An Adaptive Attack on 2-SIDH

Samuel Dobson[a], Steven D. Galbraith[a], Jason Legrow[b], Yan Bo Ti[a], and Lukas Zobernig[a]

[a]Department of Mathematics, The University of Auckland, New Zealand
samuel.dobson.nz@gmail.com, s.galbraith@auckland.ac.nz, yanbo.ti@gmail.com,
lukas.zobernig@auckland.ac.nz
[b]Department of Combinatorics and Optimization, University of Waterloo, Canada
jlegrow@waterloo.ca

Abstract

We present a polynomial-time adaptive attack on the 2-SIDH protocol. The 2-SIDH protocol is a special instance of the countermeasure proposed by Azarderakhsh, Jao and Leonardi to perform isogeny-based key exchange with static keys in the presence of an adaptive attack. This countermeasure has also been recently explicitly proposed by Kayacan.

Our attack extends the adaptive attack by Galbraith, Petit, Shani and Ti (GPST) to recover a static secret key using malformed points. The extension of GPST is non-trivial and requires learning additional information. In particular, the attack needs to recover intermediate elliptic curves in the isogeny path, and points on them. We also discuss how to extend the attack to k-SIDH when $k > 2$ and explain that the attack complexity is exponential in k.

1 Introduction

The Supersingular Isogeny Diffie–Hellman (SIDH) protocol was introduced in 2011 by Jao and De Feo [6, 3] as a post-quantum key exchange scheme. The adaptive attack on SIDH due to Galbraith, Petit, Shani, and Ti [4] (GPST) shows that SIDH cannot be used for non-interactive key exchange. Similarly if SIDH is being used for ElGamal then one needs to use a padding scheme to check correctness of the ciphertexts (in other words, SIDH-KEM can only be safely used with CCA2 protection). On the other hand, CSIDH [2] can be used for non-interactive key exchange or CCA1-secure encryption.

Azarderakhsh, Jao, and Leonardi [1] give a solution to this problem. By running k instances in parallel (e.g. for $k = 60$) they prove that the scheme is classically secure. In the non-interactive key exchange setting this requires Alice and Bob to compute about k^2 isogenies to obtain the shared key, which is inefficient. Hence, a natural question is whether $k = 2$ is sufficient for secure non-interactive key exchange. If it were the case that $k = 2$ was secure for non-interactive key exchange then the SIDH approach would be faster than CSIDH.

The aim of this work is to give an attack on the scheme with $k = 2$. Our methods clearly extend to $k > 2$ and we discuss how the complexity of the attack grows with k in Section 7.1. Unlike the original GPST attack, we cannot just "read off" the bits of the secret directly by engaging in key exchange sessions. Instead, it seems to be necessary to gradually recover the intermediate elliptic curves and certain points on them. This is because at various stages we need to compute "random" branches in the isogeny graph.

We now sketch the basic idea in the ElGamal setting, where a user Alice has $k = 2$ secret keys and Bob sends a single curve. Fix SIDH parameters E over the field $\mathbb{F}_{p^2}$ with $E[2^n]$ and $E[3^m]$ contained in $E(\mathbb{F}_{p^2})$. Let $P, Q \in E$ be a basis for $E[2^n]$. Alice has two private keys $\alpha^{(1)}$ and $\alpha^{(2)}$ and her public key is the pair

$$E_A^{(1)} = E/\langle P + [\alpha^{(1)}]Q\rangle, \qquad E_A^{(2)} = E/\langle P + [\alpha^{(2)}]Q\rangle.$$

Figure 1 shows the isogeny paths for Alice's two keys. There is also a pair of points on E that generate $E[3^m]$ and Alice sends the images of those points on $E_A^{(1)}$ and $E_A^{(2)}$ under her isogenies. In encryption, Bob computes a random 3^m-isogeny $\phi : E \to E_B$ and sends E_B, $\phi(P)$ and $\phi(Q)$. Alice and Bob compute the shared key

$$\text{Hash}\left(j\left(E_B/\langle\phi(P)+[\alpha^{(1)}]\phi(Q)\rangle\right), j\left(E_B/\langle\phi(P)+[\alpha^{(2)}]\phi(Q)\rangle\right)\right).$$

A malicious Bob will try to learn Alice's secrets $\alpha^{(1)}$ and $\alpha^{(2)}$ by sending (E_B, U, V) for carefully and adaptively chosen points $U, V \in E_B[2^n]$. The attacker will gradually learn the sequence of curves $E_i^{(1)}$ and $E_i^{(2)}$ in Figure 1. Knowledge of the secret key $\alpha^{(k)}$ is equivalent to knowledge of the kernel of the isogeny from E to $E_A^{(k)}$, and we incrementally determine this by computing the kernels of the isogenies $\phi_i^{(k)} : E_i^{(k)} \to E_{i-1}^{(k)}$.

The astute reader has already noticed that this model of 2-SIDH is not the same as that presented in [1, 7]. In Section 4.1 we show the equivalence of these two models. We now give an outline of this work. Section 2 will introduce some notation and definitions. Section 3 then recalls the GPST attack. We present the $k = 2$ protocol in Section 4. Section 5 explains the attack, which we have implemented in `MAGMA`. Code is available at the following URL

https://github.com/kiwi-crypto/2SIDH-Extended-Adaptive-Attack

Section 6 gives a few implementation details and Section 7 elaborates on extensions such as the case $k > 2$ or when Alice is attacking Bob.

2 Notation and Definitions

We begin with some important definitions and notation which we will use throughout this work. The notation is summarised in Figure 1. As shown in the figure, we have two isogenies from E, so we will use superscripts to denote which isogeny we are on. Subscripts will be used to denote bit numbers. We will use the convention that the 0-th bit is the parity bit, thus α_0 refers to the first (least significant) bit of a key α. We will often use that a key α can be written in the form $\alpha = K_i + \alpha' 2^i$ for some α'. Here the *i-th partial key* K_i of a key α is defined as

$$K_i = \sum_{k=0}^{i-1} \alpha_k 2^k.$$

We denote *point halving* with $[\frac{1}{2}]R = \{S \mid [2]S = R\}$, i.e. the set of all points S such that $R = [2]S$. The *2-neighbours* of an elliptic curve E are the codomains of all possible 2-isogenies emanating from E, and similarly the *4-neighbours* are those curves which are two 2-isogeny (non-backtracking) "steps" from E.

The $\phi_i^{(k)}$ are 2-isogenies from $E_i^{(k)} \to E_{i-1}^{(k)}$, while the $\psi_i^{(k)}$ are compositions of $\phi^{(k)}$s, such that $\psi_i^{(k)} : E \to E_i^{(k)}$ and $\psi_0^{(k)}$ is the secret 2^n-isogeny generated by the chosen secret keys in the protocol.

We will use $A^{(k)}$ to denote the kernel generators $P + [\alpha^{(k)}]Q$. We define $A_i^{(k)} = \psi_i^{(k)}(A^{(k)})$, and $Q_i^{(k)} = \psi_i^{(k)}(Q)$.

3 Adaptive Attack on SIDH

First we recall the SIDH scheme itself. Setup of the scheme involves the choice of a prime of the form $p = \ell_A^n \cdot \ell_B^m \cdot f \pm 1$, where ℓ_A, ℓ_B are small primes, $\ell_A^n \approx \ell_B^m$, and f is a small cofactor. Typically, $\ell_A = 2$ and $\ell_B = 3$, so we will assume this case in the following discussion, although the same shall always apply to the general case too. A supersingular elliptic curve E is then constructed over the field $\mathbb{F}_{p^2}$, and made public along with chosen bases $\langle P_A, Q_A\rangle = E[2^n]$, $\langle P_B, Q_B\rangle = E[3^m]$ (these torsion subgroups have order 2^{2n} and 3^{2m} respectively).

To perform the key exchange, Alice will select $0 \le a_1, a_2 < 2^n$ not both divisible by 2, and similarly Bob will select $0 \le b_1, b_2 < 3^m$ not both divisible by 3. These integers allow each party to compute a secret subgroup each

$$G_A = \langle [a_1]P_A + [a_2]Q_A\rangle, \qquad G_B = \langle [b_1]P_B + [b_2]Q_B\rangle.$$

$$\begin{tikzcd}
E = E_n \arrow[r, dashed] \arrow[r, bend left, "\psi_i^{(1)}"] \arrow[rr, bend left=60, "\psi_{i-1}^{(1)}"] & E_i^{(1)} \arrow[r, "\phi_i^{(1)}"] & E_{i-1}^{(1)} \arrow[r, dashed] & E_0^{(1)} = E_A^{(1)}
\end{tikzcd}$$

$$\begin{tikzcd}
E = E_n \arrow[r, dashed] \arrow[r, bend left, "\psi_i^{(2)}"] \arrow[rr, bend left=60, "\psi_{i-1}^{(2)}"] & E_i^{(2)} \arrow[r, "\phi_i^{(2)}"] & E_{i-1}^{(2)} \arrow[r, dashed] & E_0^{(2)} = E_A^{(2)}
\end{tikzcd}$$

Figure 1: Isogeny paths for the two instances.

