Secret Keys in 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 analyze 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. More generally, using symplectic bases enables us to classify and enumerate isogeny kernel subgroups and thus simplify the secret sampling step for general genus-2 SIDH-style constructions.
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, forcing the use of the Fujisaki–Okamoto transform for CCA2-security. 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 [NIS17]. The SIDH protocol was first described in 2011 by Jao and De Feo [JF11]. The G2SIDH key exchange [FT19] 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 ℓ-isogenies are replaced by (ℓ, ℓ)-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 the original description of G2SIDH, the secret keys were encoded by multiple secret scalars that need to fulfil a certain linear congruence property and it required the user to solve these linear congruences during the selection of secrets. This is cumbersome and increases the computational cost of key exchange during run time. Moreover, random sampling of keys was left as an open problem in [FT19].

We propose a simplification of the methods in [FT19] which makes the sampling procedure more straightforward. This is achieved by introducing specific conditions on the torsion basis points. In particular, we require the users to choose torsion generators which form a *symplectic* basis. The use of a symplectic torsion basis enables us to classify general genus-2 isogeny kernel subgroups and find canonical expressions for their generators in terms of the suggested basis points. It allows both parties of the G2SIDH key exchange protocol to choose secrets uniformly from a large keyspace simply by choosing 3 to 4 scalars, and furthermore provides a framework in which to present the adaptive attack.

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 [FT19], 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 on SIDH [GPST16] 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 [GPST16]. This attack only requires that a decryption oracle returns a single bit of information at a time. In fact, the decryption oracle can be viewed as a decisional Diffie–Hellman oracle, 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 strength 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 [DGL$^+$20], a different cryptosystem will necessitate modifications in the adaptive attack. This continues to hold true in the adaptive attack on G2SIDH.

The authors of [FT19] 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. This required a thorough analysis of the keyspace.

Contributions In this paper we present the two results:

- Use of symplectic bases enabling a classification of the isogeny kernel subgroups appearing in the genus-2 SIDH protocol. This allows us to generate secret keys without needing to solve cumbersome linear modular equations. Moreover, we propose a simplified version of secret selection that allows for the uniform selection of keys from a restricted (but large enough) keyspace. This was not addressed in the original paper [FT19]. Furthermore, the classification provides a framework for the G2SIDH attack to be carried out.
- 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 bypass 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 [FO13].

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. 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 [GPST16] with respect to our attack framework, we refer to Appendix A. More details on the type distinction algorithm for valid secret subgroups can be found in Appendix C, while Appendix D presents a first approach for an adaptive attack. While the malformed points do not pass the Weil pairing countermeasure, it serves as a good motivation for the attack presented in Section 3.

Note that this is the full version of the paper accepted to SAC 2021. In this version, we provide a contextualisation of our methods and results in the elliptic curve setting as well as supplementary material to the adaptive attack presented in this paper in the appendix.

An implementation of the attack can be found in `https://github.com/yanboti/G2SIDHAdaptiveAttack`.

Acknowledgements We would like to thank Christophe Petit and Tanja Lange for the helpful discussions. We would also like to thank Ben Smith and the anonymous reviewers at SAC for their insightful comments and advice to improve the exposition of our results. The first author was supported by the Deutsche Forschungsgemeinschaft (DFG) Cluster of Excellence 2092 CASA.

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 [Mil86], and isogeny-based cryptography using principally polarised abelian varieties of dimension two instead of elliptic curves when describing G2SIDH as in [FT19]. We assume that the reader is familiar with the general concept of elliptic curves, but refer to [Sil09], for example, for an in-depth discussion of this topic.

1.1 PPSSAS

Let p and ℓ 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 [FT19, 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 ℓ^n-torsion group of A and $\boldsymbol{\mu}_{\ell^n}$ denotes the group of ℓ^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 ℓ^n-isotropic subgroup). *Let A be an abelian variety and K a proper subgroup of $A[\ell^n]$. Then we call G a* maximal ℓ^n-isotropic subgroup *if*

(i) the ℓ^n-Weil pairing (on $A[\ell^n]$) restricts trivially to G, and
(ii) G is a maximal subgroup with respect to Property (i).

As shown by Flynn and Ti, principal polarisations of PPAS are preserved under isogenies whose kernel is a maximal ℓ^n-isotropic subgroup, hence any maximal ℓ^n-isotropic subgroup of A defines an isogeny between PPAS.

Remark 1. An isogeny $\phi : A \to A'$ defined by an ℓ^n-isotropic subgroup G can be represented as a sequence $\phi_1, \dots, \phi_n$ of (ℓ, ℓ)-isogenies between PPAS, each defined by a kernel generated by two order-ℓ elements. In this paper, we are only interested in non-backtracking isogenies, more precisely, we exclude sequences of isogenies that contain both an (ℓ, ℓ)-isogeny ϕ_i and its dual $\phi_j = \widehat{\phi_i}$. This is equivalent to the condition

$$G \not\subset A[m] \quad \text{for any } m < \ell^n .$$

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 divisible 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

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

The curve H is a double cover of the elliptic curve $E : y^2 = x^3 + 1$, given by

$$\begin{aligned} \phi_1 : E &\to H & \text{and} \quad \phi_2 : E &\to H \\ (x, y) &\mapsto (x^2, y), & (x, y) &\mapsto (x^{-2}, yx^{-3}). \end{aligned}$$

These maps induce a $(2, 2)$-isogeny from $E \times E$ to $J_H := \mathrm{Jac}(H)$ [CF96, p. 155].

Also, we have that E is supersingular and $\#E(\mathbb{F}_{p^2}) = (p + 1)^2$ since $p \equiv 2 \pmod 3$ by computing the criterion in [Sil09, Thm. V.4.1(a)]. Hence the Jacobian $J_H = \mathrm{Jac}(H)$ is indeed a PPSSAS. As a consequence $\#J_H(\mathbb{F}_{p^2}) = (p + 1)^4$ using the theorem of Tate [Tat66, Thm. 1]. Furthermore, we have that $J_H(\mathbb{F}_{p^2}) = J_H[2^{e_A}] \times J_H[3^{e_B}] \times J_H[f]$ as a group.

Then a short random walk is taken from J_H in the $(2, 2)$-isogeny graph to obtain a random PPSSAS J. It follows from the above that we can fix torsion-bases $\langle P_1, P_2, P_3, P_4\rangle = J[2^{e_A}]$ and $\langle R_1, R_2, R_3, R_4\rangle = J[3^{e_B}]$ with all points P_i and R_i defined over $\mathbb{F}_{p^2}$.

To perform the key exchange, Alice chooses a secret maximal 2^{e_A}-isotropic subgroup G_A of $J[2^{e_A}]$ corresponding to an isogeny ϕ_A with kernel G_A and codomain J_A. This kernel can be determined by three generators (one of which may be $\mathcal{O}$), given as linear combinations of the P_i known only to Alice. In particular, each generator is determined by the four coefficients of the P_i and thus, Alice's secret can be described explicitly by a collection of secret scalars that is dependent on the basis.[4] 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. He analogously completes his side of the computation so that Alice 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 torsion points with the secret scalars as coefficients which generate the kernel of her secret isogeny ϕ_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. This allows for the use of isomorphism invariants as the shared key.

Flynn and Ti outline a procedure to select Alice's scalars $\alpha_{i,1}, \dots, \alpha_{i,4} \in \mathbb{Z}/2^{e_A}\mathbb{Z}, 1 \le i \le 3$, such that the points

$$A_1 = \sum_{i=1}^{4} [\alpha_{1,i}]P_i, \quad A_2 = \sum_{i=1}^{4} [\alpha_{2,i}]P_i, \text{ and } A_3 = \sum_{i=1}^{4} [\alpha_{3,i}]P_i$$

generate a maximal 2^{e_A}-Weil isotropic subgroup of $J[2^{e_A}]$ which can be used as her secret $G_A = \langle A_1, A_2, A_3\rangle$. For details, specifically which congruences need to be satisfied to obtain a trivial Weil pairing on the A_i, we refer to [FT19]. We discuss a more efficient method of secret selection in §2.3 and also address random sampling from the keyspace, which was left as an open problem.

[4] Throughout the remainder of this work, we will use different encodings of Alice's secret subgroup G_A, such as a description in terms of scalars or as an isogeny.

2 Keyspace

A secret key of Alice can be expressed as 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. Moreover, we require that the isogeny is non-backtracking (cf. Remark 1). This allows us to identify the keyspace with the set

$$\mathcal{K}_\ell = \{G \subset J \mid G \text{ maximal } \ell^n\text{-isotropic and } G \not\subset J[m] \text{ for any } m < \ell^n\},$$

for $\ell \in \{2, 3\}$ and $n = e_A$ or e_B, respectively. The groups in $\mathcal{K}_\ell$ can be specified as follows.

Proposition 1 ([FT19, Prop. 2]). *Let $G \in \mathcal{K}_\ell$, then G is isomorphic to*

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

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, the groups in $\mathcal{K}_\ell$ are not cyclic.

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 us to classify the groups in $\mathcal{K}_\ell$ and define canonical generators. This is achieved by considering so-called symplectic bases of the ℓ^n-torsion of J.

2.1 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 tuple (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 ζ is some primitive m-th root of unity and $\delta_{ij} = \begin{cases} 1 & \textit{if } i = j, \\ 0 & \textit{otherwise.} \end{cases}$

Note that $e_m(Q_j, P_i) = \zeta^{-\delta_{ij}}$ for a symplectic basis (P_1, P_2, Q_1, Q_2) since the Weil-pairing is alternating.

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 $J[m]$ with respect to some primitive root ζ, 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 $J'[m]$ with respect to $\zeta^{\deg(\phi)}$.*

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 $J'[m]$. □

2.2 Classification of secret keys

In this section we suggest a normalisation algorithm that produces canonical generators for each group $G \in \mathcal{K}_\ell$. For this purpose, we let

$$(P_1^*, P_2^*, P_3^*, P_4^*) := (P_1, P_2, Q_1, Q_2)$$

be a symplectic basis for $J[\ell^n]$.[5] Of course, one can adapt the procedure to general bases by performing a basis change to a symplectic basis before applying the algorithm.

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 ℓ^n-isotropic. This property imposes different conditions on the scalars $(\alpha_{1,1}, \ldots, \alpha_{3,4})$.

