

The supersingular isogeny problem in genus 2 and beyond

Craig Costello, Benjamin Smith

▶ **To cite this version:**

Craig Costello, Benjamin Smith. The supersingular isogeny problem in genus 2 and beyond. PQCrypto 2020 - 11th International Conference on Post-Quantum Cryptography, Apr 2020, Paris, France. hal-02389073v2

HAL Id: hal-02389073
https://inria.hal.science/hal-02389073v2

Submitted on 27 Jan 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

The supersingular isogeny problem in genus 2 and beyond

Craig Costello[1] and Benjamin Smith[2]

[1] Microsoft Research, USA
craigco@microsoft.com
[2] Inria *and* École polytechnique, Institut Polytechnique de Paris, Palaiseau, France
smith@lix.polytechnique.fr

Abstract. Let $A/\overline{\mathbb{F}}_p$ and $A'/\overline{\mathbb{F}}_p$ be superspecial principally polarized abelian varieties of dimension $g > 1$. For any prime $\ell \neq p$, we give an algorithm that finds a path $\phi\colon A \to A'$ in the $(\ell, \dots, \ell)$-isogeny graph in $\widetilde{O}(p^{g-1})$ group operations on a classical computer, and $\widetilde{O}(\sqrt{p^{g-1}})$ calls to the Grover oracle on a quantum computer. The idea is to find paths from A and A' to nodes that correspond to products of lower dimensional abelian varieties, and to recurse down in dimension until an elliptic path-finding algorithm (such as Delfs–Galbraith) can be invoked to connect the paths in dimension $g = 1$. In the general case where A and A' are any two nodes in the graph, this algorithm presents an asymptotic improvement over all of the algorithms in the current literature. In the special case where A and A' are a known and relatively small number of steps away from each other (as is the case in higher dimensional analogues of SIDH), it gives an asymptotic improvement over the quantum claw finding algorithms and an asymptotic improvement over the classical van Oorschot–Wiener algorithm.

1 Introduction

Isogenies of supersingular elliptic curves are now well-established in cryptography, from the Charles–Goren–Lauter hash function [10] to Jao and De Feo's SIDH key exchange [27] and beyond [2,21,12,13]. While the security of isogeny-based cryptosystems depend on the difficulty of a range of computational problems, the fundamental one is the *isogeny problem*: given supersingular elliptic curves $\mathcal{E}_1$ and $\mathcal{E}_2$ over $\mathbb{F}_{p^2}$, find a walk in the ℓ-isogeny graph connecting them.

One intriguing aspect of isogeny-based cryptography is the transfer of elliptic-curve techniques from classic discrete-log-based cryptography into the post-quantum arena. In this spirit, it is natural to consider cryptosystems based on isogeny graphs of higher-dimensional abelian varieties, mirroring the transition from elliptic (ECC) to hyperelliptic-curve cryptography (HECC). Compared with elliptic supersingular isogeny graphs, the higher-dimensional graphs have more vertices and higher degrees for a given p, which allows some interesting tradeoffs (for example: in dimension $g = 2$, we get the same number of vertices with a p of one-third the bitlength).

For $g = 2$, Takashima [36] and Castryck, Decru, and Smith [7] have defined CGL-style hash functions, while Costello [11] and Flynn and Ti [19] have already proposed SIDH-like key exchanges. Generalizations to dimensions $g > 2$, using isogeny algorithms such as those in [4], are easy to anticipate; for example, a family of hash functions on isogeny graphs of superspecial abelian varieties with real multiplication was hinted at in [9].

So far, when estimating security levels, these generalizations assume that the higher-dimensional supersingular isogeny problem is basically as hard as the elliptic supersingular isogeny problem in graphs of the same size. In this article, we show that this assumption is false. The general supersingular isogeny problem can be partially reduced to a series of lower-dimensional isogeny problems, and thus recursively to a series of elliptic isogeny problems.

Theorem 1. *There exists a classical algorithm which, given a prime ℓ and superspecial abelian varieties $\mathcal{A}_1$ and $\mathcal{A}_2$ of dimension g over $\overline{\mathbb{F}}_p$ with $p \neq \ell$, succeeds with probability $\geq 1/2^{g-1}$ in computing a composition of $(\ell, \ldots, \ell)$-isogenies from $\mathcal{A}_1$ to $\mathcal{A}_2$, running in expected time $\widetilde{O}((p^{g-1}/P))$ on P processors as $p \to \infty$ (with ℓ fixed).*

Given that these graphs have $O(p^{g(g+1)/2})$ vertices, the expected runtime for generic random-walk algorithms is $\widetilde{O}(p^{g(g+1)/4}/P)$. Our algorithm therefore represents a substantial speedup, with nontrivial consequences for cryptographic parameter selection.[3] We also see an improvement in quantum algorithms:

Theorem 2. *There exists a quantum algorithm which, given a prime ℓ and superspecial abelian varieties $\mathcal{A}_1$ and $\mathcal{A}_2$ of dimension g over $\overline{\mathbb{F}}_p$ with $p \neq \ell$, computes a composition of $(\ell, \ldots, \ell)$-isogenies from $\mathcal{A}_1$ to $\mathcal{A}_2$ running in expected time $\widetilde{O}(\sqrt{p^{g-1}})$ as $p \to \infty$ (with ℓ fixed).*

This reflects the general pattern seen in the passage from ECC to HECC: the dimension grows, the base field shrinks—-and the mathematical structures become more complicated, which can ultimately reduce claimed security levels. Just as index calculus attacks on discrete logarithms become more powerful in higher genus, where useful structures appear in Jacobians [15,23,22,34], so interesting structures in higher-dimensional isogeny graphs provide attacks that become more powerful as the dimension grows. Here, the interesting structures are (relatively large) subgraphs corresponding to increasing numbers of elliptic factors in (polarized) abelian varieties. These subgraphs are relatively large, and so random-walking into them is relatively easy. We can then glue together elliptic isogenies, found with an elliptic path-finding algorithm, to form product isogenies between products of elliptic curves, and thus to solve the original isogeny problem. We will see that the path-finding problem in the superspecial graph gets asymptotically easier as the dimension grows.

[3] Our algorithms apply to the full superspecial graph; we do not claim any impact on cryptosystems that run in small and special subgraphs, such as CSIDH [8].

Notation and conventions. Throughout, p denotes a prime > 3, and ℓ a prime not equal to p. Typically, p is large, and $\ell \ll \log(p)$ is small enough that computing $(\ell, \ldots, \ell)$-isogenies of g-dimensional principally polarized abelian varieties (PPAVs) is polynomial in $\log(p)$. Similarly, we work with PPAVs in dimensions $g \ll \log p$; in our asymptotics and complexities, g and ℓ are fixed. We say a function $f(X)$ is in $\widetilde{O}(g(X))$ if $f(X) = O(h(\log X)g(X))$ for some polynomial h.

2 The elliptic supersingular isogeny graph

An elliptic curve $\mathcal{E}/\overline{\mathbb{F}}_p$ is *supersingular* if $\mathcal{E}[p](\overline{\mathbb{F}}_p) = 0$. We have a number of efficient algorithms for testing supersingularity: see Sutherland [35] for discussion.

Supersingularity is isomorphism-invariant, and any supersingular $\mathcal{E}$ has j-invariant $j(\mathcal{E})$ in $\mathbb{F}_{p^2}$; and in fact the curve $\mathcal{E}$ can be defined over $\mathbb{F}_{p^2}$. We let

$$S_1(p) := \left\{ j(\mathcal{E}) : \mathcal{E}/\mathbb{F}_{p^2} \text{ is supersingular} \right\} \subset \mathbb{F}_{p^2}$$

be the set of isomorphism classes of supersingular elliptic curves over $\overline{\mathbb{F}}_p$. It is well-known that

$$\#S_1(p) = \left\lfloor \frac{p}{12} \right\rfloor + \epsilon_p \tag{1}$$

where $\epsilon_p = 0$ if $p \equiv 1 \pmod{12}$, 2 if $p \equiv -1 \pmod{12}$, and 1 otherwise.

