Isogeny-based cryptography

Chloe Martindale, Technische Universiteit Eindhoven
Lorenz Panny, Technische Universiteit Eindhoven

chloemartindale@gmail.com
lorenz@yx7.cc


Introduction

Quantum computers threaten to break most of the cryptography we are currently using to secure critical computer systems such as the internet. A quantum computer is a machine which employs quantum-physical phenomena to perform computations in a way that's fundamentally different from a "normal", *classical*, computer. Whereas a classical computer is, at any point in time, in a fixed *state* — such as a bit string representing its memory contents — the state of a quantum computer can be a "mixture", a so-called *superposition*, of several states. Note that the internal state is *hidden*: The only way to get information about the state is to perform a *measurement*, which will return a single non-superimposed *classical* output, such as a bit string, that is *randomly distributed according to the internal state*, and the internal state gets replaced by the measurement outcome. For example, when measuring an equal superposition of the two-qubit states $|00\rangle$ and $|11\rangle$, the result would be one of the bit strings 00 or 11 with probability $1/2$ each. Now a *quantum algorithm* consists of applying a carefully crafted sequence of operations to the internal state of a quantum computer in order to amplify the desired piece of information in the superposition, followed by measurements to extract the result.

The extra computational power thanks to the ability to store and manipulate *superpositions* of states allows for more efficient algorithms to tackle some computational problems. Note that contrary to a common misconception, quantum computers are *not* known to provide massive speedups over classical computers for "many", or even "all", tasks; in fact, there is only a handful of problems where known quantum algorithms outperform the best currently known classical algorithms. Unfortunately, many of these problems are at the heart of today's cryptographic systems; we shall see an example of this in the next section.

To deal with this problem, researchers have come up with *post-quantum cryptography*, a set of proposals for solutions to the looming threat of quantum computers on the cryptography currently in use. It may seem that quantum computers effective enough to break real-world encryption are a long way off: Building a quantum computer with enough qubits, and keeping them stable for long enough to be useful, poses a set of incredibly difficult physics and engineering challenges. However, no matter whether one believes powerful quantum computers are five years off or thirty years off, there is a compelling argument for acting as early as possible: People are now sharing plenty of data via cryptographically secured channels that they intend to stay private forever — or at least for a very long time —, even when stored now and attacked later with a quantum computer. This includes online audio or video telephony, private messages sent through chat services such as WhatsApp, and other sensitive data such as medical or financial records. In the words of a recent report by the United States' National Academy of Sciences:

> Even if a quantum computer that can decrypt current cryptographic ciphers is more than a decade off, the hazard of such a machine is high enough — and the time frame for transitioning to a new security protocol is sufficiently long and uncertain — that prioritization of the development, standardization, and deployment of post-quantum cryptography is critical for minimizing the chance of a potential security and privacy disaster. [13]

Isogeny-based cryptography is a specific type of post-quantum cryptography that uses certain well-behaved maps between abelian varieties over finite fields (typically elliptic curves) as its core building block. Its main advantages are relatively small keys and its rich mathematical structure, which poses some extremely interesting questions to cryptographers and computer algebraists.

The Autumn 2018 issue of this *Rundbrief* contained an article on post-quantum cryptography giving an overview of the main families of proposed constructions. We refer interested readers there for a broader discussion of things happening in the field of post-quantum cryptography.

Classical cryptographic key exchange

An important building block of many cryptographic systems, including the ubiquitous TLS protocol used to secure communication with websites, is a secure *key exchange*. Imagine you and a friend want to be able to send each other messages in a (metaphorical) locked box over an insecure channel, so you both need the same key. Since it's often not practical to exchange keys in person with everyone you want to send messages to, you also need a way of agreeing on a shared secret key over an insecure channel, with nobody else besides you and your friend being able to figure out what that key is. The key can be anything, so long as it's the same for all the parties involved; typically it is encoded as a bit string.

There are several ways to do this, the most common being the *Diffie–Hellman key exchange*. The security of this method is based on the fact that, for a finite group G, if you share an element $g \in G$ and some randomly chosen power $g^a \in G$, it is in general very hard for a corrupt bystander to compute a. The traditional (but now mostly deprecated) instantiation consists of fixing a prime p and computing in the group $\mathbb{F}_p^*$, i.e., the element g is simply an integer and the power g^a is computed modulo p.

