Efficiency of SIDH-based signatures (yes, SIDH)

Wissam Ghantous[1], Federico Pintore[2] and Mattia Veroni[3]

[1] Mathematical Institute, University of Oxford, UK,
wissam.ghantous@maths.ox.ac.uk
[2] Department of Mathematics, University of Bari, IT,
federico.pintore@uniba.it
[3] NTNU - Norwegian University of Science and Technology, Trondheim, NO ,
mattia.veroni@ntnu.no

Abstract. In this note we assess the efficiency of a SIDH-based digital signature built on a *diminished* 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 the problem of designing a practical isogeny-based signature scheme remains largely open. However, concretely determine the current state of the art which future optimisations can compare to appears to be of relevance for a problem which has witnessed only small steps towards a solution.

Keywords: Post-quantum Cryptography · Isogeny-based Cryptography · Digital Signature

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 [DFKL+20], which is the most compact post-quantum signature scheme. On the other hand, isogeny-based cryptosystems incur in high execution times, with SQISign making no exception (despite the recent improvements in [DFLW22]). The most promising results in terms of computational efficiency have been obtained for the key-exchange SIDH [DFJP14] and the corresponding key-encapsulation mechanism SIKE [JAC+17]. However, not all schemes built on SIDH share the same quality. An example are the SIDH-based digital signatures proposed in [GPS17,YAJ+17], for which no substantial amelioration has

appeared since their publication. Nevertheless, they represented an alternative starting point for those researchers seeking for a practical isogeny-based digital signature building on existing schemes. However, three major classical attacks [CD22a,MM22,Rob22] 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[4] 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 [DDGZ21, Sec. 5.3] — denoted by Σ_{wSIDH} in the following — was originally designed for the SIDH setting, while for the second one [BCC+22, Sec. 4] — which we denote by $\Sigma^{\mathsf{base}}_{\mathsf{SECUER}}$ — the SIDH parameters are (probably the most) favorable 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 $\Sigma^{\mathsf{base}}_{\mathsf{SECUER}}$ are not affected by the attacks in [CD22a,MM22,Rob22] and hence can still be the base for constructing digital signature schemes as well as non-interactive zero-knowledge proofs (NIZKPs in short).

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 *diminished* variant of $\Sigma^{\mathsf{base}}_{\mathsf{SECUER}}$. The adjective diminished refers to the fact that $\Sigma^{\mathsf{base}}_{\mathsf{SECUER}}$ 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. The main reasons to work with Σ_{SEC} instead of other SIDH-based Σ-protocols are three. First of all, a similar assessment focused on Σ_{wSIDH} was recently conducted in [CD22b]. 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 $\Sigma^{\mathsf{base}}_{\mathsf{SECUER}}$ 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, $\Sigma^{\mathsf{base}}_{\mathsf{SECUER}}$ has been used to prove random generation of supersingular curves of unknown endomorphism rings in a distributed and trusted manner [BCC+22].

[4] Sea-Sign [DG19] is an isogeny-based digital signature scheme which works with isogenies and elliptic curves defined over prime fields. CSI-FiSh [BKV19] 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 [CSCJR22], and the instantiations of SeaSign with *bigger* parameters incur, again, in high execution times.

We conduct our analysis by applying some known cryptographic techniques to decrease the signature size. By doing so, we can shorten the signatures produced by means of Σ_{SEC} by approximately 69%. As an 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 [BKP20] 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, in order 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 [BCC+22, Sec. 5] built on top of $\Sigma^{\mathsf{base}}_{\mathsf{SECUER}}$ to collaboratively produce a random supersingular elliptic curve whose endomorphism ring is hard to compute even for the parties who did the sampling. Moreover, concretely determine the current state of the art of isogeny-based signatures which future optimisations can compare to appears to be of relevance for a problem which has witnessed only small steps towards a solution.

Related Work. The SIDH-based digital signature scheme proposed in [GPS17] produces signatures of approximately 12 KB when targeting 128 bits of classical security. For the same security target, the signature scheme in [CDMP22] (deduced from a different SIDH-based Σ-protocol proposed in [DDGZ21, 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 [CD22b] on a still-secure digital signature built on top of Σ_{wSIDH} achieves signatures of size approximately 74KB for the parameter set $\mathsf{SIKEp434}$. Note that the protocol in [GPS17] and Σ_{wSIDH} in [DDGZ21, Sec. 5]) are 2-special sound. The protocols in [DDGZ21, Sec. 6]) is instead 3-special sound.

Road Map. The paper 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 $\Sigma^{\mathsf{base}}_{\mathsf{SECUER}}$ from [BCC+22, Sec. 4]. By applying the Fiat-Shamir transform [FS86] 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 contains some closing remarks.

2 Preliminaries

In this section we list some definitions and results regarding Σ-protocols and digital signatures. We then detail the Σ-protocol Σ_{SEC} which is a diminished variant of $\Sigma^{\mathsf{base}}_{\mathsf{SECUER}}$ from [BCC+22, Sec. 4]. Throughout this work, we assume the reader is familiar with isogenies between elliptic curves over finite fields (we refer to [Sil09,Gal12] for an introduction to the topic).

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

2.1 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 $\mathcal{L}_{\mathcal{R}} = \{\mathsf{x} \in X \mid \exists\, \mathsf{w} \in Y : (\mathsf{x}, \mathsf{w}) \in \mathcal{R}\}$.

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 holding a pair $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}$ can show beyond doubt to a verifier the knowledge of a valid witness w for x, without revealing any information about w 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 $\mathcal{O}$. We may occasionally omit the superscript $\mathcal{O}$ when the meaning is clear from the context.

