Failing to hash into supersingular isogeny graphs

Jeremy Booher[1], Ross Bowden[2], Javad Doliskani[3], Tako Boris Fouotsa[4], Steven D. Galbraith[5], Sabrina Kunzweiler[6], Simon-Philipp Merz[7], Christophe Petit[8,13], Benjamin Smith[9], Katherine E. Stange[10], Yan Bo Ti[11], Christelle Vincent[12], José Felipe Voloch[1], Charlotte Weitkämper[13], and Lukas Zobernig[5]

[1] School of Mathematics and Statistics, University of Canterbury, Christchurch, New Zealand
jeremy.booher,felipe.voloch@canterbury.ac.nz
[2] Department of Computer Science, University of Bristol, Bristol, UK
ross.bowden@bristol.ac.uk
[3] Department of Computer Science, Ryerson University, Toronto, Canada
javad.doliskani@ryerson.ca
[4] LASEC, EPFL, Lausanne, Switzerland
tako.fouotsa@epfl.ch
[5] Department of Mathematics, The University of Auckland, Auckland, New Zealand
s.galbraith,lukas.zobernig@auckland.ac.nz
[6] Ruhr-Universität Bochum, Bochum, Germany
sabrina.kunzweiler@ruhr-uni-bochum.de
[7] Royal Holloway, University of London, London, UK
simon-philipp.merz.2018@rhul.ac.uk
[8] Laboratoire d'Informatique,
Université libre de Bruxelles, Bruxelles, Belgium
christophe.f.petit@gmail.com
[9] Inria and Laboratoire d'Informatique (LIX), CNRS, École polytechnique, Institut Polytechnique de Paris, Palaiseau, France
smith@lix.polytechnique.fr
[10] Department of Mathematics, University of Colorado Boulder, Boulder, Colorado, USA
kstange@math.colorado.edu
[11] DSO, Singapore
yanbo.ti@gmail.com
[12] Department of Mathematics and Statistics, University of Vermont, Burlington, Vermont, USA
christelle.vincent@uvm.edu
[13] University of Birmingham, Birmingham, UK
c.weitkaemper@pgr.bham.ac.uk

Abstract. An important open problem in supersingular isogeny-based cryptography is to produce, without a trusted authority, concrete examples of "hard supersingular curves" that is, equations for supersingular curves for which computing the endomorphism ring is as difficult as it is for random supersingular curves. A related open problem is to produce a hash function to the vertices of the supersingular ℓ-isogeny graph which does not reveal the endomorphism ring, or a path to a curve of known endomorphism ring. Such a hash function would open up interesting cryptographic applications. In this paper, we document a number of (thus far) failed attempts to solve this problem, in the hope that we may spur further research, and shed light on the challenges and obstacles to this endeavour. The mathematical approaches contained in this article include: (i) iterative root-finding for the supersingular polynomial; (ii) gcd's of specialized modular polynomials; (iii) using division polynomials to create small systems of equations; (iv) taking random walks in the isogeny graph of abelian surfaces; and (v) using quantum random walks.

[*] Jeremy Booher was supported by a grant from the Marsden Fund Council administered by the Royal Society of New Zealand. Ross Bowden was supported by EPSRC grant EP/T517872/1. Javad Doliskani was supported by the

1 Introduction

Supersingular curves (and isogenies between them) have become a hot topic in cryptography over the last ten years or so. Fortunately the theory of complex multiplication provides efficient algorithms to generate a supersingular elliptic curve over $\mathbb{F}_{p^2}$, even for the astronomically large p that are used for cryptographic applications (see Bröker [11]). It is also known how to uniformly sample a supersingular elliptic curve over $\mathbb{F}_{p^2}$: generate one curve E_0 using Bröker's method and then take a sufficiently long random walk in the supersingular isogeny graph to get a curve E.

There are several flavors of isogeny-based cryptography. One of the earliest proposals was the cryptographic hash function based on isogenies, by Charles, Goren and Lauter [19]. Another early proposal was to obtain a group action of the ideal class group on a set of elliptic curves. This was first proposed by Couveignes [24] and re-discovered by Rostovtsev and Stolbunov [61]. Class group actions were made practical with the CSIDH scheme by Castryck, Lange, Martindale, Panny and Renes [17]. Various digital signature schemes have been proposed [8, 26, 32, 33, 38, 68]. But the most studied isogeny-based cryptosystem of all is the key exchange protocol SIDH, by Jao and De Feo [42]. Public keys in the SIDH protocol include not only elliptic curves but also certain auxiliary points on these curves. The SIDH protocol has been a highly active area of research for over 10 years, but very recently major advances in cryptanalysis by Castryck and Decru [16], Maino and Martindale [49] and Robert [59] completely break SIDH, by exploiting the auxiliary points. It remains to be seen whether some variant of SIDH can be secure and practical. Note that the other areas of isogeny-based cryptography, such as CSIDH and the signature schemes, do not use auxiliary points and so are not affected by the attack. This paper was written before SIDH was broken, and we will refer to some results and papers that may not be relevant anymore. Nevertheless, the general problems considered in this paper are still relevant for isogeny cryptography and remain worthy of study.

One of the main computational problems in isogeny-based cryptography is to compute an isogeny between two given supersingular elliptic curves over the same finite field $\mathbb{F}_{p^2}$. This problem is called the *supersingular isogeny problem* or the *path finding problem in the supersingular isogeny graph*. It is believed to be hard, even for quantum computers. A related problem is the *supersingular endomorphism ring problem*: Given a supersingular elliptic curve E over $\mathbb{F}_{p^2}$, compute its endomorphism ring $\mathrm{End}(E)$ (or even just one non-trivial endomorphism of E). The supersingular endomorphism ring problem and the supersingular isogeny problem are related [30, 37, 67].

The algorithm using complex multiplication sketched in the first paragraph for generating a uniformly distributed supersingular curve has the side-effect that the person who generated the curve also knows a path from E_0 to E. In certain cryptographic applications this approach is not acceptable as it allows a user to insert a trapdoor or in some other way violate the desired security. There are a number of papers that have already mentioned this problem [1, 10, 18, 25, 52]. Currently

Natural Sciences and Engineering Research Council of Canada (NSERC). Steven Galbraith was supported by funding from the Ministry for Business, Innovation and Employment, New Zealand. Sabrina Kunzweiler was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972. Simon-Philipp Merz was supported by EPSRC and the UK government as part of the grant EP/P009301/1. Christophe Petit was partially supported by EPSRC grants EP/S01361X/1 and EP/V011324/1. Benjamin Smith was partially supported by l'Agence nationale de la recherche (ANR) program CIAO ANR-19-CE48-0008. Katherine E. Stange was supported by NSF-CAREER CNS-1652238 and by Simons Fellowship 822143. Christelle Vincent was supported by NSF grant DMS-1802323. José Felipe Voloch was partially supported by grants from the Ministry of Business, Innovation and Employment and the Marsden Fund Council administered by the Royal Society of New Zealand. Date of this document: 2022-10-19.

the only solution known is to involve some “trusted party” to generate a random curve and then “forget” any resulting secret information. There is great interest in finding better ways to solve this problem that do not require trusting a single party. Among other applications, it would circumvent the trusted setup in an isogeny-based verifiable delay function [25], in delay encryption [13] and in an SIDH-based oblivious pseudorandom function [9]. For the latter, the necessity of the trusted setup was pointed out by [6]. Before SIDH was broken, using a starting curve that is generated uniformly at random would prevent torsion point attacks [47, 55, 57].

Applications of hashing to hard supersingular curves might include hash-and-sign signatures, oblivious pseudorandom functions [9] and password-authenticated key exchange [5].

There are (at least) three general problems that are of interest for isogeny-based cryptography:

1. Given a prime p, to compute a supersingular curve E over $\mathbb{F}_{p^2}$ without revealing anything about the endomorphism ring or providing any information to help solve the isogeny problem (for isogenies from E to some other supersingular curve over $\mathbb{F}_{p^2}$). This is the problem of **demonstrating a hard curve** [10].
2. Given a prime p, to generate **uniformly random** supersingular curves E over $\mathbb{F}_{p^2}$ without revealing anything about the endomorphism ring or providing any information to help solve the isogeny problem to other supersingular curves over $\mathbb{F}_{p^2}$.
3. **Defining a hash function to the entire supersingular graph**. To produce a hash function taking arbitrary strings as input, and outputting supersingular j-invariants. The hard problems in this context include both pre-image finding and collision-finding for the hash function, and path finding and endomorphism ring computation for the output curve. We ask for these problems to remain hard on curves produced by the hash function.

There are also variants of these problems that involve sampling from (resp. mapping to) subsets of the set of supersingular curves. The most significant is **defining a hash function just to the $\mathbb{F}_p$ subgraph**.

The two obvious approaches to these problems are to use tools from the theory of complex multiplication and/or random walks. However neither method is secure for our problems. The insecurity of methods based on random walks is self-evident. The insecurity of methods based on CM is less clear, and was demonstrated by Castryck, Panny and Vercauteren [18] and Love and Boneh [10]. We refer to Section 2 for details.

Castryck–Panny–Vercauteren [18] and Wesolowski [66] have considered the analogous approach in the special case of sampling supersingular curves with j-invariant in $\mathbb{F}_p$ using CM theory. Again they show that any such approach is not secure (they show how to solve the class group action problem in subexponential and polynomial time respectively).

Hence we need new ideas. The goal of the paper is to explain some possible approaches and to discuss the obstructions to getting a practical solution.

In all cases we are interested in an efficient algorithm that takes input p, can be executed without any secret information, and that outputs (the j-invariant of) a supersingular elliptic curve over $\mathbb{F}_{p^2}$. We do not want the algorithm to provide any additional information that would be useful to the person who executes it. For the problem of generating a single hard curve (e.g., to bypass the requirement for trusted set up), the meaning of “efficient” might be relaxed, as long as it is feasible in applications.

As already mentioned, it would already be interesting to have an algorithm that returns a single curve. But the most desirable outcome is a **cryptographic hash function** $H(m)$ that takes a binary string m and returns a supersingular j-invariant and satisfies these properties:

1. It is efficient and deterministic.
2. It is hard to find a **collision**, namely two binary strings m_1 and m_2 such that $H(m_1) = H(m_2)$.
3. It is hard to **invert**, namely given an h in the codomain it is hard to compute a binary string m such that $H(m) = h$.
4. The j-invariants are uniformly distributed in the codomain.

Note that one can build an algorithm for hashing to the supersingular set by combining a standard cryptographic hash function H' (e.g., SHA-3) with a randomised algorithm to generate a supersingular curve (as in problem 2 listed above). To do this, simply compute $H'(m)$ and use it as the seed to a pseudorandom generator and then run the algorithm to generate a supersingular curve replacing all calls to randomness with this pseudorandom sequence. Hence, it suffices to focus on problems 1 and 2 above.

Several of the approaches in our paper try to bypass the problem of working with polynomials of exponentially-large degree. Section 3 sketches an approach motivated by iterated methods for root-finding (such as the Newton-Raphson method). However the main idea in this section is to avoid writing down the polynomial by indirectly computing its evaluation at a given point. This motivates a study of iterative methods in this special case. Similarly, Section 4 studies an approach based on modular curves and the fact that one can compute the roots in $\mathbb{F}_{p^2}$ of the greatest common divisor of two polynomials $F(x, x^p)$ and $G(x, x^p)$ in polynomial time in certain circumstances, even though the polynomials themselves have exponential degree. This approach does not lead to a useful solution at present, as the computation only produces curves that could feasibly have been computed using the CM method. Section 5 also attempts to control the growth of polynomials, by giving a system of low-degree polynomials whose common solution would give a desired curve.

Other methods try to use random walks in new ways. Section 6 suggests walking on the isogeny graph of abelian surfaces, until one lands on a reducible surface. The challenge faced by this method is that reducible surfaces are exponentially rare in the isogeny graph and we lack techniques to navigate to one from an arbitrary position in the graph. Finally, Section 7 suggests a way to use a quantum analog of the CGL hash to generate a random supersingular curve. The way a quantum algorithm uses randomness means this cannot be combined with a standard cryptographic hash function as described above. If properly implemented on a quantum computer, the algorithm makes the path information inaccessible to the user. But without a method to certify the use of the quantum algorithm, this approach only replaces the need for a trusted entity from one who will erase the path data to one who will promise to use a quantum computer.

Between release and revisions for this work, the concurrent work [53], which also proposes some approaches to the hashing problem, was made public. In particular, the papers appear to overlap in suggesting a system of equations based on torsion point restrictions (compare Section 5.1 and [53, Section 6.3]).

We hope the ideas and analysis in our paper will be useful to researchers. We identify a number of obstructions to efficient hashing to supersingular curves. We hope that future research might overcome one of these obstructions.

Acknowledgements

This project was initiated as part of the Banff International Research Station (BIRS) Workshop 21w5229, *Supersingular Isogeny Graphs in Cryptography.* The project owes a debt of gratitude to BIRS and to the organizers of that workshop: Victoria de Quehen, Kristin Lauter, Chloe Martindale, and Christophe Petit. The project was led by Steven Galbraith, Christophe Petit, Yan Bo Ti, and Katherine E. Stange. We would also like to thank Chloe Martindale for useful discussions, Annamaria Iezzi for her involvement in Section 4, as well as Wouter Castryck and Eyal Goren for contributing ideas to Section 6.

2 Existing methods

We briefly review the existing methods of generating supersingular curves. Neither is secure in the sense of the introduction. Nevertheless, these two paradigms form the basis of the methods proposed in this work, which fall broadly into methods based on random walks, and methods based on finding roots to high degree polynomials (or systems of such).

The Charles-Goren-Lauter hash function [19] hashes into the supersingular curves over $\mathbb{F}_{p^2}$. At each vertex of the supersingular isogeny graph, the out-directed edges are labelled in some fixed deterministic manner. Starting from a known curve such as $j = 1728$, the bitstring to be hashed is interpreted as directions for a walk through the graph, via the labelling just mentioned. If the walk is sufficiently long, it is known from the properties of the graph (it is Ramanujan) that the endpoint will be uniformly randomly chosen from amongst all the vertices of the graph [19]. However, the walk itself is a path to $j = 1728$ and therefore the path-finding problem from the endpoint is trivial, unless this information is discarded by a trusted authority.