To use this operation for a key exchange, our two parties Alice and Bob first agree on a group G and an element $g \in G$. Alice then chooses a secret positive integer a, and Bob chooses a secret positive integer b. Together, they can then compute the value g^{ab} over a public channel as follows:

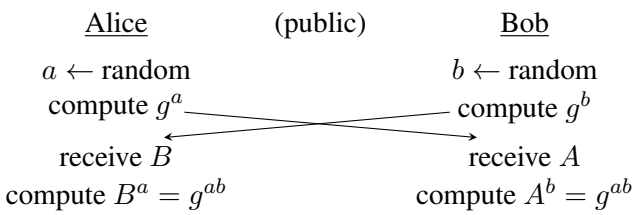

Figure 1: *The Diffie–Hellman key exchange.*

The obvious way to attack this scheme is to recover one of the secrets a and b from the publicly transmitted values g^a and g^b. This is known as the *discrete-logarithm problem*, which appears to be computationally hard for classical computers when the group G is well-chosen.

However, unfortunately, one thing that quantum computers are particularly good at is finding *periods* of computable functions using (variants of) an algorithm by Shor [11], which can be used to attack the discrete-logarithm problem as follows. Note that public values g^x are simply group elements: They can be multiplied together, and this satisfies the rule $g^x \cdot g^y = g^{x+y}$. This is exactly the operation that reduces breaking the scheme to finding a period: Given the element g and a public key $A = g^a$, we can define the group homomorphism

$$f\colon\ \mathbb{Z}^2 \to G,\ (x, y) \mapsto g^x \cdot A^y = g^{x+ay}\,.$$

Hence f is a periodic map whose period lattice is just its *kernel*, i.e., those pairs $(x, y) \in \mathbb{Z}^2$ where $g^{x+ay} = 1$. Shor's algorithm can, in polynomial time, find a basis of this lattice from an efficient description of the map f. We can then look for a vector of the form $(\tau, -1)$ in the lattice, which must equal $(a, -1)$ modulo the order of g, thus we have found a. The bottom line is that Diffie–Hellman is broken by quantum computers in all groups. Is this the end?

Luckily, in the aftermath of the discovery of Shor's algorithm, the traditional Diffie–Hellman framework has been extended to schemes which have similar traits, but do not rely on exponentiation maps in groups being one-way. One of these variants uses *isogeny graphs*.

Key exchange from graphs

Before we talk about isogeny graphs, let's first see how to get a shared value (key) from a more familiar graph. Here "graph" refers to a collection of *nodes* (dots) and *edges* (lines between them). For example, we can create a graph from a map of Manhattan by drawing a node at every junction and drawing an edge for every street. Then if Green and Red want to compute a shared value, they each choose a (secret) path from a common starting point, share the coordinates of their endpoints, then follow the same path from each other's endpoints to end up at the same final coordinates.

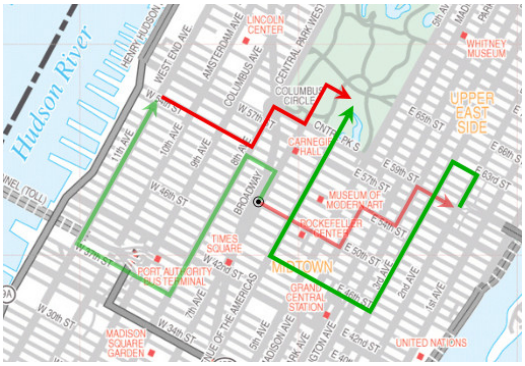

Figure 2: *Diffie–Hellman in Manhattan.*

However, this clearly isn't a secure way of exchanging keys — anyone can find a path from the common starting point to either Green or Red's end-of-path coordinates and thus compute the shared final coordinates, literally by adding and subtracting. To turn this approach into a secure key exchange, we need to replace our graph of Manhattan with a less structured graph, one in which finding a path between two given nodes is infeasible. On the other hand, the graph still needs to have *enough* structure to allow composing paths in a meaningful, commutative way, such that both parties end up at the same spot.