Definition 2 (Σ-protocols). *A Σ-protocol S for polynomially-computable binary relations $\mathcal{R} \subseteq \tilde{\mathcal{R}}$ consists of five oracle-calling polynomial-time algorithms $(\mathsf{Gen}, \mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2))$, where V_2 is deterministic, Gen, P_1, 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 has the following three-move flow:*

- *The key-generation algorithm* $\mathsf{Gen}^{\mathcal{O}}(1^\lambda)$ *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* $\mathsf{com} \longleftarrow \mathsf{P}_1^{\mathcal{O}}(\mathsf{x},\mathsf{w})$ *and sends the commitment* com *to the verifier.*
- *The verifier runs* $\mathsf{ch} \longleftarrow \mathsf{V}_1^{\mathcal{O}}(\mathsf{com})$ *to obtain a random challenge, and sends* ch *to the prover.*
- *Given* ch*, the prover computes* $\mathsf{resp} \longleftarrow \mathsf{P}_2^{\mathcal{O}}(\mathsf{x},\mathsf{w},\mathsf{com},\mathsf{ch})$ *and returns the response* resp *to the verifier.*
- *The verifier runs* $\mathsf{V}_2^{\mathcal{O}}(\mathsf{x},\mathsf{com},\mathsf{ch},\mathsf{resp})$ *and outputs 1 if they accept, 0 otherwise.*

Here $\mathcal{O}$ *is modeled as a random oracle. Moreover, a transcript* $(\mathsf{x},\mathsf{com},\mathsf{ch},\mathsf{resp}) \in X \times \mathsf{ComSet} \times \mathsf{ChSet} \times \mathsf{ResSet}$ *of the protocol is said to be valid (relative to* x*) in case* $\mathsf{V}_2(\mathsf{x},\mathsf{com},\mathsf{ch},\mathsf{resp})$ *outputs* 1*.*

We require the following properties of a Σ-protocol:

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

$$\Pr\left[\mathsf{V}_2^{\mathcal{O}}(\mathsf{x},\mathsf{com},\mathsf{ch},\mathsf{resp}) = 1 \;\middle|\; \begin{array}{c}(\mathsf{x},\mathsf{w}) \longleftarrow \mathsf{KeyGen}^{\mathcal{O}}(1^\lambda)\\ \mathsf{com} \longleftarrow \mathsf{P}_1^{\mathcal{O}}(\mathsf{x},\mathsf{w}),\\ \mathsf{ch} \longleftarrow \mathsf{V}_1^{\mathcal{O}}(\mathsf{com}),\\ \mathsf{resp} \longleftarrow \mathsf{P}_2^{\mathcal{O}}(\mathsf{x},\mathsf{com},\mathsf{w},\mathsf{ch})\end{array}\right] = 1.$$

2. **Relaxed κ-Special Soundness:** there exists a polynomial-time *extraction algorithm* Ex such that, given any κ valid transcripts $(\mathsf{x},\mathsf{com},\mathsf{ch}_1,\mathsf{resp}_1),\ldots,(\mathsf{x},\mathsf{com},\mathsf{ch}_\kappa,\mathsf{resp}_\kappa)$ relative to the same statement $\mathsf{x} \in \mathcal{L}_{\mathcal{R}}$, with the same commitment com and κ distinct challenges $\mathsf{ch}_1,\ldots,\mathsf{ch}_\kappa$, outputs w such that $(\mathsf{x},\mathsf{w}) \in \tilde{\mathcal{R}}$ (i.e., Ex is only required to recover a witness in $\tilde{\mathcal{R}}$ rather than in $\mathcal{R}$).

3. **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 $\mathcal{O}$. We say the Σ-protocol is *statistically HVZK* if there exists a PPT simulator algorithm $\mathsf{Sim}^{\mathcal{O}}$ such that, for any $(\mathsf{x},\mathsf{w}) \in \mathcal{R}$, $\mathsf{ch} \in \mathsf{ChSet}$ and any computationally unbounded adversary $\mathcal{A}$ that makes at most a polynomial number of queries to $\mathcal{O}$, we have

$$|\Pr[1 \longleftarrow \mathcal{A}^{\mathcal{O}}(\tilde{\mathsf{P}}^{\mathcal{O}}(\mathsf{x},\mathsf{w},\mathsf{ch})] - \Pr[1 \longleftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{Sim}^{\mathcal{O}}(\mathsf{x},\mathsf{ch}))]| = \mathsf{negl}(\lambda),$$

where $\tilde{\mathsf{P}}$ is a prover $\mathsf{P} = (\mathsf{P}_1,\mathsf{P}_2)$ run on (x,w) with challenge fixed as ch. If the above relation holds only for any polynomially bounded adversary $\mathcal{A}$, the protocol is said to be *computationally HVZK*.

2.2 Digital signatures

Below we recall the definition of digital signature schemes.

Definition 3 (Digital signature schemes). *A digital signature scheme* DS *consists of three algorithms* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *defined as follows:*

- $(\mathsf{vk}, \mathsf{sk}) \longleftarrow \mathsf{KeyGen}(1^\lambda)$*: on input a security parameter* λ*, the key-generation algorithm outputs a pair of verification and signing keys* $(\mathsf{vk}, \mathsf{sk})$*.*
- $\sigma \longleftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{M})$*: on input a signing key* sk *and a message* M*, the signing algorithm outputs a signature* σ*.*
- $b \in \{0,1\} \longleftarrow \mathsf{Vrfy}(\mathsf{vk}, \mathsf{M}, \sigma)$*: on input a verification key* vk*, a message* M *and a signature* σ*, the verification algorithm outputs 1 (accept) or 0 (reject).*

We require a signature scheme DS to satisfy the following two properties.

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

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

Security. We define *existential unforgeability under chosen message attack* (or $\mathsf{EUF\text{-}CMA}$ for short) by 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 $\mathcal{S} = \emptyset$.

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 $\mathcal{S} \longleftarrow \mathcal{S} \cup \{(\mathsf{M}, \sigma)\}$.

