An Isogeny-Based Password-Authenticated Key Establishment Protocol

Oleg Taraskin\textsuperscript{1}, Vladimir Soukharev\textsuperscript{2}, David Jao\textsuperscript{3,4}, and Jason LeGrow\textsuperscript{3,5}

\textsuperscript{1} Bankex Foundation, Moscow, Russian Federation
tog.postquant@gmail.com, ot@bankexfoundation.org
\textsuperscript{2} InfoSec Global, Toronto, Ontario, M2J 5C2, Canada
vladimir.soukharev@infosecglobal.com
\textsuperscript{3} Department of Combinatorics and Optimization
University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada
\{djao,jlegrow\}@uwaterloo.ca
\textsuperscript{4} evolutionQ Inc.
Waterloo Accelerator Centre
Waterloo, Ontario, N2L 6R5, Canada
david.jao@evolutionQ.com
\textsuperscript{5} Institute for Quantum Computing, Waterloo, Ontario N2L 3G1, Canada

Abstract. Password authenticated key establishment (PAKE) is a cryptographic primitive that allows two parties who share a low-entropy secret (a password) to securely establish cryptographic keys in the absence of public key infrastructure. We present the first quantum-resistant password-authenticated key exchange scheme based on supersingular elliptic curve isogenies. The scheme is built upon supersingular isogeny Diffie-Hellman \cite{9}, and uses the password to generate functions which obscure the auxiliary points used in the computation. We include a detailed security proof based on a number of reasonable computational problems on supersingular elliptic curves.

Key words: isogenies, password authenticated key establishment, key establishment, post-quantum cryptography, isogeny-based cryptography, authentication

1 Introduction

Many current cryptographic schemes are based on mathematical problems that are considered difficult for classical computers, but can easily be solved using quantum algorithms. To prepare for the emergence of quantum computers, we aim to design cryptographic primitives for common operations, such as encryption and authentication, which will resist quantum attacks. One family of such primitives, proposed by De Feo, Jao, and Plût \cite{9}, commonly referred to as SIDH, uses isogenies between supersingular elliptic curves to construct quantum-resistant cryptographic protocols for public-key encryption, two-party key establishment.

Password-Authenticated Key Exchange (PAKE) is a primitive in which parties securely establish a common cryptographic key over an insecure channel using a common password (modelled as a low-entropy secret). The first PAKE protocol was designed by Bellovin and Merritt in 1992 \cite{3}. Today, many protocols of this type exist—most are based on either multiplicative group $\mathbb{Z}_p^*$ or on a group of points of elliptic curves and are not quantum-safe. Until this work, the only PAKEs built on quantum-safe foundations are lattice-based \cite{10,22}. We propose the first PAKE based on isogenies between supersingular elliptic curves. It is derived from SIDH and its speed characteristics are very similar.

One of the most natural applications of PAKE is establishing common session secret key for mutual authentication (and, as an option, further opening a secure channel) between a smartcard and a terminal with keyboard for PIN. Note that terminal and smartcard reader can be physically separated, so payment industry standards require secure channel between them in order to prevent PIN leakage. Another important case of practical usage of PAKE is creating secure channel between process on computer and USB token (or smartcard) inserted into it. There is a number of other applications of PAKE, which include its use for client-server authentication and secure communication.
2 Background

2.1 Isogenies

We provide a brief review of the necessary background information. For further details on the mathematical foundations of isogenies, we refer the reader to [9, 12, 19].

Given two elliptic curves $E_1$ and $E_2$ over some finite field $\mathbb{F}_q$ of cardinality $q$, an isogeny $\phi$ is an algebraic morphism from $E_1$ to $E_2$ of the form

$$\phi(x, y) = \left( \frac{f_1(x, y)}{g_1(x, y)}, \frac{f_2(x, y)}{g_2(x, y)} \right),$$

such that $\phi(\infty) = \infty$ (here $f_1, f_2, g_1, g_2$ are polynomials in two variables, and $\infty$ denotes the identity element on an elliptic curve). Equivalently, an isogeny is an algebraic morphism which is a group homomorphism.

The degree of $\phi$, denoted $\deg(\phi)$, is its degree as an algebraic morphism. Two elliptic curves are isogenous if there exists an isogeny between them.

Given an isogeny $\phi: E_1 \rightarrow E_2$ of degree $n$, there exists another isogeny $\hat{\phi}: E_2 \rightarrow E_1$ of degree $n$ satisfying $\phi \circ \hat{\phi} = \hat{\phi} \circ \phi = [n]$ (where $[n]$ is the multiplication by $n$ map). It follows that the relation of being isogenous is an equivalence relation. The isogeny $\hat{\phi}$ is called the dual isogeny of $\phi$.

For any natural number $n$, we define $E[n]$ to be the subgroup $E[n] = \{ P \in E(\bar{\mathbb{F}}_q) : nP = \infty \}$. In other words, $E[n]$ is the kernel of the multiplication by $n$ map over the algebraic closure $\bar{\mathbb{F}}_q$ of $\mathbb{F}_q$. The group $E[n]$ is isomorphic to $(\mathbb{Z}/m\mathbb{Z})^2$ as a group whenever $m$ and $q$ are relatively prime [19].

We define the endomorphism ring $\text{End}(E)$ to be the set of all isogenies from $E$ to itself, defined over the algebraic closure $\bar{\mathbb{F}}_q$ of $\mathbb{F}_q$. The endomorphism ring is a ring under the operations of pointwise addition and functional composition. If $\dim_{\mathbb{Z}}(\text{End}(E)) = 2$, then we say that $E$ is ordinary; otherwise $\dim_{\mathbb{Z}}(\text{End}(E)) = 4$ and we say that $E$ is supersingular. Two isogenous curves are either both ordinary or both supersingular. All elliptic curves used in this work are supersingular.

The isogeny $\phi: E_1 \rightarrow E_2$ is defined to be separable if the extension $\bar{\mathbb{F}}_q(E_1)/\phi^*(\bar{\mathbb{F}}_q(E_2))$ of function fields is separable. In this work, we will only consider separable isogenies. An important property of a separable isogeny is that the size of the kernel of that isogeny is equal to the degree of that isogeny (as an algebraic map) [19, III.4.10(c)]. The kernel uniquely defines the isogeny up to isomorphism. Methods for computing and evaluating isogenies are given in [5, 9, 12, 13, 21]. We use the isogenies whose kernels are cyclic groups, and knowledge of the kernel, or any single generator of the kernel, allows for efficient evaluation of the isogeny (up to isomorphism); conversely, the ability to evaluate the isogeny via a black box allows for efficient determination of the kernel. Thus, in our application, the following are equivalent: knowledge of the isogeny, knowledge of the kernel, or knowledge of any generator of the kernel.

2.2 Isogeny-Based Key Establishment

The term “elliptic curve cryptography” typically encompasses cryptographic primitives and protocols whose security is based on the hardness of the discrete logarithm problem on elliptic curves. This hardness assumption is invalid against quantum computers [18]. Hence, traditional elliptic curve cryptography is not a viable foundation for constructing quantum-resistant cryptosystems. As a result, alternative elliptic curve cryptosystems based on hardness assumptions other than discrete logarithms have been proposed for use in settings where quantum resistance is desired. One early proposal by Stolbunov [20], based on isogenies between ordinary elliptic curves, was subsequently shown by Childs et al. [7] to offer only subexponential security against quantum computers.

In response to these developments, In PQCrypto 2011, Jao and De Feo [9, 12] proposed a new collection of quantum-resistant public-key cryptographic protocols for entity authentication, key exchange, and public-key cryptography, based on the difficulty of computing isogenies between supersingular elliptic curves. We review here the operation of the most fundamental protocol in the collection, the key exchange protocol, which forms the building block for our password-authenticated key exchange protocol.


\[ E_{AB} := E_B / (m_A \phi_B(P_A) + n_A \phi_B(Q_A)) \]

**Output:** \( j(E_{AB}) \)

---

**Fig. 1.** Key Establishment protocol using isogenies on supersingular curves.

Fix a prime \( p \) of the form \( \ell^e_A \ell^e_B \cdot f \pm 1 \) where \( \ell_A \) and \( \ell_B \) are small primes, \( e_A \) and \( e_B \) are positive integers, and \( f \) is some (typically very small) cofactor. Also, fix a supersingular curve \( E \) defined over \( \mathbb{F}_p^2 \), and bases \( \{ P_A, Q_A \} \) and \( \{ P_B, Q_B \} \) which generate \( E[\ell^e_A] \) and \( E[\ell^e_B] \) respectively, so that \( \langle P_A, Q_A \rangle = E[\ell^e_A] \) and \( \langle P_B, Q_B \rangle = E[\ell^e_B] \). Alice chooses two random elements \( m_A, n_A \in \mathbb{Z}/\ell^e_A \mathbb{Z} \), not both divisible by \( \ell_A \), and computes an isogeny \( \phi_A: E \to E_A \) with kernel \( K_A := \langle m_A P_A + n_A Q_A \rangle \). Alice also computes the auxiliary points \( \{ \phi_A(P_B), \phi_A(Q_B) \} \subset E_A \) obtained by applying her secret isogeny \( \phi_A \) to the basis \( \{ P_B, Q_B \} \) for \( E[\ell^e_B] \), and sends these points to Bob together with \( E_A \). Similarly, Bob selects random elements \( m_B, n_B \in \mathbb{Z}/\ell^e_B \mathbb{Z} \) and computes an isogeny \( \phi_B: E \to E_B \) having kernel \( K_B := \langle m_B P_B + n_B Q_B \rangle \), along with the auxiliary points \( \{ \phi_B(P_A), \phi_B(Q_A) \} \). Upon receipt of \( E_B \) and \( \phi_B(P_A), \phi_B(Q_A) \in E_B \) from Bob, Alice computes an isogeny \( \phi'_A: E_B \to E_{AB} \) having kernel equal to \( \langle m_A \phi_B(P_A) + n_A \phi_B(Q_A) \rangle \): Bob proceeds \textit{mutatis mutandis}.

Alice and Bob can then use the common \( j \)-invariant of

\[ E_{AB} = \phi'_B(\phi_A(E)) = \phi'_B(\phi_B(E)) = E / (m_A P_A + n_A Q_A, m_B P_B + n_B Q_B) \]

to form a secret shared key.

The full protocol is presented in Figure 1. We denote by \( A \) and \( B \) the identifiers of Alice and Bob.

**Remark 1.** Alice’s auxiliary points \( \{ \phi_A(P_B), \phi_A(Q_B) \} \) allow Bob (or any eavesdropper) to compute Alice’s isogeny \( \phi_A \) on any point in \( E[\ell^e_B] \). This ability is necessary in order for the scheme to function, since Bob needs to compute \( \phi_A(K_B) \) as part of the scheme. However, Alice must never disclose \( \phi_A(P_A) \) or \( \phi_A(Q_A) \) (or more generally any information that allows an adversary to evaluate \( \phi_A \) on \( E[\ell^e_A] \)), since disclosing this information would allow the adversary to solve a system of discrete logarithms in \( E[\ell^e_A] \) (which are easy since \( E[\ell^e_A] \) has smooth order) to recover \( K_A \).

