# An adaptive attack on Genus-2 SIDH

Sabrina Kunzweiler[1], Yan Bo Ti[2], and Charlotte Weitkämper[3]

[1] Ruhr-Universität Bochum, Germany
[2] DSO, Singapore
[3] University of Birmingham, UK

**Abstract.** We present a polynomial-time adaptive attack on the genus-2 variant of the SIDH protocol (G2SIDH) and describe an improvement to its secret selection procedure. G2SIDH is a generalisation of the Supersingular Isogeny Diffie–Hellman key exchange into the genus-2 setting which was proposed by Flynn and Ti. G2SIDH is able to achieve the same security as SIDH while using fields a third of the size.

We give a thorough analysis of the keyspace of G2SIDH and achieve an improvement to the secret selection by using symplectic bases for the torsion subgroups. This allows for the near uniform sampling of secrets without needing to solve multiple linear congruences as suggested by Flynn–Ti.

The proposed adaptive attack on G2SIDH is able to recover the secret when furnished with an oracle that returns a single bit of information. We ensure that the maliciously generated information provided by the attacker cannot be detected by implementing simple countermeasures such as checking the Weil pairing or order of the given points. We demonstrate this attack and show that it is able to recover the secret isogeny in all cases of G2SIDH using a symplectic basis before extending the strategy to arbitrary bases.

**Keywords:** Genus 2 SIDH · isogenies · adaptive attack

The Supersingular Isogeny Diffie–Hellman (SIDH) protocol is a key exchange protocol which is the basis of a third round alternative candidate in the NIST post-quantum cryptographic standardisation process [10]. The SIDH protocol was first described in 2011 by Jao and De Feo [7]. The G2SIDH key exchange [5] is a natural generalisation of SIDH to a higher dimensional setting. In this variant, the supersingular elliptic curves of SIDH are substituted with principally polarised superspecial abelian surfaces (PPSSAS) and $\ell$-isogenies are replaced by $(\ell, \ell)$-isogenies. Due to the increased number of neighboring isogenies, the security of G2SIDH can be maintained while using primes a third of the size when compared with SIDH.

In most aspects, G2SIDH closely resembles SIDH. This leads to natural generalisations of attacks on SIDH to G2SIDH, and also the concept of equivalences of secret keys. In [5], the authors noted that they expected attacks on SIDH to generalise naturally to G2SIDH. One of the contributions of this paper is to demonstrate an adaptive attack on G2SIDH which is similar, but not the same as the Galbraith–Petit–Shani–Ti (GPST) attack [6] since a straightforward adaptation thereof fails due to the difference in types of kernel subgroups for SIDH-isogenies and those in G2SIDH.

Adaptive chosen ciphertext attacks work by recovering the secret key from a decryption oracle by sending the oracle adaptively chosen inputs. Such an attack on isogeny-based cryptosystems was first introduced by Galbraith et al. in 2016 [6]. This attack only requires that a decryption oracle returns a single bit of information at a time instead of returning the decryption of the input. In fact, the decryption oracle can be viewed as a decisional Diffie–Hellman protocol, and the adaptive attack can be seen as the reduction of the computational Diffie–Hellman problem to the decisional Diffie–Hellman problem.

More practically, the use of a weaker oracle demonstrates the potency of the attack. The GPST attack meant that non-interactive key exchange implementations of SIDH are no longer secure and can only be safely used with CCA2-protections. As noted in [4], a different cryptosystem will necessitate modifications in the adaptive attack. This continues to hold true in the adaptive attack on G2SIDH.

The authors of [5] claimed that an adaptive attack which can break a static-key implementation of G2SIDH should exist. The implications of the existence of such an attack on G2SIDH would be the same as the GPST attack had on SIDH. Namely the reduction of the computational Diffie–Hellman problem to the

decisional Diffie–Hellman problem, and that static keys are insecure without CCA2-protections. However, such an attack is not found in the state-of-the-art.

The main difference between our adaptive attack and the SIDH attack lies in the number of secret scalars and the number of kernel generators associated with each cryptosystem. The original description of G2SIDH called for multiple secret scalars that need to fulfil a certain linear congruence property. One can immediately see that the number of secret scalar combinations exceeds the number of admissible isogenies. Furthermore, the original description of G2SIDH requires that the user solves these linear congruences during the selection of secrets. This is cumbersome and increases the computational costs of key exchange during run time.

We further propose a set of generators for the torsion points that allow for a straightforward selection of secrets. This description also lends us a framework to describe the adaptive attack. The set of generators of torsion points is called the *symplectic basis*. This allows both parties of the key exchange protocol to choose secrets uniformly from a large keyspace simply by choosing 3 to 4 scalars instead of 12.

### Contributions

In this paper we present the two results:

- Use of a symplectic basis which allows for the secret selection to be performed without needing to solve cumbersome linear modular equations. This provides a framework for the G2SIDH attack to be carried out. Furthermore, we propose a simplified version of secret selection that allows for the uniform selection of keys from a restricted (but large enough) keyspace. We also propose a key compression technique that is conceptually similar to the SIDH key compression.
- An adaptive attack on G2SIDH that recovers the secret kernel when provided with an oracle that returns a single bit of information. This attack will be presented with the assumption that the users are using a symplectic basis. However, we will also show how this attack can be extended to users using an arbitrary basis by recovering equivalent keys in a symplectic basis. Finally, this adaptive attack is able to by-pass simple countermeasures such as Weil pairing and order checking.
  The only countermeasure that we are aware of is to implement CCA2-protections such as the Fujisaki–Okamoto transformation.

### Outline

This paper is organised as follows. We give a brief introduction to principally polarised supersingular and superspecial abelian surfaces and the genus-2 variant of SIDH utilising these varieties in Section 1. For comparison, the original SIDH protocol for elliptic curves is recalled in Appendix A.1. In Section 2, we analyze the G2SIDH keyspace and suggest a slight restriction thereof to allow for easier uniform sampling. The adaptive attack is then presented in Section 3 where we first sketch an algorithm to determine the type of secret subgroup Alice is using, and then give a detailed description of the strategy for recovering secret keys corresponding to a certain form of kernel subgroup. We then generalise our findings. For a discussion of parallels to SIDH and an examination of the GPST attack on elliptic curve SIDH [6] with respect to our attack framework, we refer to Appendices A.2 and A.3. More details on the type distinction algorithm for valid secret subgroups can be found in Appendix C.

## 1 Preliminaries

Traditionally, isogeny-based cryptography considers isogenies between (certain types) of elliptic curves. Elliptic curves are abelian varieties of dimension one which are principally polarised, though the polarisation is not usually of concern when cryptography is instantiated with elliptic curves. It is thus natural to consider generalising isogeny-based cryptography by broadening the scope to isogenies between principally polarised abelian varieties of higher dimensions. In particular, G2SIDH is a protocol which adapts SIDH to using principally polarised abelian surfaces.

In this section, we will first give a brief introduction to PPSSAS following [8], and isogeny-based cryptography using principally polarised abelian varieties of dimension two instead of elliptic curves when describing G2SIDH as in [5]. We assume that the reader is familiar with the general concept of elliptic curves, but refer to [11], for example, for an in-depth discussion of this topic.

### 1.1 PPSSAS

Let $p$ and $\ell$ be distinct primes, let $n$ be a positive integer. Further, let $A$ be an abelian surface defined over some finite field $\mathbb{F}_q$ of characteristic $p$, i.e. an abelian variety of dimension two. Then an *isogeny* is a homomorphism between two abelian surfaces which is surjective and has a finite kernel.

In order to obtain a higher-dimensional analogue of an elliptic curve, we need to consider *principally polarised abelian surfaces* (PPAS). A *polarisation* of $A$ is an isogeny $\lambda : A \to A^\vee$, where $A^\vee$ is the dual variety of $A$, derived from some ample divisor of $A$. This polarisation is called *principal* if the isogeny is an isomorphism of varieties. If a principally polarised abelian surface $A/\mathbb{F}_q$ is isogenous to a product of supersingular elliptic curves over $\mathbb{F}_q$, we consider $A$ to be *supersingular*. If $A$ is isomorphic to a product of supersingular elliptic curves, we call $A$ a principally polarised *superspecial* abelian surface (PPSSAS).

As shown in [5, Thm. 1], any PPSSAS $A$ defined over $\overline{\mathbb{F}}_p$ is either isomorphic to the Jacobian of a smooth hyperelliptic curve of genus two, or to the product of two elliptic curves. This implies that $A$ can be explicitly represented either by the equation $y^2 = f(x)$ for some polynomial $f \in \overline{\mathbb{F}}_p[x]$ of degree 5 or 6 representing a genus-2 curve, or as the product of two elliptic curves defined by the equations $y^2 = g(x)$ and $y^2 = h(x)$ for some degree-3 polynomials $g$ and $h$. We represent the $\overline{\mathbb{F}}_q$-isomorphism class of some PPSSAS $A$ in the genus-2 setting by any valid isomorphism invariant.

As with elliptic curves, there exists a non-degenerate, alternating pairing on any abelian surface $A/\mathbb{F}_q$, the *Weil pairing*

$$e_{\ell^n} : A[\ell^n](\overline{\mathbb{F}}_q) \times A^\vee[\ell^n](\overline{\mathbb{F}}_q) \to \boldsymbol{\mu}_{\ell^n}$$

where $A[\ell^n](\overline{\mathbb{F}}_q)$ denotes the $\ell^n$-torsion group of $A$ and $\boldsymbol{\mu}_{\ell^n}$ denotes the group of $\ell^n$-th roots of unity. If $A$ is a PPAS, we can use the isomorphism $A \simeq A^\vee$ to obtain the pairing $e_{\ell^n} : A[\ell^n](\overline{\mathbb{F}}_q)^2 \to \boldsymbol{\mu}_{\ell^n}$. This pairing allows us to examine the correspondence between subgroups and isogenies of abelian surfaces which preserve the principal polarisation.

**Definition 1 (Maximal $\ell^n$-isotropic subgroup).** *Let $A$ be an abelian variety and $K$ a subgroup of $A[\ell^n]$. Then we call $K$ a* maximal $\ell^n$-isotropic subgroup *if*

- *$K \not\subset A[m]$ for any $m < \ell^n$,*
- *the $\ell^n$-Weil pairing (on $A[\ell^n]$) restricts trivially to $K$, and*
- *$K$ is a maximal subgroup with respect to the Weil pairing-property above.*

As shown by Flynn and Ti, principal polarisations of PPAS are preserved under isogenies whose kernel is a maximal $\ell^n$-isotropic subgroup, and such subgroups can be specified as follows.

**Proposition 1 ([5, Prop. 2]).** *Let $K \subset A$ be a maximal $\ell^n$-isotropic subgroup. Then $K$ is isomorphic to*

$$\mathcal{C}_{\ell^n} \times \mathcal{C}_{\ell^n} \quad \text{or} \quad \mathcal{C}_{\ell^n} \times \mathcal{C}_{\ell^{n-k}} \times \mathcal{C}_{\ell^k}$$

*for some $1 \leq k \leq \lfloor \frac{n}{2} \rfloor$.*

Note in particular that, in comparison to kernels usually considered for elliptic curve isogenies, $K$ is not cyclic.

Hence, a maximal $\ell^n$-isotropic subgroup of $A$ defines a sequence of $n$ $(\ell, \ell)$-isogenies between PPAS, each defined by a kernel generated by two order-$\ell$ elements.

### 1.2 G2SIDH

The G2SIDH key exchange scheme is a natural generalisation of SIDH to dimension two. The key exchange scheme requires the selection of a prime of the form $p = 2^{e_A} \cdot 3^{e_B} \cdot f - 1$ where $2^{e_A} \approx 3^{e_B}$ and $f$ is a small cofactor not divisble by 2 or 3. A principally polarised superspecial abelian surface is then chosen to be the base abelian variety. This is achieved by first considering the hyperelliptic curve

$$H : y^2 = x^6 + 1\,.$$

It can be shown that $H$ is a double cover of the supersingular elliptic curve $y^2 = x^3 + 1$. Hence the Jacobian $J_H$ of $H$ is indeed a PPSSAS. Then a short random walk is taken from $J_H$ in the $(2,2)$-isogeny graph to obtain a random PPSSAS $J$. Moreover $\#J(\mathbb{F}_{p^2}) = (p+1)^4$ and in particular $J(\mathbb{F}_{p^2}) = J[2^{e_A}] \times J[3^{e_B}] \times J[f]$ as a group.

Fixing the torsion-bases $\langle P_1, P_2, P_3, P_4\rangle = J[2^{e_A}]$ and $\langle R_1, R_2, R_3, R_4\rangle = J[3^{e_B}]$, we have that the points $P_i$ and $R_i$ are all defined over $\mathbb{F}_{p^2}$ by our choice of the characteristic of the field.

To perform the key exchange, Alice chooses her secret isogeny $\phi_A$ such that the kernel of the isogeny has order $2^{2e_A}$. The kernel of $\phi_A$ will be determined by a secret linear combination of the $P_i$ known only to Alice and determined by a 12-tuple of secret scalars. Let $J_A$ be the codomain of $\phi_A$. She sends the tuple

$$\big(J_A, \phi_A(R_1), \phi_A(R_2), \phi_A(R_3), \phi_A(R_4)\big)$$

to Bob.

Bob analogously completes his side of the computation and Alice thus receives the tuple

$$\big(J_B, \phi_B(P_1), \phi_B(P_2), \phi_B(P_3), \phi_B(P_4)\big)\,.$$

She can then use her linear combination of secret scalars generating the kernel of her secret isogeny $\phi_A$ to compute an isogeny from $J_B$ using $\phi_B(P_i)$ as the basis instead of $P_i$. Denote the codomain of this isogeny by $J_{AB}$. Bob will complete his side of the protocol and obtain the abelian surface $J_{BA}$. By construction, $J_{AB}$ and $J_{BA}$ are isomorphic as principally polarised abelian surfaces and thus also the corresponding hyperelliptic curves are isomorphic via Torelli's theorem [9, Cor. 12.2]

We have deliberately omitted the discussion of secret linear combinations of the basis points. We will revisit this when we present our findings in the next section, especially §2.1 and §2.4, when discussing a more efficient method of secret selection.

## 2 Keyspace

A secret key of Alice is an isogeny of principally polarised abelian varieties $\phi_A : J \to J_A$. We have seen in §1.1 that this isogeny corresponds to a maximal $2^{e_A}$-isotropic subgroup of $J$. In the same way, Bob's secret key corresponds to a maximal $3^{e_B}$-isotropic subgroup of $J$. This allows us to identify the keyspace with the set of maximal isotropic subgroups. We set

$$\mathcal{K}_\ell = \{G \subset J \mid G \text{ maximal } \ell^n\text{-isotropic}\} \quad \text{for } \ell \in \{2,3\} \text{ and } n = e_A \text{ or } e_B, \text{ respectively.}$$

In this section we analyze the set $\mathcal{K}_\ell$ and show how to (almost) uniformly sample from the entire keyspace. Moreover, we introduce the subset $\mathcal{K}_\ell^{\text{res}} \subset \mathcal{K}_\ell$. This subset is of the same order of magnitude as the entire keyspace, and allows for very simple and truly random sampling. An important step in the analysis is the normalisation of secret keys that allows to define classify the groups in $\mathcal{K}_\ell$ and define canonical generators. This is achieved by considering so-called symplectic bases of the $\ell^n$-torsion of $J$.

### 2.1 Secret sampling in the original G2SIDH

Let us briefly recall the method of sampling secrets suggested in the original G2SIDH protocol [5]. In particular, given a PPSSAS $J$ and a basis $(P_1, \dots, P_4)$ for $J[\ell^n]$ the authors describe a method to generate a secret maximal isotropic subgroup of the $\ell^n$-torsion which by Proposition 1 must be of the form

$$\mathcal{C}_{\ell^n} \times \mathcal{C}_{\ell^{n-k}} \times \mathcal{C}_{\ell^k}$$

for some $0 \leq k \leq \lfloor \frac{n}{2} \rfloor$.

First, a random $k \in [\lfloor \frac{n}{2} \rfloor + 1]$ is chosen. Note that to sample from all possible maximal $\ell^n$-isotropic subgroups uniformly, $k$ cannot be picked uniformly. We give more detail about the distribution of different values of $k$ in §2.4.

Next, four scalars $\alpha_{1,1}, \dots, \alpha_{1,4} \in [\ell^n]$ are picked at random. Since these scalars will produce the first kernel generator of full order $\ell^n$, at least one of the scalars must not be divisible by $\ell$. Finally, the remaining scalars $\alpha_{i,1}, \dots, \alpha_{i,4}$ for $i \in \{2, 3\}$ are found by solving the following linear congruences.

$$\begin{pmatrix} \alpha_{1,1}\alpha_{2,2} - \alpha_{1,2}\alpha_{2,1} & +x_{1,3}(\alpha_{1,1}\alpha_{2,3} - \alpha_{1,3}\alpha_{2,1}) & +x_{1,4}(\alpha_{1,1}\alpha_{2,4} - \alpha_{1,4}\alpha_{2,1}) \\ +x_{2,3}(\alpha_{1,2}\alpha_{2,3} - \alpha_{1,3}\alpha_{2,2}) & +x_{2,4}(\alpha_{1,2}\alpha_{2,4} - \alpha_{1,4}\alpha_{2,2}) & +x_{3,4}(\alpha_{1,3}\alpha_{2,4} - \alpha_{1,4}\alpha_{2,3}) \end{pmatrix} \equiv 0 \pmod{\ell^k}$$

and

$$\begin{pmatrix} \alpha_{1,1}\alpha_{3,2} - \alpha_{1,2}\alpha_{3,1} & +x_{1,3}(\alpha_{1,1}\alpha_{3,3} - \alpha_{1,3}\alpha_{3,1}) & +x_{1,4}(\alpha_{1,1}\alpha_{3,4} - \alpha_{1,4}\alpha_{3,1}) \\ +x_{2,3}(\alpha_{1,2}\alpha_{3,3} - \alpha_{1,3}\alpha_{3,2}) & +x_{2,4}(\alpha_{1,2}\alpha_{3,4} - \alpha_{1,4}\alpha_{3,2}) & +x_{3,4}(\alpha_{1,3}\alpha_{3,4} - \alpha_{1,4}\alpha_{3,3}) \end{pmatrix} \equiv 0 \pmod{\ell^{n-k}},$$

where $x_{i,j}$ denotes the integer such that $e_{\ell^n}(P_i, P_j) = e_{\ell^n}(P_1, P_2)^{x_{i,j}}$ for $i, j \in \{1, \dots, 4\}$ and $e_{\ell^n}(P_1, P_2)$ a primitive $\ell^n$-th root of unity.