This is where isogenies comes in: They give rise to two families of graphs which are believed to have all the required properties for a (post-quantum) key exchange. Typical examples of these graphs look like this:

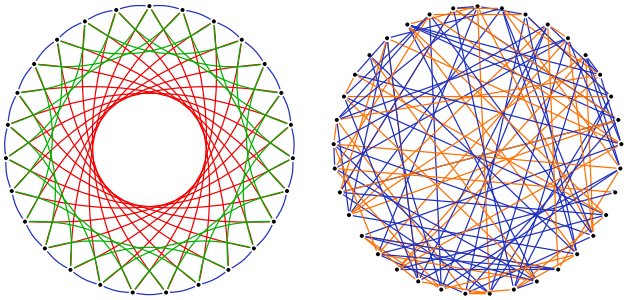

Figure 3: *Special isogeny graphs over a finite field.*

In both graphs, each node represents an *elliptic curve*, which can be represented as a certain kind of polynomial, and the edges represent maps between the elliptic curves called *isogenies*.

Elliptic curves and isogenies

We shall now explain some of the necessary background for understanding isogeny-based cryptography. Let $p \geq 5$ be a prime. An *elliptic curve over* $\mathbb{F}_{p^k}$ can then be defined as a smooth curve with equation $E\colon y^2 = f(x)$, where $f(x)$ is a degree-3 polynomial with coefficients in $\mathbb{F}_{p^k}$.

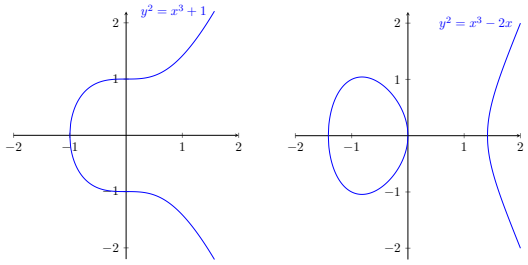

Figure 4: *Two elliptic curves over* $\mathbb{R}$.

"Smooth" means that the graph does not intersect itself or have any sharp points ("cusps"). These equations are especially interesting because the solutions that are defined over $\mathbb{F}_{p^k}$, together with one extra element, form an abelian group denoted $E(\mathbb{F}_{p^k})$. The extra element is referred to as the *point at infinity* ∞ and is defined to be lying on every vertical line that intersects the curve. The point ∞ is also the identity element of the group. The inverse of a point (a solution $P = (x_0, y_0)$ to the defining polynomial of E) is defined to be $-P = (x_0, -y_0)$. Notice that on a vertical line that intersects the elliptic curve in a point $P = (x_0, y_0)$, there are a total of three points in the group of solutions to the polynomial of E: the point P, its inverse $-P$, and the point at infinity ∞.

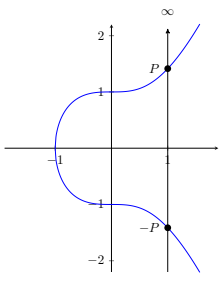

Figure 5: *Negating a point.*

In fact, this is no coincidence: any straight line that intersects the elliptic curve will intersect it exactly three times (when counting tangents as intersecting twice and also taking ∞ into account). We use this fact to define the group operation on $E(\mathbb{F}_{p^k})$: Given any two solutions $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$, draw the straight line passing through P and Q. This line will intersect the elliptic curve in exactly one more point R, and it turns out that the coordinates of this point will also be defined over $\mathbb{F}_{p^k}$. We then define a *group law*, written as $+$, by requiring that $P + Q + R = \infty$, the neutral element.

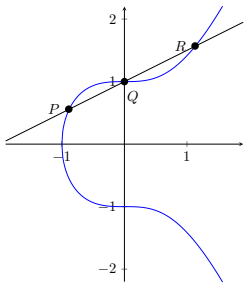

Figure 6: *Adding points.*

This group structure has led number theorists and geometers to study interesting properties of elliptic curves for centuries, and more recently elliptic curves have also enticed cryptographers, most importantly because one can base a very compact and efficient Diffie–Hellman key exchange on it. We need a little more though for a post-quantum scheme, since Shor's quantum algorithm to compute discrete logarithms of course also applies to this group.