### 2.3 Computational Assumptions for Supersingular Elliptic Curve Isogenies

In this section we define a number of computational assumptions related to computing supersingular elliptic curve isogenies. Throughout this section, we use the notation of Section 2.2 for our global parameters, and define the security parameter \( \lambda = \lfloor \log_2 p \rfloor \). In [8] was shown effective trick to make SIDH faster: the kernel group generator \( mP + nQ \) can be replaced by \( P + mQ \) essentially without loss of generality. Our protocol will use \( P + nQ \) form of kernel generator.

We begin with the fundamental computational problem which underlies SIDH.

**Definition 1 (Computational Supersingular Isogeny Diffie-Hellman Problem (C-SIDH)).** Let \( \phi_A: E \to E_A \) be an isogeny with kernel \( \langle P_A + n_A Q_A \rangle \) where \( n_A \leftarrow U(\mathbb{Z}/\ell^e_A \mathbb{Z}) \). Similarly, let \( \phi_B: E \to E_B \) be an isogeny with kernel \( \langle P_B + n_B Q_B \rangle \) where \( n_B \leftarrow U(\mathbb{Z}/\ell^e_B \mathbb{Z}) \). The supersingular computational Diffie-Hellman problem (C-SIDH) is to find the \( j \)-invariant of

\[ E_{AB} = E / \langle P_A + n_A Q_A, P_B + n_B Q_B \rangle \]

given \((E, P_A, Q_A, P_B, Q_B), (E_A, \phi_A(P_B), \phi_A(Q_B)), (E_B, \phi_B(P_A), \phi_B(Q_A)).\)
When \( ((E, P_A, Q_A, P_B, Q_B), (E_A, \phi_A(P_B), \phi_A(Q_B)), (E_B, \phi_B(P_A), \phi_B(Q_A))) \) is a valid input to the C-SIDH problem, using the notation of Definition 1, we define

\[
\text{SIDH}((E, P_A, Q_A, P_B, Q_B), (E_A, \phi_A(P_B), \phi_A(Q_B)), (E_B, \phi_B(P_A), \phi_B(Q_A))) = E_{AB}.
\]

When the global parameters and auxiliary points can be inferred from context, we abbreviate this quantity as \( E_{AB} = \text{SIDH}(E_A, E_B) \). We also define “one-sided” variants of SIDH:

- For \( E_A = E / (P_A + m_A Q_A) \) and any \( (E_B, X_B, Y_B) \), define \( \text{SIDH}_A(E_A, E_B, X_B, Y_B) = E_B / (X_B + m_A Y_B) \)
- For \( E_B = E / (P_B + m_B Q_B) \) and any \( (E_A, X_A, Y_A) \), define \( \text{SIDH}_B(E_A, E_B, X_A, Y_A) = E_A / (X_A + m_B Y_A) \)

These quantities arise naturally in SIDH by considering messages which are not necessarily well-formed.

For an algorithm \( A \) which, given a valid C-SIDH instance, produces a list of candidate solutions to the instance, define its advantage as

\[
\text{Adv}^{C\text{-SIDH}}_p(A) = P[\vartheta \leftarrow U(\text{Ins}_{\text{SIDH}}(p)) : \text{SIDH}(\vartheta) \in A(\vartheta)]
\]

where \( \text{Ins}_{\text{SIDH}}(p) \) is the set of all valid C-SIDH instances defined over \( GF(p^2) \). Further, define

\[
\text{Adv}^{C\text{-SIDH}}_{p}(t, n) = \max\{\text{Adv}^{C\text{-SIDH}}_p(A) : A \text{ runs in time } t \text{ and } |A(\vartheta)| \leq n \text{ for all } \vartheta \in \text{Ins}_{\text{SIDH}}(p)\}.
\]

Intuitively, this quantity measures the maximum probability of solving a randomly-chosen C-SIDH instance in time \( t \) if you are allowed to make \( n \) guesses. The C-SIDH assumption is that for \( t \) and \( n \) polynomial in \( \lambda \), \( \text{Adv}^{C\text{-SIDH}}_{p}(t, n) = \negl(\lambda) \).

As noted in Section 2.2, in order to compute the shared secret in SIDH the parties must share the images of the public torsion bases under the secret isogenies. Whereas previous works ignored these “auxiliary points” in favour of standard authentication methods, such as signature schemes [15] or generic transforms [11] for authenticating SIDH and preventing man-in-the-middle attacks, in this work we focus on their role in the computation and how we can disrupt man-in-the-middle attacks by obfuscating them.

Naturally, because our construction relies on obfuscating the auxiliary points used in the scheme, we require some computational assumptions related to computing auxiliary points given the relevant curves. We present these computational assumptions here. Notably, because of the inherent asymmetry of the auxiliary point computations, we will specify two computational problems—one for each \( \ell^a_B \) and \( \ell^b_B \)-isogenies.

**Definition 2 (Supersingular Isogeny Auxiliary Point Computation–A (SI-APC_A)).** Let \( \phi_A : E \rightarrow E_A \) be an isogeny with kernel \( (P_A + n_A Q_A) \) where \( n_A \leftarrow U(\mathbb{Z}/\ell^a_A \mathbb{Z}) \). The supersingular isogeny auxiliary point computation problem (type A) is to compute \( \phi_A(P_B) \) and \( \phi_A(Q_B) \) given \( (E, P_A, Q_A, P_B, Q_B, E_A) \).

**Definition 3 (Supersingular Isogeny Auxiliary Point Computation–B (SI-APC_B)).** Let \( \phi_B : E \rightarrow E_B \) be an isogeny with kernel \( (P_B + n_B Q_B) \) where \( n_B \leftarrow U(\mathbb{Z}/\ell^b_B \mathbb{Z}) \). The supersingular isogeny auxiliary point computation problem (type B) is to compute \( \phi_B(P_A) \) and \( \phi_B(Q_A) \) given \( (E, P_A, Q_A, P_B, Q_B, E_B) \).

Using the notation above, we define

\[
\text{SI-APC}_A((E, P_A, Q_A, P_B, Q_B, E_A)) = (\phi_A(P_B), \phi_A(Q_B)) , \text{ and}
\]

\[
\text{SI-APC}_B((E, P_A, Q_A, P_B, Q_B, E_B)) = (\phi_B(P_A), \phi_B(Q_A)) .
\]

We also define the advantages

\[
\{ \text{Adv}^{\text{SI-APC}_A}_p(A) = P[\vartheta \leftarrow U(\text{Ins}_{\text{SI-APC}_A}(p)) : \text{SI-APC}_A(\vartheta) \in A(\vartheta)] \}
\]

\[
\{ \text{Adv}^{\text{SI-APC}_B}_p(A) = P[\vartheta \leftarrow U(\text{Ins}_{\text{SI-APC}_B}(p)) : \text{SI-APC}_B(\vartheta) \in A(\vartheta)] \}
\]

and

\[
\left\{ \begin{array}{l}
\text{Adv}^{\text{SI-APC}_A}_{p}(t, n) = \max\{\text{Adv}^{\text{SI-APC}_A}_p(A) : A \text{ runs in time } t \text{ and } |A(\vartheta)| \leq n \text{ for all } \vartheta \in \text{Ins}_{\text{SI-APC}_A}(p)\}
\end{array} \right.
\]

\[
\left\{ \begin{array}{l}
\text{Adv}^{\text{SI-APC}_B}_{p}(t, n) = \max\{\text{Adv}^{\text{SI-APC}_B}_p(A) : A \text{ runs in time } t \text{ and } |A(\vartheta)| \leq n \text{ for all } \vartheta \in \text{Ins}_{\text{SI-APC}_B}(p)\}.
\end{array} \right.
\]

The SI-APC_A and SI-APC_B assumptions are that for \( t \) and \( n \) polynomial in \( \lambda \), \( \text{Adv}^{\text{SI-APC}_A}_{p}(t, n) = \negl(\lambda) \) and \( \text{Adv}^{\text{SI-APC}_B}_{p}(t, n) = \negl(\lambda) \), respectively. In this quantum setting, these problems can be phrased in a
more natural way: instead of looking for the images of a particular torsion basis, one can ask to search for the image of any appropriate torsion point. This equivalence is because Shor’s algorithm can be extended to solve extended discrete logarithms in Abelian groups, and so being able to find the auxiliary points allows one to find the image of any point in the appropriate torsion subgroup (and the reverse reduction is trivial). In that sense, in the quantum setting these problems can be thought of as natural easier variants of the Supersingular Isogeny Problems (Types A and B):

**Definition 4 (Supersingular Isogeny Problem–A (SSI_A)).** Let \( \phi_A : E \rightarrow E_A \) be an isogeny with kernel \( \langle P_A + n_AQ_A \rangle \) for \( n_A \leftarrow U(\mathbb{Z}/\ell_A^e \mathbb{Z}) \). The supersingular isogeny problem (type A) (SSI_A) is, given \( E, E_A, \phi_A(P_B), \) and \( \phi_A(Q_B) \), to find a generator of \( \ker \phi_A \).

**Definition 5 (Supersingular Isogeny Problem–B (SSI_B)).** Let \( \phi_B : E \rightarrow E_B \) be an isogeny with kernel \( \langle P_B + n_BQ_B \rangle \) for \( n_B \leftarrow U(\mathbb{Z}/\ell_B^e \mathbb{Z}) \). The supersingular isogeny problem (type B) (SSI_B) is, given \( E, E_B, \phi_B(P_A), \) and \( \phi_B(Q_A) \), to find a generator of \( \ker \phi_B \).

The SI-APC_A and SI-APC_B problems reduce to SSI_A and SSI_B, respectively, by noting that finding a generator of the kernel of an isogeny \( \phi \) allows one to compute the isogeny on the whole domain curve, by Vélu’s formulas \([21]\), whereas SI-APC_A and SI-APC_B simply require one to compute the isogeny on particular points (or, in the quantum setting, the restrictions \( \phi_{E|E_A}^{\ell_A^e} \) or \( \phi_{E|E_B}^{\ell_B^e} \)).

We will also make use of the decisional version of the auxiliary point computation problems, and so we present them here.

**Definition 6 (Supersingular Isogeny Auxiliary Point Decision–A (SI-APD_A)).** Let \( \phi_A : E \rightarrow E_A \) be an isogeny with kernel \( \langle P_A + n_AQ_A \rangle \) where \( n_A \leftarrow U(\mathbb{Z}/\ell_A^e \mathbb{Z}) \). The supersingular isogeny auxiliary point decision problem (type A) is, given given \( E, P_A, Q_A, P_B, Q_B, E_A \) and two points \( (X, Y) \) on \( E_A \) with \( e_{E_A}(X, Y) = e_E(P_B, Q_B) \ell_A^e \) where either

1. \( X = \phi_A(P_B) \) and \( Y = \phi_A(Q_B) \), or
2. \( (X, Y) \leftarrow U\{\{(S, T) \in E_A[\ell_A^e] \setminus \mathbb{Z}^2 : e_{E_A}(X, Y) = e_E(P_B, Q_B) \ell_A^e \text{ and } \text{ord}_{E_A}(X) = \text{ord}_{E_A}(Y) = \ell_A^e \} \}

(each with probability \( \frac{1}{2} \)) is to decide which is the case.

