Practical Supersingular Isogeny Group Key Agreement

Reza Azarderakhsh[1], Amir Jalali[1], David Jao[2], and Vladimir Soukharev[3]

[1] Department of Computer and Electrical Engineering and Computer Science,
Florida Atlantic University, FL, USA,
{ajalali2016, razarderakhsh}@fau.edu
[2] Department of Combinatorics and Optimization, University of Waterloo, Waterloo,
ON, Canada,
djao@uwaterloo.ca
[3] InfoSec Global, Toronto, ON, Canada,
Vladimir.Soukharev@infosecglobal.com

Abstract. We present the first quantum-resistant n-party key agreement scheme based on supersingular elliptic curve isogenies. We show that the scheme is secure against quantum adversaries, by providing a security reduction to an intractable isogeny problem. We describe the communication and computational steps required for n parties to establish a common shared secret key. Our scheme is the first non-generic quantum-resistant group key agreement protocol, and is more efficient than generic protocols, with near-optimal communication overhead. In addition, our scheme is contributory, which in some settings is a desirable security property: each party applies a function of their own private key to every further transmission. We implement the proposed protocol in portable C for the special case where three parties establish a shared secret. Moreover, we benchmark our software on two generations of Intel processors, highlighting the feasibility and efficiency of using the proposed scheme in practical settings. The proposed software computes the entire group key agreement in 994 and 1,374 millions of clock cycles on Intel Core i7-6500 Skylake and Core i7-2609 Sandy Bridge processors, respectively.

Keywords: Group key agreement, isogenies, post-quantum cryptography.

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 [6, 9], uses isogenies between supersingular elliptic curves to construct quantum-resistant cryptographic protocols for public-key encryption, two-party key agreement. One of the still-missing components is a multi-party key agreement scheme.

In this paper, we present a new protocol for quantum-resistant n-party key agreement based on the difficulty of computing isogenies between supersingular elliptic curves. We provide a construction which takes a two-party Diffie-Hellman style key agreement and transforms it into a n-party key agreement. The procedure is sequential from party to party and ensures that each party, before sending their next message in the sequence, applies their own private information. This approach provides authentication capability at no extra cost and has minimal transmission overhead. We present a high-speed implementation of ephemeral three-party key agreement software which provides 80 bits of quantum security and 120 bits of classical security. This implementation provides fast and compact functions to compute isogeny maps on supersingular Montgomery curves, taking advantage of their efficient x-coordinate arithmetic. The first version of our software is written in C and it is portable on various platforms. The benchmarks on the two generations of x64 Intel processors with and without extended arithmetic support, show that the whole protocol including key generations takes reasonable amount of time to compute the shared secret among three parties without using any assembly optimizations in finite field arithmetic. Moreover, because of the special design of the proposed protocol, the shared secret key is established after only four passes.

2 Comparison and related work

Group key exchange is to a large extent an unsolved problem even in the classical setting, involving some amount of compromise between functionality and performance. The only known practical one-round multiparty non-interactive key exchange (NIKE) protocol is the pairing-based protocol of Joux [11], which only handles the 3-party case. This protocol generalizes in an obvious way to n parties using multilinear maps, but currently known candidate multilinear map constructions are, to put it mildly, far from practical. Aside from multilinear maps, there are to our knowledge no existing proposals for quantum-resistant group key exchange protocols. Therefore, even relatively inefficient protocols serve to advance the state of the art in the post-quantum setting.

In the classical (non-quantum-safe) setting, there are a number of different approaches to transform two-party key agreement into multi-party key agreement. Most such constructions transform DH and ECDH protocols into multi-party key agreement protocols. Some examples of possible approaches are [1, 2, 7, 10, 12–14, 16], though some of them require a trusted third party (TTP). The performance of the resulting schemes can be roughly estimated to a first approximation by counting the number of messaging rounds and transmissions per round. To obtain an overall value, we multiply these two parameters. Let n be the number of the parties establishing the common key. For our proposed scheme, we have $2n-2$ rounds with only 1 transmission per round, for an overall value of $2n-2$, which is close to that of the (classical) optimal multi-party key agreement scheme, namely $n-1$. Many proposed schemes, which extend DH and ECDH, have a value of $n(n-1)$. From this viewpoint, our scheme is near optimal, and moreover the number of transmissions per round is constant and equal to 1, which may be advantageous in certain situations.

Contrary to some other published approaches, our proposed scheme is *contributory*: in every round, the transmitting party leaves traces on every parameter which they transmit further to the next party. No parts of the transmitted message are simply re-transmitted; the current party applies a function of their private key to the message parameters and transmits the results. This feature prevents many possible man-in-the-middle attacks on the scheme. In addition, our proposed approach does not require a TTP, which is an important property in some settings. However, our scheme does not support dynamic group membership.

3 Isogeny-based key agreement

3.1 Key Agreement

In PQCrypto 2011, Jao and De Feo [6, 9] 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 group key exchange protocol. Fix a prime p of the form $\ell_A^a \ell_B^b \cdot f \pm 1$ where ℓ_A and ℓ_B are small primes, a and 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.

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.

4 From Two-Party to Three-Party Key Agreement

For a simple, but nontrivial example that illustrates the main idea, we work out the 3-party case. Similar to two-party key exchange, we define a set of global parameters which are agreed among three parties prior to key agreement protocol.

4.1 Global Parameters

We select a prime of the form:

$$p = \ell_A^{e_A} \cdot \ell_B^{e_B} \cdot \ell_C^{e_C} \cdot f \pm 1,$$

where ℓ_A, ℓ_B, and ℓ_C are small primes and f is a co-factor.

Let E be a supersingular elliptic curve defined over $\mathbb{F}_{p^2}$ (finite field of size p^2).

We define the following torsion groups and the corresponding generators:

$$E[\ell_A^{e_A}] = \langle P_A, Q_A \rangle, E[\ell_B^{e_B}] = \langle P_B, Q_B \rangle, E[\ell_C^{e_C}] = \langle P_C, Q_C \rangle.$$

4.2 Key Generation

Each party generates two scalars as their private key and computes the corresponding isogeny kernel. The resulting curve and the image of other parties' bases points on that curve is the public key. We illustrate the whole three-party key agreement protocol in Fig. 1. The isogeny computations by each party is separated.
First, A, B, and C generate their secret-key and public-key. The secret-key is an integer randomly generated from each party subgroup order, i.e. $m, n \in \mathbb{Z}_{\ell^e}$.

4.3 Key Agreement

To establish a common shared secret key, the three parties could naively perform the two-party key exchange protocol with each other, and as a result of this first round, establish three secret keys. Then, they will need three more passes to finally establish a common key. This approach would take nine passes in total. Here, we present our approach, described in previous section, that takes only four passes.

