Research Article

Wissam Ghantous, Federico Pintore, and Mattia Veroni*

Efficiency of SIDH-based signatures (yes, SIDH)

Abstract: In this note, we assess the efficiency of a supersingular isogeny Diffie-Hellman (SIDH)-based digital signature built on a *weaker* variant of a recent identification protocol proposed by Basso et al. Despite the devastating attacks against (the mathematical problem underlying) SIDH, this identification protocol remains secure, as its security is backed by a different (and more standard) isogeny-finding problem. We conduct our analysis by applying some known cryptographic techniques to decrease the signature size by about 70% for all parameter sets (obtaining signatures of approximately 21 kB for SIKEp434). Moreover, we propose a minor optimisation to compute many isogenies in parallel from the same starting curve. Our assessment confirms that determining the most efficient methods for isogeny-based signature schemes, including optimisations such as those presented in this paper, is still a open problem, with much more work to be done.

Keywords: post-quantum cryptography, isogeny-based cryptography, digital signature

MSC 2020: 94A62

1 Introduction

Isogenies between supersingular elliptic curves have been used to construct cryptosystems supposed to be secure even in the presence of quantum attackers. The family of such cryptosystems is named isogeny-based cryptography, and its most appealing members enjoy short keys and ciphertexts. At the time of writing, the most prominent example of this attractive feature is the digital signature SQISign [1], which is the most compact post-quantum signature scheme. On the other hand, isogeny-based cryptosystems incur high execution times, with SQISign making no exception (despite the recent improvements in [2]). The most promising results in terms of computational efficiency have been obtained for the key-exchange supersingular isogeny Diffie-Hellman (SIDH) [3] and the corresponding key-encapsulation mechanism supersingular isogeny key-encapsulation (SIKE) [4]. However, not all schemes built on SIDH share the same quality. An example is the SIDH-based digital signatures proposed in [5,6], for which no substantial amelioration has appeared since their publication. Nevertheless, they represented an alternative starting point for a practical isogeny-based digital signature building on existing schemes. However, three major classical attacks [7–9] were devised in 2022, which make SIDH, SIKE, and most of the SIDH-based cryptosystems – including the signature schemes based on SIDH mentioned above – completely insecure. As a consequence, SQISign and Sea-Sign/CSI-FiSh[1] were, until recently, the only isogeny-based digital signature schemes still secure.

Fortunately, two isogeny-based Σ-protocols that were recently proposed have restored the family of SIDH-based digital signatures (and non-interactive zero-knowledge proofs). The first one [13, Sec. 5.3] – denoted by Σ_{wSIDH} in the following – was originally designed for the SIDH setting, while for the second one [14, Sec. 4] –

* **Corresponding author: Mattia Veroni,** NTNU – Norwegian University of Science and Technology, Trondheim, Norway, e-mail: mattia.veroni@ntnu.no

Wissam Ghantous: Mathematical Institute, University of Oxford, Oxford OX1 2JD, UK, e-mail: wissam.ghantous@maths.ox.ac.uk

Federico Pintore: Department of Mathematics, University of Bari, 70121 Bari BA, Italy, e-mail: federico.pintore@uniba.it

which we denote by $\Sigma_{\mathsf{SECUER}}^{\mathsf{base}}$ – the SIDH parameters are (probably the most) favourable in terms of practical efficiency, despite it being designed for a general scenario. Consequently, their implementations can take advantage of the optimised implementations for determining and evaluating isogenies in the SIDH configuration. Even so, both Σ_{wSIDH} and $\Sigma_{\mathsf{SECUER}}^{\mathsf{base}}$ are not affected by the attacks in [7–9] and hence can still be the base for constructing digital signature schemes as well as non-interactive zero-knowledge proofs (NIZKPs in short).

1.1 Our contribution

In this note, we assess the compactness and efficiency that can be currently reached by digital signatures (and NIZKPs) based on the SIDH setting. In doing so, we restrict our attention to a digital signature built on a slightly weaker variant of $\Sigma_{\mathsf{SECUER}}^{\mathsf{base}}$. We talk about weaker variant because $\Sigma_{\mathsf{SECUER}}^{\mathsf{base}}$ was designed to satisfy statistical honest-verifier zero-knowledge, which is not necessary for our case study. The variant we consider – denoted by Σ_{SEC} – only achieves computational honest-verifier zero-knowledge, but it allows for shorter isogenies, therefore a better efficiency. There are three main reasons to work with Σ_{SEC} instead of other SIDH-based Σ-protocols. First, a similar assessment focused on Σ_{wSIDH} was recently conducted in [15]. Even more importantly, despite the similarities between Σ_{wSIDH} and Σ_{SEC}, the latter has a more lightweight design, leading to smaller transcripts and faster execution times. Last but not least, the optimisations we apply to (t parallel executions of) Σ_{SEC} are applicable also to $\Sigma_{\mathsf{SECUER}}^{\mathsf{base}}$ and are relevant to any application of these Zero-Knowledge Proof systems. In fact, using the Fiat-Shamir transform to remove interactivity from a Σ-protocol has applications beyond digital signatures. For example, $\Sigma_{\mathsf{SECUER}}^{\mathsf{base}}$ has been used to prove random generation of supersingular curves of unknown endomorphism rings in a distributed and trusted manner [14].

We conduct our analysis by applying some known signature-shortening techniques. By doing so, we can shorten the signatures produced by means of Σ_{SEC} by approximately 69%. For example, we obtain signatures of approximately 21 kB for the parameter set SIKEp434. In addition, we propose minor optimisations to compute many isogenies in parallel from the same starting curve.

One of the techniques we consider to shorten the signatures is the *unbalanced challenge space technique*, firstly proposed in the study by Beullens et al. [16] for a Σ-protocol obtained by running parallel executions of a base Σ-protocol with soundness error 1/2. In this work, however, we apply it to a base Σ-protocol with soundness error 2/3, which requires a non-trivial generalisation of the original proposal. In fact, to determine the number of parallel executions which are required in such case for an unbalanced challenge space, we deduce some combinatorial results. Our findings can be readily applied to every possible soundness error of the base Σ-protocol, and therefore are of independent interest.

Our assessment confirms that the problem of designing a practical isogeny-based signature scheme remains largely open. Nonetheless, the proposed optimisations can be applied to the distributed trusted-setup protocol [14, Sec. 5] built on top of $\Sigma_{\mathsf{SECUER}}^{\mathsf{base}}$ to collaboratively produce a random supersingular elliptic curve whose endomorphism ring is hard to compute even for the involved parties. Determining the best techniques to design secure and efficient isogeny-based signatures, against which future protocols can be benchmarked, is still a quite new and very open problem.

---

1 Sea-Sign [10] is an isogeny-based digital signature scheme which works with isogenies and elliptic curves over prime fields. CSI-FiSh [11] is an optimisation of Sea-Sign for a specific set of parameters, named CSIDH-512. It is worth noticing that the security provided by CSIDH-512 is still an active area of research [12], and instantiating SeaSign with *larger* parameters leads to long execution times.

1.2 Related work

The SIDH-based digital signature scheme proposed in the study by Galbraith et al. [5] produces signatures of approximately 12 kB when targeting 128 bits of classical security. For the same security target, the signature scheme in the study by Chi-DomAηnguez et al. [17] (deduced from a different SIDH-based Σ-protocol proposed in the study by Chi-DomAηnguez et al. [13, Sec. 6]) outputs signatures of approximately 61 kB. Both these signature schemes are no longer secure after the cryptanalytic attacks against SIDH. The analysis conducted in the study by Chi-Domínguez et al. [15] on a still-secure digital signature built on top of Σ_{wSIDH} achieves signatures of size approximately 74 kB for the parameter set SIKEp434. Note that the protocol in the study by Galbraith et al. [5] and Σ_{wSIDH} in the study by Chi-DomAηnguez et al. [13, Sec. 5] are 2-special sound. The protocols in the study by Chi-DomAηnguez et al. [13, Sec. 6] is instead 3-special sound.

1.3 Roadmap

This article is organised as follows. In Section 2, we recall some cryptographic preliminaries and we provide a description of the Σ-protocol Σ_{SEC}, which is a diminished version of $\Sigma_{\mathsf{SECUER}}^{\mathsf{base}}$ from the study by Basso et al. [14, Sec. 4]. By applying the Fiat-Shamir transform [18] on Σ_{SEC}, an SIDH-based signature scheme $\mathsf{DS}_{\mathsf{SEC}}$ is obtained. In Section 3, we apply some optimisation techniques to reduce the size of the signatures produced by $\mathsf{DS}_{\mathsf{SEC}}$: commitment recoverability (Section 3.1), response compression (Section 3.2), and seed trees (Section 3.3). In Section 3.4, unbalanced challenge spaces are also taken into account. We conclude the section highlighting the overall gain in applying these optimisations with respect to the original scheme. In Section 4, we suggest two optimisations for the computation of several isogenies of the same degree from the same starting curve. They consists in a pre-computation of repeated initial steps (Section 4.1) and in the parallelisation of kernel generators computation (Section 4.2). Section 5 presents some closing remarks.

2 Preliminaries

In this section, we list some definitions and results regarding isogenies between supersingular elliptic curves, Σ-protocols, and digital signatures. We then detail the Σ-protocol Σ_{SEC}, a diminished variant of $\Sigma_{\mathsf{SECUER}}^{\mathsf{base}}$ from the study by Basso et al. [14, Sec. 4].

Remark 1. In the following, commitment schemes (C) and pseudorandom number generators (Expand) are instantiated with a hash function modelled as a random oracle O. We always assume the input domain of the random oracle is appropriately separated when instantiating different cryptographic primitives. With an abuse of notation, we will write $O(\mathsf{Expand}||\cdot)$ instead of $\mathsf{Expand}(\cdot)$ and $O(\mathsf{Com}||\cdot)$ instead of $\mathsf{C}(\cdot)$ to make the usage of the random oracle explicit. Here, we identify Expand and Com with unique strings.

2.1 Supersingular elliptic curves, isogenies, and hardness assumptions

We refer the reader to [19,20] for a more detailed introduction to the topic.

Let q be a power of a prime $p \geq 5$, and let $\mathbb{F}_q$ be a finite field with q elements. An isogeny $\varphi : E \longrightarrow E'$ between two elliptic curves E and E' over $\mathbb{F}_q$, with points at infinity denoted by 0_E and $0_{E'}$ respectively, is a non-constant regular rational map mapping $\varphi(0_E)$ into $0_{E'}$. Every isogeny φ can be written in its polynomial form $(F_1(x)/F_2(x), yG_1(x)/G_2(x))$, where F_1, F_2, G_1, and G_2 are polynomials over the algebraic closure of $\mathbb{F}$, F_1 is coprime with F_2, and G_1 is coprime with G_2. The isogeny φ is said to be defined over $\mathbb{F}_{q^k}$ if the coefficients

of the above polynomials are contained in $\mathbb{F}_{q^k}$; in this case, we say that E, E' are isogenous over $\mathbb{F}_{q^k}$. Tate's theorem states that E, E' are isogenous over $\mathbb{F}_{q^k}$ if and only if $\#E(\mathbb{F}_{q^k}) = \#E'(\mathbb{F}_{q^k})$.

An invertible isogeny is an isomorphism; in addition, if its domain and image coincide, it is an endomorphism. The set of all endomorphisms of an elliptic curve E together with the zero map form a ring under pointwise addition and composition, called the endomorphism ring of E and denoted by $\mathsf{End}(E)$. If $\mathsf{End}(E)$ is not commutative, then E is said to be supersingular. Every supersingular elliptic curve defined over $\mathbb{F}_{p^k}$ for some $k \in \mathbb{N}$ is isomorphic to an elliptic curve defined over $\mathbb{F}_{p^2}$. The degree $\deg(\varphi)$ of an isogeny φ is the maximum among $\{\deg(F_1), \deg(F_2)\}$; we say that φ is a d-isogeny. Two elliptic curves E and E' are d-isogenous if there exists an isogeny $\varphi : E \longrightarrow E'$ of degree d. Given a power q of a prime $p > 5$ and a prime number $\ell \neq p$, we denote by $\mathcal{G}_q(\ell)$ the graph whose vertices are $\mathbb{F}_q$-isomorphism classes of supersingular elliptic curves over $\mathbb{F}_q$ and whose edges are equivalence classes of ℓ-isogenies (two isogenies are in the same class if they have the same kernel). The composition of two isogenies of degrees d_1 and d_2 is an isogeny of degree $d_1 d_2$.

The kernel of an isogeny is finite, and its size is equal to the degree of the isogeny itself if the isogeny is separable. Vice versa, if H is a finite subgroup of an elliptic curve E, then the elliptic curve E/H and a separable isogeny $\psi : E \longrightarrow E'$ of kernel $\ker(\psi) = H$ are unique (modulo isomorphism). Both E/H and ψ can be computed with complexity $O(\#H)$ using Velu's formulas. We say that φ is a cyclic isogeny when $\ker(\varphi)$ is a cyclic group. Given $\ell \in \mathbb{N}$, we denote by $E[\ell]$ the ℓ-torsion subgroup $\{P \in E | [\ell]P = 0_E\}$ of E. When ℓ and p are relatively prime, $E[\ell] \simeq (\mathbb{Z}/\ell\mathbb{Z}) \times (\mathbb{Z}/\ell\mathbb{Z})$.

