DEFEND: Towards Verifiable Delay Functions from Endomorphism Rings

Knud Ahrens and Jens Zumbrägel
University of Passau, Germany
{knud.ahrens, jens.zumbraegel}@uni-passau.de

Abstract

We present a verifiable delay function based on isogenies of supersingular elliptic curves, using Deuring correspondence and computation of endomorphism rings for the delay. For each input x a verifiable delay function has a unique output y and takes a predefined time to evaluate, even with parallel computing. Additionally, it generates a proof π by which the output can efficiently be verified. In our approach the input is a path in the 2-isogeny graph and the output is the maximal order isomorphic to the endomorphism ring of the curve at the end of that path. This approach is presumably quantum-secure, does not require a trusted setup or special primes and the verification is independent from the delay. It works completely within the isogeny setting and the computation of the proof π causes no overhead. The efficient sampling of challenges however remains an open problem.

Keywords: Verifiable delay function, isogeny walks, modular polynomials, Deuring correspondence.

1 Introduction

A verifiable delay function (VDF) is a function which requires some specified time to be computed yet its output can be verified fast. Such functions have interesting applications such as resource-efficient blockchains or randomness beacons. Further use cases and more details can be found in [2]. In this work we propose a construction of a VDF using endomorphism rings of supersingular elliptic curves. Its security is based on problems in isogeny-based cryptography, thereby offering a scheme being presumably quantum-resistant.

As defined by [2] a VDF consists of three algorithms `Setup`, `Eval`, `Verify`. The `Setup` provides some public parameters based on the desired security λ and delay t. Then `Eval` takes the public parameters and a challenge $x \in X$ as input and produces a response $y \in Y$ with a proof π. Finally `Verify` checks if a presented response and proof are valid with respect to the challenge. While the algorithms `Setup`, `Verify` and the creation of challenges have to be efficient, `Eval` should require the desired time t. The VDF should also be *correct*, i.e. `Verify` accepts the output of `Eval` if the protocol is followed through as intended, *sound*, i.e. `Verify` will accept something that is not the output of `Eval` only with negligible probability, and *sequential*, i.e. a polynomially bounded adversary has only negligible chance to forge a correct response in time less than t, even with parallel computation.

The first approaches towards practical VDFs are based on group actions in groups of unknown order. They use for example repeated squaring and the proof contains intermediate steps [2, 16]. The downside of these approaches is that they are not quantum-secure. Isogeny-based cryptography is comparatively slow, but allows to build quantum-secure protocols, so it seems to be a natural candidate for VDFs. There are already several approaches to realise a VDF based on isogenies. De Feo, Masson, Petit and Sanso [8] use the evaluation of an isogeny of large degree at a given point and verify the result with pairings. This scheme needs a trusted setup, is not quantum-secure and its `Setup` takes the same time as its `Eval`. Shani [13] presents a hybrid VDF, but it is not quantum-secure and deviates from the original definition. In the paper by Chavez-Saab, Rodríguez-Henríquez and Tibouchi [5] a walk on the isogeny graph is defined via modular polynomials (similar to the hash function by Charles, Goren and Lauter [4]) and verified with SNARGs that are computed along the way. Here the running time of `Verify` depends on the delay and we need SNARGs in addition to the already involved topic of isogenies. Decru, Maino and Sanso propose a novel isogeny-based VDF [10], where verification uses Kani's criterion for abelian surfaces. It employs a purely algebraic construction, yet requires a trusted setup and its implementation poses some challenges.

In this work we use a similar `Setup` as in [5] but in addition to the starting curve an effective representation of its endomorphism ring is given. The modular polynomials are then used as a hash function to implicitly describe a path in the supersingular 2-isogeny graph. The input of the hash function and the final j-invariant are the input to `Eval`. The challenge is then to give a representation of the endomorphism ring of a supersingular elliptic curve with the given j-invariant. Finding the endomorphism ring is as hard as finding an isogeny from a curve with known endomorphism ring to that curve and both problems are considered hard [17]. This so-called endomorphism ring problem is also the foundation for CSIDH [3] and SQISign [9, 6]. Thus the easiest way to compute the endomorphism ring appears to push the generators of the endomorphism ring of the starting curve through the isogeny walk defined by the modular polynomials. Then `Verify` checks if the presented endomorphisms belong to a curve with the right j-invariant and generate its endomorphism ring.

This approach is presumably quantum-secure, does not need a trusted setup or special primes and the algorithms `Setup` and `Verify` are fast. It works completely within the isogeny setting and the time for verification is independent from the delay t. Also the computation of the proof π causes no overhead, since it is already part of computing the response y. An important issue however remains to find an efficient way to sample challenge inputs to `Eval` that include enough information for fast verification.

In the following we first give proper definitions for a VDF and a description of our idea on an abstract level. Then we provide the preliminaries of isogeny-based cryptography, including its quaternion aspects and some computational problems. With these we can present our VDF, discuss the three parts Setup, Evaluation and Verification and make a brief security analysis. Next we show how this can be implemented before finally giving a summary and outlook for future research.