Now fix a prime $\ell \neq p$, and consider the directed multigraph $\Gamma_1(\ell; p)$ whose vertex set is $S_1(p)$, and whose edges correspond to ℓ-isogenies between curves (again, up to isomorphism). The graph $\Gamma_1(\ell; p)$ is $(\ell+1)$-regular: there are (up to isomorphism) $\ell + 1$ distinct ℓ-isogenies from a supersingular elliptic curve $\mathcal{E}/\mathbb{F}_{p^2}$ to other elliptic curves, corresponding to the $\ell + 1$ order-ℓ subgroups of $\mathcal{E}[\ell](\overline{\mathbb{F}}_p) \cong (\mathbb{Z}/\ell\mathbb{Z})^2$ that form their kernels. But since supersingularity is isogeny-invariant, the codomain of each isogeny is again supersingular; that is, the $\ell+1$ order-ℓ subgroups of $\mathcal{E}[\ell]$ are in bijection with the edges out of $j(\mathcal{E})$ in $\Gamma_1(\ell; p)$.

Definition 1. *A* walk *of length n in $\Gamma_1(\ell; p)$ is a sequence of edges $j_0 \to j_1 \to \cdots \to j_n$. A* path *in $\Gamma_1(\ell; p)$ is an acyclic (and, in particular, non-backtracking) walk: that is, a walk $j_0 \to j_1 \to \cdots \to j_n$ such that $j_i = j_{i'}$ if and only if $i = i'$.*

Pizer [32] proved that $\Gamma_1(\ell; p)$ is Ramanujan: in particular, $\Gamma_1(\ell; p)$ is a connected expander graph, and its diameter is $O(\log p)$. We therefore expect the end-points of short random walks from any given vertex j_0 to quickly yield a uniform distribution on $S_1(p)$. Indeed, if j_0 is fixed and j_n is the end-point of an n-step random walk from j_0 in $\Gamma_1(\ell; p)$, then [21, Theorem 1] shows that

$$\left| \Pr[j_n = j] - \frac{1}{\#S_1(p)} \right| \leq \left(\frac{2\sqrt{\ell}}{\ell+1} \right)^n \qquad \text{for all } j \in S_1(p)\,. \tag{2}$$

The *isogeny problem* in $\Gamma_1(\ell; p)$ is, given j_0 and j in $S_1(p)$, to find a path (of any length) from j_0 to j in $\Gamma_1(\ell; p)$. The difficulty of the isogeny problem underpins the security of the Charles–Goren–Lauter hash function (see §3 below).

The isogeny problem is supposed to be hard. Our best generic classical path-finding algorithms look for collisions in random walks, and run in expected time the square root of the graph size: in this case, $\widetilde{O}(\sqrt{p})$. In the special case of supersingular isogeny graphs, we can make some practical improvements but the asymptotic complexity remains the same: given j_0 and j in $F_1(p;\ell)$, we can compute a path $j_0 \to j$ in $\widetilde{O}(\sqrt{p})$ classical operations (see [14]).

The best known quantum algorithm for path-finding [3] instead searches for paths from $j_0 \to j_0'$ and from $j \to j'$, where j_0' and j' are both in $\mathbb{F}_p$. Of the $O(p)$ elements in $S_1(p)$, there are $O(\sqrt{p})$ elements contained in $\mathbb{F}_p$; while a classical search for elements this sparse would therefore run in time $O(\sqrt{p})$, Grover's quantum algorithm [24] completes the search in expected time $O(\sqrt[4]{p})$. It remains to find a path from j_0' to j'. This could be computed classically in time $\widetilde{O}(\sqrt[4]{p})$ using the Delfs–Galbraith algorithm, but Biasse, Jao and Sankar [3] show that a quantum computer can find paths between subfield curves in subexponential time, yielding an overall algorithm that runs in expected time $O(\sqrt[4]{p})$.

We can also consider the problem of finding paths of a fixed (and typically *short*) length: for example, given $e > 0$ and j_0 and j in $S_1(p)$ such that there exists a path $\phi : j_0 \to \cdots \to j$ of length e, find ϕ. This problem arises in the security analysis of SIDH, for example.

3 Cryptosystems in the elliptic supersingular graph

The Charles–Goren–Lauter hash function (CGL). Supersingular isogenies appeared in cryptography with the CGL hash function, which operates in $\Gamma_1(2;p)$. Fix a base point j_0 in $S_1(p)$, and one of the three edges in $\Gamma_1(2;p)$ leading into it: $j_{-1} \to j_0$, say. To hash an n-bit message $m = (m_0, m_1, \ldots, m_{n-1})$, we let m drive a non-backtracking walk $j_0 \to \cdots \to j_n$ on $\Gamma_1(2;p)$: for each $0 \le i < n$, we compute the two roots α_0 and α_1 of $\Phi_2(j_i, X)/(j_{i-1} - X)$ to determine the neighbours of j_i that are not j_{i-1}, numbering the roots with respect to some ordering of $\mathbb{F}_{p^2}$ (here $\Phi_2(Y, X)$ is the classical modular polynomial), and set $j_{i+1} = \alpha_{m_i}$.

Once we have computed the entire walk $j_0 \to \cdots \to j_n$, we can derive a $\log_2 p$-bit hash value $H(m)$ from the end-point j_n; we call this step *finalisation*. Charles, Goren, and Lauter suggest applying a linear function $f : \mathbb{F}_{p^2} \to \mathbb{F}_p$ to map j_n to $H(m) = f(j_n)$. For example, if $\mathbb{F}_{p^2} = \mathbb{F}_p(\omega)$ then we can map $j_n = j_{n,0} + j_{n,1}\omega$ (with $j_{n,0}$ and $j_{n,1}$ in $\mathbb{F}_p$) to $H(m) = aj_{n,0} + bj_{n,1}$ for some fixed random choice of a and b in $\mathbb{F}_p$. Heuristically, for general f, if we suppose $S_1(p)$ is distributed uniformly in $\mathbb{F}_{p^2}$, then roughly one in twelve elements of $\mathbb{F}_p$ appear as hash values, and each of those has only one expected preimage in $S_1(p)$.

Finding a preimage for a given hash value h in $\mathbb{F}_p$ amounts to finding a path $j_0 \to \cdots \to j$ such that $f(j) = h$: that is, solving the isogeny problem. We note that inverting the finalisation seems hard: for linear $f : \mathbb{F}_p^2 \to \mathbb{F}_p$, we know of no efficient method which given h in $\mathbb{F}_p$ computes a supersingular j such that $f(j) = h$. (Brute force search requires $O(p)$ trials.) Finalisation thus gives us some protection against meet-in-the-middle isogeny algorithms. Finding

collisions and second preimages for H amounts to finding cycles in $\Gamma_1(2;p)$. For well-chosen p and j_0, this is roughly as hard as the isogeny problem [10, §5].

SIDH. Jao and De Feo's SIDH key exchange [27] begins with a supersingular curve $\mathcal{E}_0/\mathbb{F}_{p^2}$, where p is in the form $c\cdot 2^a 3^b - 1$, with fixed torsion bases $\langle P_2, Q_2\rangle = \mathcal{E}_0[2^a]$ and $\langle P_3, Q_3\rangle = \mathcal{E}_0[3^b]$ (which are rational because of the special form of p). Alice computes a secret walk $\phi_A : \mathcal{E}_0 \to \cdots \to \mathcal{E}_A$ of length a in $\Gamma_1(2;p)$, publishing $\mathcal{E}_A$, $\phi_A(P_3)$, and $\phi_A(Q_3)$; similarly, Bob computes a secret walk $\phi_B : \mathcal{E}_0 \to \cdots \to \mathcal{E}_B$ of length b in $\Gamma_1(3;p)$, publishing $\mathcal{E}_B$, $\phi_B(P_2)$, and $\phi_B(Q_2)$. The basis images allow Alice to compute $\phi_B(\ker \phi_A)$, and Bob $\phi_A(\ker \phi_B)$; Alice can thus "repeat" her walk starting from $\mathcal{E}_B$, and Bob his walk from $\mathcal{E}_A$, to arrive at curves representing the same point in $S_1(p)$, which is their shared secret.

