 seconds. The client's final computation takes 10.9 seconds.

Comparison with previous constructions. The more direct comparison is with the OPRF by Basso [4], since the two constructions share a similar high-level design: our OPRF is significantly more compact (about $278\times$ smaller in bandwidth) and vastly more efficient. While the author does not provide a detailed analysis of the performance of the OPRF, the protocol requires computing more than 1700 large-degree isogenies where $\log p \approx 8868$: our construction, conversely, requires computing only 40 isogenies of smaller degree and with a prime characteristic that is $5.7\times$ smaller. Moreover, our construction improves on

⁷ In the implementation, the client's request does not randomize the blinding isogeny as discussed in Remark 13; the results may thus slightly underestimate the request time.

the construction by Basso by not needing a trusted setup. Other isogeny-based OPRFs have been proposed based on CSIDH in [34], which fixed and improved an older construction [10, §8], and in [55]. The first construction [34], called OPUS, is not round-optimal nor verifiable, and it requires computing $\approx 3\lambda$ isogenies, and thus is expected to have a higher computational cost. The communication cost is also higher, as the client communicates 66 kB and the server sends 131.6 kB of data. The second construction [55] is round-optimal and verifiable, but it requires computing $\approx 8\lambda$ isogenies and communicating ≈ 68 kB of data (when instantiated with $p = 2048$; with $p = 4096$ the bandwidth becomes ≈ 135 kB). Thus, our construction significantly outperforms all isogeny-based OPRFs proposed so far, both in terms of communication and computational costs.

If we extend our comparison to non-isogeny-based protocols, the only post-quantum OPRFs reported in the literature that are secure against malicious clients are due to Albrecht, Davidson, Deo, and Smart [2], Albrecht, Davidson, Deo, and Gardham [1], and Faller, Ottenhues, and Ottenhues [29]. The first construction [2] is possibly the first proposed post-quantum OPRF, but it is highly inefficient (it requires >128 GB of communication). The second construction is based on a homomorphic-encryption evaluation of the Crypto Dark Matter PRF, and it is significantly more efficient, especially when batched. The most expensive computation, the server’s blind evaluation, requires 7 seconds, which reduces to 151 ms when run on a 64-core machine. However, this may (or may not, see [1, Sec. 1.3 and 6.1]) underestimate the real cost by an order of magnitude. This uncertainty, together with the lack of an optimized implementation of our OPRF, makes a direct performance comparison hard. However, the communication cost (>41 MB, for both verifiable and non-verifiable variants) is still significantly higher than our construction. Lastly, the third construction [29] is based on a garbled-circuit evaluation of an AES circuit. It is not round-optimal nor verifiable, but achieves running times of around 47 ms. However, the communication cost is still much higher than our construction, at 4.7 MB.

7 Conclusion

We introduced a framework to work with isogenies represented by higher-dimensional isogenies, and we used this framework to propose three new protocols. The first is a PKE that outperforms all previous isogeny-based PKEs in terms of computational cost, while also being one of the most compact post-quantum PKEs. The second is a split KEM that could be used to instantiate a post-quantum version of the Signal X3DH protocol; its adaptive security is based on a novel technique to validate public keys. The third construction is a round-optimal verifiable OPRF, based on the split KEM and its validation method. The OPRF is significantly more efficient than all previous isogeny-based OPRFs, and competitive in terms of running time with other post-quantum OPRFs, while being significantly more compact. This may provide one of the first practical post-quantum OPRFs, at least for applications that tolerate longer running times.

However, these are but just three applications of the POKE framework. Being able to manipulate irrational isogenies is an important tool that opens up new possibilities, and we expect that the POKE framework will be used to develop new cryptographic protocols and applications. In future work, we aim to explore more of this applications, as well as develop optimized implementations of the proposed constructions to better understand and compare their performance.