2 Verifiable Delay Functions

Let us recall the original definition by Boneh, Bonneau, Bünz and Fisch [2]. We have three algorithms $\texttt{Setup}, \texttt{Eval}, \texttt{Verify}$ and want $\texttt{Verify}$ (and $\texttt{Setup}$) to be fast, while $\texttt{Eval}$ should be exponentially slower, or more formally:

$\texttt{Setup}(\lambda, t) \longrightarrow \mathbf{pp} = (\mathbf{ek}, \mathbf{vk})$ polynomial in λ (t is subexponential in λ)

$\texttt{Eval}(\mathbf{ek}, x) \longrightarrow (y, \pi)$ with parallel time t on $\mathrm{poly}(\lambda, \log t)$ processors

$\texttt{Verify}(\mathbf{vk}, x, y, \pi) \longrightarrow (\mathsf{valid}/\mathsf{invalid})$ polynomial in λ and $\log t$

Here λ is the security parameter and t is a delay parameter. The evaluation key $\mathbf{ek}$ and the verification key $\mathbf{vk}$ can be identical and the public parameters $\mathbf{pp}$ should define a challenge space X and a response space Y such that $x \in X$ and $y \in Y$. The proof π is allowed to be empty.

In addition, the algorithms have to satisfy the following properties: Honest outputs should be accepted, y is unique or at least hard to forge and y cannot be found significantly faster than with $\texttt{Eval}$.

Definition 2.1 (Correctness) *For all λ, t, x if $(\mathbf{ek}, \mathbf{vk}) \leftarrow \texttt{Setup}(\lambda, t)$ and $(y, \pi) \leftarrow \texttt{Eval}(\mathbf{ek}, x)$ then $\texttt{Verify}(\mathbf{vk}, x, y, \pi)$ outputs valid.*

Definition 2.2 (Soundness) *For all adversaries $\mathcal{A}(\lambda, t, \mathbf{pp}) \longrightarrow (x, y, \pi)$ polynomial in λ, t the probability to find a y that gets accepted by $\texttt{Verify}(\mathbf{vk}, x, y, \pi)$, but differs from the output of $\texttt{Eval}(\mathbf{ek}, x)$ under the condition that $\mathbf{pp}$ is honestly generated is a negligible function in λ.*

Definition 2.3 (Sequentiality) *For $\sigma(t) \lesssim t$ and $p(t)$ polynomial in t a VDF is called (σ, p)-sequential, if for all adversaries $\mathcal{A}_0(\lambda, t, \mathbf{pp}) \longrightarrow L$ polynomially bounded in λ, t (preprocessing $\mathbf{pp}$) and $\mathcal{A}_1(\mathbf{pp}, L, x) \longrightarrow y_A$ with parallel time $\sigma(t)$ on at most $p(t)$ processors the probability that y_A is equal to the output of $\texttt{Eval}(\mathbf{ek}, x)$ is a negligible function in λ.*

The name delay function is based on the idea that we have an actual function $f\colon X \to Y$ in the mathematical sense and $\texttt{Eval}$ has to evaluate f at x and present a proof π to verify the result $f(x) = y$. Since $\texttt{Eval}$ needs a challenge $x \in X$ as input, $\texttt{Setup}$ should also provide an efficient algorithm $\texttt{Sample}$ or $\texttt{Hash}$ to select a random element from X or to have a function $h\colon S \to X$ that maps some seed $s \in S$ to an element in X, respectively. This might depend on the application of the VDF.

3 Basic Idea

We start with an abstract high-level description of our approach that might also allow for different applications, and fill in the details in a later section.

Let G be a connected graph with vertex set V. We need a function $g\colon V \to Y$ that is hard to compute, i.e. given $v \in V$ it is difficult to obtain $g(v)$. However, the values $g(v)$ can be *transported* along a path in the graph (for a cost polynomial in its length) as follows. If γ is a path in G from v_0 to v_1, then there is a computable map $\tilde{\gamma}\colon Y \to Y$ such that $g(v_1) = \tilde{\gamma}(g(v_0))$, so that the diagram in Figure 1 commutes.

$$\begin{tikzcd}
v_0 \arrow[r, "\gamma"] \arrow[d, "g"'] & v_1 \arrow[d, "g"] \\
y_0 \arrow[r, "\tilde{\gamma}"] & y_1
\end{tikzcd}$$

Figure 1: The function g can be transported along a path γ if the diagram commutes.

The idea is to have a starting vertex v_0 with known $g(v_0) = y_0$ in the public parameters and the challenge $x = (\gamma_x, v_x)$ is a path γ_x on the graph starting at v_0 and ending at some vertex v_x. Then the evaluator has to compute $g(v_x) = y_x$ and provide additional information π such that verification is fast. The corresponding algorithms are

$\texttt{Setup}(\lambda, t) \longrightarrow \mathbf{pp} = \mathbf{ek} = \mathbf{vk} = (v_0, y_0, X, Y, \texttt{Sample}/\texttt{Hash})$

$\quad \texttt{Sample} \longrightarrow x = (\gamma_x, v_x)$

$\quad \texttt{Hash}(s) \longrightarrow x = (\gamma_x, v_x)$

$\texttt{Eval}(\mathbf{ek}, x) \longrightarrow (y_x, \pi_x)$

$\texttt{Verify}(\mathbf{vk}, x, y_x, \pi_x) \longrightarrow (\mathsf{valid}/\mathsf{invalid})$

where the delay is encoded in the length of the paths in X.

Whether or not the protocol is correct or sound depends heavily on the details of `Verify`. Since g is hard to compute, a faster way to get $g(v_x) = y_x$ is to move y_0 along the path γ_x. In fact this intended way has to be the fastest one, in order to achieve sequentiality. Therefore the path γ_x has to be given in a way that does not allow to find shortcuts easily. So either it has to be loop-free or given implicitly, i.e. not as a list of vertices. In addition finding a different, shorter path from v_0 to v_x has to be at least as hard as using γ_x. If this is true then the protocol is sequential if the transport $\tilde{\gamma}$ in Figure 1 is sequential.