The CM method of Bröker [11] finds supersingular roots of a Hilbert class polynomial. The Hilbert class polynomial $H_{\mathcal{O},p}$ for a quadratic order $\mathcal{O}$ modulo p is a polynomial in $\mathbb{F}_p[x]$ whose roots in $\overline{\mathbb{F}}_p$ are the j-invariants of elliptic curves whose endomorphism rings contain a copy of $\mathcal{O}$. In order to apply known root-finding algorithms, or indeed, to obtain the polynomial at all, the degree of $H_{\mathcal{O},p}$ must be small. This implies that $\mathcal{O}$ itself has non-integral elements of small norm. The images of such elements in the endomorphism ring are termed *small endomorphisms*, and so all the curves obtained have small endomorphisms. At the very least, it follows that the curves obtained are far from uniformly distributed. Furthermore, having a small endomorphism is known to be a serious vulnerability [10, 18]. Precisely, Castryck, Panny and Vercauteren [18] study the CSIDH case and show how to efficiently compute an ideal class as required to break CSIDH when given a small degree endomorphism. Love and Boneh [10] consider the more general case, such as arises in SIDH, and also show a general and efficient approach to computing isogenies between any two such curves in this setting.

3 Iterating to supersingular j-invariants

In this section, we propose a method for generating a hard curve. For a prime number $p > 2$, define the polynomial $H_p(t)$, known as the *Hasse polynomial* or *supersingular polynomial*, by

$$H_p(t) = \sum_{j=0}^{(p-1)/2} \binom{\frac{p-1}{2}}{j}^2 t^j. \tag{1}$$

Proposition 1. *Let E_λ denote the elliptic curve whose Legendre form is $y^2 = x(x-1)(x-\lambda)$. Then for $\lambda \in \mathbb{F}_p$*

$$\#E_\lambda(\mathbb{F}_p) \equiv p + 1 - H_p(\lambda) \pmod{p}.$$

Similarly for $\lambda \in \mathbb{F}_{p^2}$

$$\#E_\lambda(\mathbb{F}_{p^2}) \equiv p^2 + 1 - H_p(\lambda)^{p+1} \pmod{p}.$$

Proof. This follows from the proof of [63, Theorem V.4.1(b)]. □

Thus $\lambda \in \mathbb{F}_{p^2}$ is a root of $H_p(t)$ if and only if E_λ is a supersingular elliptic curve. It is known that all the roots belong to $\mathbb{F}_{p^2}$ and that, for $p \equiv 3 \pmod 4$, we have that $p^{1/2+o(1)}$ of them belong to $\mathbb{F}_p$. None belong to $\mathbb{F}_p$ when $p \equiv 1 \pmod 4$. This follows since the number of supersingular curves over $\mathbb{F}_p$ is $p^{1/2+o(1)}$ by combining [39] and [27, Eq (1)] and such a curve can be put in Legendre form if and only if all of its 2-torsion is rational, which is only possible when $p \equiv 3 \pmod 4$.

The basic idea is to compute a random root of the polynomial, thus giving a random supersingular elliptic curve. At first glance this seems impractical, as representing the polynomial $H_p(t)$ takes exponential space, and computing $H_p(t)$ would take exponential time. However, we can compute $H_p(\lambda)$ for $\lambda \in \mathbb{F}_p$ in polynomial time using Schoof's algorithm to compute $\#E_\lambda(\mathbb{F}_p)$. We can similarly compute $H_p(\lambda)^{p+1}$ by computing $\#E_\lambda(\mathbb{F}_{p^2})$. It is unclear whether there is a fast way to compute $H_p(\lambda)$ for $\lambda \in \mathbb{F}_{p^2}$.

3.1 Iterating to a root

One approach to finding a root of $H_p(t)$ is to iterate a polynomial function over a finite field as inspired by the Newton-Raphson method. Recall that the Newton-Raphson method finds a root of a polynomial $f(x)$ by first picking a point on the domain t_0 and iteratively computing

$$t_{n+1} = t_n - \frac{f(t_n)}{f'(t_n)}. \tag{2}$$

If a fixed point $t_{m+1} = t_m$ is found, then we can conclude that $f(t_m) = 0$ and that we have found a root.

In this vein, our "preliminary" idea is to find the roots using the same method. So one picks some $t_0 \in \mathbb{F}_p$ (or $\mathbb{F}_{p^2}$), and then defines

$$t_{n+1} = t_n - \frac{H_p(t_n)}{H_p'(t_n)}. \tag{3}$$

It is clear that if $t_{m+1} = t_m$, we must have that $H_p(t_m) = 0$, and we have found a supersingular elliptic curve. Furthermore, this method could allow us to define a hash function into supersingular curves, by using the hash input to determine t_0 and then iterating (3).

However, there are three issues with this idea:

1. The algorithm may not halt at a fixed point (the iteration may become stuck in a cycle).
2. The algorithm may reach a fixed point, but require too many iterations to efficiently compute.
3. We do not know how to compute $H_p'(t)$ efficiently, or compute $H_p(t)$ efficiently for $t \in \mathbb{F}_{p^2} \setminus \mathbb{F}_p$.

To eliminate the third obstacle, we can consider the following alternatives to the Newton-Raphson method which share the key property that fixed points of the iteration correspond to roots of $H_p(t)$:

$$t_{n+1} = t_n - H_p(t_n) \tag{4}$$

$$t_{n+1} = t_n - H_p(t_n)^{p+1}. \tag{5}$$

The denominator $H_p'(t)$ in the previous attempt speeds up convergence to a root in a field with a metric, if we are already close to a root. In a finite field with the discrete topology, this plays no role. Using Schoof's algorithm we may efficiently iterate (4) over $\mathbb{F}_p$ (which is only of interest when $H_p(t)$ has roots over $\mathbb{F}_p$, i.e. $p \equiv 3 \pmod 4$), while we may efficiently iterate (5) over $\mathbb{F}_{p^2}$.

The first two obstacles are thornier to tackle: there are plenty of choices of t_0 where iteration leads to a cycle and not a fixed point, and paths to a fixed point can be very long. These are fundamental obstructions which we will discuss experimentally and compare to the behavior of random mappings.

3.2 Does Iteration Mimic Iteration of a Random Function

In terms of understanding whether these iterative methods are useful for finding supersingular elliptic curves, the important quantities to understand for the iteration are:

1. the number of fixed points;
2. the number of points which eventually reach a fixed point upon iteration;
3. for these points, the maximum number of iterations needed to reach the fixed point; and
4. the number of points which reach a fixed point after k iterations.

We will be mainly interested in understanding to what extent these iterative methods look like iterating a random function (which we can understand theoretically).

Consider a random function from a set S of size n to itself. In other words, the image of each element of S is chosen independently and uniformly at random from S. The expected number of fixed points for a random function is one. Thus the iterative methods we are considering, which have many fixed points, do not appear random in this respect. However, they appear random in many other ways after taking this into account.

It appears experimentally that "many" points eventually reach a fixed point after iteration, which means that our iterative methods have a reasonable chance of finding a supersingular curve. However, the maximum number of iterations needed seems to be on the order of $\sqrt{n}$, which is too long to be practical. This is in line with the expected "tail length" of a random mapping [50, Theorem 8.4.8]. (Section 8.4 of *loc. cit.* contains a survey of the properties of random mappings.) Finally, the number of points which reach one of the m fixed points after k iterations appears to be on the order of $(k+1)m$, at least when k is small relative to n. This is supported by our analysis of random functions with many fixed points in Section 3.3.

We will now briefly discuss experimental results for different kinds of iterations. We have focused on iteration (4) over $\mathbb{F}_p$ when $p \equiv 3 \pmod 4$ as it is efficiently computable and well-motivated by analogy with the Newton-Raphson method. However, the behavior we are seeing does not seem sensitive to the exact iterative method used.

Example 1. When using the original Newton iteration (3), many points eventually reach a fixed point upon iteration. For example, when $p = 101$ the polynomial $H_p(t)$ has 50 roots all defined over $\mathbb{F}_{101^2}$. If one iterates using (3), 328 of the elements of $\mathbb{F}_{101^2}$ eventually end up at a fixed point (about three percent). In those cases it took at most 10 iterations to reach a fixed point. Similarly, when $p = 211$ around twenty eight percent (12747 out of 211^2 of the elements) eventually reach a fixed point. The maximum number of iterations needed was 90. When $p = 1009$, eight out of ten randomly chosen elements of $\mathbb{F}_{p^2}$ eventually reached a fixed point.

Example 2. The behavior when iterating using (4) over $\mathbb{F}_{p^2}$ is broadly similar; removing division by $H_p'(t_n)$ does not seem to have a significant effect. For example, if we look at all primes between

30 and 200 and compute the percentage of elements of $\mathbb{F}_{p^2}$ which eventually reach a fixed point, the minimum and maximum percentages are about 1.9 percent and 81 percent. The mean is about 30 percent. There are often quite long paths which eventually lead to a fixed point. When $p = 1009$, 8 of 10 randomly chosen elements of $\mathbb{F}_{1009^2}$ ended in a fixed point, and 6 out of 10 for $p = 10007$.

Example 3. Iterating using (4) over $\mathbb{F}_p$ is only interesting when there are fixed points defined over $\mathbb{F}_p$, i.e. when $p \equiv 3 \pmod 4$. It appears broadly similar to the previous iterations considered. The number of fixed points is $p^{1/2+o(1)}$. Experimentally it looks like a sizeable fraction of the points of $\mathbb{F}_p$ eventually reach a fixed point, and that for small k the number of points which reach a fixed point after k iterations is about $(k+1)$ times the number of fixed points (so on the order of $(k+1)\sqrt{p}$). The largest number of iterations needed to reach a fixed point appears to be on the order of $\sqrt{p}$.

To quantify this, we computed the minimum and maximum values for:

- the number of fixed points divided by $\sqrt{p}$, denoted $F_1(p)$;
- the number of elements of $\mathbb{F}_p$ iterating to a fixed point, divided by p, denoted $F_2(p)$;
- the largest number of iterations needed to reach a fixed point divided by $\sqrt{p}$, denoted $F_3(p)$.

Table 1 shows the minimum and maximum values of these values for primes in several ranges.

p in Range:	min $F_1(p)$	max $F_1(p)$	min $F_2(p)$	max $F_2(p)$	min $F_3(p)$	max $F_3(p)$
100 to 2000	.23	3.9	.019	.93	.034	.95
2000 to 3000	.29	4.0	.014	.61	.062	.64
20000 to 21000	.27	4.0	.0085	.46	.035	.47

Table 1. Statistics about iteration (4) over $\mathbb{F}_p$

Figure 1 shows a graph of the ratio of the number of elements of $\mathbb{F}_p$ which reach a fixed point after 5 iterations of (4) and of the number of fixed points, versus p. As expected, this appears to be around 6 but is somewhat noisy.

Example 4. Iterating using (5) over $\mathbb{F}_{p^2}$ preserves the cosets of $\mathbb{F}_p$ in $\mathbb{F}_{p^2}$. At first glance, it looks like for most cosets the map behaves like a random map: each orbit has very few fixed points and about $\sqrt{p}$ points in each orbit lead to the fixed points.

Based on this behavior, using iteration to efficiently find a supersingular elliptic curve (or to hash to a supersingular elliptic curve) does not seem to be practical. For concreteness, we will focus on iterating (4) over $\mathbb{F}_p$ when $p \equiv 3 \pmod 4$. The basic idea would be to pick a random starting element and iterate k times, hoping to find a fixed point. Thus the key property of the iteration to understand is the number of points which iterate to a fixed point in k steps. Assuming the iteration is a random function that has many fixed points, we expect on the order of $(k+1)\sqrt{p}$ points with this property (see Section 3.3).

The chance of randomly choosing to start at one of these is on the order of $(k+1)/\sqrt{p}$. The iteration requires k evaluations of $H_p(t)$, each of which can be done in polynomial time using Schoof's algorithm. For this to be efficient, the number of iterations k must be polynomial in $\log(p)$, so this approach is quite unlikely to find a fixed point efficiently.

Remark 1. This is no better than finding a supersingular curve (root of $H_p(t)$) by randomly guessing. The probability of a random curve being supersingular is on the order of $1/\sqrt{p}$. We can check

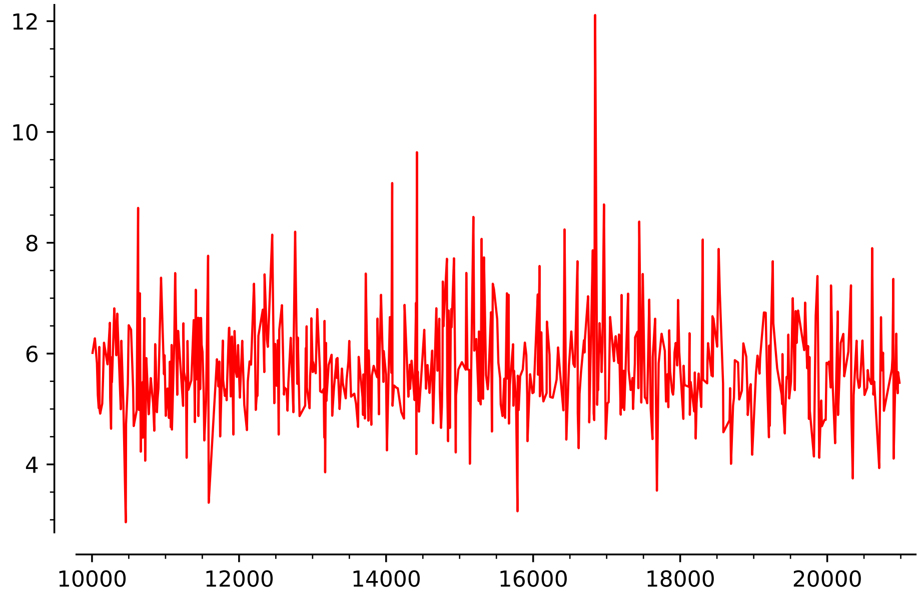

Fig. 1. Number of elements which reach a fixed point after 5 iterations of $x \mapsto x - H_p(x)$ divided by the number of fixed points, versus p

each guess by evaluating $H_p(t)$ (equivalently counting points using Schoof's algorithm). Checking k random curves would require k evaluations of $H_p(t)$, and would find a supersingular one with similar probability to the iterative method.

