Post-Quantum Static-Static Key Agreement Using Multiple Protocol Instances

Reza Azarderakhsh[3], David Jao[1,2] and Christopher Leonardi[1]

[1] University of Waterloo, {djao,cfoleona}@uwaterloo.ca
[2] evolutionQ, Inc., david.jao@evolutionq.com
[3] Florida Atlantic University, razarderakhsh@fau.edu

Abstract. Some key agreement protocols leak information about secret keys if dishonest participants use specialized public keys. We formalize these protocols and attacks, and present a generic transformation that can be made to such key agreement protocols to resist such attacks. Simply put, each party generates k different keys, and two parties perform key agreement using all k^2 combinations of their individual keys. We consider this transformation in the context of various post-quantum key agreement schemes and analyze the attacker's success probabilities (which depend on the details of the underlying key agreement protocol) to determine the necessary parameter sizes for 128-bit security. Our transformation increases key sizes by a factor of k and computation times by k^2, which represents a significant cost—but nevertheless still feasible. Our transformation is particularly well-suited to supersingular isogeny Diffie-Hellman, in which one can take $k = 113$ instead of the usual $k = 256$ at the 128-bit quantum security level. These results represent a potential path forward towards solving the open problem of securing long-term static-static key exchange against quantum adversaries.

Keywords: post-quantum cryptography, key agreement, isogenies, supersingular isogeny Diffie-Hellman

1 Introduction

In Asiacrypt 2016, Galbraith, Petit, Shani, and Ti [13] introduced an active attack against the supersingular isogeny-based cryptosystem of De Feo, Jao, and Plût [10], which circumvents all extant (at the time) direct validation techniques. The attack allows an attacker who interacts with a static key over multiple rounds of key exchange to efficiently compute the private key corresponding to the static key over multiple sessions. When communicating, the participants in an SIDH key exchange each send a supersingular elliptic curve and two points on the curve. By manipulating the values of the two points, the attacker can learn one bit of information about the other participant's private key (depending on whether or not the key exchange operation succeeds using the manipulated points), and then repeat this process over additional sessions to learn additional private key bits. As stated in [13], a countermeasure to their attack was already available in the earlier work of Kirkwood et al. [17], who proposed so-called "indirect key validation" using a Fujisaki-Okamoto type transform [12] in order to allow the honest participant to detect whether or not the other party is manipulating points. Unfortunately, this countermeasure requires the untrusted party to disclose their SIDH private key, precluding the use of SIDH as a drop-in replacement for Diffie-Hellman or other protocols that support static-static key exchange using direct key validation.

Although [13] specifically targets SIDH, similar attacks apply against all other available post-quantum cryptosystems. No currently known post-quantum scheme achieves secure static-static key exchange without the use of ephemeral keys or indirect validation techniques that would expose one's key in the static-static setting. Major lattice-based key establishment schemes such as "A New Hope" [1] and "Frodo" [5] achieve only passive security and are intended and designed to be used with ephemeral keys. Peikert's Ring-LWE based scheme [19] is a key encapsulation mechanism that uses a Fujisaki-Okamoto type transform to achieve IND-CCA security [19, §5]. In Peikert's scheme, the encrypting participant must reveal their random coins to the decrypting participant, and so one member must use an ephemeral key. The Module-LWE key exchange Kyber [6, §5] has at least one party using an ephemeral key, and both parties using both a static and ephemeral key in the authenticated variant. In Niederreiter hybrid encryption [23, §3.1], the error vector is revealed and used to derive the shared symmetric key. Similarly, in McEliece encryption [18], although the error vector is not explicitly used in decryption, it is trivial to compute once the message is determined, and therefore one party must use an ephemeral key.

In this work we present a new generic transformation that takes any key establishment protocol satisfying certain security properties (see Definition 3) and converts it into a different protocol that is immune to attacks of the form presented in [13]. In our transformation, each party generates k different key pairs and publishes for their public key the list of k individual public keys. During key agreement, two parties compute k^2 different shared secrets obtained by performing shared key agreement with each of their keys in all possible combinations, and hashing the shared secrets to derive a final shared key. Under this scheme, any use of an invalid public key will, with all but negligible probability, cause at least one of the k^2 shared secret computations to fail, which neutralizes the attack of [13]. Moreover, the number of possible failure outcomes is exponential in k, making it impossible for an attacker to predict a likely failure outcome in advance and lie about the value of their final shared key in order to salvage the attack of [13].

The necessary value of k depends on the details of the original protocol with which we started. The easiest (and worst) case is where each invalid key attempt in the original protocol leads to one of two possible (invalid) shared secret computations on the part of the honest party, depending on the value of one of the bits in the honest party's private key. In this case, one simply needs $k \approx \ell$ to achieve ℓ-bit classical security, and $k \approx 2\ell$ in the quantum case to account for Grover search. However, if there are more possible invalid outcomes, then the attacker's job is harder, and (as a designer) we can use a smaller value of k while still achieving ℓ-bit security. For example in Section 3.4 we perform a detailed analysis of SIDH and conclude that a value of $k = 113$ is sufficient to achieve 128-bit quantum security. While a key size penalty of a factor of $O(\ell)$ and performance penalty of a factor of $O(\ell^2)$ might seem untenable, we point out that our scheme is far from the worst in this regard compared to some recently published articles such as [3].

In Section 2 we present our security theorem which states that, for SIDH and other suitable protocols, our transformation is secure in the sense that finding even a single invalid key resulting in a successful key exchange (in the sense that the attacker can guess the shared secret computed by the honest party under this invalid key) is equivalent to breaking the passive security of the original un-transformed protocol. We recognize and emphasize that our security reduction falls short of a full proof of active security, as it only shows that attacks of the type that involve feeding an honest party invalid keys must fail, and not that arbitrary attacks must fail. Nevertheless, we suggest that our results provide a useful foundation for building secure static-static key agreement protocols, and is worthy of further study, especially in the post-quantum setting where the question of achieving secure static-static key agreement remains an open problem.

2 Multiple Instances of Key Agreement

We begin with a review of the format for key agreement protocols. The content of this paper focuses on two participants establishing a shared secret key that depends on inputs from both members, it does not address authentication.

Definition 1. *We let* $\boldsymbol{KE}$ *be a key establishment function (the requirements of which will be stated shortly). A key agreement protocol,* $\boldsymbol{KA}$*, for Alice and Bob using* $\boldsymbol{KE}$ *consists of the phases:*

0. ***Setup****: Both members obtain a valid copy of the global parameters, gp.*
1. ***Key Generation****: Alice generates a secret key* s_A *and public key* p_A*, likewise Bob generates* s_B *and* p_B*.*
2. ***Communication****: Alice obtains* p_B *and Bob obtains* p_A*.*
3. ***Key Establishment****: Alice computes* $\boldsymbol{KE}(gp, p_B, s_A)$ *and Bob computes* $\boldsymbol{KE}(gp, p_A, s_B)$*.*
4. ***Verification****: If applicable, each participant test the validity of the others public key. Alice and Bob verify that they have computed the same shared secret. If they have not, communication is terminated.*

For the verification step to succeed, clearly the key establishment function $\boldsymbol{KE}$ *has the requirement that these two outputs are equal when the participants operate honestly. Additionally, the following values must be computationally infeasible to compute: a secret key from its corresponding public key, a secret key* s *from* $\boldsymbol{KE}(gp, p, s)$*, and* $\boldsymbol{KE}(gp, p_B, s_A)$ *from* gp, p_B*, and* p_A*.*

Note, this protocol is incomplete as it does not state how Alice and Bob check if they computed the same secret in the verification phase. However this step of the protocol will become explicit below, and the security of our choice will be examined in detail. We now formally state and analyze the security of performing multiple simultaneous instances of key agreement. First is the attack model that will be used throughout.