References

1. Albrecht, M.R., Davidson, A., Deo, A., Gardham, D.: Crypto dark matter on the torus: Oblivious PRFs from shallow PRFs and FHE. Cryptology ePrint Archive, Report 2023/232 (2023), <https://eprint.iacr.org/2023/232> 28, 33, 36
2. Albrecht, M.R., Davidson, A., Deo, A., Smart, N.P.: Round-optimal verifiable oblivious pseudorandom functions from ideal lattices. In: Garay, J. (ed.) PKC 2021, Part II. LNCS, vol. 12711, pp. 261–289. Springer, Heidelberg (May 2021). https://doi.org/10.1007/978-3-030-75248-4_10 28
3. Azarderakhsh, R., Jao, D., Leonardi, C.: Post-quantum static-static key agreement using multiple protocol instances. In: Adams, C., Camenisch, J. (eds.) SAC 2017. LNCS, vol. 10719, pp. 45–63. Springer, Heidelberg (Aug 2017). https://doi.org/10.1007/978-3-319-72565-9_3 16
4. Basso, A.: A post-quantum round-optimal oblivious PRF from isogenies. Cryptology ePrint Archive, Report 2023/225 (2023), <https://eprint.iacr.org/2023/225> 19, 20, 23, 24, 25, 27
5. Basso, A., Codogni, G., Connolly, D., De Feo, L., Fouotsa, T.B., Lido, G.M., Morrison, T., Panny, L., Patranabis, S., Wesolowski, B.: Supersingular curves you can trust. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part II. LNCS, vol. 14005, pp. 405–437. Springer, Heidelberg (Apr 2023). https://doi.org/10.1007/978-3-031-30617-4_14 36
6. Basso, A., Fouotsa, T.B.: New SIDH countermeasures for a more efficient key exchange. In: Guo, J., Steinfeld, R. (eds.) Advances in Cryptology – ASIACRYPT 2023. pp. 208–233. Springer Nature Singapore, Singapore (2023) 2, 3, 4, 5, 11, 13, 15, 36
7. Basso, A., Kutas, P., Merz, S.P., Petit, C., Sanso, A.: Cryptanalysis of an oblivious PRF from supersingular isogenies. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part I. LNCS, vol. 13090, pp. 160–184. Springer, Heidelberg (Dec 2021). https://doi.org/10.1007/978-3-030-92062-3_6 14, 25
8. Basso, A., Maino, L., Pope, G.: Festa: Fast encryption from supersingular torsion attacks. In: Guo, J., Steinfeld, R. (eds.) Advances in Cryptology – ASIACRYPT 2023. pp. 98–126. Springer Nature Singapore, Singapore (2023) 2, 4, 7, 11, 13, 36
9. Bellare, M., Palacio, A.: The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 273–289. Springer, Heidelberg (Aug 2004). https://doi.org/10.1007/978-3-540-28628-8_17 19
10. Boneh, D., Kogan, D., Woo, K.: Oblivious pseudorandom functions from isogenies. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 520–550. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-64834-3_18 19, 20, 23, 28

11. Bonnetain, X., Schrottenloher, A.: Quantum security analysis of CSIDH. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 493–522. Springer, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45724-2_17 12
12. Brendel, J., Fischlin, M., Günther, F., Janson, C., Stebila, D.: Towards post-quantum security for Signal’s X3DH handshake. In: Dunkelman, O., Jr., M.J.J., O’Flynn, C. (eds.) SAC 2020. LNCS, vol. 12804, pp. 404–430. Springer, Heidelberg (Oct 2020). https://doi.org/10.1007/978-3-030-81652-0_16 3, 4, 13
13. Campos, F., Chavez-Saab, J., Chi-Domínguez, J.J., Meyer, M., Reijnders, K., Rodríguez-Henríquez, F., Schwabe, P., Wiggers, T.: Optimizations and practicality of high-security csidh. Cryptology ePrint Archive, Paper 2023/793 (2023), <https://eprint.iacr.org/2023/793>, <https://eprint.iacr.org/2023/793> 12
14. Castryck, W., Decru, T.: An efficient key recovery attack on SIDH. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 423–447. Springer, Heidelberg (Apr 2023). https://doi.org/10.1007/978-3-031-30589-4_15 1, 10, 11, 14, 36
15. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: An efficient post-quantum commutative group action. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part III. LNCS, vol. 11274, pp. 395–427. Springer, Heidelberg (Dec 2018). https://doi.org/10.1007/978-3-030-03332-3_15 12, 35
16. Castryck, W., Vercauteren, F.: A polynomial time attack on instances of M-SIDH and FESTA. In: Guo, J., Steinfeld, R. (eds.) Advances in Cryptology – ASIACRYPT 2023. pp. 127–156. Springer Nature Singapore, Singapore (2023) 7, 20, 35, 36
17. Chávez-Saab, J., Chi-Domínguez, J.J., Jaques, S., Rodríguez-Henríquez, F.: The sqale of csidh: sublinear vélu quantum-resistant isogeny action with low exponents. *Journal of Cryptographic Engineering* **12**(3), 349–368 (Sep 2022). <https://doi.org/10.1007/s13389-021-00271-w>, <https://doi.org/10.1007/s13389-021-00271-w> 12
18. Chen, M., Imran, M., Ivanyos, G., Kutas, P., Leroux, A., Petit, C.: Hidden stabilizers, the isogeny to endomorphism ring problem and the cryptanalysis of psidh. In: Guo, J., Steinfeld, R. (eds.) Advances in Cryptology – ASIACRYPT 2023. pp. 99–130. Springer Nature Singapore, Singapore (2023) 25, 35
19. Chen, M., Leroux, A.: Scallop-hd: group action from 2-dimensional isogenies. Cryptology ePrint Archive, Paper 2023/1488 (2023), <https://eprint.iacr.org/2023/1488>, <https://eprint.iacr.org/2023/1488> 2
20. Costello, C.: B-SIDH: Supersingular isogeny Diffie-Hellman using twisted torsion. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 440–463. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-64834-3_15 12
21. Dartois, P., Leroux, A., Robert, D., Wesolowski, B.: Sqisignhd: New dimensions in cryptography. Cryptology ePrint Archive, Paper 2023/436 (2023), <https://eprint.iacr.org/2023/436>, <https://eprint.iacr.org/2023/436> 2
22. Dartois, P., Maino, L., Pope, G., Robert, D.: An algorithmic approach to (2, 2)-isogenies in the theta model and applications to isogeny-based cryptography. Cryptology ePrint Archive, Paper 2023/1747 (2023), <https://eprint.iacr.org/2023/1747>, <https://eprint.iacr.org/2023/1747> 2, 12, 13, 27
23. De Feo, L.: Mathematics of isogeny based cryptography (2017) 4
24. De Feo, L., Dobson, S., Galbraith, S.D., Zobernig, L.: SIDH proof of knowledge. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part II. LNCS, vol. 13792, pp. 310–339. Springer, Heidelberg (Dec 2022). https://doi.org/10.1007/978-3-031-22966-4_11 17

