Cryptanalysis of an Oblivious PRF from Supersingular Isogenies

Andrea Basso[1(✉)], Péter Kutas[1,6(✉)], Simon-Philipp Merz[2(✉)], Christophe Petit[1,3(✉)], and Antonio Sanso[4,5(✉)]

[1] University of Birmingham, Birmingham, UK
a.basso@cs.bham.ac.uk, p.kutas@bham.ac.uk
[2] Royal Holloway, University of London, Egham, UK
simon-philipp.merz.2018@rhul.ac.uk
[3] Université Libre de Bruxelles, Brussels, Belgium
christophe.petit@ulb.be
[4] Ethereum Foundation, Zug, Switzerland
antonio.sanso@ethereum.org
[5] Ruhr Universität Bochum, Bochum, Germany
[6] Eötvös Loránd University, Budapest, Hungary

Abstract. We cryptanalyse the SIDH-based oblivious pseudorandom function from supersingular isogenies proposed at Asiacrypt'20 by Boneh, Kogan and Woo. To this end, we give an attack on an assumption, the auxiliary one-more assumption, that was introduced by Boneh et al. and we show that this leads to an attack on the oblivious PRF itself. The attack breaks the pseudorandomness as it allows adversaries to evaluate the OPRF without further interactions with the server after some initial OPRF evaluations and some offline computations. More specifically, we first propose a polynomial-time attack. Then, we argue it is easy to change the OPRF protocol to include some countermeasures, and present a second subexponential attack that succeeds in the presence of said countermeasures. Both attacks break the security parameters suggested by Boneh et al. Furthermore, we provide a proof of concept implementation as well as some timings of our attack. Finally, we examine the generation of one of the OPRF parameters and argue that a trusted third party is needed to guarantee provable security.

1 Introduction

An oblivious pseudorandom function (OPRF) is a two-party protocol between a client and a server that computes a pseudorandom function (PRF) on a client's input with the server's key. At the end, the server does not learn anything about the client's input or the output of the function and the client learns the evaluation of the OPRF but nothing about the server's key. In particular, a client should not be able to compute the OPRF on any input without the server's participation.

Moreover, a *verifiable* oblivious pseudo random function (VOPRF) is an OPRF, where a server commits to some key and the client is ensured that the server used this key to evaluate the OPRF. In particular, the client is guaranteed that a server does not change their secret key across different executions of the protocol.

Oblivious pseudorandom functions are an important building block in many cryptographic applications. They can be used for private set intersection [23], which in turn has many applications such as private contact discovery for messaging services [14] or checking for compromised credentials [25]. Further applications of (V)OPRFs include password-authenticated key exchange [22], password-management systems [16], adaptive oblivious transfer [24], password-protected secret sharing [21] and privacy-preserving CAPTCHA systems [10].

Apart from their theoretical relevance in cryptography, OPRFs have had significant real-world impact recently. The password-authenticated key exchange OPAQUE [22] which is built on an OPRF is intended for use in TLS 1.3 [33].

The privacy-preserving authorisation mechanism known as Privacy Pass by Davidson et al. [10] is also based entirely on the security of a VOPRF. Privacy Pass is currently used at scale by Cloudflare. There is an ongoing effort to standardise OPRFs at the Crypto Forum Research Group (CFRG) [11].

Generic techniques from two-party computation and zero-knowledge proofs can be used to construct verifiable OPRFs. However, the resulting protocols might be inefficient. Therefore, all of the real-world use-cases of (V)OPRFs are currently instantiated with performant (V)OPRFs which are based on classical security assumptions. Practical constructions are currently based either on the hardness of the decisional Diffie-Hellman problem, called DH-OPRF [21], or they are derived from RSA blind signatures [8,11].

For quantum-secure OPRFs, there are only few proposals. Indeed, only three such solutions appear in the literature to date. In 2019, Albrecht et al. proposed a lattice-based VOPRF [1] based on the ring learning with errors problem and the short integer solution problem in one dimension. Seres et al. constructed an OPRF based on the shifted Legendre symbol problem [31] and Boneh et al. presented two isogeny-based (V)OPRFs at ASIACRYPT 2020 [3].

Isogeny-based cryptography is one of the branches of post-quantum cryptography that are currently being explored. The particularly small key sizes required by isogeny-based cryptosystems make them very attractive in some areas of information security. Isogeny-based cryptography was first proposed by Couveignes in 1997 [9]. However, his ideas were not published at the time and they were independently rediscovered by Rostovtsev and Stolbunov [30]. The idea of Couveignes and Rostovtsev-Stolbunov (CRS) was to build a Diffie-Hellman type key exchange using the class group of the endomorphism ring of ordinary elliptic curves. However, neither of the suggested schemes was efficient enough to be considered practical. Meanwhile, supersingular elliptic curves were first used in cryptography by Charles, Lauter and Goren [7] to build a hash function.

Jao and De Feo took a different approach to isogeny-based cryptography when they introduced the *supersingular isogeny Diffie-Hellman* (SIDH) key exchange [20]. Instead of computing class group actions as in the case of CRS, Jao and De Feo use the following observation. Two subgroups of an elliptic curve of coprime cardinality are only intersecting at the point at infinity. Independent of the order in which two such subgroups are divided out of an elliptic curve, the resulting curve will be equal up to isomorphism. The only isogeny-based cryptosystem submitted to NIST's ongoing post-quantum standardization process is the SIDH-based candidate SIKE [2,19] which has been selected as one of the alternate finalists.

Later, the idea of CRS-type schemes was resurrected, when Castryck et al. adapted it to supersingular elliptic curves and managed to eliminate most of its performance issues [5]. The resulting scheme is called CSIDH.

In their ASIACRYPT 2020 paper [3], Boneh et al. propose an augmentable commitment framework that can be used to build an OPRF and is instantiated with both an SIDH-based scheme that can be made verifiable, and with a CSIDH-based one. The SIDH-based variant relies on the hardness of the decisional supersingular isogeny problem, a standard assumption in the area, and a novel 'one-more' isogeny assumption.

Our Contributions. In this paper, we cryptanalyse the SIDH-based 'one-more' assumption introduced by Boneh, Kogan and Woo. We first give multiple variants of an attack on the assumption itself. A first variant leads to a polynomial-time attack against the proposed SIDH-based OPRF protocol. We then argue that a simple modification of the (V)OPRF protocol prevents such an attack. Then, we show that a second variant of the attack leads to an attack on the protocol even in the presence of those countermeasures. This attack has a subexponential complexity, but there appear to be no simple countermeasures. Developing countermeasures is left as an open problem. As a result of our attack, the parameters suggested by Boneh et al. fall short of their estimated security level.

The attacks on the OPRF allow malicious clients to evaluate the OPRF on arbitrary inputs after some initial queries to the server, without even interacting with the server any further. This breaks the pseudorandomness property of the OPRF and could lead to significant attacks on OPRF-based protocols. In the context of private set intersection based on oblivious PRFs, the proposed attack allows the attacker to brute-force the other party's set elements and break the privacy requirement. In the Privacy Pass protocol used to guarantee privacy-preserving CAPTCHAs, our attack allows the attacker to generate unlimited tokens, thus avoiding solving CAPTCHAs and fully breaking the security of the system.

Furthermore, we discuss how one of the parameters of the SIDH-based OPRF by Boneh et al. is generated and which party should compute it. We argue there are security implications if the server, the client or any third party maliciously generates this parameter. The client or a third party can introduce a backdoor through this parameter to recover the secret key of the server, whereas if the server is malicious, they can break the supersingular-isogeny collision assumption on which Boneh et al.'s security proofs are built. We suggest that a trusted setup may be needed to guarantee provable security.

Finally, we want to emphasise that the CSIDH-based OPRF proposal by Boneh et al. is not affected by our attacks.

Outline. In Sect. 2, we introduce some background on isogeny-based cryptography, the security properties of (verifiable) oblivious PRFs and Boneh et al.'s construction. The attacks against the 'one-more' assumption are presented in Sect. 3, while their application to the OPRF protocol by Boneh et al. is discussed in Sect. 4. We present our implementation of the attack and discuss its results in Sect. 5. In Sect. 6, we argue that a trusted setup should be used for the OPRF and briefly sketch two attacks that follow a lack of trusted setup before concluding the paper in Sect. 7.

2 Preliminaries

In this section we introduce the necessary mathematical background on isogenies and the SIDH key exchange, we summarize the security properties of OPRFs and we briefly recall Boneh et al.'s OPRF construction [3].

2.1 Mathematical Background on Isogenies

Let $\mathbb{F}_q$ be a finite field of characteristic p. In the following, we assume $p \geq 3$ and therefore an elliptic curve E over $\mathbb{F}_q$ can be defined by its short Weierstrass form

$$E(\mathbb{F}_q) = \{(x, y) \in \mathbb{F}_q^2 \mid y^2 = x^3 + Ax + B\} \cup \{\mathcal{O}_E\}$$

with $A, B \in \mathbb{F}_q$ such that the discriminant is non-zero and $\mathcal{O}_E$ denotes the point $(X : Y : Z) = (0 : 1 : 0)$ on the associated projective curve $Y^2Z = X^3 + AXZ^2 + BZ^3$. The *j-invariant* of an elliptic curve is $j(E) = 1728\frac{4A^3}{4A^3+27B^2}$.