Definition 2. *Consider the attack model on a key agreement protocol where Bob may use a specially chosen public key/private key* (p_B, s_B) *and additionally act dishonestly in the verification phase.*

Following [13, §3] we define a two types of oracles that we will consider Bob having access to once per verification phase:

1. $Oracle_1(p_B) = \boldsymbol{KE}(gp, p_B, s_A)$*, which corresponds to Bob somehow obtaining the output of Alice's key establishment function.*
2. $Oracle_2(p_B, h')$ *returns* 1 *if* $h' = \boldsymbol{KE}(gp, p_B, s_A)$*, and returns* 0 *otherwise, which corresponds to Alice either terminating or continuing a session after she and Bob performed verification in which Bob used some* h' *as his secret.*

Suppose Bob chooses p_B *in such a way that a response from a type* (1) *oracle, or a response of* 1 *from a type* (2) *oracle, will reveal* $\kappa(p_B)$ *bits of Alice's secret key to Bob (where* $\kappa(\cdot)$ *returns non-negative integers). Then the output of* $Oracle_1(p_B)$ *follows some discrete probability distribution (as those* $\kappa(p_B)$ *bits vary); denote the corresponding probability mass function by* $\chi_{KE}(p_B, \cdot)$*. Likewise for the type* (2) *oracle, let* $\chi_{KE}(p_B, h')$ *denote the probability that* $Oracle_2(p_B, h') = 1$*.*

In protocols where these attacks apply, a malicious Bob will typically know the distribution $\chi_{KE}(p_B, \cdot)$ (loosely speaking, if p_B is "close" to the actual public key derived by s_B, then $\mathbf{KE}(gp, p_B, s_A)$ will be "close" to $\mathbf{KE}(gp, p_A, s_B)$). Then Bob can use the values of h' for which $\chi_{KE}(p_B, h') > 0$ and have Alice respond as a type (2) oracle during verification which reveals those $\kappa(p_B)$ bits of her private key when he guesses h' correctly. Our goal is to modify key agreements susceptible to such attacks so that we can bound all probabilities in $\chi_{KE}(\cdot)$ arbitrarily from above. We first need to define a specific type of key agreement protocol.

Definition 3. *Let* $\boldsymbol{KA}$ *be a key agreement protocol which uses the key establishment function* $\boldsymbol{KE}(gp, \cdot, \cdot)$*, for some global parameters* gp*. If Bob has a public key/secret key pair* (p_B, s_B) *for* $\boldsymbol{KA}$ *and is given two public keys* p_1 *and*

*p_2 (derived from some secret keys s_1, s_2 which are unknown to Bob), then $\boldsymbol{KE}(gp, p_B, s_1) = \boldsymbol{KE}(gp, p_1, s_B)$ and $\boldsymbol{KE}(gp, p_B, s_2) = \boldsymbol{KE}(gp, p_2, s_B)$ by requirement of **KE**. A public key which has been altered in any way will be referred to as **modified**. A modified public key p^* that is guaranteed to satisfy:*

1. *p^* passes all validation tests Alice performs in the verification phase,*
2. *$\kappa(p^*) > 0$,*
3. *$\boldsymbol{KE}(gp, p^*, s_1) = \boldsymbol{KE}(gp, p_B, s_1)$, and*
4. *$\boldsymbol{KE}(gp, p^*, s_2) = \boldsymbol{KE}(gp, p_B, s_2)$,*

*will be called **malicious**. If it is computationally infeasible for Bob to modify his public key to some malicious p^* then we will say **KA** is **irreducible**.*

We can now define our key agreement transformation. With the above general framework for a key agreement in mind, consider the following variant.

Definition 4. *Let $\mathbf{KE}$ be a key establishment function as above, let k be a positive integer, and let $\mathbf{H}$ be a preimage resistant hash function. Consider the following key agreement process between Alice and Bob, called $k - \mathbf{KA}$:*

0. ***Setup**: Both members obtain a valid copy of the global parameters, gp.*
1. ***Key Generation**: Alice generates k secret key/public key pairs (s_{Ai}, p_{Ai}), $1 \leq i \leq k$. Likewise Bob generates (s_{Bi}, p_{Bi}) for $1 \leq i \leq k$.*
2. ***Communication**: Alice initiates communication and sends all k of her public keys to Bob. Bob then sends all k of his public keys to Alice.*
3. ***Key Establishment**: Alice computes $z_{i,j} \leftarrow \boldsymbol{KE}(gp, p_{Bi}, s_{Aj})$ for every pair $1 \leq i, j \leq k$, then computes*

   $$h \leftarrow \boldsymbol{H}(z_{1,1}, \ldots, z_{1,k}, z_{2,1}, \ldots, z_{2,k}, \ldots, z_{k,1}, \ldots, z_{k,k}).$$

   Similarly, Bob computes $z'_{i,j} \leftarrow \boldsymbol{KE}(gp, p_{Aj}, s_{Bi})$ for each pair $1 \leq i, j \leq k$, and then computes

   $$h' \leftarrow \boldsymbol{H}(z'_{1,1}, \ldots, z'_{1,k}, z'_{2,1}, \ldots, z'_{2,k}, \ldots, z'_{k,1}, \ldots, z'_{k,k}).$$

4. ***Verification**: If applicable, Alice and Bob test the validity of each others public keys. Alice and Bob verify that h is equal to h' as follows: Alice sends $\mathbf{H}(\mathbf{H}(h))$ to Bob, and Bob responds with $\mathbf{H}(h')$. Alice checks that $\mathbf{H}(h) = \mathbf{H}(h')$ and Bob checks that $\mathbf{H}(\mathbf{H}(h')) = \mathbf{H}(\mathbf{H}(h))$. Either party terminates the session if their verification fails.*

When Alice and Bob perform honestly, it is clear that they will share the same key and verification will pass on both ends. We now present our main theorem which explains how the parameter k can affect the security of the protocol from attacks of the type mentioned in Definition 2.

Theorem 1. *Let* $\boldsymbol{KA}$ *be an irreducible key agreement protocol which uses the key establishment function* $\boldsymbol{KE}(gp, \cdot, \cdot)$*, for some global parameters gp. Let* p^* *be a modified public key with* $\kappa(p^*) > 0$ *that passes all validity tests of* $\boldsymbol{KA}$*, and let* ρ *denote the largest probability in the image of* $\chi_{KE}(p^*, \cdot)$*. Suppose that in* k*-*$\boldsymbol{KA}$ *one of the* k *parts to Bob's public key is* p^**. If Bob has access to a type* (1) *oracle for* k*-*$\boldsymbol{KA}$*, then the largest probability in* $\chi_{k\text{-}KA}(p_B)$ *is* ρ^{k-1}*.*

In k-**KA** Bob has access to a type (2) oracle (see Definition 2) in the form of Alice sending $\mathbf{H}(\mathbf{H}(h))$ (or $\mathbf{H}(h)$ if the roles are reversed) as he can guess at the preimage and check his guess. However we are assuming that Bob has access to a type (1) oracle, that is he somehow recovers h from Alice during verification, which provides the adversary with greater capabilities. We now prove Theorem 1.