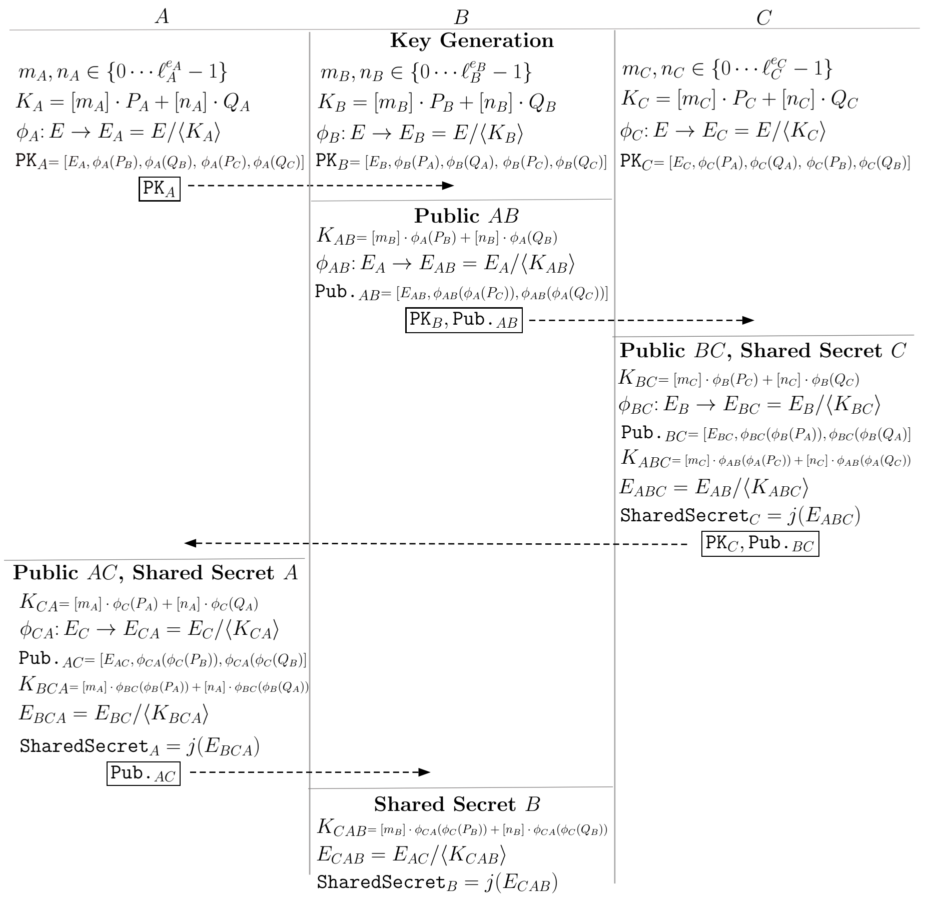
A
B
C
Key Generation
$m_A, n_A \in \{0 \cdots \ell_A^{e_A} - 1\}$
$K_A = [m_A] \cdot P_A + [n_A] \cdot Q_A$
$\phi_A \colon E \to E_A = E/\langle K_A \rangle$
$\mathtt{PK}_A = [E_A, \phi_A(P_B), \phi_A(Q_B), \phi_A(P_C), \phi_A(Q_C)]$
$m_B, n_B \in \{0 \cdots \ell_B^{e_B} - 1\}$
$K_B = [m_B] \cdot P_B + [n_B] \cdot Q_B$
$\phi_B \colon E \to E_B = E/\langle K_B \rangle$
$\mathtt{PK}_B = [E_B, \phi_B(P_A), \phi_B(Q_A), \phi_B(P_C), \phi_B(Q_C)]$
$m_C, n_C \in \{0 \cdots \ell_C^{e_C} - 1\}$
$K_C = [m_C] \cdot P_C + [n_C] \cdot Q_C$
$\phi_C \colon E \to E_C = E/\langle K_C \rangle$
$\mathtt{PK}_C = [E_C, \phi_C(P_A), \phi_C(Q_A), \phi_C(P_B), \phi_C(Q_B)]$
$\boxed{\mathtt{PK}_A}$
Public AB
$K_{AB} = [m_B] \cdot \phi_A(P_B) + [n_B] \cdot \phi_A(Q_B)$
$\phi_{AB} \colon E_A \to E_{AB} = E_A/\langle K_{AB} \rangle$
$\mathtt{Pub.}_{AB} = [E_{AB}, \phi_{AB}(\phi_A(P_C)), \phi_{AB}(\phi_A(Q_C))]$
$\boxed{\mathtt{PK}_B, \mathtt{Pub.}_{AB}}$
Public BC, Shared Secret C
$K_{BC} = [m_C] \cdot \phi_B(P_C) + [n_C] \cdot \phi_B(Q_C)$
$\phi_{BC} \colon E_B \to E_{BC} = E_B/\langle K_{BC} \rangle$
$\mathtt{Pub.}_{BC} = [E_{BC}, \phi_{BC}(\phi_B(P_A)), \phi_{BC}(\phi_B(Q_A))]$
$K_{ABC} = [m_C] \cdot \phi_{AB}(\phi_A(P_C)) + [n_C] \cdot \phi_{AB}(\phi_A(Q_C))$
$E_{ABC} = E_{AB}/\langle K_{ABC} \rangle$
$\mathtt{SharedSecret}_C = j(E_{ABC})$
$\boxed{\mathtt{PK}_C, \mathtt{Pub.}_{BC}}$
Public AC, Shared Secret A
$K_{CA} = [m_A] \cdot \phi_C(P_A) + [n_A] \cdot \phi_C(Q_A)$
$\phi_{CA} \colon E_C \to E_{CA} = E_C/\langle K_{CA} \rangle$
$\mathtt{Pub.}_{AC} = [E_{AC}, \phi_{CA}(\phi_C(P_B)), \phi_{CA}(\phi_C(Q_B))]$
$K_{BCA} = [m_A] \cdot \phi_{BC}(\phi_B(P_A)) + [n_A] \cdot \phi_{BC}(\phi_B(Q_A))$
$E_{BCA} = E_{BC}/\langle K_{BCA} \rangle$
$\mathtt{SharedSecret}_A = j(E_{BCA})$
$\boxed{\mathtt{Pub.}_{AC}}$
Shared Secret B
$K_{CAB} = [m_B] \cdot \phi_{CA}(\phi_C(P_B)) + [n_B] \cdot \phi_{CA}(\phi_C(Q_B))$
$E_{CAB} = E_{AC}/\langle K_{CAB} \rangle$
$\mathtt{SharedSecret}_B = j(E_{CAB})$