This produces the tuple $(\alpha_{1,1}, \dots, \alpha_{3,4})$ of secret scalars defining the generators of the kernel $G = \langle G_1, G_2, G_3 \rangle$ corresponding to the secret isogeny as

$$G_1 = \sum_{i=1}^{4} [\alpha_{1,i}]P_i, \quad G_2 = \sum_{i=1}^{4} [\alpha_{2,i}]P_i, \text{ and } G_3 = \sum_{i=1}^{4} [\alpha_{3,i}]P_i\,.$$

By construction, the points $G_1, G_2$ and $G_3$ are of order $\ell^n, \ell^{n-k}$ and $\ell^k$, respectively, and satisfy the necessary Weil pairing-conditions $e_{\ell^n}(G_1, G_2) = e_{\ell^n}(G_1, G_3) = e_{\ell^n}(G_2, G_3) = 1$.

This implies that $\langle G_1 \rangle \simeq \mathcal{C}_{\ell^n}$, $\langle G_2 \rangle \simeq \mathcal{C}_{\ell^{n-k}}$, $\langle G_3 \rangle \simeq \mathcal{C}_{\ell^k}$ and that $G = \langle G_1, G_2, G_3 \rangle$ is isotropic. However, the congruences above do not always guarantee that $G$ is isomorphic to $\mathcal{C}_{\ell^n} \times \mathcal{C}_{\ell^{n-k}} \times \mathcal{C}_{\ell^k}$. In the worst case, one could have chosen $\alpha_{2,1}, \dots, \alpha_{3,4}$ such that $G_2 = \ell^k G_1$ and $G_3 = \ell^{n-k} G_1$. In this case $G = \langle G_1 \rangle$ is a cyclic group of order $\ell^n$, and in particular not maximal $\ell^n$-isotropic. This issue is not addressed in [5], but it is fixed using the selection process that we describe in the subsequent sections.

### 2.2 Symplectic basis

Let $m$ be an integer not divisible by $p$. The $m$-torsion of $J$ is a finitely generated group of rank 4, more precisely

$$J[m] \;\xrightarrow{\sim}\; (\mathbb{Z}/m\mathbb{Z})^4.$$

**Definition 2.** *We say that a set $(P_1, P_2, Q_1, Q_2)$ is a* basis *for $J[m]$ if it generates $J[m]$ as a group. We say that the basis $(P_1, P_2, Q_1, Q_2)$ for $J[m]$ is* symplectic *with respect to the Weil pairing if*

$$e_m(P_i, Q_j) = \zeta^{\delta_{ij}}, \quad e_m(P_1, P_2) = e_m(Q_1, Q_2) = \zeta^0 = 1,$$

*where $\zeta$ is some primitive $m$-th root of unity and $\delta_{ij}$ denotes the Kronecker delta.*

Another way of phrasing the definition is to say that a basis is symplectic if the associated pairing matrix is of the form

$$(\log(\zeta, e_m(P,Q)))_{P,Q\in\{P_1,P_2,Q_1,Q_2\}} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix},$$

where the logarithm is taken with respect to $\zeta$.

There always exists a symplectic basis for $J[m]$. Indeed, given any basis for $J[m]$ it can be easily transformed into a symplectic basis using Algorithm 1 which is presented in Appendix B.

Finally, symplectic bases are preserved under isogenies as shown in the lemma to come. This allows us to use symplectic bases in G2SIDH.

**Lemma 1.** *Let $(P_1, P_2, Q_1, Q_2)$ be a symplectic basis of $J[m]$, and let $\phi : J \to J'$ be an isogeny whose degree is coprime to $m$. Then $\big(\phi(P_1), \phi(P_2), \phi(Q_1), \phi(Q_2)\big)$ is a symplectic basis of $J'[m]$.*

*Proof.* Observe that we have

$$e_m(\phi(P_i), \phi(Q_j)) = e_m(P_i, Q_j)^{\deg\phi} = 1 \text{ and } e_m(\phi(Q_i), \phi(P_j)) = e_m(P_j, Q_i)^{-\deg\phi} = 1$$

for all $i \neq j$. Likewise, we have that

$$e_m(\phi(P_i), \phi(Q_i)) = e_m(P_i, Q_i)^{\deg\phi} = \zeta^{\deg\phi}$$

for $i = 1, 2$. Finally, since $\langle \phi(P_1), \phi(P_2), \phi(Q_1), \phi(Q_2)\rangle = J'[m]$ and the torsion subgroup is of rank 4, we can conclude that $\big(\phi(P_1), \phi(P_2), \phi(Q_1), \phi(Q_2)\big)$ is a symplectic basis of $J'[m]$. □

### 2.3 Classification of secret keys

Let $(\alpha_{1,1}, \ldots, \alpha_{3,4})$ be the secret scalars defining the group $G = \langle G_1, G_2, G_3\rangle \in \mathcal{K}_\ell$, where

$$G_1 = \sum_{i=1}^{4} [\alpha_{1,i}]P_i, \quad G_2 = \sum_{i=1}^{4} [\alpha_{2,i}]P_i, \text{ and } G_3 = \sum_{i=1}^{4} [\alpha_{3,i}]P_i\,.$$

By definition $G$ is maximal $\ell^n$-isotropic. This property imposes different conditions on the scalars $(\alpha_{1,1}, \ldots, \alpha_{3,4})$. In this section we suggest a normalisation algorithm that produces canonical generators for each group $G \in \mathcal{K}_\ell$. For this purpose, we assume that $(P_1, P_2, P_3, P_4)$ is a symplectic basis. Of course, one can adapt this procedure to general bases by performing a base change to a symplectic basis before applying the algorithm.

The idea for the normalisation is similar to Gaussian elimination. We set

$$A = \begin{pmatrix} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} \\ \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} \end{pmatrix} \in M_{3,4}(\mathbb{Z}/\ell^n\mathbb{Z}).$$

Using elementary row operations and permuting columns if necessary, we can obtain a matrix of the form

$$A \sim_\sigma \begin{pmatrix} 1 & 0 & * & * \\ 0 & 1 & * & * \\ 0 & 0 & 0 & 0 \end{pmatrix} \text{ if } G \simeq \mathcal{C}_{\ell^n} \times \mathcal{C}_{\ell^n} \qquad \text{or} \quad A \sim_\sigma \begin{pmatrix} 1 & * & * & * \\ 0 & \ell^k & * & * \\ 0 & 0 & \ell^{n-k} & * \end{pmatrix} \text{ if } G \simeq \mathcal{C}_{\ell^n} \times \mathcal{C}_{\ell^{n-k}} \times \mathcal{C}_{\ell^k}. \tag{1}$$

Here $\sigma$ denotes the permutation $\sigma \in S_4$ corresponding to the permutation of the columns, and $*$ is meant as a placeholder. The first case is obtained from the second by setting $k = 0$, hence this will not appear explicitly in the discussion below.

The next lemma shows that this does not affect the corresponding group.

**Lemma 2.** *Let*

$$A' = \begin{pmatrix} \alpha'_{1,1} & \alpha'_{1,2} & \alpha'_{1,3} & \alpha'_{1,4} \\ \alpha'_{2,1} & \alpha'_{2,2} & \alpha'_{2,3} & \alpha'_{2,4} \\ \alpha'_{3,1} & \alpha'_{3,2} & \alpha'_{3,3} & \alpha'_{3,4} \end{pmatrix}$$

*be obtained from $A$ by applying elementary row operations and potentially swapping columns. Let $\sigma \in S_4$ denote the corresponding permutation of the columns. Then*

$$G = \langle G'_1, G'_2, G'_3 \rangle,$$

*where*

$$G'_1 = \sum_{i=1}^{4} [\alpha'_{1,i}] P_{\sigma(i)}, \quad G'_2 = \sum_{i=1}^{4} [\alpha'_{2,i}] P_{\sigma(i)}, \quad G'_3 = \sum_{i=1}^{4} [\alpha'_{3,i}] P_{\sigma(i)}\,.$$

*Proof.* Clear. □

We only used the knowledge of the group structure of $G$ to obtain a presentation as an upper triangular matrix as in (1). Additionally, we also know that the Weil pairing $e_{\ell^n}(G_i, G_j) = 1$ for all $i, j \in \{1, 2, 3\}$. Using this property, we can work out the relations between the non-zero entries of the matrices. The result is captured in the following proposition.

**Proposition 2.** *Let $A$ be a matrix corresponding to a maximal $\ell^n$-isotropic subgroup of the form $\mathcal{C}_{\ell^n} \times \mathcal{C}_{\ell^{n-k}} \times \mathcal{C}_{\ell^k}$ for some integer $0 \leq k \leq \lfloor \frac{n}{2} \rfloor$. Then there exist a permutation $\sigma \in D_8 = \langle (1234), (13) \rangle$ and scalars $a \in [\ell^n]$, $b \in [\ell^{n-k}]$, $c \in [\ell^{n-2k}]$, $d \in [\ell^k]$ such that*

$$A \sim_\sigma A' = \begin{pmatrix} 1 & d & a & b \\ 0 & \ell^k & s_\sigma\, \ell^k\, (b - cd) & \ell^k\, c \\ 0 & 0 & -s_\sigma\, \ell^{n-k}\, d & \ell^{n-k} \end{pmatrix},$$

*where $s_\sigma = \mathrm{sgn}(\sigma)$ denotes the sign of the permutation $\sigma$.*

*On the other hand, if $A'$ is as above and $G' = \langle G'_1, G'_2, G'_3 \rangle$, where*

$$G'_1 = \sum_{i=1}^{4} [\alpha'_{1,i}] P_{\sigma(i)}, \quad G'_2 = \sum_{i=1}^{4} [\alpha'_{2,i}] P_{\sigma(i)}, \quad G'_3 = \sum_{i=1}^{4} [\alpha'_{3,i}] P_{\sigma(i)}$$

*for some $\sigma \in D_8$, then $G'$ is maximal $\ell^n$-isotropic.*

*Proof.* Following the Gaussian elimination process one obtains a matrix $A'$ of the form given in Equation (1). Note that the rank-2 case is just the special case obtained by setting $k = 0$. Examining this process more closely and swapping the last two columns in (1), one sees that $\sigma$ can be chosen to lie in the dihedral group $D_8 = \langle (1234), (13) \rangle$.[4]

Let us write

$$A' = \begin{pmatrix} 1 & d & a & b \\ 0 & \ell^k & \ell^k x & \ell^k\, c \\ 0 & 0 & \ell^{n-k}\, y & \ell^{n-k} \end{pmatrix}$$

for some $a, b, c, d, x, y \in [\ell^n]$. First, note that after adding a multiple of the second line to the first line of $A'$, we may assume that $d \in [\ell^k]$. Similarly, we can achieve $b \in [\ell^{n-k}]$ and $c \in [\ell^{n-2k}]$. It remains to show that $x$ and $y$ are determined by the scalars $a, b, c, d$. This is done using the Weil pairing. For the following computation, it is important to note that

$$e_{\ell^n}(P_{\sigma(1)}, P_{\sigma(3)}) = e_{\ell^n}(P_{\sigma(2)}, P_{\sigma(4)})^{s_\sigma} \tag{2}$$

[4] In the rank-2 case, we moreover have $\sigma \in V_4 = \langle (13), (24) \rangle \subset D_8$.

for all $\sigma \in D_8$.

Let $G'_1, G'_2, G'_3$ be the generators corresponding to the matrix $A'$. Then

$$\begin{aligned} e_{\ell^n}(G'_1, G'_2) &= e_{\ell^n}(P_{\sigma(1)} + [d]P_{\sigma(2)} + [a]P_{\sigma(3)} + [b]P_{\sigma(4)}, \ell^k \cdot (P_{\sigma(2)} + [x]P_{\sigma(3)} + [c]P_{\sigma(4)})) \\ &= e_{\ell^n}(P_{\sigma(1)}, P_{\sigma(3)})^{\ell^k x} \cdot e_{\ell^n}(P_{\sigma(2)}, P_{\sigma(4)})^{\ell^k(cd-b)} \end{aligned}$$

Using property (2), we obtain the condition $\ell^k x = s_\sigma \ell^k (b - cd)$. Computing the Weil pairing on $G_2$ and $G_3$ shows that $\ell^{n-k} y = -s_\sigma \ell^{n-k} d$.

For the other direction, it remains to show that the group $G' = \langle G'_1, G'_2, G'_3 \rangle$ is maximal $\ell^n$-isotropic. This can be done by verifying that $G'$ meets the criteria from Definition 1.

□

The following consequence of the proposition will be helpful for determining the type of a group in the adaptive attack (cf. §3.3).

**Corollary 1.** *Let $(P_1, P_2, P_3, P_4)$ be a symplectic basis for $J[\ell^n]$ and let $G \subset J$ be an isotropic group isomorphic to $\mathcal{C}_{\ell^n} \times \mathcal{C}_{\ell^{n-k}} \times \mathcal{C}_{\ell^k}$. Assume that $G_1 = P_{\sigma(1)} + [d]P_{\sigma(2)} + [a]P_{\sigma(3)} + [b]P_{\sigma(4)} \in G$ for some permutation $\sigma \in D_8$ and scalars $a, b, d$. Then*

$$\mathcal{C}_{\ell^k} \times \mathcal{C}_{\ell^k} \simeq \ell^{n-k} \left\langle P_{\sigma(2)} + [s_\sigma \cdot b]P_{\sigma(3)},\ [-s_\sigma \cdot d]P_{\sigma(3)} + P_{\sigma(4)} \right\rangle \subset G.$$

*Remark 1.* Phrased differently, the proposition above shows that a group $G$ is maximal isotropic if and only if there exist generators $G_1, G_2, G_3$ of the form

$$\begin{aligned} G_1 =& P_{\sigma(1)} + [d]P_{\sigma(2)} + [a]P_{\sigma(3)} + [b]P_{\sigma(4)}, \\ G_2 =& \ell^k \cdot (P_{\sigma(2)} + [s_\sigma \cdot (b - dc)]P_{\sigma(3)} + [c]P_{\sigma(4)}), \\ G_3 =& \ell^{n-k} \cdot ([-s_\sigma \cdot d]P_{\sigma(3)} + P_{\sigma(4)}), \end{aligned}$$

where $a, b, c, d$ and $\sigma$ are as in the statement of the proposition.

Note that in the marginal cases $k \in \{0, \frac{n}{2}\}$, the presentation for the kernel generators simplifies and the setup is more symmetric. If $k = 0$, then $d = 0$ and $a, b, c \in [\ell^n]$. If $k = \frac{n}{2}$, then $c = 0$, $a \in [\ell^n]$ and $b, d \in [\ell^k]$.

*Remark 2.* Proposition 2 shows that each group $G \in \mathcal{K}_\ell$ may be represented by a tuple of the form $(a, b, c, d, k, \sigma)$, where

$$a \in [\ell^n],\ b \in [\ell^{n-k}],\ c \in [\ell^{n-2k}],\ d \in [\ell^k], \quad 0 \le k \le \left\lfloor \frac{n}{2} \right\rfloor \quad \text{and } \sigma \in D_8.$$

Clearly, such a presentation is not unique in most cases. However, it is possible to make the elimination algorithm deterministic by imposing conditions on the choice of the permutation $\sigma \in D_8$. In that way, it is possible to obtain canonical representatives of the form $(a, b, c, d, k, \sigma)$, where each $\sigma \neq \mathrm{id}$ comes with some additional constraints on the parameters $a, b, c, d$. For $k = 0$, we obtain 4 different types of rank-2 groups. We assign the labels $2.1, 2.2., 2.3, 2.4$ to these types. In the case $0 < k < \frac{n}{2}$, there are 8 different types, which we label by $3.1 - 3.8$ and if $k = \frac{n}{2}$ there are 4 types labelled by $4.1 - 4.4$. Table 1 summarizes the classification of the groups in $\mathcal{K}_\ell$.

