An Isogeny-Based Password-Authenticated Key Establishment Protocol

Oleg Taraskin[1], Vladimir Soukharev[2], David Jao[3,4], and Jason LeGrow[3,5]

[1] Bankex Foundation, Moscow, Russian Federation
tog.postquant@gmail.com, ot@bankexfoundation.org
[2] InfoSec Global, Toronto, Ontario, M2J 5C2, Canada
vladimir.soukharev@infosecglobal.com
[3] Department of Combinatorics and Optimization
University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada
{djao,jlegrow}@uwaterloo.ca
[4] evolutionQ Inc.
Waterloo Accelerator Centre
Waterloo, Ontario, N2L 6R5, Canada
david.jao@evolutionQ.com
[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 [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 [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 [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 [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* ϕ 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 ∞ denotes the identity element on an elliptic curve). Equivalently, an isogeny is an algebraic morphism which is a group homomorphism. The degree of ϕ, 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 $\phi\colon E_1 \to E_2$ of degree n, there exists another isogeny $\hat{\phi}\colon E_2 \to E_1$ 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 ϕ.

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* $\mathrm{End}(E)$ 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 $\dim_{\mathbb{Z}}(\mathrm{End}(E)) = 2$, then we say that E is *ordinary*; otherwise $\dim_{\mathbb{Z}}(\mathrm{End}(E)) = 4$ 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 $\phi\colon E_1 \to E_2$ is defined to be *separable* if the extension $\mathbb{F}_q(E_1)/\phi^*(\mathbb{F}_q(E_2))$ 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.

$\mathcal{A}$		$\mathcal{B}$
Input: A, B		**Input:** B
$m_A, n_A \in_R \mathbb{Z}/\ell_A^{e_A}\mathbb{Z}$		$m_B, n_B \in_R \mathbb{Z}/\ell_B^{e_B}\mathbb{Z}$
$\phi_A := E \to E/\langle m_A P_A + n_A Q_A \rangle$		$\phi_B := E \to E/\langle m_B P_B + n_B Q_B \rangle$
	$\xrightarrow{A, \phi_A(P_B), \phi_A(Q_B), E_A}$	
	$\xleftarrow{B, \phi_B(P_A), \phi_B(Q_A), E_B}$	
$E_{AB} := E_B/\langle m_A \phi_B(P_A) + n_A \phi_B(Q_A) \rangle$		$E_{BA} := E_A/\langle m_B \phi_A(P_B) + n_B \phi_A(Q_B) \rangle$
Output: $j(E_{AB})$		**Output:** $j(E_{BA})$

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

Fix a prime p of the form $\ell_A^{e_A}\ell_B^{e_B} \cdot f \pm 1$ where ℓ_A and ℓ_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 $E[\ell_A^{e_A}]$ and $E[\ell_B^{e_B}]$ respectively, so that $\langle P_A, Q_A \rangle = E[\ell_A^{e_A}]$ and $\langle P_B, Q_B \rangle = E[\ell_B^{e_B}]$. Alice chooses two random elements $m_A, n_A \in_R \mathbb{Z}/\ell_A^{e_A}\mathbb{Z}$, not both divisible by ℓ_A, and computes an isogeny $\phi_A \colon E \to E_A$ 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 ϕ_A to the basis $\{P_B, Q_B\}$ for $E[\ell_B^{e_B}]$, and sends these points to Bob together with E_A. Similarly, Bob selects random elements $m_B, n_B \in_R \mathbb{Z}/\ell_B^{e_B}\mathbb{Z}$ and computes an isogeny $\phi_B \colon E \to E_B$ 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 $\phi'_A \colon E_B \to E_{AB}$ having kernel equal to $\langle m_A \phi_B(P_A) + n_A \phi_B(Q_A) \rangle$; Bob proceeds *mutatis mutandis*. Alice and Bob can then use the common j-invariant of

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

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 ϕ_A on any point in $E[\ell_B^{e_B}]$. 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 ϕ_A on $E[\ell_A^{e_A}]$), since disclosing this information would allow the adversary to solve a system of discrete logarithms in $E[\ell_A^{e_A}]$ (which are easy since $E[\ell_A^{e_A}]$ 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 $\lambda = \lceil \log_2 p \rceil$. 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 $\phi_A \colon E \to E_A$ be an isogeny with kernel $\langle P_A + n_A Q_A \rangle$ where $n_A \leftarrow U(\mathbb{Z}/\ell_A^{e_A}\mathbb{Z})$. Similarly, let $\phi_B \colon E \to E_B$ be an isogeny with kernel $\langle P_B + n_B Q_B \rangle$ where $n_B \leftarrow U(\mathbb{Z}/\ell_B^{e_B}\mathbb{Z})$. 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

$$\mathrm{SIDH}\big((E, P_A, Q_A, P_B, Q_B), (E_A, \phi_A(P_B), \phi_A(Q_B)), (E_B, \phi_B(P_A), \phi_B(Q_A))\big) = E_{AB}.$$

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

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

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

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

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

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

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

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 λ, $\mathrm{Adv}_p^{\text{C-SIDH}}(t, n) = \mathrm{negl}(\lambda)$.

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 $\ell_A^{e_A}$- and $\ell_B^{e_B}$-isogenies.

Definition 2 (Supersingular Isogeny Auxillary Point Computation–A (SI-APC$_A$)). *Let $\phi_A \colon E \to E_A$ be an isogeny with kernel $\langle P_A + n_A Q_A\rangle$ where $n_A \leftarrow U(\mathbb{Z}/\ell_A^{e_A}\mathbb{Z})$. 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 Auxillary Point Computation–B (SI-APC$_B$)). *Let $\phi_B \colon E \to E_B$ be an isogeny with kernel $\langle P_B + n_B Q_B\rangle$ where $n_B \leftarrow U(\mathbb{Z}/\ell_B^{e_B}\mathbb{Z})$. 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

$$\begin{aligned}\text{SI-APC}_A((E, P_A, Q_A, P_B, Q_B, E_A)) &= (\phi_A(P_B), \phi_A(Q_B)), \text{ and}\\ \text{SI-APC}_B((E, P_A, Q_A, P_B, Q_B, E_B)) &= (\phi_B(P_A), \phi_B(Q_A)).\end{aligned}$$

We also define the advantages

$$\begin{cases}\mathrm{Adv}_p^{\text{SI-APC}_A}(\mathcal{A}) = \mathbb{P}[\vartheta \leftarrow \mathtt{Ins}_{\text{SI-APC}_A}(p) : \text{SI-APC}_A(\vartheta) \in \mathcal{A}(\vartheta)]\\ \mathrm{Adv}_p^{\text{SI-APC}_B}(\mathcal{A}) = \mathbb{P}[\vartheta \leftarrow \mathtt{Ins}_{\text{SI-APC}_B}(p) : \text{SI-APC}_B(\vartheta) \in \mathcal{A}(\vartheta)]\end{cases}$$

and

$$\begin{cases}\mathrm{Adv}_p^{\text{SI-APC}_A}(t, n) = \max\{\mathrm{Adv}_p^{\text{SI-APC}_A}(\mathcal{A}) : \mathcal{A} \text{ runs in time } t \text{ and } |\mathcal{A}(\vartheta)| \leq n \text{ for all } \vartheta \in \mathtt{Ins}_{\text{SI-APC}_A}(p)\}\\ \mathrm{Adv}_p^{\text{SI-APC}_B}(t, n) = \max\{\mathrm{Adv}_p^{\text{SI-APC}_B}(\mathcal{A}) : \mathcal{A} \text{ runs in time } t \text{ and } |\mathcal{A}(\vartheta)| \leq n \text{ for all } \vartheta \in \mathtt{Ins}_{\text{SI-APC}_B}(p)\}.\end{cases}$$

The SI-APC$_A$ and SI-APC$_B$ assumptions are that for t and n polynomial in λ, $\mathrm{Adv}_p^{\text{SI-APC}_A}(t, n) = \mathrm{negl}(\lambda)$ and $\mathrm{Adv}_p^{\text{SI-APC}_A}(t, n) = \mathrm{negl}(\lambda)$, 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 $\phi_A \colon E \to E_A$ be an isogeny with kernel $\langle P_A + n_A Q_A \rangle$ for $n_A \leftarrow U(\mathbb{Z}/\ell_A^{e_A}\mathbb{Z})$. 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 $\phi_B \colon E \to E_B$ be an isogeny with kernel $\langle P_B + n_B Q_B \rangle$ for $n_B \leftarrow U(\mathbb{Z}/\ell_B^{e_B}\mathbb{Z})$. 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 ϕ 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 $\phi|_{E[\ell_A^{e_A}]}$ or $\phi|_{E[\ell_B^{e_B}]}$.)