Remark 2. For the iterative method to offer an improvement, we would need a way to make a "giant step" and efficiently iterate multiple times at once. For example, given the nth iteration we would like to be able to efficiently compute the $2n$th iteration. We do not know if this is possible.

Remark 3. This iterative method should not produce supersingular curves uniformly at random. Taking $p = 1019$, when iterating (4) there are fixed points which no other elements of $\mathbb{F}_p$ reach upon iteration but there is also a fixed points that 50 other elements of $\mathbb{F}_p$ reach upon iteration.

3.3 Random functions with fixed points

We use the functional graph perspective on random mappings and the asymptotic analysis developed in Flajolet and Odlyzko [34] to analyze functions with many fixed points. A function on n elements with m fixed points is represented by:

- A functional graph, consisting of m rooted trees (one for each fixed point) plus a set of components without fixed points;
- Each component without fixed points is a collection of trees of length greater than 1 with the roots are permuted cyclically;
- Each rooted tree is a node (the root) together with a possibly empty set of rooted trees that are the children.

Note that all of these objects are labeled.

As in [34], there is a standard method to give relationships between exponential generating functions for these objects. Let $F_m(z)$ be the exponential generating function for random mappings with exactly m fixed points. This means that

$$F_m(z) = \sum_{n=0}^{\infty} F_{m,n} z^n / n!$$

where $F_{m,n}$ is the number of such functions with n total elements. Equivalently, it is the sum of $z^{|\varphi|}/|\varphi|!$ over all functions φ with m fixed points. Likewise let $C(z)$ and $T(z)$ be the exponential generating functions for components and trees, and let $C_{fpf}(z)$ be the exponential generating function for fixed-point-free components.

Lemma 1. *We have the following relationships:*

$$F_m(z) = T(z)^m \cdot \exp(C_{fpf}(z)) \tag{6}$$

$$C_{fpf}(z) = -\log(1 - T(z)) - T(z) \tag{7}$$

$$T(z) = z \exp(T(z)). \tag{8}$$

Proof. The first is a consequence of the fact that a function with m fixed points consists of m rooted trees plus a set of components with no fixed points. It is standard that

$$C(z) = \sum_{k \geq 1} \frac{1}{k} T(z)^k = \log(1/(1 - T(z)))$$

as a connected component based on a cycle of length k is built out of k trees and one can cyclically permute them. Therefore we see that

$$C_{fpf}(z) = C(z) - T(z) = \log(1/(1 - T(z))) - T(z).$$

The third is standard, a consequence of the fact that a tree is a node plus a set of trees.

We can use asymptotic analysis to compute the number of random functions with m fixed points. Flajolet and Odlyzko [34, Proposition 1] give an asymptotic expansion

$$T(z) = 1 - \sqrt{2}\sqrt{1 - ez} - 1/3(1 - ez) + O\left((1 - ez)^{3/2}\right)$$

of $T(z)$ around its singularity at $z = 1/e$. We can rewrite $F_m(z) = T(z)^m \exp(C_{\geq 2}(z))$ in terms of $T(z)$ as

$$F_m(z) = T(z)^m \frac{1}{1 - T(z)} \exp(-T(z)) = T(z)^{m-1} \frac{z}{1 - T(z)}$$

using that $T(z) = z \exp(T(z))$. We have that $z = \frac{1}{e} - \frac{1}{e}(1 - ez)$, so the leading term in the asymptotic expansion of $F_m(z)$ is

$$(e\sqrt{2}\sqrt{1 - ez})^{-1}.$$

Using [34, Theorem 1] gives the asymptotic

$$\frac{F_{m,n}}{n!} \sim \frac{e^{n-1}}{\sqrt{2\pi n}}. \tag{9}$$

For example, taking $m = 0$ gives the asymptotic $\frac{1}{e}\frac{1}{\sqrt{2\pi n}}e^n$ for $F_{0,n}/n!$. In comparison, Flajolet and Odlyzko's asymptotic analysis gave the known fact (letting F_n denote the number of functions on n elements) that $F_n/n! \sim \frac{1}{\sqrt{2\pi n}}e^n$. As expected, this implies that about $\frac{1}{e}$ of randomly chosen functions do not have a fixed point.

Remark 4. Note that if m is fixed as $n \to \infty$, the precise value of m has no effect on the asymptotics of $F_{m,n}$.

We will now modify our generating functions to take into account the number of elements which reach a fixed point after k iterations of a random function. The key case is for trees, where we consider the exponential generating function $T_k(z,u)$ where the coefficient of $z^n u^\ell$ is the number of rooted trees of size n with ℓ nodes that are distance at most k from the root, divided by $n!$.

Lemma 2. *We have that $T_0(z,u) = uT(z)$, that $T_k(z,u) = zu\exp(T_{k-1}(z,u))$, and that $T_k(z,1) = T(z)$.*

Proof. The first equality reflects that each tree has exactly one node at distance 0 from the root. The second comes from the fact that a rooted tree is a root plus a collection of child trees, and a node has distance at most k from the root if it either is the root or has distance at most $k-1$ from the root of one of the child trees. The third equality is clear.

We likewise modify $F_m(z)$ to become a bi-variate exponential generating function $F_{m,k}(z,u)$ which counts nodes with distance at most k to one of the m fixed points. It satisfies

$$F_{m,k}(z,u) = T_k(z,u)^m \exp(C_{fpf}(z)) = T_k(z,u)^m \frac{z}{(1-T(z))T(z)}.$$

Lemma 3. *The exponential generating function for the sum of the number of elements which reach a fixed point after k iterations for functions with m fixed points is*

$$\frac{dF_{m,k}(z,u)}{du}|_{u=1}.$$

Proof. This follows from viewing the bi-variate exponential generating function as a sum over all functions with m fixed points.

Proposition 2. *For fixed m and k, the number of elements which reach a fixed point after k iterations for a random function on n elements with m fixed points is asymptotically $(k+1)m$ as $n \to \infty$.*

Proof. We compute that $\frac{T_0(z,u)}{du}|_{u=1} = T(z)$ and

$$\begin{aligned}\frac{dT_k(z,u)}{du}|_{u=1} &= z\exp(T_{k-1}(z,1)) + z\exp(T_{k-1}(z,1))\frac{dT_{k-1}(z,u)}{du}|_{u=1}\\ &= T(z) + T(z)\frac{dT_{k-1}(z,u)}{du}|_{u=1}\\ &= T(z) + T(z)^2 + \ldots + T(z)^{k+1}\end{aligned}$$

with the last step following by induction. Thus we have that

$$\frac{dF_{m,k}(z,u)}{du}|_{u=1} = mT(z)^{m-1}(T(z) + T(z)^2 + \ldots + T(z)^{k+1})\frac{z}{(1-T(z))T(z)}.$$

The leading term in the asymptotic expansion at $z = 1/e$ is $(k+1)m(e\sqrt{2}\sqrt{1-ez})^{-1}$ again using [34, Proposition 1], so applying [34, Theorem 1] and comparing with (9) gives the result.

Remark 5. These results require that k and m be fixed as n grows. A more careful analysis should give that they continue to hold as long as k and m grow "slowly" compared to n, but we do not pursue this here.

In light of this analysis of functions with many fixed points, the iterative methods investigated in Section 3.2 behave exactly like random functions with the correct number of fixed points.

4 Modular polynomials and curves isogenous to their conjugates

4.1 Overview

As described in the introduction and Section 2, Bröker's method is limited by the degree of the Hilbert polynomials, upon which the runtime depends. However, taking small-degree Hilbert polynomials leads to curves with small endomorphisms (a vulnerability). In this section, we consider using polynomials whose roots correspond to curves with endomorphisms of exponentially large degree. The hope is, at least, to demonstrate a hard curve. The process we will describe does not, naïvely, appear likely to generate curves in a uniformly random manner, although perhaps it can be adapated.

If n is a positive integer coprime to p, then the classical modular polynomial $\Phi_n(x,y) \in \mathbb{Z}[x,y]$ is defined as follows. For any elliptic curve E, let $S_{E,n} = \{C \subseteq E[n] : C \text{ cyclic}, \#C = n\}$. There are $\psi(n)$ elements of $S_{E,n}$, where ψ is the Dedekind psi function (recall that $\psi(\ell^k) = (\ell+1)\ell^{k-1}$ for ℓ prime, and ψ is multiplicative; in particular, $\psi(n) > n$). Write E/C for the codomain of a separable n-isogeny from E with kernel C. Then

$$\Phi_n(j(E), y) = \prod_{C \in S_{E,n}} (y - j(E/C)).$$

In other words, $\Phi_n(x,y) = 0$ if and only if x and y are j-invariants related by a cyclic n-isogeny. This remains the case over any field. (See [48, Chapter 5] for background.)

Now, consider the roots of the univariate polynomial $\Phi_n(x, x^p)$. These roots are the j-invariants of curves with cyclic n-isogenies to their conjugates (with root multiplicities equal to the number of distinct n-isogeny kernels), and hence with an inseparable np-endomorphism (composing the n-isogeny with the inseparable p-powering isogeny from the conjugate back to the starting curve; see e.g. [21]). There is no particular reason why these curves should also have small-degree non-integer endomorphisms.

The collection of *supersingular* curves with an n-isogeny to the conjugate has been studied [2, 3, 21], and plays a role in the security of the path-finding problem [31]. In particular, the class group of $\mathbb{Q}(\sqrt{-np})$ acts on this set, and these curves form CSIDH-like graphs which could be used for cryptographic purposes [21]. Thus, a construction for random supersingular curves involving $\Phi_n(x, x^p)$ may lead to a means of sampling from these CSIDH-like graphs. As in the CSIDH setting, there are subexponential quantum algorithms to solve the *vectorization* or *class group action* problem (see [4, Section 9.1], [21] and [66]). Thus, if there is a curve of known endomorphism ring in this set (see for example [20]), one may be able to solve the fundamental isogeny problems (path-finding and endomorphism ring computation) in quantum subexponential time. This is still far from polynomial and may be considered secure for some applications.

For $p > n$, the polynomial $\Phi_n(x, x^p)$ has degree $\psi(n)p$, which is exponential with respect to $\log p$. While this polynomial is quite sparse, especially when $p \gg n$, we cannot compute its roots efficiently. The idea is to reduce that degree, and make computations manageable, by instead computing roots of the factor(s)

$$f_{n,m,p}(x) := \gcd(\Phi_n(x, x^p), \Phi_m(x, x^p))$$

for some auxiliary m, without explicitly computing $\Phi_n(x, x^p)$ or $\Phi_m(x, x^p)$.

The proposed approach for constructing supersingular curves is then:

1. Choose n and m.
2. Compute one or more roots of $f_{n,m,p}(x)$ in $\mathbb{F}_{p^2}$. There are $O(nm)$ of these roots, and we can compute them in polynomial time with respect to n, m, and $\log p$ (see §4.2).
3. Test each root to see if it is a supersingular j-invariant, using e.g. Sutherland's supersingularity test [65]; we give heuristics for this step in §4.3.

This method produces curves known to have endomorphisms of degree nm, np and mp. In fact, we know slightly more: the endomorphisms of degree np and mp have trace zero in the supersingular case (that is, they act like $\pm\sqrt{np}$ and $\pm\sqrt{mp}$, respectively) provided $n < p$ (see [19, Lemma 6]). Since we wish to avoid endomorphisms of small degree, the presence of the degree-mn endomorphism means that we should take at least one of n and m to be exponentially large. Nevertheless, it is plausible that the information about the endomorphism leaked from the process of construction is not enough to allow us to compute $\mathrm{End}(E)$ efficiently (i.e., in polynomial time).

4.2 Computing roots of $f_{n,m,p}$

We want to compute roots of $f_{n,m,p}(x) = \gcd(\Phi_n(x, x^p), \Phi_n(x, x^p))$ in $\mathbb{F}_{p^2}$. Note that simply computing $\Phi_m(x, x^p)$ and $\Phi_n(x, x^p)$ in $\mathbb{F}_p[x]$, computing their gcd, and finding its roots is exponential in $\log p$, because $\deg \Phi_m(x, x^p) > mp$ and $\Phi_n(x, x^p) > np$; these polynomials are sparse for large p, but generic gcd computations (which are quasilinear in the maximum of the degrees of the inputs [51]) cannot take advantage of this.

Algorithm 1 computes all of the $\mathbb{F}_{p^2}$-roots[14] of $f_{n,m,p}(x)$ in polynomial time with respect to m, n, and $\log p$. The key to its polynomial runtime in $\log p$ is that the polynomials F_m and F_n constructed in Lines 3 and 4 satisfy (by definition)

$$\Phi_m(j, j^p) = \Phi_n(j, j^p) = 0 \iff F_m(j_0, j_1) = F_n(j_0, j_1) = 0$$

for all $j = j_0 + j_1\sqrt{\delta}$ in $\mathbb{F}_{p^2} = \mathbb{F}_p(\sqrt{\delta})$, and it is much easier to solve the bivariate system $F_m(X_0, X_1) = F_n(X_0, X_1) = 0$ than it is to compute $\gcd(\Phi_m(x, x^p), \Phi_n(x, x^p))$ when p is large.