Fig. 1: Three-party group key agreement diagram

1. **A to B (pass one).** A sends to B its public-key which contains E_A and the image of P_B, Q_B, P_C, and Q_C on E_A. B, upon receiving data from A, computes the shared public-key $\mathtt{Pub.}_{AB}$ by computing the curve E_{AB} and the image of P_C and Q_C on E_{AB}.
2. **B to C (pass two).** After computing, B sends its public-key and the computed $\mathtt{Pub.}_{AB}$ to C. C, upon receiving data from B, C can compute the shared secret-key and the $\mathtt{Pub.}_{BC}$ using B's public key.
3. **C to A (pass three).** After computing $\mathtt{Pub.}_{BC}$, C sends its public-key and the generated $\mathtt{Pub.}_{BC}$ to A. A, upon receiving data from C, computes the shared-secret key and $\mathtt{Pub.}_{AC}$ to pass to B for the final round of the protocol.
4. **A to B (pass four)**. After computing $\mathtt{Pub.}_{AC}$, A sends only the generated $\mathtt{Pub.}_{AC}$ to B. Upon receiving the shared data between A and C (second round) from A, B can compute the shared secret-key.

The shared key is $j(E_{ABC})$. Note that all resulting curves, E_{ABC}, E_{BCA}, and E_{CAB} are isomorphic to $E/\langle K_A, K_B, K_C\rangle$ and hence have the same j-invariant. Observe that the above scheme has only four passes, namely, A to B, B to C, C to A, and A to B (second time).

5 Proposed n-party Key Agreement Protocol

Let n be the number of parties, where $n \geq 2$. Let $U_1, U_2, \ldots, U_n$ be any n parties.

5.1 Protocol Parameters

Let p be a prime of the form $\ell_1^{e_1}\ell_2^{e_2}\cdots\ell_n^{e_n}\cdot f \pm 1$, where ℓ_i's are small distinct primes. Fix a supersingular curve $E\colon y^2 = x^3 + x$ with j-invariant 1728 over $\mathbb{F}_{p^2}$ together with bases $\{P_1, Q_1\}, \{P_2, Q_2\}, \ldots, \{P_n, Q_n\}$ of $E[\ell_1^{e_1}], E[\ell_2^{e_2}], \ldots, E[\ell_n^{e_n}]$, respectively.

5.2 Key Generation

A party U_i generates two secret random integers $m_i, n_i \in \mathbb{Z}/\ell_i^{e_i}\mathbb{Z}$, not both divisible by ℓ_i, obtains $K_i = [m_i]P_i + [n_i]Q_i$ and computes $E_i = E/\langle K_i\rangle$. Let ϕ_i be the isogeny from E to E_i. Then, U_i computes

$$\begin{aligned}&\phi_i(P_1), \phi_i(Q_1), \phi_i(P_2), \phi_i(Q_2),\\ &\qquad \ldots, \phi_i(P_{i-1}), \phi_i(Q_{i-1}), \phi_i(P_{i+1}), \phi_i(Q_{i+1}), \ldots,\\ &\qquad\qquad \phi_i(P_n), \phi_i(Q_n).\end{aligned}$$

Thus, for each party U_i:

- Private key: $m_i, n_i \in \mathbb{Z}/\ell_i^{e_i}\mathbb{Z}$.
- Public key: E_i and

$$\begin{aligned}&\phi_i(P_1), \phi_i(Q_1), \phi_i(P_2), \phi_i(Q_2),\\ &\qquad \ldots, \phi_i(P_{i-1}), \phi_i(Q_{i-1}), \phi_i(P_{i+1}), \phi_i(Q_{i+1}), \ldots,\\ &\qquad\qquad \phi_i(P_n), \phi_i(Q_n) \in E_i.\end{aligned}$$

5.3 Protocol

Suppose, parties $U_1, U_2, \ldots, U_n$ wish to establish a common secret key. This will involve passes and calculations by each party.

There will be two rounds of passes. We begin with the first round. For $i \in \{2, \ldots, n\}$, the following passes and computations take place (sequentially). Define $N = \{1, 2, \ldots, n\}$. Let $P_{i,w}$ the value of the point P_i, evaluated by all isogenies, sequentially, mapping from curve E to E_w. $Q_{i,w}$ is defined similarly for the point Q_i.

$U_{(i-1)}$ sends to U_i the following:

- $E_{i-1}, E_{(i-2)(i-1)}, \ldots, E_{12\cdots(i-1)}$
- $\phi_{(i-1)}(P_j), \phi_{(i-1)}(Q_j), \forall j \in N \setminus \{i-1\}$
- $P_{j,(i-2)(i-1)}, Q_{j,(i-2)(i-1)}, \forall j \in N \setminus \{i-2, i-1\}$

  $\vdots$
- $P_{j,12\cdots(i-1)}, Q_{j,12\cdots(i-1)}, \forall j \in N \setminus \{1, \ldots, i-2, i-1\}$

U_i computes the following:

- $\phi_{(i-1)i} \colon E_{(i-1)} \to E_{(i-1)i} = E_{(i-1)}/\langle [m_i]\phi_{(i-1)}(P_i) + [n_i]\phi_{(i-1)}(Q_i)\rangle$
- $\phi_{(i-2)(i-1)i} \colon E_{(i-2)(i-1)} \to E_{(i-2)(i-1)i} = E_{(i-2)(i-1)}/\langle [m_i]P_{i,(i-2)(i-1)} + [n_i]Q_{i,(i-2)(i-1)}\rangle$

  $\vdots$
- $\phi_{12\cdots(i-1)i} \colon E_{12\cdots(i-1)} \to E_{12\cdots(i-1)i} = E_{12\cdots(i-1)}/\langle [m_i]P_{i,12\cdots(i-1)} + [n_i]P_{i,12\cdots(i-1)}\rangle$
- $P_{j,(i-1)i}, Q_{j,(i-1)i}, \forall j \in N \setminus \{i-2, i-1, i\}$
- $P_{j,(i-2)(i-1)i}, Q_{j,(i-2)(i-1)i}, \forall j \in N \setminus \{i-2, i-1, i\}$

  $\vdots$
- $P_{j,12\cdots(i-1)i}, Q_{j,12\cdots(i-1)i}, \forall j \in N \setminus \{1, \ldots, i-1, i\}$

This process continues until U_{n-1} sends the data to U_n. U_n performs a standard computation and then uses the curve $E_{12\cdots(n-1)n}$ to compute its j-invariant, which becomes the common shared secret key. Then U_n sends to U_1 all the curves, except $E_{12\cdots(n-1)n}$ and the corresponding images of bases points.