Proof. During the k-**KA** session, denote by $(p_{A1}, s_{A1}), \ldots, (p_{Ak}, s_{Ak})$ the keys generated by Alice and likewise $(p_{B2}, s_{B2}), \ldots, (p_{Bk}, s_{Bk})$ the keys generated by Bob, along with $p_{B1} = p^*$ (without loss of generality). Bob can potentially learn about Alice's secret keys during the verification phase. Alice will compute $z_{1,j} \leftarrow \mathbf{KE}(gp, p^*, s_{Aj})$ and $z_{i,j} \leftarrow \mathbf{KE}(gp, p_{Bi}, s_{Aj})$ for every $2 \leq i \leq k$ and $1 \leq j \leq k$. She then computes

$$h \leftarrow \mathbf{H}(z_{1,1}, \ldots, z_{1,k}, z_{2,1}, \ldots, z_{2,k}, \ldots, z_{k,1}, \ldots, z_{k,k}).$$

We are assuming Bob has access to a type (1) oracle, and so he has obtained h from Alice. As $\mathbf{H}$ is preimage resistant, in order to learn anything about Alice's secret keys Bob must guess at the preimage of h. Bob can easily compute $z_{i,j} = \mathbf{KE}(gp, p_{Aj}, s_{Bi})$ for all $2 \leq i \leq k$, $1 \leq j \leq k$. Therefore determining the preimage relies completely on Bob's ability to find $z_{1,j} = \mathbf{KE}(gp, p^*, s_{Aj})$ for every $1 \leq j \leq k$, each of which is an instance of the original **KA** protocol, however he is only able to test a guess for the tuple $(z_{1,1}, \ldots, z_{1,k})$ instead of each one individually. By assumption, **KA** is irreducible and p^* is modified with $\kappa(p^*) > 0$ and passes all applicable validity tests. It follows that $\mathbf{KE}(gp, p_{Aj}, s_{B1})$ is not guaranteed to be equal to $\mathbf{KE}(gp, p^*, s_{Aj}) = z_{1,j}$ for more than one value of j. Bob can therefore be certain of no more than one value of $z_{1,j}$ before testing guesses.

Note that if Bob guesses $(x_1, \ldots, x_k) = (z_{1,1}, \ldots, z_{1,k})$, then the probability of success is unaffected by his previous guesses. Therefore the probability that each of Bob's guesses of $z_{1,j}$ is bounded above by ρ, except for possibly the one value which can be forced to be $\mathbf{KE}(gp, p_{Aj}, s_{B1})$ by Bob's choice of p^*. Since the type (2) oracle only returns 1 if all k instances are correct, Bob's maximum probability of success on any guess is ρ^{k-1}. □

More than the theorem's result, the proof shows that the probability that a guess $(x_1, \ldots, x_k)$ is equal to $(z_{1,1}, \ldots, z_{1,k})$ is the product that each individual x_j is equal to $z_{1,j}$, $1 \leq j \leq k$, with the exclusion of no more than one j by the irreducibility assumption.

3 Multiple Instances of SIDH

In this section we will apply the previous theory to the SIDH key agreement protocol to enable secure use of static keys. We then estimate the expected amount of work required to break our transformation in this case.

3.1 Preliminaries

For general background on elliptic curves we refer the reader to [21]. Throughout, we let E be an elliptic curve over a finite field $\mathbb{F}_q$ and use $[m]P$ to denote applying the multiplication-by-m map to the point P (adding P to itself m times) for any $m \in \mathbb{Z}$. We denote the m-torsion subgroup of E, the subgroup of points $P \in E(\overline{\mathbb{F}}_q)$ such that $[m]P$ is the identity on E, by $E[m]$. If $q = p^n$, then those elliptic curves for which $E[p^r]$ is the trivial subgroup (for all $r \in \mathbb{N}$) are called supersingular elliptic curves. Otherwise $E[p^r] \cong \mathbb{Z}/p^r\mathbb{Z}$ for all $r \in \mathbb{N}$ and such elliptic curves are called ordinary. Supersingular elliptic curves are all defined over $\mathbb{F}_{p^2}$.

Let E' be a second elliptic curve defined over the finite field $\mathbb{F}_q$. An isogeny $\phi : E \to E'$ over $\mathbb{F}_q$ is a non-constant rational map defined over $\mathbb{F}_q$, mapping identity to identity, and is a group homomorphism from $E(\mathbb{F}_q)$ to $E'(\mathbb{F}_q)$ [21, III.4]. The elliptic curves E and E' defined over $\mathbb{F}_q$ are then said to be isogenous over $\mathbb{F}_q$. For each subgroup G of E, there is up to isomorphism a unique isogeny ϕ with domain E and kernel G [21, III.4.12], which we will denote E/G. The degree, $\deg(\phi)$, is its degree as a rational map which is equal to the size of its kernel for separable isogenies. If ϕ has degree ℓ, we will frequently refer to ϕ as an ℓ-isogeny. Every isogeny with $\deg(\phi) > 1$ can be represented uniquely (up to isomorphism) as a composition of prime degree isogenies over $\overline{\mathbb{F}}_q$ [9]. For every isogeny ϕ there exists a dual isogeny $\hat{\phi} : E' \to E$ of equal degree [21, III.6] and it follows that being isogenous over $\mathbb{F}_q$ is an equivalence relation on the set of $\overline{\mathbb{F}}_q$-isomorphism classes of elliptic curves which are defined over $\mathbb{F}_q$. If E is supersingular and $\ell \nmid p$, then E is ℓ-isogenous to $\ell + 1$ supersingular elliptic curves (counting multiplicites).

Associated to each elliptic curve is a j-invariant, and two elliptic curves are isomorphic over $\overline{\mathbb{F}}_q$ if and only if they have the same j-invariant [21, III.1.4]. Therefore we can refer to the $\overline{\mathbb{F}}_q$-isomorphism classes of elliptic curves over $\mathbb{F}_q$ by their j-invariant. If the elliptic curve is represented as $E : y^2 = x^3 + ax + b$ with $a, b \in \mathbb{F}_q$, then

$$j(E) = 1728 \frac{4a^3}{4a^3 + 27b^2} \in \mathbb{F}_q.$$

For any integer $\ell > 0$ with $p \nmid \ell$, the Weil pairing is a bilinear form that we denote by

$$e_\ell : E[\ell] \times E[\ell] \to \mu_\ell,$$

where $\mu_\ell = \{x \in \mathbb{F}_q | x^\ell = 1\}$. The following remark connecting the Weil pairing and isogenies follows immediately from [21, III.8.2].

Remark 1. Let E be an elliptic curve and $R, S \in E[\ell]$ for some positive integer ℓ. If $\phi : E \to E'$ is an isogeny, then

$$e_\ell(\phi(R), \phi(S)) = e_\ell(R, S)^{\deg(\phi)}.$$

3.2 Supersingular Isogeny Diffie-Hellman Key Agreement

We give a simplified overview of the original SIDH key-establishment protocol [10] in the format of §2 and the Galbraith et al. attack [13].

Setup: The global parameters consist of a prime number $p = 2^m 3^n f \pm 1$ where f is 1 or a small prime, a supersingular elliptic curve $E/\mathbb{F}_{p^2}$, and four points $P_A, Q_A, P_B, Q_B \in E(\mathbb{F}_{p^2})$ such that $\langle P_A, Q_A \rangle = E[2^m]$ and $\langle P_B, Q_B \rangle = E[3^n]$.

Key Generation: The key generation function takes in E, p, P_A, Q_A, P_B, Q_B and $r \in \{0, 1\}$. Upon input of $r = 0$, the key generation function computes:

$$\begin{aligned} \alpha &\leftarrow_R \mathbb{Z}/2^m\mathbb{Z}, \\ \phi_A \colon E \to E_A &= E/\langle P_A + [\alpha]Q_A \rangle, \\ (R_A, S_A) &\leftarrow (\phi_A(P_B), \phi_A(Q_B)). \end{aligned}$$

The key generation function then outputs the private key α and the public key (E_A, R_A, S_A). Upon input of 1 the key generation function performs the analogous computations with some $\beta \leftarrow_R \mathbb{Z}/3^n\mathbb{Z}$, and outputs the private key β and the public key (E_B, R_B, S_B). Additionally, to prevent the recently discovered fault attack [22] Alice and Bob each check the order of the points in their own public key. This is efficient since the order of each point is known.