As has already been noted, for security in applications, at least one of n and m must be exponentially large. But if n (or m) is super-polynomially large with respect to $\log p$, then Algorithm 1 requires super-polynomial time and space, since it must work explicitly with the polynomials Φ_n. Hence a natural question is whether we can do better than Algorithm 1 when one (or both) of n and m is large. This is an open question. If m is small and n is very large then a "dream" approach would be to compute F_m using the classical algorithm and then somehow compute $\mathrm{Resultant}(F_m, F_n; X_0)$ directly by some form of "square-and-multiply" approach without explicitly computing F_n.

[14] Algorithm 1 ignores root multiplicities, but can be easily modified to take them into account if required.

Algorithm 1: Compute the set of roots of $f_{n,m,p}(x) = \gcd(\Phi_n(x, x^p), \Phi_n(x, x^p))$ in $\mathbb{F}_{p^2}$.

```
   Input: $m$, $n$, $p$
   Output: The set of roots of $f_{n,m,p}(x)$ in $\mathbb{F}_{p^2}$
 1 Compute $\Phi_m(X, Y)$ and $\Phi_n(X, Y)$ in $\mathbb{F}_p[X, Y]$      // Using e.g. the algorithm of [12]
 2 Compute a nonsquare $\delta$ in $\mathbb{F}_p$, and its square root $\sqrt{\delta}$ in $\mathbb{F}_{p^2}$
 3 $F_m \leftarrow \Phi_m(X_0 + \sqrt{\delta}X_1, X_0 - \sqrt{\delta}X_1)$ in $\mathbb{F}_p[X_0, X_1]$      // $F_m \in \mathbb{F}_p[X_0, X_1]$ because $\Phi_m$ is symmetric
 4 $F_n \leftarrow \Phi_n(X_0 + \sqrt{\delta}X_1, X_0 - \sqrt{\delta}X_1)$ in $\mathbb{F}_p[X_0, X_1]$      // $F_n \in \mathbb{F}_p[X_0, X_1]$ because $\Phi_n$ is symmetric
 5 $R \leftarrow$ Resultant($F_m$, $F_n$; $X_0$)      // Bivariate resultant $\mathrm{Res}_{X_0}(F_m, F_n)$ in $\mathbb{F}_p[X_1]$
 6 $\mathcal{J}_1 \leftarrow$ Roots($R$, $\mathbb{F}_p$)
 7 $\mathcal{S} \leftarrow \emptyset$
 8 for $j_1 \in \mathcal{J}_1$ do
 9     $G \leftarrow$ GCD($F_m(X_0, j_1)$, $F_n(X_0, j_1)$)
10     $\mathcal{J}_0 \leftarrow$ Roots($G$, $\mathbb{F}_p$)
11     $\mathcal{S} \leftarrow \mathcal{S} \cup \{j_0 + j_1\sqrt{\delta} : j_0 \in \mathcal{J}_0\}$
12 return $\mathcal{S}$
```

4.3 Supersingular roots of $f_{n,m,p}$

Now we consider the question of how many of the roots of $f_{n,m,p}(x)$ might be supersingular j-invariants. The individual polynomials $\Phi_n(x, x^p)$ should be expected to have overwhelmingly ordinary roots, but there are some heuristic reasons to expect $f_{n,m,p}(x)$ to have a higher proportion of supersingular roots and we give some evidence for this in Section 4.4.

A heuristic lower bound on the number of supersingular roots of $f_{n,m,p}(x)$ can be obtained as follows. Note that the resultant in line 5 of Algorithm 1 has degree $O(mn)$, so there are $O(mn)$ roots of $f_{n,m,p}(x)$ in $\mathbb{F}_{p^2}$ (to see this, apply Bézout's theorem to the polynomials F_m and F_n in Algorithm 1). There are $\approx p/12$ supersingular curves over $\mathbb{F}_{p^2}$, and $O(\sqrt{np})$ of them have an n-isogeny to their conjugate (see e.g. [21, §4.3]). Hence, we can think that a "random" supersingular curve has probability $\sqrt{12n/p}$ to have an n-isogeny to its conjugate. Applying this to the $O(\sqrt{mp})$ supersingular curves that are roots of $\Phi_n(x, x^p)$, we conclude that there should be $O(\sqrt{mn})$ supersingular roots of $f_{n,m,p}(x)$.

Hence, if we expect that the properties of having an n-isogeny and having an m-isogeny to the conjugate are in an appropriate sense "independent," then one might expect the supersingular portion of $f_{n,m,p}(x)$ to have size $O(\sqrt{nm})$. This would seem to be too large. Of course, the independence assumption cannot be expected to hold indiscriminately; perhaps there are choices of n and m for which this can be improved[15]. Given the degree estimate just described, one might consider taking the gcd of three different modular polynomials. This will almost certainly have a smaller degree: the same heuristic argument as above would lead to degree $O(\sqrt{nmr/p})$ for the gcd of $\Phi_n(x, x^p)$, $\Phi_m(x, x^p)$ and $\Phi_r(x, x^p)$. With such a degree, one might consider taking $n \sim m \sim \sqrt{p}$ and r polynomial in $\log p$. One might expect the 3-way gcd to have supersingular roots, provided it is not 1, by the same heuristics as above.

Remark 6. If E has an n-isogeny and an m-isogeny to its conjugate $E^{(p)}$, then it also has an nm-endomorphism to itself. When p is inert in $\mathbb{Q}(\sqrt{-nm})$, some such E will be reductions modulo

[15] We note that one might expect that since the supersingular roots are definitely in $\mathbb{F}_{p^2}$, and one might expect the ordinary roots to belong in general to larger fields, this would ensure that by restricting the search only to $\mathbb{F}_{p^2}$ roots, most of them will be supersingular. However, we have found experimentally that this is not the case.

p of curves over $\overline{\mathbb{Q}}$ with CM by $\mathbb{Z}[\sqrt{-nm}]$, specifically those where the reduction of the nm-endomorphism factors through the conjugate. In this case, we can expect a nontrivial gcd between $f_{n,m,p}(x)$ and the Hilbert class polynomial for $\mathbb{Q}(\sqrt{-nm})$.

Remark 7. If one desired uniformly randomly generated supersingular curves, one might consider using randomly generated n and m of a certain size. It is unclear what distribution of n and m would lead to a uniformly random distribution of supersingular curves, if any.

4.4 Experimental evidence

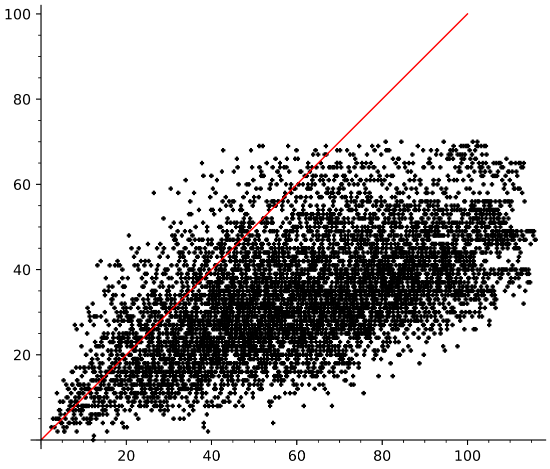

Fig. 2. Scatterplot of $f_{n,m,983}$: x-axis is $\sqrt{nm}$, y-axis is the number of $\mathbb{F}_{p^2}$ roots. The line $y = x$ is shown for reference. There are a total of 7046 data points.

data set	points	avg. $\frac{\text{supersingular roots}}{\mathbb{F}_{p^2}\text{ roots}}$	avg. $\frac{\mathbb{F}_{p^2}\text{ roots}}{\sqrt{nm}}$
all	7046	0.9362	0.6251
coprime	4286	0.9189	0.6555
not coprime	2760	0.9632	0.5779
$\left(\frac{-nm}{p}\right) = -1$	3572	0.9424	0.6373
$\left(\frac{-nm}{p}\right) = 1$	3474	0.9300	0.6126

Table 2. Statistics for various subsets of the data set for $p = 983$. The first row ('all') contains all the polynomials $f_{n,m,983}$ collected as described in the beginning of the section. The other rows give statistics for subsets of the data where n and m satisfy some criterion: the row 'coprime' (respectively 'not coprime') refers to those data points where $\gcd(n, m) = 1$ (respectively, $\gcd(n, m) \neq 1$), and the final two rows include points where n and m satisfy the indicated equality.

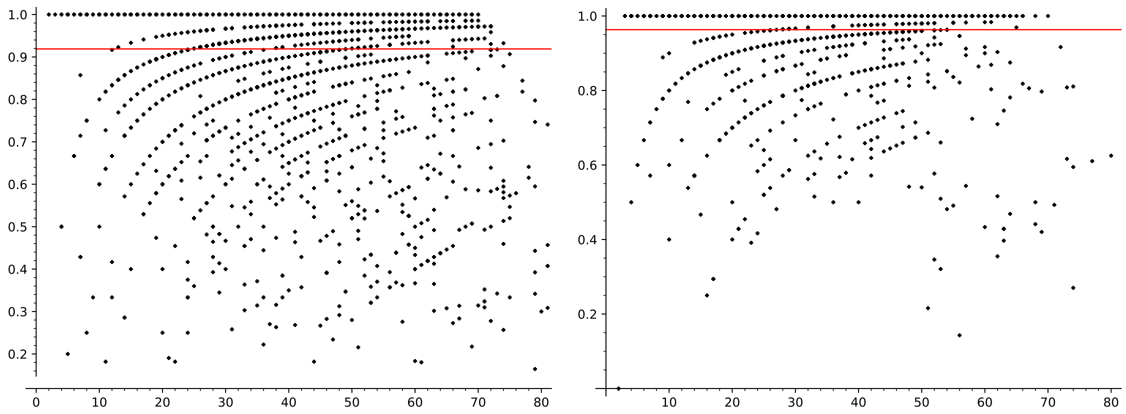

Fig. 3. Scatterplot of $f_{n,m,983}$: x-axis is degree, y-axis is the ratio of supersingular roots to all $\mathbb{F}_{p^2}$ roots. The visible hyperbolas correspond to the existence of 0, 1, 2 etc. non-supersingular roots. At left, 4286 pairs (n, m) which are coprime; at right, 2760 pairs which are not coprime. The red line indicates the average ratio across all pairs.

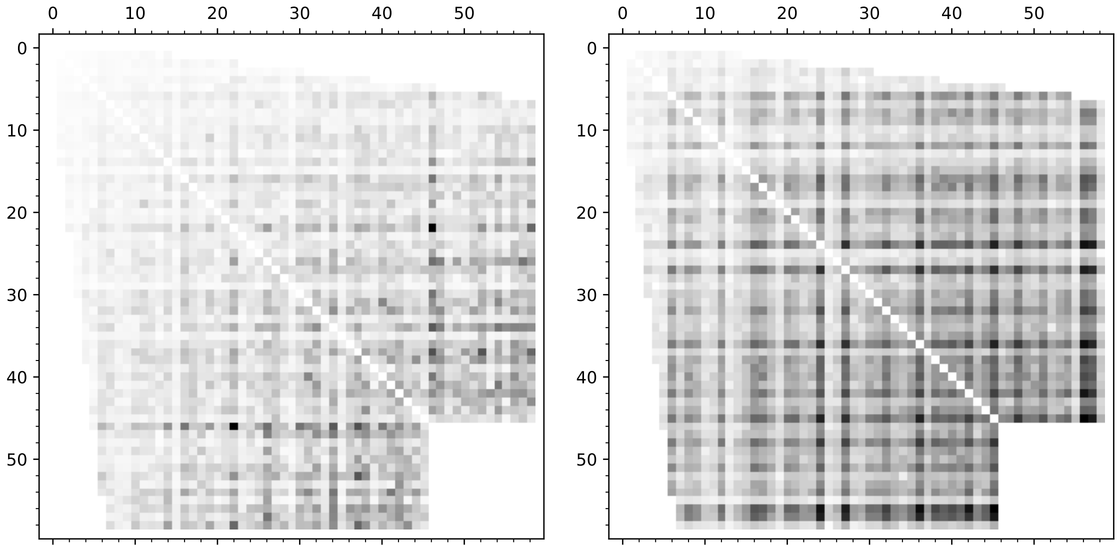

Fig. 4. Plot of the number of $\mathbb{F}_{p^2}$ roots (left) and supersingular $\mathbb{F}_{p^2}$ roots (right) of $f_{n,m,983}$ as a function of n and m (x and y axes). Dark = more (maximum = 238 $\mathbb{F}_{p^2}$ roots at left; 70 supersingular roots at right); light = fewer (minimum = 0); white = uncomputed.

To test the heuristics of the previous section, the polynomials $f_{n,m,p}(x)$ were computed for a fixed prime $p = 983$ with pairs (n, m) ranging over $2 \leq n \leq 45$, $n+1 \leq m < 8n$. Figure 2 shows the degree of the $\mathbb{F}_{p^2}$ part of the polynomial as compared with $\sqrt{nm}$. Figure 3 shows the proportion of supersingular roots. Table 2 gives the average values of these quantities for various subsets of the data, including where (n, m) is coprime or not, and where p is inert or split in $\mathbb{Q}(\sqrt{-nm})$. Figure 4 gives a sense of how the number of $\mathbb{F}_{p^2}$ roots of $f_{n,m,p}$ varies in an intricate manner as a function of n and m for fixed p.