2.2 Σ-protocols

Let X and Y be two sets whose sizes depend on a security parameter λ. Then $\mathcal{R} \subset X \times Y$ is a *polynomially computable binary relation over X and Y* if, for any $(\mathsf{x}, \mathsf{w}) \in X \times Y$, whether $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}$ can be decided in time $\mathrm{poly}(|\mathsf{x}|)$. If $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}$, we call w a *witness* for the *statement* x. The *language* corresponding to $\mathcal{R}$ is $\mathcal{L}_{\mathcal{R}} = \{\mathsf{x} \in X | \exists\, \mathsf{w} \in Y : (\mathsf{x}, \mathsf{w}) \in \mathcal{R}\}$.

A Σ-*protocol* for a polynomially-computable binary relation $\mathcal{R}$ is a public-coin three-move interactive protocol between a prover and a verifier. Informally, a prover can demonstrate knowledge of a valid witness for a certain statement without revealing any information about the witness itself. Below, we define a relaxed version of sigma protocols where the special-soundness extractor only extracts a witness for a slightly larger relation $\tilde{\mathcal{R}}$, with $\mathcal{R} \subseteq \tilde{\mathcal{R}}$. Furthermore, the definition is given in the random oracle model, i.e. prover and verifier have access to a random oracle O. We may occasionally omit the superscript O when the meaning is clear from the context.

Definition 1. (Σ-protocols) A Σ-protocol S for polynomially-computable binary relations $\mathcal{R} \subseteq \tilde{\mathcal{R}}$ consists of five polynomial-time algorithms $(\mathsf{Gen}, \mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2))$ with oracle-access, where V_2 is deterministic, $\mathsf{Gen}, \mathsf{P}_1, \mathsf{P}_2$, and V_1 are probabilistic, and $\mathsf{P}_1, \mathsf{P}_2$ share states. We denote by ComSet, ChSet, and ResSet the commitment space, challenge space, and response space, respectively. In the random-oracle model, the protocol goes as follows:

- The key-generation algorithm $\mathsf{Gen}^O(1^\lambda)$ takes the security parameter 1^λ as input, and outputs a statement-witness pair $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}$.
- On input $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}$, the prover computes $\mathsf{com} \leftarrow \mathsf{P}_1^O(\mathsf{x}, \mathsf{w})$ and sends the commitment com to the verifier.
- The verifier runs $\mathsf{ch} \leftarrow \mathsf{V}_1^O(\mathsf{com})$ to obtain a random challenge, and sends ch to the prover.
- Given ch, the prover computes $\mathsf{resp} \leftarrow \mathsf{P}_2^O(\mathsf{x}, \mathsf{w}, \mathsf{com}, \mathsf{ch})$ and returns the response resp to the verifier.
- The verifier runs $\mathsf{V}_2^O(\mathsf{x}, \mathsf{com}, \mathsf{ch}, \mathsf{resp})$ and outputs 1 if they accept, 0 otherwise.

Here O is modelled as a random oracle. Moreover, a transcript $(x, com, ch, resp) \in X \times ComSet \times ChSet \times ResSet$ of the protocol is said to be valid (relative to x) in case $V_2(x, com, ch, resp)$ outputs 1.

We require the following properties of a Σ-protocol:

• **Correctness:** All honestly generated transcripts must be valid. Formally, it is required that

$$\Pr\left[V_2^O(x, com, ch, resp) = 1 \,\middle|\, \begin{array}{l} (x, w) \leftarrow KeyGen^O(1^\lambda) \\ com \leftarrow P_1^O(x, w), \\ ch \leftarrow V_1^O(com), \\ resp \leftarrow P_2^O(x, com, w, ch) \end{array}\right] = 1.$$

• **Relaxed κ-special soundness:** There exists a polynomial-time *extraction algorithm* Ex such that, given any κ valid transcripts $(x, com, ch_1, resp_1), \cdots, (x, com, ch_\kappa, resp_\kappa)$ relative to the same statement $x \in \mathcal{L}_\mathcal{R}$, with the same commitment com and κ distinct challenges $ch_1, \cdots, ch_\kappa$, outputs w such that $(x, w) \in \tilde{\mathcal{R}}$ (note that Ex is only required to recover a witness in $\tilde{\mathcal{R}} \supseteq \mathcal{R}$).

• **Statistical and computational honest-verifier zero-knowledge (HVZK):** Within this definition, we allow the adversary, the prover, and the simulator to make queries to a common random oracle O. We say the Σ-protocol is *statistically HVZK* if there exists a probabilistic polynomial-time (PPT) simulator algorithm Sim^O such that, for any $(x, w) \in \mathcal{R}$, any honestly chosen $ch \in ChSet$ and any computationally unbounded adversary $\mathcal{A}$ that makes at most a polynomial number of queries to O, we have

$$\Pr\left[\mathcal{A}^O(com, ch, resp) = 1 \,\middle|\, \begin{array}{l} com \leftarrow P_1^O(x, w); \\ resp \leftarrow P_2^O(x, com, w, ch) \end{array}\right] -$$
$$\Pr[\mathcal{A}^O(com, ch, resp) = 1 \,|\, (com, resp) \leftarrow Sim^O(x, ch)] = negl(\lambda).$$

If the aforementioned relation holds only for computationally bounded adversaries, the protocol is said to be *computationally HVZK*.

2.3 Digital signatures

Below we recall the definition of digital signature schemes, correctness, and unforgeability.

Definition 2. (Digital signature schemes) A digital signature scheme DS consists of three algorithms (KeyGen, Sign, Vrfy) defined as follows:

• $(vk, sk) \leftarrow KeyGen(1^\lambda)$: On input a security parameter λ, the key-generation algorithm outputs a pair of verification and signing keys (vk, sk).

• $\sigma \leftarrow Sign(sk, M)$: On input a signing key sk and a message M, the signing algorithm outputs a signature σ.

• $b \in \{0, 1\} \leftarrow Vrfy(vk, M, \sigma)$: On input a verification key vk, a message M, and a signature σ, the verification algorithm outputs 1 (accept) or 0 (reject).

2.3.1 Correctness

For every security parameter $\lambda \in \mathbb{N}$ and every message M, a signature scheme is *correct* if the following holds:

$$\Pr\left[\mathsf{Vrfy}(\mathsf{vk}, \mathsf{M}, \sigma) = 1 \,\middle|\, \begin{array}{l} (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda), \\ \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{M}) \end{array}\right] = 1.$$

2.3.2 Security

We define *existential unforgeability under chosen message attack*(EUF-CMA) with the following game between an adversary $\mathcal{A}$ and a challenger.

Setup: The challenger runs $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and provides the adversary $\mathcal{A}$ with the verification key vk. It also prepares an empty set $\mathcal{S} = \varnothing$.

Signing queries: The adversary $\mathcal{A}$ may adaptively submit messages M to the challenger. The challenger responds with $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{M})$ to $\mathcal{A}$'s query on a message M and updates the set $\mathcal{S} \leftarrow \mathcal{S} \cup \{(\mathsf{M}, \sigma)\}$.

Output: Finally, $\mathcal{A}$ outputs a forgery (M^*, σ^*). We say that the adversary $\mathcal{A}$ wins if $(\mathsf{M}^*, \cdot) \notin \mathcal{S}$ and $\mathsf{Vrfy}(\mathsf{vk}, \mathsf{M}^*, \sigma^*) = 1$.

We then say that the signature scheme DS is EUF-CMA-*secure* if, for all PPT adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in winning the above game is negligible in the security parameter λ:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{EUF\text{-}CMA}}(\lambda) \coloneqq \Pr[\mathcal{A} \text{ wins}] = \mathrm{negl}(\lambda).$$

Via the Fiat-Shamir transform [18], a Σ-protocol S for a binary relation $\mathcal{R}$ can be turned into a digital signature scheme. The resulting scheme *FS*(S) differs from S in the challenge computation, as the challenge is set equal to the digest $H(\mathsf{com}, \mathsf{M})$ – where M is the message to sign and H a hash function - instead of being randomly produced by the verifier. If the binary relation $\mathcal{R}$ is based on a hard problem, then *FS*(S) can be proved EUF-CMA secure.

2.4 The Σ_{SEC} protocol

In this section, we describe the Σ-protocol Σ_{SEC}, a weaker variant of $\Sigma_{\mathsf{SECUER}}^{\mathsf{base}}$ from the study by Basso et al. [14, Section 4] (see Remark 2 for the differences between the two protocols).

For every possible value of the security parameter λ, p will denote a prime of the form $p = \ell_1^{e_1}\ell_2^{e_2}f \pm 1$ (where ℓ_1, ℓ_2 are small primes such that $\ell_1^{e_1} \approx \ell_2^{e_2}$ and $f \in \mathbb{N}$ is a small cofactor), E_0 is a fixed supersingular elliptic curve over $\mathbb{F}_{p^2}$ such that $\#E_0(\mathbb{F}_{p^2}) = (\ell_1^{e_1}\ell_2^{e_2}f)^2$ (when considering SIKE parameters, we will take $E_0 : y^2 = x^3 + 6x^2 + x$), and $\{P_1, Q_1\}$ and $\{P_2, Q_2\}$ basis for $E_0[\ell_1^{e_1}]$ and $E_0[\ell_2^{e_2}]$, respectively. Then, the tuple $\mathsf{pp} = (p, \ell_1, \ell_2, e_1, e_2, f, E_0, P_1, Q_1, P_2, Q_2)$ forms the public parameter for the protocol.

The Σ-protocol Σ_{SEC} consists of five oracle-calling algorithms $(\mathsf{Gen}, \mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2))$, where:

- $(E_1, \varphi) \leftarrow \mathsf{Gen}(1^\lambda)$: On input a security parameter, the key-generation algorithm uniformly samples s from $\mathbb{Z}/\ell_1^{e_1}\mathbb{Z}$ and computes the cyclic isogeny $\varphi : E_0 \to E_1 \coloneqq E_0/\langle P_1 + [s]Q_1\rangle$ having $\langle P_1 + [s]Q_1\rangle$ as kernel. It returns the statement-witness pair (E_1, φ).

- $\mathsf{com} \leftarrow \mathsf{P}_1(E_1, \varphi)$: Given a statement E_1 and a corresponding witness φ, the prover uniformly samples r from $\mathbb{Z}/\ell_2^{e_2}\mathbb{Z}$ and computes the point $R = P_2 + [r]Q_2$ – whose order is $\ell_2^{e_2}$ – and the elliptic curves $E_2 = E_0/\langle R\rangle$ and $E_3 = E_1/\langle\varphi(R)\rangle$. Then, it uniformly samples b_2, b_3 from $\{0, 1\}^\lambda$ and commits to E_2 and E_3 computing $\mathsf{com}_1 \leftarrow O(\mathsf{Com}\|E_2\|b_2)$ and $\mathsf{com}_2 \leftarrow O(\mathsf{Com}\|E_3\|b_3)$ via the random oracle O. The output is $\mathsf{com} = (\mathsf{com}_1, \mathsf{com}_2)$.

- $\{-1, 0, 1\} \leftarrow \mathsf{V}_1(\mathsf{com})$: On input a commitment com, V_1 outputs a random challenge $\mathsf{ch} \in \{-1, 0, 1\}$.

- $\mathsf{resp} \leftarrow \mathsf{P}_2(E_1, \varphi, \mathsf{com}, \mathsf{ch})$: On input a statement E_1, a corresponding witness φ, a commitment $\mathsf{com} = (\mathsf{com}_1, \mathsf{com}_2)$ and a challenge $\mathsf{ch} \in \{-1, 0, 1\}$, it outputs a response resp defined as follows. If

$\mathsf{ch} = -1$, then $\mathsf{resp} = (E_2, r, b_2)$; if $\mathsf{ch} = 1$, then $\mathsf{resp} = (E_3, \varphi(R), b_3)$; if $\mathsf{ch} = 0$, then $\mathsf{resp} = (E_2, \psi(\mathsf{Ker}(\varphi)), E_3, b_2, b_3)$, where ψ is the isogeny from E_0 having $\langle R \rangle$ as kernel.