These subgroups are then used via Vélu's formulae [8] to construct isogenies $\phi_i : E \to E_i = E/G_i$, $i \in \{A, B\}$. Alice will send to Bob the triple $\big(E_A, \phi_A(P_B), \phi_A(Q_B)\big)$ and Bob will reciprocate appropriately, and the exchanged points will allow Bob (and in the same manner Alice) to compute

$$\langle [b_1]\phi_A(P_B) + [b_2]\phi_A(Q_B)\rangle = \langle \phi_A([b_1]P_B + [b_2]Q_B)\rangle = \phi_A(G_B).$$

With these new subgroups, Vélu's formulae can then be used to compute isogenies $E_A \to E_A/\phi_A(G_B) \simeq E_B/\phi_B(G_A) \simeq E/\langle G_A, G_B\rangle$. Because the j-invariant $j\left(E/\langle G_A, G_B\rangle\right)$ of these curves is isomorphism-invariant, it can be used as the shared key. This is summarised in Figure 2.

$$\begin{tikzcd}
E \arrow[r, "\phi_A"] \arrow[d, "\phi_B"'] & E/\langle G_A\rangle \arrow[d, "\phi_B'"] \\
E/\langle G_B\rangle \arrow[r, "\phi_A'"'] & E/\langle G_A, G_B\rangle
\end{tikzcd}$$

Figure 2: The SIDH key exchange.

As is now standard (see for example Lemma 1 of [4]) any choice of secret integers (a_1, a_2) is equivalent to either $(1, \alpha)$ or $(\alpha', 1)$. It thus suffices to consider only these cases. For the rest of this work we use keys of the form $(1, \alpha)$.

The adaptive GPST attack is an active attack, in which one party (we shall assume Alice) uses a fixed key for each exchange. This could, for example, be a webserver using a static key when deriving secrets with each visitor to the site. The other party uses the key exchange protocol to slowly leak secret information and eventually reveal Alice's static secret key. The adaptive attack works in the two attack models defined, each in terms of access to a different oracle:

1. $O(E, R, S) = E/\langle R + [\alpha]S\rangle$.
2. $O(E, R, S, E') = 1$ if $j\left(E/\langle R + [\alpha]S\rangle\right) = j\left(E'\right)$ and 0 otherwise.

The second is weaker so the attack is described in this model for illustration of its potency.

Remark 1. Our analysis, as with GPST, assumes that $j(E/G) = j(E/H)$ if and only if $G = H$ for two cyclic subgroups G and H. This statement is false in general: In the supersingular ℓ-isogeny graphs that we consider here, two curves E and E' could be connected by different ℓ^k-isogenies (for some $k \in \mathbb{N}$) — this is equivalent to $\mathrm{End}(E)$ containing an element of degree ℓ^{2k} which passes through E' after k steps of ℓ-isogenies.

Let us estimate the probability that a uniformly chosen supersingular curve over $\mathbb{F}_{p^2}$ has an endomorphism of degree ℓ^{2k}. Specifically, we are fixing ℓ and k, and choosing uniformly a supersingular j-invariant. We would like to study the probability that the corresponding elliptic curve has an endomorphism of degree ℓ^{2k}. For our purposes it suffices to consider endomorphisms that arise from cyclic isogenies.

For $N \in \mathbb{N}$, the *classical modular polynomial* $\Phi_N(X,Y) \in \mathbb{Z}[X,Y]$ parametrises elliptic curves that are related by a cyclic isogeny of degree N. Fix a j-invariant j_0, then the roots of $\Phi_N(X,j_0)$ are all the j-invariants of elliptic curves that are N-isogenous to the elliptic curve with j-invariant j_0 via a cyclic isogeny of degree N. The polynomial $\Phi_N(X,Y)$ is symmetric in X,Y and it holds that $\deg_X \Phi_N(X,Y) = N\prod_{\ell|N}(1+1/\ell)$. Hence for $N=\ell^{2k}$, we have that $\deg \Phi_{\ell^{2k}}(X,X) \leq 2\ell^{2k-1}(\ell+1)$. Hence, in the worst case (assuming that every j-invariant that is a root of $\Phi_{\ell^{2k}}(X,X)$ has supersingular reduction modulo p) the probability for a supersingular elliptic curve over $\mathbb{F}_{p^2}$ to have an endomorphism of degree ℓ^{2k} is bounded by $24\ell^{2k-1}(\ell+1)/p$. Here we used that the number of supersingular elliptic curves is $\lfloor p/12 \rfloor + e$ where $e = 0,1,1,2$ if $p \equiv 1,5,7,1 \mod 12$.

From this we see that most supersingular elliptic curves over $\mathbb{F}_{p^2}$ will have such an endomorphism for large k but on the other hand it becomes progressively unlikely that we encounter a curve that has such an endomorphism for smaller and smaller k. Specifically (assuming $p \approx \ell^{2n}$ for SIDH parameters) the probability is negligible in the gap between n and k: $\Pr\left[\exists \phi \in \operatorname{End}(E) : \deg\phi = \ell^{2k}\right] \approx 1/\ell^{2(n-k)}$. This should come to no surprise, the supersingular isogeny graph is connected and Ramanujan after all. Thinking in terms of norm forms, we also see that the probability that it is possible to represent ℓ^{2k} in a non-trivial way (with respect to p) is negligible in the gap $2(n-k)$.

In the SIDH protocol, E_B is not sampled uniformly in the Ramanujan graph (since $n < \log(p/12)$). By this heuristic, we can ignore situations where where E and $E/\langle G_A\rangle$ are in such a position in the isogeny graph that there exists the aforementioned cycle of length 2^{2n} passing through $E/\langle G_A\rangle$ after n steps. We could also find ourselves in the situation where there exists a unique path from E to some intermediate curve E' such that E' has an endomorphism of degree $2^{2n'}$ passing through $E/\langle G_A\rangle$ after $n' < n$ steps. But since $n' < n$ decreases as the distance between E' and E grows, by above reasoning the probability that such an endomorphism exists in the first place also decreases (we found that it is negligible in the gap $2(n-n')$).

Hence, for the remainder of this work we shall assume that we are always in the *good* case in which E and $E/\langle G_A\rangle$ are connected by a unique path of 2-isogenies. Charles, Goren, and Lauter [5] have considered using the supersingular elliptic curve isogeny graph to build cryptographic hash functions. They reduced the hardness of finding hash collisions to the hardness of finding cycles in the supersingular isogeny graph. This is the same problem that we are considering in the above discussion and is more evidence that considering the *good* case is sufficient.

3.1 The GPST Attack

The first step of the attack simultaneously reveals which of two oracle cases we are in as well as the first bit of α. This is done by honestly generating the ephemeral key $(E_B, R = \phi_B(P_A), S = \phi_B(Q_A))$, and querying the oracle on $(E_B, R, S + [2^{n-1}]R, E_{AB})$. A result of 1 from the oracle informs the attacker that

$$E_{AB} \simeq E_B/\langle R + [\alpha](S + [2^{n-1}]R)\rangle$$

so $\langle R + [\alpha](S+[2^{n-1}]R)\rangle = \langle R + [\alpha]S\rangle$. Hence, the key is of the first form, and α is even, by the following lemma ([4], Lemma 2).

Lemma 3.1. *Let $R, S \in E[2^n]$ be linearly independent points of order 2^n and let $\alpha \in \mathbb{Z}$. Then*

$$\langle R + [\alpha](S + [2^{n-1}]R)\rangle = \langle R + [\alpha]S\rangle$$

if and only if α is even.

Otherwise, the attacker learns that α is odd.

We now describe the remainder of the attack. Suppose we have learnt the first i bits of α. We can write α in the form $\alpha = K_i + \alpha_i 2^i + \alpha' 2^{i+1}$, K_i is the i-th partial key defined above, $\alpha_i \in \{0,1\}$ is the next bit of α

we hope to learn, and α' is unknown. The attacker will honestly generate $(E_B, R = \phi_B(P_A), S = \phi_B(Q_A))$ and E_{AB} as before, according to the protocol. He will then query the oracle with

$$\left(E_B, [\theta](R - [2^{n-i-1}K_i]S), [\theta]([1 + 2^{n-i-1}]S), E_{AB}\right)$$

where θ is a scaling parameter included to avoid detection of the attack with Weil pairing validation (this will not be discussed further here, please see [4] for full details). Both points sent to the oracle have the correct order so the attack is not detectable by order validation. If the response of the oracle is 1, then $\alpha_i = 0$, otherwise $\alpha_i = 1$, because

$$\begin{aligned}
&\langle R - [2^{n-i-1}K_i]S + [\alpha][1 + 2^{n-i-1}]S\rangle \\
=&\langle R + [\alpha]S + [-2^{n-i-1}K_i + 2^{n-i-1}(K_i + 2^i\alpha_i + 2^{i+1}\alpha')]S\rangle \\
=&\langle R + [\alpha]S + [\alpha_i 2^{n-1}]S\rangle \\
=&\begin{cases} \langle R + [\alpha]S\rangle & \text{if } \alpha_i = 0\,, \\ \langle R + [\alpha]S + [2^{n-1}]S\rangle & \text{if } \alpha_i = 1\,. \end{cases}
\end{aligned}$$

The last two bits $\alpha_{n-2}, \alpha_{n-1}$ should be brute-forced because for these last two bits, there may not exist a suitable scaling θ, making the attack detectable otherwise.