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 Auxillary Point Decision–A (SI-APD_A)). *Let $\phi_A \colon E \to E_A$ be an isogeny with kernel $\langle P_A + n_A Q_A \rangle$ where $n_A \leftarrow U(\mathbb{Z}/\ell_A^{e_A}\mathbb{Z})$. 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 $e_{E_A}(X,Y) = e_E(P_B, Q_B)^{\ell_A^{e_A}}$ where either*

1. *$X = \phi_A(P_B)$ and $Y = \phi_A(Q_B)$, or*
2. *$(X,Y) \leftarrow U(\{(S,T) \in E_A[\ell_B^{e_B}]^2 \,:\, e_{E_A}(X,Y) = e_E(P_B,Q_B)^{\ell_A^{e_A}}$ and $\mathrm{ord}_{E_A}(X) = \mathrm{ord}_{E_A}(Y) = \ell_B^{e_B}\})$*

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

Definition 7 (Supersingular Isogeny Auxillary Point Decision–B (SI-APD_B)). *Let $\phi_B \colon E \to E_B$ be an isogeny with kernel $\langle P_B + n_B B_B \rangle$ where $n_B \leftarrow U(\mathbb{Z}/\ell_B^{e_B}\mathbb{Z})$. 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 $e_{E_B}(X,Y) = e_E(P_A, B_A)^{\ell_B^{e_B}}$ where either*

1. *$X = \phi_B(P_A)$ and $Y = \phi_B(B_A)$, or*
2. *$(X,Y) \leftarrow U(\{(S,T) \in E_B[\ell_A^{e_A}]^2 \,:\, e_{E_B}(X,Y) = e_E(P_A,B_A)^{\ell_B^{e_B}}$ and $\mathrm{ord}_{E_B}(X) = \mathrm{ord}_{E_B}(Y) = \ell_A^{e_A}\})$*

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

As before, we define the advantages

$$\begin{cases} \mathrm{Adv}_p^{\text{SI-APD}_A}(\mathcal{A}) = \mathbb{P}[\vartheta \leftarrow \mathtt{Ins}_{\text{SI-APC}_A}(p) : \mathcal{A}(\vartheta) \text{ is correct}] \\ \mathrm{Adv}_p^{\text{SI-APD}_B}(\mathcal{A}) = \mathbb{P}[\vartheta \leftarrow \mathtt{Ins}_{\text{SI-APC}_B}(p) : \mathcal{A}(\vartheta) \text{ is correct}] \end{cases}$$

and

$$\begin{cases} \mathrm{Adv}_p^{\text{SI-APD}_A}(t,n) = 2\max\{\mathrm{Adv}_p^{\text{SI-APC}_A}(\mathcal{A}) : \mathcal{A} \text{ runs in time } t \text{ for all } \vartheta \in \mathtt{Ins}_{\text{SI-APD}_A}(p)\} - 1 \\ \mathrm{Adv}_p^{\text{SI-APD}_B}(t,n) = 2\max\{\mathrm{Adv}_p^{\text{SI-APC}_B}(\mathcal{A}) : \mathcal{A} \text{ runs in time } t \text{ for all } \vartheta \in \mathtt{Ins}_{\text{SI-APD}_B}(p)\} - 1. \end{cases}$$

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

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 ℓ and an integer e, we define

$$\begin{aligned}\mathrm{SL}_2(\ell, e) &= \left\{\Psi \in (\mathbb{Z}/\ell^e\mathbb{Z})^{2\times 2} \,:\, \det A \equiv 1 \pmod{\ell^e}\right\} \\ \Upsilon_2(\ell, e) &= \{\Psi \in \mathrm{SL}_2(\ell, e) \,:\, A \text{ is upper triangular modulo } \ell\}\end{aligned}$$

to be the special linear and special reduced upper triangular groups modulo ℓ^e, respectively. If $p = \ell_A^{e_A}\ell_B^{e_B}f \pm 1$ is a prime and E is a supersingular elliptic curve defined over $\mathbb{F}_{p^2}$, $\Upsilon_2(\ell_A, e_A)$ acts on $E[\ell_A^{e_A}]^2$ in a way completely analogous with ordinary matrix-vector multiplication over a ring: if $\Psi = \left[\begin{smallmatrix}\alpha & \beta \\ \gamma & \delta\end{smallmatrix}\right]$ 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 $E[\ell_A^{e_A}]$, then so is $\{X', Y'\}$, defined by $\left[\begin{smallmatrix}X' \\ Y'\end{smallmatrix}\right] = \Psi^{-1}\Psi' \left[\begin{smallmatrix}X \\ Y\end{smallmatrix}\right]$ for any $\Psi, \Psi' \in \Upsilon_2(\ell_A, e_A)$.
2. For any $X, Y \in E[\ell_A^{e_A}]$ and for any $\Psi, \Psi' \in \Upsilon_2(\ell_A, e_A)$, for $\left[\begin{smallmatrix}X' \\ Y'\end{smallmatrix}\right] = \Psi^{-1}\Psi' \left[\begin{smallmatrix}X \\ Y\end{smallmatrix}\right]$ we have $e(X, Y) = e(X', Y')$, where e is the Weil pairing on E. Moreover, $\mathrm{SL}_2(\ell_A, e_A)$ is the unique maximal matrix group acting as defined above which has this property.

In the context of isogeny computations, this (left) action of $\Upsilon_2(\ell_A, e_A)$ induces a new (right) action on $\mathbb{Z}/\ell_A^{e_A}\mathbb{Z}$ in the following way: observe that if $\left[\begin{smallmatrix}X' \\ Y'\end{smallmatrix}\right] = \Psi \left[\begin{smallmatrix}X \\ Y\end{smallmatrix}\right]$ where $\Psi = \left[\begin{smallmatrix}\alpha & \beta \\ \gamma & \delta\end{smallmatrix}\right]$, then for any $m \in \mathbb{Z}/\ell_A^{e_A}\mathbb{Z}$

$$\begin{aligned} E/\left\langle X' + mY' \right\rangle &= E/\left\langle (\alpha X + \beta Y) + m(\gamma X + \delta Y) \right\rangle \\ &= E/\left\langle X + \frac{\delta m + \beta}{\gamma m + \alpha} Y \right\rangle; \end{aligned}$$

we thus might naturally define the right action $m^\Psi := \frac{\delta m + \beta}{\gamma m + \alpha}$. The two actions are related as described above: mapping the auxiliary basis according to the action of Ψ on $E[\ell_A^{e_A}]^2$ is equivalent to mapping the integer m which defines the kernel of the isogeny according to the action of Ψ on $\mathbb{Z}/\ell_A^{e_A}\mathbb{Z}$. This action relates to a sort of "related-key attack" on SIDH: in an unauthentiacted 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 Ψ.

