SPRINT: New Isogeny Proofs of Knowledge and Isogeny-Based Signatures

Thomas den Hollander(✉)[1] Shai Levin[2] Marzio Mula[1]
Robi Pedersen[3] Daniel Slamanig[1] Sebastian A. Spindler[1]

Research Institute CODE, Universität der Bundeswehr München
{thomasdh,marzio.mula,daniel.slamanig,s.spindler}@unibw.de
Chalmers University of Technology, University of Gothenburg
shai.levin@chalmers.se
Technical University of Denmark
robi.pedersen@protonmail.com

Abstract

Zero-knowledge proofs of knowledge are a fundamental building block in many isogeny-based cryptographic protocols, such as signature schemes based on identification-to-signature transformations, or multi-party ceremonies that avoid a trusted setup, in particular for generating supersingular elliptic curves with unknown endomorphism rings.

In this paper, we construct *SPRINT*, an efficient polynomial IOP-based proof system that encodes the radical 2-isogeny formulas into a system of multivariate polynomials. When combined with the recent polynomial commitment scheme (PCS) DeepFold, our construction yields substantial improvements over state-of-the-art isogeny proofs of knowledge. For the SQIsign NIST-I prime $p = 5 \cdot 2^{248} - 1$, our implementation takes only a few milliseconds for proving and verification, with proof sizes around 80 kB. Compared to the previous state-of-the-art proof system by den Hollander, Mula, Slamanig & Spindler (PQCrypto'26), we achieve speedups ranging from 1.1× to 22.7× for the prover and from 4.4× to 109× for the verifier, while achieving proof sizes that are 1.4× to 17.5× smaller, across different instantiations and parameter sets.

Moreover, we study the weak simulation extractability of our proof system, which we can use as a starting point for a modular construction of signatures. We show that any Fiat–Shamir compiled interactive proof with a so-called canonical simulator is weakly simulation-extractable. We expect this general result to be applicable to other proof systems and thus of independent interest.

Building on SPRINT and our wSE result, we introduce a new family of signature schemes whose security solely relies on the ℓ-isogeny path problem, a foundational problem in isogeny-based cryptography.

As a concrete instantiation, we construct a signature scheme using DeepFold as the PCS. Across the different parameter sets, a prototype implementation of our scheme sees significantly faster signing times than the highly optimized NIST specification for SQIsign, at the cost of much larger signature sizes and excluding a one-time precomputation phase. When this phase is instead included with the signing time, we still get almost on-par timings. Meanwhile our signature scheme relies on weaker assumptions and is constant-time when instantiated with a suitable PCS. Even though our signatures are large, the framework offers flexibility for tradeoffs and optimizations – both within a given PCS and by switching to alternative PCS constructions. In particular, it will naturally inherit efficiency gains from future advances in plausibly post-quantum secure PCS constructions.

1 Introduction

Isogenies are one of a handful of cryptographic paradigms that are presumed secure against attacks by both classical and quantum algorithms. Compared to its competitors, isogeny-based cryptography allows particularly small parameters and hence key sizes and signatures. On the other hand,

isogenies rank among the slowest protocols that achieve post-quantum security. This drawback is particularly noticeable in the context of proving knowledge of isogenies, which often requires a significant number of repetitions to provide reasonable soundness guarantees [BKV19, BCC+23, DDGZ22, BDGP23].

An alternative proof of knowledge can be built around knowledge of the endomorphism ring of a given curve E. Such proofs have become particularly practical since the SIDH attacks [CD23, MMP+23, Rob23] and the resulting new representation of isogenies through higher-dimensional isogenies between abelian varieties, see e.g. [Rob24]. This approach has been used in signatures, such as PRISM [BBC+25, BBC+26] and the more recent variants of SQIsign [DLRW24, DF24, NOC+24, BDD+24, NO25, XLZO25], resulting in very compact signatures and good computational efficiency. However, secret knowledge of an endomorphism ring $\mathrm{End}(E)$ is only possible by walking from an elliptic curve of known endomorphism ring via a (secret) isogeny $\phi\colon E_0 \to E$, see e.g. [BBD+22, MMP24]. Usually, the curve E_0 is public and the distinguished curve of j-invariant 1728.

For many applications, however, the starting curve explicitly has to be of unknown endomorphism ring, such as in the CGL hash function [CLG09], in the (pre-quantum) verifiable delay function [DMPS19] and delay encryption [BD21] schemes, in oblivious transfer over $\mathbb{F}_p$ [LGD21], isogeny commitment schemes [Ste22], public-key encryption [FMP23, Mor23], oblivious pseudo-random functions [Bas24], and verifiable random functions [LP25]. Particularly, the last two also employ proof systems without having the power to leverage the endomorphism ring to their advantage.

Knowledge of the endomorphism ring of curves is also an issue in protocols between multiple parties, where parties sequentially compute chains of secret isogenies, and cannot know the endomorphism ring of these intermediate curves. This is relevant in multi-party trusted parameter generation [BCC+23] as well as distributed key generation and threshold signature algorithms [CS20, BDPV21, CM22, ABCP23], where they play the role of proofs-of-possession (PoPs) – which can also be used, in a more applied (post-quantum) setting, to certify KEM keys for KEMTLS [SSW20]. The multi-party parameter generation proposed by [BCC+23] ended up presenting proof systems of almost 20 seconds per party for NIST level I security. This result motivated [CLL23] to use generic proof systems as an alternative to expensive proofs tailored to specific relations.[1] With the goal of also achieving statistical zero-knowledge in a tailor-made approach, [CLL23] reduced these costs down to about a second, a figure which was recently even further improved by [dHKM+25, LP25, dHMSS26]. However, there still remains the question whether these results can be improved to obtain highly efficient isogeny proofs of knowledge without the requirement of knowing endomorphism rings.

In addition to efficiency and generality, a property that over the last years turned out to be highly relevant for the practical use of zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARKs) is that they are non-malleable, i.e., that modifying proofs (or statements in a weaker variant) without the knowledge of the corresponding witness is prevented. This is covered by the soundness notion of (weak) simulation-extractability [Sah99, DDO+01, Gro16], which guarantees that no cheating prover can break knowledge soundness even after asking a ZK simulator to produce proofs on adaptively chosen statements. While simulation extractability (SE) has become a very popular and important property for zk-SNARKs (cf. [FFK+23, KPT23, FFR24, CFR25]), research has mostly focused on the strong variant, which is usually harder to prove or satisfy. Moreover, in the context of isogeny proofs of knowledge, the SE property has so far not been studied at all. Interestingly, once we have a proof system that is at least weakly SE, we automatically obtain a new class of isogeny-based signatures.

1.1 Previous Work

An overview of tailor-made isogeny proofs of knowledge can be found in the 2023 survey paper [BDGP23] and the further references in the introduction. In this section, we focus on the use of generic proof systems for proving isogeny-knowledge.

Proving Isogenies using SNARKs. The first use of succinct non-interactive arguments (SNARGs) for isogeny proofs was proposed in [CSRT22]. In this work, the authors proposed to use a sumcheck

[1] We note that [MJ23] provides an alternative way of producing explicitly $\mathbb{F}_p$-rational elliptic curves via walks in the $\mathbb{F}_p$-rational graph, where their protocols rely on an ad-hoc adaptation of the Knowledge-of-Exponent Assumption (KEA) to CSIDH.

protocol based on modular polynomial relations in the 2-isogeny graph. Since the goal was to obtain a verifiable delay function (VDF), the suggested proof system did not need to satisfy knowledge soundness nor the zero-knowledge property. Furthermore, the practicality of their results was never properly assessed.

A few years later, Cong, Lai and Levin [CLL23] proposed a different approach to prove statements of the same relation. Instead of building their own proof system, the authors showed that expressing modular polynomial relations as R1CS constraints, and plugging them into the existing zk-SNARKs Aurora [BCR+19], Ligero [AHIV17] and Limbo [DOT21], proved very fruitful. The authors managed to construct a proof system for a NIST-level I instance that runs in just above a second.

These results in turn sparked more research into optimizing the R1CS constraints, with three works trying to reduce the number of constraints using different approaches. The authors in [dHKM+25] use canonical modular polynomials instead of the classical ones to obtain smaller constraint systems. As an additional contribution, the authors also propose constraint systems for isogenies of the higher degrees $\ell \in \{3, 5, 7, 13\}$. In the follow-up work [dHMSS26], the authors switch to the Atkin modular polynomials to achieve even smaller constraint systems and to extend the list of usable prime degrees further. A different path is taken in the work [LP25], which leverages the recent radical isogeny formulas [CD20, CDV20] instead of a modular polynomial for $\ell = 2$. These radical isogeny formulas have the additional advantage that they automatically prevent backtracking, and due to their simplicity they yield again a more efficient proof system.

A parallel approach is explored in [MNV24], where isogeny walks, computed via Vélu's formulas, are arithmetized and proven using the verifiable folding scheme Nova [KST22]. In multiparty protocols, this lets per-party proofs be folded into a single recursive proof. In the authors' implementation, proving a 2^{200}-isogeny takes around 2 minutes. The main limitation of this approach is that Nova is discrete logarithm based. Although there are alternative post-quantum folding schemes [BC24, FKNP24, BC25], their use would come with the price of extra lattice-based assumptions to prove knowledge of isogenies, which is unwanted.

While the recent advancements in isogeny proofs of knowledge are promising, there is still the question whether one can further improve their efficiency. Moreover, as mentioned above, these proof systems so far do not explicitly consider simulation extractability. One consequence is that they (currently) cannot be directly used as a basis for signature schemes that are efficient and only based on well-established assumptions.[2]

Isogeny-based signatures. The recent PRISM [BBC+25, BBC+26] scheme and the latest variants of SQIsign [DLRW24, DF24, NOC+24, BDD+24, NO25, XLZO25] are currently the most promising isogeny-based signature schemes. Their compactness allows them to rival even some post-quantum signature schemes standardized by NIST (i.e., ML-DSA based on Dilithium [LDK+22] or FN-DSA based on Falcon [PFH+22]), while their efficiency – notoriously one of the bottlenecks of isogeny-based protocols – has significantly improved over the previous state-of-the-art. On the downside, both PRISM and SQIsign rely on non-standard security assumptions. Namely, for SQIsign proving zero-knowledge requires either interacting with *ad hoc* oracles or adopting new variants of the classic isogeny problems and the Fiat–Shamir heuristic [ABD+25]. PRISM instead follows the hash-and-sign paradigm (inspired by [Ler25]), but when proven in the standard model requires a new interactive assumption (i.e., adaptive access to a so-called special degree isogeny oracle). When switching to the random oracle model, this interactive assumption can be replaced by a new non-static "q-type" assumption with the parameter q being linear in the number of signing and hash queries.

To find an isogeny-based signature scheme that relies on more standard isogeny problems, we need to take a step back to a conceptual ancestor of SQIsign, the GPS signature scheme introduced in [GPS17, §4]. Here, as in SQIsign, an identification protocol is turned into a signature scheme via the Fiat–Shamir (resp. Unruh) transform in the ROM (resp. QROM). The bottleneck is that the underlying identification scheme has a binary challenge space, which forces λ repetitions and hence yields signature sizes quadratic in λ, making the scheme impractical for both size and speed. Hence the current state of affairs in isogeny-based signatures is that practically efficient schemes whose

[2] One exception would be to apply the Bellare–Goldwasser template [BG90] for which a sound NIZK suffices. However, this would introduce additional assumptions (for the commitments and PRF), increase signature size and decrease performance.

security is only based on standard isogeny assumptions are not available.

1.2 Our Contributions

We first prove a general result establishing weak simulation-extractability for Fiat–Shamir-compiled interactive proofs, which we expect to be widely applicable to post-quantum proof systems that are used in practice.

We then present a new zero-knowledge argument for the knowledge of an isogeny, obtained by arithmetizing radical isogeny formulas via a tailored polynomial interactive oracle proof (PIOP). When used as a building block for multiparty generation of supersingular elliptic curves of unknown endomorphism ring (SECUERs), we obtain substantial improvements over the state of the art.

Finally, we present a family of isogeny-based signature schemes, which can be instantiated with any polynomial commitment scheme (PCS). Our approach allows us to use a more standard security assumption than SQIsign and PRISM, at the cost of larger signatures. We detail our contributions below.

Simulation-extractability of NARGs with canonical simulators. Motivated by the application of proof systems to construct signature schemes, we look at the non-malleability [Sah99, DDO+01] and in particular simulation extractability [Gro06] of proof systems. Recently, most research has focused on *strong simulation extractability* (sSE), i.e., where a malicious prover only succeeds by creating a new proof π despite having access to the simulator. There is extensive literature on this version of SE, falling roughly into three categories: either they provide a compiler to make any proof system SE [Sah99, DDO+01, KZM+15, ARS20, GKO+23], they provide a general framework for proving existing SNARKs SE [FKMV12, GKK+22, GOP+22, DG23, KPT23, FFR24, CFR25, GOP+25], or they build or prove a specific SNARK to provide SE [Gro06, GM17, AB19]. The first category incurs additional overhead, while the second type usually requires some form of unique prover responses, which is not satisfied by many proof systems. For signature schemes however, it is actually sufficient to start from a proof system that provides *weak simulation extractability* (wSE) [KZM+15, BKSV21], where an adversary succeeds by forging a proof for a statement for which the simulator was not queried. This has been shown to hold for Groth16 [BKSV21]. A more general result is [CF24], which relies on the straight-line extractability of Merkle trees and round-by-round knowledge soundness to prove (a weakly simulation extractable version of) UC-security of the Micali [Mic00] and BCS [BCS16] constructions.

In this work we show that, in fact, *any* Fiat–Shamir-compiled interactive proof with a so-called canonical simulator is weak SE. Such a simulator only programs the random oracle to obtain verifier challenges for a Fiat–Shamir transformed protocol, and this method of simulation is common among proof systems and much easier to verify than, for example, unique response. We thus expect this result to be directly applicable to many other post-quantum proof systems.[3] Where [CF24] gives sufficient conditions for PCP and IOPs to be UC-secure, our result bounds the advantage of a simulation extractability adversary compared to the regular extractability adversary with regard to any Fiat–Shamir compiled proof system, in particular also those with a rewinding extractor.

Novel isogeny proofs of knowledge. We design SPRINT (Succinct Proof for Radical Isogenies, Nimbly Transformed), a novel proof of isogeny knowledge based on the radical isogeny formulas due to Castryck, Decru and Vercauteren [CDV20]. These formulas have recently been used by Levin and Pedersen [LP25] to arithmetize Charles–Goren–Lauter (CGL) [CLG09] walks to R1CS for generic proof systems. We observe that, for proofs of knowledge of isogenies, we can embed the radical isogeny constraints directly into a polynomial IOP (PIOP) by exploiting their repetitive structure.

More precisely, we directly model the resulting equations into a multivariate polynomial and use a sumcheck-based protocol to create an efficient and perfectly zero-knowledge proof system. Unlike previous works, which consider only (knowledge) soundness, we prove round-by-round knowledge soundness, preventing the knowledge soundness losses that may otherwise occur when using the Fiat–Shamir transform on multi-round protocols. Additionally, we achieve logarithmic verification and zero-knowledge through some additional effort.

[3] For example, the Halo family [BGH19, ha21], which so far has not been proven (w/s)SE.

We provide an implementation for our proof system using the DeepFold PCS [GLH+25] and achieve substantial improvements across all metrics over the state of the art. Explicitly, with respect to [dHMSS26], we achieve a 1.1-8.1× faster prover speed than their Ligero implementation, previously achieving the fastest proofs in the literature, and a 1.4-24.0× faster verification speed than their Aurora implementation, previously achieving the fastest verification, across different parameter sets. Furthermore, we outperform the currently smallest proof sizes, achieved in SECUER [BCC+23], by a factor of 1.2-1.5×. Notably, we achieve these improvements *simultaneously*, whereas previous approaches optimizing one measure suffered from significant drawbacks in the others. Ours is also the first proof system to have a full implementation since the SECUER proof of knowledge [BCC+23], over which we achieve far greater improvements. The use of our proof of knowledge directly improves the performance of multiparty ceremonies for the generation of SECUERs. A fundamental feature in this context is that the prover does not need to know the endomorphism ring of the domain or the codomain curve.

A new family of isogeny-based signature schemes. We propose a new family of highly flexible isogeny-based signature schemes. For this we use a compiler to obtain EUF-CMA secure signatures from any wSE NARG system.[4] Here, signatures are proofs for a hard relation, defining the verification and signing keys. The hash of the message to be signed is included into the relation.

Our approach yields a very versatile framework, since the polynomial commitment scheme (PCS) used with our PIOP to obtain our NARG can easily be exchanged for other PCSs. A further source of flexibility is the option to use (exclusively) curves with unknown endomorphism ring among the protocol parameters. This eliminates the traditional dependence on the curve of j-invariant 1728 (or, more generally, on a curve with known endomorphism ring), on which schemes like SQIsign and PRISM crucially rely. In terms of security, the unforgeability of our signature relies on a well-established isogeny-based assumption, the ℓ^e-ISOGENYPATH problem, on the weak simulation extractability of the underlying zk-NARK, and on the Fiat–Shamir heuristic. We also point out that, unlike SQIsign and PRISM, our signature scheme can straightforwardly be made constant time.

We showcase one member of our family of signature schemes by using the recent DeepFold PCS [GLH+25]. In terms of computational efficiency, our scheme improves key generation and signing times by 2-3× compared to the NIST specification of SQIsign [AAA+25], while achieving timings almost on par with the highly optimized salted version of PRISM [BBC+26] and the Qlapoti version of SQIsign [BCE+25]. DeepFold naturally has a pre-computation phase, independent of the statement being proven. When this phase is excluded from the timings, our verifier is $3.75 - 6.47\times$ faster than the optimized Qlapoti SQIsign implementation. With the use of compression, key sizes match existing isogeny-based schemes, while for signature sizes our approach is less favorable, resulting in signatures of around 80kB. We note that this is not a lower bound as we still see much potential for optimization. Tweaking the parameters for the PCS also provides interesting design flexibility, e.g., changing the rate of code-based commitment schemes gives a natural trade-off between proving time and signature size. Switching to an alternative PCS is likely to give additional improvements. As an example, WHIR [ACFY25] yields proof sizes as small as 36kB for the NIST-I parameters, while retaining comparable timings (ignoring its current lack of zero-knowledge[5]).

Given its large signatures, we view the current design as a promising proof of concept; in contrast, our proof of knowledge already improves upon the state of the art across all metrics. However, we are optimistic that the performance of our signature scheme has room for considerable improvements, some of which are exemplified in Section 6.3. For comparison, consider the evolution of MPC/VOLEitH signatures over the last few years, i.e., coming down from signature sizes of around 40kB for the initial version of Picnic [CDG+17] when relying on a non-standard assumption (the LowMC blockcipher), to around 4.5kB for FAEST [BBB+25], now even relying on a standard assumption (the security of AES). Similarly, signing times for SQIsign improved by 7-9× from the previous implementation of SQIsign2D-West [BDD+24], which already drastically improved upon the previous SQIsign variants [DKL+20, DLRW24], to the specification for the SQIsign NIST round 2 submission [AAA+25] and the recent Qlapoti optimization [BCE+25].

[4] This result has long been folklore and was to the best of our knowledge first formalized in [DHLW10].

[5] A subsequent paper describes the changes needed for zero-knowledge [CFW26] but is not yet fully implemented at the moment of writing.

Note on Wesolowski's attack. Very recently, Wesolowski [Wes26] presented, under a plausible smoothness heuristic, an algorithm solving the supersingular isogeny problem in time and memory $p^{1/3+o(1)}$, improving upon the previous best time $p^{1/2} \cdot (\log p)^{O(1)}$. Combined with the fixed-degree-isogeny algorithm of [AAF+26, §5.2], it yields an attack on the ℓ-isogeny path problem underlying our scheme. This does not affect knowledge soundness or zero knowledge of our proof system, but lowers the estimated hardness of the underlying relation and thus the estimated security of the resulting signature scheme. A conservative, albeit costly, countermeasure is to increase the bit length of p from $\approx 2\lambda$ to $\approx 3\lambda$. Our current understanding suggests that signature path lengths may remain approximately 2λ. This does not apply to applications such as SECUER generation, whose output must be close to uniform and therefore requires[6] walks of length $\log(p) + 2\lambda + O(\log \lambda)$ [BCC+23, Thm. 11 and Lem. 14]. The concrete impact on our parameter sets remains unclear owing to the attack's recency, the factors hidden by the $o(1)$ term, and its high memory cost. We therefore retain the 'historical' primes for benchmarking, also preserving direct comparison with the current SQIsign and PRISM parameter sets.