4 2-SIDH

The usual way to secure SIDH when one party is using static keys is the Fujisaki–Okamoto transform, which allows to detect malicious behaviour. But this cannot be used in the static-static setting.

A different countermeasure proposed by Azarderakhsh, Jao and Leonardi [1] circumvents the attack by having each party generate multiple instances of the SIDH protocol. This is known as the k-SIDH key agreement protocol. In their work, they proposed the use of $k = 60$ if the static-key user is working in 2^n-torsion group to achieve 128-bits of classical security, and $k = 113$ to achieve 128-bits of quantum security. If the static-key user is working in the 3^m-torsion group, they proposed $k = 50$ and $k = 94$ to achieve 128-bits of classical and quantum security respectively. More recently, Kayacan [7] proposed two countermeasures to thwart the adaptive attack. The first protocol proposed is simply the 2-SIDH key agreement protocol.

The 2-SIDH protocol has the same set-up as the SIDH protocol with the addition of a preimage resistant hash function $\mathrm{Hash}(\cdot)$. A prime number $p = 2^n \cdot 3^m \cdot f \pm 1$ is chosen, and a supersingular elliptic curve E defined over $\mathbb{F}_{p^2}$ and four points $P_A, Q_A, P_B, Q_B \in E(\mathbb{F}_{p^2})$ are chosen such that $\langle P_A, Q_A\rangle = E[2^n]$ and $\langle P_B, Q_B\rangle = E[3^m]$.

Alice chooses secrets $\alpha^{(1)} \leftarrow_R \mathbb{Z}/2^n\mathbb{Z}$ and $\alpha^{(2)} \leftarrow_R \mathbb{Z}/2^n\mathbb{Z}$ uniformly random, which she would use to compute the isogenies

$$\phi_A^{(k)} : E \to E_A^{(k)} = E/\langle P_A + [\alpha^{(k)}]Q_A\rangle$$

for $k = 1, 2$, and the points $R^{(k)} = \phi_A^{(k)}(P_B)$ and $S^{(k)} = \phi_A^{(k)}(Q_B)$. She then sends Bob her public key

$$\left((E_A^{(1)}, R^{(1)}, S^{(1)}), (E_A^{(2)}, R^{(2)}, S^{(2)})\right).$$

Bob will perform a similar procedure with points in $E[3^m]$. He chooses secrets $\beta^{(1)} \leftarrow_R \mathbb{Z}/3^m\mathbb{Z}$ and $\beta^{(2)} \leftarrow_R \mathbb{Z}/3^m\mathbb{Z}$ and computes

$$\phi_B^{(k)} : E \to E_B^{(k)} = E/\langle P_B + [\beta^{(k)}]Q_B\rangle$$

for $k = 1, 2$, and the points $U^{(k)} = \phi_B^{(k)}(P_A)$ and $V^{(k)} = \phi_B^{(k)}(Q_A)$. He then sends Alice his public key

$$\left((E_B^{(1)}, U^{(1)}, V^{(1)}), (E_B^{(2)}, U^{(2)}, V^{(2)})\right).$$

To obtain the shared secret, Alice takes Bob's public key and computes

$$z_{k,l} = j\left(E_B^{(k)}/\langle U^{(k)} + [\alpha^{(l)}]V^{(k)}\rangle\right)$$

and the hash

$$h = \mathrm{Hash}\left(z_{1,1}, z_{1,2}, z_{2,1}, z_{2,2}\right).$$

She will use the hash as the shared secret with Bob.

4.1 Attack Models

We will frame our attack by the use of oracle models that will model the information learned by an attacker against a user with a static secret key. In the following, we will assume that Alice is the user with a static secret key and the attacker is playing the role of Bob. This is the same as the oracle models of the original adaptive attack which will serve as the basis for the oracle model that we will use here. We will see that the oracle model used later will differ slightly from the 2-SIDH protocol as presented in §4, however we will show that this simpler model is equivalent to the original 2-SIDH protocol.

Adapting the oracle model found in [4], we see that in our case, we have the following two oracle models:

1. $O(E^{(1)}, E^{(2)}, R^{(1)}, S^{(1)}, R^{(2)}, S^{(2)}) = \text{Hash}\left(z_{1,1}, z_{1,2}, z_{2,1}, z_{2,2}\right)$, where
$$z_{k,l} = j\left(E^{(k)}/\langle R^{(k)} + [\alpha^{(l)}]S^{(k)}\rangle\right).$$
2. $O(E^{(1)}, E^{(2)}, R^{(1)}, S^{(1)}, R^{(2)}, S^{(2)}, h)$ which returns 1 if
$$h = \text{Hash}\left(z_{1,1}, z_{1,2}, z_{2,1}, z_{2,2}\right),$$
where $z_{k,l}$ is as above, and 0 otherwise.

We will focus on the second model. Furthermore, we can simplify this oracle model by having Bob send Alice a specific form of public key in the attack. This is possible because Bob can choose a public key of any form of his liking. We stress that the 2-SIDH (or generally k-SIDH) key exchange protocol is *stateless* and hence certain attack detection strategies are simply not possible. Such mechanisms include Alice testing whether the curve she received from Bob is special in any way (for example a fixed j-invariant) or static over multiple executions of the protocol. Rather than generating two secret keys $\beta^{(1)}, \beta^{(2)}$ and sending Alice his public key as described above, his behaviour can be modelled by selecting a single $\beta^{(1)}$ and duplicating the curves and points he sends in his public key

$$\left((E_B^{(1)}, U^{(1)}, V^{(1)}), (E_B^{(1)}, U^{(1)}, V^{(1)})\right).$$

Note that it would also suffice for Bob to hold his second secret key constant and vary only the first between queries. Upon receipt of this key, Alice will compute

$$\begin{aligned}&\text{Hash}\begin{pmatrix} j\left(E_B^{(1)}/\langle U^{(1)} + [\alpha^{(1)}]V^{(1)}\rangle\right), j\left(E_B^{(1)}/\langle U^{(1)} + [\alpha^{(2)}]V^{(1)}\rangle\right), \\ j\left(E_B^{(1)}/\langle U^{(1)} + [\alpha^{(1)}]V^{(1)}\rangle\right), j\left(E_B^{(1)}/\langle U^{(1)} + [\alpha^{(2)}]V^{(1)}\rangle\right)\end{pmatrix} \\ &= \text{Hash}(z_{1,1}, z_{1,2}, z_{1,1}, z_{1,2})\end{aligned}$$

If we consider a tweaked hash function

$$H'(m_1, m_2) = \text{Hash}(m_1, m_2, m_1, m_2)$$

it becomes obvious that we can define a third oracle model in the following way:

3. $O(E^{(1)}, R^{(1)}, S^{(1)}, h)$ which returns 1 if $h = H'\left(z_{1,1}, z_{1,2}\right)$, where $z_{k,l}$ is as above, and 0 otherwise.

Making this simplification, we can assume that the 2-SIDH protocol during the attack has the following description. Without loss of generality, we continue assuming that Alice is using a normalised static key $(\alpha^{(1)}, \alpha^{(2)})$, and that Bob is dishonest and is trying to recover the long-term secret $(\alpha^{(1)}, \alpha^{(2)})$. The discussion to come will assume that the points are in the 2^n-torsion, but it can be shown that the following methods and arguments will carry over to other torsion points.

Simplified 2-SIDH Protocol. Let p be a prime such that $p = 2^n \cdot 3^m \cdot f \pm 1$, where f is small and $2^n \approx 3^m$. More generally, we allow $p = \ell_A^n \cdot \ell_B^m \cdot f \pm 1$ where ℓ_A, ℓ_B are small primes, but for ease of exposition, we will let $\ell_A = 2$ and $\ell_B = 3$. Fix the field $\mathbb{F}_{p^2}$, and let E be a supersingular elliptic curve over this field. We fix generators P_A, Q_A for the torsion subgroup $E[2^n]$, and P_B, Q_B for $E[3^m]$.