A non-constant rational map $\phi : E_1 \to E_2$ between two elliptic curves is an *isogeny* if it sends the point at infinity of E_1 to the point at infinity of E_2. Equivalently, an isogeny is a rational map which is also a group homomorphism. Thus an isogeny is the natural morphism of the category of elliptic curves. An isogeny ϕ induces a field extension between the function fields of E_1 and E_2. The degree of this extension is the degree of the isogeny. We call an isogeny *separable* if this field extension is separable. The kernel of a separable isogeny as a group homomorphism is finite and is equal to the degree of the isogeny. If $\phi : E_1 \to E_2$ is an isogeny of degree d, then there exists a unique isogeny $\hat{\phi}$ of degree d such that $\phi \circ \hat{\phi} = [d]$, where $[d]$ denotes multiplication by d. The isogeny $\hat{\phi}$ is called the *dual isogeny* of ϕ. An isomorphism of elliptic curves is an isogeny of degree 1 and there is an isomorphism of curves $f : E_0 \to E_1$ if and only if $j(E_0) = j(E_1)$.

An *endomorphism* of E is an isogeny from E to itself. Endomorphisms of E form a ring under composition and addition denoted by $\mathrm{End}(E)$. The endomorphism ring of an elliptic curve over a finite field is either an order in an imaginary quadratic field (in which case the curve is called *ordinary*) or a maximal order in the quaternion algebra ramified at infinity and p (in which case the curve is called *supersingular*).

The j-invariant of any supersingular elliptic curve defined over $\mathbb{F}_q$ lies in $\mathbb{F}_{p^2}$.

For a thorough introduction to elliptic curves and isogeny-based cryptography, we refer to Silverman [32] and De Feo [12], respectively.

2.2 SIDH

We briefly recall the *supersingular isogeny Diffie-Hellman* key exchange introduced by Jao and De Feo [20].

Let E_0 be a supersingular elliptic curve defined over $\mathbb{F}_{p^2}$, where p is a prime of the form $f \cdot N_1 N_2 \pm 1$. Here $f \in \mathbb{Z}$ is a small cofactor and N_1, N_2 are two coprime smooth integers (e.g. a power of 2 and 3 respectively). Furthermore, fix two bases P_A, Q_A and P_B, Q_B such that $\langle P_A, Q_A \rangle = E_0[N_1]$ and $\langle P_B, Q_B \rangle = E_0[N_2]$. To agree on a secret key over an insecure channel, Alice and Bob proceed as follows:

1. Alice chooses a random cyclic subgroup of $E_0[N_1]$ generated by a point of the form $A = P_A + [x_A]Q_A$ as her secret. Similarly, Bob chooses his secret as $\langle B \rangle := \langle P_B + [x_B]Q_B \rangle \subset E_0[N_2]$.
2. Then, Alice and Bob compute their secret isogeny $\varphi_A : E_0 \to E_0/\langle A \rangle$ and $\varphi_B : E_0 \to E_0/\langle B \rangle$, respectively.
3. Alice sends the curve $E_A := E_0/\langle A \rangle$ and the points $\varphi_A(P_B), \varphi_A(Q_B)$ to Bob. Mutatis mutandis, Bob sends $E_B := E_0/\langle B \rangle, \varphi_B(P_A)$ and $\varphi_B(Q_A)$ to Alice.
4. Both Alice and Bob can compute the shared secret curve $E_{AB} := E_0/\langle A, B \rangle$ up to isomorphism as

$$E_{AB} \cong E_B/\langle \varphi_B(P_A) + [x_A]\varphi_B(Q_A) \rangle \cong E_A/\langle \varphi_A(P_B) + [x_B]\varphi_A(Q_B) \rangle.$$

Since isomorphic curves have the same j-invariant, Alice and Bob use $j(E_{AB})$ as their shared secret.

2.3 Security Properties of (V)OPRFs

In the following, we will call a function $\mu : \mathbb{N} \to \mathbb{R}$ *negligible* if for every positive polynomial $\mathsf{poly}(\cdot)$ there exists an integer $N_{\mathsf{poly}} > 0$ such that for all $x > N_{\mathsf{poly}}$, we have $|\mu(x)| < 1/\mathsf{poly}(x)$.

The security properties of an oblivious pseudorandom function (OPRF) include those of a standard pseudorandom function (PRF).

Definition 1. *Let $F : K \times X \to Y$ be an efficiently computable function. F is a* pseudorandom function *(PRF) if for all probabilistic polynomial-time distinguishers D, there is a negligible function negl such that*

$$\mathbb{P}[D^{F(k,\cdot)}(1^n) = 1] - \mathbb{P}[D^{f(\cdot)}(1^n) = 1] \leq \mathsf{negl}(n),$$

where the first probability is taken over uniform choices of $k \in \{0,1\}^n$ and the randomness of D, and the second probability is taken over uniform choices of functions $f : X \to Y$ and the randomness of D.

A consequence of pseudorandomness is that one cannot compute a new evaluation of $F(k, \cdot)$ from existing evaluations. However, our attack on Boneh et al.'s OPRF will allow adversaries to evaluate $F(k, \cdot)$ on arbitrary inputs after some initial evaluations.

Furthermore, an OPRF is oblivious in the following sense.

Definition 2 **([17]).** *Let $F : K \times X \rightarrow Y$ be a PRF. A protocol between a client with input $x \in X$ and a server with key $k \in K$ is called* oblivious *PRF, if the client learns $F(k, x)$ and nothing else and the server learns nothing about x or $F(k, x)$ at the end of the protocol.*

In particular, the server will learn nothing about the input x of the client and the client will learn nothing about the server's key k. Additionally, an OPRF can be verifiable.

Definition 3. *An OPRF is said to be* verifiable *if the evaluation y that the client obtains at the end of the protocol is correct, i.e. if it satisfies $y = F(k, x)$, where $x \in X$ is the client's input and $k \in K$ is the server's private key.*

In practice, verifiability is ensured by the server committing to a key k prior to the execution of the verifiable OPRF (VOPRF) and providing a zero-knowledge proof that the VOPRF execution uses the same key as the committed value.

2.4 An Isogeny-Based OPRF by Boneh, Kogan and Woo

We provide a simplified description of Boneh et al.'s OPRF based on the SIDH key exchange protocol.

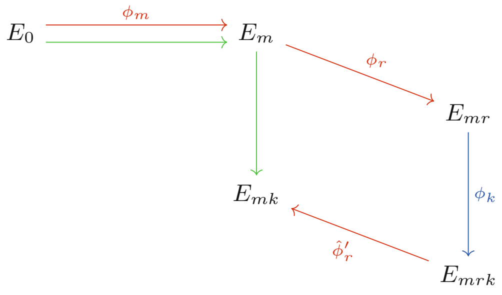
Let λ be the security parameter and let $p = fN_{\mathsf{K}}N_{\mathsf{M}}N_{\mathsf{V}}N_{\mathsf{R}}N_{\mathsf{S}} - 1$ be a prime where $f \in \mathbb{Z}$ is a small cofactor and N_i are powers of distinct small primes such that $N_{\mathsf{K}}, N_{\mathsf{M}}, N_{\mathsf{V}}, N_{\mathsf{R}}$ are roughly of size $2^{5\lambda/2}$ and $N_{\mathsf{S}} \approx 2^{2\lambda}$. To prevent an attack by Merz et al. [27], the factors $N_{\mathsf{K}}, N_{\mathsf{M}}, N_{\mathsf{V}}, N_{\mathsf{R}}$ are of size $2^{5\lambda/2}$ instead of the more common size $2^{2\lambda}$ in the SIDH setting. Moreover, let $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_{N_{\mathsf{M}}}$ be a cryptographic hash function. In their proofs, Boneh et al. treat H_1 as a random oracle. Finally, let E_0 be a randomly chosen supersingular elliptic curve over $\mathbb{F}_{p^2}$ and let $\{P_i, Q_i\}$ denote a basis of $E_0[N_i]$ for $i = K, M, V, R, S$. While Boneh et al. only require E_0 to be a randomly chosen elliptic curve, we will discuss how it is generated in Sect. 6 and argue that this choice should be done by a trusted third party.

First, the server chooses their private key k which is the PRF key and publishes a commitment to this key. To evaluate the OPRF at an input x in the input space, a client computes the hash $H_1(x) = m \in \mathbb{Z}_{N_{\mathsf{M}}}$. Furthermore, the client randomly chooses an element $r \in \mathbb{Z}_{N_{\mathsf{R}}}$.

The client computes the isogenies $\phi_m : E_0 \rightarrow E_m := E_0/\langle P_M + [m]Q_M \rangle$ and $\phi_r : E_m \rightarrow E_{mr} := E_m/\langle \phi_m(P_R) + [r]\phi_m(Q_R) \rangle$. Then, the client sends E_{mr} together with the torsion point images of P_i, Q_i for $i = V, K, S$ to the server as well as a basis of $E_{mr}[N_{\mathsf{R}}]$. To avoid active attacks like the GPST attack [18], where a malicious client tries to recover information about the server's private key by sending manipulated torsion point information, the client proves to the server in a non-interactive zero-knowledge proof that they know the kernel of the isogeny from E_0 to E_{mr} and that the provided torsion point images are indeed the images under this isogeny. For full details about the zero-knowledge proof we refer to Sect. 5 of [3].