- $1/0 \leftarrow V_2(E_1, \mathsf{com}, \mathsf{ch}, \mathsf{resp})$: It takes as input a statement E_1, a commitment $\mathsf{com} = (\mathsf{com}_1, \mathsf{com}_2)$, a challenge $\mathsf{ch} \in \{-1, 0, 1\}$, and a response resp. Depending on ch, the algorithm performs a check. In particular, if $\mathsf{ch} = -1$ then $\mathsf{resp} = (E, r, b)$ and the algorithm checks whether the isogeny from E_0 with kernel equal to $\langle P_2 + [r]Q_2 \rangle$ goes to E and whether $\mathsf{com}_1 = O(\mathsf{Com}||E||b)$. If $\mathsf{ch} = 1$, then $\mathsf{resp} = (E, T, b)$ and the algorithm checks whether the point T is in E_1, the order of T is $\ell_2^{e_2}$, the isogeny from E_1 with kernel $\langle T \rangle$ goes to E and $\mathsf{com}_2 = O(\mathsf{Com}||E||b)$. Finally, if $\mathsf{ch} = 0$, then $\mathsf{resp} = (E, T, \tilde{E}, b, \tilde{b})$ and the algorithm checks whether the point T is in E, the order of T is $\ell_1^{e_1}$, the isogeny from E with kernel $\langle T \rangle$ goes to $\tilde{E}$, $\mathsf{com}_1 = O(\mathsf{Com}||E||b)$, and $\mathsf{com}_2 = O(\mathsf{Com}||\tilde{E}||\tilde{b})$. If the check is successful, then it outputs 1, and 0 otherwise.

Let X be the set of supersingular elliptic curves E_1 over $\mathbb{F}_{p^2}$ having the same number of rational points of E_0, and Y be the set of all separable isogenies with domain E_0. Define the relation

$$\mathcal{R}_{\mathsf{SEC}} = \left\{ (E_1, \varphi) | E_1 \in X, \varphi \in Y, \varphi : E_0 \to E_1, \mathsf{deg}(\varphi) = \ell_1^{e_1} \right\}$$

and the relaxed relation

$$\tilde{\mathcal{R}}_{\mathsf{SEC}} = \left\{ (E_1, \mathsf{w}) \left| \begin{array}{c} E_1 \in X \quad \text{and} \\ \mathsf{w} = \varphi : E_0 \to E_1, \varphi \in Y, \mathsf{deg}(\varphi) = \ell_2^{2i}\ell_1^{e_1} \text{ with } 0 \le i \le e_2 \\ \text{or } \mathsf{w} = (x, x') \quad \text{s.t. } x \ne x', O(\mathsf{Com}||x) = O(\mathsf{Com}||x') \end{array} \right. \right\}.$$

It is not difficult to see that the Σ-protocol Σ_{SEC} described earlier is correct and has relaxed 3-special soundness for the relations $\mathcal{R}_{\mathsf{SEC}}$ and $\tilde{\mathcal{R}}_{\mathsf{SEC}}$. Furthermore, under the assumption that the following problem is hard, we prove in Proposition 1 that Σ_{SEC} is computationally HVZK.

Problem 1. (Decisional supersingular product problem) Let $\varphi : E_0 \to E_1$ be an isogeny of degree $\ell_1^{e_1}$. Given (E_2, E_3, φ') sampled with probability 1/2 from one of the following distributions, the decisional supersingular product problem $\mathsf{DSSP}_{\mathsf{pp}}$ requires to det ermine which distribution it is from:

- Choose a random point $R \in E_0[\ell_2^{e_2}]$ of order $\ell_2^{e_2}$. Let $\psi : E_0 \to E_2$ and $\psi' : E_1 \to E_3$ be the isogenies with kernels $\langle R \rangle$ and $\langle \varphi(R) \rangle$, respectively. Then let $\varphi' : E_1 \to E_2$ be the isogeny having $\langle \psi(\mathsf{Ker}(\varphi)) \rangle$ as kernel, where $\mathsf{deg}(\varphi') = \ell_1^{e_1}$.
- Choose E_2 randomly among all the supersingular elliptic curves defined over $\mathbb{F}_{p^2}$ having the same number of rational points as E_0. Then, choose a random point $U \in E_2$ of order $\ell_1^{e_1}$ and compute the isogeny $\varphi' : E_2 \to E_3$ having $\langle U \rangle$ as kernel.

Proposition 1. *Let λ be a security parameter, and let $\mathsf{pp} = (p, \ell_1, \ell_2, e_1, e_2, f, E_0, P_1, Q_1, P_2, Q_2)$ be the public parameters. The Σ-protocol Σ_{SEC} is computationally HVZK for the relation $\mathcal{R}_{\mathsf{SEC}}$ under $\mathsf{DSSP}_{\mathsf{pp}}$, assuming that the commitment oracle is computationally hiding.*

Proof. For $\mathsf{ch} = -1$, the simulator Sim^O uniformly samples r from $\mathbb{Z}/\ell_2^{e_2}\mathbb{Z}$ and computes the isogeny ψ of kernel generator $R = P_2 + [r]Q_2$ and the elliptic curve $E_2 = E_0/\langle R \rangle$. It then uniformly samples $b_2, b_3 \xleftarrow{\$} \{0, 1\}^\lambda$ and sets $\mathsf{com}_1 \leftarrow O(\mathsf{Com}||E_2||b_2)$ and $\mathsf{com}_2 \leftarrow O(\mathsf{Com}||1||b_3)$. The isogeny ψ is computed as in the original protocol, so the transcript is valid. Under the assumption that the commitment oracle is computationally hiding, an adversary cannot distinguish between the simulated com_2 and a commitment computed following the protocol.

For $\mathsf{ch} = 1$, the simulator uniformly samples r from $\mathbb{Z}/\ell_2^{e_2}\mathbb{Z}$, computes a basis $\{P_2', Q_2'\}$ of $E_1[\ell_2^{e_2}]$ and the isogeny ψ' of kernel generator $R' = P_2' + [r]Q_2'$, with codomain $E_3 = E_1/\langle R' \rangle$. It then uniformly samples $b_2, b_3 \xleftarrow{\$} \{0, 1\}^\lambda$ and sets $\mathsf{com}_1 \leftarrow O(\mathsf{Com}||1||b_2)$ and $\mathsf{com}_2 \leftarrow O(\mathsf{Com}||E_3||b_3)$. The transcript is valid, since by construction we have that $R' \in E_1$, that its order is $\ell_2^{e_2}$ and that the resulting φ' has image E_3. Without

knowing the witness, an adversary cannot tell R', of order $\ell_2^{e_2}$, from a point of order $\ell_2^{e_2}$ that is an image through φ. The mixing properties of supersingular isogeny graphs ensure that E_3 is randomly distributed, as it would be if it was computed following the protocol. Finally, if the commitment oracle is computationally hiding, the adversary cannot distinguish the simulated com_1 from a properly formed one.

For $\mathsf{ch} = 0$, the simulator samples r from $\mathbb{Z}/\ell_2^{e_2}\mathbb{Z}$ and computes the isogeny ψ of kernel generator $R = P_2 + [r]Q_2$ and the elliptic curve $E_2 = E_0/\langle R \rangle$. Then it computes a basis $\{P', Q'\}$ of $E_2[\ell_1^{e_1}]$, it samples s from $\mathbb{Z}/\ell_1^{e_1}\mathbb{Z}$ and the isogeny φ' of kernel generator $S' = P' + [s]Q'$, with codomain $E_3 = E_2/\langle S' \rangle$. It then uniformly samples $b_2, b_3 \xleftarrow{\$} \{0,1\}^\lambda$ and sets $\mathsf{com}_1 \leftarrow \mathcal{O}(\mathsf{Com}||E_2||b_2)$ and $\mathsf{com}_2 \leftarrow \mathcal{O}(\mathsf{Com}||E_3||b_3)$. The transcript is valid, since $S' \in E_2$ has order $\ell_1^{e_1}$ and the kernel it generates is that of an isogeny from E_2 to E_3. The adversary cannot tell the simulated point S' from a properly formed $\psi(\mathrm{ker}(\varphi))$, otherwise it would have solved the $\mathrm{DSSP}_{\mathrm{pp}}$ instance. $\qquad\square$

Remark 2. Within the protocol $\Sigma_{\mathsf{SECUER}}^{\mathsf{base}}$, the degree of the isogenies from E_0 to E_1 and from E_2 to E_3 is equal to $\ell_1^{d_1}$ (with d_1 a suitable natural number bigger than e_1), while the degree of the isogenies from E_0 to E_2 and from E_1 to E_3 is equal to $\ell_2^{d_2}$ (with d_2 a suitable natural number bigger than e_2). In this way, the protocol can be proved to be statistically HVZK [14, Prop. 17]. The conditions to satisfy this stronger property heavily affect both the transcript size and the execution times. Since in our work we are only interested in standard digital signatures, it is enough to rely on the computational HVZK property of the Σ-protocol. This allows us to preserve in full the SIDH parameters, including the degrees of the isogenies.

$\underline{\mathsf{P}_1(E_1, \varphi)}:$

1: $(r_1, r_2, \ldots, r_t) \xleftarrow{\$} (\mathbb{Z}/\ell_2^{e_2}\mathbb{Z})^t$
2: $(b_{2,1}, \ldots, b_{2,t}) \xleftarrow{\$} \{0,1\}^{\lambda t}$
3: $(b_{3,1}, \ldots, b_{3,t}) \xleftarrow{\$} \{0,1\}^{\lambda t}$
4: **for** $i = 1, 2, \ldots, t$ **do**
5: $\quad E_{2,i} \leftarrow E_0/\langle P_2 + [r_i]Q_2 \rangle$
6: $\quad E_{3,i} \leftarrow E_1/\langle \varphi(P_2 + [r_i]Q_2) \rangle$
7: $\quad \mathsf{com}_{i,1} \leftarrow \mathcal{O}(\mathsf{Com}||E_{2,i}||b_{2,i})$
8: $\quad \mathsf{com}_{i,2} \leftarrow \mathcal{O}(\mathsf{Com}||E_{3,i}||b_{3,i})$
9: **return** $\mathsf{com} \leftarrow (\mathsf{com}_{i,1}, \mathsf{com}_{i,2})_{i=1}^t$

$\underline{\mathsf{P}_2(E_1, \varphi, \mathsf{com}, \mathsf{ch})}:$

1: **for** $i = 1, 2, \ldots, t$ **do**
2: $\quad$ **if** $\mathsf{ch}_i = -1$ **then**
3: $\quad\quad \mathsf{resp}_i \leftarrow (E_{2,i}, r_i, b_{2,i})$
4: $\quad$ **else if** $\mathsf{ch}_i = 1$ **then**
5: $\quad\quad \mathsf{resp}_i \leftarrow (E_{3,i}, \varphi(P_2 + [r_i]Q_2), b_{3,i})$
6: $\quad$ **else**
7: $\quad\quad \psi_i \leftarrow \texttt{IsogenyFromKernel}(E_0,$
$\quad\quad \langle P_2 + [r_i]Q_2 \rangle)$
8: $\quad\quad \mathsf{resp}_i \leftarrow (E_{2,i}, \psi_i(\mathsf{Ker}(\varphi)), E_{3,i}, b_{2,i}, b_{3,i})$
9: **return** $\mathsf{resp} \leftarrow (\mathsf{resp}_i)_{i=1}^t$

$\underline{\mathsf{V}_2(E_1, \mathsf{com}, \mathsf{ch}, \mathsf{resp})}:$

1: **for** $i = 1, 2, \ldots, t$ **do**
2: $\quad$ **if** $\mathsf{ch}_i = -1$ **then**
3: $\quad\quad \mathsf{resp}_i \leftarrow (E, r, b)$
4: $\quad\quad$ **if** $E_0/\langle P_2 + [r]Q_2 \rangle \neq E$ **or** $\mathcal{O}(\mathsf{Com}||E||b) \neq \mathsf{com}_{i,1}$ **then**
5: $\quad\quad\quad$ **return** 0
6: $\quad$ **else if** $\mathsf{ch}_i = 1$ **then**
7: $\quad\quad \mathsf{resp}_i \leftarrow (E, T, b)$
8: $\quad\quad$ **if** $T \notin E_1[\ell_2^{e_2}]_{max}$ **or** $E_1/\langle T \rangle \neq E$ **or** $\mathcal{O}(\mathsf{Com}||E||b) \neq \mathsf{com}_{i,2}$ **then**
9: $\quad\quad\quad$ **return** 0
10: $\quad$ **else**
11: $\quad\quad \mathsf{resp}_i \leftarrow (E, T, \tilde{E}, b, \tilde{b})$
12: $\quad\quad$ **if** $T \notin E[\ell_2^{e_2}]_{max}$ **or** $E/\langle T \rangle \neq \tilde{E}$ **or** $\mathcal{O}(\mathsf{Com}||E||b) \neq \mathsf{com}_{i,1}$
$\quad\quad$ **or** $\mathcal{O}(\mathsf{Com}||\tilde{E}||\tilde{b}) \neq \mathsf{com}_{i,2}$ **then**
13: $\quad\quad\quad$ **return** 0
14: **return** 1

Figure 1: Algorithms in Σ_{SEC}^t. Given a supersingular elliptic curve E, `IsogenyFromKernel`$(E, \cdot)$ denotes an algorithm which, on input a subgroup $S \subset E$, computes an isogeny from E with kernel S. Moreover, $E[\ell^e]_{\mathrm{max}}$ denotes the points of order ℓ^e in $E[\ell^e]$, when ℓ is prime and $e \in \mathbb{N}$.

