Authenticated key exchange for SIDH

Steven D. Galbraith

Mathematics Department, University of Auckland, NZ. s.galbraith@auckland.ac.nz

March 13, 2018

Abstract. We survey authenticated key exchange (AKE) in the context of supersingular isogeny Diffie-Hellman key exchange (SIDH). We discuss different approaches to achieve authenticated key exchange, and survey the literature. We explain some challenges that arise in the SIDH setting if one wants to do a "Diffie-Hellman-like" AKE, and present several candidate authenticated key exchange protocols suitable for SIDH. We also discuss some open problems.

1 Introduction

A fundamental operation in many security systems is to establish an ephemeral shared key between two parties. This can be done using trusted third parties, using symmetric crypto, using an interactive protocol like Diffie-Hellman key exchange, or using public key encryption as key transport. In this paper we are interested in schemes that are based on computational assumptions in mathematics and so are in the public key crypto setting. Basic Diffie-Hellman key exchange is vulnerable to man-in-the-middle attacks, and so it is necessary to provide some form of authentication in most applications.

Supersingular isogeny Diffie-Hellman key exchange (SIDH) allows two parties to generate a shared random key. The security relies on computational problems regarding the computation of isogenies between elliptic curves, and these problems are believed to be hard for quantum computers. Hence SIDH is a candidate for post-quantum crypto.

The basic SIDH scheme is an analogue of the Diffie-Hellman key exchange concept and so is vulnerable to man-in-the-middle attacks. A natural problem is to design an authenticated key exchange (AKE) scheme from the basic SIDH primitive.

To our knowledge there are few papers in the literature that discuss AKE based on supersingular isogeny problems[1]. The thesis of LeGrow [28] is the most extensive document that we are aware of. We also mention the work of Kirkwood et al [23]. They give a scheme where Bob has a public key that enables Alice to share a key with Bob. Alice is certain that only Bob can complete the protocol and compute the key, and Bob is certain that his partner Alice is behaving honestly. But Bob has no assurance about the identity of his partner and so this does not satisfy the definition of AKE. Similarly, using public key encryption for key transport (or a KEM) only provides authentication in one direction.

A standard solution to authenticated key exchange is to use digital signatures (i.e., both players sign their protocol messages with respect to their long-term public keys) or MACs.[2] One can get a secure AKE scheme by combining SIDH with a post-quantum secure signature scheme. Unfortunately, there is no practical digital signature scheme whose security relies on computing isogenies [14, 37]. Hence, if we wish to have an efficient system whose security relies entirely on isogeny problems, it is necessary to consider key exchange schemes that provide implicit authentication.

Another approach is to build authenticated key exchange from public key encryption. There are a large number of papers on this topic and we survey them in Section 4 and discuss isogeny variants in Section 5.

As we discuss in Section 3, there are a number of direct constructions of AKE schemes in the literature in the case of discrete logarithms. However there are several challenges that arise in the SIDH context that do not arise in the discrete logarithms context:

[1] The two papers [17, 18] are based on ordinary isogeny graphs and, despite their titles, neither of the papers provides an authenticated protocol.

[2] As far as I can tell, the scheme of LeGrow [28] can be interpreted as SIDH protected by a MAC coming from a long-term static shared Diffie-Hellman key. But I might be wrong about that. LeGrow writes that the scheme is applying a post-quantum transform to a signature scheme. The scheme also requires a chameleon hash, and does not provide a post-quantum instantiation of such a hash.

1. Discrete logarithms provide a richer algebraic structure that allows more sophisticated protocols.
2. There are adaptive attacks in the SIDH setting that make certain operations involving long-term private keys dangerous.
3. There are gap assumptions in the discrete logarithm setting that are useful for security proofs, but these assumptions do not hold in the SIDH setting (because the decisional variant of the main problem is equivalent to the computational variant).

Hence many existing AKE schemes and their security proofs cannot be adapted to the SIDH case. Luckily, an AKE scheme (and its security proof in the random oracle model) due to Jeong, Katz and Lee [21] are adaptable to this setting. We give the details in Section 7.

The aims of this paper are to highlight some of the challenges in authenticated key exchange for SIDH and to present some basic schemes (together with security analysis) as a starting point for future research. We emphasise that the goal is to obtain practical and efficient AKE schemes based on isogeny problems, with *classical* security proofs in the random oracle model (ROM). We argue in Section 3.2 that classical security proofs are sufficient for post-quantum cryptography using classical devices.

Acknowledgements

The seeds of this project were sown during a summer research project by Alex Coombs supervised by the author in 2017. The author thanks Jonathan Katz for answering questions about his work [21]. The author is especially grateful to the reviewers of PQCrypto 2018 for their many helpful suggestions and pointers to relevant literature.

2 Background on SIDH

For general background on elliptic curves and isogenies we refer to [10, 12, 15, 32, 36]. For background on SIDH we refer to the original papers [9, 19].

The SIDH scheme of Jao and De Feo [19] (also see De Feo, Jao and Plût [9]) has a very special set-up. First choose distinct small primes ℓ_1, ℓ_2 (typically $\ell_1 = 2$ and $\ell_2 = 3$) and choose $e_1, e_2 \in \mathbb{N}$ such that $\ell_1^{e_1} \approx \ell_2^{e_2} \approx 2^\lambda$ where λ is some security parameter (typically, $\lambda = 256$). Next choose a small integer $f \in \mathbb{N}$ such that $p = \ell_1^{e_1}\ell_2^{e_2} f \pm 1$ is prime. Choose E to be a supersingular elliptic curve over $\mathbb{F}_{p^2}$ such that $E(\mathbb{F}_{p^2})$ has group structure a product of two cyclic groups of order $\ell_1^{e_1}\ell_2^{e_2} f$. Fix points $P_A, Q_A \in E[\ell_1^{e_1}]$ such that the group generated by P_A and Q_A is the whole group $E[\ell_1^{e_1}]$. Similarly, choose P_B, Q_B such that $\langle P_B, Q_B\rangle = E[\ell_2^{e_2}]$. The **SIDH system parameters** are (E, P_A, Q_A, P_B, Q_B).

The **SIDH key exchange protocol** works as follows: Alice chooses a secret subgroup of $E[\ell_1^{e_1}]$ by choosing an integer $0 \leq a < \ell_1^{e_1}$ and setting $S_A = P_A + [a]Q_A$. Alice computes an isogeny $\phi_A : E \to E_A$ with kernel generated by S_A and publishes $(E_A, \phi_A(P_B), \phi_A(Q_B))$. Similarly, Bob chooses $0 \leq b < \ell_2^{e_2}$, computes $\phi_B : E \to E_B$ with kernel generated by $S_B = P_B + [b]Q_B$ and publishes $(E_B, \phi_B(P_A), \phi_B(Q_A))$. To compute the shared key, Alice computes

$$S'_A = \phi_B(P_A) + [a]\phi_B(Q_A) = \phi_B(P_A + [a]Q_A) = \phi_B(S_A)$$

and then computes an isogeny $\phi'_A : E_B \to E_{AB}$ with kernel generated by S'_A. The composition $\phi'_A \circ \phi_B : E \to E_{AB}$ has kernel $\langle S_A, S_B\rangle$. Similarly, Bob computes an isogeny $\phi'_B : E_A \to E'_{AB}$ with kernel $\langle \phi_A(P_B) + [b]\phi_A(Q_B)\rangle$. The shared key is the j-invariant $j(E_{AB}) = j(E'_{AB})$.

We will use the notation $G_A = \langle S_A\rangle$ and $E_A = E/G_A$. So we can write E_{AB} as $(E/G_B)/\phi_B(G_A) \cong E/\langle G_A, G_B\rangle$.

For more discussion of the protocol and its implementation and security we refer to [19, 9, 7]. In particular, there are compression methods [8] to minimise the bandwidth when sending triples like $(E_A, \phi_A(P_B), \phi_A(Q_B))$.

The security of the SIDH key exchange protocol relies on the hardness of the following computational problem.

Problem 1. **SIDH problem**: Let (E, P_A, Q_A, P_B, Q_B) be SIDH system parameters. Let E_A be such that there is an isogeny $\phi_A : E \to E_A$ of degree $\ell_1^{e_1}$ with kernel G_A. Let $P'_B = \phi_A(P_B), Q'_B = \phi_A(Q_B)$. Let E_B be such that there is an isogeny $\phi_B : E \to E_B$ of degree $\ell_2^{e_2}$ with kernel G_B. Let $P'_A = \phi_B(P_A), Q'_A = \phi_B(Q_A)$.

The problem is: Given $(E, P_A, Q_A, P_B, Q_B, E_A, P'_B, Q'_B, E_B, P'_A, Q'_A)$ to determine $j(E_{AB})$ where $E_{AB} \cong E/\langle G_A, G_B\rangle$.

Problem 1 is believed to be hard for quantum computers. We refer to [15] for a survey of algorithmic results.

It is obvious that the SIDH protocol is not secure against a man-in-the-middle attack. Hence, in reality, there is a need to provide some kind of authentication to users of the protocol.

We mention here an issue with using this scheme in a multi-user setting. Suppose Alice engages in the SIDH protocol with Bob and also with Carol. Then, for example, Alice may use points of order $\ell_1^{e_1}$ while Bob and Carol use points of order $\ell_2^{e_2}$. What if now Bob wants to agree a key with Carol? One of them will use points of order $\ell_1^{e_1}$ while the other will use points of order $\ell_2^{e_2}$. This is fine when all elements of the protocol are ephemeral, but it creates a problem when we move to the authenticated setting. We call it the *three body problem*.

As observed in several papers [15, 34], decisional variants of problem 1 are equivalent to the computational problem. Hence there is no chance to get a "gap problem" in this setting (i.e., a situation where the computational problem can be hard even in the presence of a decisional oracle). This prevents us from adapting certain security arguments from the discrete logarithm setting to the SIDH setting.