Breaking Alice's public key amounts to solving an isogeny problem in $\Gamma_1(2;p)$ subject to the constraint that the walk have length a (which is particularly short). The 3^b-torsion basis may give some useful information here, though so far this is only exploited in attacks on artificial variants of SIDH [31]. Similarly, breaking Bob's public key amounts to solving a length-b isogeny problem in $\Gamma_1(3;p)$. Alternatively, we can compute these short paths by computing endomorphism rings: [20, Theorem 4.1] states that if $\mathcal{E}$ and $\mathcal{E}'$ are in $S_1(p)$ and we have *explicit descriptions* of $\mathrm{End}(\mathcal{E})$ and $\mathrm{End}(\mathcal{E}')$, then we can efficiently compute the *shortest* path from $\mathcal{E}$ to $\mathcal{E}'$ in $\Gamma_1(\ell;p)$ (see [29,20,17] for further details on this approach).

4 Abelian varieties and polarizations

An abelian variety is a smooth projective algebraic group variety. An isogeny of abelian varieties is a surjective finite morphism $\phi : \mathcal{A} \to \mathcal{A}'$ such that $\phi(0_{\mathcal{A}}) = 0_{\mathcal{A}'}$. In dimension $g = 1$, these definitions coincide with those for elliptic curves.

The proper higher-dimensional generalization of an elliptic curve is a *principally polarized abelian variety* (PPAV). A *polarization* of $\mathcal{A}$ is an isogeny $\lambda : \mathcal{A} \to \widehat{\mathcal{A}}$, where $\widehat{\mathcal{A}} \cong \mathrm{Pic}^0(\mathcal{A})$ is the *dual* abelian variety; λ is *principal* if it is an isomorphism. If $\mathcal{A} = \mathcal{E}$ is an elliptic curve, then there is a canonical principal polarization $\lambda : P \mapsto [(P) - (\infty)]$, and every other principal polarization is isomorphic to λ (via composition with a suitable translation and automorphism). The Jacobian $\mathcal{J}_{\mathcal{C}}$ of a curve $\mathcal{C}$ also has a canonical principal polarization defined by the theta divisor, which essentially corresponds to an embedding of $\mathcal{C}$ in $\mathcal{J}_{\mathcal{C}}$, and thus connects $\mathcal{J}_{\mathcal{C}}$ with the divisor class group of $\mathcal{C}$.

We need a notion of compatibility between isogenies and principal polarizations. First, recall that every isogeny $\phi : \mathcal{A} \to \mathcal{A}'$ has a dual isogeny $\widehat{\phi} : \widehat{\mathcal{A}'} \to \widehat{\mathcal{A}}$. Now, if $(\mathcal{A}, \lambda)$ and $(\mathcal{A}', \lambda')$ are PPAVs, then $\phi : \mathcal{A} \to \mathcal{A}'$ is an *isogeny of PPAVs* if $\widehat{\phi} \circ \lambda' \circ \phi = [d]\lambda$ for some integer d. We then have $\phi^\dagger \circ \phi = [d]$ on $\mathcal{A}$ (and $\phi \circ \phi^\dagger = [d]$ on $\mathcal{A}'$), where $\phi^\dagger := \lambda^{-1} \circ \widehat{\phi} \circ \lambda'$ is the *Rosati dual*. Intuitively, ϕ will be defined by homogeneous polynomials of degree d with respect to projective coordinate systems on $\mathcal{A}$ and $\mathcal{A}'$ corresponding to λ and λ', respectively. There

is a simple criterion on subgroups $S \subset \mathcal{A}[d]$ to determine when an isogeny with kernel S is an isogeny of PPAVs: the subgroup should be *Lagrangian.*[4]

Definition 2. *Let $\mathcal{A}/\overline{\mathbb{F}}_p$ be a PPAV and let m be an integer prime to p. A* Lagrangian subgroup *of $\mathcal{A}[m]$ is a maximal m-Weil isotropic subgroup of $\mathcal{A}[m]$.*

If $\ell \neq p$ is prime, then $\mathcal{A}[\ell^n] \cong (\mathbb{Z}/\ell^n\mathbb{Z})^{2g}$ for all $n > 0$. If $S \subset \mathcal{A}[\ell]$ is Lagrangian, then $S \cong (\mathbb{Z}/\ell\mathbb{Z})^g$. Any Lagrangian subgroup of $A[\ell^n]$ is isomorphic to $(\mathbb{Z}/\ell\mathbb{Z})^{n_1} \times \cdots \times (\mathbb{Z}/\ell\mathbb{Z})^{n_g}$ for some $n_1 \geq \cdots \geq n_g$ with $\sum_i n_i = gn$ (though not every $(n_1, \ldots, n_g)$ with $\sum_i n_i = gn$ occurs in this way).

We now have almost everything we need to generalize supersingular isogeny graphs from elliptic curves to higher dimension. The elliptic curves will be replaced by PPAVs; ℓ-isogenies will be replaced by isogenies with Lagrangian kernels in the ℓ-torsion—called $(\ell, \ldots, \ell)$-isogenies—and the elliptic dual isogeny will be replaced by the Rosati dual. It remains to define the right analogue of supersingularity in higher dimension, and study the resulting graphs.

5 The superspecial isogeny graph in dimension g

We need an appropriate generalization of elliptic supersingularity to $g > 1$. As explained in [7], it does not suffice to simply take the PPAVs $\mathcal{A}/\overline{\mathbb{F}}_p$ with $\mathcal{A}[p] = 0$.

Definition 3. *A PPAV $\mathcal{A}$ is* ***supersingular*** *if the Newton polygon of its Frobenius endomorphism has all slopes equal to $1/2$, and* ***superspecial*** *if Frobenius acts as 0 on $H^1(\mathcal{A}, \mathcal{O}_\mathcal{A})$. Superspecial implies supersingular; in dimension $g = 1$, the definitions coincide.*

All supersingular PPAVs are isogenous to a product of supersingular elliptic curves. Superspecial abelian varieties are isomorphic to a product of supersingular elliptic curves, though generally only as *unpolarized* abelian varieties. The special case of Jacobians is particularly relevant for us when constructing examples: $\mathcal{J}_\mathcal{C}$ is superspecial if and only if the Hasse–Witt matrix of $\mathcal{C}$ vanishes.

It is argued in [7] that the world of superspecial (and not supersingular) PPAVs is the correct setting for supersingular isogeny-based cryptography. We will not repeat this argument here; but in any case, every higher-dimensional "supersingular" cryptosystem proposed so far has in fact been superspecial.

In analogy with the elliptic supersingular graph, then, we define

$$S_g(p) := \left\{\mathcal{A} : \mathcal{A}/\mathbb{F}_{p^2} \text{ is a superspecial } g\text{-dimensional PPAV}\right\} / \cong .$$

Our first task is to estimate the size of $S_g(p)$.

Lemma 1. *We have $\#S_g(p) = O(p^{g(g+1)/2})$.*

[4] Isogenies with strictly smaller kernels exist—isogenies with cyclic kernel are treated algorithmically in [16]— but these isogenies are not relevant to this investigation.

Proof. See [18, §5]. This follows from the Hashimoto–Ibukiyama mass formula

$$\sum_{A \in S_g(p)} \frac{1}{\#\mathrm{Aut}(A)} = \prod_{i=1}^{g} \frac{B_{2i}}{4i}(1 + (-p)^i)\,,$$

where B_{2i} is the $2i$-th Bernoulli number. In particular, $\#S_g(p)$ is a polynomial in p of degree $\sum_{i=1}^{g} i = g(g+1)/2$. □

Note that $\#S_g(p)$ grows *quadratically* in g (and exponentially in $\log p$): we have $\#S_1(p) = O(p)$, $\#S_2(p) = O(p^3)$, $\#S_3(p) = O(p^6)$, and $\#S_4(p) = O(p^{10})$.

For each prime $\ell \neq p$, we let $\Gamma_g(\ell; p)$ denote the (directed) graph on $S_g(p)$ whose edges are $\overline{\mathbb{F}}_p$-isomorphism classes of $(\ell, \cdots, \ell)$-isogenies of PPAVs: that is, isogenies whose kernels are Lagrangian subgroups of the ℓ-torsion. Superspeciality is invariant under $(\ell, \ldots, \ell)$-isogeny, so to determine the degree of the vertices of $\Gamma_g(\ell; p)$ it suffices to enumerate the Lagrangian subgroups of a g-dimensional PPAV. A simple counting argument yields Lemma 2.