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 $\mathsf{EUF\text{-}CMA}-$*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:

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

Via the Fiat-Shamir transform [FS86], a Σ-protocol S for a binary relation $\mathcal{R}$ can be turned into a digital signature scheme. The resulting scheme $FS(\mathsf{S})$ 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 $FS(\mathsf{S})$ can be proved EUF-CMA secure.

2.3 The Σ_{SEC} protocol

In this section we describe the Σ-protocol Σ_{SEC}, which is a diminished variant of $\Sigma^{\mathsf{base}}_{\mathsf{SECUER}}$ from [BCC+22, Sec. 4] (see Remark 5 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 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 $E_0 : y^2 = x^3 + 6x^2 + x$), $\{P_1, Q_1\}$ 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:

- $(E_1, \varphi) \longleftarrow \mathsf{Gen}(1^\lambda)$: 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 $\varphi : E_0 \longrightarrow E_1 := E_0/\langle P_1 + [s]Q_1\rangle$ having $\langle P_1 + [s]Q_1\rangle$ as kernel. It returns the statement-witness pair (E_1, φ).
- $\mathsf{com} \longleftarrow \mathsf{P}_1(E_1, \varphi)$: given a statement E_1 and a corresponding witness φ, the prover P_1 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 $\mathsf{com}_1 \longleftarrow \mathcal{O}(\mathsf{Com}||E_2||b_2)$ and $\mathsf{com}_2 \longleftarrow \mathcal{O}(\mathsf{Com}||E_3||b_3)$ via the random oracle $\mathcal{O}$. The output is $\mathsf{com} = (\mathsf{com}_1, \mathsf{com}_2)$.
- $\{-1, 0, 1\} \longleftarrow \mathsf{V}_1(\mathsf{com})$: on input a commitment com, V_1 outputs a random challenge $\mathsf{ch} \in \{-1, 0, 1\}$.
- $\mathsf{resp} \longleftarrow \mathsf{P}_2(E_1, \varphi, \mathsf{com}, \mathsf{ch})$: 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.
- $1/0 \longleftarrow \mathsf{V}_2(E_1, \mathsf{com}, \mathsf{ch}, \mathsf{resp})$: 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 $\mathsf{com}_1 = \mathcal{O}(\mathsf{Com}||E||b)$. 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 $\mathsf{com}_2 = \mathcal{O}(\mathsf{Com}||E||b)$. 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 $\tilde{E}$, $\mathsf{com}_1 = \mathcal{O}(\mathsf{Com}||E||b)$ and $\mathsf{com}_2 = \mathcal{O}(\mathsf{Com}||\tilde{E}||\tilde{b})$. 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

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

and the relaxed relation

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

It is not difficult to prove that the Σ-protocol Σ_{SEC} described above 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, Σ_{SEC} is computationally HVZK.

Problem 4 (Decisional Supersingular Product Problem). Let $\varphi : E_0 \longrightarrow E_1$ be an isogeny of degree $\ell_1^{e_1}$. Given (E_2, E_3, ψ) sampled with probability $1/2$ from one of the following distributions, the decisional supersingular product problem $\mathrm{DSSP}_{\mathsf{pp}}$ requires to determine which distribution it is from:

- choose a random point $R \in E_0[\ell_2^{e_2}]$ of order $\ell_2^{e_2}$. Let $\psi : E_0 \longrightarrow E_2$ and $\psi' : E_1 \longrightarrow E_3$ be the isogenies with kernels $\langle R \rangle$ and $\langle \varphi(R) \rangle$, respectively. Then let $\varphi' : E_1 \longrightarrow E_2$ be the isogeny having $\langle \psi(\mathsf{Ker}(\varphi)) \rangle$ as kernel, where $\mathsf{deg}(\phi') = \ell_1^{e_1}$.
- 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 $\varphi' : E_2 \longrightarrow E_3$ having $\langle U \rangle$ as kernel.

Remark 5. Within the protocol $\Sigma^{\mathsf{base}}_{\mathsf{SECUER}}$, 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 [BCC+22, Prop. 17]. The conditions to satisfy this stronger property heavily affect both the transcript size and the execution times. As in our work we are only intersted 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.

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

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

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

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: $\quad\quad \psi_i \leftarrow$ `IsogenyFromKernel`$(E_0, \langle P_2 + [r_i]Q_2\rangle)$
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** $\mathcal{O}(\mathsf{Com}||E||b) \neq \mathsf{com}_{i,1}$ **or** $\mathcal{O}(\mathsf{Com}||\tilde{E}||\tilde{b}) \neq \mathsf{com}_{i,2}$ **then**
13: $\quad\quad\quad$ **return** 0
14: **return** 1

Fig. 1. Algorithms which form 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, $E[\ell^e]_{max}$ denotes the point of order ℓ^e in $E[\ell^e]$ (when $\ell, e \in \mathbb{N}$ and ℓ is a prime) while the commitment scheme C is modelled by a random oracle $\mathcal{O}$.

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

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

where the first term is size of the commitment, the second one that of the challenge and the third one that of the response (the terms within the brackets correspond to the sizes of the responses to challenge -1,1 and 0, respectively).

SIKE parameters	$\lceil \log p \rceil$	λ	t	Transcript length	Signature length
SIKEp434	434	128	218	543692	543256
SIKEp503	503	128	218	606404	605968
SIKEp610	610	192	326	1163277	1162625
SIKEp751	751	256	435	1956775	1955905

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

Within an execution of Σ^t_{SEC}, 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^t_{SEC} is turned into the digital signature scheme $\mathsf{DS}_{\mathsf{SEC}}$ via the Fiat-Shamir transform, the security of $\mathsf{DS}_{\mathsf{SEC}}$ is guaranteed by the hardness of the relation $\tilde{\mathcal{R}}_{\mathsf{SEC}}$, 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 $\mathsf{DS}_{\mathsf{SEC}}$ is just a transcript of Σ^t_{SEC} 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^t_{SEC}, and the computational cost to sign is that undergone by the prover in an execution of Σ^t_{SEC}, plus one hash-function evaluation. Therefore, the efficiency analysis presented above applies almost directly to $\mathsf{DS}_{\mathsf{SEC}}$ (see the last column of Table 1 for the signature sizes of $\mathsf{DS}_{\mathsf{SEC}}$ for different SIDH/SIKE parameters).