Especially important in isogeny-based cryptography is a specific subclass of elliptic curves: *Supersingular* elliptic curves. An elliptic curve E defined over $\mathbb{F}_{p^k}$ is supersingular if $p \mid \left(p^k + 1 - \#E(\mathbb{F}_{p^k})\right)$. The most important special cases that come up are E defined over $\mathbb{F}_p$ with $\#E(\mathbb{F}_p) = p + 1$, and E defined over $\mathbb{F}_{p^2}$ with $\#E(\mathbb{F}_{p^2}) = (p + 1)^2$. In a nutshell, this is useful because it allows to easily enforce a special group order: Given any prime p and $k \in \{1, 2\}$, it is known how to generate a supersingular elliptic curve with $(p + 1)^k$ points over $\mathbb{F}_{p^k}$, hence we can control the group structure by choosing p in a special way. By contrast, it is not generally known how to efficiently find an elliptic curve with a given number of points, except in particularly nice special cases.

Recall that each node in the graphs we want to use for our post-quantum key exchange represents an elliptic curve. Since these graphs also have edges we need a way of passing from one node to another, which naturally will be a rational map that maps one curve to the other, and we will also want these maps to preserve the group structure of the elliptic curves. An *isogeny* is a non-zero map between elliptic curves that satisfies these things. More precisely, it is a surjective morphism of abelian varieties with finite kernel. The kernel subgroup is of utmost importance: In fact, one can prove that a (separable) isogeny is essentially uniquely defined by its kernel subgroup, and one can compute an isogeny from

its kernel in time linear in the size. Every isogeny has a *degree*, and typically (for separable isogenies) the degree is equal to the size of the kernel. Thus in a sense, the degree quantifies the algebraic and algorithmic complexity of an isogeny. However, since the secret keys in our cryptosystems are isogenies, we will use isogenies with "crypto-sized" (big) degrees! So how do we compute these isogenies quickly? The solution is to use an isogeny of very *smooth* degree, say $\deg \varphi = \ell^k$ for a small ℓ, and factor it into a composition of much smaller prime-degree maps:

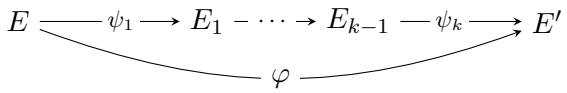

Figure 7: *Decomposing a smooth-degree isogeny.*

Note that the sequence $(\psi_1, ..., \psi_k)$ can be computed in $O(k \cdot \ell^2)$ field operations, whereas naïvely computing the entire map φ all at once would take time $\Theta(\deg \varphi) = \Theta(\ell^k)$: *exponentially* more.

The mathematics behind all of this is much richer than we can show in this short article. Interested readers are kindly referred to Luca De Feo's lecture notes [4].

Key exchange on isogeny graphs

In Figure 3 we saw two very different isogeny graphs that are used in cryptographic protocols. The two protocols built on these two types of graphs are called *CSIDH* [2] (pronounced "seaside", stands for "Commutative Supersingular-Isogeny Diffie–Hellman") and *SIDH* [8, 7] (pronounced as individual letters, stands for "Supersingular-Isogeny Diffie–Hellman").

CSIDH

We will show the CSIDH key exchange on a small (definitely not cryptographically-sized) example. We have seen how to perform an (insecure) key exchange on the graph on Manhattan — CSIDH is an implementation of the same idea on an *isogeny graph* with:

- Nodes given by *supersingular elliptic curves* E_A with equation $y^2 = x^3 + Ax^2 + x$ for $A \in \mathbb{F}_{419}$.
- Edges given by 3-, 5-, and 7-isogenies.