Communication: Bob initiates conversation and sends his public key,

$$(E_B, R_B, S_B),$$

to Alice. Alice then responds with her public key,

$$(E_A, R_A, S_A).$$

Key Establishment: Alice computes

$$E_{BA} = E_B/\langle R_B + [\alpha]S_B \rangle, \quad \text{and } S = j(E_{BA}).$$

Bob computes

$$E_{AB} = E_A/\langle R_A + [\beta]S_A \rangle, \quad \text{and } S' = j(E_{AB}).$$

Verification: Both Alice and Bob perform validation on the public key the received by the other via the methods proposed by Costello et al. [8, §9], verifying that the points have the correct order and are independent. This includes Alice verifying that $\langle R_B, S_B\rangle = E_B[2^m]$ and $e_{2^m}(R_B, S_B) = e_{2^m}(P_A, Q_A)^{3^n}$, and Bob acting *mutatis mutandis.* Additionally, Alice and Bob check that they have computed the same secret key. If any of the tests fail, then the session is terminated. Otherwise they continue communication with $S = S'$ as their shared secret key.

As in Definition 1 this protocol as defined is incomplete since it does not state how Alice and Bob check if they computed the same secret in the verification phase. This step is made explicit when we apply our multiple instances model.

In its original form [10], the key generation phase produces two values, say α_1 and α_2 (not both divisible by 2) as Alice's private key, her isogeny ϕ_A has kernel $\langle[\alpha_1]P_A + [\alpha_2]Q_A\rangle$, and she takes the analogous linear combination during the key establishment phase. However, through a change of variables one can always obtain kernel $\langle P_A + [\alpha]Q_A\rangle$ or $\langle[\alpha]P_A + Q_A\rangle$ since at least one of α_1 or α_2 is invertible modulo 2^m. Throughout the remainder of this work we assume without loss of generality that we fall into the former case (as we stated in our definition of SIDH) because it simplifies our analysis.

The SIDH key-establishment protocol relies on the difficulty of the following computation problem.

Definition 5. *Let E be a supersingular elliptic curve defined over $\mathbb{F}_{p^2}$, with $p = \ell_A^m \ell_B^n f \pm 1$, and let $P_A, Q_A \in E(\mathbb{F}_{p^2})$ be such that $\langle P_A, Q_A\rangle = E[\ell_A^m]$.*

Given an elliptic curve E_A defined over $\mathbb{F}_{p^2}$ which is ℓ_A^m-isogenous to E, the Supersingular Isogeny (SSI) problem *is to find an isogeny over $\mathbb{F}_{p^2}$ of degree ℓ_A^m from E to E_A with a cyclic kernel. Since the isogeny itself can be infeasible to store, a solution to the SSI problem is an integer $\alpha \in \mathbb{Z}/\ell_A^m\mathbb{Z}$ such that $\langle P_A + [\alpha]Q_A\rangle$ is the kernel of the isogeny.*

As mentioned, this Diffie-Hellman type protocol is susceptible to an active attack if Alice uses the same private key in different sessions [13, §3]. We will describe it now. For this discussion we will assume $\ell_A = 2$ and $\ell_B = 3$, a similar attack applies when this is not the case.

Instead of using the public key $(E_B, \phi_B(P_A), \phi_B(Q_A))$ when communicating with Alice, a dishonest Bob can send

$$(E_B, R, S) = (E_B, [\theta]\phi_B(P_A), [\theta](\phi_B(Q_A) + [2^{m-1}]\phi_B(P_A))),$$

where θ is chosen such that $e_{2^m}(R, S) = e_{2^m}(P_A, Q_A)^{3^n}$. This modified public key is certain to pass the validation methods in [8, §9]. The parity of Alice's private key α can then be determined as follows. The subgroup computed by Alice during key establishment is $\langle R + [\alpha]S\rangle$. When α is even this subgroup is equal to $\langle \phi_B(P_A) + [\alpha]\phi_B(Q_A) \rangle$, but the subgroup will be different when α is odd. Therefore, if Bob performs his half of the key establishment honestly and uses the shared secret key $E_A / \langle \phi_A(P_B) + [\beta]\phi_A(Q_B) \rangle$ during verification, then he can determine the parity of α based on Alice terminating the session or not. This attack can be extended adaptively to learn each bit of α efficiently and without detection when using the described validation methods. An indirect validation technique [17] is available which prevents the attack, but at the cost of Bob revealing his private key so that Alice can verify the message he sends was computed honestly.

This active attack suggests that static keys can no longer be used for SIDH key exchange unless the other party is using an ephemeral key. In addition, it requires that all holders of static keys must double their computational costs, recomputing the other participant's message in order to verify the validity of the message.

3.3 *k*-SIDH Key Agreement Protocol

We now apply the multiple instances model of §2 to create a k-**KA** scheme based on supersingular isogenies. For the security proof of Theorem 1 to apply we need to show that SIDH is irreducible as defined in Definition 3. We first address the case where a malicious Bob scales his public torsion points by some invertible element.

Lemma 1. *Suppose Alice and Bob participate in an instance of SIDH key-agreement and that Bob uses the dishonest public key*

$$p^* = (E_B, [\mu]\phi_B(P_A), [\mu]\phi_B(Q_A))$$

for some μ coprime to order of P_A and Q_A. Then p^* is not a malicious key in the sense of Definition 3.

Proof. Denote the order of Alice's torsion subgroup by ℓ_A^m and Bob's by ℓ_B^n. The verification phase of SIDH consists of checking that the two torsion points are independent, have the correct order, satisfy the Weil pairing condition, and that both parties compute the same shared secret key. The order and independence conditions follow immediately from the assumption that ℓ_A and μ are coprime. By Remark 1 and the bilinearity of the Weil pairing,

$$e_{\ell_A^m}([\mu]\phi_B(P_A), [\mu]\phi_B(Q_A)) = e_{\ell_A^m}(P_A, Q_A)^{\mu^2 \ell_B^n}.$$

Therefore p^* passes the Weil pairing test if and only if $\mu^2 \equiv 1 \mod \ell_A^m$. Lastly, if we denote Alice's private key by α, then

$$\begin{aligned}\langle [\mu]\phi_B(P_A) + [\alpha]\left([\mu]\phi_B(Q_A)\right) \rangle &= \langle [\mu]\left(\phi_B(P_A) + [\alpha]\phi_B(Q_A)\right) \rangle \\ &= \langle \phi_B(P_A) + [\alpha]\phi_B(Q_A) \rangle,\end{aligned}$$

where the second equality follows from μ being coprime to ℓ_A.

This shows that if Bob modifies his public key in this way, then Alice will compute the same shared secret independent of her private key. Therefore no more information about her private key can be leaked by Alice accepting (or rejecting if $\mu^2 \not\equiv 1$) than is already leaked when Bob performs honestly. Hence, $\kappa(p^*) = 0$ and this modification does not result in a malicious public key. $\square$

It is worth noting that if Bob scales his two torsion points by different scalers, say μ_1 and μ_2, then they will no longer generate the same subgroup under Alice's private key by the independence of $\phi_B(P_A)$ and $\phi_B(Q_A)$, again resulting in a public key which is not malicious. Now we can prove that isogenies lend themselves to the transform of §2.