Finally, we mention the adaptive attacks due to Galbraith, Petit, Shani and Ti [13] on the case where one party uses a static key. Hence, if we build an authenticated key exchange using static and ephemeral components then we will need to keep these attacks in mind.

3 Authenticated encryption in the discrete logarithm setting

Let G be a group with generator g. The Diffie-Hellman scheme involves Alice sending g^x to Bob and Bob sending g^y to Alice and then both parties computing g^{xy}. As is well known, this is not secure against a man-in-the-middle attack.

A general solution is to sign all protocol messages using digital signatures (this is folklore and is analysed in [6]; we refer to [1] for a recent application of the idea). The security of the resulting protocol then depends on the Diffie-Hellman assumption and whatever assumptions are necessary for the security of the digital signature scheme.

There are a huge number of papers that provide a higher level of security by including long-term public keys as well as ephemeral messages. Such schemes all fall under the heading of authenticated key exchange (AKE).

Throughout the paper we use the following notation: $H : \{0,1\}^* \to \{0,1\}^l$ is a cryptographic hash function. For $a_1, \ldots, a_t \in \{0,1\}^*$ we write $H(a_1, \ldots, a_t)$ for the hash of the concatenation of the strings. When writing things like $H(g^{xy})$ we assume a well-defined specification for how group elements are encoded as binary strings.

3.1 Brief summary of popular AKE schemes

We do not give an exhaustive survey here, but start our historical summary with a survey paper by Blake-Wilson and Menezes [2]. Their notation is for Alice to have static key $Y_A = g^a$ and ephemeral (one-time) protocol messages $T_A = g^x$. Similarly Bob has public key $Y_B = g^b$ and protocol messages $T_B = g^y$. The paper recalls the **KEA** scheme: Alice sends g^x and Bob sends g^y and the shared key is $g^{ay} + g^{bx}$. Next this is **unified** to a scheme (protocol 5 in their paper) with shared key $H(g^{ab}, g^{xy})$ where H is a cryptographic hash function. Next they discuss **MQV**, which has a complex shared key of the form $g^{(x+aU)(y+bV)}$ for some values U and V that are derived from the protocol messages and are "short" to provide efficiency. Protocol 8 of their paper is a more complex scheme that also computes a shared key of the form $H(g^{ab}, g^{xy})$. We also refer to Okamoto [31] for a similar protocol.

Krawczyk [24] proposed the **HMQV** protocol and gave a proof of security in the Canetti-Krawczyk model. Menezes [30] pointed out that certain validation steps are still necessary with HMQV. The **NAXOS** protocol of LaMacchia, Lauter and Mityagin [26] (also see [27, 35]) is a protocol that allows Alice and Bob to share the key $H(A, B, g^{ay}, g^{bx}, g^{xy})$ where the ephemeral values are themselves generated using a hash function (modelled as a random oracle).

Jeong, Katz and Lee [21] also present a model for "one-round" protocols (allowing simultaneous communication between Alice and Bob) and give three schemes. Their scheme TS2 has shared secret $H(A, B, g^x, g^y, g^{xy}, g^{ab})$ and the security proof in the random oracle model relies on the computational Diffie-Hellman problem (CDH). They also give a scheme TS3 that is proved secure in the standard model under the DDH assumption and the security of a MAC. The scheme TS3 is similar to the solution of signing protocol messages: Alice and Bob derive a static Diffie-Hellman key g^{ab} from their respective public keys g^a and g^b and use it as the key to a MAC for authenticating their ephemeral Diffie-Hellman messages.

We stress that these schemes provide what is known as **implicit authentication**, which means that Alice knows that the only other person who could compute the same key as her is Bob (and vice versa). In practice this means that if Alice later receives a message together with, say, a MAC tag that verifies with respect to the key generated from the session, then Alice is assured that the MAC tag must have been generated by Bob.

Note that most AKE schemes require a strong public key validation stage during registration with the public key infrastructure. This is a process that requires a user, during registration of their public key, to prove that they know the private key. It prevents certain attacks that involve Eve registering a public key that is in some algebraic way related to the public key of another user. Most protocols also require players to check validity of values sent during the protocol (e.g., checking that bitstrings correspond to group elements of the correct order). We refer to Menezes [30] for an overview of such attacks in the context of MQV and HMQV, and we refer to Boyd et al [5] for a broader discussion of how to model certification authorities.

We now recall the Kaliski attack [22] on MQV (an unknown key share attack): Suppose Alice sends $T_A = g^x$ and this is intercepted by Eve. Eve chooses a random r, computes $T_E = T_A Y_A^U g^{-r} = g^{x+Ua-r}$ and computes the corresponding U' by the MQV process. Then Eve sets $e = U'^{-1}r$ and registers public key $Y_E = g^e$. Note that Eve knows the private key for this public key, and so can perform the key validation process. Then Eve sends T_E to Bob, as coming from her public key Y_E. Bob then sends $T_B = g^b$ to Eve and computes key

$$(T_E Y_E^{U'})^{(y+Vb)} = (T_A Y_A^U g^{-r}(g^{U'^{-1}r})^{U'})^{(y+Vb)} = g^{(x+Ua)(y+Vb)}.$$

Eve then forwards T_B to Alice and she computes the same key. It means that Alice and Bob share a key. Alice thinks she shares the key with Bob, but Bob thinks he shares the key with Eve. Hence Eve is enabled to "take credit" for information coming from Alice.

3.2 Security models for AKE

Everything in this section is classical; we do not consider quantum adversaries or quantum-secure models for AKE. LeGrow [28] has considered a quantum-secure model for AKE (similar to the work of Soukharev, Jao and Seshadri [33]) but this is not our concern. We are concerned with classical devices and classical communication channels. The adversary may have access to a quantum computer in order to solve some computational problem, but currently deployed devices and communication channels do not provide quantum access to adversaries.

Our security proof is in the random oracle model, but this is just a proof technique to model attacks. The real cryptosystem uses a fixed hash function. Hence we believe it is not necessary to consider that the attacker has quantum acccess to a random oracle. While we appreciate that research on the quantum random oracle model is of value, we believe the quantum random oracle model is not necessary for understanding the security of AKE protocols that use fixed hash functions and work with classical devices over classical channels.

Formal security analysis typically proceeds in the model of Canetti and Krawczyk [6]. We also follow the formulation from Jeong, Katz and Lee [21].

The model considers a collection of n players, each with a public key. Each player engages in a number of sessions (indexed by j). Session j of player i has an associated session identifier $\mathsf{sid}_{i,j}$. We denote by Π_i^j the instance of player i running in session j. Session j of player i and session j' of player i' are said to be *matching* or *partners* if player i believes session j has partner i' and session id $\mathsf{sid}_{i,j}$ and player i' believes session j' has partner i and session id $\mathsf{sid}_{i',j'} = \mathsf{sid}_{i,j}$.

An instance Π_i^j is *completed* if either i initiates a protocol instance in session j and receives a valid response from its partner, or if session j of i receives a valid initial protocol message and responds to it. The correctness condition is that completed partners should compute the same session key.

The adversary in the model controls all communication between players, and may modify or replace any message from player i to player i'. This is formalised by giving the adversary access to a number of oracles.

$\mathsf{Initiate}(i, i')$**:** Triggers player i to initiate a protocol session with player i'. This returns a protocol message (that includes a session identifier).

$\mathsf{Send}(i, j, M)$**:** Sends message M to Π_i^j (session j of player i), in response to which Π_i^j performs the specified steps of the protocol and returns its protocol message as appropriate. This query enables the protocol to operate and it

also allows to model active attacks where the value sent to a player is not the true value that was sent by another player.

Execute(i, i')**:** Starts a fresh protocol interaction between players i and i', initiated by i and completed by i', that is conducted in a correct and undisrupted way. This models passive eavesdropping.

Corrupt**:** The adversary learns the long-term private key of player i, and all session keys agreed so far, and any other local state information. The adversary can then create a new public key for that user (possibly depending on other user's public keys and/or protocol messages).

Session-Key**:** (Also called Reveal or Session-key reveal.) When a player completes a key exchange protocol instance this query provides the adversary with the resulting session key.

Session-State**:** (Also called Session-state reveal.) During a key exchange protocol, before the protocol completes, this query provides the adversary with any values that are stored by the player that are necessary for completion of the protocol.[3]

After interacting with the players the adversary eventually makes a Test query (i, j) on a completed session j of player i. Let i' be player i's partner in session j (so that session j' of player i' is a matching session with session j of player i). It is required that players i and i' have not been corrupted, and that session j of player i and session j' of player i' have not been the victim of state reveal or key reveal queries. In response to the test query, an unbiased coin b is flipped. If $b = 0$ the adversary is given the session key computed by Π_i^j, and if $b = 1$ then a random string is returned to the adversary. The adversary is required to guess the bit b with noticeable advantage.

Note that Jeong, Katz and Lee [21] extend this model to include forward security, but for simplicity in this paper we restrict to the "key independence" model (which is essentially the Cannett-Krawczyk formulation). The proof techniques also hold in the forward security model, but the probability calculations in the security proof are more complex.

One issue with the Canetti-Krawczyk model is an ambiguity with the meaning of state reveal queries. Do these queries just yield the ephemeral secret x stored by Alice? Or do they give access to values that are temporarily stored by Alice during the computation of the secret key? For example, suppose we are discussing the unified scheme $H(g^{ab}, g^{xy})$ where Alice has key g^a and sends g^x to Bob. Clearly Alice needs to store the ephemeral value x so that she can compute T_B^x in the second phase of the protocol. So it is natural that a state reveal query to Alice, between the first and second stage of the protocol, will reveal x. But the adversary can also make a state reveal query during the second part of Alice's computation, so does this mean the adversary can also obtain the intermediate value g^{ab} that Alice must compute before she computes the hash value? The literature seems to be ambiguous about this. Essentially, the issue is to what extent state reveal queries are modelling side-channel attacks. Criticisms like these have been made in [30, 26].