3 Signature-size Optimisations

In this section, we apply some known cryptographic techniques to $\mathsf{DS}_{\mathsf{SEC}}$ in order 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 $\mathsf{DS}_{\mathsf{SEC}}$.

3.1 Challenge and commitment recoverability

A Σ-protocol $(\mathsf{Gen}, \mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2))$ is said to be *commitment-recoverable* if, with overwhelming probability over the random choice of a statement-witness pair $(\mathsf{x}, \mathsf{w}) \longleftarrow \mathsf{Gen}(1^\lambda)$, for any $\mathsf{ch} \in \mathsf{ChSet}$ and $\mathsf{resp} \in \mathsf{ResSet}$, there

exists a unique commitment $\mathsf{com} \in \mathsf{ComSet}$ that makes $(\mathsf{x}, \mathsf{com}, \mathsf{ch}, \mathsf{resp})$ a valid transcript and such a commitment can be publicly computed by means of an algorithm taking $(\mathsf{x}, \mathsf{ch}, \mathsf{resp})$ 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^t_{SEC} - described in Figure 1 - does not satisfy commitment recoverability (for example, the response resp_i when $\mathsf{ch}_i = -1$ does not allow to recover $\mathsf{com}_{i,2}$). However, we can modify (Σ_{SEC} and) Σ^t_{SEC} in such a way that the new protocol(s) are commitment-recoverable.

The modification of Σ^t_{SEC} which we suggest[5] is detailed in Figure 2. In particular, P_1 remains unchanged, while the algorithm P_2 outputs the response $\mathsf{resp}_i = (r_i, b_{2,i}, \mathsf{com}_{i,2})$ when $\mathsf{ch}_i = -1$; the response $\mathsf{resp}_i = (\varphi(P_2 + [r_i]Q_2), b_{3,i}, \mathsf{com}_{i,1})$ when $\mathsf{ch}_i = 1$; the response $\mathsf{resp}_i = (E_{2,i}, \phi_i(\mathsf{Ker}(\varphi)), b_{2,i}, b_{3,i})$ when $\mathsf{ch}_i = 0$, 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 components of a transcript of the modified Σ^t_{SEC} protocol (excluding the statement) are approximated by

$$\begin{aligned} &|\mathsf{com}| = 4\lambda t, \qquad |\mathsf{ch}| = \lceil \log(3) t \rceil, \\ &|\mathsf{resp}| = \tfrac{t}{3}\big((\lceil \log(\ell_2^{e_2}) \rceil + 3\lambda) + (2\lceil \log p \rceil + 1 + 3\lambda) + (4\lceil \log p \rceil + 1 + 2\lambda) \big) \end{aligned} \tag{3}$$

(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 parameters SIKEp434, SIKEp503, SIKEp610 and SIKEp751.

SIKE param.	$\lceil \log p \rceil$	λ	t	$\lvert\mathsf{com}\rvert$	$\lvert\mathsf{ch}\rvert$	$\lvert\mathsf{resp}\rvert = \lvert\text{signature}\rvert$	gain
SIKEp434	434	128	218	111616	10	279622	48.53%
SIKEp503	503	128	218	111616	10	312249	48.47%
SIKEp610	610	192	326	250368	10	597993	48.57%
SIKEp751	751	256	435	445440	11	1005575	48.59%

Table 2. Sizes (in bits) of the signatures of $\mathsf{DS}_{\mathsf{SEC}}$ (i.e. the transcripts of Σ^t_{SEC} excluding statements and challenges) produced by the modified Σ^t_{SEC} 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.

Remark 6. The gain column in Table 2 indicates how many bits (in percentage) we save when storing the response computed by the modified Σ^t_{SEC} 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^t_{SEC}, and s_1 is the

[5] We stress that the modification we suggest is analogous to the one proposed in [CD22b, Sec. 3.2] for the protocol Σ_{wSIDH}.

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

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** $\mathsf{com} \leftarrow (\mathsf{com}_{i,1}, \mathsf{com}_{i,2})_{i=1}^t$

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

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

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

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

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

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^t_{SEC} 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^t_{SEC} 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^t_{SEC} 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^t_{SEC} protocol is exactly the same as in the original Σ^t_{SEC} protocol.

3.2 Compressed responses

In Σ^t_{SEC}, 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 $\psi'_i : E_1 \longrightarrow E_{3,i}$. 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 [CLN16,AJK+16], 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 $\iota_i := 1$ and $\gamma_i := \alpha_i^{-1}\beta_i$, otherwise we let $\gamma_i := 1$ and $\iota_i := \beta_i^{-1}\alpha_i$. 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 components, α, β and γ, ω, respectively, with respect to 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 for the 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

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

Remark 7. The above 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.

SIKE param.	$\lceil \log p \rceil$	λ	t	$\lvert\mathsf{resp}\rvert = \lvert\text{signature}\rvert$	gain
SIKEp434	434	128	218	232388	8.69%
SIKEp503	503	128	218	257531	9.03%
SIKEp610	610	192	326	498563	8.55%
SIKEp751	751	256	435	842740	8.33%

Table 3. Sizes (in bits) of the responses produced by the modified Σ^t_{SEC} with compressed responses for different SIDH/SIKE parameters. Commitment and challenge lengths remain unchanged w.r.t. 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.

3.3 Seed trees

A primitive called *seed tree* [BKP20] 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 $\mathsf{Expand} : \{0,1\}^{\lambda+\lceil \log_2(t-1)\rceil} \to \{0,1\}^{2\lambda}$ is a Pseudorandom Generator (PRG, is short) for any $\lambda, t \in \mathbb{N}$, instantiated by a random oracle $\mathcal{O}$.

