Succinct Non-interactive Arguments of Knowledge from Supersingular Isogenies

Paulo L. Barreto[1], **Marcos A. Simplicio Jr**[2], **Gustavo H. M. Zanon**[1]

[1] University of Washington — Tacoma.
Tacoma (WA), US

[2]Escola Politécnica, University of São Paulo.
São Paulo (SP), Brazil

pbarreto@uw.edu, {msimplicio,gzanon}@larc.usp.br

Abstract. *A succinct non-interactive argument of knowledge (SNARK) enables a party to convince another of some statement (typically, knowledge of some information) by means of a short argument, while ensuring it is infeasible for an adversary to create a short argument of the opposite statement. We hereby describe a SNARK for CSI-FiSh signatures, whose security stems from hard problems involving supersingular isogenies. Although the scheme looks analogous to a SNARK for conventional Schnorr signatures, it is non-trivial in that, as we also show, a similar SNARK for another isogeny-based signature scheme (SQISign) is not viable. As a bonus, we also discuss how to drastically reduce the memory needed to implement the CSIDH framework required by CSI-FiSh signatures.*

1. Introduction

A succinct non-interactive argument of knowledge (SNARK) is a protocol designed to allow a user to prove that some statement is true via an argument of soft-polynomial size in the length of the statement. Typically, this is meant to argue about the knowledge of some information without revealing it, either in part or in full. If the protocol reveals nothing about that information, it is called a zero-knowledge SNARK (zk-SNARK). A plain SNARK might involve revealing part of the information, but in such a way that it is infeasible to retrieve the remaining part.

Although a plain SNARK cannot be used when traceability is a concern (e.g. to obtain fully anonymous credentials, which should not be traceable), it is useful in situations when that is a desirable or required feature. This is the case of delegated or proxy credentials: in that scenario, Alice delegates to Peter a credential to act as her proxy, and it is important that, even though Alice's identity may remain hidden, Peter can be traced down to conduct actions corresponding to her, and her alone (rather than being able to, say, impersonate another anonymous user Bob on a par with Alice). It is also the case of one-time credentials: Alice can give Peter a credential that remains anonymous for one single application, since it becomes traceable when used more than once. Credential delegation may involve Alice creating for Peter a signature that he will need to keep partially or entirely private, despite being necessary in his role as a proxy. In other words, Peter must resort to a SNARK of such a signature.

Full zero-knowledge SNARKs of Fiat-Shamir signatures [Fiat and Shamir 1987] are possibly overkill. For instance, they might involve computationally expensive and

complex garbled circuits or homomorphic operations to compute hash values, and consequently, also group/ring/field arithmetic. The use of a plain SNARK as a credential, whenever possible, may offer a substantial computational advantage[1].

To address this and other scenarios, we focus our attention on the problem of obtaining SNARKs for *isogeny-based* signature schemes. This is a much less explored area than other more well-known settings like lattices, and arguably more challenging due to the exquisite algebraic framework it involves. Yet, it is also an area that has been attracting growing interest on its own, and for its closer relationship to rich pre-quantum schemes like those based on elliptic curves.

Contributions. We show how to construct a SNARK of Commutative Supersingular Isogeny-based Fiat-Shamir (CSI-FiSh) digital signatures [Beullens et al. 2019], a signature scheme whose security stems from hard computational problems involving supersingular isogenies, particularly CSIDH [Castryck et al. 2018]. Our proposal is similar to a SNARK for conventional Schnorr signatures, but it is non-trivial because: (1) CSI-FiSh signatures typically use not a single private signing key, but a whole collection of them; and (2) a corresponding construction for another isogeny-based signature scheme, namely SQISign [De Feo et al. 2020], not only fails to work but apparently cannot be built along the same lines. We also describe a technique to reduce drastically (by a factor of at least 18 for lower-security parameters, and increasingly more so for higher security levels) the work space needed to implement the CSIDH framework required by CSI-FiSh signatures.

Remark. The recent Castryck-Decru attack [Castryck and Decru 2022] against SIDH very specifically depends on knowledge of isogeny images at two torsion base points, and thus does not apply to the CSIDH and CSI-FiSh schemes on which this work builds as no such points are ever revealed nor needed.

Organization. The rest of this paper is organized as follows. Sec. 2 reviews the notions of SNARKs and zk-SNARKs. Sec. 3 reviews Schnorr signatures, how they can be used as a zk-SNARK of a discrete logarithm, and how to obtain a SNARK of a Schnorr signature itself, preparing the reader for its isogeny-based counterpart. Sec. 4 proposes a supersingular isogeny-based SNARK of the CSI-Fish signature scheme (defined in Sec. 4.1), which can be itself viewed as a zk-SNARK of (a set of) private isogenies between supersingular curves. Sec. 5 briefly considers another isogeny-based signature scheme (SQISign), showing why it fails to yield a SNARK built under the same principles. Sec. 6 suggests a work space (and, to some extent, time) improvement for the decomposition of class group actions with large exponents in terms of efficiently computable ideals with small exponents. This technique has independent interest, and applicable in the implementation of other CSIDH-style cryptosystems. Sec. 7 concludes the work.

2. Succinct non-interactive arguments of knowledge

Hereafter we very closely follow Groth [Groth 2010] in the definitions of a SNARK and a zk-SNARK (see also [Camenisch and Lysyanskaya 2004] and [Ganesh et al. 2021]).

[1]Creating a zk-SNARK for other kinds of signatures may be simpler, as is the case of many pairing-based signature schemes like BBS+ [Au et al. 2006]. However, no such scheme seems to be currently known in a post-quantum setting.

Let R be an efficiently computable binary relation. For pairs $(C, w) \in R$ we call C the *statement* and w the *witness*. Let L be the NP-language consisting of statements with witnesses in R. For statements of size N, these are written respectively L_N and R_N.

Intuitively, a non-interactive argument of knowledge for R will consist of three algorithms Setup, Prove and Verify that run in probabilistic polynomial time:

- On input a security parameter λ the Setup algorithm produces a common reference string σ.
- On input (σ, C, w), the Prove algorithm produces an argument of knowledge π.
- On input (σ, C, π), the Verify algorithm outputs 1 if the argument is acceptable and 0 otherwise.

Formally, the tuple $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ is called a non-interactive argument of knowledge for R if it satisfies the following properties of *completeness* and *soundness*. It is called a SNARK if it also satisfies the following property of *succinctness*. Finally, it is called a zero-knowledge non-interactive argument of knowledge if it also satisfies the ensuing property of being *zero-knowledge*.