3.3 Comments on discrete logarithm based AKE protocols

We now give some remarks and examples to clarify certain issues that will be of concern later in the paper.

Remark 1. Consider a basic protocol with shared key $H(g^{ab}, g^{xy})$: If a state reveal query at the appropriate time can leak g^{ab} then Eve can now intercept any key exchange session between A and B and create a shared key k with A and a shared key k' with B, while A and B believe they share the key with each other.

On the other hand, if the key is $H(g^{ay}, g^{bx})$ then an attacker who replaces the messages g^x and g^y will still not be able to compute the value computed by Alice or Bob. For example, Alice will compute g^{bx} and the attacker does not know b or x.

Remark 2. Consider a protocol with shared key $H(g^{ay}, g^{bx})$. Alice computes this key by taking Bob's public key Y_B and Bob's message T_B and computing $H(T_B^a, Y_B^x)$. It is well-known that this is vulnerable to a small subgroup attack by Bob: Bob may send a value T_B of small order ℓ. Bob knows b and so can compute g^{bx}. Hence, by doing a session key reveal query on Alice, Bob can test a guess u for $a \pmod{\ell}$ by seeing if $H(T_B^u, g^{bx})$ is equal to the claimed value.

As we explain in Section 6.1, a variant of this attack that is hard to detect can be mounted in the SIDH setting, and so we need to pay close attention to it.

[3] It seems that Jeong-Katz-Lee omit this query.

Remark 3. A typical way to prove security is explained in general terms by Kudla and Paterson [25]. This involves a reduction to the gap-CDH problem (i.e., the CDH problem in the setting where the solver has access to a DDH oracle). The reduction inserts a CDH instance (g, g^b, g^x) into the protocol as follows: Set up a user B and set their public key to be g^b; choose public keys for all other users so that the private keys are known to the reduction; put g^x into one of the protocol sessions between A and B; hope that B is not corrupted by the user and that the protocol session with g^x is the subject of the test query. The problem is how can the reduction answer the key reveal queries made to B, since the private key is not known to the reduction. This is done by using the fact that there should be a random oracle query of the form $H(g^{ay}, g^{bx})$. One can use a decision oracle to search the list of random oracle queries and identify any suitable choice as a response to the reveal query. Similar ideas are used in [26, 35]. Such a proof methodology cannot be used in our case since the decisional variant of the SIDH assumption is equivalent to the computational variant.

There is no security proof known for the MQV protocol. Krawczyk [24] gave a proof of security for HMQV in the Canetti-Krawczyk model. His security proof relies on the computational Diffie-Hellman problem (the security proof is complex and leverages the security of a certain digital signature scheme).

The NAXOS protocol is proved secure based on a gap assumption. Lee and Park [27] give a variant of the NAXOS protocol that does not require a gap assumption. They use a standard "trapdoor test" idea in their proof, but this exploits algebraic structures that are not available in the isogeny setting.

4 Generic constructions of AKE from IND-CCA KEMs or public key encryption

There are a number of papers that discuss generic constructions of secure authenticated key exchange from basic primitives like IND-CCA encryption/KEMs and MACs, PRFs etc. We very briefly summarise some of these papers.

The first main result is due to Bellare, Canetti and Krawczyk [3]. They show how to add "authenticators" to a non-authenticated key exchange protocol like Diffie-Hellman. The drawback is that the number of rounds increases.

There are a substantial number of papers following on from this work, and we will simply mention four of them. We do not give full details, and for simplicity blur the distinction between KEMs and public key encryption. We will use the notation pk_A as the public key for a user A and $\mathsf{Enc}_{pk_A}(r)$ for either IND-CCA public key encryption of a random bitstring r to A, or else the IND-CCA encapsulation of a random key r to A.

In 2009, Boyd, Cliff, Gonzlez Nieto and Paterson [4] gave a simple scheme with two message passes. Alice sends to Bob $(A, C_A = \mathsf{Enc}_{pk_B}(r_A))$ and Bob sends to Alice $(B, C_B = \mathsf{Enc}_{pk_A}(r_B))$. Alice and Bob derive a key from the shared data $(A, C_A, B, C_B, r_A, r_B)$. This scheme (Protocol 1 in their paper) does not provide forward security. A second scheme (Protocol 2 in [4]) provides "(weak) forward secrecy". The second scheme runs an additional unauthenticated key exchange protocol in parallel. Both protocols provide KCI resistance.

In 2014, Li, Schäge, Yang, Bader and Schwenk [29] gave a scheme that uses Encryption and MACs and requires 5 rounds of communication. We do not describe it here.

In 2015, Fujioka, Suzuki, Xagawa and Yoneyama [11] gave a variant of the Boyd et al scheme that still only requires two rounds but now also features a "one-time" public key generated by Alice (much of the paper is about identity-based KEMs, but it seems the methods also work in the public key setting). So the protocol is: Alice sends to Bob $(A, C_A = \mathsf{Enc}_{pk_B}(r_A), pk_T)$ and Bob sends to Alice $(B, C_B = \mathsf{Enc}_{pk_A}(r_B), C_T = \mathsf{Enc}_{pk_T}(r_T))$. Alice and Bob derive a key from the shared data $(A, C_A, B, C_B, C_T, r_A, r_B, r_T)$.

In 2017, Guilhem, Smart and Warinschi [16] gave a three round scheme. They start with an unauthenticated key exchange protocol (such as Diffie-Hellman) that involves a message m_1 from Alice to Bob and a message m_2 from Bob to Alice. (This is similar to Protocol 2 in [4], which also includes an unauthenticated key exchange protocol.) Rather than exchanging m_1 and m_2, these values are now encrypted using an IND-CCA secure public key encryption scheme. So Alice sends to Bob $C_A = \mathsf{Enc}_{pk_B}(m_1)$ and Bob sends to Alice $C_B = \mathsf{Enc}_{pk_A}(m_2)$. After decrypting they computed the shared key k resulting from the unauthenticated key exchange protocol. This key k is used to derive a MAC key k_1 and a session key $k_2 = H(k, Alice, Bob)$. Finally, Alice sends to Bob a MAC value $\mathsf{Mac}(k_1, Alice \| Bob)$ which is checked by Bob.

The above brief discussion gives a flavour of the approaches but omits many details (e.g., we have not discussed the tightness of security reductions). Please see the original papers for details. In particular, the exact security properties vary. For example, the focus in [16] is on forward security, while the focus in [11] is on key compromise impersonation (KCI) and maximal exposure attacks (MEX).

5 Adapting generic constructions to the SIDH case

5.1 IND-CCA KEM based on SIDH

One can take any of the schemes from the previous section and insert an IND-CCA secure KEM based on SIDH. The details of an IND-CCA KEM are given in the SIKE submission to the NIST PQCrypto competition (see Section 4.3.3 of [20]), and we briefly sketch it now.

The system parameters, as usual, are (E, P_A, Q_A, P_B, Q_B) and also the description of three hash functions F, G, H. The public key is $pk_B = (E_B, \phi_B(P_A), \phi_B(Q_A))$ where $\phi_B : E \to E_B$ is the secret isogeny of degree $\ell_2^{e_2}$. The secret key is an integer β such that $\ker(\phi_B) = \langle P_B + [\beta]Q_B \rangle$.
The **encapsulation algorithm** is:

- Chooses a random string $m \in \{0,1\}^n$ and compute $r = G(m, pk_B)$ where G is a hash function and $0 \leq r < \ell_1^{e_1}$.
- Compute $\phi_A : E \to E_A$ such that $\ker(\phi_A) = \langle P_A + [r]Q_A \rangle$.
- Compute $\phi_{AB} : E_B \to E_{AB}$ such that $\ker(\phi_{AB}) = \langle \phi_B(P_A) + [r]\phi_B(Q_A) \rangle$.
- Compute $c_1 = F(j(E_{AB})) \oplus m$ where F is a hash function to $\{0,1\}^n$.
- Output the ciphertext $c = (E_A, \phi_A(P_B), \phi_A(Q_B), c_1)$.
- Output the key $K = H(m, c)$.

The **decapsulation algorithm** of ciphertext $c = (E_A, \phi_A(P_B), \phi_A(Q_B), c_1)$ is:

- Compute $\phi_{BA} : E_A \to E_{BA}$ such that $\ker(\phi_{BA}) = \langle \phi_A(P_B) + [\beta]\phi_A(Q_B) \rangle$.
- Compute $m' = F(j(E_{BA})) \oplus c_1$.
- Compute $r' = G(m', pk_B)$.
- Compute $\phi'_A : E \to E'_A$ such that $\ker(\phi'_A) = \langle P_A + [r']Q_A \rangle$.
- If $(j(E'_A) = j(E_A)$ and $\phi_A(P_B) = \phi'_A(P_B)$ and $\phi_A(Q_B) = \phi'_A(Q_B))$ then output $K = H(m', c)$; otherwise choose random $s \in \{0,1\}^n$ and output $H(s, c)$.

Note that encapsulation requires two isogeny computations (both of degree $\ell_1^{e_1}$) and decapsulation also requires two isogeny computations (the first of degree $\ell_2^{e_2}$ and the second of degree $\ell_1^{e_1}$).

5.2 The transforms

Adapting the schemes by Boyd, Cliff, Gonzlez Nieto and Paterson [4] is immediate. For Protocol 1 in [4], Alice and Bob exchange random encapsulations and use the results to derive their session key. Both players perform one encapsulation and one decapsulation, so perform 4 isogeny computations in total. For Protocol 2 in [4] there is an additional SIDH key exchange protocol: Each player performs one isogeny computation to produce the message and a second isogeny computation to compute the shared key, giving 2 additional isogeny computations for each player.