Note that for a given E_A for which $E_A = E/\left\langle P_A + m_A Q_B \right\rangle$ and $\Psi \in \Upsilon(\ell_A, e_A)$ it is easy to find $(E_B, X_B, Y_B), E'$ satisfying

$$E' = E_B/\left\langle X_B + m_A^\Psi Y_B \right\rangle;$$

simply set $E_B = E/\ker\phi$ for some isogeny ϕ, 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 $\phi_A : E \to E_A$ be an isogeny with kernel $\langle P_A + m_A Q_A\rangle$ for $n_A \leftarrow U(\mathbb{Z}/\ell_A^{e_A}\mathbb{Z})$, and let $\Psi_1, \Psi_2 \leftarrow U(\Upsilon_2(\ell_A, e_A))$. 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 $E_1 = E_B/\langle X_B + m_A^{\Psi_1} Y_B\rangle$ and $E_2 = E_B/\langle X_B + m_A^{\Psi_2} Y_B\rangle$ 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 $\phi_B : E \to E_B$ be an isogeny with kernel $\langle P_B + m_B Q_B\rangle$ for $n_B \leftarrow U(\mathbb{Z}/\ell_B^{e_B}\mathbb{Z})$, and let $\Psi_1, \Psi_2 \leftarrow U(\Upsilon_2(\ell_B, e_B))$. 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 $E_1 = E_A/\langle X_A + m_B^{\Psi_1} Y_A\rangle$ and $E_2 = E_A/\langle X_A + m_B^{\Psi_2} Y_A\rangle$ given (E, P_A, Q_A) and $(E_B, \phi_B(P_A), \phi_B(Q_A))$.*

We define $\mathrm{SGA}_A((E, P_A, Q_A, P_B, Q_B), E_A, \Psi_1, \Psi_2))$ and $\mathrm{SGA}_B((E, P_A, Q_A, P_B, Q_B), E_B, \Psi_1, \Psi_2))$ to be the set of all solutions to the respective problems with the given parameters, and we define the advantages

$$\begin{cases} \mathrm{Adv}_p^{\text{C-SGA}_A}(\mathcal{A}) = \mathbb{P}[\vartheta \leftarrow \mathtt{Ins}_{\text{SI-APC}_A}(p) : \mathrm{SGA}_A(\vartheta) \cap \mathcal{A}(\vartheta) \neq \varnothing] \\ \mathrm{Adv}_p^{\text{C-SGA}_B}(\mathcal{A}) = \mathbb{P}[\vartheta \leftarrow \mathtt{Ins}_{\text{SI-APC}_B}(p) : \mathrm{SGA}_B(\vartheta) \cap \mathcal{A}(\vartheta) \neq \varnothing] \end{cases}$$

and

$$\begin{cases} \mathrm{Adv}_p^{\text{C-SGA}_A}(t, n) = \max\{\mathrm{Adv}_p^{\text{C-SGA}_A}(\mathcal{A}) : \mathcal{A} \text{ runs in time } t \text{ and } |\mathcal{A}(\vartheta)| \leq n \text{ for all } \vartheta \in \mathtt{Ins}_{\text{C-SGA}_A}(p)\} \\ \mathrm{Adv}_p^{\text{C-SGA}_B}(t, n) = \max\{\mathrm{Adv}_p^{\text{C-SGA}_B}(\mathcal{A}) : \mathcal{A} \text{ runs in time } t \text{ and } |\mathcal{A}(\vartheta)| \leq n \text{ for all } \vartheta \in \mathtt{Ins}_{\text{C-SGA}_B}(p)\}. \end{cases}$$

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 $E/\langle X + m_A^{\Psi} Y\rangle$ or $E/\langle X + m_B^{\Psi} Y\rangle$ for any Ψ 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 Π 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* $\mathfrak{C}$ and *servers* $\mathfrak{S}$. 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 $P \in \mathfrak{C} \sqcup \mathfrak{S}$ 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 $A \in \mathfrak{C}$ has a *password* pwd_A, and each server has a collection $\{\text{pwd}_A^B\}_{A \in \mathfrak{C}}$ of *transformed passwords.* We may consider either a *symmetric* scenario, where $\text{pwd}_A^B = \text{pwd}_A \; \forall A \in \mathfrak{C}, B \in \mathfrak{S}$ (*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 pwd_A from a reasonably-sized[6] subset of $\{\text{pwd}_A^B\}_{B \in \mathfrak{S}}$, 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 (*e.g.,* [2, 6, 14]) in a pre-protocol "initialization" phase. Similarly, in this model of password-authenticated key establishment there is a pre-protocol initialization phase where users generate passwords and "install" them on servers (*i.e.*, servers get the corresponding transformed passwords) securely.

Party Instances. Associated to each party $U \in \mathfrak{C} \sqcup \mathfrak{S}$ 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 one additional "formal" party who "administrates" the game—the *challenger*. The security experiment proceeds in three stages

1. **Initialization:** The adversary chooses disjoint sets $\mathfrak{C}$ and $\mathfrak{S}$ of client and server identifiers. Each client $A \in \mathfrak{C}$ generates a password according to a probability distribution on their password-space: $\text{pwd}_A \leftarrow \mathcal{P}_A \; \forall A \in \mathfrak{C}$; then, each server $B \in \mathfrak{S}$ computes the transformed passwords $\{\text{pwd}_A^B\}_{A \in \mathfrak{C}}$. Password-generation happens out-of-view of the adversary, though the adversary knows each $\mathcal{P}_A$ and the maps $\text{pwd}_A \mapsto \text{pwd}_A^B$.
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) $\texttt{Send}(U, n, \text{msg})$: Message M 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 Π.
   (b) $\texttt{Reveal}(U, n)$: The party instance $\Omega_U^{(n)}$ reveals its session key $\text{sk}_U^{(n)}$ (if it exists).
   (c) $\texttt{Execute}(A, n, B, m)$: If $A \in \mathfrak{C}$, $B \in \mathfrak{S}$, and neither $\Omega_A^{(n)}$ nor $\Omega_B^{(m)}$ has been used, the challenger $\mathcal{C}$ instructs $\Omega_A^{(n)}$ to execute Π with $\Omega_B^{(n)}$. The transcript of this execution is then provided to $\mathcal{A}$.

[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.

(d) `Corrupt`(U): If $U \in \mathfrak{C}$, this returns pwd_U. If $U \in \mathfrak{S}$, this returns $\{\text{pwd}_A^U\}_{A \in \mathfrak{C}}$.
(e) `Test`(U, n): The challenger $\mathcal{C}$ chooses $b \in \{0, 1\}$ uniformly at random; if $b = 0$, the challenger delivers $\text{sk}_U^{(n)}$ to $\mathcal{A}$, while if $b = 1$, the challenger chooses $\widetilde{\text{sk}}$ from the secret key space uniformly at random and delivers $\widetilde{\text{sk}}$ to $\mathcal{A}$.

Remark 2. Our definition of `Corrupt` in item (d) above is known as the *weak corruption model* [1, Remark 3]. In the *strong corruption model*, a `Corrupt`(U) query also reveals the state of each oracle $\Omega_U^{(n)}, n \in \mathbb{N}$; this presents another way by which a secret session key can become known to the adversary, and the definition of freshness (Definition 11) would have to be modified in order for two-pass protocols (like the one we present in Section 3) to have any hope of being secure [1, Remark 7].

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

Definition 10 (Partnered Oracles). *A pair of oracles $\Omega_A^{(n)}$ and $\Omega_B^{(m)}$ are called* partnered *if all of the following are true:*
(a) $A \in \mathfrak{C}$ and $B \in \mathfrak{S}$, or $A \in \mathfrak{S}$ and $B \in \mathfrak{C}$.
(b) $\Omega_A^{(n)}$ and $\Omega_B^{(m)}$ have both accepted.
(c) $\Omega_A^{(n)}$'s peer is B, and $\Omega_B^{(m)}$'s peer is A (that is, A believes she is establishing a key with B, and vice versa*).*
(d) $\Omega_A^{(n)}$ and $\Omega_B^{(m)}$ have the same message transcript and session key.
(e) No other oracle $\Omega_U^{(k)}$ accepts with the same message transcript.

Definition 11 (Fresh Oracle). *An oracle $\Omega_U^{(n)}$ 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) `Reveal`(U, n) has been issued.
(b) `Reveal`(V, m) has been issued, where $\Omega_V^{(m)}$ is the partner to $\Omega_U^{(n)}$.
(c) `Corrupt`(V, i) was issued for some $V \in \mathfrak{C} \sqcup \mathfrak{S}$ and `Send`(U, n, M) was issued for some $M \in \mathcal{M}$.