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

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

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

Note that this normalisation procedure does not affect the corresponding group. More precisely, 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

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

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 ℓ^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 $a \in \{0, \ldots, \ell^n - 1\}$, $b \in \{0, \ldots, \ell^{n-k} - 1\}$, $c \in \{0, \ldots, \ell^{n-2k} - 1\}$, $d \in \{0, \ldots, \ell^k - 1\}$ such that*

$$A \sim_\sigma A' = \begin{pmatrix} 1 & d & a & b \\ 0 & \ell^k & s_\sigma \, \ell^k \, (b - cd) & \ell^k \, c \\ 0 & 0 & -s_\sigma \, \ell^{n-k} \, d & \ell^{n-k} \end{pmatrix} \in M_{3,4}(\mathbb{Z}/\ell^n\mathbb{Z}),$$

[5] The notation $(P_1^*, P_2^*, P_3^*, P_4^*)$ is necessary because we are going to work with permutations of the basis elements. It is only used in this part of the notes.

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

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 ℓ^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, one sees that σ can be chosen to lie in the dihedral group $D_8 = \langle (1234), (13) \rangle$.[6]

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 $a, b, c, d, x, y \in \{0, \dots, \ell^n - 1\}$, now including the divisibility by ℓ-powers which was omitted in (1). First, note that after adding a multiple of the second line to the first line of A', we may assume that $d \in \{0, \dots, \ell^k - 1\}$. Similarly, we can achieve $b \in \{0, \dots, \ell^{n-k} - 1\}$ and $c \in \{0, \dots, \ell^{n-2k} - 1\}$. 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

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

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 ℓ^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*

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

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, σ), where

$$a \in \{0, \dots, \ell^n - 1\},\ b \in \{0, \dots, \ell^{n-k} - 1\},\ c \in \{0, \dots, \ell^{n-2k}\},\ d \in \{0, \dots, \ell^k - 1\}, \quad 0 \le k \le \left\lfloor \frac{n}{2} \right\rfloor \quad \text{and } \sigma \in D_8.$$

Clearly, such a representation 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, σ), where each $\sigma \neq \mathrm{id}$ comes with some additional constraints on the parameters a, b, c, d.

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

Definition 3 (Classification). *Let $(P_1^*, P_2^*, P_3^*, P_4^*)$ be a symplectic basis for $J[\ell^n]$ and denote by $\mathbf{P}^*$ the column vector $\begin{pmatrix}P_1^* & P_2^* & P_3^* & P_4^*\end{pmatrix}^T$. For a group $G = \langle G_1, G_2\rangle \simeq \mathcal{C}_{\ell^n} \times \mathcal{C}_{\ell^n}$ in $\mathcal{K}_\ell$, we say that G_1, G_2 are the* canonical generators *if one of the following is true for some $a, b, c \in \mathbb{Z}/\ell^n\mathbb{Z}$.*

2.1 $\begin{pmatrix}G_1\\G_2\end{pmatrix} = \begin{pmatrix}1 & 0 & a & b\\0 & 1 & b & c\end{pmatrix}\mathbf{P}^*$.

2.2 $\begin{pmatrix}G_1\\G_2\end{pmatrix} = \begin{pmatrix}1 & b & a & 0\\0 & c & -b & 1\end{pmatrix}\mathbf{P}^*$, *and* $\ell \mid c$.

2.3 $\begin{pmatrix}G_1\\G_2\end{pmatrix} = \begin{pmatrix}a & 0 & 1 & b\\-b & 1 & 0 & c\end{pmatrix}\mathbf{P}^*$, *and* $\ell \mid a, b$.

2.4 $\begin{pmatrix}G_1\\G_2\end{pmatrix} = \begin{pmatrix}a & b & 1 & 0\\b & c & 0 & 1\end{pmatrix}\mathbf{P}^*$, *and* $\ell \mid a, b, c$.

For a group $G = \langle G_1, G_2, G_3\rangle \simeq \mathcal{C}_{\ell^n} \times \mathcal{C}_{\ell^{n-k}} \times \mathcal{C}_{\ell^k}$ with $0 < k < \frac{n}{2}$ in $\mathcal{K}_\ell$, we say that G_1, G_2, G_3 are the canonical generators *if one of the following is true for some $a \in \{0, \ldots, \ell^n - 1\},\ b \in \{0, \ldots, \ell^{n-k} - 1\},\ c \in \{0, \ldots, \ell^{n-2k} - 1\},\ d \in \{0, \ldots, \ell^k - 1\}$.*

3.1 $\begin{pmatrix}G_1\\G_2\\G_3\end{pmatrix} = \begin{pmatrix}1 & d & a & b\\0 & \ell^k & \ell^k(b-cd) & \ell^k c\\0 & 0 & -\ell^{n-k}d & \ell^{n-k}\end{pmatrix}\mathbf{P}^*$.

3.2 $\begin{pmatrix}G_1\\G_2\\G_3\end{pmatrix} = \begin{pmatrix}1 & b & a & d\\0 & \ell^k c & -\ell^k(b-cd) & \ell^k\\0 & \ell^{n-k} & \ell^{n-k}d & 0\end{pmatrix}\mathbf{P}^*$,
and $\ell \mid c$.

3.3 $\begin{pmatrix}G_1\\G_2\\G_3\end{pmatrix} = \begin{pmatrix}a & d & 1 & b\\-\ell^k(b-cd) & \ell^k & 0 & \ell^k c\\\ell^{n-k}d & 0 & 0 & \ell^{n-k}\end{pmatrix}\mathbf{P}^*$,
and $\ell \mid a$.

3.4 $\begin{pmatrix}G_1\\G_2\\G_3\end{pmatrix} = \begin{pmatrix}a & b & 1 & d\\\ell^k(b-cd) & \ell^k c & 0 & \ell^k\\-\ell^{n-k}d & \ell^{n-k} & 0 & 0\end{pmatrix}\mathbf{P}^*$,
and $\ell \mid a, c$.

3.5 $\begin{pmatrix}G_1\\G_2\\G_3\end{pmatrix} = \begin{pmatrix}d & 1 & b & a\\\ell^k & 0 & \ell^k c & \ell^k(b-cd)\\0 & 0 & \ell^{n-k} & -\ell^{n-k}d\end{pmatrix}\mathbf{P}^*$,
and $\ell \mid b, d$.

3.6 $\begin{pmatrix}G_1\\G_2\\G_3\end{pmatrix} = \begin{pmatrix}b & 1 & d & a\\\ell^k c & 0 & \ell^k & -\ell^k(b-cd)\\\ell^{n-k} & 0 & 0 & \ell^{n-k}d\end{pmatrix}\mathbf{P}^*$,
and $\ell \mid b, c, d$.

3.7 $\begin{pmatrix}G_1\\G_2\\G_3\end{pmatrix} = \begin{pmatrix}d & a & b & 1\\\ell^k & -\ell^k(b-cd) & \ell^k c & 0\\0 & \ell^{n-k}d & \ell^{n-k} & 0\end{pmatrix}\mathbf{P}^*$,
and $\ell \mid a, b, d$.

3.8 $\begin{pmatrix}G_1\\G_2\\G_3\end{pmatrix} = \begin{pmatrix}b & a & d & 1\\\ell^k c & \ell^k(b-cd) & \ell^k & 0\\\ell^{n-k} & -\ell^{n-k}d & 0 & 0\end{pmatrix}\mathbf{P}^*$,
and $\ell \mid a, b, c, d$.

For a group $G = \langle G_1, G_2, G_3\rangle \simeq \mathcal{C}_{\ell^n} \times \mathcal{C}_{\ell^k} \times \mathcal{C}_{\ell^k}$ with $k = \frac{n}{2}$ in $\mathcal{K}_\ell$, we say that G_1, G_2, G_3 are the canonical generators *if one of the following is true for some $a \in \{0, \ldots, \ell^n - 1\},\ b, d \in \{0, \ldots, \ell^k - 1\}$.*

4.1 $\begin{pmatrix}G_1\\G_2\\G_3\end{pmatrix} = \begin{pmatrix}1 & d & a & b\\0 & \ell^k & \ell^k b & 0\\0 & 0 & -\ell^k d & \ell^k\end{pmatrix}\mathbf{P}^*$.

4.2 $\begin{pmatrix}G_1\\G_2\\G_3\end{pmatrix} = \begin{pmatrix}a & d & 1 & b\\-\ell^k b & \ell^k & 0 & 0\\\ell^k d & 0 & 0 & \ell^k\end{pmatrix}\mathbf{P}^*$, *and* $\ell \mid a$.

4.3 $\begin{pmatrix}G_1\\G_2\\G_3\end{pmatrix} = \begin{pmatrix}d & 1 & b & a\\\ell^k & 0 & 0 & \ell^k b\\0 & 0 & \ell^k & -\ell^k d\end{pmatrix}\mathbf{P}^*$, *and* $\ell \mid b, d$.

4.4 $\begin{pmatrix}G_1\\G_2\\G_3\end{pmatrix} = \begin{pmatrix}d & a & b & 1\\\ell^k & -\ell^k b & 0 & 0\\0 & \ell^k d & \ell^k & 0\end{pmatrix}\mathbf{P}^*$, *and* $\ell \mid a, b, d$.

Moreover we say that a group $G \in \mathcal{K}_\ell$ is of Type $2.i$, $3.i$ *or* $4.i$ *for $i \in \{1, \ldots, 8\}$ depending on which of the cases above applies.*

Table 1 summarizes the classification of the groups in $\mathcal{K}_\ell$ defined above. 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 [FT19, Prop. 3].

	type	σ	condition on (a,b,c,d)	cardinality
$k=0$	2.1	id	-	ℓ^{3n}
	2.2	(24)	$\ell \mid c$	ℓ^{3n-1}
	2.3	(13)	$\ell \mid a, b$	ℓ^{3n-2}
	2.4	(13)(24)	$\ell \mid a, b, c$	ℓ^{3n-3}
$0<k<\frac{n}{2}$	3.1	id	-	ℓ^{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	-	ℓ^{2n}
	4.2	(13)	$\ell \mid a$	ℓ^{2n-1}
	4.3	(12)(34)	$\ell \mid b, d$	ℓ^{2n-2}
	4.4	(1432)	$\ell \mid a, b, d$	ℓ^{2n-3}

Table 1. Classification of maximal ℓ^n-isotropic subgroups.