Adapting the Fujioka, Suzuki, Xagawa and Yoneyama [11] scheme is relatively straightforward. The initiator Alice performs one encapsulation and also needs to generate a new SIDH public key pk_T, which involves one isogeny computation. Bob responds by performing two encapsulations. Bob also needs to decapsulate the message from Alice, while Alice needs to decap both messages from Bob. Overall, Alice performs key generation, one encapsulation and two decapsulations, which is 7 isogeny computations. Bob performs two encapsulations and one decapsulation, which is 6 isogeny computations.

The Guilhem, Smart and Warinschi [16] scheme starts with unauthenticated SIDH (so each player performs one isogeny computation to produce the message and a second isogeny computation to compute the shared key). The encryption and decryption operations add an extra 4 isogeny computations for each player. Plus there is the additional round of sending a MAC value.

We summarise these results in the following table. This shows that our protocol is more efficient than the protocols obtained by applying the generic solutions. However, a more careful analysis would compare the precise security properties of these protocols.

Authors	One-round	Total message flows	Initiator # isog	Responder # isog
Boyd et al. P1	Yes	2	4	4
Boyd et al. P2	Yes	2	6	6
Fujioka et al.	No	2	7	6
Guilhem et al.	No	3	6	6
This work	Yes	2	3 (2)	3 (2)

Table 1. Comparison of generic AKE protocols from public key encryption or KEMS, and our proposal from Section 7.1. Recall that "one-round" means that Alice and Bob can send their messages simultaneously. The table lists the number of isogeny computations. The final row is for our scheme in Section 7, which requires 3 isogeny computations by each player, unless the long-term secret value $j(E_{AB})$ is stored locally, in which case only 2 isogeny computations are required by each player.

6 Adapting non-generic constructions to the SIDH case

The most immediate problem, if one wants to adapt previous work to the SIDH case, is the lack of a ring structure "in the exponent". A protocol like MQV involves computations like $T_A Y_A^U$, which evaluate to g^{x+aU}. With isogenies we do not have such a rich suite of operations (this, by the way, is also the primary obstacle to having an efficient public key signature scheme based on isogenies). Hence, we have to ignore complex protocols like MQV and study simpler ones like the "unified" scheme [2], Jeong, Katz and Lee [21], and NAXOS [26].

Further issues arise in the security proofs. The constrained algebraic structure prohibits random-self-reductions, self-correctors, trapdoor tests and other tools that are often used in security proofs for AKE schemes. There is no gap assumption available, since there is a reduction from the search variant of SIDH to the decisional variant. Hence, most of the proof techniques in previous work go out the window.

Even worse, as we explain in the next subsection, there are schemes that would be secure in the discrete logarithm setting but whose SIDH variants are insecure!

6.1 Adaptive attack

One of the most serious issues with SIDH using static keys is the adaptive attack from Galbraith, Petit, Shani and Ti [13]. Suppose a protocol involves Bob computing a curve E_{BX} where $E_B = E/G_B$ is a fixed public key and E_X is an ephemeral value sent by Alice. It is shown in [13] how a malicious Alice can send (E_X, R', S'), where R' and S' are specially chosen points that differ from the protocol specification, in such a way that Alice can gradually learn Bob's long-term secret. The attack is undetectable by Bob (there is no way for Bob to tell whether the points R' and S' are the correct points to be sent by Alice) without performing an additional round of computation (as mentioned by Kirkwood et al [23]). The attack is symmetric: Bob can also mount a similar attack on Alice if she is computing a key of the form E_{AY}.

For future reference we recall the details of the attack in the context where Bob has a long term public key $E_B = E/\langle P_B + \beta Q_B \rangle$, where P_B, Q_B are points of order $\ell_2^{e_2}$ and $0 \leq \beta < \ell_2^{e_2}$ Let $\phi_X : E \to E_X$ be Alice's isogeny and let $R = \phi_X(P_B)$ and $S = \phi_X(Q_B)$. Bob computes his kernel by computing the point $R + [\beta]S$. Alice's attack learns Bob's key "bit-by-bit" (really "coefficient by coefficient" in the base ℓ_2 representation). Suppose, for step i of the attack, that Alice knows an integer K_i with $0 \leq K_i < \ell_2^i$ such that $\beta = K_i + \ell_2^i z$ for some unknown z. Let $z_0 \in \{0, 1, \ldots, \ell_2 - 1\}$ be a guess for $z \pmod{\ell_2}$. The attack allows to determine whether the guess z_0 is correct. The attack is to choose $R' = R + [-\ell_2^{m-1-i} K_i - \ell_2^{m-1} z_0]S$ and $S' = [1 + \ell_2^{m-i-1}]S$ and to send (E_X, R', S') to Bob. Bob computes

$$\begin{aligned} R' + [\beta]S' &= (R + [\beta]S) \ + \ [-\ell_2^{m-1-i} K_i - \ell_2^{m-1} z_0 + \ell_2^{m-i-1}(K_i + \ell_2^i z)]S \\ &= (R + [\beta]S) \ + \ [(z - z_0)\ell_2^{m-1}]S \end{aligned}$$

which generates the correct kernel if $z \equiv z_0 \pmod{\ell_2}$ and generates a different kernel otherwise. Note that Alice can compute a curve isomorphic to $E_X/\langle R + [\beta]S \rangle$ by computing $E_B/\langle P'_A + [\alpha]Q'_A \rangle$, where α is her ephemeral key. By making a key reveal query (or interacting with the corresponding session) Alice can determine which of these

two cases occurs. It follows that after at most $(\ell_2 - 1)e_2$ malicious interactions with Bob, Alice has learned Bob's long-term key.

The actual attack uses a further scaling of the points, to prevent the attack from being detected by using the Weil pairing. We refer to [13] for the details. The issue is that there is no way for Bob to know that he is the victim of this attack, apart from an additional round of communication and verification of Alice's ephemeral key.

In the classical discrete logarithm setting then it is perfectly reasonable to have a key exchange protocol that allows Alice and Bob to compute a shared key $H(g^{ay}, g^{bx})$. But this attack shows that for a SIDH protocol with shared a key $H(j(E_{AY}), j(E_{BX}))$ then a malicious Bob (who knows the private key for E_B) would be able to learn Alice's long-term key.

One of our contributions is to explain a new way to secure against such an attack (see Section A.3).

7 SIDH variant of the Jeong-Katz-Lee protocol

One problem with SIDH in the multi-user setting, already mentioned earlier, is the requirement to have public keys to both initiate and respond to key exchange sessions (the "three body problem"). Similar issues arise in the work of LeGrow [28]. Hence we are going to be encumbered with a larger public key than considered in most other works on SIDH.

Set system parameters (E, P_1, Q_1, P_2, Q_2) as in Section 2, so that P_i and Q_i have order $\ell_i^{e_i}$ for $i = 1, 2$ and $\ell_1^{e_1} \approx \ell_2^{e_2} \approx 2^\lambda$ where λ is the security parameter.

A user A generates a public key as follows: Choose a secret a_1 such that $0 \leq a_1 < \ell_1^{e_1}$ and generate kernel $G_{A,1}$ of order $\ell_1^{e_1}$ from point $P_1 + [a_1]Q_1$. Similarly choose a secret a_2 such that $0 \leq a_2 < \ell_2^{e_2}$ and generate kernel $G_{A,2}$ of order $\ell_2^{e_2}$ from point $P_2 + [a_2]Q_2$. The user's key is a tuple $(E_{A,1}, P_{A,1}, Q_{A,1}, E_{A,2}, P_{A,2}, Q_{A,2})$ where $E_{A,1} = E/G_{A,1}$ and $P_{A,1} = \phi_{A,1}(P_2)$, $Q_{A,1} = \phi_{A,1}(Q_2)$ etc.

Essentially, $E_{A,1}$ is used when initiating a key exchange session and $E_{A,2}$ is used when responding to a session.

7.1 SIDH-AKE Protocol

This protocol is based on the Jeong, Katz and Lee [21] scheme TS2.
When **Alice initiates** a session of the protocol she performs these operations:

- Chooses $0 \leq x < \ell_1^{e_1}$ and sets $S_X = P_1 + [x]Q_1$, sets $G_X = \langle S_X \rangle$ and computes $\phi_X : E \to E_X = E/G_X$.
- Chooses session identifier sid. To be able to apply the Jeong, Katz and Lee proof we must define sid to be the triple $(E_X, \phi_X(P_2), \phi_X(Q_2))$ (which is already being sent by Alice and so sid is implicit at this stage).
- Sends $(\text{Alice}, \text{Bob}, \mathsf{sid}, E_X, P'_2 = \phi_X(P_2), Q'_2 = \phi_X(Q_2))$ to Bob.
- Saves $x, j(E_X)$ in protected temporary storage.

On receipt of $(\text{Alice}, \text{Bob}, \mathsf{sid} = (E_X, P'_2, Q'_2))$ Bob does:

- Checks that E_X is a supersingular elliptic curve over $\mathbb{F}_{p^2}$ and that P'_2, Q'_2 are independent points on E_X of order $\ell_2^{e_2}$.
  (These checks require significant overhead, though are less costly than actually computing an $\ell_2^{e_2}$-isogeny. We refer to Section 9 of Costello, Longa and Naehrig [7] and also [13] for details about validation.)
- Chooses $0 \leq y < \ell_2^{e_2}$ and sets $S_Y = P_2 + [y]Q_2$, sets $G_Y = \langle S_Y \rangle$ and computes $\phi_Y : E \to E_Y = E/G_Y$.
- Sends $(\text{Alice}, \text{Bob}, \mathsf{sid}, E_Y, P'_1 = \phi_Y(P_1), Q'_1 = \phi_Y(Q_1))$ to Alice.
  Note that since $\mathsf{sid} = (E_X, \phi_X(P_2), \phi_X(Q_2))$ this means Bob's communication to Alice is roughly twice as long as Alice's communication to Bob.
- Computes $G_{XY} = \langle P'_2 + [y]Q'_2 \rangle$ and $E_{XY} = E_X/G_{XY}$.
- Looks up Alice's public key $(E_{A,1}, P_{A,1}, Q_{A,1})$, verifies her certificate, looks up his long-term private key b_2 from secure storage and computes $G_{AB} = \langle P_{A,1} + [b_2]Q_{A,1} \rangle$ and $E_{AB} = E_{A,1}/G_{AB}$.
- Computes the session key

$$k = H(\text{Alice}, \text{Bob}, j(E_X), P'_2, Q'_2, j(E_Y), P'_1, Q'_1, j(E_{XY}), j(E_{AB})).$$

– Flushes his working storage.