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

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

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:

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

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 $A \in \mathfrak{C}$ wishes to establish a key with a server $B \in \mathfrak{S}$, who share a common password pwd_A. The setup is as follows: Fix a prime p of the form $\ell_A^{e_A} \ell_B^{e_B} \cdot f \pm 1$ where ℓ_A and ℓ_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 $\text{codom}\, H_A = \Upsilon_2(\ell_B, e_B)$ and $\text{codom}\, H_B = \Upsilon_2(\ell_A, e_A)$. They follow the following process:

1. A generates an ephemeral key and sends public information to B. In particular, A:
   (a) Chooses $n_A \in \{0, 1, \ldots, \ell_A^{e_A} - 1\}$ uniformly at random;
   (b) Constructs $G_A = P_A + n_A Q_A$;
   (c) Defines $E_A = E_K/\langle G_A \rangle$ and ϕ_A to be the isogeny defined on E_K with kernel $\langle G_A \rangle$;
   (d) Defines $\Psi_A = H_A(j(E_A)||\text{pwd}_A)$
   (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 $\ell_B^{e_B}$, abort;
   (b) Checks that $e_E(X, Y) = e_{E_K}(P_B, Q_B)^{\ell_A^{e_A}}$—if not, abort;
   (c) Computes $\Psi_{AB} = H_A(j(E)||\text{pwd}_A)$
   (d) Computes
   $$\begin{bmatrix} P_{AB} \\ Q_{AB} \end{bmatrix} = \Psi_{AB}^{-1} \cdot \begin{bmatrix} X \\ Y \end{bmatrix}$$
   (e) Chooses $n_B \in \{0, 1, \ldots, \ell_B^{e_B} - 1\}$ uniformly at random;
   (f) Constructs $G_B = P_B + n_B Q_B$;
   (g) Defines $E_B = E_K/\langle G_B \rangle$ and ϕ_B to be the isogeny defined on E_K with kernel $\langle G_B \rangle$;
   (h) Computes $\Psi_B = H_B(j(E_B)||\text{pwd}_A)$
   (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
   $$K_B = \text{KDF}((E, X, Y)||(E_B, X_B, Y_B)||j(E/\langle P_{AB} + n_B Q_{AB} \rangle)||\Psi_{AB}||\Psi_B)$$
3. Upon receiving (E', X', Y') from B, A:
   (a) Computes $\text{ord}_{E'}(X')$ and $\text{ord}_{E'}(Y')$—if either is not $\ell_A^{e_A}$, abort;
   (b) Checks that $e_{E'}(X', Y') = e_{E_K}(P_A, Q_A)^{\ell_B^{e_B}}$—if not, abort;
   (c) Computes $\Psi_{BA} = H_B(j(E')||\text{pwd}_A)$;
   (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
   $$K_A = \text{KDF}((E_A, X_A, Y_A)||(E', X', Y')||j(E/\langle P_{BA} + n_B Q_{BA} \rangle)||\Psi_A||\Psi_{BA})$$

4 Security of the Scheme

We prove the following result:

Theorem 1. *Suppose $\mathcal{A}$ plays Γ with protocol Π, runs in time T, and makes at most n_S, n_E, n_O, n_K queries to* `Send`, `Execute`, *the random oracles, and* KDF, *respectively. Then*

$$\begin{aligned}
\text{Adv}_\Gamma^\Pi(\mathcal{A}) \leq & \frac{n_S}{L} + \frac{2(n_S + n_E)(n_S + n_E + n_O + n_K)}{\min\{\ell_A^{e_A}, \ell_B^{e_B}\}} + \frac{2n_K}{\min\{\ell_A^{3e_A-2}(\ell_A - 1), \ell_B^{3e_B-2}(\ell_B - 1)\}} \\
& + 2n_S \left(\text{Adv}_p^{\text{C-SGA}_A}(t, n_K(n_K - 1)) + \text{Adv}_p^{\text{C-SGA}_B}(t, n_K(n_K - 1)) \right) + 2n_E \text{Adv}_p^{\text{C-SIDH}}(t, n_K) \\
& + 2(n_S + n_E) \left((\ell_B + 1)\text{Adv}_p^{\text{SI-APC}_A}(t, n_K) + (\ell_A + 1)\text{Adv}_p^{\text{SI-APC}_B}(t, n_K) \right) \\
& + 4(n_S + n_E) \left(\text{Adv}_p^{\text{SI-APD}_A}(t) + \text{Adv}_p^{\text{SI-APD}_B}(t) \right)
\end{aligned}$$

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:

- $\texttt{ClientPasswordTest}(A, i, B, \text{pwd})$: For some $E, X, Y, E', X', Y', \Psi, \Psi'$, $\mathcal{A}$ makes queries:
  - $\text{KDF}((E, X, Y)||(E', X', Y')||j(E'')||\Psi||\Psi')$;
  - $\texttt{Send}(A, i, (\texttt{Initiate}, B))$ with output (E, X, Y);
  - $\texttt{Send}(A, i, (E', X', Y'))$;
  - $H_A(j(E)||\text{pwd}) = \Psi$ and $H_B(j(E')||\text{pwd}) = \Psi'$

  and $E'' \cong \text{SIDH}_A(E, E', \Psi'^{-1}\left[\begin{smallmatrix} X' \\ Y' \end{smallmatrix}\right])$. The value associated to this event is

  $$\text{sk}_A^{(i)} = \text{KDF}((E, X, Y)||(E', X', Y')||j(E'')||\Psi||\Psi')$$

  Intuitively, $\mathcal{A}$ is testing whether $\text{pwd}_A = \text{pwd}$ by running a key establishment instance with client A (pretending to be server B).
- $\texttt{ServerPasswordTest}(B, j, A, \text{pwd})$: For some $E, X, Y, E', X', Y', \Psi, \Psi'$, $\mathcal{A}$ makes queries:
  - $\text{KDF}((E, X, Y)||(E', X', Y')||j(E'')||\Psi||\Psi')$;
  - $\texttt{Send}(B, j, (E, X, Y))$ with output (E', X', Y');
  - $H_A(j(E)||\text{pwd}) = \Psi$ and $H_B(j(E')||\text{pwd}) = \Psi'$

  and $E'' \cong \text{SIDH}_B(E, E', \Psi^{-1}\left[\begin{smallmatrix} X \\ Y \end{smallmatrix}\right])$. The value associated to this event is

  $$\text{sk}_B^{(i)} = \text{KDF}((E, X, Y)||(E', X', Y')||j(E'')||\Psi||\Psi')$$

  Intuitively, $\mathcal{A}$ is testing whether $\text{pwd}_A = \text{pwd}$ by running a key establishment instance with server B (pretending to be client A).
- $\texttt{PassivePasswordTest}(A, i, B, j, \text{pwd})$: For some $E, X, Y, E', X', Y', \Psi, \Psi'$, $\mathcal{A}$ makes queries:
  - $\text{KDF}((E, X, Y)||(E', X', Y')||j(E'')||\Psi||\Psi')$
  - $\texttt{Execute}(A, i, B, j)$ with transcript $\text{msg}_A^{(i)} = (E, X, Y), \text{msg}_B^{(j)} = (E', X', Y')$;
  - $H_A(j(E)||\text{pwd}) = \Psi$ and $H_B(j(E')||\text{pwd}) = \Psi'$

  where $E'' \cong \text{SIDH}(E, E')$. The value associated to this event is

  $$\text{sk}_A^{(i)} = \text{sk}_B^{(j)} = \text{KDF}((E, X, Y)||(E', X', Y')||j(E'')||\Psi||\Psi')$$

  Intuitively, $\mathcal{A}$ is testing whether $\text{pwd}_A = \text{pwd}$ by passively observing an instance of A establishing a key with B and guessing at the password.
- $\texttt{ActiveCorrectPassword}$: Either a $\texttt{ClientPasswordTest}(A, i, B, \text{pwd}_A)$ event occurs for some $A \in \mathfrak{C}, B \in \mathfrak{S}$ and $i \in \mathbb{N}$, or a $\texttt{ServerPasswordTest}(B, j, A, \text{pwd}_A)$ event occurs for some $A \in \mathfrak{C}, B \in \mathfrak{S}$ and $j \in \mathbb{N}$. As the name suggests, this is precisely the event that $\mathcal{A}$ has guessed a client's password in a session in which they have actively participated.
- $\texttt{PassiveCorrectPassword}$: A $\texttt{PassivePasswordTest}(A, i, B, j, \text{pwd}_A)$ event occurs for some $A \in \mathfrak{C}, B \in \mathfrak{S}$, and $i, j \in \mathbb{N}$. As the name suggests, this is precisely the event that $\mathcal{A}$ guesses a client's password correctly in a session which they have only observed passively.
- $\texttt{2ClientPassword}$: 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 $\texttt{ClientPasswordTest}(A, i, B, \text{pwd})$ and $\texttt{ClientPasswordTest}(A, i, B, \text{pwd}')$ occur.
- $\texttt{2ServerPassword}$: 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 $\texttt{ServerPasswordTest}(B, j, A, \text{pwd})$ and $\texttt{ServerPasswordTest}(B, j, A, \text{pwd}')$ occur.