2 Preliminaries

We first introduce some notation. We write $[n] = \{1, \ldots, n\}$. Given an index i, we will write $\vec{i}$ for its binary expansion in $\{0,1\}^n \subset \mathbb{F}^n$. We further define $\tilde{\mathrm{eq}}(\vec{x}, \vec{y})$ for $\vec{x}, \vec{y} \in \mathbb{F}^n$ as the multi-linear equality function

$$\tilde{\mathrm{eq}}(\vec{x}, \vec{y}) = \prod_{i \in [n]} (x_i y_i + (1 - x_i)(1 - y_i)) \ ,$$

i.e. the unique polynomial linear in $x_1, \ldots, x_n, y_1, \ldots, y_n$ such that in the domain $\{0,1\}^n \times \{0,1\}^n$, we have $\tilde{\mathrm{eq}}(\vec{x}, \vec{y}) = 1$ if $\vec{x} = \vec{y}$ and 0 otherwise. Given a function on the boolean hypercube, we may on occasion collect inputs to the function as a vector, i.e. for $f : \{0,1\}^n \to \mathbb{F}$ and $x \in \{0,1\}^n$, $f(x_1, \ldots x_n) = f(x)$.

2.1 Non-interactive Arguments

A *non-interactive argument* (NARG) in the ROM is a tuple $\mathtt{NARG} = (\mathcal{P}, \mathcal{V})$ where $\mathcal{P}$ is an oracle algorithm known as the *argument prover* and $\mathcal{V}$ is an oracle algorithm known as the *argument verifier*. We refer to [CY24] for definitions of completeness, zero-knowledge and succinctness. Here, we provide formal definitions for knowledge soundness and also adapt these to weak simulation extractability. First, we define the *oracle configuration* [CY24, Definition 7.1.1] $\mathsf{cnf}(\lambda, n)$, which outputs a list of output sizes $(\ell_i)_{i \in [k]}$. For a Fiat–Shamir transformed protocol, in round $i \in [k]$, the oracle constructed from cnf will output an ℓ_i bit challenge.

We define knowledge soundness (KS), the requirement that a successful prover must 'know' the witness, as well as weak simulation extractability (wSE), i.e. that this must hold even if an adversary can obtain simulated proofs. These are presented jointly, where the gray background color indicates the changes/additions needed to strengthen KS into wSE.

Definition 1 (Straight-line KS [CY24, Definition 7.1.5] / Straight-line wSE)**.** A non-interactive argument $\mathsf{NARG} = (\mathcal{P}, \mathcal{V})$ for a relation $\mathcal{R}$ is *straight-line knowledge sound* / *straight-line weak simulation-extractable with error* κ if there exists a polynomial-time deterministic algorithm $\mathcal{E}$ (the extractor) such that for every security parameter $\lambda \in \mathbb{N}$, statement size bound $n \in \mathbb{N}$, query bound $t \in \mathbb{N}$, and t-query deterministic argument prover $\tilde{\mathcal{P}}$,

$$\Pr\left[\begin{array}{lll} & |\mathbf{x}| \le n & \\ \wedge & (\mathbf{x}, \mathbf{w}) \notin \mathcal{R} & \\ \wedge & \mathcal{V}^f(\mathbf{x}, \pi) = 1 & \\ \wedge & \mathbf{x} \notin Q_{\mathcal{S}} & \end{array} \middle| \begin{array}{l} f \leftarrow \mathcal{U}(\mathsf{cnf}(\lambda, n)) \\ (\mathbf{x}, \pi) \xleftarrow{\mathrm{tr}} \tilde{\mathcal{P}}^{f, \mathcal{S}} \\ \mathbf{w} \leftarrow \mathcal{E}(\mathbf{x}, \pi) \end{array}\right] \le \kappa(\lambda, t, n)$$

Here, $Q_{\mathcal{S}}$ stores all statements for which $\mathcal{S}$ was asked for a simulated proof and f is a programmable RO.

[6] Per party, to ensure that having at least one honest party guarantees a secure curve.

If an argument system permits a (black-box) *straight-line* extractor, it must extract from a single execution of $\tilde{\mathcal{P}}$, seeing only its output $(\mathbf{x}, \pi)$ and query trace. Other argument systems are only known to be knowledge sound with respect to a *rewinding* extractor, which is given (black-box) access to $\tilde{\mathcal{P}}$ and re-runs it with fresh answers to some of its oracle queries in order to obtain further accepting proofs. Whether such re-runs succeed depends on how often $\tilde{\mathcal{P}}$ outputs an accepting proof at all, so the error of a rewinding extractor is allowed to depend on the *failure probability* of $\tilde{\mathcal{P}}$, which we define first. For wSE, outputting a statement in $Q_{\mathcal{S}}$ also counts as a failure, since a proof for such a statement is not a forgery.

Definition 2 (Failure probability [CY24, Definition 7.1.6])**.** Let $\mathsf{NARG} = (\mathcal{P}, \mathcal{V})$ be a non-interactive argument. A deterministic argument prover $\tilde{\mathcal{P}}$ has *failure probability* $\delta_{\tilde{\mathcal{P}},\mathcal{S}}$ *in the presence of* $\mathcal{S}$ if for every security parameter $\lambda \in \mathbb{N}$ and statement size bound $n \in \mathbb{N}$,

$$\Pr\left[\begin{array}{ll} & |\mathbf{x}| > n \\ \vee & \mathcal{V}^f(\mathbf{x}, \pi) = 0 \\ \vee & \mathbf{x} \in Q_{\mathcal{S}} \end{array} \middle| \begin{array}{l} f \leftarrow \mathcal{U}(\mathsf{cnf}(\lambda, n)) \\ (\mathbf{x}, \pi) \leftarrow \tilde{\mathcal{P}}^{f,\mathcal{S}} \end{array}\right] \leq \delta_{\tilde{\mathcal{P}},\mathcal{S}}(\lambda, n).$$

Definition 3 (Rewinding KS [CY24, Definition 7.1.7] / Rewinding wSE)**.** A non-interactive argument $\mathsf{NARG} = (\mathcal{P}, \mathcal{V})$ for a relation $\mathcal{R}$ is *rewinding knowledge sound* / *weak simulation-extractable with error* κ *and extraction time* $\mathsf{et}_{\mathsf{ARG}}$ if there exists a probabilistic algorithm $\mathcal{E}$ (the *extractor*) such that for every security parameter $\lambda \in \mathbb{N}$, statement size bound $n \in \mathbb{N}$, and t-query deterministic argument prover $\tilde{\mathcal{P}}$ with failure probability $\delta_{\tilde{\mathcal{P}}}$ and running time $\tau_{\tilde{\mathcal{P}}}$,

$$\Pr\left[\begin{array}{ll} & |\mathbf{x}| \leq n \\ \wedge & (\mathbf{x}, \mathbf{w}) \notin \mathcal{R} \\ \wedge & b = 1 \\ \wedge & \mathbf{x} \notin Q_{\mathcal{S}} \end{array} \middle| \begin{array}{l} f \leftarrow \mathcal{U}(\mathsf{cnf}(\lambda, n)) \\ (\mathbf{x}, \pi) \xleftarrow{\mathsf{tr}} \tilde{\mathcal{P}}^{f,\mathcal{S}} \\ b \xleftarrow{\mathsf{tr}_{\mathcal{V}}} \mathcal{V}^f(\mathbf{x}, \pi) \\ \mathbf{w} \leftarrow \mathcal{E}(\mathbf{x}, \pi, \mathsf{tr}, \mathsf{tr}_{\mathcal{V}}, \tilde{\mathcal{P}}) \end{array}\right] \leq \kappa(\lambda, t, n, \delta_{\tilde{\mathcal{P}},\mathcal{S}}(\lambda, n)).$$

2.2 Interactive Proofs

We give preliminaries on interactive proofs from [CY24, Chapter 13]. An interactive proof (IP) is a tuple of algorithms $\mathtt{IP} = (\mathcal{P}, \mathcal{V})$ that works as follows. The IP prover $\mathcal{P}$ receives as input a statement $\mathbf{x}$ and a witness $\mathbf{w}$, and the IP verifier $\mathcal{V}$ receives as input the statement $\mathbf{x}$. They interact over some number k of rounds, where in each round $i \in [k]$ the IP prover $\mathcal{P}$ sends a message α_i and then the IP verifier $\mathcal{V}$ sends a message ρ_i. After the interaction, the IP verifier $\mathcal{V}$ outputs a bit denoting whether to accept or reject, computed based on the statement $\mathbf{x}$, the received prover messages $(\alpha_i)_{i\in[k]}$, and the IP verifier's own randomness (these form the entire view of the IP verifier). Both algorithms may be probabilistic.

We omit definitions for completeness and (knowledge) soundness, only providing a formal definition for zero-knowledge, which is roughly the requirement that a verifier learns nothing about $\mathbf{w}$ except for its existence.

Definition 4 (Verifier's View [CY24, Definition 13.1.6])**.** The IP verifier's $\mathtt{View}$ in $\mathtt{IP} = (\mathcal{P}, \mathcal{V})$ on the statement-witness pair $(\mathbf{x}, \mathbf{w})$, denoted $\mathtt{View}(\mathcal{P}, \mathcal{V}, \mathbf{x}, \mathbf{w})$, is the random variable $(\mathbf{x}, \rho, (\alpha_i)_{i\in[k]})$ where:

- ρ is a random choice of randomness for the IP verifier $\mathcal{V}$; and
- $(\alpha_i)_{i\in[k]}$ are the prover messages received in an interaction between $\mathcal{P}(\mathbf{x}, \mathbf{w})$ and $\mathcal{V}(\mathbf{x}, \rho)$.

Note that the honest IP prover $\mathcal{P}(\mathbf{x}, \mathbf{w})$ may use its own private randomness (and is not part of the IP verifier's view). If the IP is public coin then the view shows each round's randomness: $(\mathbf{x}, (\rho_i)_{i\in[k]}, (\alpha_i)_{i\in[k]})$.

Definition 5 (Zero Knowledge [CY24, Definition 13.1.7])**.** $\mathtt{IP} = (\mathcal{P}, \mathcal{V})$ for a relation $\mathcal{R}$ has *honest-verifier zero-knowledge error* z_{IP} if there exists a polynomial-time probabilistic algorithm $\mathcal{S}$ such that for every statement-witness pair $(\mathbf{x}, \mathbf{w}) \in \mathcal{R}$ the following random variables are $z_{\mathtt{IP}}(\mathbf{x})$-close in statistical distance:

$$\mathtt{View}(\mathcal{P}, \mathcal{V}, \mathbf{x}, \mathbf{w}) \text{ and } \mathcal{S}(\mathbf{x}).$$

We additionally define

$$z_{\mathtt{IP}}(n) = \max\{z_{\mathtt{IP}}(\mathbf{x}) \mid \mathbf{x}, \mathbf{w} \in \{0,1\}^* \text{ with } |\mathbf{x}| \leq n \text{ and } (\mathbf{x}, \mathbf{w}) \in \mathcal{R}\}.$$

2.3 Polynomial IOP

A *polynomial interactive oracle proof* (PIOP) [KPT23, Definition 2.1] is a type of interactive protocol. In the *Interaction Phase*, $\mathcal{P}$ sends polynomial oracles, i.e. in round $i \in [k]$ it sends $(p_{i,1}, \ldots, p_{i,s(i)})$, and gets challenges ρ_i back. Here $s(i)$ is a function that specifies the number of polynomials sent in round i. We write $\vec{p}_i = (p_{i,j})_{j \in [s(i)]}$. In the *Query Phase*, $\mathcal{V}$ queries these polynomials at a number of points. Finally, $\mathcal{V}$ accepts or rejects in the *Decision Phase*. Let $\langle \mathcal{P}(\mathbf{x}, \mathbf{w}), \mathcal{V}(\mathbf{x}) \rangle$ denote $\mathcal{V}$'s output after the interaction with $\mathcal{P}$. We define $\texttt{View}$ as in Definition 5, except that $(\alpha_i)_i$ consists of the evaluations of the polynomial oracles $\vec{p}_i$.

Our PIOP will satisfy completeness, zero-knowledge and round-by-round knowledge soundness. Completeness and zero-knowledge are as for IPs (cf. Definition 5): the honest prover always convinces the verifier of a true statement, and a simulator given only $\mathbf{x}$ reproduces the verifier's view, here consisting of the oracle evaluations. We require both to hold perfectly.

Round-by-round knowledge soundness [CY24, Definitions 31.1.1 & 31.1.4] is phrased via a *state function*, which marks each partial transcript as *doomed* ($\texttt{RBRState} = 0$) or not. The empty transcript is doomed, prover messages cannot leave a doomed state, and a doomed full transcript is rejected, so a cheating prover can only escape through a lucky verifier challenge. Round-by-round knowledge soundness bounds the probability of such an escape in each round unless a witness can be extracted from the prover's messages. Unlike plain knowledge soundness, this is well-behaved under the Fiat–Shamir transformation (see Section 5).

Definition 6 (Completeness)**.** $\texttt{PIOP} = (\mathcal{P}, \mathcal{V})$ for a relation $\mathcal{R}$ has *perfect completeness* if for every $(\mathbf{x}, \mathbf{w}) \in \mathcal{R}$,

$$\Pr[\langle \mathcal{P}(\mathbf{x}, \mathbf{w}), \mathcal{V}(\mathbf{x}) \rangle = 1] = 1.$$

Definition 7 (Perfect Zero Knowledge)**.** $\texttt{PIOP} = (\mathcal{P}, \mathcal{V})$ for relation $\mathcal{R}$, is *(perfect) honest-verifier zero-knowledge* if there exists a PPT simulator $\mathcal{S}$ such that for any tuple $(\mathbf{x}, \mathbf{w}) \in \mathcal{R}$, the following random variables are equally distributed:

$$\texttt{View}(\mathcal{P}, \mathcal{V}, \mathbf{x}, \mathbf{w}) \text{ and } \mathcal{S}(\mathbf{x}).$$

Definition 8 (State Function)**.** A *state function* for $\texttt{PIOP} = (\mathcal{P}, \mathcal{V})$ is a deterministic (possibly inefficient) function $\texttt{RBRState}$ that receives as input a statement $\mathbf{x}$ and an interaction transcript $\texttt{trc}$ and outputs a bit for which the following holds.

- Empty transcript: if $\texttt{trc} = \emptyset$, then $\texttt{RBRState}(\mathbf{x}, \texttt{trc}) = 0$.
- Prover moves: if $\texttt{trc} = (\vec{p}_1, \rho_1, \ldots \vec{p}_i, \rho_i)$ is a transcript for i rounds (for $i < k$) with $\texttt{RBRState}(\mathbf{x}, \texttt{trc}) = 0$ then, for every $p_{\vec{i+1}}$, it holds that $\texttt{RBRState}(\mathbf{x}, (\texttt{trc}, p_{\vec{i+1}})) = 0$.
- Full transcript: if $\texttt{trc} = (\vec{p}_1, \rho_1, \ldots \vec{p}_k, \rho_k)$ is a full transcript and $\texttt{RBRState}(\mathbf{x}, \texttt{trc}) = 0$, then $\mathcal{V}$ outputs decision bit $b = 0$.

Definition 9 (Round by Round Knowledge Soundness)**.** A $\texttt{PIOP} = (\mathcal{P}, \mathcal{V})$ for a relation $\mathcal{R}$ has *round-by-round knowledge errors* $(\epsilon_i)_{i \in [k]}$ if there exists an extractor $\mathcal{E}$ and state function $\texttt{RBRState}$ such that for every statement $\mathbf{x}$, round index $i \in [k]$, and malicious prover $\tilde{\mathcal{P}}$ the following holds:

$$\Pr\left[\begin{array}{ll|l} & \texttt{RBRState}(\mathbf{x}, (\vec{p}_1, \rho_1, \ldots \vec{p}_i)) = 0 & ((\vec{p}_i)_{i \in [k]}, (\rho_j)_{j \in [i-1]}) \leftarrow \tilde{\mathcal{P}} \\ \wedge & \texttt{RBRState}(\mathbf{x}, (\vec{p}_1, \rho_1, \ldots \vec{p}_i, \rho_i)) = 1 & \rho_i \in \texttt{Ch} \\ \wedge & (\mathbf{x}, \mathbf{w}) \notin \mathcal{R} & \mathbf{w} \leftarrow \mathcal{E}(\mathbf{x}, (\vec{p}_i)_{i \in [k]}) \end{array}\right] \leq \epsilon_i(\mathbf{x}).$$

The PIOP has *round-by-round knowledge error* ϵ if for every $i \in [k]$, $\epsilon_i \leq \epsilon$.

2.4 Sumcheck Protocol

The *sumcheck protocol* [LFKN92, Set20] is an interactive protocol to reduce a relation over a finite field $\mathbb{F}$ of the form $\sum_{\vec{z} \in \{0,1\}^n} p(\vec{z}) = 0$ to the relation $p(\vec{r}_z) = H$, where $\vec{r}_z \in \mathbb{F}^n$ is a uniformly random assignment for $\vec{z}$. The protocol consists of n rounds of interaction, where in round i the prover $\mathcal{P}$ sends a univariate polynomial $\hat{p}_i$ such that

$$\hat{p}_i(X) = \sum_{\vec{z} \in \{0,1\}^{n-i}} p(r_{z1}, \ldots, r_{zi-1}, X, \vec{z}).$$

$\mathcal{V}$ checks consistency with the previous round by checking that $\hat{p}_i(0) + \hat{p}_i(1) = \hat{p}_{i-1}(r_{zi-1})$, sampling a random assignment $r_{zi} \in \mathbb{F}$ and sending it to $\mathcal{P}$. After n rounds of interaction, $\mathcal{V}$ ends up with a final claim, which can be checked by a single evaluation of p.

2.5 Isogenies and Their Computation

We assume some familiarity with isogenies of elliptic curves and revisit the main definitions and arithmetic aspects of isogeny computations in Appendix A. Note that we only consider supersingular elliptic curves (with representation over $\mathbb{F}_{p^2}$) and separable isogenies throughout this work. We denote the ℓ-isogeny graph as $\mathcal{G}_\ell(p)$ and call chains of ℓ-isogenies in this graph isogeny paths.

A foundational assumption in isogeny-based cryptography is that, for long enough paths (that is, for large enough e), the following problem is hard [CLG09].

Problem 1 (ℓ^e-IsogenyPath)**.** Given distinct primes ℓ and p, a positive integer e and a pair $E, E' \in \mathcal{G}_\ell(p)$, find an ℓ-isogeny path $\phi\colon E \to E'$ of length e.

Our security relies on this problem, where the path length e and first curve E is fixed. Such a problem arises from the CGL hash function and its variants; hardness of GPS [GPS17] signature key-recovery, the first in the class of isogeny-based signatures; and is strictly harder than the (now broken) computational assumption underlying the SIDH family of constructions. We emphasise that the attacks on SIDH did not arise from breaking this variant of the problem, but rather from a related problem with additional structure admitted.

For most cryptographic applications, we are only interested in paths that are not *backtracking*, i.e., paths for which no step is the dual of the previous, as any path can be turned into a (possibly shorter) path without backtracking. Importantly, isogenies corresponding to paths without backtracking are *cyclic* [CLG09, Prop. 1]. The only isogenies that will be relevant in this work are isogenies of degree 2^e for some positive power e. The computational cost of an isogeny depends on its degree, so computing a chain of e consecutive 2-isogenies instead of a single isogeny of degree 2^e will generally give an exponential speedup in the computational cost. Classical approaches to compute such non-backtracking paths are revisited in Appendix A. Throughout this work, we will rely on the more recently introduced *radical isogenies* computational approach.