- *Perfect completeness.* Completeness captures the notion that an honest prover should be able to convince an honest verifier that the statement is true. For all adversaries $\mathcal{A}$ that output $(C, w) \in R$:

$$\Pr\left[\ \sigma \leftarrow \mathsf{Setup}(1^\lambda); (C, w) \leftarrow \mathcal{A}(\sigma); \pi \leftarrow \mathsf{Prove}(\sigma, C, w) : \mathsf{Verify}(\sigma, C, \pi) = 1\ \right] = 1.$$

- *Computational soundness.* Soundness captures the notion that it should be infeasible (in respect to security parameter k) for an adversary to come up with an accepting argument for a false statement. For all non-uniform polynomial-time adversaries $\mathcal{A}$:

$$\Pr\left[\ \sigma \leftarrow \mathsf{Setup}(1^\lambda); (C, \pi) \leftarrow \mathcal{A}(\sigma) : C \notin L \wedge \mathsf{Verify}(\sigma, C, \pi) = 1\ \right] \lesssim 2^{-k}.$$

- *Succinctness.* For any $(C, w) \in R$, the length of the proof π is $|\pi| = \mathrm{poly}(\lambda) \cdot \mathrm{polylog}(|C| + |w|)$.
- *Perfect zero-knowledge.* Intuitively, an argument is zero-knowledge if it does not leak any information besides the truth of the statement. If so, it can be indistinguishably replaced by a simulator where there is no information at all to be leaked. We say a non-interactive argument of knowledge $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ is perfect zero-knowledge if there exists a polynomial time simulator $S := (S_1, S_2)$ such that S_1 outputs a simulated common reference string σ and a simulation trapdoor τ, while S_2 takes the common reference string, the simulation trapdoor and a statement as input and produces a simulated argument, and all stateful interactive adversaries $\mathcal{A}$ that output $(C, w) \in R$:

$$\begin{aligned}&\Pr\left[\ \sigma \leftarrow \mathsf{Setup}(1^\lambda); (C, w) \leftarrow \mathcal{A}(\sigma); \pi \leftarrow \mathsf{Prove}(\sigma, C, w) : \mathcal{A}(\pi) = 1\ \right] = \\ &\Pr\left[\ (\sigma, \tau) \leftarrow S_1(1^\lambda); (C, w) \leftarrow \mathcal{A}(\sigma); \pi \leftarrow S_2(\sigma, \tau, C) : \mathcal{A}(\pi) = 1\ \right].\end{aligned}$$

3. The Schnorr signature scheme

In what follows, let $\mathbb{G}$ be a (multiplicative) group of prime order n where the discrete logarithm problem is assumed to be hard, let $\mathbb{Z}_n := \mathbb{Z}/n\mathbb{Z}$ denote the ring (actually the

field) of integers modulo n, and let $\mathcal{H}_k : \mathbb{G}^k \to \mathbb{Z}_n$ and $\mathcal{H}_k^* : \mathbb{G}^k \times \{0,1\}^* \to \mathbb{Z}_n$ be random oracles. Also, let $x \xleftarrow{\$} \mathbb{Z}_n$ denote the act of picking $x \in \mathbb{Z}_n$ uniformly at random.

The Schnorr identification scheme [Schnorr 1990] for a group generator $g \in \mathbb{G}$ and a public key $y \in \mathbb{G}$ (corresponding to a private signing key $s \in \mathbb{Z}_n$) is a challenge-response protocol where, given a commitment $u \in \mathbb{G}$ and a challenge $c \in \mathbb{Z}_n$, the response $z \in \mathbb{Z}_n$ satisfies $u = g^z y^c$. It can be made non-interactive by redefining $c = \mathcal{H}_3(g, y, u)$.

The corresponding signature scheme includes the message $m \in \{0,1\}^*$ in the challenge as $c = \mathcal{H}_3^*(g, y, u, m)$, yielding a short transcript $(c, z) \in \mathbb{Z}_n^2$ satisfying $c = \mathcal{H}_3(g, y, u)$ with $u = g^z y^c$. Equivalently, one can define the signature as the short transcript $(u, z) \in \mathbb{G} \times \mathbb{Z}_n$ satisfying $u = g^z y^c$ with $c = \mathcal{H}_3(g, y, u)$.

3.1. ZK-proving knowledge of discrete logarithms

In this section we essentially follow Camenisch [Camenisch 1998, Definitions 3.1 and 3.3] with a slightly different notation.

A zk-SNARK of the discrete logarithm $s \in \mathbb{Z}_n$ of a group element $y \in \mathbb{G}$ to the base $g \in \mathbb{G}$, denoted $\mathrm{SPK}\{(s) : y = g^s\}$, consists simply of a Schnorr signature, that is, a pair $(c, z) \in \mathbb{Z}_n^2$ satisfying $c = \mathcal{H}_3(g, y, u)$ with $u = g^z y^c$.

A zk-SNARK of t simultaneous discrete logarithms $s_j \in \mathbb{Z}_n$ of corresponding group elements $y_j \in \mathbb{G}$ to the respective bases $g_j \in \mathbb{G}$, denoted $\mathrm{SPK}\{(s_j) : y_j = g_j^{s_j} \mid 1 \le j \le t\}$, consists of a tuple $(c, z_1, \ldots, z_t) \in \mathbb{Z}_n^{t+1}$ satisfying $c = \mathcal{H}_{3t}(g_1, \ldots, g_t, y_1, \ldots, y_t, u_1, \ldots, u_t)$ with $u_j = g_j^{z_j} y_j^c$ for all $1 \le j \le t$, implying t signatures sharing the same challenge. If some components are repeated in some scenario, one can simplify the computations by shortening the hash argument list by convention, omitting the repetitions. For instance, if the generator g is the same for all simultaneous discrete logarithms, one could set $c = \mathcal{H}_{1+2t}(g, y_1, \ldots, y_t, u_1, \ldots, u_t)$ instead.

The above constructions could equivalently adopt the alternative form of Schnorr signatures. For instance, one could define $\mathrm{SPK}\{(s_j) : y_j = g_j^{s_j} \mid 1 \le j \le t\}$ as a tuple $(u_1, \ldots, u_t, z_1, \ldots, z_t) \in \mathbb{G}^t \times \mathbb{Z}_n^t$ satisfying $u_j = g_j^{z_j} y_j^c$ for all $1 \le j \le t$, with $c = \mathcal{H}_{3t}(g_1, \ldots, g_t, y_1, \ldots, y_t, u_1, \ldots, u_t)$.

From this point onward, call Alice the entity who creates a zk-SNARK of a private signing key corresponding to a given public key (that is, Alice is the original signer), and call Peter the entity who creates a SNARK of a signature generated by Alice for some message.

3.2. SNARK of a Schnorr signature

One can obtain a SNARK of a full Schnorr signature that is written in the equivalent form $(u, z) \in \mathbb{G} \times \mathbb{Z}_n$ by revealing only u while keeping z private, redefining the message space to $\mathbb{Z}_n$, and replacing the message m by $r_m := g^m$ as an argument to the hash function, that is, redefining $c = \mathcal{H}_4(g, y, u, r_m)$. Specifically, from g, y, u, and m, Peter creates a SNARK of z and m by using these two quantities as Schnorr signing keys for the public keys $r_z := g^z = u/y^c$ and $r_m := g^m$, with $c = \mathcal{H}_4(g, y, u, r_m)$. This is the idea underlying the signature scheme described in [Galindo and Garcia 2009].