The next round of passes begins. For $i \in \{1, \ldots, n-2\}$, the following passes and computations take place (sequentially).

U_i computes the following:

- $\phi_{(i+1)(i+2)\cdots n12\cdots(i-1)i} \colon$
  $E_{(i+1)(i+2)\cdots n12\cdots(i-1)} \to E_{(i+1)(i+2)\cdots n12\cdots(i-1)i} =$
  $E_{(i+1)(i+2)\cdots n12\cdots(i-1)}/\langle [m_i]P_{i,(i+1)(i+2)\cdots n12\cdots(i-1)} + [n_i]Q_{i,(i+1)(i+2)\cdots n12\cdots(i-1)}\rangle$
- $\phi_{(i+2)\cdots n12\cdots(i-1)i} \colon$
  $E_{(i+2)\cdots n12\cdots(i-1)} \to E_{(i+2)\cdots n12\cdots(i-1)i} =$
  $E_{(i+2)\cdots n12\cdots(i-1)}/\langle [m_i]P_{i,(i+2)\cdots n12\cdots(i-1)} + [n_i]Q_{i,(i+2)\cdots n12\cdots(i-1)}\rangle$

  $\vdots$
- $\phi_{n12\cdots(i-1)i} \colon$
  $E_{n12\cdots(i-1)} \to E_{n12\cdots(i-1)i} =$
  $E_{n12\cdots(i-1)}/\langle [m_i]P_{i,n12\cdots(i-1)} + [n_i]Q_{i,n12\cdots(i-1)}\rangle$
- $P_{j,(i+2)\cdots n12\cdots(i-1)i}, Q_{j,(i+2)\cdots n12\cdots(i-1)i}$ for $j \in \{i+1\}$
- $P_{j,(i+3)\cdots n12\cdots(i-1)i}, Q_{j,(i+3)\cdots n12\cdots(i-1)i}$ for $j \in \{i+1, i+2\}$

  $\vdots$
- $P_{j,n12\cdots(i-1)i}, Q_{j,n12\cdots(i-1)i}$ for $j \in \{i+1, i+2, \ldots, n-1\}$

Then U_i computes the j-invariant of $E_{(i+1)(i+2)\cdots n12\cdots(i-1)i}$, which is the common shared secret key.

U_i sends to U_{i+1} the following:

- $E_{(i+2)\cdots n12\cdots(i-1)i}$
- $E_{(i+3)\cdots n12\cdots(i-1)i}$

  $\vdots$
- $E_{n12\cdots(i-1)i}$
- $P_{j,(i+2)\cdots n12\cdots(i-1)i}, Q_{j,(i+2)\cdots n12\cdots(i-1)i}$ for $j \in \{i+1\}$
- $P_{j,(i+3)\cdots n12\cdots(i-1)i}, Q_{j,(i+3)\cdots n12\cdots(i-1)i}$ for $j \in \{i+1, i+2\}$

  $\vdots$
- $P_{j,n12\cdots(i-1)i}, Q_{j,n12\cdots(i-1)i}$ for $j \in \{i+1, i+2, \ldots, n-1\}$

This continues until the final pass from U_{n-2} to U_{n-1}. Finally U_{n-1} computes $E_{n12\cdots(n-1)} = E_{n12\cdots(n-2)}/\langle[m_{n-1}]P_{n-1,n12\cdots(n-2)}+[n_{(n-1)}]Q_{n-1,n12\cdots(n-2)}\rangle$ and its j-invariant, which is the common shared secret key.

At this point, all n parties $U_1, U_2, \ldots, U_n$ possess the common shared secret key, which they can use now for their group communication purposes.

Figure 2 summarizes which curves generated by each respective user and then all of the the, except the ones in red, are passed on to the the next user.

5.4 Communication Overhead

Each party performs 2 passes, except for U_{n-1} and U_n, who perform one pass. The total number of passes is $2n-2$. In the first $n-1$ passes, the communication overhead of the pass from party U_i to U_{i+1} includes i elliptic curves and $i(2n-1-i)/2$ corresponding pairs of points. For the next pass from U_n to U_1, there are $n-1$ elliptic curves and $n(n-1)/2$ pairs of points. For the last $n-2$ passes, the pass from party U_i to U_{i+1} includes $n-1-i$ elliptic curves and $(n-1-i)(n-i)/2$ pairs of points.

Overall, each party sends $n-1$ elliptic curves and $n(n-1)/2$ pairs of elliptic curve points. In general, without compression or optimization, it takes two finite field elements to represent the curve and one finite field element plus one bit to represent the point. If p is a k-bit prime, then each party transfers:

$$(n-1) \cdot 2 \cdot (2k) + (n(n-1)/2) \cdot (2k+1) \text{ bits.}$$

A time-space trade-off is possible: For any collection of points lying on the same curve and having orders which are pairwise relatively prime, we can transmit this entire collection by sending only a single point on the curve, using the projection onto the elliptic curve of the standard Chinese Remainder Theorem isomorphism $\mathbb{Z}/a \times \mathbb{Z}/b \cong \mathbb{Z}/ab$, valid for $\gcd(a, b) = 1$. The trade-off is that performing this conversion each way takes time. In our software we do not perform this conversion.

5.5 Protocol Optimization

The group key protocol which is explained above is an instance of "book" description of the protocol following the notation of the original Diffie-Hellman

$$\begin{tikzcd}
U_1 & & & & E_1 \arrow[dr] & & & \\
U_2 & & & E_2 \arrow[dr] & & E_{12} \arrow[dr] & & \\
U_3 \arrow[d, dotted, no head] & & E_3 \arrow[dr, dotted] & & E_{23} \arrow[dr, dotted] & & E_{123} \arrow[dr, dotted] & \\
U_n & E_n \arrow[dr] & & E_{(n-1)n} \arrow[dr] & \cdots\cdots & E_{23\cdots n} \arrow[dr] & & {\color{red}E_{12\cdots n}} \\
U_1 \arrow[d, dotted, no head] & & E_{n1} \arrow[dr, dotted] & & E_{(n-1)n1} \arrow[dr, dotted] & \cdots\cdots & {\color{red}E_{23\cdots n1}} & \\
U_{n-2} & & & E_{n12\cdots(n-2)} \arrow[dr] & & {\color{red}E_{(n-1)n1\cdots(n-2)}} & & \\
U_{n-1} & & & & {\color{red}E_{n1\cdots(n-1)}} & & &
\end{tikzcd}$$

Fig. 2: Protocol Diagram

key exchange protocol by Jao and De Feo [9] which can be adopted on generic form of elliptic curves. However, we can reduce the size of the overhead and computations by applying optimization techniques similar to the ones in [5]. We briefly describe these optimization techniques in the following, and further we adopt them in the practical implementation of our proposed protocol in section 8.