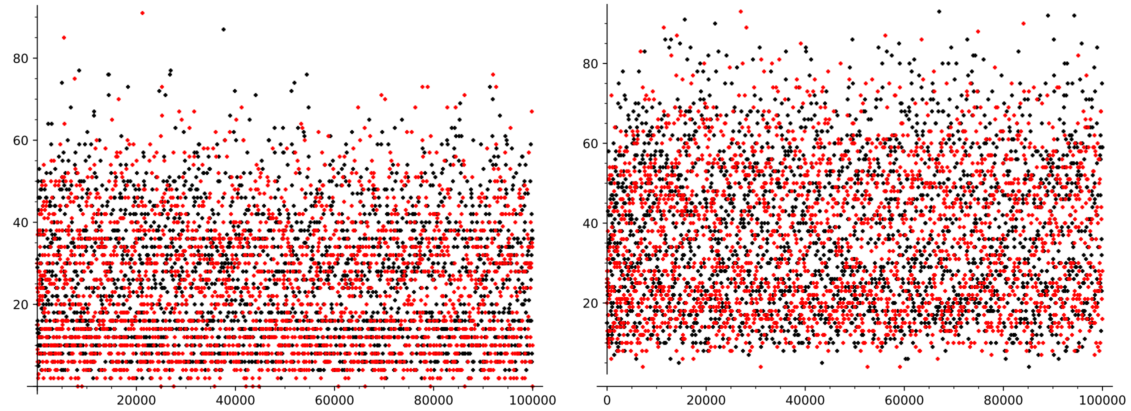

Fig. 5. Plots of deg $f_{n,m,p}$ as a function of p. At left, $n = 8$, $m = 12$. At right, $n = 8$, $m = 13$. When p is inert in $\mathbb{Q}(\sqrt{-nm})$, the plotted points are black. When p is split, the plotted points are red. In both plots, there are 4808 points in total, representing a random selection of primes which are $3 \pmod 4$ in the given range.

In addition, the polynomials $f_{n,m,p}(x)$ were computed for various fixed pairs (n, m) with p ranging over all primes less than 10^6 with $p \equiv 3 \pmod 4$. Figure 5 shows the degrees of $f_{8,12,p}$ and $f_{8,13,p}$ with respect to p.

In general, the data seems to support the following tentative patterns: (i) the degrees of the $f_{n,m,p}$ may be similar to or slightly less than predicted by heuristics, (ii) the proportion of supersingular roots among $\mathbb{F}_{p^2}$ roots is often high, (iii) there is variation in the ratio of supersingular roots with n and m, with slightly higher proportions found amongst n and m not coprime, and (iv) as p varies, the degree of $f_{n,m,p}$ is relatively constant, but is dependent upon the coprimality, not just size, of n and m.

5 Constructing supersingular curves using constraints on their torsion

A supersingular curve is characterized by the number of points over any extension. Provided a curve, Schoof's algorithm [62] computes the trace. When hashing into supersingular graphs, we know the trace and we want to find a curve. Thus, one may try to use Schoof's algorithm "backwards," by setting up a system of equations restricting the trace (or, more directly, the field of definition of torsion points), and looking for solutions. This method may lead to a way to generate supersingular curves uniformly randomly, since some such systems have all supersingular curves as roots.

5.1 A system of equations

To introduce the approach, let us first discuss the case when p is a prime of the form $p+1 = \prod_i \ell_i$, where ℓ_i are small distinct odd primes. For such p, the approach could proceed as follows. Let a be some parameter for the curve, like the j-invariant or the Montgomery coefficient. For every i, write $\Psi_{\ell_i}(x_{\ell_i}, a)$ for the division polynomial of order ℓ_i of the curve parameterized by a. These polynomials can be efficiently computed. Consider the system

$$\begin{cases} \Psi_{\ell_i}(x_{\ell_i}, a) = 0 & \forall\ \ell_i | p+1 \\ x_{\ell_i}^{p^2} - x_{\ell_i} \ = 0 & \forall\ \ell_i | p+1, \end{cases} \tag{10}$$

with variables x_{ℓ_i} and a. The equations of this system force the ℓ_i-torsion points of the curve with parameter a to be defined over $\mathbb{F}_{p^2}$ for all i. Therefore the $p+1$ torsion is also defined over $\mathbb{F}_{p^2}$, which implies that any curve with parameter a being a solution of (10) is supersingular. Taking the resultant of all polynomials in the system with respect to all variables but a gives a polynomial whose roots are all parameters a that correspond to supersingular curves.

More generally when $p+1$ is not smooth, one can fix a set of small primes or prime powers ℓ_i such that their product is above the Hasse bound, and replace the equations $x_{\ell_i}^{p^2} - x_{\ell_i} = 0$ in (10) by alternative equations forcing the endomorphisms

$$\pi^2 + [p-1]\pi + p^2$$

on the curve with parameter a to act trivially on the ℓ_i torsion.

For primes of the form $p+1 = f\ell_2^{e_2}\ell_3^{e_3}$ where f, ℓ_2, ℓ_3 are small integers (as used in the SIDH key exchange [42]) one can replace a single equation $\Psi_{\ell_i}(x_{\ell_i}, a) = 0$ in (10) by a polynomial system in the variables x_{ij} and a

$$\begin{cases} \Psi_{\ell_i}(x_{i1}, a) & = 0 \\ [\ell_i]_a(x_{i(j+1)}, -) = (x_{ij}, -) & \text{for } 1 \leq j \leq e_i - 1, \end{cases} \tag{11}$$

where $[\ell_i]_a$ are "x-only" multiplication-by-ℓ polynomials on the curve of parameter a. For any solution to this system, x_{ij} is the x-coordinate of a point $(x_{ij}, -)$ of order ℓ_i^j on the curve with parameter a. Note that the equations $[\ell_i]_a(x_{i(j+1)}, -) = (x_{ij}, -)$ are of degree roughly ℓ_i^2 and $\Psi_{\ell_i}(x_{i1}, a)$ is of degree $(\ell_i^2 - 1)/2$.

As with other approaches involving large polynomial systems or large degree equations, the cost and optimal strategy to solve these systems are not obvious. We observe that the polynomial system (11) contains equations in $e_1 + e_2 + 1$ variables of degree roughly ℓ_i^2 and ℓ_i together with the equation translating the fact that the torsion points lie in $\mathbb{F}_{p^2}$ of degree p^2. Yet, compared to generic polynomial systems of the same degree and with an equal number of variables, the given polynomial systems have only a few mixed monomial terms. Further, they exhibit a certain block structure. Instead of using generic algorithms such as Gröbner basis computations, taking the full monomial structure into account might help to solve the polynomial systems faster. This might be feasible using algorithms such as Rojas' algorithm for sparse polynomial systems [60]. However, further research is needed to draw conclusions about the concrete speedup that can be achieved using this additional structure and to assess the cost of solving the polynomial systems given in this section.

Unlike the Hasse polynomial of Section 3 and arguably the function $f_{n,m,p}$ of Section 4, the polynomial system to be solved in this section can certainly be stored in polynomial space.

5.2 Variants

Reducing the number of solutions: Instead of computing a random solution to the polynomial systems described in the previous section and thus a random curve with the correct number of points, some applications require computing only one curve with unknown endomorphism ring. To achieve this, one could add additional equations to the systems (11) to reduce the number of expected solutions – potentially all the way to 1, when solving the system would mean to select a single curve.

One approach could be to restrict the x-coordinate of torsion points to random cosets of multiplicative subgroups, namely replacing $x_{\ell_i}^{p^2} - x_{\ell_i} = 0$ for some i by

$$(\mu_i x_{\ell_i})^{r_i} - 1 = 0$$

for suitable r_i dividing $p^2 - 1$, and random μ_i in $\mathbb{F}_{p^2}$. This will decrease the degrees of equations in the system, as well as the number of solutions. If one does not restrict the field equations for all i, one may want to choose some i uniformly at random.

Assuming that the solutions to the system (11) are "randomly" distributed among all cosets of the multiplicative subgroup, the expected number of solutions to the system is reduced by the number of such cosets. If one of the remaining solutions is chosen uniformly at random and if the cosets for different i were chosen uniformly at random, then the supersingular elliptic curve corresponding to the final solution is a random supersingular elliptic curve. One could consider various versions of this, leaving more or fewer solutions.

Hybrid version: Another variant is to drop some equations in the polynomial system (11). The resulting system has then more solutions. Each solution to the resulting system leads to a curve with a number of points N with trace not fixed modulo the Hasse bound. That is, the curve generated might be of order N different from the order $(p+1)^2$ we would like to find. Hereby, the number of equations dropped from the system (11) controls the size of $\gcd(N, (p+1)^2)$. Thus, to compute a supersingular elliptic curve one may want to proceed as follows. One generates a system with fewer equations and keeps computing random solutions until the resulting curve has the correct order. We leave it for future research to examine how much easier it is to solve the resulting systems compared to (11).

6 Genus 2 Walks

In this section, we explore several approaches to sample a uniformly random supersingular elliptic curve based on the following general idea: start with a known supersingular elliptic curve $E_0/\mathbb{F}_q$, glue it to itself to construct a genus-2 Jacobian $A \cong \mathrm{Jac}(C)$ explicitly isogenous to E_0^2, and then connect A with a new random-looking elliptic product using Richelot isogenies, or through geometric inspection of the Jacobian (via its Kummer surface). The hope is that these genus-2 operations will "hide" obvious isogenies between the elliptic curves involved.

Let A be a principally polarised abelian surface (PPAS) over a finite field $\mathbb{F}_q$ of characteristic $p > 2$. The correct generalisation of the notion of supersingularity to genus 2 is to say that A is supersingular if and only if the Newton polygon of its Weil polynomial has all its slopes equal to $1/2$; this is the case if and only if the p-torsion $A[p]$ is isomorphic (as a BT_1 group scheme) to either $I_{2,1}$ or $I_{1,1} \oplus I_{1,1}$, where $I_{1,1}$ is the p-torsion group scheme of a supersingular elliptic curve (see Pries [56] for further detail). In the latter case, we say A is principally polarized *superspecial* abelian surface (PPSSAS).

Every PPAS A is isomorphic (as a principally polarized abelian variety) to either the Jacobian $\mathrm{Jac}(C)$ of some genus-2 curve C, or the product $E_1 \times E_2$ of two elliptic curves (which are both supersingular if A is superspecial). Oort [54] has shown that every superspecial abelian surface is isomorphic *as an unpolarized abelian variety* to a product of supersingular elliptic curves, and that every supersingular abelian surface is at least isogenous to a product of supersingular elliptic curves (if the abelian surface is supersingular but not superspecial, then the isogeny is inseparable).

We can construct a superspecial Jacobian A isogenous to a product of supersingular elliptic curves E_1 and E_2 by *gluing* them along their 2-torsion, say. This corresponds to a Richelot isogeny [58] $E_1 \times E_1 \to (E_1 \times E_2)/G \cong A$, where $G \leq (E_1 \times E_2)[2]$ is the graph of an isomorphism of group schemes $\psi : E_1[2] \to E_2[2]$ that is an anti-isometry with respect to the 2-Weil pairing (see [45]); the resulting A is always a Jacobian. (We can also glue along the ℓ-torsion for $\ell > 2$, and there is an analogous inseparable construction in [54] for gluing along the p-divisible group schemes $E_i[\mathrm{Fr}_p]$, but the case $\ell = 2$ is sufficient to illustrate our ideas. There is no reason to suspect that $\ell > 2$ or $\ell = p$ will give better results. The case $\ell = 2$ also has the advantage of being completely explicit.)

6.1 Random Walks

Our first idea is simple: We begin with a supersingular elliptic curve and glue it to itself which induces an isogeny to an abelian surface. We then take a random walk on the isogeny graph of abelian surfaces. Finally, we find the closest reducible surface and return one of its supersingular elliptic factors. The idea can be summarised in the following diagram:

$$\begin{tikzcd} E \times E \arrow[r, "\text{glue}"] & A \arrow[r, "\text{rand. walk}"] & A' \arrow[r, "\text{unglue}"] & E' \times E'' \end{tikzcd}$$

The initial A is superspecial, and so superspeciality is preserved so long as the isogenies in the random walk are of degree prime to the characteristic. This means that we are walking in the superspecial graph.

A similar situation occurs in [23], where the authors consider the supersingular isogeny problem in genus 2 and higher. We will only sketch the outline of their arguments and will refer interested readers to find details in their paper: In genus 2, given two superspecial abelian surfaces A and A', the idea is to reduce the problem of finding an isogeny $\phi : A \to A'$ to the problem of finding a factored isogeny $\psi : E_1 \times E_2 \to E_1' \times E_2'$ and (un)gluings $\pi : A \to E_1 \times E_2$ and $\pi' : E_1' \times E_2' \to A'$. Finding the isogenies π and $\widehat{\pi'}$ is essentially done by taking random walks of length $O(\log(p))$. Such a walk encounters a product of elliptic curves with probability $O(1/p)$, so after $O(p)$ many random walks we should have found the required π and $\widehat{\pi'}$. (The heuristics of [23] are made more rigorous in [36].)

Translating this to our setting, we see that random walks away from a fixed superspecial abelian surface have no better expected runtime at encountering a supersingular elliptic curve than simply searching for one directly by randomly sampling j-invariants and testing if they correspond to supersingular elliptic curves.

Ultimately, for this approach to give any advantage over simply taking a random walk in the elliptic supersingular graph, we need the genus-2 walk to "hide" information about the relative endomorphism rings of the starting and ending elliptic factors. But as noted in [41, Section 2], by fixing a supersingular elliptic curve over a finite field it is possible to parametrise the space of PPSSASs by positive-definite hermitian matrices which are elements of the matrix algebra $M_2(B_{p,\infty})$, where $B_{p,\infty}$ is the definite quaternion algebra that is ramified at p. Furthermore, isogenies between PPSSASs can be represented by conjugation by matrices in the same matrix algebra. Thus, knowledge of the random walk in the genus-2 graph may allow the construction of a matrix in $M_2(B_{p,\infty})$ that can be used to construct a path between our base and final supersingular elliptic curves.

Lastly, knowledge of the genus-2 walk may allow for the adversary to compute the endomorphism ring of the target surface, by computing the matrix that corresponds to the isogeny walk. The endomorphism ring of an elliptic product contains the endomorphism ring of each factor as a direct summand, so this information should allow an adversary to compute the endomorphism ring of the resulting (supersingular) elliptic curve.

6.2 Constructing curves on the Kummer surface

We saw above that random walks in the superspecial genus-2 graph give no real advantage over random walks in the elliptic supersingular graph when constructing new supersingular elliptic curves—and in addition, they may reveal information about the endomorphism ring. But we know that every superspecial abelian surface A is isomorphic to an elliptic product *as an unpolarised abelian variety*, so why not go looking for a new supersingular elliptic curve directly in A?