The classification of the groups in $\mathcal{K}_\ell$ also allows us to determine the cardinality of $\mathcal{K}_\ell$. The number of groups of a given type can be directly read off from the description and is provided in the last column of Table 1. Adding up the numbers for Types $2.1, 2.2, 2.3, 2.4$, we obtain $\ell^{3n-3}(\ell^2 + 1)(\ell + 1)$, the number of maximal isotropic subgroups of rank 2. Adding up the numbers for Types $3.1 - 3.8$, we find that there are $\ell^{3n-2k-4}(\ell^2 + 1)(\ell + 1)^2$ groups isomorphic to $\mathcal{C}_{\ell^n} \times \mathcal{C}_{\ell^{n-k}} \times \mathcal{C}_{\ell^k}$, where $0 < k < \frac{n}{2}$. Finally the sum over the numbers for Types $4.1 - 4.4$ is equal to $\ell^{2n-3}(\ell^2 + 1)(\ell + 1)$, the number of groups isomorphic to $\mathcal{C}_{\ell^n} \times \mathcal{C}_{\ell^k} \times \mathcal{C}_{\ell^k}$, where $2k = n$. These cardinalities coincide with the numbers provided in [5, Prop. 3].

| | label | $\sigma$ | condition on $(a,b,c,d)$ | cardinality |
|---|---|---|---|---|
| $k=0$ | 2.1 | id | - | $\ell^{3n}$ |
| | 2.2 | (24) | $\ell \mid c$ | $\ell^{3n-1}$ |
| | 2.3 | (13) | $\ell \mid a, b$ | $\ell^{3n-2}$ |
| | 2.4 | (13)(24) | $\ell \mid a, b, c$ | $\ell^{3n-3}$ |
| $0 < k < \frac{n}{2}$ | 3.1 | id | - | $\ell^{3n-2k}$ |
| | 3.2 | (24) | $\ell \mid c$ | $\ell^{3n-2k-1}$ |
| | 3.3 | (13) | $\ell \mid a$ | $\ell^{3n-2k-1}$ |
| | 3.4 | (13)(24) | $\ell \mid a, c$ | $\ell^{3n-2k-2}$ |
| | 3.5 | (12)(34) | $\ell \mid b, d$ | $\ell^{3n-2k-2}$ |
| | 3.6 | (1234) | $\ell \mid b, c, d$ | $\ell^{3n-2k-3}$ |
| | 3.7 | (1432) | $\ell \mid a, b, d$ | $\ell^{3n-2k-3}$ |
| | 3.8 | (14)(23) | $\ell \mid a, b, c, d$ | $\ell^{3n-2k-4}$ |
| $2k=n$ | 4.1 | id | - | $\ell^{2n}$ |
| | 4.2 | (13) | $\ell \mid a$ | $\ell^{2n-1}$ |
| | 4.3 | (12)(34) | $\ell \mid b, d$ | $\ell^{2n-2}$ |
| | 4.4 | (1432) | $\ell \mid a, b, d$ | $\ell^{2n-3}$ |

**Table 1.** Classification of maximal $\ell^n$-isotropic subgroups.

### 2.4 New uniform sampling from the keyspace

In the previous section we have described a classification of the groups in $\mathcal{K}_\ell$. This can be used to sample uniformly from the entire keyspace. Here, we introduce a slightly restricted keyspace $\mathcal{K}_\ell^{\text{res}}$ that allows a particularly easy and truly random sampling of the keys. For the convenience of the reader, Figure 1 provides an explicit description of the G2SIDH protocol in this setting.

**Setup**

- prime $p = 2^{e_A} \cdot 3^{e_B} \cdot f - 1$.
- superspecial hyperelliptic curve $H/\mathbb{F}_{p^2}$ with Jacobian $J$.
- symplectic bases $(P_{A,1}, P_{A,2}, Q_{A,1}, Q_{A,2})$ for $J[2^{e_A}]$ and $(P_{B,1}, P_{B,2}, Q_{B,1}, Q_{B,2})$ for $J[3^{e_B}]$.

**Key Generation**

| | | |
|---|---|---|
| – $a_1, a_2, a_3 \xleftarrow{\$} [2^{e_A}]$<br>– $A_1 = P_{A,1} + [a_1]Q_{A,1} + [a_2]Q_{A,2}$<br>$A_2 = P_{A,2} + [a_2]Q_{A,1} + [a_3]Q_{A,2}$<br>– $\phi_A : J \to J_A = J/\langle A_1, A_2\rangle$ | $J_A, \phi_A(P_{B,i}), \phi_A(Q_{B,i})$ for $i \in \{1,2\}$ $\longrightarrow$<br>$\longleftarrow$ $J_B, \phi_B(P_{A,i}), \phi_B(Q_{A,i})$ for $i \in \{1,2\}$ | – $b_1, b_2, b_3 \xleftarrow{\$} [3^{e_B}]$<br>– $B_1 = P_{B,1} + [b_1]Q_{B,1} + [b_2]Q_{B,2}$<br>$B_2 = P_{B,2} + [b_2]Q_{B,1} + [b_3]Q_{B,2}$<br>– $\phi_B : J \to J_B = J/\langle B_1, B_2\rangle$ |

**Shared Key**

$$J_B/\langle \phi_B(A_1), \phi_B(A_2)\rangle = \quad J/\langle A_1, A_2, B_1, B_2\rangle \quad = J_A/\langle \phi_A(B_1), \phi_A(B_2)\rangle$$

**Fig. 1.** G2SIDH with restricted keyspace $\mathcal{K}_\ell^{res}$.

For some fixed symplectic basis $(P_1, P_2, Q_1, Q_2)$ of $J[\ell^n]$, we define the restricted keyspace as

$$\mathcal{K}_\ell^{\text{res}} = \{\langle P_1 + [a]Q_1 + [b]Q_2,\ P_2 + [b]Q_1 + [c]Q_2\rangle \mid a, b, c \in [\ell^n]\}\,.$$

In the terminology of the previous section this means that $\mathcal{K}_\ell^{\text{res}}$ is the set of all groups of Type 2.1.

First of all, note that every secret key $sk \in \mathcal{K}_\ell^{\text{res}}$ is indeed a maximal $\ell^n$-isotropic subgroup as per Proposition 2. A very beneficial feature of the new keyspace is that every secret key $sk \in \mathcal{K}_\ell^{\text{res}}$ is uniquely encoded by a tuple $(a, b, c) \in [\ell^n]^3$. This means that a secret key can be sampled by choosing three random integers $a, b, c \in [\ell^n]$.

Moreover, the restricted keyspace still has the same order of magnitude as the original keyspace. To see this recall the number of maximal $\ell^n$-isotropic subgroups from [5, Thm. 2]:

$$\#\mathcal{K}_\ell = \ell^{2n-3}(\ell^2+1)(\ell+1)\left(\ell^n + \frac{\ell^{n-1}-1}{\ell-1}\right) = \ell^{3n} \cdot \underbrace{\frac{(\ell^2+1)(\ell+1)}{\ell^3}\left(1 + \frac{\ell^{n-1}-1}{\ell^n(\ell-1)}\right)}_{\alpha_\ell}$$

Evaluating the expression on the right, one finds $\alpha_2 \approx \frac{45}{16}$ and $\alpha_3 \approx \frac{140}{81}$ for large $n$.

We would like to point out that a similar restriction of the keyspace is made in the SIDH protocol for elliptic curves. While the space of all $\ell^n$-isogenies from a fixed starting curve is of size $(\ell+1)\ell^{n-1}$, the keyspace used in the optimised implementation is only of size $\ell^n$.

*Remark 3.* We can also obtain uniform sampling on the entire keyspace $\mathcal{K}_\ell$ by taking into consideration the canonical generators and the distribution of the possible subgroup structures in the keyspace.

Suppose $\ell = 2$. The formulae of [5, Thm. 2] and [5, Prop. 3] then show that the proportion of rank-2 subgroups among all admissible subgroups is

$$\frac{2^n}{2^n + 2^{n-1} - 1} \approx \frac{2}{3}$$

if $n$ is large.

Performing the same computation on rank-3 subgroups, for large $n$ we have

$$\frac{3 \cdot 2^{n-2k}}{3 \cdot 2^n - 2} \approx \frac{1}{2^{2k}},$$

where $k$ is the parameter determining the subgroup structure.

Therefore, we obtain a method to almost uniformly sample the keyspace. First, $0 \leq k \leq N$ is determined for some bound $N \leq \lfloor \frac{n}{2} \rfloor$, weighted according to the proportion stated above. Next, one has to make a choice of canonical generators based on the distribution of the different types presented in Table 1. Finally, uniformly selecting the required scalars will ensure the near-uniform sampling from the keyspace.

### 2.5 Point compression

Message compression is a technique used in SIDH to reduce message sizes at the expense of computational overhead. Recall that SIDH messages are of the form $(E, P, Q)$. Since the protocol works in the field $\mathbb{F}_{p^2}$, this message without any compression will be of size $4 \log p$ for the curve $E$, $4 \log p$ for each of the two points $P, Q$, hence $12 \log p$ in total. Azarderakhsh et al.[2] proposed a technique that does not require users of the protocol to send the points $P$ and $Q$. Under the assumption that a deterministic algorithm is able to compute torsion subgroup bases for any elliptic curve, they propose sending scalars that define the points $P$ and $Q$ instead of the points themselves. For instance, suppose that $E$ is the elliptic curve, and the deterministic algorithm returns the torsion subgroup basis $(A, B)$, then one can express $P = [\alpha]A + [\beta]B$ and $Q = [\gamma]A + [\delta]B$. The message should then contain the tuple $(E, \alpha, \beta, \gamma, \delta)$ which has size $4 \log p$. Note that $\alpha, \beta, \gamma, \delta$ are of size $O(0.5 \log p)$.

Using the symplectic basis, we can also use the same ideas for genus two. Observe that since Algorithm 1 (in Appendix B) is deterministic, all we need is an algorithm that can generate an arbitrary basis in a deterministic way. In short, this procedure consists of three parts:

1. Generate elements of the Jacobian deterministically.

2. Form an arbitrary basis deterministically.
3. Form a symplectic basis deterministically.

For the first part, we will show how to generate elements in the Jacobian deterministically.

Given a Jacobian whose hyperelliptic curve is of the form $Y^2 = f(X)$, we use a deterministic pseudo-random number generator to generate a random element of $\mathbb{F}_{p^2}$ and call it $x_1$. Form a quadratic equation in $Y$ given by $Y^2 = f(x_1)$. Solve the quadratic equation and let $y_1$ denote the "smaller" of the two roots. Then $P_1 = (x_1, y_2)$ is a point on the hyperelliptic curve. Repeating this procedure would return $P_1'$. We then use these two points to define the divisor $D_1 = P_1 + P_1'$ in the Jacobian of the hyperelliptic curve.

This gives us a method to deterministically generate elements in the Jacobian. For the next step, we simply have to form an arbitrary basis.

Use the method in the first part to obtain an element in the Jacobian, and call it $D_1$. After multiplying the element $D_1$ with the appropriate cofactor, we can obtain an element of the Jacobian with the correct order.

Repeating this procedure, we are able to obtain $D_2$. Check for linear dependencies between $D_1$ and $D_2$ by checking if there are non-trivial intersections between the two subgroups $\langle D_1 \rangle$ and $\langle D_2 \rangle$.

This procedure is repeated until we have obtained a basis $(D_1, D_2, D_3, D_4)$.

Observe that the vanilla G2SIDH would require $38 \log p$ bits, $6 \log p$ for the full description of $G_2$-invariants, and 4 lots of $8 \log p$ bits, since we have 4 auxiliary points expressed in Mumford representations, and each element in the Mumford representation is described using 4 coefficients in $\mathbb{F}_{p^2}$. Using the point compression technique sketched above, we can reduce the size of the message to $14 \log p$ bits. The $G_2$-invariants will still require $6 \log p$ bits to describe, but auxiliary points would require $2 \log p$ bits each. To see this, recall that $2^n \approx 3^m \approx \sqrt{p}$, and so the when expressing auxiliary points as linear combinations of the symplectic basis, we require 4 coefficients each of $0.5 \log p$ bits.

## 3 Adaptive attack on G2SIDH

The final attack as presented in this section is able to recover the secret kernel of Alice using a static secret kernel which is maximal $2^n$-isotropic. In particular, we will describe a method that can recover secret kernels of various group structures. In the exposition to come, the scalars $\theta$ are used to ensure Weil pairing countermeasures are unable to detect our attack. This method is employed in tandem with the symplectic transformations that are primarily used to isolate the bit under attack. The adaptive attack on G2SIDH is similar to adaptive attacks on SIDH [6], 2-SIDH [4], and Jao–Urbanik's variant [3]. It interacts with an oracle by sending points on some starting variety that correspond to the auxiliary points provided in the protocol. The oracle is "weak" in the sense that only one bit is returned per query. By sending malformed points, the adaptive attack is able to recover scalars that determine the secret kernels.

The first step of the adaptive attack is to recover the kernel structure used by Alice and is presented in §3.3. The next step then recovers the scalars associated with the kernel structure recovered in the first step and is divided into two parts depending on the rank of the kernel structure (§3.4 for rank 2 and §3.5 for rank 3). In each case, we will recover the first bit of the secret scalars before iteratively recovering the remaining bits.

Note that a first approach to recovering the secret scalars is included in Appendix D. The malformed points used in this alternative version can be detected by checking the Weil pairing values of the received points are correct. However, we hope that the accompanying kernel computations may be useful in future cryptanalytic attempts.

In the following, we will assume that all users of the G2SIDH protocol (or at least Alice, the honest party whose key we want to recover) are using a symplectic basis as described in §2.2. This attack will still work on users not using a symplectic basis as one can perform a linear transformation from an arbitrary torsion basis into a symplectic basis. For clarity, we present the attack directly on a symplectic torsion basis here and describe the extension to arbitrary bases in §3.6.

**Notations and set-up**

Let us fix some notation. Let $J$ be the starting variety, and let $J_A$ be the codomain of the secret isogeny with kernel $G_A = \langle A_1, A_2, A_3 \rangle$, where the orders of the points are $2^n$, $2^{n-k}$, $2^k$ respectively.

Furthermore, suppose $\langle P_1, P_2, Q_1, Q_2 \rangle = J[2^n]$ is a symplectic basis such that $e_{2^n}(P_i, Q_i) = \zeta^{\delta_{ij}}$, where $\zeta$ is a primitive $2^n$-th root of unity, and $e_{2^n}(P_1, P_2) = e_{2^n}(Q_1, Q_2) = \zeta^0 = 1$.

We write $\phi_B : J \to J_B$ for Bob's isogeny. Then $(\phi_B(P_1), \phi_B(P_2), \phi_B(Q_1), \phi_B(Q_2))$ is a symplectic basis for $J_B[2^n]$ as per Proposition 1. To ease notation, we set

$$R_1 = \phi_B(P_1),\ R_2 = \phi_B(P_2),\ S_1 = \phi_B(Q_1),\ S_2 = \phi_B(Q_2).$$

We will assume that Alice is the party under attack, and that she is using secret scalars $\alpha_{1,1}, \ldots, \alpha_{3,4}$ which define a maximal $\ell^n$-isotropic subgroup of $J[2^n]$. We can write any of the secret scalars, say $a$, as $a = \sum_{i=0}^{n-1} 2^i a_i$ for bits $a_i \in \{0, 1\}$. For $i = 1, \ldots, n-1$, let us then denote the partial key consisting of the first $i$ bits of $a$ as $K_i^a = \sum_{j=0}^{i-1} 2^j a_j$ so that $a = K_i^a + 2^i a_i + 2^{i+1} a'$ for some $a'$. This convention will help us keep track of the known information at each step of the attack below.

### 3.1 Attack model and oracle

The attack we present in the following assumes that the honest party Alice uses a static key which a malicious Bob is trying to learn through repeatedly providing malformed torsion point information to Alice during the G2SIDH protocol execution. Bob's overall goal is to recover Alice's full key or a valid tuple of scalars forming an equivalent key. While this means we explicitly work with elements of $J[2^n]$ and focus on recovering a kernel corresponding to a sequence of Richelot isogenies, this strategy can be translated to recover a key for more general small primes $\ell$ and therefore $\ell^{e_\ell}$-torsions of $J$ due to Proposition 2. The resulting attack on different $\ell$ may not return a bit of information with every query, but the attack can still be carried out successfully.

It is customary in similar attacks to consider two distinct oracles which can model the information obtained by the attacker which differ in their inherent strength. One which, on input of a variety $J$ and four points $R'_1, \ldots, R'_4 \in J[2^n]$, provides the isomorphism invariants of the codomain variety $J/G_A$ of the isogeny corresponding to the kernel subgroup $G_A = \langle \sum_{i=1}^4 [\alpha_{1,i}] R'_i, \sum_{i=1}^4 [\alpha_{2,i}] R'_i, \sum_{i=1}^4 [\alpha_{3,i}] R'_i \rangle$. The second, less powerful oracle is the one we will utilise to model our attack in the following.

Our oracle

$$O\left(J, J', (R'_1, R'_2, R'_3, R'_4)\right)$$

returns 1 whenever the subgroup $G_A = \langle \sum_{i=1}^4 [\alpha_{1,i}] R'_i, \sum_{i=1}^4 [\alpha_{2,i}] R'_i, \sum_{i=1}^4 [\alpha_{3,i}] R'_i \rangle$ is isotropic and the variety $J/G_A$ has the same isomorphism invariants as the second input variety $J'$. Otherwise, it returns 0. Moreover we assume that the oracle checks whether an input is valid and returns $\perp$ if this is not the case. Here, we say that a tuple $(J, J', (R'_1, R'_2, S'_1, S'_2))$ is a *valid* input if $(R'_1, R'_2, S'_1, S'_2)$ is a symplectic basis for $J[2^n]$ and $e_{2^n}(R'_i, S'_i) = e_{2^n}(P_i, Q_i)^{3^{e_B}}$. Note that an honest run of the protocol generates the valid input $(J_B, J_{AB}, (R_1, R_2, S_1, S_2))$.

For ease of reading, we will represent malformed points to be queried by displaying them in brackets and laid out in a $2 \times 2$ matrix. So when querying the oracle on the points $R'_1, R'_2, S'_1, S'_2$ to obtain a reference variety, we say that we are sending the points

$$\begin{Bmatrix} R'_1, & R'_2, \\ S'_1, & S'_2 \end{Bmatrix}.$$

### 3.2 Symplectic transformations

When constructing malformed torsion points for the oracle queries, we need to make sure that the input is still valid. In our setting, an oracle query

$$O\left(J_B, J_{AB}, (R'_1, R'_2, S'_1, S'_2)\right)$$

is valid if and only if $(R_1', R_2', S_1', S_2')$ is a symplectic basis and

$$e_{2^n}(R_i', S_j') = e_{2^n}(R_i, S_j) \quad \text{for } i, j \in \{1, 2\}.$$

A change of basis $(R_1', R_2', S_1', S_2') \leftarrow (R_1, R_2, S_1, S_2)$ with this property is called a *symplectic transformation*.

Using symplectic transformations has yet another advantage. Let $G = \langle G_1, G_2, G_3 \rangle \subset J$ be maximal $2^n$-isotropic and $t : J[2^n] \to J[2^n]$ a symplectic transformation, then $G' = \langle t(G_1), t(G_2), t(G_3) \rangle$ is maximal $2^n$-isotropic as well. Note that is not true for general isomorphisms of $J[2^n]$.

One can verify that the group of symplectic transformations is generated by the following transformations.