Alice picks two random integers $0 \le \alpha^{(k)} < 2^n$, where $k = 1, 2$. Alice will then compute

$$G_A^{(k)} = \langle P_A + [\alpha^{(k)}]Q_A \rangle,$$

for $k = 1, 2$. At this step, we will use the normalised kernels (see [4, Lemma 1]), and represent $G_A^{(k)} = \langle P_A + [\alpha^{(k)}]Q_A \rangle$. She will use Vélu's formula to compute $\phi_A^{(k)} : E \to E_A^{(k)}$, where $\ker \phi_A^{(k)} = G_A^{(k)}$. She will compute and send to Bob the following tuple:

$$\left(E_A^{(1)}, E_A^{(2)}, \phi_A^{(1)}(P_B), \phi_A^{(1)}(Q_B), \phi_A^{(2)}(P_B), \phi_A^{(2)}(Q_B) \right).$$

Bob would perform a similar computation but only select a single pair of random integers, and output the tuple:

$$\left(E_B^{(1)}, E_B^{(1)}, \phi_B^{(1)}(P_A), \phi_B^{(1)}(Q_A), \phi_B^{(1)}(P_A), \phi_B^{(1)}(Q_A) \right).$$

Upon receiving of Bob's message, to derive the shared key, Alice would compute the two subgroups

$$H_A^{(k)} = \langle \phi_B^{(1)}(P_A) + [\alpha^{(k)}]\phi_B^{(1)}(Q_A) \rangle = \phi_B^{(1)}(G_A^{(k)}).$$

and the isogenies from $E_B^{(1)}$, with kernel equal to these subgroups. Call the codomain of these isogenies $E_{AB}^{(k)}$, then Alice uses

$$H'\left(j\left(E_{AB}^{(1)}\right), j\left(E_{AB}^{(2)}\right) \right)$$

as the shared key.

The attacker will assume the role of Bob and from that point of view, we use the following set-up and notations. The attacker will generate E_B honestly along with the points $\phi_B(P_A)$ and $\phi_B(Q_A)$. Observe that the attacker is only using a single branch. To simplify the notation, we will call these parameters E, $P^{(k)}$, $Q^{(k)}$. Now let $(\alpha^{(1)}, \alpha^{(2)}) \in \{1, \ldots, 2^n\}^2$ be the secrets unknown to the attacker, and let $E^{(1)} = E/\langle P + [\alpha^{(1)}]Q \rangle$ and $E^{(2)} = E/\langle P + [\alpha^{(2)}]Q \rangle$. The attacker's goal is to recover secrets equivalent to $\alpha^{(1)}$ and $\alpha^{(2)}$. The method for doing so will be to query model the interaction with Alice as Oracle (3).

5 The Extended Adaptive Attack

At a high level, the attack follows the same methodology as the GPST attack. However there is one major difficulty that needs to be overcome. Suppose at step i we have computed the first i bits of $\alpha^{(1)}$ and $\alpha^{(2)}$, so that we have

$$\alpha^{(1)} = K_i^{(1)} + \alpha_i^{(1)} 2^i + \alpha'^{(1)} 2^{i+1} \quad \text{and} \quad \alpha^{(2)} = K_i^{(2)} + \alpha_i^{(2)} 2^i + \alpha'^{(2)} 2^{i+1}.$$

To learn the next bit of $\alpha^{(1)}$ we will make an oracle query on points $U = R - [2^{n-i-1} K_i^{(1)}]S$ and $V = [1 + 2^{n-i-1}]S$ and get back a hash value $h = H'(j(E/\langle U + [\alpha^{(1)}]V \rangle), j(E/\langle U + [\alpha^{(2)}]V \rangle))$. To learn the next bit we need to check whether or not $j(E/\langle U + [\alpha^{(1)}]V \rangle) = j(E_A^{(1)})$. But we do not see this j-invariant, we only see the hash value h. Since H' is a hash function, the only way to check equality of the j invariant is to compute $H'(j(E_A^{(1)}, \star)$ for some appropriate j-invariant $\star$. The difficulty is that $j(E/\langle U + [\alpha^{(2)}]V \rangle)$ is not necessarily close in the isogeny graph to $j(E_A^{(2)})$. The solution is to use the available partial knowledge of $\alpha^{(2)}$ to compute a small list of candidate j-invariants, so that we can compute a few hash values and determine whether or not $j(E/\langle U + [\alpha^{(1)}]V \rangle) = j(E_A^{(1)})$.

To do this we need to compute what we call the "intermediate images" of the kernel and the point Q. More precisely we compute candidates for $A_{i+1}^{(k)}$ and $[2^{n-(i+1)}]Q_{i+1}^{(k)}$ (these points were defined in Section 2). Hence, the extended adaptive attack we will present in this section has two stages: one to recover the bits of the secret scalar, and another to recover the intermediate images.

The first stage for the recovery of bits is almost identical to the adaptive attack presented in [4]. The difference from the GPST adaptive attack arises from the way one instance interferes with the other.

This results in the necessity of making more guesses (but bounded at each step) to recover the bit. The guesses are generated by intermediate images which we recover in the second stage.

In the second stage, the goal is to recover the intermediate images required for the first stage of the next step. At this stage, we will use the bits recovered from the first stage to pull back the intermediate images used in the first stage. Not all the points in the pull back are suitable for use in the next step, so we will need to remove those points using queries to the oracle. We will show that the number of intermediate images remains constant at each step.

We now give some basic facts that will help us determine these points.

Lemma 5.1. *Consider the intermediate 2-isogeny $\phi_i^{(k)} : E_i^{(k)} \to E_{i-1}^{(k)}$ (for any $1 \le i \le n$). Then it holds that*

1. $\ker \phi_i^{(k)} = \langle [2^{i-1}] A_i^{(k)} \rangle$, *and*
2. $\ker \widehat{\phi}_i^{(k)} = \langle [2^{n-1}] Q_{i-1}^{(k)} \rangle$.

Proof. The first part (1) is apparent from the fact that $A^{(k)}$ has order 2^n so $A_i^{(k)}$ has order 2^i, and by inspecting Figure 1.

For (2), we first note that for any N-isogeny $\phi : E \to E'$, $\ker \hat{\phi} = \phi(E[N])$. Hence, we have

$$\begin{aligned} \ker \widehat{\phi}_i^{(k)} &= \phi_i^{(k)}(E_i[2]) \\ &= \psi_{i-1}^{(k)}(E[2])\,. \end{aligned}$$

We know that $E[2] = \langle [2^{n-1}]A, [2^{n-1}]Q \rangle$, hence

$$\begin{aligned} \psi_{i-1}^{(k)}(E[2]) &= \langle \psi_{i-1}^{(k)}([2^{n-1}]A), \psi_{i-1}^{(k)}([2^{n-1}]Q) \rangle \\ &= \langle [2^{n-i}]A_{i-1}^{(k)}, [2^{n-1}]Q_{i-1}^{(k)} \rangle \end{aligned}$$

But since $[2^{n-i}]A_{i-1}^{(k)} = \mathcal{O}$, we finally have our result. $\square$

We remark that these properties do not uniquely determine the points $A_{i+1}^{(k)}$ and $[2^{n-(i+1)}]Q_{i+1}^{(k)}$. For example, $\ker \phi_i^{(k)} = \langle [2^{i-1}][\lambda] A_i^{(k)} \rangle$ for any odd integer λ. Similarly, $\ker \widehat{\phi}_i^{(k)} = \langle [2^{n-1}][\lambda] Q_{i-1}^{(k)} \rangle$ for any odd integer λ. So the points we are computing are not uniquely determined.

5.1 Determining the First Bits and Intermediate Curves

The recovery of the first bits will be different from the subsequent bits. In this subsection, we will show how an attacker can recover the first bits using only the public information sent by Alice. This will provide the basis for the iterative step to recover the entire secret in the next section.

Recovering the first bit of each secret

To recover the first bits, we first send Alice the points P, $[1 + 2^{n-1}]Q$ and note that she would compute

$$\begin{aligned} P + [\alpha^{(1)}]Q + [\alpha^{(1)}][2^{n-1}]Q &= A^{(1)} + [\alpha_0^{(1)}][2^{n-1}]Q \\ &= \begin{cases} A^{(1)} & \text{if } \alpha_0^{(1)} = 0\,, \\ A^{(1)} + [2^{n-1}]Q & \text{if } \alpha_0^{(1)} = 1\,, \end{cases} \end{aligned}$$

and

$$\begin{aligned} P + [\alpha^{(2)}]Q + [\alpha^{(2)}][2^{n-1}]Q &= A^{(2)} + [\alpha_0^{(2)}][2^{n-1}]Q \\ &= \begin{cases} A^{(2)} & \text{if } \alpha_0^{(2)} = 0\,, \\ A^{(2)} + [2^{n-1}]Q & \text{if } \alpha_0^{(2)} = 1\,. \end{cases} \end{aligned}$$

Now, let $E'^{(k)} = E/\langle A^{(k)} + [2^{n-1}]Q\rangle$. We receive a response from the oracle with which we can recover the first bit of the two secrets. This is done by computing all the 4-neighbours $E'^{(k)}_\ell$ of $E^{(k)}_0$ ($\ell = 1, \dots, 6$), computing the hash of different combinations of the known $j(E^{(k)})$ and the j-invariants of the 4-neighbours $j_{k,\ell} = j\left(E'^{(k)}_\ell\right)$, and using them as oracle queries (see Figure 3). If the oracle returns 1, then we will be able to recover the bits. For example, if we find a match $h = H'(j(E^{(1)}_0), j_{2,4})$, we know that the $\alpha^{(1)}_0 = 0$ and $\alpha^{(2)}_0 = 1$. In the case of $\alpha^{(k)}_0 = 1$, we have also recovered the curve $E'^{(k)} = E'^{(k)}_4$ in the process.