- $\mathsf{SeedTree}^{\mathcal{O}}(\mathsf{seed}_{\mathsf{root}}, t) \to \{\mathsf{leaf}_i\}_{i\in\{1,\ldots,t\}}$: On input a root seed $\mathsf{seed}_{\mathsf{root}} \in \{0,1\}^\lambda$ 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 children seeds. Calls to the random oracle are of the form $\mathcal{O}(\mathsf{Expand}\|\mathsf{seed}\|h)$, where $h \in \{1,\ldots,t-1\}$ identifies the position of seed in the binary tree. The algorithm finally outputs the list of seeds associated with the t leaves.
- $\mathsf{ReleaseSeeds}^{\mathcal{O}}(\mathsf{seed}_{\mathsf{root}}, \mathbf{c}, j) \to \mathsf{seeds}_{\mathsf{internal}}$: On input a root seed $\mathsf{seed}_{\mathsf{root}} \in \{0,1\}^\lambda$, a bit string $\mathbf{c} \in \{-1,0,1\}^t$, and $j \in \{-1,0,1\}$, it outputs the list of seeds $\mathsf{seeds}_{\mathsf{internal}}$ 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 $\mathsf{seeds}_{\mathsf{internal}}$ is coupled with an index which identifies its position in the binary tree.

– $\mathsf{RecoverLeaves}^{\mathcal{O}}(\mathsf{seeds}_{\mathsf{internal}}, \mathbf{c}, j) \to \{\overline{\mathsf{leaf}_i}\}_{i \text{ s.t. } c_i=j}$: On input a set $\mathsf{seeds}_{\mathsf{internal}}$, 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 $\mathsf{seeds}_{\mathsf{internal}}$.

By construction, the leaves $\{\mathsf{leaf}_i\}_{i \text{ s.t. } c_i=j}$ output by $\mathsf{SeedTree}(\mathsf{seed}_{\mathsf{root}}, t)$ are the same as those output by $\mathsf{RecoverLeaves}(\mathsf{ReleaseSeeds}(\mathsf{seed}_{\mathsf{root}}, \mathbf{c}, j), \mathbf{c}, j)$ for any $\mathbf{c} \in \{0,1\}^t$ and $j \in \{-1,0,1\}$. We observe that the last algorithm $\mathsf{SimulateSeeds}$ 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 $\mathsf{seeds}_{\mathsf{internal}}$, $\mathbf{c}$ and j.

We now describe how seed trees can be used to optimise the modified Σ^t_{SEC} 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 $\{r_1, \ldots, r_t\} \xleftarrow{\$} (\mathbb{Z}/\ell_2^{e_2}\mathbb{Z})^t$ for the commitment curves $(E_{2,i}, E_{3,i})_{i=1}^t$ and two sets of random strings $(b_{2,1}, \ldots, b_{2,t})$, $(b_{3,1}, \ldots, b_{3,t}) \xleftarrow{\$} \{0,1\}^{\lambda t}$. 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 $\mathsf{seed}^{\mathsf{coeff}}_{\mathsf{root}}$ and one for the random bit-strings which originates from the root $\mathsf{seed}^{\mathsf{str}}_{\mathsf{root}}$. 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 $\mathsf{seed}^{\mathsf{str}}_{\mathsf{root}}$ 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 $\mathsf{seeds}_{\mathsf{internal}} \leftarrow \mathsf{ReleaseSeeds}(\mathsf{seed}_{\mathsf{root}}\mathsf{coeff}, \mathsf{ch}, -1)$. The verifier would then use $\mathsf{seeds}_{\mathsf{internal}}$ along with ch and $j = -1$ to recover the required seeds by running $\mathsf{RecoverLeaves}$.

Let us analyse how generating all random strings from a single root seed $\mathsf{seed}^{\mathsf{str}}_{\mathsf{root}}$ 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 $\mathsf{seed}^{\mathsf{str}}_{\mathsf{root}}$, 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 $r_1, \ldots, r_t$, since the amount of internal seeds that need to be revealed depends on how -1, 0 and 1 are distributed over the challenge string. In the worst-case scenario, i.e. when all the leaf seeds that are expanded to the coefficients r_i with $\mathsf{ch}_i = -1$ need to be revealed, in-

stead 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^t_{SEC} 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 lenghts 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 $\mathsf{seed}^{\mathsf{str}}_{\mathsf{root}}$.

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

In Table 4 we report the numbers produced by Equation (5) for different SIKE parameters. With respect to Table 3, only the columns labelled by "$|\mathsf{resp}| = |\text{signature}|$" and "gain" are modified.

SIKE param.	$\lceil \log p \rceil$	λ	t	$\lvert\mathsf{com}\rvert$	$\lvert\mathsf{ch}\rvert$	$\lvert\mathsf{resp}\rvert = \lvert\text{signature}\rvert$	gain
SIKEp434	434	128	218	111616	10	188771	8.03%
SIKEp503	503	128	218	111616	10	211370	7.62%
SIKEp610	610	192	326	250368	10	403020	8.22%
SIKEp751	751	256	435	445440	11	676681	8.49%

Table 4. Sizes (in bits) of the different components of the transcripts (excluding the statement) produced by the modified Σ^t_{SEC} 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.

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 [BKP20, 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 $K \in \{0, ..., \lceil t/3 \rceil\}$ 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 8. *We can express* $\mathfrak{n}$ *as follows*

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

Proof. Let S be the set of all subsets U of $\{-1, 0, 1\}^M$ composed by 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.

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

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).

Proposition 9. *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 $\{K, ..., M\}$, 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 $h \geq K$,

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

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

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

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.