Radical 2-isogenies. In [CD20], Castryck and Decru propose an alternative way of computing chains of 2-isogenies $E_0 \to E_1 \to \cdots \to E_e$, a result that was later generalized in [CDV20]. The idea is to work over elliptic curves that have $(0,0)$ ready in the 2-torsion and then remove the need to sample points by post-composing the isogeny computation with an isomorphism that makes $(0,0) \in E_{i+1}[2]$ a non-backtracking point again. For isogenies of degree 2, we can choose variants of the Montgomery form, i.e.

$$E : y^2 = x^3 + Ax^2 + Cx\,.$$

Clearly $(0,0)$ is well-defined over this curve. By applying the 2-isogeny formulas from e.g. [Vél71], we find that $E' = E/\langle (0,0) \rangle : y^2 = x^3 + A'x^2 + C'x$ has coefficients

$$A' = -2A, \quad \text{and} \quad C' = A^2 - 4C\,.$$

On E', $(0,0)$ is now the backtracking point, and the other 2-torsion points are given by $(A + 2\sqrt{C}, 0)$ and are distinguished by the choice of the root. By translating either of these points to $(0,0)$, we end up with the curve $E'' : y^2 = x^3 + A''x^2 + C''x$, where

$$A'' = 6\sqrt{C} + A, \quad \text{and} \quad C'' = 4A\sqrt{C} + 8C\,. \tag{1}$$

From this new curve, we can continue the walk in the isogeny graph by again quotienting out $(0,0)$. Depending on the choice of the root of C, this defines either of the two outgoing vertices in the graph.

Proof of knowledge. More recently, [LP25] have constructed a proof of knowledge for a chain of 2-isogenies, by arithmetizing the constraints from Equation (1) into a Rank-1 Constraint System (R1CS). Here, the coefficients A and C for curves along the walk are indexed by $i \in \{0, \ldots, e\}$, with

(A_0, C_0) representing the starting curve and (A_e, C_e) the end curve. The resulting constraint system can then be proven generically by any R1CS-compatible proof system. On top of these constraints, they also present a proof of the correct evaluation of a CGL hash function [CLG09], as well as a verifiable random function.

2.6 Signature Schemes

Definition 10 (Signature Scheme)**.** A *signature scheme* is a tuple of algorithms $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ which are defined as follows:

- $\mathsf{Gen}(\mathsf{par})$: The randomized *key generation* algorithm Gen takes as input public parameters par (or just the a security parameter 1^λ in unary) and outputs a secret/public key pair $(\mathsf{sk}, \mathsf{pk})$.
- $\mathsf{Sign}(\mathsf{sk}, m)$: The (possibly) randomized *signing* algorithm Sign takes as input a secret key sk and a message m and outputs a signature σ.
- $\mathsf{Verify}(m, \sigma, \mathsf{pk})$: The deterministic *verification* algorithm Verify takes as input a public key pk, a signature σ, and a message m. It outputs either 1 (accept) or 0 (reject).

Definition 11 (Correctness)**.** A signature scheme Σ is correct if for every security parameter λ, message m such that $\mathsf{pp} \leftarrow \mathsf{PPGen}(1^\lambda)$ and $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(\mathsf{pp})$, we have:

$$\Pr\left[\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m) : \mathsf{Verify}(m, \sigma, \mathsf{pk}) = 1\right] = 1.$$

Definition 12 (EUF-CMA)**.** A signature scheme Σ is *existentially unforgeable under adaptively chosen-message attacks*, if for all PPT adversaries $\mathcal{A}$ with access to a signing oracle Sign, the following probability is negligible,

$$\Pr\left[\begin{array}{l} \mathsf{pp} \leftarrow \mathsf{PPGen}(1^\lambda);\ (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(\mathsf{pp}); (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}, \cdot)}(\mathsf{pk}) : \\ \forall m \in Q : m^* \neq m \wedge \mathsf{Verify}(m^*, \sigma^*, \mathsf{pk}) = 1 \end{array}\right]$$

where Q is the set of queries that $\mathcal{A}$ has issued to the signing oracle.

For unforgeability, one can also consider the strong variant denoted sEUF-CMA security, in which the oracle keeps track of message-signature pairs and the adversary wins if $(m^*, \sigma^*) \notin Q$.

3 Canonical Simulation Implies Weak-SE

Before we construct our proof of knowledge, we recall our focus on *weak simulation extractability* (wSE), where an adversary only succeeds if they forge a proof π for a statement $\mathbf{x}$ for which they did not query the simulator. Since actual and simulated proofs are indistinguishable, this implies that proofs for one statement cannot be used to forge a proof for another statement. So far there are no general results for establishing wSE for transparent multi-round protocols. While strong SE is a powerful notion, proving it is often complex with existing frameworks imposing additional constraints.[7] Meanwhile, for many applications such as signature schemes, wSE is sufficient. We show that weak simulation extractability is achieved automatically for a wide class of proof systems, namely those with a *canonical simulator*. This is a simulator that first creates a simulated transcript and then constructs a query-answer list such that the transcript becomes a valid Fiat–Shamir proof under programmable random oracles $f = (f_i)_{i \in [k]}$.

Definition 13 (Canonical simulator, [CY24, Construction 16.2.2])**.** The *canonical simulator* is an algorithm $\mathcal{S}(\mathbf{x})$ that works as follows. Below we denote by $\mathcal{S}_{\mathtt{IP}}$ the honest-verifier zero-knowledge simulator of the IP.

1. Sample a simulated view of the IP verifier: $(\mathbf{x}, (\rho_i)_{i \in [k]}, (\alpha_i)_{i \in [k]}) \leftarrow \mathcal{S}_{\mathtt{IP}}(\mathbf{x})$.
2. For $i = 1 \ldots k$:
   (a) Sample a random salt $\tau_i \in \{0,1\}^s$.

[7] The unique response property in particular may not be satisfied by protocols that randomly reveal entries in vector commitments.

(b) Set $x_i = \begin{cases} (\mathbf{x}, \alpha_1, \tau_1) & \text{if } i = 1, \\ (\rho_{i-1}, \alpha_i, \tau_i) & \text{if } i > 1. \end{cases}$

(c) Set the query-answer list $\mu_i := \{(x_i, \rho_i)\}$, to be used to program oracle f_i.

3. Set $\mu := (\mu_i)_{i \in [k]}$.

4. Output (π, μ).

Note that $\mathcal{S}$ does not yet program f. We write $\mathcal{S}'$ for a *programming* canonical simulator, which first runs the canonical simulator $\mathcal{S}$ to obtain (π, μ), programs f_i using μ_i and finally outputs π. In other words, it is the canonical simulator that not only generates the query-answer list, but also programs the RO.

Theorem 1. *Let $\mathsf{NARG} = (\mathcal{P}, \mathcal{V})$ be a Fiat–Shamir Transformed IP for relation $\mathcal{R}$, i.e. $\mathsf{NARG} := \boldsymbol{FS}[IP, \lambda, s]$ with security parameter λ and privacy parameter s, and let $\mathcal{S}'$ be a programming canonical simulator. Let $t \in \mathbb{N}$ be a query bound for f, $n \in \mathbb{N}$ be a statement size bound, and $q_{\mathcal{S}'} \in \mathbb{N}$ be a query bound for the simulator. Let $\Delta = t q_{\mathcal{S}'} \sum_{i \in [k-1]} 2^{-\ell_i}$.*

- *If NARG is straight-line knowledge sound with knowledge soundness error $\kappa_{KS}(\lambda, t, n)$, then it is also straight-line weak simulation-extractable with knowledge soundness error*

$$\kappa_{wSE}(\lambda, t, n) = \kappa_{KS}(\lambda, t, n) + \Delta.$$

- *If NARG has rewinding knowledge soundness error $\kappa_{KS}(\lambda, t, n, \delta_{\mathcal{P}}(\lambda, n))$, then it is also rewinding weak simulation-extractable with knowledge soundness error*

$$\kappa_{wSE}(\lambda, t, n, \delta_{\mathcal{P},\mathcal{S}'}(\lambda, n)) = \kappa_{KS}(\lambda, t, n, \delta_{\mathcal{P},\mathcal{S}'}(\lambda, n) + \Delta) + \Delta.$$

We recall that ℓ_i denotes the challenge size for the RO of round $i \in [k]$ as given in Section 2.1.

Proof. Consider the rewinding case. We prove the result by contradiction: assume that for every extractor $\mathcal{E}$, there exists a weak-SE prover $\tilde{\mathcal{P}}_{\mathsf{wSE}}$ such that

$$\Pr\left[\begin{array}{lll} & |\mathbf{x}| \le n & f \leftarrow \mathcal{U}(\mathsf{cnf}(\lambda, n)) \\ \wedge & (\mathbf{x}, \mathbf{w}) \notin \mathcal{R} & (\mathbf{x}, \pi) \xleftarrow{\mathsf{tr}} \tilde{\mathcal{P}}_{\mathsf{wSE}}^{f, \mathcal{S}'} \\ \wedge & b = 1 & b \xleftarrow{\mathsf{tr}_{\mathcal{V}}} \mathcal{V}^f(\mathbf{x}, \pi) \\ \wedge & \mathbf{x} \notin Q_{\mathcal{S}'} & \mathbf{w} \leftarrow \mathcal{E}(\mathbf{x}, \pi, \mathsf{tr}, \mathsf{tr}_{\mathcal{V}}, \tilde{\mathcal{P}}_{\mathsf{wSE}}) \end{array}\right] > \kappa_{\mathsf{wSE}}(\lambda, t, n, \delta_{\tilde{\mathcal{P}}_{\mathsf{wSE}},\mathcal{S}'}(\lambda, n)). \tag{2}$$

We build from $\tilde{\mathcal{P}}_{\mathsf{wSE}}$ a knowledge soundness prover $\tilde{\mathcal{P}}_{\mathsf{KS}}$ for which

$$\Pr\left[\begin{array}{lll} & |\mathbf{x}| \le n & f \leftarrow \mathcal{U}(\mathsf{cnf}(\lambda, n)) \\ \wedge & (\mathbf{x}, \mathbf{w}) \notin \mathcal{R} & (\mathbf{x}, \pi) \xleftarrow{\mathsf{tr}} \tilde{\mathcal{P}}_{\mathsf{KS}}^{f} \\ \wedge & b = 1 & b \xleftarrow{\mathsf{tr}_{\mathcal{V}}} \mathcal{V}^f(\mathbf{x}, \pi) \\ & & \mathbf{w} \leftarrow \mathcal{E}(\mathbf{x}, \pi, \mathsf{tr}, \mathsf{tr}_{\mathcal{V}}, \tilde{\mathcal{P}}_{\mathsf{KS}}) \end{array}\right] > \kappa_{\mathsf{KS}}(\lambda, t, n, \delta_{\tilde{\mathcal{P}}_{\mathsf{KS}}}(\lambda, n)). \tag{3}$$

Since this is a contradiction, it follows that there exists at least one extractor $\mathcal{E}$ for which there are no provers $\tilde{\mathcal{P}}_{\mathsf{wSE}}$ that succeed at cheating in the weak-SE case with sufficient probability.

To construct $\tilde{\mathcal{P}}_{\mathsf{KS}}$ and show that Equation (2) implies Equation (3), we emulate a programmable random oracle. Let $\mathcal{F} = (\mathcal{F}_i)_{i \in [k]}$ be a list of functionalities which work as follows: whenever f_i would be programmed, $\mathcal{F}_i$ adds the elements of μ_i to the initially empty set $\mu_{i,\mathcal{F}}$. Whenever $\mathcal{F}_i$ is queried with x_i, it outputs ρ_i if $(x_i, \rho_i) \in \mu_{i,\mathcal{F}}$, and otherwise queries f_i for x_i and outputs the answer.

We let $\tilde{\mathcal{P}}_{\mathsf{KS}}$ be identical to $\tilde{\mathcal{P}}_{\mathsf{wSE}}$, except for the following:

- $\mathcal{S}'$ is incorporated into $\tilde{\mathcal{P}}_{\mathsf{KS}}$, such that on a call to $\mathcal{S}'$, $\tilde{\mathcal{P}}_{\mathsf{KS}}$ performs any action that $\mathcal{S}'$ would have taken.
- For any such call, the prover keeps track of all queried statements in set $\tilde{Q}_{\mathcal{S}'}$ to mirror $Q_{\mathcal{S}'}$.
- Any interaction with f, including any calls that $\mathcal{S}'$ would have made, are replaced by interactions with $\mathcal{F}$.

Note that this transformation does not cause the prover to exceed the query bound t, since $\mathcal{S}'$ never queries f, nor does it change its running time.

Since $\tilde{\mathcal{P}}_{\mathsf{KS}}$ perfectly mimics $\mathcal{S}'$'s actions and $\mathcal{F}$'s behavior is indistinguishable from f's,

$$\Pr\left[\begin{array}{lll} & |\mathbf{x}| \leq n & \\ \wedge & (\mathbf{x},\mathbf{w}) \notin \mathcal{R} & \\ \wedge & b = 1 & \\ \wedge & \mathbf{x} \notin Q_{\mathcal{S}'} & \end{array} \middle| \begin{array}{l} f \leftarrow \mathcal{U}(\mathsf{cnf}(\lambda,n)) \\ (\mathbf{x},\pi) \xleftarrow{\mathsf{tr}} \tilde{\mathcal{P}}_{\mathsf{wSE}}^{f,\mathcal{S}'} \\ b \xleftarrow{\mathsf{tr}_{\mathcal{V}}} \mathcal{V}^{f}(\mathbf{x},\pi) \\ \mathbf{w} \leftarrow \mathcal{E}(\mathbf{x},\pi,\mathsf{tr},\mathsf{tr}_{\mathcal{V}},\tilde{\mathcal{P}}_{\mathsf{wSE}}) \end{array}\right]$$

$$= \Pr\left[\begin{array}{lll} & |\mathbf{x}| \leq n & \\ \wedge & (\mathbf{x},\mathbf{w}) \notin \mathcal{R} & \\ \wedge & b = 1 & \\ \wedge & \mathbf{x} \notin \tilde{Q}_{\mathcal{S}'} & \end{array} \middle| \begin{array}{l} f \leftarrow \mathcal{U}(\mathsf{cnf}(\lambda,n)) \\ (\mathbf{x},\pi) \xleftarrow{\mathsf{tr}} \tilde{\mathcal{P}}_{\mathsf{KS}}^{\mathcal{F}} \\ b \xleftarrow{\mathsf{tr}_{\mathcal{V}}} \mathcal{V}^{\mathcal{F}}(\mathbf{x},\pi) \\ \mathbf{w} \leftarrow \mathcal{E}(\mathbf{x},\pi,\mathsf{tr},\mathsf{tr}_{\mathcal{V}},\tilde{\mathcal{P}}_{\mathsf{KS}}) \end{array}\right] = (\star).$$

In other words, the cheating probability of cheating prover $\tilde{\mathcal{P}}_{\mathsf{KS}}$ against $\mathcal{V}^{\mathcal{F}}$ must be identical to that of $\tilde{\mathcal{P}}_{\mathsf{wSE}}$ against $\mathcal{V}^{f}$. We now show that

$$(\star) \leq \Pr\left[\begin{array}{lll} & |\mathbf{x}| \leq n & \\ \wedge & (\mathbf{x},\mathbf{w}) \notin \mathcal{R} & \\ \wedge & b = 1 & \end{array} \middle| \begin{array}{l} f \leftarrow \mathcal{U}(\mathsf{cnf}(\lambda,n)) \\ (\mathbf{x},\pi) \xleftarrow{\mathsf{tr}} \tilde{\mathcal{P}}_{\mathsf{KS}}^{\mathcal{F}} \\ b \xleftarrow{\mathsf{tr}_{\mathcal{V}}} \mathcal{V}^{f}(\mathbf{x},\pi) \\ \mathbf{w} \leftarrow \mathcal{E}(\mathbf{x},\pi,\mathsf{tr},\mathsf{tr}_{\mathcal{V}},\tilde{\mathcal{P}}_{\mathsf{KS}}) \end{array}\right] + tq_{\mathcal{S}'} \sum_{i\in[k-1]} 2^{-\ell_i}.$$

There are two differences between the probabilities on either side of the inequality. First, the condition $\mathbf{x} \notin \tilde{Q}_{\mathcal{S}'}$ is removed. Second, on the right hand side, $\mathcal{V}$ uses the real f, which $\tilde{\mathcal{P}}_{\mathsf{KS}}$ cannot program. Since the removal of the condition $\mathbf{x} \notin \tilde{Q}_{\mathcal{S}}$ can only increase the success probability, the only situations where the inequality becomes false are those where $\tilde{\mathcal{P}}$ uses a programmed query in their proof, which $\mathcal{V}$ then rejects because it was not programmed in f. By Definition 13, there are three cases that we need to consider.

- In the first round, $i = 1$. If x_1 is a programmed query, the prover does not succeed in either case. The prover on the left does not succeed since this means $\mathbf{x} \in Q_{\mathcal{S}'}$, while the prover on the right does not succeed since f was not actually programmed.
- In subsequent rounds, if $x_i = \hat{x}_i$ for some programmed query $\hat{x}_i$, then $\rho_{i-1} = \hat{\rho}_{i-1}$ where $\hat{\rho}_{i-1}$ is the previous round challenge of the simulated transcript. This could mean that $x_{i-1} = \hat{x}_{i-1}$, i.e. that the query for the previous round already matches that of a simulated transcript.
- Or, it could mean that the prover was lucky and even though x_{i-1} was not programmed, still $\rho_{i-1} = \hat{\rho}_{i-1}$ for some programmed response.

It follows by induction that the prover must have found a collision ρ_i for some $1 \leq i < k$. Since at most $q_{\mathcal{S}'}$ such queries have been programmed, this happens with probability across all rounds of at most $\Delta = tq_{\mathcal{S}'} \sum_{i\in[k-1]} 2^{-\ell_i}$. Note that the failure probability attains the same factor, i.e. $\delta_{\tilde{\mathcal{P}}_{\mathsf{KS}}}(\lambda,n) \leq \delta_{\tilde{\mathcal{P}}_{\mathsf{wSE}},\mathcal{S}'}(\lambda,n) + \Delta$. From this, we conclude that, if $\tilde{\mathcal{P}}_{\mathsf{wSE}}$ succeeds with probability more than $\kappa_{\mathsf{wSE}}(\lambda,t,n,\delta_{\tilde{\mathcal{P}}_{\mathsf{wSE}},\mathcal{S}'}(\lambda,n))$, then $\tilde{\mathcal{P}}_{\mathsf{KS}}$ succeeds with probability more than

$$\kappa_{\mathsf{KS}}(\lambda,t,n,\delta_{\tilde{\mathcal{P}}_{\mathsf{KS}}}(\lambda,n)) = \kappa_{\mathsf{wSE}}(\lambda,t,n,\delta_{\tilde{\mathcal{P}}_{\mathsf{KS}}}(\lambda,n) - \Delta) - \Delta.$$

It follows that if there exists an extractor $\mathcal{E}$ for which all provers $\tilde{\mathcal{P}}$ succeed with probability at most κ_{KS}, then there also exists an extractor for which all weak-SE provers $\tilde{\mathcal{P}}$ succeed with probability at most κ_{wSE}.

The statement for the straight-line extractor is a special case of the rewinding case. Instead of considering all extractors $\mathcal{E}(\mathbf{x},\pi,\mathtt{tr},\mathtt{tr}_{\mathcal{V}},\mathcal{P})$, we only consider extractors that do not access $\mathcal{P}$. The result follows. □

Note that the added success probability for the prover is an artifact of the optimized variant of Fiat–Shamir, where the challenge of the previous round is hashed instead of the full preceding transcript. In the latter case, $\mathbf{x}$ appears in every programmed query, and thus a prover never succeeds (i.e. $\Delta = 0$). Since this version is more straightforward and the optimized version appears in practice, we prove the statement for the optimized version.

4 Constructing a Novel Isogeny Proof of Knowledge

We now present our proof of knowledge of isogeny relations. Like [LP25], we use the radical 2-isogenies to construct a proof system for an isogeny walk of degree 2^e. However, instead of R1CS to be used in a generic proof system, we construct a tailored PIOP, which can then be compiled using any polynomial commitment scheme. Since the PIOP + PCS approach of constructing proof systems is very common, this still gives us the flexibility of the generic approach, while achieving major speedups and reductions in proof sizes.