5.6 Projective points and curve coefficient.

Following [5], and setting up the base curve as an instance of a supersingular Montgomery curve, we can perform all the isogeny computations in projective coordinates, taking advantage of fast arithmetic associated with Kummer varieties of Montgomery curves. Therefore, all the points are represented by x-coordinate in $\mathbb{P}^1$, and all the field inversions are eliminated except for the very last step of each party computations. Using this technique, we can compute the field inversion using *constant-time* algorithms such as FLT, providing a fast and constant-time implementation of the protocol. We adopt the same technique inside our software, More details are given in section 8.

5.7 Communication overhead.

For each pair of elliptic curve points (P, Q) can be represented by three x-coordinates of P, Q, and $R = P - Q$. Using this representation, we do not need extra information for the curve coefficient as long as we are working on the Montgomery curves, and we can retrieve the curve coefficient A using the x-coordinates of these three points anytime inside the protocol computations as it is described in details in section 8.2. In that case, we get the following overhead size:

$$(n(n-1)/2) \cdot (6k) \text{ bits},$$

where k is the prime bit-length. Note that, on average, each pass is half the size of what is stated above.

6 Complexity Assumptions

The complexity assumptions for this scheme are the same as for the original scheme by Jao, De Feo and Plût [6]. We list them here.

As before, let p be a prime of the form $\ell_1^{e_1}\ell_2^{e_2}\cdots\ell_n^{e_n}\cdot f \pm 1$, and fix a supersingular curve E over $\mathbb{F}_{p^2}$ together with bases $\{P_1, Q_1\}, \{P_2, Q_2\}, \ldots, \{P_n, Q_n\}$ of $E[\ell_1^{e_1}], E[\ell_2^{e_2}], \ldots, E[\ell_n^{e_n}]$, respectively.

Originally these assumptions were stated for primes p constructed from two primes ℓ_1 and ℓ_2. However, the general case can actually be viewed as a special case of the two-prime case, since the extra primes can be incorporated into the cofactor f.

6.1 Decisional Supersingular Isogeny (DSSI) problem.

Let E_i be another supersingular curve defined over $\mathbb{F}_{p^2}$. Decide whether E_i is $\ell_i^{e_i}$-isogenous to E.

Computational Supersingular Isogeny (CSSI) problem Let the map $\phi_i \colon E \to E_i$ be an isogeny whose kernel is $\langle [m_i]P_i + [n_i]Q_i\rangle$, where m_i and n_i are chosen at random from $\mathbb{Z}/\ell_i^{e_i}\mathbb{Z}$ and not both divisible by ℓ_i. Given E_i and the values $\phi_i(P_j)$, $\phi_i(Q_j)$, for $j \neq i$, find a generator R_i of $\langle [m_i]P_i + [n_i]Q_i\rangle$.

Supersingular Computational Diffie-Hellman (SSCDH) problem Let $\phi_i \colon E \to E_i$ be an isogeny whose kernel is equal to $\langle [m_i]P_i + [n_i]Q_i\rangle$, and let $\phi_j \colon E \to E_j$ $(j \neq i)$ be an isogeny whose kernel is $\langle [m_j]P_j + [n_j]Q_j\rangle$, where m_i, n_i (respectively m_j, n_j) are chosen at random from $\mathbb{Z}/\ell_i^{e_i}\mathbb{Z}$ (respectively $\mathbb{Z}/\ell_j^{e_j}\mathbb{Z}$) and not both divisible by ℓ_i (respectively ℓ_j). Given the curves E_i, E_j and the points $\phi_i(P_j)$, $\phi_i(Q_j)$, $\phi_j(P_i)$, $\phi_j(Q_i)$, find the j-invariant of $E/\langle [m_i]P_i + [n_i]Q_i, [m_j]P_j + [n_j]Q_j\rangle$.

Supersingular Decision Diffie-Hellman (SSDDH) problem Given a tuple sampled with probability 1/2 from one of the following two distributions:

- $(E_i, \phi_i(P_j), \phi_i(Q_j), E_j, \phi_j(P_i), \phi_j(Q_i), E_{ij})$, wherein the quantities E_i, $\phi_i(P_j)$, $\phi_i(Q_j)$, E_j, $\phi_j(P_i)$, and $\phi_j(Q_i)$ are as in the SSCDH problem and

$$E_{ij} \cong E/\langle [m_i]P_i + [n_i]Q_i, [m_j]P_j + [n_j]Q_j \rangle,$$

- $(E_i, \phi_i(P_j), \phi_i(Q_j), E_j, \phi_j(P_i), \phi_j(Q_i), E_C)$, wherein the quantities E_i, $\phi_i(P_j)$, $\phi_i(Q_j)$, E_j, $\phi_j(P_i)$, and $\phi_j(Q_i)$ are as in the SSCDH problem and

$$E_C \cong E/\langle [m_i']P_i + [n_i']Q_i, [m_j']P_j + [n_j']Q_j \rangle,$$

  where m_i', n_i' (respectively m_j', n_j') are chosen at random from $\mathbb{Z}/\ell_i^{e_i}\mathbb{Z}$ (respectively $\mathbb{Z}/\ell_j^{e_j}\mathbb{Z}$) and not both divisible by ℓ_i (respectively ℓ_j),

determine from which distribution the tuple is sampled.

7 Security

The presented protocol, just like the original two-party key agreement, is session-key secure in the authenticated-links adversarial model similar to Canetti and Krawczyk [3] model for two-party key agreement under the assumption that SSDDH is hard.

The proof of Theorem 5.4 from [9] can be easily adapted to prove security of the protocol presented in this paper. We provide an outline for the proof.

It is obvious that, if all the parties are uncorrupted, they will produce the same key as the output. We are going to prove that the advantage of an adversary making the right guess is negligible. Assume the contrary, that it is not negligible and that we have an oracle which solves that with an advantage ϵ. The adversary can take control and pretend to be all the parties except the target two parties U_i and U_j, which communicate, meaning that $i = j \pm 1$. Without loss of generality, assume $i = j - 1$. The goal is to determine whether the observed curve E_{ij} is a valid curve produced by U_i and U_j, or is it a random curve. For all the parties under control of the adversary, the values of the private scalars can be all set to zero and 1 (or any other value coprime to the order Q_i's). This makes the final expected output to be of the form $E_{12\cdots ij\cdots n}$. Adversary runs the protocol with the two parties U_i and U_j and observes their auxiliary points and has the final curve. He runs the oracle to check if the final curve is valid. If the final curve is valid, then this means that E_{ij} is valid. Otherwise, it would mean that E_{ij} is invalid. Assume there are k sessions. The adversary can solve this problem using the given oracle with an advantage ϵ, if the session matches. Given that the adversary can run at most k sessions (where k is an upper bound on the number of sessions activated by the adversary in any interaction), we have $1/k$ chance that the session actually matches. Solving this problem now directly yields a solution to SSDDH. Hence we get an advantage of ϵ/k, which is non-negligible.