Almost as though by magic, this graph turns out to be very structured: Every node has exactly two outgoing edges of each colour, and the resulting cycles are compatible in the sense that a red step is always equivalent to the same number of blue steps, etc., independent of the position:

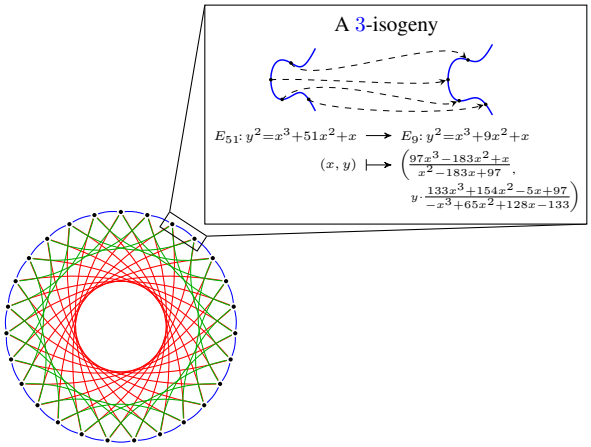

Figure 8: *An isogeny graph with a zoomed-in edge.*

Now if Alice and Bob want to compute a shared value in this graph, they each choose a (secret) path from a common starting point, share their endpoints, then follow the same path from the respective other party's endpoint to end up at the same final node — just like in the Manhattan example. We describe a path by a list of directions: one step clockwise (+) or anticlockwise (−) on the subgraph of a given colour.

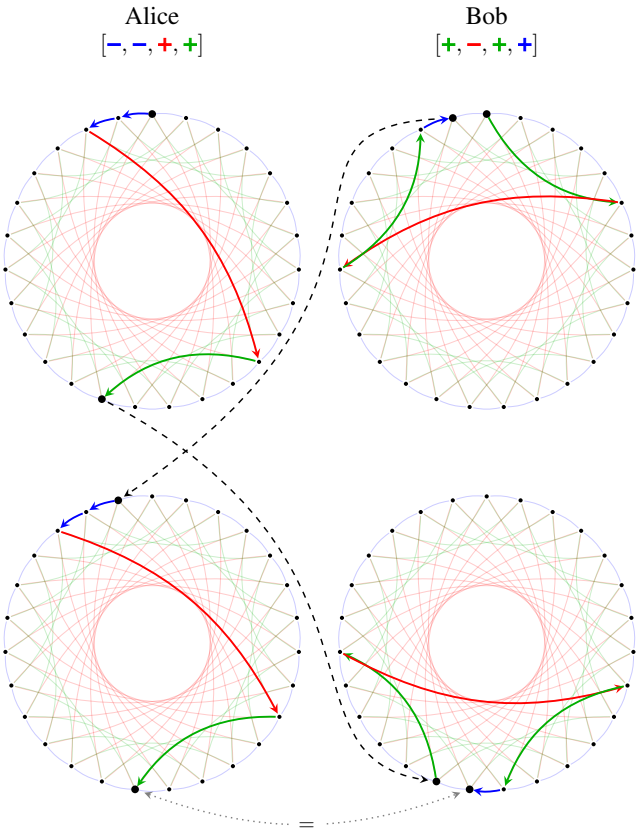

Figure 9: *Key exchange on a regular isogeny graph.*

On an unstructured graph, there is a priori no reason why Alice and Bob would end up at the same node after following this "graph key exchange" method. So what is it about the structure of *this* graph that makes this work?

[1] For number theory experts: a (separable) isogeny is (up to isomorphism) uniquely defined by its kernel, which corresponds to an ideal in the common $\mathbb{F}_p$-rational endomorphism ring $\mathbb{Z}[\sqrt{-p}]$ of every such elliptic curve. The *codomains* of such isogenies then only depend on the *class* of the corresponding ideal, hence the action of G can be considered an action of (a subgroup of) the class group $\mathrm{cl}(\mathbb{Z}[\sqrt{-p}])$.

Consider the set of clockwise and anticlockwise ℓ-isogenies as ℓ ranges over the non-negative integers, taken up to isomorphism. This set forms a *commutative group* G that *acts* on the set of supersingular elliptic curves $E_A\colon y^2 = x^3 + Ax^2 + x$ with $A \in \mathbb{F}_p$.[1] Write f_ℓ^+ and f_ℓ^- for clockwise and anticlockwise ℓ-isogenies respectively. In our key-exchange example above, Alice computes her curve E_A by computing the *action* of

$$f_A = f_7{}^+ \cdot f_5{}^+ \cdot f_3{}^- \cdot f_3{}^-$$

on E_0, written as $E_A := f_A * E_0$, and Bob computes his elliptic curve E_B by computing the action of