SIKE parameters	$\lceil \log p \rceil$	λ	M	K	$h_{\max}$	$\lvert\mathsf{resp}\rvert$	gain
SIKEp434	434	128	250	48	96	154783	6.26%
SIKEp503	503	128	250	48	96	169041	6.99%
SIKEp610	610	192	362	76	152	336095	5.76%
SIKEp751	751	256	478	103	206	571453	5.38%

Table 5. Values of M and K for the unbalanced challenge space that minimise the response length of the modified Σ^M_{SEC} 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.

In order 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 $K \in \{0, ..., \lceil t/3 \rceil\}$ and pick out the values (M, K) minimizing $\lvert\mathsf{resp}\rvert$. 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 66.31KB to at most 18.89KB, corresponding to a reduction of at least 71.51%
- for SIKEp503 we shorten the signature from 73.97KB to at most 20.63KB, corresponding to a reduction of at least 72.10%
- for SIKEp610 we shorten the signature from 141.92KB to at most 41.03KB, corresponding to a reduction of at least 71.09%
- for SIKEp751 we shorten the signature from 238.76KB to at most 69.76KB, 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 [BCC+22, Sec. 5] built on top of $\Sigma^{\mathsf{base}}_{\mathsf{SECUER}}$ to collaboratively produce a random supersingular elliptic curve whose endomorphism ring is hard to compute even for the parties who did the sampling.

4 Running-time optimisations

In an execution of the Σ-protocol Σ^t_{SEC}, $2t$ 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^M_{SEC} (see Section 3.4). All such isogenies have degree $\ell_2^{e_2}$; half of them originate from E_0, 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^t_{SEC} (and, implicitly, on Σ^M_{SEC}), both optimisations could be extended to the distributed trusted-setup protocol [BCC+22, Sec. 5] built on top of $\Sigma^{\mathsf{base}}_{\mathsf{SECUER}}$.

In order 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 $R := P + [r]Q$ 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 $\psi^{(b-1)}(\ldots(\psi^{(1)}(R))\ldots)$.

The strategies described in [ACC+20, 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. In order 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 to the bottom-left corner, represent the kernels of $\psi^{(1)}, \psi^{(2)}, \ldots, \psi^{(b)}$, 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 [ACC+20, Appendix D] for the case $b = 6$ (the general strategy for a generic b is just a generalisation of it).

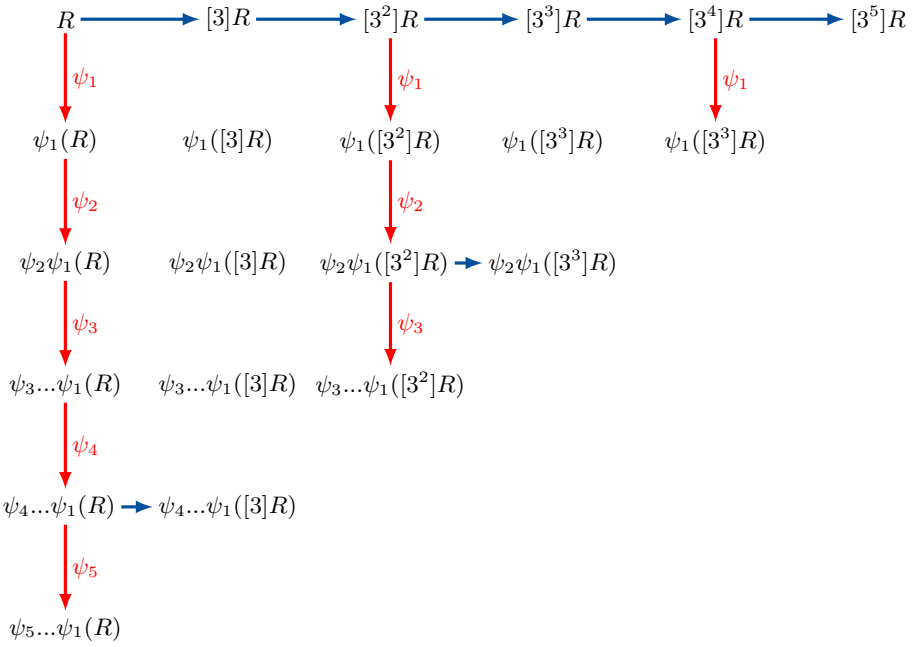

Fig. 3. Graphical representation of the optimal strategy proposed in [ACC+20, Appendix D] for the case $b = 6$ to compute a 3^6-isogeny from a kernel generator R. Blue arrows represent scalar multiplications by 3, red arrows represent 3-isogeny evaluations.

4.1 Computing several isogenies in parallel

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

As we saw above, for $i \in \{1, \ldots, t\}$, 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[6]. In particular, for $j \in \{1, \ldots, b\}$ and two different $i, i' \in \{1, \ldots, t\}$, it holds that

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

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 $r_i \pmod{3^5}$ and $t = 218$, the fourth is the last factor where we can still expect a good amount[7] of repetitions.

Given the above 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 [ACC+20, 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.

In order 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 [ACC+20, Appendix D] to compute isogenies of degree 3^b:

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

[6] 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.

[7] 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.