Letting X be the set of all pairs (γ_x, v_x) with γ_x a path in G starting from v_0 and ending in $v_x \in V$, this VDF thus computes the function

$$f\colon X \to Y, \quad (\gamma_x, v_x) \mapsto \tilde{\gamma}_x(y_0) = g(v_x).$$

4 Isogeny-based Cryptography

In this section we provide the necessary basics for isogeny-based cryptography, quaternion algebras and the Deuring correspondence. We also discuss some computational problems in this area.

4.1 Elliptic Curves

Elliptic curves have ties to different fields resulting in several equivalent definitions. We will mostly follow the notation of Silverman [14], but restrict ourselves to aspects relevant for this paper.

Definition 4.1 (Elliptic Curve) *An* elliptic curve *is a pair (E, O), where E is a curve of genus one and $O \in E$. It is* defined *over a field K, if it is defined over K as a curve and $O \in E(K)$.*

We can define an addition of points on the curve making $(E, +)$ an additive group where O is the neutral element. This permits scalar multiplication written as $[m]\colon E \to E$ and torsion subgroups $E[m] := \{P \in E \mid [m]P = O\}$.

Definition 4.2 (Isogeny) *Let E and E' be elliptic curves. Then a morphism $\varphi\colon E \to E'$ such that $\varphi(O) = O$ is called an* isogeny*. If a non-zero isogeny $\varphi\colon E \to E'$ exists, then E and E' are called* isogenous*.*

In fact, every isogeny between two curves is also a group homomorphism. The isogenies from a curve E into itself form the endomorphism ring $\operatorname{End} E$. Isogenies can be written as rational maps and their degree is defined by this map. Thus, the degree $\deg(\varphi \circ \varphi') = \deg \varphi \deg \varphi'$ is multiplicative. In addition each isogeny $\varphi\colon E \to E'$ has a unique dual isogeny $\hat{\varphi}\colon E' \to E$ such that the composition $\hat{\varphi} \circ \varphi = [\deg \varphi]$ is the multiplication by the degree. The isogenies of degree 1 are the isomorphisms, and each isomorphism class can be labelled by the so-called j-invariant. This allows to construct the ℓ-isogeny graph that has those j-invariants as vertices and isogenies of degree ℓ as edges.

Definition 4.3 (Supersingularity) *Let K be a field of characteristic p and E an elliptic curve defined over K. The curve E is* supersingular *if the torsion group $E[p]$ is trivial. Equivalently, this means that the endomorphism ring $End E$ is an order in a quaternion algebra.*

Many isogeny-based protocols rely on secret walks in isogeny graphs of supersingular elliptic curves. The fact that the endomorphism ring is non-commutative gives rise to presumably quantum-secure protocols and the graphs have fast mixing properties, meaning that we reach an almost uniform distribution on the graph after a short random walk.

For the rest of this paper $p > 3$ will be a large prime. This allows us to write every elliptic curve in short Weierstraß form as $E\colon y^2 = x^3 + Ax + B$ with $j(E) = 108(4A)^3/(4A^3 + 27B^2)$. For supersingular curves there is always a representation with A, B, j in $\mathbb{F}_{p^2}$. In this setting we have the isogenies

$$\begin{aligned} [-1]\colon (x, y) &\mapsto (x, -y)\,, \\ [\mathrm{i}]\colon (x, y) &\mapsto (-x, \mathrm{i}y) \quad \text{and} \\ \phi\colon (x, y) &\mapsto (x^p, y^p) \end{aligned}$$

where ϕ is called Frobenius map. Note that

$$\begin{aligned} [\mathrm{i}] \circ [\mathrm{i}] &= [-1]\,, \\ \phi \circ \phi &= [-p] \quad \text{for supersingular curves,} \\ [\mathrm{i}] \circ \phi &= [-1] \circ \phi \circ [\mathrm{i}] \quad \text{for } p \equiv 3 \bmod 4 \end{aligned}$$

and [i] is not necessarily an endomorphism.

4.2 Quaternion Algebras

We have already seen in Definition 4.3 that supersingular curves are related to quaternion algebras. We are interested in the quaternion algebra $\mathcal{B}_{p,\infty}$ ramified at p and infinity with $\mathbb{Q}$-basis $\{1, \mathrm{i}, \mathrm{j}, \mathrm{k}\}$ such that

$$\mathrm{i}^2 = -1, \quad \mathrm{j}^2 = -p, \quad \mathrm{k} = \mathrm{ij} = -\mathrm{ji}.$$

The (reduced) norm of an element $\alpha = a_1 + a_2\mathrm{i} + a_3\mathrm{j} + a_4\mathrm{k} \in \mathcal{B}_{p,\infty}$ is given by $\alpha\bar{\alpha}$ for $\bar{\alpha} = a_1 - a_2\mathrm{i} - a_3\mathrm{j} - a_4\mathrm{k}$. An order in $\mathcal{B}_{p,\infty}$ is a lattice that is also a subring, and it is maximal if its discriminant equals p.

An elliptic curve E is supersingular if and only if $\operatorname{End} E$ is isomorphic to a maximal order $\mathcal{O}$ in $\mathcal{B}_{p,\infty}$, i.e. $\mathbb{Q} \otimes \operatorname{End} E \cong \mathcal{B}_{p,\infty}$. If $E\colon y^2 = x^3 + Ax$ and $p \equiv 3 \bmod 4$ we have seen that there are the isogenies [i] and ϕ with the same properties as i and j. Deuring even proved that the isomorphism classes of supersingular elliptic curves correspond to the isomorphism classes of invertible left $\mathcal{O}$-ideals in the quaternion algebra [11]. This so-called Deuring correspondence also gives us that an ℓ-isogeny φ starting at E corresponds to a left ideal I_φ of norm ℓ in $\mathcal{O} \cong \operatorname{End} E$ and the image curve has an endomorphism ring isomorphic to the right order $\mathcal{O}_R(I_\varphi) = \{\alpha \in \mathcal{B}_{p,\infty} \mid I_\varphi \alpha \subseteq I_\varphi\}$ of I_φ, see [15, Ch. 42] for more details.