2.3 New uniform sampling from the keyspace

In the previous section we 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 way of sampling from the keyspace which chooses elements uniformly at random. 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

- $a_1, a_2, a_3 \overset{\$}{\leftarrow} \{0, \dots, 2^{e_A}-1\}$
- $A_1 = P_{A,1} + [a_1]Q_{A,1} + [a_2]Q_{A,2}$
  $A_2 = P_{A,2} + [a_2]Q_{A,1} + [a_3]Q_{A,2}$
- $\phi_A : J \to J_A = J/\langle A_1, A_2\rangle$

$$\xrightarrow{J_A,\ \phi_A(P_{B,i}),\ \phi_A(Q_{B,i}) \text{ for } i \in \{1,2\}}$$

$$\xleftarrow{J_B,\ \phi_B(P_{A,i}),\ \phi_B(Q_{A,i}) \text{ for } i \in \{1,2\}}$$

- $b_1, b_2, b_3 \overset{\$}{\leftarrow} \{0, \dots, 3^{e_B}-1\}$
- $B_1 = P_{B,1} + [b_1]Q_{B,1} + [b_2]Q_{B,2}$
  $B_2 = P_{B,2} + [b_2]Q_{B,1} + [b_3]Q_{B,2}$
- $\phi_B : J \to J_B = J/\langle B_1, B_2\rangle$

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

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

In the terminology of the previous section this means that $\mathcal{K}_\ell^{\mathrm{res}}$ is the set of all groups of Type 2.1 (cf. Def. 3 and Table 1).

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

Moreover, the restricted keyspace still has the same order of magnitude as the original keyspace. To see this recall the number of maximal ℓ^n-isotropic subgroups from [FT19, 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 ℓ^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 ℓ^n.

Remark 2. 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 [FT19, Thm. 2] and [FT19, 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.

3 Adaptive attack on G2SIDH

The attack as presented in this section is able to recover Alice's secret kernel when she uses 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 θ_i 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 [GPST16], 2-SIDH [DGL$^+$20], and Jao–Urbanik's variant [BKM$^+$20]. 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 §2.1. 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 $\langle A_1, A_2, A_3\rangle$, 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 $e_{2^n}(P_i, Q_j) = \zeta^{\delta_{ij}}$, where ζ 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 2^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 an honest Alice uses a static key which a malicious Bob is trying to learn through repeatedly providing malformed torsion point information 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 ℓ and therefore ℓ^{e_ℓ}-torsions of J due to Proposition 2. The resulting attack on different ℓ may not return a bit of information with every single query, but may require a small number of additional queries to determine a bit of information. 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, as is done in [GPST16].

Our oracle, which replaces Alice in an honest execution of the protocol,

$$O\left(J, J', (R_1', R_2', R_3', R_4')\right)$$

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 as linear combinations of R_1, R_2, S_1, S_2 and laid out in a 4×4 matrix. That is, for any points R'_1, R'_2, S'_1, S'_2 that the adversary sends to the oracle, we can write

$$\begin{pmatrix} R'_1 \\ R'_2 \\ S'_1 \\ S'_2 \end{pmatrix} = \begin{pmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \\ d_1 & d_2 & d_3 & d_4 \end{pmatrix} \begin{pmatrix} R_1 \\ R_2 \\ S_1 \\ S_2 \end{pmatrix},$$

and we will represent the queries R'_1, R'_2, S'_1, S'_2 by the 4×4 matrix.

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 $t : (R'_1, R'_2, S'_1, S'_2) \leftarrow (R_1, R_2, S_1, S_2)$ with this property is called a *symplectic transformation.* The matrices corresponding to symplectic transformations are called *symplectic matrices.* We are going to write M_t for the matrix corresponding to the transformation t.

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 this is not true for general isomorphisms of $J[2^n]$.

One can easily verify that the following matrices are symplectic. We will use different combinations of these to construct the transformations for the oracle queries.

$$M_{t_0} = \begin{pmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 1&0&1&0 \\ 0&0&0&1 \end{pmatrix}, \ M_{t_1} = \begin{pmatrix} 1&0&1&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \end{pmatrix}, \ M_{t_2} = \begin{pmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&1&0&1 \end{pmatrix},$$

$$M_{t_3} = \begin{pmatrix} 1&0&0&0 \\ 0&1&0&1 \\ 0&0&1&0 \\ 0&0&0&1 \end{pmatrix}, \ M_{t_4} = \begin{pmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&1&1&0 \\ 1&0&0&1 \end{pmatrix}, \ M_{t_5} = \begin{pmatrix} 1&0&0&1 \\ 0&1&1&0 \\ 0&0&1&0 \\ 0&0&0&1 \end{pmatrix}.$$

Proposition 3. *The following matrices are symplectic for any values $x, x_0, x_1, x_2, x_3, x_4, x_5$ and ivertible elements $\theta_1, \theta_2 \in \mathbb{Z}/2^n\mathbb{Z}$.*

$$M_1 = \begin{pmatrix} \theta_1 & \theta_2 x & 0 & 0 \\ 0 & \theta_2 & 0 & 0 \\ 0 & 0 & \theta_1^{-1} & 0 \\ 0 & 0 & -\theta_1^{-1}x & \theta_2^{-1} \end{pmatrix} \quad \textit{and} \quad M_2 = \begin{pmatrix} 1&0&x_1&x_5 \\ 0&1&x_5&x_3 \\ 0&0&1&0 \\ 0&0&0&1 \end{pmatrix},$$

$$M_3 = \begin{pmatrix} \theta_1(1+x_0x_1 - x_4x_5(1+x_0x_1)) & \theta_2x_2x_5 & \theta_1^{-1}x_1(1+x_4x_5) & \theta_2^{-1}x_5 \\ \theta_1x_0x_5 & \theta_2(1+x_2x_3+x_4x_5(1+x_2x_3)) & \theta_1^{-1}x_5 & \theta_2^{-1}x_3(1+x_4x_5) \\ \theta_1x_0 & \theta_2x_4(1+x_2x_3) & \theta_1^{-1} & \theta_2^{-1}x_3x_4 \\ \theta_1x_4(1+x_0x_1) & \theta_2x_2 & \theta_1^{-1}x_1x_4 & \theta_2^{-1} \end{pmatrix}.$$

Proof. It is easy to check that M_1 is symplectic since the scalars satisfy $\theta_1\theta_1^{-1} = \theta_2\theta_2^{-1} = 1$. The matrix M_2 can be easily written in terms of the transformations t_i, namely $M_2 = M_{t_1}^{x_1} \cdot M_{t_3}^{x_3} \cdot M_{t_5}^{x_5}$. Finally, M_3 can be written as

$$M_3 = M_1 \cdot M_{t_0}^{x_0} \cdot M_{t_1}^{x_1} \cdot M_{t_2}^{x_2} \cdot M_{t_3}^{x_3} \cdot M_{t_4}^{x_4} \cdot M_{t_5}^{x_5}.$$

□

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. This means $A_1 = R_1 + [d]R_2 + [a]S_1 + [b]S_2,\ A_2 = 2^k(R_2 + [b-cd]S_1 + [c]S_2)\ A_3 = 2^{n-k}([-d]S_1 + S_2)$. As in §2.2, we let A be the associated matrix, i.e. here

$$A = \begin{pmatrix} 1 & d & a & b \\ 0 & 2^k & 2^k(b-cd) & 2^k c \\ 0 & 0 & -2^{n-k}d & 2^{n-k} \end{pmatrix}.$$

Applying a basis transformation t corresponds to computing $A' = A \cdot M_t$. As an example, consider the second basis transformation from the above proposition.

$$A' = A \cdot M_2 = \begin{pmatrix} 1 & d & a + x_1 + dx_5 & b + x_5 + dx_3 \\ 0 & 2^k & 2^k(b-cd) + 2^k x_5 & 2^k c + 2^k x_3 \\ 0 & 0 & -2^{n-k}d & 2^{n-k} \end{pmatrix} = A + \begin{pmatrix} 0 & 0 & x_1 + dx_5 & x_5 + dx_3 \\ 0 & 0 & 2^k x_5 & 2^k x_3 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

This means that the matrices A and A' correspond to the same group G_A if and only if $\langle [x_1 + dx_5]S_1 + [x_5 + dx_3]S_2, [2^k]([x_5]S_1 + [x_3]S_2)\rangle \subset G_A$.

3.3 Case distinction

Recall that in [FT19], Alice's secret can be described by $(\alpha_{1,1}, \dots, \alpha_{3,4})$. 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. Def. 3, Table 1). In total, when $k = 0$ there are 4 types of maximal 2^n-isotropic groups, 8 different 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 §2.2. 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 details for one of the paths in the decision tree in Appendix C.

Assuming that the key $(\alpha_{1,1}, \dots, \alpha_{3,4})$ 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 §2.3 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 value 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 $\langle A_1\rangle$. Therefore 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 thus assume knowledge of $b \pmod{2^k}$ and d for any rank-3 kernel subgroups.