$$f_B = f_7{}^+ \cdot f_5{}^- \cdot f_7{}^+ \cdot f_3{}^+$$

on E_0, written as $E_B := f_B * E_0$. Then Alice and Bob send each other E_A and E_B and compute the actions $f_A * E_B$ and $f_B * E_A$ respectively. Now, as f_A and f_B are both elements in a *commutative* group,

$$\begin{aligned} f_A * E_B &= (f_A \cdot f_B) * E_0 \\ &= (f_B \cdot f_A) * E_0 = f_B * E_A. \end{aligned}$$

So Alice's endpoint $f_A * E_B$ and Bob's endpoint $f_B * E_A$ are the same!

Observe that, during this exchange, Alice never needs to communicate the isogeny group element f_A; this is her *private key*. With a much larger example, i.e., replacing 419 by a prime of several hundred (or even thousand) bits, and using a much longer list of ℓs, recovering this secret isogeny group element given just the start and end curve becomes infeasible for an attacker.

You may be thinking: doesn't Shor's algorithm apply to groups? Can I attack this commutative "isogeny group" with a quantum computer? The fundamental difference is that in traditional Diffie–Hellman, the public keys themselves are elements of the group, whereas in CSIDH, only the private keys are elements of a group — the public keys are elements of a set on which the group acts, and the operation $g^x \cdot g^y = g^{x+y}$ we've seen exploited earlier to apply Shor's algorithm to the discrete-logarithm problem simply does not exist. However, there is a quantum algorithm due to Kuperberg which attacks the action of a commutative group on the set of public keys to get a *subexponential* (but still super-polynomial) quantum attack [9, 10, 3]. In practice, this means that in order to be post-quantum secure, the parameters (like the prime p) have to be chosen larger than if the best attack was exponential; exactly how much larger is an ongoing research question.

SIDH

On the second graph in Figure 3, there is no evident group action — it looks just random. This rightfully suggests that Kuperberg's algorithm may not apply to finding a path on this graph, so the security level might scale better. However, it is not obvious how to even make a key-exchange protocol *work* on this graph: The extremely regular structure of the CSIDH graph aided in getting Alice and Bob's operations to commute, whereas in this case everything looks rather messy.

Happily, the graph does still carry enough structure, due to the fact that an isogeny is uniquely defined by its kernel. Alice and Bob (publicly) agree on a common starting curve $E/\mathbb{F}_{p^2}$ and choose secret subgroups A and B of $E(\mathbb{F}_{p^2})$. Writing φ_A and φ_B for the isogenies with those subgroups as kernel, the following diagram commutes (up to isomorphism) when $A' = \varphi_B(A)$ and $B' = \varphi_A(B)$, and therefore an isomorphism invariant of the curve $E/\langle A, B\rangle$ can be used as a shared secret:

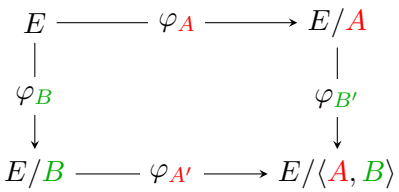

Figure 10: *High-level view of SIDH.*

Here, the horizontal arrows are computed by Alice and the vertical arrows are computed by Bob. Focussing on Alice, the only problem left is that she needs to somehow obtain $A' = \varphi_B(A)$, but Bob cannot give out φ_B since that's his secret.

The solution is that isogenies are also group homomorphisms on the corresponding groups of elliptic curve points: While Bob cannot give out φ_B, he *can* evaluate this map on publicly known points $P, Q \in E(\mathbb{F}_{p^2})$ and reveal $P' = \varphi_B(P)$ and $Q' = \varphi_B(Q)$. If Alice then chooses her subgroup A of the form $\langle P + [a]Q\rangle$ (that is, a cyclic group generated by $P + [a]Q$), she can simply compute A' as $\langle P' + [a]Q'\rangle$.

Similarly, Alice publishes images of known points under her own secret φ_A, allowing Bob to find B'.