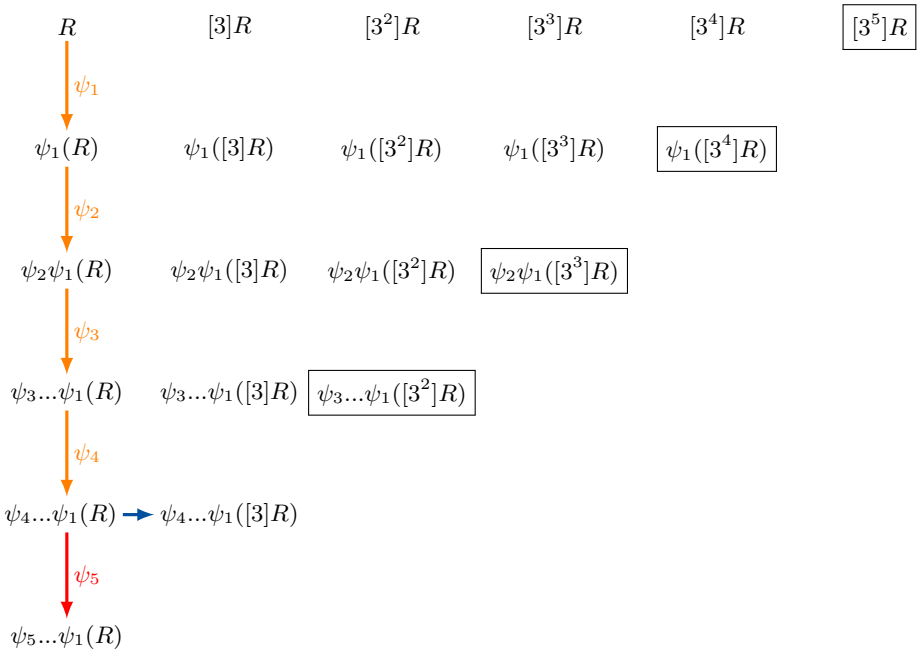

Fig. 4. By doing some pre-computation or avoiding the multiple computation of 3-isogenies, the optimal strategy from [ACC+20, Appendix D] to compute and evaluate a 3^6-isogeny from a kernel generator R is granted the possibility to move from line 1 to line $\alpha+1$ (with $\alpha = 3$ in the figure) without the need to reach the right most of the first α lines. In particular, the points in boxes can obtained instantly from modular-arithmetic checks, and they determine, in turn, the vertical orange arrows.

We borrow from [ACC+20, 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 [ACC+20, 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 6. For example, for SIKEp434, t is equal to 218, $b = 137$, $p_3 = 5322$ and $q_3 = 5282$, with the total cost for computing t isogenies of degree 3^{137} via the optimal strategy being

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

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 above, 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, $H'_{\text{os}}(137) = H_{\text{os}}(137-4)$ horizontal arrows and $V'_{\text{os}}(137) = V_{\text{os}}(137-4)+4$ vertical arrows are required to computate a single 3^{137}-isogeny. The total cost

of computing 218 isogenies of degree 3^{137} in parallel is

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

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 6, where we also indicate the cost of storing the pre-computed kernel generators of all possible initial steps $\psi_1, \dots \psi_4$.

Protocol	SIKE parameters	t	b	old cost (cc)	new cost (cc)	gain	storage (KB)
Σ^t_{SEC}	SIKEp434	218	137	5716545548	5400988804	5.52%	13.035
	SIKEp503	218	159	7642101180	7275879492	4.79%	15.105
	SIKEp610	326	192	16365527304	15704201096	4.04%	55.41
	SIKEp751	435	239	33908315250	32272252410	4.82%	68.2
$\Sigma^{t,U}_{\mathsf{SEC}}$	SIKEp434	250	137	6555671500	6019478500	8.18%	39.43
	SIKEp503	250	159	8763877500	8140531500	7.11%	45.65
	SIKEp610	362	192	18172763448	17438407352	4.04%	55.41
	SIKEp751	478	239	37260171700	35462383108	4.82%	68.2

Table 6. Costs (in clock cycles and kilobytes of storage) and percentage gains in using the pre-computation tweak to compute t commitment isogenies in Σ^t_{SEC} and in its unbalanced challenge variant $\Sigma^{t,U}_{\mathsf{SEC}}$ compared to those of using the optimal strategies from [ACC+20, Appendix D], for different SIKE parameter sets.

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 initial steps. When computing the second isogeny, we first check whether $r_2 \equiv r_1 (\mod 3^j)$ 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 $\psi_1, \dots, \psi_t$ 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 3 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 9 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 get the 81 possibilities for $\psi_i^{(4)}$. The total cost of this strategy is therefore

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

for SIKEp434 and SIKEp503 and

$$t \cdot (V_{\text{os}}(b-5) \cdot q_3 + 5 \cdot q_3 + H_{\text{os}}(b-5) \cdot p_3) + (264 \cdot b - 1254) \cdot p_3 + 90 \cdot q_3,$$

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 precomputation strategy. The results for all SIKE parameter sets are presented in Table 7.

SIKE parameters	t	b	old cost (cc)	new cost (cc)	gain	storage (KB)
SIKEp434	218	137	5716545548	5422890556	5.14%	13.035
SIKEp503	218	159	7642101180	7301293764	4.46%	15.105
SIKEp610	326	192	16365527304	15967764224	2.43%	55.41
SIKEp751	435	239	33908315250	32601850914	3.85%	68.2

Table 7. Costs (in clock cycles and kilobytes of storage) and percentage gains in computing t commitment isogenies in Σ^t_{SEC} following the strategy that avoids precomputation but performs modular-arithmetic checks on-the-fly, compared to those of the optimal strategy from [ACC+20, Appendix D], for different SIKE parameter sets.

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^t_{SEC} (or in one of its variants), and are the kernel generators $R_1, R_2, \dots R_t$, 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 get 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 $r_i = (r_i^0, r_i^1, \dots, r_i^{m-1})_2$ is the little-endian binary representation of $r_i \in \mathbb{Z}/\ell_2^{e_2}\mathbb{Z}$.

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

for $i = 1, 2, \ldots, t$ **do**
  $R_i \leftarrow 0$
for $j = 0, 1, \ldots, m-2$ **do**
  for $i = 1, 2, \ldots, t$ **do**
    if $r_i^j = 1$ **then**
      $R_i \leftarrow R_i + Q_2$
  $Q_2 \leftarrow [2]Q_2$
for $i = 1, 2, \ldots, t$ **do**
  if $r_i^{m-1} = 1$ **then**
    $R_i \leftarrow R_i + Q_2$
  $R_i \leftarrow R_i + P_2$
return $(R_1, R_2, \ldots, R_t)$

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

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

In this note we have assessed the efficiency of a SIDH-based digital signature built on a weaker but more efficient variant of the recent identification protocol $\Sigma^{\mathsf{base}}_{\mathsf{SECUER}}$ from[BCC+22, Sec. 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 [BCC+22, Sec. 5] built

on top of $\Sigma^{\mathsf{base}}_{\mathsf{SECUER}}$ to collaboratively produce a random supersingular elliptic curve whose endomorphism ring is hard to compute even for the parties who did the sampling.