Theorem 2. *Under the assumption that the SSI problem is intractable, it is computationally infeasible for a malicious Bob with non-negligible probability to modify his public key* $(E_B, \phi_B(P_A), \phi_B(Q_A))$ *to some* $p^* = (E_B, R, S)$ *which is malicious for SIDH.*

Proof. Let $p = \ell_A^m \ell_B^n f \pm 1$ be prime, let E be an elliptic curve defined over $\mathbb{F}_{p^2}$, and let P_A, Q_A, P_B and Q_B be points on $E(\mathbb{F}_{p^2})$ such that $\langle P_A, Q_A \rangle = E[\ell_A^m]$ and $\langle P_B, Q_B \rangle = E[\ell_B^n]$. Alice has some public key/secret key pair

$$\phi_{A1} : E \to E_{A1} = E/\langle P_A + [\alpha_1]Q_A \rangle,\ \alpha_1 \in \mathbb{Z}/\ell_A^m\mathbb{Z}.$$

Bob knows the global parameters p, P_A, Q_A, P_B and Q_B, and receives the public key $(E_{A1}, \phi_{A1}(P_B), \phi_{A1}(Q_B))$ from Alice. By the assumption of intractability of the SSI problem, it should be infeasible for Bob to compute α_1. The goal of our proof is to show that if Bob can violate the definition of irreducibility by computing p^* in the statement of the theorem, then he can compute α_1 efficiently which violates the SSI assumption.

Bob uses the SIDH key generation algorithm twice, to generate some

$$\alpha_2 \in \mathbb{Z}/\ell_A^m\mathbb{Z},\ \phi_{A2} : E \to E_{A2} = E/\langle P_A + [\alpha_2]Q_A \rangle, \text{ and}$$

$$\beta \in \mathbb{Z}/\ell_B^n\mathbb{Z},\ \phi_B : E \to E_B = E/\langle P_B + [\beta]Q_B \rangle.$$

Suppose for contradiction that Bob is able to modify $(E_B, \phi_B(P_A), \phi_B(Q_A))$ to some malicious public key (E_B, R, S), violating irreducibility as stated in Definition 3. That is:

- (E_B, R, S) passes all validation tests,
- $j(E_B/\langle R + [\alpha_1]S \rangle) = j(E_B/\langle \phi_B(P_A) + [\alpha_1]\phi_B(Q_A) \rangle)$,
- $j(E_B/\langle R + [\alpha_2]S \rangle) = j(E_B/\langle \phi_B(P_A) + [\alpha_2]\phi_B(Q_A) \rangle)$, and
- $\kappa(E_B, R, S) > 0$.

Since we cannot fully characterize public keys with $\kappa(p^*) > 0$ in this setting, we instead use the condition that $(R, S) \neq ([\mu]\phi_B(P_A), [\mu]\phi_B(Q_A))$ for some μ coprime to ℓ_A. By Lemma 1 these public keys satisfy $\kappa(p^*) = 0$, so we are assuming a potentially weaker condition than $\kappa(p^*) > 0$ by excluding only public keys of this type.

To simplify notation for the remainder of this proof we set $\ell = \ell_A$. The subgroups $\langle R + [\alpha_1]S\rangle$ and $\langle R + [\alpha_2]S\rangle$ are guaranteed to be kernels of isogenies from E to elliptic curves isomorphic to E_{A1} and E_{A2} respectively by the j-invariant requirements. For the first subgroup one of two cases is true:

i The isogeny with kernel $\langle R + [\alpha_1]S\rangle$ is isomorphic to the isogeny with kernel $\langle \phi_B(P_A) + [\alpha_1]\phi_B(Q_A)\rangle$,
ii The isogeny with kernel $\langle R + [\alpha_1]S\rangle$ is not isomorphic to the isogeny with kernel $\langle \phi_B(P_A) + [\alpha_1]\phi_B(Q_A)\rangle$.

Likewise, there are two cases for α_2 and the isogeny to E_{A2}. For the remainder of the proof we assume that both isogenies fall into case (i) as our reduction only applies in this situation. This point will be examined in greater detail in the run-time analysis at the end of the proof. This distinction of cases must be made as it is possible for the two isogenies to be non-isomorphic and yet the torsion points R and S (or some scaling of them) still satisfy all the requirements of the verification phase, including the Weil pairing test that $e_{\ell^m}(R, S) = e_{\ell^m}(P_A, Q_A)^{\ell_B^n}$ (see [13, §3.2] for details).

Suppose the isogeny with kernel $\langle \phi_B(P_A)+[\alpha_i]\phi_B(Q_A)\rangle$ is isomorphic to that of $\langle R + [\alpha_i]S\rangle$ for both $i \in \{1, 2\}$. Then the two subgroups themselves are equal for each i. It follows that their generators must then differ by a scalar multiple coprime to the order of the subgroup. We can then write

$$[\lambda_i](\phi_B(P_A) + [\alpha_i]\phi_B(Q_A)) = R + [\alpha_i]S, \tag{1}$$

for some $\lambda_i \in \mathbb{Z}/\ell^m\mathbb{Z}$ coprime to ℓ^m (i.e. coprime with ℓ), for both $i \in \{1, 2\}$.

Since ℓ is a small prime, the elliptic curve discrete log problem is tractable on $E_B[\ell^m]$ using Pohlig-Hellman [20] and the Weil or Tate pairing (see [2, §3.2] and optimization [7, §4-5]). Solving two instances of the two-dimensional ECDLP provides $a, b, c, d \in \mathbb{Z}/\ell^m\mathbb{Z}$ such that

$$R = [a]\phi_B(P_A) + [b]\phi_B(Q_A), \text{ and } S = [c]\phi_B(P_A) + [d]\phi_B(Q_A). \tag{2}$$

Substituting these decompositions into (1) and rearranging we obtain

$$[\lambda_1](\phi_B(P_A) + [\alpha_1]\phi_B(Q_A)) = [a + \alpha_1 c]\phi_B(P_A) + [b + \alpha_1 d]\phi_B(Q_A).$$

The points P_A and Q_A are independent—there does not exist $t \in \mathbb{Z}/\ell^m\mathbb{Z}$ such that $P_A = [t]Q_A$. Therefore $\phi_B(P_A)$ and $\phi_B(Q_A)$ are independent as well. Comparing coefficients of $\phi_B(P_A)$ implies that $\lambda_1 \equiv a + \alpha_1 c \bmod \ell^m$. Comparing coefficients of $\phi_B(Q_A)$ then gives the congruence

$$b + \alpha_1 d \equiv \lambda_1\alpha_1 \equiv (a + \alpha_1 c)\alpha_1 \bmod \ell^m. \tag{3}$$

Similar analysis of the subgroups associated with α_2 result in the congruence

$$b + \alpha_2 d \equiv (a + \alpha_2 c)\alpha_2 \bmod \ell^m. \tag{4}$$

Rearranging (3) and (4) gives

$$c\alpha_1^2 + (a-d)\alpha_1 - b \equiv 0 \bmod \ell^m, \text{ and } c\alpha_2^2 + (a-d)\alpha_2 - b \equiv 0 \bmod \ell^m.$$

Therefore α_1 and α_2 are solutions to the quadratic congruence relation

$$cx^2 + (a-d)x - b \equiv 0 \bmod \ell^m. \tag{5}$$

Bob has the ability to construct this polynomial. One approach to solving this equation comes from the assumption that α_1 and α_2 are simple roots modulo ℓ (this is the same assumption required in Hensel's lemma) as it implies $\alpha_1 - \alpha_2$ is invertible modulo ℓ^m. By subtracting (4) from (3) and multiplying the result by $(\alpha_1 - \alpha_2)^{-1} \mod \ell^m$ we obtain

$$d \equiv a + c(\alpha_1 + \alpha_2) \mod \ell^m, \tag{6}$$

and it follows that

$$b \equiv -c\alpha_1\alpha_2 \mod \ell^m. \tag{7}$$

Therefore, if $\ell^r \mid c$, then $\ell^r \mid a-d$ and $\ell^r \mid b$ too. If $c \equiv 0 \mod \ell^m$, then $b \equiv 0$ and $a \equiv d \mod \ell^m$, which contradicts the assumption that (E_B, R, S) is malicious by Lemma 1.

