Intuitive Understanding of Quantum Computation and Post-Quantum Cryptography (Chapter 3)

Nguyen Thoi, Minh Quan [*]

3 Supersingular Isogeny Diffie-Hellman(SIDH)

SIDH [1][2][3][4] is the hardest to understand post-quantum cryptography construction. If you read my previous article "Intuitive Advanced Cryptography" [5], you would know that ECDH uses 1 elliptic curve, then pairing increases the difficulty significantly by using 2 elliptic curves. SIDH is the next difficulty level where it uses a graph (set) of elliptic curves. Therefore, this chapter requires prerequisite: you must watch the movie Inception by Christopher Nolan. I'm serious! Furthermore, we'll give up on the ambitious goal of analyzing SIDH's security. Instead, we'll set a humble (not-so-humble) goal to understand how SIDH works. If we can't achieve our goal, change it to an easier one :)

Alice		**Bob**
$A = aG$	$\xrightarrow{\quad A \quad}$	
	$\xleftarrow{\quad B \quad}$	$B = bG$
$aB = a(bG) = abG$		$bA = b(aG) = baG$

Let's recall a few terminologies in elliptic curve and ECDH protocol. In this article, we'll use supersingular [1] elliptic curve E defined over a finite field $F_q = F_{p^2}$. An extraordinary property of elliptic curve is that we can define addition operation $+$ between its points, i.e., given 2 points $P, Q \in E$, the operation $P + Q$ makes sense and the result is another point $R \in E$. How points addition operation $+$ is defined is not our concern, what we care is that $(E, +)$ forms a group. The order of a point P is the smallest number n for which $nG = 0$ where 0 denotes the identity of E. The ECDH protocol works as follows. Alice generates a random private key a, computes her public key $A = aG$ and sends A to Bob. Bob generates a random private key b, computes his public key $B = bG$ and sends B to Alice. Now Alice computes $aB = a(bG) = abG$ and Bob computes $bA = b(aG) = baG$. We notice that Alice and Bob compute the same result abG and it's their shared key that can be used for encryption.

Let's look at ECDH from a different perspective using graph terminology. Let consider points in the elliptic curve E as nodes in a graph and 2 points are connected by an edge if their difference is G. Using graph terminology, Alice computes her public key aG by travelling through the following path in the graph: $P_1 = G \xrightarrow{\text{G}} P_2 = 2G \xrightarrow{\text{G}} P_3 = 3G \cdots \xrightarrow{\text{G}} P_a = aG$. The security relies on the fact that knowing the destination point $P_a = aG$ doesn't help the adversary compute

[*] https://www.linkedin.com/in/quan-nguyen-a3209817, https://scholar.google.com/citations?user=9uUqJ9IAAAAJ, msuntmquan@gmail.com

[1] Let's ignore what supersingular means. For advanced readers, the word supersingular probably triggers your negative reaction because using supersingular curves in ECDH is not safe. However, SIDH uses a graph of elliptic curves, not points within 1 elliptic curve like ECDH, so the computational hard problem is different.

how many edges $\xrightarrow{G}$ that Alice has gone through. In terms of practical computation, Alice does not compute $P_a = aG$ linearly as shown above, instead she uses a double-and-add algorithm. In special case when $a = 2^k$, Alice computes $P_1 = G, P_2 = 2P_1, P_4 = 2P_2, \cdots, P_{2^k} = 2P_{2^{k-1}}$. In other words, in ECDH protocol, each node is a point and we travel from points to points within 1 elliptic curve. In SIDH protocol, a node is an elliptic curve [2] and we travel from elliptic curves to elliptic curves and if we look inside 1 node/elliptic curve we'll see multiple points inside it. It's similar to dreams in the movie Inception. Inside each node (elliptic curve) in SIDH protocol, there are other nodes (points) within that elliptic curve. From another perspective, it's similar to abstraction layers. At the SIDH layer, each node in a graph is an elliptic curve and we abstract away the detail in the lower layer of how points within that elliptic curve look like. We talked about nodes, but we haven't defined edges in SIDH protocol. I.e. what connects 1 node (1 elliptic curve) to another?