The desired SNARK of a Schnorr signature (u, z) for a message m, verifiable under the public key y given u, is denoted $\mathrm{SPK}\{(z, m) : r_z = g^z \wedge r_z = u/y^c \wedge r_m = g^m \wedge$

$c = \mathcal{H}_4(g, y, u, r_m)\}(u)$ and consists of a tuple $(u, r_m, d, w_z, w_m) \in \mathbb{G}^2 \times \mathbb{Z}_n^3$ satisfying $d = \mathcal{H}_5(g, r_z, r_m, v_z, v_m)$ with $c := \mathcal{H}_4(g, y, u, r_m)$, $r_z := u/y^c$, $v_z := g^{w_z} r_z^d$, $v_m := g^{w_m} r_m^d$. Notice that this contains a zk-SNARK of two simultaneous discrete logarithms, z and m.

Our construction of a SNARK of a CSI-FiSh signature will be inspired on this construction, after some important differences between Schnorr and CSI-FiSh signatures are taken care of.

4. CSI-FiSh signatures, SNARKs and zk-SNARKs

Like in the CSI-FiSh scheme [Beullens et al. 2019], start by computing a prime $p := 4\prod_{j=1}^{n} \ell_j - 1$ for some n, as follows: for $j < n$, the set of ℓ_j are picked as the first $n-1$ odd primes; then, $\ell_n > \ell_{n-1}$ is picked as the smallest prime such that p computed in this manner is prime. The target size of p depends on the underlying CSIDH structure; for example, for CSIDH-512 [Castryck et al. 2018], one would have $\lceil \lg p \rceil = 511$. Let $\mathcal{E}\ell\ell$ be the set of $\mathbb{F}_p$-isomorphism classes of supersingular elliptic curves with $\mathbb{F}_p$-rational endomorphism ring $\mathcal{O}$. Any element of $\mathcal{E}\ell\ell$ can be represented by some Montgomery curve [Montgomery 1987] $E_A : y^2 = x^3 + Ax^2 + x$, where $A \in \mathbb{F}_p$ and $\#E_A = p + 1$ (in particular, $E_0 : y^2 = x^3 + x$ is such a curve), and $\mathcal{O} = \mathrm{End}_{\mathbb{F}_p}(E_0)$ can be seen as the set of all $\mathbb{F}_p$-isogenies from E_0 to itself. The (quadratic) twist of such a curve E_A for $A \neq 0$ is the curve $E'_A := E_{-A}$. For convenience, define the map $\psi : \mathcal{E}\ell\ell \to \mathbb{F}_p$ as $\psi(E_A) := A$.

The ideal class group of $\mathcal{O}$, denoted $\mathrm{Cl}(\mathcal{O})$, which is the quotient of the group of fractional invertible ideals in $\mathcal{O}$ by the principal fractional invertible ideals, acts freely and transitively on $\mathcal{E}\ell\ell$. The action of an ideal $\mathfrak{a} \in \mathrm{Cl}(\mathcal{O})$ on a curve $E \in \mathcal{E}\ell\ell$ corresponds to applying to E an isogeny whose kernel is the intersection of all kernels of the endomorphisms in $\mathfrak{a}$. Of particular interest are the ideals $\mathfrak{l}_j := \langle \ell_j, \pi - 1 \rangle$ (where π is the $\mathbb{F}_p$-Frobenius endomorphism), which correspond to isogenies with kernels of order ℓ_j defined over $\mathbb{F}_p$. These are easily computable using Vélu's formulas. We will henceforth assume the CSIDH/CSI-FiSh setting whereby the class group $\mathrm{Cl}(\mathcal{O})$ is cyclic of known order $N := \#\mathrm{Cl}(\mathcal{O})$ generated by the class of some ideal $\mathfrak{g}$. For instance, for CSIDH-512, where $\lceil \lg p \rceil = 511$ and $\lceil \lg N \rceil = 258$, one could take $\mathfrak{g} = \mathfrak{l}_1 = \langle 3, \pi - 1 \rangle$. Thus, any ideal has the form $\mathfrak{a} = \mathfrak{g}^a$ for some $a \in \mathbb{Z}_N$, and its action on a curve E can be conveniently written as $[a]E := \mathfrak{a} * E = \mathfrak{g}^a * E$.

The following property is responsible for an efficiency improvement in CSI-FiSh signatures, to be discussed later in the context of a SNARK of such signatures:

Property 1. For any $a \in \mathbb{Z}_N$ and any $A \in \mathbb{F}_p$ such that E_A is supersingular, $\psi([-a]E_{-A}) = -\psi([a]E_A)$, i.e., $[a]E_A = E_B \Rightarrow [-a]E_{-A} = E_{-B}$.

Remark 1. Computing the group action $[a]E_0$ is carried out by expressing $\mathfrak{g}^a \in \mathrm{Cl}(\mathcal{O})$ as a short(ish) vector on the lattice defined by the easily computable actions of the ideals $\mathfrak{l}_j$, i.e., writing $\mathfrak{g}^a = \prod_j \mathfrak{l}_j^{e_j}$ for small(ish) integers e_j, e.g. by applying Babai's nearest-plane algorithm — especially when combined with the decomposition technique to eliminate multiple-precision floating-point arithmetic described in Section 6.

4.1. The CSI-FiSh scheme

We now succinctly describe the original CSI-FiSh scheme. Some modifications will be made latter to obtain variants more amenable to SNARKs and zk-SNARKs.

- **Setup:** Choose a suitable CSIDH parameter set, with known class group order $N := \#\mathrm{Cl}(\mathcal{O})$, together with a Fiat-Shamir security parameter t and a number C of curves. For convenience, the message space is $\mathbb{Z}_N$, and the hash function is now a map $\mathcal{H}_t^* : \mathbb{F}_p^t \times \mathbb{Z}_N \to \{-C+1, \ldots, C-1\}^t$.
- **Key generation:** Pick $C-1$ private keys $s_k \xleftarrow{\$} \mathbb{Z}_N$ (corresponding to ideals $\mathfrak{s}_k := \mathfrak{g}^{s_k}$) and compute $E_{A_k} \leftarrow [s_k]E_0$. The key pair collection is $\mathsf{sk} := (s_k \mid 1 \leq k < C) \in \mathbb{Z}_N^{C-1}$, $\mathsf{pk} := (A_k \mid 1 \leq k < C) \in \mathbb{F}_p^{C-1}$. Curve twists are implicitly used as well, with the convention $s_{-k} := -s_k$ (and $s_0 := 0$), and corresponding public keys $E_{-A_k} := [-s_k]E_0$ (and E_0 itself), for a total of $2C-1$ key pairs.
- **Signing and verification:** A valid CSI-FiSh signature for a message $m \in \mathbb{Z}_N$ is a tuple $(c_1, \ldots, c_t, a_1, \ldots, a_t) \in \{-C+1, \ldots, C-1\}^t \times \mathbb{Z}_N^t$ such that $(c_1, \ldots, c_t) = \mathcal{H}_t^*(B_1, \ldots, B_t, m)$, where $B_j := \psi([a_j]E_{A_{c_j}})$ for all $1 \leq j \leq t$. This is obtained by sampling t nonces $r_j \xleftarrow{\$} \mathbb{Z}_N$, computing $B_j := \psi([r_j]E_0)$, hashing $(c_1, \ldots, c_t) \leftarrow \mathcal{H}_t^*(B_1, \ldots, B_t, m)$, and setting $a_j \leftarrow r_j - s_{c_j} \pmod N$ for $1 \leq j \leq t$.