Subsequently, the server computes their secret isogeny $\phi_k : E_{mr} \to E_{mrk}$, where $E_{mrk} := E_{mr}/\langle \phi_r \circ \phi_m(P_K) + [k]\phi_r \circ \phi_m(Q_K)\rangle$. Moreover, the server computes the images of the order N_{V} torsion points and the basis of $E_{mr}[N_{\mathsf{R}}]$ provided by the client. The server sends E_{mrk} together with the torsion point information to the client. Using an interactive zero-knowledge proof with a cut-and-choose approach between server and client, the server can prove to the client that it computed the isogeny and the torsion point images correctly. This proof uses the torsion point images of order N_{V} and the server's initial commitment to the key k. Details about this zero-knowledge proof can be found in Sect. 6 of [3].

After executing the zero-knowledge proof with the server to convince itself of the correctness of the server's reply, the client uses the images of the $E_{mr}[N_{\mathsf{R}}]$ torsion to "unblind" E_{mrk}. The unblinding isogeny $\hat{\phi}'_r$ is a translation of the dual of ϕ_r starting from E_{mrk}. This allows the client to compute a curve isomorphic to $E_{mk} := E_m/\langle \phi_m(P_K) + k\phi_m(Q_K)\rangle$ without knowing k at any point in time. Hashing the input together with the j-invariant of E_{mk} and the server's initial commitment to his key k yields the output of the VOPRF. The entire evaluation is sketched in Fig. 1.



$$\begin{tikzcd}
E_0 \arrow[r, shift left, "\phi_m"] \arrow[r, shift right] & E_m \arrow[d] \arrow[dr, "\phi_r"] & \\
 & E_{mk} & E_{mr} \arrow[d, "\phi_k"] \\
 & & E_{mrk} \arrow[ul, "\hat{\phi}'_r"]
\end{tikzcd}$$

Fig. 1. Sketch of Boneh et al.'s isogeny-based VOPRF. The isogenies computed by the client are marked in red (ϕ_m, ϕ_r, and $\hat{\phi}'_r$) while the server's isogeny is noted in blue (ϕ_k). The green isogenies represent the PRF which is jointly evaluated by the client and the server. The output of the OPRF is computed as $F(k, x) = H(x, j(E_{mk}), \mathsf{pk})$, where H is a cryptographic hash function and pk is the server's (public) commitment to his key k.

3 Attacks on the Auxiliary One-More SIDH Assumption

In [3], Boneh et al. introduce the auxiliary one-more SIDH assumption. This is a new security assumption to prove the unpredictability of their isogeny-based VOPRF. In this section we challenge the validity of this assumption and we present multiple attacks on the corresponding computational problem.

All of the attacks follow a similar strategy. First, an attacker recovers certain torsion point images up to a scalar under the secret isogeny using queries in the security game. Having recovered these torsion point images, an attacker is capable of answering any challenge set by the challenger correctly. This breaks the security assumption and also leads to an attack on Boneh et al.'s (V)OPRF.

We start by recalling the security assumption introduced by Boneh et al. [3]. Then, we show that recovering said torsion point images up to a scalar is sufficient to compute the correct answer to arbitrary challenges in the corresponding security game. Subsequently, we give multiple approaches to recover these torsion point images. In Sect. 4, we will show how the attack on the security assumption translates to an attack on the (V)OPRF itself.

3.1 The Auxiliary One-More SIDH Assumption

First, we recall the game underlying the auxiliary one-more SIDH assumption as defined by Boneh et al. [3]. While Boneh et al. use the "decision queries" defined in the following game in their security proofs, our attacks will not make use of decision queries and a reader may ignore this additional ability of an adversary.

Game 1 (Auxiliary One-More SIDH). Let $p = f \cdot N_1 \cdots N_n - 1$ be a prime depending on the security level λ and n, where N_i are smooth coprime integers and f is a small cofactor, and let $\mathsf{M}, \mathsf{K} \in \{1, \ldots, n\}$ be two distinct indices. Consider the following game between a challenger and an adversary:

- The challenger chooses a random supersingular curve $E_0/\mathbb{F}_{p^2}$ and a basis $\{P, Q\}$ of $E_0[(p+1)/(N_\mathsf{M} \cdot N_\mathsf{K})]$. Moreover, it chooses $K \in E_0$ of order N_K, computes $\phi_K : E_0 \to E_K := E_0/\langle K \rangle$, and sends E_0, P, Q, and E_K to the adversary.
- The adversary can make a sequence of queries of the following types to the challenger:
  - Challenge query: The challenger chooses $M \in E_0[N_\mathsf{M}]$ randomly and sends it to the adversary
  - Solve query: The adversary submits $V \in E_0[(p+1)/N_\mathsf{K}]$ to the challenger[1], who computes $\phi_{KV} : E_0 \to E_0/\langle K, V \rangle$ and sends $j(E_0/\langle K, V \rangle)$, $\phi_{KV}(P)$, and $\phi_{KV}(Q)$ to the adversary.
  - Decision query: The adversary submits a pair (i, j) to the challenger, where i is a positive integer bounded by the number of challenge queries made so far, and $j \in \mathbb{F}_{p^2}$. The challenger responds true if $j = j(E_0/\langle K, M \rangle)$, where M is the challenger's response to the ith challenge query, and false otherwise.
- The adversary outputs a list of distinct pairs of the form (i, j), where i is a positive integer bounded by the number of challenge queries made and $j \in \mathbb{F}_{p^2}$.

We call an output-pair (i, j) correct, if j is the j-invariant of $E_0/\langle K, M \rangle$, where M is the challenger's response to the ith challenge query. An adversary wins the game, if the number of correct pairs exceeds the number of Solve queries.

Assumption 2 (Auxiliary One-More SIDH [3]). For every constant n and every distinct $\mathsf{M}, \mathsf{K} \in \{1, \dots, n\}$, every efficient adversary wins the above game with probability negligible in λ.

In the following, we will show that the auxiliary one-more SIDH assumption does not hold. We will give different attacks on the security problem underlying Assumption 2 that follow a similar strategy. Let K be the server's secret, determining the isogeny $\phi_K : E_0 \to E_0/\langle K\rangle$. The idea is to use a number of solve queries to subsequently predict $E_0/\langle K, M\rangle$ for any $M \in E_0[N_\mathsf{M}]$. To this end, we will derive a method to extract the subgroup generated by $\phi_K(P)$ for any $P \in E_0[N_\mathsf{M}]$ with a number of solve queries. Using this procedure, an adversary can extract the subgroups generated by $\phi_K(P_M)$, $\phi_K(Q_M)$ and $\phi_K(P_M + Q_M)$, where $\{P_M, Q_M\}$ is a basis of $E_0[N_\mathsf{M}]$.

Knowing these subgroups allows the adversary to compute the subgroups generated by $\phi_K(M)$ for arbitrary $M \in E_0[N_\mathsf{M}]$ without any further solve queries. Given a generator of $\langle\phi_K(M)\rangle$, the adversary can compute the j-invariant of $E_0/\langle K, M\rangle$ as $E_0/\langle K, M\rangle \cong E_K/\langle\phi_K(M)\rangle$. In particular, the adversary can produce arbitrarily many correct output-pairs and win the security game underlying the auxiliary one-more SIDH assumption (Assumption 2).

3.2 Winning the Security Game Given Torsion Point Images

In this section, we show how mapping three different N_M-order subgroups to $E_K := E_0/\langle K\rangle$ is enough to recover sufficient information to compute a generator of $\langle\phi_K(M)\rangle \in E_K$ for any point $M \in E_0[N_\mathsf{M}]$.

Lemma 1. *Let P_V, Q_V, $R_V := P_V + Q_V \in E_0$ be pairwise linearly independent points of smooth order N_M and let $\phi_K : E_0 \to E_K$ be an unknown isogeny of degree coprime to N_M. Given the points P_V, Q_V, R_V and the subgroups $\langle\phi_K(P_V)\rangle$, $\langle\phi_K(Q_V)\rangle$ and $\langle\phi_K(R_V)\rangle$, an adversary can compute $\langle\phi_K(M)\rangle$ for arbitrary $M \in E_0[N_\mathsf{M}]$.*

Proof. Fix P', Q', and R' to be generators of $\langle\phi_K(P_V)\rangle$, $\langle\phi_K(Q_V)\rangle$ and $\langle\phi_K(R_V)\rangle$ respectively. Note that the given information $\langle\phi_K(P_V)\rangle$, $\langle\phi_K(Q_V)\rangle$ and $\langle\phi_K(R_V)\rangle$ is the same as knowing $\phi_K(P_V)$, $\phi_K(Q_V)$, $\phi_K(R_V)$ up to a scalar multiple. There are many different generators for the groups $\langle\phi_K(P_V)\rangle$, $\langle\phi_K(Q_V)\rangle$ and $\langle\phi_K(R_V)\rangle$ but for any fixed choice we have

$$\begin{aligned} P' &= \alpha\phi_K(P_V), \\ Q' &= \beta\phi_K(Q_V), \\ R' &= \gamma\phi_K(R_V) \end{aligned}$$

[1] In Algorithm 1, we will describe how an adversary can win the game in polynomial time, if the point V is not required to be of full order.

for some (unknown) integers α, β, γ coprime to N_{M}. As isogenies are homomorphisms, we have $\phi_K(R_V) = \phi_K(P_V) + \phi_K(Q_V)$. One finds a, b such that $R' = aP' + bQ'$, which can be done efficiently as computing discrete logarithms is easy in a group of smooth order N_{M}. We have $\gamma = a\alpha = b\beta$. Thus, it is possible for the attacker to recover the ratio $\alpha/\beta = b/a$.