25. De Feo, L., Fouotsa, T.B., Kutas, P., Leroux, A., Merz, S.P., Panny, L., Wesolowski, B.: SCALLOP: Scaling the CSI-FiSh. In: Boldyreva, A., Kolesnikov, V. (eds.) PKC 2023, Part I. LNCS, vol. 13940, pp. 345–375. Springer, Heidelberg (May 2023). https://doi.org/10.1007/978-3-031-31368-4_13 2
26. De Feo, L., Jao, D., Plût, J.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. *Journal of Mathematical Cryptology* **8**(3), 209–247 (2014). <https://doi.org/doi:10.1515/jmc-2012-0015>, <https://doi.org/10.1515/jmc-2012-0015> 1
27. De Feo, L., Kohel, D., Leroux, A., Petit, C., Wesolowski, B.: SQISign: Compact post-quantum signatures from quaternions and isogenies. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part I. LNCS, vol. 12491, pp. 64–93. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-64837-4_3 2, 23
28. De Feo, L., Leroux, A., Longa, P., Wesolowski, B.: New algorithms for the deuring correspondence - towards practical and secure SQISign signatures. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 659–690. Springer, Heidelberg (Apr 2023). https://doi.org/10.1007/978-3-031-30589-4_23 23
29. Faller, S., Ottenhues, A., Ottenhues, J.: Composable oblivious pseudo-random functions via garbled circuits. *Cryptology ePrint Archive*, Paper 2023/1176 (2023). https://doi.org/https://doi.org/10.1007/978-3-031-44469-2_13, <https://eprint.iacr.org/2023/1176>, <https://eprint.iacr.org/2023/1176> 28
30. Feo, L.D., Fouotsa, T.B., Panny, L.: Isogeny problems with level structure. *Cryptology ePrint Archive*, Paper 2024/459 (2024), <https://eprint.iacr.org/2024/459>, <https://eprint.iacr.org/2024/459> 6, 35
31. Flynn, E.V., Ti, Y.B.: Genus two isogeny cryptography. In: Ding, J., Steinwandt, R. (eds.) Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019. pp. 286–306. Springer, Heidelberg (2019). https://doi.org/10.1007/978-3-030-25510-7_16 1
32. Fouotsa, T.B., Moriya, T., Petit, C.: M-SIDH and MD-SIDH: Countering SIDH attacks by masking information. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 282–309. Springer, Heidelberg (Apr 2023). https://doi.org/10.1007/978-3-031-30589-4_10 2, 5, 10, 11, 35
33. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M.J. (ed.) CRYPTO'99. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (Aug 1999). https://doi.org/10.1007/3-540-48405-1_34 10
34. Heimerger, L., Hennerbichler, T., Meisingseth, F., Ramacher, S., Rechberger, C.: Oprfs from isogenies: Designs and analysis. *Cryptology ePrint Archive*, Paper 2023/639 (2023), <https://eprint.iacr.org/2023/639>, <https://eprint.iacr.org/2023/639> 28
35. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the Fujisaki-Okamoto transformation. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 341–371. Springer, Heidelberg (Nov 2017). https://doi.org/10.1007/978-3-319-70500-2_12 10
36. Jao, D., De Feo, L.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In: Yang, B.Y. (ed.) Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011. pp. 19–34. Springer, Heidelberg (Nov / Dec 2011). https://doi.org/10.1007/978-3-642-25405-5_2 1, 4
37. Kiltz, E., Malone-Lee, J.: A general construction of ind-cca2 secure public key encryption. In: Paterson, K.G. (ed.) *Cryptography and Coding*. pp. 152–166. Springer Berlin Heidelberg, Berlin, Heidelberg (2003) 10