- $t_0 : S_1 \leftarrow R_1 + S_1$
- $t_1 : R_1 \leftarrow R_1 + S_1$
- $t_2 : S_2 \leftarrow R_2 + S_2$
- $t_3 : R_2 \leftarrow R_2 + S_2$
- $t_4 : S_1 \leftarrow R_2 + S_1,\ S_2 \leftarrow R_1 + S_2$
- $t_5 : R_1 \leftarrow R_1 + S_2,\ R_2 \leftarrow R_2 + S_1$

To ease notation, we only indicated which basis elements change under a given transformation, i.e. this should be read as $t_0 : (R_1, R_2, S_1, S_2) \leftarrow (R_1, R_2, S_1 + R_1, S_2)$, and similarly for the other transformations.

**Proposition 3.** *The following transformations are symplectic for any values $x, x_0, x_1, x_2, x_3, x_4, x_5$ and odd scalars $\theta_1, \theta_2, \theta_3, \theta_4 \in [2^n]$ such that $\theta_1\theta_3 \equiv \theta_2\theta_4 \equiv 1 \pmod{2^n}$.*

$$\left\{\begin{matrix}[\theta_1]R_1 + [\theta_2][x]R_2, & [\theta_2]R_2, \\ [\theta_3]S_1, & [\theta_4]S_2 - [\theta_3][x]S_1,\end{matrix}\right\} \quad \textit{and} \quad \left\{\begin{matrix}R_1 + [x_1]S_1 + [x_5]S_2, & R_2 + [x_3]S_2 + [x_5]S_1, \\ S_1, & S_2,\end{matrix}\right\},$$

$$\left\{\begin{matrix}[\theta_1][1 + x_0x_1 + x_4x_5(1 + x_0x_1)]R_1 + [\theta_2][x_2x_5]R_2 + [\theta_3][x_1][1 + x_4x_5]S_1 + [\theta_4][x_5]S_2, \\ [\theta_1][x_0x_5]R_1 + [\theta_2][1 + x_2x_3 + x_4x_5(1 + x_2x_3)]R_2 + [\theta_3][x_5]S_1 + [\theta_4][x_3(1 + x_4x_5)]S_2, \\ [\theta_1][x_0]R_1 + [\theta_2][x_4(1 + x_2x_3)]R_2 + [\theta_3]S_1 + [\theta_4][x_3x_4]S_2, \\ [\theta_1][x_4(1 + x_0x_1)]R_1 + [\theta_2][x_2]R_2 + [\theta_3][x_1x_4]S_1 + [\theta_4]S_2\end{matrix}\right\}.$$

*Proof.* The first transformation is obtained by first scaling with the odd scalars and then applying the transformation $t_2^x t_1^{-x} t_4 t_1^x t_4^{-1}$.

The second transformation is obtained by applying $t_i$ for $i = 1, 3, 5$ a total of $x_i$ times.

The third transformation is again obtained by first scaling with the odd scalars, and then applying $t_i$ from $i = 0$ to $i = 5$ a total of $x_i$ times each. □

All our queries to the oracle are obtained by combining the transformations in the proposition above. In order to choose a transformation, it is necessary to examine the effect of a transformation on a secret subgroup. To illustrate this, assume that Alice uses a group $\langle A_1, A_2, A_3 \rangle$ of Type 3.1. Applying the first transformation from the proposition as the malformed points, we have that Alice would compute with points generating the subgroups

$$\begin{aligned}
\langle [\theta_3]A_1' \rangle &= \langle R_1 + [\theta_2\theta_3x_5]R_2 + [d\theta_2\theta_3]R_2 + [a\theta_3^2]S_1 + [b\theta_3\theta_4]S_2 - [b\theta_3^2x_5]S_1 \rangle \\
&= \langle A_1 + [\theta_2\theta_3x_5 + d(\theta_2\theta_3 - 1)]R_2 - [a(\theta_3^2 - 1) - \theta_3^2x_5b]S_1 + [(\theta_3\theta_2^{-1} - 1)b]S_2 \rangle \\
\langle [\theta_4]A_2' \rangle &= \langle 2^k\big(R_2 + [\theta_3\theta_4(b - cd)]S_1 + [\theta_4^2c]S_2 - [c\theta_3\theta_4x_5]S_1\big) \rangle, \\
&= \langle A_2 + 2^k\big([(\theta_2^{-1}\theta_3 - 1)(b - cd) - \theta_3x_5c]S_1 + [(\theta_2^{-2} - 1)c]S_2\big) \rangle, \\
\langle [\theta_3^{-1}]A_3' \rangle &= \langle 2^{n-k}\big( - [d]S_1 + [\theta_4\theta_3^{-1}]S_2 - [x_5]S_1\big) \rangle \\
&= \langle A_3 + 2^{n-k}\big( - [x_5]S_1 - [\theta_4\theta_3^{-1} - 1]S_1\big) \rangle.
\end{aligned}$$

### 3.3 Case distinction

Recall that Alice's secret is of the form $(\alpha_{1,1}, \ldots, \alpha_{3,4})$. A priori we do not know, if the group $G_A$ defined by these scalars has rank 2 or 3. Moreover we do not know which canonical form is obtained when normalising the generators (cf. Table 1). In total there are 4 types of maximal $2^n$-isotropic groups when $k = 0$, 8 different

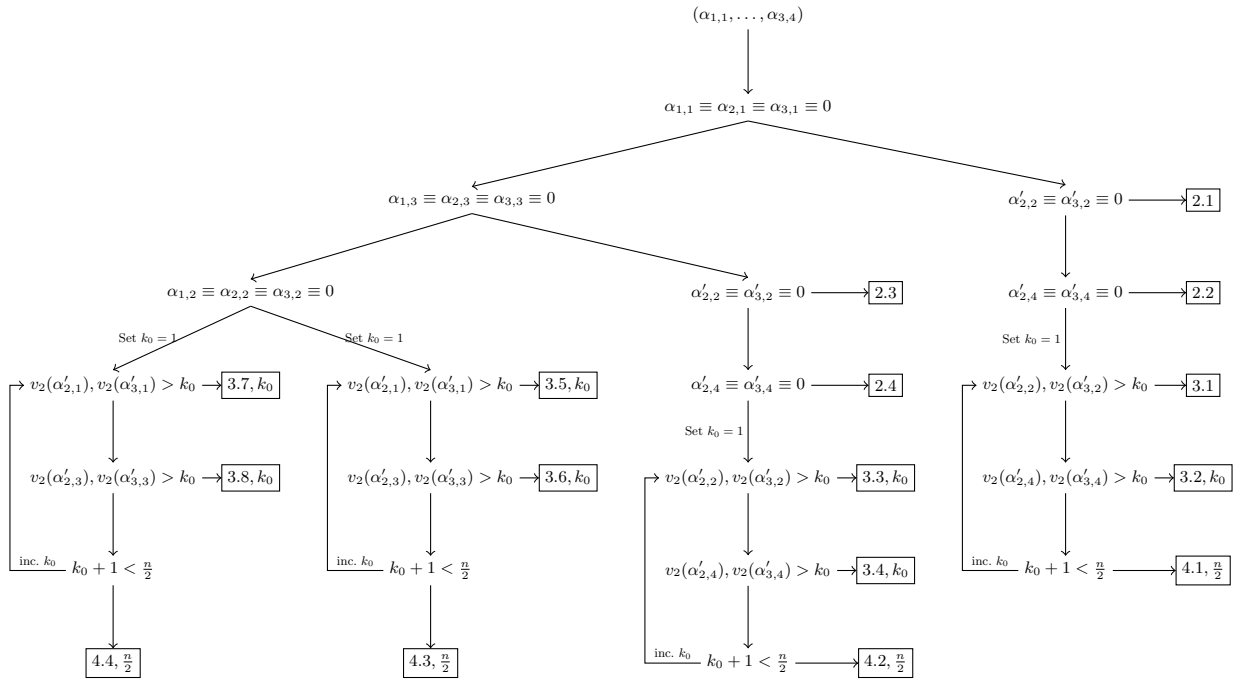


**Fig. 2.** Strategy for type distinction of normalised kernel generators as in Table 1. We begin with Alice's scalars $(\alpha_{1,1}, \ldots, \alpha_{3,4})$. Each node below represents one or multiple malformed queries which determine whether the displayed condition holds. All equivalence conditions are viewed modulo 2 here. A true response will take the left path while the right path is taken when the condition is not fulfilled. We use iterative queries to distinguish between types of rank-3 subgroups by way of finding $k$. Note that when an odd scalar is found, the subsequent conditions use further normalised scalars denoted by $\alpha'_{i,j}$; for an integer $x$, $v_2(x)$ denotes the largest integer such that $2^{v_2(x)}$ divides $x$.

types when $0 < k < \frac{n}{2}$ and 4 different types when $k = \frac{n}{2}$. The type can be recovered by sending at most $4k + 4$ queries that mimic the normalisation process outlined in §2.3. The approach is illustrated in the decision tree in Figure 2, where each node is labelled with the condition we want to test for. Note that at most two queries have to be made per "equivalence" node while at most four queries are necessary to test for divisibility by a power of 2. We provide exemplary details for one of the paths in the decision tree in Appendix C.

Assuming that the key $(\alpha_{1,1}, \ldots, \alpha_{3,4})$ is drawn uniformly at random from the entire key space $\mathcal{K}_2$, the algorithm illustrated by the decision tree will in many cases terminate at an early stage.

Recall from §2.4 that roughly one third of the key space consists of groups of Type 2.1. In that case the algorithm terminates after three queries; one to find that one of $\alpha_{1,1}, \alpha_{2,1}, \alpha_{3,1}$ is odd, and another two to determine that one of $\alpha_{2,2}, \alpha_{3,2}$ is odd. In total, the rank-2 subgroups constitute two thirds of the key space, in which case the algorithm terminates after having made at most six queries. Finally, if we encounter a rank-3 group, it will usually not be necessary to perform many iterations to find $k$ because the probability that $k > k_0$ for some fixed $k_0$ is less than $\frac{1}{3 \cdot 2^{2k_0}}$.

Observe that an attacker obtains some information about the parity of certain bits during the course of the type distinction. In particular for rank-3 groups, we recover normalised scalars $b \pmod{2^k}$ and $d \pmod{2^k} = d$ via the iterative queries. At each step of the iteration, we aim to find out whether $2^{k_0+1}$ divides the coefficients of $P_{\sigma^{-1}(2)}$ and $P_{\sigma^{-1}(4)}$ in the canonical generators of $\langle A_2 \rangle$ and $\langle A_3 \rangle$. In order to achieve this, we need to eliminate the possibility that an oracle query returns 0 because $\langle A_1' \rangle \neq \langle A_1 \rangle$. Hence, we need to query twice for each possible further bit of the coefficients of $P_{\sigma^{-1}(2)}$ and $P_{\sigma^{-1}(4)}$ in $\langle A_1 \rangle$. Hence we recover the first $k$ bits of $b$ and $d$ fully while we determine the type of $G_A$. This information can then be used to drastically reduce the number of queries in the main attack algorithm presented in §3.5, and we therefore assume knowledge of $b \pmod{2^k}$ and $d$ for any rank-3 kernel subgroups.

### 3.4 Kernels of rank 2

As discussed above, there are multiple canonical forms for rank-2 kernels. In this section, we assume that we have applied the method from §3.3 to find the correct canonical form of the kernel generators, which is

$$\begin{aligned} A_1 &= R_1 \quad\quad + [a]S_1 + [b]S_2, \\ A_2 &= \quad R_2 + [b]S_1 + [c]S_2. \end{aligned}$$

Should the generators be of a different canonical form, slight alterations to the malformed points in the exposition of the attack below will suffice to still recover the correct scalars.

**Parity bits.** We want to employ symplectic transformations so that the Weil pairing countermeasure is unable to detect that malformed points have been sent. Table 2 presents transformations that return information about the parity bits, and Figure 3 illustrates how one can use the transformations to get an optimal adaptive attack.

As an example, we examine how the first transformation, ${t_4}^{2^{n-1}}$, affects the kernel generators. This step corresponds to sending malformed points

$$\begin{Bmatrix} R_1, & R_2, \\ S_1 + [2^{n-1}]R_2, & S_2 + [2^{n-1}]R_1 \end{Bmatrix}$$

and leads to Alice (or the oracle) using

$$\begin{aligned} \langle A_1' \rangle &= \langle [1 + 2^{n-1}b]R_1 + [2^{n-1}a]R_2 + [a]S_1 + [b]S_2 \rangle = \langle A_1 + [2^{n-1}(b^2 + ac)]S_2 \rangle, \\ \langle A_2' \rangle &= \langle [2^{n-1}c]R_1 + [1 + 2^{n-1}b]R_2 + [b]S_1 + [c]S_2 \rangle = \langle A_2 + [2^{n-1}(b^2 + ac)]S_1 \rangle, \end{aligned}$$

during their internal computations. Note that the final equality in each line above stems from a transformation of the kernel back into the canonical form. From the resulting kernel generators, we can observe that $O(J_B, J_{AB}, (R_1', R_2', S_1', S_2')) = 1$ if and only if $2^{n-1}(b^2 + ac) \equiv 0 \pmod{2^n}$. This occurs whenever $b \equiv ac \pmod 2$, as displayed in Table 2, and from the response we can determine whether $[a_0, b_0, c_0]$ is among $\{[0,0,0], [0,0,1], [1,0,0], [1,1,1]\}$ or $\{[0,1,0], [0,1,1], [1,0,1], [1,1,0]\}$.

| transformation | oracle outputs 1 iff |
|---|---|
| ${t_0}^{2^{n-1}}$ | $a \equiv b \equiv 0$ |
| ${t_2}^{2^{n-1}}$ | $b \equiv c \equiv 0$ |
| ${t_4}^{2^{n-1}}$ | $b \equiv ac$ |
| ${t_0}^{2^{n-1}}{t_1}^{2^{n-1}}$ | $a + 1 \equiv b \equiv 0$ |
| ${t_0}^{2^{n-1}}{t_3}^{2^{n-1}}$ | $a \equiv b + 1 \equiv 0$ |
| ${t_2}^{2^{n-1}}{t_1}^{2^{n-1}}$ | $b + 1 \equiv c \equiv 0$ |
| ${t_2}^{2^{n-1}}{t_3}^{2^{n-1}}$ | $b \equiv c + 1 \equiv 0$ |

**Table 2.** Table of symplectic transformations and how parity bits affect the codomain. Second column are all equivalence in modulo 2.

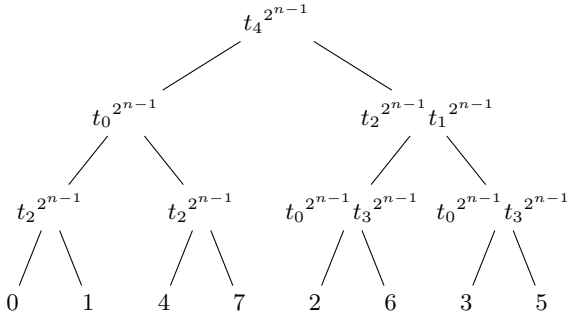


**Fig. 3.** Optimal strategy for recovering parity bits. The top node represents the first malformed query which will use the $t_4$ transformation. At each level, a true response from the oracle will take the left path, while a false takes the right. Leaves are decimal representations of the parity bits $a_0, b_0, c_0$, i.e. 6 corresponds to $[a_0, b_0, c_0] = [1, 1, 0]$.

**Iterative step.** The recovery of subsequent bits will not follow the optimal strategy from the recovery of the parity bits. However, it will still recover a bit of information per query on average.

Suppose now that we have learned the first $i$ bits of each key scalar. Then we know $K_i^a, K_i^b$ and $K_i^c$, where $a = \sum_{j=0}^{n-1} 2^j a_j = K_i^a + \sum_{j=i}^{n-1} 2^j a_j$ (and similarly for $b$ and $c$).

We assume that $i < n - 3$. As a consequence the element $T = 1 - 2^{n-i-1}$ is a quadratic residue modulo $2^n$. In the following $\theta \in [2^n]$ denotes one of the square roots, i.e. $\theta^2 \equiv T \pmod{2^n}$. Note that $\theta$ is necessarily odd, hence there exists an inverse $\theta^{-1}$ modulo $2^n$. Intuitively, $\theta$ is a masking scalar that allows us to defeat the Weil pairing countermeasure.

We use three different sets of malformed points to determine $a_i$ and $c_i$, and then learn $b_i$ with one further query.

First, we send the malformed points

$$\left\{ \begin{matrix} [\theta](R_1 - [T^{-1}2^{n-i-1}K_i^a]S_1),\, [\theta^{-1}](R_2 + [2^{n-i-1}K_i^c]S_2), \\ [\theta^{-1}]S_1, \qquad\qquad\qquad [\theta]S_2 \end{matrix} \right\}.$$

Upon which, the oracle will preform the internal computation[5]

$$\begin{aligned}\langle A_1', A_2'\rangle &= \langle [\theta^{-1}]A_1', [\theta]A_2'\rangle \\ &= \left\langle \begin{matrix} R_1 + [T^{-1}a - 2^{n-i-1}K_i^a T^{-1}]S_1 + [b]S_2 \\ R_2 + [b]S_1 + [Tc + 2^{n-i-1}K_i^c]S_2 \end{matrix} \right\rangle \\ &= \left\langle \begin{matrix} R_1 + [a]S_1 + [b]S_2 + [T^{-1}2^{n-i-1}(K_i^a - a)]S_1, \\ R_2 + [b]S_1 + [c]S_2 + [2^{n-i-1}(K_i^c - c)]S_2 \end{matrix} \right\rangle \\ &= \left\langle \begin{matrix} A_1 + [T^{-1}2^{n-1}a_i]S_1, \\ A_2 + [2^{n-1}c_i]S_2 \end{matrix} \right\rangle.\end{aligned}$$

This is the same group as $\langle A_1, A_2\rangle$ exactly when both $a_i$ and $c_i$ are zero.