Both orders and ideals can also be written as lattices such that

$$(\alpha_1\ \alpha_2\ \alpha_3\ \alpha_4) = (1\ \mathrm{i}\ \mathrm{j}\ \mathrm{k})L$$

is a basis of the lattice and L is a 4×4 matrix over $\mathbb{Q}$. We will identify a lattice with this matrix and always use Hermite normal form of L to have a unique representation.

4.3 Computational Problems

In this subsection we list some (relevant) computational problems in the field of isogeny-based cryptography. This toolbox will then be used to build the VDF. They are grouped into three categories: easy, medium and hard.

The easy problems can be solved fast and have a polynomial or even constant complexity.

Easy 1: Compute isogenies of small or smooth degree.

Easy 2: Given two elliptic curves E, E', an isogeny $\varphi\colon E \to E'$ as well as the corresponding order $\mathcal{O} \cong \operatorname{End} E$ and ideal I_φ, compute $\mathcal{O}' \cong \operatorname{End} E'$.

Easy 3: Given two elliptic curves E, E', and the corresponding orders $\mathcal{O} \cong \operatorname{End} E$, $\mathcal{O}' \cong \operatorname{End} E'$, compute a connecting ideal I corresponding to an isogeny $\varphi_I\colon E \to E'$.

Easy 4: Given a left ideal I of a maximal order $\mathcal{O} \subset \mathcal{B}_{p,\infty}$, find an equivalent ideal such that its norm is small or a prime power.

Easy 1 can be solved using Vélu's formulae. For Easy 2 we can compute $\mathcal{O}'$ as $\mathcal{O}_R(I_\varphi)$ and the connecting ideal I in Easy 3 satisfies $\mathcal{O} = \mathcal{O}_L(I)$, where the left order $\mathcal{O}_L(I)$ is defined analogously to the right order $\mathcal{O}_R(I)$, and $\mathcal{O}' = \mathcal{O}_R(I)$. Easy 4 is solved by the KLPT algorithm [12]. For VDFs we need moderately hard problems, which are still polynomial in complexity but might take a considerable time to compute. In [6] the last one is mentioned as bottleneck.

Medium 1: Compute isogenies of medium or non-smooth degree.

Medium 2: Given $\mathcal{O} \cong \operatorname{End} E$, translate between isogenies $\varphi\colon E \to E'$ and their corresponding left $\mathcal{O}$-ideals I_φ.

Depending on the degree we can use Vélu's formulae or the $\sqrt{\text{élu}}$ algorithm [1] to solve Medium 1. Meduim 2 is the core of our approach and is discussed in more detail in Section 6.4. The hard problems are the basis for encryption or signature schemes like CSIDH [3] or SQISign [6] and are equivalent [17].

Hard 1: Given two (isogenous) supersingular elliptic curves E, E' and a prime ℓ, find a path from E to E' in the ℓ-isogeny graph.

Hard 2: Given a supersingular elliptic curve E, find four endomorphisms that generate $\operatorname{End} E$ as a lattice.

Hard 3: Given a supersingular elliptic curve E, find four quaternions in $\mathcal{B}_{p,\infty}$ that generate a maximal order $\mathcal{O}$ such that $\mathcal{O} \cong \operatorname{End} E$.

Remark 4.4 *Knowledge of endomorphism rings can break the hard problems. If we know both endomorphism rings the first hard problem becomes medium using* Easy 3, 4 *and* Medium 2*. If we know an isogeny from a curve with known endomorphism ring to our curve also the second hard problem becomes medium by* Medium 2 *and* Easy 2*. The third hard problem reduces to the second via* Medium 2*.*

Finding supersingular elliptic curves can be done in two ways. One can reduce an elliptic curve in characteristic 0 modulo a prime and check if the resulting curve is supersingular, or take a random isogeny starting at one of these curves. In both cases the endomorphism ring of the final curve can be computed either via reduction or by transport along the isogeny. But as discussed in Remark 4.4 this weakens the hard problem cryptosystems may be based on and hence those require curves with unknown endomorphism ring. This in turn forces them to use a multi-party computation or a trusted authority in their setup to ensure that no single participant knows a complete path from a curve with known endomorphism ring to the one used.

5 VDF from Endomorphism Rings

The idea for our protocol is to exploit the fact that we can compute the endomorphism ring for the image curve of any known isogeny. Broadly speaking, the `Setup` should provide a starting curve E_0, some information about its endomorphism ring $\operatorname{End} E_0$ and an algorithm `Hash` to generate paths in the isogeny graph starting at E_0 and ending at some other curve E_x. `Eval` then has to compute $\operatorname{End} E_x$ and finally `Verify` checks this result. A schematic overview is given in Figure 2.