**Definition 7 (Supersingular Isogeny Auxiliary Point Decision–B (SI-APD_B)).** Let \( \phi_B : E \rightarrow E_B \) be an isogeny with kernel \( \langle P_B + n_BQ_B \rangle \) where \( n_B \leftarrow U(\mathbb{Z}/\ell_B^e \mathbb{Z}) \). The supersingular isogeny auxiliary point decision problem (type B) is, given given \( E, P_B, B_B, P_A, B_A, E_B \) and two points \( (X, Y) \) on \( E_B \) with \( e_{E_B}(X, Y) = e_E(P_A, B_A) \ell_B^e \) where either

1. \( X = \phi_B(P_A) \) and \( Y = \phi_B(B_A) \), or
2. \( (X, Y) \leftarrow U\{\{(S, T) \in E_B[\ell_B^e] \setminus \mathbb{Z}^2 : e_{E_B}(X, Y) = e_E(P_A, B_A) \ell_B^e \text{ and } \text{ord}_{E_B}(X) = \text{ord}_{E_B}(Y) = \ell_B^e \} \}

(each with probability \( \frac{1}{2} \)) is to decide which is the case.

As before, we define the advantages

\[
\begin{align*}
\text{Adv}^{\text{SI-APD}_A}(A) &= \mathbb{P}[\vartheta \leftarrow \text{Ins}_{\text{SI-APC}_A}(p) : A(\vartheta) \text{ is correct}] \\
\text{Adv}^{\text{SI-APD}_B}_p(A) &= \mathbb{P}[\vartheta \leftarrow \text{Ins}_{\text{SI-APC}_B}(p) : A(\vartheta) \text{ is correct}]
\end{align*}
\]

and

\[
\begin{align*}
\text{Adv}^{\text{SI-APD}_A}(t, n) &= 2\max\{\text{Adv}^{\text{SI-APC}_A}(A) : A \text{ runs in time } t \text{ for all } \vartheta \in \text{Ins}_{\text{SI-APD}_A}(p)\} - 1 \\
\text{Adv}^{\text{SI-APD}_B}_p(t, n) &= 2\max\{\text{Adv}^{\text{SI-APC}_B}_p(A) : A \text{ runs in time } t \text{ for all } \vartheta \in \text{Ins}_{\text{SI-APD}_B}(p)\} - 1.
\end{align*}
\]

The corresponding computational assumptions are that for \( t \) polynomial in \( \lambda \), \( \text{Adv}^{\text{SI-APD}_A}(t) = \negl(\lambda) \) and \( \text{Adv}^{\text{SI-APD}_B}_p(t) = \negl(\lambda) \).

Finally, we must discuss a computational problem which, in the context of our protocol, will be related to password-guessing. This computational problem will be related to certain group actions of a matrix group on groups related to our PAKE. We cover the relevant topics here.
For a prime $\ell$ and an integer $e$, we define
\[
\text{SL}_2(\ell, e) = \{ \Psi \in (\mathbb{Z}/\ell^e\mathbb{Z})^{2 \times 2} : \det A \equiv 1 \mod \ell^e \}
\]
\[
\mathcal{T}_2(\ell, e) = \{ \Psi \in \text{SL}_2(\ell, e) : A \text{ is upper triangular modulo } \ell \}
\]
to be the special linear and special reduced upper triangular groups modulo $\ell^e$, respectively. If $p = \ell_A^e \ell_B^p f \pm 1$ is a prime and $E$ is a supersingular elliptic curve defined over $\mathbb{F}_{q^2}$, $\mathcal{T}_2(\ell_A, \ell_B) = \text{SL}_2(\ell_A, \ell_B)$ acts on $E[\ell_A^e \ell_B^p]$ in a way completely analogous with ordinary matrix-vector multiplication over a ring: if $\Psi = \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix}$ then
\[
\psi \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} \alpha X + \beta Y \\ \gamma X + \delta Y \end{bmatrix}.
\]

It is this action that we will use to obfuscate the auxiliary points in our protocol. This action has two important qualities which make it amenable to our PAKE (in particular, they do not enable a particular kind of offline dictionary attack):

1. If $\{X, Y\}$ is a $\mathbb{Z}$-basis for $E[\ell_A^e \ell_B^p]$, then so is $\{X', Y'\}$, defined by $\begin{bmatrix} X' \\ Y' \end{bmatrix} = \Psi^{-1} \begin{bmatrix} X \\ Y \end{bmatrix}$ for any $\Psi, \Psi' \in \mathcal{T}_2(\ell_A, \ell_B)$.
2. For any $X, Y \in E[\ell_A^e \ell_B^p]$ and for any $\Psi, \Psi' \in \mathcal{T}_2(\ell_A, \ell_B)$, for $\begin{bmatrix} X' \\ Y' \end{bmatrix} = \Psi^{-1} \begin{bmatrix} X \\ Y \end{bmatrix}$ we have $e(X, Y) = e(X', Y')$, where $e$ is the Weil pairing on $E$. Moreover, $\text{SL}_2(\ell_A, \ell_B)$ is the unique maximal matrix group acting as defined above which has this property.

In the context of isogeny computations, this (left) action of $\mathcal{T}_2(\ell_A, \ell_B)$ induces a new (right) action on $\mathbb{Z}/\ell_A^e \mathbb{Z}$ in the following way: observe that if $\begin{bmatrix} X' \\ Y' \end{bmatrix} = \Psi \begin{bmatrix} X \\ Y \end{bmatrix}$ where $\Psi = \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix}$, then for any $m \in \mathbb{Z}/\ell_A^e \mathbb{Z}$
\[
E/\langle X' + mY' \rangle = E/\langle (\alpha X + \beta Y) + m(\gamma X + \delta Y) \rangle
= E/\langle X + \gamma m + \alpha \rangle;
\]
we thus might naturally define the right action $m \Psi := \begin{bmatrix} \gamma m + \alpha \\ \delta m + \beta \end{bmatrix}$. The two actions are related as described above: mapping the auxiliary basis according to the action of $\Psi$ on $E[\ell_A^e \ell_B^p]^2$ is equivalent to mapping the integer $m$ which defines the kernel of the isogeny according to the action of $\Psi$ on $\mathbb{Z}/\ell_A^e \mathbb{Z}$. This action relates to a sort of “related-key attack” on SIDH: in an unauthentiated setting, an active adversary can force an honest party to compute a shared key corresponding to an ephemeral key chosen by the adversary and the image of the honest party’s ephemeral key under the action of a known matrix $\Psi$.

Note that for a given $E_A$ for which $E_A = E/\langle P_A + m_A Q_B \rangle$ and $\Psi \in \mathcal{T}(\ell_A, \ell_B)$ it is easy to find $(E_B, X_B, Y_B), E'$ satisfying
\[
E' = E_B/\langle X_B + m_A^\Psi Y_B \rangle;
\]
simply set $E_B = E/\ker \phi$ for some isogeny $\phi$, and $[X_B, Y_B]^T = \Psi^{-1} \phi(P_B), \phi(Q_B)]^T$ (and of course, this works just the same with $A$ and $B$ switched). In the context of our protocol, this computation corresponds to guessing a single password in an online attack, and is essentially unavoidable in any PAKE protocol. What we must prevent is online guessing of two or more passwords in a single session. The following computational problems underly such online guessing:

**Definition 8 (Computational Simultaneous Group Action Problem–A (C-SGA$_A$))**. Let $\phi_A : E \rightarrow E_A$ be an isogeny with kernel $\langle P_A + m_A Q_B \rangle$ for $n_A \leftarrow U(\mathbb{Z}/\ell_A^e \mathbb{Z})$, and let $\Psi_1, \Psi_2 \leftarrow U(\mathcal{T}_2(\ell_A, \ell_B))$. The Computational Simultaneous Group Action Problem (Type A) (C-SGA$_A$) is to find $(E_B, X_B, Y_B), E_1, E_2$ which satisfy $E_1 = E_B/\langle X_B + m_A^\Psi Y_B \rangle$ and $E_2 = E_B/\langle X_B + m_A^\Psi Q_B \rangle$ given $(E, P_B, Q_B)$ and $(E_A, \phi_A(P_B), \phi_A(Q_B))$.

**Definition 9 (Computational Simultaneous Group Action Problem–B (C-SGA$_B$))**. Let $\phi_B : E \rightarrow E_B$ be an isogeny with kernel $\langle P_B + m_B Q_B \rangle$ for $n_B \leftarrow U(\mathbb{Z}/\ell_B^e \mathbb{Z})$, and let $\Psi_1, \Psi_2 \leftarrow U(\mathcal{T}_2(\ell_B, \ell_B))$. The Computational Simultaneous Group Action Problem (Type B) (C-SGA$_B$) is to find $(E_A, X_A, Y_A), E_1, E_2$ which satisfy $E_1 = E_A/\langle X_A + m_B^\Psi Y_A \rangle$ and $E_2 = E_A/\langle X_A + m_B^\Psi Q_A \rangle$ given $(E, P_A, Q_A)$ and $(E_B, \phi_B(P_A), \phi_B(Q_A))$. 
We define $SGA_A((E, P_A, Q_A, P_B, Q_B, E_A, \Psi_1, \Psi_2))$ and $SGA_B((E, P_A, Q_A, P_B, Q_B, E_B, \Psi_1, \Psi_2))$ to be the set of all solutions to the respective problems with the given parameters, and we define the advantages

$$
\begin{align*}
\text{Adv}^{SGA_A}_p &= \mathbb{P}[\theta \leftarrow \text{Ins}_{SLAPC_A}(p) : SGA_A(\theta) \cap A(\theta) \neq \emptyset] \\
\text{Adv}^{SGA_B}_p &= \mathbb{P}[\theta \leftarrow \text{Ins}_{SLAPC_B}(p) : SGA_B(\theta) \cap A(\theta) \neq \emptyset]
\end{align*}
$$

and

$$
\begin{align*}
\text{Adv}^{SGA_A}_{t,n} &= \max\{\text{Adv}^{SGA_A}_p(A) : A \text{ runs in time } t \text{ and } |A(\theta)| \leq n \text{ for all } \theta \in \text{Ins}_{SGA_A}(p)\} \\
\text{Adv}^{SGA_B}_{t,n} &= \max\{\text{Adv}^{SGA_B}_p(A) : A \text{ runs in time } t \text{ and } |A(\theta)| \leq n \text{ for all } \theta \in \text{Ins}_{SGA_B}(p)\}.
\end{align*}
$$

Again, these computational problems reduce to the corresponding isogeny computation problems, by noting that one can recover the secret integer $m_A$ or $m_B$ from a generator of ker $\phi_A$ or ker $\phi_B$, and from there can compute $E/(X + m_A^\Psi Y)$ or $E/(X + m_B^\Psi Y)$ for any $\Psi$ and any $(E, X, Y)$.

2.4 Applications

Password-authenticated key establishment has many applications and is in wide use. There is a variety of PAKE-based protocols and standards, as well as program implementations within certain well-known and widely used cryptographic libraries such as OpenSSL [17] and Bouncy Castle [4]. Given emerging quantum threat and ubiquity of PAKEs, it is imperative to construct a post-quantum PAKE.