The connection from 1 node (1 elliptic curve) to another in SIDH protocol is an isogeny. Formally, isogeny is a non-constant rational map $E_1 \xrightarrow{\phi} E_2$ which is also a group homomorphism from $E_1(F_q)$ to $E_2(F_q)$. What rational map means is not our concern but we'll pay attention to what group homomorphism is. In this context, group homomorphism means that $\phi(\alpha P + \beta Q) = \alpha\phi(P) + \beta\phi(Q)$ where $P, Q \in E_1, \alpha, \beta \in \mathbb{Z}$. I.e., if we're given $\phi(P), \phi(Q)$ then we can compute ϕ of any linear combination $\alpha P + \beta Q$ of P and Q. In the next paragraphs, we'll introduce a few new concepts and isogeny's properties.

The kernel of $E_1 \xrightarrow{\phi} E_2$ is the set of points $P \in E_1$ such as $\phi(P) = 0$.

The degree of isogeny $E_1 \xrightarrow{\phi} E_2$, denoted as $\deg\phi$, is the number of elements in the kernel $\ker(\phi)$.

The multiplication-by-n map $E \xrightarrow{[n]} E$ maps a point P to nP. The n-torsion group of E, denoted $E[n]$ is the set of all points $P \in E$ such that nP is the identity. In other words, n-torsion group is the kernel of a multiplication-by-n map.

The composition of isogenies $E_0 \xrightarrow{\phi_0} E_1$ and $E_1 \xrightarrow{\phi_1} E_2$ is an isogeny $\phi_1 \circ \phi_0 : E_0 \to E_2$ ($E_0 \xrightarrow{\phi_0} E_1 \xrightarrow{\phi_1} E_2$) and $\deg(\phi_1 \circ \phi_0) = \deg\phi_1 * \deg\phi_0$. This composition property is important as it allows us to combine edges to create paths such as $E_0 \xrightarrow{\phi_0} E_1 \xrightarrow{\phi_1} E_2$. Let's look at a path with more than 2 edges. If we have a series of k edges (isogenies), each have degree 2 as follows: $E_0 \xrightarrow{\phi_0} E_1 \xrightarrow{\phi_1} E_2 \xrightarrow{\phi_2} E_3 \cdots \xrightarrow{\phi_{k-1}} E_k$ then the composition $\phi_k \circ \phi_{k-1} \cdots \phi_0 : E_0 \to E_k$ has order 2^k. Do you notice the similarity with ECDH?

ECDH:	$P_1 = G$	$P_2 = 2P_1$	$P_4 = 2P_2 \cdots$	$P_{2^k} = 2P_{2^{k-1}}$
SIDH:	$E_1 = \phi_0(E_0)$	$E_2 = \phi_1(E_1)$	$E_3 = \phi_2(E_2) \cdots$	$E_k = \phi_{k-1}(E_{k-1})$

The previous paragraph assumes that we know how to construct isogeny ϕ_i. It's time to discuss isogeny's construction. Given a finite subgroup $G \in E_1$, Vélu formula [3] constructs an isogeny $\phi : E_1 \to E_2$ that has G as kernel and we write $E_2 = \phi(E_1) = E_1/\langle G \rangle$. What it means is that if we start at E_1, choose a subgroup G as kernel, then we not only can construct an edge (isogeny) from E_1 but we also know how to compute the other end E_2 of the constructed edge. In other words, an edge (isogeny) is determined by its kernel (subgroup of starting node). We'll come back to this property in later paragraphs.