Lemma 2. *If $\mathcal{A}/\overline{\mathbb{F}}_p$ is a g-dimensional PPAV, then the number of Lagrangian subgroups of $\mathcal{A}[\ell]$, and hence the number of edges leaving $\mathcal{A}$ in $\Gamma_g(\ell; p)$, is*

$$N_g(\ell) := \sum_{d=0}^{g} \begin{bmatrix} g \\ d \end{bmatrix}_\ell \cdot \ell^{\binom{g-d+1}{2}}\,.$$

(The ℓ-binomial coefficient $\begin{bmatrix} n \\ k \end{bmatrix}_\ell := \frac{(n)_\ell \cdots (n-k+1)_\ell}{(k)_\ell \cdots (1)_\ell}$, where $(i)_\ell := \frac{\ell^i - 1}{\ell - 1}$, counts the k-dimensional subspaces of $\mathbb{F}_\ell^n$.) In particular, $\Gamma_g(\ell; p)$ is $N_g(\ell)$-regular; and $N_g(\ell)$ is a polynomial in ℓ of degree $g(g+1)/2$.

We do not yet have analogues of Pizer's theorem to guarantee that $\Gamma_g(\ell; p)$ is Ramanujan when $g > 1$, though this is proven for superspecial abelian varieties with real multiplication [26]. We therefore work on the following hypothesis:

Hypothesis 1. *The graph $\Gamma_g(\ell; p)$ is Ramanujan.*

We need Hypothesis 1 in order to obtain the following analogue of Eq. 2 (a standard random walk theorem, as in [25, §3]): if we fix a vertex $\mathcal{A}_0$ and consider n-step random walks $\mathcal{A}_0 \to \cdots \to \mathcal{A}_n$, then

$$\left| \Pr[\mathcal{A}_n \cong \mathcal{A}] - \frac{1}{\#S_g(p)} \right| \leq \left(\frac{2\sqrt{N_g(\ell) - 1}}{N_g(\ell)} \right)^n \qquad \text{for all } \mathcal{A} \in S_g(p)\,. \tag{3}$$

That is, random walks in $\Gamma_g(\ell; p)$ converge exponentially quickly to the uniform distribution: after $O(\log p)$ steps in $\Gamma_g(\ell; p)$ we are uniformly distributed over $S_g(p)$. Given specific ℓ and g, we can explicitly derive the constant hidden by the big-O to bound the minimum n yielding a distribution within $1/\#S_g(p)$ of uniform.

Remark 1. Existing proposals of higher-dimensional supersingular isogeny-based cryptosystems all implicitly assume (special cases of) Hypothesis 1. For the purposes of attacking their underlying hard problems, we are comfortable making the same hypothesis. After all, if our algorithms are less effective because the expansion properties of $\Gamma_g(\ell;p)$ are less than ideal, then the cryptosystems built on $\Gamma_g(\ell;p)$ will fail to be effective by the same measure.

6 Superspecial cryptosystems in dimension $g = 2$

Before attacking the isogeny problem in $\Gamma_g(\ell;p)$, we consider some of the cryptosystems that have recently been defined in $\Gamma_2(\ell;p)$. This will also illustrate some methods for computing in these graphs, and as well as special cases of the general phenomena that can help us solve the isogeny problem more efficiently. For the rest of this section, therefore, we restrict to dimension $g = 2$.

Every 2-dimensional PPAV is isomorphic (as a PPAV) to either the Jacobian of a genus-2 curve, or to a product of two elliptic curves. We can therefore split $S_2(p)$ naturally into two disjoint subsets: $S_2(p) = S_2(p)^J \sqcup S_2(p)^E$, where

$$S_2(p)^J := \{\mathcal{A} \in S_2(p) : \mathcal{A} \cong \mathcal{J}_\mathcal{C} \text{ with } g(\mathcal{C}) = 2\} \quad \text{and}$$
$$S_2(p)^E := \{\mathcal{A} \in S_2(p) : \mathcal{A} \cong \mathcal{E}_1 \times \mathcal{E}_2 \text{ with } \mathcal{E}_1, \mathcal{E}_2 \in S_1(p)\} \ .$$

Vertices in $S_2(p)^J$ are "general", while vertices in $S_2(p)^E$ are "special". We can make the estimates implied by Lemma 1 more precise: if $p > 5$, then

$$\#S_2(p)^J = \frac{1}{2880}p^3 + \frac{1}{120}p^2 \qquad \text{and} \qquad \#S_2(p)^E = \frac{1}{288}p^2 + O(p)$$

(see e.g. [7, Proposition 2]). In particular, $\#S_2(p)^E/\#S_2(p) = 10/p + o(1)$.

Takashima's hash function. Takashima [36] was the first to generalize CGL to $g = 2$. We start with a distinguished vertex $\mathcal{A}_0$ in $S_2(p)$, and a distinguished incoming edge $\mathcal{A}_{-1} \to \mathcal{A}_0$ in $\Gamma_2(\ell;p)$. Each message m then drives a walk in $\Gamma_2(\ell;p)$: at each vertex we have a choice of 14 forward isogenies (the 15th is the dual of the previous, which is a prohibited backtracking step). The message m is therefore coded in base 14. While traversing the graph, the vertices are handled as concrete genus-2 curves representing the isomorphism classes of their Jacobians. Lagrangian subgroups correspond to factorizations of the hyperelliptic polynomials into a set of three quadratics, and the isogenies are computed using Richelot's formulæ (see [6, Chapters 9-10] and [33, Chapter 8]). We derive a hash value From the final vertex $\mathcal{A}_n$ as the Igusa–Clebsch invariants of the Jacobian, in $\mathbb{F}_{p^2}^3$; Takashima does not define a finalisation map (into $\mathbb{F}_p^3$, for example).

Flynn and Ti observe in [19] that this hash function has a fatal weakness: it is trivial to compute length-4 cycles starting from any vertex in $\Gamma_2(2;p)$, as in Example 1. Every cycle produces infinitely many hash collisions.

Example 1. Given some $\mathcal{A}_0$ in $S_2(p)$, choose a point P of order 4 on $\mathcal{A}_0$. There exist Q and R in $\mathcal{A}_0[2]$ such that $e_2([2]P, Q) = 1$ and $e_2([2]P, R) = 1$, but $e_2(Q, R) \neq 1$. The Lagrangian subgroups $K_0 := \langle [2]P, Q\rangle$ and $K_0' := \langle [2]P, R\rangle$ of $\mathcal{A}_0[2]$ are kernels of $(2,2)$-isogenies $\phi_0 : \mathcal{A}_0 \to \mathcal{A}_1 \cong \mathcal{A}_0/K_0$ and $\phi_0' : \mathcal{A}_0 \to \mathcal{A}_1' \cong \mathcal{A}_0/K_0'$; and in general, $\mathcal{A}_1 \ncong \mathcal{A}_1'$. Now $K_1 := \phi_0(K_0')$ and $K_1' := \phi_0'(K_0)$ are Lagrangian subgroups of $\mathcal{A}_1[2]$. Writing $I_1 = \ker {\phi_1}^\dagger$ and $I_1' = \ker {(\phi_1')}^\dagger$, we see that $K_1 \cap I_1 = \langle \phi_1(R)\rangle$ and $K_1' \cap I_1' = \langle \phi_1(Q)\rangle$. We thus define another pair of $(2,2)$-isogenies, $\phi_1 : \mathcal{A}_1 \to \mathcal{A}_2 \cong \mathcal{A}_1/K_1$ and $\phi_1' : \mathcal{A}_1' \to \mathcal{A}_2' \cong \mathcal{A}_1'/K_1'$. We have $\ker(\phi_1 \circ \phi_0) = \ker(\phi_1' \circ \phi_0')$, so $\mathcal{A}_2 \cong \mathcal{A}_2'$. Now let $\psi := {(\phi_0')}^\dagger \circ {(\phi_1')}^\dagger \circ \phi_1 \circ \phi_0$. We have $\psi \cong [4]_{\mathcal{A}_0}$, but ψ does not factor over $[2]_{\mathcal{A}_0}$ (since $\mathcal{A}_1 \ncong \mathcal{A}_1'$). Hence ψ represents a nontrivial cycle of length 4 in the graph.