Given any $M \in E_0[N_{\mathsf{M}}]$, an adversary can compute integers k_1, k_2 such that $M = k_1 P_V + k_2 Q_V$ (which again is possible because N_{M} is smooth) and obtain $\langle \phi_K(M) \rangle$ by computing $\langle k_1 \phi_K(P) + k_2 \phi_K(Q) \rangle = \langle k_1 P' + k_2 \frac{\alpha}{\beta} Q' \rangle$. $\square$

In particular, an adversary who knows $\phi_K(P_V)$, $\phi_K(Q_V)$ and $\phi_K(R_V)$ up to a scalar and $E_K := E_0/\langle K \rangle$ can compute $E_0/\langle K, M \rangle \cong E_K/\langle \phi_K(M) \rangle$ for any $M \in E_0[N_{\mathsf{M}}]$.

3.3 Recovering Points in $\phi_K(E_0[N_{\mathsf{M}}])$ Up to a Scalar

The previous subsection shows that $E_0/\langle K, M \rangle$ can be computed by an adversary for arbitrary $M \in E_0[N_{\mathsf{M}}]$ as long as they can recover images of points in $E_0[N_{\mathsf{M}}]$ under the secret isogeny ϕ_K up to scalar. In this section, we will present multiple ways an adversary can recover this information. For didactic purposes, we include not only a polynomial and a subexponential attack (in case countermeasures to prevent the former one are put in place) but also an exponential attack in our exposition.

Query points of arbitrary order. Let $M \in E_0[N_{\mathsf{M}}]$. We are interested in recovering $\phi_K(M)$ up to a scalar, given access to the oracle provided by the "solve queries" in Game 1. Note that our attack will not use "decision queries" as defined in the same game.

There is a simple procedure to compute an isogeny between E_K and $E_M := E_K/\langle \phi_K(M) \rangle$ and therefore $\phi_K(M)$ up to scalar, if "solve queries" are allowed for points of arbitrary order. Recall that during a solve query in Game 1, an adversary gets to submit points $V \in E_0[(p+1)/N_{\mathsf{K}}]$ to the challenger, who replies with the j-invariant of $E_0/\langle K, V \rangle$ and some additional torsion point images. Algorithm 1 describes how an adversary can recover $\phi_K(M)$ up to a scalar for arbitrary $M \in E_0[N_{\mathsf{M}}]$. The Algorithm recovers the isogeny from E_K to $E_K/\langle \phi_K(M) \rangle$ by using solve queries to obtain all intermediate curves. This allows to recover the isogeny $E_K \to E_K/\langle \phi_K(M) \rangle$ one step at a time and therefore its kernel $\langle \phi_K(M) \rangle$.

Algorithm 1: Computation of $\langle \phi_K(M) \rangle$ using solve queries on points of arbitrary order

```
Let $\{l_i\}_{i=0}^{n}$ be an integer sequence of all divisors of $N_{\mathsf{M}}$ such that $l_{i+1}/l_i$ is a
prime, $l_i < l_{i+1}$, with $l_0 := 1$, $l_n := N_{\mathsf{M}}$.
Input: $E_K$, $M \in E_0[N_{\mathsf{M}}]$ and access to an oracle answering solve queries as
       defined in Game 1.
Output: A generator of $\langle \phi_K(M) \rangle$

1 $E^{(n)} \leftarrow E_0/\langle K \rangle$ ;
2 for $i = n-1, \ldots, 0$ do
3     Query the oracle with the point $V_i := [l_i]M$ and obtain the curve
      $E^{(i)} := E_0/\langle K, V_i \rangle = E_0/\langle K, [l_i]M \rangle = E_K/\langle [l_i]\phi_K(M) \rangle$;
4     Find $l_{i+1}/l_i$-isogeny $\phi_i$ from $E^{(i+1)}$ to $E^{(i)}$;
5 return A generator of $\ker(\phi_0 \circ \cdots \circ \phi_{n-1})$;
```

Lemma 2. *Algorithm 1 returns $\lambda\phi_K(M)$, where $\lambda \in \mathbb{Z}$ is coprime to N_{M}.*

Proof. Let ψ_M be the isogeny from E_K to $E_K/\phi_K(M)$. Then the claim follows from the observation that $E_0/\langle K, [l_i]M \rangle \cong E_0/\langle [l_i]K, [l_i]M \rangle$, since l_i is coprime to the order of K. □

Remark 1. Note that an attacker can easily change the attack to require fewer queries. Instead of using one query for each intermediate curve, an attacker can choose any factorisation $f_1 \cdots f_t$ of N_{M} such that f_i are roughly of equal size and query the oracle with $\left[\prod_{j=1}^{b} f_i\right] M$ for $b = 1, \ldots, t$. Then, the attacker is left to recover the isogeny between any two consecutive queries, i.e. the isogenies of degree f_i for $i = 1, \ldots, t$, using a meet-in-the-middle attack.

In Game 1, Boneh et al. did not specify any restrictions on the points of $E_0[(p+1)/N_{\mathsf{K}}]$ that can be submitted to the solve queries. However, in the context of the game, this attack can be easily thwarted by answering to a solve query only if the submitted point is of order $(p+1)/N_{\mathsf{K}}$. This property can be checked efficiently by the challenger. In Sect. 4, we discuss how this polynomial-time attack and its countermeasures translate to the VOPRF protocol.

Query Points of Order $(p+1)/N_{\mathsf{K}}$. Next, we present how an attacker can retrieve the necessary information even if they are only allowed to send solve queries on points of order $(p+1)/N_{\mathsf{K}}$, i.e. if the challenger checks the order of a submitted point and only replies to a query if the point is of order $(p+1)/N_{\mathsf{K}}$.

Let ϕ_V denote the isogeny $E_K \rightarrow E_0/\langle V, K \rangle$ of degree $(p+1)/N_{\mathsf{K}}$ and let $\phi_V = \phi_{V'} \circ \phi_M$ be its decomposition into a degree $(p+1)/(N_{\mathsf{K}}N_{\mathsf{M}})$ and a degree N_{M} isogeny. Our attack aims to recover the image of multiple subgroups of $E_0[N_{\mathsf{M}}]$ under the isogeny ϕ_K, i.e. we are interested in the kernel of the isogeny ϕ_M for different points V. The isogenies are depicted in Fig. 2.

$$\begin{tikzcd}
E_0 \arrow[r, "\phi_K"] & E_K \arrow[r, "\phi_M"] \arrow[rr, bend left, "\phi_V"] & \bullet \arrow[r, "\phi_{V'}"] & E_0/\langle K, V\rangle
\end{tikzcd}$$

Fig. 2. Depiction of the isogenies of a solve query

Recovering $\phi_{V'}$ from torsion point information. Let $P, Q \in E_0[(p+1)/N_{\mathsf{M}}N_{\mathsf{K}}]$ be the torsion point basis provided by the challenger and let $V \in E_0[(p+1)/N_{\mathsf{K}}]$ be linearly independent of P or Q. Then, we can use the torsion point images provided during a solve queries to compute $\hat{\phi}_V'$ as follows.

Let $P' := \phi_V \circ \phi_K(P)$, $Q' := \phi_V \circ \phi_K(Q)$ be the images of the torsion points provided by the challenger. The adversary can compute $\hat{\phi}_{V'}$ as the isogeny from $E_0/\langle K, V\rangle$ with kernel $\langle P', Q'\rangle$. Note that $\langle P', Q'\rangle \subset \ker(\hat{\phi}_{V'})$, because $\hat{\phi}_{V'} \circ \phi_{V'} = [(p+1)/N_{\mathsf{M}}N_{\mathsf{K}}]$ is the order of the points P, Q. As V is linearly independent to at least one of P and Q, the other inclusion follows from $\langle P', Q'\rangle$ spanning a subgroup of size $(p+1)/N_{\mathsf{M}}N_{\mathsf{K}}$.

Choosing P_V, Q_V as a basis of $E_0[(p+1)/N_{\mathsf{K}}]$ such that $[N_{\mathsf{M}}]P_V = P + [(p+1)/N_{\mathsf{M}}N_{\mathsf{K}}]Q$ and $[N_{\mathsf{M}}]Q_V = [(p+1)/N_{\mathsf{M}}N_{\mathsf{K}}]P + Q$, every point of the form $P_V + [i]Q_V$ or $[i]P_V + Q_V$ will be linearly independent of P or Q.

As a consequence of $\phi_{V'}$ being easy to recover, we may assume that during a solve query an attacker can send a point M of order N_{M} to the challenger who returns $E_0/\langle K, M\rangle$. We are left to recover the kernel of ϕ_M.

Naïve attack to recover ϕ_M. Next we describe an exponential attack that recovers $\hat{\phi}_M$ using meet-in-the-middle (MITM) computations of increasing size. In the subsequent part, we will introduce a trade-off between queries and computation costs that reduces the complexity of the attack to subexponential.

Let P_M, Q_M denote a basis of $E_0[N_{\mathsf{M}}]$. For simplicity of exposition we treat N_{M} as a prime power and we write $N_{\mathsf{M}} = \ell_M^{e_M}$. The attack recovers $\phi_M : E_K \to E_K/\langle P_M\rangle$ by recovering each of the e_M intermediate curves one at a time.