If we have not yet recovered the two bits in question, we proceed with sending malformed points

$$\left\{ \begin{matrix} [\theta](R_1 - [T^{-1}2^{n-i-1}(K_i^a + 2^i)]S_1), & [\theta^{-1}](R_2 + [2^{n-i-1}K_i^c]S_2), \\ [\theta^{-1}]S_1, & [\theta]S_2 \end{matrix} \right\}.$$

In this case

$$\langle A_1', A_2'\rangle = \left\langle \begin{matrix} A_1 + [T^{-1}][2^{n-1}(a_i + 1)]S_1, \\ A_2 + [2^{n-1}c_i]S_2 \end{matrix} \right\rangle.$$

This coincides with the group $\langle A_1, A_2\rangle$ exactly when $a_i = 1$ and $c_i = 0$.

If both queries fail to recover the bits $a_i$ and $c_i$, i.e. $(a_i, c_i) \notin \{(0,0),(1,0)\}$, then we can conclude that $c_i = 1$. To find the bit $a_i$, we then send the third set of malformed points

$$\left\{ \begin{matrix} [\theta](R_1 - [T^{-1}2^{n-i-1}K_i^a]S_1), & [\theta^{-1}](R_2 + [2^{n-i-1}K_{i+1}^c]S_2), \\ [\theta^{-1}]S_1, & [\theta]S_2 \end{matrix} \right\}.$$

Here, the oracle will return 1 exactly when $a_i = 0$. If this is not the case, then $a_i = 1$.

After these series of queries, we have recovered the bits $a_i$ and $c_i$, hence we know $K_{i+1}^a$ and $K_{i+1}^c$. It remains to recover the bit $b_i$. This is done by querying the oracle on the following malformed points

$$\left\{ \begin{matrix} [\theta^{-1}](R_1 + [2^{n-i-1}K_{i+1}^a]S_1 + [2^{n-i-1}K_i^b]S_2), & [\theta^{-1}](R_2 + [2^{n-i-1}K_i^b]S_1 + [2^{n-i-1}K_{i+1}^c]S_2), \\ [\theta]S_1, & [\theta]S_2 \end{matrix} \right\}.$$

Here,

$$\begin{aligned}\langle A_1', A_2'\rangle &= \langle \theta A_1', \theta A_2'\rangle \\ &= \left\langle \begin{matrix} R_1 + [a]S_1 + [b]S_2 + [2^{n-i-1}(K_{i+1}^a - a)]S_1 + [2^{n-i-1}(K_i^b - b)]S_2, \\ R_2 + [b]S_1 + [c]S_2 + [2^{n-i-1}(K_i^b - b)]S_1 + [2^{n-i-1}(K_{i+1}^c - c)]S_2 \end{matrix} \right\rangle \\ &= \left\langle \begin{matrix} R_1 + [a]S_1 + [b]S_2 + [2^{n-1}b_i]S_2, \\ R_2 + [b]S_1 + [c]S_2 + [2^{n-1}b_i]S_1 \end{matrix} \right\rangle.\end{aligned}$$

The oracle returns 0 exactly when $b_i = 0$. Otherwise, we know that $b_i = 1$.

It follows from Proposition 3 that all of the transformations used in these queries are symplectic. Therefore they constitute valid queries in our oracle model. As a consequence the attack is not detectable by the Weil pairing.

Note that we are not able to use these transformation for $i \in \{n-3, n-2, n-1\}$. We suggest to use a brute force method to deduce the last three bits of each scalar. This is consistent with the adaptive attack described in [6].

[5] To verify the computation below note that $T^{-1} \equiv 1 + 2^{n-i-1}T^{-1} \pmod{2^n}$.

*Complexity.* Taking into account that the case distinction strategy outlined in §3.3 requires at most 6 queries to determine any type of rank-2 kernel subgroup as well as the information we learn about the parity of Alice's scalars throughout the process, we find that this attack requires at most $6 + 4(n-4) = 4n - 10$ queries, each corresponding to one isogeny computation. This leaves 3 bits per secret scalar, hence 9 bits in total, to be recovered through brute force.

### 3.5 Kernels of rank 3

Now suppose Alice's secret kernel subgroup has three generators, i.e. $k > 0$. Let $1 \leq k \leq \lfloor \frac{n}{2} \rfloor$ be fixed. We assume that the attacker has determined the type of Alice's secret subgroup as outlined in §3.3, and therefore knows $k$. Then, possibly modulo some permutation of the basis points, the generators are of the form

$$\begin{aligned} A_1 &= \quad R_1 + [d]R_2 + [a]S_1 \quad + [b]S_2, \\ A_2 &= 2^k( \quad R_2 \quad + [b-dc]S_1 + [c]S_2), \text{ and} \\ A_3 &= 2^{n-k}( \quad\quad - [d]S_1 \quad + S_2) \end{aligned}$$

for some $(a, b, c, d) \in [2^n] \times [2^{n-k}] \times [2^{n-2k}] \times [2^k]$, where $b \pmod{2^k}$ and $d$ are known from the case distinction algorithm. As usual, we denote the resulting variety $J_B/\langle A_1, A_2, A_3 \rangle$ by $J_{AB}$.

We first fix

$$l = n - i - 1, \quad T = 1 - 2^l, \quad \theta = \sqrt{T}$$

for $1 \leq i \leq n - 4$.

**Recovering $a \pmod{2^{k-1}}$.** We first recover the parity of the secret scalar $a$ by sending the malformed points

$$\begin{Bmatrix} R_1, & R_2, \\ S_1 + [2^{n-1}]R_1, & S_2 \end{Bmatrix}.$$

These allow us to recover the bit $a_0$ since Alice performs the following internal computations, where $A'_1, A'_2, A'_3$ denote the images of $A_1, A_2, A_3$ under the above transformation,

$$\begin{aligned} \langle A'_1, A'_2, A'_3 \rangle &= \langle\ R_1 + [d]R_2 + [a]S_1 + [a][2^{n-1}]R_1 + [b]S_2, \\ &\qquad 2^k(R_2 + [b-dc]S_1 + [b-dc][2^{n-1}]R_1 + [c]S_2), \\ &\qquad 2^{n-k}(-[d]S_1 + \ S_2)\rangle \\ &= \langle\ A_1 + [a^2]S_1, A_2, A_3 \rangle\,. \end{aligned}$$

Note that a transformation into the canonical representation of the first kernel generator which does not change the resulting variety $J'_{AB}$ is applied here. Since the second and third kernel generator are unchanged from the computation yielding the original variety $J_{AB}$ and we know that $\langle A'_1, A'_2, A'_3 \rangle = G_A$ if and only if $a_0 = 0$, we can deduce $a_0$ from the oracle response.

Now, we iteratively recover the bit $a_i$ for $i = 1, \ldots, k-2$ using the knowledge of $K_i^a = \sum_{j=0}^{i-1} 2^j a_j$ obtained from the previous steps. Fix

$$\alpha = -2^l(dK_k^b + K_i^a)T^{-1}, \quad \delta = -2^l dT^{-1}.$$

and send the malformed points

$$\begin{Bmatrix} [\theta](R_1 + [\delta]R_2 + [\alpha]S_1), & [\theta^{-1}]R_2, \\ [\theta^{-1}]S_1, & [\theta]([-\delta]S_1 + S_2) \end{Bmatrix}.$$

This transformation leaves the second and third kernel generator unchanged since $[\theta]A'_2 = A_2$ and $[\theta]A'_3 = A_3$, and affects the first generator as follows. From

$$[\theta^{-1}]A'_1 = R_1 + [\delta + \theta^{-2}d]R_2 + [\alpha + \theta^{-2}a - \delta b]S_1 + [b]S_2 = R_1 + [d]R_2 + [T^{-1}(a - 2^l K_i^a)]S_1 + [b]S_2$$

and the observation that $T^{-1} = 1 + 2^l$ (when $k \geq 1$ and $i \leq k-1$ as is the case here), it follows that $[\theta^{-1}]A'_1 = A_1 + [2^{n-1}][a_i]S_1$. Hence, we can determine the desired bit from the oracle response since $O(J_B, J_{AB}, (R'_1, R'_2, S'_1, S'_2)) = 1$ implies $a_i = 0$, and $a_i = 1$ otherwise.

**Recovering $a \pmod{2^{n-k-1}}$ and $c$.** We recover the bits $a_i$ and $c_{i-k+1}$ for $i = k-1, \ldots, n-k-2$ simultaneously. Recall that we know the first $k$ bits of $b$, i.e. $K_k^b$, as well as $d$ from the type distinction of kernel subgroups. In the following we assume that $d$ is an odd integer. The queries can be easily adapted to the case where $d$ is even by shifting the indices of $c$ accordingly.

In the first query we send the malformed points

$$\begin{Bmatrix} [\theta](R_1 + [\delta]R_2 + [\alpha]S_1 + [\beta]S_2), & [\theta^{-1}](R_2 + [T\beta]S_1 + [\gamma]S_2), \\ [\theta^{-1}]S_1, & [\theta]([-\delta]S_1 + S_2) \end{Bmatrix}$$

where

$$\alpha = -2^l K_i^a T^{-1}, \quad \beta = -2^{l-1} K_{i-k+1}^c d T^{-1}, \quad \gamma = 2^l K_{i-k+1}^c, \quad \delta = -2^{l-1} d T^{-1}.$$

Then we obtain

$$\begin{aligned} [\theta^{-1}]A_1' &= A_1 + [-\delta]R_2 + [2^l(a - K_i^a) - \delta b + \beta T]S_1 + [\beta S_2], \\ [\theta]A_2' &= A_2 + 2^k\left([2^{l-1}d(c - K_{i-k+1}^c)]S_1 + [2^l(c - K_{i-k+1}^c)]S_2\right), \\ [\theta^{-1}]A_3' &= A_3. \end{aligned}$$

Since $i \geq k-1$, we find that

$$[\theta]A_2' = A_2 + [2^{n-1}][dc_{i-k+1}]S_1,$$

from which we obtain the condition $c_{i-k+1} = 0$.

In order to compare $A_1'$ and $A_1$, we consider the following simplification.

$$\begin{aligned} [\theta^{-1}]A_1 - [2^{l-k-1}T^{-1}d][\theta]A_2 =& A_1 + [2^l(a - K_i^a) + 2^{l-1}d^2(c - K_{i-k+1}^c)]S_1 + [2^{l-1}dT^{-1}(K_{i-k+1}^c - c)]S_2 \\ =& A_1 + [2^{n-1}a_i]S_1 + [2^{n-k-1}d^2 c_{i-k+1}]([-d]S_1 + S_2). \end{aligned}$$

This shows that the oracle returns 1 if and only if $a_i = c_{i-k+1} = 0$.

If the oracle returns 0, we proceed with a query to test if $a_i = 1$ and $c_{i-k+1} = 0$. This is achieved by setting $\alpha = -2^l(K_i^a + 2^i)T^{-1}$ in the query above.

Similarly, we test for $a_i = 0$ and $c_{i-k+1} = 1$ by setting $\beta = -2^{l-1}(K_{i-k+1}^c - 2^{i-k+1})dT^{-1}$ and $\gamma = 2^l(K_{i-k+1}^c + 2^{i-k+1})$.

**Recovering $b$.** Recall that $K_{n-k-1}^a, K_k^b, c$ and $d$ are known from previous oracle queries. We now utilise this knowledge to find the remaining bits of $b$. Again, assuming $d$ to be odd here allows us to perform the queries below for any $k \leq i < n - \max\{k, 3\}$ which can be adapted via some shift in indices to accommodate even $d$. Let

$$\alpha = 2^l(K_{n-k-1}^a + K_i^b d - cd^2), \quad \beta = 2^l cd, \quad \gamma = 2^l T^{-1} c, \quad \delta = 2^l d.$$

We then send malformed points

$$\begin{Bmatrix} [\theta^{-1}]\left(R_1 + [\delta]R_2 + [\alpha]S_1 + [\beta]S_2\right), & [\theta]\left(R_2 + [\beta T^{-1}]S_1 - [\gamma]S_2\right), \\ [\theta]S_1, & [\theta^{-1}]\left([-\delta]S_1 + S_2\right) \end{Bmatrix}$$

to the oracle so that it performs the internal computation

$$\begin{aligned} \langle A_1', A_2', A_3' \rangle &= \langle [\theta]A_1', [\theta^{-1}]A_2', [\theta]A_3' \rangle \\ &= \left\langle \begin{matrix} R_1 + [\delta + Td]R_2 + [\alpha + \beta\delta - \delta b + T(K_{n-k-1}^a + \beta d)]S_1 + [\beta - \gamma\delta + b - T\gamma]S_2, \\ 2^k\left(R_2 + [\beta + b - dc - T^{-1}\delta c]S_1 + [T^{-1}c - \gamma]S_2\right), \\ 2^{n-k}\left([-Td - \delta]S_1 + S_2\right) \end{matrix} \right\rangle \\ &= \left\langle \begin{matrix} A_1 + 2^l d(K_i^b - b)S_1, \\ A_2, \\ A_3 \end{matrix} \right\rangle \end{aligned}$$

and returns 1 if $b_i = 0$, and 0 if $b_i = 1$.

**Recovering $a$.** It remains to recover the last bits of $a$, given $K^a_{n-k-1}$ as well as $b, c$ and $d$. Let $i = n-k-1, \ldots, n-3$. We again fix

$$\alpha = 2^l(K_i^a + bd - cd^2), \quad \beta = 2^l cd, \quad \gamma = 2^l T^{-1} c, \quad \delta = 2^l d$$

and query the oracle with the symplectic transformation

$$\begin{Bmatrix} [\theta^{-1}]\left(R_1 + [\delta]R_2 + [\alpha]S_1 + [\beta]S_2\right), & [\theta]\left(R_2 + [\beta T^{-1}]S_1 - [\gamma]S_2\right), \\ [\theta]S_1, & [\theta^{-1}]\left([-\delta]S_1 + S_2\right) \end{Bmatrix}.$$

Again, $[\theta^{-1}]A'_2 = A_2$ and $[\theta]A'_3 = A_3$ while $[\theta]A'_1 = A_1 + [2^l(K_i^a - a)]S_1 = A_1 - [2^{n-1}][a_i]S_1$. Hence, we can deduce the bit $a_i$ from the response of the oracle whereby $O(J_B, J_{AB}, (R'_1, R'_2, S'_1, S'_2)) = 1$ implies $a_i = 0$, and $a_i = 1$ otherwise.

Since the square root of $T = 1 - 2^l$ is not defined when $i \geq n-3$, we cannot scale the malformed points in order to obtain a valid symplectic transformation. Therefore, the last three bits of $a$ need to be recovered by brute force.

*Complexity.* If Alice's kernel has rank 3, we can learn the type of the subgroup as well as the scalar $d$ and $b \pmod{2^k}$ following §3.3 with at most $4 + 4k$ queries. We further require $k-1$ queries, one for each of the first $k-1$ bits of $a$, and then 3 queries for each step of the parallel recovery of $a_i$ and $c_{i-k+1}$, summing to $4 + 4k + k - 1 + 3(n - 2k - 2) = 3n - k - 3$ queries thus far. Each remaining bit of $b$ and $a$, potentially bar the last $4-k$ and 3 bits respectively, requires exactly one query to recover, adding $n-6$ queries. This leads to a total number of at most $4n - k - 9$ queries to recover Alice's secret key while leaving 3 bits of $a$ as well as $4-k$ bits of $b$ and $3-k$ bits of $c$ (if $k < 4$ and $k < 3$, respectively) to brute force.

### 3.6 Adaptive attack on arbitrary basis

In the above, we were able to show that the adaptive attack is able to recover a static key when a symplectic basis is used. In this section, we will present an extension of the attack to recover a static key when an arbitrary basis is used (as originally described in [5]).

As shown in Algorithm 1 (cf. Appendix B), we are able to obtain a symplectic basis from an arbitrary basis using a $4 \times 4$ change of basis matrix. In particular, such a basis has the following form:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ \gamma_1 & -\gamma_2 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ \mu_3^{-1}\mu_1 & -\mu_3^{-1}\mu_2 & 0 & \mu_3^{-1} \end{pmatrix}$$

up to swapping certain rows depending on the result of the `if` branch of Algorithm 1. This matrix, together with its inverse, allows us to transform points in one basis to another. Each time we need to query the oracle on a particular set of malformed points, we map these malformed points under the inverse of the matrix to get the malformed corresponding points of the arbitrary basis. We still obtain the same bit of information in return: either the oracle returns the reference variety, or it does not. This ultimately allows us to recover the secret for a symplectic basis which is equivalent to knowing the secret isogeny. Note that the attack is still not detectable by the Weil pairing if the transformations from the previous sections are applied.

## References


1. Azarderakhsh, R., Campagna, M., Costello, C., De Feo, L., Hess, B., Hutchinson, A., Jalali, A., Jao, D., Karabina, K., Koziel, B., LaMacchia, B., Longa, P., Naehrig, M., Pereira, G., Renes, J., Soukharev, V., Urbanik, D.: Supersingular isogeny key encapsulation (SIKE). Updated specifications for NIST Post-Quantum Standardization project (2020), `http://sike.org/`

2. Azarderakhsh, R., Jao, D., Kalach, K., Koziel, B., Leonardi, C.: Key compression for isogeny-based cryptosystems. In: Proceedings of the 3rd ACM International Workshop on ASIA Public-Key Cryptography. pp. 1–10 (2016)
3. Basso, A., Kutas, P., Merz, S., Petit, C., Weitkämper, C.: On adaptive attacks against Jao-Urbanik's isogeny-based protocol. In: Nitaj, A., Youssef, A.M. (eds.) Progress in Cryptology - AFRICACRYPT 2020 - 12th International Conference on Cryptology in Africa, Cairo, Egypt, July 20-22, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12174, pp. 195–213. Springer (2020), https://doi.org/10.1007/978-3-030-51938-4_10
4. Dobson, S., Galbraith, S.D., LeGrow, J.T., Ti, Y.B., Zobernig, L.: An adaptive attack on 2-SIDH. Int. J. Comput. Math. Comput. Syst. Theory **5**(4), 282–299 (2020), https://doi.org/10.1080/23799927.2020.1822446
5. Flynn, E.V., Ti, Y.B.: Genus two isogeny cryptography. In: International Conference on Post-Quantum Cryptography. pp. 286–306. Springer (2019)
6. Galbraith, S.D., Petit, C., Shani, B., Ti, Y.B.: On the security of supersingular isogeny cryptosystems. In: Advances in Cryptology - ASIACRYPT 2016. pp. 63–91 (2016), https://doi.org/10.1007/978-3-662-53887-6_3
7. Jao, D., Feo, L.D.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In: Yang, B. (ed.) Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29 - December 2, 2011. Proceedings. Lecture Notes in Computer Science, vol. 7071, pp. 19–34. Springer (2011), https://doi.org/10.1007/978-3-642-25405-5_2
8. Milne, J.S.: Abelian varieties. In: Cornell, G., Silverman, J.H. (eds.) Arithmetic Geometry, pp. 103–150. Springer New York, New York, NY (1986)
9. Milne, J.S.: Jacobian varieties. In: Cornell, G., Silverman, J.H. (eds.) Arithmetic Geometry, pp. 167–212. Springer New York, New York, NY (1986)
10. NIST (National Institute of Standards and Technology): NIST post-quantum cryptography project. http://csrc.nist.gov/groups/ST/post-quantum-crypto/ (2017)
11. Silverman, J.H.: The arithmetic of elliptic curves, vol. 106. Springer Science & Business Media (2009)

# A Revisiting SIDH

In this paper we studied different aspects of the generalisation of the SIDH scheme to abelian surfaces. For that purpose it was necessary to introduce different notions that did not appear in the description of the SIDH protocol for elliptic curves. In order to give some intuition on the different terms, we explain their meaning in the case of elliptic curves and demonstrate the analogies to our setting.