The ubiquity of these length-4 cycles does not mean that $\Gamma_2(2;p)$ is no use for hashing: it just means that we must use a stronger rule than backtrack-avoidance when selecting steps in a walk. The following hash function does just this.

The Castryck–Decru–Smith hash function (CDS). Another generalization of CGL from $\Gamma_1(2;p)$ to $\Gamma_2(2;p)$, neatly avoiding the length-4 cycles of Example 1, is defined in [7]. Again, we fix a vertex $\mathcal{A}_0$ and an isogeny $\phi_{-1} : \mathcal{A}_{-1} \to \mathcal{A}_0$; we let $I_0 \subset \mathcal{A}_0[2]$ be the kernel of the Rosati dual $\phi_{-1}^\dagger$. Now, let $m = (m_0, \ldots, m_{n-1})$ be a $3n$-bit message, with each $0 \leq m_i < 8$. The sequence $(m_0, \ldots, m_{n-1})$ drives a path through $\Gamma_2(2;p)$ as follows: our starting point is $\mathcal{A}_0$, with its distinguished subgroup I_0 corresponding to the edge $\mathcal{A}_{-1} \to \mathcal{A}_0$. For each $0 \leq i < n$, we compute the set of eight Lagrangian subgroups $\{S_{i,0}, \ldots, S_{i,7}\}$ of $\mathcal{A}_i[2]$ such that $S_{i,j} \cap I_i = 0$, numbering them according to some fixed ordering on the encodings of Lagrangian subgroups. Then we compute $\phi_i : \mathcal{A}_i \to \mathcal{A}_{i+1} \cong \mathcal{A}_i/S_{i,m_i}$, and let $I_{i+1} := \phi_i(\mathcal{A}_i[2]) = \ker {\phi_i}^\dagger$. Once we have computed the entire walk $\mathcal{A}_0 \to \cdots \to \mathcal{A}_n$, we can derive a $3\log_2 p$-bit hash value $H(m)$ from the isomorphism class of $\mathcal{A}_n$ (though such a finalisation is unspecified in [7]). The subgroup intersection condition ensures that the composition of the isogenies in the walk is a $(2^n, \ldots, 2^n)$-isogeny, thus protecting us from the small cycles of Example 1.

Putting this into practice reveals an ugly technicality. As in Takashima's hash function, we compute with vertices as genus-2 curves, encoded by their hyperelliptic polynomials, with $(2,2)$-isogenies computed using Richelot's formulæ. Walk endpoints are mapped to Igusa–Clebsch invariants in $\mathbb{F}_{p^2}^3$. But these curves, formulæ, and invariants only exist for vertices in $S_2(p)^J$. We can handle vertices in $S_2(p)^E$ as pairs of elliptic curves, with pairs of j-invariants for endpoints, and there are explicit formulæ to compute isogenies in to and out of $S_2(p)^E$ (see e.g. [7, §3]). Switching between representations and algorithms (to say nothing of finalisation, where $S_2(p)^E$ would have a smaller, easily distinguishable, and easier-to-invert image) seems like needless fiddle when the probability of stepping onto a vertex in $S_2(p)^E$ is only $O(1/p)$, which is negligible for cryptographic p.

In [7], this issue was swept under the rug by defining simpler algorithms which efficiently walk in the subgraph of $\Gamma_2(2;p)$ supported on $S_2(p)^J$, and simply fail if they walk into $S_2(p)^E$. This happens with probability $O(1/p)$, which may seem acceptable—however, this also means that *it is exponentially easier to find a*

message where the hash fails than it is to find a preimage with a square-root algorithm. The former requires $O(p)$ work, the latter $O(p^{3/2})$. In this, as we will see, the simplified CDS hash function contains the seeds of its own destruction.

Genus-2 SIDH. Flynn and Ti [19] defined an SIDH analogue in dimension $g = 2$. As in the hash functions above, Richelot isogenies are used for Alice's steps in $\Gamma_2(2; p)$, while explicit formulæ for $(3, 3)$-isogenies on Kummer surfaces are used for Bob's steps in $\Gamma_2(3; p)$. Walks may (improbably) run into $S_2(p)^E$, as with the hash functions above; but the same work-arounds apply without affecting security. (Further, if we generate a public key in $S_2(p)^E$, then we can discard it and generate a new one in $S_2(p)^J$.) As with SIDH, breaking public keys amounts to computing *short* solutions to the isogeny problem in $\Gamma_2(2; p)$ or $\Gamma_2(3; p)$, though presumably endomorphism attacks generalizing [17] also exist.

7 Attacking the isogeny problem in superspecial graphs

We want to solve the isogeny problem in $\Gamma_g(\ell; p)$. We can always do this using random walks in $O(\sqrt{\#S_g(p)}) = O(p^{g(g+1)/4})$ classical steps.

Our idea is that $S_{g-1}(p) \times S_1(p)$ maps into $S_g(p)$ by mapping a pair of PPAVs to their product equipped with the product polarization, and the image of $S_{g-1}(p) \times S_1(p)$ represents a large set of easily-identifiable "distinguished vertices" in $\Gamma_g(\ell; p)$. Indeed, since the map $S_{g-1}(p) \times S_1(p) \to S_g(p)$ is generically finite, of degree independent of p, Lemma 1 implies that

$$\#S_g(p)/\#(\text{image of } S_{g-1}(p) \times S_1(p)) = O(p^{g-1}) \qquad \text{for } g > 1\,. \tag{4}$$

We can efficiently detect such a step into a product PPAV in a manner analogous to that of the failure of the CDS hash function: for example, by the breakdown of a higher-dimensional analogue of Richelot's formulæ such as [30].

We can walk into this subset, then recursively solve the path-finding problem in the subgraphs $\Gamma_{g-1}(\ell; p), \ldots, \Gamma_1(\ell; p)$ (each time walking from $\Gamma_i(\ell; p)$ into $\Gamma_{i-1}(\ell; p) \times \Gamma_1(\ell; p)$) before gluing the results together to obtain a path in $\Gamma_g(\ell; p)$.

Lemma 3. *Let $\alpha : A \to A'$ and $\beta : B \to B'$ be walks in $\Gamma_i(\ell; p)$ and $\Gamma_j(\ell; p)$ of lengths a and b, respectively. If $a \equiv b \pmod 2$, then we can efficiently compute a path of length $\max(a, b)$ from $A \times B$ to $A' \times B'$ in $\Gamma_{i+j}(\ell; p)$.*

Proof. Write $\alpha = \alpha_1 \circ \cdots \circ \alpha_a$ and $\beta = \beta_1 \circ \cdots \circ \beta_b$ as compositions of $(\ell, \cdots, \ell)$-isogenies. WLOG, suppose $a \geq b$. Set $\beta_{b+1} = {\beta_b}^\dagger$, $\beta_{b+2} = \beta_b$, ..., $\beta_{a-1} = {\beta_b}^\dagger$, $\beta_a = \beta_b$; then $\alpha \times \beta : (\alpha_1 \times \beta_1) \circ \cdots \circ (\alpha_a \times \beta_a)$ is a path from $A \times B$ to $A' \times B'$. □

Equations 3 and 4 show that a walk of length $O(\log p)$ lands in the image of $S_{g-1}(p) \times S_1(p)$ with probability $O(1/p^{g-1})$, and after $O(p^{g-1})$ such short walks we are in $S_{g-1}(p) \times S_1(p)$ with probability bounded away from zero. More generally, we can walk into the image of $S_{g-i}(p) \times S_i(p)$ for any $0 < i < g$; but the probability of this is $O(1/p^{i(g-i)})$, which is maximised by $i = 1$ and $g - 1$.