From the malicious public key (E_B, R, S), Bob can now efficiently solve for α_1 and α_2 using the following process:

1 Compute the discrete log coefficients $a, b, c, d \in \mathbb{Z}/\ell^m\mathbb{Z}$ as above.
2 Write $c = \ell^r g$ for some g indivisible by ℓ and $0 \le r < m$.
3 Let $K = g^{-1}\dfrac{a-d}{\ell^r} \mod \ell^{m-r}$ and $L = -g^{-1}\dfrac{b}{\ell^r} \mod \ell^{m-r}$, where the inverse of g is computed modulo ℓ^{m-r}.
4 α_1 and α_2 are roots of the quadratic $x^2 + Kx + L \equiv 0 \bmod \ell^{m-r}$ by (5). Solve for all roots of this polynomial modulo ℓ^{m-r}.
5 For each root, u, extend it to an integer mod ℓ^m, say u', and test if it is equal to α_1. This test can be performed by computing the image curve of the isogeny with $\langle P_A + [u']Q_A \rangle \subset E(\mathbb{F}_{p^2})$ as its kernel and comparing its j-invariant with $j(E_{A1})$ (the image curve of the isogeny with $\langle P_A + [\alpha_1]Q_A \rangle$ as its kernel).

What remains is to analyze the computational cost of this reduction and the probability of success. For this analysis, we need to know the likelihood of our assumptions, the probable size of the value r, and the number of roots of the quadratic congruence.

The first assumption is that the subgroup associated to the points R and S is the same as the isogeny kernel in the SIDH instance. The existence of multiple isogenies of degree ℓ^m between two fixed supersingular elliptic curves is possible, but unlikely under the Galbraith et al. heuristic of [13, §4.2]. For instance there can exist multiple isogenies of degree ℓ from one j-invariant, j_0, to another and this occurs exactly when the classical modular polynomial $\mathbf{\Phi}_\ell(j_0, x)$ has repeated roots in x. The set of possible roots grows with p and yet its degree in x is fixed by $\ell + 1$, so this situation unlikely for large p.

Next we examine the value r when $\alpha_1 \equiv \alpha_2 \mod \ell$. When $\ell = 2$, we have that $r \geq 3$ whenever $m > 3$. From the distribution of multiples of ℓ in $\mathbb{Z}/2^m\mathbb{Z}$, we have $r = j$ for $3 \leq j < m$ with probability $\frac{1}{2^{j-2}}$, and the probability that $r = m$ (i.e. $c = 0$) is $\frac{1}{2^{m-3}}$. When ℓ is odd, only $r \geq 1$ is guaranteed. For $\mathbb{Z}/\ell^m\mathbb{Z}$ with odd ℓ, we have $r = j$ for $1 \leq j < m$ with probability $\frac{1}{\ell^j}$, and the probability that $r = m$ is $\frac{1}{2^{m-1}}$. Hence, it is most likely that $r = 3$ or 4 when $\ell = 2$, and $r = 1$ or 2 when ℓ is an odd prime.

Lastly, we look at the number of solutions to (5). If $\alpha_1 \not\equiv \alpha_2 \bmod \ell$ and $\ell \nmid c$, then there are exactly two solutions modulo ℓ^m, namely α_1 and α_2. Letting r be the ℓ-adic valuation of c as above, the number of solutions to this quadratic congruence is $2\ell^r$, namely

$$\alpha_i + z\ell^{m-r-1},\ 0 \leq z \leq \ell^r - 1,\ i \in \{1, 2\}.$$

Even though the number of roots to check grows exponentially in r, the probability of each successive value of r occurring decreases exponentially (see the previous paragraph).

When α_2 is chosen to be congruent to α_1 modulo ℓ, b and d are not necessarily of the form (6) and (7). This makes solving for α_1 much harder, and in some cases, impossible. However, this only happens with probability $\frac{1}{\ell}$. By the previous paragraph we see that if Bob counts the number of roots of (5) modulo ℓ^{m-r} before solving for α_1, then verifying there are less than ℓ^{r+1} of them can serve to test for when $\alpha_1 \equiv \alpha_2 \mod \ell$. If the test fails then Bob can reuse the key generation algorithm until the private key provided is incongruent to the initial α_2, and then repeat the process above (he never has to run this process more than twice).

We conclude that if Bob can violate this irreducibility condition, then he can efficiently solve the SSI problem. □

Combining Theorem 2 with the fact that there are currently no know attacks on SIDH that involve modified elliptic curves (as opposed to modified torsion points) we conclude that SIDH is irreducible for all known modified public keys. We now give an explicit statement of the k-SIDH protocol.

Setup: A preimage resistant hash function H, a prime number $p = 2^m 3^n f \pm 1$, a supersingular elliptic curve $E/\mathbb{F}_{p^2}$, and four points $P_A, Q_A, P_B, Q_B \in E(\mathbb{F}_{p^2})$ such that $\langle P_A, Q_A \rangle = E[2^m]$ and $\langle P_B, Q_B \rangle = E[3^n]$.

Key Generation: Upon input of 0, the key generation function computes, for $1 \leq i \leq k$:

$$\begin{aligned} \alpha_i &\leftarrow_R \mathbb{Z}/2^m\mathbb{Z}, \\ \phi_{Ai} \colon E \to E_{Ai} &= E/\langle P_A + [\alpha_i]Q_A \rangle, \\ (R_i, S_i) &\leftarrow (\phi_{Ai}(P_B), \phi_{Ai}(Q_B)). \end{aligned}$$

The key generation function then outputs the private key $(\alpha_1, \ldots, \alpha_k)$ and the public key $(E_{A1}, R_1, S_1), \ldots, (E_{Ak}, R_k, S_k)$. The recipient checks that the order of each R_i and S_i is 3^n to ensure no faults were induced.

Upon input of 1 the key generation function computes, for $1 \leq j \leq k$:

$$\begin{aligned} \beta_j &\leftarrow_R \mathbb{Z}/3^n\mathbb{Z}, \\ \phi_{Bj} : E \to E_{Bi} &= E/\langle P_B + [\beta_j]Q_B \rangle, \\ (U_j, V_j) &\leftarrow (\phi_{Bj}(P_A), \phi_{Bj}(Q_A)). \end{aligned}$$

The key generation function then outputs the private key $(\beta_1, \ldots, \beta_k)$ and the public key $(E_{B_1}, U_1, V_1), \ldots, (E_{B_k}, U_k, V_k)$. The recipient checks that the order of each U_j and V_j is 2^m to ensure no faults were induced.

Communication: Bob initiates conversation and sends his public key to Alice. Alice responds with her public key.