This is summarised as Algorithm 1. Following this algorithm, there is a 1/4 chance that the attacker recovers the first bits after a single oracle query. However, as emphasised earlier, this is not the only information that we require: we also require the recovery of intermediate curves and points. Hence the 1/4 chance of early bit recovery still entails further computation and hence incurs further complexity to the attack.

$$\begin{tikzcd} {} \arrow[r, dashed] & E^{(1)}_2 \arrow[r, "\phi^{(1)}_2"] & E^{(1)}_1 \arrow[r, "\phi^{(1)}_1"] \arrow[dr] & E^{(1)}_0 \\ & & & E'^{(1)} \end{tikzcd} \qquad \begin{tikzcd} {} \arrow[r, dashed] & E^{(2)}_2 \arrow[r, "\phi^{(2)}_2"] & E^{(2)}_1 \arrow[r, "\phi^{(2)}_1"] \arrow[dr] & E^{(2)}_0 \\ & & & E'^{(2)} \end{tikzcd}$$

Figure 3: Branching at the first bit of the secrets

Algorithm 1: Recovering the first bits using $O(E, R, S, h')$.

Data: E, $R = P$, $S = [1 + 2^{n-1}]Q$, $E^{(1)}_0$, $E^{(2)}_0$
Result: $(\alpha^{(1)} \mod 2, \alpha^{(2)} \mod 2)$

Set $h_0 \leftarrow H'\left(j(E^{(1)}_0), j(E^{(2)}_0)\right)$;
if $O(E, R, S, h_0) = 1$ **then** Return $(0, 0)$;
for $i \leftarrow 1$ **to** 6 **do**
  Set $h \leftarrow H'(j(E^{(1)}_0), j(E'^{(1)}_i))$;
  if $O(E, R, S, h) = 1$ **then** Return $(0, 1)$;
end
for $i \leftarrow 1$ **to** 6 **do**
  Set $h \leftarrow H'(j(E'^{(1)}_i), j(E^{(2)}_0))$;
  if $O(E, R, S, h) = 1$ **then** Return $(1, 0)$;
end
Return $(1, 1)$;

Recovering the first intermediate curves

Having recovered the $\alpha^{(k)}_0$ bits, we would like to recover the curves $E'^{(k)}$. This is done using a branch, as in Figure 3 - if we have $E^{(k)}_0$ and $E'^{(k)}$, we can find $E^{(k)}_1$ simply by finding their common neighbour in the isogeny graph. If either of the bits $\alpha^{(k)}_0 = 1$, a branch would already have occurred and we would know $E'^{(k)}$. But if $\alpha^{(k)}_0 = 0$, we need to force a branch to happen. This is done by sending Alice the points $P + [2^{n-1}]Q$, $[1 + 2^{n-1}]Q$. One can verify that this indeed produces a branch, and again finding the common neighbour will reveal $E^{(k)}_1$. Additionally, we can recover $E^{(k)}_2$ since there are only three 2-isogenies from $E^{(k)}_1$, and two are known.

Pulling back A and Q's

We currently know

$$E^{(k)}_0, E^{(k)}_1, E^{(k)}_2, E'^{(k)}.$$

Given these curves, one computes the kernel subgroups of the 2-isogenies

$$\phi_2^{(k)} : E_2^{(k)} \to E_1^{(k)}, \qquad \phi_1^{(k)} : E_1^{(k)} \to E_0^{(k)}.$$

The kernels of the $\phi_1^{(k)}$ are generated by $A_1^{(k)} = \psi_1^{(k)}(A^{(k)})$ by Lemma 5.1. Also, we can obtain $[2^{n-1}]Q_1^{(k)} = \psi_1^{(k)}([2^{n-1}]Q)$ as the kernels of the duals $\widehat{\phi}_2^{(k)}$.

Now we set

$$\mathcal{A}_1^{(k)} = A_1^{(k)}, \qquad \mathcal{Q}_1^{(k)} = \left\{[2^{n-1}]Q_1^{(k)}\right\}$$

and proceed with $\mathcal{A}_1^{(k)}$ and $\mathcal{Q}_1^{(k)}$ to the main iterative step in §5.2 to recover the remaining bits of the secrets.

5.2 The Main Iterative Step

Now that we are set up with the bootstrap information from the previous section, we may proceed to recover the remaining $2(n-1)$ bits of Alice's secret $(\alpha^{(1)}, \alpha^{(2)})$. The rough outline of the attack is that for each additional pair of bits we proceed in two stages: A combination of the GPST attack (cf. § 3) to determine the bits $\alpha_i^{(1)}, \alpha_i^{(2)}$, and an additional step that pulls back generator points between intermediate curves and filters out invalid preimages.

We will need a few supporting lemmas, which we state now.

Lemma 5.2. *Let $T = A_{i-1}^{(k)}$ and $X = Q_{i-1}^{(k)}$ be the images of the kernel generators under the partial $2^{n-(i-1)}$-isogeny $\psi_{i-1}^{(k)} : E_n \to E_{i-1}^{(k)}$. Then the following holds:*

1. *The preimage of T under $\phi_i^{(k)}$ is given by*

$$\mathcal{T}_i = \left\{A_i^{(k)}, [1+2^{i-1}]A_i^{(k)}\right\}.$$

2. *The preimage of X under $\phi_i^{(k)}$ is given by*

$$\mathcal{X}_i = \left\{Q_i^{(k)}, Q_i^{(k)} + [2^{i-1}]A_i^{(k)}\right\}.$$

Proof. The preimage of T under $\phi_i^{(k)}$ is exactly given by $A_i^{(k)} + \ker \phi_i^{(k)}$ and since $\ker \phi_i^{(k)}$ is a 2-torsion, the result follows from Lemma 5.1 (1). The same argument holds for the preimage of X under $\phi_i^{(k)}$. □

Lemma 5.3. *Using the notation from Lemma 5.2, $\langle T\rangle = \langle[\lambda]T\rangle$ if λ is odd. Furthermore, all points in the pull-back of both $A_{i-1}^{(k)}$ and $[\lambda]A_{i-1}^{(k)}$ under $\phi_i^{(k)}$ generate the same subgroups.*

Proof. The first statement is clear.
From the previous lemma, we have that the preimage of $A_{i-1}^{(k)}$ under $\phi_{i-1}^{(k)}$ is $A_i^{(k)}$ and $[1+2^{i-1}]A_i^{(k)}$. Hence, using the first statement, we are done.

Next, the preimage of $[\lambda]A_{i-1}^{(k)}$, where λ is odd is given by $[\lambda]A_i^{(k)}$ and $[\lambda][1+2^{i-1}]A_i^{(k)}$, so the statement follows. □

Remark 2. For the ease of exposition we will assume that

$$K_i^{(1)} - K_i^{(2)} \notin 2\mathbb{Z}, \tag{1}$$

i.e. that the difference of the partial keys is odd. This is the case if and only if the first bits of the two secrets $\alpha_0^{(1)} \neq \alpha_0^{(2)}$. Special care has to be taken in case Equation (1) does not hold. We discuss this case in Section 5.3. See also the attack implementation for details.

Theorem 5.4 (2-SIDH Key Recovery)**.** *Let $\alpha^{(1)} - \alpha^{(2)}$ odd. Assume that E_n and $E_0^{(k)}$ are connected by a unique 2^n-isogeny for $k = 1, 2$. Assuming there exists an oracle $O(E_1, R_1, S_1, h)$ as in §4.1 (3), then there exists a polynomial-time algorithm that finds Alice's secrets in the 2-SIDH protocol.*

Proof. We are able to recover the first bits using §5.1, so the task is to recover the rest of the bits, which we will do by induction. The induction step is given by the following two stages and a boundedness argument showing that the algorithm is polynomial-time. The tuple (a, b) is either $(1, 2)$ or $(2, 1)$, distinguishing the two instances.

Stage 1 (Recovering Bits). Assume we have recovered the first i bits of each secret, and are attempting to recover the $(i+1)$-th bits.