Setting $\mathbb{F} = \mathbb{F}_{p^2}$, we start from the constraints in Equation (1), where we eliminate the factors $\sqrt{C}$ by squaring the first formula and substituting the first into the second. We obtain the following two constraints for $i \in \{0, \dots, e-1\}$,

$$-\frac{1}{12} A_i (A_{i+1} - A_i) = C_i - \frac{1}{8} C_{i+1}, \tag{4}$$

$$\frac{1}{36} (A_{i+1} - A_i)^2 = C_i, \tag{5}$$

from which we derive the following relation[8],

$$\mathcal{R} = \left\{ (\mathbf{x}, \mathbf{w}) : \begin{array}{l} \mathbf{x} = (A_0, C_0, A_e, C_e),\ \mathbf{w} = \Big((A_i)_{i \in [e-1]}, (C_i)_{i \in [e-1]} \Big) \\ \forall i \in \{0, \dots, e-1\} : \quad \begin{array}{ll} -\frac{1}{12} A_i (A_{i+1} - A_i) & = C_i - \frac{1}{8} C_{i+1} \\ \wedge\ \frac{1}{36} (A_{i+1} - A_i)^2 & = C_i \end{array} \end{array} \right\}. \tag{6}$$

In the remainder of this section, we construct a PIOP which can then be compiled to a proof of knowledge for relation $\mathcal{R}$. We use Section 4.1 to construct the polynomial that is at the basis of this relation. As we will see, it encodes both constraints at every step of the isogeny path, which reduces to a claim on a single evaluation point using the sumcheck protocol.

At each index i of the isogeny walk, we use coefficients at indices i and $i+1$. To ensure the polynomial that we will use in a sumcheck protocol has a constant degree in every variable, we introduce a new multilinear polynomial $\tilde{Q}$. We prove in Section 4.2 that $\tilde{Q}$ is efficiently evaluable, even outside of the boolean hypercube, which is necessary to obtain a succinct verifier.

In Section 4.3, we discuss how to adapt our PIOP to isogeny walks of lengths e that are not a power of two. Finally, Section 4.4 demonstrates how to make the resulting system zero-knowledge.

4.1 Polynomial Relation from Radical Isogenies

To encode Equations (4) and (5) into a single polynomial, we introduce the multilinear polynomial $\tilde{B}$, such that for all $i \in \{0, \dots, e\}$, $\tilde{B}(0, 0, \vec{i}) = A_i$ and $\tilde{B}(0, 1, \vec{i}) = C_i$. The remaining entries are selected uniformly at random (under the constraint that the mask M, defined in Section 4.4, sums to 0 over the boolean hypercube). We fix the linear polynomials a_{A_j}, a_{A_i} and a_{C_j} with

$$\begin{array}{lll} a_{A_j}(0) = 0, & a_{A_i}(0) = -\frac{1}{12}, & a_{C_j}(0) = -\frac{1}{8}, \\ a_{A_j}(1) = \frac{1}{36}, & a_{A_i}(1) = -\frac{1}{36}, & a_{C_j}(1) = 0, \end{array}$$

to encode the coefficients. Using these ingredients, we can define the polynomial

$$\begin{aligned} P'(\vec{h}, \vec{k}, \vec{i}, x, \vec{j}) = \tilde{\mathrm{eq}}(h_0, 1)\, \tilde{\mathrm{eq}}(x, 0) \tilde{Q}(\vec{i}, \vec{j}, \vec{k}) \\ \Big(\Big[a_{A_j}(h_1) \tilde{B}(x, 0, \vec{j}) + a_{A_i}(h_1) \tilde{B}(x, 0, \vec{i}) \Big] \Big[\tilde{B}(x, 0, \vec{j}) - \tilde{B}(x, 0, \vec{i}) \Big] \\ - \Big[\tilde{B}(x, 1, \vec{i}) + a_{C_j}(h_1) \tilde{B}(x, 1, \vec{j}) \Big] \Big) \end{aligned} \tag{7}$$

as well as the masked version

$$P(\vec{h}, \vec{k}, \vec{i}, x, \vec{j}) = P'(\vec{h}, \vec{k}, \vec{i}, x, \vec{j}) + M(\vec{i}, x, \vec{j}) \tag{8}$$

[8] Here, (A_i, C_i) correspond to curve E_i's coordinates in the ℓ^e-IsogenyPath Problem for $\ell = 2$. In Appendix C we discuss how we can use the statement $\mathbf{x} = (j(E_0), j(E_e))$.

where $x \in \mathbb{F}$, $\vec{h} \in \mathbb{F}^2$, $\vec{k}, \vec{i}, \vec{j} \in \mathbb{F}^n$. Here, h_0 ensures independence between the mask and the constraints, while h_1 selects which constraint is checked, and x is used to ensure only the evaluations $\tilde{B}(0, \dots)$ have contributions in the constraints, since the product is automatically 0 if $x = 1$. Notably, M is not multilinear, but instead has degree 3 in all of its variables. By construction, it evaluates to 0 when summed over the boolean hypercube. To ensure that our constraints run over consecutive curves of our path, we define $\tilde{Q}$ as follows:

$$\tilde{Q}(\vec{i}, \vec{j}, \vec{k}) = \sum_{l \in \{0,1\}^n \setminus \{1\dots1\}} \tilde{\mathrm{eq}}(\overrightarrow{(l+1)}, \vec{j})\, \tilde{\mathrm{eq}}(\vec{l}, \vec{i})\, \tilde{\mathrm{eq}}(\vec{l}, \vec{k}). \tag{9}$$

In words, $\tilde{Q}$ is the unique multilinear polynomial that returns 1 if and only if $\vec{i} = \vec{k}$ and $\vec{j} = (\vec{i+1})$ on the boolean hypercube. We define the sum

$$q(\vec{h}, \vec{k}) = \sum_{\vec{i} \in \{0,1\}^n} \sum_{x \in \{0,1\}} \sum_{\vec{j} \in \{0,1\}^n} P(\vec{h}, \vec{k}, \vec{i}, x, \vec{j}), \tag{10}$$

which allows us to present the following lemma:

Lemma 2. Relation $\mathcal{R}$ in Equation (6) is equivalent to the following relation:

$$\mathcal{R}' = \left\{ (\mathbf{x}, \mathbf{w}) : \begin{array}{ll} \mathbf{x} = (A_0, C_0, A_e, C_e),\ \mathbf{w} = \tilde{B}(\cdot) & \\ \forall i \in \{0, \dots, e\} : & \tilde{B}(0,0,\vec{i}) = A_i, \tilde{B}(0,1,\vec{i}) = C_i \\ \forall \vec{h} \in \{0,1\}^2, \vec{k} \in \{0,1\}^n : & q(\vec{h}, \vec{k}) = 0 \end{array} \right\}.$$

Proof. We have already seen how $\mathbf{w}$ for $\mathcal{R}$ is transformed to the polynomial $\tilde{B}$ and vice versa. It remains to be shown that $\mathcal{R}$ is satisfied if and only if $\mathcal{R}'$ is.

First, note that the factor $\tilde{\mathrm{eq}}(h_0, 1)$ ensures that

$$\sum_{\vec{i} \in \{0,1\}^n} \sum_{x \in \{0,1\}} \sum_{\vec{j} \in \{0,1\}^n} P'(\vec{h}, \vec{k}, \vec{i}, x, \vec{j}) = \sum_{\vec{i} \in \{0,1\}^n} \sum_{x \in \{0,1\}} \sum_{\vec{j} \in \{0,1\}^n} M(\vec{i}, x, \vec{j}) = 0,$$

where the latter equality holds by construction. Since $P'(\vec{h}, \vec{k}, \vec{i}, 1, \vec{j}) = 0$ due to the factor $\tilde{\mathrm{eq}}(x, 0)$, it suffices to focus on $P'(1, h_1, \vec{k}, \vec{i}, 0, \vec{j}) = 0$. Here, $\tilde{Q}$ gives only a non-zero contribution if $\vec{i} = \vec{k}$ and $\vec{j} = k \vec{+} 1$ and therefore

$$\begin{aligned} &\sum_{\vec{i} \in \{0,1\}^n} \sum_{x \in \{0,1\}} \sum_{\vec{j} \in \{0,1\}^n} P'(1, h_1, \vec{k}, \vec{i}, 0, \vec{j}) \\ &\quad = \Big(\Big[a_{A_j}(h_1) \tilde{B}(0,0,k \vec{+} 1) + a_{A_i}(h_1) \tilde{B}(0,0,\vec{k}) \Big] \Big[\tilde{B}(0,0,k \vec{+} 1) - \tilde{B}(0,0,\vec{k}) \Big] \\ &\qquad\qquad - \Big[\tilde{B}(0,1,\vec{k}) + a_{C_j}(h_1) \tilde{B}(0,1,k \vec{+} 1) \Big] \Big). \end{aligned}$$

We see that $h_1 \in \{0,1\}$ recovers exactly the constraints in $\mathcal{R}$. □

The above polynomial will be the basis of our PIOP. Instead of considering all assignments $(\vec{h}, \vec{k}) \in \{0,1\}^{2+n}$, the verifier will provide the challenge $(\vec{r_h}, \vec{r_k}) \in \mathbb{F}^{2+n}$. This is valid due to linearity in $\vec{h}$ and $\vec{k}$, and subsequently applying the Schwartz–Zippel lemma. Additional work is required to ensure the protocol is efficient, zero-knowledge and applicable to walks with length not equal to a power of two, which we cover in the remainder of this section.

Remark 1. One might be tempted to write $\vec{j}(\vec{i}) = (\vec{i+1})$ as a multilinear extension to eliminate the input variables $\vec{j}$ and $\vec{k}$. Unfortunately, any bit of $\vec{i}$ affects the value of $(\vec{i+1})$. For example, j_n flips from 0 to 1 if $i_{n-1} = \cdots = i_1 = 1$, i.e. based on the product $\prod_{v \in [n-1]} i_v$. While this is still multilinear, we then use it as an input to $\tilde{B}$, which again has a linear term in all variables. The degree of P' thus becomes linear in n in every variable, which increases the communication and verifier complexity from $O(n)$ to $O(n^2)$.

4.2 Computing $\tilde{Q}$

One potential hurdle towards succinctness is the matter of computing the value of $\tilde{Q}$, defined in Equation (9). It is clear that this function is efficiently evaluated over the boolean hypercube, since it is 1 iff $\vec{i} = \vec{k}$ and $\vec{j} = (\vec{k+1})$ and 0 otherwise. When one index is from the entire field $\mathbb{F}$ instead, we may consider the line defined by keeping this index variable and extrapolating from the values of $\tilde{Q}$ where this line intersects the boolean hypercube.

It is however not clear that $\tilde{Q}$ is efficiently evaluable when all its inputs lie in $\mathbb{F}$. The rest of the protocol takes $O(n)$ work for the verifier – logarithmic in the size of the path. Naively, $\tilde{Q}$ would need to be evaluated in time $O(2^n)$, due to the sum over $l \in \{0,1\}^n \setminus \{1 \dots 1\}$. Fortunately, this product can be evaluated more efficiently:

Lemma 3. Let $\vec{i}, \vec{j}, \vec{k} \in \mathbb{F}^n$. The polynomial

$$\tilde{Q}(\vec{i}, \vec{j}, \vec{k}) = \sum_{l \in \{0,1\}^n \setminus \{1 \dots 1\}} \tilde{\mathrm{eq}}(\overrightarrow{(l+1)}, \vec{j})\, \tilde{\mathrm{eq}}(\vec{l}, \vec{i})\, \tilde{\mathrm{eq}}(\vec{l}, \vec{k})$$

may be evaluated in $6n - 4$ field multiplications and $10n - 4$ additions.

Proof. First, observe that we can compute the v'th bit of $\overrightarrow{(l+1)}$ as follows:

$$(l+1)_v = l_v + (1 - 2l_v) \prod_{w<v} l_w.$$

This is the arithmetization of an adder with one of the addends fixed at 1 – it is equal to l_v plus the carry of all lower bits, which is 1 if and only if all lower bits are equal to 1. Furthermore, if both l_v and the carry are 1, the result should be 0 and so 2 is subtracted from the total. We can define $\tilde{Q}_m$ as

$$\begin{aligned}\tilde{Q}_m(\vec{i}, \vec{j}, \vec{k}) = \sum_{l \in \{0,1\}^m \setminus \{1 \dots 1\}} \prod_{v \in [m]} &(l_v i_v k_v + (1 - l_v)(1 - i_v)(1 - k_v)) \\ &\cdot \left(\left(l_v + (1 - 2l_v) \prod_{w<v} l_w \right) j_v + \left(1 - l_v - (1 - 2l_v) \prod_{w<v} l_w \right) (1 - j_v) \right)\end{aligned}$$

with $\tilde{Q}_n = \tilde{Q}$.

We then split $\tilde{Q}_m$'s sum into two cases: where $l_m \neq 1$ and $l_{m-1} \dots l_1 = 1 \dots 1$, and where $l_{m-1} \dots l_1 \neq 1 \dots 1$. This yields the recursion

$$\begin{aligned}\tilde{Q}_m(\vec{i}, \vec{j}, \vec{k}) = (1 - i_m)(1 - k_m) j_m \prod_{w<m} i_w k_w (1 - j_w) \\ + \left[i_m k_m j_m + (1 - i_m)(1 - k_m)(1 - j_m) \right] \tilde{Q}_{m-1}(\vec{i}_{m-1\dots1}, \vec{j}_{m-1\dots1}, \vec{k}_{m-1\dots1}),\end{aligned}$$

where we define the base case as $\tilde{Q}_1(i_1, j_1, k_1) = (1 - i_1)(1 - k_1) j_1$ (the only valid assignment is $\vec{i} = \vec{k} = 0$ and $\vec{j} = 1$).

From this, it follows that we can compute $\tilde{Q}(\vec{i}, \vec{j}, \vec{k})$ efficiently, not just over the boolean hypercube but over $\mathbb{F}^{2n+1}$, as follows: precompute the products $i_m j_m k_m, \dots, (1 - i_m)(1 - j_m)(1 - k_m)$, after that compute $\tilde{Q}_1$ and then incrementally the values $\tilde{Q}_v(i_v \dots i_1, j_v \dots j_1, k_v \dots k_1)$ and $\prod_{w<v} i_w k_w (1 - j_w)$ for $v \in \{2, \dots, n\}$ until we arrive at $\tilde{Q} = \tilde{Q}_n$. □

4.3 Supporting Arbitrary Length Walks

Since in our construction the path length e is halved in every round, natively only path lengths that are powers of two are supported. While this is unlikely to be an issue for most applications, where the goal is simply to attain at least some 'security level', it is nonetheless a worthwhile extension to be able to support isogeny walks of arbitrary length.

To achieve this, we 'puncture' the polynomial $\tilde{Q}$ to allow a prover to provide only a partial path of length e'. The new multilinear polynomial $\tilde{Q}_{e'}(\vec{i}, \vec{j}, \vec{k})$ is constructed such that all terms with $i > e'$ receive a factor 0.

This does not affect the proof size, but does require the verifier to do a tiny amount of additional work. The prover can simply skip these indices when computing $\tilde{Q}$'s evaluation table and is thus not impacted either. For more details on this approach see Appendix D, where we additionally show (Lemma 10) that the verifier needs to do just $2(n-1)$ additional field multiplications compared to Lemma 3. This may also allow tighter packing of the masking values, and potentially enable optimizations for multilinear PCS's since we now know that a fixed number of evaluations is always zero. We leave these optimizations for future work.

4.4 Adding Zero-Knowledge

There are two categories of leakage that we must prevent to make our protocol zero-knowledge: first, the evaluations of $\tilde{B}$ that are revealed as openings of the polynomial commitment at the end of the protocol. These are masked by adding the variable x to $\tilde{B}$, thereby doubling the size of the polynomial, and sampling the additional coefficients uniformly at random. For example, at the end of the sumcheck protocol, we will open $\tilde{B}(r_x, 0, \vec{r_i})$. This hides $\tilde{B}(0, 0, \vec{r_i})$, which would otherwise leak information about $(A_i)_{i \in \{0,\dots,e\}}$.

The second form of leakage is due to the sumcheck protocol. Here we have added the mask M. We would like to embed this mask in the multilinear polynomial $\tilde{B}$. However, since P has degree 3 in every variable (Equation (8)), this is not straightforward. Fortunately, M does not need to be an arbitrary polynomial of degree 3 in every variable, it is enough that it hides the specific revealed evaluations, as observed in the work of Libra [XZZ+19]. Unlike Libra, which adds a new univariate polynomial of low degree per sumcheck round, we would like to avoid sending additional polynomials, instead packing the mask in the existing polynomial $\tilde{B}$. To this end, we let

$$M(\vec{z}) = \tilde{B}(1, 0, \vec{0}) + \sum_{d \in [3]} \sum_{v \in [2n+1]} \tilde{B}(1, 0, 3(v - \vec{1}) + d) z_v^d, \tag{11}$$

so that M is a polynomial of degree 3 in every variable, its coefficients being the first $6n + 4$ evaluations of $\tilde{B}$ (in lexicographic ordering). The value of $\tilde{B}(1, \vec{0})$ is set such that M sums to 0 over the boolean hypercube. This avoids having to send the initial sumcheck value.

While M fully masks the sumcheck evaluations, it may seem that we have simply shifted the issue – to verify the evaluation of M we need to open $\tilde{B}$ at all $6n + 4$ points to obtain M's coefficients. This would fully reveal the masking polynomial, defeating its purpose. Therefore, as soon as $\vec{r_z}$ has been fixed, $\mathcal{P}$ instead sends

$$M'(\vec{r_z}, y) = \tilde{B}(1, y, \vec{0}) + \sum_{d \in [3]} \sum_{v \in [2n+1]} \tilde{B}(1, y, 3(v - \vec{1}) + d)(\vec{r_z})_v^d. \tag{12}$$

$\mathcal{V}$ chooses a random value r_y, after which the evaluations $\tilde{B}(1, r_y, \vec{c})$ for $\vec{c} \in \{0, \dots, 6n + 3\}$ are revealed. $\mathcal{V}$ then checks that these match $M'(\vec{r_z}, r_y)$.

4.5 The PIOP

With the relevant building blocks in place, we now present the full PIOP in Figure 1. We can see how the prover constructs polynomial $\tilde{B}$ and subsequently receives challenges $\vec{r_h}$ and $\vec{r_k}$. We then perform $2n + 1$ rounds of sumcheck on polynomial P constructed in Section 4.1. In the last steps, the verifier samples challenge r_y to ensure that the components of the mask can be opened without revealing their respective contributions. The openings of $\tilde{B}$ are masked by a linear combination due to r_x. The proving complexity is $O(2^n) = O(e)$, linear in the length of the walk, and the verification complexity is $O(n) = O(\log(e))$. $\mathcal{P}$ sends $6n + 5$ field elements in addition to the polynomial oracle $\tilde{B}$. We analyze the security of our PIOP in Section 5.

5 Security of our PIOP

We now prove that PIOP in Figure 1 is secure.

Lemma 4. PIOP has perfect completeness.

Interaction phase: The prover and verifier engage in the following protocol:

1. The prover $\mathcal{P}$ computes a multilinear polynomial $\tilde{B}$, where

$$\begin{aligned} \forall i \in \{0,\ldots,e\}: &\quad \tilde{B}(0,0,\vec{i}) = A_i, \quad \tilde{B}(0,1,\vec{i}) = C_i, \\ \forall i \in \{1,\ldots,2e+1\}: &\quad \tilde{B}(1,\vec{i}) \in_R \mathbb{F}, \end{aligned}$$

   and $\tilde{B}(1,\vec{0})$ is chosen s.t. $\sum_{\vec{i}\in\{0,1\}^n}\sum_{x\in\{0,1\}}\sum_{\vec{j}\in\{0,1\}^n} M(\vec{i},x,\vec{j}) = 0$, for M defined in Equation (11).
2. $\mathcal{P}$ sends $\tilde{B}$ to $\mathcal{V}$.
3. $\mathcal{V}$ samples $\vec{r_h} \in (\mathbb{F}^\times)^2$, $\vec{r_k} \in (\mathbb{F}^\times)^n$ and sends these values to $\mathcal{P}$.
4. $\mathcal{P}$ and $\mathcal{V}$ perform $2n+1$ rounds of sumcheck: in the i'th round, $\mathcal{P}$ sends polynomial $\hat{p}_i(X)$ and $\mathcal{V}$ sends $\rho_i \in \mathbb{F}^\times$, where

$$\hat{p}_i(X) = \sum_{\vec{z}\in\{0,1\}^{2n+1-i}} P(\vec{r_h},\vec{r_k},\rho_1,\ldots,\rho_{i-1},X,\vec{z}).$$

   Instead of sending $\hat{p}_i$ in its entirety, it is derived from the constraint that $\hat{p}_i(0)+\hat{p}_i(1) = \hat{p}_{i-1}(\rho_{i-1})$ for $i > 1$ or $\hat{p}_i(0)+\hat{p}_i(1) = 0$ for $i = 1$.
5. $\mathcal{P}$ sends $M'(\vec{r_i}, r_x, \vec{r_j}, y)$ with M' as defined in Eq. (12).
6. $\mathcal{V}$ samples $r_y \in \mathbb{F}^\times$ uniformly at random and sends it to $\mathcal{P}$.