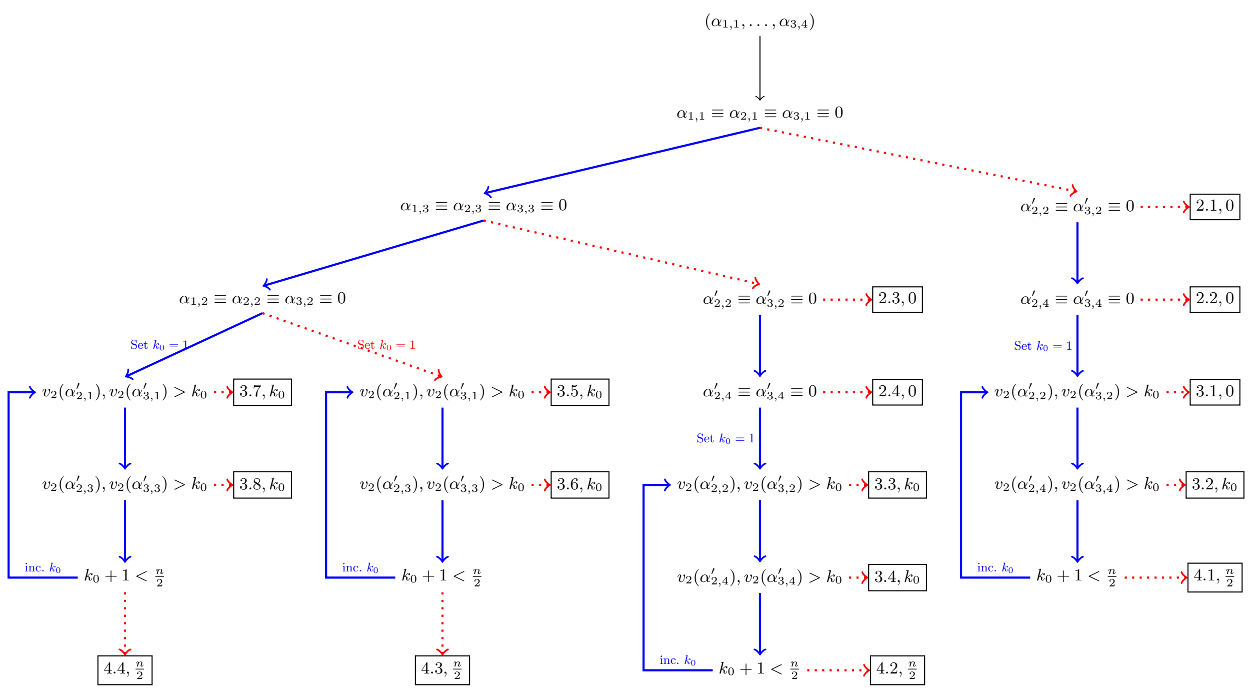
$(\alpha_{1,1},\dots,\alpha_{3,4})$
$\alpha_{1,1} \equiv \alpha_{2,1} \equiv \alpha_{3,1} \equiv 0$
$\alpha_{1,3} \equiv \alpha_{2,3} \equiv \alpha_{3,3} \equiv 0$
$\alpha'_{2,2} \equiv \alpha'_{3,2} \equiv 0$
$\boxed{2.1, 0}$
$\alpha_{1,2} \equiv \alpha_{2,2} \equiv \alpha_{3,2} \equiv 0$
$\alpha'_{2,2} \equiv \alpha'_{3,2} \equiv 0$
$\boxed{2.3, 0}$
$\alpha'_{2,4} \equiv \alpha'_{3,4} \equiv 0$
$\boxed{2.2, 0}$
Set $k_0 = 1$
Set $k_0 = 1$
Set $k_0 = 1$
$v_2(\alpha'_{2,1}), v_2(\alpha'_{3,1}) > k_0$
$\boxed{3.7, k_0}$
$v_2(\alpha'_{2,1}), v_2(\alpha'_{3,1}) > k_0$
$\boxed{3.5, k_0}$
$\alpha'_{2,4} \equiv \alpha'_{3,4} \equiv 0$
$\boxed{2.4, 0}$
$v_2(\alpha'_{2,2}), v_2(\alpha'_{3,2}) > k_0$
$\boxed{3.1, 0}$
Set $k_0 = 1$
$v_2(\alpha'_{2,3}), v_2(\alpha'_{3,3}) > k_0$
$\boxed{3.8, k_0}$
$v_2(\alpha'_{2,3}), v_2(\alpha'_{3,3}) > k_0$
$\boxed{3.6, k_0}$
$v_2(\alpha'_{2,2}), v_2(\alpha'_{3,2}) > k_0$
$\boxed{3.3, k_0}$
$v_2(\alpha'_{2,4}), v_2(\alpha'_{3,4}) > k_0$
$\boxed{3.2, k_0}$
inc. k_0
$k_0 + 1 < \frac{n}{2}$
inc. k_0
$k_0 + 1 < \frac{n}{2}$
$v_2(\alpha'_{2,4}), v_2(\alpha'_{3,4}) > k_0$
$\boxed{3.4, k_0}$
inc. k_0
$k_0 + 1 < \frac{n}{2}$
$\boxed{4.1, \frac{n}{2}}$
inc. k_0
$k_0 + 1 < \frac{n}{2}$
$\boxed{4.2, \frac{n}{2}}$
$\boxed{4.4, \frac{n}{2}}$
$\boxed{4.3, \frac{n}{2}}$

Fig. 2. Strategy for type distinction of normalised kernel generators as in Table 1. We begin with Alice's scalars $(\alpha_{1,1},\dots,\alpha_{3,4})$. Each node below represents one or multiple malformed queries which determine whether the displayed condition holds. All equivalence conditions are viewed modulo 2 here. For example, the first query node corresponds to checking whether $\alpha_{1,1} \equiv \alpha_{2,1} \equiv \alpha_{3,1} \equiv 0 \pmod 2$ which can be done with the transformation ${t_1}^{2^{n-1}}$.
At each node, a true response indicates that the next query can be found along the blue and solid arrow, while the red and dotted path is taken when the condition is not fulfilled. Note that when an odd scalar is found, the subsequent conditions use further normalised scalars denoted by $\alpha'_{i,j}$. Leaves show which type of normalised generators define the secret subgroup Alice uses (as classified in Table 1), followed by k which indicates the order of the generators of the subgroup.
For distinguishing types of rank-3 subgroups, it is necessary to use iterative queries to find the correct type and determine the value of k. At each step, we test whether $k = k_0$ for increasing values of $0 < k_0 < \frac{n}{2} - 1$ by checking if certain scalars are divisible by 2^{k_0+1}. We use that for any integer x, $v_2(x)$ denotes the largest integer such that $2^{v_2(x)}$ divides x. If a scalar is found to not satisfy the divisibility condition, we can again normalise at this position and deduce the type of the subgroup along with k indicating the order of its generators.

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. We illustrate the attack for Type 2.1, where

$$\begin{pmatrix} A_1 \\ A_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & a & b \\ 0 & 1 & b & c \end{pmatrix} \cdot \begin{pmatrix} R_1 \\ R_2 \\ S_1 \\ S_2 \end{pmatrix}.$$

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.

transformation	same j-invariant 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. The equivalences in the second column are all modulo 2.

$$\begin{tikzcd}
& & & & & & & {t_4}^{2^{n-1}} \arrow[dllll] \arrow[drrrr, dotted] & & & & & & & \\
& & & {t_0}^{2^{n-1}} \arrow[dll] \arrow[drr, dotted] & & & & & & & & {t_2}^{2^{n-1}}{t_1}^{2^{n-1}} \arrow[dll] \arrow[drr, dotted] & & & \\
& {t_2}^{2^{n-1}} \arrow[dl] \arrow[dr, dotted] & & & & {t_2}^{2^{n-1}} \arrow[dl] \arrow[dr, dotted] & & & & {t_0}^{2^{n-1}}{t_3}^{2^{n-1}} \arrow[dl] \arrow[dr, dotted] & & & & {t_0}^{2^{n-1}}{t_3}^{2^{n-1}} \arrow[dl] \arrow[dr, dotted] & \\
0 & & 1 & & 4 & & 7 & & 2 & & 6 & & 3 & & 5
\end{tikzcd}$$

Fig. 3. Optimal strategy for recovering parity bits. The top node represents the first malformed query which will use the t_4 transformation to determine whether $b \equiv ac \pmod 2$, as shown in Table 2. A true response indicates that the next query can be found along the blue and solid arrow, while the red and dotted path is taken when the condition is false. Leaves are decimal representations of the parity bits a_0, b_0, c_0, i.e. 6 corresponds to $[a_0, b_0, c_0] = [1, 1, 0]$.

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 obtained via the matrix

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 2^{n-1} & 1 & 0 \\ 2^{n-1} & 0 & 0 & 1 \end{pmatrix}$$

and leads to Alice using

$$A' = \begin{pmatrix} 1 & 0 & a & b \\ 0 & 1 & b & c \end{pmatrix} \cdot M = \begin{pmatrix} 1 + 2^{n-1}b & 2^{n-1}a & a & b \\ 2^{n-1}c & 1 + 2^{n-1}b & b & c \end{pmatrix} \sim A + \begin{pmatrix} 0 & 0 & 0 & 2^{n-1}(b^2 + ac) \\ 0 & 0 & 2^{n-1}(b^2 + ac) & 0. \end{pmatrix}$$

during her internal computations. Note that in the last step, Gaussian elimination is used to normalize A'. 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 $\{[0, 1, 0], [0, 1, 1], [1, 0, 1], [1, 1, 0]\}$.

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).

Now assume that $i < n-3$ and set $e_i := n-i-1$. By the lemma below, the element $T_i = 1 - 2^{e_i}$ is thus a quadratic residue modulo 2^n.

Lemma 2 ([GPST16, Lem. 4]). *Let $n \geq 5$ and $i \in \{1, \dots, n-4\}$. Then $T_i := 1 - 2^{n-i-1}$ is a quadratic residue modulo 2^n.*

In the following, $\theta_i \in \{0, \dots, 2^n - 1\}$ denotes one of the square roots of T_i, i.e. $\theta_i^2 \equiv T_i \pmod{2^n}$. Note that θ_i is necessarily odd, hence there exists an inverse θ_i^{-1} modulo 2^n. Intuitively, θ_i 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 obtained from

$$\theta_i^{-1} \begin{pmatrix} T_i & 0 & -2^{e_i} K_i^a & 0 \\ 0 & 1 & 0 & 2^{e_i} K_i^c \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & T_i \end{pmatrix}.$$

Upon which, the subgroup computation[7] will entail

$$\begin{aligned} A' &= \theta^{-1} \begin{pmatrix} T_i & 0 & -2^{e_i} K_i^a + a & T_i b \\ 0 & 1 & b & 2^{e_i} K_i^c + T_i c \end{pmatrix} \\ &\sim \begin{pmatrix} 1 & 0 & a + T_i^{-1} 2^{e_i}(K_i^a - a) & b \\ 0 & 1 & b & c + 2^{e_i}(K_i^c - c) \end{pmatrix} = A + \begin{pmatrix} 0 & 0 & T_i^{-1} 2^{n-1} a_i & 0 \\ 0 & 0 & 0 & 2^{n-1} c_i \end{pmatrix}. \end{aligned}$$

Hence A' defines the same group as A, 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 corresponding to the transformation matrix

$$\theta_i^{-1} \begin{pmatrix} T_i & 0 & -2^{e_i}(K_i^a + 2^i) & 0 \\ 0 & 1 & 0 & 2^{e_i} K_i^c \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & T_i \end{pmatrix}.$$

In this case

$$A' \sim A + \begin{pmatrix} 0 & 0 & T_i^{-1} 2^{n-1}(a_i + 1) & 0 \\ 0 & 0 & 0 & 2^{n-1} c_i \end{pmatrix}.$$

The groups associated to A and A' coincide 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 obtained from

$$\theta_i^{-1} \begin{pmatrix} T_i & 0 & -2^{e_i} K_i^a & 0 \\ 0 & 1 & 0 & 2^{e_i} K_{i+1}^c \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & T_i \end{pmatrix}.$$

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

[7] To verify the computation below note that $T_i^{-1} \equiv 1 + 2^{e_i} T_i^{-1} \pmod{2^n}$.