Also assume we are given $\mathcal{A}_i^{(b)}$, a valid pull back of $A_{i-1}^{(b)}$, and $\mathcal{Q}_i^{(b)}$ a set (of size at most 2) of valid pull backs for $[\frac{1}{2}][2^{n-(i-1)}]Q_{i-1}^{(b)}$. Thus $\mathcal{A}_i^{(b)}$ and the points in $\mathcal{Q}_i^{(b)}$ are on the curve $E_i^{(b)}$.

First, we set

$$P' = P - [K_i^{(a)}][2^{n-(i+1)}]Q \quad \text{and} \quad Q' = [1 + 2^{n-(i+1)}]Q.$$

For each $Y \in [\frac{1}{2}]\mathcal{Q}_i^{(b)}$ we compute

$$E' = E/\langle \mathcal{A}_i^{(b)} + [K_i^{(b)} - K_i^{(a)}]Y \rangle.$$

For all 2-neighbours E'_ℓ of E' $(\ell = 1, 2, 3)$ we determine $j_{b,\ell} = j\,(E'_\ell)$. Also let $j_i = j(E_0^{(i)})$ be the j-invariants of the curves in Alice's public key. Note that we check the 2-neighbours of E' because the first step of the isogeny actually maps into $E_{i+1}^{(b)}$. Hence the isogeny from $E_i^{(b)}$, where the points originate from cannot reach the curve mapped from E. Therefore, we are forced to compute the 2-neighbours of E' to bridge that gap.

To recover the bit, we query the oracle on $O(E, P', Q', h)$ with $h = H'(j_{1,\ell}, j_2)$ (or $h = H'(j_1, j_{2,\ell})$, depending on the instance) for all $\ell = 1, 2, 3$. The oracle will have the following intermediate computations:

$$\begin{aligned} &P + [\alpha^{(a)}]Q - [K_i^{(a)}][2^{n-(i+1)}]Q + [K_i^{(a)}][2^{n-(i+1)}]Q + [\alpha_i^{(a)}][2^{n-1}]Q \\ &= A^{(a)} + [\alpha_i^{(a)}][2^{n-1}]Q \\ &= \begin{cases} A^{(a)} & \text{if } \alpha_i^{(a)} = 0\,, \\ A^{(a)} + [2^{n-1}]Q & \text{if } \alpha_i^{(a)} = 1\,, \end{cases} \end{aligned} \tag{2}$$

and

$$\begin{aligned} &P + [\alpha^{(b)}]Q - [K_i^{(a)}][2^{n-(i+1)}]Q + [K_i^{(b)}][2^{n-(i+1)}]Q + [\alpha_i^{(b)}][2^{n-1}]Q \\ &= A^{(b)} + [K_i^{(b)} - K_i^{(a)}][2^{n-(i+1)}]Q + [\alpha_i^{(b)}][2^{n-1}]Q \\ &= \begin{cases} A^{(b)} + [K_i^{(b)} - K_i^{(a)}][2^{n-(i+1)}]Q & \text{if } \alpha_i^{(b)} = 0\,, \\ A^{(b)} + [K_i^{(b)} - K_i^{(a)}][2^{n-(i+1)}]Q + [2^{n-1}]Q & \text{if } \alpha_i^{(b)} = 1\,, \end{cases} \end{aligned} \tag{3}$$

These points generate a subgroup which is the kernel of some isogeny. In the following, we have points in the intermediate curves that generate subgroups which is the kernel of an isogeny with the same codomain as the ones above.

$$\begin{array}{ll} \psi_i^{(b)}(A^{(b)}) + [K_i^{(b)} - K_i^{(a)}]\psi_i^{(b)}([2^{n-(i+1)}]Q) & \text{if } \alpha_i^{(b)} = 0\,, \\ \psi_i^{(b)}(A^{(b)}) + [K_i^{(b)} - K_i^{(a)}]\psi_i^{(b)}([2^{n-(i+1)}]Q) + \psi_i^{(b)}([2^{n-1}]Q) & \text{if } \alpha_i^{(b)} = 1\,. \end{array}$$

Hence, it can be checked that if any of the oracle outputs is 1, we must have that $\alpha_i^{(a)} = 0$ and $\alpha_i^{(a)} = 1$ otherwise. This yields the $(i+1)$-th partial keys

$$K_{i+1}^{(a)} = K_i^{(a)} + \alpha_i^{(a)} 2^i.$$

Stage 2 (Pulling Back Generators). Recall that Stage 1 used the point $\mathcal{A}_i^{(b)}$ and the set $\mathcal{Q}_i^{(b)}$. In this stage we determine valid pull backs of $A_i^{(b)}$ and $[\frac{1}{2}][2^{n-i}]Q_i^{(b)}$ which will yield $\mathcal{A}_{i+1}^{(b)}$ and $\mathcal{Q}_{i+1}^{(b)}$ for the next iteration. We note that from Lemma 5.1 (2), for any $Y \in \mathcal{Q}_i^{(b)}$ we have that $[2^{i-1}]Y = [2^{n-1}]Q_i^{(b)}$ determines the dual intermediate isogeny $\hat{\phi}_{i+1}^{(b)} : E_i^{(b)} \to E_{i+1}^{(b)}$ which allows us to compute the pull back.

First, we compute the preimages of $\mathcal{A}_i^{(b)}$ and the points $Y \in [\frac{1}{2}]\mathcal{Q}_i^{(b)}$ under the intermediate isogeny $\phi_{i+1}^{(b)} : E_{i+1}^{(b)} \to E_i^{(b)}$ and record them as candidates for $A_{i+1}^{(b)}$ and $[2^{n-(i+1)}]Q_{i+1}^{(b)}$, and call those sets $\mathcal{Z}_{i+1}^{(b)}$ and $\mathcal{Y}_{i+1}^{(b)}$. The size of $\mathcal{Z}_{i+1}^{(b)}$ is 2 and the size of $\mathcal{Y}_{i+1}^{(b)}$ is 16.

We now test the validity of the points in $\mathcal{Z}_{i+1}^{(b)}$ and in $\mathcal{Y}_{i+1}^{(b)}$. Similar to Stage 1, we set

$$P' = P - [K_{i+1}^{(a)}][2^{n-(i+1)}]Q \quad \text{and} \quad Q' = [1 + 2^{n-(i+1)}]Q.$$

Note that this time, we use the $(i+1)$-th partial keys $K_{i+1}^{(a)}, K_{i+1}^{(b)}$ that we recovered in Stage 1. For each $A \in \mathcal{Z}_{i+1}^{(b)}$ and for each $Y \in \mathcal{Y}_{i+1}^{(b)}$ we compute

$$j_b = j\left(E/\langle A + [K_{i+1}^{(b)} - K_{i+1}^{(a)}]Y\rangle\right) \tag{4}$$

and set $j_a = j\left(E^{(a)}\right)$. We then query the oracle on $O(E, P', Q', h)$ with $h = H'(j_1, j_2)$.

Using Equations (2) and (3), one can check that if the oracle output is 1, we may keep the tuple (A, Y) as valid data for the next round.

The following two claims complete the inductive hypothesis:

Claim 1. A valid tuple (A, Y) will be such that $A \in [\lambda]\mathcal{T}_{i+1}$ and $Y \in [\lambda]\mathcal{X}_{i+1}$ for some $\lambda \in (\mathbb{Z}/2^{i+1}\mathbb{Z})^*$, as defined in Lemma 5.2.

Claim 2. We can select any two (different) tuples $(A, Y_1), (A, Y_2)$ with a common A, and set $\mathcal{A}_{i+1}^{(b)} = A$ and $\mathcal{Q}_{i+1}^{(b)} = \{Y_1, Y_2\}$.

Boundedness. At the beginning of Stage 2 we quadruple the number of points in $\mathcal{Q}_i^{(b)}$ by computing $[\frac{1}{2}]\mathcal{Q}_i^{(b)}$ and then we double this number by determining the preimages of the halved points under the intermediate isogeny $\phi_{i+1}^{(b)} : E_{i+1}^{(b)} \to E_i^{(b)}$. Hence it is crucial to ensure that we reduce the total amount again.

To be precise, the set of pull backs of $[\frac{1}{2}][\lambda][2^{n-i}]Q_i^{(b)}$ for $\lambda \in (\mathbb{Z}/2^i\mathbb{Z})^*$ is given by

$$[\lambda][2^{n-(i+1)}]\begin{Bmatrix} Q_{i+1}^{(b)}, & [1+2^i]Q_{i+1}^{(b)}, \\ Q_{i+1}^{(b)} + [2^{i-1}]A_{i+1}^{(b)}, & [1+2^i]Q_{i+1}^{(b)} + [2^{i-1}]A_{i+1}^{(b)}, \\ Q_{i+1}^{(b)} + [2^{i}]A_{i+1}^{(b)}, & [1+2^i]Q_{i+1}^{(b)} + [2^{i}]A_{i+1}^{(b)}, \\ Q_{i+1}^{(b)} - [2^{i-1}]A_{i+1}^{(b)}, & [1+2^i]Q_{i+1}^{(b)} - [2^{i-1}]A_{i+1}^{(b)}, \end{Bmatrix}, \tag{5}$$

see Lemma 5.2.