38. Kohel, D., Lauter, K., Petit, C., Tignol, J.P.: On the quaternion ℓ -isogeny path problem. Cryptology ePrint Archive, Report 2014/505 (2014), <https://eprint.iacr.org/2014/505> 7, 8
39. Kunzweiler, S., Ti, Y.B., Weitkämper, C.: Secret keys in genus-2 SIDH. In: AlTawy, R., Hülsing, A. (eds.) SAC 2021. LNCS, vol. 13203, pp. 483–507. Springer, Heidelberg (Sep / Oct 2022). https://doi.org/10.1007/978-3-030-99277-4_23 1
40. Kutas, P., Petit, C.: Torsion point attacks on “SIDH-like” cryptosystems. Cryptology ePrint Archive, Report 2022/654 (2022), <https://eprint.iacr.org/2022/654> 36
41. LeGrow, J.T., Ti, Y.B., Zobernig, L.: Supersingular non-superspecial abelian surfaces in cryptography. Cryptology ePrint Archive, Report 2022/650 (2022), <https://eprint.iacr.org/2022/650> 1
42. Maino, L., Martindale, C., Panny, L., Pope, G., Wesolowski, B.: A direct key recovery attack on SIDH. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 448–471. Springer, Heidelberg (Apr 2023). https://doi.org/10.1007/978-3-031-30589-4_16 1, 10, 11, 14, 36
43. Marlinspike, M., Perrin, T.: The X3DH key agreement protocol (2016), <https://www.signal.org/docs/specifications/x3dh/x3dh.pdf> 3, 13
44. Mokrani, Y., Jao, D.: Zero-knowledge proofs for sidh variants with masked degree or torsion. In: Regazzoni, F., Mazumdar, B., Parameswaran, S. (eds.) Security, Privacy, and Applied Cryptography Engineering. pp. 48–65. Springer Nature Switzerland, Cham (2024) 17
45. Moriya, T.: Is-cube: An isogeny-based compact kem using a boxed sidh diagram. Cryptology ePrint Archive, Paper 2023/1506 (2023), <https://eprint.iacr.org/2023/1506>, <https://eprint.iacr.org/2023/1506> 11, 36
46. Moriya, T., Onuki, H.: The wrong use of festa trapdoor functions leads to an adaptive attack. Cryptology ePrint Archive, Paper 2023/1092 (2023), <https://eprint.iacr.org/2023/1092>, <https://eprint.iacr.org/2023/1092> 3, 15
47. Moriya, T., Onuki, H., Takagi, T.: SiGamal: A supersingular isogeny-based PKE and its application to a PRF. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 551–580. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-64834-3_19 3, 7, 10
48. Nakagawa, K., Onuki, H.: QFESTA: Efficient algorithms and parameters for FESTA using quaternion algebras. Cryptology ePrint Archive, Paper 2023/1468 (2023), <https://eprint.iacr.org/2023/1468>, <https://eprint.iacr.org/2023/1468> 2, 7, 11, 13, 36
49. Oudompheng, R., Pope, G.: A note on reimplementing the castryck-decru attack and lessons learned for SageMath. Cryptology ePrint Archive, Report 2022/1283 (2022), <https://eprint.iacr.org/2022/1283> 2
50. Peikert, C.: He gives C-sieves on the CSIDH. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 463–492. Springer, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45724-2_16 12
51. Pereira, G., Doliskani, J., Jao, D.: x-only point addition formula and faster compressed sike. Journal of Cryptographic Engineering **11**(1), 57–69 (Apr 2021). <https://doi.org/10.1007/s13389-020-00245-4>, <https://doi.org/10.1007/s13389-020-00245-4> 7, 20
52. Petit, C.: Faster algorithms for isogeny problems using torsion point images. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part II. LNCS, vol. 10625, pp. 330–353. Springer, Heidelberg (Dec 2017). https://doi.org/10.1007/978-3-319-70697-9_12 35, 36