It should be noted that proposed parameters become heavily unbalanced when the number of parties grows which makes our proposed scheme vulnerable to a passive key recovery attack by Petit in [15]. Based on the discussion on that paper, the attack against the unbalanced version of SIDH is applicable as soon as n exceeds a certain bound. Heuristically, the attack becomes applicable when n exceeds the bit length of ℓ^e, which is three times the desired security level. Therefore for a typical size of ℓ^e ($\sim 3 \cdot 128$ bits) our scheme becomes insecure as soon as about 400 parties run the protocol. This number is large enough that it should not affect most applications.

7.1 Parameter Sizes

We need to state parameter sizes to provide a given security level. Assume that we require an s-bit quantum security level. The best know quantum attack is an exponential time complexity attack of cube root of the degree of the isogeny (i.e. cube root of $\ell_i^{e_i}$). Hence, for each party i the size of $\ell_i^{e_i}$ must be $3s$ bits. In case of a key agreement for n parties, the size of the prime p needs to be $3ns$.

8 Supersingular Isogeny Group Key Performance

In this section, we present the implementation detail of the supersingular isogeny group key agreement protocol in the case of three parties. The parties compute different large-degree isogenies over different torsion subgroups of a supersingular elliptic curve to efficiently establish a shared secret key. Our software is publicly available[4] for evaluation and reproducibility.

8.1 Implementation Summary

Public parameters The finite field is constructed over a special form 747-bit prime

$$p_{747} = 2^{260} \cdot 3^{153} \cdot 5^{105} - 1,$$

which provides three different torsion subgroups, i.e. $E[2^{260}]$, $E[3^{153}]$, and $E[5^{105}]$ for isogeny computations. Moreover, the subgroup orders $2^{260} \approx 3^{153} \approx 5^{105}$ are almost equal for achieving a balance security among parties. Note that since the group key protocol is established by three parties, the overall security level of the protocol for the same size finite field is less than two parties Diffie-Hellman like key exchange. Therefore, the proposed prime provides 80-bit quantum and 120-bit classical security level, respectively.

The special form of the proposed prime helps us to exploit the customized Montgomery reduction algorithm which is discussed in details in [5]. Similar to other optimized implementation of supersingular isogeny-based protocols, we set the starting curve E as the special instance of supersingular Montgomery curve

[4] `https://github.com/amirjalali65/PQCisogenyGroupKey`

$E/\mathbb{F}_{p^2} : y^2 = x^3 + x$ with $\#E(\mathbb{F}_{p^2}) = (2^{260} \cdot 3^{153} \cdot 5^{105})^2$ and j-invariant equal to 1728.

Three pairs of base points are defined corresponding to each torsion subgroups such that $\{P_2, Q_2, R_2\} \in E[2^{260}]$, $\{P_3, Q_3, R_3\} \in E[3^{153}]$, and $\{P_5, Q_5, R_5\} \in E[5^{105}]$, where we use an auxiliary point $R_i = Q_i - P_i, i \in \{2, 3, 5\}$ to encode the base points following [8].

Key generation Each party randomly generates a secret key $\mathcal{K} \in \{0 \cdots \ell^e - 1\}$ from the key-space corresponding to their subgroup order and computes the isogeny kernel $R = P + [\mathcal{K}]Q$. This kernel point is further used to compute isogenies using optimal strategy. Furthermore, each party needs to compute the image of the base points which are belong to the torsion subgroups of other parties, using their secret isogeny. Since our software computes the key agreement among three parties, we denote these parties as A, B, and C and the corresponding public keys are defined as:

$$\begin{aligned}
\texttt{PK}_A = [&\phi_A(x_{P_B}), \phi_A(x_{Q_B}),\\
&\phi_A(x_{R_B}), \phi_A(x_{P_C}), \phi_A(x_{Q_C}), \phi_A(x_{R_C})] \in \mathbb{F}_{p^2}^6,\\
\texttt{PK}_B = [&\phi_B(x_{P_A}), \phi_B(x_{Q_A}),\\
&\phi_B(x_{R_A}), \phi_B(x_{P_C}), \phi_B(x_{Q_C}), \phi_B(x_{R_C})] \in \mathbb{F}_{p^2}^6,\\
\texttt{PK}_C = [&\phi_C(x_{P_A}), \phi_C(x_{Q_A}),\\
&\phi_C(x_{R_A}), \phi_C(x_{P_B}), \phi_C(x_{Q_B}), \phi_C(x_{R_B})] \in \mathbb{F}_{p^2}^6.
\end{aligned}$$

We state that encoding the base points using the auxiliary point R is very useful for computing 5-degree isogenies which is explained in more details in the following section.

Shared public keys Communication parties also require to generate a shared public key with other communication parties and pass it to their neighbor as it was discussed in details in section 4.

These shared public keys are generated based on other party's public key as follows:

$$\begin{aligned}
\texttt{Pub.}_{AB} &= [\phi_B(\phi_A(x_{P_C})), \phi_B(\phi_A(x_{Q_C})), \phi_B(\phi_A(x_{R_C}))] \in \mathbb{F}_{p^2}^3,\\
\texttt{Pub.}_{BC} &= [\phi_C(\phi_B(x_{P_A})), \phi_C(\phi_B(x_{Q_A})), \phi_C(\phi_B(x_{R_A}))] \in \mathbb{F}_{p^2}^3,\\
\texttt{Pub.}_{AC} &= [\phi_A(\phi_C(x_{P_B})), \phi_A(\phi_C(x_{Q_B})), \phi_A(\phi_C(x_{R_B}))] \in \mathbb{F}_{p^2}^3.
\end{aligned}$$

Shared secret key. Upon receiving the shared public keys, each party performs the final isogey computations using the private kernels to reach the final curve. Finally, the common j-invariant is regarded as the group shared secret. In our software this key for each party is computed as follows:

$$\begin{aligned}
\texttt{SharedSK}_A &= j(E_{ABC}) \in \mathbb{F}_{p^2},\\
\texttt{SharedSK}_B &= j(E_{BAC}) \in \mathbb{F}_{p^2},\\
\texttt{SharedSK}_C &= j(E_{CAB}) \in \mathbb{F}_{p^2},
\end{aligned}$$

where $j(E_{ABC}) = j(E_{BAC}) = j(E_{CAB})$. In the following section, we describe the implementation detail of computing isogenies and curve coefficients for each party.