One of the main PAKE applications is smartcard- or token-based security. Generally, software that interacts with smartcards and USB tokens mainly use standard programming interfaces which reduce duplicate code while performing the same job and enables the program to support devices from different vendors.

ISO 7816 and PKCS#11 are the two most popular and de facto mandatory standards for device vendors meaning that a smartcard/token manufacturer will support at least one of those standards. ISO 7816 is a standard for establishing communication with a smartcard on a very low (signal) level. Among other things, this standard defines the structure of the basic level APDU commands used to interact with the device. Most vendors manufacture devices with the same APDU commands for such simple actions as selecting, reading, or writing files. Most likely, the APDUs for more complex actions, such as generating a key pair, will be different: in those cases, PKCS#11 should be used, as its library encapsulates APDUs. Besides, PKCS#11 is a high-level standard which uses C language to describe function prototypes for interacting with abstract devices that can execute cryptographic operations and store keys. Such abstraction helps programmers to concentrate on cryptography-related tasks without having to deal with the details of platform-related APDUs or smartcard filesystems.

There are two distinct principles of interaction between a computer program and a smartcard/token connected to a computer, with or without establishing a secure channel. Without a secure channel, the APDU transmission speed is higher, because the device does not need time for encryption, decryption, and MAC calculations. Also, it is easier to support the host software and the devices microprogram. The main disadvantage of the absence of a secure channel is the vulnerability to an attack: all APDUs can be eavesdropped or maliciously modified by an attacker. The attacker can obtain such secret information as PIN and symmetric session key by eavesdropping. The attacker can also force a party to generate a digital signature. Therefore, the communication channel should be protected.

One of the practical advantages of PAKE is that it can be “hidden” inside high-level library (like PKCS#11 or CSP(Cryptographic Service Provider)) without changing library’s interface functions, so that developer who integrates it into program, don’t have to care about implementation of PAKE protocol and simply can use just the same calls of library functions that work with non-PAKE smartcard or token.

2.5 Security Model and Definitions

We prove the security of our protocol in the model of Bellare, Pointcheval, and Rogaway [1]. We review the essential features of the model here.

**Protocols.** Fundamentally, a protocol $\Pi$ is a probabilistic algorithm which maps strings (in the context of password-authenticated key establishment, these strings will be a concatenation of passwords, randomness, and protocol-specific messages and global parameters) to strings (keys).
Parties and Party Identifiers. Participants in protocols are called parties. Parties come in two distinct flavours: clients $\mathcal{C}$ and servers $\mathcal{S}$. As the names suggest, in this model, protocols are always initiated by clients and responded to by servers. All clients are served by all servers.

Each party $P \in \mathcal{C} \sqcup \mathcal{S}$ is uniquely identified by a string of some fixed length (depending on the number of parties); if $P$ is a party, we will also use $P$ to refer to $P$’s identifier string when no confusion will arise (in particular, as arguments to functions).

Passwords and Transformed Passwords. Each client $A \in \mathcal{C}$ has a password $\text{pwd}_A$, and each server has a collection $\{\text{pwd}_B^A\}_{A \in \mathcal{C}}$ of transformed passwords. We may consider either a symmetric scenario, where $\text{pwd}_B^A = \text{pwd}_A \forall A \in \mathcal{C}$, $B \in \mathcal{S}$ (i.e., servers know their clients’ passwords) or an asymmetric scenario, where servers store an actually-transformed password. In the asymmetric setting it is often desirable for it to be difficult to recover $\text{pwd}_A$ from a reasonably-sized\textsuperscript{6} subset of $\{\text{pwd}_B^A\}_{B \in \mathcal{S}}$, so that the protocol is better able to resist server information leakage.

Password Initialization. In models of authenticated key establishment, parties establish public-key/private-key pairs and securely publish their public keys (i.e., servers know their clients’ passwords) or an asymmetric scenario, where servers generate passwords and “install” them on servers (i.e., servers get the corresponding transformed passwords) securely.

Party Instances. Associated to each party $U \in \mathcal{C} \sqcup \mathcal{S}$ is a collection of party instances $\{\Omega_U^{(n)}\}_{n \in \mathbb{N}}$. When the adversary interacts with a party (in one of the ways described below) he may be required to specify an instance with which to interact. Intuitively, these instances model the fact that parties may establish many keys at different times and with different partners, and these different key-establishing sessions may be attacked differently (or not at all) by the adversary. As a practical matter, explicitly modelling party instances make it easier compartmentalize keys established by the parties (and their associated private ephemeral information), and simplify the security definition.

Acceptance and Termination. We say that a party instance “accepts” when they compute a session key, and “terminates” when it will not send any more messages. In our protocol, acceptance is always followed immediately by termination, but termination may occur without acceptance (in the case of ill-formed incoming messages, for instance).

The Security Experiment. Informally, the security experiment sees a new entity—the adversary—attempt to break semantic security of the protocol after interacting with the parties and an additional “formal” party who “administrates” the game—the challenger. The security experiment proceeds in three stages:

1. Initialization: The adversary chooses disjoint sets $\mathcal{C}$ and $\mathcal{S}$ of client and server identifiers. Each client $A \in \mathcal{C}$ generates a password according to a probability distribution on their password-space: $\text{pwd}_A \leftarrow \mathcal{P}_A \forall A \in \mathcal{C}$; then, each server $B \in \mathcal{S}$ computes the transformed passwords $\{\text{pwd}_B^A\}_{A \in \mathcal{C}}$. Password-generation happens out-of-view of the adversary, though the adversary knows each $\mathcal{P}_A$ and the maps $\text{pwd}_A \mapsto \text{pwd}_B^A$.

2. Information Gathering: During this phase, the adversary is allowed to perform computations on his own machine and interact with the parties and challenger. The computational power of the adversary is not fixed in the model, but his interactive abilities are limited to the following five interaction commands:

   (a) $\text{Send}(U, n, \text{msg})$: Message $\text{msg}$ is sent to $\Omega_U^{(n)}$. As a result, $\Omega_U^{(n)}$’s internal state (including which step in the protocol it has reached, its ephemeral key for this instance, etc.) is updated appropriately according to the specification of $\Pi$.

   (b) $\text{Reveal}(U, n)$: The party instance $\Omega_U^{(n)}$ reveals its session key $\text{sk}_U^{(n)}$ (if it exists).

   (c) $\text{Execute}(A, n, B, m)$: If $A \in \mathcal{C}$, $B \in \mathcal{S}$, and neither $\Omega_A^{(n)}$ nor $\Omega_B^{(m)}$ has been used, the challenger $\mathcal{C}$ instructs $\Omega_A^{(n)}$ to execute $\Pi$ with $\Omega_B^{(n)}$. The transcript of this execution is then provided to $\mathcal{A}$.

---

\textsuperscript{6} What size is “reasonable” depends on how powerful an adversary we consider. Typically we would consider polynomially-sized sets, but we do not assume this a priori. Similarly, what is meant by “difficult” is context-dependent.
pwd

points. Our scheme uses the password to generate coefficients which scramble the auxiliary points. That original scheme, the public information sent by each user consists of the image curve and two auxiliary points.

The protocol is based on the (ephemeral) SIDH scheme for key establishment by De Feo, Jao, and Plût [9]. In Section 3

3. The Game: To begin, we must define what it means for an oracle \( \Omega^{(n)}_{U} \) to be fresh, which itself requires that we define partnered sessions:

Definition 10 (Partnered Oracles). A pair of oracles \( \Omega^{(n)}_{A} \) and \( \Omega^{(m)}_{B} \) are called partnered if all of the following are true:

(a) \( A \in C \) and \( B \in \mathcal{S} \), or \( A \in \mathcal{S} \) and \( B \in C \).

(b) \( \Omega^{(n)}_{A} \) and \( \Omega^{(m)}_{B} \) have both accepted.

(c) \( \Omega^{(n)}_{A} \)'s peer is \( B \), and \( \Omega^{(m)}_{B} \)'s peer is \( A \) (that is, \( A \) believes she is establishing a key with \( B \), and vice versa).

(d) \( \Omega^{(n)}_{A} \) and \( \Omega^{(m)}_{B} \) have the same message transcript and session key.

(e) No other oracle \( \Omega^{(k)}_{U} \) accepts with the same message transcript.

Definition 11 (Fresh Oracle). An oracle \( \Omega^{(n)}_{U} \) is called fresh (or, to emphasize that we are explicitly considering forward secrecy in the weak corruption model, weak forward secrecy fresh) if none of the following is true:

(a) \( \text{Reveal}(U, n) \) has been issued.

(b) \( \text{Reveal}(V, m) \) has been issued, where \( \Omega^{(m)}_{V} \) is the partner to \( \Omega^{(n)}_{U} \).

(c) \( \text{Corrupt}(V, i) \) was issued for some \( V \in C \sqcup \mathcal{S} \) and \( \text{Send}(U, n, M) \) was issued for some \( M \in \mathcal{M} \).

Now, at any point during the protocol, \( A \) may make a \( \text{Test}(U, n) \) query. If \( \Omega^{(n)}_{U} \) is not fresh or this is not the first \( \text{Test} \) query of the game, \( A \) loses the game; otherwise, the challenger answers the query appropriately. The game continues, and eventually \( A \) makes a guess \( b' \) at the value of \( b \) that the challenger chose in response to the \( \text{Test} \) query; the adversary wins if \( b' = b \) and loses otherwise. We define the adversary’s advantage to be

\[
\text{Adv}^H_{\Gamma}(A) = 2\mathbb{P}[A \text{ wins the game } \Gamma \text{ with protocol } II] - 1.
\]

In our proof we will consider a sequence of games, and how the adversary’s advantage changes when we go from one game to the next. The following fact is helpful:

\[
\mathbb{P}[A \text{ wins } \Gamma \text{ with protocol } II] \leq \mathbb{P}[A \text{ wins } \Gamma' \text{ with protocol } II] + \epsilon \implies \text{Adv}^H_{\Gamma}(A) \leq \text{Adv}^H_{\Gamma'}(A) + 2\epsilon.
\]

3 Our Protocol

The protocol is based on the (ephemeral) SIDH scheme for key establishment by De Feo, Jao, and Plût [9]. In that original scheme, the public information sent by each user consists of the image curve and two auxiliary points. Our scheme uses the password to generate coefficients which scramble the auxiliary points.

Suppose a client \( A \in C \) wishes to establish a key with a server \( B \in \mathcal{S} \), who share a common password pwd\( _A \). The setup is as follows: Fix a prime \( p \) of the form \( t_A t_B \cdot f + 1 \) where \( t_A \) and \( t_B \) are small primes, \( e_A \) and \( e_B \) are positive integers, and \( f \) is some (typically very small) cofactor. Also, fix a supersingular curve \( E_K \) defined over \( \mathbb{F}_{p^2} \), and bases \( \{P_A, Q_A\} \) and \( \{P_B, Q_B\} \). Let \( H_A \) and \( H_B \) be random oracles with codom \( H_A = Y_2(t_B, e_B) \) and codom \( H_B = Y_2(t_A, e_A) \). They follow the following process:
1. A generates an ephemeral key and sends public information to B. In particular, A:
   (a) Chooses \( n_A \in \{0, 1, \ldots , t_A^e - 1 \} \) uniformly at random;
   (b) Constructs \( G_A = P_A + n_A Q_A \);
   (c) Defines \( E_A = E_K/\langle G_A \rangle \) and \( \phi_A \) to be the isogeny defined on \( E_K \) with kernel \( \langle G_A \rangle \);
   (d) Defines \( \Psi_A = H_A(j(E_A)||pwd_A) \)
   (e) Defines
   \[
   \begin{bmatrix}
   X_A \\
   Y_A
   \end{bmatrix} = \Psi_A \cdot \begin{bmatrix}
   \phi_A(P_B) \\
   \phi_A(Q_B)
   \end{bmatrix}
   \]
   (f) Sends \( (E_A, X_A, Y_A) \) to B.