We're not ready to describe the protocol yet. The elliptic curves in SIDH have special structure, so we have to look at them first. We will work with extension field $F_q = F_{p^2}$ where p is a prime of the form $p = l_A^{e_A} l_B^{e_B} - 1$, $l_A = 2, l_B = 3$. Fix a supersingular curve E_0 over F_q. We have $E_0[2^{e_A}] \cong Z_{2^{e_A}} \times Z_{2^{e_A}}$. In layman's term, it means that there exist 2 points P_A, Q_A of order 2^{e_A} and they're the basis of 2^{e_A}-torsion group of E_0 (aka kernel of multiplication-by-2^{e_A} map). We write $E_0[2^{e_A}] = \langle P_A, Q_A \rangle$. Similarly, $\langle P_B, Q_B \rangle = E_0[2^{e_B}]$. All E_0, P_A, Q_A, P_B, Q_B are public parameters.

[2] For strict readers, a node is a set of isomorphic elliptic curves that have the same j-invariants.

[3] The explicit formula is complicated, so we'll skip it.

Let's combine all the knowledge in the previous paragraphs and see how far we can go. If we fix a point $R_0 = [m_A]P_A + [n_A]Q_A, m_A, n_A \in \mathbb{Z}$ of order 2^{e_A} then we can efficiently construct an isogeny $\phi_A : E_0 \to E_A$ having $\langle R_0 \rangle$ as kernel. Wait a second, we knew how to compute ϕ_A from kernel $\langle R_0 \rangle$ using Vélu's formula, so what's new? The key word is "efficient". Note that the kernel $\langle R_0 \rangle$ is a large group with 2^{e_A} elements and Vélu's formula requires us to explicitly list all the elements in the kernel. We'll use composition of isogenies property to decompose ϕ_A into smaller isogenies. Recall that the composition of isogenies of degree 2 is the isogeny $\phi_k \circ \phi_{k-1} \cdots \phi_0 : E_0 \to E_k$ of order 2^k: $E_0 \xrightarrow{\phi_0} E_1 \xrightarrow{\phi_1} E_2 \xrightarrow{\phi_2} E_3 \cdots \xrightarrow{\phi_{k-1}} E_k$. The idea to construct $\phi_A : E_0 \to E_A$ is to find $\phi_i, i = 0, \cdots, e_A - 1$ (each has small order 2 and hence is efficiently computable) such that $\phi_A = \phi_{e_A} \cdots \phi_1 \circ \phi_0$ and $E_k = E_A$ has $\langle K_A \rangle$ as kernel. Let's see how we can construct isogeny of order 2: $E_0 \xrightarrow{\phi_0} E_1$. To do that, we need to find a point of order 2 that generates the kernel of ϕ_0 (recall that isogeny is determined by its kernel). The point that we're looking for is $2^{e_A-1}R_0$. Why's that? We know that R_0 has order 2^{e_A}, i.e., $2^{e_A}R_0 = 0$ or $2(2^{e_A-1}R_0) = 0$. In other words, $2^{e_A-1}R_0$ is a point of order 2 and it's the point we're looking for. Denote $R_1 = \phi_0(R_0)$ then R_1 has order 2^{e_A-1} in E_1. Using a similar argument, we see that $2^{(e_A-1)-1}R_1$ is the kernel of isogeny $\phi_1 : E_1 \to E_2$. In general, the following algorithm will construct $E_0, E_1, \cdots, E_{e_A-1}$ and isogenies $\phi_0, \phi_1, \cdots, \phi_{e_A-1}$

$$E_{i+1} = E_i/\langle 2^{e_A-i-1}R_i \rangle, \phi_i : E_i \to E_{i+1}, R_{i+1} = \phi_i(R_i)$$

Finally, after all the hard work, we're ready to describe SIDH protocol.