On receipt of $(\text{Alice}, \text{Bob}, \mathsf{sid}, E_Y, P'_1, Q'_1)$ Alice completes the protocol as follows:

– Checks that E_Y is a supersingular elliptic curve over $\mathbb{F}_{p^2}$ and that P'_1, Q'_1 are independent points on E_Y of order $\ell_1^{e_1}$.
– Retrieves $x, j(E_X)$ from temporary storage associated with sid and computes $G_{XY} = \langle P'_1 + [x]P'_2 \rangle$ and $E_{XY} = E_Y/G_{XY}$.
– Looks up Bob's public key $(E_{B,2}, P_{B,2}, Q_{B,2})$, verifies his certificate, looks up her long-term private key a_1 from secure storage and computes $G_{AB} = \langle P_{B,2} + [a_1]Q_{B,2} \rangle$ and $E_{AB} = E_{B,2}/G_{AB}$.
– Computes the session key

$$k = H(\text{Alice}, \text{Bob}, j(E_X), P'_2, Q'_2, j(E_Y), P'_1, Q'_1, j(E_{XY}), j(E_{AB})).$$

– Flushes her working storage.

The correctness of the protocol is easily verified, based on the correctness of the Jao-De Feo SIDH concept.

Note that compression techniques can be used to reduce the bandwidth of triples such as (E_X, P'_2, Q'_2), and the input to the hash computation should also use the compressed representation.

The computation costs are easily estimated. Alice and Bob need one isogeny each to create their protocol message, and two isogenies each to compute the curves E_{XY} and E_{AB}. If Alice and Bob are in regular contact then $j(E_{AB})$ could be cached, in which case the "online" cost is just two isogenies (though we need to take into account the risk of leakage of this long-term secret, as discussed in Remark 1). This explains the numbers given in Table 1 earlier.

One of the features of the design is to include the term $j(E_{XY})$. This provides forward security: if a player's long-term private key is obtained by an attacker, previous session keys are still secure since an attacker cannot compute E_{XY} when given E_X and E_Y.

7.2 Key validation

The security of our scheme depends on public keys being correctly and honestly formed. Hence Alice needs to prove to the PKI that her key is correctly formed. In this section we provide two simple ways to do this.

Alice's key is a tuple $(E_{A,1}, P_{A,1}, Q_{A,1}, E_{A,2}, P_{A,2}, Q_{A,2})$. Here $E_{A,1} = E/G_{A,1}$ for a group $G_{A,1}$ of order $\ell_1^{e_1}$ and $P_{B,1} = \phi_{A,1}(P_1)$ etc. Also $E_{A,2} = E/G_{A,2}$ for a group $G_{A,2}$ of order $\ell_2^{e_2}$ and $P_{A,2} = \phi_{A,2}(P_2)$ etc. The PKI needs to verify that everything is correctly constructed. So it must be the case that each curve is correctly formed as the image of an isogeny of the correct degree, and it must be the case that the points $P_{A,1}, Q_{A,1}, P_{A,2}, Q_{A,2}$ are the correct images of the points on the base curve.

The first solution, which is too inefficient, is using a "cut and choose" protocol: Alice runs her key generation algorithm k times (say, $k = 2^{20}$) to generate tuples $(E_{A,1}^{(i)}, P_{A,1}^{(i)}, Q_{A,1}^{(i)}, E_{A,2}^{(i)}, P_{A,2}^{(i)}, Q_{A,2}^{(i)})$ for $1 \leq i \leq k$. She then uploads all tuples to the PKI. The PKI then chooses one of them to be Alice's public key and then asks Alice to reveal the private key for all $k-1$ remaining tuples. The PKI then checks that all $k-1$ keys are correctly formed. The probability that Alice can successfully cheat this protocol is $1/k$.

The second, and much more efficient, solution is to use an interactive protocol similar to the zero-knowledge proof of identity in Section 3.1 of [9]: To prove knowledge of an isogeny $\phi : E \to E_A$ of degree d such that $\phi(P_i) = R_i$ for $1 \leq i \leq t$, the prover chooses a subgroup $G \subseteq E$ of order d' coprime to d and the order of the points P_i. The prover constructs isogenies $\rho : E \to \mathcal{E}$ and $\psi : E_A \to \mathcal{E}_A$ with $\ker(\rho) = G$ and $\ker(\psi) = \phi(G)$. The prover sends $(\mathcal{E}, \mathcal{E}_A)$ and $\rho(P_i)$ and $\psi(R_i)$ to the verifier. The verifier sends a bit b. If $b = 0$ the prover responds with G and $\phi(G)$, while if $b = 1$ the prover responds with $\rho(\ker(\phi))$. The verifier checks the consistency of the prover's responses. The probability Alice can cheat one round of this protocol is $1/2$. By repeating this argument the verifier becomes convinced of the validity of the prover's key (one needs to be careful to choose G from a fixed subgroup $E[n]$ to prevent leaking too much information).

An interesting problem for future research would be to consider other solutions.

7.3 Formal Security analysis

We closely follow the security proof of Section 4.2 of Jeong, Katz and Lee [21]. A key property of their approach is that the session identifier sid includes the values E_X and E_Y so that by definition a matching session must involve the correct values E_X and E_Y selected by the players during their protocol execution, and so has not been tampered with. This aspect of the protocol design basically implies that there can be no man-in-the-middle attack, since any change to the protocol messages does not lead to a matching session. The proof in [21] exploits random-self-reduction of CDH tuples, which is not available in our setting, and so the reduction in the SIDH case is even less tight than the result in [21]. We specify that the session state reveal query only returns the ephemeral values x or y, and not other intermediate values such as $j(E_{AB})$, hence the issue of Remark 1 is avoided in the security proof.

Theorem 1. *Suppose Problem 1 is hard. Then, in the random oracle model, the SIDH-AKE Protocol is a secure authenticated key exchange protocol in the Canetti-Krawczyk model.*

Proof. Let A be an adversary to the SIDH-AKE protocol in the Canetti-Krawczyk model that succeeds with noticeable advantage. Let there be n users in the system. Let m be an upper bound for the number of sessions participated in by each player i. Note that n and m are polynomial in the security parameter.

The proof proceeds in two stages. The first stage is to show that any successful adversary must make a random oracle query of a certain form with noticeable probability. The second stage is to give a simulation of the protocol such that when the adversary makes the specific query then a challenge SIDH instance is solved.

Stage one: The adversary chooses some player (i, j) as the object of the Test query. This means that this player is not the object of a Corrupt or Session-State or Session-Key query and that it has completed the protocol and has a matching partner session that is also not the object of a Corrupt or Session-State or Session-Key query. Let $\mathsf{sid} = (j(E_X), P'_2, Q'_2, j(E_Y), P'_1, Q'_1)$ be the session identifier for this matching session and let E_A and E_B be the public keys for players i and j respectively. We define two events:

q_1 is the event that there are two non-partnered protocol instances Π_i^j and Π_k^l with the same session identifier.
q_2 is the event that there is a query to the random oracle of the form $H(A, B, \mathsf{sid}, j(E_{XY}), j(E_{AB}))$.

Since each player generates a fresh ephemeral key, the probability in any given instance that the player generates any specific curve E that has been already used by another player is at most $1/2^\lambda$ (where λ is the security parameter). Since there are at most nm sessions in total there are at most $(mn)^2$ possible pairs of conflicts to avoid, and so $\Pr(q_1) \leq (nm)^2/2^\lambda$.

We now prove that if A succeeds in the Test query with noticeable advantage then $\Pr(q_2)$ is noticeable. Let b be the hidden bit in the security game played by A, and b' the bit output by A. The probability distribution is over the choices made by A and the simulation. The advantage $\mathrm{Adv}(A)$ of A is

$$\begin{aligned}
\left|\Pr(b = b') - \tfrac{1}{2}\right| &= \left|\Pr(b = b' \wedge q_1) + \Pr(b = b' \wedge \bar{q_1}) - \tfrac{1}{2}\right| \\
&= \Big|\Pr(b = b'|q_1)\Pr(q_1) + \Pr(b = b' \wedge \bar{q_1} \wedge q_2) \\
&\quad + \Pr(b = b' \wedge \bar{q_1} \wedge \bar{q_2}) - \tfrac{1}{2}\Big| \\
&= \Big|\Pr(b = b'|q_1)\Pr(q_1) + \Pr(b = b'|\bar{q_1} \wedge q_2)\Pr(\bar{q_1} \wedge q_2) \\
&\quad + \Pr(b = b'|\bar{q_1} \wedge \bar{q_2})\Pr(\bar{q_1} \wedge \bar{q_2}) - \tfrac{1}{2}\Big|.
\end{aligned}$$

Now, $\Pr(b = b'|\bar{q_1} \wedge \bar{q_2}) = \frac{1}{2}$ since if H is never queried on the value of the test query, and if there is no way to get access to the corresponding key via a session key reveal query, then there is no information about the hidden bit b. For brevity define $\alpha = \Pr(q_1)$ and $\beta = \Pr(\bar{q_1} \wedge q_2)$. Then

$$\Pr(\bar{q_1} \wedge \bar{q_2}) = 1 - \alpha - \beta.$$

Hence, the advantage can be written as

$$\begin{aligned}
&\left|\Pr(b = b'|q_1)\alpha + \Pr(b = b'|\bar{q_1} \wedge q_2)\beta + \tfrac{1}{2}(1 - \alpha - \beta) - \tfrac{1}{2}\right| \\
&= \left|\Pr(b = b'|q_1)\alpha + \Pr(b = b'|\bar{q_1} \wedge q_2)\beta - \tfrac{1}{2}(\alpha + \beta)\right| \\
&\leq \tfrac{1}{2}(\alpha + \beta).
\end{aligned}$$

It follows that $\Pr(q_2) \geq \beta \geq 2\text{Adv}(A) - \alpha \geq 2\text{Adv}(A) - (nm)^2/2^\lambda$ and so is noticeable.