Claim 1. Let $T = A_{i+1}^{(b)}$ and $X = [2^{n-(i+1)}]Q_{i+1}^{(b)}$ be the *correct* images of the kernel generators. Looking back at Equation (4), we find matching hashes (i.e. the oracle outputs 1) if and only if the following two subgroups are equal

$$\langle [\mu]T + [x]([\nu]X + [\xi]T)\rangle = \langle T + [x]X\rangle, \tag{6}$$

where $x = K_i^{(b)} - K_i^{(a)}$, and $\mu, \nu \in (\mathbb{Z}/2^{i+1}\mathbb{Z})^*$ and $\xi \in \mathbb{Z}/2^{i+1}\mathbb{Z}$. Here $\langle T + [x]X\rangle$ is the correct kernel, see Equation (3). In particular, ξ is a multiple of 2^{i-1} since a pullback of $[\frac{1}{2}][2^{n-i}]Q_i^{(b)}$ is determined up to an element of order 4, and so $\xi \in \{0, 2^i, \pm 2^{i-1}\} \pmod{2^{i+1}}$.

Without loss of generality, we may ignore any common scalar factor $\lambda \in (\mathbb{Z}/2^i\mathbb{Z})^*$. By assumption, x is odd and so Equation (6) implies that

$$\mu + x\xi = \nu \pmod{2^{i+1}}. \tag{7}$$

Lemma 5.2 and Equation (5) show that $\mu, \nu \in \{1, 1+2^i\}$. The only possible solutions to Equation (7) with $\mu, \nu \in \{1, 1+2^i\}$ are when $\xi \in \{0, 2^i\}$. Hence, for every $A \in \mathcal{Z}_{i+1}^{(b)}$ only two tuples (A, Y_1) and (A, Y_2) pass Stage 2, where each A corresponds to one choice of ν. ☐

Claim 2. This follows from Lemma 5.3 and Claim 1 because each choice of A corresponds to an odd scalar multiple of T which will generate the same kernel. Thus for a specific choice of A, we select the two corresponding tuples. These two tuples (A, Y_1) and (A, Y_2) represent the only two solutions to Equation (7). □

Initially, from §5.1 we have $\mathcal{Q}_1^{(k)}$ for $k = 1, 2$ constructed with only one $Q_1^{(k)}$, and the corresponding $\mathcal{A}_1^{(k)}$. By the inductive hypothesis above, $\mathcal{Q}_i^{(k)}$ remains at most size 2, completing the proof. □

The code referenced at the start of this work contains the implementation of this attack. As such, implementation details can be found in the code.

Remark 3. A simple verification can be performed using Weil pairings to detect this adaptive attack. This verification is delineated in Section 2.4 of [4]. However, this countermeasure can be circumvented with the use of a scalar θ just as described in Section 3.2 of [4]. One can put the map $[\theta]$ into appropriate parts of Section 5.1 and Section 5.2.

5.3 Forcing a Branch

In this section we will deal with the case when $\alpha^{(2)} - \alpha^{(1)}$ is arbitrary, and in particular even.

Theorem 5.5 (2-SIDH Key Recovery). *Assume that E_n and $E_0^{(k)}$ are connected by a unique 2^n-isogeny for $k = 1, 2$. Assuming there exists an oracle $O(E_1, R_1, S_1, h)$ as in §4.1 (3), then there exists a polynomial-time algorithm that finds Alice's secrets in the 2-SIDH protocol.*

Sketch of proof. Suppose that we are in the situation given by Stage 1 of Theorem 5.4, and so the aim is to recover the i-th bit of $\alpha^{(1)}$ and $\alpha^{(2)}$.

Stage 1. As in Stage 1 of Theorem 5.4, the attacker sets

$$P' = P - [K_i^{(1)}][2^{n-i}]Q \quad \text{and} \quad Q' = [1 + 2^{n-i}]Q.$$

Hence the attacker can determine the bits by querying the oracle with the information in Stage 1. In the case where $\alpha^{(2)} - \alpha^{(1)}$ is even, we do not need to check the 2-neighbours of E' since the first isogeny no longer maps into $E_{i+1}^{(k)}$. However, the guesses that the attacker makes must align with Equations (2) and (3).

Stage 2. The aim at this stage is the same as before: to find suitable images of points to use in the recovery of the next bit. The complication with having an even parity in $\alpha^{(2)} - \alpha^{(1)}$ is that the ambiguities in the pull-back of Q_*^* are killed by the even parity of the scalar. The solution is to query the oracle with the following points

$$P' = P - [K_{i+1}^{(a)} + y][2^{n-i}]Q \quad \text{and} \quad Q' = [1 + 2^{n-i}]Q.$$

for some odd y, and noting that we are using information from the previous stage. This results in the following intermediate computations:

$$A_i^{(1)} + [x][2^{n-i}]Q_i \text{ and } A_i^{(2)} + [x + K_{i+1}^{(2)} - K_{i+1}^{(1)}][2^{n-i-1}]Q_i.$$

We can then recover the candidates using the same procedure as before.

Boundedness. The boundedness argument from Theorem 5.4 carries over verbatim to this case, by adding on odd scalar y to x.

□

Note that this is the slower case. This is due to the fact that the implementation has to test the two branches simultaneously instead of testing each branch at a time. Refer to implementation for details.

5.4 Dividing the Work by Three

In this section, we will make an observation about the GPST adaptive attack if more information is recovered at each step.

For the simplicity of the following exposition, we will assume that the torsion subgroup used by Alice is of this particular form: $E[2^{3n}]$. Otherwise, bruteforcing one or two steps, respectively, of 2-isogenies allows us to force the remaining number of steps to be divisible by 3. The following diagram illustrates the isogeny path corresponding to the static key:

$$E_B \to E_{3n-1} \to E_{3n-2} \to \cdots \to E_2 \to E_1 \to E_{BA}$$

Note the order of the indices of the intermediate curves between E_B and E_{BA}. Also, we let the isogeny $\psi_i : E_B \to E_i$ be the isogeny from E_B to E_i, and let P and Q be points such that $\langle P, Q\rangle = E[2^{3n}]$.

Now, suppose, hypothetically, that such an attacker is also able to learn $[2^{3n-i}]Q_i$ along with the $(i-1)$-st bit of the secret at each stage of the attack. Then, such an attacker will only need to run the adaptive attack n times. This is because, at the n-th stage of the attack, the attacker would have recovered the following information:

$$\alpha \pmod{2^n}, \quad \text{and} \quad [2^{3n-i}]Q_i \text{ for } 0 \le i \le n\,.$$

Using $\alpha \pmod{2^n}$, the attacker would be able to compute

$$E_B \to E_{3n-1} \to E_{3n-2} \to \cdots \to E_{2n+1}.$$

Then, the parent curve of the points $[2^{3n-i}]Q_i$ for $0 \le i \le n$, would give us the curves (and the isogenies between them)

$$E_n \to E_{n-1} \to E_{n-2} \to \cdots \to E_{BA}.$$

Finally, using the point $[2^{2n}]Q_n$, we are able to obtain the path

$$E_{2n} \to E_{2n-1} \to E_{2n-2} \to \cdots \to E_n.$$

Indeed, this is because we have taken the convention that the kernel of $\phi : E_B \to E_{BA}$ is of the form $P + [\alpha]Q$ for some secret α, therefore $Q \notin \ker\phi$.

One can see that this exposition clearly applies to the extended adaptive attack described in the earlier parts of this section. Hence, an attack would be able to recover the entire secret with just a partial recovery.

6 Implementation

We have implemented the attack in `MAGMA` and code is available at the URL above in Section 1. Assuming a fixed value of k, for example $k = 2$ in this attack, the algorithm performs $O(n)$ isogeny computations, where n is the bitlength of the secrets. We can reduce the number of isogeny computations by a factor of three using the observation in Section 5.4. Most of the overhead in the implementation is due to the computation of isogenies, which can be optimised further (see [3]).

We remark that the starting curve of the attack E_B is unimportant to the attack but Bob should choose this curve to be 3^m-isogenous from E_0 as the points $\phi_B(P_A)$ and $\phi_B(Q_A)$ he sends to Alice can be used to detect an attack via Weil pairings. In the implementation we choose to work with E_0, but as mentioned, this is only for the ease of programming.

7 Extensions

In this section we briefly discuss extensions of our attack: When $k \ge 3$ and the case of attacking Bob or other participants with primes not equal to 2.