## A.1 SIDH protocol

Let $E$ be a supersingular elliptic curve. In the setup, one chooses two small primes $\ell_A$ and $\ell_B$ and a prime $p$ which is of the form $p = \ell_A^{e_A}\ell_B^{e_B}f - 1$, where $f$ is a small cofactor and $e_A, e_B$ are large integers. Due to efficiency reasons in [7], the authors suggest the use of $\ell_A = 2$ and $\ell_B = 3$, while ensuring that parameters are balanced, i.e. $\ell_A^{e_A} \approx \ell_B^{e_B}$, provides security. Let $P_A, Q_A$ be generators of the $\ell_A^{e_A}$-torsion and let $P_B, Q_B$ be generators of the $\ell_B^{e_B}$-torsion of $E$. Then the protocol is as follows; a sketch of it can be found in Figure 4.

1. Alice chooses a random cyclic subgroup of $E[\ell_A^{e_A}]$ of order $\ell_A^{e_A}$. As $P_A, Q_A$ form a basis of the $\ell_A^{e_A}$-torsion, there exist integers $x_A, y_A$ such that $A = [x_A]P_A + [y_A]Q_A$ generates this subgroup. Similarly, Bob chooses a random cyclic subgroup of $E[\ell_B^{e_B}]$ of order $\ell_B^{e_B}$ generated by $B = [x_B]P_B + [y_B]Q_B$ for some $x_B, y_B$.
2. Alice computes the isogeny $\phi_A : E \to E/\langle A\rangle$ and Bob computes the isogeny $\phi_B : E \to E/\langle B\rangle$.
3. Alice sends the curve $E/\langle A\rangle$ and the points $\phi_A(P_B)$ and $\phi_A(Q_B)$ to Bob and Bob similarly sends ($E/\langle B\rangle$, $\phi_B(P_A)$, $\phi_B(Q_A)$) to Alice.
4. Alice and Bob both use the images of the torsion points to compute the shared secret which is the curve $E/\langle A, B\rangle$ (e.g. Alice can compute $\phi_B(A) = [x_A]\phi_B(P_A) + [y_A]\phi_B(Q_A)$ and $E/\langle A, B\rangle = E_B/\langle\phi_B(A)\rangle$).

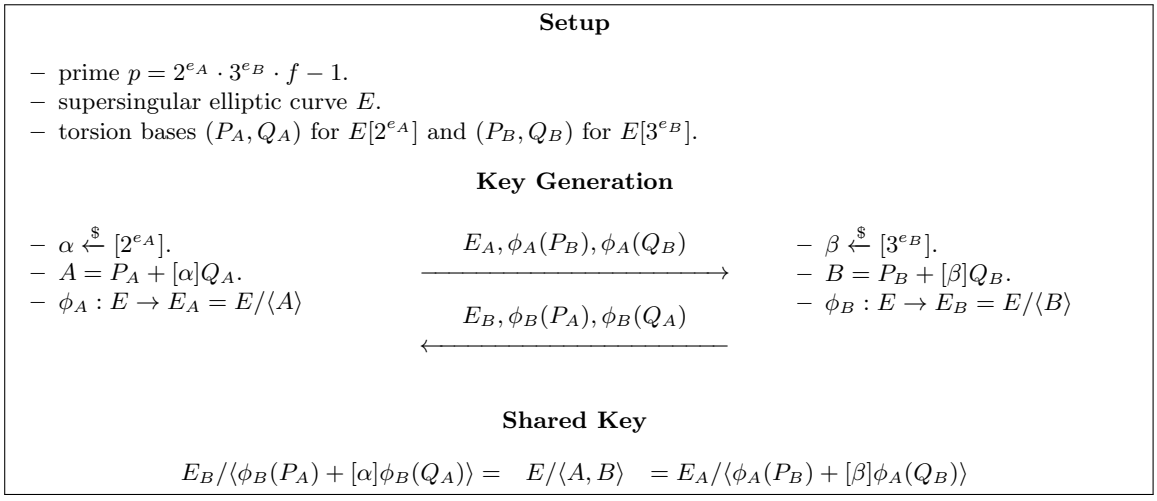


**Fig. 4.** SIDH protocol with restricted keyspace after normalisation as in [6, Lemma 2.1].

## A.2 Keyspace for SIDH

First of all, we would like to point out that in the case of elliptic curves, one automatically deals with symplectic bases.

**Lemma 3.** *Every basis $(P, Q)$ for $E[\ell^n]$ is symplectic.*

*Proof.* Let $(P, Q)$ be any basis for $E[\ell^n]$ and write $e_{\ell^n}$ for the Weil pairing on $E[\ell^n]$. Then $e_{\ell^n}(P, Q) = \mu$ is a primitive $\ell^n$-th root of unity. This means that $(P, Q)$ is automatically a symplectic basis with respect to the Weil pairing. □

The keyspace $\mathcal{K}_\ell$ consists of all cyclic subgroups of $E[\ell^n]$ of order $\ell^n$. As one might expect, it turns out that these are precisely the maximal $\ell^n$-isotropic subgroups.

**Lemma 4.** *A group $K \subset E[\ell^n]$ is cyclic of order $\ell^n$ if and only if $K$ is maximal $\ell^n$-isotropic.*

*Proof.* Let $K = \langle R \rangle$ be such a group. Then for any two elements $R_1, R_2 \in K$, there exist $\alpha_1, \alpha_2 \in [\ell^n]$ satisfying $R_i = [\alpha_i]R$ for $i \in \{1, 2\}$. This implies

$$e_{\ell^n}(R_1, R_2) = e_{\ell^n}(R, R)^{\alpha_1\alpha_2} = 1,$$

i.e. the Weil pairing restricts trivially onto $K$. Moreover $\mathrm{ord}(R) = \ell^n$, hence $K$ cannot be contained in any torsion group $E[m]$ with $m < \ell^n$. It remains to prove the maximality condition, i.e. we need to show that we cannot add an element $R'$ to $K$ without obtaining a non-trivial pairing. For that purpose, take $S$ to be some element $E[\ell^n]$ such that $e_{\ell^n}(R, S) = \mu$ has order $\ell^n$. Then clearly, $(R, S)$ is a basis for $E[\ell^n]$ and we can write $R' = [r]R + [s]S$ with $s \neq 0$ for any element $R' \notin K$. But this implies $e_{\ell^n}(R, R') = \mu^s \neq 1$.

The other direction is left to the reader. □

Recall that the keyspace of SIDH, $\mathcal{K}_\ell$, can be divided into two disjoint sets as follows

$$\mathcal{K}_\ell = \{\langle P + [\alpha]Q\rangle \mid \alpha \in [\ell^n]\} \cup \{\langle [\ell\alpha]P + Q\rangle \mid \alpha \in [\ell^{n-1}]\}.$$

In the terminology of §2.3, this means that there are two types of groups in $\mathcal{K}_\ell$ as opposed to the multitude of types in the genus-2 setting (cf. Table 1). To draw analogies later, we assign the labels (1.1) and (1.2) to these two types. Note that the entire keyspace has cardinality $\ell^{n-1}(\ell + 1)$. However in practice one restricts to the groups of Type (1.1). This restricted key space has cardinality $\ell^n$ [1, Sec. 1.3.9]. This is very similar to the restriction suggested in §2.4.

### A.3 Adaptive attack

In [6], the authors present an adaptive attack on SIDH. Here, we will briefly present their strategy and explain the connection to the adaptive attack on G2SIDH we present. In particular, we will show that the attack on SIDH is implied by our attack.

A major obstruction when devising an attack strategy for G2SIDH was to avoid detection by the Weil pairing. We overcame potential detection by only allowing symplectic transformations of the basis elements for the oracle queries. Indeed this strategy is also followed in [6] albeit not explicitly phrased in this way.

Assume that Alice uses a fixed secret key $x_A, y_A$ that define the group $G_A = \langle [x_A]P_A + [y_A]Q_A\rangle \in \mathcal{K}_2$. Her public key is of the form $(E_A, \phi_A(P_B), \phi_A(Q_B))$, where $E_A$ is the codomain of the isogeny $\phi_A : E \to E_A$ with kernel $G_A$. It is assumed that the attacker has access to the oracle

$$O(E_1, E_2, (R, S)) = \begin{cases} \perp & \text{if } e_{2^n}(R, S) \neq e_{2^n}(P_A, Q_A)^{3^m}, \\ 1 & \text{if } E_2 \simeq E_1/\langle [x_A]R + [y_A]S\rangle, \\ 0 & \text{otherwise.} \end{cases}$$

Honestly running the protocol, the attacker first generates Bob's ephemeral values $(E_B, R = \phi_B(P_A), S = \phi_B(Q_A))$ and computes the elliptic curve $E_{AB}$.[6] Then they send different queries to the oracle with fixed curves $E_B$ and $E_{AB}$, while the basis $(R, S)$ is modified in each step. In order to create a valid query it is necessary that the Weil pairing on the malformed basis elements $(R', S')$ coincides with that on $(R, S)$. Phrased differently, the basis transformation

$$(R' = [a_{1,1}]R + [a_{1,2}]S,\ S' = [a_{2,1}]R + [a_{2,2}]S) \leftarrow (R, S)$$

[6] Note that by construction $O(E_B, E_{AB}, (R, S)) = 1$.

has to be a symplectic transformation. Note that this transformation can be represented by the $2 \times 2$ matrix

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}.$$

It is easy to see that this matrix is symplectic if and only if its determinant $\det(A) = 1$. However this criterion is unique to the case of $2 \times 2$ matrices.

**Case distinction.** The first step in the attack is to distinguish between groups of Type (1.1) and (1.2). This is done by sending the malformed points $(R + [2^{n-1}]S, S)$ to the oracle. Note that

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

This coincides with $\langle [x_A]R + [y_A]S \rangle$ if and only if $x_A$ is even. It follows that

$$O(E_B, E_{AB}, (R + [2^{n-1}]S,\ S)) = \begin{cases} 1 & \text{if } G_A \text{ has Type 1.2,} \\ 0 & \text{if } G_A \text{ has Type 1.1.} \end{cases}$$

In the following we assume that $G_A$ has Type (1.1) and denote the normalised generator by $A = P_A + \alpha Q_A$.

**First bit recovery.** In order to find the first bit of $\alpha$, the attacker queries the oracle on the malformed points $R,\ [2^{n-1}]R + S$. It is easy to check that the query is valid and that

$$O(E_B, E_{AB}, (R,\ [2^{n-1}]R + S)) = \begin{cases} 1 & \text{if } \alpha \text{ is even,} \\ 0 & \text{if } \alpha \text{ is odd.} \end{cases}$$

**Iterative step.** Assume the attacker has recovered the first $i$ bits of $\alpha$. Then we know $K_i^\alpha$, where $\alpha = K_i^\alpha + \sum_{j=i}^{n-1} \alpha_j 2^j$. If $i < n-3$, there exists $\theta$ satisfying $\theta^2 \equiv 1 + 2^{n-i-1} \pmod{2^n}$, [6, Lemma 3.3]. Necessarily $\theta$ is an odd integer, hence invertible modulo $2^n$. Consider the basis

$$(R' = [\theta^{-1}]R - [\theta^{-1}2^{n-i-1}K_i^\alpha]S,\ S' = [\theta]S).$$

The matrix associated to the transformation $(R', S') \to (R, S)$ is $\begin{pmatrix} [\theta^{-1}] & [-\theta^{-1}2^{n-i-1}K_i^\alpha] \\ 0 & [\theta] \end{pmatrix}$, which has determinant 1, hence is symplectic. This means that the tuple $(E_B, E_{AB}, (R', S'))$ defines a valid query. Note that

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

In conclusion

$$O(E_B, E_{AB}, (R', S')) = \begin{cases} 1 & \text{if } \alpha_i \text{ is even,} \\ 0 & \text{if } \alpha_i \text{ is odd.} \end{cases}$$

This means that the attacker can iteratively recover the first $n-3$ bits of $\alpha$. For the remaining three bits, the authors suggest to use a brute force method.

*Remark 4.* It is a priori not obvious how to find symplectic transformations which can be used to recover the parity of some bit $\alpha_i$ of the secret scalar. In [6], the authors use the following strategy.

First, they find a transformation $(R', S') \leftarrow (R, S)$ such that

$$\langle R + [\alpha]S \rangle = \langle R' + [\alpha]S' \rangle \quad \Leftrightarrow \quad a_i \text{ is even.}$$

This can be achieved using some clever reverse engineering.

In the second step, the authors multiply $R'$ and $S'$ by a scalar to make the transformation symplectic. This corresponds to multiplying the associated matrix by a scalar such that its determinant is 1. The latter only works if the determinant is a square modulo $2^n$. This condition also prevents the use of the same attack approach for recovering the last three bits.

The first step could be translated directly to the G2SIDH-setting. However, the scaling in the second step is not possible in the genus-2 case. Let $(R_1, R_2, S_1, S_2)$ be an arbitrary basis for $J[2^n]$. Then in general there will not exist an integer $\lambda$ with the property that $(\lambda R_1, \lambda R_2, \lambda S_1, \lambda S_2)$ is symplectic. As a consequence, it was necessary to use a different strategy for finding suitable symplectic transformations.

## B Symplectic basis algorithm

Let $J$ be a PPSSAS over some finite field $\mathbb{F}_{p^2}$ and $m$ an integer not divisible by $p$. Given an arbitrary basis $(R_1, R_2, R_3, R_4)$ for $J[m]$, Algorithm 1 can be used to construct a symplectic basis $(P_1, P_2, Q_1, Q_2)$ for $J[m]$ (cf. §2.2). The algorithm resembles the Gram-Schmidt process for orthonormalisation.

**Algorithm 1:** Converting an arbitrary set of generators of the torsion subgroup to a symplectic basis.

**Data:** Basis $(R_1, R_2, R_3, R_4)$ for $J[m]$
**Result:** Symplectic basis $(P_1, P_2, Q_1, Q_2)$ for $J[m]$
// $P_1$ arbitrary
Set $P_1 \leftarrow R_1$;
// $Q_1$ such that $e(P_1, Q_1) = \zeta_m$
**if** $\mathrm{ord}(e(P_1, R_2)) = m$ **then**
  Set $Q_1 \leftarrow R_2$;
**else if** $\mathrm{ord}(e(P_1, R_3)) = m$ **then**
  Set $Q_1 \leftarrow R_3$;
  Set $R_3 \leftarrow R_2$;
**else**
  Set $Q_1 \leftarrow R_4$;
  Set $R_4 \leftarrow R_2$;
Set $\zeta \leftarrow e(P_1, Q_1)$;
// $P_2, Q_2$ should be ''orthogonal'' to $P_1, Q_1$
Set $\lambda_1 \leftarrow \log(\zeta, e(Q_1, R_3))$, $\lambda_2 \leftarrow \log(\zeta, e(P_1, R_3))$;
Set $P_2 \leftarrow R_3 + [\lambda_1]P_1 - [\lambda_2]Q_1$;
Set $\mu_1 \leftarrow \log(\zeta, e(Q_1, R_4))$, $\mu_2 \leftarrow \log(\zeta, e(P_1, R_4))$, $\mu_3 \leftarrow \log(\zeta, e(P_2, R_4))$;
Set $Q_2 \leftarrow [\mu_3^{-1}](R_4 + [\mu_1]P_1 - [\mu_2]Q_1)$;
Return $(P_1, P_2, Q_1, Q_2)$;

## C Case distinction of kernel subgroups

In this section, we will give some more details on the case distinction strategy sketched in §3.3 to determine the canonical form of normalised generators for an admissible kernel subgroup (cf. Table 1). While Figure 5 gives explicit transformations for each step of the distinction process, we will run through the queries required for classifying one type of rank-3 subgroup as an example, i.e. we will present the queries along one path in the decision trees. The queries required to check the conditions along other paths are very similar and hence omitted.

Suppose Alice's secret is of the form $(\alpha_{1,1}, \ldots, \alpha_{3,4})$. We do not initially know the rank of the group $G_A$ defined by these scalars, which canonical form is obtained when normalising the generators, nor their respective order. We proceed as follows to find the type of $G_A$ according to our classification from §2.3.