From a computational point of view, it is easier to work with curves on the *Kummer surface*, which is the quotient of A by the action of the involution $[-1]$. The projective embeddings of the Kummer surface $A/\langle\pm 1\rangle$ are easier to manage than those of the abelian surface A, since they involve fewer equations and lower-dimensional ambient spaces; but they also retain much of the information of A.

In this part, we consider the singular model K^{sing} in $\mathbb{P}^3$ of the Kummer surface of an abelian surface A. The model K^{sing} is defined by a single quartic equation (see e.g. [14, Eq. 3.1.8]). We write $\pi : A \to K^{\text{sing}} = A/\langle\pm 1\rangle$ for the degree-two quotient map; this map is ramified precisely at the sixteen 2-torsion points of A, and the images of these points under π are the singular points of K^{sing}, known as *nodes*. We denote the set of nodes by $S \subset K^{\text{sing}}$.

If $E \subset A$ is an elliptic curve, then the restriction of π to E defines a double cover of curves $\pi : E \to E' := \pi(E) \subset K^{\text{sing}}$. It follows from the Riemann–Hurwitz formula that E' is either an elliptic curve or a genus-0 curve; E' is an elliptic curve if and only if π is unramified along E; and E' is a genus-0 curve if and only if π is ramified at precisely 4 points.

This observation provides two ideas for constructing a new supersingular elliptic curve from a superspecial abelian surface A:

1. Find an elliptic curve on K^{sing} that does not go through any of the nodes of K^{sing}.
2. Find a genus-0 curve on K^{sing} that goes through precisely 4 of the nodes of K^{sing}.

For both approaches, we consider the intersection of K^{sing} with a hyperplane H.

Approach 1: For any hyperplane $H \subset \mathbb{P}^3$, the intersection $K^{\text{sing}} \cap H$ is a plane quartic curve C. If C is non-singular then it is a genus-3 curve. If on the other hand C is singular and has precisely two nodes then its (geometric) genus is $3-2=1$. Hence, it is possible to obtain such genus-1 curves by constructing hyperplanes that contain precisely two of the nodes of K^{sing}. Each pair of nodes determines a one-parameter family of hyperplanes passing through them, and imposing singularity of the intersection C at the nodes gives simple algebraic conditions on the parameter that let us choose "good" hyperplanes. (If required, one may define a birational map from C to an elliptic curve in Weierstrass form.) There is an important caveat here: even if C has genus 1, it may not be the image of an elliptic curve in A.

In our experiments, we took K^{sing} to be the Kummer surface of the Jacobian of the superspecial curve $y^2 = x^6 - x$ over $\mathbb{F}_{p^2}$ with $p \equiv 4 \pmod 5$. We note that this Jacobian is *not* Richelot-isogenous to any elliptic product (see e.g. [35, §4.15]), so we can be confident that any elliptic curves we find are not connected with some gluing along 2-torsion. Unfortunately, none of the elliptic curves we found using this approach were supersingular. We discuss reasons for this in §6.4 below.

Approach 2: This approach is doomed to fail: it is impossible to construct a hyperplane H passing through precisely 4 of the nodes of K^{sing}. Any three of the singular points in K^{sing} already define a hyperplane H, and it turns out that this hyperplane must pass through exactly 6 of the nodes. These hyperplanes, known as the *tropes* of the Kummer, are classical objects of study; there are sixteen of them, and the incidence structure formed by the intersections of tropes and nodes is a $(16, 6)$-configuration [40, §26].

If H is a trope, then it is tangent to K^{sing}. The intersection is a smooth conic, taken twice, and the preimage of this conic in A is isomorphic to the genus-2 curve generating A as a Jacobian; its Weierstrass points are the ramified points above the six nodes (see [14, §3.7] for further details, including the explicit recovery of the genus-2 curve). This curve may degenerate to a union of two elliptic curves joined at one point, but then A is an elliptic product itself, and these two elliptic curves are isomorphic to the factors—so we cannot obtain any new supersingular elliptic curves in this way.

6.3 Genus-5 curves on the desingularised Kummer

We can find more elliptic curves by computing the desingularization $\phi : K^{\mathrm{sm}} \to K^{\mathrm{sing}}$ of the Kummer surface, which yields a smooth model K^{sm} in $\mathbb{P}^5$ (see [14, Chapter 16] for more details). Concretely, let $Y : y^2 = \prod_{i=0}^{5}(x - a_i)$ be a hyperelliptic curve. Then $K^{\mathrm{sm}} = V(\Omega_0, \Omega_1, \Omega_2) \subset \mathbb{P}^5$, where

$$\Omega_0 : \sum_{i=0}^{5} X_i^2 = 0, \quad \Omega_1 : \sum_{i=0}^{5} a_i X_i^2 = 0, \quad \Omega_2 : \sum_{i=0}^{5} a_i^2 X_i^2 = 0\,,$$

is a smooth model of the Kummer surface of the Jacobian variety of Y (see Klein [46], and the survey articles by Dolgachev [28] and Edge [29]). As an intersection of three quadrics in $\mathbb{P}^4$, the intersection of K^{sm} with a hyperplane is a non-hyperelliptic genus-5 curve C. We first explain how to construct different elliptic curves that arise as quotients of the curve C, and later explore an alternative path where we choose hyperplanes in such a way that the curve C is singular and its irreducible components are elliptic curves.

Elliptic curves as quotients The intersection of the variety K^{sm} with a hyperplane defined by $X_i = 0$ for some $i \in \{0, \dots, 5\}$ yields a non-hyperelliptic genus-5 curve C_i. We are interested in certain elliptic curves $E_{i,j}$ with $j \in \{0, 1, 2, 3, 4, 5\} \setminus \{i\}$ that arise as quotients of the curve C_i. This situation is also studied by Stoll in [64]. The construction is depicted in Figure 6.

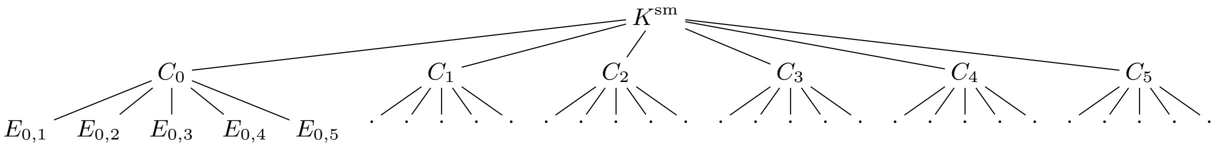

Fig. 6. Elliptic curves $E_{i,j}$ contained in the Kummer surface K^{sm}

Lemma 4. *Let $j \in \{0, 1, 2, 3, 4, 5\} \setminus \{i\}$ and consider the involution $\tau_j : X_j \mapsto -X_j$ in $\mathbb{P}^4$. Then*

$$E_{i,j} = C_i / \langle \tau_j \rangle$$

*is a genus-*1 *curve.*

Proof. The quotient map $\phi : C_i \to E_{i,j}$ has degree 2. It is ramified at $C_i \cap \{X_j = 0\} \subset \mathbb{P}^4$, a set of 8 points, each with ramification index 2. The Riemann–Hurwitz formula gives $2g(C_i) - 2 = 2 \cdot (2g(E_{i,j}) - 2) + 8 \cdot (2 - 1)$, whence $g(E_{i,j}) = 1$. □

We now show how to compute a Weierstrass equation for $E_{i,j} = C_i / \langle \tau_j \rangle$ by example of the genus-5 curve $C_5 = V(\Omega'_0, \Omega'_1, \Omega'_2) \subset \mathbb{P}^4$, where

$$\Omega'_0 : \sum_{i=0}^{4} X_i^2 = 0, \quad \Omega'_1 : \sum_{i=0}^{4} a_i X_i^2 = 0, \quad \Omega'_2 : \sum_{i=0}^{4} a_i^2 X_i^2 = 0.$$

Moreover, we assume that $j \in \{0, 1, 2\}$ since the other cases are obtained by permuting the variables.

First we simplify the equations defining C_5 using Gaussian elimination to obtain equations of the form

$$\begin{array}{llllll} \Omega''_0 : X_0^2 & & & +\lambda_{0,3}X_3^2 & +\lambda_{0,4}X_4^2 & = 0, \\ \Omega''_1 : & X_1^2 & & +\lambda_{1,3}X_3^2 & +\lambda_{1,4}X_4^2 & = 0, \\ \Omega''_2 : & & X_2^2 & +\lambda_{2,3}X_3^2 & +\lambda_{2,4}X_4^2 & = 0. \end{array}$$

The quotient $E_{5,j}$ for $j \in \{0, 1, 2\}$ is defined as the zero set of the two equations

$$\begin{array}{lllll} \Omega''_{j_1} : X_{j_1}^2 & & +\lambda_{j_1,3}X_3^2 & +\lambda_{j_1,4}X_4^2 & = 0, \\ \Omega''_{j_2} : & X_{j_2}^2 & +\lambda_{j_2,3}X_3^2 & +\lambda_{j_2,4}X_4^2 & = 0 \end{array}$$

in $\mathbb{P}^4$, where j_1, j_2 are such that $\{j_1, j_2, j\} = \{0, 1, 2\}$. This corresponds to the image of C_5 under the projection $\pi_j : \mathbb{P}^5 \to \mathbb{P}^4$ projecting away from X_j.

Note that $E_{5,j}$ is defined as the intersection of two quadrics in $\mathbb{P}^3$. To find a Weierstrass equation for this curve, let $P \in E_{5,j}$ be a rational point. First perform a coordinate transformation such that $P = (0 : 0 : 0 : 1)$ and then consider the projection $\mathbb{P}^3 \to \mathbb{P}^2$ projecting away from the last coordinate. The restriction of this map to $E_{5,j}$ is birational and in particular the image of $E_{5,j}$ is a curve in $\mathbb{P}^2$ defined by a cubic equation.

Singular hyperplane intersections In this part we consider singular curves that arise as the intersection of K^{sm} with a hyperplane defined as $L : \sum_{i=0}^{5} b_i X_i = 0$ for some coefficients $b_i \in k$. Such singular curves have geometric genus 5 and there are different configurations that can occur. Since our goal is to find an elliptic curve, we are interested in singular curves that consist of several components with at least one of these an elliptic curve. Here, we discuss the construction of singular curves that consist of two elliptic curves intersecting in 4 different points. This configuration is depicted in Figure 7.

Finding parameters b_i such that the intersection is singular can be solved efficiently using linear algebra. For that purpose, one considers the jacobian matrix M of the variety $C = K^{\mathrm{sm}} \cap L$. Let $M(P) \in M_{4,6}(k)$ denote the evaluation of M at a point $P = [x_0 : \cdots : x_5] \in C$. Then C is singular in P if and only if $\mathrm{rank}(M(P)) = 3$. Note that the last row of the matrix is given by the vector $b = (b_0, \ldots, b_5)$, hence the parameters must be chosen such that b is a linear combination of the first three rows of the matrix so that C is singular.

For most choices of b, the curve C will consist of only one irreducible component with precisely one singular point. As mentioned before, we intend to construct a curve C with two genus-1 components and 4 singular points. One possibility to achieve this is to choose b such that $b_i = b_j = 0$

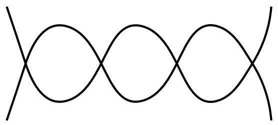

Fig. 7. Configuration of a singular genus-5 curve consisting of two elliptic curves.

for two indices $i \neq j$ in $\{0, \ldots, 5\}$. In that case, not only $\mathrm{rank}(M(P)) = 3$, but $M(P')$ has rank 3 for every $P' = [x'_0 : \cdots : x'_5]$ with $x'_k = x_k$ if $k \notin \{i, j\}$ and $x'_k \in \{\pm x_k\}$ otherwise.

We used this approach for different Kummer surfaces K^{sm} coming from a superspecial abelian variety. We obtained singular genus-5 curves C that consisted of two elliptic curves intersecting in 4 points. The configuration is depicted in Figure 7. However, none of the elliptic curves obtained in that way were supersingular.

6.4 Why do we only obtain ordinary elliptic curves?

In §6.2 and §6.3 we succeeded in constructing elliptic curves from the Kummer surfaces of superspecial abelian surfaces. However, these elliptic curves were not supersingular in most cases. At first glance this might contradict the intuition that we expect elliptic curves on superspecial abelian surfaces to be supersingular. To understand this situation, it is necessary to study the preimages of the constructed elliptic curves in the corresponding abelian surface.

Let us consider the second approach from §6.3, where we constructed elliptic curves in K^{sm}. If $E \subset K^{\mathrm{sm}}$ is an elliptic curve, then $\phi(E) \subset K^{\mathrm{sing}}$ is a (possibly singular) genus-1 curve. On the other hand $C = \pi^{-1}(E')$ has genus 1 if and only if the cover π is unramified along C. This means that E' must not go through any of the singular points $S \subset K^{\mathrm{sing}}$. The preimages of the sixteen nodes of K^{sing} are lines in K^{sm}; we write $L \subset K^{\mathrm{sm}}$ for this set of lines. Translating our condition on E' to K^{sm}, we see that E should not intersect with L. But using explicit descriptions of L (see e.g. [15, §2.2]), it is easy to see that there does not exist a hyperplane in $\mathbb{P}^5$ having trivial intersection with all of these lines. This shows that the elliptic curve E does not correspond to an elliptic curve in A.

A similar argument holds for the elliptic curves constructed in §6.2. The situation in the first approach of §6.3, where elliptic curves where constructed as quotients of genus-5 curves on K^{sm} is different. One can show that the Jacobian of the genus-5 curve C_i as above, is isogenous to $\prod_{j=1}^{5} E_{i,j}$. But it is not clear if there is a relation to $\mathrm{Jac}(Y)$. We leave this as an open question.

Question 1. What is the relation between the elliptic curves $E_{i,j}$ and the Jacobian $\mathrm{Jac}(Y)$ of the initial hyperelliptic curve?