4.2. Signatures as zk-SNARKs of private keys

Using a Schnorr signature as a zk-SNARK of the private signing key is straightforward because, among other things, there is only a single key to prove. We propose to apply the same basic principle to CSI-FiSh signatures as a zk-SNARK of the private signing key(s), but need to handle some small yet relevant differences.

For a scheme like CSI-FiSh, although using a single private key is possible, it would typically incur efficiency issues (a larger number of repetitions of the implicit challenge-response protocol and larger signatures), so in general the actual signing key is an entire collection of individual but related keys, namely, isogenies from the same starting curve E_0. This means that a CSI-FiSh signature is a succinct non-interactive argument of knowledge of *some* of those individual keys, but not necessarily all of them. For instance, the proposed setting [Beullens et al. 2019, Table 3] where $C = 2$ and $t = 56$, that is, with a single private signing key and a public key consisting of curves $E_{\pm A}$ and E_0, is arguably a SNARK of that private key. In contrast, in the setting where $C = 2^{15}$ and $t = 7$, that is, with $2^{15} - 1$ private keys and only 7 implicit iterations of the underlying identification scheme, one may be left with the impression that a SNARK would not cover all of the keys but rather only 7 of them at best.

From a practical point of view, this is not really an issue. The challenge (that is, the actual choice of a public-key curve to come from in order to reach the commitment curve) is not only unknown to the signer beforehand but uncontrollable as well. Thus the probability of successfully forging a signature when some but not all private keys are known can be made as low as desired. This means that what a CSI-FiSh signature is actually a SNARK of is not an individual private key, but the whole set, as though it were an equivalence class. Yet, for an application where the signature plays the role of a SNARK it makes more sense to impose tighter constraints on the allowed parameters. For the λ-bit security level, we thus require that the number t of iterations of the implicit challenge-response protocol be *larger* than the number of private keys by at least $\lambda/2$. This means the probability of any individual private key in the set never being considered is no larger than $2^{-\lambda/2}$. Asking for a larger margin is possible, but less meaningful due to

the unavoidable possibility of birthday collisions. Of course, there is always the possibility that a private key is left untouched by sheer chance, but this could happen even in the single-key case (namely, it is possible if unlikely that the challenge will be repeated every time), and going too far beyond the proposed parameter constraint will incur a substantial performance penalty.

In summary, we henceforth plausibly and sensibly require $t \geq C - 1 + \lambda/2$ for any CSI-FiSh signature that is meant as a zk-SNARK of the corresponding private key. This being said, we can proceed and state analogous definitions for zk-SNARKs of private isogenies in the form of ideals in the class group.

4.3. ZK-proving knowledge of (sets of) ideals

Again, we essentially follow Camenisch [Camenisch 1998, Definition 3.3], this time adapting the notation to the CSI-FiSh isogeny setting. We use parametrized hash functions $\mathcal{H}_{k,t,C} : \mathbb{F}_{p^2}^k \to \{0, \ldots, C-1\}^t$ and $\mathcal{H}^*_{k,t,C} : \mathbb{F}_{p^2}^k \times \mathbb{Z}_N \to \{-C+1, \ldots, C-1\}^t$. Contrary to the discrete logarithm setting, only the knowledge of a number of simultaneous ideals need to be considered, since CSI-FiSh already typically uses more than one private isogeny. For brevity, we only state the definition for the former hash, $\mathcal{H}_{k,t,C}$; the latter case is completely analogous.

A zk-SNARK of $C-1$ simultaneous ideals $\mathfrak{g}^{s_j}$, represented by $s_j \in \mathbb{Z}_N$ and specifying isogenies from a starting curve $E_{A_0} \in \mathcal{E}\ell\ell$ to curves $E_{A_j} \in \mathcal{E}\ell\ell$, is denoted $\mathrm{SPK}\{(s_j) : E_{A_j} = [s_j]E_{A_0} \mid 1 \leq j < C\}$ and consists of a tuple $(c_1, \ldots, c_t, a_1, \ldots, a_t) \in \{0, \ldots, C-1\}^t \times \mathbb{Z}_N^t$ satisfying $(c_1, \ldots, c_t) = \mathcal{H}_{C+t,t,C}(A_0, \ldots, A_{C-1}, B_1, \ldots, B_t)$, with $B_j := \psi([a_j]E_{A_{c_j}})$ for all $1 \leq j \leq t$. This implies t single-key CSI-FiSh signatures sharing the same challenge. It also immediately generalizes to a zk-SNARK of more than one set of private keys.

4.4. Toward a SNARK of a CSI-FiSh signature

In the case of Schnorr signatures, the underlying principle to obtain a SNARK of a signature is that one signature component, namely $z \in \mathbb{Z}_n$, has precisely the same nature as the private signing key itself, $s \in \mathbb{Z}_n$. We now show how to apply the same basic principle to obtain a SNARK of a CSI-FiSh signature.

In a CSI-FiSh signature, the sequence $c := (c_1, \ldots, c_t) \in \{-C+1 \ldots C-1\}^t$ indicates which keys are used for each signature component. The number of times each key appears on this list varies at random, but overall the keys are partitioned into $2C-1$ equivalence classes. To obtain a SNARK of a CSI-FiSh signature along similar lines to a SNARK of a Schnorr signature, Peter might think of committing to $2C-1$ curves $E'_j := [\zeta_j]E_{A_j}$ with $\zeta_j \xleftarrow{\$} \mathbb{Z}/n$ and $-C+1 \leq j \leq C-1$, and then using $2C-1$ CSI-FiSh signatures to perform one step of a zk-SNARK of all signature components associated to the starting curve E_{A_j} at once, for all $j \in \{-C+1, \ldots, C-1\}$. The number of protocol repetitions would depend on how many curves constitute each equivalence class, and revealing c would actually make this a SNARK rather than a zk-SNARK, but the idea is promising. However, it does have issues, as we will see (and handle them).

Example 1. We will use here a simple color code to make it easier to keep track of related information.