After these series of queries, we have recovered the bits a_i and c_i, hence we know K^a_{i+1} and K^c_{i+1}. It remains to recover the bit b_i. This is done by querying the oracle on the points corresponding to the matrix

$$\theta_i^{-1}\begin{pmatrix} 1 & 0 & 2^{e_i}K^a_{i+1} & 2^{e_i}K^b_i \\ 0 & 1 & 2^{e_i}K^b_i & 2^{e_i}K^c_{i+1} \\ 0 & 0 & T_i & 0 \\ 0 & 0 & 0 & T_i \end{pmatrix}.$$

Here,

$$A' \sim A + \begin{pmatrix} 0 & 0 & 2^{e_i}(K^a_{i+1}-a) & 2^{e_i}(K^b_i-b) \\ 0 & 0 & 2^{e_i}(K^b_i-b) & 2^{e_i}(K^c_{i+1}-c) \end{pmatrix} = A + \begin{pmatrix} 0 & 0 & 0 & 2^{n-1}b_i \\ 0 & 0 & 2^{n-1}b_i & 0 \end{pmatrix}.$$

The oracle returns 1 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 [GPST16].

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 rank 3, i.e. $k > 0$. Let $1 \le k \le \lfloor \frac{n}{2} \rfloor$ 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. We present the attack for a kernel of Type 3.1, hence the generators are of the form

$$\begin{pmatrix} A_1 \\ A_2 \\ A_3 \end{pmatrix} = \begin{pmatrix} 1 & d & a & b \\ 0 & 2^k & 2^k(b-cd) & c \\ 0 & 0 & -2^{n-k}d & 2^{n-k} \end{pmatrix} \cdot \begin{pmatrix} R_1 \\ R_2 \\ S_1 \\ S_2 \end{pmatrix}$$

for some $(a,b,c,d) \in \{0,\ldots,2^n-1\} \times \{0,\ldots,2^{n-k}-1\} \times \{0,\ldots,2^{n-2k}-1\} \times \{0,\ldots,2^k-1\}$, 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 again fix

$$e_i = n-i-1, \quad T_i = 1 - 2^{e_i} \in \mathbb{Z}/\ell^n\mathbb{Z}, \quad \theta_i^2 = T_i \in \mathbb{Z}/\ell^n\mathbb{Z}$$

for $1 \le i \le n-4$, where θ_i is any one of the two square roots. Recall that θ_i exists since $T_i \equiv 1 \pmod 8$.

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

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 2^{n-1} & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

These allow us to recover the bit a_0 since

$$A' = \begin{pmatrix} 1+2^{n-1}a & d & a & b \\ 0 & 2^k & 2^k(b-cd) & 2^k c \\ 0 & 0 & -2^{n-k}d & 2^{n-k} \end{pmatrix} \sim A + \begin{pmatrix} 0 & 0 & 2^{n-1}a^2 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

This means that A and A' correspond to the same group if and only if $a_0 = 0$, hence 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

$$\alpha = -2^{e_i}(dK_k^b + K_i^a), \quad \delta = -2^{e_i}d.$$

and send the malformed points obtained from the transformation

$$\theta_i^{-1} \begin{pmatrix} T_i & \delta & \alpha & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\delta & T_i \end{pmatrix}.$$

This transformation applied to A yields

$$A' = \theta^{-1} \begin{pmatrix} T_i & \delta+d & \alpha+a-\delta b & T_i b \\ 0 & 2^k & 2^k(b-cd-\delta c) & 2^k T_i c \\ 0 & 0 & 2^{n-k}(-d-\delta) & 2^{n-k}T_i \end{pmatrix} \sim \begin{pmatrix} 1 & d & a+2^{e_i}(a-K_i^a) & b \\ 0 & 2^k & 2^k(b-cd) & 2^k c \\ 0 & 0 & -2^{n-k}d & 2^{n-k} \end{pmatrix}$$

$$\sim A + \begin{pmatrix} 0 & 0 & 2^{n-1}a_i & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

To verify the above simplifications, note that $T_i^{-1} = 1 + 2^{e_i}$ (when $k \geq 1$ and $i \leq k-1$ as is the case here). 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 $\boldsymbol{a}$ (mod $\mathbf{2^{n-k-1}}$) and $\boldsymbol{c}$. 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 obtained from the transformation

$$\theta_i^{-1} \begin{pmatrix} T_i & \delta & \alpha & \beta \\ 0 & 1 & \beta & \gamma \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\delta & T_i \end{pmatrix}$$

where

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

Then we obtain

$$A' \sim A + \begin{pmatrix} 0 & 0 & 2^{n-1}a_i - 2^{n-k-1}d^2 c_{i-k+1} & 2^{n-k-1}dc_{i-k+1} \\ 0 & 0 & 2^{n-1}dc_{i-k+1} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

This means that A and A' define the same group if both $[2^{n-1}a_i]S_1 + [2^{n-k-1}dc_{i-k+1}]([-d]S_1 + S_2)$ and $[2^{n-1}dc_{i-k+1}]S_1$ are in G_A. Recall that we assume d odd and note that $[2^{n-k}]([-d]S_1 + S_2) \in G_A$. Hence 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 $\alpha = -2^{e_i}(K_i^a + 2^i)T_i^{-1}$ in the query above. Similarly, we test for $a_i = 0$ and $c_{i-k+1} = 1$ by setting $\beta = -2^{e_i-1}(K_{i-k+1}^c - 2^{i-k+1})dT_i^{-1}$ and $\gamma = 2^{e_i}(K_{i-k+1}^c + 2^{i-k+1})$.

Recovering *b*. Recall that K^a_{n-k-1}, K^b_k, 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 $k \le i < n - \max\{k, 3\}$ which can be adapted via some shift in indices to accommodate even d. Let

$$\alpha = 2^{e_i}(K^a_{n-k-1} + K^b_i d - cd^2), \quad \beta = 2^{e_i}cd, \quad \gamma = 2^{e_i}c, \quad \delta = 2^{e_i}d.$$

We then send malformed points obtained via

$$\theta_i^{-1}\begin{pmatrix}1 & \delta & \alpha & \beta\\ 0 & T_i & \beta & -\gamma\\ 0 & 0 & T_i & 0\\ 0 & 0 & -\delta & 1\end{pmatrix}$$

to the oracle resulting in

$$A' \sim \begin{pmatrix}1 & \delta + T_i d & \alpha + \beta d - \delta b + T_i\alpha & \beta - \gamma d + b\\ 0 & 2^k & 2^k(T_i^{-1}\beta + b - dc - T_i^{-1}\delta c) & 2^k T_i^{-1}(c-\gamma)\\ 0 & 0 & 2^{n-k}(-T_i d - \delta) & 2^{n-k}\end{pmatrix} = A + \begin{pmatrix}0 & 0 & 2^{e_i}d(K^b_i - b) & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{pmatrix}$$

so that the oracle 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

$$\alpha = 2^{e_i}(K^a_i + bd - cd^2), \quad \beta = 2^{e_i}cd, \quad \gamma = 2^{e_i}c, \quad \delta = 2^{e_i}d$$

and query the oracle with the symplectic transformation

$$\theta_i^{-1}\begin{pmatrix}1 & \delta & \alpha & \beta\\ 0 & T_i & \beta & -\gamma\\ 0 & 0 & T_i & 0\\ 0 & 0 & -\delta & 1\end{pmatrix}.$$

We obtain

$$A' \sim A + \begin{pmatrix}0 & 0 & 2^{e_i}(K^a_i - a) & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{pmatrix}.$$

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 $T_i = 1 - 2^{e_i}$ is not defined when $i \ge n-3$, 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 [FT19]).

As shown in Algorithm 1 (cf. Appendix B), we are able to obtain a symplectic basis from an arbitrary basis using a 4×4 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

[ACC$^+$20] Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Aaron Hutchinson, Amir Jalali, David Jao, Koray Karabina, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Geovandro Pereira, Joost Renes, Vladimir Soukharev, and David Urbanik, *Supersingular isogeny key encapsulation (SIKE)*, Updated specifications for NIST Post-Quantum Standardization project (2020).

[BKM$^+$20] Andrea Basso, Péter Kutas, Simon-Philipp Merz, Christophe Petit, and Charlotte Weitkämper, *On adaptive attacks against Jao-Urbanik's isogeny-based protocol*, Progress in Cryptology - AFRICACRYPT 2020 - 12th International Conference on Cryptology in Africa, Cairo, Egypt, July 20-22, 2020, Proceedings (Abderrahmane Nitaj and Amr M. Youssef, eds.), Lecture Notes in Computer Science, vol. 12174, Springer, 2020, pp. 195–213.

[CF96] J. W. S. Cassels and E. V. Flynn, *Prolegomena to a middlebrow arithmetic of curves of genus 2*, London Mathematical Society Lecture Note Series, Cambridge University Press, 1996.

[DGL$^+$20] Samuel Dobson, Steven D. Galbraith, Jason T. LeGrow, Yan Bo Ti, and Lukas Zobernig, *An adaptive attack on 2-SIDH*, Int. J. Comput. Math. Comput. Syst. Theory **5** (2020), no. 4, 282–299.

[FO13] Eiichiro Fujisaki and Tatsuaki Okamoto, *Secure integration of asymmetric and symmetric encryption schemes*, J. Cryptology **26** (2013), no. 1, 80–101.

[FT19] E. Victor Flynn and Yan Bo Ti, *Genus two isogeny cryptography*, International Conference on Post-Quantum Cryptography, Springer, 2019, pp. 286–306.

[GPST16] Steven D. Galbraith, Christophe Petit, Barak Shani, and Yan Bo Ti, *On the security of supersingular isogeny cryptosystems*, Advances in Cryptology - ASIACRYPT 2016, 2016, pp. 63–91.

[JF11] David Jao and Luca De Feo, *Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies*, Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29 - December 2, 2011. Proceedings (Bo-Yin Yang, ed.), Lecture Notes in Computer Science, vol. 7071, Springer, 2011, pp. 19–34.

[Mil86] J. S. Milne, *Abelian varieties*, Arithmetic Geometry (Gary Cornell and Joseph H. Silverman, eds.), Springer New York, New York, NY, 1986, pp. 103–150.

[NIS17] NIST (National Institute of Standards and Technology), *NIST post-quantum cryptography project*, `http://csrc.nist.gov/groups/ST/post-quantum-crypto/`, 2017.

[Sil09] Joseph H. Silverman, *The arithmetic of elliptic curves*, vol. 106, Springer Science & Business Media, 2009.