Stage two: Let $(E, P_1, Q_1, P_2, Q_2, E_X, P'_2, Q'_2, E_Y, P'_1, Q'_1)$ be a challenge instance of Problem 1. We will construct a simulator that uses A to solve this problem instance (in other words, computes $j(E_{XY})$ where $E_X = E/G_X$, $E_Y = E/G_Y$ and $E_{XY} \cong E/\langle G_X, G_Y \rangle$).

The simulation begins with the simulator creating n users i for $1 \leq i \leq n$ and public keys for them. The simulator generates public keys correctly and knows the corresponding private keys. The simulator randomly chooses two users i_0 and i'_0. The simulator randomly chooses session $1 \leq j_0 \leq m$ for user i_0, and session $1 \leq j'_0 \leq m$ for user i'_0. The simulator is hoping that session j_0 of player i_0 will match session j'_0 of player i'_0, and that the test query will be applied to either the j_0-th session of player i_0 or the j'_0-th session of player i'_0.

The simulation handles random oracle queries in the usual way: On input u the simulation checks whether $H(u)$ has already been queried and, if not, uniformly chooses a binary string $h \in \{0,1\}^l$ and returns h and adds (u, h) to the table of random oracle values.

We now explain how to handle the queries required in the Canetti-Krawczyk model.

Initiate(i, i')**:** Let j be the next available session for player i. If $(i, j) = (i_0, j_0)$ then the output is (E_X, P'_2, Q'_2), coming from the challenge SIDH instance. If $(i, j) \neq (i_0, j_0)$ then a fresh value $0 \leq x < \ell_1^{e_1}$ is chosen (and stored) and ϕ_X and $E_X = E/\langle P_1 + [x]Q_1 \rangle$ are constructed and the output is the correctly formed triple (E_X, P'_2, Q'_2) as in the protocol.

Send(i, j, M)**:** If $(i, j) = (i'_0, j'_0)$ then the output is (E_Y, P'_1, Q'_1) from the challenge SIDH instance and no computation is performed (a key is not computed). If $(i, j) = (i_0, j_0)$ then do nothing (again, no key is computed). Otherwise, execute the protocol appropriately, using the private key and ephemeral keys that are known to the player in this protocol instance (in particular, do not perform any further computations if M is not a bitstring that satisfies the validity checks of the protocol).

Execute(i, i')**:** Determine the next available sessions j and j' for players i and i' respectively. Then run the Initiate and Send algorithms as just explained. In particular, if $(i, j) = (i_0, j_0)$ and $(i', j') = (i'_0, j'_0)$ then the protocol messages come from the challenge SIDH instance.

Corrupt(i)**:** Output the long-term private key of user i, together with all session keys for completed sessions up to this point in the game. Note that the private keys can always be output by the simulation. The session keys are handled as in the next item.

Session-Key(i, j) If $(i, j) \neq (i_0, j_0), (i'_0, j'_0)$ then the simulator knows all the relevant ephemeral information of player i and so can compute the relevant elliptic curves and perform a hash query and hence return the correct key

$$H(A, B, j(E_X), P'_2, Q'_2, j(E_Y), P'_1, Q'_1, j(E_{XY}), j(E_{AB})).$$

Note that this is true even if the values sent to player i by the Send query have been maliciously chosen.

We are assuming the Session-Key query is not made on the values (i_0, j_0) or (i'_0, j'_0). If it is then we just respond with a randomly chosen binary string in $\{0,1\}^l$.

Session-State(i, j) If $(i, j) \neq (i_0, j_0), (i'_0, j'_0)$ then returns the ephemeral value x used to generate E_X (respectively y to generate E_Y). We are assuming the Session-State query is not made on the values (i_0, j_0) or (i'_0, j'_0).

We are hoping that the Test query will be made on the instance $\Pi_{i_0}^{j_0}$ or $\Pi_{i'_0}^{j'_0}$ in the correct matching session, and that the adversary makes a random oracle query of the form $H(A, B, \mathsf{sid}, j(E_X), j(E_Y), j(E_{XY}), j(E_{AB}))$. The simulator looks up in the random oracle table all values of the form $(A, B, j(E_X), *, j(E_Y), *, u, j(E_{AB}))$, chooses one of them at random, and outputs the value u. This completes the description of the simulator.

We now discuss the correctness of the simulation in the random oracle model. All functions are correctly computed except for those involving Π_i^j with $(i, j) = (i_0, j_0)$ or (i'_0, j'_0). The only way the simulation cannot be perfect is if there is a conflict in the responses to the random oracle queries. This can happen in two ways. Either the ephemeral values E_X and E_Y are repeated (so a sid is repeated), or a query is made of the form $H(A, B, \mathsf{sid}, j(E_X), j(E_Y), j(E_{XY}), j(E_{AB}))$ where these are the specific values arising in the Test query as above. The former event is q_1 and we have already shown it is negligible. The latter event is q_2, and we have shown in stage one of the proof it is noticeable if the advantage of A is noticeable. However, this is the event we are looking for, so the potential failure of the simulation is no longer a concern at this point.

Finally, we discuss the success probability of the simulator to solve the SIDH instance. With probability at least $1/(mn)$ the Test query is made on either the instance $\Pi_{i_0}^{j_0}$ or $\Pi_{i'_0}^{j'_0}$. With probability at least $1/(mn)$ the partner of this instance is the other one desired by the simulation. Hence, with probability at least $1/(mn)^2$ we are in the setting that the simulator is hoping for.[4] Now the adversary plays the game and, as already noted, makes the desired hash query with noticeable probability. The simulation outputs one of the values from the list of hash inputs of the correct form (there are only polynomially many such entries). Hence, the simulation succeeds with noticeable probability. □

8 Other SIDH-AKE Protocols

As already discussed, one can get post-quantum secure AKE from SIDH by signing the protocol messages with a post-quantum secure digital signature scheme, such as a hash-based signature scheme. We leave the analysis of such protocols for future work.

One can also consider the TS3 scheme from Section 4.3 of Jeong, Katz and Lee [21].[5] In this scheme Alice has SIDH public key $(E_{A,1}, P_{A,1}, Q_{A,1}, E_{A,2}, P_{A,2}, Q_{A,2})$ and Bob has a public key $(E_{B,1}, P_{B,1}, Q_{B,1}, E_{B,2}, P_{B,2}, Q_{B,2})$. Knowing each other's public keys, Alice and Bob can agree on a static shared session key E_{AB} (where Alice, as initiator, uses $E_{A,1}$ and Bob as responder uses $E_{B,2}$). From this they derive a MAC key k by applying a hash function or strong extractor. The protocol is then for Alice to send to Bob $(E_X, P'_2, Q'_2, \tau_A = \mathsf{Mac}_k(E_X, P'_2, Q'_2))$ and for Bob to check the MAC and respond with $(E_Y, P'_1, Q'_1, \tau_A = \mathsf{Mac}_k(E_Y, P'_1, Q'_1))$. The shared key is $j(E_{XY})$.

The security proof of the TS3 protocol is long and complex (see the full version of [21] for the corrected version). It also relies on random-self-reduction of DDH tuples, which we do not have. Once again, we leave for future work a full analysis of this protocol in the SIDH setting.

In Appendix A we give another SIDH-AKE protocol. The shared key for this protocol is of the form $H(j(E_{AY}), j(E_{BX}), j(E_{XY}))$. This protocol has some features that may make it preferable in practice to the Jeong-Katz-Lee protocols (though it is computationally less efficient). We believe that a variant of the protocol can be proved secure using the same proof methodology as in Section 7.3.

9 Open problems and conclusions

We have surveyed the problem of authenticated key exchange in the context of supersingular isogeny Diffie-Hellman. We have highlighted a number of technical issues that explain why it is non-trivial to adapt results from the discrete logarithms setting to this new context. We have presented an SIDH variant of a protocol due to Jeong, Katz and Lee, and proved its security in the same model as [21]. We have also sketched several other SIDH-AKE protocols.

There are many open problems in this area. Foremost is the need to find new techniques to design and tightly prove security of AKE protocols in the SIDH setting. It is also necessary to give a full security analysis of these protocols that includes the widest possible variety of adversarial goals (because, as is known, not every possible security issue in AKE is captured by the formal models in [6, 21]). The following problems are also worth investigation.

1. Thoroughly review the literature to precisely compare the existing solutions from the point of view of security (e.g., forward security) and tightness of the security reduction.
2. Determine which protocols are most efficient (in terms of bandwidth and computation) for practical applications. For example, is it really necessary to include $j(E_X)$ and $j(E_Y)$ and the respective points as inputs to the hash function for the protocol in Section 7? These values are critical to the security proof, since they form the session identifier sid, and the strong condition on matching session identifiers allows us to eliminate certain attacks as outside the model.
3. Give more efficient solutions to public key validation for SIDH keys. (This is probably equivalent to more efficient isogeny signatures.)