Query phase: $\mathcal{V}$ queries $\tilde{B}$ at $(r_x,0,\vec{r_i})$, $\tilde{B}(r_x,0,\vec{r_j})$, $\tilde{B}(r_x,1,\vec{r_i})$ and $\tilde{B}(r_x,1,\vec{r_j})$, as well as the starting and end points $\tilde{B}(0,0,\vec{0})$, $\tilde{B}(0,1,\vec{0})$, $\tilde{B}(0,0,\vec{1})$ and $\tilde{B}(0,1,\vec{1})$, and evaluations required for the mask: $\tilde{B}(1,r_y,\vec{c})$ for $c \in \{0,\ldots 6n+3\}$.

Decision phase: $\mathcal{V}$ accepts if

- The starting and end points match:
  - $\tilde{B}(0,0,\vec{0}) = A_0$ and $\tilde{B}(0,0,\vec{1}) = A_e$,
  - $\tilde{B}(0,1,\vec{0}) = C_0$ and $\tilde{B}(0,1,\vec{1}) = C_e$.
- The constraint system is satisfied:

$$\begin{aligned} \hat{p}(\rho_i) = \tilde{\mathrm{eq}}(\vec{r_{h0}},1)\,\tilde{\mathrm{eq}}(r_x,0)\tilde{Q}(\vec{r_i},\vec{r_j},\vec{r_k}) \\ \Big(\Big[a_{A_j}(\vec{r_{h1}})\tilde{B}(r_x,0,\vec{r_j}) + a_{A_i}(\vec{r_{h1}})\tilde{B}(r_x,0,\vec{r_i})\Big]\big[\tilde{B}(r_x,0,\vec{r_j}) - \tilde{B}(r_x,0,\vec{r_i})\big] \\ - \Big[\tilde{B}(r_x,1,\vec{r_i}) + a_{C_j}(\vec{r_{h1}})\tilde{B}(r_x,1,\vec{r_j})\Big]\Big) + M'(\vec{r_i},r_x,\vec{r_j},0). \end{aligned}$$

- The mask was correctly evaluated (Eq. (12) with $\vec{r_z} = (\vec{r_i},r_x,\vec{r_j})$):

$$M'(\vec{r_z},r_y) = \tilde{B}(1,r_y,\vec{0}) + \sum_{d\in[3]}\sum_{v\in[2n+1]} \tilde{B}(1,r_y,3(v-\vec{1})+d)(\vec{r_z})_v^d.$$

Figure 1: The full PIOP for relation $\mathcal{R}$ (Equation 6).

Proof. For a valid witness $\mathbf{w} = ((A_i)_{i\in\{0,\dots,e\}}, (C_i)_{i\in\{0,\dots,e\}})$, we have for every $\vec{k} \in \mathbb{F}^n$ and $\vec{h} \in \mathbb{F}^2$:

$$\sum_{\vec{i},\vec{j}\in\{0,1\}^n, x\in\{0,1\}} P(\vec{h},\vec{k},\vec{i},x,\vec{j}) = 0,$$

since for every $\vec{k}$, this relation captures one step of the isogeny path. This means that the claim at the start of Step 4 holds. Next the prover and verifier perform a sumcheck argument, arriving at a new relation that holds by construction. The prover finally sends polynomial M' honestly. Since the witness is valid and the polynomials sent by the prover are correct w.r.t. $\tilde{B}$, the evaluations of $\tilde{B}$ are valid and final checks succeed and verifier outputs 1. □

Multi-round protocols that are only proven to be knowledge sound may suffer from large soundness losses when made non-interactive using the Fiat–Shamir transformation. Therefore, instead of regular knowledge soundness, we focus on the stronger notion of *round-by-round knowledge soundness* [CCH+19], which is equivalent to *state-restoration knowledge soundness* [BCS16, Hol19]. Such a PIOP may be compiled to a state-restoration knowledge sound argument, i.e. with a knowledge soundness error that is 'well-behaved' under the Fiat–Shamir transformation [CGKY25].

Lemma 5. `PIOP` has round-by-round knowledge soundness errors

$$\left(\frac{n+2}{|\mathbb{F}|-1}, \frac{3}{|\mathbb{F}|-1}, \dots \frac{3}{|\mathbb{F}|-1}, \frac{1}{|\mathbb{F}|-1}\right)_{i\in[k]}.$$

Proof. We define state function `RBRState` that indicates whether the protocol execution is doomed ($\texttt{RBRState}(\mathbf{x},\texttt{trc}) = 0$) or not ($\texttt{RBRState}(\mathbf{x},\texttt{trc}) = 1$). In the following, let $\texttt{trc}'$ denote transcript `trc` without the last message.

- Empty transcript: If $\texttt{trc} = \emptyset$ or $\texttt{trc} = (\tilde{B})$, $\texttt{RBRState}(\mathbf{x},\texttt{trc}) = 0$.
- Prover moves: If the prover sent the last message, $\texttt{RBRState}(\mathbf{x},\texttt{trc}) = \texttt{RBRState}(\mathbf{x},\texttt{trc}')$.
- If $\texttt{trc} = (\tilde{B}, (\vec{r_h}, \vec{r_k}))$, then $\forall i \in \{0,\dots,e\}$ let $A_i = \tilde{B}(0,0,\vec{i})$ and $C_i = \tilde{B}(0,1,\vec{i})$.
  Then $\texttt{RBRState}(\mathbf{x},\texttt{trc}) = 0$ if and only if
  - The polynomial $\tilde{B}$ is not a valid witness for $\mathbf{x}$, i.e.
    - the starting and end points do not match: $\mathbf{x} \neq (A_0, C_0, A_e, C_e)$, or
    - $(A_i)_{i\in\{0,\dots,e\}}$, $(C_i)_{i\in\{0,\dots,e\}}$ do not satisfy Equations (4) and (5), or
    - $\sum_{\vec{i}\in\{0,1\}^n}\sum_{x\in\{0,1\}}\sum_{j\in\{0,1\}^n} M(\vec{i},x,\vec{j}) \neq 0$,
  - *and*, $\sum_{\vec{i}\in\{0,1\}^n}\sum_{x\in\{0,1\}}\sum_{\vec{j}\in\{0,1\}^n} P(\vec{r_h},\vec{r_k},\vec{i},x,\vec{j}) \neq 0$.
- During Step 4, in the i'th sumcheck round, let $\texttt{trc} = (\tilde{B}, (\vec{r_h},\vec{r_k}), \hat{p}_1, r_{z1}, \dots, \hat{p}_i, r_{zi})$.
  Then $\texttt{RBRState}(\mathbf{x},\texttt{trc}) = 0$ if and only if
  - the starting and end points do not match, or
  - $\hat{p}_i(r_{zi}) \neq \sum_{\vec{z}\in\{0,1\}^{2n+1-i}} P(\vec{r_h},\vec{r_k},r_{z1},\dots r_{zi-1},r_{zi},\vec{z})$.
- After Step 6, for $\texttt{trc} = (\dots, \hat{M}', r_y)$, let $\texttt{RBRState}(\mathbf{x},\texttt{trc}) = 0$ if and only if any of the verification equations are not satisfied.

Clearly `RBRState` is correct: the empty transcript is always doomed, a prover message can never make a doomed state not doomed, and if the final transcript is doomed, then verification fails. It remains to be shown that if there is a non-negligible chance of leaving a doomed state, then $\mathcal{E}$ can obtain a valid witness. It follows from Lemma 2 that a valid assignment for $\mathcal{R}$ satisfying Equations (4) and (5) can be read off from an assignment for $\mathcal{R}'$.

The first point where a prover can leave a doomed state, is after the first challenge $(\vec{r_h},\vec{r_k})$ has been sent, which comes after $\mathcal{P}$ has sent $\tilde{B}$. A doomed transcript can become valid if either $\tilde{B}$ is a

valid witness, in which case $\mathcal{E}$ clearly succeeds, or if $\tilde{B}$ is not a valid witness. In this case $(\vec{r_h}, \vec{r_k})$ is a root of q, i.e.

$$q(\vec{h}, \vec{k}) = \sum_{\vec{i} \in \{0,1\}^n, x \in \{0,1\}, \vec{j} \in \{0,1\}^n} P(\vec{h}, \vec{k}, \vec{i}, x, \vec{j}) \neq 0,$$

but $q(\vec{r_h}, \vec{r_k}) = 0$. This happens with probability at most $\frac{n+2}{|\mathbb{F}|-1}$.

In the first sumcheck round, we have $\hat{p}_1(0) + \hat{p}_1(1) = 0$, and so $\hat{p}_1(X) \neq \sum_{\vec{z} \in \{0,1\}^{2n}} P(\vec{r_h}, \vec{r_k}, X, \vec{z})$. Since $\texttt{RBRState}(\mathbf{x}, \texttt{trc}') = 0$, the starting and end points do not match, and therefore the only way to end up in a transcript that is not doomed is by receiving a challenge r_{z1} for which it holds that $\hat{p}_1(r_{z1}) = \sum_{\vec{z} \in \{0,1\}^{2n}} P(\vec{r_h}, \vec{r_k}, r_{z1}, \vec{z})$. This happens with probability at most $\frac{3}{|\mathbb{F}|-1}$.

By recursion, for the other sumcheck rounds

$$\hat{p}_i(0) + \hat{p}_i(1) = \hat{p}_{i-1}(r_{zi-1}) \neq \sum_{\vec{z} \in \{0,1\}^{2n+2-i}} P(\vec{r_h}, \vec{r_k}, r_{z1}, \ldots, r_{zi-1}, \vec{z}),$$

and so $\hat{p}_i(X) \neq \sum_{\vec{z} \in \{0,1\}^{2n+1-i}} P(\vec{r_h}, \vec{r_k}, r_{z1}, \ldots, r_{zi-1}, X, \vec{z})$. By selecting a random value r_{zi} for X, there is at most a probability $\frac{3}{|\mathbb{F}|-1}$ of ending up in a state that is not doomed.

At the end of the protocol, if the transcript was doomed in the second to last state, either the starting and end points do not match, or

$$\hat{p}_{2n+1}(r_{z2n+1}) \neq P(\vec{r_h}, \vec{r_k}, r_{z1}, \ldots, r_{z2n+1}) = P(\vec{r_h}, \vec{r_k}, \vec{r_i}, r_x, \vec{r_j}).$$

In the former case, the final state will always be doomed. In the latter case, suppose that, for the last prover message $\hat{M}'(Y)$ (claimed to equal M'),

$$\hat{M}'(Y) = M'(\vec{r_i}, r_x, \vec{r_j}, Y) = \tilde{B}(1, Y, \vec{0}) + \sum_{d \in [3]} \sum_{v \in [2n+1]} \tilde{B}(1, Y, 3(v - \vec{1}) + d)(\vec{r_z})_v^d.$$

Then

$$\begin{aligned}\hat{p}_{2n+1}((\vec{r_z})_{2n+1}) \neq\ & \tilde{\mathrm{eq}}(\vec{r_{h0}}, 1)\, \tilde{\mathrm{eq}}(r_x, 0) \tilde{Q}(\vec{r_i}, \vec{r_j}, \vec{r_k}) \\ & \Big(\Big[a_{A_j}(\vec{r_{h1}}) \tilde{B}(r_x, 0, \vec{r_j}) + a_{A_i}(\vec{r_{h1}}) \tilde{B}(r_x, 0, \vec{r_i}) \Big] \Big[\tilde{B}(r_x, 0, \vec{r_j}) - \tilde{B}(r_x, 0, \vec{r_i}) \Big] \\ & \quad - \Big[\tilde{B}(r_x, 1, \vec{r_i}) + a_{C_j}(\vec{r_{h1}}) \tilde{B}(r_x, 1, \vec{r_j}) \Big] \Big) + \hat{M}'(\vec{r_i}, r_x, \vec{r_j}, 0),\end{aligned}$$

and verification will fail. If, on the other hand $\hat{M}'(Y) \neq M'(Y)$, then $\hat{M}'(r_y) \neq M'(r_y)$ except with probability $\frac{1}{|\mathbb{F}|-1}$. This means that the final transcript is doomed except with probability $\frac{1}{|\mathbb{F}|-1}$. □

Lemma 6. PIOP is perfect honest-verifier zero-knowledge.

Proof. In both the honest and the simulated protocol, the challenges are picked uniformly at random from $\mathbb{F}^\times$. It thus remains to be shown that the prover messages are identically distributed.

In the sumcheck rounds, $\mathcal{P}$ reveals the following evaluation points:

- $\forall X \in [3] : \sum_{\vec{i} \in \{0,1\}^{n-1}, x \in \{0,1\}, \vec{j} \in \{0,1\}^n} P(\vec{r_h}, \vec{r_k}, X, i_2 \ldots i_n, x, \vec{j})$,
- $\forall X \in [3] : \sum_{\vec{i} \in \{0,1\}^{n-2}, x \in \{0,1\}, \vec{j} \in \{0,1\}^n} P(\vec{r_h}, \vec{r_k}, (\vec{r_i})_1, X, i_3 \ldots i_n, x, \vec{j})$,
- $\ldots$,
- $\forall X \in [3] : P(\vec{r_h}, \vec{r_k}, \vec{r_i}, r_x, (\vec{r_j})_{1 \ldots n-1}, X)$.

Due to the contribution of $M(\vec{i}, x, \vec{j})$, these are all uniformly distributed over $\mathbb{F}$. In the honest execution, the polynomial $\tilde{B}$ satisfies the constraint system and thus passes all checks. In the simulated execution, $\vec{r_h}$ and thus also the linear dependence between the constraints and the mask are known and therefore $\mathcal{S}$ is able to pick a value $\tilde{B}(1, \vec{0})$ such that the system is satisfied, i.e. that $q(\vec{h}, \vec{k}) = 0$ (Equation (10)). The corresponding transcripts are therefore identically distributed.

The simulator $\mathcal{S}$ produces a simulated view of the interaction between $\mathcal{P}$ and $\mathcal{V}$:

1. Sample $r_x, r_y \in \mathbb{F}$, $\vec{r}_h \in \mathbb{F}^2$, $\vec{r}_i, \vec{r}_j, \vec{r}_k \in \mathbb{F}^n$ uniformly at random.
2. Let $\tilde{B}(1, \vec{i}) = 0$ for $i \in \{0, \ldots, 2e+1\}$, i.e. set the part corresponding to $(A_i)_{i\in\{0,\ldots,e\}}$ and $(C_i)_{i\in\{0,\ldots,e\}}$ to 0.
3. Let $\tilde{B}(0, \vec{i}) \in \mathbb{F}$ for $i \in \{1, \ldots, 2e+1\}$, i.e. sample the remainder of $\tilde{B}$ uniformly at random.
4. Let
$$\tilde{B}(1, \vec{0}) = \frac{-1}{2^{2n+1}} \sum_{\vec{z}\in\{0,1\}^{2n+1}} \left[P'(\vec{r}_h, \vec{r}_k, \vec{z}) + \sum_{d\in[3]} \sum_{v\in[2n+1]} \tilde{B}(1, 0, 3(v - \vec{1}) + d) z_v^d \right].$$
5. Continue by following Protocol 1 honestly, starting from Step 2.

Figure 2: The canonical simulator $\mathcal{S}$ for the PIOP.

The polynomial $M'(\vec{r}_i, r_x, \vec{r}_j, y)$ is fully determined by the verifier checks on the final openings of $\tilde{B}$. It suffices to show that the openings are uniformly distributed. For $\tilde{B}(r_x, 0, \vec{r}_i)$, $\tilde{B}(r_x, 0, \vec{r}_j)$, $\tilde{B}(r_x, 1, \vec{r}_i)$, and $\tilde{B}(r_x, 1, \vec{r}_j)$, note how $\tilde{B}$ may be written as $\tilde{B}(\vec{z}) = \sum_{\vec{z'}\in\{0,1\}^{n+2}} \tilde{\text{eq}}(\vec{z}, \vec{z'})\tilde{B}(\vec{z'})$. In particular, consider the contributions

$$\begin{aligned} \cdots &+ \tilde{\text{eq}}(\vec{z}, (1,0,0,\vec{1}))\tilde{B}(1,0,0,\vec{1}) + \tilde{\text{eq}}(\vec{z}, (1,0,1,\vec{1}))\tilde{B}(1,0,1,\vec{1}) \\ &+ \tilde{\text{eq}}(\vec{z}, (1,1,1,\vec{1}))\tilde{B}(1,1,1,\vec{1}) + \tilde{\text{eq}}(\vec{z}, (1,1,0,\vec{1}))\tilde{B}(1,1,0,\vec{1}). \end{aligned}$$

The evaluations $\tilde{B}(r_x, 0, \vec{r}_i)$ and $\tilde{B}(r_x, 0, \vec{r}_j)$ get a non-zero contribution from the first two terms of this sum, while $\tilde{B}(r_x, 1, \vec{r}_i)$ and $\tilde{B}(r_x, 1, \vec{r}_j)$ get a non-zero contribution from the second two terms of this sum. Furthermore, the contributions are all linearly independent unless $\vec{r}_i = \vec{r}_j$ in which case the evaluations are actually pairwise identical, in both the honest as well as the simulated case. The evaluation points do not overlap with the mask since the mask requires $6n+4$ evaluation points. Its binary value will thus not reach $\vec{1} \in \mathbb{F}^{n-1}$ as long as $n \geq 7$. Since these values are picked uniformly at random, these openings are identically distributed in both the simulated and the honest case.

Finally, for the revealed mask we note that the evaluations $\tilde{B}(1, r_y, \vec{c})$ for $\vec{c} \in \{0, \ldots, 6n+3\}$ all receive a non-zero contribution from $\tilde{\text{eq}}(r_y, 1)\tilde{B}(1, 1, \vec{c})$, which are uniformly distributed over $\mathbb{F}$ for all such $\vec{c}$. We conclude that honest protocol execution and a simulated transcript have the same distribution. □

Theorem 7. *PIOP can be compiled to a zero-knowledge, succinct weakly simulation extractable NARG for relation $\mathcal{R}$.*

Proof. It follows from Lemmas 4, 5 and 6 that PIOP is complete, round-by-round knowledge sound and zero-knowledge. We apply the compiler from [CGKY25] with a state-restoration binding PCS to obtain a complete, zero-knowledge and state-restoration knowledge sound argument. By its state-restoration soundness the Fiat–Shamir-transformed argument is a knowledge sound zk-SNARK. Succinctness in size follows from the fact that we send a constant number of field elements in logarithmically many rounds, as well as the succinctness of the PCS. The verifier performs logarithmically many operations (see Section 4.2). Applying Theorem 1 we conclude that this argument is weak simulation extractable. □

We will refer to the NARG described in Theorem 7 as *SPRINT* (Succinct Proof for Radical Isogenies, Nimbly Transformed).

6 Applications of SPRINT

In this section we present two applications of SPRINT (defined after Theorem 7). First, we show how it can be used as an alternative proof system within the multi-party generation of supersingular curves of unknown endomorphism ring proposed in [BCC+23]. Second, using a compiler from [DHLW10], we turn SPRINT into a signature scheme. As the PCS underlying our PIOP, we rely on DeepFold [GLH+25], and for this reason we refer to the resulting PoK/signature by the name *DeepSPRINT*. We benchmark our prototypical implementation and compare it to the current state-of-the-art for both applications.