Alice

$E_A = E_0/\langle [m_A]P_A + [n_A]Q_A \rangle = E_0/\langle K_A \rangle$

$E_0 \xrightarrow{\phi_A} E_A$

Bob

$E_B = E_0/\langle [m_B]P_B + [n_B]Q_B \rangle = E_0/\langle K_B \rangle$

$E_0 \xrightarrow{\phi_B} E_B$

$\phi_A(P_B)$,
$\phi_A(Q_B)$,
E_A

$\longrightarrow$

$\phi_B(P_A)$,
$\phi_B(Q_A)$,
E_B

$\longleftarrow$

$E_B \xrightarrow{\phi'_A} E_{AB}$

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

$E_A \xrightarrow{\phi'_B} E_{BA}$

$E_{BA} =$
$E_A/\langle [m_B]\phi_A(P_B) + [n_B]\phi_A(Q_B) \rangle$

The starting node (elliptic curve) for both Alice and Bob is E_0. Alice chooses her secret isogeny (path) $\phi_A : E_0 \to E_A$ by choosing a secret kernel $K_A = \langle [m_A]P_A + [n_A]Q_A \rangle$ where m_A, n_A are randomly generated in $\{0, \cdots, 2^{e_A-1}\}$. We know how to compute ϕ_A efficiently from the previous paragraphs. Alice then computes $\phi_A(P_B), \phi_A(Q_B)$ and sends her public key $\{E_A, \phi_A(P_B), \phi_A(Q_B)\}$ to Bob. Bob chooses his secret isogeny (path) $\phi_B : E_0 \to E_B$ by choosing a secret kernel $K_B = \langle [m_B]P_B + [n_B]Q_B \rangle$ where m_B, n_B are randomly generated in $\{0, \cdots, 3^{e_B-1}\}$. Bob computes $\phi_B(P_A), \phi_B(Q_A)$ and sends his public key $\{E_B, \phi_B(P_A), \phi_B(Q_A)\}$ to Alice.

After receiving Bob's public key, Alice uses $E_B, \phi_B(P_A), \phi_B(Q_A)$ to construct isogeny $\phi'_A : E_B \to E_{AB}$ with kernel $\langle [m_A]\phi_B(P_A) + [n_A]\phi_B(Q_A) \rangle$. Similarly, after receiving Bob's public key, Bob uses $E_A, \phi_A(P_B), \phi_A(Q_B)$ to construct isogeny $\phi'_B : E_A \to E_{BA}$ with kernel $\langle [m_B]\phi_A(P_B) + [n_B]\phi_A(Q_B) \rangle$.

At this point, I guess you have many questions. What are $\phi_B(P_A)$, $\phi_B(Q_A)$ for? What does complicated formula $E_0/\langle [m_A]\phi_B(P_A) + [n_A]\phi_B(Q_A) \rangle$ mean? As isogeny is group homomorphism, we have $\phi_B([m_A]P_A + [n_A]Q_A) = [m_A]\phi_B(P_A) + [n_A]\phi_B(Q_A)$. Furthermore, we have $K_A = [m_A]P_A + [n_A]Q_A$, so $\phi_B(K_A) = [m_A]\phi_B(P_A) + [n_A]\phi_B(Q_A)$. This means that $\phi_B(P_A), \phi_B(Q_A)$ allows Alice to compute $\phi_B(K_A)$ although Alice does not know Bob's secret isogeny ϕ_B. This discussion shed some light on the cumbersome formula $E_B/\langle[m_A]\phi_B(P_A)+[n_A]\phi_B(Q_A)\rangle$, it is just $E_B/\langle\phi_B(K_A)\rangle$. Now, let's revisit $E_{AB} : E_0 \xrightarrow{\phi_B} E_B \xrightarrow{\phi'_A} E_{AB}$, we have $E_{AB} = \phi'_A(\phi_B(E_0)) = E_0/\langle K_A, K_B\rangle = \phi'_B(\phi_A(E_0)) = E_{BA}$. I.e., Alice and Bob compute the same shared secret.

References

[1] David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies.

[2] Craig Costello. Supersingular isogeny key exchange for beginners.

[3] David Urbanik. A friendly introduction to supersingular isogeny diffie-hellman.

[4] Craig Costello. An introduction to supersingular isogeny-based cryptography.

[5] Nguyen Thoi Minh Quan. Intuitive advanced cryptography.