2. Upon receiving \( (E, X, Y) \) from A, B:
   (a) Computes \( \text{ord}_E(X) \) and \( \text{ord}_E(Y) \)—if either is not \( \ell_B^e \), abort;
   (b) Checks that \( e_E(X, Y) = e_{E_k}(P_B, Q_B) \ell_A^e \)—if not, abort;
   (c) Computes \( \Psi_{AB} = H_B(j(E)||pwd_A) \)
   (d) Computes
   \[
   \begin{bmatrix}
   P_{AB} \\
   Q_{AB}
   \end{bmatrix} = \Psi_{AB}^{-1} \cdot \begin{bmatrix}
   X \\
   Y
   \end{bmatrix}
   \]
   (e) Chooses \( n_B \in \{0, 1, \ldots , t_B^e - 1 \} \) uniformly at random;
   (f) Constructs \( G_B = P_B + n_B Q_B \);
   (g) Defines \( E_B = E_K/\langle G_B \rangle \) and \( \phi_B \) to be the isogeny defined on \( E_K \) with kernel \( \langle G_B \rangle \);
   (h) Computes \( \Psi_B = H_B(j(E_B)||pwd_B) \)
   (i) Computes
   \[
   \begin{bmatrix}
   X_B \\
   Y_B
   \end{bmatrix} = \Psi_B \cdot \begin{bmatrix}
   \phi_B(P_A) \\
   \phi_B(Q_A)
   \end{bmatrix}
   \]
   (j) Sends \( (E_B, X_B, Y_B) \) to A; and,
   (k) Constructs the key
   \[
   K_B = \text{KDF}((E, X, Y)||(E_B, X_B, Y_B)||j(E/(P_{AB} + n_B Q_{AB}))||\Psi_{AB}||\Psi_B)
   \]

3. Upon receiving \( (E', X', Y') \) from B, A:
   (a) Computes \( \text{ord}_E(X') \) and \( \text{ord}_E(Y') \)—if either is not \( \ell_A^e \), abort;
   (b) Checks that \( e_{E'}(X', Y') = e_{E_k}(P_A, Q_A) \ell_B^e \)—if not, abort;
   (c) Computes \( \Psi_{BA} = H_B(j(E')||pwd_B) \)
   (d) Computes
   \[
   \begin{bmatrix}
   P_{BA} \\
   Q_{BA}
   \end{bmatrix} = \Psi_{BA}^{-1} \cdot \begin{bmatrix}
   X' \\
   Y'
   \end{bmatrix}
   \]
   and,
   (e) Constructs the key
   \[
   K_A = \text{KDF}((E_A, X_A, Y_A)||(E', X', Y')||j(E/(P_{BA} + n_B Q_{BA}))||\Psi_A||\Psi_{BA})
   \]

4 Security of the Scheme

We prove the following result:

**Theorem 1.** Suppose A plays \( \Gamma \) with protocol \( \Pi \), runs in time \( T \), and makes at most \( n_S, n_E, n_O, n_K \) queries to \( \text{Send}, \text{Execute} \), the random oracles, and \( \text{KDF} \), respectively. Then

\[
\text{Adv}_\Gamma^\Pi(A) \leq \frac{n_S}{L} + \frac{2(n_S + n_E)(n_S + n_E + n_O + n_K)}{\min\{\ell_A^e, \ell_B^e\}} + \frac{2n_K}{\min\{\ell_A^{e-1}(\ell_A - 1), \ell_B^{e-2}(\ell_B - 1)\}} + \frac{2n_S}{L} \left( \text{Adv}_p^{\text{SIDH}}(t, n_K) \right) + 2(n_S + n_E) \left( (\ell_B + 1) \text{Adv}_p^{\text{SIDH}}(t, n_K) + (\ell_A + 1) \text{Adv}_p^{\text{SIDH}}(t, n_K) \right) + 4(n_S + n_E) \left( \text{Adv}_p^{\text{SIDH}}(t) + \text{Adv}_p^{\text{SIDH}}(t) \right)
\]
Before we begin the proof, as in [16] (and later in [10]), for notational reasons it will be convenient to define some events which correspond intuitively to the Adversary making online password guesses:

- **ClientPasswordTest**(A, i, B, pwd): For some E, X, Y, E’, X’, Y’, Ψ, Ψ’, A makes queries:
  - \( \text{KDF}((E, X, Y)||j(E’)|||Y||Ψ||Ψ’) \)
  - \( \text{Send}(A, i, (\text{Initiate}, B)) \) with output (E, X, Y);
  - \( \text{Send}(A, i, (E’, X’, Y’)) \);
  - \( H_A(j(E)||pwd) = Ψ \) and \( H_B(j(E’)||pwd) = Ψ’ \)
  and \( E'' \triangleq \text{SIDH}_A(E, E’, Ψ’\cdot Ψ^{-1}[{X'}]) \). The value associated to this event is
    \[
    \text{sk}_A^{(i)} = \text{KDF}((E, X, Y)||j(E’)|||Y||Ψ||Ψ’)
    \]
  Intuitively, \( A \) is testing whether pwd\(_A\) = pwd by running a key establishment instance with client \( A \) (pretending to be server \( B \)).

- **ServerPasswordTest**(B, j, A, pwd): For some E, X, Y, E’, X’, Y’, Ψ, Ψ’, A makes queries:
  - \( \text{KDF}((E, X, Y)||j(E’)|||Y||Ψ||Ψ’) \)
  - \( \text{Send}(B, j, (E, X, Y)) \) with output (E’, X’, Y’);
  - \( H_A(j(E)||pwd) = Ψ \) and \( H_B(j(E’)||pwd) = Ψ’ \)
  and \( E'' \triangleq \text{SIDH}_B(E, E’, Ψ’\cdot Ψ^{-1}[{X'}]) \). The value associated to this event is
    \[
    \text{sk}_B^{(i)} = \text{KDF}((E, X, Y)||j(E’)|||Y||Ψ||Ψ’)
    \]
  Intuitively, \( A \) is testing whether pwd\(_A\) = pwd by running a key establishment instance with server \( B \) (pretending to be client \( A \)).

- **PassivePasswordTest**(A, i, B, j, pwd): For some E, X, Y, E’, X’, Y’, Ψ, Ψ’, A makes queries:
  - \( \text{KDF}((E, X, Y)||j(E’)|||Y||Ψ||Ψ’) \)
  - \( \text{Execute}(A, i, B, j) \) with transcript \( \text{msg}_A^{(i)} = (E, X, Y) \), \( \text{msg}_B^{(j)} = (E’, X’, Y’) \);
  - \( H_A(j(E)||pwd) = Ψ \) and \( H_B(j(E’)||pwd) = Ψ’ \)
  where \( E'' \triangleq \text{SIDH}(E, E’) \). The value associated to this event is
    \[
    \text{sk}_A^{(i)} = \text{sk}_B^{(j)} = \text{KDF}((E, X, Y)||j(E’)|||Y||Ψ||Ψ’)
    \]
  Intuitively, \( A \) is testing whether pwd\(_A\) = pwd by passively observing an instance of \( A \) establishing a key with \( B \) and guessing at the password.

- **ActiveCorrectPassword**: Either a **ClientPasswordTest**(A, i, B, pwd\(_A\)) event occurs for some \( A \in \mathcal{C}, B \in \mathcal{S} \) and \( i \in \mathbb{N} \), or a **ServerPasswordTest**(B, j, A, pwd\(_A\)) event occurs for some \( A \in \mathcal{C}, B \in \mathcal{S} \) and \( j \in \mathbb{N} \). As the name suggests, this is precisely the event that \( A \) has guessed a client’s password in a session in which they have actively participated.

- **PassiveCorrectPassword**: A **PassivePasswordTest**(A, i, B, pwd\(_A\)) event occurs for some \( A \in \mathcal{C}, B \in \mathcal{S} \), and \( i, j \in \mathbb{N} \). As the name suggests, this is precisely the event that \( A \) guesses a client’s password correctly in a session which they have only observed passively.

- **2ClientPassword**: This is the event that the adversary is detected checking two distinct passwords in an online dictionary attack against a client: for some \( \text{pwd} \neq \text{pwd}’ \), both **ClientPasswordTest**(A, i, B, pwd) and **ClientPasswordTest**(A, i, B, pwd’) occur.

- **2ServerPassword**: This is the event that the adversary is detected checking two distinct passwords in an online dictionary attack against a server: for some \( \text{pwd} \neq \text{pwd}’ \), both **ServerPasswordTest**(B, j, A, pwd) and **ServerPasswordTest**(B, j, A, pwd’) occur.

\[
\text{Proof}: \quad \text{We give a sequence of games } \{G_i\}_{i=0}^9, \text{ where } G_0 \text{ is the original security game, and } G_i \text{ admits nothing more than a simple online dictionary attack. In the following series of claims, we bound the advantage difference between consecutive games, and hence bound the advantage for the original security game.}
\]

\( G_0 \): The original security game.
$\Gamma_1$: The same as $\Gamma_0$, except that the party oracles are all simulated by the challenger rather than being directly accessible to the adversary. In particular:

1. During setup, the challenger chooses each party’s password uniformly at random from the dictionary $\mathcal{P}$, and stores it.
2. Upon receiving a $\text{Send}$ query, the challenger computes the appropriate party’s output and forwards it to the adversary.
3. Upon receiving an $\text{Execute}$ query, the challenger simulates the appropriate parties and forwards the transcript of the session to the adversary.
4. Upon receiving a $\text{Reveal}$ query, the challenger reveals the appropriate session key (if it exists).
5. Upon receiving a $\text{Corrupt}$ query, the challenger reveals the appropriate password.

Claim. For any adversary $\mathcal{A}$, $\text{Adv}^H_{\Gamma_0}(\mathcal{A}) = \text{Adv}^H_{\Gamma_1}(\mathcal{A})$.

Proof. It is clear that the two games are indistinguishable from the perspective of the adversary. □

$\Gamma_2$: The same as $\Gamma_1$, except that if the challenger chooses an ephemeral key $n_U$ for a party $U$ which has it has chosen previously for any party (in step 1a or 2d of a protocol execution, as appropriate) or for which the curve generated by this randomness ($E_K/(P_A + n_UQ_A)$ or $E_K/(P_B + n_UQ_B)$ as appropriate) has been used by the adversary as input to a $\text{Send}$, random oracle, or KDF query then the challenger ends the game and the adversary loses.