The attacker starts by querying the solve oracle with two points $V_0 := P_M$ and $V_1 := P_M + [\ell_M^{e_M-1}]Q_M$. Note that the curves $E_K/\langle\phi_K(V_0)\rangle$ and $E_K/\langle\phi_K(V_1)\rangle$ are ℓ_M^2-isogenous, since they are both ℓ_M-isogenous to $E_K/\langle[\ell_M]\phi_K(V_0)\rangle = E_K/\langle[\ell_M]\phi_K(V_1)\rangle$. The attacker recovers the curve $E_K/\langle[\ell_M]\phi_K(V_0)\rangle$, which is the first intermediate curve on the ϕ_M isogeny path by computing the common neighbour of $E_K/\langle\phi_K(V_0)\rangle$ and $E_K/\langle\phi_K(V_1)\rangle$.

The rest of the attacks proceeds similarly. The attacker queries with the points $V_i := P_M + [\ell_M^{e_M-i}]Q_M$, $i = 1, \ldots, e_M/2$ and runs a MITM attack to recover $E_K/\langle[\ell_M^i]\phi_K(V_0)\rangle$ given $E_K/\langle\phi_K(V_i)\rangle$ and $E_K/\langle[\ell_M^{i-1}]\phi_K(V_0)]\rangle$. This could be repeated e_M times to recover the entire isogeny ϕ_M. However, the attacker does not need to recover the last part of the isogeny through this strategy, since it is faster to directly compute the MITM between $E_K/\langle[\ell_M^{e_M/2}]V_0\rangle$ and the starting curve E_K. The attack with the required meet-in-the-middle computations is shown in Fig. 3.

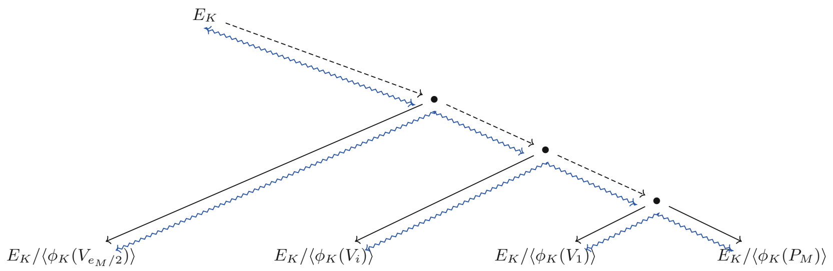

E_K
$E_K/\langle\phi_K(V_{e_M/2})\rangle$
$E_K/\langle\phi_K(V_i)\rangle$
$E_K/\langle\phi_K(V_1)\rangle$
$E_K/\langle\phi_K(P_M)\rangle$

Fig. 3. Naïve attack where isogenies of increasing length need to be recovered. The blue lines represent the meet-in-the-middle computations. (Color figure online)

Note that the isogenies that need to be recovered using MITM grow at each step. To recover the i-th intermediate curve, the attacker needs to compute an isogeny between two curves that are $\ell_M^{(i+1)}$-isogenous, which takes roughly $O(\ell_M^{(i+1)/2})$.

Clearly, this attack can be optimised by recovering multiple steps of ϕ_M at a time, and by making sure that the different MITM attacks that need to be executed have similar complexity. We will discuss these improvements in the following.

Full attack with query-time trade-off. We can reduce the complexity of the naïve attack by introducing a trade-off between queries and the cost of MITM computations. This is because the attacker recovers the whole path between two isogenies during a MITM computation. Thus, it is possible to recover more than one intermediate curve with a single (longer) MITM computation. Moreover, the queries can be spaced out more in order to reduce the length of the isogenies that have to be recovered using MITM strategies.

More formally, let 2^q denote the number of queries that an attacker can (or wants to) send to the challenger. For simplicity of this exposition, assume that $2e_m$ is divisible by $q+2$. The attacker chooses the V_i such that $E_0/\langle K, V_i\rangle$ correspond to curves that are the leaves of a binary isogeny tree. The V_i should be chosen such that there is an $\ell_M^{2e_M/(q+2)}$ isogeny between any two siblings in the binary tree and the curve that is $\ell_M^{e_M/(q+2)}$-isogenous to both leaves is their parent in the tree. Again, the parent and its sibling should be $\ell_M^{2e_M/(q+2)}$-isogenous, etc.

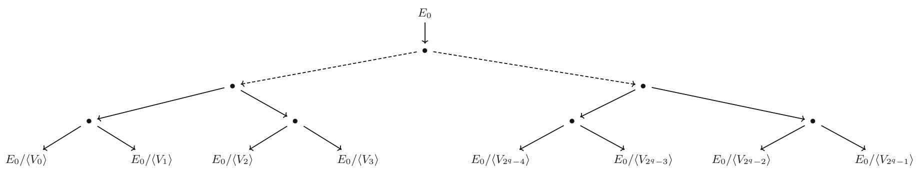

E_0
$E_0/\langle V_0\rangle$
$E_0/\langle V_1\rangle$
$E_0/\langle V_2\rangle$
$E_0/\langle V_3\rangle$
$E_0/\langle V_{2^q-4}\rangle$
$E_0/\langle V_{2^q-3}\rangle$
$E_0/\langle V_{2^q-2}\rangle$
$E_0/\langle V_{2^q-1}\rangle$

Fig. 4. The attacker queries the challenger on points corresponding to isogeny kernels leading to the leaves of this binary tree

Remark 2. Note that it is easy to choose such a set of points V_i. Let P_M, Q_M be a basis of $E_0[\ell_M^{e_M}]$. The attacker can choose

$$V_0 := P_M$$
$$V_i := V_{i-2^{\lfloor \log i \rfloor}} + [\ell_M^{e_M - (\lfloor \log i \rfloor + 1)2e_M/(q+2)}]Q_M$$

Lemma 3. *Let $E_0/\langle V_i\rangle$ and $E_0/\langle V_j\rangle$ be ℓ_M^k isogenous curves. Then $E_K/\langle \phi_K(V_i)\rangle$ and $E_K/\langle \phi_K(V_j)\rangle$ are ℓ_M^k-isogenous curves too.*

Proof. This follows from $N_{\mathsf{K}} = \deg(\phi_K)$ being coprime to ℓ_M^k. □

In particular, $\{\phi_K(P_M), \phi_K(Q_M)\}$ is a basis of $E_K[N_{\mathsf{M}}]$ and $E_K/\langle \phi_K(V_i)\rangle$ are the leaves in a binary tree where all siblings are $\ell_M^{2e_M/(q+2)}$ isogenous.

After querying the oracle to obtain $E_K/\langle \phi_K(V_i)\rangle = E_0/\langle K, V_i\rangle$, an attacker recovers iteratively parent nodes in the binary tree using a meet-in-the-middle approach. Any siblings in the tree correspond to curves that are $\ell_M^{2e_M/(q+2)}$-isogenous, thus this can be done in $O(\ell_M^{e_M/(q+2)})$. Note that the root of the binary tree is recovered after $2^q - 1$ such meet-in-the-middle instances, i.e. the number of internal nodes in the binary tree. This root of the binary tree is then by construction $\ell_M^{2e_M/(q+2)}$-isogenous to E_0. This final isogeny can be recovered using meet-in-the-middle again. An attacker recovers and saves the intermediate nodes and isogenies from E_K to the leaf $E_K/\phi_K(V_0)$. Clearly, the kernel of this isogeny is $\phi_K(V_0)$.

In summary, we can recover the isogeny from $E_K \to E_K/\langle \phi_K(P_M)\rangle$ for any P_M with 2^q queries to the challenger and 2^q instances of meet-in-the-middle isogeny computations with cost of $O(\ell_M^{e_M/(q+2)})$ each.

Remark 3. If $\ell_M = 2$, we get q bits for free, i.e. one additional bit per layer of the binary tree. This is because every parent node in the binary tree has three outgoing edges: two edges leading to its children and one edge leading towards the root. Thus, having recovered both paths to the children an attacker gets one step towards the root for free.

3.4 Attack Analysis

The proposed attack is composed of two stages: firstly the generators of $\langle \phi_K(P_V)\rangle$, $\langle \phi_K(Q_V)\rangle$, and $\langle \phi_K(R_V)\rangle$ are recovered, and then these points are used to recover $\phi_K(M)$ for any possibly challenge $M \in E_0[N_{\mathsf{M}}]$.

The second part consists mostly of pairing evaluations and discrete log computations in groups of smooth order. Thus, it runs in polynomial time. The complexity of the attack is dominated by the complexity of recovering the subgroups in the first step.

The algorithm proposed in Sect. 3.3 offers different trade-offs between computation costs and solve queries. As little as two solve queries can be enough to recover ϕ_M with two meet-in-the-middle computations. If we write $N_{\mathsf{M}} \approx 2^m$, each meet-in-the-middle requires $O(2^{m/3})$ operations. This is already an improvement over the standard meet-in-the-middle attack that requires $O(2^{m/2})$ time. The OPRF protocol targets 128 bits of security, which corresponds to $m \approx 5\lambda/2 = 320$. Thus six queries (two per generator) are enough to reduce the security to $m/3 = 106$ bits. The number of solve queries can be significantly increased to obtain a faster attack. Note that OPRF protocols are usually used for applications such as private set intersection, that support a large number of queries. Thus, common scenarios where the OPRF may be used would easily lend themselves to an attack with many queries.