Algorithm 1: Computing isogeny paths in $\Gamma_g(\ell; p)$

Input: $\mathcal{A}$ and $\mathcal{A}'$ in $S_g(p)$
Output: A path $\phi : \mathcal{A} \to \mathcal{A}'$ in $\Gamma_g(\ell; p)$
1 Find a path ψ from $\mathcal{A}$ to some point $\mathcal{B} \times \mathcal{E}$ in $S_{g-1}(p) \times S_1(p)$
2 Find a path ψ' from $\mathcal{A}'$ to some point $\mathcal{B}' \times \mathcal{E}'$ in $S_{g-1}(p) \times S_1(p)$
3 Find a path $\beta : \mathcal{B} \to \mathcal{B}'$ in $\Gamma_{g-1}(\ell; p)$ using Algorithm 1 recursively if $g - 1 > 1$, or elliptic path-finding if $g - 1 = 1$
4 Find a path $\eta : \mathcal{E} \to \mathcal{E}'$ in $\Gamma_1(\ell; p)$ using elliptic path-finding
5 Let $b = \text{length}(\beta)$ and $e = \text{length}(\eta)$. If $b \not\equiv e \pmod 2$, then fail and return $\perp$ (or try again with another ψ and/or ψ', β, or η)
6 Construct the product path $\pi : \mathcal{B} \times \mathcal{E} \to \mathcal{B}' \times \mathcal{E}'$ defined by Lemma 3.
7 **return** the path $\phi := \psi'^{\dagger} \circ \pi \circ \psi$ from $\mathcal{A}$ to $\mathcal{A}'$.

Proof of Theorem 1 Algorithm 1 implements the approach above, and proves Theorem 1. Step 1 computes ψ by taking $O(p^{g-1})$ non-backtracking random walks of length $O(\log(p))$ which can be trivially parallelized, so with P processors we expect $\widetilde{O}(p^{g-1}/P)$ steps before finding ψ. (If $\mathcal{A}$ is a fixed public base point then we can assume ψ is already known). Likewise, Step 2 takes $\widetilde{O}(p^{g-1}/P)$ steps to compute ψ'. After $g-1$ recursive calls, we have reduced to the problem of computing paths in $\Gamma_1(\ell; p)$ in Step 4, which can be done in time $O(\sqrt{p}/P)$. Step 7 applies Lemma 3 to compute the final path in polynomial time. At each level of the recursion, we have a $1/2$ chance of having the same walk-length parity; hence, Algorithm 1 succeeds with probability $1/2^{g-1}$. This could be improved by computing more walks when the parities do not match, but $1/2^{g-1}$ suffices to prove the theorem. The total runtime is $\widetilde{O}(p^{g-1}/P)$ isogeny steps.

Proof of Theorem 2 Algorithm 1 can be run in a quantum computation model as follows. First, recall from the proof of Theorem 1 that Steps 1 and 2 find product varieties by taking $O(p^{g-1})$ walks of length $O(\log(p))$. Here we proceed following Biasse, Jao and Sankar [3, §4]. Let N be the number of walks in $O(p^{g-1})$ of length λ (in $O(\log(p))$). To compute ψ, we define an injection

$$f\colon [1, \ldots, N] \longrightarrow \{\text{nodes of distance } \lambda \text{ starting from } \mathcal{A}\},$$

and a function $C_f\colon [1, \ldots, N] \to \{0, 1\}$ by $C_f(x) = 1$ if $f(x)$ is in $S_{g-1}(p) \times S_1(p)$, and 0 otherwise. If there is precisely one x with $C_f(x) = 1$, Grover's algorithm [24] will find it (with probability $\geq 1/2$) in $O(\sqrt{N})$ iterations. If there are an unknown $t \geq 1$ such solutions, then Boyer–Brassard–Høyer–Tapp [5] finds one in $O(\sqrt{N/t})$ iterations. Hence, if we take λ large enough to expect at least one solution, then we will find it in $O(\sqrt{p^{g-1}})$ Grover iterations. We compute ψ' (and any recursive invocations of Steps 1 and 2) similarly.

For the elliptic path finding in Steps 3 and 4, we can apply (classical) Pollard-style pseudorandom walks which require $\widetilde{O}(\sqrt{p})$ memory and $\widetilde{O}(\sqrt{p})$ operations to find an ℓ-isogeny path. Alternatively, we can reduce storage costs by applying

Grover's algorithm to the full graph $\Gamma_1(\ell; p)$ to find an ℓ-isogeny path in expected time $O(\sqrt{p})$. Finally, Step 7 applies Lemma 3 to compute the final path.

Remark 2. We can use the same approach as Algorithm 1 to compute explicit endomorphism rings of superspecial PPAVs. Suppose we want to compute $\mathrm{End}(\mathcal{A})$ for some g-dimensional $\mathcal{A}$ in $S_g(p)$. Following the first steps of Algorithm 1, we compute a walk ϕ from $\mathcal{A}$ into $S_{g-1}(p) \times S_1(p)$, classically or quantumly, recursing until we end up at some $\mathcal{E}_1 \times \cdots \times \mathcal{E}_g$ in $S_1(p)^g$. Now we apply an elliptic endomorphism-ring-computing algorithm to each of the $\mathcal{E}_i$; this is equivalent to solving the isogeny problem in $\Gamma_1(\ell; p)$ (see [17, §5]), so its cost is in $\widetilde{O}(\sqrt{p})$. The products of the generators for the $\mathrm{End}(\mathcal{E}_i)$ form generators for $\mathrm{End}(\mathcal{E}_1 \times \cdots \times \mathcal{E}_g)$, which we can then pull back through ϕ to compute a finite-index subring of $\mathrm{End}(\mathcal{A})$ that is maximal away from ℓ. The total cost is a classical $\widetilde{O}(p^{g-1}/P)$ (on P processors), or a quantum $\widetilde{O}(\sqrt{p^{g-1}})$, plus the cost of the pullback.

Remark 3. Algorithm 1 computes compositions of $(\ell, \ldots, \ell)$-isogenies. If we relax and allow arbitrary-degree isogenies, not just paths in $\Gamma_g(\ell; p)$ for fixed ℓ, then the elliptic path-finding steps can use the classical Delfs–Galbraith [14] or quantum Biasse–Jao–Sankar [3] algorithms. While this would not change the asymptotic runtime of Algorithm 1 (under the reasonable assumption that the appropriate analogue of vertices "defined over $\mathbb{F}_p$" with commutative endomorphism rings form a subset of size $O(\sqrt{\#S_g(p)})$), both of these algorithms have low memory requirements and are arguably more implementation-friendly than Pollard-style pseudorandom walks [14, §4].

8 Cryptographic implications

Table 1 compares Algorithm 1 with the best known attacks for dimensions $g \leq 6$. For general path-finding, the best known algorithms are classical Pollard-style pseudorandom walks and quantum Grover search [24,5]. As noted in Remark 3, higher-dimensional analogues of Delfs–Galbraith [14] or Biasse–Jao–Sankar [3] might yield practical improvements, without changing the asymptotic runtime.

Table 1. Logarithms (base p) of asymptotic complexities of algorithms for solving the isogeny problems in $\Gamma_g(\ell; p)$ for $1 \leq g \leq 6$. Further explanation in text.

	Dimension g	1	2	3	4	5	6
Classical	**Algorithm 1**	—	**1**	**2**	**3**	**4**	**5**
	Pollard/Delfs–Galbraith [14]	0.5	1.5	3	5	7.5	10.5
Quantum	**Algorithm 1**	—	**0.5**	**1**	**1.5**	**2**	**2.5**
	Grover/Biasse–Jao–Sankar [3]	0.25	0.75	1.5	2.5	3.75	4.25

The paths in $\Gamma_g(\ell; p)$ constructed by Algorithm 1 are generally too long to be private keys for SIDH analogues, which are paths of a fixed and typically shorter

length. Extrapolating from $g = 1$ [27] and $g = 2$ [19], we suppose that the secret keyspace has size $O(\sqrt{\#S_g(p)}) = O(p^{g(g+1)/4})$ and the target isogeny has degree in $O(\sqrt{p})$, corresponding to a path of length roughly $\log_\ell(p)/2$ in $\Gamma_g(\ell; p)$. On the surface, therefore, Algorithm 1 does not yield a direct attack on SIDH-style protocols; or, at least, not a direct attack that succeeds with high probability. (Indeed, to resist direct attacks from Algorithm 1, it would suffice to abort any key generations passing through vertices in $S_{g-1}(p) \times S_1(p)$.)