Key Establishment: For each $1 \leq i, j \leq k$, Alice computes

$$z_{i,j} = j(E_{Bj}/\langle U_j + [\alpha_i]V_j \rangle),$$

and then she calculates the hash

$$h = H(z_{1,1}, \ldots, z_{1,k}, z_{2,1}, \ldots, z_{2,k}, \ldots, z_{k,1}, \ldots, z_{k,k}).$$

Similarly, for each $1 \leq i, j \leq k$, Bob computes

$$z'_{i,j} = j(E_{Ai}/\langle R_i + [\beta_j]S_i \rangle),$$

and calculates the hash

$$h' = H(z'_{1,1}, \ldots, z'_{1,k}, z'_{2,1}, \ldots, z'_{2,k}, \ldots, z'_{k,1}, \ldots, z'_{k,k}).$$

Verification: Alice verifies that for each $1 \leq j \leq k$ the pair U_j and V_j are independent points of order 2^m on the curve E_{Bj} [8, §9]. Additionally Alice verifies that $e_{2^m}(U_j, V_j) = e_{2^m}(P_A, Q_A)^{3^n}$. Likewise Bob verifies that each pair R_i and S_i are independent points of order 3^n on the curve E_{Ai} and that $e_{3^n}(R_i, S_i) = e_{3^n}(P_B, Q_B)^{2^m}$. Alice sends $H(H(h))$ to Bob who verifies it is equal to $H(H(h'))$. Bob sends $H(h')$ to Alice who verifies it is equal to $H(h)$. If they have different secret keys, or any of the public key pairs fail the verification, then the session is terminated. Otherwise they continue communication with $h = h'$ as their shared secret key.

3.4 Security Analysis and Key Size

Before the security of k-SIDH can be properly analyzed we need the following simple result. As before, let $\ell = \ell_A$ denote the prime defining Alice's torsion subgroup. Recall that an ℓ^m-degree isogeny can be expressed uniquely as a composition of m ℓ-degree isogenies. The following result tells us that, given the shared j-invariant and Alice's public key, Bob is unable to determine the final ℓ-isogeny in the composition of Alice's isogeny (under the SSI assumption).

Theorem 3. *Suppose Alice and Bob perform the standard SIDH key-agreement protocol as described §3.2. In the key establishment phase Alice computes a secret isogeny $\phi_A : E_B \to E_{BA}$ of degree ℓ^m ($\ell \in \{2, 3\}$) as the composition of m isogenies of degree ℓ, say $\phi_A = \phi_m \circ \cdots \circ \phi_1$. Let $\phi' = \phi_{m-1} \circ \cdots \circ \phi_1$ be the isogeny whose image curve is ℓ isogenous to E_{AB}, say $\phi' : E_B \to E'$. Bob also knows the curve E_{BA} by performing his half of the key establishment. If Bob has access to an efficient, deterministic algorithm which produces E' from E, E_A, E_B, E_{BA} and ℓ^m, then Bob can efficiently solve the SSI problem.*

Proof. The elliptic curve E' is ℓ isogenous to E_{BA}. Given E' Bob can then determine ϕ_m as there are only $\ell + 1$ choices which he can test exhaustively. Repeated iterations of the procedure, replacing the target curve adaptively and decreasing the exponent of the degree iteratively by 1, will return each ℓ-isogeny in the composition. This procedure will reveal ϕ_A, breaking SSI. □

The security of this scheme is based on the amount of work Bob must do in the proof of Theorem 1 to compute the preimage of h. There are two benchmarks that we could use when choosing k: the expected number of hashes Bob will compute before correctly hashing the preimage, or the number of hashes before the solution is found with probability $\frac{1}{2}$. The former is asymptotically greater in our case, and so it is irrelevant when setting a security level.

The runtime depends on the order Bob guesses at solutions, so we always assume he does so optimally. We index Bob's guesses by i, and denote the associated probability of success by P_i. The proof of Theorem 1 shows that for Bob to determine the preimage of h he must correctly guess at least $k-1$ independent samples from some distribution. We now determine that distribution for SIDH.

In the attack of Galbraith et al. [13], the public key with the greatest ratio of revealed bits of Alice's private key to probability of success that Bob could use is $p^* = (E_B, \phi_B(P_A), \phi_B(Q_A) + [\ell^{m-1}]\phi_B(P_A))$. Bob knows the shared key that Alice computes were he to participate honestly,

$$j_0 = j(E_A/\langle \phi_A(P_B) + [\beta]\phi_B(Q_A) \rangle),$$

and when using this dishonest p^* he knows Alice will compute either j_0 or some other j-invariant which is ℓ^2-isogenous to j_0. With overwhelming probability there are $\ell(\ell + 1)$ distinct isomorphism families which are ℓ^2-isogenous to any isomorphism family (not represented by the j-invariant 0 or 1728). Combining this fact with the Theorem 3 shows that k-SIDH exhibits the following probability distribution for each of Bob's k guesses:

$$\{\frac{1}{2}, \frac{1}{2\ell(\ell+1)}, \dots, \frac{1}{2\ell(\ell+1)}\},$$

where $\dfrac{1}{2\ell(\ell+1)}$ occurs $\ell(\ell+1)$ times. For example, if $\ell = 2$, then the honestly computed j-invariant, j_0, occurs with probability $\frac{1}{2}$, and there are six j-invariants which are 4-isogenous to j_0 each occurring with probability $\frac{1}{12}$.

The guess that maximizes Bob's probability of success is j_0 for each of the $k-1$ unknown values, resulting in $P_1 = \dfrac{1}{2^{k-1}}$. The next most likely outcomes are those with j_0 for $k-2$ of the values and one of the other $\ell(\ell+1)$ j-invariants, each occurring with probability

$$P_i = \frac{1}{2^{k-2} \cdot (2\ell(\ell+1))} \text{ for } 2 \le i \le (k-1)(\ell(\ell+1)) + 1.$$

From this we calculate r, the number of hashes that Bob computes before his probability of success is $\frac{1}{2}$ by solving $\dfrac{1}{2} = \sum_{i=1}^{r} P_i$.

The first step is to collect all guesses with the same probability of success, that is, those which select the same number of j_0. To achieve this we change from the variable r, the total number of guesses Bob makes, to t which represents the quantity of non-j_0 elements in Bob's choice. They are related by

$$r = \sum_{i=0}^{t} \binom{k-1}{i} (\ell(\ell+1))^i$$

as each term in the summand is the number of possibilities with i non-j_0 elements. Therefore,

$$\sum_{i=1}^{r} P_i = \sum_{i=0}^{t} \frac{1}{2^{k-1-i}(2\ell(\ell+1))^i} \binom{k-1}{i} (\ell(\ell+1))^i = \frac{1}{2^{k-1}} \sum_{i=0}^{t} \binom{k-1}{i},$$

and this final sum equals $\frac{1}{2}$ exactly when $t = \frac{k-2}{2}$ (if k is odd then the sum needs one half times the $\binom{k-1}{\frac{k-1}{2}}$ term) by the symmetry of the binomial coefficient. This implies that the number of hashes required by Bob to learn the first bit of each of Alice's k secret keys is

$$r = \sum_{i=0}^{\frac{k-2}{2}} \binom{k-1}{i} (\ell(\ell+1))^i. \tag{8}$$

If $\ell = 2$, then $k = 60$ gives $r = 2^{130}$; for $\ell = 3$, 2^{131} hashes is achieved by $k = 50$.