53. de Quehen, V., Kutas, P., Leonardi, C., Martindale, C., Panny, L., Petit, C., Stange, K.E.: Improved torsion-point attacks on SIDH variants. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part III. LNCS, vol. 12827, pp. 432–470. Springer, Heidelberg, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84252-9_15 35, 36
54. Robert, D.: Breaking SIDH in polynomial time. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 472–503. Springer, Heidelberg (Apr 2023). https://doi.org/10.1007/978-3-031-30589-4_17 1, 10, 11, 14, 36
55. de Saint Guilhem, C.D., Pedersen, R.: New proof systems and an oprf from csidh. Cryptology ePrint Archive, Paper 2023/1614 (2023), <https://eprint.iacr.org/2023/1614>, https://eprint.iacr.org/2023/1614_28
56. The Sage Developers: SageMath, the Sage Mathematics Software System (Version 10.1) (2024), <https://www.sagemath.org> 12, 27
57. Tyagi, N., Celi, S., Ristenpart, T., Sullivan, N., Tessaro, S., Wood, C.A.: A fast and simple partially oblivious PRF, with applications. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part II. LNCS, vol. 13276, pp. 674–705. Springer, Heidelberg (May / Jun 2022). https://doi.org/10.1007/978-3-031-07085-3_23 33

A Security definitions of OPRFs

In this work, we rely on the game-based security definition of an OPRF proposed in [57]. The definition is based on two games: $\text{OPFR}_{\text{OPRF}, S, \text{RO}}^{A,b}$ and $\text{OPRIV}_{\text{OPRF}, \text{RO}}^{A,b}$. The first game guarantees the pseudorandomness of the OPRF, which, informally, guarantees that a malicious client can obtain OPRF evaluations only by interacting with the server: after n interactions with the server, the client cannot produce $n + 1$ valid pairs of OPRF inputs and outputs. On the other hand, the $\text{OPRIV}_{\text{OPRF}, \text{RO}}^{A,b}$ game guarantees the privacy of the client: a honest-but-curious server cannot link the client’s requests to the corresponding messages. We refer to [57,1] for a detailed description of the security games.

Definition 20 (Pseudorandomness (OPFR) [57,1]). *An oblivious pseudorandom function OPRF is pseudorandom if for all PPT adversaries \mathcal{A} , there exists a simulator S such that the following advantage is negligible*

$$\left| \Pr(\text{OPFR}_{\text{OPRF}, S, \text{RO}}^{A,1}(\lambda) = 1) - \Pr(\text{OPFR}_{\text{OPRF}, S, \text{RO}}^{A,0}(\lambda) = 1) \right|.$$

Definition 21 (Client’s Privacy (OPRIV) [57,1]). *An oblivious pseudorandom function OPRF has client’s privacy if for all PPT adversaries \mathcal{A} , the following advantage is negligible*

$$\left| \Pr(\text{OPRIV}_{\text{OPRF}, \text{RO}}^{A,b}(\lambda) = 1) - \Pr(\text{OPRIV}_{\text{OPRF}, \text{RO}}^{A,b}(\lambda) = 1) \right|.$$

B Alternative Key Generation for the P(O)KE PKE

The PKE protocol, as presented in Section 4, starts from a curve E_0 whose endomorphism ring is known, since knowledge of the endomorphism ring is used