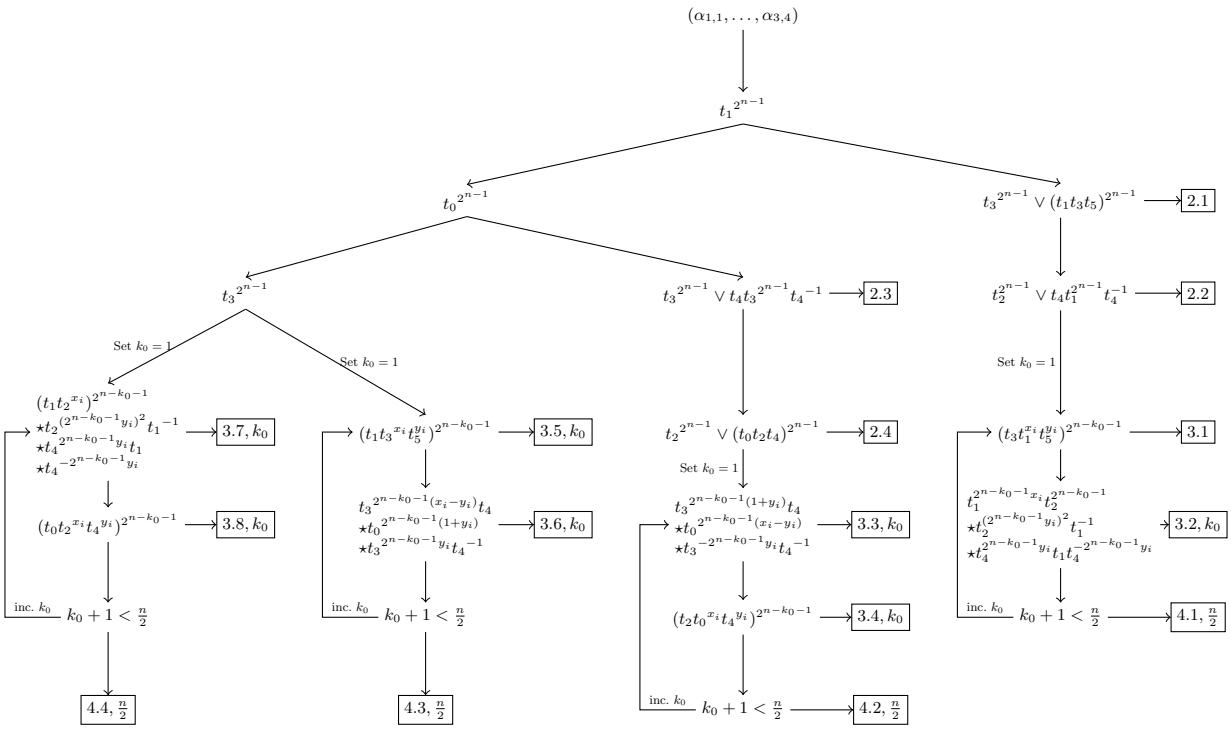


**Fig. 5.** Explicit transformations to supplement strategy for type distinction of normalised kernel generators as displayed in Figure 2 and described in Section 3.3 and Appendix C. At each node, the left path is taken when the query (or one of the queries) leads the oracle to return 1. Else, the right path is taken.

**Step 1.** We start by testing whether $\alpha_{1,1} \equiv \alpha_{2,1} \equiv \alpha_{3,1} \equiv 0 \pmod 2$ with the query

$$\begin{Bmatrix} R_1 + [2^{n-1}]S_1, & R_2, \\ S_1, & S_2 \end{Bmatrix}$$

corresponding to the transformation $t_1$. The kernel is affected in the following way.

$$\begin{aligned}
&\langle [\alpha_{1,1}]R_1' + [\alpha_{1,2}]R_2' + [\alpha_{1,3}]S_1' + [\alpha_{1,4}]S_2', \\
&\ [\alpha_{2,1}]R_1' + [\alpha_{2,2}]R_2' + [\alpha_{2,3}]S_1' + [\alpha_{2,4}]S_2', \\
&\ [\alpha_{3,1}]R_1' + [\alpha_{3,2}]R_2' + [\alpha_{3,3}]S_1' + [\alpha_{3,4}]S_2' \rangle \\
&= \langle [\alpha_{1,1}]R_1 + [\alpha_{1,2}]R_2 + [\alpha_{1,3} + 2^{n-1}\alpha_{1,1}]S_1 + [\alpha_{1,4}]S_2, \\
&\quad [\alpha_{2,1}]R_1 + [\alpha_{2,2}]R_2 + [\alpha_{2,3} + 2^{n-1}\alpha_{2,1}]S_1 + [\alpha_{2,4}]S_2, \\
&\quad [\alpha_{3,1}]R_1 + [\alpha_{3,2}]R_2 + [\alpha_{3,3} + 2^{n-1}\alpha_{3,1}]S_1 + [\alpha_{3,4}]S_2 \rangle.
\end{aligned}$$

The latter is equal to $\phi_B(G_A)$ if and only if $[2^{n-1}][\alpha_{1,1}]S_1, [2^{n-1}][\alpha_{2,1}]S_1, [2^{n-1}][\alpha_{3,1}]S_1 \in \phi_B(G_A)$. This is the case if and only if $\alpha_{1,1}, \alpha_{2,1}$ and $\alpha_{3,1}$ are all even. This leads to the first case distinction depending on the output of the oracle which signifies whether the permutation $\sigma \in D_8$ corresponding to the normalisation of $G_A$ fixes the first basis point:

$$O(J_B, J_{AB}, (R_1', R_2', S_1', S_2')) = \begin{cases} 1 & : \text{cases } (2.3), (2.4), (3.3) - (3.8), (4.2) - (4.4). \\ 0 & : \text{cases } (2.1), (2.2), (3.1), (3.2), (4.1). \end{cases}$$

Assume that the answer is 0. This implies that at least one of the coefficients of $R_1$ is invertible and we can perform a first normalisation step. We obtain elements $\alpha_{i,2}', \alpha_{i,3}', \alpha_{i,4}'$ for $i \in \{1, 2, 3\}$ such that

$$\begin{aligned}
\phi_B(G_A) = \langle R_1 + &[\alpha_{1,2}']R_2 + [\alpha_{1,3}']S_1 + [\alpha_{1,4}']S_2, \\
&[\alpha_{2,2}']R_2 + [\alpha_{2,3}']S_1 + [\alpha_{2,4}']S_2, \\
&[\alpha_{3,2}']R_2 + [\alpha_{3,3}']S_1 + [\alpha_{3,4}']S_2 \rangle.
\end{aligned}$$

**Step 2.** Now we test whether one of $\alpha_{2,2}'$ or $\alpha_{3,2}'$ is invertible. This requires at most two queries. First, we send

$$\begin{Bmatrix} R_1, & R_2 + [2^{n-1}]S_2, \\ S_1, & S_2 \end{Bmatrix}$$

corresponding to $t_3$ to the oracle. Similarly to the strategy above, $O(J, J', (R_1', R_2', S_1', S_2')) = 1$ if and only if all of $\alpha_{1,2}', \alpha_{2,2}', \alpha_{3,2}'$ are even. On the other hand if $O(J, J', (R_1', R_2', S_1', S_2')) = 0$, we only know that at least one of the three coefficients of $R_2$ is odd. We want to distinguish between the two cases where $\alpha_{1,2}'$ is odd and both $\alpha_{2,2}', \alpha_{3,2}'$ are even, or where at least one of $\alpha_{2,2}', \alpha_{3,2}'$ is odd. Therefore, we also send the query

$$\begin{Bmatrix} R_1 + [2^{n-1}]S_1 + [2^{n-1}]S_2, & R_2 + [2^{n-1}]S_1 + [2^{n-1}]S_2, \\ S_1, & S_2 \end{Bmatrix}$$

corresponding to $t_1t_3t_5$. Taking into account that the output of the previous query was 0, the answer of this query is 1 if $\alpha_{1,2}'$ is odd and both $\alpha_{2,2}', \alpha_{3,2}'$ are even, and it is 0 otherwise.

Note that we are done with the case distinction if both of the previous queries returned 0. In that case we can simply normalise the coefficient of $R_2$ to 1 and obtain a representation of the form

$$\phi_B(G_A) = \langle R_1 + [\alpha_{1,3}'']S_1 + [\alpha_{1,4}'']S_2,\ R_2 + [\alpha_{2,3}'']S_1 + [\alpha_{2,4}'']S_2,\ [\alpha_{3,3}'']S_1 + [\alpha_{3,4}'']S_2 \rangle.$$

Using the fact that the group $\phi_B(G_A)$ is isotropic, we see that $\alpha_{3,3}'' = \alpha_{3,4}'' = 0$, and $\phi_B(G_A)$ is a rank-2 group with canonical form 2.1.

On the other hand if one of the queries returned 1, then none of $\alpha'_{2,2}, \alpha'_{3,2}$ are invertible. This leaves the following possibilities for the group structure.

$$2.2 \ \ \langle R_1 + [a]S_1 + [b]R_2,\ S_2 + [b]S_1 + [c]R_2\rangle,$$

where $c$ is even, and the three rank-3 types

$$3.1 \ \ \langle R_1 + [d]R_2 + [a]S_1 + [b]S_2,\ 2^k(R_2 + [b-cd]S_1 + [c]S_2),\ 2^{n-k}([-d]S_1 + S_2)\rangle,$$
$$3.2 \ \ \langle R_1 + [d]S_2 + [a]S_1 + [b]R_2,\ 2^k(S_2 + [cd-b]S_1 + [c]R_2),\ 2^{n-k}([d]S_1 + R_2)\rangle,$$
$$4.1 \ \ \langle R_1 + [d]R_2 + [a]S_1 + [b]S_2,\ 2^{n/2}(R_2 + [cd-b]S_1 + [c]S_2),\ 2^{n/2}([d]S_1 + S_2)\rangle.$$

Note that we can also deduce the parity of $d$ (resp. $b$) for Type 3.1 and 4.1 (resp. Type 3.2) from the previous queries.

**Step 3.** In this step we distinguish between Type 2.2 and the possible rank-3 types. For that purpose, we check whether one of $\alpha'_{2,4}$ or $\alpha'_{3,4}$ is invertible using the transformations

$$\begin{Bmatrix} R_1, & R_2, \\ S_1, & S_2 + [2^{n-1}]R_2 \end{Bmatrix} \text{ and } \begin{Bmatrix} R_1 + [2^{n-1}]R_2 + [2^{n-1}]S_1, & R_2, \\ S_1, & S_2 + [2^{n-1}]R_2 + [2^{n-1}]S_1 \end{Bmatrix}.$$

If the queries show that one (or both) of $\alpha'_{2,4}$ or $\alpha'_{3,4}$ is invertible, then $\phi_B(G_A)$ is of Type 2.2. Otherwise, if both $\alpha'_{2,4}$ and $\alpha'_{3,4}$ are even we know that $\phi_B(G_A)$ is a rank-3 group, and we continue with the next step.

**Step 4.** In order to distinguish between Type 3.1, 3.2 and Type 4.1, we have to compare the elements $\alpha'_{2,2}, \alpha'_{3,2}$ and $\alpha'_{2,4}, \alpha'_{3,4}$. Recall that all of these scalars are necessarily even, hence we can find positive integers $k_{2,2}, k_{2,4}, k_{3,2}, k_{3,4}$ and odd numbers $\beta_{2,2}, \beta_{2,4}, \beta_{3,2}, \beta_{3,4}$ such that $\alpha'_{i,j} = 2^{k_{i,j}}\beta_{i,j}$ for $(i,j) \in I = \{(2,2),(2,4),(3,2),(3,4)\}$. Our goal is to determine

$$k = \min\{k_{i,j} \mid (i,j) \in I\}.$$

This minimum can be found iteratively. We start with $k_0 = 1$ and increase $k_0$ by one if the following queries are not successful. Before describing the queries, note that

$$2^{n-k}\langle R_2 + [\alpha'_{1,4}]S_1,\ S_2 - [\alpha'_{1,2}]S_1\rangle \subset \phi_B(G_A) \tag{3}$$

as per Corollary 1. Property (3) will be used multiple times in this step.

The first query is to test whether $\phi_B(G_A)$ is of Type 3.1. Recall that we know the parity of $\alpha'_{1,2}$ and $\alpha'_{1,4}$ from the previous queries. Hence, for $k_0 = 1$, we only need to send one of the following queries. For each iterative step however, we have determined $\alpha'_{1,2} \pmod{2^{k_0}}$ and $\alpha'_{1,4} \pmod{2^{k_0}}$ through previous queries, and send the following query

$$\begin{Bmatrix} R_1 + [2^{n-k_0-1}]([x_i]S_1 + [y_i]S_2),\ R_2 + [2^{n-k_0-1}]([y_i]S_1 + S_2), \\ S_1, \qquad\qquad S_2 \end{Bmatrix}$$

first with $y_1 = -\alpha'_{1,2} \pmod{2^{k_0}}$ and $x_1 = y_1^2$. Using Property (3), this leaves $\phi_B(G_A)$ unchanged exactly if the $k_0$-th bit of $\alpha'_{1,2}$ equals 0 and $2^{k_0+1}$ divides $\alpha'_{2,2}$ and $\alpha'_{3,2}$, i.e. $k_{2,2} > k_0$ and $k_{3,2} > k_0$. If the oracle query returns 0, we resend the query with $y_2 = -\alpha'_{1,2} \pmod{2^{k_0}} - 2^{k_0}$ and $x_2 = y_2^2$. If these malformed points produce a group distinct from $\phi_B(G_A)$, we can deduce that one of $k_{2,2}, k_{3,2}$ equals $k_0$. Hence, the group must be of Type 3.1 with $k = k_0$. Otherwise, $O(J, J', (R'_1, R'_2, S'_1, S'_2)) = 1$ implies that we have determined the correct next bit of $\alpha'_{1,2}$ and that $k_{2,2} > k_0$ and $k_{3,2} > k_0$. We thus proceed with the next set of queries to test whether the group is of Type 3.2. We send the two queries

$$\begin{Bmatrix} R_1 + [2^{n-k_0-1}]([x_i]S_1 + [y_i]R_2), & R_2, \\ S_1, & S_2 + [2^{n-k_0-1}]([-y_i]S_1 + R_2) \end{Bmatrix}$$

with $y_1 = -\alpha'_{1,4} \pmod{2^{k_0}}$, $y_2 = -\alpha'_{1,4} \pmod{2^{k_0}} - 2^{k_0}$ and $x_i = -y_i^2$. With analogous reasoning as before, we can deduce from the oracle responses whether both $k_{2,4} > k_0$ and $k_{3,4} > k_0$, and learn the next bit of $\alpha'_{1,4}$ if any query returns 1. Thus we can either determine the type of $\phi_B(G_A)$ to be 3.2 with $k = k_0$, or increase $k_0$ by 1 and repeat the queries in this step.

If we have not managed to determine that $\phi_B(G_A)$ is of Type 3.1 or 3.2 with $k_0 = \lfloor \frac{n}{2} \rfloor$, we conclude that indeed $\phi_B(G_A)$ must be of Type 4.1 with $k = \frac{n}{2}$.

# D A first adaptive attack on G2SIDH

In this section, we give a first set of malformed points to query the oracle with in order to recover secret G2SIDH scalars. This version of the attack can be detected by the honest party, but we hope the details given below will be useful for future efforts to cryptanalyze similar schemes.

We use the It interacts with an oracle by sending the oracle points on some starting variety that correspond to the auxiliary points provided in the protocol. By sending malformed points, the adaptive attack is able to recover scalars that determine the secret kernels.

Again, we assume that all users of the G2SIDH protocol (or at least Alice) are using a symplectic basis as this allows us to work with the canonical representations of kernel generators and refer to §3.6 for how to translate the attack to a setting where Alice uses an arbitrary basis.

*Remark 5.* Note that Alice is able to detect the following attack easily if she checks the Weil pairing of the points she receives from the other party. If Bob sends honestly generated points

$$\begin{Bmatrix} R_1 = \phi_B(P_1),\ R_2 = \phi_B(P_2), \\ S_1 = \phi_B(Q_1),\ S_2 = \phi_B(Q_2) \end{Bmatrix}$$

then the only non-trivial pairings should satisfy

$$e_{2^n}(R_1, S_1) = e_{2^n}(R_2, S_2) = \zeta^{\deg \phi_B},$$

where $\zeta = e_{2^n}(P_1, Q_1)$. Furthermore, some of the kernels obtained from the malformed points in the attack on rank-3 kernels do not always generate maximal $2^n$-isotropic subgroups. In these cases, the honest party will not be able to compute a valid variety from the points provided. Therefore, checking whether the points received from the other party generate an admissible subgroup is a further means of detecting an attack such as the one below.

## D.1 Kernels of rank 2

If the kernel of Alice's isogeny has rank 2 (i.e. when $k = 0$), we again assume that the kernel generators $Q_1$ and $Q_2$ can be written in the form

$$\begin{aligned} A_1 &= R_1 \quad\ \ + [a]S_1 + [b]S_2 \text{ and} \\ A_2 &= \quad R_2 + [b]S_1 + [c]S_2 \end{aligned}$$

for some $(a, b, c) \in [2^n]^3$ in accordance with the sampling of keys for symplectic bases as discussed in §2.4 and §3.4.

**Recovering the first bits.** With access to the oracle $O$, we at most need the following four queries:

To determine the parity of $a$ and $b$, we send multiple queries.

First, we check whether $a_0 = b_0 = 0$ by sending

$$\begin{Bmatrix} R_1, & R_2, \\ [1 + 2^{n-1}]S_1, & S_2 \end{Bmatrix}.$$

In this case $\langle A_1', A_2' \rangle = \langle A_1 + [2^{n-1}][a_0]S_1, A_2 + [2^{n-1}][b_0]S_1 \rangle$ is the same torsion-subgroup as $\langle A_1, A_2 \rangle$ exactly when both scalars in question are even.

Second, we can check whether $a_0 = 1$ and $b_0 = 0$ by sending points

$$\begin{Bmatrix} R_1 - [2^{n-1}]S_1, & R_2, \\ [1 + 2^{n-1}]S_1, & S_2 \end{Bmatrix}.$$

This results in Alice calculating with the subgroup $\langle A_1', A_2' \rangle = \langle A_1 + [2^{n-1}][a_0 - 1]S_1, A_2 + [2^{n-1}][b_0]S_1 \rangle$. This subgroup is equal to $\langle A_1, A_2 \rangle$ and corresponds to an isogeny with codomain variety $J_{AB}$ if and only if $a$ is odd while $b$ is even.