Since OPRFs are used in protocols where the clients interact several times with the server, we can expect the attacker to be able to run several OPRF instances. Thus, we model a solve query as an oracle query, where it has a unitary complexity. Then, the overall complexity of recovering a generator of $\langle \phi_K(P_V) \rangle$ with 2^q solve queries is $O(2^{m/(q+2)+q})$ operations, since the attacker needs to compute 2^q meet-in-the-middle instances between curves which are $2^{2m/(q+2)}$-isogenous. In terms of the security parameter, that complexity is equivalent to $O(2^{5\lambda/2(q+2)\,+\,q})$, since the OPRF protocol suggests using $m \approx 5\lambda/2$. If the number of solve queries is unrestricted, the complexity of the attack is minimized for $q = \sqrt{5\lambda/2} - 2$, which gives an overall complexity of $O(2^{\sqrt{10\lambda}-2})$, or using the L-notation $L[1/2, c]$, for some constant c. This shows the attack is subexponential, assuming that the solve query complexity is $O(1)$.

At 128-bit of security, our attack becomes feasible with around 64 solve queries, when it requires 64 meet-in-the-middle computations between curves which are 2^{80}-isogenous, i.e. each MITM has a complexity of 2^{40} operations. If the number of solve queries is unrestricted, an attacker can use 2^{18} solve queries to reduce the overall complexity of the attack to 2^{18} MITM computations, where each MITM operations has a complexity of 2^{16} operations.

The high-level attack does not generally require much memory. Storing the isogeny tree in memory is not particularly demanding, especially if the tree is traversed depth-first. In particular, memory is used only to store the part of the recovered isogeny, together with the two curves between which the meet-in-the-middle needs to be computed. However, a more significant amount of memory is used by the meet-in-the-middle computations, and indeed we see that the memory used by a single meet-in-the-middle generally outweighs the memory used by the rest of the attack. Meet-in-the-middle computations between curves which are 2^n-isogenous require to store $2^{n/2}$ curves. Thus, their memory requirements are given by $2 \cdot 2^{n/2} \log p$, since each curve can be represented by its j-invariant in $\mathbb{F}_{p^2}$. For common security levels, such as those proposed by Boneh et al. [3], the memory requirements remain moderate. In Sect. 5, we show that indeed our attack requires about 3 GB of memory to break 128 bits of security. However, for a more complete asymptotical analysis, we note that the memory requirements may become a bottleneck for the attack against higher security levels. In those instances, it may be preferable to substitute the meet-in-the-middle approach with the van Oorschot-Wiener algorithm [28]. This reduces the memory consumption at the cost of higher asymptotic complexity. In particular, the vOW algorithm requires $O(2^{3n/4})$ computations (compared to $O(2^{n/2})$ of MITM) to recover the halfway curve between curves which are 2^n-isogenous. Thus, while the concrete performance of the attack may differ, its asymptotic complexity remains subexponential.

Future improvements. A natural question to ask is whether the proposed attack that queries points of the correct order may be improved to achieve a polynomial running time. Consider that an attacker chooses an isogeny $\phi_V : E_0 \rightarrow E_0/\langle V \rangle$ and he is given the curve $E_0/\langle K, V \rangle$. Since the attacker knows the entire isogeny ϕ_V, backtracking from $E_0/\langle K, V \rangle$ to E_K to recover $\phi_K(V)$ in polynomial time does not seem too far fetched, since the attacker knows the entire isogeny ϕ_V. A possible strategy may start by retrieving $E_0/\langle K, V \rangle$ and $E_0/\langle K, V + \ell_M^{e_M} V' \rangle$, for a point V' linearly independent of V. Their common ℓ_M-neighbour is the first curve on the isogeny path. Then, the attacker may use the knowledge of ϕ_V starting from E_0 to distinguish between the ℓ_M possible candidates for the next curve on the isogeny path. Unfortunately, our efforts to develop such an attack did not succeed. It remains an open problem whether such an attack is possible.

4 Attack on the OPRF

Having presented an attack on one of the security assumptions underlying the isogeny-based OPRF by Boneh et al., we investigate how an adversary can use the same method to attack the OPRF itself.

We will show that a malicious client can send carefully crafted queries to the server for which it can produce all necessary NIZK proofs required by the protocol that was summarized in Sect. 2.4. However, after some offline computation analogously to the attack on the auxiliary one-more SIDH assumption outlined in the previous section, the malicious client can evaluate the OPRF on any input without the help of the server. Even though the malicious client does not recover the server's secret key k, this breaks the "pseudorandomness", Definition 1, of the OPRF. We will use the same notation as in Sect. 2.4 to refer to different isogenies of the OPRF.

A malicious client will not use a hashed input to obtain the kernel for the first isogeny $\phi_m : E_0 \rightarrow E_m$ but rather choose the kernel of this first isogeny maliciously. The choice is analogous to the points from $E_0[N_{\mathsf{M}}]$ that the attacker submitted to the solve queries in the attack of the previous section. In other words, instead of computing E_m as $E_0/\langle P + H(x)Q \rangle$ for some input x, the malicious client chooses a point V_i and computes E_m as $E_0/\langle V_i \rangle$ in the i-th evaluation of the OPRF.

The rest of the protocol is executed honestly. The malicious client can pick some $r \in N_{\mathsf{R}}$ to blind his maliciously chosen E_m. And it can compute the torsion point information for the server honestly since it knows the kernel of the isogeny $E_0 \to E_{mr} = E_m/\langle \phi_m(P_R) + [r]\phi_m(Q_R) \rangle$. In particular, the malicious client will always be able to produce the valid non-interactive zero-knowledge proof of knowledge for the kernel of $E_0 \to E_{mr}$ and the correct computation of the torsion point information.

Following through with the rest of the OPRF protocol, the malicious client obtains the j-invariant of the curve $E_0/\langle V_i, K \rangle$ after unblinding. Here K denotes the server's secret $P_K + [k]Q_K$. This is exactly what corresponds to a "solve query" in the auxiliary one-more SIDH game, Game 1.

Now the malicious client can proceed as in the attacks on the auxiliary one-more SIDH assumption.

In the attack using points of arbitrary order dividing N_{M}, the malicious client recovers the isogeny $E_K \to E_K/\langle \phi_K(P) \rangle = E_0/\langle K, P \rangle$ and therefore $\langle \phi_K(P) \rangle$ for any $P \in E_0[N_{\mathsf{M}}]$ in polynomial time. This is done by submitting points of lower order, i.e. choosing the isogeny $E_0 \to E_m$ shorter, to recover the isogeny stepwise. After recovering three such isogenies corresponding to pairwise linearly independent points $P, Q, P+Q$, the malicious client can compute $E_0/\langle M, K \rangle$ for any $M \in E_0[N_{\mathsf{M}}]$ as was shown in Sect. 3.2.

Then, the malicious client can evaluate the OPRF on arbitrary inputs x as follows: They compute the point $M := P_M + H_1(x)Q_M$ as in the honest evaluation and then they compute $j(E_0/\langle M, K \rangle)$ directly. Hashing this j-invariant together with the input x and public information of the server yields the output of the OPRF. Note that the malicious client does not even need to interact with the server to evaluate the OPRF on arbitrary inputs.

Clearly, this breaks the pseudorandomness property of an OPRF, see Definition 1, as a malicious client will be able to predict the output of the OPRF for any input after the initial queries.

Remark 4. The SIDH-based OPRF protocol by Boneh et al. does not prohibit malicious clients from using points of smaller order dividing N_{M}, i.e. from using a shorter isogeny $E_0 \to E_m$. However, this attack could be thwarted if the server checked that the submitted curve is of correct distance from the starting curve. A simple test using pairing computations on the provided torsion point information may be tricked, but the NIZK of the client could be extended to include a proof that the client's witness, i.e. the kernel of the isogeny $E_0 \to E_{mr}$, is of full order $N_{\mathsf{M}}N_{\mathsf{R}}$.

Even if countermeasures for this polynomial-time attack are put in place, we are left with the following subexponential attack when points of full order are used.

The client evaluates the OPRF on a certain number of inputs that correspond to solve queries in the auxiliary one-more game. More precisely, the client chooses the kernel of his first isogeny as in the subexponential attack of the previous section. After blinding, evaluation of the server and unblinding, the client obtains what would have been the result of a "solve query" in the previous section. After the offline computation which, using meet-in-the-middle routines, recovers the binary tree described in Sect. 5, the client obtains torsion point images of $E_0[N_{\mathsf{M}}]$ up to scalar under the isogeny $E_0 \to E_K := E_0/\langle P_K + [k]Q_K\rangle$. Again this is enough to compute $E_0/\langle M, K\rangle$ for any $M \in E_0[N_{\mathsf{M}}]$ by Sect. 3.2, allowing the client to compute the OPRF on arbitrary inputs and therefore breaking the pseudorandomness property.