Proof. We give a sequence of games $\{\Gamma_i\}_{i=0}^{9}$, where Γ_0 is the original security game, and Γ_t 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.

Γ_0*:* The original security game.

Γ_1: The same as Γ_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 `Send` query, the challenger computes the appropriate party's output and forwards it to the adversary.
3. Upon receiving an `Execute` query, the challenger simulates the appropriate parties and forwards the transcript of the session to the adversary.
4. Upon receiving a `Reveal` query, the challenger reveals the appropriate session key (if it exists).
5. Upon receiving a `Corrupt` query, the challenger reveals the appropriate password.

Claim. For any adversary $\mathcal{A}$, $\mathrm{Adv}^{\Pi}_{\Gamma_0}(\mathcal{A}) = \mathrm{Adv}^{\Pi}_{\Gamma_1}(\mathcal{A})$.

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

Γ_2: The same as Γ_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 ($E_K/\langle P_A + n_U Q_A\rangle$ or $E_K/\langle P_B + n_U Q_B\rangle$ as appropriate) has been used by the adversary as input to a `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 `Send`, `Execute`, the random oracles, and KDF, respectively, we have $\mathrm{Adv}^{\Pi}_{\Gamma_1}(\mathcal{A}) \leq \mathrm{Adv}^{\Pi}_{\Gamma_2}(\mathcal{A}) + \frac{2(n_S+n_E)(n_S+n_E+n_O+n_K)}{\min\{\ell_A^{e_A}, \ell_B^{e_B}\}}$.

Proof. The numbers of valid ephemeral keys for clients and for servers are $\ell_A^{e_A}$ and $\ell_B^{e_B}$, respectively. It follows then that if the challenger participating in game Γ_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

$$\mathbb{P}[\Omega_U^{(i)}\text{'s ephemeral key is unused}] \geq \begin{cases} \frac{\ell_A^{e_A}-(k-1)}{\ell_A^{e_A}} \geq \frac{\ell_A^{e_A}-(n_S+n_E+n_O+n_K)}{\ell_A^{e_A}} & \text{If } U \in \mathfrak{C} \\ \frac{\ell_B^{e_B}-(k-1)}{\ell_B^{e_B}} \geq \frac{\ell_B^{e_B}-(n_S+n_E+n_O+n_K)}{\ell_B^{e_B}} & \text{If } U \in \mathfrak{S} \end{cases}$$

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

$$\begin{aligned} \mathbb{P}[\Omega_U^{(i)}\text{'s ephemeral key has been used}] &\leq \begin{cases} \frac{(n_S+n_E+n_O+n_K)}{\ell_A^{e_A}} & \text{If } U \in \mathfrak{C} \\ \frac{(n_S+n_E+n_O+n_K)}{\ell_B^{e_B}} & \text{If } U \in \mathfrak{S} \end{cases} \\ &\leq \max\left\{\frac{(n_S+n_E+n_O+n_K)}{\ell_A^{e_A}}, \frac{(n_S+n_E+n_O+n_K)}{\ell_B^{e_B}}\right\} \\ &= \frac{(n_S+n_E+n_O+n_K)}{\min\{\ell_A^{e_A}, \ell_B^{e_B}\}} \end{aligned}$$

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

$$\mathrm{Adv}^{\Pi}_{\Gamma_1}(\mathcal{A}) - \mathrm{Adv}^{\Pi}_{\Gamma_2}(\mathcal{A}) \leq \frac{2(n_S+n_E)(n_S+n_E+n_O+n_K)}{\min\{\ell_A^{e_A}, \ell_B^{e_B}\}}$$

from which the result follows. □

Γ_3: The same as Γ_2, except that `Send` and `Execute` queries are executed without using random oracles, and future random oracle queries are answered in a way which is consistent with previous `Send` and `Execute` queries whenever possible. In particular,

1. In an `Execute`(A, i, B, j) query for which $\Omega_A^{(i)}$ and $\Omega_B^{(j)}$ are unused, set

(a) $n_A^{(i)} \leftarrow U(\{0,1,\ldots,\ell_A^{e_A}-1\})$ and $n_B^{(j)} \leftarrow U(\{0,1,\ldots,\ell_B^{e_B}-1\})$
(b) $\phi_A^{(i)}$ to be the isogeny defined on E_K with kernel $\langle P_A + n_A^{(i)} Q_A \rangle$, and $\phi_B^{(j)}$ to be the isogeny defined on E_K with kernel $\langle P_B + n_B^{(j)} Q_B \rangle$
(c) $E_A^{(i)} = E_K/\ker \phi_A^{(i)}$ and $E_B^{(j)} = E_K/\ker \phi_B^{(j)}$
(d) $\Psi_A \leftarrow U(\Upsilon_2(\ell_A, e_A))$
(e) $\Psi_B \leftarrow U(\Upsilon_2(\ell_B, e_B))$
(f) $\mathrm{msg}_A^{(i)} = (E_A^{(i)}, \Psi_A \left[\begin{smallmatrix} \phi_A(P_B) \\ \phi_A(Q_B) \end{smallmatrix}\right])$
(g) $\mathrm{msg}_B^{(i)} = (E_B^{(i)}, \Psi_B \left[\begin{smallmatrix} \phi_B(P_A) \\ \phi_B(Q_A) \end{smallmatrix}\right])$
(h) $\mathrm{sk}_A^{(i)} = \mathrm{sk}_B^{(j)} \leftarrow U(\{0,1\}^\lambda)$

2. When $A \in \mathfrak{C}$ is instructed to initiate unused session i with $B \in \mathfrak{S}$, set
(a) $n_A^{(i)} \leftarrow U(\{0,1,\ldots,\ell_A^{e_A}-1\})$;
(b) $\phi_A^{(i)}$ to be the isogeny defined on E_K with kernel $\langle P_A + n_A^{(i)} Q_A \rangle$;
(c) $E_A^{(i)} = E_K/\ker \phi_A^{(i)}$;
(d) $\Psi_A \leftarrow U(\Upsilon_2(\ell_B, e_B))$
(e) $\mathrm{msg}_A^{(i)} = (E_A^{(i)}, \Psi_A \left[\begin{smallmatrix} \phi_A(P_B) \\ \phi_A(Q_B) \end{smallmatrix}\right])$

3. In a $\texttt{Send}(B, j, \mathrm{msg})$ query where $B \in \mathfrak{S}$ and $\Omega_B^{(j)}$ is unused, set
(a) $n_B^{(j)} \leftarrow U(\{0,1,\ldots,\ell_B^{e_B}-1\})$;
(b) $\phi_B^{(j)}$ to be the isogeny defined on E_K with kernel $\langle P_B + n_B^{(j)} Q_B \rangle$;
(c) $E_B^{(j)} = E_K/\ker \phi_B^{(j)}$;
(d) $\Psi_B \leftarrow U(\Upsilon_2(\ell_A, e_A))$
(e) $\mathrm{msg}_B^{(i)} = (E_B^{(i)}, \Psi_B \left[\begin{smallmatrix} \phi_B(P_A) \\ \phi_B(Q_A) \end{smallmatrix}\right])$
(f) $\mathrm{sk}_B^{(j)} \leftarrow U(\{0,1\}^\lambda)$.

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

5. In a $\mathrm{KDF}((E,X,Y)||(E',X',Y')||j(E'')||\Psi||\Psi')$ query:
(a) If this query causes an $\texttt{ActiveCorrectPassword}$ or $\texttt{PassiveCorrectPassword}$ event, output the associated value; otherwise,
(b) Output a uniformly random element of $\{0,1\}^\lambda$.