8.2 Isogenies and Curve Coefficients Computations

In order to implement three-party group key agreement we need to have three different torsion subgroups from which each communication party can construct the secret isogeny map. In this work, we chose to work with isogenies of degree 3, 4, and 5 due to their efficient and compact formulas. For the isogeny map and evaluation of degree 3 and 4, we followed the same implementation from SIDH/SIKE library[5], while we developed a new set of functions for computing degree 5 isogeny, based on the proposed algorithms in [4]. These formulas are derived from Vélu's formula for computing small degree isogenies, while they are projectivized and customized for a special form of the Montgomery curves in this work. We refer the readers to [6, 5, 4] for further detail.

3-isogeny. Following [5], computing the image of a projective point $P = (X : Z)$ and the projective curve coefficients $(A : C)$ via 3-isogeny map can be efficiently computed as:

$$(A' : C') = (Z_3^4 + 18X_3^2Z_3^2 - 27X_3^4 : 4X_3Z_3^3),$$
$$(X' : Z') = (X(X_3X - Z_3Z)^2 : Z(Z_3X - X_3Z)^2),$$

where $P_3 = (X_3 : Z_3)$ is a kernel point of order 3 on the curve.

4-isogeny. Computing 4-isogeny evaluation and map are expressed in projective coordinates as:

$$(A' : C') = (2(2X_4^4 - Z_4^4) : Z_4^4),$$
$$(X' : Z') = (X(2X_4Z_4Z - X(X_4^2 + Z_4^2))(X_4X - Z_4Z)^2 :$$
$$Z(2X_4Z_4X - Z(X_4^2 + Z_4^2))(Z_4X - X_4Z)^2),$$

where $P_4 = (X_4, Z_4)$ is a projective kernel point of order 4 on the curve [5].

Note that, as the degree of isogeny grows, the number of required arithmetic operations for computing an ℓ-isogeny is increased notably. However, the total number of isogeny walks, i.e., e to compute ℓ^e-isogeny is decreased. For instance, in our proposed prime, 2-isogeny, 3-isogeny, and 5-isogeny are evaluated by 260, 153, and 105 compositions, respectively.

As a result, there is a trade-off between choosing the degree of isogeny which defines a single isogeny computation performance, and the total number of steps to compute the ℓ^e large-degree isogeny. However, from the implementation viewpoint, smaller degree isogenies can be implemented using more compact functions and they are more desirable.

5-isogeny. Computing isogenies with degree larger than 4 require at least two points as the input. In particular, torsion subgroup of 5 on an elliptic curve can be denoted as $E[5] = \{-2P_5, -P_5, \mathcal{O}, P_5, 2P_5\}$ where $P_5 = (X_5, Z_5)$ is a projective point of order 5 on the curve. Based on Vélu's formula [17], we need to push the coordinates of P_5 and $2P_5$ to evaluate the 5-isogeny map. Therefore, before each 5-isogeny computation, the projective coordinates of isogeny kernel $P_5 = (X_5 : Z_5)$ and $2P_5 = (X_5' : Z_5')$ are required to be computed using point doubling formula.

Moreover, we need to develop a quintupling function inside the optimal strategy for isoegny computations to compute the coordinates of $[5]P$. We developed this function

[5] `https://github.com/Microsoft/PQCrypto-SIDH`

based on the Montgomery ladder algorithm using point doubling and differential addition projectively. In our software, `xQNTPL` function implements quintupling of a point based on curve projective coefficients A and C.

Costello et al. [4] proposed a set of projective compact and efficient algorithms for computing arbitrary odd degree isogenies. Following their work, we developed a function for computing the image of a projective point $P = (X : Z)$ via 5-isogeny map. Since the explicit formula for 5-isogeny point evaluation is relatively long, we avoid providing it in this section and we just state that each 5-isogeny map costs 8**M**+2**S**+6**a**[6]. We refer the readers to the implementation of `eval_5_isog` function inside our software for further detail.

For the curve coefficient map, implementation is more complicated. In particular, we note that there are two different ways suggested in [4] for retrieving the isomorphic curve coefficient of an odd isogeny map. The first method computes the curve coefficient a as:

$$a = \frac{(1 - x_P x_Q - x_P x_R - x_Q x_R)^2}{4 x_p x_Q x_R} - x_p - x_Q - x_R, \qquad (1)$$

where P, Q, and $R = Q - P$ are three points on the isomorphic curve. This function requires one expensive inversion, and it is not efficient to use it as it is for 5-isogeny map computations. Alternatively, we can use the projective version of the above formula which costs 8**M**+5**S**+11**a** when the coordinates of P, Q, and R are available. Fortunately, this is the case in group key agreement protocol for the key generation and the shared public key generation procedure. However, for the group secret shared key computations, we have to compute three redundant 5-isogeny evaluations in each step to be able to use this technique. We implemented the projective version of (1), denoted by `get_AC_proj` inside our software. This function computes $(A + 2C : 4C)$ values which are required to pass to the `xQNTPL` function.

The second method of computing the odd isogeny map, as discussed in details in [4], is by evaluating 2-torsion points. The idea is based on the relation between 2-torsion point $P_\alpha = (\alpha, 0)$ which projectively denoted as $P = (X_\alpha : Z_\alpha)$, and the curve coefficient a on the Montgomery curve defined by $E : y^2 = x^3 + ax^2 + x$. In this case, we can retrieve curve coefficients as:

$$(A + 2C : 4C) = ((X_\alpha + Z_\alpha)^2 : (X_\alpha - Z_\alpha)^2 - (X_\alpha + Z_\alpha)^2),$$

using 2**S**+3**a**. However, before this operation, the coordinates of P_α need to evaluated on the isomorphic curve using 5-isogeny evaluation. Therefore the total cost of this technique is 8**M**+4**S**+9**a** which is slightly faster than the first method. We also implemented this technique as `get_AC_alpha` function in our library. Since this function is slightly faster than `get_AC_proj`, it is more efficient to only use this method for 5-isogeny map, however, in our library and considering our parameters set, we have to use both approaches for the following reason.