Claim. For any adversary $\mathcal{A}$ which makes $n_S, n_E, n_O, n_K$ queries to $\text{Send}, \text{Execute}$, the random oracles, and KDF, respectively, we have $\text{Adv}^H_{\Gamma_1}(\mathcal{A}) \leq \text{Adv}^H_{\Gamma_2}(\mathcal{A}) + \frac{2(n_S + n_E)(n_S + n_E + n_O + n_K)}{\min\{\ell_A^A, \ell_B^A\}}$.

Proof. The numbers of valid ephemeral keys for clients and for servers are $\ell_A^A$ and $\ell_B^A$, respectively. It follows then that if the challenger participating in game $\Gamma_2$ is choosing an ephemeral key after $k-1$ total ephemeral key choices, random oracle queries, and KDF queries, the probability that the randomness is distinct from the previous randomnesses is

$$\mathbb{P}[\Omega_{U}^{(i)}'s \text{ ephemeral key is unused}] \geq \begin{cases} \ell_A^A - (k-1) & \text{if } U \in \mathcal{C} \\ \ell_B^B - (k-1) & \text{if } U \in \mathcal{S} \end{cases}$$

for each $1 \leq k \leq (n_S + n_E)$. Then the probability that an ephemeral key pair has already been used satisfies

$$\mathbb{P}[\Omega_{U}^{(i)}'s \text{ ephemeral key has been used}] \leq \begin{cases} \frac{(n_S + n_E + n_O + n_K)}{(n_S + n_E + n_O + n_K)} & \text{if } U \in \mathcal{C} \\ \frac{(n_S + n_E + n_O + n_K)}{(n_S + n_E + n_O + n_K)} & \text{if } U \in \mathcal{S} \end{cases}$$

$$\leq \max \left\{ \frac{(n_S + n_E + n_O + n_K)}{\ell_A^A}, \frac{(n_S + n_E + n_O + n_K)}{\ell_B^B} \right\}$$

$$= \frac{(n_S + n_E + n_O + n_K)}{\min\{\ell_A^A, \ell_B^A\}}$$

for each $k$. Since the adversary makes at most $(n_S + n_E)$ $\text{Send}$ and $\text{Execute}$ queries, and each such query loses the game with probability bounded as above, it follows that

$$\text{Adv}^H_{\Gamma_1}(\mathcal{A}) - \text{Adv}^H_{\Gamma_2}(\mathcal{A}) \leq \frac{2(n_S + n_E)(n_S + n_E + n_O + n_K)}{\min\{\ell_A^A, \ell_B^A\}}$$

from which the result follows. □

$\Gamma_3$: The same as $\Gamma_2$, except that $\text{Send}$ and $\text{Execute}$ queries are executed without using random oracles, and future random oracle queries are answered in a way which is consistent with previous $\text{Send}$ and $\text{Execute}$ queries whenever possible. In particular,

1. In an $\text{Execute}(A, i, B, j)$ query for which $\Omega_A^{(i)}$ and $\Omega_B^{(j)}$ are unused, set
Proof. Consider the ways that a KDF query could occur, and what happens in $\Omega_B \in \mathcal{C}$ and $\Omega_A \in \mathcal{C}$, respectively, we have

For any adversary $A \in \mathcal{C}$, we consider the following:

1. When $A \in \mathcal{C}$ is instructed to initiate unused session $i$ with $B \in \mathcal{S}$, set
   \begin{enumerate}
   
   \item $n_A^{(i)} \leftarrow U \{0, 1, \ldots, \ell_A^{\text{A}} - 1\}$;
   \item $\phi_A^{(i)}$ to be the isogeny defined on $E_K$ with kernel $\langle P_A + n_A^{(i)} Q_A \rangle$;
   \item $E_A^{(i)} = E_K / \ker \phi_A^{(i)}$;
   \item $\Psi_A \leftarrow U(\mathcal{T}_2(\ell_A, e_A))$
   \item $\msg_A^{(i)} = (E_A^{(i)}, \Psi_A \phi_A(P A))$
   \item $\sk_A^{(i)} = \msg_A^{(i)} \leftarrow U(\{0, 1\}^\lambda)$
   \end{enumerate}

2. When $A \in \mathcal{C}$ is instructed to initiate unused session $i$ with $B \in \mathcal{S}$, set
   \begin{enumerate}
   
   \item $n_B^{(i)} \leftarrow U \{0, 1, \ldots, \ell_B^{\text{B}} - 1\}$;
   \item $\phi_B^{(i)}$ to be the isogeny defined on $E_K$ with kernel $\langle P_B + n_B^{(i)} Q_B \rangle$;
   \item $E_B^{(i)} = E_K / \ker \phi_B^{(i)}$;
   \item $\Psi_B \leftarrow U(\mathcal{T}_2(\ell_B, e_B))$
   \item $\msg_B^{(i)} = (E_B^{(i)}, \Psi_B \phi_B(P A))$
   \item $\sk_B^{(i)} = \msg_B^{(i)} \leftarrow U(\{0, 1\}^\lambda)$
   \end{enumerate}

3. In a Send($B$, $j$, msg) query where $B \in \mathcal{S}$ and $\Omega_B^{(j)}$ is unused, set
   \begin{enumerate}
   
   \item $n_B^{(j)} \leftarrow U \{0, 1, \ldots, \ell_B^{\text{B}} - 1\}$;
   \item $\phi_B^{(j)}$ to be the isogeny defined on $E_K$ with kernel $\langle P_B + n_B^{(j)} Q_B \rangle$;
   \item $E_B^{(j)} = E_K / \ker \phi_B^{(j)}$;
   \item $\Psi_B \leftarrow U(\mathcal{T}_2(\ell_B, e_B))$
   \item $\msg_B^{(j)} = (E_B^{(j)}, \Psi_B \phi_B(P A))$
   \item $\sk_B^{(j)} \leftarrow U(\{0, 1\}^\lambda)$
   \end{enumerate}

4. In a Send($A$, $i$, msg) query where $A \in \mathcal{C}$ and $\Omega_A^{(j)}$ has not yet computed a session key:
   \begin{enumerate}
   
   \item If $\Omega_A^{(i)}$ is partnered with $\Omega_B^{(j)}$ for some $B \in \mathcal{S}$, set $\sk_A^{(i)} = \sk_B^{(j)}$; otherwise,
   \item If this query causes ClientPasswordTest($A$, $i$, $S$, pwd$_A$), set $\sk_A^{(i)}$ to the corresponding value; otherwise,
   \item Set $\sk_A^{(i)} \leftarrow U(\{0, 1\}^\lambda)$.
   \end{enumerate}

5. In a KDF($\langle E, X, Y \rangle || \langle E', X', Y' \rangle || \langle E'' \rangle || \langle \Psi \rangle || \Psi \rangle$ query:
   \begin{enumerate}
   
   \item If this query causes an ActiveCorrectPassword or PassiveCorrectPassword event, output the associated value; otherwise,
   \item Output a uniformly random element of $\{0, 1\}^\lambda$.
   \end{enumerate}

Claim. For any adversary $A$ which runs in time $t$ and makes at most $n_S, n_E, n_K$ queries to Send, Execute, and KDF, respectively, we have

$$
\text{Adv}^H_{\mathcal{T}_2}(A) \leq \text{Adv}^H_{\mathcal{T}_2}(A) + \frac{2n_K}{\min\{\ell_A^{\text{A}} - 1, \ell_B^{\text{B}} - 1\}} + 2(n_S + n_E) \left( (\ell_B + 1)\text{Adv}^{\text{SI-APC}}_{\mathcal{A}}(t, n_K) + (\ell_A + 1)\text{Adv}^{\text{SI-APC}}_{\mathcal{B}}(t, n_K) \right)
$$

Proof. Consider the ways that a KDF query could occur, and what happens in $\Gamma_2$ when they do:

1. When $\Omega_B^{(j)}$ ($B \in \mathcal{S}$) receives a message and generates a key, it calls KDF on an input which has never been called before since, if it had been called before, we should have aborted either because an earlier server oracle had chosen the same randomness as $\Omega_B^{(j)}$ or the adversary had made a random oracle or KDF query with the corresponding randomness. So the output of this KDF query is independent from everything that has happened previously.

2. When $\Omega_A^{(j)}$ ($A \in \mathcal{C}$) receives a message and generates a key, there are three possibilities:
(a) $\Omega_B^{(j)}$ is partnered with some $\Omega_B^{(i)}$ for some $B \in \mathcal{K}$, in which case $sk_A^{(i)} = sk_B^{(j)}$; or,

(b) $\Omega_A^{(i)}$ is unpartnered and a ClientPasswordTest($A,i,U,pwd_A$) event occurs, in which case $sk_A^{(i)}$ is set to the value of the event (the corresponding KDF value); or,

(c) $\Omega_A^{(i)}$ is unpartnered and no ClientPasswordTest($A,i,U,pwd_A$) event occurs, in which case $sk_A^{(i)}$ is chosen independently of all previous queries, since the KDF query that yields it is new.

3. When $A$ makes a KDF((E, X, Y)||(E', X', Y')||j(E'')||ψ'') query, there are four (not mutually exclusive) possibilities:

(a) It causes a ClientPasswordTest, ServerPasswordTest or PassivePasswordTest event, in which case the output is the associated session key;

(b) $ψ$ is correct, but no $H_A(j(E)||pwd_A)$ query has been made;

(c) $ψ'$ is correct, but no $H_B(j(E')||pwd_A)$ query has been made, or;

(d) This KDF query is new, and so the output is chosen uniformly at random from $\{0,1\}^\lambda$.

It is clear that $\Gamma_3$ is indistinguishable from $\Gamma_2$ unless case 3.(b) or 3.(c) occurs. Consider the circumstances under which a KDF((E, X, Y)||(E', X', Y')||j(E'')||ψ'') query in one of the two cases could occur:

1. $(E, X, Y)$ has not been the output of a Send($A,i,(\text{Initiate},B)$) query, and $ψ$ is correct. This case occurs with probability $\ell_B^{2-3\epsilon}(\ell_B - 1)^{-1}$, since this matrix has not been computed before and so $A$ cannot do any better than guessing, and $|T(\ell_B, e_B)| = \ell_B^{3\epsilon-2}(\ell_B - 1)$.

2. $(E', X', Y')$ has not been the output of a Send($B,j,A,(E,X,Y)$) query, and $ψ'$ is correct. This case occurs with probability $\ell_A^{2-3\epsilon}(\ell_A - 1)^{-1}$, since this matrix has not been computed before and so $A$ cannot do any better than guessing, and $|T(\ell_A, e_A)| = \ell_A^{3\epsilon-2}(\ell_A - 1)$.

Notice that cases 1. and 2. occur at most $n_K$ times in total, and so the total advantaged gained by the adversary by considering both of these cases is bounded by

$$\min\{\ell_A^{3\epsilon-2}(\ell_A - 1), \ell_B^{3\epsilon-2}(\ell_B - 1)\}$$

3. $(E, X, Y)$ has been the output of a Send($A,i,(\text{Initiate},B)$) query, and $ψ$ is correct. We use an adversary who causes this case to solve the SI-APCA problem.