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

$$\mathrm{Adv}^{\Pi}_{\Gamma_2}(\mathcal{A}) \leq \mathrm{Adv}^{\Pi}_{\Gamma_3}(\mathcal{A}) + \frac{2n_K}{\min\{\ell_A^{3e_A-2}(\ell_A-1), \ell_B^{3e_B-2}(\ell_B-1)\}} + 2(n_S+n_E)\left((\ell_B+1)\mathrm{Adv}_p^{\text{SI-APC}_A}(t,n_K) + (\ell_A+1)\mathrm{Adv}_p^{\text{SI-APC}_B}(t,n_K)\right)$$

Proof. Consider the ways that a KDF query could occur, and what happens in Γ_2 when they do:

1. When $\Omega_B^{(j)}$ ($B \in \mathfrak{S}$) 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 $\Omega_A^{(i)}$ ($A \in \mathfrak{C}$) receives a message and generates a key, there are three possibilities:

(a) $\Omega_A^{(i)}$ is partnered with some $\Omega_B^{(j)}$ for some $B \in \mathfrak{S}$[7], in which case $\mathrm{sk}_A^{(i)} = \mathrm{sk}_B^{(j)}$; or,
(b) $\Omega_A^{(i)}$ is unpartnered and a $\texttt{ClientPasswordTest}(A, i, U, \mathrm{pwd}_A)$ events occurs, in which case $\mathrm{sk}_A^{(i)}$ is set to the value of the event (the corresponding KDF value); or
(c) $\Omega_A^{(i)}$ is unpartnered and no $\texttt{ClientPasswordTest}(A, i, U, \mathrm{pwd}_A)$ events occurs, in which case $\mathrm{sk}_A^{(i)}$ is chosen independently of all previous queries, since the KDF query that yields it is new.

3. When $\mathcal{A}$ makes a $\mathrm{KDF}((E, X, Y)||(E', X', Y')||j(E'')||\Psi||\Psi')$ query, there are four (not mutually exclusive) possibilities:
(a) It causes a $\texttt{ClientPasswordTest}, \texttt{ServerPasswordTest}$ or $\texttt{PassivePasswordTest}$ event, in which case the output is the associated session key;
(b) Ψ is correct, but no $H_A(j(E)||\mathrm{pwd}_A)$ query has been made;
(c) Ψ' is correct, but no $H_B(j(E')||\mathrm{pwd}_A)$ 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 Γ_3 is indistinguishable from Γ_2 unless case 3.(b) or 3.(c) occurs. Consider the circumstances under which a $\mathrm{KDF}((E, X, Y)||(E', X', Y')||j(E'')||\Psi||\Psi')$ query in one of the two cases could occur:

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

$$\frac{2n_K}{\min\{\ell_A^{3e_A-2}(\ell_A - 1), \ell_B^{3e_B-2}(\ell_B - 1)\}}$$

3. (E, X, Y) has been the output of a $\texttt{Send}(A, i, (\texttt{Initiate}, B))$ query, and Ψ is correct. We use an adversary who causes this case to solve the SI-APC$_A$ problem.
Given an instance $(E_K, P_A, Q_A, P_B, Q_B, E_A)$ of the SI-APC$_A$ problem and an adversary $\mathcal{A}$ who causes this scenario with probability ϵ, run $\mathcal{A}$ 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 $\widehat{\mathrm{msg}} = (E_A, X, Y)$ for (X, Y) chosen uniformly at random from

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

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

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

(where ϕ_A is the true isogeny $\phi_A \colon E_K \to E_A$) satisfies $\Psi = \left[\begin{smallmatrix} \alpha & \beta \\ \gamma & \delta \end{smallmatrix}\right] \in \Upsilon_2(\ell_A, e_A)$.
By choice of W, $\Psi \in \mathrm{SL}_2(\ell_A, e_A)$, and so $\Psi \in \Upsilon_2(\ell_A, e_A)$ with probability

$$\mathbb{P}[\Psi \in \Upsilon_2(\ell_A, e_A)] = [\mathrm{SL}_2(\ell_A, e_A) : \Upsilon_2(\ell_A, e_A)]^{-1} = (\ell_A + 1)^{-1}.$$

When $\widehat{\mathrm{msg}}$ is a correctly-formatted message, the simulation is indistinguishable from Γ_3 from the perspective of the adversary. In any case, with probability ϵ, the scenario occurs in the way described, and with probability $(n_S + n_E)^{-1}$ it occurs for $\widehat{\mathrm{msg}}$. Moreover, at most n_K queries are made at all, and we have at most n_K queries of the form $\mathrm{KDF}(\widehat{\mathrm{msg}}||\cdot||\cdot||\Psi_i||\cdot)$. The probability that the correct Ψ 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 $\{\Psi_i^{-1}\left[\begin{smallmatrix} X \\ Y \end{smallmatrix}\right]\}_i$ to our instance of SI-APC$_A$. It follows that

$$\epsilon \leq (n_S + n_E)(\ell_B + 1)\mathbb{P}[\Psi \in \{\Psi_i\}_i] \leq (n_S + n_E)(\ell_B + 1)\mathrm{Adv}_p^{\text{SI-APC}_A}(t, n_K)$$

[7] Notably, because of the definition of partnering, this partner $\Omega_B^{(j)}$ is unique.

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

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

By the above arguments,

$$\begin{aligned}\text{Adv}_{\Gamma_2}^{\Pi}(\mathcal{A}) \leq \text{Adv}_{\Gamma_3}^{\Pi}(\mathcal{A}) + \frac{2n_K}{\min\{\ell_A^{3e_A-2}(\ell_A-1), \ell_B^{3e_B-2}(\ell_B-1)\}} \\ + 2(n_S+n_E)\left((\ell_B+1)\text{Adv}_p^{\text{SI-APC}_A}(t,n_K) + (\ell_A+1)\text{Adv}_p^{\text{SI-APC}_B}(t,n_K)\right)\end{aligned}$$

as required. □

Γ_4: The same as Γ_3, except that there is no checking for `PassiveCorrectPassword` events.

Claim. For any adversary $\mathcal{A}$ that runs in time t and makes at most n_K queries to the KDF oracle we have $\text{Adv}_{\Gamma_3}^{\Pi}(\mathcal{A}) \leq \text{Adv}_{\Gamma_4}^{\Pi}(\mathcal{A}) + 2n_E\text{Adv}_p^{\text{C-SIDH}}(t, n_K)$.

Proof. It is clear that Γ_3 is indistinguishable from Γ_4 unless `PassiveCorrectPassword` occurs. Suppose it occurs with probability ϵ. We build an SIDH 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 `Execute` query. When $\mathcal{A}$ makes its k^{th} `Execute` query, instead of choosing the messages as usual, instead choose $\Psi \leftarrow U(\Upsilon(\ell_B, e_B))$ and $\Psi' \leftarrow U(\Upsilon(\ell_A, e_A))$ and set the transcript as

$$\widehat{\text{msg}}_A = (E_A, \Psi \left[\begin{smallmatrix} X_A \\ Y_A \end{smallmatrix}\right]), \text{ and } \widehat{\text{msg}}_B = (E_B, \Psi' \left[\begin{smallmatrix} X_B \\ Y_B \end{smallmatrix}\right]).$$

With probability ϵ, $\mathcal{A}$ will cause `PassiveCorrectPassword`, and with probability at least n_E^{-1} it will occur for the `Execute` command we chose. Now, $\mathcal{A}$ makes at most n_K queries of the form $\text{KDF}(\widehat{\text{msg}}_A||\widehat{\text{msg}}_B||j(\hat{E}_i)||\Psi||\Psi')$ and so produces a list of size at most n_K of candidate solutions $\hat{E}_j$ to the SIDH instance. It follows that $\mathbb{P}[\text{SIDH}(E_A, E_B) \in \{\hat{E}_j\}_j] \geq n_E^{-1}\epsilon$, which implies $\epsilon \leq n_E\mathbb{P}[\text{SIDH}(E_A, E_B) \in \{\hat{E}_j\}_j] \leq n_E\text{Adv}_p^{\text{SIDH}}(t, n_K)$ from which the result follows. □

Γ_5: The same as Γ_4, except that if `ActiveCorrectPassword` occurs, the game ends and the adversary wins.

Claim. For any adversary $\mathcal{A}$, $\text{Adv}_{\Gamma_4}^{\Pi}(\mathcal{A}) \leq \text{Adv}_{\Gamma_5}^{\Pi}(\mathcal{A})$.

Proof. This is clear; we have simply added more win conditions for the adversary. □

Γ_6: The same as Γ_5, except that if `2ServerPassword` occurs, the game ends and the adversary loses.