5.1 Setup

The `Setup` takes two arguments as input. The security parameter λ influences the size of prime p, i.e. the characteristic of the underlying field, and the delay

$$\begin{tikzcd}
E_0 \arrow[r, "\text{known}"] \arrow[d, "\varphi"', "\texttt{Hash}"] & \mathcal{O}_0 \arrow[r, "\sim", "\iota_0"'] \arrow[d, "I_\varphi"'] & \operatorname{End} E_0 \arrow[d, "\texttt{Eval}"] \\
E_x \arrow[r, "\text{hard}"] & \mathcal{O}_x \arrow[r, "\sim", "\iota_x"'] & \operatorname{End} E_x
\end{tikzcd}$$

Figure 2: Main idea of our VDF using endomorphism rings.

parameter t governs the length of the paths generated by Hash. For the starting curve any supersingular curve with known endomorphism ring can do, but we will take $E_0\colon y^2 = x^3 + x$. Since Definition 2.3 permits adversaries with polynomial precomputation, we may give all (useful) information regarding the endomorphism ring of E_0. This can include a generating set of endomorphisms, the corresponding set of ideals generating the order $\mathcal{O}_0 \cong \operatorname{End} E_0$ and an explicit and efficiently computable isomorphism ι_0 between these two representations.

For generating a challenge $x \in X$ we will use a hash function. If we need random elements, we can take a random seed. Our Hash will be either the function described in [4] or a similar one constructed in [5]. Both take a binary string as input and output a j-invariant j_x. Internally they walk a path in the 2-isogeny graph that starts at $j(E_0)$ and ends at the output j_x, thus implicitly describing an isogeny from E_0 to a curve E_x with $j(E_x) = j_x$. Since the zeroth step is always to j-invariant 287496, i.e. $E_1\colon y^2 = x^3 + 11x \pm 14\mathrm{i}$, Setup will precompute $\mathcal{O}_1 \cong \operatorname{End} E_1$ and we say that Hash starts at E_1.

Written as algorithms we have $\texttt{Setup}(\lambda, t) \longrightarrow \mathbf{pp} = (E_0, D_0, X, Y, \texttt{Hash})$ and $\texttt{Hash}(s) \longrightarrow x = (s, j_x)$ where E_0 is the curve $y^2 = x^3 + x$ and $D_0 = (\mathcal{O}_0, E_1, \mathcal{O}_1)$. For paths of length n the input for Hash is $s \in S = \{0,1\}^n$ and the output is in $X = S \times V$ with V the vertex set of the 2-isogeny graph labelled by j-invariants. A response $y \in Y = \mathrm{Mat}_{4\times 4}(\mathbb{Q})$ is the unique lattice basis $y = H$ in Hermite normal form of $\mathcal{O}_x \cong \operatorname{End} E_x$. Note that we do not have a fast and secure way to implement Hash yet. See Section 6 for more details.

5.2 Evaluation

Our idea is to iteratively translate the path or isogeny implicitly given by Hash into the corresponding ideal and use that to compute $\mathcal{O}_x$. For a VDF to be sequential the intended way to find y also has to be the fastest one. Since finding the endomorphism ring of a random curve is hard, we have to use an isogeny from a curve with known endomorphism ring to compute it. Finding an isogeny between two curves with at least one unknown endomorphism ring is hard too and to shorten an isogeny one has to translate it into an ideal first. So we can enforce the usage of the long isogeny given by Hash and translating it into an ideal will be the first step in any reasonable approach. Therefore this is the primary source for our delay and thus allows us to adjust the delay via the length of the path.

Hence, the algorithm $\texttt{Eval}(E_0, D_0, \texttt{Hash}, (s, j_x)) \longrightarrow (H, E_x)$ first extracts an (iterated) isogeny $\varphi_1 \circ \cdots \circ \varphi_n = \varphi\colon E_1 \to E_x$ with $j(E_x) = j_x$ from Hash with input $s \in \{0,1\}^n$ and $\varphi_i\colon E_i \to E_{i+1}$ of degree 2. Then for each step it uses $\mathcal{O}_i$ to translate φ_i into the corresponding ideal I_i and computes $\mathcal{O}_{i+1}$ as the right order of I_i. This is done with the algorithms explained in Section 6 which are similar to those in SQISign [6]. Finally for $E_{n+1} = E_x$ it computes the order $\mathcal{O}_x$. This is a lattice and we can give a basis A_x for it in terms of $\{1, \mathrm{i}, \mathrm{j}, \mathrm{k}\}$. If we interpret A_x as a matrix with the basis vectors as columns we can compute its Hermite normal form H. This gives a unique basis for $\mathcal{O}_x$ and will be the unique part $y = H$ of the output. The proof π contains additional information for faster verification. In our case we have $\pi = E_x$ and this causes no overhead, since computing E_x is part of finding $\mathcal{O}_x$.

5.3 Verification

The verifier has to check that H is a basis for $\mathcal{O}_x \cong \operatorname{End} E_x$ and that E_x is a valid curve. This can be done as follows: First check if $j(E_x) = j_x$, $|E_x(\mathbb{F}_{p^2})| = (p+1)^2$ and if H is in Hermite normal form. To do this it is necessary that j_x is either part of the challenge $x = (s, j_x)$ or that it can be validated in an other efficient way, i.e. not computing the hash function. Then we can compute the order generated by H and verify that its discriminant is p. This proves that it is a maximal order. Finally we check that the columns of H correspond to endomorphisms of $E_x/\mathbb{F}_{p^2}$. We do this by computing their action on a set of points $\{P, Q\}$ generating the $\mathbb{F}_{p^2}$-rational points of E_x. In total this allows the algorithm $\texttt{Verify}((s, j_x), H, E_x) \longrightarrow (\mathsf{valid}/\mathsf{invalid})$ to verify that E_x and $\mathcal{O}_x \cong \operatorname{End} E_x$ have been computed correctly.

5.4 Security Analysis

In this subsection we discuss the correctness, soundness and sequentiality of our protocol. Since there are no heuristic processes involved, our mathematical foundation provides correctness.