This is the high-level mathematical overview of the protocol — of course there are many more interesting details in practice, for example one still has to ensure that the points P', Q' do not leak computationally useful[2] information about φ_B (similarly for φ_A), and choose the other parameters of the system in such a way that everything Alice and Bob need to compute is efficient in practice.

More advanced protocols

In this article we've only shown two simple key-exchange protocols using different kinds of isogeny graphs. However, the underlying mathematical ideas give rise to many other interesting cryptographic constructions, some of which seem impossible or harder to build without the use of isogenies.

For instance, one can build a *verifiable delay function* from isogenies [6]: A VDF is a random-looking function which is inherently slow to compute — independently of the algorithms used, the amount of

[2] We emphasize that while the action on a set of points often uniquely identifies an isogeny, it is not generally known how to *compute* the isogeny from that information.

hardware available, and parallelism — but on the other hand it is efficient for anyone to verify afterwards that the result is correct. (In the isogeny-based construction, the slow part consists of a long isogeny evaluation, and the verification part is a single elliptic-curve pairing.) This functionality may seem like not more than an odd curiosity, but in fact it has enough relevance in the distributed-systems world that blockchain projects are currently investing one hundred thousand US dollars [14] in the development of VDF technology (albeit founded on different mathematical ideas).

Another interesting example is the construction of *digital signatures* from isogenies: A recent paper [1] proposes a practical signature scheme CSI-FiSh (pronounced "seafish", stands for "Commutative Supersingular-Isogeny Fiat–Shamir") based on CSIDH's 512-bit parameter set. The main contribution is a massive precomputation effort in the form of a record-breaking *class-group computation*, which allows uniform sampling of isogeny walks — an important ingredient of the signature scheme. It is not known how to adapt this scheme to bigger (read: more secure) parameters: the effort for the class-group computation quickly grows too big for currently known techniques.

References

[1] Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. In *Cryptology ePrint Archive*, Report 2019/498. 2019. https://ia.cr/2019/498.

[2] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: an efficient post-quantum commutative group action. In *ASIACRYPT (3)*, volume 11274 of *LNCS*, pages 395–427. Springer, 2018. https://ia.cr/2018/383.

[3] Andrew M. Childs, David Jao, and Vladimir Soukharev. Constructing elliptic curve isogenies in quantum subexponential time. In *J. Mathematical Cryptology*, volume 8, pages 1–29. 2014. https://arxiv.org/abs/1012.4019v1.

[4] Luca De Feo. Mathematics of isogeny based cryptography. 2017. https://arxiv.org/abs/1711.04062.

[5] Luca De Feo and Steven D. Galbraith. SeaSign: Compact isogeny signatures from class group actions. In *EUROCRYPT (3)*, volume 11478 of *LNCS*, pages 759–789. Springer, 2019. https://ia.cr/2018/824.

[6] Luca De Feo, Simon Masson, Christophe Petit, and Antonio Sanso. Verifiable Delay Functions from supersingular isogenies and pairings. In *Cryptology ePrint Archive*, Report 2019/166. 2019. https://ia.cr/2019/166.

[7] David Jao, Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalali, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Joost Renes, Vladimir Soukharev, and David Urbanik. SIKE. Submission to [12]. http://sike.org.

[8] David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In *PQCrypto*, volume 7071 of *LNCS*, pages 19–34. Springer, 2011. https://ia.cr/2011/506.

[9] Greg Kuperberg. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM J. Comput.*, 35(1):170–188, 2005. https://arxiv.org/abs/quant-ph/0302112.

[10] Greg Kuperberg. Another subexponential-time quantum algorithm for the dihedral hidden subgroup problem. In *TQC*, volume 22 of *LIPIcs*, pages 20–34. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2013. https://arxiv.org/abs/1112.3333.

[11] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. In *SIAM Journal on Computing*, volume 26(5), pages 1484–1509. 1997.

[12] National Institute of Standards and Technology. Post-quantum cryptography standardization, December 2016. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization.

[13] National Academy of Sciences, Engineering, and Medicine. Quantum computing: Progress and prospects. ISBN: 978-0-309-47969-1. 2019. https://nap.edu/catalog/25196.

[14] VDF Alliance. FPGA design competition. See https://vdfalliance.org/contest.