[Tat66] John Tate, *Endomorphisms of abelian varieties over finite fields*, Invent. Math. **2** (1966), 134–144.

A Revisiting SIDH

In the main body of 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 ℓ_A and ℓ_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. We follow the suggestion from [JF11] to use $\ell_A = 2$ and $\ell_B = 3$. Let P_A, Q_A be generators of the 2^{e_A}-torsion and let P_B, Q_B be generators of the 3^{e_B}-torsion of E. Then the protocol is as follows.

1. Alice chooses a random cyclic subgroup of $E[2^{e_A}]$ of order 2^{e_A}. As P_A, Q_A form a basis of the 2^{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 $E[3^{e_B}]$ of order 3^{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 $E/\langle A, B\rangle = E_B/\langle \phi_B(A)\rangle$).

A.2 Keyspace for SIDH

A secret key of Alice is a cyclic subgroup of E order 2^{e_A}, and similarly Bob's secret key is a cyclic subgroup of order 3^{e_A}. In analogy to the definition in §2, we denote

$$\mathcal{K}_\ell = \{G \subset E \mid G \text{ cyclic and } \#G = \ell^n\},$$

for $\ell \in \{2, 3\}$ and $n = e_A$ or e_B, respectively.

It is easy to see that a group $G \in \mathcal{K}_\ell$ is maximal ℓ^n-isotropic with respect to the Weil-pairing on E.[8] Moreover $G \not\subset E[m]$ for any $m < \ell^n$, since a generator of G has order ℓ^n. In particular, we may use the equivalent definition

$$\mathcal{K}_\ell = \{G \subset E \mid G \text{ maximal } \ell^n\text{-isotropic and } G \not\subset E[m] \text{ for any } m < \ell^n\},$$

which resembles the definition in §2 more closely.

An important ingredient in the classification of secret keys for G2SIDH is the use of symplectic bases for the torsion groups $J[\ell^n]$. In the elliptic curve setting one automatically works with symplectic bases.

Lemma 3. *Let $\ell \neq p$ be a prime and $n > 0$. Then every basis (P, Q) for $E[\ell^n]$ is symplectic with respect to the Weil pairing.*

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

$$\mathcal{K}_\ell = \{\langle P + [a]Q\rangle \mid a \in \mathbb{Z}/\ell^n\mathbb{Z}\} \cup \{\langle [\ell a]P + Q\rangle \mid a \in \mathbb{Z}/\ell^n\mathbb{Z}\}.$$

In the terminology of §2.2, 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. Definition 3, Table 1). To make the analogy more explicit, we introduce the following terminolgy.

Definition 4. *Let (P, Q) be a basis for $E[\ell^n]$. For a group $G = \langle G_1\rangle \simeq \mathcal{C}_{\ell^n}$ in $\mathcal{K}_\ell$, we say that G_1 is the* canonical generator *of G if one of the following is true for some $a \in \mathbb{Z}/\ell^n\mathbb{Z}$.*

[8] Isotropy follows from the fact that G is cyclic. To see that G is maximal with this property, consider some $R \in G$ with $G = \langle R\rangle$ and note that for any $R' \in E \setminus G$, the Weil pairing $e_{\ell^n}(R, R')$ is non-trivial.

1.1 $G_1 = \begin{pmatrix}1 & a\end{pmatrix} \cdot \begin{pmatrix}P \\ Q\end{pmatrix}$. *1.2* $G_1 = \begin{pmatrix}a & 1\end{pmatrix} \cdot \begin{pmatrix}P \\ Q\end{pmatrix}$ *and* $\ell \mid a$.

Moreover we say that a group $G \in \mathcal{K}_\ell$ *is of* Type 1.*i* *for* $i \in \{1, 2\}$ *depending on which of the cases above applies.*

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 keyspace has cardinality ℓ^n [ACC+20, Sec. 1.3.9]. This is very similar to the restriction suggested in §2.3. A sketch of the SIDH protocol using the restricted keyspace is provided in Figure 4.

Setup

- prime $p = 2^{e_A} \cdot 3^{e_B} \cdot f - 1$.
- supersingular elliptic curve E.
- torsion bases (P_A, Q_A) for $E[2^{e_A}]$ and (P_B, Q_B) for $E[3^{e_B}]$.

Key Generation

- $a \overset{\$}{\leftarrow} \{0, \dots, 2^{e_A} - 1\}$.
- $A = P_A + [a]Q_A$.
- $\phi_A : E \to E_A = E/\langle A \rangle$

$$\xrightarrow{E_A, \phi_A(P_B), \phi_A(Q_B)}$$

$$\xleftarrow{E_B, \phi_B(P_A), \phi_B(Q_A)}$$

- $b \overset{\$}{\leftarrow} \{0, \dots, 3^{e_B} - 1\}$.
- $B = P_B + [b]Q_B$.
- $\phi_B : E \to E_B = E/\langle B \rangle$

Shared Key

$$E_B/\langle \phi_B(P_A) + [a]\phi_B(Q_A) \rangle = \quad E/\langle A, B \rangle \quad = E_A/\langle \phi_A(P_B) + [b]\phi_A(Q_B) \rangle$$

Fig. 4. SIDH protocol with restricted keyspace.

A.3 Adaptive attack

In [GPST16], the authors present an adaptive attack on SIDH. Here, we will briefly present their strategy. In order to illustrate the connection to our adaptive attack on G2SIDH, we use a different terminology for this presentation.

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 [GPST16] albeit not explicitly phrased in this way.

Assume that Alice uses a fixed secret key x_A, y_A defining 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}.[9] Then they send different queries to the oracle with fixed

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

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

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

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

$$M = \begin{pmatrix} m_{1,1} & m_{1,2} \\ m_{2,1} & m_{2,2} \end{pmatrix}.$$

It is easy to see that this matrix is symplectic if and only if $\det(M) = 1$. [10]

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. These malformed points are obtained from the transformation

$$M = \begin{pmatrix} 1 & 2^{n-1} \\ 0 & 1 \end{pmatrix}.$$

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 canonical generator by $A_1 = R + [a]S$. In order to be consistent with the notation from §3, we set $A = \begin{pmatrix} 1 & a \end{pmatrix}$, hence $A_1 = A \cdot \begin{pmatrix} R & S \end{pmatrix}^t$.

First bit recovery. In order to find the first bit of a, the attacker queries the oracle on the malformed points $(R', S')^t = M_0 \cdot (R, S)^t$, where

$$M_0 = \begin{pmatrix} 1 & 0 \\ 2^{n-1} & 1 \end{pmatrix}.$$

Note that $A \cdot (R', S')^t = A' \cdot (R, S)^t$, where

$$A' = A \cdot M_0 = \begin{pmatrix} 1 & a \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 2^{n-1} & 1 \end{pmatrix} = \begin{pmatrix} 1 + 2^{n-1}a & a \end{pmatrix}.$$

It follows that

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

Iterative step. Assume the attacker has recovered the first i bits of a. Then we know K_i^a, where $a = K_i^a + \sum_{j=i}^{n-1} a_j 2^j$. If $i < n - 3$, there exists θ satisfying $\theta^2 \equiv 1 + 2^{n-i-1} \pmod{2^n}$, [GPST16, Lemma 3.3]. Necessarily θ is an odd integer, hence invertible modulo 2^n. Consider the transformation $(R', S')^t = M_i \cdot (R, S)^t$, where

$$M_i = \theta^{-1} \cdot \begin{pmatrix} 1 & -2^{n-i-1}K_i^a \\ 0 & 1 + 2^{n-i-1} \end{pmatrix}.$$

Clearly M_i is symplectic, hence the tuple $(E_B, E_{AB}, (R', S'))$ defines a valid query. Here, we obtain

$$\begin{aligned} A' = A \cdot M_i &= \begin{pmatrix} 1 & a \end{pmatrix} \cdot \theta^{-1} \cdot \begin{pmatrix} 1 & -2^{n-i-1}K_i^a \\ 0 & 1 + 2^{n-i-1} \end{pmatrix} = \theta^{-1} \begin{pmatrix} 1 & a + 2^{n-i-1}a - 2^{n-i-1}K_i^a \end{pmatrix} \\ &= \theta^{-1} \begin{pmatrix} 1 & a + 2^{n-1}a_i \end{pmatrix}. \end{aligned}$$

[10] Note that this criterion is unique to the case of 2×2 matrices. For $2n \times 2n$ matrices with $n > 1$ the condition $\det(M) = 1$ is necessary but not sufficient.

This shows that $\langle A \cdot (R,S)^t \rangle = \langle A' \cdot (R,S)^t \rangle$ if and only if a_i is even. In conclusion

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

This shows that the attacker can iteratively recover the first $n-2$ bits of a. For the remaining two bits, the authors suggest to use a brute force method.

Remark 3. It is a priori not obvious how to find symplectic transformations which can be used to recover the parity of some bit a_i of the secret scalar. In [GPST16], the authors use the following strategy.

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

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

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 bits.

The first step could be translated directly to the G2SIDH-setting. This approach is developed in AppendixD. 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 λ 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. §2.1). 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
1  Set $P_1 \leftarrow R_1$;
   // $Q_1$ such that $e(P_1, Q_1) = \zeta_m$
2  if $\mathrm{ord}(e(P_1, R_2)) = m$ then
3      Set $Q_1 \leftarrow R_2$;
4  else if $\mathrm{ord}(e(P_1, R_3)) = m$ then
5      Set $Q_1 \leftarrow R_3$;
6      Set $R_3 \leftarrow R_2$;
7  else
8      Set $Q_1 \leftarrow R_4$;
9      Set $R_4 \leftarrow R_2$;
10 Set $\zeta \leftarrow e(P_1, Q_1)$;
   // $P_2, Q_2$ should be ``orthogonal'' to $P_1, Q_1$
11 Set $\lambda_1 \leftarrow \log(\zeta, e(Q_1, R_3))$, $\lambda_2 \leftarrow \log(\zeta, e(P_1, R_3))$;
12 Set $P_2 \leftarrow R_3 + [\lambda_1]P_1 - [\lambda_2]Q_1$;
13 Set $\mu_1 \leftarrow \log(\zeta, e(Q_1, R_4))$, $\mu_2 \leftarrow \log(\zeta, e(P_1, R_4))$, $\mu_3 \leftarrow \log(\zeta, e(P_2, R_4))$;
14 Set $Q_2 \leftarrow [\mu_3^{-1}](R_4 + [\mu_1]P_1 - [\mu_2]Q_1)$;
15 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. Def. 3, Table 1). We will run through the explicit 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 tree of Figure 2. The queries required to check the conditions along other paths are very similar and hence omitted.