As the protocol Σ_{SEC} has a soundness error $\varepsilon = 2/3$, it is necessary to repeat its execution in parallel t times to obtain a negligible soundness error. It is customary to set t as the minimum positive integer such that $\varepsilon^t < 2^{-\lambda}$. Therefore, we obtain

$$t > \frac{1}{\log_2(3) - 1}\lambda \approx 1.7 \cdot \lambda. \tag{1}$$

The Σ-protocol that results from repeating Σ_{SEC} in parallel t-times, which will be denoted by Σ_{SEC}^t in the following, is depicted in Figure 1.

Assuming a supersingular elliptic curve over $\mathbb{F}_{p^2}$ is identified by a single element of $\mathbb{F}_{p^2}$, the average size (in bits) of a transcript of Σ_{SEC}^t (excluding the statement) is approximated by

$$|\text{transcript}| = \boldsymbol{4\lambda t} + \lceil \boldsymbol{\log(3)} \rceil \boldsymbol{t} + \frac{t}{3}((2\lceil \log p \rceil + \lceil \log \ell_2^{e_2} \rceil + \lambda) + +(4\lceil \log p \rceil + 1 + \lambda) + (6\lceil \log p \rceil + 1 + 2\lambda)). \tag{2}$$

where the terms indicate **commitment**, **challenge**, and response sizes (the terms within brackets correspond to the sizes of the responses to challenge -1, 1, and 0).

Within an execution of Σ_{SEC}^t, the prover computes $2t$ elliptic-curve scalar multiplications, $2t$ isogenies and $2t$ commitments to produce the commitment. In addition, to produce the response, the prover evaluates an $\ell_1^{e_1}$-isogeny (on a point of order $\ell_2^{e_2}$) $t/3$ times, and an $\ell_2^{e_2}$-isogeny (on a point of order $\ell_1^{e_1}$) $t/3$ times, on average.

When Σ_{SEC}^t is turned into the digital signature scheme DS_{SEC} via the Fiat-Shamir transform, the security of DS_{SEC} is guaranteed by the hardness of the relation $\tilde{\mathcal{R}}_{\text{SEC}}$, as the problem of finding an isogeny between two given isogenous elliptic curves is still believed to be hard (and it has not been affected by the recent cryptanalytic attacks on SIDH). A signature produced by DS_{SEC} is just a transcript of Σ_{SEC}^t without the statement and the challenge (as the latter can be easily recovered, being it the digest of a hash function on the message m to sign and the commitment com). Consequently, the signature size is slightly smaller than the size of a transcript (without the statement) of Σ_{SEC}^t, and the computational cost to sign is that undergone by the prover in an execution of Σ_{SEC}^t, plus one hash-function evaluation. Therefore, the efficiency analysis presented above applies almost directly to DS_{SEC} (see the last column of Table 1 for the signature sizes of DS_{SEC} for different SIDH/SIKE parameters).

3 Signature-size optimisations

In this section, we apply some known cryptographic techniques to DS_{SEC} to decrease the size of the signatures it produces. We start with some optimisations that determine a reduction of the signature size without causing any increase of the signing computations, and then we discuss those that have an impact on the signing time. We stress that none of the considered optimisations affects the security of DS_{SEC}.

Table 1: Average sizes (in bits) of the transcripts (excluding the statement) produced by Σ_{SEC}^t, and the average length of the signatures produced by DS_{SEC}, working with different SIDH/SIKE parameters. The signature sizes are obtained from equation (2) minus the challenge length $\lceil \log(3) \rceil t$

SIKE parameters	$\lceil \log p \rceil$	λ	t	Transcript length	Signature length
SIKEp434	434	128	218	543,692	543,256
SIKEp503	503	128	218	606,404	605,968
SIKEp610	610	192	326	1,163,277	1,162,625
SIKEp751	751	256	435	1,956,775	1,955,905

3.1 Challenge and commitment recoverability

A Σ-protocol (Gen, P = (P$_1$, P$_2$), V = (V$_1$, V$_2$)) is said to be *commitment-recoverable* if, with overwhelming probability over the random choice of a pair (x, w) $\leftarrow$ Gen(1^λ), for any ch $\in$ ChSet and resp $\in$ ResSet, there exists a unique commitment com $\in$ ComSet that makes (x, com, ch, resp) a valid transcript, and such a commitment can be publicly computed by means of an algorithm taking (x, ch, resp) as input. This property allows for shorter signatures by omitting com from them, and letting the verifier re-compute it. Its correctness is then checked by means of the challenge ch.

The original version of the Σ-protocol Σ_{SEC}^t, described in Figure 1, does not satisfy commitment recoverability (e.g., the response resp$_i$ when ch$_i$ = −1 does not allow to recover com$_{i,2}$). However, we can modify (Σ_{SEC} and) Σ_{SEC}^t in such a way that the new protocol(s) are commitment recoverable.

The modification of Σ_{SEC}^t which we suggest[2] is detailed in Figure 2. In particular, P$_1$ remains unchanged, while the algorithm P$_2$ outputs the response resp$_i$ = (r_i, $b_{2,i}$, com$_{i,2}$) when ch$_i$ = −1; the response resp$_i$ = ($\varphi(P_2 + [r_i]Q_2)$, $b_{3,i}$, com$_{i,1}$) when ch$_i$ = 1; and the response resp$_i$ = ($E_{2,i}$, $\psi_i(\mathsf{Ker}(\varphi))$, $b_{2,i}$, $b_{3,i}$) when ch$_i$ = 0, where ψ_i is the isogeny with kernel $\langle P_2 + [r_i]Q_2 \rangle$ from E_0. The verifier then re-computes part of the commitment, and checks whether it corresponds to that received by P$_1$.

The expected sizes (in bits) of the elements in a transcript of the modified Σ_{SEC}^t protocol (excluding the statement) are approximated by

$$|\mathsf{com}| = 4\lambda t, \quad |\mathsf{ch}| = \lceil \log(3)t \rceil, |\mathsf{resp}| = \frac{t}{3}((\lceil \log(\ell_2^{e_2}) \rceil + 3\lambda) + (2\lceil \log p \rceil + 1 + 3\lambda) + (4\lceil \log p \rceil + 1 + 2\lambda)) \quad (3)$$

(the terms within the brackets corresponds to the size of the responses to challenge −1, 1, and 0, respectively).

In Table 2, we lists the approximated sizes (in bits) of commitments, challenges, and responses for the four SIKE parameter sets.

Remark 3. The gain column in Table 2 indicates how many bits (in percentage) we save when storing the response computed by the modified Σ_{SEC}^t after applying commitment and challenge recoverability. Each value is computed as $(s_0 - s_1)/s_0$, where s_0 is the average length of the response output by Σ_{SEC}^t, and s_1 is the average length of the response output by the modified version. Every time we will introduce a new optimisation, we will compute the gain it provides as $(s_i - s_{i-1})/s_0$, where s_i is the average response length produced by the version of Σ_{SEC}^t with the current and all previous modifications, and s_{i-1}, the response length with only the previous modifications. Note that in this way the overall gain provided by our optimisations can be obtained by simply adding together all the intermediate gains.

Thanks to commitment recoverability, when the modified Σ-protocol is turned into a digital signature, the commitment com does not need to be part of the corresponding signature. In principle, the challenge ch should now be part of the signature, as it necessary to recover the commitment com. However, the modified Σ_{SEC}^t protocol is also *challenge-recoverable*, and then ch can be excluded from the signature. We recall that a Σ-protocol is *challenge-recoverable* if the challenge ch in a transcript (x, com, ch, resp) can be reconstructed from resp. This is the case for both Σ_{SEC}^t and its modification, since the type of each resp$_i$ in resp uniquely determines the corresponding challenge bit ch$_i$ in ch. Thus, one can simply omit the challenge from a transcript and let it be deduced from the response. Consequently, both challenge and commitment can be reconstructed by the verifier. Therefore, the signature sizes for different SIDH/SIKE parameters are equal to the response sizes in Table 2. Moreover, we note that the computational effort made by the prover in the modified Σ_{SEC}^t protocol is exactly the same as in the original Σ_{SEC}^t protocol.

---

2 We stress that our modification is analogous to the one proposed in the study by Chi-DomAηηguez [15, Sec. 3.2] for the protocol Σ_{wSIDH}.

$\underline{\mathsf{P}_1(E_1, \varphi):}$

1: $(r_1, r_2, \ldots, r_t) \xleftarrow{\$} (\mathbb{Z}/\ell_2^{e_2}\mathbb{Z})^t$
2: $(b_{2,1}, \ldots, b_{2,t}) \xleftarrow{\$} \{0, 1\}^{\lambda t}$
3: $(b_{3,1}, \ldots, b_{3,t}) \xleftarrow{\$} \{0, 1\}^{\lambda t}$
4: **for** $i = 1, 2, \ldots, t$ **do**
5:      $E_{2,i} \leftarrow E_0/\langle P_2 + [r_i]Q_2 \rangle$
6:      $E_{3,i} \leftarrow E_1/\langle \varphi(P_2 + [r_i]Q_2) \rangle$
7:      $\mathsf{com}_{i,1} \leftarrow \mathcal{O}(\mathsf{Com}||E_{2,i}||b_{2,i})$
8:      $\mathsf{com}_{i,2} \leftarrow \mathcal{O}(\mathsf{Com}||E_{3,i}||b_{3,i})$
9: **return** $\mathsf{com} \leftarrow (\mathsf{com}_{i,1}, \mathsf{com}_{i,2})_{i=1}^t$

$\underline{\mathsf{P}_2(E_1, \varphi, \mathsf{com}, \mathsf{ch}):}$

1: **for** $i = 1, 2, \ldots, t$ **do**
2:      **if** $\mathsf{ch}_i = -1$ **then**
3:          $\mathsf{resp}_i \leftarrow (r_i, b_{2,i}, \mathsf{com}_{i,2})$
4:      **else if** $\mathsf{ch}_i = 1$ **then**
5:          $\mathsf{resp}_i \leftarrow (\varphi(P_2 + [r_i]Q_2), b_{3,i}, \mathsf{com}_{i,1})$
6:      **else**
7:          $\psi_i \leftarrow \mathtt{IsogenyFromKernel}(E_0,$
         $\langle P_2 + [r_i]Q_2 \rangle)$
8:          $\mathsf{resp}_i \leftarrow (E_{2,i}, \psi_i(\mathsf{Ker}(\varphi)), b_{2,i}, b_{3,i})$
9: **return** $\mathsf{resp} \leftarrow (\mathsf{resp}_i)_{i=1}^t$

$\underline{\mathsf{V}(E_1, \mathsf{com}, \mathsf{ch}, \mathsf{resp}):}$

1: **for** $i = 1, 2, \ldots, t$ **do**
2:      **if** $\mathsf{ch}_i = -1$ **then**
3:          $\mathsf{resp}_i \leftarrow (r, b, c)$
4:          $E \leftarrow E_0/\langle P_2 + [r]Q_2 \rangle$
5:          **if** $\mathcal{O}(\mathsf{Com}||E||b) \neq \mathsf{com}_{i,1}$ **or** $c \neq \mathsf{com}_{i,2}$ **then**
6:              **return** 0
7:      **else if** $\mathsf{ch}_i = 1$ **then**
8:          $\mathsf{resp}_i \leftarrow (T, b, c)$
9:          **if** $T \notin E_1[\ell_2^{e_2}]_{max}$ **or** $\mathcal{O}(\mathsf{Com}||E_1/\langle T \rangle||b) \neq \mathsf{com}_{i,2}$ **or** $c \neq \mathsf{com}_{i,1}$ **then**
10:             **return** 0
11:      **else**
12:          $\mathsf{resp}_i \leftarrow (E, T, b, \tilde{b})$
13:          **if** $T \notin E[\ell_2^{e_2}]_{max}$ **or** $\mathcal{O}(\mathsf{Com}||E||b) \neq \mathsf{com}_{i,1}$ **or** $\mathcal{O}(\mathsf{Com}||E/\langle T \rangle||\tilde{b}) \neq \mathsf{com}_{i,2}$ **then**
14:             **return** 0
15: **return** 1

Figure 2: Modified Σ_{SEC}^t protocol which enjoys commitment recoverability. The blue text marks the differences with the original scheme depicted in Figure 1.