6.1 Supersingular Curves You Can Trust

Since SPRINT does not need knowledge of the endomorphism ring of the starting curve, it can be used as a replacement of currently deployed proofs of knowledge that also do not have this requirement. One such example is the computation of supersingular elliptic curves of unknown endomorphism ring (SECUERs).

In general, the generation of SECUERs can be realized either via a trusted setup or via a multiparty ceremony. The latter approach is explored in [BCC+23], where the following scheme is described: to sequentially construct a walk $E_0 \to E_1 \to \cdots \to E_e$ in the isogeny graph, each party i, on input E_{i-1}, outputs the codomain E_i of a random isogeny $\varphi_i \colon E_{i-1} \to E_i$ and provides a NIZK proof that they know φ_i. Assuming that the NIZK proof is knowledge sound and statistically zero-knowledge, the hardness of the endomorphism ring problem and GRH, such a protocol produces a SECUER if at least one of the parties is honest (i.e., discards its secret isogeny) [BCC+23, Thm. 21]. They also opt for 2-isogeny walks of length $k \approx \log(p) + 2\lambda \approx 6\lambda$ in order to guarantee the output curve is statistically close to uniformly distributed in the set of supersingular curves[9].

This scheme can be instantiated with any PoK satisfying the above assumptions, provided that it does not require the prover to know the endomorphism ring of the domain or codomain curve. The one proposed in [BCC+23] achieves statistical zero-knowledge via relatively long coprime-degree isogeny walks, which negatively impact its efficiency. Moreover, it enjoys special soundness only for the relaxed relation

$$\mathcal{R}^* = \{((E, E'), \phi) \mid \phi \colon E \to E' \text{ is an isogeny of degree } \ell^k d^{2i} \text{ for } 0 \leq i \leq n\},$$

where n is a sufficiently large integer depending on p, ℓ, d and the targeted security level λ. This means that the above approach proves knowledge of isogenies of given degree only up to a powersmooth factor.

Both limitations are addressed in [CLL23], where the isogeny relation is expressed as a polynomial relation from modular polynomials, turned into a rank-1 constraint system (R1CS) over either $\mathbb{F}_{p^2}$ or $\mathbb{F}_p$, and plugged into a generic proof system. At the price of slightly worse proof sizes, it reduces proof and verification times by an order of magnitude. Further improvements were recently proposed in [dHKM+25, LP25, dHMSS26], which describe R1CS relations based on canonical modular polynomials, radical isogenies and Atkin modular polynomials, respectively.

Our proof system lies conceptually between the tailor-made approach of [BCC+23] and the generic one of [CLL23], leveraging a suitably tailored PIOP in which any PCS can be plugged in.

We prototypically implement DeepSPRINT in Rust[10] and compare it with the previously mentioned approaches[11]. For a direct comparison, we opt to instantiate our protocol with the same choice of primes and path lengths, emphasizing that primes of this form are not necessary for our construction. Indeed, a smaller prime of length 2λ is sufficient to achieve λ bits of security against the attack of [DG16][12], and allows for uniform mixing in walks of length $\approx \log(p) + 2\lambda$.

Table 1 contains benchmarks for the SIDH parameter sets `p434`, `p503`, `p610` and `p751`, while Table 2 gives benchmarks[13] for the SQIsign primes $5 \cdot 2^{248} - 1$, $65 \cdot 2^{376} - 1$ and $27 \cdot 2^{500} - 1$.

[9] See [BCC+23, Thm. 7 & Lemma 14] for a tighter calculation of uniform mixing path lengths.

[10] The DeepFold implementation we use has no support for (multi-point, single-polynomial) batching, and instead proves only a single evaluation. Batching would involve taking a linear combination and continuing as normal; based on our measurements we expect a performance impact of less than one percent.

[11] The PoK from [LP25] does not provide benchmarks for comparison. However, since their R1CS has the same number of constraints as the one from [dHMSS26], we expect the performance and proof sizes to roughly be the same.

[12] As noted in Section 1.2, we do not account for the recent attack by Wesolowski [Wes26] in our security discussion.

[13] The SECUER PoK is only implemented for the SIDH parameter sets.

Table 1: Benchmarks of our isogeny proof of knowledge DeepSPRINT and recent isogeny proofs of knowledge for the SIDH parameter sets `p434`, `p503`, `p610`, `p751`, using an Intel Core i7-1370P at 1.9GHz with turbo boost disabled and restricted to one thread for SECUER, the classical modular polynomial approach and our scheme, resp. projected (cf. [dHMSS26, Table 3]) for the other modular polynomial approaches. Timings include DeepFold's preprocessing.

	Prime p	Path length e	zk-SNARK	Proof size (kB)	Time (s) Prove	Verify
SECUER[1a] [BCC+23]	$2^{216}3^{137}-1$	705	–	191	14.9	1.69
	$2^{250}3^{159}-1$	774	–	216	24.3	2.74
	$2^{305}3^{192}-1$	1010	–	404	76.5	8.16
	$2^{372}3^{239}-1$	1280	–	663	157.3	17.9
Classical MP[1b] [CLL23]	$2^{216}3^{137}-1$	705	Aurora	277	5.16	0.40
			Ligero	3281	1.66	1.69
	$2^{250}3^{159}-1$	774	Aurora	313	6.28	0.45
			Ligero	3778	2.18	2.22
	$2^{305}3^{192}-1$	1010	Aurora	570	7.65	0.87
			Ligero	4568	3.73	3.04
	$2^{372}3^{239}-1$	1280	Aurora	852	17.6	2.55
			Ligero	11302	5.91	14.0
Canonical MP [dHKM+25]	$2^{216}3^{137}-1$	705	Aurora	257	3.73	0.30
			Ligero	2840	1.20	1.26
	$2^{250}3^{159}-1$	774	Aurora	291	4.54	0.33
			Ligero	3271	1.57	1.67
	$2^{305}3^{192}-1$	1010	Aurora	531	5.53	0.65
			Ligero	3955	2.70	2.28
	$2^{372}3^{239}-1$	1280	Aurora	795	12.8	1.91
			Ligero	9786	4.28	10.5
Atkin MP [dHMSS26]	$2^{216}3^{137}-1$	705	Aurora	231	2.35	0.20
			Ligero	2319	0.76	0.84
	$2^{250}3^{159}-1$	774	Aurora	261	2.87	0.22
			Ligero	2670	0.99	1.11
	$2^{305}3^{192}-1$	1010	Aurora	479	3.50	0.43
			Ligero	3229	1.71	1.52
	$2^{372}3^{239}-1$	1280	Aurora	720	8.09	1.27
			Ligero	7990	2.71	7.00
DeepSPRINT[1c]	$2^{216}3^{137}-1$	705	–	160	0.44	0.04
	$2^{250}3^{159}-1$	774	–	178	0.61	0.05
	$2^{305}3^{192}-1$	1010	–	266	0.86	0.08
	$2^{372}3^{239}-1$	1280	–	456	2.52	0.22

[1a] C source code: `https://github.com/trusted-isogenies/SECUER-pok`
[1b] C++ source code: `https://github.com/levanin/libiop-fp2`
[1c] Rust source code: `https://github.com/QuSAC/DeepSPRINT`. We use the technique described in Section 4.3 to adapt our scheme to the given path lengths.

Table 2: Benchmarks of our isogeny proof of knowledge DeepSPRINT and recent isogeny proofs of knowledge for the SQIsign primes and paths of length $e \approx \log(p)$, using an Intel Core i7-1370P at 1.9GHz with turbo boost disabled and restricted to one thread for the classical modular polynomial approach and our scheme, resp. projected (cf. [dHMSS26, Table 3]) for the other modular polynomial approaches. Unlike Table 4, timings include DeepFold's preprocessing step.

	Prime p	Path length e	zk-SNARK	Proof size (kB)	Time (ms)	
					Prove	Verify
Classical MP[2a] [CLL23]	$5 \cdot 2^{248} - 1$	256	Aurora	162	1365	109
			Ligero	1298	485	526
	$65 \cdot 2^{376} - 1$	384	Aurora	355	2542	304
			Ligero	2627	1940	1786
	$27 \cdot 2^{500} - 1$	512	Aurora	677	5795	959
			Ligero	5206	1855	4346
Canonical MP [dHKM+25]	$5 \cdot 2^{248} - 1$	256	Aurora	149	979	82
			Ligero	1123	348	394
	$65 \cdot 2^{376} - 1$	384	Aurora	328	1829	228
			Ligero	2274	1396	1338
	$27 \cdot 2^{500} - 1$	512	Aurora	627	4178	719
			Ligero	4506	1337	3271
Atkin MP [dHMSS26]	$5 \cdot 2^{248} - 1$	256	Aurora	131	613	55
			Ligero	917	218	263
	$65 \cdot 2^{376} - 1$	384	Aurora	291	1149	152
			Ligero	1857	877	892
	$27 \cdot 2^{500} - 1$	512	Aurora	559	2631	479
			Ligero	3679	842	2181
DeepSPRINT[2b]	$5 \cdot 2^{248} - 1$	256	–	82	27	3.5
	$65 \cdot 2^{376} - 1$	384	–	171	131	15
	$27 \cdot 2^{500} - 1$	512	–	246	166	20

[2a] C++ source code: `https://github.com/levanin/libiop-fp2`
[2b] Rust source code: `https://github.com/QuSAC/DeepSPRINT`

For the SIDH parameters we achieve 1.2-1.5× smaller proofs than SECUER, 1.1-2.0× faster proving[14] than the Ligero instantiation of [dHMSS26] and 4.4-5.8× faster verification than the Aurora instantiation, all the previously smallest/fastest of their kind. For the SQIsign primes, we achieve 5.1-8.1× faster proving times than Atkin-Ligero, while achieving 1.6-2.3× smaller proof sizes and 10-24× faster verification times than Atkin-Aurora.[15] Note that all of the above improvements over the prior approaches are attained simultaneously, whereas SECUER suffers from significantly slower timings, Ligero from significantly larger proof sizes and verification times, and Aurora from slightly larger proof sizes and significantly slower proving times.

In contrast to the `libiop` library used for benchmarking the R1CS-based approaches, the implementation of SECUER benefits noticeably from parallelization. For this reason we also compare the fully implemented SECUER PoK to our implementation using 20 logical cores in Table 3; there we see that we still achieve 10-30× faster proving and 14-38× faster verification.

[14] The 'low' improvement factor of 1.1 for the `p751` proving time comes from the large gap between the path length 1280 and the next 2-power 2048.

[15] Surprisingly, we also see in Table 2 that Ligero's proof generation is slightly faster for the level V prime than for level III; as this behavior persisted across multiple tests, we expect it to be caused by unfavorable padding of the level III polynomial used in Ligero.

Table 3: Benchmarks of our isogeny proof of knowledge DeepSPRINT and the isogeny proof of knowledge SECUER for the SIDH parameter sets `p434`, `p503`, `p610`, `p751`, using an Intel Core i7-1370P at 1.9GHz with turbo boost disabled and multithreading using 20 logical cores.

	Prime p	Path length e	Proof size (kB)	Time (s)	
				Prove	Verify
SECUER[3a] [BCC+23]	$2^{216}3^{137}-1$	705	191	4.38	0.47
	$2^{250}3^{159}-1$	774	216	8.13	0.75
	$2^{305}3^{192}-1$	1010	404	25.0	2.73
	$2^{372}3^{239}-1$	1280	663	57.2	6.42
DeepSPRINT[3b]	$2^{216}3^{137}-1$	705	162	0.43	0.03
	$2^{250}3^{159}-1$	774	179	0.60	0.05
	$2^{305}3^{192}-1$	1010	270	0.82	0.07
	$2^{372}3^{239}-1$	1280	454	2.46	0.21

[3a] C source code: https://github.com/trusted-isogenies/SECUER-pok

[3b] Rust source code: https://github.com/QuSAC/DeepSPRINT. We use the technique described in Section 4.3 to adapt our scheme to the given path lengths.

6.2 A New Family of Isogeny-Based Signature Schemes

In this section, we first recall a folklore approach to construct EUF-CMA secure signatures from weakly simulation-extractable proof systems, and then build a family of signature schemes from SPRINT.

EUF-CMA Signatures from wSE Proof Systems. Given a (w)SE NIZK argument system $\Pi = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$, there is a folklore way to obtain EUF-CMA secure signatures. In particular, a signature simply represents a proof for an augmented relation

$$\mathcal{R}^\dagger = \{((\mathbf{x}, m), \mathbf{w}) : (\mathbf{x}, \mathbf{w}) \in \mathcal{R} \wedge m \in \{0,1\}^n\},$$

where $\mathcal{R}$ is a hard relation defining verification- and signing-key pairs $(vk, sk) \leftarrow \mathsf{Gen}_{\mathcal{R}}(1^\lambda)$, and m is the message to be signed. We note that, in practice, one would use a collision-resistant hash function H and replace m by $H(m)$ (e.g., as in [GM17]).

A signature σ for message m is then simply obtained by running $\sigma \leftarrow \Pi.\mathsf{Prove}(crs, (vk, m), sk)$ to get the proof σ with $crs \leftarrow \Pi.\mathsf{Setup}(1^\lambda)$, and verification is done by running $b \leftarrow \Pi.\mathsf{Verify}(crs, vk, \sigma)$.

To the best of our knowledge, this folklore approach has first been formalized in [DHLW10] (cf. also [Bel21] for the case of strongly EUF-CMA signatures from an SE NIZK), and later specifically for Fiat–Shamir transformed NIZKs in [FKMV12]. We recall a theorem from [DHLW10] here for completeness, which we straightforwardly adapt to the case of conventional signatures (i.e., their focus is leakage-resilience, but we can simply set the leakage to be the zero function) and instead of using labels to integrate messages we directly augment the statement by (a hash of) the message. Moreover, we note that [DHLW10] actually considers *true* wSE, meaning that only true statements are allowed to be queried to the simulator, which is clearly implied by wSE.

Theorem 8 ([DHLW10, Thm. 4.3]). *Let $\mathcal{R}^\dagger$ be a family of hard relations and let Π be a wSE NIZK argument for $\mathcal{R}^\dagger$. Then the above signature scheme is EUF-CMA secure.*

In particular, the EUF-CMA security of the resulting signature is tight given the straight-line extractability of the underlying NIZK argument. We hence obtain a family of signature schemes by augmenting the relation in Theorem 7 by the public message m in the statement and then using the compiler from [DHLW10] with any zero-knowledge round-by-round binding PCS. In

$\mathsf{Setup}(1^\lambda)$

1. Fix a 2λ-bit prime $p = c2^f - 1$ with $c > 0$ a small integer, and a path length e.
2. Return the public parameters $\mathsf{pp} = (p, E_0/\mathbb{F}_{p^2}, e, n)$, where the curve E_0 is encoded as $(A_0, C_0) = (0, 1) \in \mathbb{F}_{p^2}$, and n is the bit-length of the message space.

$\mathsf{KeyGen}(\mathsf{pp})$

1. Sample a random path $s \leftarrow \{0,1\}^e$ of length e.
2. Starting from (A_0, C_0), iterate according to the radical isogeny formulas (1), deterministically selecting one of the two roots $\sqrt{C_i}$ according to the bit s_i.
3. Output the signing key $sk = \left((A_i)_{i\in[e-1]}, (C_i)_{i\in[e-1]}\right)$ as well as the verification key $vk = (A_e, C_e)$.

$\mathsf{Sign}(vk, m, sk; \mathsf{pp})$

1. Return $\sigma \leftarrow \Pi.\mathsf{Prove}(\mathsf{pp}, (vk, m), sk)$.

$\mathsf{Verify}(vk, m, \sigma; \mathsf{pp})$

1. Return $\Pi.\mathsf{Verify}(\mathsf{pp}, (vk, m), \sigma)$.

Figure 3: The algorithms of our signature scheme.

Figure 3 we give an algorithmic description of our signature scheme based on our NARG $\Pi = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ from Theorem 7.

Note that the augmented relation of our signature scheme is

$$\mathcal{R}^* = \left\{ (\mathbf{x}_m, \mathbf{w}) : \begin{array}{rl} \mathbf{x}_m = (A_0, C_0, A_e, C_e, m) & \\ \mathbf{w} = \left((A_i)_{i\in[e-1]}, (C_i)_{i\in[e-1]}\right) & \\ \forall i \in \{0, \dots e-1\} : & \\ -\frac{1}{12}A_i(A_{i+1} - A_i) & = C_i - \frac{1}{8}C_{i+1} \\ \wedge \frac{1}{36}(A_{i+1} - A_i)^2 & = C_i \end{array} \right\}. \tag{13}$$

For the historical parameters considered here, $\log(p) \approx e \approx 2\lambda$. See Section 1.2 for the effect of the recent attack [Wes26] on these choices. Note also that one needs to consider the extended statement $(\mathbf{x}, m)$ in the first Fiat–Shamir challenge only, with no effect on the rest of the NIZK. Further, we do not require a trusted setup – $\Pi.\mathsf{Setup}$ simply generates some public parameters.

The DeepSPRINT Signature Scheme. As previously mentioned, we instantiate our family of signature schemes using DeepFold[16] as the PCS.

Corollary 9. Instantiating our NARG 'SPRINT' (Theorem 7) using the DeepFold PCS and the relation with the statement extended by message m yields the EUF-CMA secure signature scheme DeepSPRINT given the hardness of the relation in Equation (6).

We compare DeepSPRINT to the NIST standardization candidate SQIsign [AAA+25, BCE+25] and the other state-of-the-art isogeny-based signature PRISM [BBC+25, BBC+26], which we both revisit in Appendix B, including discussions on their design and security. Here, we compare our signature to these other schemes in terms of security assumptions and performance.

Security assumptions. The EUF-CMA security of our signature scheme relies solely on the ℓ^e-IsogenyPath problem, on the wSE of the chosen NIZK and on the Fiat–Shamir heuristic. This is a

[16] DeepFold's security proof actually shows round-by-round binding, even if the statement is for regular binding.

rather conservative approach compared to other isogeny-based signatures, in particular PRISM and SQIsign.

The EUF-CMA security of PRISM, for instance, relies on oracles that produce isogenies with large prime factors. Assuming the random oracle model, this can be replaced with a new non-static "q-type" assumption with the parameter q being linear in the number of signing and hash queries. The most recent approach to prove security of SQIsign, on the other hand, is based on a new framework called *Fiat–Shamir with hints* [ABD+25], in which EUF-CMA is reduced, in the ROM, to the *endomorphism ring with hints* problem, which gives extra information to the simulator, but also to the adversary. Furthermore, this framework relies on another decisional assumption concerning the indistinguishability of hints. The same security reduction is carried out in the quantum setting in [ABR26], where the authors introduce the *Algebraic Isogeny Model* (AIM) and consider the QROM with adversaries restricted to the AIM. In this model, whenever an adversary outputs an elliptic curve, they must also provide an *explanation*, i.e., a description of an isogeny from that curve to a public (or previously seen) curve.

Thus, both PRISM and SQIsign rely on ad-hoc assumptions, which give attackers extra information when compared to more standard isogeny hardness assumptions. Whether either of these reduce to the foundational hardness assumptions of isogeny-based cryptography – such as the isogeny path problem or the endomorphism ring problem – is currently open. In fact, the only other signature schemes we are aware of that rely on "pure" isogeny assumptions are GPS [GPS17, §4] and group-action based schemes, such as CSI-FiSh [BKV19], although the latter rely on the isogeny path problem in a subgraph of the full isogeny graph. However, neither of these signatures is currently competitive in the NIST security parameter range.

Lastly, we note a side-channel advantage of our approach compared to other isogeny-based signatures: Our PIOP uses a constant number of operations, and thus, when compiled with a constant-time PCS, yields a constant-time signature. This is in contrast to PRISM and SQIsign, which cannot be made constant-time without significant effort.

Starting curve. It is standard for protocols not vulnerable to known endomorphism ring attacks to choose the starting curve E_0 such that $j(E_0) = 1728$. In our choice of curve model, this corresponds to $(A_0, C_0) = (0, 1) \in \mathbb{F}_{p^2}$. We remark that due to the nature of the radical isogeny formulae, we incur 1-bit of security loss, since an adversary may always compute the first j-invariant in the isogeny path as $j_1 = j(E_0/\langle(0,0)\rangle)$. However, after this step, due to the post-composition of a secret curve isomorphism, no further steps in the witness can be determined.