Suppose Alice's secret is of the form $(\alpha_{1,1}, \ldots, \alpha_{3,4})$ and let A be the 3×4 matrix defined by these scalars. 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 §2.2.

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 (R_1', R_2', S_1', S_2') obtained from the transformation

$$M_{t_1}{}^{2^{n-1}} = \begin{pmatrix} 1 & 0 & 2^{n-1} & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Applied to A this transformation yields

$$A' = A \cdot M_t = A + 2^{n-1} \cdot \begin{pmatrix} 0 & 0 & \alpha_{1,1} & 0 \\ 0 & 0 & \alpha_{2,1} & 0 \\ 0 & 0 & \alpha_{3,1} & 0 \end{pmatrix}.$$

The matrices A and A' define the same group G_A if and only if $[2^{n-1}][\alpha_{1,1}]S_1,\ [2^{n-1}][\alpha_{2,1}]S_1,\ [2^{n-1}][\alpha_{3,1}]S_1 \in G_A$. This is the case if and only if $\alpha_{1,1}, \alpha_{2,1}$ and $\alpha_{3,1}$ are all even. One direction is easy. For the other direction, note that if $[2^{n-1}][\alpha_{j,1}]S_1 \in G_A$, then isotropy implies $1 = e_{2^n}([\alpha_{i,1}]R_1 + [\alpha_{i,2}]R_2 + [\alpha_{i,3}]S_1 + [\alpha_{i,4}]S_2,\ [2^{n-1}\alpha_{j,1}]S_1) = e_{2^n}(R_1, S_1)^{2^{n-1}\alpha_{i,1}\cdot\alpha_{j,1}}$ for all $i \in \{1, 2, 3\}$. 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:

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

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

$$A \sim \begin{pmatrix} 1 & \alpha_{1,2}' & \alpha_{1,3}' & \alpha_{1,4}' \\ 0 & \alpha_{2,2}' & \alpha_{2,3}' & \alpha_{2,4}' \\ 0 & \alpha_{3,2}' & \alpha_{3,3}' & \alpha_{3,4}' \end{pmatrix}.$$

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 the basis obtained from the transformation

$$M_{t_3}^{2^{n-1}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2^{n-1} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

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 obtained from

$$M_{t_1}^{2^{n-1}} M_{t_3}^{2^{n-1}} M_{t_5}^{2^{n-1}} = \begin{pmatrix} 1 & 0 & 2^{n-1} & 2^{n-1} \\ 0 & 1 & 2^{n-1} & 2^{n-1} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

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 find that

$$A \sim \begin{pmatrix} 1 & 0 & \alpha''_{1,3} & \alpha''_{1,4} \\ 0 & 1 & \alpha''_{2,3} & \alpha''_{2,4} \\ 0 & 0 & \alpha''_{3,3} & \alpha''_{3,4} \end{pmatrix}.$$

Using the fact that the group G_A is isotropic, we see that $\alpha''_{3,3} = \alpha''_{3,4} = 0$, and G_A is a rank-2 group of Type 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.

$$2.2 \quad \begin{pmatrix} 1 & b & a & 0 \\ 0 & c & -b & 1 \end{pmatrix}$$

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

$$3.1 \quad \begin{pmatrix} 1 & d & a & b \\ 0 & \ell^k & \ell^k(b-cd) & \ell^k c \\ 0 & 0 & -\ell^{n-k}d & \ell^{n-k} \end{pmatrix}, \qquad 3.2 \quad \begin{pmatrix} 1 & b & a & d \\ 0 & \ell^k c & \ell^k(cd-b) & \ell^k \\ 0 & \ell^{n-k} & \ell^{n-k}d & 0 \end{pmatrix}, \qquad 4.1 \quad \begin{pmatrix} 1 & d & a & b \\ 0 & \ell^k & \ell^k b & 0 \\ 0 & 0 & -\ell^k d & \ell^k \end{pmatrix}.$$

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

$$M_{t_2}^{2^{n-1}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 2^{n-1} & 0 & 1 \end{pmatrix} \quad \text{and} \quad M_{t_4} \cdot M_{t_1}^{2^{n-1}} \cdot M_{t_4}^{-1} = \begin{pmatrix} 1 & 2^{n-1} & 2^{n-1} & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 2^{n-1} & 2^{n-1} & 1 \end{pmatrix}.$$

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

Step 4. In order to distinguish between Types 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

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

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

The first query is to test whether G_A is of Type 3.1 with $k = k_0$. Recall that we know the parity of $\alpha'_{1,2}$ and $\alpha'_{1,4}$ from the previous queries. For each iterative step, we have determined $\alpha'_{1,2} \pmod{2^{k_0}}$ and $\alpha'_{1,4} \pmod{2^{k_0}}$ from the previous queries. We write $\alpha'_{1,2} = K_{\alpha'_{1,2}} + 2^{k_0}\alpha'_{1,2,k_0} + 2^{k_0+1}\alpha''_{1,2}$ and $\alpha'_{1,4} = K_{\alpha'_{1,4}} + 2^{k_0}\alpha'_{1,4,k_0} + 2^{k_0+1}\alpha''_{1,4}$. We send the following query

$$(M_{t_3}{M_{t_1}}^{x_i}{M_{t_5}}^{y_i})^{2^{n-2k_0-1}} = \begin{pmatrix} 1 & 0 & 2^{n-2k_0-1}x_i & 2^{n-2k_0-1}y_i \\ 0 & 1 & 2^{n-2k_0-1}y_i & 2^{n-2k_0-1} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

first with $y_1 = -K_{\alpha'_{1,2}}$ and $x_1 = y_1^2$. This transformation leaves the kernel unchanged if and only if

$$\begin{aligned} 2^{n-k_0-1}\alpha_{1,2,k_0}([-\alpha_{1,2}]S_1 + S_2) &\in G_A, \\ 2^{n-2k_0-1}\alpha_{2,2}([-\alpha_{1,2}]S_1 + S_2) &\in G_A, \\ 2^{n-2k_0-1}\alpha_{3,2}([-\alpha_{1,2}]S_1 + S_2) &\in G_A. \end{aligned}$$

Using Property (3), this translates to the conditions $\alpha'_{1,2,k_0} = 0$ 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 $y_2 = -(K_{\alpha'_{1,2}} + 2^{k_0})$ and $x_2 = y_2^2$. If these malformed points produce a group distinct from 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 corresponding to the transformation $M_t = M_{t_1}^{2^{n-2k_0-1}x_i} M_{t_2}^{2^{n-2k_0-1}} M_{t_2}^{(2^{n-2k_0-1}y_i)^2} M_{t_1}^{-1} M_{t_4}^{2^{n-2k_0-1}y_i} M_{t_1} M_{t_4}^{-2^{n-2k_0-1}y_i}$ such that

$$M_t = \begin{pmatrix} 1 & 2^{n-2k_0-1}y_i & 2^{n-2k_0-1}x_i & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 2^{n-2k_0-1} & -2^{n-2k_0-1}y_i & 1 \end{pmatrix}$$

with $y_1 = -K_{\alpha'_{1,4}}$, $y_2 = -(K_{\alpha'_{1,4}} + 2^{k_0})$ 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 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 G_A is of Type 3.1 or 3.2 with $k_0 < \frac{n}{2}$, we conclude that indeed 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.

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 4. 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

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

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 G_A of Alice's isogeny has rank 2 (i.e. when $k = 0$), we again assume that the kernel generators A_1 and A_2 can be written as

$$\begin{array}{ll} A_1 = R_1 & +[a]S_1 +[b]S_2, \\ A_2 = R_2 & +[b]S_1 +[c]S_2, \end{array} \quad \text{or} \quad \begin{pmatrix} 1 & 0 & a & b \\ 0 & 1 & b & c \end{pmatrix}$$

for some $(a, b, c) \in [2^n]^3$ in accordance with the sampling of keys for symplectic bases as discussed in §2.3 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 points corresponding to

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1+2^{n-1} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

In this case

$$A' = A + \begin{pmatrix} 0 & 0 & 2^{n-1}a_0 & 0 \\ 0 & 0 & 2^{n-1}b_0 & 0 \end{pmatrix}$$

defines the same torsion subgroup as A exactly when both scalars in question are even.

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

$$\begin{pmatrix} 1 & 0 & -2^{n-1} & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1+2^{n-1} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

This results in Alice calculating with the subgroup given by

$$A' = A + \begin{pmatrix} 0 & 0 & 2^{n-1}(a_0 - 1) & 0 \\ 0 & 0 & 2^{n-1}b_0 & 0 \end{pmatrix}.$$

This subgroup is equal to G_A 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 obtained from the transformation

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -2^{n-1} & 0 \\ 0 & 0 & 1+2^{n-1} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

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 output 0, 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 try the transformation

$$\begin{pmatrix} 1 & 0 & 0 & -2^{n-1}b_0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1+2^{n-1} \end{pmatrix}.$$

Then the first kernel generator A_1 is unchanged and the second one will hold information about c_0. Explicitly, this means

$$A' = A + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2^{n-1}c_0 \end{pmatrix}.$$

Hence we can conclude that $c_0 = 0$ when the oracle returns 1, and $c_0 = 1$ 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 corresponding to

$$\begin{pmatrix} 1 & 0 & -2^{n-i-1}K_i^a & 0 \\ 0 & 1 & -2^{n-i-1}K_i^b & 0 \\ 0 & 0 & 1+2^{n-i-1} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

leads to Alice's internal computation including

$$A' = A + \begin{pmatrix} 0 & 0 & 2^{n-1}a_i & 0 \\ 0 & 0 & 2^{n-1}b_i & 0 \end{pmatrix}.$$

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

If the bits remain undetermined, we proceed with sending malformed points obtained from the transformations

$$\begin{pmatrix} 1 & 0 & -2^{n-i-1}(K_i^a + 2^i) & 0 \\ 0 & 1 & -2^{n-i-1}K_i^b & 0 \\ 0 & 0 & 1+2^{n-i-1} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{ and } \begin{pmatrix} 1 & 0 & -2^{n-i-1}K_i^a & 0 \\ 0 & 1 & 2^{n-i-1}(K_i^b + 2^i) & 0 \\ 0 & 0 & 1+2^{n-i-1} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

leading to the use of subgroups corresponding to

$$A' = A + \begin{pmatrix} 0 & 0 & 2^{n-1}(a_i - 1) & 0 \\ 0 & 0 & 2^{n-1}b_i & 0 \end{pmatrix} \text{ and } A' = A + \begin{pmatrix} 0 & 0 & 2^{n-1}a_i & 0 \\ 0 & 0 & 2^{n-1}(b_i - 1) & 0 \end{pmatrix},$$

respectively. These coincide with G_A 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 all returned 0.