Third, we send points

$$\begin{Bmatrix} R_1, & R_2 - [2^{n-1}]S_1, \\ [1 + 2^{n-1}]S_1, & S_2 \end{Bmatrix}$$

to check for $a_0 = 0$ and $b_0 = 1$ which produce the desired variety exactly when the scalar $a$ is even and $b$ is odd.

If all three oracle queries fail (i.e. do not yield the same variety as $J_{AB}$), we can conclude that both scalars in question are odd, so $a_0 = b_0 = 1$.

Once we have recovered $a_0$ and $b_0$, we can send

$$\begin{Bmatrix} R_1 - [2^{n-1}][b_0]S_2, & R_2, \\ S_1, & [1 + 2^{n-1}]S_2 \end{Bmatrix}.$$

Then the first kernel generator $A_1$ is unchanged and the second one will hold information about $c_0$. Explicitly, this means $\langle A_1', A_2' \rangle = \langle A_1, A_2 + [2^{n-1}][c_0]S_2 \rangle$. Hence we can conclude that $c_0 = 0$ when the oracle returns a match between the computed variety and $J_{AB}$, while $c_0 = 1$ otherwise.

**Iterative step.** Suppose now that we have learned the first $i$ bits of each key scalar, so we know $K_i^a, K_i^b$ and $K_i^c$, where $a = \sum_{j=0}^{n-1} 2^j a_j = K_i^a + \sum_{j=i}^{n-1} 2^j a_j$ (and similarly for $b$ and $c$).

Again, we can use three different sets of malformed points to determine $a_i$ and $b_i$, and then learn $c_i$ with one further query.

Sending malformed points

$$\begin{Bmatrix} R_1 - [2^{n-i-1}][K_i^a]S_1, & R_2 - [2^{n-i-1}][K_i^b]S_1, \\ [1 + 2^{n-i-1}]S_1, & S_2 \end{Bmatrix}$$

leads to Alice's internal computation including

$$\begin{aligned} \langle A_1', A_2' \rangle &= \langle A_1 + [2^{n-i-1}][a - K_i^a]S_1, A_2 + [2^{n-i-1}][b - K_i^b]S_1 \rangle \\ &= \langle A_1 + [2^{n-i-1}][2^i a_i]S_1, A_2 + [2^{n-i-1}][2^i b_i]S_1 \rangle \\ &= \langle A_1 + [2^{n-1}][a_i]S_1, A_2 + [2^{n-1}][b_i]S_1 \rangle . \end{aligned}$$

For the oracle to return a match between varieties, we require $a_i$ and $b_i$ both to be zero.

If required, we proceed with sending malformed points

$$\begin{Bmatrix} R_1 - [2^{n-i-1}][K_i^a + 2^i]S_1, & R_2 - [2^{n-i-1}][K_i^b]S_1, \\ [1 + 2^{n-i-1}]S_1, & S_2 \end{Bmatrix}$$

and

$$\begin{Bmatrix} R_1 - [2^{n-i-1}][K_i^a]S_1, & R_2 - [2^{n-i-1}][K_i^b + 2^i]S_1, \\ [1 + 2^{n-i-1}]S_1, & S_2 \end{Bmatrix}$$

which yield subgroups of the form

$$\begin{aligned}\langle A_1', A_2'\rangle &= \langle A_1 + [2^{n-i-1}][a - K_i^a - 2^i]S_1, A_2 + [2^{n-i-1}][b - K_i^b]S_1\rangle \\ &= \langle A_1 + [2^{n-1}][a_i - 1]S_1, A_2 + [2^{n-1}][b_i]S_1\rangle\,,\end{aligned}$$

$$\begin{Bmatrix} R_1 - [2^{n-i-1}][K_i^a]S_1,\ R_2 - [2^{n-i-1}][K_i^b + 2^i]S_1, \\ [1 + 2^{n-i-1}]S_1, \qquad\qquad S_2 \end{Bmatrix}$$

respectively. These coincide with kernels generating $J_{AB}$ when we have $a_i = 1$ and $b_i = 0$,and $a_i = 0$ and $b_i = 1$, respectively. Again, we conclude that $a_i = b_i = 1$ if the previous queries did not show a match.

Hence, we have recovered $K_{i+1}^a$ and $K_{i+1}^b$. To determine $c_i$ and therefore $K_{i+1}^c$, we can send points of the form

$$\begin{Bmatrix} R_1 - [2^{n-i-1}][K_{i+1}^b]S_2,\ R_2 - [2^{n-i-1}][K_i^c]S_2, \\ S_1, \qquad\qquad [1 + 2^{n-i-1}]S_2 \end{Bmatrix}.$$

Then the first kernel generator is unchanged since

$$\begin{aligned}A_1' &= A_1 + [2^{n-i-1}][a - K_{i+1}^a]S_2 \\ &= A_1 + [2^{n-i-1}][\sum_{j=i+1}^{n-1} 2^j a_j]S_2 \\ &= A_1,\end{aligned}$$

while the second generates the subgroup

$$\begin{aligned}\langle A_2'\rangle &= \langle A_2 + [2^{n-i-1}][c - K_i^c]S_2\rangle \\ &= \langle A_2 + [2^{n-1}][c_i]S_2\rangle.\end{aligned}$$

Therefore, if the oracle confirms the computed variety is equivalent to $J_{AB}$, we can deduce that $c_i = 0$, and $c_i = 1$ otherwise.

This way, we can recover the full key. However, additionally to being detectable by pairing checks, the last two bits of each key scalar will need to be brute forced since the last two steps would send points of incorrect order.

*Remark 6.* Note that, for example, the queries recovering the first bits $a_0$ and $b_0$ can easily be made undetectable. Instead of scaling a single point and hence altering the corresponding pairing values, the transformation $tr_0$ can be applied multiple times to the honestly generated points instead. It is not as straightforward for all sets of malformed points to obtain Weil pairing-compatible versions thereof, but finding (compositions of) symplectic transformations which produce kernel generators exposing the desired bits lead to the attack outlined in Section 3.4.

### D.2 Kernels of rank 3

Now suppose that Alice's secret kernel subgroup has three generators, i.e. $k > 0$. Let $1 \leq k \leq \lfloor \frac{n}{2} \rfloor$ be fixed. Then the generators are of the form

$$\begin{aligned}A_1 = &\quad R_1 + [d]R_2 + [a]S_1 \quad + [b]S_2, \\ A_2 = 2^k( &\quad R_2 \quad + [b - dc]S_1 + [c]S_2) \text{ and} \\ A_3 = 2^{n-k}( &\quad\quad - [d]S_1 \quad + S_2)\end{aligned}$$

for some $(a, b, c, d) \in [2^n] \times [2^{n-k}] \times [2^{n-2k}] \times [2^k]$.

As we desrcibed in §3, we again make the assumption that $k$ is guessed to equal zero first and then iterate over increasing values of $k$ until an isogeny is recovered which produces the correct codomain. On average, an attacker will not need to repeat the attack too often since 99% of the keyspace has $k \leq 3$.

**Recovering first bits of $a$, $b$ and $d$.** We first recover the parity of the secret scalars $a, b$ and $d$ by sending the malformed points

$$\begin{Bmatrix} R_1, & R_2, \\ [1+2^{n-1}]S_1, & S_2 \end{Bmatrix}, \quad \begin{Bmatrix} R_1, & R_2, \\ S_1, & [1+2^{n-1}]S_2 \end{Bmatrix}, \quad \text{and} \quad \begin{Bmatrix} R_1, & [1+2^{n-1}]R_2, \\ S_1, & S_2 \end{Bmatrix}$$

respectively. By sending the first set of malformed points, we can recover the bit $a_0$ since Alice performs the following internal computations

$$\begin{aligned} A'_1 &= A_1 + [2^{n-1}][a]S_1 &&= A_1 + [2^{n-1}][a_0]S_1, \\ A'_2 &= A_2 + [2^{n-1}][2^k][b-dc]S_1 &&= A_2, \\ A'_3 &= A_3 + [2^{n-1}][2^{n-k}][-d]S_1 &&= A_3, \end{aligned}$$

allowing her to recover a variety $J'_{AB}$. The second and third kernel generator are unchanged from the computation yielding the original variety $J_{AB}$, and we know that $A'_1 = A_1$ if and only if $a_0 = 0$. Hence we can deduce the value of $a_0$ from comparing $J'_{AB}$ and $J_{AB}$, i.e. $a_0 = 0$ whenever the varieties are the same and $a_0 = 1$ otherwise.

Analogously, when sending the malformed points querying for $b_0$ and $d_0$, respectively, only the first kernel generator $A'_1$ is potentially different from $A_1$ and allows an attacker to read off the bit in question.

**Iterative step for recovering partial keys of $a$, $b$ and $d$ while $i < k$.** Assume we have so far recovered the first $i$ key bits of $a, b$ and $d$, $K_i^a, K_i^b$ and $K_i^d$, where $K_i^x + 2^i x_0 + \sum_{j=i+1}^{n-1} 2^j x_j = x$ for any $x \in \{a, b, d\}$, for some $i < k$.

Sending the malformed points

$$\begin{Bmatrix} R_1 - [2^{n-i-1}K_i^a]S_1, & R_2, \\ [1+2^{n-i-1}]S_1, & S_2 \end{Bmatrix}$$

leads Alice to use kernel generators $A'_1, A'_2$ and $A'_3$ during her isogeny computation as follows:

$$\begin{aligned} A'_1 &= A_1 + [2^{n-i-1}][a - K_i^a]Q_1 &&= A_1 + [2^{n-1}][a_i]S_1, \\ A'_2 &= A_2 + [2^{n-i-1}][2^k][b-dc]Q_1 &&= A_2, \\ A'_3 &= A_3 + [2^{n-i-1}][2^{n-k}][-d]Q_1 &&= A_3, \end{aligned}$$

where the equalities for $A'_2$ and $A'_3$ follow from the fact that $i + 1 \leq k$.

Similarly, we can send the points

$$\begin{Bmatrix} R_1 - [2^{n-i-1}K_i^b]S_2, & R_2, \\ S_1, & [1+2^{n-i-1}]S_2 \end{Bmatrix} \quad \text{and} \quad \begin{Bmatrix} R_1 - [2^{n-i-1}K_i^d]R_2, & [1+2^{n-i-1}]R_2, \\ S_1, & S_2 \end{Bmatrix}$$

producing the kernels $\langle A_1 + [2^{n-1}][b_i]S_2, A_2, A_3\rangle$ and $\langle A_1 + [2^{n-1}][d_i]R_2, A_2, A_3\rangle$, respectively, to recover the bits $b_i$ and $d_i$.

Hence, this procedure allows an attacker to recover the partial keys $K_k^a, K_k^b$ as well as $K_k^d = d$.

**Recovering remaining scalars if $k = \frac{n}{2}$.** Note first that if $k = n/2$, $c = 0$ since it is a value modulo $2^{n-2k} = 2^0$. Further, after recovering $K_k^a, K_k^b$ and $K_k^d$ as above we have both $b = K_k^b$ and $d = K_k^d$. Hence, it only remains to recover the more significant bits of $a$, namely $a_k, \ldots, a_{n-1}$.

So for $k \leq i < n - 3$, we can query iteratively with the malformed points

$$\begin{Bmatrix} R_1 - [2^{n-i-1}][K_i^a]S_1, & R_2 - [2^{n-i-1}][b]S_1, \\ [1+2^{n-i-1}]S_1, & S_2 + [2^{n-i-1}][d]S_1 \end{Bmatrix}.$$

Then Alice's internal computation uses as kernel generators

$$\begin{aligned} A_1' &= A_1 + [2^{n-i-1}][-K_i^a - bd + a + bd]S_1 \\ &= A_1 + [2^{n-i-1}][a - K_i^a]S_1 \\ &= A_1 + [2^{n-1}][a_i]S_1 \end{aligned}$$

and the unchanged points

$$\begin{aligned} A_2' &= A_2 + [2^{n-i-1}][2^k][-b + b - dc + dc]S_1 = A_2, \\ A_3' &= A_3 + [2^{n-i-1}][2^k][-d + d]S_1 \qquad\quad = A_3. \end{aligned}$$

Hence, if Alice computes a kernel resulting in the variety $J_{AB}$, we can conclude that the first kernel generator was also unchanged, yielding $a_i = 0$. Otherwise, we may deduce that $a_i = 1$. This way, we can recover $K_{n-2}^a$ and brute-force the two most significant bits of $a$.

From now on, we assume that $0 < k < \frac{n}{2}$.

**Recovering $c_0, b_k$ and $a_k$.** Sending queries with malformed points

$$\begin{Bmatrix} R_1 - [2^{n-k-1}][K_k^b]S_2, & R_2, \\ S_1, & [1 + 2^{n-k-1}]S_2 \end{Bmatrix} \text{ and } \begin{Bmatrix} R_1 - [2^{n-k-1}][K_k^b + 2^k]S_2, & R_2, \\ S_1, & [1 + 2^{n-k-1}]S_2 \end{Bmatrix}$$

leads to the oracle computing varieties corresponding to the kernels

$$\begin{aligned} &\langle A_1 + [2^{n-1}][b_k]S_2, A_2 + [2^{n-1}][c_0]S_2, A_3\rangle \text{ and} \\ &\langle A_1 + [2^{n-1}][b_k - 1]S_2, A_2 + [2^{n-1}][c_0]S_2, A_3\rangle \end{aligned}$$

respectively. If $c_0 = 0$, then one of the resulting varieties must equal $J_{AB}$, depending on the value of $b_k$. If neither query gives a match with $J_{AB}$, we can conclude $c_0 = 1$, but do not learn $b_k$ as in the case where $c_0 = 0$ straightaway.

However, we can send another query to the oracle with points

$$\begin{Bmatrix} (R_1 + [dc_0 - K_k^b][2^{n-k-1}]S_2, & R_2 - [2^{n-k-1}c_0]S_2, \\ S_1, & [1 + 2^{n-k-1}]S_2) \end{Bmatrix}$$

where $c_0 = 1$ to prompt Alice to use

$$\langle A_1 + [2^{n-1}][b_k]S_2, A_2, A_3\rangle$$

during her computation. Clearly, this results in computing the variety $J_{AB}$ exactly when $b_k = 0$, and in a different variety or a failure to compute otherwise.

The next bit of $a$ can be recovered by sending malformed points

$$\begin{Bmatrix} R_1 - [2^{n-k-1}][d(d_0c_0 - b_0) + K_k^a]S_1, & R_2 - [2^{n-k-1}][b_0 - d_0c_0]S_1, \\ [1 + 2^{n-k-1}]Q_1, & S_2 \end{Bmatrix}.$$

The kernel subgroup corresponding to these points is generated by $A_1' = A_1 + [2^{n-1}][a_k]S_1$, $A_2' = A_2$ and $A_3' = A_3$. Again we can conclude that $a_k = 0$ if and only if the oracle computes a variety matching $J_{AB}$.

**Iterative step for recovering remaining scalars when $i \geq k+1$.** (Here, we use different notation for the partial scalar $c$: $K_i^c = \sum_{j=0}^{i-k-1} 2^j c_j$, so that e.g. $K_{k+1}^c = c_0$.)

As before, the bits $b_i$ and $c_{i-k}$ have to be recovered simultaneously. We again send three queries to check for distinct bit combinations for $(c_{i-k}, b_i)$. We first try

$$\begin{Bmatrix} R_1 + [2^{n-i-1}][dK_i^c - K_i^b]S_2,\ R_2 - [2^{n-i-1}][K_i^c]S_2, \\ S_1, \qquad\qquad [1+2^{n-i-1}]S_2 \end{Bmatrix} \text{ and } \begin{Bmatrix} R_1 + [2^{n-i-1}][dK_i^c - K_i^b - 2^i]S_2,\ R_2 - [2^{n-i-1}][K_i^c]S_2, \\ S_1, \qquad\qquad [1+2^{n-i-1}]S_2 \end{Bmatrix}$$

which yield kernels $\langle A_1 + [2^{n-1}][b_i]S_2, A_2 + [2^{n-1}][c_{i-k}]S_2, A_3\rangle$ and $\langle A_1 + [2^{n-1}][b_i - 1]S_2, A_2 + [2^{n-1}][c_{i-k}]S_2, A_3\rangle$, respectively. Hence, we can conclude $(c_{i-k}, b_i) = (0,0)$ or $(c_{i-k}, b_i) = (0,1)$ if one of the resulting varieties is $J_{AB}$.

Hence, we have either determined that $c_{i-k} = 0$ or can deduce that $c_{i-k} = 1$. Therefore, we now know $K_{i+1}^c$ and can send a last set of malformed points

$$\begin{Bmatrix} R_1 + [2^{n-i-1}][dK_{i+1}^c - K_i^b]S_2,\ R_2 - [2^{n-i-1}][K_{i+1}^c]S_2, \\ S_1, \qquad\qquad [1+2^{n-i-1}]S_2 \end{Bmatrix}$$

to determine the bit $b_i$ from the kernel $\langle A_1 + [2^{n-1}][b_i]S_2, A_2, A_3\rangle$ Alice uses in her computation.

For recovering $a_i$ for $i \leq n-3$, we again send

$$\begin{Bmatrix} R_1 - [2^{n-i-1}][K_i^a]S_1,\ R_2 - [2^{n-i-1}][K_{i+1}^b]S_1, \\ [1+2^{n-i-1}]S_1, \qquad S_2 + [2^{n-i-1}][d]S_1 \end{Bmatrix}.$$

This query uses the kernel $\langle A_1 + [2^{n-1}][a_i]S_1, A_2, A_3\rangle$ and we obtain the variety $J_{AB}$ exactly when $a_i = 0$. The remaining two bits can easily be brute-forced.

Note that we will have determined both $b$ and $c$ fully after the $(n-k-1)$-th iterative step and only need to iterate the single query for $a_i$ when $i \leq n-k$ using $b = K_{i+1}^b$ and $c = K_{i+1}^c$.