Key compression. Currently, the signing key is quite large at $2(e-1)\log p$ bits. Instead of storing the signing key as the entire chain of curve coefficients, one can also simply store the random path s sampled in the first step of the key generation algorithm and recover the coefficient chain during the signing algorithm. This reduces the signing key size from $2(e-1)\log p$ bits to e bits at only a small computational overhead for the signer.

If one requires more compact verification keys, the latter can also be halved by representing the final elliptic curve using its j-invariant. The curve coefficients can be provided as part of the signature, and the verifier can simply check if (A_e, C_e) define an elliptic curve of the correct j-invariant.

Efficiency and sizes. We compare the public key and signature sizes, as well as benchmarks of our prototype implementation, against original PRISM and SQIsign as well as the current state-of-the-art versions of both of these schemes; these benchmarks can be found in Table 4.[17] Our public keys are two coefficients over $\mathbb{F}_{p^2}$, but they can be compressed to a single j-invariant, making them similar in size to other designs. DeepFold requires a precomputation of its evaluation domain, depending only on the field used and the size of the witness. DeepSPRINT does this once for each parameter set, i.e. one doesn't need to rerun for new keys or messages. For this reason, the table contains a separate column with precomputation timings. A signer or verifier may either thus compute this result once at initialization, or load it from disk. The uncompressed sizes are 2MB, 6MB and 8MB respectively.

[17] Our parameters are chosen with respect to the attacks against ℓ^e-IsogenyPath from [AAF+26, BKM+24]. As a consequence, we can adopt the same parameters as those used in the current SQIsign or PRISM instantiations, which further allows a direct comparison of the benchmarks over the same fields. As observed in Section 1.2, these parameters do not take the recent attack of Wesolowski [Wes26] into account.

Table 4: Benchmarks for our signature scheme DeepSPRINT and the isogeny-based state-of-the-art signatures on a MacBook Air with Apple M4 CPU (10 cores, up to 4.4 GHz), 32 GB memory, macOS 26.4.1 (Tahoe), using the implementations without optimized finite field arithmetic. Public-key sizes are given in both their uncompressed (as used in the benchmarks) and compressed version (in brackets). DeepSPRINT signature sizes are broken down into the contributions from the NARG and the PCS. Unlike in Table 1 and Table 2, we separate DeepFold's one-time preprocessing times from signing/verification.

	Sec. level[4a]	Prime p	Size (bytes)		Time (ms)			
			Pub. key	Signature	Precomp	Keygen	Sign	Verify
PRISM[4b] [BBC+25]	NIST-I	$5 \cdot 2^{248} - 1$	66	189	-	16.9	22.7	4.23
	NIST-III	$65 \cdot 2^{376} - 1$	98	288	-	45.1	59.6	7.9
	NIST-V	$27 \cdot 2^{500} - 1$	130	388	-	124.3	112.3	–
salt-PRISM[4c] [BBC+26]	NIST-I	$5 \cdot 2^{248} - 1$	66	191	-	9.77	11.6	1.98
	NIST-III	$65 \cdot 2^{376} - 1$	98	272	-	32.2	48.4	4.7
	NIST-V	$27 \cdot 2^{500} - 1$	130	350	-	34.8	84.9	8.7
SQIsign (NIST)[4d] [AAA+25]	NIST-I	$5 \cdot 2^{248} - 1$	65	148	-	23.5	45.2	1.65
	NIST-III	$65 \cdot 2^{376} - 1$	97	224	-	61.9	151.8	4.43
	NIST-V	$27 \cdot 2^{500} - 1$	129	292	-	112.8	297.6	8.83
SQIsign (Qlapoti)[4e] [BCE+25]	NIST-I	$5 \cdot 2^{248} - 1$	65	148	-	9.77	17.89	1.65
	NIST-III	$65 \cdot 2^{376} - 1$	97	224	-	30.3	59.8	4.35
	NIST-V	$27 \cdot 2^{500} - 1$	129	292	-	35.2	85.7	8.8
DeepSPRINT[4f]	NIST-I	$5 \cdot 2^{248} - 1$	128 (64)	82056 (7048 + 75008)	1.25	2.93	17.9	0.44
	NIST-III	$65 \cdot 2^{376} - 1$	192 (96)	171016 (11688 + 159328)	5.97	16.52	71.19	1.01
	NIST-V	$27 \cdot 2^{500} - 1$	256 (128)	246024 (15560 + 230464)	9.37	33.84	90.57	1.36

[4a] This column refers to the schemes' 'historical' security levels. As explained in Section 1.2, these labels do not account for the recent attack of [Wes26].

[4b] C source code: https://github.com/KULeuven-COSIC/PRISM. Currently, the C code provided for PRISM benchmarking throws a segmentation fault for NIST-V on verification.

[4c] C source code: https://github.com/mariascrs/PRISM_v2.

[4d] C source code: https://github.com/SQIsign/the-sqisign

[4e] C source code: https://github.com/KULeuven-COSIC/Qlapoti

[4f] Rust source code: https://github.com/QuSAC/DeepSPRINT

In terms of computational efficiency, across all of the benchmarked parameter sets, DeepSPRINT achieves the fastest key generation among the compared schemes, improving key generation by 1.03–8.02× over prior PRISM and SQIsign implementations. Our signing time is roughly twice as fast compared to the specification of the SQIsign NIST submission [AAA+25], and on par with the variant using Qlapoti [BCE+25] or slightly less efficient if we also add the precomputation time. Similar results hold for the C implementation of PRISM [BBC+25] and the newer salted version of PRISM [BBC+26], which has undergone parameter improvements with respect to the initial version. Compared to the original PRISM implementation, DeepSPRINT achieves signing performance in a range of 0.79-1.19× that of PRISM and 1.06-1.54× that of salt-PRISM. Regarding verification, assuming the precomputation result is preloaded, we significantly outperform salt-PRISM by 4.50-6.40× and Qlapoti SQIsign by 3.75-6.47×. Even when we include precomputation in the signing times, we achieve on-par to slightly subpar performance, ranging from 0.85-1.49× salt-PRISM's performance and 1.02-1.22× Qlapoti SQIsign's. Our design has a much less competitive signature size than the PRISM or SQIsign variants. The dominant cost here is the underlying PCS DeepFold, which for all parameter sets accounts for over 90% of the size.

6.3 Optimizations and Improvements

We achieve major improvements across all metrics for our PoK, while for our signature the picture is mixed: signing times are on par with the state-of-the-art, verification times show significant improvement when using one-time precomputation, and we do not rely on ad-hoc assumptions, but this comes at the cost of significantly bigger sizes. We thus consider the latter a promising approach with potential for improvement. To illustrate this, we want to briefly mention some ways in which we expect that performance can be improved in the future, both in terms of timings and sizes. This includes merging our sumcheck argument into the PCS's, not opening the mask's contributions individually but opening the linear combination directly[18] and halving the field size by using a PCS over $\mathbb{F}_p$. We can also leverage the flexibility of our design, in particular changing the rate of code-based commitment schemes gives a natural trade-off between proving time and proof size. Alternatively, we can switch from DeepFold to any other PCS with a canonical simulator (thus achieving wSE). One example is WHIR [ACFY25], whose non-zero-knowledge version achieves proof sizes down to 36kB with similar performance to DeepFold. A zero-knowledge version has been proposed [CFW26], but as of yet is not fully implemented. Another option is the use of specialized PCS's for low-degree polynomials, in particular SmallWood [FR25], which achieves even smaller proof sizes for our parameter ranges.

Acknowledgements. We want to thank Akira Takahashi for answering our questions about the work in [KPT23], Giacomo Fenzi for making us aware of previous work in [CF24], and Steven Galbraith for helpful feedback on the manuscript. We also thank the PRISM team for informing us of an updated version of their code, which enabled us to revise our benchmarks. This work was supported by Research grant VIL53029 from VILLUM FONDEN, by the European Union's Horizon research and innovation program under grant agreement number 101190366 (PiQASO), and in part by a grant from the Ministry for Business Innovation and Employment, New Zealand. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the funding agencies. Neither the European Union nor the granting authorities can be held responsible for them.

References

[AAA+25] Marius A. Aardal, Gora Adj, Diego F. Aranha, Andrea Basso, Isaac Andrés Canales Martínez, Jorge Chávez-Saab, Maria Corte-Real Santos, Pierrick Dartois, Luca De Feo, Max Duparc, Jonathan Komada Eriksen, Tako Boris Fouotsa, Décio Luiz Gazzoni Filho, Basil Hess, David Kohel, Antonin Leroux, Patrick Longa, Luciano Maino, Michael Meyer, Kohei Nakagawa, Hiroshi Onuki, Lorenz Panny, Sikhar Patranabis, Christophe Petit, Giacomo Pope, Krijn Reijnders, Damien Robert, Francisco

[18] For instance by making non-blackbox use of the PCS's batching, or by using the more expressive language of IOPPs for constrained Reed-Solomon codes [ACFY25].

Rodríguez-Henríquez, Sina Schaeffler, and Benjamin Wesolowski. SQIsign. Technical report, National Institute of Standards and Technology, 2025. Version 2.0.1 from 2025-07-07.

[AAF+26] Marius A. Aardal, Diego F. Aranha, Yansong Feng, Yiming Gao, and Yanbin Pan. Better bounds for finding fixed-degree isogenies via Coppersmith's method. In Joan Daemen and Emmanuel Thomé, editors, *Advances in Cryptology – EUROCRYPT 2026*, pages 541–572, Cham, 2026. Springer Nature Switzerland.

[AB19] Shahla Atapoor and Karim Baghery. Simulation extractability in groth's zk-SNARK. Cryptology ePrint Archive, Report 2019/641, 2019.

[ABCP23] Shahla Atapoor, Karim Baghery, Daniele Cozzo, and Robi Pedersen. Practical robust DKG protocols for CSIDH. In Mehdi Tibouchi and Xiaofeng Wang, editors, *ACNS 2023, Part II*, volume 13906 of *LNCS*, pages 219–247. Springer, Cham, June 2023.

[ABD+25] Marius A. Aardal, Andrea Basso, Luca De Feo, Sikhar Patranabis, and Benjamin Wesolowski. A complete security proof of SQIsign. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VI*, volume 16005 of *LNCS*, pages 190–222. Springer, Cham, August 2025.

[ABR26] Marius A. Aardal, Andrea Basso, and Doreen Riepel. The algebraic isogeny model: A general model with applications to SQIsign and key exchanges. Cryptology ePrint Archive, Paper 2026/032, 2026. To appear at EUROCRYPT 2026.

[ACFY25] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. WHIR: Reed-solomon proximity testing with super-fast verification. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part IV*, volume 15604 of *LNCS*, pages 214–243. Springer, Cham, May 2025.

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 2087–2104. ACM Press, October / November 2017.

[ARS20] Behzad Abdolmaleki, Sebastian Ramacher, and Daniel Slamanig. Lift-and-shift: Obtaining simulation extractable subversion and updatable SNARKs generically. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1987–2005. ACM Press, November 2020.

[Bas24] Andrea Basso. A post-quantum round-optimal oblivious PRF from isogenies. In Claude Carlet, Kalikinkar Mandal, and Vincent Rijmen, editors, *SAC 2023*, volume 14201 of *LNCS*, pages 147–168. Springer, Cham, August 2024.

[BBB+25] Carsten Baum, Ward Beullens, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Christian Majenz, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger, Lawrence Roy, and Peter Scholl. Shorter, tighter, FAESTer: Optimizations and improved (QROM) analysis for VOLE-in-the-head signatures. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VI*, volume 16005 of *LNCS*, pages 124–156. Springer, Cham, August 2025.

[BBC+25] Andrea Basso, Giacomo Borin, Wouter Castryck, Maria Corte-Real Santos, Riccardo Invernizzi, Antonin Leroux, Luciano Maino, Frederik Vercauteren, and Benjamin Wesolowski. PRISM: Simple and compact identification and signatures from large prime degree isogenies. In Tibor Jager and Jiaxin Pan, editors, *PKC 2025, Part III*, volume 15676 of *LNCS*, pages 300–332. Springer, Cham, May 2025.

[BBC+26] Andrea Basso, Giacomo Borin, Wouter Castryck, Maria Corte-Real Santos, Riccardo Invernizzi, Antonin Leroux, Luciano Maino, Frederik Vercauteren, and Benjamin Wesolowski. PRISM with a pinch of salt: Simple, efficient and strongly unforgeable signatures from isogenies. *IACR Cryptol. ePrint Arch.*, 2026:443, 2026.

[BBD+22] Jeremy Booher, Ross Bowden, Javad Doliskani, Tako Boris Fouotsa, Steven D. Galbraith, Sabrina Kunzweiler, Simon-Philipp Merz, Christophe Petit, Benjamin Smith, Katherine E. Stange, Yan Bo Ti, Christelle Vincent, José Felipe Voloch, Charlotte Weitkämper, and Lukas Zobernig. Failing to hash into supersingular isogeny graphs. Cryptology ePrint Archive, Report 2022/518, 2022.

[BC24] Dan Boneh and Binyi Chen. LatticeFold: A lattice-based folding scheme and its applications to succinct proof systems. Cryptology ePrint Archive, Report 2024/257, 2024.

[BC25] Dan Boneh and Binyi Chen. LatticeFold+: Faster, simpler, shorter lattice-based folding for succinct proof systems. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VII*, volume 16006 of *LNCS*, pages 327–361. Springer, Cham, August 2025.

[BCC+23] 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. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 405–437. Springer, Cham, April 2023.

[BCE+25] Giacomo Borin, Maria Corte-Real Santos, Jonathan Komada Eriksen, Riccardo Invernizzi, Marzio Mula, Sina Schaeffler, and Frederik Vercauteren. Qlapoti: Simple and efficient translation of quaternion ideals to isogenies. In Goichiro Hanaoka and Bo-Yin Yang, editors, *ASIACRYPT 2025, Part IV*, volume 16248 of *LNCS*, pages 174–205. Springer, Singapore, December 2025.

[BCR+19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 103–128. Springer, Cham, May 2019.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60. Springer, Berlin, Heidelberg, October / November 2016.

[BD21] Jeffrey Burdges and Luca De Feo. Delay encryption. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 302–326. Springer, Cham, October 2021.

[BDD+24] Andrea Basso, Pierrick Dartois, Luca De Feo, Antonin Leroux, Luciano Maino, Giacomo Pope, Damien Robert, and Benjamin Wesolowski. SQIsign2D-West - the fast, the small, and the safer. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part III*, volume 15486 of *LNCS*, pages 339–370. Springer, Singapore, December 2024.

[BDGP23] Ward Beullens, Luca De Feo, Steven D. Galbraith, and Christophe Petit. Proving knowledge of isogenies: a survey. *DCC*, 91(11):3425–3456, 2023.

[BDPV21] Ward Beullens, Lucas Disson, Robi Pedersen, and Frederik Vercauteren. CSI-RAShi: Distributed key generation for CSIDH. In Jung Hee Cheon and Jean-Pierre Tillich, editors, *PQCrypto 2021*, pages 257–276. Springer, Cham, 2021.

[Bel21] Mihir Bellare. Lectures on NIZKs: A concrete security treatment. https://cseweb.ucsd.edu/~mihir/cse208-Wi20/main.pdf, 2021.

[BG90] Mihir Bellare and Shafi Goldwasser. New paradigms for digital signatures and message authentication based on non-interactive zero knowledge proofs. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 194–211. Springer, New York, August 1990.

[BGH19] Sean Bowe, Jack Grigg, and Daira Hopwood. Halo: Recursive proof composition without a trusted setup. Cryptology ePrint Archive, Report 2019/1021, 2019.

[BKM+24] Benjamin Benčina, Péter Kutas, Simon-Philipp Merz, Christophe Petit, Miha Stopar, and Charlotte Weitkämper. Improved algorithms for finding fixed-degree isogenies between supersingular elliptic curves. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part V*, volume 14924 of *LNCS*, pages 183–217. Springer, Cham, August 2024.

[BKSV21] Karim Baghery, Markulf Kohlweiss, Janno Siim, and Mikhail Volkhov. Another look at extraction and randomization of groth's zk-SNARK. In Nikita Borisov and Claudia Díaz, editors, *FC 2021, Part I*, volume 12674 of *LNCS*, pages 457–475. Springer, Berlin, Heidelberg, March 2021.

[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, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 227–247. Springer, Cham, December 2019.

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-Shamir: from practice to theory. In Moses Charikar and Edith Cohen, editors, *51st ACM STOC*, pages 1082–1090. ACM Press, June 2019.

[CD20] Wouter Castryck and Thomas Decru. CSIDH on the surface. In Jintai Ding and Jean-Pierre Tillich, editors, *PQCrypto 2020*, pages 111–129. Springer, Cham, 2020.

[CD23] Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 423–447. Springer, Cham, April 2023.

[CDG+17] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1825–1842. ACM Press, October / November 2017.

[CDV20] Wouter Castryck, Thomas Decru, and Frederik Vercauteren. Radical isogenies. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 493–519. Springer, Cham, December 2020.

[CF24] Alessandro Chiesa and Giacomo Fenzi. zkSNARKs in the ROM with unconditional UC-security. In Elette Boyle and Mohammad Mahmoody, editors, *TCC 2024, Part I*, volume 15364 of *LNCS*, pages 67–89. Springer, Cham, December 2024.

[CFR25] Matteo Campanelli, Antonio Faonio, and Luigi Russo. SNARKs for virtual machines are non-malleable. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part IV*, volume 15604 of *LNCS*, pages 153–183. Springer, Cham, May 2025.

[CFW26] Alessandro Chiesa, Giacomo Fenzi, and Guy Weissenberg. Zero-knowledge IOPPs for constrained interleaved codes. *IACR Cryptol. ePrint Arch.*, 2026:391, 2026.

[CGKY25] Alessandro Chiesa, Ziyi Guan, Christian Knabenhans, and Zihan Yu. On the Fiat–Shamir security of succinct arguments from functional commitments. Cryptology ePrint Archive, Paper 2025/902, 2025.

[CL24] Giulio Codogni and Guido Lido. Spectral theory of isogeny graphs. arXiV:2308.13913v3 [math.NT], July 2024.

[CLG09] Denis Xavier Charles, Kristin E. Lauter, and Eyal Z. Goren. Cryptographic hash functions from expander graphs. *Journal of Cryptology*, 22(1):93–113, January 2009.

[CLL23] Kelong Cong, Yi-Fu Lai, and Shai Levin. Efficient isogeny proofs using generic techniques. In Mehdi Tibouchi and Xiaofeng Wang, editors, *ACNS 2023, Part II*, volume 13906 of *LNCS*, pages 248–275. Springer, Cham, June 2023.

[CM22] Fabio Campos and Philipp Muth. On actively secure fine-grained access structures from isogeny assumptions. In Jung Hee Cheon and Thomas Johansson, editors, *PQCrypto 2022*, pages 375–398. Springer, Cham, September 2022.

[CS20] Daniele Cozzo and Nigel P. Smart. Sashimi: Cutting up CSI-FiSh secret keys to produce an actively secure distributed signing protocol. In Jintai Ding and Jean-Pierre Tillich, editors, *PQCrypto 2020*, pages 169–186. Springer, Cham, 2020.

[CSRT22] Jorge Chávez-Saab, Francisco Rodríguez-Henríquez, and Mehdi Tibouchi. Verifiable isogeny walks: Towards an isogeny-based postquantum VDF. In Riham AlTawy and Andreas Hülsing, editors, *SAC 2021*, volume 13203 of *LNCS*, pages 441–460. Springer, Cham, September / October 2022.

[CY24] Alessandro Chiesa and Eylon Yogev. Building cryptographic proofs from hash functions. https://github.com/hash-based-snargs-book/hash-based-snargs-book/blob/main/snargs-book.pdf, 2024.

[DDGZ22] Luca De Feo, Samuel Dobson, Steven D. Galbraith, and Lukas Zobernig. SIDH proof of knowledge. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part II*, volume 13792 of *LNCS*, pages 310–339. Springer, Cham, December 2022.