The only responses $(y, \pi) = (H, E_x)$ that get accepted have to have $j(E_x) = j_x$, $|E_x(\mathbb{F}_{p^2})| = (p+1)^2$ and $\mathcal{O}_x \cong \operatorname{End} E_x$ a maximal order in $\mathcal{B}_{p,\infty}$. We require curves E_x with the right j-invariant, that are isogenous to E_0 over $\mathbb{F}_{p^2}$, i.e. also having $(p+1)^2$ $\mathbb{F}_{p^2}$-rational points. These curves are isomorphic over $\mathbb{F}_{p^2}$ and post-composing an isogeny from E_0 to E_x with an isomorphism does not change is kernel. Thus, the corresponding ideals I and their right orders $\mathcal{O}_R(I) = \mathcal{O}$ are the same. Fixing the isomorphism $\iota_x \colon \mathcal{O}_x \to \operatorname{End} E_x$ by mapping i to [i] and j to the Frobenius map ϕ also fixes $\mathcal{O}_x$. The Hermite normal form then gives a unique representation of the lattice basis of $\mathcal{O}_x$. This makes $y = H$ unique and our protocol sound.

The sequentiality is based on the hard problems presented in Section 4.3. As mentioned in Section 5.2, computing the order $\mathcal{O}_x$ directly is hard and finding a path from E_0 to E_x is also hard, since we do not know $\mathcal{O}_x$ yet. So we have to use the long path defined by the hash function. The number of steps should be in $\log p$ and the number of vertices in the 2-isogeny graph is roughly $p/12$. Due to the fast mixing properties of the graph, the probability of having a loop in the path is negligible for sufficiently large p. To find an equivalent isogeny with smaller degree, we have to translate the path into an isogeny and then into a single large ideal to use algorithms like Algorithm 8 from [6] to get an equivalent ideal of smaller norm. Another option is to map all four generating endomorphism α_i to $\varphi \circ \alpha_i \circ \hat{\varphi} / \deg \varphi$ (either iteratively or in one go) and translate these into ideals. The last two approaches need points of large order, which only exist over field extensions of $\mathbb{F}_{p^2}$. This makes every computation slower and is at least as hard as using the small steps directly as intended. Also there is not much improvement to be gained using parallel computing since most of the steps are intrinsically sequential. Regarding precomputation we could only provide the first m possible steps of the hash function, but since there are 2^m options this would need sufficient time and memory to compute and store 2^m elliptic curves and their corresponding orders. This also becomes infeasible quite fast and therefore does not constitute a significant advantage.

6 Implementation

We provide a proof-of-concept implementation, while the optimised algorithms used in SQISign will also improve our approach even though the parameters might be different. We mostly follow the notation of SQISign. Our Sage implementation can be found at https://git.fim.uni-passau.de/zumbraegel/defend.

We have a toy example and one that resembles the final protocol more closely. The toy example adopts the above algorithms more naively and does everything in one big step instead of the iterated approach with the hash function. This has the advantage that we can do almost everything on E_0 and $\mathcal{O}_0$. The downside is that almost everything can be precomputed, so there is not much delay, but it shows all the necessary computations for an individual step in the proper protocol. It samples a point P on E_0 of given order D as the kernel of our isogeny $\varphi\colon E_0 \to E_x$, translates that into one $\mathcal{O}_0$-ideal I_φ and computes the order $\mathcal{O}_x$ directly as $\mathcal{O}_R(I_\varphi)$. The other example uses small parameters and not-optimised algorithms, but comprises all the parts of the final protocol.

6.1 Parameters

As mentioned before we take $E_0\colon y^2 = x^3 + x$. Our prime p has to satisfy $p \equiv 3 \bmod 4$ to have $[\mathrm{i}] \circ \phi = -\phi \circ [\mathrm{i}]$ and $\operatorname{End} E_0$ isomorphic to $\mathcal{O}_0 = \langle 1, \mathrm{i}, \frac{\mathrm{i}+\mathrm{j}}{2}, \frac{1+\mathrm{k}}{2} \rangle_{\mathbb{Z}}$. Other than that we have no restrictions on p, since we only need 2-isogenies and the 2-torsion is always rational over $\mathbb{F}_{p^2}$. The security parameter λ governs the size of p as $\log p \approx \lambda$ and the delay parameter $t = n$ should be polynomial in $\log p$. The isomorphism mapping $\mathcal{O}_0$ to $\operatorname{End} E_0$ is given by sending i to [i] and j to the Frobenius map ϕ. This also implies an isomorphism $\mathcal{O} \to \operatorname{End} E$ for curves E via an isogeny $\varphi\colon E_0 \to E$ [6]. In a slight abuse of notation elements $\alpha \in \mathcal{O}$ are also used as maps $\alpha \in \operatorname{End} E$ implicitly applying this isomorphism.

6.2 Precomputation

To speed-up the following computations and reduce the advantage a possible attacker might have, we can provide auxiliary information. As stated earlier we will precompute the zeroth (internal) step of the hash function from $E_0\colon y^2 = x^3 + x$ with j-invariant 1728 to $E_1\colon y^2 = x^3 + 11x \pm 14\mathrm{i}$ with j-invariant 287496 and give a lattice basis of $\mathcal{O}_1 \cong \operatorname{End} E_1$.

For our toy example we can precompute a generating set $\{P_D, Q_D\}$ for the D-torsion group $E_0[D]$ and the evaluation of $\theta_0 = \mathrm{j} + \frac{1+\mathrm{k}}{2}$ and $\eta_0 = \mathrm{i}$ at those points.