Table 2: Sizes (in bits) of the signatures of $\mathsf{DS}_{\mathsf{SEC}}$ (i.e. the transcripts of Σ_{SEC}^t excluding statements and challenges) produced by the modified Σ_{SEC}^t after applying challenge and commitment recoverability, for different SIDH/SIKE parameters. The "gain" column indicates by how much the signature lengths have reduced compared to those in Table 1

SIKE param.	$\lceil \log p \rceil$	λ	t	|com|	|ch|	|resp| = |signature|	Gain (%)
SIKEp434	434	128	218	111,616	10	279,622	48.53
SIKEp503	503	128	218	111,616	10	312,249	48.47
SIKEp610	610	192	326	250,368	10	597,993	48.57
SIKEp751	751	256	435	445,440	11	1,005,575	48.59

3.2 Compressed responses

In Σ_{SEC}^t, when $\mathsf{ch}_i = 1$ the prover responds with the point $\varphi(P_2 + [r_i]Q_2)$ generating the kernel of the commitment isogeny $\psi_i' : E_1 \rightarrow E_{3,i}$. Being P_2 and Q_2 over $\mathbb{F}_{p^2}$ by construction, this requires the transmission of $2 \cdot \log p + 1$ bits.

Following the algorithmic improvements proposed in the studies by Costello et al. and Azarderakhsh et al. [21,22], we can deterministically compute a torsion basis $\{P', Q'\}$ of $E_1[\ell_2^{e_2}]$ for any statement/public key E_1. Then, the response can be set as the result (α_i, β_i) of a double discrete logarithm, with α_i, β_i such that $[\alpha_i]P' + [\beta_i]Q' = \varphi(P_2 + [r_i]Q_2)$. Since $\varphi(P_2 + [r_i]Q_2)$ is of order $\ell_2^{e_2}$, one of the two coefficients α_i, β_i must be invertible modulo $\ell_2^{e_2}$; if it is α_i, we let $\iota_i \coloneqq 1$ and $\gamma_i \coloneqq \alpha_i^{-1}\beta_i$, otherwise we let $\gamma_i \coloneqq 1$ and $\iota_i \coloneqq \beta_i^{-1}\alpha_i$. The response can then be set as (ι_i, γ_i), and the kernel generator computed as $[\iota_i]P' + [\gamma_i]Q'$.

With this method, the size of the response is therefore reduced to $\lceil \log(\ell_2^{e_2}) \rceil + 1$ bits, at the cost of computing a deterministic torsion basis both by signer and verifier, and determining a double discrete logarithm only on the signer's side. Note that the basis P', Q' can be computed once for all by adding it to the statement/public key E_1. The new public key would look exactly like an *old* SIDH public key, with the crucial difference that the basis is computed independently of the secret isogeny φ, preventing the applicability of the attacks on SIDH to this context. Moreover, the following pre-computation would allow us to use the method of compressed responses at no additional computation cost. The prover needs to determine $\varphi(P_2)$ and $\varphi(Q_2)$, and then compute their respective components α, β and γ, ω in the basis $\{P', Q'\}$. Then, to compute a response, the prover would only need to calculate one multiplications and two sums in $\mathbb{Z}/\ell_2^{e_2}\mathbb{Z}$, since $\varphi(P_2 + [r_i]Q_2) = [\alpha + \gamma]P' + [r_i \cdot(\beta + \omega)Q']$.

We stress that, since the number of bits of the response on challenge $\mathsf{ch}_i = -1$ is one bit shorter than the compressed response for the challenge $\mathsf{ch}_i = 1$, the challenge recoverability of the protocol is preserved. The new response length is then computed as follows:

$$|\mathsf{resp}| = \frac{t}{3}((\lceil \log(\ell_2^{e_2}) \rceil + 3\lambda) + (\lceil \log(\ell_2^{e_2}) \rceil + 1 + 3\lambda) + (4\lceil \log p \rceil + 1 + 2\lambda)). \tag{4}$$

Remark 4. The aforementioned compression method could also be applied for the case where $\mathsf{ch} = 0$, at the cost of a slow down, since a new canonical basis would need to be computed. Indeed, unlike for $\mathsf{ch} \in \{\pm 1\}$, the curve E_2 varies for each challenge. We will therefore not consider this compression method for $\mathsf{ch} = 0$ in Table 3.

3.3 Seed trees

A primitive called *seed tree* [16] can be used to first generate a number of pseudorandom values and later efficiently disclose an arbitrary subset of them, without revealing any information on the values which are not disclosed. More precisely, a seed tree is a complete binary tree (i.e. a binary tree in which every level, except possibly the last, is completely filled, and all nodes are as far left as possible) of λ-bit seed values such that the left (resp. right) child of a λ-bit seed seed is the left (resp. right) half of the bit string $\mathsf{Expand}(\mathsf{seed}\|h)$, where h is a unique identifier for the position of seed in the binary tree. The seed values of a subset of the set of leaves can be efficiently revealed by sharing the appropriate set of internal seeds in the tree. As a simple example, if the sender (who created the complete binary tree) only provides the seed value associated to the left child of the root of the tree, then the recipient will only be able to recover the seed values associated to the leaves in the left half of the tree. Notably, the recipient will not learn any information about the leaves in the right half of the tree. A seed tree consists of four oracle-calling algorithms: $\mathsf{SeedTree}$, $\mathsf{ReleaseSeeds}$, $\mathsf{RecoverLeaves}$, $\mathsf{SimulateSeeds}$. Below, we recall the formal definitions of the first three algorithms, where $\mathsf{Expand} : \{0,1\}^{\lambda + \lceil \log_2(t-1) \rceil} \longrightarrow \{0,1\}^{2\lambda}$ is a Pseudo-random generator for any $\lambda, t \in \mathbb{N}$, instantiated by a random oracle O.

- $\mathsf{SeedTree}^O(\mathsf{seed}_{\mathrm{root}}, t) \to \{\mathsf{leaf}_i\}_{i \in \{1,\dots,t\}}$: On input a root seed $\mathsf{seed}_{\mathrm{root}} \in \{0,1\}^\lambda$ and an integer $t \in \mathbb{N}$, the algorithm constructs a complete binary tree with t leaves by recursively expanding each seed to obtain its

Table 3: Sizes (in bits) of the responses produced by the modified Σ_{SEC}^t with compressed responses for different SIDH/SIKE parameters. Commitment and challenge lengths remain unchanged with respect to Table 2. The "gain" column is computed as the difference between the new signature lengths and the ones in Table 2 divided by the signature lengths from Table 1

SIKE param.	$\lceil \log p \rceil$	λ	t	$|\mathsf{resp}| = |\mathsf{signature}|$	Gain (%)
SIKEp434	434	128	218	232,388	8.69
SIKEp503	503	128	218	257,531	9.03
SIKEp610	610	192	326	498,563	8.55
SIKEp751	751	256	435	842,740	8.33

children seeds. Calls to the random oracle are of the form $O(\text{Expand}\|\text{seed}\|h)$, where $h \in \{1, \dots, t-1\}$ identifies the position of seed in the binary tree. The algorithm finally outputs the list of seeds associated with the t leaves.

- $\text{ReleaseSeeds}^O(\text{seed}_{\text{root}}, \mathbf{c}, j) \rightarrow \text{seeds}_{\text{internal}}$: On input a root seed $\text{seed}_{\text{root}} \in \{0,1\}^\lambda$, a bit string $\mathbf{c} \in \{-1, 0, 1\}^t$, and $j \in \{-1, 0, 1\}$, it outputs the list of seeds $\text{seeds}_{\text{internal}}$ that covers all the leaves with index i such that $c_i = j$. Here, we say that a set of nodes F *covers* a set of leaves S if the union of the leaves of the subtrees rooted at each node $v \in F$ is exactly the set S. Here, we note that each seed in $\text{seeds}_{\text{internal}}$ is coupled with an index which identifies its position in the binary tree.
- $\text{RecoverLeaves}^O(\text{seeds}_{\text{internal}}, \mathbf{c}, j) \rightarrow \{\overline{\text{leaf}_i}\}_{i \text{ s.t. } c_i = j}$: On input a set $\text{seeds}_{\text{internal}}$, a bit string $\mathbf{c} \in \{0,1\}^t$ and a chosen $j \in \{-1, 0, 1\}$, it computes and outputs all the leaves of the subtrees rooted at the seeds in $\text{seeds}_{\text{internal}}$.

By construction, the leaves $\{\text{leaf}_i\}_{i \text{ s.t. } c_i = j}$ output by $\text{SeedTree}(\text{seed}_{\text{root}}, t)$ are the same as those output by $\text{RecoverLeaves}(\text{ReleaseSeeds}(\text{seed}_{\text{root}}, \mathbf{c}, j), \mathbf{c}, j)$ for any $\mathbf{c} \in \{0,1\}^t$ and $j \in \{-1, 0, 1\}$. We observe that the last algorithm SimulateSeeds can be used to argue that the seeds associated with all the leaves with index i such that $c_i \neq j$ are indistinguishable from uniformly random values for a recipient that is only given $\text{seeds}_{\text{internal}}$, $\mathbf{c}$ and j.

We now describe how seed trees can be used to optimise the modified Σ_{SEC}^t on two fronts. The optimisation we propose in the following is motivated by the observation that in the first three lines of P_1 (Figure 2), all the elements necessary to compute the inputs to the commitment oracle are sampled: the t random coefficients $\{r_1, \dots, r_t\} \xleftarrow{\$} (\mathbb{Z}/\ell_2^{e_2}\mathbb{Z})^t$ for the commitment curves $(E_{2,i}, E_{3,i})_{i=1}^t$ and two sets of random strings $(b_{2,1}, \dots, b_{2,t}), (b_{3,1}, \dots, b_{3,t}) \xleftarrow{\$} \{0,1\}^{\lambda t}$. Of these inputs, only the coefficients need to be selectively opened, while the entirety of the random strings can be revealed (as their disclosure does not impact the computational HVZK property of the protocol).

Therefore, instead of independently choosing t coefficients and $2t$ random bit-strings, $3t$ seeds could be generated using two distinct seed trees, one for the coefficients which originates from the root $\text{seed}_{\text{root}}^{\text{coeff}}$ and one for the random bit-strings which originates from the root $\text{seed}_{\text{root}}^{\text{str}}$. Then, instead of selectively revealing a subset of the $2t$ bit strings according to the response algorithm P_2, the prover could directly sends the initial seed $\text{seed}_{\text{root}}^{\text{str}}$ used to generate them, letting the verifier compute them all. On the other hand, instead of responding with the random coefficients r_i for the challenge bits $\mathsf{ch}_i = -1$, the prover could output $\text{seeds}_{\text{internal}} \leftarrow \text{ReleaseSeeds}(\text{seed}_{\text{root}}\text{coeff}, \mathsf{ch}, -1)$. The verifier would then use $\text{seeds}_{\text{internal}}$ along with ch and $j = -1$ to recover the required seeds by running RecoverLeaves.

Let us analyse how generating all random strings from a single root seed $\text{seed}_{\text{root}}^{\text{str}}$ and revealing it to the verifier affects the response length. Each random string is represented by λ bits, and without the use of seed trees, one of them is communicated if $\mathsf{ch}_i = -1$ or $\mathsf{ch}_i = 1$, and two of them if $\mathsf{ch}_i = 0$. For t responses resp_i on challenges evenly distributed over $\{-1, 0, 1\}$, this amounts to $\frac{1}{3}t(\lambda + \lambda + 2\lambda) = \frac{4}{3}t\lambda$ bits. If all random strings are generated with a seed tree from a root seed $\text{seed}_{\text{root}}^{\text{str}}$, releasing just the root seed requires only λ bits.

Such neat analysis cannot be performed on the application of the seed tree primitive to the generation of the coefficients $r_1, \dots, r_t$, since the amount of internal seeds that need to be revealed depends on how -1, 0, and 1

Table 4: Sizes (in bits) of the different components of the transcripts (excluding the statement) produced by the modified Σ_{SEC}^t of Figure 2 when it also incorporates compressed responses (for challenges $\mathsf{ch}_i = 1$) and seed trees. The "gain" indicates by how much the signature lengths have reduced compared to those in Table 3

SIKE param.	$\lceil \log p \rceil$	λ	t	|com|	|ch|	|resp| = |signature|	Gain (%)
SIKEp434	434	128	218	111,616	10	188,771	8.03
SIKEp503	503	128	218	111,616	10	211,370	7.62
SIKEp610	610	192	326	250,368	10	403,020	8.22
SIKEp751	751	256	435	445,440	11	676,681	8.49

are distributed over the challenge string. In the worst-case scenario, i.e. when all the leaf seeds need to be revealed, instead of $\frac{\log p}{2}$ bits for the coefficient r_i, only λ bits for the generating seed need to be communicated.

The following equation (5) determines how seed trees affect the lengths of the responses produced by the modified Σ_{SEC}^t of Figure 2 when it also incorporates compressed responses (for challenges $\mathsf{ch}_i = 1$). In parenthesis, we add response lengths for $\mathsf{ch}_i = -1, 1, 0$ respectively, where the 2λ addends represent the necessary information for commitment recoverability; the lengths of random strings $(b_{2,1}, \ldots, b_{2,t}), (b_{3,1}, \ldots, b_{3,t})$ is removed from each response and replaced by a unique λ addend representing $\mathsf{seed}_{\mathsf{root}}^{\mathsf{str}}$.

$$|\mathsf{resp}| = \lambda + \frac{t}{3}\left((\lambda + 2\lambda) + (\lceil \log(\ell_2^{e_2}) \rceil + 1 + 2\lambda) + (4\lceil \log p \rceil + 1)\right). \tag{5}$$

In Table 4, we report the numbers produced by equation (5) for different SIKE parameters. Table 4 differs from Table 3 only in the last two columns.