Given an instance $(E_K, P_A, Q_A, P_B, Q_B, E_A)$ of the SI-APCA problem and an adversary $A$ who causes this scenario with probability $\epsilon$, run $A$ with global curve and torsion bases $(E, P_A, Q_A, P_B, Q_B)$. Choose one of the fewer than $n_S + n_E$ client messages uniformly at random, and replace its message by $\tilde{\text{msg}} = (E_A, X, Y)$ for $(X, Y)$ chosen uniformly at random from

$$W = \left\{(S, T) \in E_A[\ell_A^e]^2 : e_{E_A}(S, T) = e_{E_K}(P_B, Q_B)^{\ell_A^e} \text{ and } \text{ord}_{E_A}(S) = \text{ord}_{E_A}(T) = \ell_B^e\right\}.$$ 

Observe that $\tilde{\text{msg}}$ is valid if and only if the (unique) representation of $X$ and $Y$ as

$$\begin{align*}
X &= \alpha \phi_A(P_B) + \beta \phi_A(Q_B) \\
Y &= \gamma \phi_A(P_B) + \delta \phi_A(Q_B)
\end{align*}$$

(where $\phi_A$ is the true isogeny $\phi_A : E_K \to E_A$) satisfies $\psi = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \in \mathcal{Y}_2(\ell_A, e_A)$.

By choice of $W$, $\psi \in \text{SL}_2(\ell_A, e_A)$, and so $\psi \in \mathcal{Y}_2(\ell_A, e_A)$ with probability

$$\mathbb{P}[\psi \in \mathcal{Y}_2(\ell_A, e_A)] = |\text{SL}_2(\ell_A, e_A) : \mathcal{Y}_2(\ell_A, e_A)|^{-1} = (\ell_A + 1)^{-1}.$$ 

When $\tilde{\text{msg}}$ is a correctly-formatted message, the simulation is indistinguishable from $\Gamma_3$ from the perspective of the adversary. In any case, with probability $\epsilon$, the scenario occurs in the way described, and with probability $(n_S + n_E)^{-1}$ it occurs for $\tilde{\text{msg}}$. Moreover, at most $n_K$ queries are made at all, and we have at most $n_K$ queries of the form KDF($\tilde{\text{msg}}||\cdot||||\psi||$). The probability that the correct $\psi$ is in the list $\{\psi_i\}_i$ is thus bounded by

$$\mathbb{P}[\psi \in \{\psi_i\}_i] \geq \epsilon \cdot (\ell_B + 1)^{-1} \cdot (n_S + n_E)^{-1}.$$ 

The set $\{\psi_i\}_i$ gives us a list of size at most $n_K$ of candidate solutions $\{\psi^{-1}[\chi]\}_i$ to our instance of SI-APCA. It follows that

$$\epsilon \leq (n_S + n_E)(\ell_B + 1)\mathbb{P}[\psi \in \{\psi_i\}_i] \leq (n_S + n_E)(\ell_B + 1)\text{Adv}_{\psi}^{\text{SI-APCA}}(t, n_K)$$

Notably, because of the definition of partnering, this partner $\Omega_B^{(j)}$ is unique.
4. \((E', X', Y')\) has been the output of a \(\text{Send}(B, j, \text{msg})\) query, and \(\Psi'\) is correct. By an analogous argument to point 3. above, we find that the probability that this occurs is bounded by

\[
(n_S + n_E)(\ell_A + 1)\text{Adv}^{\text{SI-APC}_B}_p(t, n_K).
\]

By the above arguments,

\[
\text{Adv}^{\Pi}_p(A) \leq \text{Adv}^{\Pi}_p(A) + \frac{2n_K}{\min\{\ell_A, \ell_B\}} \left(\ell_A^{-2}(\ell_A - 1), \ell_B^{-2}(\ell_B - 1)\right)
\]

+ \(2(n_S + n_E)\left(\ell_B + 1\right)\text{Adv}^{\text{SI-APC}_A}(t, n_K) + (\ell_A + 1)\text{Adv}^{\text{SI-APC}_B}(t, n_K)\)

as required.

\(\square\)

\(I_4\): \(\Gamma_4\) is indistinguishable from \(\Gamma_3\), except that there is no checking for \text{PassiveCorrectPassword} events.

\textbf{Claim.} For any adversary \(\mathcal{A}\) that runs in time \(t\) and makes at most \(n_K\) queries to the KDF oracle we have

\[
\text{Adv}^{\Pi}_p(\mathcal{A}) \leq \text{Adv}^{\Pi}_p(\mathcal{A}) + 2n_E \text{Adv}^{\text{C-SIDH}}_p(t, n_K).
\]

\textbf{Proof.} It is clear that \(\Gamma_3\) is indistinguishable from \(\Gamma_4\) unless \text{PassiveCorrectPassword} occurs. Suppose it occurs with probability \(\epsilon\). We build an SDH solver from \(\mathcal{A}\) as follows:

Suppose we are given an instance \(((E, K, P_A, Q_A, P_B, Q_B), (E_A, X_A, Y_A), (E_B, X_B, Y_B))\) of SIDH. Run \(\mathcal{A}\) as usual, with global curve and torsion bases \((E_K, P_A, Q_A, P_B, Q_B)\), and at the beginning of the game, choose an index \(1 \leq k \leq n_E\) of a potential \text{Execute} query. When \(\mathcal{A}\) makes its \(k\)th \text{Execute} query, instead of choosing the messages as usual, instead choose \(\Psi \leftarrow U(T(\ell_B, e_B))\) and \(\Psi' \leftarrow U(T(\ell_A, e_A))\) and set the transcript as

\[
\text{msg}_A = (E_A, \Psi \left[ X_A \right]), \quad \text{msg}_B = (E_B, \Psi' \left[ X_B \right]).
\]

With probability \(\epsilon\), \(\mathcal{A}\) will cause \text{PassiveCorrectPassword}, and with probability at least \(n_E^{-1}\) it will occur for the \text{Execute} command we chose. Now, \(\mathcal{A}\) makes at most \(n_K\) queries of the form \(\text{KDF}(\text{msg}_A || \text{msg}_B || j(\hat{E}_j)) || |\Psi|| |\Psi'|\) and so produces a list of size at most \(n_K\) of candidate solutions \(\hat{E}_j\) to the SIDH instance. It follows that

\[
P(\text{SIDH}(E_A, E_B) \in \hat{E}_j) \geq n_E^{-1},
\]

which implies \(\epsilon \leq n_E P(\text{SIDH}(E_A, E_B) \in \hat{E}_j) \leq n_E \text{Adv}^{\text{SIDH}}_p(t, n_K)\) from which the result follows.

\(\square\)

\(I_5\): \(\Gamma_5\) is indistinguishable from \(\Gamma_4\), except that if \text{ActiveCorrectPassword} occurs, the game ends and the adversary wins.

\textbf{Claim.} For any adversary \(\mathcal{A}\), \(\text{Adv}^{\Pi}_p(\mathcal{A}) \leq \text{Adv}^{\Pi}_p(\mathcal{A})\).

\textbf{Proof.} This is clear; we have simply added more win conditions for the adversary.

\(\square\)

\(I_6\): \(\Gamma_6\) is indistinguishable from \(\Gamma_5\), except that if \text{2ServerPassword} occurs, the game ends and the adversary loses.

\textbf{Claim.} For any adversary \(\mathcal{A}\) which runs in time \(t\) and makes at most \(n_S\) \text{Send} queries and \(n_K\) KDF queries, we have

\[
\text{Adv}^{\Pi}_p(\mathcal{A}) \leq \text{Adv}^{\Pi}_p(\mathcal{A}) + 2n_S \text{Adv}^{\text{C-SGA}_B}_p(t, n_K(n_K - 1))
\]

\textbf{Proof.} Given an instance \((\Psi_1, \Psi_2, (E_B, \phi_B(P_A), \phi_B(Q_A)))\) of C-SGA, choose an index \(1 \leq k_S \leq n_S\) of a potential \text{Send} query that \(\mathcal{A}\) can make. Run the protocol as usual until the \(k\)th \text{Send} query occurs. If the query is submitted to a client, abort and try again. Otherwise, the query is of the form \(\text{Send}(B, j, A, (E, X, Y))\) for \(A \in \mathcal{C}\), \(B \in \mathcal{S}\), and \(\Omega_B^{(j)}\) unused. Set the response message to this \text{Send} query as \(\text{msg}_B = (E_B, \Psi \left[ \phi_B(P_A) \right])\).

\(\mathcal{A}\) makes fewer than \(n_K\) queries of the form \(\text{KDF}((E, X, Y)) || \text{msg}_B || j(E_i) || |\Psi_i|| |\Psi_i'\); hence (letting \(\text{Min}(\Psi)\) denote the matrix of minors of \(\Psi\)) the list of all ordered pairs

\[
\{( (E''_i, \text{Min}(\Psi_i)^{-1} [X][Y]), (E''_j, \text{Min}(\Psi_j)^{-1} [X][Y])) \}_{i \neq j}
\]

has size at most \(n_K(n_K - 1)\). If \text{2ServerPassword} occurs, then with probability \(n_S^{-1}\) it occurs for the \text{Send} query we chose, and in this case it must hold that for some \(i \neq j\),

\[
E''_i = \text{SIDH}_B(E, E_B, \Psi_i^{-1} [X][Y]) \quad \text{and} \quad E''_j = \text{SIDH}_B(E, E_B, \Psi_j^{-1} [X][Y]).
\]
Note that
\[
E''_i = E/\langle [1, n_B]|\psi''^{-1}_i [\chi] \rangle
= E/\langle [1, n_B]|\text{Min}(\psi_1)^T (\text{Min}(\psi_1)^{-T}\psi''^{-1}_i [\chi]) \rangle
= E/\langle [1, n_B]| (\text{Min}(\psi_1)^{-T}\psi''^{-1}_i [\chi]) \rangle
\]
and
\[
E''_j = E/\langle [1, n_B]| (\text{Min}(\psi_2)^{-T}\psi''^{-1}_j [\chi]) \rangle
\]
But then \((E''_i, \text{Min}(\psi_1)^{-T}\psi''^{-1}_i [\chi]), (E''_j, \text{Min}(\psi_2)^{-T}\psi''^{-1}_j [\chi])\) is precisely a solution to our instance of C-SGA\(_B\). It follows that if 2ServerPassword occurs with probability \(\epsilon\), we have
\[
n_S^{-1}\epsilon \leq \text{Adv}_{p}^{c-SGA_B}(t, n_K(n_K-1)) \implies \epsilon \leq 2\text{Adv}_{p}^{c-SGA_B}(t, n_K(n_K-1)).
\]
Since games \(\Gamma_5\) and \(\Gamma_6\) are indistinguishable unless 2ServerPassword occurs, we find that
\[
\text{Adv}_{\Gamma_5}^{H}(A) \leq \text{Adv}_{\Gamma_6}^{H}(A) + 2n_S\text{Adv}_{p}^{c-SGA_B}(t, n_K(n_K-1))
\]
as required. \(\square\)

\(\Gamma_7\): The same as \(\Gamma_6\), except that if 2ClientPassword occurs, the game ends and the adversary loses.