However, we can anticipate an attack via endomorphism rings, generalizing the attack described at the end of §3, using the algorithm outlined in Remark 2. If we assume that what is polynomial-time for elliptic endomorphisms remains so for (fixed) $g > 1$, then we can break g-dimensional SIDH keys by computing shortest paths in $\Gamma_g(\ell; p)$ with the same complexity as Algorithm 1: that is, classical $\widetilde{O}(p^{g-1}/P)$ and quantum $\widetilde{O}(p^{(g-1)/2})$ for $g > 1$.

This conjectural cost compares very favourably against the best known classical and quantum attacks on g-dimensional SIDH. In the classical paradigm, a meet-in-the-middle attack would run in $\widetilde{O}(p^{g(g+1)/8})$, with similar storage requirements. In practice the best attack is the golden-collision van Oorschot–Wiener (vOW) algorithm [38] investigated in [1], which given storage w runs in expected time $\widetilde{O}(p^{3g(g+1)/16}/(P\sqrt{w}))$. For fixed w, the attack envisioned above gives an asymptotic improvement over vOW for all $g > 1$. If an adversary has access to a large amount of storage, then vOW may still be the best classical algorithm for $g \leq 5$, particularly when smaller primes are used to target lower security levels. (vOW becomes strictly worse for all $g > 5$, even if we assume unbounded storage.) In the quantum paradigm, Tani's algorithm [37] would succeed in $\widetilde{O}(p^{g(g+1)/12})$, meaning we get the same asymptotic complexities for dimensions 2 and 3, and an asymptotic improvement for all $g > 3$. Moreover, Jaques and Schanck [28] suggest a significant gap between the asymptotic runtime of Tani's algorithm and its actual efficacy in any meaningful model of quantum computation. On the other hand, the bottleneck of the quantum attack forecasted above is a relatively straightforward invocation of Grover search, and the gap between its asymptotic and concrete complexities is likely to be much closer.

Like the size of $S_g(p)$, the exponents in the runtime complexities of all of the algorithms above are quadratic in g. Indeed, this was the practical motivation for instantiating isogeny-based cryptosystems in $g > 1$. In contrast, the exponents for Algorithm 1 and our proposed SIDH attack are linear in g. This makes the potential trade-offs for cryptosystems based on higher-dimensional supersingular isogeny problems appear significantly less favourable, particularly as g grows and the gap between the previous best attacks and Algorithm 1 widens.

References

1. G. Adj, D. Cervantes-Vázquez, J. Chi-Domínguez, A. Menezes, and F. Rodríguez-Henríquez. On the cost of computing isogenies between supersingular elliptic curves. In *25th Conference on Selected Areas in Cryptography (SAC 2018), to appear*, 2018, preprint: `https://eprint.iacr.org/2018/313`.

2. R. Azarderakhsh, B. Koziel, M. Campagna, B. LaMacchia, C. Costello, P. Longa, L. De Feo, M. Naehrig, B. Hess, J. Renes, A. Jalali, V. Soukharev, D. Jao, and D. Urbanik. Supersingular isogeny key encapsulation, 2017.
3. J. Biasse, D. Jao, and A. Sankar. A quantum algorithm for computing isogenies between supersingular elliptic curves. In W. Meier and D. Mukhopadhyay, editors, *Progress in Cryptology - INDOCRYPT 2014 - 15th International Conference on Cryptology in India, New Delhi, India, December 14-17, 2014, Proceedings*, volume 8885 of *Lecture Notes in Computer Science*, pages 428–442. Springer, 2014.
4. G. Bisson, R. Cosset, and D. Robert. AVIsogenies – a library for computing isogenies between abelian varieties, November 2012. URL: `http://avisogenies.gforge.inria.fr`.
5. M. Boyer, G. Brassard, P. Høyer, and A. Tapp. Tight bounds on quantum searching. *Fortschritte der Physik: Progress of Physics*, 46(4-5):493–505, 1998.
6. J. W. S. Cassels and E. V. Flynn. *Prolegomena to a middlebrow arithmetic of curves of genus 2*, volume 230 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, 1996.
7. W. Castryck, T. Decru, and B. Smith. Hash functions from superspecial genus-2 curves using Richelot isogenies. Cryptology ePrint Archive, Report 2019/296, 2019. To appear in the proceedings of NuTMiC 2019.
8. W. Castryck, T. Lange, C. Martindale, L. Panny, and J. Renes. CSIDH: An efficient post-quantum commutative group action. In T. Peyrin and S. Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part III*, pages 395–427. Springer International Publishing, 2018.
9. D. X. Charles, E. Z. Goren, and K. E. Lauter. Families of Ramanujan graphs and quaternion algebras. In J. Harnad and P. Winternitz, editors, *Groups and symmetries, from neolithic scots to John McKay*, pages 53–80. AMS, 2009.
10. D. X. Charles, K. E. Lauter, and E. Z. Goren. Cryptographic hash functions from expander graphs. *Journal of Cryptology*, 22(1):93–113, 2009.
11. C. Costello. Computing supersingular isogenies on Kummer surfaces. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 428–456. Springer, 2018.
12. L. De Feo and S. Galbraith. SeaSign: Compact isogeny signatures from class group actions. In *Advances in Cryptology – EUROCRYPT 2019*, 2019. To appear.
13. L. De Feo, S. Masson, C. Petit, and A. Sanso. Verifiable delay functions from supersingular isogenies and pairings. Cryptology ePrint Archive, Report 2019/166, 2019.
14. C. Delfs and S. D. Galbraith. Computing isogenies between supersingular elliptic curves over $\mathbb{F}_p$. *Des. Codes Cryptography*, 78(2):425–440, 2016.
15. C. Diem. An index calculus algorithm for plane curves of small degree. In F. Hess, S. Pauli, and M. E. Pohst, editors, *Algorithmic Number Theory, 7th International Symposium, ANTS-VII, Berlin, Germany, July 23-28, 2006, Proceedings*, volume 4076 of *Lecture Notes in Computer Science*, pages 543–557. Springer, 2006.
16. A. Dudeanu, D. Jetchev, D. Robert, and M. Vuille. Cyclic Isogenies for Abelian Varieties with Real Multiplication, Nov. 2017. preprint.
17. K. Eisenträger, S. Hallgren, K. Lauter, T. Morrison, and C. Petit. Supersingular isogeny graphs and endomorphism rings: reductions and solutions. In J. B. Nielsen and V. Rijmen, editors, *Advances in cryptology—EUROCRYPT 2018. Part III*, pages 329–368. Springer International Publishing, 2018.
18. T. Ekedahl. On supersingular curves and abelian varieties. *Mathematica Scandinavica*, 60:151–178, 1987.

19. E. V. Flynn and Y. B. Ti. Genus two isogeny cryptography. In J. Ding and R. Steinwandt, editors, *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019, Chongqing, China, May 8-10, 2019 Revised Selected Papers*, volume 11505 of *Lecture Notes in Computer Science*, pages 286–306. Springer, 2019.
20. S. D. Galbraith, C. Petit, B. Shani, and Y. B. Ti. On the security of supersingular isogeny cryptosystems. In J. H. Cheon and T. Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 63–91, 2016.
21. S. D. Galbraith, C. Petit, and J. Silva. Identification protocols and signature schemes based on supersingular isogeny problems. In T. Takagi and T. Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2017.
22. P. Gaudry. Index calculus for abelian varieties of small dimension and the elliptic curve discrete logarithm problem. *J. Symb. Comput.*, 44(12):1690–1702, 2009.
23. P. Gaudry, E. Thomé, N. Thériault, and C. Diem. A double large prime variation for small genus hyperelliptic index calculus. *Math. Comput.*, 76(257):475–492, 2007.
24. L. K. Grover. A fast quantum mechanical algorithm for database search. In G. L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 212–219. ACM, 1996.
25. S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. *Bulletin (New Series) of the American Mathematical Society*, 43(4):439–561, 2006.
26. M.-N. Hubert. *Superspecial abelian varieties, theta series and the Jacquet–Langlands correspondence*. PhD thesis, McGill University, 2005.
27. D. Jao and L. De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In *International Workshop on Post-Quantum Cryptography*, pages 19–34. Springer, 2011.
28. S. Jaques and J. M. Schanck. Quantum cryptanalysis in the RAM model: Claw-finding attacks on SIKE. In A. Boldyreva and D. Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 32–61. Springer, 2019.
29. D. Kohel, K. Lauter, C. Petit, and J.-P. Tignol. On the quaternion ℓ-isogeny path problem. *LMS J. Comput. Math.*, 17(suppl. A):418–432, 2014.
30. D. Lubicz and D. Robert. Arithmetic on abelian and Kummer varieties. *Finite Fields and Their Applications*, 39:130–158, 2016.
31. C. Petit. Faster algorithms for isogeny problems using torsion point images. In T. Takagi and T. Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II*, volume 10625 of *Lecture Notes in Computer Science*, pages 330–353. Springer, 2017.
32. A. K. Pizer. Ramanujan graphs and hecke operators. *Bull. Am. Math. Soc.*, 23(1), 1990.
33. B. Smith. *Explicit endomorphisms and correspondences*. PhD thesis, University of Sydney, 2005.