3.4 Unbalanced challenge space

Equation (5) clearly shows that the response resp_i when $\mathsf{ch}_i = 0$ is significantly bigger than when $\mathsf{ch}_i \in \{-1, 1\}$. As a consequence, one might consider to *unbalance* the challenge string ch in order to decrease the overall size of $\mathsf{resp} = (\mathsf{resp}_i)_{i=1}^t$. Such modification was proposed in the study by Beullens et al. [16, Sec. 3.4.1], and has the extra positive effect of making the transcript/signature size constant. To be more concrete, the modification consists in choosing a positive integer K and performing M parallel executions of Σ_{SEC}, exactly K of which use the unfavourable challenge bit 0. The number of challenges in $\{-1, 0, 1\}^M$ having exactly K components equal to 0 is $\binom{M}{K} \cdot 2^{M-K}$. Equation 1 dictates that, for a given K, M should be selected in such a way that the success probability of a dishonest prover is bounded above by $2^{-\lambda}$, i.e.

$$\binom{M}{K} \cdot \frac{2^{M-K}}{\mathfrak{n}} \geq 2^{\lambda}. \tag{6}$$

Therefore, for generic M and K, it is necessary to find the maximal number $\mathfrak{n} = \mathfrak{n}_{M,K}$ of challenges to which a dishonest prover would be able to correctly reply. Afterwards, we will find the optimal $M \in \mathbb{N}$ and $K \in \{0, \ldots, \lceil t/3 \rceil\}$ such that

$$|\mathsf{resp}| = \lambda + \left\lceil \frac{M-K}{2} \right\rceil \left((\lambda + 2\lambda) + (\lceil \log(\ell_2^{e_2}) \rceil + 1 + 2\lambda)\right) + K(4\lceil \log p \rceil + 1) \tag{7}$$

is minimal. We first start by finding $\mathfrak{n}$.

Lemma 1. *We can express* $\mathfrak{n}$ *as follows*:

$$\mathfrak{n} = \max\left\{\binom{h}{K} \cdot 2^{M-h} : h \in \{K, \ldots, M\}\right\}. \tag{8}$$

Proof. Let S be the set of all subsets U of $\{-1, 0, 1\}^M$ consisting of elements of Hamming weight $M - K$ (i.e. elements with $M - K$ non-zero components) and such that, for any index $i \in \{1, \ldots, M\}$, U does not contain three elements whose i-th components are all distinct. Then $\mathfrak{n}$ is the maximum cardinality among the sets in S. Given a set $U \in S$, let h_U denote the number of indices i such that there exists a sequence in U that has a zero at index i, i.e.

$$h_U := \#\{i \in \{1, \ldots, M\} : \exists x \in U, x_i = 0\}.$$

Hence, for a set $U \in S$, we can have $\binom{h_U}{K}$ choices for the entries that are zero. The remaining $M - h_U$ entries can be either 1 or -1, giving us 2^{M-h_U} choices. Therefore, the maximal size of a set $U \in S$ is $\mathfrak{n}$ as in equation (8). $\quad\square$

Proposition 2. *Let $h_{\max}$ denote a value of h realising the maximum of the set in Equation (8). Then $h_{\max} = 2K$ and*

$$\mathfrak{n} = \binom{2K}{K} \cdot 2^{M-2K}.$$

Proof. Let us study the behaviour of the discrete function $f(h) := \binom{h}{K} \cdot 2^{M-h}$ taking values in $\{K, \ldots, M\}$, with parametrised integers $K < M$. We start by noticing that the left factor $\binom{h}{K}$ is monotonically increasing, while the right factor 2^{M-h} is monotonically decreasing, with ratio $2^{M-h}/2^{M-(h+1)} = 2$ for any value of h.

The function $f(h)$ is initially increasing, since the left factor grows faster than how the right factor decreases. In fact, for any $h \geq K$,

$$\frac{\binom{h+1}{K}}{\binom{h}{K}} = \frac{h+1}{h+1-K} > 2 \iff h < 2K - 1.$$

For $h = 2K - 1$, the ratio between the binomial coefficients for h and $h+1$ is exactly 2, so $f(2K - 1) = f(2K)$.

For any $h > 2K - 1$, the function f is decreasing, since $\binom{h+1}{K} / \binom{h}{K} < 2$ for $2K \leq h < M$.

We conclude by arbitrarily choosing $h_{\max} = 2K$ as a value of h maximising f (one can equivalently set $h_{\max} = 2K - 1$, obtaining a less neat formula), and thus $\mathfrak{n} = \binom{2K}{K} \cdot 2^{M-2K}$. $\square$

As an example, when $\lambda = 128$ and K is set to 75, it is sufficient to have $M = 247$ parallel runs of Σ_{SEC}, since here the value $h_{\max}$ giving us the maximal size of U is 150. The values of M and K that optimally minimise the length of the response for different SIDH/SIKE parameters are collected in Table 5.

To obtain the values in Table 5, we simply run through all values of M, up to a very large upper bound (say twice the value of the corresponding t), and all values of $K \in \{0, \ldots, \lceil t/3 \rceil\}$ and pick out the values (M, K) minimizing $|\mathsf{resp}|$. As expected, the values of M obtained end up being very close (just a little bit bigger) than the corresponding values of t (which can be found, for example, in Table 4).

3.5 Summary

We conclude this section highlighting the overall gain in applying challenge and commitment recoverability, compressed responses, seed trees, and unbalanced challenge space optimisations to the Σ_{SEC} protocol. We phrase the results in terms of signature sizes:

- for SIKEp434, we shorten the signature from 66.31 kB to at most 18.89 kB, corresponding to a reduction of at least 71.51%,

Table 5: Values of M and K for the unbalanced challenge space that minimise the response length of the modified $\Sigma_{\mathsf{SEC}}^{M}$ for different SIDH/SIKE parameters still guaranteeing a negligible soundness error. The size of resp is in bits, while the "gain" column reports by how much the signature lengths have reduced compared to those in Table 4

SIKE parameters	$\lceil \log p \rceil$	λ	M	K	$h_{\max}$	|resp|	Gain (%)
SIKEp434	434	128	250	48	96	154,783	6.26
SIKEp503	503	128	250	48	96	169,041	6.99
SIKEp610	610	192	362	76	152	336,095	5.76
SIKEp751	751	256	478	103	206	571,453	5.38

- for SIKEp503, we shorten the signature from 73.97 kB to at most 20.63 kB, corresponding to a reduction of at least 72.10%,
- for SIKEp610, we shorten the signature from 141.92 kB to at most 41.03 kB, corresponding to a reduction of at least 71.09%,
- for SIKEp751, we shorten the signature from 238.76 kB to at most 69.76 kB, corresponding to a reduction of at least 70.78%.

We also note that all the optimisations discussed in this section could be extended to the distributed trusted-setup protocol [14, Sec. 5] built on top of $\Sigma_{\text{SECUER}}^{\text{base}}$ to collaboratively produce a random supersingular elliptic curve whose endomorphism ring is hard to compute even for the parties who did the sampling (Table 6).

4 Running-time optimisations

In an execution of the Σ-protocol Σ_{SEC}^t, $2t$ commitment isogenies need to be computed. The same holds for all the modified protocols introduced in Section 3, including the one that considers fixed-weight challenges, which we denote by Σ_{SEC}^M (see Section 3.4). All such isogenies have degree $\ell_2^{e_2}$ – half of them originate from E_0 and half from E_1.

We now present two optimisations that take advantage of the computation of several isogenies of the same degree from the same supersingular elliptic curve. Despite focusing on Σ_{SEC}^t (and, implicitly, on Σ_{SEC}^M), both optimisations could be extended to the distributed trusted-setup protocol [14, Sec. 5] built on top of $\Sigma_{\text{SECUER}}^{\text{base}}$.

To better explain such optimisations, we recall the fastest generic method to compute a cyclic isogeny of degree $\ell_2^{e_2}$ from its kernel. For simplicity, we specialise our presentation to the SIKE parameters, and therefore in the following, we will replace ℓ_1 with 2, ℓ_2 with 3, e_1 with a and e_2 with b.

Let ψ be an isogeny of degree 3^b from a supersingular elliptic curve E over $\mathbb{F}_{p^2}$, with kernel generated by $R \coloneqq P + [r]Q$ for some basis $\{P, Q\}$ of $E[3^b]$ and some $r \in \mathbb{Z}/3^b\mathbb{Z}$. The isogeny ψ can be expressed as the composition $\psi = \psi^{(b)} \circ \psi^{(b-1)} \circ \cdots \circ \psi^{(1)} = \prod_{j=0}^{b-1}\psi^{(b-j)}$, where each $\psi^{(j)}$ has degree 3. The first isogeny $\psi^{(1)}$ of such decomposition is the isogeny whose kernel is generated by $[3^{b-1}]R$. Then $\psi^{(2)}$ is the isogeny with kernel generated by $[3^{b-2}]\psi^{(1)}(R)$, and so on until $\psi^{(b)}$, the last 3-isogeny with kernel generated by $\psi^{(b-1)}(...(\psi^{(1)}(R))...\,)$.

The strategies described in [23, Appendix D] speed up the computation of $\psi = \psi^{(b)} \circ \psi^{(b-1)} \circ \cdots \circ \psi^{(1)}$ by minimising the number of operations to execute. We give a high-level description of these strategies in the following lines. To recursively determine the kernels of and computing the 3-isogenies in the decomposition $\psi = \prod_{j=0}^{b-1}\psi^{(b-j)}$, the strategies combine two operations: scalar multiplication and isogeny evaluation. Each of these two operations runs in a certain time, with the latter slightly faster than the former. The goal of the strategies is that of minimising the overall computational cost. Referring to Figure 3, we can graphically describe their goal and how they operate. In particular, they aim to obtain the points on the hypotenuse of the right-angled triangle using the least amount of arrows – with blue arrows representing scalar multiplications by 3 and red arrows representing 3-isogeny evaluations – under the condition that the $(i + 1)$-th line from the top cannot be *accessed* before reaching the rightmost point on the i-th line from the top, i.e. before the computation of the 3-isogeny $\psi^{(i)}$. In fact, the elements on the hypotenuse, from the top-right corner

Table 6: Overall gains in applying all the proposed optimisations to the Σ_{SEC} protocol

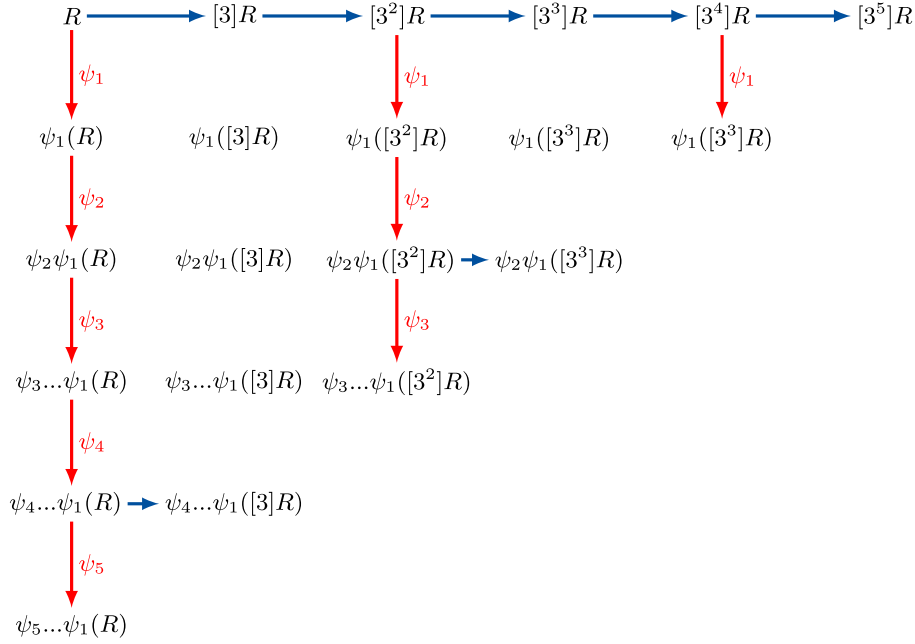| SIKE parameters	$\lceil \log p \rceil$	λ	|resp| (kB)	Total gain (%)
SIKEp434	434	128	18.89	71.51
SIKEp503	503	128	20.63	72.1
SIKEp610	610	192	41.03	71.09
SIKEp751	751	256	69.76	70.78

Figure 3: Graphical representation of the optimal strategy proposed in the study by Azarderakhsh et al. [23, Appendix D] for the case $b = 6$ to compute a 3^6-isogeny from a kernel generator R. Blue arrows represent scalar multiplications by 3, red arrows represent 3-isogeny evaluations.

to the bottom-left corner, represent the kernels of $\psi^{(1)}, \psi^{(2)}, \dots, \psi^{(b)}$, respectively. The naive (standard) approach to compute $\psi = \prod_{j=0}^{b-1} \psi^{(b-j)}$ would be to start at the first line, go all the way to the right, then move down to the next line, and go all the way to the right, and so on. However, there exist alternative strategies which accelerate the computations. In particular, Figure 3 depicts the optimal strategy proposed in [23, Appendix D] for the case $b = 6$ (the general strategy for a generic b is just a generalisation of it).