Let $C = 3$ and $t = 15$ so that $c \in \{-2 \dots 2\}^{15}$, and consider a public key consisting of curves E_A, and $E_{A'}$, their respective twists E_{-A} and $E_{-A'}$, and the curve E_0. On average, each value from $\{-2, -1, 0, 1, 2\}$ occurs roughly $t/(2C-1) = 3$ times on c. Suppose that, for some specific signature, the hash value is $c = (0, 2, 1, 2, -1, -1, 0, 0, -2, -1, 1, 1, -2, 0, 2)$. This corresponds to the choice

$$(E_0, E_{A'}, E_A, E_{A'}, E_{-A}, E_{-A}, E_0, E_0, E_{-A'}, E_{-A}, E_A, E_A, E_{-A'}, E_0, E_{A'})$$

of public-key curves for that specific signature, and these in turn are the starting curves of isogenies leading to the signature curves

$$(E_{B_1}, E_{B_2}, E_{B_3}, E_{B_4}, E_{B_5}, E_{B_6}, E_{B_7}, E_{B_8}, E_{B_9}, E_{B_{10}}, E_{B_{11}}, E_{B_{12}}, E_{B_{13}}, E_{B_{14}}, E_{B_{15}}),$$

in that order. That is, $E_{B_1} = [a_1]E_0$, $E_{B_2} = [a_2]E_{A'}$, $E_{B_3} = [a_3]E_A$, and so on. Thus, if Peter commits to a curve $E'_{-2} := [\zeta_{-2}]E_{-A'}$ for a secret $\zeta_{-2} \xleftarrow{\$} \mathbb{Z}/n$, he can respond to a challenge to reach E'_{-2} from any of $E_{-A'}$, E_{B_9}, or $E_{B_{13}}$. Similarly, if he commits to $E'_{-1} := [\zeta_{-1}]E_{-A}$ for $\zeta_0 \xleftarrow{\$} \mathbb{Z}/n$, he can reach E'_{-1} from any of E_{-A}, E_{B_5}, E_{B_6}, or $E_{B_{10}}$. If he commits to $E'_0 := [\zeta_0]E_0$ for $\zeta_0 \xleftarrow{\$} \mathbb{Z}/n$, he can reach E'_0 from any of E_0, $E_{\pm B_1}$, $E_{\pm B_7}$, $E_{\pm B_8}$, or $E_{\pm B_{14}}$. If he commits to $E'_1 := [\zeta_1]E_A$ for $\zeta_1 \xleftarrow{\$} \mathbb{Z}/n$, he can reach E'_1 from any of E_A, E_{B_3}, $E_{B_{11}}$, or $E_{B_{12}}$. And if he commits to $E'_2 := [\zeta_1]E_{A'}$ for $\zeta_1 \xleftarrow{\$} \mathbb{Z}/n$, he can reach E'_2 from any of $E_{A'}$, E_{B_2}, E_{B_4}, or $E_{B_{15}}$.

One complicating issue is that the $2C-1$ equivalence classes of curves are likely of different sizes, making the corresponding CSI-FiSh parameters highly heterogeneous. In Example 1, Peter would have to create a SNARK from $2C-1 = 5$ sets of "public keys" derived from the signature Alice created for him: $\{E_{-A'}, E_{B_9}, E_{B_{13}}\}$, $\{E_{-A}, E_{B_5}, E_{B_6}, E_{B_{10}}\}$, $\{E_0, E_{\pm B_1}, E_{\pm B_7}, E_{\pm B_8}, E_{\pm B_{14}}\}$, $\{E_A, E_{B_3}, E_{B_{11}}, E_{B_{12}}\}$, and $\{E_{A'}, E_{B_2}, E_{B_4}, E_{B_{15}}\}$. This means that different hash functions and number of implicit iterations of the challenge-response protocol would have to be adopted for each key equivalence class.

While this feature is theoretically not an issue, it may be a practical hindrance. At the very least, security levels and matching parameters depend on how many curves there are in a set of such "public keys" and the smallest one would be a bottleneck, making the larger ones into sources of inefficiency as they would contribute far too many more curves than needed.

For that reason, we will make a threefold homogenizing assumption, namely:

1. The technique of using twists, which is only available when the starting curve is E_0 (as a consequence of Property 1), is avoided altogether for the CSI-FiSh signatures that constitute the SNARK, though it is still an option for the signatures one wishes to create a SNARK for;
2. CSI-FiSh parameters are chosen so that $2C-1 \mid t$, that is, $\rho := t/(2C-1)$ is an integer;
3. The hash functions $\mathcal{H}_{C+t,t,C} : \mathbb{F}_{p^2}^{C+t} \to \{0, \dots, C-1\}^t$ and $\mathcal{H}^*_{C+t,t,C} : \mathbb{F}_{p^2}^{C+t} \times \mathbb{Z}_N \to \{-C+1, \dots, C-1\}^t$ are redefined so that each value in $\{-C+1, \dots, C-1\}$ occurs exactly ρ times, that is, the hash values are *permutations* of the sequence $(-C+1, \dots, C-1)^t$.

Obtaining hash functions as required in the last item above is not hard. One can start from a simpler hash that maps to $\mathbb{Z}_M$ or $\mathbb{Z}_{M^*}$, where $M := t!/(\rho!)^C$ is the number of permutations of the C values from $\{0, \ldots, C-1\}$ and $M^* := t!/(\rho!)^{2C-1}$ is the number of permutations of the $2C-1$ values from $\{-C+1, \ldots, C-1\}$. From the simpler hash value one can easily obtain a unique permutation from the required sequence via permutation unranking [Ruskey 2003]. As for the simpler hash itself, it can be obtained from a conventional hash function (say, a cryptographic sponge [Bertoni et al. 2008]).

Example 2. Consider the same setting as in Example 1, but now using the three-fold homogenizing assumption above. Then each value from $\{-2, -1, 0, 1, 2\}$ occurs *exactly* $t/(2C-1) = 3$ times on c. For instance, the hash value might be $c = (0, 2, 1, 2, -1, -1, 0, -2, -2, -1, 1, 1, -2, 0, 2)$. This corresponds to the choice

$$(E_0, E_{A'}, E_A, E_{A'}, E_{-A}, E_{-A}, E_0, E_{-A'}, E_{-A'}, E_{-A}, E_A, E_A, E_{-A'}, E_0, E_{A'})$$

of public-key curves for that specific signature, and these in turn are the starting curves of isogenies leading to the signature curves

$$(E_{B_1}, E_{B_2}, E_{B_3}, E_{B_4}, E_{B_5}, E_{B_6}, E_{B_7}, E_{B_8}, E_{B_9}, E_{B_{10}}, E_{B_{11}}, E_{B_{12}}, E_{B_{13}}, E_{B_{14}}, E_{B_{15}}),$$

in that order. That is, $E_{B_1} = [a_1]E_0$, $E_{B_2} = [a_2]E_{A'}$, $E_{B_3} = [a_3]E_A$, and so on. This implies the new equivalence classes are $\{E_{-A'}, E_{B_8}, E_{B_9}, E_{B_{13}}\}$, $\{E_{-A}, E_{B_5}, E_{B_6}, E_{B_{10}}\}$, $\{E_0, E_{B_1}, E_{B_7}, E_{B_{14}}\}$, $\{E_A, E_{B_3}, E_{B_{11}}, E_{B_{12}}\}$, and $\{E_{A'}, E_{B_2}, E_{B_4}, E_{B_{15}}\}$, all with the same number of keys, namely, $t/(2C-1)+1$. Thus, if Peter commits to a curve starting from any of Alice's keys $E_{-A'}$, E_{-A}, E_0, E_A, $E_{A'}$, he can reach the committed curve from any of an equal number of curves, namely, any curve in the equivalence class of the same color. Therefore, the hash functions and number of implicit protocol repetitions can be the same for all of them.