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

$$\text{Adv}_{\Gamma_5}^{\Pi}(\mathcal{A}) \leq \text{Adv}_{\Gamma_6}^{\Pi}(\mathcal{A}) + 2n_S\text{Adv}_p^{\text{C-SGA}_B}(t, n_K(n_K-1))$$

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

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

$$\left\{\left((E_i'', \text{Min}(\Psi_1)^{-T}\Psi_i''^{-1}\left[\begin{smallmatrix} X \\ Y \end{smallmatrix}\right]), (E_j'', \text{Min}(\Psi_2)^{-T}\Psi_j''^{-1}\left[\begin{smallmatrix} X \\ Y \end{smallmatrix}\right])\right)\right\}_{i \neq j}$$

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

$$E_i'' = \text{SIDH}_B(E, E_B, \Psi_i''^{-1}\left[\begin{smallmatrix} X \\ Y \end{smallmatrix}\right]) \text{ and } E_j'' = \text{SIDH}_B(E, E_B, \Psi_j''^{-1}\left[\begin{smallmatrix} X \\ Y \end{smallmatrix}\right]).$$

Note that

$$\begin{aligned} E_i'' &= E/\left\langle [1, n_B]\Psi_i''^{-1}\left[\begin{smallmatrix} X \\ Y \end{smallmatrix}\right]\right\rangle \\ &= E/\left\langle [1, n_B]\mathrm{Min}(\Psi_1)^T\left(\mathrm{Min}(\Psi_1)^{-T}\Psi_i''^{-1}\left[\begin{smallmatrix} X \\ Y \end{smallmatrix}\right]\right)\right\rangle \\ &= E/\left\langle [1, n_B^{\Psi_1}]\left(\mathrm{Min}(\Psi_1)^{-T}\Psi_i''^{-1}\left[\begin{smallmatrix} X \\ Y \end{smallmatrix}\right]\right)\right\rangle \end{aligned}$$

and

$$E_j'' = E/\left\langle [1, n_B^{\Psi_2}]\left(\mathrm{Min}(\Psi_2)^{-T}\Psi_j''^{-1}\left[\begin{smallmatrix} X \\ Y \end{smallmatrix}\right]\right)\right\rangle$$

But then $\left((E_i'', \mathrm{Min}(\Psi_1)^{-T}\Psi_i''^{-1}\left[\begin{smallmatrix} X \\ Y \end{smallmatrix}\right]), (E_j'', \mathrm{Min}(\Psi_2)^{-T}\Psi_j''^{-1}\left[\begin{smallmatrix} X \\ Y \end{smallmatrix}\right])\right)$ is precisely a solution to our instance of C-SGA$_B$. It follows that if `2ServerPassword` occurs with probability ϵ, we have

$$n_S^{-1}\epsilon \leq \mathrm{Adv}_p^{\text{C-SGA}_B}(t, n_K(n_K-1)) \implies \epsilon \leq n_S \mathrm{Adv}_p^{\text{C-SGA}_B}(t, n_K(n_K-1)).$$

Since games Γ_5 and Γ_6 are indistinguishable unless `2ServerPassword` occurs, we find that

$$\mathrm{Adv}_{\Gamma_5}^{\Pi}(\mathcal{A}) \leq \mathrm{Adv}_{\Gamma_6}^{\Pi}(\mathcal{A}) + 2n_S \mathrm{Adv}_p^{\text{C-SGA}_B}(t, n_K(n_K-1))$$

as required. □

Γ_7: The same as Γ_6, except that if `2ClientPassword` occurs, the game ends and the adversary loses.

Claim. For any adversary $\mathcal{A}$ which runs in time t and makes at most n_S `Send` queries and n_K KDF queries,

$$\mathrm{Adv}_{\Gamma_6}^{\Pi}(\mathcal{A}) \leq \mathrm{Adv}_{\Gamma_7}^{\Pi}(\mathcal{A}) + 2n_S \mathrm{Adv}_p^{\text{C-SGA}_A}(t, n_K(n_K-1)).$$

Proof. This follows by an argument analogous to the previous claim. □

Γ_8: The same as Γ_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

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

Claim. For any adversary $\mathcal{A}$ running in time t,

$$\mathrm{Adv}_{\Gamma_7}^{\Pi}(\mathcal{A}) \leq \mathrm{Adv}_{\Gamma_8}^{\Pi}(\mathcal{A}) + 2(n_S + n_E)\mathrm{Adv}_p^{\text{SI-APD}_A}(t)$$

Proof. We proceed by a hybrid argument; let $\{\Gamma_7^{(j)}\}_{j=0}^{n_S+n_E}$ be a sequence obtained from Γ_7 by replacing only the first j client messages as described. We will show that for $1 \leq j \leq n_S + n_E$,

$$\mathrm{Adv}_{\Gamma_7^{(j-1)}}^{\Pi}(\mathcal{A}) \leq \mathrm{Adv}_{\Gamma_7^{(j)}}^{\Pi}(\mathcal{A}) + 2\mathrm{Adv}_p^{\text{SI-APD}_A}(t).$$

Suppose we are faced with an instance (E_A, X_A, Y_A) of SI-APD$_A$. Run the simulation against $\mathcal{A}$ as usual, but replace the first $j-1$ client messages as described, and replace the j^{th} client message with $\widehat{\mathrm{msg}}_A = (E_A, \Psi\left[\begin{smallmatrix} X_A \\ Y_A \end{smallmatrix}\right])$ for $\Psi \leftarrow U(\Upsilon(\ell_B, e_B))$. Play the game to completion with $\mathcal{A}$, 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 $\mathcal{A}$ wins the security game, and guess that they are random points otherwise. Observe that

$$\begin{aligned} \mathbb{P}[\text{We guess correctly}] &= \mathbb{P}[(X_A,Y_A) \text{ are correct}|\mathcal{A} \text{ wins}] + \mathbb{P}[(X_A,Y_A) \text{ are random}|\mathcal{A} \text{ does not win}] \\ &\geq \mathbb{P}[(X_A,Y_A) \text{ are correct}] \cdot \mathbb{P}[\mathcal{A} \text{ wins}|(X_A,Y_A) \text{ are correct}] \\ &\quad + \mathbb{P}[(X_A,Y_A) \text{ are random}] \cdot \mathbb{P}[\mathcal{A} \text{ does not win}|(X_A,Y_A) \text{ are random}] \\ &= \frac{1}{2}\left(\frac{1}{2} + \frac{1}{2}\mathrm{Adv}_{\Gamma_7^{(j-1)}}^{\Pi}(\mathcal{A})\right) + \frac{1}{2}\left(\frac{1}{2} - \frac{1}{2}\mathrm{Adv}_{\Gamma_7^{(j)}}^{\Pi}(\mathcal{A})\right). \end{aligned}$$

This procedure takes time at most t, and so by definition $\mathbb{P}[\text{We guess correctly}] \leq \frac{1}{2} + \mathrm{Adv}_p^{\text{SI-APD}_A}(t)$. Substituting this into the above and simplifying we find that for each j,

$$\frac{1}{2}+\mathrm{Adv}_p^{\text{SI-APD}_A}(t) \geq \frac{1}{2}+\frac{1}{4}\mathrm{Adv}^{\Pi}_{\Gamma_7^{(j-1)}}(\mathcal{A})-\frac{1}{4}\mathrm{Adv}^{\Pi}_{\Gamma_7^{(j)}}(\mathcal{A}) \implies \mathrm{Adv}^{\Pi}_{\Gamma_7^{(j-1)}}(\mathcal{A}) \leq \mathrm{Adv}^{\Pi}_{\Gamma_7^{(j)}}(\mathcal{A})+4\mathrm{Adv}_p^{\text{SI-APD}_A}(t).$$

There are $n_S + n_E + 1$ hybrids, and so, substituting $\Gamma_7^{(0)} = \Gamma_7$ and $\Gamma_7^{(n_S+n_E)} = \Gamma_8$ we obtain

$$\mathrm{Adv}^{\Pi}_{\Gamma_7}(\mathcal{A}) \leq \mathrm{Adv}^{\Pi}_{\Gamma_8}(\mathcal{A}) + 4(n_S + n_E)\mathrm{Adv}_p^{\text{SI-APD}_A}(t)$$

as required. □

Γ_9*:* The same as Γ_8, except that whenever a server would send a message (E_B, X_B, Y_B), we instead send $(E_B, \hat{X}_B, \hat{Y}_B)$, for $(\hat{X}_B, \hat{Y}_B)$ chosen uniformly at random from

$$W = \left\{(S,T) \in E_B[\ell_A^{e_A}]^2 \,:\, e_{E_B}(S,T) = e_{E_K}(P_A, Q_A)^{\ell_B^{e_B}} \text{ and } \mathrm{ord}_{E_B}(S) = \mathrm{ord}_{E_B}(T) = \ell_A^{e_A}\right\}.$$

Claim. For any adversary $\mathcal{A}$ running in time t, $\mathrm{Adv}^{\Pi}_{\Gamma_8}(\mathcal{A}) \leq \mathrm{Adv}^{\Pi}_{\Gamma_9}(\mathcal{A}) + 2(n_S + n_E)\mathrm{Adv}_p^{\text{SI-APD}_B}(t)$

Proof. This follows by an argument analogous to the previous claim. □

Claim. For any adversary $\mathcal{A}$ that makes at most n_S `Send` queries, $\mathrm{Adv}^{\Pi}_{\Gamma_9}(\mathcal{A}) \leq \frac{n_S}{L}$ where L is the size of the password space.