Possible countermeasures. In the case where the degree of the client's isogeny is forced to be $N_{\mathsf{M}}N_{\mathsf{R}}$, the proposed attack has subexponential complexity, and thus possible countermeasures may include increasing the parameter sizes. However, the solve queries to time trade-off may reduce the feasibility of such an approach. If the number of possible solve queries is unrestricted, to get 128-bit security one would need the isogeny degree N_{M} to be $\approx 2^{(128^2)}$. This can be partially mitigated by guaranteeing security only up to a certain number of queries. Given a limit of 2^Q queries, the exponent m needs to guarantee that $\min\{2^{\sqrt{m}-2}, 2^{m/(Q+2)+Q}\}$ is at least 2^λ. Thus, for 128-bit security, with $Q = 64$ the isogeny degree N_{M} would have to be increased to $\approx 2^{4224}$, whereas $Q = 32$ would require a degree $N_{\mathsf{M}} \approx 2^{3264}$. Note that handling 2^{32} queries may well be within the scope of several OPRF applications, and isogenies of such a size may become impractically large. Their feasibility, however, depends on the specifics of the OPRF application and its time and bandwidth requirements. Thus, while the attack is subexponential (assuming $O(1)$ complexity for solve queries), increasing the parameter size comes at a significant performance and communication cost.

Therefore, it is important to consider possible algorithmic countermeasures. Firstly, note that the attacker submits seemingly valid requests, so the server cannot stop such interactions. Even if the server did want to prevent these requests, it may not be able to detect them. This is because the attacker only submits the image curve and some torsion point images under an isogeny with chosen kernel.

However, the attack strongly depends on the attacker choosing the point V. If the input points V were randomized, the attack as such could not work. The OPRF protocol requires that such points are obtained via hashing the client's PRF input x, but it does not enforce it. Hence, a possible countermeasure to the proposed attack would be requiring the client to provide a zero-knowledge proof that the curve E_{mr} is not only the result of honest isogeny computations, but also that the kernel of ϕ_m is the result of some hash function. However, developing an ad-hoc and efficient proof that can prove such statements remains an open problem.

5 Attack Implementation

We implemented the subexponential attack of Sect. 3.3 in SageMath to demonstrate the correctness of the algorithm and prove its feasibility. The source code is freely available at https://github.com/isogenists/isogeny-OPRF. We remark that this implementation is to be regarded only as a proof-of-concept and that several subroutines can be further optimized. Improving their performance and using lower-level languages, such as C, as well as platform-specific instructions, such as AVX, could significantly reduce the running time of the attack.

The proposed attack has two distinguishing features that help its implementation: it can be easily parallelized, and it has very low memory requirements. Indeed, the computations to recover the generators of $\langle \phi_K(P_V) \rangle$, $\langle \phi_K(Q_V) \rangle$ and $\langle \phi_K(P_V + Q_V) \rangle$ are independent of each other. It is also possible to achieve a higher degree of parallelization. Within each computation to recover a single generator, the meet-in-the-middle operations within each layer of the tree are also independent of each other, and they can thus be parallelized. In this case, the tree is generated layer-by-layer in a breadth-first manner. Note that while this may require a sizeable amount of memory to fully store an entire layer, the memory requirements are hardly the bottleneck. An attack with 2^{20} queries requires to store, at most, 2^{19} curves. Since an elliptic curve can be represented by its j-invariant, the memory limit is $2^{19} \cdot 2 \log p$. With a prime of size $\approx 2^{1500}$, as proposed in the OPRF protocol, the memory limit is about 196 MB. Alternatively, it is possible to traverse the tree in a depth-first manner to further lower the memory requirement, but this may limit the degree of parallelization. We remark that while parallelization only provides a linear speed-up, its effects can be significant. Our implementation provides parallelized meet-in-the-middle computations with a configurable number of cores in parallel.

Results. The majority of the attack's subroutines have polynomial complexity and they are optimized enough that their performance does not affect the overall running time. The building block that most affects the performance of the attack is the meet-in-the-middle computation. Indeed, the timings of the attack are directly correlated to the timing of a single meet-in-the-middle and the total number of queries. The memory requirements of the attack are given by the amount of memory needed for a single meet-in-the-middle, which in turn depends on the distance between the two curves. For parallelized implementations of the attack, the memory requirements correspond to as many meet-in-the-middle computations as there are parallel instances.

Table 1 shows the running times at different security levels on an Apple M1 CPU clocked at 3.20 GHz with 4 CPUs running in parallel. Up to 32 bits of security, the results come from running the entire attack, whereas for higher security levels the results are estimated based on those of a single MITM computation. The estimated time t is computed as

$$t = \frac{3(M+Q)2^q}{C}, \tag{1}$$

where M is the average running time of a MITM computation, Q is the average running time of a solve query computation, 2^q is the number of queries and C is the number of CPUs running in parallel. This formula follows from the fact that there are 2^q MITM computations and 2^q solve queries for each generator recovery, and three of those are needed. Moreover, parallelization gives a linear speed-up, and the remaining computations (such as those of Sect. 3.2) are extremely fast when compared to the rest of the attack, and thus negligible. Running computations at lower security levels and computing Eq. 1 does indeed estimate the running time accurately. It should be noted that this remains an estimate and the real results may vary to some degree.

We estimate that our non-optimized implementation running on a laptop with 4 CPUs can break 64 bits[2] of security in less than two days and 128 bits of security in about 5 years. If the same attack was performed with more powerful hardware and an optimized implementation, the running time could easily be reduced to a matter of months, if not weeks. We remark that if a server rotates its keys often, an attack that breaks the server one-more unpredictability after the key has changed still leads to significant attacks. For instance, in the case of OPRF-based private set intersection protocols, breaking the one-more unpredictability property allows the attacker to break the privacy property of the server's set at the time when that specific key was used.

Lastly, note that in the implementation solve queries are simulated locally. A real attack would interact with the server, and thus the "correct" attack time should not include the query computation times. For completeness, Table 1 reports the running time of the entire implementation, including the solve queries.

6 Trusted Setup

In the OPRF protocol of Boneh et al., the authors suggest using a random supersingular elliptic curve as starting curve. However, there is currently no known algorithm to generate a random supersingular elliptic curve such that its endomorphism ring is unknown to the person who generated it. Some attempts to solve this problem have been proposed in [26] and further studied in [6]. This motivates the following question:

[2] We report the results for $e_M = 169$, which corresponds to $\lambda = 67$. That is because our implementation requires $(q + 2) \mid e_M$, and 169 allows choosing $q = 11$. Using $e_M = 160$ would have required using significantly more queries or a longer MITM, thus resulting in worse performance. Note that the requirement that $(q + 2) \mid e_M$ is a limitation of the implementation and not of the attack itself.

Table 1. Results of our proof-of-concept implementation of the attack, running on an Apple M1 CPU clocked at 3.20 GHz with 4 CPUs in parallel and SageMath version 9.2. Results for $\lambda = 128$ are estimated based on the average running time of a meet-in-the-middle computation. Parameters include the size of the prime p, the security level λ, the degree of the isogeny written as $N_{\mathsf{M}} = 2^{e_M}$, and the number of queries 2^q. The MITM section reports the distance between the curves and memory needed to compute a single meet-in-the-middle.

Parameters				MITM		Running time
$\log p$	λ	e_M	q	Distance	Memory (kB)	(s)
112	8	20	3	8	3.5	15
216	16	40	6	10	13.8	212 (3.53 m)
413	32	80	8	16	211.4	1,371 (22.85 m)
859	67	169	11	26	14,073	163,869 (1.89 d)
1,614	128	320	18	40	3,384,803	174,709,440 (5.54 y)

Is a trusted third party needed to generate the starting curve E_0?

Phrased differently, would choosing the starting curve E_0 and therefore knowledge of its endomorphism ring allow a malicious server, client or third party to break security properties of the (V)OPRF?

We first discuss whether a server may know the endomorphism ring of the starting curve E_0. The security proof by Boneh et al.'s OPRF relies on the hardness of finding two distinct isogenies (up to isomorphism) of the same degree from E_0 to a second curve [3, Lemma 29]. If the server chooses the starting curve and therefore knows its endomorphism ring, they are able to produce such collisions by breaking the collision resistance of the CGL hash function as in [15,29]. To guarantee provable security, a server should therefore not choose the starting curve.

However, breaking the verifiability insured by the zero-knowledge proof [3, Protocol 17] or the weak binding property [3, Game 3] of the protocol seems harder than finding collisions. Indeed, the server would need to produce two isogenies of degree dividing N_{K} such that both isogenies have the same action on the N_{V}-torsion for a chosen starting curve. We leave adapting the security proofs or finding an attack on the zero-knowledge proof for future work.

We now argue that any other party, either the client or a third party, cannot choose the curve E_0 either without compromising the security of the protocol. In [13], the authors describe algorithms for finding a secret isogeny when torsion information is provided. Their algorithms can be split into two categories: one where the starting curve has j-invariant 1728 and one where the starting curve is a trapdoor curve from which one can solve the isogeny problem faster than generic meet-in-the-middle algorithms. Trapdoor curves are parameterized by a pair (A, B) where A corresponds to the degree of the secret isogeny and B to the order of torsion points whose image under the secret isogeny is known. When $B \approx A^2$ or larger, then one can construct (A, B) trapdoor curves from which one can retrieve secret isogenies of degree A in polynomial time, if the action on the B-torsion is known [13, Theorem 15].

Attacks from the special starting curve with j-invariant 1728 do not apply here, since the starting curve cannot have j-invariant 1728 because the endomorphism ring needs to be unknown to the server. However, trapdoor curves have the property that without extra information they are difficult to distinguish from a random supersingular curve.