6.3 Hash Function

The first live operation is the hash function based on modular polynomials like in [5]. An ℓ-isogeny graph is $\ell + 1$ regular and the modular polynomial for a curve in that graph is of degree $\ell + 1$ and has the $\ell + 1$ neighbouring j-invariants as roots. We work with the 2-isogeny graph and in our path we already know the j-invariant we came from. Therefore we can factor out this root, solve a quadratic equation and label the solutions with 0 and 1, allowing us to encode a path of length n in the 2-isogeny graph by a binary string $s = \{0, 1\}^n$. The curve E_0 is special, as it has one 2-isogeny connecting back to itself and two 2-isogenies to the same j-invariant. So we can factor out its own j-invariant and consider this as the zeroth step. The first proper step of `Hash` starts at E_1 with $j(E_1) = 287496$.

However, this way of generating a challenge $x = (s, j_x)$ requires $\Omega(n)$ operations, like `Eval`, and is therefore slow. Moreover, it allows the generator of the challenge to cheat and always evaluate faster. A possible solution would be an efficient way to verify the final j-invariant without computing the hash function, thereby allowing to simply take $X = S$ as the challenge space. A less elegant workaround would be to have a trusted authority (that must not participate in computing `Eval`) sampling and publishing the challenges $x = (s, j_x)$. This does not solve the problem of `Hash` being slow and therefore challenges still take a long time to compute.

6.4 Isogeny to Ideal

Second is the translation of a (separable) isogeny $\varphi\colon E \to E'$ of degree 2 into its corresponding ideal I_φ. The general description $I_\varphi = \{\alpha \in \mathcal{O} \mid \alpha(P) = O \text{ for all } P \in \ker\varphi\}$ does not really help with computing it, but we follow the approach of SQISign to write it as as $I_\varphi = \mathcal{O}\langle\alpha, 2\rangle$.

The kernel of φ is generated by a point P of order 2 and $\alpha(P)$ should be O. To find this P we can simply check the codomain of the three isogenies, whose kernels are generated by one of the three non-trivial points in $E[2]$. This can be done in parallel for a more reliable timing. Note that this also gives us the proof $\pi = E_x$

For isogenies starting at E_0 in SQISign one takes $\theta_0 = \mathrm{j} + \frac{1+\mathrm{k}}{2}$ and $\eta_0 = \mathrm{i}$ orthogonal to θ_0 to have a basis $\{P, \theta_0(P)\}$ of $E_0[2]$. Then one solves $[a]P + [b]\theta_0(P) = \eta_0(P)$ to get $\alpha = a + b\theta_0 - \eta_0$. This is useful for the precomputation, but for the following steps we have to take the corresponding elements/maps of $\mathcal{O} \cong \operatorname{End} E$.

In our toy example this can be sped-up by using the precomputed basis $\{P_D, Q_D\}$ of $E_0[D]$ with precomputed $\theta_0(P_D), \theta_0(Q_D), \eta_0(P_D), \eta_0(Q_D)$ to write P as $P = [u]P_D + [v]Q_D$ and solve

$$[a]([u]P_D + [v]Q_D) + [b]([u]\theta_0(P_D) + [v]\theta_0(Q_D)) = [u]\eta_0(P_D) + [v]\eta_0(Q_D)$$

using matrix multiplication.

6.5 Right Order

Next is the computation of the right order $\mathcal{O}_R(I)$ of a left $\mathcal{O}$ ideal I. Again the straight forward definition $\mathcal{O}_R(I) = \{\alpha \in \mathcal{B}_{p,\infty} \mid I\alpha \subset I\}$ is not helpful, but we can think of the ideal as a lattice and use the right transporter or colon lattice, as done in SQISign.

We know a basis of $\mathcal{O}$ and a representation of I as $I = \mathcal{O}\alpha + \mathcal{O}2$. So we can compute the lattice multiplications $\mathcal{O}\alpha, \mathcal{O}2$, take their lattice union and the Hermite normal form of the result is the lattice L corresponding to I. Now we want to find the lattice $L' = \{\alpha \in \mathcal{B}_{p,\infty} \mid L\alpha \subset L\}$. We can write $\alpha = a + b\mathrm{i} + c\mathrm{j} + d\mathrm{k}$ and compute the lattice multiplications

$$A_1 = L,\ A_\mathrm{i} = L\mathrm{i},\ A_\mathrm{j} = L\mathrm{j},\ A_\mathrm{k} = L\mathrm{k}$$

to get the new constraint $aA_1 + bA_\mathrm{i} + cA_\mathrm{j} + dA_\mathrm{k} \subset L$. Applying the dual of L, i.e. computing the matrix multiplications

$$B_1 = L^{-1}A_1,\ B_\mathrm{i} = L^{-1}A_\mathrm{i},\ B_\mathrm{i} = L^{-1}A_\mathrm{j},\ B_\mathrm{k} = L^{-1}A_\mathrm{k},$$

yields that $aB_1 + bB_\mathrm{i} + cB_\mathrm{j} + dB_\mathrm{k}$ has to be integral and therefore $(a, b, c, d) \subset (1/\det L)\mathbb{Z}^4$. Now L' is the kernel of the map

$$(a, b, c, d) \to aB_1 + bB_\mathrm{i} + cB_\mathrm{j} + dB_\mathrm{k} \bmod m$$

where m is the numerator of $\det L$. This can be computed via the Smith normal form.