The 2-torsion technique is only meaningful for odd degree isogeny when the order of 2-torsion points are preserved through isogeny evaluations. Considering our parameter set, party A computes 4-isogeny. Party C requires to receive the image of P_α on E_{AB} to compute the shared secret, i.e. $j(E_{CAB})$ by computing 5-isogeny steps on this curve. However, since A computes even degree 4-isogeny, the order of P_α on E_{AB} is not preserved and we cannot use this technique for computing 5-isogeny maps during the

[6] We use **M**, **S**, and **a** to denote multiplication, squaring, and addition/subtraction in $\mathbb{F}_{p^2}$

Table 1: Cycle counts (presented in millions of clock cycles) for Ephemeral isogeny-based group key agreement. Timing benchmarks were taken on an Intel Core i7-6500 Skylake processor running Ubuntu 16.04 LTS and an Intel Core i7-2609 Sandy Bridge with TurboBoost disabled. Executables are generated by GNU-`gcc` version 5.4.0.

Operation	Skylake	Sandy Bridge
Keygen A	113	168
Keygen B	111	166
Keygen C	149	227
SharedPublic B	97	144
SharedPublic + SharedSecret C	269	400
SharedPublic + SharedSecret A	174	269
SharedSecret B	81	134
Total	**994**	**1,374**

shared secret computations for party C. However, we still can benefit from 2-torsion evaulation for the key generation of C.

Considering the above statements, for achieving more efficiency, we use 2-torsion point technique for 5-isogeny map inside the ephemeral key generation of C, while we evaluate the curve coefficients using `get_AC_proj` for shared public key and secret key computations.

8.3 Implementation Results and Discussion

In this section, we present the performance of our three-party ephemeral group key agreement. This version of our software is a fully portable implementation in C, supporting both 32- and 64-bit processors. We developed both field and quadratic extension field arithmetic operations by modifying the arithmetic functions inside the SIDH/SIKE library, customizing them for our special prime. At the API level, our software provides interfaces for each party and efficiently generates public keys, shared public keys, and shared secret keys. Party B requires separate functions to generate shared public key and shared secret key since this party should wait for the $\texttt{Pub.}_{AC}$, generated by A during the final pass, to be able to compute the group shared secret.

We benchmarked our software on two popular families of Intel processors, an Intel i7-6700 Skylake processor, running Ubuntu 16.04 LTS, and an Intel i7-2609 Sandy Bridge processor running Ubuntu 14.04.5 LTS. Table 1 presents the performance of each group key agreement operation presented in millions of clock cycles. We state that the significant performance difference between the two generation of target processors is not only related to the improvements in micro-architecture design, but it is also because of extended arithmetic support, i.e. `MULX` and `ADDX` operations on Skylake processors.

Based on the optimized results in [5, 8], we expect to achieve significant performance improvement by implementing target-specific finite field arithmetic in AMD assembly language. We leave the design of hardened and more optimized implementations for future work.

9 Conclusion

We present the first isogeny-based multi-party group key agreement scheme that is secure against quantum adversaries. Our construction is based on the two-party key agreement scheme by De Feo, Jao, and Plût [6, 9]. The transformation from 2-party to n-party key agreement is inspired by previous work, but is not identical to any previously published transformation. The resulting scheme has the same security reduction as the original (two-party) key agreement scheme. Our scheme is contributory with an optimal number of transmissions per round and a near-optimal number of rounds. The overhead size is close to optimal, as the isogeny-based schemes are known to have the lowest communication overhead, combining with almost minimal number of passes, which enables us to obtain optimal results.

We implemented a special case three-party key agreement to evaluate the performance of the proposed scheme at 80-bit quantum and 120-bit classical security level. Using a set of fast and compact isogeny evaluation functions, we show that the group shared secret for three communication parties can be generated efficiently in a practical performance. Moreover, based on the previous optimized implementation of isogeny-based Diffie-Hellman key exchange, we expect to obtain further performance enhancement by developing target-specific implementation. We leave the investigation of optimized assembly implementation of our library for future work.

References

1. Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Dynamic group Diffie-Hellman key exchange under standard assumptions. In Lars R. Knudsen, editor, *Advances in Cryptology — EUROCRYPT 2002: International Conference on the Theory and Applications of Cryptographic Techniques Amsterdam, The Netherlands, April 28 – May 2, 2002 Proceedings*, pages 321–336, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.
2. Mike Burmester and Yvo Desmedt. A secure and efficient conference key distribution system. In Alfredo De Santis, editor, *Advances in Cryptology — EUROCRYPT'94: Workshop on the Theory and Application of Cryptographic Techniques Perugia, Italy, May 9–12, 1994 Proceedings*, pages 275–286, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.
3. Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474. Springer, 2001.
4. Craig Costello and Huseyin Hisil. A simple and compact algorithm for SIDH with arbitrary degree isogenies. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 303–329. Springer, 2017.
5. 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.
6. Luca De Feo, David Jao, and Jérôme Plût. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. *J. Math. Cryptol.*, (to appear). `http://eprint.iacr.org/2011/506`.

7. Lein Harn and Changlu Lin. Efficient group Diffie-Hellman key agreement protocols. *Comput. Electr. Eng.*, 40(6):–, August 2014.
8. David Jao, Reza Azarderakhsh, Matthew Campagna, Craig Costello, Amir Jalali, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Joost Renes, Vladimir Soukharev, and David Urbanik. Supersingular isogeny key encapsulation. *NIST submissions*, 2017.
9. 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.
10. Nam-Su Jho, Myung-Hwan Kim, Dowon Hong, and Byung-Gil Lee. Multiparty key agreement using bilinear map. *IACR Cryptology ePrint Archive*, 2007:439, 2007.
11. Antoine Joux. The Weil and Tate pairings as building blocks for public key cryptosystems. In *Algorithmic number theory (Sydney, 2002)*, volume 2369 of *Lecture Notes in Comput. Sci.*, pages 20–32. Springer, Berlin, 2002.
12. Jonathan Katz and Moti Yung. Scalable protocols for authenticated group key exchange. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003: 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003. Proceedings*, pages 110–125, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.
13. Young-Ran Lee, Hyang-Sook Lee, and Ho-Kyu Lee. Multi-party authenticated key agreement protocols from multi-linear forms. *Applied Mathematics and Computation*, 159(2):317 – 331, 2004.
14. S. Mandal and S. Mohanty. Multi-party key-exchange with perfect forward secrecy. In *2014 International Conference on Information Technology*, pages 362–367, Dec 2014.
15. Christophe Petit. Faster algorithms for isogeny problems using torsion point images. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II*, volume 10625 of *Lecture Notes in Computer Science*, pages 330–353. Springer, 2017.
16. Michael Steiner, Gene Tsudik, and Michael Waidner. Diffie-Hellman key distribution extended to group communication. In *Proceedings of the 3rd ACM Conference on Computer and Communications Security*, CCS '96, pages 31–37, New York, NY, USA, 1996. ACM.
17. Jacques Vélu. Isogénies entre courbes elliptiques. *C. R. Acad. Sci. Paris Sér. A-B*, 273:A238–A241, 1971.