4.1 Computing several isogenies in parallel

We now go back to the computation of $2t$ commitment isogenies within an execution of Σ_{SEC}^t (or within one of its variations). To outline the first optimisation we propose, we restrict our attention to the isogenies $\psi_1, \dots, \psi_t$ which originate from E_0. Analogous considerations hold for the isogenies originating from E_1.

As we saw earlier, for $i \in \{1, \dots, t\}$, the fastest method to compute ψ_i from its kernel $\langle P_2 + [r_i]Q_2 \rangle$ – where $\{P_2, Q_2\}$ is a basis of $E_0[3^b]$ – is that of determining the composition $\psi_i^{(b)} \circ \psi_i^{(b-1)} \circ \cdots \circ \psi_i^{(1)}$ of 3-isogenies. We then observe that there are 3 possible values for $\psi_i^{(1)}$, 3^2 for $\psi_i^{(2)}$, 3^3 for $\psi_i^{(3)}$, and so on. For example, the three possible values for $\psi_i^{(1)}$ have $\langle [3^{b-1}]P_2 + [0 \cdot 3^{b-1}]Q_2 \rangle$, $\langle [3^{b-1}]P_2 + [1 \cdot 3^{b-1}]Q_2 \rangle$ and $\langle [3^{b-1}]P_2 + [2 \cdot 3^{b-1}]Q_2 \rangle$ as kernels. Since $t = 218$ for SIKEp434 and SIKEp503, $t = 326$ for SIKEp610 and $t = 435$ for SIKEp751, some of the possible 3-isogenies will surely occur multiple times[3]. In particular, for $j \in \{1, \dots, b\}$ and two different $i, i' \in \{1, \dots, t\}$, it holds that

$$\psi_i^{(j)} \circ \cdots \circ \psi_i^{(1)} = \psi_{i'}^{(j)} \circ \cdots \circ \psi_{i'}^{(1)} \iff r_i \equiv r_{i'} \pmod{3^j}. \tag{9}$$

---

3 When using an unbalanced challenge space, as discussed in Section 3.4, Σ_{SEC} is repeated M times, with $M > t$ for all SIKE parameters.

After a few steps, however, repetitions are expected to stop occurring (for the rapid-mixing property of supersingular isogeny graphs). For example, considering the SIKEp434 parameters, since there are 3^5 = 243 possible values for r_i (mod 3^5) and t = 218, the fourth is the last factor where we can still expect a good amount[4] of repetitions.

Given the aforementioned observations, a speed-up in computing the t isogenies of degree 3^b from E_0 can be obtained by avoiding repeatedly computing 3-isogenies which occur multiple times. To be more precise, this can be achieved by pre-computing all possible values for $\psi_i^{(1)}, \ldots, \psi_i^{(\alpha)}$ – with α being the biggest positive integer such that $3^\alpha < t$ – and then, for every $i \in \{1, \ldots, t\}$, calculating the congruence classes of r_i modulo 3, $3^2, \ldots, 3^\alpha$ to determine $\psi_i^{(1)}, \ldots, \psi_i^{(\alpha)}$, respectively. Alternatively, for each $i \in \{1, \ldots, t\}$ and $j \in \{1, \ldots, \alpha\}$, the congruence class of r_i modulo 3^j can be determined before computing the kernel of $\psi_i^{(j)}$ and the isogeny itself. If such congruence class matches the one of a coefficient r_k with $k \in \{1, \ldots, j - 1\}$, then the kernel of $\psi_i^{(j)}$ and the isogeny itself do not need to be re-computed, since $\psi_i^{(j)} = \psi_h^{(j)}$.

Figure 4 shows how the modular-arithmetic checks can be exploited within the optimal strategies proposed in the study by Azarderakhsh et al. [23, Appendix D]. In particular, they grant the possibility of moving from line 1 to line $\alpha + 1$ (where α is equal to 3 in the toy example depicted in the figure) without the need to reach the rightmost of the first α lines. In other words, the vertical orange arrows can be simply determined form the modular-arithmetic checks.

To evaluate the advantage in applying the described tweak, we first deduce formulas for the number of horizontal and vertical arrows, respectively, required in the optimal strategies from the study by Azarderakhsh et al. [23, Appendix D] to compute isogenies of degree 3^b:

$$H_{os}(b) = \left\lfloor \frac{3b - 4}{2} \right\rfloor, \quad V_{os}(b) = \left\lfloor \frac{b + 1}{2} \right\rfloor^2 - (b \bmod 2). \tag{10}$$

We borrow from the study by Azarderakhsh et al. [23, Appendix D] the costs (in cycles) of a point-tripling operation, p_3, and of computation and evaluation of a 3-isogeny, q_3. With them and Equation (10), we calculate the cost of the optimal strategies from [23, Appendix D] to compute t isogenies of degree 3^b, and we compare it with the cost when pre-computation is performed. The results are presented in Table 7. For example, for SIKEp434, t is equal to 218, b = 137, p_3 = 5,322, and q_3 = 5,282, with the total cost for computing t isogenies of degree 3^{137} via the optimal strategy being

$$218(V_{os}(137) \cdot q_3 + H_{os}(137) \cdot p_3) = 5{,}716{,}545{,}548.$$

If isogenies are pre-computed and selected using modular arithmetics according to equation (9), we can start each isogeny computation from the fifth line (as noted earlier, we expect a good amount of repetitions in the first four factors when t = 218), thus reducing the amount of arrows to compute. In this case, $H'_{os}(137) = H_{os}(137 - 4)$ horizontal arrows and $V'_{os}(137) = V_{os}(137 - 4) + 4$ vertical arrows are required to compute a single 3^{137}-isogeny. The total cost of computing 218 isogenies of degree 3^{137} in parallel is

$$218(V_{os}(133) \cdot q_3 + 4 \cdot q_3 + H_{os}(133) \cdot p_3) = 5{,}396{,}382{,}900,$$

which corresponds to an efficiency increase of 5.60% for SIKEp434 parameters.

We can analogously compute the savings determined by our tweak for the other SIKE parameter sets, pre-computing the first 5 steps for SIKEp610 and SIKEp751 (in which cases $3^5 < t$). The results are summarised in Table 7, where we also indicate the cost of storing the pre-computed kernel generators of all possible initial steps $\psi_1, \ldots \psi_4$.

For comparison with the parallel strategy with pre-computation, we now describe a sequential approach to speed up Σ^t_{SEC} by avoiding recomputing the same steps several times, but without introducing any pre-computations. When we compute the first isogeny ψ_1 from its coefficient r_1, we store the kernels of its first four

---

4 Some repetitions are also expected in the fifth factor. In particular, an average of 28.35 repetitions occur. In other words, if we were to pick 128 items from a set of 3^5 = 243, then on average (repeating this experiment many times), we would get around 28 repetitions.
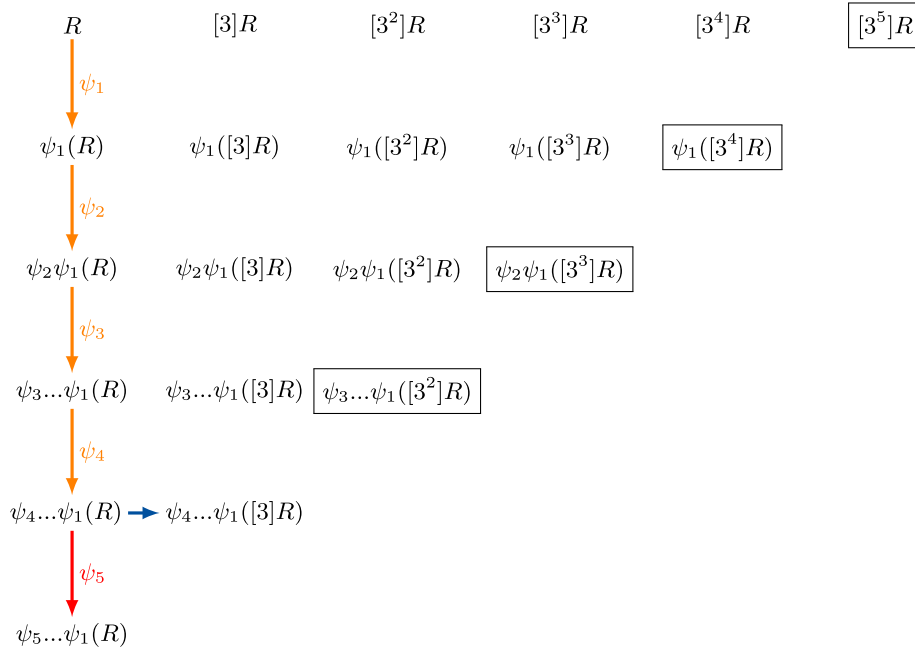
$$R \qquad [3]R \qquad [3^2]R \qquad [3^3]R \qquad [3^4]R \qquad \boxed{[3^5]R}$$

$$\downarrow \psi_1$$

$$\psi_1(R) \qquad \psi_1([3]R) \qquad \psi_1([3^2]R) \qquad \psi_1([3^3]R) \qquad \boxed{\psi_1([3^4]R)}$$

$$\downarrow \psi_2$$

$$\psi_2\psi_1(R) \qquad \psi_2\psi_1([3]R) \qquad \psi_2\psi_1([3^2]R) \qquad \boxed{\psi_2\psi_1([3^3]R)}$$

$$\downarrow \psi_3$$

$$\psi_3...\psi_1(R) \qquad \psi_3...\psi_1([3]R) \qquad \boxed{\psi_3...\psi_1([3^2]R)}$$

$$\downarrow \psi_4$$

$$\psi_4...\psi_1(R) \longrightarrow \psi_4...\psi_1([3]R)$$

$$\downarrow \psi_5$$

$$\psi_5...\psi_1(R)$$

Figure 4: By doing some pre-computation or avoiding the multiple computation of 3-isogenies, the optimal strategy from [23, Appendix D] to compute and evaluate a 3^6-isogeny from a kernel generator R is granted the possibility to move from line 1 to line $\alpha + 1$ (with $\alpha = 3$ in the figure) without the need to reach the right most of the first α lines. In particular, the points in boxes can be obtained instantly from modular-arithmetic checks, and they determine, in turn, the vertical orange arrows.

Table 7: Costs (in clock cycles and kilobytes of storage) and percentage gains in using the pre-computation tweak to compute t commitment isogenies in Σ_{SEC}^{t} and in its unbalanced challenge variant $\Sigma_{\text{SEC}}^{t,U}$ compared to those of using the optimal strategies from the study by Azarderakhsh et al. [23, Appendix D], for different SIKE parameter sets

Protocol	SIKE parameters	t	b	Old cost (cc)	New cost (cc)	Gain (%)	Storage (kB)
Σ_{SEC}^{t}	SIKEp434	218	137	5,716,545,548	5,400,988,804	5.52%	13.035
	SIKEp503	218	159	7,642,101,180	7,275,879,492	4.79	15.105
	SIKEp610	326	192	16,365,527,304	15,704,201,096	4.04	55.41
	SIKEp751	435	239	33,908,315,250	32,272,252,410	4.82	68.2
$\Sigma_{\text{SEC}}^{t,U}$	SIKEp434	250	137	6,555,671,500	6,019,478,500	8.18	39.43
	SIKEp503	250	159	8,763,877,500	8,140,531,500	7.11	45.65
	SIKEp610	362	192	18,172,763,448	17,438,407,352	4.04	55.41
	SIKEp751	478	239	37,260,171,700	35,462,383,108	4.82	68.2

initial steps. When computing the second isogeny, we first check whether $r_2 \equiv r_1 (\bmod 3^j)$ for any $j = 1, 2, 3, 4$: if so, we already have the kernel for that step, and we can save all horizontal lines that would be required to compute it; if not, we compute the kernel corresponding to that step and store it. Repeating this simple procedure would allow us to compute the initial steps of all t isogenies $\psi_1, ..., \psi_t$ without repeating the same unnecessary scalar multiplications, at the cost of evaluating modular equivalences. To analyse the cost of this strategy, let us start from the one when we considered pre-computations, and let us increase it by the extra cost of performing scalar multiplications in the first 4 steps. In particular, we need $3 \cdot (b-1)$ scalar multiplications to obtain the three possibilities for $\psi_i^{(1)}$; as per the optimal strategy, we need 3^2 isogeny evaluations on the kernels $3^{b-2}R_i$ to obtain the nine possibilities for $\psi_i^{(2)}$. Then again, we perform $3^3 \cdot (b-3)$ scalar multiplications to obtain the 27 possibilities for $\psi_i^{(3)}$, and finally 3^4 isogeny evaluations to obtain the 81 possibilities for $\psi_i^{(4)}$. The total cost of this strategy is therefore