Proof. Note that

$$\begin{aligned}\mathbb{P}[\mathcal{A} \text{ succeeds}] \leq\ & \mathbb{P}[\texttt{ActiveCorrectPassword}] \\ & + \mathbb{P}[\mathcal{A} \text{ succeeds} | \text{No } \texttt{ActiveCorrectPassword}] \cdot \mathbb{P}[\text{No } \texttt{ActiveCorrectPassword}]\end{aligned}$$

In game Γ_9, the messages contain no information about the passwords, since the auxiliary points are random. It follows that $\mathbb{P}[\texttt{ActiveCorrectPassword}] \leq \frac{n_S}{L}$, since the adversary causes at most n_S total `ClientPasswordTest` and `ServerPasswordTest` events.

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

There are two kinds of queries we need to consider:

1. `Reveal` queries. Since $\Omega_U^{(i)}$ is fresh, $\mathcal{A}$ has made neither a $\texttt{Reveal}(U,i)$ query nor a $\texttt{Reveal}(V,j)$ query, where $\Omega_V^{(j)}$ is the partner to $\Omega_U^{(i)}$ (if it exists). For any other instance $\Omega_W^{(k)}$, the session key $\mathrm{sk}_W^{(k)}$ is computed as
$$\mathrm{sk}_W^{(k)} = \mathrm{KDF}(\mathrm{msg}_W^{(k)} || \mathrm{msg}_{W'}^{(k')} || j(E_W^{(k)}) || \Psi || \Psi')$$
where $\mathrm{msg}_W, \mathrm{msg}_{W'}$ is the transcript of the protocol. By Γ_2, we know that the message transcript from the perspective of $\Omega_U^{(i)}$ is different from that of $\Omega_W^{(k)}$, since otherwise we would have aborted. Since KDF is a random oracle, $\mathrm{sk}_W^{(i)}$ is independent from $\mathrm{sk}_U^{(i)}$, and hence a $\texttt{Reveal}(W,k)$ query gives no advantage to $\mathcal{A}$ for the given `Test` query.
2. KDF queries. Note that in games Γ_4 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 $\mathrm{sk}_U^{(i)}$ is set. There are three cases to consider:
   (a) $U \in \mathfrak{C}$, and $\Omega_U^{(i)}$ is unpartnered. In this case, cases 4.(a) and 4.(b) in the description of Γ_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 $\mathrm{sk}_U^{(i)}$ is chosen without regard for anything that has previously occurred.

(b) $U \in \mathfrak{C}$, and $\Omega_U^{(i)}$ is partnered with some $\Omega_V^{(j)}$. In this case, $\mathrm{sk}_U^{(i)} = \mathrm{sk}_V^{(j)}$. But according to point 3.(f) in the description of Γ_3, $\mathrm{sk}_V^{(j)}$ is chosen without regard for anything that has previously occurred.

(c) $U \in \mathfrak{S}$. Again we are in point 3.(f) in the description of Γ_3, and so $\mathrm{sk}_U^{(i)}$ is chosen without regard for anything that has previously occurred.

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

Now,

$$\begin{aligned}
\mathbb{P}[\mathcal{A}\text{ succeeds}] &= \mathbb{P}[\texttt{ActiveCorrectPassword}] + \frac{1}{2}\mathbb{P}[\text{No } \texttt{ActiveCorrectPassword}] \\
&= \mathbb{P}[\texttt{ActiveCorrectPassword}] + \frac{1}{2}\left(1 - \mathbb{P}[\texttt{ActiveCorrectPassword}]\right) \\
&= \frac{1}{2} + \frac{1}{2}\mathbb{P}[\texttt{ActiveCorrectPassword}] \\
&\leq \frac{1}{2} + \frac{1}{2}\frac{n_S}{L}
\end{aligned}$$

from which it immediately follows that $\mathrm{Adv}_{\Gamma_9}^{\Pi}(\mathcal{A}) \leq \frac{n_S}{L}$. Using the preceding 8 claims, we finally obtain the desired result. □

Corollary 1. *Under the* SIDH, SI-APC$_A$, SI-APC$_B$, SI-APD$_A$, SI-APD$_B$, 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

1. Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Bart Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, pages 139–155, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.
2. Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *Advances in Cryptology — CRYPTO '93*, pages 232–249, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.
3. Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *IEEE Symposium On Research In Security And Privacy*, pages 72–84, 1992.
4. Bouncy castle project. https://www.bouncycastle.org/.
5. Reinier Bröker, Denis Charles, and Kristin Lauter. Evaluating large degree isogenies and applications to pairing based cryptography. In *Pairing '08: Proceedings of the 2nd International Conference on Pairing-Based Cryptography*, pages 100–112, 2008.
6. Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In Birgit Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, pages 453–474, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.
7. Andrew Childs, David Jao, and Vladimir Soukharev. Constructing elliptic curve isogenies in quantum subexponential time. *J. Math. Cryptol.*, 8(1):1–29, 2014.

8. Craig Costello, Patrick Longa, and Michael Naehrig. Efficient algorithms for supersingular isogeny diffie-hellman. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, pages 572–601, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.
9. Luca De Feo, David Jao, and Jérôme Plût. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. *J. Math. Cryptol.*, 8:209–247, 2014.
10. Jintai Ding, Saed Alsayigh, Jean Lancrenon, Saraswathy RV, and Michael Snook. Provably secure password authenticated key exchange based on rlwe for the post-quantum world. In Helena Handschuh, editor, *Topics in Cryptology – CT-RSA 2017*, pages 183–204, Cham, 2017. Springer International Publishing.
11. David Jao, Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalali, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Joost Renes, Vladimir Soukharev, and David Ubanik. Supersingular isogeny key encapsulation. Technical report, EvolutionQ, 2017.
12. David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In Bo-Yin Yang, editor, *PQCrypto*, volume 7071 of *Lecture Notes in Computer Science*, pages 19–34. Springer, 2011.
13. David Jao and Vladimir Soukharev. A subexponential algorithm for evaluating large degree isogenies. In *Algorithmic number theory*, volume 6197 of *Lecture Notes in Comput. Sci.*, pages 219–233. Springer, Berlin, 2010.
14. Brian LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger security of authenticated key exchange. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *Provable Security: First International Conference*, pages 1–16, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.
15. Jason LeGrow. Post-quantum security of authenticated key establishment protocols. Master's thesis, University of Waterloo, 2016.
16. Philip MacKenzie. The pak suite: Protocols for password-authenticated key exchange. In *IEEE P1363.2*, 2002.
17. Openssl project. http://www.openssl.org/.
18. Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997. Preliminary version in FOCS '94. `arXiv:quant-ph/9508027v2`.
19. Joseph H. Silverman. *The arithmetic of elliptic curves*, volume 106 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1992. Corrected reprint of the 1986 original.
20. Anton Stolbunov. Constructing public-key cryptographic schemes based on class group action on a set of isogenous elliptic curves. *Adv. Math. Commun.*, 4(2):215–235, 2010.
21. Jacques Vélu. Isogénies entre courbes elliptiques. *C. R. Acad. Sci. Paris Sér. A-B*, 273:A238–A241, 1971.
22. Jiang Zhang and Yu Yu. Two-round pake from approximate sph and instantiations from lattices. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 37–67, Cham, 2017. Springer International Publishing.