Game $\text{OPFR}_{\text{OPRF}, S, \text{RO}}^{\mathcal{A}, b}(\lambda)$	Oracle $\text{EVAL}(m)$
$q_{t,s}, q_t \leftarrow 0, 0$ $\text{pp}_1 \leftarrow \$ \text{OPRF.Setup}(\lambda)$ $\text{st}_S, \text{pk}_0, \text{pp}_0 \leftarrow \$ \text{S.Init}(\text{pp}_1)$ $(\text{pk}_1, \text{sk}) \leftarrow \$ \text{OPRF.KeyGen}^{\text{RO}}(\text{pp}_1)$ $b' \leftarrow \$ \mathcal{A}^{\text{EVAL}, \text{BLINDEVAL}, \text{PRIM}}(\text{pp}_b, \text{pk}_b)$ Return b'	$z_0 \leftarrow \text{RandomFunction}(x)$ $z_1 \leftarrow \text{OPRF.Eval}_{\text{pp}_1}^{\text{RO}}(\text{sk}, x)$ Return z_b
Oracle $\text{BLINDEVAL}(\text{req})$	Oracle $\text{LIMITEVAL}(m)$
$q_t \leftarrow q_t + 1$ $\text{rsp}_0, \text{st}_S \leftarrow \$ \text{S.BlindEval}^{\text{LIMITEVAL}}(\text{req}, \text{st}_S)$ $\text{rsp}_1 \leftarrow \$ \text{OPRF.BlindEval}_{\text{pp}_1}^{\text{RO}}(\text{sk}, \text{req})$	$q_{t,s} \leftarrow q_{t,s} + 1$ If $q_{t,s} \leq q_t$ Return $\text{EVAL}(m)$ Return \perp
	Oracle $\text{PRIM}(x)$
	$h_0, \text{st}_S \leftarrow \$ \text{S.Eval}^{\text{LIMITEVAL}}(x, \text{st}_S)$ $h_1 \leftarrow \text{RO}(x)$ Return h_b

Fig. 2: The $\text{OPFR}_{\text{OPRF}, S, \text{RO}}^{\mathcal{A}, b}$ game.

Game $\text{OPRIV1}_{\text{OPRF}, \text{RO}}^{\mathcal{A}, b}(\lambda)$	Oracle $\text{RUN}(m_0, m_1)$
$\text{pp} \leftarrow \$ \text{OPRF.Setup}(\lambda)$ $(\text{pk}, \text{sk}) \leftarrow \$ \text{OPRF.KeyGen}(\text{pp})$ $b' \leftarrow \$ \mathcal{A}^{\text{RUN}, \text{RO}}(\text{pp}, \text{pk}, \text{sk})$ Return b'	$\text{st}_0, \text{req}_0 \leftarrow \$ \text{OPRF.req}^{\text{RO}}(\text{pk}, m_0)$ $\text{st}_1, \text{req}_1 \leftarrow \$ \text{OPRF.req}^{\text{RO}}(\text{pk}, m_1)$ $\text{rsp}_0 \leftarrow \$ \text{OPRF.BlindEval}^{\text{RO}}(\text{sk}, \text{req}_0)$ $\text{rsp}_1 \leftarrow \$ \text{OPRF.BlindEval}^{\text{RO}}(\text{sk}, \text{req}_1)$ $y_0 \leftarrow \text{OPRF.Finalize}^{\text{RO}}(\text{rsp}_0; \text{st}_0)$ $y_1 \leftarrow \text{OPRF.Finalize}^{\text{RO}}(\text{rsp}_1; \text{st}_1)$ $\tau \leftarrow (\text{req}_b, \text{rsp}_b, y_0)$ $\tau' \leftarrow (\text{req}_{1-b}, \text{rsp}_{1-b}, y_1)$ Return τ, τ'

Fig. 3: The $\text{OPRIV1}_{\text{OPRF}, \text{RO}}^{\mathcal{A}, b}$ game.

to compute an isogeny of non-smooth degree during Key Generation. While E_0 having a known endomorphism ring does not affect the security of the protocol, it requires greater care in the choice of the points P_0, Q_0 . Moreover, several attacks against other isogeny-based constructions rely on the knowledge of the endomorphism ring of the starting curve [52,53,18,16]. As a defense-in-depth measure, it may be prudent to replace E_0 with a curve of unknown endomorphism ring.

To do so, Alice can start from the curve with j -invariant 1728 and compute key generation as described. After obtaining $(E_A, P_A, Q_A, R_A, S_A, X_A)$, she may compute parallel 3^b -isogenies $\psi : E_0 \rightarrow \bar{E}_0$ and $\psi : E_A \rightarrow \bar{E}_A$, similarly to what is done during encryption. She then obtains the points $\psi(P_0), \psi(Q_0)$ on E_0 and their images $\psi(P_A), \psi(Q_A)$, from which she can obtain a two-dimensional representation of the $q(2^b - q)$ -isogeny $\bar{\phi}_A : \bar{E}_0 \rightarrow \bar{E}_A$. Then, Alice (deterministically) generates a 2^a -basis of $\bar{E}_0[2^a]$, a 3^b -basis of $\bar{E}_0[3^b]$, and a point of order x . Using the representation of $\bar{\phi}_A$, she can map these points on \bar{E}_A and scales them as in the original KeyGen. The resulting public key consists of \bar{E}_0, \bar{E}_A , and the points on \bar{E}_A . Hence, with this modified key generation, Alice can generate a public key where E_0 has unknown (to anyone else) endomorphism ring; this, however, comes at the cost of a longer key generation procedure and a larger public key.