Table 8: Costs (in clock cycles and kilobytes of storage) and percentage gains in computing t commitment isogenies in Σ_{SEC}^t following the strategy that avoids pre-computation but performs modular-arithmetic checks on-the-fly, compared to those of the optimal strategy from [23, Appendix D], for different SIKE parameter sets

SIKE parameters	t	b	Old cost (cc)	New cost (cc)	Gain (%)	Storage (kB)
SIKEp434	218	137	5,716,545,548	5,422,890,556	5.14	13.035
SIKEp503	218	159	7,642,101,180	7,301,293,764	4.46	15.105
SIKEp610	326	192	16,365,527,304	15,967,764,224	2.43	55.41
SIKEp751	435	239	33,908,315,250	32,601,850,914	3.85	68.2

Parallel scalar multiplication$(P_2, Q_2, (r_1, r_2, \ldots, r_t))$:

1: **for** $i = 1, 2, \ldots, t$ **do**
2:      $R_i \leftarrow 0_{E_0}$
3: **for** $j = 0, 1, \ldots, m - 2$ **do**
4:      **for** $i = 1, 2, \ldots, t$ **do**
5:          **if** $r_i^j = 1$ **then**
6:              $R_i \leftarrow R_i + Q_2$
7:      $Q_2 \leftarrow [2]Q_2$
8: **for** $i = 1, 2, \ldots, t$ **do**
9:      **if** $r_i^{m-1} = 1$ **then**
10:        $R_i \leftarrow R_i + Q_2$
11:      $R_i \leftarrow R_i + P_2$
12: **return** $(R_1, R_2, \ldots, R_t)$

Figure 5: Algorithm to compute t coefficients $R_i = P_2 + [r_i]Q_2$ in parallel.

$$t \cdot (V_{os}(b - 4) \cdot q_3 + 4 \cdot q_3 + H_{os}(b - 4) \cdot p_3) + (30 \cdot b - 84) \cdot p_3 + 90 \cdot q_3,$$

for SIKEp434 and SIKEp503 and

$$t \cdot (V_{os}(b - 5) \cdot q_3 + 5 \cdot q_3 + H_{os}(b - 5) \cdot p_3) + (264 \cdot b - 1{,}254) \cdot p_3 + 90 \cdot q_3,$$

for SIKEp610 and SIKEp751 (considering the fact that repetitions occur in the fifth step as well), and it requires the same amount of extra storage as the pre-computation strategy. The results for all SIKE parameter sets are presented in Table 8.

4.2 Computing multiple scalar multiplications in parallel

Following the optimisations presented in the previous section, we note that many points belonging to E_0 must be computed. These points are used in the commitment generation of Σ_{SEC}^t (or in one of its variants), and are the kernel generators $R_1, R_2, \ldots R_t$, where each $R_i = P_2 + [r_i]Q_2$ is obtained by randomly sampling r_i from $\mathbb{Z}/\ell_2^{e_2}\mathbb{Z}$ (or by using a seed tree, as shown in Section 3.3).

     We now discuss how to calculate all R_i's in parallel and obtain some computational savings. An analogous strategy can be applied to parallelise the computation of the kernel generators of the commitment isogenies which originate from E_1.

     For each R_i, the scalar multiplication $[r_i]Q_2$ can be performed using the classical double-and-add strategy, in which the points that obtain doubled and (possibly) added at each step are multiples of Q_2. Hence, we can perform the multiple doublings of Q_2 only once for all R_i's, as detailed in Figure 5, where m is the minimum number of bits necessary to represent any coefficient in $\mathbb{Z}/\ell_2^{e_2}\mathbb{Z}$, and $r_i = (r_i^0, r_i^1, \ldots, r_i^{m-1})_2$ is the little-endian binary representation of $r_i \in \mathbb{Z}/\ell_2^{e_2}\mathbb{Z}$.

In analysing how much this strategy saves us, let $\mathsf{Hw}(r_i)$ denote the Hamming weight (i.e. the number of non-zero components) of the binary representation of the coefficient r_i, and let cADD and cDBL denote the cost (in cycles) of adding and doubling points over an elliptic curve, respectively. With a naive approach, we would perform $m - 1$ doublings and $\mathsf{Hw}(r_i) + 1$ additions (with "+1" counting for the last addition by P_2) for each R_i, at a total cost of

$$t(m - 1)\cdot \mathsf{cDBL} + \sum_{i=1}^{t}(\mathsf{Hw}(r_i) + 1)\cdot \mathsf{cADD}.$$

With our parallelised approach presented in Figure 5, we still perform the same amount $\mathsf{Hw}(r_i) + 1$ of additions for each R_i, but the $m - 1$ doublings are performed once for all the R_i's, which saves us $(t - 1)(m - 1)\mathsf{cDBL}$ (at least 99% of the doublings for any $t \geq 100$, as in our case study).

5 Conclusions

We have assessed the efficiency of a SIDH-based digital signature built on a weaker but more efficient variant of the recent identification protocol $\Sigma_{\mathsf{SECUER}}^{\mathsf{base}}$ from the study by Basso et al. [14, Section 4]. The Σ-protocol we consider only achieves computational honest-verifier zero-knowledge instead of the stronger notion of statistical honest-verifier zero-knowledge, but it allows for shorter isogenies. We have conducted our analysis by applying some known cryptographic techniques to decrease the signature size and proposing a minor optimisation to compute many isogenies in parallel from the same starting curve. In addition, we provide novel results on unbalanced challenge space with ternary challenges. Our assessment confirms that the problem of designing a practical isogeny-based signature scheme remains largely open. Nonetheless, the proposed optimisations can be applied to the distributed trusted-setup protocol [14, Sec. 5] built on top of $\Sigma_{\mathsf{SECUER}}^{\mathsf{base}}$ to collaboratively produce a random supersingular elliptic curve whose endomorphism ring is hard to compute even for the parties who participated in the sampling.

Author contributions: All authors have accepted responsibility for the entire content of this manuscript and approved its submission. As common practice in cryptography, the authors have not followed the SDC approach; instead, they have sorted the authors alphabetically.

Conflict of interest: Dr. Federico Pintore is an Editorial Board member of the Journal of Mathematical Cryptology and was not involved in the review process of this article.

Data availability statement: Data sharing is not applicable to this article as no datasets were generated or analysed during the current study.

References

[1]   De Feo L, Kohel D, Leroux A, Petit C, Wesolowski B. SQISign: compact post-quantum signatures from quaternions and isogenies. International Conference on the Theory and Application of Cryptology and Information Security. 2020. Vol. 8. Issue 3; p. 64–93.

[2]   Feo LD, Leroux A, Longa P, Wesolowski B. New algorithms for the Deuring correspondence - towards practical and secure SQISign signatures. In: Hazay C, Stam M, editors. Advances in Cryptology - EUROCRYPT 2023 – 42nd Annual International Conference on the

Theory and Applications of Cryptographic Techniques, Lyon, France, April 23–27, 2023, Proceedings, Part V. Vol. 14008 of Lecture Notes in Computer Science. Springer; 2023. p. 659–90. doi: 10.1007/978-3-031-30589-4_23.

[3] De Feo L, Jao D, Pluuuut J. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. J Math Cryptol. 2014;8(3):209–47.

[4] Jao D, Azarderakhsh R, Campagna M, Costello C, De Feo L, Hess B, et al. SIKE. National Institute of Standards and Technology; 2017. //csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[5] Galbraith SD, Petit C, Silva J. Identification protocols and signature schemes based on supersingular isogeny problems. J Cryptol. 2020;33(1):130–75.

[6] Yoo Y, Azarderakhsh R, Jalali A, Jao D, Soukharev V. A post-quantum digital signature scheme based on supersingular isogenies. In: FC. Springer; 2017. p. 163–81.

[7] Castryck W, Decru T. An efficient key recovery attack on SIDH. In: Hazay C, Stam M, editors. Advances in cryptology - EUROCRYPT 2023. Cham: Springer Nature Switzerland; 2023. p. 423–47.

[8] Maino L, Martindale C, Panny L, Pope G, Wesolowski B. A direct key recovery attack on SIDH. In: Hazay C, Stam M, editors. Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23–27, 2023, Proceedings, Part V. Vol. 14008 of Lecture Notes in Computer Science. Springer; 2023. p. 448–71. doi: 10.1007/978-3-031-30589-4_16.

[9] Robert D. Breaking SIDH in polynomial time. In: Hazay C, Stam M, editors. Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23–27, 2023, Proceedings, Part V. Vol. 14008 of Lecture Notes in Computer Science. Springer; 2023. p. 472–503. doi: 10.1007/978-3-031-30589-4_17.

[10] De Feo L, Galbraith SD. SeaSign: compact isogeny signatures from class group actions. In: Ishai Y, Rijmen V, editors. Advances in Cryptology - EUROCRYPT 2019, Part III. Vol. 11478 of Lecture Notes in Computer Science. Darmstadt, Germany: Springer, Heidelberg, Germany; 2019. p. 759–89.

[11] Beullens W, Kleinjung T, Vercauteren F. CSI-FiSh: efficient isogeny based signatures through class group computations. In: Galbraith SD, Moriai S, editors. Advances in Cryptology - ASIACRYPT 2019, Part I. Vol. 11921 of Lecture Notes in Computer Science. Kobe, Japan: Springer, Heidelberg, Germany; 2019. p. 227–47.

[12] Chávez-Saab J, Chi-Domiiiinguez JJ, Jaques S, Rodríguez-Henríquez F. The SQALE of CSIDH: sublinear Vélu quantum-resistant isogeny action with low exponents. J Cryptograph Eng. 2022 Sep;12(3):349–68.

[13] Feo LD, Dobson S, Galbraith SD, Zobernig L. SIDH proof of knowledge. In: Agrawal S, Lin D, editors. Advances in Cryptology - ASIACRYPT 2022 – 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5–9, 2022, Proceedings, Part II. Vol. 13792 of Lecture Notes in Computer Science. Springer; 2022. p. 310–39. doi: 10.1007/978-3-031-22966-4_11.

[14] Basso A, Codogni G, Connolly D, Feo LD, Fouotsa TB, Lido GM, et al. Supersingular curves you can trust. In: Hazay C, Stam M, editors. Advances in Cryptology - EUROCRYPT 2023 – 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23–27, 2023, Proceedings, Part II. Vol. 14005 of Lecture Notes in Computer Science. Springer; 2023. p. 405–37. doi: 10.1007/978-3-031-30617-4_14.

[15] Chi-Domínguez JJ. A Note on Constructing SIDH-PoK-based Signatures after Castryck-Decru Attack; 2022. //eprint.iacr.org/2022/1479. Cryptology ePrint Archive, Paper 2022/1479. //eprint.iacr.org/2022/1479.

[16] Beullens W, Katsumata S, Pintore F. Calamari and Falafl: logarithmic (linkable) ring signatures from isogenies and lattices. In: International Conference on the Theory and Application of Cryptology and Information Security; 2020. p. 464–92.

[17] Chi-DomAηnguez JJ, Mateu V, Perin LP. SIDH-sign: an efficient SIDH PoK-based signature; 2022. //eprint.iacr.org/2022/475. Cryptology ePrint Archive, Paper 2022/475. //eprint.iacr.org/2022/475.

[18] Fiat A, Shamir A. How to prove yourself: practical solutions to identification and signature problems. In: Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings. Vol. 263 of Lecture Notes in Computer Science. Springer; 1986. p. 186–94.

[19] Silverman JH. The arithmetic of elliptic curves. Vol. 106. New York: Springer; 2009.

[20] Galbraith SD. Mathematics of public key cryptography. Cambridge, UK: Cambridge University Press; 2012. //books.google.no/books?id=owd76BElvosC.

[21] Costello C, Longa P, Naehrig M. Efficient algorithms for supersingular isogeny Diffie-Hellman. In: Robshaw M, Katz J, editors. Advances in Cryptology - CRYPTO 2016, Part I. Vol. 9814 of Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany; 2016. p. 572–601.

[22] Azarderakhsh R, Jao D, Kalach K, Koziel B, Leonardi C. Key compression for isogeny-based cryptosystems. In: Proceedings of the 3rd ACM International Workshop on ASIA Public-Key Cryptography; 2016. p. 1–10.

[23] Azarderakhsh R, Campagna M, Costello C, De Feo L, Hess B, Jalali A, et al. Supersingular isogeny key encapsulation november 30, 2017. Third Round Candidate of the NIST's post-quantum cryptography standardization process; 2020. //sikeorg/.