References

ACC+20. Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalali, David Jao, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Geovandro Pereira, Joost Renes, Vladimir Soukharev, and David Urbanik. Supersingular isogeny key encapsulation november 30, 2017. Third Round Candidate of the NIST's post-quantum cryptography standardization process. Available at: https://sike.org/, 2020.

AJK+16. Reza Azarderakhsh, David Jao, Kassem Kalach, Brian Koziel, and Christopher Leonardi. Key compression for isogeny-based cryptosystems. In Proceedings of the 3rd ACM International Workshop on ASIA Public-Key Cryptography, pages 1–10, 2016.

BCC+22. Andrea Basso, Giulio Codogni, Deirdre Connolly, Luca De Feo, Tako Boris Fouotsa, Guido Maria Lido, Travis Morrison, Lorenz Panny, Sikhar Patranabis, and Benjamin Wesolowski. Supersingular curves you can trust. Cryptology ePrint Archive, 2022/1469, 2022.

BKP20. Ward Beullens, Shuichi Katsumata, and Federico Pintore. Calamari and falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. In International Conference on the Theory and Application of Cryptology and Information Security, pages 464–492, 2020.

BKV19. Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. In Steven D. Galbraith and Shiho Moriai, editors, Advances in Cryptology – ASIACRYPT 2019, Part I, volume 11921 of Lecture Notes in Computer Science, pages 227–247, Kobe, Japan, December 8–12, 2019. Springer, Heidelberg, Germany.

CD22a. Wouter Castryck and Thomas Decru. An efficient key recovery attack on sidh (preliminary version). Cryptology ePrint Archive, Paper 2022/975, 2022. https://eprint.iacr.org/2022/975.

CD22b. Jesús-Javier Chi-Domínguez. A note on constructing sidh-pok-based signatures after castryck-decru attack. Cryptology ePrint Archive, Paper 2022/1479, 2022. https://eprint.iacr.org/2022/1479.

CDMP22. Jesús-Javier Chi-Domínguez, Victor Mateu, and Lucas Pandolfo Perin. Sidh-sign: an efficient sidh pok-based signature. Cryptology ePrint Archive, Paper 2022/475, 2022. https://eprint.iacr.org/2022/475.

CLN16. Craig Costello, Patrick Longa, and Michael Naehrig. Efficient algorithms for supersingular isogeny Diffie-Hellman. In Matthew Robshaw and Jonathan Katz, editors, Advances in Cryptology – CRYPTO 2016, Part I, volume 9814 of Lecture Notes in Computer Science, pages 572–601, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

CSCJR22. Jorge Chávez-Saab, Jesús-Javier Chi-Domínguez, Samuel Jaques, and Francisco Rodríguez-Henríquez. The SQALE of CSIDH: sublinear Vélu quantum-resistant isogeny action with low exponents. Journal of Cryptographic Engineering, 12(3):349–368, September 2022.

DDGZ21. Luca De Feo, Samuel Dobson, Steven D. Galbraith, and Lukas Zobernig. SIDH proof of knowledge. Cryptology ePrint Archive, Report 2021/1023, 2021. https://eprint.iacr.org/2021/1023.

DFJP14. Luca De Feo, David Jao, and Jérôme Plût. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. Journal of Mathematical Cryptology, 8.3:209–247, 2014.

DFKL+20. Luca De Feo, David Kohel, Antonin Leroux, Christophe Petit, and Benjamin Wesolowski. Sqisign: compact post-quantum signatures from quaternions and isogenies. International Conference on the Theory and Application of Cryptology and Information Security, 8.3:64–93, 2020.

DFLW22. Luca De Feo, Antonin Leroux, , and Benjamin Wesolowski. New algorithms for the deuring correspondence: Sqisign twice as fast. Cryptology ePrint Archive, 2022/234, 2022.

DG19. Luca De Feo and Steven D. Galbraith. SeaSign: Compact isogeny signatures from class group actions. In Yuval Ishai and Vincent Rijmen, editors, Advances in Cryptology – EUROCRYPT 2019, Part III, volume 11478 of Lecture Notes in Computer Science, pages 759–789, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany.

FS86. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings, volume 263 of Lecture Notes in Computer Science, pages 186–194. Springer, 1986.

Gal12. S.D. Galbraith. Mathematics of Public Key Cryptography. Cambridge University Press, 2012.

GPS17. Steven D Galbraith, Christophe Petit, and Javier Silva. Identification protocols and signature schemes based on supersingular isogeny problems. In ASIACRYPT, pages 3–33. Springer, 2017.

JAC+17. David Jao, Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalali, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Joost Renes, Vladimir Soukharev, and David Urbanik. Sike. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

MM22. Luciano Maino and Chloe Martindale. An attack on sidh with arbitrary starting curve. Cryptology ePrint Archive, Paper 2022/1026, 2022. https://eprint.iacr.org/2022/1026.

Rob22. Damien Robert. Breaking sidh in polynomial time. Cryptology ePrint Archive, Paper 2022/1038, 2022. https://eprint.iacr.org/2022/1038.

Sil09. Joseph H. Silverman. The arithmetic of elliptic curves. New York, Springer, Vol. 106, 2009.

YAJ+17. Youngho Yoo, Reza Azarderakhsh, Amir Jalali, David Jao, and Vladimir Soukharev. A post-quantum digital signature scheme based on supersingular isogenies. In FC, pages 163–181. Springer, 2017.