4.5. SNARK of a CSI-FiSh signature

We are now in a position to describe how to obtain a SNARK of a CSI-FiSh signature.

As before, we redefine the message space, this time as $\mathbb{Z}_N^\rho$ with $\rho := t/(2C-1)$, and we replace the message $m := (m_1, \ldots, m_\rho)$ as an argument to the hash function by $B_{t+i} := \psi([m_i]E_0)$ for $1 \le i \le \rho$. Correspondingly, we increase the challenge size from t to $t+\rho$ components. Thus, the CSI-FiSh challenge becomes $(c_1, \ldots, c_t, c_{t+1}, \ldots, c_{t+\rho}) \leftarrow \mathcal{H}_{C+t+\rho,t+\rho,C}(B_1, \ldots, B_t, B_{t+1}, \ldots, B_{t+\rho})$. Now we restate the signature as the tuple $(c_1, \ldots, c_t, B_1, \ldots, B_{t+\rho}) \in \{-C+1, \ldots, C-1\}^t \times \mathbb{F}_{p^2}^{t+\rho}$ constrained to satisfy the above challenge, and we finally combine it with zk-SNARKs of all $a_1, \ldots, a_t, a_{t+\rho}$ where $B_j = \psi([a_j]E_{A_{c_j}})$ for all $1 \le j \le t$, and $a_{t+i} := m_i$ for all $1 \le i \le \rho$.

Notice that all of the above zk-SNARKs can be combined into $2C-1$ equivalence classes, each containing $t/(2C-1)$ curves, with a_{t+1} through $a_{t+\rho}$ already being in a class of their own. So they can use the CSI-FiSh parameters for that number of curves and corresponding number of implicit protocol iterations. They can then be merged together into a single challenge for all equivalence classes of keys, thereby reducing the hash part of the resulting SNARK. We point out here that, since the merged signature is meant to be all-or-nothing (either all challenges are suitably responded to, or else the whole signature is rejected), a smaller number of implicit iterations may be needed than recommended

at the end of Section 4.2. Yet, the number of iterations only decreases modestly with an increased number of keys: for instance, it only decreases by a factor 8 when the number of keys is increased by a factor 2^{14} in the CSIDH-512 parameter set, so this optimization is not expected to have high impact. We leave this possibility for future investigation, and stick to that more conservative choice for the time being.

Interestingly, the ideas underlying this construction will *not* generalize to other isogeny-based signatures. We next discuss the case of the SQISign scheme and why the construction fails for it.

5. The SQISign identification and signature scheme

The SQISign identification scheme works as follows.

- **Setup:** Given a security parameter λ, pick a prime number p and a supersingular elliptic curve $E_0/\mathbb{F}_p$ with known special extremal endomorphism ring $\mathcal{O}_0$. Select an odd λ-bit smooth number $D_c = \prod_{j=1}^{m} \ell_j^{e_j}$ and $D := 2^e$ where e exceeds the diameter of the supersingular 2-isogeny graph.
- **Keygen:** Pick a random isogeny walk $\tau : E_0 \to E_A$ of degree N_τ (a large *prime* number), leading to a random elliptic curve E_A. The public key is $\mathsf{pk} := E_A$, and the secret key is the isogeny $\mathsf{sk} := \tau$.

Remark 2. For λ-bit security, $\deg \tau = N_\tau$ with $\lg N_\tau \approx \lg(p)/4 \approx (\lambda/2)$, computing τ explicitly would incur exponential effort. SQISign avoids the need to do so by using the corresponding ideal I_τ instead, and computing a different isogeny with the same codomain to obtain the public key E_A.

SQISign identification protocol: To prove knowledge of the secret τ (or more exactly a corresponding ideal I_τ), the prover P engages in the following challenge-response protocol with the verifier V.

- **Commitment:** P generates a random (secret) isogeny walk $\psi : E_0 \to E_1$, and sends E_1 to V.
- **Challenge:** V sends the description of a cyclic isogeny $\varphi : E_1 \to E_2$ of degree D_c to P.
- **Response:** From the isogeny $\varphi \circ \psi \circ \hat{\tau} : E_A \to E_2$, P constructs a new isogeny $\sigma : E_A \to E_2$ of degree D such that $\hat{\varphi} \circ \sigma$ is cyclic, and sends σ to V.
- **Verification:** V accepts iff σ is an isogeny of degree D from E_A to E_2 and $\hat{\varphi} \circ \sigma$ is cyclic, otherwise rejects.

SQISign signature protocol: The SQISign scheme requires an especially tailored hash function $\hat{\mathcal{H}}^* : \mathbb{F}_{p^2} \times \{0,1\}^* \to [1 \dots \mu(D_c)]$, where[2] $\mu(D_c) := \prod_{j=1}^{m} \ell_j^{e_j - 1}(\ell_j + 1)$, and also a fixed but arbitrary function $\Phi_{D_c}(E, s)$ mapping $s \in [1 \dots \mu(D_c)]$ to non-backtracking sequences of isogenies of total degree D_c starting at E.

- **Sign:** Given a private key sk and a message m, pick a random (secret) isogeny $\psi : E_0 \to E_1$. Compute $s \leftarrow \hat{\mathcal{H}}^*(j(E_1), m)$ and then the isogeny $\varphi \leftarrow \Phi_{D_c}(E_1, s) : E_1 \to E_2$. From the knowledge of $\mathcal{O}_A$ and of the isogeny $\varphi \circ \psi : E_0 \to E_2$, construct an isogeny $\sigma : E_A \to E_2$ of degree D such that $\hat{\varphi} \circ \sigma$ is cyclic. The signature is the pair $\Sigma := (E_1, \sigma)$.

[2] NB: this is *not* the Möbius function.

- **Verify:** Given a public key pk, a message m and a purported signature $\Sigma := (E_1, \sigma : E_A \to E_2)$, compute $s \leftarrow \mathcal{H}^*(j(E_1), m)$, recover the isogeny $\varphi \leftarrow \Phi_{D_c}(E_1, s) : E_1 \to E_2$. Accept iff $\deg \sigma = D$ and $\hat{\varphi} \circ \sigma$ is cyclic.

5.1. Why SQISign fails to yield a similar SNARK

In contrast with Schnorr, where part of the signature (the z component) has exactly the same nature as a private signing key, or CSI-FiSh signatures, where the corresponding part of the signature has the same nature as a collection of private signing keys, SQISign keys and signatures are in disjoint and incompatible categories of their own.