[4] In the Jeong-Katz-Lee case they use a random-self-reduction to ensure that the challenge instance is inserted into all sessions, and so this loss of tightness does not appear in their security result.

[5] Bellare, Canetti and Krawczyk [3] also consider adding an authenticator to plain Diffie-Hellman key exchange.

4. Study post-quantum signature schemes and MAC schemes that can be used most effectively in practice to obtain good AKE schemes.

The aim of this paper is not to make a recommendation about which scheme is most suitable for isogeny crypto. We leave this as an open problem.

References

1. Christoph Bader, Dennis Hofheinz, Tibor Jager, Eike Kiltz and Yong Li. Tightly-Secure Authenticated Key Exchange. In Y. Dodis and J. B. Nielsen (eds.) TCC, Springer LNCS 9014 (2015) 629–658.
2. Simon Blake-Wilson and Alfred Menezes. Authenticated Diffie-Hellman Key Agreement Protocols. In S. E. Tavares and H. Meijer (eds.), Selected Areas in Cryptography '98, Springer LNCS 1556 (1998) 339-361.
3. M. Bellare, R. Canetti and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. Proceedings of 30th Annual Symposium on the Theory of Computing, ACM, 1998.
4. Colin Boyd, Yvonne Cliff, Juan Manuel Gonzlez Nieto, Kenneth G. Paterson: One-round key exchange in the standard model. IJACT 1(3): 181-199 (2009)
5. Colin Boyd, Cas Cremers, Michèle Feltz, Kenneth G. Paterson, Bertram Poettering and Douglas Stebila. ASICS: Authenticated Key Exchange Security Incorporating Certification Systems. In J. Crampton, S. Jajodia and K. Mayes (eds.) ESORICS 2013, Springer (2013) 381–399.
6. Ran Canetti and Hugo Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In B. Pfitzmann (ed.) EUROCRYPT 2001. Springer LNCS 2045 (2001) 453–474.
7. Craig Costello, Patrick Longa and Michael Naehrig. Efficient algorithms for supersingular isogeny Diffie-Hellman. In M. Robshaw and J. Katz (eds.), CRYPTO 2016, Part I, Springer LNCS 9814 (2016) 572–601.
8. Craig Costello, David Jao, Patrick Longa, Michael Naehrig, Joost Renes and David Urbanik. Efficient Compression of SIDH Public Keys. In J.-S. Coron and J. B. Nielsen (eds.), EUROCRYPT 2017, Part 1, Springer LNCS 10210 (2017) 679–706.
9. Luca De Feo, David Jao and Jérôme Plût. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. J. Mathematical Cryptology, 8(3) (2014) 209–247.
10. Luca De Feo. Mathematics of Isogeny Based Cryptography. Notes from a summer school on Mathematics for Post-quantum cryptography. `http://defeo.lu/ema2017/poly.pdf`
11. Atsushi Fujioka, Koutarou Suzuki, Keita Xagawa, Kazuki Yoneyama. Strongly secure authenticated key exchange from factoring, codes, and lattices. Des. Codes Cryptography 76(3): 469–504 (2015)
12. Steven D. Galbraith. Mathematics of Public Key Cryptography. Cambridge University Press, 2012.
13. Steven D. Galbraith, Christophe Petit, Barak Shani and Yan Bo Ti. On the security of supersingular isogeny cryptosystems. In J.-H. Cheon and T. Takagi (eds.), ASIACRYPT 2016, Springer LNCS 10031 (2016) 63–91.
14. Steven D. Galbraith, Christophe Petit and Javier Silva. Signature Schemes Based On Supersingular Isogeny Problems. In T. Takagi and T. Peyrin (eds), ASIACRYPT 2017, Springer LNCS 10624 (2017) 3–33.
15. Steven D. Galbraith and Frederik Vercauteren. Computational problems in supersingular elliptic curve isogenies. eprint 2017/774.
16. Cyprien de Saint Guilhem, Nigel P. Smart, Bogdan Warinschi. Generic Forward-Secure Key Agreement Without Signatures. in P. Q. Nguyen and K. Zhou (eds.) ISC 2017, Springer LNCS 10599 (2017) 114–133.
17. Han Weiwei, He Debiao. An Authenticated Key Agreement Protocol Using Isogenies Between Elliptic Curves. Second International Workshop on Education Technology and Computer Science (ETCS) (2010) 366–369.
18. Debiao He, Jianhua Chen, Jin Hu. An Authenticated Key Agreement Protocol Using Isogenies Between Elliptic Curves. International Journal of Computers Communications & Control, Vol 6, No 2 (2011) 258–265.
19. David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In B.-Y. Yang (ed.), PQCrypto 2011, Springer LNCS 7071 (2011) 19–34.
20. David Jao (Principal submitter). Supersingular Isogeny Key Encapsulation. Submission to NIST, November 30, 2017.
21. Ik Rae Jeong, Jonathan Katz and Dong Hoon Lee. One-Round Protocols for Two-Party Authenticated Key Exchange. In M. Jakobsson, M. Yung and J. Zhou (eds.), ACNS 2004, Springer LNCS 3089 (2004) 220–232.
22. Burton S. Kaliski Jr., An unknown key-share attack on the MQV key agreement protocol, ACM Trans. Inf. Syst. Secur., vol. 4, no. 3 (2001) 275–288.
23. 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, 2015, Workshop on Cybersecurity in a Post-Quantum World.
24. Hugo Krawczyk. HMQV: A High-Performance Secure Diffie-Hellman Protocol. In V. Shoup (ed.), CRYPTO 2005. Springer LNCS 3621 (2005) 546–566.

25. Caroline Kudla and Kenneth G. Paterson. Modular Security Proofs for Key Agreement Protocols. In B. K. Roy (ed.), ASIACRYPT 2005, Springer LNCS 3788 (2005) 549–565.
26. Brian A. LaMacchia, Kristin E. Lauter and Anton Mityagin. Stronger Security of Authenticated Key Exchange. In W. Susilo, J. K. Liu and Y. Mu (eds.), ProvSec 2007, Springer LNCS 4784 (2007) 1–16.
27. Jooyoung Lee and Je Hong Park. Efficient and Secure Authenticated Key Exchange Protocols in the eCK Model. IEICE Transactions, Vol. 94-A, no. 1 (2011) 129–138.
28. Jason LeGrow. Post-Quantum Security of Authenticated Key Establishment Protocols. University of Waterloo Masters Thesis (2016) `http://hdl.handle.net/10012/10386`
29. Yong Li, Sven Schäge, Zheng Yang, Christoph Bader, Jörg Schwenk. New Modular Compilers for Authenticated Key Exchange. In I. Boureanu, P. Owesarski and S. Vaudenay (eds.), ACNS 2014, Springer LNCS 8479 (2014) 1–18.
30. Alfred Menezes. Another look at HMQV. J. Mathematical Cryptology, Vol. 1, No. 1 (2007) 47–64.
31. Tatsuaki Okamoto. Authenticated Key Exchange and Key Encapsulation in the Standard Model. In K. Kurosawa (ed.), ASIACRYPT 2007, Springer LNCS 4833 (2007) 474–484.
32. Joseph H. Silverman. The arithmetic of elliptic curves. Springer GTM 106, Springer, 1986.
33. Vladimir Soukharev, David Jao and Srinath Seshadri. Post-Quantum Security Models for Authenticated Encryption. In T. Takagi (ed.) PQCrypto 2016, Springer LNCS 9606 (2016) 64–78.
34. Erik Thormarker. Post-Quantum Cryptography: Supersingular Isogeny Diffie-Hellman Key Exchange. Thesis, Stockholm University, 2017.
35. Berkant Ustaoglu. Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. Des. Codes Cryptography, 46, no. 3 (2008) 329–342.
36. Lawrence C. Washington. Elliptic curves: Number theory and cryptography, 2nd ed., CRC Press, 2008.
37. Youngho Yoo, Reza Azarderakhsh, Amir Jalali, David Jao and Vladimir Soukharev. A Post-Quantum Digital Signature Scheme Based on Supersingular Isogenies. In A. Kiayias (ed.) Financial Cryptography and Data Security FC 2017, Springer LNCS 10322 (2017) 163–181.

A Variant of the NAXOS scheme

This section presents a scheme that may be attractive in some practical applications, though it is less efficient than our SIDH variant of the Jeong-Katz-Lee schemes. It is closely related to other schemes in the discrete logarithm setting, such as the NAXOS scheme [26] by LaMacchia, Lauter and Mityagin. This scheme deliberately does not use $j(E_{AB})$, due to the issues mentioned in Remark 1.

A.1 Protocol

As in Section 7 the system parameters are (E, P_1, Q_1, P_2, Q_2). User Alice has public key $(E_{A,1}, P_{A,1}, Q_{A,1}, E_{A,2}, P_{A,2}, Q_{A,2})$, such that there is a $\ell_1^{e_1}$-isogeny $\phi_{A,1} : E \to E_{A,1}$ with kernel $\langle P_1 + a_1 Q_1 \rangle$ and $P_{A,1} = \phi_{A,1}(P_2), Q_{A,1} = \phi_{A,1}(Q_2)$ are points of order $\ell_2^{e_2}$ etc. Recall that $\ell_1^{e_1} \approx \ell_2^{e_2} \approx 2^\lambda$ where λ is the security parameter.

When **Alice initiates** a session of the protocol she performs these operations:

- Chooses $0 \leq x < \ell_1^{e_1}$ and sets $S_X = P_1 + [x]Q_1$, sets $G_X = \langle S_X \rangle$ and computes $\phi_X : E \to E_X = E/G_X$.
- Chooses session ID sid. This could be a random binary string, or one could avoid an explicit sid and instead define it to be the triple $(E_X, \phi_X(P_2), \phi_X(Q_2))$ that is already being sent.
- Sends $(\text{Alice}, \text{Bob}, \mathsf{sid}, E_X, \phi_X(P_2), \phi_X(Q_2))$ to Bob (compression is used here).
- Saves x in protected temporary storage.