C Hardness analysis of Problem 4

In Section 4, we introduced new security assumptions. In particular, the security of Alice’s secret isogeny against key-recovery attacks relies on Problem 3, which asks to recover an isogeny ϕ of unknown degree given its action on:

- a 2^a -torsion basis, scaled by two independent scalars α_2 and α'_2 in $\mathbb{Z}_{2^a}^*$;
- a 3^b -torsion basis, scaled by a scalar α_3 in $\mathbb{Z}_{3^b}^*$;
- a point of order x , scaled by a scalar α_x in \mathbb{Z}_x^* .

At the cost of being less explicit, it is possible to simplify the formulation of the assumption. Using the same notation as Problem 3, we can sum the torsion points to obtain

$$U_0 = P_0 + R_0 + X_0, \quad V_0 = Q_0 + S_0,$$

and express the problem relative to such points.

Thus, Problem 3 is equivalent to ask to recover the isogeny ϕ given only $[\alpha]\phi(U_0)$ and $[\alpha']\phi(V_0)$, where U_0, V_0 form a basis of $E_0[2^a 3^b x]$, where $\alpha, \alpha' \in \mathbb{Z}_{2^a 3^b x}^*$ such that $\alpha \equiv \alpha' \pmod{2^a x}$ and $\alpha' \equiv 0 \pmod{x}$. These modular constraints translate the requirement that the 3^b -torsion points are masked by the same scalar and that only one point of order x is revealed.

Such a formulation shows that Problem 3 belongs to the category of security problems that asks to find a connecting isogeny between two nodes in the supersingular isogeny graph *with level structure* [30]. In the same category, we can find the security assumptions used in CSIDH [15], M-SIDH, MD-SIDH [32],

binSIDH, terSIDH, and their hybrid variants [6], FESTA [8], QFESTA [48], and IS-CUBE [45].

The level structure used in [Problem 3](#) is a combination of a diagonal level structures, such as those used by CSIDH, FESTA, and bin/terSIDH, level structures with scalar multiples of the identity, such as those used by M-SIDH and MD-SIDH, and a Borel level structures (where only one image point is revealed), such as those used by the SIDH proofs of knowledge [5]. Crucially, however, the isogeny in [Problem 3](#) relies on an isogeny of *unknown degree*. This makes the assumption more similar to the CSIDH assumption; however, in our case, the unknown degree is fundamental to security. Since the 3^b -torsion is scaled only by a single scalar, it would be easy to recover the exact torsion images and apply the SIDH attacks if the degree were known. Nonetheless, this does not seem to pose a security risk: finding the degree of a connecting isogeny is deeply linked to the decisional isogeny problem, especially in its variant where it asks whether there exists a connecting isogeny of a given degree. The problem has been studied, both as a pure problem and in the context of validating SIDH public keys, and there are no techniques reported in the literature that do not rely on computing the isogeny explicitly. More generally, there exist no known techniques to recover the degree of an isogeny between two curves, even when torsion information is given, besides trivial attacks based on pairing information: this is avoided in our case since the 3^b -torsion is masked. An attacker may hope to sidestep the issue altogether and find an attack that does not require the degree: while that is theoretically possible, it appears to be unlikely, since all existing attacks that recover an isogeny from its torsion information [52,53,40,14,42,54,16] fundamentally rely on knowledge of the degree.

Lastly, our assumption differs from those in the literature by the amount of torsion information provided: in particular, if we compare the order of the torsion bases whose images are revealed ($\approx 2^{3.5\lambda}$) with the degree of the isogeny ($\approx 2^{2\lambda}$), we see that this is significantly less than what is revealed in FESTA and QFESTA, where the order of the torsion bases is more than three times the degree of the secret isogenies.