34. B. A. Smith. Isogenies and the discrete logarithm problem in jacobians of genus 3 hyperelliptic curves. *J. Cryptology*, 22(4):505–529, 2009.
35. A. V. Sutherland. Identifying supersingular elliptic curves. *LMS Journal of Computation and Mathematics*, 15:317–325, 2012.
36. K. Takashima. *Mathematical Modelling for Next-Generation Cryptography: CREST Crypto-Math Project*, chapter Efficient Algorithms for Isogeny Sequences and Their Cryptographic Applications, pages 97–114. Springer Singapore, Singapore, 2018.
37. S. Tani. Claw finding algorithms using quantum walk. *Theor. Comput. Sci.*, 410(50):5285–5297, 2009.
38. P. C. van Oorschot and M. J. Wiener. Parallel collision search with cryptanalytic applications. *J. Cryptology*, 12(1):1–28, 1999.

A A proof-of-concept implementation

We include a naive Magma implementation of the product finding stage (i.e. Steps 1-3) of Algorithm 1 in dimension $g = 2$ with $\ell = 2$. First, it generates a challenge by walking from the known superspecial node corresponding to the curve $\mathcal{C}\colon y^2 = x^5 + x$ over a given $\mathbb{F}_{p^2}$ to a random abelian surface in $\Gamma_2(2; p)$, which becomes the target $\mathcal{A}$. Then it starts computing random walks of length slightly larger than $\log_2(p)$, whose steps correspond to $(2, 2)$-isogenies. As each step is taken, it checks whether we have landed on a product of two elliptic curves (at which point it will terminate) before continuing.

Magma's built-in functionality for $(2, 2)$-isogenies makes this rather straightforward. At a given node, the function `RichelotIsogenousSurfaces` computes all 15 of its neighbours, so our random walks are simply a matter of generating enough entropy to choose one of these neighbours at each of the $O(\log(p))$ steps. For the sake of replicability, we have used Magma's inbuilt implementation of SHA-1 to produce pseudo-random walks that are deterministically generated by an input seed. SHA-1 produces 160-bit strings, which correspond to 40 integers in $[0, 1, \ldots, 15]$; this gives a straightforward way to take 40 pseudo-random steps in $\Gamma_2(2; p)$, where no step is taken if the integer is 0, and otherwise the index is used to choose one of the 15 neighbours.

The seed `processor` can be used to generate independent walks across multiple processors. We always used the seed "0" to generate the target surface, and set `processor` to be the string "1" to kickstart a single process for very small primes. For the second and third largest primes, we used the strings "1", "2", ..., "16" as seeds to 16 different deterministic processes. For the largest prime, we seeded 128 different processes.

For the prime $p = \mathbf{127} = 2^7 - 1$, the seed "0" walks us to the starting node corresponding to $C_0/\mathbb{F}_{p^2}\colon y^2 = (41i + 63)x^6 + \cdots + (6i + 12)x + 70$. The single processor seeded with "1" found a product variety $E_1 \times E_2$ on its second walk after taking **53 steps** in total, with $E_1/\mathbb{F}_{p^2}\colon y^2 = x^3 + (93i + 43)x^2 + (23i + 93)x + (2i + 31)$ and $E_2/\mathbb{F}_{p^2}\colon y^2 = x^3 + (98i + 73)x^2 + (30i + 61)x + (41i + 8)$.

For the prime $p = \mathbf{8191} = 2^{13} - 1$, the single processor seeded with "1" found a product variety on its 175-th walk after taking **6554 steps** in total.

For the prime $p = \mathbf{524287} = 2^{19} - 1$, all 16 processors were used. The processor seeded with "2" was the first to find a product variety on its 311-th walk after taking 11680 steps in total. Given that all processors walk at roughly the same pace, at this stage we would have walked close to $16 \cdot 11680 = \mathbf{186880}$ **steps**.

For the 25-bit prime $p = \mathbf{17915903} = 2^{13}3^7 - 1$, the processor seeded with "13" found a product variety after taking 341 walks and a total of 12698 steps. At this stage the 16 processors would have collectively taken around **203168 steps**.

The largest experiment that we have conducted to date is with the prime $p = \mathbf{2147483647} = 2^{31} - 1$, where 128 processors walked in parallel. Here the processor seeded with "95" found a product variety after taking 10025 walks and a total of 375703 steps. At this stage the processors would have collectively taken around **48089984 steps**.

In all of the above cases we see that product varieties are found with around p steps. The Magma script that follows can be used to verify the experiments[5], or to experiment with other primes.

[5] Readers without access to Magma can make use of the free online calculator at `http://magma.maths.usyd.edu.au/calc/`, omitting the "Write" functions at the end that are used to print to local files.

```
////////////////////////////////////////////////////////////
clear;

processor:="1";

p:=2^13-1;
Fp:=GF(p);
Fp2<i>:=ExtensionField<Fp,x|x^2+1>;
_<x>:=PolynomialRing(Fp2);

////////////////////////////////////////////////////////////

Next_Walk := function(str)
    H := SHA1(str);
        steps := [ StringToInteger(x, 16): x in ElementToSequence(H) | x ne "0"];
    return steps ,H;
end function;

////////////////////////////////////////////////////////////

Walk_To_Starting_Jacobian:=function(str)

        steps,H:= Next_Walk(str);

        C0:=HyperellipticCurve(x^5+x);
        J0:=Jacobian(C0);
        for i:=1 to #steps do
                neighbours:=RichelotIsogenousSurfaces(J0);
                if Type(neighbours[steps[i]]) ne SetCart then
                        J0:=neighbours[steps[i]];
                end if;
        end for;

        return J0;

end function;

////////////////////////////////////////////////////////////

Walk_Until_Found:=function(seed,J0);

        found:=false;
        H:=seed;
        found:=false;
        walks_done:=0;
        steps_done:=0;

        while not found do

                walks_done+:=1;
                walks_done, "walks and",steps_done, "steps on core", processor, "for p=",p;
                J:=J0;
                steps,H:=Next_Walk(H);

                for i:=1 to #steps do
                        steps_done+:=1;
                        J:=RichelotIsogenousSurfaces(J)[steps[i]];
                        if Type(J) eq SetCart then
                                found:=true;
                                index:=i;
                                break;
                        end if;
                end for;

        end while;

        return steps,index,walks_done,steps_done,J;

end function;

////////////////////////////////////////////////////////////

file_name:="p" cat IntegerToString(p)  cat "-" cat processor cat ".txt";
J0:=Walk_To_Starting_Jacobian("0");
steps,index,walks_done,steps_done,J:=Walk_Until_Found(processor,J0);

Write(file_name, "walks done =");
Write(file_name, walks_done);
Write(file_name, "steps_done =");
Write(file_name, steps_done);
Write(file_name, "steps=");
Write(file_name, steps);
Write(file_name, "index=");
Write(file_name, index);
Write(file_name, "Elliptic Product=");
Write(file_name, J);

////////////////////////////////////////////////////////////
```