On receipt of $(\text{Alice}, \text{Bob}, \mathsf{sid}, E_X, P'_2, Q'_2)$ Bob does:

- Checks that E_X is a supersingular elliptic curve over $\mathbb{F}_{p^2}$ and that P'_2, Q'_2 are independent points on E_X of order $\ell_2^{e_2}$.
- Chooses $0 \leq y < \ell_2^{e_2}$ and sets $S_Y = P_2 + [y]Q_2$, sets $G_Y = \langle S_Y \rangle$ and computes $\phi_Y : E \to E_Y = E/G_Y$.
- Sends $(\text{Alice}, \text{Bob}, \mathsf{sid}, E_Y, P'_1 = \phi_Y(P_1), Q'_1 = \phi_Y(Q_1))$ to Alice (compressed).
- Computes $G_{XY} = \langle P'_2 + [y]Q'_2 \rangle$ and $E_{XY} = E_X/G_{XY}$.
- Looks up his long-term private key b_2 from secure storage and computes $G_{XB} = \langle P'_2 + [b_2]Q'_2 \rangle$ and $E_{XB} = E_X/G_{XB}$.

– Looks up Alice's public key, verifies her certificate, and computes $G_{AY} = \langle P_{A,12} + [y]Q_{A,1} \rangle$ and $E_{AY} = E_{A,1}/G_{AY}$.
– Computes the session key $k = H(\text{Alice}, \text{Bob}, \mathsf{sid}, j(E_{AY}), j(E_{BX}), j(E_{XY}))$.
– Flushes his working storage.

On receipt of $(\text{Alice}, \text{Bob}, \mathsf{sid}, E_Y, P'_1, Q'_1)$ Alice completes the protocol as follows:

– Checks that E_Y is a supersingular elliptic curve over $\mathbb{F}_{p^2}$ and that P'_1, Q'_1 are independent points on E_Y of order $\ell_1^{e_1}$.
– Retrieves x from temporary storage and computes $G_{XY} = \langle P'_1 + [x]Q'_1 \rangle$ and $E_{XY} = E_Y/G_{XY}$.
– Looks up Bob's public key, verifies his certificate, and computes $G_{BX} = \langle P_{B,2} + [x]Q_{B,2} \rangle$ and $E_{BX} = E_{B,2}/G_{BX}$.
– Retrieves her long-term private key a_1 from secure storage and computes $G_{AY} = \langle P'_1 + [a_1]Q'_1 \rangle$ and $E_{AY} = E_Y/G_{AY}$.
– Computes the session key $k = H(\text{Alice}, \text{Bob}, \mathsf{sid}, j(E_{AY}), j(E_{BX}), j(E_{XY}))$.
– Flushes her working storage.

The correctness of the protocol is easily verified. Note that both players perform 4 isogeny computations in total, so this is less efficient than our earlier scheme.

A.2 Informal Security Analysis

We now give an informal analysis of the security of the protocol. For this analysis we assume that Problem 1 is hard, that H is a preimage-resistant hash function, that there is a public key infrastructure (PKI) that ensures authenticity and integrity of public keys, and that the PKI enforces public key validation for all users.

Consider an attacker Eve who tries to interfere with the protocol messages sent between Alice and Bob. Eve can replace Alice's message $(\text{Alice}, \text{Bob}, \mathsf{sid}, E_X, P'_2, Q'_2)$ with a message $(\text{Alice}, \text{Bob}, \mathsf{sid}, E_{X'}, P''_2, Q''_2)$ of her choosing. She can also replace Bob's message $(\text{Alice}, \text{Bob}, \mathsf{sid}, E_Y, P'_1, Q'_1)$ with $(\text{Alice}, \text{Bob}, \mathsf{sid}, E_{Y'}, P''_1, Q''_1)$. However, without knowing the long-term secret keys of Alice and Bob, she cannot compute either key. For example, to compute Alice's session key $H(\cdots, j(E_{AY'}), j(E_{BX}), j(E_{XY'}))$ it is necessary to compute the curve E_{BX} which cannot be done by Eve unless she knows Bob's private key or can determine Alice's ephemeral secret. Similarly, to compute Bob's session key $H(\cdots, j(E_{AY}), j(E_{BX'}), j(E_{X'Y}))$ Eve would need to compute E_{AY}. Hence, the use of the long-term public keys E_A and E_B is expected to provide **implicit key confirmation**. This means that Alice is convinced that Bob is the only other person who could compute k, and vice versa. So if Alice receives a message encrypted/MAC'd with the key k then Alice can be sure it comes from Bob.

There seems to be no way to mount an attack like the Kaliski attack (see Section 3.1), that "eliminates" part of the public key by choosing an appropriate protocol message. This is due to the lack of algebraic structure in the SIDH system: there seems to be no way to "combine" an isogeny with domain E_A and an isogeny with domain E into a single object.

The above informal arguments suggest that there is no simple algebraic way for Eve to learn or manipulate the keys agreed by users of the protocol. Adaptive attacks to learn a user's secret key are discussed in Section A.3 below. Forward security is provided by the value $j(E_{XY})$.

Note that we could have included E_{AB} in the session key as well. The choice to ignore this is because it feels less secure to rely on a single fixed shared long-term key E_{AB}. This is also the sort of shape one wants to have for future security proofs, and at least allows a proof in a weaker model that does not include session key reveal queries.

A.3 Resistance to adaptive attack

We now consider the attack from Section 6.1 and explain why it cannot be mounted on this protocol.

We explain the situation in the case of a malicious Alice; the case of Bob is exactly the same by relabelling. Recall that $\phi_X : E \to E_X$ is Alice's isogeny and that $R = \phi_X(P_B)$ and $S = \phi_X(Q_B)$ are the correct points. Bob needs to compute both E_{BX} and E_{XY}. Let b be the secret for E_B and y the secret for E_Y, so that $E_Y = E/\langle P_B + [y]Q_B \rangle$. Suppose, as previously, that at step i of the attack Alice knows an integer K_i with $0 \leq K_i < \ell_2^i$ such that $b =$

$K_i + \ell_2^i z$ for some unknown z. Let $z_0 \in \{0, 1, \ldots, \ell_2 - 1\}$ be a guess for $z \pmod{\ell_2}$. The attack is to choose $R' = R + [-\ell_2^{m-1-i} K_i - \ell_2^{m-1} z_0]S$ and $S' = [1 + \ell_2^{m-i-1}]S$. As we explained in Section 6.1, if the guess z_0 is correct then $E_X/\langle R' + [b]S' \rangle$ will be isomorphic to the curve that Alice can compute.

However, this protocol has key $H(j(E_{AY}), j(E_{BX}), j(E_{XY}))$ and so Alice cannot test a guess for the key without computing $j(E_{XY})$ (the pre-image resistance of H is necessary here). Bob computes E_{XY} by computing the quotient of E_X by $\langle R' + [y]S' \rangle$. The problem for Alice is that

$$R' + [y]S' \;=\; (R + [y]S) + [\ell_2^{m-1-i}(-K_i + y - \ell_2^i z_0)]S$$

where y is a freshly chosen random integer with $0 \leq y < \ell_2^m$. Hence, Alice has no knowledge of this point, or how to compute the corresponding points on E_Y that would allow her to compute E_{XY}. To conclude, Alice cannot produce a candidate for the key $H(j(E_{AY}), j(E_{BX}), j(E_{XY}))$ and so has no way to verify whether her guess z_0 is correct.

This idea gives an alternative to the Kirkwood et al [23] approach to preventing the adaptive attack.

A.4 Formal Security analysis

By including the protocol messages $(E_X, P'_2, Q'_2 E_Y, P'_1, Q'_1)$ in the session identifier (and hence including them in the hash inputs for generating the key), one can apply the proof technique of Jeong-Katz-Lee and prove the security of the scheme. The security reduction is not tight. We omit the details due to lack of enthusiasm. However, it is also interesting to discuss whether security can be proved using similar techniques to those used by LaMacchia et al [26] (potentially obtaining a tighter reduction).

We now discuss what would be required to get a rigorous security proof for the scheme in the random oracle model. The natural approach would be to develop a simulator that takes as input an instance $(E, E_X, P'_A, Q'_A, E_B, P'_B, Q'_B)$ of the SIDH problem and sets up (E_B, P'_B, Q'_B) as the public key of some user Bob and (E_X, P'_A, Q'_A) as the initial message in session j between Alice to Bob. The hope is that Bob is never corrupted by the adversary and that the session j is the object of the test query.

The problem, as already mentioned, is how to handle session key reveal queries to Bob. Consider a user Alice (controlled by the adversary) who sends a value E_X to Bob and then receives Bob's answer E_Y. Alice, knowing the ephemeral secret for E_X and also her long-term private key, can compute $j(E_{AY})$, $j(E_{BX})$, and $j(E_{XY})$. Adversarial Alice can therefore make a number of hash queries of the form $H(j(E_{AY}), u, j(E_{XY}))$ for different values u, one of them being the correct value $y = j(E_{BX})$. If a session key reveal query is made to Bob, how is the simulator to respond (even though the simulator controls the random oracle)? Knowing the secret key for E_Y, the simulator can compute $j(E_{AY})$ and $j(E_{XY})$, so can look up all random oracle queries of the form $H(j(E_{AY}), u, j(E_{XY}))$. But it cannot determine which of these values is the correct one to output as the session key. In the Jeong-Katz-Lee proof this is not a problem (the event q_2 has taken place), but this leads to a lack of tightness. This issue is resolved in the classical case (see LaMacchia et al [26]) by using a decision oracle, but we are unable to use such an assumption in the isogeny case. Hence we are unable to adapt their tighter proof to this protocol. However, in a weaker model that does not permit session key reveal queries, then it is clear that the security proof can work.