When considering security against a quantum enabled adversary, one would expect a quadratic speedup because the runtime of Grover's algorithm [14] on a non-uniform distribution is still $O(\sqrt{N})$ when searching for one item, where N is the size of the domain [4]. The domain for k-SIDH when Bob uses a malicious public key is all possible preimages to Alice's hash. Considering an initial state of each possible preimage (where each preimage is a collection of qubits representing the associated j-invariants) all with amplitude $\frac{1}{\sqrt{(2(\ell(\ell+1))^{k-1})}}$, and searching for an element of the marked set (the collection of qubits corresponding to the correct preimage) gives a quantum algorithm with runtime $\frac{\pi}{4}2^{\frac{k-1}{2}}(\ell(\ell+1))^{\frac{k-1}{4}}$ and requires at least $(2(\ell(\ell+1))^{k-1}$ qubits. We then calculate the minimal k such that Bob is required to compute 2^{128} quantum operations before successfully calculating the preimage of Alice's hash, which would reveal k bits of her secret key. Setting $k = 113$ is required when $\ell = 2$, and $k = 94$ suffices when $\ell = 3$.

These choices of k are based on the currently best known attack that satisfy Definition 2. There is the possibility that other attacks will be discovered such as modifying the elliptic curve in a public key instead of the torsion points, or perhaps stronger attacks using modified torsion points. However, if such attacks are discovered, the generality of the Theorem 1 shows that only a recalculation of k is needed to adapt as these qualify as malicious public key attacks.

To achieve a specified level of security for k-SIDH each individual SIDH instance must also meet that security level. Using the compression techniques of [7, §7], at the 128-bit quantum security (or 192-bit classical security) level a k-SIDH pubic key requires 37 kb when $\ell = 2$ and 31 kb when $\ell = 3$.

3.5 Other Applicable Post-Quantum Schemes and Future Work

The proposed k-instances model applies to key agreement schemes in which the resulting shared secret is dependent on input from both parties (not encapsulation methods) where the use of static keys may reveal private keys to a malicious participant. We have seen that this applies to SIDH [13], but it may also apply to lattice based schemes. The ring-LWE key agreement protocol by Ding et al. [16] satisfies this criterion as it is susceptible to such an active attack [11]. However, one would have to show that this protocol is irreducible (Definition 3).

The computational costs of k-SIDH are naively k^2 that of standard SIDH, in which parties simply perform k^2 independent SIDH operations. Economies of scale could be realized in an optimized implementation using (for example) SIMD, since the key establishments can be organized into k groups of k such that all SIDH operations in a group have one half in common.

k-SIDH only addresses the problem of dishonest users who manipulate elliptic curve points. It does not address the case where the curves themselves are manipulated. It may be worth examining whether approaches like k-SIDH can help protect against attacks involving manipulated curves. Another interesting problem comes from the heuristic assumption from [13, §4.2] which was used in the proof of Theorem 2. Although this assumption seems plausible in light of the Ramanujan property of the supersingular ℓ-isogeny graph, a proof would be preferable, perhaps under a standard assumption such as GRH. Similar results have been achieved in the ordinary case [15].

4 Conclusion

We presented a new key agreement model which performs k^2 simultaneous key agreements and defends against a specific class of active adversaries when certain assumptions about the underlying key agreement protocol are satisfied. We showed that supersingular isogeny key agreement satisfies these assumptions provided its computational problem is intractable. Using this new model, we determined that performing $60 \cdot 50 = 3000$ simultaneous instances of SIDH will protect both participants from leaking any information of their secret key against these active adversaries with classical capabilities, and $113 \cdot 94 = 10622$ suffices for protection against quantum adversaries.

Acknowledgments. The authors would like to thank the reviewers for their comments. This work is supported in parts by the grants NIST-60NANB17D184, NIST-60NANB16D246, and NSF CNS-1661557. This work is also partially supported by NSERC, CryptoWorks21, and Public Works and Government Services Canada.

References

1. Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange—a new hope. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 327–343, Austin, TX, 2016. USENIX Association.
2. Reza Azarderakhsh, David Jao, Kassem Kalach, Brian Koziel, and Christopher Leonardi. Key compression for isogeny-based cryptosystems. In *Proceedings of the 3rd ACM International Workshop on ASIA Public-Key Cryptography*, AsiaPKC '16, pages 1–10. ACM, 2016.
3. Daniel J. Bernstein, Nadia Heninger, Paul Lou, and Luke Valenta. Post-quantum RSA. Cryptology ePrint Archive, Report 2017/351, 2017. http://eprint.iacr.org/2017/351.
4. David Biron, Ofer Biham, Eli Biham, Markus Grassl, and Daniel A. Lidar. Generalized Grover search algorithm for arbitrary initial amplitude distribution. In *1st NASA Conference on Quantum Computing and Quantum Communications*, 1998.
5. Joppe Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! practical, quantum-secure key exchange from LWE. In *Proc. 23rd ACM Conference on Computer and Communications Security (CCS) 2016*. ACM, October 2016.
6. Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, and Damien Stehlé. CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM. Cryptology ePrint Archive, Report 2017/634, 2017.

7. Craig Costello, David Jao, Patrick Longa, Michael Naehrig, Joost Renes, and David Urbanik. Efficient compression of SIDH public keys. Cryptology ePrint Archive, Report 2016/963, 2016.
8. Craig Costello, Patrick Longa, and Michael Naehrig. *Efficient Algorithms for Supersingular Isogeny Diffie-Hellman*, pages 572–601. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.
9. Jean-Marc Couveignes. Hard homogenous spaces. http://eprint.iacr.org/2006/291/, 2006.
10. Luca De Feo, David Jao, and Jérôme Plût. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. *Journal of Mathematical Cryptology*, 8(3):209–247, 2014.
11. Scott Fluhrer. Cryptanalysis of ring-LWE based key exchange with key share reuse. Cryptology ePrint Archive, Report 2016/085, 2016. http://eprint.iacr.org/2016/085.
12. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO '99: 19th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 1999, Proceedings*, pages 537–554, 1999.
13. Steven D. Galbraith, Christophe Petit, Barak Shani, and Yan Bo Ti. On the security of supersingular isogeny cryptosystems. Cryptology ePrint Archive, Report 2016/859, 2016. http://eprint.iacr.org/2016/859.
14. Lov K. Grover. A fast quantum mechanical algorithm for database search. *In Proc. ACM STOC*, 1996.
15. David Jao, Stephen D. Miller, and Ramarathnam Venkatesan. Expander graphs based on GRH with an application to elliptic curve cryptography. *J. Number Theory*, 129(6):1491–1504, 2009.
16. Xiaodong Lin Jintai Ding, Xiang Xie. A simple provably secure key exchange scheme based on the learning with errors problem. Cryptology ePrint Archive, Report 2012/688, 2012. http://eprint.iacr.org/2012/688.
17. Daniel Kirkwood, Bradley C. Lackey, John McVey, Mark Motley, Jerome A. Solinas, and David Tuller. Failure is not an option: Standardization issues for post-quantum key agreement. *Workshop on Cybersecurity in a Post-Quantum World*, 2015.
18. Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo S. L. M. Barreto. MDPC—McEliece: New McEliece Variants from Moderate Density Parity-Check Codes. In *IEEE International Symposium on Information Theory - ISIT 2013*, pages 2069–2073, 2013.
19. Chris Peikert. *Lattice Cryptography for the Internet*, pages 197–219. Springer International Publishing, 2014.
20. S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. In *IEEE Transactions on Information Theory*, volume 24, pages 106–110, 1978.
21. J. Silverman. *The Arithmetic of Elliptic Curves*, volume 106 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2 edition, 2009.
22. Yan Bo Ti. Fault attack on supersingular isogeny cryptosystems. Cryptology ePrint Archive, Report 2017/379, 2017. http://eprint.iacr.org/2017/379.
23. Ingo von Maurich, Lukas Heberle, and Tim Güneysu. *IND-CCA Secure Hybrid Encryption from QC-MDPC Niederreiter*, pages 1–17. Springer International Publishing, 2016.