Experimental results show that for each genus-2 curve, we find 15 isomorphism classes of elliptic curves $E_{i,j}$. In most cases, the elliptic curves are not supersingular. When starting with $Y : y^2 = x^6 - 1$, we obtain a mix of ordinary and supersingular elliptic curves. If it is supersingular, the j-invariant is 1728.

7 Quantum algorithm for sampling a hard curve

On a classical computer, the CGL hash function returns a random curve in the supersingular ℓ-isogeny graph. As described in the introduction, if one wishes the curve to be a "hard curve" then the drawback to this approach is the need for a trusted party who will throw away the path information generated by the hash function. Classically, the trusted party seems difficult to avoid. In this section, we explore the possibility of using a quantum computer to efficiently sample a hard curve from the isogeny graph without leaking any information about the endomorphism ring of the curve.

Although it may be possible to create a quantum algorithm that, when run on a quantum computer, makes the path information inaccessible, there is still a drawback. Given a curve E, we do not know if it was sampled using a classical computer (with an algorithm leaking information about $\mathrm{End}(E)$) or a quantum computer. Perhaps one can imagine a situation in which all parties inspect the quantum computer and agree it is a quantum computer, and run the program under observation. However, one may debate whether this situation differs appreciably from the situation in which all parties inspect a classical algorithm designed to delete the path information during its execution, and agree that it will delete it before it can be accessed. Perhaps one can hope for a means of making the quantum computation "auditable" in some way, but we do not have such a method here. In particular, even if this method samples a uniformly random supersingular curve, it cannot be turned into a hash function in the manner described in the introduction.

Leaving these concerns aside for now, we present below a novel mathematical approach to producing random supersingular curves. We use the idea of continuous-time quantum walks on isogeny graphs of supersingular elliptic curves in characteristic p. The idea was first proposed by Kane, Sharif and Silverberg [43, 44] for constructing public-key quantum money. In their scheme, quantum walks are carried out over the ideal class group of a quaternion algebra; we adapt these walks to isogeny graphs. The key observation we make here is that the distribution of the curves defined by our sampling algorithm coincides with the limiting distribution of the quantum walks on the graphs.

7.1 Quantum computing background

A qubit holds a quantum state that is a superposition (unit length $\mathbb{C}$-linear combination) of the two possible classical states of a bit, i.e. an element of complex norm 1 of $\mathbb{C}|0\rangle \oplus \mathbb{C}|1\rangle$. An n-qubit quantum register holds a quantum state that is a higher-dimensional analogue: an element $\sum_{x=0}^{2^n-1} \alpha_x |x\rangle$ of complex norm 1 in $\bigoplus_{0\leq x<2^n} \mathbb{C}|x\rangle$. Given any orthonormal basis $|y_i\rangle$ of the $\mathbb{C}$-vector space, we can rewrite the state in that basis: $\sum_i \beta_i |y_i\rangle$. Some of the power of quantum computers comes from the fact that superpositions of n qubits lie in an 2^n dimensional state space: the n-fold tensor product of the individual 2-dimensional state spaces (indeed $(\mathbb{C}|0\rangle \oplus \mathbb{C}|1\rangle)^{\otimes n} = \bigoplus_{0\leq x<2^n} \mathbb{C}|x\rangle$). Most of those states are *entangled*, meaning that they are not simple tensors in the bases for the individual qubits.

A quantum state $\sum_i \beta_i |y_i\rangle$ cannot be observed except by *measurement in an orthonormal basis* $|y_i\rangle$, a process which collapses the state to one of the basis elements $|y_i\rangle$, where state $|y_i\rangle$ is obtained with probability $|\beta_i|^2$ (the unit length condition implies a valid probability distribution). If there are several registers, we can measure just one, obtaining a superposition of the remaining registers. In a superposition $\sum_{x,y} \alpha_{xy} |x\rangle|y\rangle$, if we measure the first register, we obtain state $C \sum_y \alpha_{x_0 y} |x_0\rangle|y\rangle$ (where $C \in \mathbb{R}$ is chosen to scale to unit length) for some x_0, with probability $\sum_y |\alpha_{x_0 y}|^2$.

To get started on a quantum computer, one can initialize simple states such as uniform superpositions $\frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}|i\rangle$. A quantum computer then operates on quantum states by unitary operators. Among the most famous is the quantum Fourier transform, whose matrix is that of the inverse discrete Fourier transform. In particular, it operates by

$$\sum_{x=0}^{N-1}\alpha_x|x\rangle \mapsto \sum_{x=0}^{N-1}\left(\frac{1}{\sqrt{N}}\sum_{y=0}^{N-1}\alpha_y e^{2\pi iyx/N}\right)|x\rangle.$$

Classical algorithms can be performed in a quantum manner on one quantum register to store the output in another. In particular, for an efficiently computable function f we can perform the operation

$$\sum_x \alpha_x|x\rangle|0\rangle \mapsto \sum_x \alpha_x|x\rangle|f(x)\rangle.$$

7.2 Sampling curves on a quantum computer

A naïve approach. To mimic the CGL algorithm in superposition, we first generate the superposition

$$\frac{1}{\sqrt{N}}\sum_{x=1}^{N}|x\rangle,$$

where N is the number of supersingular curves. Then simultaneously for each x, we use the classical CGL algorithm to compute a curve E_x, at the end of the path associated to x, storing the result in a second register. The resulting superposition is

$$\frac{1}{\sqrt{N}}\sum_{x=1}^{N}|x\rangle|E_x\rangle.$$

Measuring this state collapses the superposition to a classical state $|x\rangle|E_x\rangle$ for some uniformly random $x \in \mathbb{Z}_N$. This is exactly the output of the CGL algorithm for a random input x, so the above procedure does not do anything more than the classical CGL. In particular, the path is stored in the first register. One way to avoid revealing the path is to apply the quantum Fourier transform to the first register and measure the result. The state we get is

$$\frac{1}{\sqrt{N}}\sum_{x=1}^{N}\omega_N^{xt}|E_x\rangle$$

for some uniformly random $t \in \mathbb{Z}_N$. Now, measuring this state produces a uniformly random curve E_x without revealing anything about the path x. However, this approach does not have any advantage over the classical CGL algorithm, as performing the quantum Fourier transform to "hide" the path information is analogous to including instructions to discard the path information in the classical CGL. In particular, if one measured the first register before the quantum Fourier transform is applied, one could recover the path information. Such runtime interference would not be detectable from the output state alone.

Continuous-time quantum walk algorithm. One way to model random walks on a graph is to apply the adjacency matrix as an operator on the real vector space generated by the vertices (a Markov process). Naïvely, one might hope to mimic this on a superposition of the vertices, but unfortunately, this matrix is not unitary. The substitute is the notion of a *quantum walk*, where the adjacency matrix is replaced by its exponential, which is unitary.

The adjacency matrix of the ℓ-isogeny graph is an $N \times N$ matrix T_ℓ called the Brandt matrix. Let us assume, for simplicity, that T_ℓ is symmetric.[16] Let S be the set of supersingular elliptic curves in characteristic p. The operator T_ℓ acts on the module

$$M = \bigoplus_{E \in S} \mathbb{Z}E.$$

In the quantum setting, we will work with the complex Euclidean space

$$\mathcal{X} = M \otimes_{\mathbb{Z}} \mathbb{C} = \bigoplus_{E \in S} \mathbb{C}|E\rangle.$$

Note that in order to implement this space on a quantum computer, we use a computational basis of j-invariants, so we will include ordinary curves also. However the random walk, if initiated with a supersingular curve, will restrict itself to the subspace $\mathcal{X}$ generated by the set of supersingular curves.

Let $U_\ell = \exp(iT_\ell)$. The operator U_ℓ is unitary (since T_ℓ is hermitian) and its eigenvalues are $e^{i\lambda}$ for the eigenvalues λ of T_ℓ. The operator U_ℓ^t implements a continuous-time quantum walk at time t on the ℓ-isogeny graph. The application of this for us is that from this quantum walk we can obtain a certain probability distribution on supersingular elliptic curves, and the ability to draw from this distribution to produce a random supersingular elliptic curve (once again, according to this distribution). This is done in the following way: fix an initial supersingular curve E_0 and a bound $T > 0$, pick a time $t \in (0, T]$ uniformly at random, compute $U_\ell^t|E_0\rangle$ and measure in the basis $\{|E\rangle\}_{E \in S}$. The probability of measuring a curve $E \in S$ is then given by [22, Chapter 16]

$$p_{E_0 \to E}(T) = \frac{1}{T} \int_0^T |\langle E| e^{iT_\ell t} |E_0\rangle|^2 dt. \tag{12}$$

For this process to be useful, we must answer two questions about the distribution (12) on the vertices of the ℓ-isogeny graph: (i) How efficient is sampling from this distribution? and (ii) Do samples leak information about endomorphism rings?

We comment on the second question first: The question of information leakage requires that we understand the distribution (12) and the endomorphism rings of its outputs. However, given an initial curve E_0, this distribution seems difficult to analyse. In particular, it is not the same as the distribution of endpoints of a classical random walk on the ℓ-isogeny graph.

Regarding efficiency, for any prime $\ell \leq \mathrm{poly}(\log N)$, the operator T_ℓ is sparse in the sense that there are only $\ell + 1 = \mathrm{poly}(\log N)$ nonzero entries in each row or column. Therefore, T_ℓ is a good candidate for a Hamiltonian of continuous-time quantum walks; we can use standard Hamiltonian simulation techniques to implement the quantum walk operator U_ℓ^t. However, the running time of the best known simulation algorithm depends linearly on ℓt [7]. Therefore, these quantum walks can efficiently be performed only for time $t \leq \mathrm{poly}(\log N)$.

[16] This assumption is satisfied for a mild condition on the characteristic p.

Moving to a limiting distribution. To remedy these issues, we consider the limiting distribution of (12). Let $|\phi_j\rangle$, $j = 1, \ldots, N$ be a set of eigenvectors of T_ℓ and let λ_j be the corresponding eigenvalues. It can be shown that [22, Section 16.6]

$$\lim_{T\to\infty} p_{E_0\to E}(T) = \sum_{j=1}^{N} |\langle E_0|\phi_j\rangle\langle E|\phi_j\rangle|^2. \tag{13}$$

This limiting distribution is more tractable than (12), as it is stated in terms of the spectral theory of the graph. In practice, for the distribution (12) to be negligibly close to (13), the value $T/(\lambda_j - \lambda_k)$ must be large for any j, k. However, the eigenvalues of T_ℓ are all in the range $[-2\sqrt{\ell}, 2\sqrt{\ell}]$, so there are some eigenvalues that are exponentially close to each other. This means that for us to assume that we are sampling according to (13), we must select T to be exponentially large. But, as mentioned above, we can only implement the walk operator U_ℓ^t for polynomially large t. Therefore, if we wish to use this nicer distribution, we need a different sampling algorithm which is efficient for larger T.

There is a (heuristic) polynomial time algorithm for sampling according to the limiting distribution (13) using phase estimation. This algorithm is based on the crucial fact that the set of operators $\{T_\ell\}_{\ell \text{ prime}}$ have a simultaneous set of eigenstates, namely the $|\phi_j\rangle$, $j = 1, \ldots, N$ from above. Since $\{|\phi_j\rangle\}$ is a basis, we can write

$$|E_0\rangle = \sum_{j=1}^{N} \langle \phi_j|E_0\rangle|\phi_j\rangle.$$

Now let $\ell_1, \ell_2, \ldots, \ell_r$ be a set of primes of size $\operatorname{poly}(\log N)$. Quantum phase estimation is an algorithm to recover the phase (which contains the eigenvalue information) of a unitary operator U. Specifically, if $U|\phi_j\rangle = e^{i\lambda_j}|\phi_j\rangle$ for $j = 1, \ldots, N$, the algorithm recovers an approximation to λ_j. We will use phase estimation on the operator U_{ℓ_1} with the input state $|E_0\rangle$. Let $\lambda_{1,j}$ be the eigenvalue of T_{ℓ_1} corresponding to the eigenstate $|\phi_j\rangle$. Then, because of the relationship between the eigenvalues of T_ℓ and those of U_ℓ, after phase estimation we obtain the state

$$\sum_{j=1}^{N} \langle \phi_j|E_0\rangle|\phi_j\rangle|\tilde{\lambda}_{1,j}\rangle \tag{14}$$

where $|\lambda_{1,j} - \tilde{\lambda}_{1,j}| \le 1/\operatorname{poly}(\log N)$. Measuring the second register (which reveals a value $\tilde{\lambda}_{1,j}$) we obtain a state $|\psi_1\rangle$ that is a projection of the state (14) onto a smaller subspace $\mathcal{X}_1 \subset \mathcal{X}$. If we repeat this procedure but now with the operator U_{ℓ_2} and the input state $|\psi_1\rangle$, we get a new state $|\psi_2\rangle$ that is the projection of $|\psi_1\rangle$ onto a smaller subspace $\mathcal{X}_2 \subset \mathcal{X}_1$. If r is large enough, repeating this procedure for all the remaining T_{ℓ_i} we end up with some eigenstate $|\phi_j\rangle$ with probability $|\langle E_0|\phi_j\rangle|^2$; see [43, 44] for a detailed analysis of this claim. Now, if we measure $|\phi_j\rangle$ in the basis $\{|E\rangle\}_{E\in S}$, we obtain a curve E with probability $|\langle E|\phi_j\rangle|^2$. Therefore, E is a sample from the distribution (13).

Challenges. This proposed method still presents a few important questions. First, a theoretical analysis of the distribution (13) is needed. As the ℓ-isogeny graph is heuristically believed to behave as a random $(\ell+1)$-regular graph, one hopes this distribution will approach the uniform distribution over supersingular curves mod p. Second, the measurement process for phase estimation reveals a series $\tilde{\lambda}_{1,j}, \tilde{\lambda}_{2,j}, \ldots, \tilde{\lambda}_{r,j}$ of approximations to the eigenvalues $\lambda_{1,j}, \lambda_{2,j}, \ldots, \lambda_{r,j}$ of the eigenstate $|\phi_j\rangle$ under the operators $T_{\ell_1}, T_{\ell_2} \ldots, T_{\ell_r}$. It is unknown whether revealing this partial eigensystem reveals any information, for example about the likely endomorphism ring of the resulting curve.