D Proof of [Theorem 16](#)

Theorem. *The OPFR protocol ([Protocol 12](#)) satisfies the POPFR property ([Definition 20](#)), with $\mathcal{H}_m, \mathcal{H}_{\text{chall}}, \mathcal{H}_{\text{out}}$ modeled as random oracles, if*

- *The one-more unpredictability assumption ([Assumption 15](#)) holds,*
- *The uniSIDH validation assumption ([Assumption 11](#)) holds,*
- *$\Pi = (\text{KeyGen}^{\text{sym}}, \text{Enc}^{\text{sym}}, \text{Dec}^{\text{sym}})$ is IND-CPA secure.*

Proof. The high-level structure of the proof follows the proof of [1, Theorem 2], and it relies on a simulator S that behaves as an honest server, except when the random oracle is queried on a message and j -invariant pair (m, j_m) : in that case, the simulator programs the random oracle to produce the same output as the Eval function. More precisely, define the following simulator S :

$S.\text{Init}()$: sample random parameters $\text{pp}_0 = (E_0, P_0, Q_0, R_0)$, sample a random secret key $\text{st}_S = S'$ dividing S , and set $\text{pk}_0 = \perp$.
 $S.\text{BlindEval}(\text{req}, \text{st}_S)$: return $\mathcal{F}.\text{BlindEval}_{\text{pp}_0}(\text{req}, S')$, i.e. an honest evaluation of the OPRF on req with secret key S' .
 $S.\text{Eval}^{\text{LimitEval}}((\text{pk}_S, m, j_m), \text{st}_S)$: if the query (m, j_m) appears among the previous queries stored in st_S , return the same output. Otherwise, if j_m is the j -invariant of the curve associated with message m and server's key S' , i.e. $j_m = F_{S'}(m)$, query $\text{LimitEval}(m)$ and return its output h ; if $j_m \neq F_{S'}(m)$, return a uniformly random value h . In both cases, store the pair $((m, j_m), h)$ in st_S .

The adversary \mathcal{A} has access to the OPRF oracles Eval , BlindEval , and Prim . When S is defined as above, the three oracles behave identically in the cases $b = 0$ and $b = 1$, unless the simulator S obtains \perp from LimitEval . This only happens when the number of LimitEval queries exceeds the number of Eval queries, which means that the adversary has produced $n + 1$ valid pairs of OPRF inputs and outputs $(m, F_{S'}(m))$ after querying Eval only n times. We call such an event E .

We now show that such an event happens with negligible probability: the one-more unpredictability assumption guarantees that an adversary cannot generate $n + 1$ valid pairs of OPRF inputs and outputs after n queries *if* all the queries are honest. To handle the case of non-honest queries, we rely on the extractor introduced in [Assumption 11](#): when queries are not honestly generated, the simulator replace its responses with random values. More formally, we introduce the following games:

Game 0. This is the original game $\text{OPFR}_{\text{OPRF}, S, \text{RO}}^{\mathcal{A}, b}$, where the simulator S is defined as above.

Game 1. The simulator S runs the extractor $\bar{\mathcal{A}}$, as defined in [Assumption 11](#). If the extraction fails, the simulator replaces the responses rsp_i used in the validation method with randomly sampled ones.

Game 2. The simulator S runs the extractor $\bar{\mathcal{A}}$, as defined in [Assumption 11](#). If the extraction fails, the simulator also replaces the response rsp , dependent on the long-term key, with a randomly sampled one.

Let us write E_i to denote the event E in Game i . Under [Assumption 11](#), the probability $|\Pr(E_1) - \Pr(E_0)|$ is negligible: if it were not, the attacker could rely on it to distinguish between Game 0 and Game 1. However, the two games cannot be distinguished with non-negligible probability: they only differ if the extraction fails, but the extraction failing means the attacker \mathcal{A} cannot win [Game 14](#) with non-negligible probability (by [Assumption 11](#)), which in turn means the attacker cannot distinguish between Game 0 and Game 1.

Similarly, we have $|\Pr(E_2) - \Pr(E_1)|$ is negligible: if it were not, the attacker could rely on it to distinguish between Game 1 and Game 2. But the two games differ only for the value rsp to be encrypted under a random key K (since the values rsp_i have been replaced with random ones in Game 1), and thus any attacker that can distinguish between Game 1 and Game 2 with non-negligible probability can also win the IND-CPA game of Π with non-negligible probability.

Lastly, in Game 2, the simulator responds with non-random values only to honest queries. The probability $\Pr(E_2)$ is thus exactly the advantage that the attacker has in winning [Game 14](#), which is negligible by [Assumption 15](#). \square