Specifically, a SQISign private key τ must be an isogeny of large *prime* degree (otherwise it would leak enough information to reveal it to an attacker), while the signature σ must be an isogeny of large *smooth composite* degree (otherwise signing and verifying would be rendered infeasible).

At this time, obtaining a SNARK of a SQISign signature as efficient as a SNARK for other Fiat-Shamir signature schemes like CSI-FiSh is left as a research problem.

6. A work space (and time) improvement for CSI-FiSh (and CSIDH)

Any ideal $\mathfrak{g}$ in the class group can be written as $\mathfrak{g} = \prod_i \mathfrak{l}_i^{g_i}$ for efficiently computable ideals $\mathfrak{l}_i := \langle \ell_i, \pi - 1 \rangle$ and small integers g_i. The original CSI-FiSh strategy to compute $\mathfrak{g}^a$ (for a large, at least 512-bit integer a) is to decompose it as $\mathfrak{g}^a = \prod_i \mathfrak{l}_i^{e_i}$ for small integers e_i obtained via lattice reduction. Notice that the obvious decomposition $e_i = a \cdot g_i$ involves exceedingly large exponents, and *no polynomial-time exponentiation algorithm is known* for the class group (in fact, this is one of the reasons that make this cryptosystem quantum-resistant, albeit in the sense of superpolynomial complexity), so this approach is infeasible. The starting point for lattice reduction in the original CSI-FiSh scheme is the vector $[a, 0, \ldots, 0]$ (of dimension n for some fixed n), corresponding to a decomposition where the initial exponents are $e_1 = a$ for $\mathfrak{g} := \mathfrak{l}_1 = \langle 3, \pi - 1 \rangle$ and $e_i = 0$ for $i > 1$, since the ideal $\langle 3, \pi - 1 \rangle$ alone is known to generate the whole class group (for practical CSI-FiSh parameters).

Lattice reduction, conducted using Babai's nearest-plane method [Babai 1986] (Algorithm 1, where $\Lambda(B)$ stands for the lattice generated by B) and mostly ad-hoc refinements, can bring the ideal exponents down to feasible sizes, say, 1-byte values, but this typically forces the floating-point precision required for the rounding step indicated on line 3 and the projection step indicated on line 4 to be substantially larger than the size $|\mathrm{Cl}(\mathcal{O})|$, which also increases the running time by an even larger factor. For instance, for the CSIDH-512 parameters, where $|\mathrm{Cl}(\mathcal{O})|$ is about 258 bits long, the official CSI-FiSh implementation sets the precision to 1000 bits per limb, or about *18-fold* worse than simple precision (53 bits per limb). Specifically, the precomputed LLL-reduced basis $B = (\mathbf{u}_j \mid i \in [n])$ consists of n vectors in dimension n, or n^2 floating-point values, and the precomputed norms $\langle \mathbf{u}_j, \mathbf{u}_j \rangle$ take n more such values, where $n = 74$ for CSIDH-512. Hence, this yields $n(n+1)$ 1000-bit precomputed values, occupying 693750 bytes overall. Updated CSIDH parameters are likely to noticeably increase both n and $|\mathrm{Cl}(\mathcal{O})|$, and hence the space requirements.

Fortunately, we can do much better. Let b and m be such that all integers modulo $|\mathrm{Cl}(\mathcal{O})|$ can be expressed with no more than m digits in base b, i.e., m is the smallest

Algorithm 1 Babai's nearest-plane algorithm

INPUT: LLL-reduced basis $B = (\mathbf{u}_j \mid 0 \le j < n)$ and vector $\mathbf{v}$.
OUTPUT: $\mathbf{w} \in \Lambda(B)$ such that $|\mathbf{v} - \mathbf{w}| \le 2^{d/2}\mathrm{dist}(\mathbf{v}, \Lambda(B))$.

$\mathbf{w} \leftarrow \mathbf{v}$
for $j \leftarrow n-1$ **down to** 0 **do**
  $c_j \leftarrow \lfloor \langle \mathbf{w}, \mathbf{u}_j \rangle / \langle \mathbf{u}_j, \mathbf{u}_j \rangle \rceil$ ▷ rounding step
  $\mathbf{w} \leftarrow \mathbf{w} - c_j \mathbf{u}_j$ ▷ projection step
end for
$\mathbf{w} \leftarrow \mathbf{v} - \mathbf{w}$
return $\mathbf{w}$

integer such that $b^m \ge |\mathrm{Cl}(\mathcal{O})|$. Our space-saving technique consists of precomputing lattice-reduced vectors $\mathfrak{g}^{b^i} = \prod_j \mathfrak{l}_j^{e_{i,j}}$ for small exponents $e_{i,j}$, all $0 \le i < m$, and all $1 \le j \le n$. Since $\mathfrak{g} = \mathfrak{l}_1$ corresponds to the vector of exponents $(e_{0,j}) = [1, 0, \ldots, 0]$, $\mathfrak{g}^b$ maps simply to $(e_{1,j}) = [b, 0, \ldots, 0]$ if b itself is small. In turn, $\mathfrak{g}^{b^2}$ maps to $[b^2, 0, \ldots, 0]$ which, albeit longer, is still short enough that lattice reduction shrinks it right away to a vector $(e_{2,j}) = [e_{2,1}, \ldots, e_{2,n}]$, where each $e_{2,j}$ is in the same ballpark as b (rather than b^2). From then on, computing $\mathfrak{g}^{b^i}$ for each i consists of multiplying the previous vector by b, yielding $[b \cdot e_{i-1,1}, \ldots, b \cdot e_{i-1,n}]$ with coefficients in the ballpark of b^2, and repeating the lattice reduction step to obtain $[e_{i,1}, \ldots, e_{i,n}]$, where each e_{ij} is again in the same ballpark as b. Thus, if b is properly chosen (say, $b = 16$), we never expect to have vector components larger than about 8 bits each, and a lattice reduction algorithm such as Babai's will be especially fast and effective to bring them down to about 4 bits each again. Algorithm 2 summarizes this process.

Algorithm 2 Precomputation of the powers $\mathfrak{g}^{b^i}$ for $0 \le i \le m$, $\mathfrak{g} := \langle 3, \pi - 1 \rangle$

INPUT: m is the smallest integer such that $b^m \ge |\mathrm{Cl}(\mathcal{O})|$.
OUTPUT: matrix $(e_{i,j})$ such that $\mathfrak{g}^{b^i} = \prod_j \mathfrak{l}_j^{e_{i,j}}$, for all $0 \le i < m$ and $1 \le j \le n$.