Bibliography

[1] Navid Alamati, Luca De Feo, Hart Montgomery, and Sikhar Patranabis. Cryptographic group actions and applications. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020 Proceedings, Part II*, volume 12492 of *Lecture Notes in Computer Science*, pages 411–439. Springer, 2020.

[2] Sarah Arpin. Adding level structure to supersingular elliptic curve isogeny graphs. arXiv:2203.03531, 2022.

[3] Sarah Arpin, Catalina Camacho-Navarro, Kristin Lauter, Joelle Lim, Kristina Nelson, Travis Scholl, and Jana Sotáková. Adventures in supersingularland. *Experimental Mathematics*, 0(0):1–28, 2021.

[4] Sarah Arpin, Mingjie Chen, Kristin E. Lauter, Renate Scheidler, Katherine E. Stange, and Ha T. N. Tran. Orienteering with one endomorphism. *IACR Cryptol. ePrint Arch.*, page 098, 2022.

[5] Reza Azarderakhsh, David Jao, Brian Koziel, Jason T LeGrow, Vladimir Soukharev, and Oleg Taraskin. How not to create an isogeny-based pake. In *International Conference on Applied Cryptography and Network Security*, pages 169–186. Springer, 2020.

[6] Andrea Basso, Péter Kutas, Simon-Philipp Merz, Christophe Petit, and Antonio Sanso. Cryptanalysis of an oblivious PRF from supersingular isogenies. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 160–184. Springer, 2021.

[7] Dominic W. Berry, Andrew M. Childs, and Robin Kothari. Hamiltonian simulation with nearly optimal dependence on all parameters. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 792–809. IEEE, 2015.

[8] Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. In *ASIACRYPT (1)*, volume 11921 of *Lecture Notes in Computer Science*, pages 227–247. Springer, 2019.

[9] Dan Boneh, Dmitry Kogan, and Katharine Woo. Oblivious pseudorandom functions from isogenies. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 520–550. Springer, 2020.

[10] Dan Boneh and Jonathan Love. Supersingular curves with small noninteger endomorphisms. In *ANTS XIV—Proceedings of the Fourteenth Algorithmic Number Theory Symposium*, volume 4 of *Open Book Series*, pages 7–22. Mathematical Sciences Publishers, 2020.

[11] Reinier Bröker. Constructing supersingular elliptic curves. *J. Comb. Number Theory*, 1(3):269–273, 2009.

[12] Reinier Bröker, Kristin Lauter, and Andrew V. Sutherland. Modular polynomials via isogeny volcanoes. *Math. Comp.*, 81(278):1201–1231, 2012.

[13] Jeffrey Burdges and Luca De Feo. Delay encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 302–326. Springer, 2021.

[14] 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, Cambridge, 1996.

[15] Abel Castorena and Juan Bosco Frías-Medina. Geometric aspects on Humbert-Edge's curves of type 5, Kummer surfaces and hyperelliptic curves of genus 2. arXiv:2106.00813, 2021.

[16] Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH (preliminary version). Cryptology ePrint Archive, Paper 2022/975, 2022. https://eprint.iacr.org/2022/975.

[17] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: An efficient post-quantum commutative group action. In Thomas Peyrin and Steven D. Galbraith, editors, *ASIACRYPT 2018*, volume 11274 of *Lecture Notes in Computer Science*, pages 395–427. Springer, 2018.

[18] Wouter Castryck, Lorenz Panny, and Frederik Vercauteren. Rational isogenies from irrational endomorphisms. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020 Proceedings, Part II*, volume 12106 of *Lecture Notes in Computer Science*, pages 523–548. Springer, 2020.

[19] Denis X. Charles, Kristin E. Lauter, and Eyal Z. Goren. Cryptographic hash functions from expander graphs. *J. Cryptology*, 22(1):93–113, 2009.

[20] Mingjie Chen and Jiangwei Xue. On $\mathbb{F}_p$-roots of the Hilbert class polynomial modulo p. arXiv:2202.04317, 2022.

[21] Mathilde Chenu and Benjamin Smith. Higher-degree supersingular group actions. *Mathematical Cryptology*, 1(2):85–101, 2022.

[22] Andrew Childs. Lecture notes on quantum algorithms. https://www.cs.umd.edu/~amchilds/qa/.

[23] Craig Costello and Benjamin Smith. The supersingular isogeny problem in genus 2 and beyond. In *International Conference on Post-Quantum Cryptography, PQCrypto 2020*, pages 151–168. Springer, 2020.

[24] Jean-Marc Couveignes. Hard homogeneous spaces. Cryptology ePrint Archive, Paper 2006/291, 2006. https://eprint.iacr.org/2006/291.

[25] Luca De Feo, Simon Masson, Christophe Petit, and Antonio Sanso. Verifiable delay functions from supersingular isogenies and pairings. In *ASIACRYPT 2019*, pages 248–277. Springer, 2019.

[26] Thomas Decru, Lorenz Panny, and Frederik Vercauteren. Faster SeaSign signatures through improved rejection sampling. In *PQCrypto 2019*, volume 11505 of *Lecture Notes in Computer Science*, pages 271–285. Springer, 2019.

[27] Christina Delfs and Steven D. Galbraith. Computing isogenies between supersingular elliptic curves over $\mathbb{F}_p$. *Des. Codes Cryptogr.*, 78(2):425–440, 2016.

[28] Igor Dolgachev. Kummer surfaces: 200 years of study. *Notices of the American Mathematical Society*, 67(10), 2019.

[29] W. L. Edge. A new look at the Kummer surface. *Canadian Journal of Mathematics*, 19:952–967, 1967.

[30] Kirsten Eisenträger, Sean Hallgren, Kristin E. Lauter, Travis Morrison, and Christophe Petit. Supersingular isogeny graphs and endomorphism rings: Reductions and solutions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018 Proceedings, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 329–368. Springer, 2018.

[31] Kirsten Eisenträger, Sean Hallgren, Chris Leonardi, Travis Morrison, and Jennifer Park. Computing endomorphism rings of supersingular elliptic curves and connections to path-finding in isogeny graphs. In *ANTS XIV—Proceedings of the Fourteenth Algorithmic Number Theory Symposium*, volume 4 of *Open Book Series*, pages 215–232. Mathematical Sciences Publishers, 2020.

[32] Luca De Feo and Steven D. Galbraith. Seasign: Compact isogeny signatures from class group actions. In *EUROCRYPT (3)*, volume 11478 of *Lecture Notes in Computer Science*, pages 759–789. Springer, 2019.

[33] Luca De Feo, David Kohel, Antonin Leroux, Christophe Petit, and Benjamin Wesolowski. SQISign: Compact post-quantum signatures from quaternions and isogenies. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020*, volume 12491 of *Lecture Notes in Computer Science*, pages 64–93. Springer, 2020.

[34] Philippe Flajolet and Andrew M. Odlyzko. Random mapping statistics. In *Advances in cryptology—EUROCRYPT '89 (Houthalen, 1989)*, volume 434 of *Lecture Notes in Comput. Sci.*, pages 329–354. Springer, Berlin, 1990.

[35] Enric Florit and Benjamin Smith. An atlas of the Richelot isogeny graph. *RIMS Kôkyûroku Bessatsu*, 2022. To appear.

[36] Enric Florit and Benjamin Smith. Automorphisms and isogeny graphs of abelian varieties, with applications to the superspecial Richelot isogeny graph. In Samuele Anni, Valentijn Karemaker, and Elisa Lorenzo García, editors, *Arithmetic, geometry, cryptography and coding theory 2021*, Contemporary Mathematics. American Mathematical Society, 2022. To appear.

[37] Steven D. Galbraith, Christophe Petit, Barak Shani, and Yan Bo Ti. On the security of supersingular isogeny cryptosystems. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016 Proceedings, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 63–91, 2016.

[38] Steven D. Galbraith, Christophe Petit, and Javier Silva. Identification protocols and signature schemes based on supersingular isogeny problems. *Journal of Cryptology*, 33(1):130–175, 2020.

[39] Dorian M. Goldfeld. A simple proof of Siegel's theorem. *Proc. Nat. Acad. Sci. U.S.A.*, 71:1055, 1974.

[40] Ronald William Henry Turnbull Hudson. *Kummer's quartic surface.* Cambridge University Press, Cambridge, 1905.

[41] Tomoyoshi Ibukiyama, Toshiyuki Katsura, and Frans Oort. Supersingular curves of genus two and class numbers. *Compositio Mathematica*, 57(2):127–152, 1986.

[42] David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In *International Workshop on Post-Quantum Cryptography*, pages 19–34. Springer, 2011.

[43] Daniel M. Kane. Quantum money from modular forms. arXiv:1809.05925, 2018.

[44] Daniel M. Kane, Shahed Sharif, and Alice Silverberg. Quantum money from quaternion algebras. arXiv:2109.12643, 2021.

[45] Ernst Kani. The number of curves of genus two with elliptic differentials. *J. reine angew. Math.*, 1997(485):93–122, 1997.

[46] Felix Klein. Zur Theorie der Liniencomplexe des ersten und zweiten Grades. *Mathematische Annalen*, 2(2):198–226, 1870.

[47] Péter Kutas, Simon-Philipp Merz, Christophe Petit, and Charlotte Weitkämper. One-way functions and malleability oracles: Hidden shift attacks on isogeny-based protocols. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021 Proceedings, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 242–271. Springer, Springer, 2021.

[48] Serge Lang. *Elliptic functions*, volume 112 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, second edition, 1987. With an appendix by J. Tate.

[49] Luciano Maino and Chloe Martindale. An attack on SIDH with arbitrary starting curve. Cryptology ePrint Archive, Paper 2022/1026, 2022. https://eprint.iacr.org/2022/1026.

[50] Rodrigo Martins, Daniel Panario, and Claudio Qureshi. A survey on iterations of mappings over finite fields. In Kai-Uwe Schmidt and Arne Winterhof, editors, *Combinatorics and Finite Fields: Difference Sets, Polynomials, Pseudorandomness and Applications*, volume 23 of *Radon Ser. Comput. Appl. Math.*, pages 135–172. De Gruyter, 2019.

[51] Niels Möller. On Schönhage's algorithm and subquadratic integer GCD computation. *Math. Comp.*, 77(261):589–607, 2008.

[52] Tomoki Moriya, Katsuyuki Takashima, and Tsuyoshi Takagi. Group key exchange from csidh and its application to trusted setup in supersingular isogeny cryptosystems. In Zhe Liu and Moti Yung, editors, *Information Security and Cryptology*, pages 86–98, Cham, 2020. Springer International Publishing.

[53] Marzio Mula, Nadir Murru, and Federico Pintore. On random sampling of supersingular elliptic curves. Cryptology ePrint Archive, Paper 2022/528, 2022. https://eprint.iacr.org/2022/528.

[54] Frans Oort. Which abelian surfaces are products of elliptic curves? *Mathematische Annalen*, 214:35–48, 1975.

[55] Christophe Petit. Faster algorithms for isogeny problems using torsion point images. In *ASIACRYPT 2017 Proceedings, Part II*, pages 330–353, 2017.

[56] Rachel Pries. A short guide to p-torsion of abelian varieties in characteristic p. In *Computational arithmetic geometry*, volume 463 of *Contemporary Mathematics*, pages 121–129. American Mathematical Society, 2008.

[57] Victoria de Quehen, Péter Kutas, Chris Leonardi, Chloe Martindale, Lorenz Panny, Christophe Petit, and Katherine E. Stange. Improved torsion-point attacks on SIDH variants. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021 Proceedings, Part III*, volume 12827 of *Lecture Notes in Computer Science*, pages 432–470. Springer, Springer, 2021.

[58] Fried. Jul. Richelot. De transformatione integralium Abelianorum primi ordinis commentatio. *Journal für die reine und angewandte Mathematik*, 16:285–341, 1837.

[59] Damien Robert. Breaking SIDH in polynomial time. Cryptology ePrint Archive, Paper 2022/1038, 2022. https://eprint.iacr.org/2022/1038.

[60] J. Maurice Rojas. Solving degenerate sparse polynomial systems faster. *Journal of Symbolic Computation*, 28(1-2):155–186, 1999.

[61] Alexander Rostovtsev and Anton Stolbunov. Public-key cryptosystem based on isogenies. Cryptology ePrint Archive, Paper 2006/145, 2006. https://eprint.iacr.org/2006/145.

[62] René Schoof. Elliptic curves over finite fields and the computation of square roots mod p. *Mathematics of Computation*, 44(170):483–494, 1985.

[63] Joseph H. Silverman. *The arithmetic of elliptic curves*, volume 106 of *Graduate Texts in Mathematics*. Springer, Dordrecht, second edition, 2009.

[64] Michael Stoll. Diagonal genus 5 curves, elliptic curves over $\mathbb{Q}(t)$, and rational diophantine quintuples. *Acta Arithmetica*, 190(3):239–261, 2019.

[65] Andrew V. Sutherland. Identifying supersingular elliptic curves. *LMS Journal of Computation and Mathematics*, 15:317–325, 2012.

[66] Benjamin Wesolowski. Orientations and the supersingular endomorphism ring problem. *IACR Cryptol. ePrint Arch.*, page 1583, 2021.

[67] Benjamin Wesolowski. The supersingular isogeny path and endomorphism ring problems are equivalent. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1100–1111. IEEE, 2022.

[68] Youngho Yoo, Reza Azarderakhsh, Amir Jalali, David Jao, and Vladimir Soukharev. A post-quantum digital signature scheme based on supersingular isogenies. In *International Conference on Financial Cryptography and Data Security*, volume 10322 of *Lecture Notes in Computer Science*, pages 163–181. Springer, Springer, 2017.