6.6 Ideal to Isogeny

Finally we have to translate the basis of $\mathcal{O} \cong \operatorname{End} E$ into explicit endomorphisms. This is done by the same isomorphism as for $\mathcal{O}_0$, namely mapping i to [i] and j to the Frobenius map ϕ. The only difficulty is to map half-integers, but we can at least evaluate them at a point P using points Q such that $[2]Q = P$.

7 Summary and Outlook

Our approach to use the computation of endomorphism rings or rather their corresponding maximal orders allows for a presumably quantum-secure VDF with basically constant verification time. In addition the implementation does not require special primes, a trusted setup or involved algorithms. The drawback is that sampling challenges efficiently and without computing part of the evaluation is still an open problem. One solution might be to find a way to verify the output of the hash function without computing it. A workaround could be using a trusted authority to create challenges.

A natural next step is to implement the protocol in a faster language and integrate the state-of-the-art algorithms developed for SQISign. This also permits an estimation of (relative) running times and complexities. An interesting question for future work is whether there are other ways to instantiate this idea. For example, can we use orientations like in OSIDH [7] to prove that we took the correct path since different paths generate different orientations?

Acknowledgements

We are grateful to Antonio Sanso and all the people, who offered valuable feedback on a previous version of this paper. Knud Ahrens would also like to thank the participants of the Brainstorm Session held by the isogeny club at Eurocrypt 2023 in Lyon for their input and inspiring discussions.

References

[1] Daniel J. Bernstein, Luca De Feo, Antonin Leroux, and Benjamin Smith. Faster computation of isogenies of large prime degree. In Steven D. Galbraith, editor, *Proceedings of the Fourteenth Algorithmic Number Theory Symposium*, pages 39–55, Berkeley, 2020. Mathematical Sciences Publishers. doi: 10.2140/obs.2020.4.39.

[2] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, pages 757–788, Cham, 2018. Springer International Publishing. doi: 10.1007/978-3-319-96884-1_25.

[3] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: An efficient post-quantum commutative group action. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018*, pages 395–427, Cham, 2018. Springer International Publishing. doi: 10.1007/978-3-030-03332-3_15.

[4] Denis X. Charles, Kristin E. Lauter, and Eyal Z. Goren. Cryptographic hash functions from expander graphs. *J. Cryptology*, 22(1):93–113, 2009. doi: 10.1007/s00145-007-9002-x.

[5] Jorge Chavez-Saab, Francisco Rodríguez-Henríquez, and Mehdi Tibouchi. Verifiable isogeny walks: Towards an isogeny-based postquantum VDF. In Riham AlTawy and Andreas Hülsing, editors, *Selected Areas in Cryptography*, pages 441–460, Cham, 2022. Springer International Publishing. doi: 10.1007/978-3-030-99277-4_21.

[6] Jorge Chavez-Saab, Maria Corte-Real Santos, Luca De Feo, Jonathan Komada Eriksen, Basil Hess, David Kohel, Antonin Leroux, Patrick Longa, Michael Meyer, Lorenz Panny, Sikhar Patranabis, Christophe Petit, Francisco Rodríguez Henríquez, Sina Schaeffler, and Benjamin Wesolowski. SQIsign algorithm specifications and supporting documentation. Project Homepage, 2023. `https://sqisign.org/spec/sqisign-20230601.pdf`.

[7] Leonardo Coló and David Kohel. Orienting supersingular isogeny graphs. *J. Math. Cryptology*, 14(1):414–437, 2020. doi: 10.1515/jmc-2019-0034.

[8] Luca De Feo, Simon Masson, Christophe Petit, and Antonio Sanso. Verifiable delay functions from supersingular isogenies and pairings. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019*, pages 248–277, Cham, 2019. Springer International Publishing. doi: 10.1007/978-3-030-34578-5_10.

[9] 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, *Advances in Cryptology – ASIACRYPT 2020*, pages 64–93, Cham, 2020. Springer International Publishing. doi: 10.1007/978-3-030-64837-4_3.

[10] Thomas Decru, Luciano Maino, and Antonio Sanso. Towards a quantum-resistant weak verifiable delay function. In Abdelrahaman Aly and

Mehdi Tibouchi, editors, *Progress in Cryptology – LATINCRYPT 2023*, pages 149–168, Cham, 2023. Springer Nature Switzerland. doi: 10.1007/978-3-031-44469-2_8.

[11] Max Deuring. Die Typen der Multiplikatorenringe elliptischer funktionenkörper. *Abh. Math. Sem. Hansischen Univ.*, 14:197–272, 1941. doi: 10.1007/BF02940746.

[12] David Kohel, Kristin Lauter, Christophe Petit, and Jean-Pierre Tignol. On the quaternion ℓ-isogeny path problem. *LMS J. Comput. Math.*, 17: 418–432, 2014. doi: 10.1112/S1461157014000151.

[13] Barak Shani. A note on isogeny-based hybrid verifiable delay functions. Cryptology ePrint Archive, Paper 2019/205, 2019. `https://eprint.iacr.org/2019/205`.

[14] Joseph H. Silverman. *The Arithmetic of Elliptic Curves*, volume 106 of *Graduate Texts in Mathematics*. Springer New York, 1986. doi: 10.1007/978-1-4757-1920-8.

[15] John Voight. *Quaternion algebras*, volume 288 of *Graduate Texts in Mathematics*. Springer Cham, 2021. doi: 10.1007/978-3-030-56694-4.

[16] Benjamin Wesolowski. Efficient verifiable delay functions. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019*, pages 379–407, Cham, 2019. Springer International Publishing. doi: 10.1007/978-3-030-17659-4_13.

[17] 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, 2022. doi: 10.1109/FOCS52979.2021.00109.