7.1 Towards k-SIDH for $k \geq 3$

We have successfully demonstrated an attack on 2-SIDH, but we are now interested in generalising this attack to higher k, and finding the complexity w.r.t. k. First we observe the brute-force of the first bit of each secret. We generalise the hash function H' to k inputs, and for each index i in H', we must iterate over all combinations of the six 4-neighbours of $E_0^{(i)}$ (which we denote as $j_{i,\ell}$ for $\ell = 1, \ldots, 6$ as above), as well as the curve $E_0^{(i)}$ itself, giving 7 choices. This gives a complexity just for the first bit of $O(7^k)$ - exponential in k. With this complexity, we suggest even $k = 46$ would give sufficient security against this attack, a lower bound than that suggested by [1].

The complexity of the main iterative step is more interesting, though, because the first bit brute force is a naive attack and we expect it to scale worse. The best case for the attacker is when one of the secrets has a different parity to all of the other secrets (this corresponds to the "easier" case for $k = 2$). WLOG assume that it is $\alpha^{(1)}$ which has a different parity than all the $\alpha^{(2)}, ..., \alpha^{(k)}$. The attacker will send $P' = P - [K^{(1)}][2^{n-i-1}]Q$, $Q' = [1 + 2^{n-i-1}]Q$. Alice will compute, for each $y = 1, ..., k$

$$\begin{cases} A^{(y)} + [K_i^{(y)} - K_i^{(1)}][2^{n-(i+1)}]Q & \text{if } \alpha_i^{(y)} = 0\,, \\ A^{(y)} + [K_i^{(y)} - K_i^{(1)}][2^{n-(i+1)}]Q + [2^{n-1}]Q & \text{if } \alpha_i^{(y)} = 1\,, \end{cases} \tag{8}$$

with corresponding intermediate points given by

$$\begin{cases} \psi_i^{(y)}(A^{(y)}) + [K_i^{(y)} - K_i^{(1)}]\psi_i^{(y)}([2^{n-(i+1)}]Q) & \text{if } \alpha_i^{(y)} = 0\,, \\ \psi_i^{(y)}(A^{(y)}) + [K_i^{(y)} - K_i^{(1)}]\psi_i^{(y)}([2^{n-(i+1)}]Q) + \psi_i^{(y)}([2^{n-1}]Q) & \text{if } \alpha_i^{(y)} = 1\,. \end{cases}$$

Now for each y, as before, we will have to pull back candidates $\mathcal{A}_i^{(y)}$ and $\mathcal{Q}_i^{(y)}$, and use oracle queries to test each combination of j-invariants from each of the candidates. As before, for each $Y^{(y)} \in [\frac{1}{2}]\mathcal{Q}_i^{(y)}$ we compute

$$E^{(y)'} = E/\langle A^{(y)} + [K_i^{(y)} - K_i^{(1)}]Y^{(y)}\rangle.$$

Using the j-invariants of these curves, we calculate

$$H'(j(E_0^{(1)}), j(E^{(2)'}), ..., j(E^{(k)'})).$$

Because $|\mathcal{Q}_i^{(y)}| = 16$ (all preimages of division points of the two candidates), this will result in 16^{k-1} different combinations in the hash function, meaning the attack uses $O(16^k)$ oracle queries.

Of course, the number of oracle queries required is greatly reduced if we instead operate in attack model (1) of Section 4.1. With access to an oracle which returns the hash value directly to the attacker, the hash comparisons can be done locally without further queries. Thus the number of oracle queries in this model is $O(n)$ (or $O(n/3)$ by using the observation in Section 5.4), but the work required locally by the attacker still grows exponentially in k.

7.2 Attacking Bob

The attack can be generalised to Bob's computations using torsion points with orders a power of three. The difference is that each step of the attack recovers not a binary bit, but a ternary bit instead.

We can see that the number of 3-neighbours is twelve compared to the six 2-neighbours in the case of $\ell = 2$. Hence, the recovery of the first ternary bit would require 12 hash comparisons compared to 6 when $\ell = 2$. Extending this to beyond 3, we have that the number of hash comparisons required is $\ell(\ell + 1)$.

In the intermediate stages of the key recovery we will also need to deal with a larger number of possible pull-back points as the ℓ-torsion subgroups grow in number of elements. In the case when $\ell = 3$, the number of possible pull-back points through the 3-isogeny and the multiplication by 3 is 27. The Boundedness argument in Theorem 5.4 still holds but with minor changes to the scalars μ, ν, and ξ, where $\mu, \nu \in \{1, 1 \pm 3^i\}$, and $\xi \in \{0, \pm 3^i, \pm 2 \cdot 3^{i-1}, \pm 4 \cdot 3^{i-1}\}$. Following the same arguments, the boundedness argument should still hold. Furthermore, although the same assumptions on the co-primality of the scalars must still be made, the same work-arounds will still work when $\ell = 3$. We leave a detailed analysis as further work.

Acknowledgments

We thank the reviewers for their helpful suggestions. This research was supported by the Royal Society of New Zealand Marsden project UOA1630, and the Ministry of Business, Innovation & Employment Catalyst project UOAX1933; a Michael Smith Foreign Study Supplement from the Natural Sciences and Engineering Research Council of Canada, and a David Johnston International Experience Award from the University of Waterloo.

References

[1] Reza Azarderakhsh, David Jao, and Christopher Leonardi. Post-quantum static-static key agreement using multiple protocol instances. In *SAC 2017*, pages 45–63. Springer, 2017.

[2] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: An efficient post-quantum commutative group action. In *ASIACRYPT 2018*, pages 395–427. Springer, 2018.

[3] Luca De Feo, David Jao, and Jérôme Plût. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. *J. Mathematical Cryptology*, 8(3):209–247, 2014.

[4] Steven D. Galbraith, Christophe Petit, Barak Shani, and Yan Bo Ti. On the security of supersingular isogeny cryptosystems. In *ASIACRYPT 2016*, pages 63–91. Springer, 2016.

[5] Dennis X. Charles, Kristin E. Lauter, and Eyal Z. Goren. Cryptographic Hash Functions from Expander Graphs. *J. Cryptology*, 22:93–113, 2009.

[6] David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In *PQCrypto 2011*, pages 19–34. Springer, 2011.

[7] Selçuk Kayacan. A note on the static-static key agreement protocol from supersingular isogenies. Cryptology ePrint Archive, Report 2019/815, 2019. https://eprint.iacr.org/2019/815.

[8] Jacques Vélu. Isogénies entre courbes elliptiques. *C. R. Acad. Sci. Paris Sér. A-B*, 273:A238–A241, 1971.

A Adaptive Attack with Points of Small Order

In this section, we show an attack that works, but can be foiled by a simple check on the order of the points.

Again, we will suppose that Alice's secret is $(\alpha^{(1)}, \alpha^{(2)})$, and our strategy in this section is to recover the secrets by sending points P and $[2^k]Q$.

To recover the first bits, we do the following: First, we send points $P, [2^{n-1}]Q$ to the oracle and receive a response. Notice that the oracle would compute the following values:

$$P + [\alpha^{(1)}][2^{n-1}]Q = \begin{cases} P & \text{if } \alpha_0^{(1)} = 0\,, \\ P + [2^{n-1}]Q & \text{if } \alpha_0^{(1)} = 1\,, \end{cases}$$
$$P + [\alpha^{(2)}][2^{n-1}]Q = \begin{cases} P & \text{if } \alpha_0^{(2)} = 0\,, \\ P + [2^{n-1}]Q & \text{if } \alpha_0^{(2)} = 1\,. \end{cases}$$

Since we know E and P and Q, we are able to work out the two points

$$P, P + [2^{n-1}]Q$$

and the elliptic curves E' and E'', which are the codomain for isogenies whose kernels are generated by the two points. We are also able to compute the four hash values given by $h(E', E')$, $h(E', E'')$, $h(E'', E')$, $h(E'', E'')$. By comparing the response from the oracle and the four values, we are able to determine the first bits of $\alpha^{(1)}, \alpha^{(2)}$.

Now, suppose we would like to recover the $(i-1)$-st bit of each secret (i.e. $\alpha_i^{(1)}, \alpha_i^{(2)}$). We send the points P and $[2^{n-i-1}]Q$ to the oracle and receive the response. The oracle would compute

$$P + [\alpha^{(1)}][2^{n-i-1}]Q = \begin{cases} P + [K_1]Q & \text{if } \alpha_i^{(1)} = 0\,, \\ P + [K_1]Q + [2^{n-1}]Q & \text{if } \alpha_i^{(1)} = 1\,, \end{cases}$$
$$P + [\alpha^{(2)}][2^{n-i-1}]Q = \begin{cases} P + [K_2]Q & \text{if } \alpha_i^{(2)} = 0\,, \\ P + [K_2]Q + [2^{n-1}]Q & \text{if } \alpha_i^{(2)} = 1\,. \end{cases}$$

Since we have P, Q and K_1, K_2, we are able to compute all four points and obtain the elliptic curve mapped by the isogeny whose kernel is generated by these points. We compare the four values with the value obtained from the oracle and conclude the two bits $\alpha_i^{(1)}, \alpha_i^{(2)}$.