Suppose that a malicious party generates the starting curve E_0 in the following way. They generate a curve E' which is a trapdoor $(N_{\mathsf{K}}, N_{\mathsf{V}}N_{\mathsf{R}})$-curve and then perform a random walk of length $N_{\mathsf{M}}N_{\mathsf{R}}$ to obtain E_0 which is sent to the server. Now the malicious party poses as a client and instead of honestly complying with the protocol, they use E' as E_{mr}. They can prove knowledge of a suitable isogeny and torsion point images as they know an isogeny of the correct degree from E_0. Then the server computes E_{mrk} and reveals the action of the $N_{\mathsf{V}}N_{\mathsf{R}}$-torsion. Since E_{mr} was chosen to be a trapdoor curve and $N_{\mathsf{V}}N_{\mathsf{R}} \approx N_{\mathsf{K}}^2$, the malicious party can retrieve this isogeny in polynomial time.

Such an attack can be thwarted by applying a trusted setup in which E_0 is a truly random curve. In [4, §4], an efficient way to perform a distributed trusted setup is described, ensuring that, if at least one participant is honest, the setup can be trusted. In that case, torsion point attacks are not applicable. The attack can also be weakened by substantially increasing N_{K}. However, this might be susceptible to future improvements of trapdoor curve constructions.

7 Conclusion

In this paper, we perform a thorough cryptanalysis of Boneh et al.'s SIDH-based oblivious pseudorandom function. The security of this OPRF is based on a new hardness assumption, the auxiliary one-more assumption. We investigate this assumption and we show how an attacker can win the corresponding security game in polynomial time, or with the appropriate countermeasures in subexponential time.

The attack on the underlying hardness assumption leads to an attack on the pseudorandomness of the OPRF itself. We show how a malicious client can extract enough information from a number of initial executions of the OPRF protocol to subsequently evaluate the OPRF on arbitrary inputs without further interaction with the server. In particular, this attack breaks the security parameters provided by Boneh et al. As a proof of concept, we implement the attack in SageMath, verified its correctness and give timings for various security levels.

Furthermore, we discuss the security implications following from a lack of a trusted setup when generating the starting curve parameter in the SIDH-based OPRF. Note that Boneh et al. do not explicitly require a trusted setup. We show how a client or a third party generating the starting curve can backdoor it to retrieve the server's secret key, while a malicious server could generate the starting curve to break the supersingular-isogeny collision assumption.

This work leads to some open problems. On one hand, one could improve and extend the proposed attack, with a particular focus on reducing the complexity of the subexponential attack to polynomial time, as well as extending it to the CSIDH-based OPRF. On the other hand, further work is needed to develop efficient countermeasures against the subexponential attack or to design a novel SIDH-based VOPRF. Future research will also focus on understanding the implications of breaking the supersingular-isogeny collision assumption on the OPRF protocol itself, and whether it is possible to avoid a trusted setup.

Acknowledgments. We would like to thank Dan Boneh, Jesús Javier Chi Domínguez, Luca De Feo, Enric Florit, Dmitry Kogan and Simon Masson for fruitful discussions. Péter Kutas, Simon-Philipp Merz and Christophe Petit were supported by EPSRC and the UK government as part of the grant EP/S01361X/1 for Péter Kutas and Christophe Petit and the grant EP/P009301/1 for Simon-Philipp Merz. Further, Péter Kutas was supported by the Ministry of Innovation and Technology and the National Research, Development and Innovation Office within the Quantum Information National Laboratory of Hungary.

References

1. Albrecht, M.R., Davidson, A., Deo, A., Smart, N.P.: Round-optimal verifiable oblivious pseudorandom functions from ideal lattices. In: Garay, J.A. (ed.) PKC 2021. LNCS, vol. 12711, pp. 261–289. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-75248-4_10
2. Azarderakhsh, R., et al.: Supersingular isogeny key encapsulation. Updated parameters for round 2 of NIST Post-Quantum Standardization project (2019)
3. Boneh, D., Kogan, D., Woo, K.: Oblivious pseudorandom functions from isogenies. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12492, pp. 520–550. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64834-3_18
4. Burdges, J., Feo, L.D.: Delay encryption. Cryptology ePrint Archive, Report 2020/638 (2020). https://eprint.iacr.org/2020/638
5. 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. LNCS, vol. 11274, pp. 395–427. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03332-3_15
6. Castryck, W., Panny, L., Vercauteren, F.: Rational isogenies from irrational endomorphisms. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12106, pp. 523–548. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45724-2_18
7. Charles, D.X., Lauter, K.E., Goren, E.Z.: Cryptographic hash functions from expander graphs. J. Cryptology **22**(1), 93–113 (2009)
8. Chaum, D.: Blind signatures for untraceable payments. In: Advances in Cryptology: Proceedings of CRYPTO 1982, Santa Barbara, California, USA, 23–25 August 1982, pp. 199–203 (1982)
9. Couveignes, J.M.: Hard homogeneous spaces. IACR Cryptology ePrint Archive **2006**, 291 (1999)
10. Davidson, A., Goldberg, I., Sullivan, N., Tankersley, G., Valsorda, F.: Privacy pass: bypassing internet challenges anonymously. Proc. Priv. Enhancing Technol. **2018**(3), 164–180 (2018)

11. Davidson, A., Sullivan, N., Wood, C.A.: Oblivious Pseudorandom Functions (OPRFs) using Prime-Order Groups. Internet-Draft draft-sullivan-cfrg-voprf-03, Internet Engineering Task Force (2019), work in Progress
12. De Feo, L.: Mathematics of isogeny based cryptography. arXiv preprint: arXiv:1711.04062 (2017)
13. de Quehen, V., et al.: Improved torsion point attacks on SIDH variants. arXiv e-prints arXiv:2005.14681 (May 2020)
14. Demmler, D., Rindal, P., Rosulek, M., Trieu, N.: PIR-PSI: scaling private contact discovery. Proc. Priv. Enhancing Technol. **2018**(4), 159–178 (2018)
15. Eisenträger, K., Hallgren, S., Lauter, K., Morrison, T., Petit, C.: Super singular isogeny graphs and endomorphism rings: reductions and solutions. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10822, pp. 329–368. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78372-7_11
16. Everspaugh, A., Chatterjee, R., Scott, S., Juels, A., Ristenpart, T.: The pythia PRF service. In: Jung, J., Holz, T. (eds.) 24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, 12–14 August 2015, pp. 547–562. USENIX Association (2015)
17. Freedman, M.J., Ishai, Y., Pinkas, B., Reingold, O.: Keyword search and oblivious pseudorandom functions. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 303–324. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30576-7_17
18. Galbraith, S.D., Petit, C., Shani, B., Ti, Y.B.: On the security of super singular isogeny cryptosystems. In: Advances in Cryptology - ASIACRYPT 2016, pp. 63–91 (2016). https://doi.org/10.1007/978-3-662-53887-6_3
19. Jao, D., et al.: SIKE: Supersingular isogeny key encapsulation http://sike.org/ (2017)
20. Jao, D., De Feo, L.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In: Yang, B.-Y. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 19–34. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25405-5_2
21. Jarecki, S., Kiayias, A., Krawczyk, H.: Round-optimal password-protected secret sharing and T-PAKE in the password-only model. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8874, pp. 233–253. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45608-8_13
22. Jarecki, S., Krawczyk, H., Xu, J.: OPAQUE: an asymmetric PAKE protocol secure against pre-computation attacks. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10822, pp. 456–486. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78372-7_15
23. Jarecki, S., Liu, X.: Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection. In: Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, 15–17 March 2009. Proceedings, pp. 577–594 (2009)
24. Jarecki, S., Liu, X.: Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 577–594. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00457-5_34
25. Li, L., Pal, B., Ali, J., Sullivan, N., Chatterjee, R., Ristenpart, T.: Protocols for checking compromised credentials. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, 11–15 November 2019, pp. 1387–1403. ACM (2019)
26. Love, J., Boneh, D.: Supersingular curves with small noninteger endomorphisms. Open Book Ser. **4**(1), 7–22 (2020)

27. Merz, S.-P., Minko, R., Petit, C.: Another look at some isogeny hardness assumptions. In: Jarecki, S. (ed.) CT-RSA 2020. LNCS, vol. 12006, pp. 496–511. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-40186-3_21
28. van Oorschot, P.C., Wiener, M.J.: Parallel collision search with cryptanalytic applications. J. Cryptology **12**(1), 1–28 (1999)
29. Petit, C., Lauter, K.E.: Hard and easy problems for supersingular isogeny graphs. IACR Cryptol. ePrint Arch. 2017, 962 (2017). http://eprint.iacr.org/2017/962
30. Rostovtsev, A., Stolbunov, A.: Public-key cryptosystem based on isogenies. IACR Cryptology ePrint Archive **2006**, 145 (2006)
31. Seres, I.A., Horváth, M., Burcsi, P.: The legendre pseudorandom function as a multivariate quadratic cryptosystem: Security and applications. IACR Cryptol. ePrint Arch. 2021, 182 (2021). https://eprint.iacr.org/2021/182
32. Silverman, J.H.: The Arithmetic of Elliptic Curves. GTM, vol. 106. Springer, New York (2009). https://doi.org/10.1007/978-0-387-09494-6
33. Sullivan, N., Krawczyk, D.H., Friel, O., Barnes, R.: OPAQUE with TLS 1.3. Internet-Draft draft-sullivan-tls-opaque-01, Internet Engineering Task Force (2021), work in Progress