$(e_{0,j}) \leftarrow [1, 0, \ldots, 0]$ ▷ $\mathfrak{g}^{b^0} = \mathfrak{g}$
for $i \leftarrow 1$ **to** $m-1$ **do**
  $(e_{i,j}) \leftarrow [b \cdot e_{i-1,1}, \ldots, b \cdot e_{i-1,n}]$ ▷ $\mathfrak{g}^{b^i} = (\mathfrak{g}^{b^{i-1}})^b$
  Apply Babai's nearest-plane algorithm to reduce $(e_{i,j})$.
end for
return $(e_{i,j})$

Given this precomputation, we can express $a = \sum_i a_i b^i$ as an m-digit number in base b, $0 \le a_i < b$. Thus $\mathfrak{g}^a = \mathfrak{g}^{\sum_i a_i b^i} = \prod_i \left(\mathfrak{g}^{b^i}\right)^{a_i} = \prod_i \left(\prod_j \mathfrak{l}_j^{e_{i,j}}\right)^{a_i} = \prod_j \mathfrak{l}_j^{\sum_i a_i e_{i,j}}$. The exponents $\sum_i a_i e_{i,j}$, while not fully reduced, are way smaller than the original a, and for $b = 16$ they can be reduced via Babai's method and eventual refinements with simple floating-point precision. This procedure is summarized[3] in Algorithm 3.

Shrinking all this down to single-precision as we propose reduces the CSIDH-512 space requirements to 39174 bytes, or no more than 46805 bytes in a more straightfor-

[3]The same ad-hoc refinements as in the original CSI-FiSh algorithm are possible after the nearest-plane reduction. These are omitted in Algorithm 3 for the sake of brevity.

Algorithm 3 Single-precision decomposition into the $\mathfrak{l}_j$ ideal base

INPUT: $a \in \mathbb{Z}_N$; precomputed matrix $(e_{i,j})$ such that $\mathfrak{g}^{b^i} = \prod_j \mathfrak{l}_j^{e_{i,j}}$.
OUTPUT: vector $(e_j) := [e_1, \ldots, e_n]$ of small components such that $\mathfrak{g}^a = \prod_j \mathfrak{l}_j^{e_j}$.

1: Write $a = \sum_i a_i b^i$ as an m-digit number in base b.
2: $(e_j) \leftarrow [\sum_i a_i e_{i,1}, \ldots, \sum_i a_i e_{i,n}]$
3: Apply Babai's nearest-plane algorithm to reduce (e_j).
4: **return** (e_j)

ward implementation with 64-bit floating-point words and individual bytes for pairs of $(e_{i,j})$ components, and leads to much faster lattice reduction (about 81-fold faster, assuming Karatsuba multiplication for extended-precision floating-point arithmetic). In fact, the required precision may be even smaller in practice, opening the possibility of doing away with floating point arithmetic entirely. This would, however, require emulating rational arithmetic, making the implementation somewhat more involved. The kind of platform where this might be an advantage (an embedded or otherwise highly constrained processor) is, however, hardly suitable for the currently available techniques to implement CSIDH, which notoriously require intensive computational capabilities, and related cryptosystems like CSI-FiSh. For this reason, we restrained from further attempts at entirely eliminating floating-point arithmetic.

7. Conclusions

We have described the construction of a SNARK of CSI-FiSh digital signatures using signatures of the same kind. Our proposal is inspired on a SNARK of conventional Schnorr signatures, yet non-trivial given the peculiar adoption of multiple private keys in the CSI-FiSh scheme, as well as the somewhat surprising fact that the same idea fails for SQISign, another isogeny-based signature scheme. We have also proposed a technique to reduce the work space needed to implement the CSIDH framework required by CSI-FiSh signatures by a factor of at least 18 (or more for higher security levels). The same technique also substantially reduces the associated processing time, by a factor that can reach almost two orders of magnitude, depending on the implementation.

Acknowledgements. Marcos A. Simplicio Jr. was funded by CNPq (research productivity grant 304643/2020-3) and by Ripple via the *University Blockchain Research Initiative*.

References

Au, M. H., Susilo, W., and Mu, Y. (2006). Constant-size dynamic k-TAA. In *Security and Cryptography for Networks*, pages 111–125, Berlin, Heidelberg. Springer. DOI: 10.1007/11832072_8.

Babai, L. (1986). On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6:1–13. DOI:10.1007/BF02579403.

Bertoni, G., Daemen, J., Peeters, M., and Van Assche, G. (2008). On the indifferentiability of the sponge construction. In *Advances in Cryptology – EUROCRYPT 2008*, pages 181–197, Berlin, Heidelberg. Springer. DOI:10.1007/978-3-540-78967-3_11.

Beullens, W., Kleinjung, T., and Vercauteren, F. (2019). CSI-FiSh: Efficient isogeny based signatures through class group computations. In *Advances in Cryptology –*

ASIACRYPT 2019, pages 227–247, Cham. Springer International Publishing. DOI:10.1007/978-3-030-34578-5_9.

Camenisch, J. (1998). *Group signature schemes and payment systems based on the discrete logarithm problem*. PhD thesis, ETH Zürich. DOI:10.3929/ethz-a-001923735.

Camenisch, J. and Lysyanskaya, A. (2004). Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology – CRYPTO 2004*, pages 56–72, Berlin, Heidelberg. Springer. DOI:10.1007/978-3-540-28628-8_4.

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

Castryck, W., Lange, T., Martindale, C., Panny, L., and Renes, J. (2018). CSIDH: An efficient post-quantum commutative group action. In *Advances in Cryptology – ASIACRYPT 2018*, pages 395–427, Cham. Springer International Publishing. DOI:10.1007/978-3-030-03332-3_15.

De Feo, L., Kohel, D., Leroux, A., Petit, C., and Wesolowski, B. (2020). SQISign: Compact post-quantum signatures from quaternions and isogenies. In *Advances in Cryptology – ASIACRYPT 2020*, pages 64–93, Cham. Springer International Publishing. DOI:10.1007/978-3-030-64837-4_3.

Fiat, A. and Shamir, A. (1987). How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology — CRYPTO' 86*, pages 186–194, Berlin, Heidelberg. Springer. DOI:10.1007/3-540-47721-7_12.

Galindo, D. and Garcia, F. (2009). A Schnorr-like lightweight identity-based signature scheme. In *Progress in Cryptology – AFRICACRYPT 2009*, pages 135–148, Berlin, Heidelberg. Springer.

Ganesh, C., Nitulescu, A., and Soria-Vazquez, E. (2021). Rinocchio: SNARKs for ring arithmetic. Cryptology ePrint Archive, Paper 2021/322. https://eprint.iacr.org/2021/322.

Groth, J. (2010). Short pairing-based non-interactive zero-knowledge arguments. In *Advances in Cryptology - ASIACRYPT 2010*, pages 321–340, Berlin, Heidelberg. Springer. DOI:10.1007/978-3-642-17373-8_19.

Montgomery, P. L. (1987). Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of computation*, 48(177):243–264. DOI:10.1090/S0025-5718-1987-0866113-7.

Ruskey, F. (2003). Combinatorial generation. Draft book. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.93.5967.

Schnorr, C. P. (1990). Efficient identification and signatures for smart cards. In *Advances in Cryptology — CRYPTO '89 Proceedings*, pages 239–252, New York, NY. Springer New York. DOI:10.1007/0-387-34805-0_22.