Claim. For any adversary \(A\) which runs in time \(t\) and makes at most \(n_S\) Send queries and \(n_K\) KDF queries,
\[
\text{Adv}_{\Gamma_6}^{H}(A) \leq \text{Adv}_{\Gamma_5}^{H}(A) + 2n_S\text{Adv}_{p}^{c-SGA_A}(t, n_K(n_K-1))
\]
Proof. This follows by an argument analogous to the previous claim. \(\square\)

\(\Gamma_8\): The same as \(\Gamma_7\), except that whenever a client would send a message \((E_A, X_A, Y_A)\), we instead send \((E_A, \hat{X}_A, \hat{Y}_A)\), for \((\hat{X}_A, \hat{Y}_A)\) chosen uniformly at random from
\[
W = \left\{ (S, T) \in E_A[\ell_B] : e_{E_A}(S, T) = e_{E_K}(P_B, Q_B) e_{A}^{A} \text{ and } \text{ord}_{E_A}(S) = \text{ord}_{E_A}(T) = \ell_B \right\}.
\]
Claim. For any adversary \(A\) running in time \(t\),
\[
\text{Adv}_{\Gamma_5}^{H}(A) \leq \text{Adv}_{\Gamma_4}^{H}(A) + 2(n_S + n_E)\text{Adv}_{p}^{SI-APDA}(t)
\]
Proof. We proceed by a hybrid argument; let \(\{\Gamma_{7,j}^{(j)}\}_{j=0}^{n_S+n_E}\) be a sequence obtained from \(\Gamma_7\) by replacing only the first \(j\) client messages as described. We will show that for \(1 \leq j \leq n_S + n_E\),
\[
\text{Adv}_{\Gamma_{7,j-1}^{(j)}}^{H}(A) \leq \text{Adv}_{\Gamma_{7,j}^{(j)}}^{H}(A) + 2\text{Adv}_{p}^{SI-APDA}(t).
\]
Suppose we are faced with an instance \((E_A, X_A, Y_A)\) of SI-APDA. Run the simulation against \(A\) as usual, but replace the first \(j-1\) client messages as described, and replace the \(j\)th client message with \(\text{msg}_A = (E_A, \Psi \left[ X_A \right]) \) for \(\Psi \leftarrow U(\gamma, \ell_B, e_B)\). Play the game to completion with \(A\), or terminate it if it exceeds its time or query bounds or fails. Upon completion, guess that \((X_A, Y_A)\) are the correct auxiliary points for \(E_A\) if \(A\) wins the security game, and guess that they are random points otherwise. Observe that
\[
\mathbb{P}[\text{We guess correctly}] = \mathbb{P}[(X_A, Y_A) \text{ are correct} | A \text{ wins}] + \mathbb{P}[(X_A, Y_A) \text{ are random} | A \text{ does not win}]
\geq \mathbb{P}[(X_A, Y_A) \text{ are correct}] \cdot \mathbb{P}[A \text{ wins} | (X_A, Y_A) \text{ are correct}]
+ \mathbb{P}[(X_A, Y_A) \text{ are random}] \cdot \mathbb{P}[A \text{ does not win} | (X_A, Y_A) \text{ are random}]
= \frac{1}{2} \left( \frac{1}{2} + \frac{1}{2} \text{Adv}_{\Gamma_{7,j-1}^{(j)}}^{H}(A) \right) + \frac{1}{2} \left( \frac{1}{2} - \frac{1}{2} \text{Adv}_{\Gamma_{7,j}^{(j)}}^{H}(A) \right).
\]
This procedure takes time at most $t$, and so by definition $\mathbb{P}[\text{We guess correctly}] = \frac{1}{2} + \text{Adv}_{\text{P}}^{\text{SI-APD}A}(t)$. Substituting this into the above and simplifying we find that for each $j$,
\[
\frac{1}{2} + \text{Adv}_{\text{P}}^{\text{SI-APD}A}(t) \geq \frac{1}{2} - \frac{1}{4} \text{Adv}_{\text{R}^{(j)}}^{\text{H}}(A) - \frac{1}{4} \text{Adv}_{\text{R}^{(j)}}^{\text{H}}(A) \implies \text{Adv}_{\text{R}^{(j)}}^{\text{H}}(A) \leq \text{Adv}_{\text{R}^{(j)}}^{\text{H}}(A) + 4 \text{Adv}_{\text{P}}^{\text{SI-APD}A}(t).
\]
There are $n_S + n_E + 1$ hybrids, and so, substituting $I_{7}^{(0)} = I_{7}$ and $I_{7}^{(n_S + n_E)} = I_{8}$ we obtain
\[
\text{Adv}_{\text{R}^{(j)}}^{\text{H}}(A) \leq \text{Adv}_{\text{R}^{(j)}}^{\text{H}}(A) + 4(n_S + n_E) \text{Adv}_{\text{P}}^{\text{SI-APD}A}(t)
\]
as required.

$I_9$: The same as $I_8$, except that whenever a server would send a message $(E_B, X_B, Y_B)$, we instead send $(E_B, X_B, Y_B)$, for $(X_B, Y_B)$ chosen uniformly at random from
\[
W = \left\{ (S, T) \in E_B[\mathcal{E}_A]^2 : e_{E_B}(S, T) = e_{E_K}(P_A, Q_A)^{e_B} \text{ and } \text{ord}_{E_B}(S) = \text{ord}_{E_B}(T) = e_A \right\}.
\]

**Claim.** For any adversary $A$ running in time $t$, $\text{Adv}_{\text{R}^{(j)}}^{\text{H}}(A) \leq \text{Adv}_{\text{R}^{(j)}}^{\text{H}}(A) + 2(n_S + n_E) \text{Adv}_{\text{P}}^{\text{SI-APD}A}(t)$

**Proof.** This follows by an argument analogous to the previous claim. $\Box$

**Claim.** For any adversary $A$ that makes at most $n_S$ Send queries, $\text{Adv}_{\text{R}^{(j)}}^{\text{H}}(A) \leq \frac{n_S}{L}$ where $L$ is the size of the password space.

**Proof.** Note that
\[
\mathbb{P}[A \text{ succeeds}] \leq \mathbb{P}[\text{ActiveCorrectPassword}] + \mathbb{P}[A \text{ succeeds}|\text{No ActiveCorrectPassword}] \cdot \mathbb{P}[\text{No ActiveCorrectPassword}]
\]

In game $I_9$, the messages contain no information about the passwords, since the auxiliary points are random. It follows that $\mathbb{P}[\text{ActiveCorrectPassword}] \leq \frac{n_S}{L}$, since the adversary causes at most $n_S$ total ClientPasswordTest and ServerPasswordTest events.

It remains to compute $\mathbb{P}[A \text{ succeeds}|\text{No ActiveCorrectPassword}]$. The only way for $A$ to win if it does not cause an ActiveCorrectPassword event is to make a Test query to a fresh instance $\Omega^{(i)}_U$ and successfully guess whether the returned key is true or random. We show that the $A$’s view is independent of $\text{sk}_U^{(i)}$, and hence that $A$ wins in this way with probability exactly $\frac{1}{2}$.

There are two kinds of queries we need to consider:

1. **Reveal** queries. Since $\Omega^{(i)}_U$ is fresh, $A$ has made neither a Reveal($U,i$) query nor a Reveal($V,j$) query, where $\Omega^{(j)}_V$ is the partner to $\Omega^{(i)}_U$ (if it exists). For any other instance $\Omega^{(k)}_W$, the session key $\text{sk}_W^{(k)}$ is computed as
\[
\text{sk}_W^{(k)} = \text{KDF} \left( \text{msg}_W^{(k)} || \text{msg}_W^{(k')} || j(E_W^{(k')} || \psi || \psi') \right)
\]
where $\text{msg}_W$, $\text{msg}_W'$ is the transcript of the protocol. By $I_2$, we know that the message transcript from the perspective of $\Omega^{(i)}_U$ is different from that of $\Omega^{(k)}_W$, since otherwise we would have aborted. Since KDF is a random oracle, $\text{sk}_W^{(k)}$ is independent of $\text{sk}_U^{(i)}$, and hence a Reveal($W,k$) query gives no advantage to $A$ for the given Test query.

2. **KDF** queries. Note that in games $I_3$ and beyond, the output of KDF queries is independent of any previously computed session keys, since there are no checks for PassiveCorrectPassword, and ActiveCorrectPassword events cause the simulation to end. So we need only consider KDF queries that occur before $\text{sk}_U^{(i)}$ is set. There are three cases to consider:

(a) $U \in \mathcal{C}$, and $\Omega^{(i)}_U$ is unpartnered. In this case, cases 4.(a) and 4.(b) in the description of $I_3$ do not occur (because $U$ is unpartnered, and ActiveCorrectPassword events end the simulation) and so we must be in case 4.(c); but here $\text{sk}_U^{(i)}$ is chosen without regard for anything that has previously occurred.
(b) \( U \in \mathcal{E} \), and \( \Omega^{(i)}_U \) is partnered with some \( \Omega^{(j)}_V \). In this case, \( \text{sk}^{(i)}_U = \text{sk}^{(j)}_V \). But according to point 3.(f) in the description of \( \Gamma_3 \), \( \text{sk}^{(j)}_V \) is chosen without regard for anything that has previously occurred.

(c) \( U \in \mathcal{F} \). Again we are in point 3.(f) in the description of \( \Gamma_3 \), and so \( \text{sk}^{(i)}_U \) is chosen without regard for anything that has previously occurred.

Altogether, this says that \( A \)'s view in independent from \( \text{sk}^{(i)}_U \) when \text{ActiveCorrectPassword} does not occur, and so \( \mathbb{P}[A \text{ succeeds}|\text{No ActiveCorrectPassword}] = \frac{1}{2} \).

Now,

\[
\mathbb{P}[A \text{ succeeds}] = \mathbb{P}[\text{ActiveCorrectPassword}] + \frac{1}{2}\mathbb{P}[\text{No ActiveCorrectPassword}]
\]

\[
= \mathbb{P}[\text{ActiveCorrectPassword}] + \frac{1}{2}(1 - \mathbb{P}[\text{ActiveCorrectPassword}])
\]

\[
= \frac{1}{2} + \frac{1}{2}\mathbb{P}[\text{ActiveCorrectPassword}]
\]

\[
\leq \frac{1}{2} + \frac{1}{2}\mathbb{P}[\text{ActiveCorrectPassword}] 
\]

from which it immediately follows that \( \text{Adv}^{H}_H({\mathcal{A}}) \leq \frac{n_S}{2} \). Using the preceding 8 claims, we finally obtain the desired result. \( \square \)

**Corollary 1.** Under the SIDH, SI-APCA, SI-APCB, SI-APDA, SI-APDB, C-SGA\(_A\) and C-SGA\(_B\) assumptions, if \( H_A, H_B, \) and KDF are modelled as random oracles, our protocol is secure in the model of [1] against polynomial-time adversaries.

### 5 Conclusions and Future Work

We have given a construction for a PAKE in the random oracle model whose security is based on the hardness of computational problems related to isogenies of supersingular elliptic curves—in particular, problems which are conjectured to be resistant to quantum algorithms. This is the first proposed PAKE from quantum-safe primitives which is not lattice-based.

Going forward there are two main questions related to this work that remain open:

1. Can we construct an isogeny-based PAKE in the quantum random oracle model (where quantum access to all hash functions are allowed) or with no random oracles?
2. How hard is the C-SGA problem? Can we reduce a better-understood problem to it?

### References