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 derived from

$$\begin{pmatrix} 1 & 0 & 0 & -2^{n-i-1}K_{i+1}^b \\ 0 & 1 & 0 & -2^{n-i-1}K_i^c \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1+2^{n-i-1} \end{pmatrix}.$$

This results in a subgroup determined by

$$A' = A + \begin{pmatrix} 0\,0\,0 & 0 \\ 0\,0\,0 & 2^{n-1}c_i \end{pmatrix}$$

which coincides with G_A whenever $c_i = 0$ and the oracle returns 1, otherwise we conclude $c_i = 1$.

This way, we can recover the full key. However, additionally to being detectable by pairing checks, the query for the last bit of each key scalar can be detected by checking all points are of the correct order.

Remark 5. 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 t_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 G_A has three generators, i.e. $k > 0$. Let $1 \leq k \leq \lfloor \frac{n}{2} \rfloor$ be fixed. Then the generators can be represented by

$$\begin{array}{llllll} A_1 = & R_1 & +[d]R_2 & +[a]S_1 & +[b]S_2, & \\ A_2 = 2^k(& & R_2 & +[b-dc]S_1 & +[c]S_2), & \text{or} \\ A_3 = 2^{n-k}(& & & -[d]S_1 & +\ S_2) & \end{array} \begin{pmatrix} 1 & d & a & b \\ 0 & 2^k & 2^k(b-dc) & 2^k c \\ 0 & 0 & -2^{n-k}d & 2^{n-k} \end{pmatrix}$$

for some $(a, b, c, d) \in \{0, \dots, 2^n - 1\} \times \{0, \dots, 2^{n-k} - 1\} \times \{0, \dots, 2^{n-2k} - 1\} \times \{0, \dots, 2^k - 1\}$.

We assume here, like in the attack presented in §3, that we know the specific value of k for Alice's subgroup G_A.

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 corresponding to the transformations

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1+2^{n-1} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1+2^{n-1} \end{pmatrix}, \quad \text{and} \quad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1+2^{n-1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

respectively. By sending the first set of malformed points, we can recover the bit a_0 since Alice obtains a subgroup determined by

$$A' \sim A + \begin{pmatrix} 0 & 0 & 2^{n-1}a_0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

allowing her to recover a variety J'_{AB}. We know that this group coincides with G_A if the oracle outputs 1, and we can then deduce that $a_0 = 0$.

Analogously, when sending the malformed points querying for b_0 and d_0, respectively, only the first kernel generator might be affected which 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 given by the transformation

$$\begin{pmatrix} 1 & 0 & -2^{n-i-1}K_i^a & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1+2^{n-i-1} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

leads Alice to obtain a kernel subgroup

$$A' = A + \begin{pmatrix} 0 & 0 & 2^{n-i-1}(a-K_i^a) & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} = A + \begin{pmatrix} 0 & 0 & 2^{n-1}a_i & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

which she uses for her isogeny computations. Note here that $i+1 \leq k$.

Similarly, we can send the points obtained from the transformations

$$\begin{pmatrix} 1 & 0 & 0 & -2^{n-i-1}K_i^b \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1+2^{n-i-1} \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 1 & -2^{n-i-1}K_i^d & 0 & 0 \\ 0 & 1+2^{n-i-1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

and producing the kernel subgroups

$$A' = A + \begin{pmatrix} 0 & 0 & 0 & 2^{n-1}b_i \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad A' = A + \begin{pmatrix} 0 & 2^{n-1}d_i & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

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 given by

$$\begin{pmatrix} 1 & 0 & -2^{n-i-1}K_i^a & 0 \\ 0 & 1 & -2^{n-i-1}b & 0 \\ 0 & 0 & 1+2^{n-i-1} & 0 \\ 0 & 0 & 2^{n-i-1}d & 1 \end{pmatrix}.$$

Then Alice uses as kernel subgroup

$$A' = A + \begin{pmatrix} 0 & 0 & 2^{n-i-1}(-K_i^a - bd + a + bd) & 0 \\ 0 & 0 & 2^{n-i-1}2^k(-b+b-dc+dc) & 0 \\ 0 & 0 & 2^{n-i-1}2^{n-k}(-d+d) & 0 \end{pmatrix} = A + \begin{pmatrix} 0 & 0 & 2^{n-1}a_i & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Hence, if Alice's kernel is equal to G_A as indicated by an oracle output of 1, we can conclude $a_i = 0$. Otherwise, we may deduce that $a_i = 1$. This way, we can recover K_{n-1}^a and brute-force the most significant bit of a as the query for $i = n-1$ is detectable by checking the order of the received points.

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

Recovering c_0, b_k and a_k. Sending queries with malformed points obtained from

$$\begin{pmatrix} 1\,0\,0 & -2^{n-k-1}K_k^b \\ 0\,1\,0 & 0 \\ 0\,0\,1 & 0 \\ 0\,0\,0 & 1+2^{n-k-1} \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 1\,0\,0 & -2^{n-k-1}(K_k^b+2^k) \\ 0\,1\,0 & 0 \\ 0\,0\,1 & 0 \\ 0\,0\,0 & 1+2^{n-k-1} \end{pmatrix}$$

leads to the oracle computing varieties corresponding to the kernels determined by

$$A' = A + \begin{pmatrix} 0\,0\,0 & 2^{n-1}b_k \\ 0\,0\,0 & 2^{n-1}c_0 \\ 0\,0\,0 & 0 \end{pmatrix} \quad \text{and} \quad A' = A + \begin{pmatrix} 0\,0\,0 & 2^{n-1}(b_k-1) \\ 0\,0\,0 & 2^{n-1}c_0 \\ 0\,0\,0 & 0 \end{pmatrix}$$

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 results in 1, we can conclude $c_0 = 1$, but do not learn b_k straightaway as before.

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

$$\begin{pmatrix} 1\,0\,0 & 2^{n-k-1}(dc_0 - K_k^b) \\ 0\,1\,0 & -2^{n-k-1}c_0 \\ 0\,0\,1 & 0 \\ 0\,0\,0 & 1+2^{n-k-1} \end{pmatrix}$$

where $c_0 = 1$ to prompt Alice to use the subgroup derived from

$$A' = A + \begin{pmatrix} 0\,0\,0 & 2^{n-1}b_k \\ 0\,0\,0 & 0 \\ 0\,0\,0 & 0 \end{pmatrix}$$

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 given by

$$\begin{pmatrix} 1\,0 & -2^{n-k-1}(d(d_0c_0 - b_0) + K_k^a) & 0 \\ 0\,1 & -2^{n-k-1}(b_0 - d_0c_0) & 0 \\ 0\,0 & 1+2^{n-k-1} & 0 \\ 0\,0 & 0 & 1 \end{pmatrix}.$$

The kernel subgroup corresponding to these points is generated by

$$A' = A + \begin{pmatrix} 0\,0 & 2^{n-1}a_k & 0 \\ 0\,0 & 0 & 0 \\ 0\,0 & 0 & 0 \end{pmatrix}$$

Again we can conclude that $a_k = 0$ if and only if the oracle outputs 1.

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 (b_i, c_{i-k}). We first try

$$\begin{pmatrix} 1\,0\,0 & 2^{n-i-1}(dK_i^c - K_i^b) \\ 0\,1\,0 & -2^{n-i-1}K_i^c \\ 0\,0\,1 & 0 \\ 0\,0\,0 & 1+2^{n-i-1} \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 1\,0\,0 & 2^{n-i-1}(dK_i^c - K_i^b - 2^i) \\ 0\,1\,0 & -2^{n-i-1}K_i^c \\ 0\,0\,1 & 0 \\ 0\,0\,0 & 1+2^{n-i-1} \end{pmatrix}$$

which yield kernel subgroups defined by

$$A' = A + \begin{pmatrix} 0\ 0\ 0 & 2^{n-1}b_i \\ 0\ 0\ 0 & 2^{n-1}c_{i-k} \\ 0\ 0\ 0 & 0 \end{pmatrix} \quad \text{and} \quad A' = A + \begin{pmatrix} 0\ 0\ 0 & 2^{n-1}(b_i - 1) \\ 0\ 0\ 0 & 2^{n-1}c_{i-k} \\ 0\ 0\ 0 & 0 \end{pmatrix}$$

respectively. Hence, we can conclude $(b_i, c_{i-k}) = (0, 0)$ or $(b_i, c_{i-k}) = (1, 0)$ if one of the oracle queries returns 1.

From the previous queries, we have either determined that $c_{i-k} = 0$ or can deduce that $c_{i-k} = 1$. Therefore, we now know K^c_{i+1} and can send a last set of malformed points obtained from

$$\begin{pmatrix} 1\ 0\ 0 & 2^{n-i-1}(dK^c_{i+1} - K^b_i) \\ 0\ 1\ 0 & -2^{n-i-1}K^c_{i+1} \\ 0\ 0\ 1 & 0 \\ 0\ 0\ 0 & 1 + 2^{n-i-1} \end{pmatrix}$$

to determine the bit b_i from the kernel subgroup corresponding to

$$A' = A + \begin{pmatrix} 0\ 0\ 0 & 2^{n-1}b_i \\ 0\ 0\ 0 & 0 \\ 0\ 0\ 0 & 0 \end{pmatrix}$$

which Alice uses in her computation.

For recovering a_i for $i \leq n-2$, we again use the transformation

$$\begin{pmatrix} 1\ 0 & -2^{n-i-1}K^a_i & 0 \\ 0\ 1 & -2^{n-i-1}K^b_{i+1} & 0 \\ 0\ 0 & 1 + 2^{n-i-1} & 0 \\ 0\ 0 & 2^{n-i-1}d & 1 \end{pmatrix}.$$

This query leads to the use of the kernel subgroup corresponding to

$$A' = A + \begin{pmatrix} 0\ 0 & 2^{n-1}a_i & 0 \\ 0\ 0 & 0 & 0 \\ 0\ 0 & 0 & 0 \end{pmatrix}$$

which is equal to G_A and hence leads to an oracle output of 1 exactly when $a_i = 0$. The remaining bit can 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 $i \geq n-k$ using $b = K^b_{i+1}$ and $c = K^c_{i+1}$.