[DDO+01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 566–598. Springer, Berlin, Heidelberg, August 2001.

[DF24] Max Duparc and Tako Boris Fouotsa. SQIPrime: A dimension 2 variant of SQISignHD with non-smooth challenge isogenies. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part III*, volume 15486 of *LNCS*, pages 396–429. Springer, Singapore, December 2024.

[DG16] Christina Delfs and Steven D. Galbraith. Computing isogenies between supersingular elliptic curves over $\mathbb{F}_p$. *DCC*, 78(2):425–440, 2016.

[DG23] Quang Dao and Paul Grubbs. Spartan and bulletproofs are simulation-extractable (for free!). In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 531–562. Springer, Cham, April 2023.

[DHK+23] Julien Duman, Dominik Hartmann, Eike Kiltz, Sabrina Kunzweiler, Jonas Lehmann, and Doreen Riepel. Generic models for group actions. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part I*, volume 13940 of *LNCS*, pages 406–435. Springer, Cham, May 2023.

[dHKM+25] Thomas den Hollander, Sören Kleine, Marzio Mula, Daniel Slamanig, and Sebastian A. Spindler. More efficient isogeny proofs of knowledge via canonical modular polynomials. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part I*, volume 16000 of *LNCS*, pages 131–166. Springer, Cham, August 2025.

[DHLW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 613–631. Springer, Berlin, Heidelberg, December 2010.

[dHMSS26] Thomas den Hollander, Marzio Mula, Daniel Slamanig, and Sebastian A. Spindler. On the use of Atkin and Weber modular polynomials in isogeny proofs of knowledge. In Magali Bardet and Ruben Niederhagen, editors, *Post-Quantum Cryptography*, pages 39–73, Cham, 2026. Springer Nature Switzerland.

[DKL+20] Luca De Feo, David Kohel, Antonin Leroux, Christophe Petit, and Benjamin Wesolowski. SQISign: Compact post-quantum signatures from quaternions and isogenies. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part I*, volume 12491 of *LNCS*, pages 64–93. Springer, Cham, December 2020.

[DLRW24] Pierrick Dartois, Antonin Leroux, Damien Robert, and Benjamin Wesolowski. SQIsignHD: New dimensions in cryptography. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part I*, volume 14651 of *LNCS*, pages 3–32. Springer, Cham, May 2024.

[DMPS19] Luca De Feo, Simon Masson, Christophe Petit, and Antonio Sanso. Verifiable delay functions from supersingular isogenies and pairings. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 248–277. Springer, Cham, December 2019.

[DOT21] Cyprien Delpech de Saint Guilhem, Emmanuela Orsini, and Titouan Tanguy. Limbo: Efficient zero-knowledge MPCitH-based arguments. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 3022–3036. ACM Press, November 2021.

[FFK+23] Antonio Faonio, Dario Fiore, Markulf Kohlweiss, Luigi Russo, and Michal Zajac. From polynomial IOP and commitments to non-malleable zkSNARKs. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023, Part III*, volume 14371 of *LNCS*, pages 455–485. Springer, Cham, November / December 2023.

[FFR24] Antonio Faonio, Dario Fiore, and Luigi Russo. Real-world universal zkSNARKs are non-malleable. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 3138–3151. ACM Press, October 2024.

[FKMV12] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the Fiat-Shamir transform. In Steven D. Galbraith and Mridul Nandi, editors, *INDOCRYPT 2012*, volume 7668 of *LNCS*, pages 60–79. Springer, Berlin, Heidelberg, December 2012.

[FKNP24] Giacomo Fenzi, Christian Knabenhans, Ngoc Khanh Nguyen, and Duc Tu Pham. Lova: Lattice-based folding scheme from unstructured lattices. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part IV*, volume 15487 of *LNCS*, pages 303–326. Springer, Singapore, December 2024.

[FMP23] Tako Boris Fouotsa, Tomoki Moriya, and Christophe Petit. M-SIDH and MD-SIDH: Countering SIDH attacks by masking information. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 282–309. Springer, Cham, April 2023.

[FR25] Thibauld Feneuil and Matthieu Rivain. SmallWood: Hash-based polynomial commitments and zero-knowledge arguments for relatively small instances. Cryptology ePrint Archive, Report 2025/1085, 2025.

[GKK+22] Chaya Ganesh, Hamidreza Khoshakhlagh, Markulf Kohlweiss, Anca Nitulescu, and Michal Zajac. What makes Fiat-Shamir zkSNARKs (updatable SRS) simulation extractable? In Clemente Galdi and Stanislaw Jarecki, editors, *SCN 22*, volume 13409 of *LNCS*, pages 735–760. Springer, Cham, September 2022.

[GKO+23] Chaya Ganesh, Yashvanth Kondi, Claudio Orlandi, Mahak Pancholi, Akira Takahashi, and Daniel Tschudi. Witness-succinct universally-composable SNARKs. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 315–346. Springer, Cham, April 2023.

[GLH+25] Yanpei Guo, Xuanming Liu, Kexi Huang, Wenjie Qu, Tianyang Tao, and Jiaheng Zhang. DeepFold: Efficient multilinear polynomial commitment from reed-solomon code and its application to zero-knowledge proofs. In Lujo Bauer and Giancarlo Pellegrino, editors, *USENIX Security 2025*, pages 3497–3516. USENIX Association, August 2025.

[GM17] Jens Groth and Mary Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 581–612. Springer, Cham, August 2017.

[GOP+22] Chaya Ganesh, Claudio Orlandi, Mahak Pancholi, Akira Takahashi, and Daniel Tschudi. Fiat-Shamir bulletproofs are non-malleable (in the algebraic group model). In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 397–426. Springer, Cham, May / June 2022.

[GOP+25] Chaya Ganesh, Claudio Orlandi, Mahak Pancholi, Akira Takahashi, and Daniel Tschudi. Fiat-Shamir bulletproofs are non-malleable (in the random oracle model). *Journal of Cryptology*, 38(1):11, January 2025.

[GPS17] Steven D. Galbraith, Christophe Petit, and Javier Silva. Identification protocols and signature schemes based on supersingular isogeny problems. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 3–33. Springer, Cham, December 2017.

[Gro06] Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459. Springer, Berlin, Heidelberg, December 2006.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Berlin, Heidelberg, May 2016.

[ha21] halo2 authors. Design - The halo2 Book — zcash.github.io. https://zcash.github.io/halo2/design.html, 2021. [Accessed 06-05-2026].

[Hol19] Justin Holmgren. On round-by-round soundness and state restoration attacks. Cryptology ePrint Archive, Report 2019/1261, 2019.

[JD11] David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In Bo-Yin Yang, editor, *PQCrypto 2011*, pages 19–34. Springer, Berlin, Heidelberg, November / December 2011.

[KPT23] Markulf Kohlweiss, Mahak Pancholi, and Akira Takahashi. How to compile polynomial IOP into simulation-extractable SNARKs: A modular approach. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023, Part III*, volume 14371 of *LNCS*, pages 486–512. Springer, Cham, November / December 2023.

[KST22] Abhiram Kothapalli, Srinath Setty, and Ioanna Tzialla. Nova: Recursive zero-knowledge arguments from folding schemes. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 359–388. Springer, Cham, August 2022.

[KZM+15] Ahmed Kosba, Zhichao Zhao, Andrew Miller, Yi Qian, Hubert Chan, Charalampos Papamanthou, Rafael Pass, abhi shelat, and Elaine Shi. C∅c∅: A framework for building composable zero-knowledge proofs. Cryptology ePrint Archive, Report 2015/1093, 2015.

[LDK+22] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[Ler25] Antonin Leroux. Verifiable random function from the deuring correspondence and higher dimensional isogenies. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part VII*, volume 15607 of *LNCS*, pages 167–194. Springer, Cham, May 2025.

[LFKN92] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, October 1992.

[LGD21] Yi-Fu Lai, Steven D. Galbraith, and Cyprien Delpech de Saint Guilhem. Compact, efficient and UC-secure isogeny-based oblivious transfer. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 213–241. Springer, Cham, October 2021.

[LP25] Shai Levin and Robi Pedersen. Faster proofs and VRFs from isogenies. In Goichiro Hanaoka and Bo-Yin Yang, editors, *ASIACRYPT 2025, Part IV*, volume 16248 of *LNCS*, pages 307–340. Springer, Singapore, December 2025.

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000.

[MJ23] Youcef Mokrani and David Jao. Generating supersingular elliptic curves over $\mathbb{F}_p$ with unknown endomorphism ring. In Anupam Chattopadhyay, Shivam Bhasin, Stjepan Picek, and Chester Rebeiro, editors, *INDOCRYPT 2023, Part I*, volume 14459 of *LNCS*, pages 159–174. Springer, Cham, December 2023.

[MMP+23] Luciano Maino, Chloe Martindale, Lorenz Panny, Giacomo Pope, and Benjamin Wesolowski. A direct key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 448–471. Springer, Cham, April 2023.

[MMP24] Marzio Mula, Nadir Murru, and Federico Pintore. On random sampling of supersingular elliptic curves. *Annali di Matematica Pura ed Applicata*, 204:1293–1335, 2024.

[MNV24] Krystal Maughan, Joseph P. Near, and Christelle Vincent. Foldable, recursive proofs of isogeny computation with reduced time complexity. In Marek Osinski, Brian La Cour, and Lia Yeh, editors, *IEEE International Conference on Quantum Computing and Engineering, QCE 2024, Montreal, QC, Canada, September 15-20, 2024*, pages 1304–1308. IEEE, 2024.

[Mor23] Tomoki Moriya. IS-CUBE: An isogeny-based compact KEM using a boxed SIDH diagram. Cryptology ePrint Archive, Report 2023/1506, 2023.

[NO25] Kohei Nakagawa and Hiroshi Onuki. SQIsign2DPush: Faster signature scheme using 2-dimensional isogenies. Cryptology ePrint Archive, Paper 2025/897, 2025.

[NOC+24] Kohei Nakagawa, Hiroshi Onuki, Wouter Castryck, Mingjie Chen, Riccardo Invernizzi, Gioella Lorenzon, and Frederik Vercauteren. SQIsign2D-East: A new signature scheme using 2-dimensional isogenies. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part III*, volume 15486 of *LNCS*, pages 272–303. Springer, Singapore, December 2024.

[PFH+22] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[Piz90] Arnold K. Pizer. Ramanujan graphs and Hecke operators. *Bull. Am. Math. Soc., New Ser.*, 23(1):127–137, 1990.

[Rob23] Damien Robert. Breaking SIDH in polynomial time. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 472–503. Springer, Cham, April 2023.

[Rob24] Damien Robert. On the efficient representation of isogenies (a survey). Cryptology ePrint Archive, Report 2024/1071, 2024.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, October 1999.

[Set20] Srinath Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 704–737. Springer, Cham, August 2020.

[SSW20] Peter Schwabe, Douglas Stebila, and Thom Wiggers. Post-quantum TLS without handshake signatures. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1461–1480. ACM Press, November 2020.

[Ste22] Bruno Sterner. Commitment schemes from supersingular elliptic curve isogeny graphs. *Transactions on Mathematical Cryptology*, 1(2):40—-51, Mar. 2022.

[Vél71] Jacques Vélu. Isogénies entre courbes elliptiques. *Comptes-Rendus de l'Académie des Sciences*, 273:238–241, 1971.

[Wes26] Benjamin Wesolowski. The supersingular isogeny problem in time and memory $p^{1/3+o(1)}$. Cryptology ePrint Archive, Paper 2026/1486, 2026.

[XLZO25] Zheng Xu, Kaizhan Lin, Chang-An Zhao, and Yi Ouyang. SQIsign2D^2: New SQIsign2D variant by leveraging power smooth isogenies in dimension one. In Goichiro Hanaoka and Bo-Yin Yang, editors, *ASIACRYPT 2025, Part IV*, volume 16248 of *LNCS*, pages 341–371. Springer, Singapore, December 2025.

[XZZ+19] Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 733–764. Springer, Cham, August 2019.

A Isogeny Background

Let E, E' be two supersingular elliptic curves defined over a finite field $\mathbb{F}_q$ of characteristic p. An isogeny $\phi\colon E \to E'$ defined over $\mathbb{F}_q$ is a non-constant algebraic map that is also a group homomorphism from $E(\mathbb{F}_q)$ to $E'(\mathbb{F}_q)$, the groups of $\mathbb{F}_q$-rational points on E and E', respectively. Isogenies have degrees, which are defined as the degree of the rational map that defines them. Isogenies of degree 1 are also called *isomorphisms*. The kernel of an isogeny is a finite subgroup of $E(\overline{\mathbb{F}_q})$. Throughout this work, we will only consider *separable* isogenies, for which it holds that $\deg\phi = \#\ker\phi$.

For every isogeny $\phi\colon E \to E'$ there exists a dual isogeny $\hat{\phi}\colon E' \to E$, such that $\hat{\phi}\circ\phi = [\deg\phi]$, where $[m]$ is the scalar multiplication endomorphism (on E). It further holds that for any finite subgroup $G \subseteq E(\mathbb{F}_q)$, there exists a unique (up to equivalence) separable isogeny with kernel G. We are mainly interested in the case where G is cyclic, i.e., generated by a single point $P \in E(\mathbb{F}_q)$.

If the degree of an isogeny ϕ is composite, we can decompose the isogeny into a sequence of isogenies $\phi = \phi_e \circ \cdots \circ \phi_1$ of smaller (usually prime) degree, where it holds that $\deg\phi = \prod_{i=1}^{e}\deg\phi_i$. Due to this property, it is often sufficient to work with isogenies of prime degrees.

ℓ-isogeny graphs. For a given prime $\ell \neq p$, the *supersingular ℓ-isogeny graph* $\mathcal{G}_\ell(p)$ has nodes consisting of all isomorphism classes of supersingular elliptic curves over $\mathbb{F}_{p^2}$ (usually labeled by their j-invariants), and edges correspond to (equivalence classes of) the ℓ-isogenies connecting representatives. Due to the existence of dual isogenies, this graph is (almost[19]) undirected; further it is connected and $(\ell+1)$-regular, i.e., each vertex has $\ell+1$ outgoing edges. A particularly useful property of ℓ-isogeny graphs is the fact that they are Ramanujan [Piz90, CL24], which gives rise to cryptographic hardness assumptions, such as the ℓ^e-IsogenyPath problem.

A *path* is a sequence of ℓ-isogenies in the isogeny graph $\mathcal{G}_\ell(p)$:

$$E_0 \xrightarrow{\phi_1} E_1 \xrightarrow{\phi_2} \cdots \xrightarrow{\phi_{e-1}} E_{e-1} \xrightarrow{\phi_e} E_e = E.$$

Note that due to the decomposition properties of isogenies, any such path of length e defines an isogeny of degree ℓ^e. Throughout this work, we only look at the case $\ell = 2$.

[19] With exceptions occurring only at j-invariants 0 and 1728.

Computing paths of 2-isogenies. Vélu [Vél71] provides explicit formulas to compute isogenies of any degree from a given kernel, so a straightforward way to compute a chain of 2-isogenies would be to generate a point of order 2 on the starting curve E_0, use Vélu's formulas to compute the 2-isogeny $\phi_0 \colon E_0 \to E_1$, and then sample another point of order 2 on the codomain E_1 and so on. To avoid backtracking of the walk, e.g. $\phi_i \colon E_i \to E_{i-1}$, we have to make sure that the point sampled on E_i does not generate the dual of ϕ_i. A simple way to do this has been proposed in [CLG09]: Let $P_i \in E_i[2]$ be the generator of $\ker \phi_i$ and let $Q_i \in E_i[2] \setminus \ker \phi_i$, then since $\hat{\phi}_i \circ \phi_i(Q_i) = [2]Q_i = O$, it must be that $\phi_i(Q_i) \in \ker \hat{\phi}_i$. So, by computing $\phi_i(Q_i)$, we know what $\ker \hat{\phi}_i$ is and can sample a different 2-torsion point on E_{i+1}, which then generates a 2-isogeny that is not backtracking. A faster way of computing chains of 2-isogenies without backtracking has been proposed in [JD11] in which rational 2^e-torsion is immediately made accessible on E_0. Then, any point $P_0 \in E_0[2^e]$ defines a non-backtracking walk in the 2-isogeny graph as follows. At every step, compute $\left[\frac{\#E(\mathbb{F}_q)}{2^i}\right] P_i$, and use this point of order 2 to generate the kernel of the isogeny ϕ_i, then push through $P_{i+1} = \phi_i(P_i)$. Unfortunately, scalar multiplication and pushing through points are expensive operations and increase the total cost of the isogeny computations considerably. [JD11] shows that we can cleverly optimize the trade-off between scalar multiplication and pushing through points, but does not get rid of these operations.

B Overview of SQIsign and PRISM

SQIsign. A candidate in NIST's additional call for post-quantum digital signatures is SQIsign [AAA+25], whose original design was introduced in [DKL+20]. At a high level, SQIsign is obtained by applying the Fiat–Shamir transform of the following Sigma protocol: to prove knowledge of an isogeny $\varphi_{\mathrm{sk}} \colon E_0 \to E_{\mathrm{pk}}$, the prover computes a random (secret) isogeny $\varphi_{\mathrm{com}} \colon E_0 \to E_{\mathrm{com}}$ and commits to it by sending E_{com} to the verifier, who replies with a challenge isogeny $\varphi_{\mathrm{chl}} \colon E_{\mathrm{pk}} \to E_{\mathrm{chl}}$. The prover then computes and outputs a response isogeny φ_{rsp}, which is accepted by the verifier if and only if φ_{rsp} is an isogeny $E_{\mathrm{com}} \to E_{\mathrm{chl}}$ with a suitable degree. The resulting isogeny diagram is sketched in Figure 4.

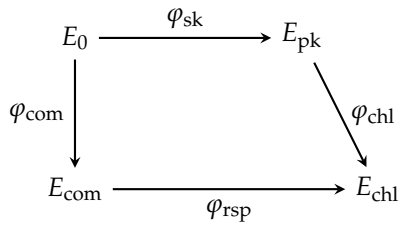


Figure 4: The identification protocol underlying SQIsign.

An analogous identification scheme underlies several successors building on SQIsign, namely SQIsignHD [DLRW24], SQIsign2D-West [BDD+24], SQIsign2D-East [NOC+24], SQIPrime [DF24], SQIsign2DPush [NO25], and SQIsign2D^2 [XLZO25]. The main difference is that, while φ_{rsp} has powersmooth degree in the original SQIsign, these new protocols introduced the use of higher-dimensional representations of isogenies, which enables greater flexibility in the choice of the degree of φ_{rsp}, together with more compact keys/signatures and faster timings. Notably, the higher-dimensional techniques from SQIsign2D-West have now been also implemented in the most recent version of SQIsign submitted to the NIST call [AAA+25].

In terms of security, the most difficult task is proving zero-knowledge of the underlying Sigma protocol, i.e., simulating valid transcripts that are statistically (or, at least, computationally) indistinguishable from valid executions of the protocol. Unfortunately, this task is difficult both when there are smoothness restrictions on the degree of φ_{rsp} and when such restrictions are relaxed by leveraging higher-dimensional techniques. In the first case, the difficulty consists in uniformly sampling from a certain subset of isogenies with powersmooth degrees, while in the second case the problem is that higher-dimensional representations cannot be simulated without knowing the endomorphism ring of the public key curve – which is equivalent to knowing the secret key. This issue was initially addressed by requiring that the simulator interacts with *ad hoc* oracles that produce

higher-dimensional representations of random isogenies. As [ABD+25] pointed out, however, it is not clear whether the Fiat–Shamir transform is actually applicable when relying on the interaction with such oracles. To circumvent this issue, [ABD+25] introduces a new framework called *Fiat–Shamir with hints*, where such oracles are replaced by non-interactive hints that can be used by the simulation to produce these higher-dimensional isogenies. The authors prove that this variant of Fiat–Shamir makes the signature scheme EUF-CMA secure in the ROM, basing the security on the underlying Sigma protocol and two new hardness assumptions, the computational *(one) endomorphism problem with hints* assumption and a decisional *hint indistinguishability* assumption. Currently, no reduction from these assumptions to more standard assumptions exists. The EUF-CMA security of SQIsign, under the same assumptions, has also been proven in the QROM in [ABR26]. The proof, however, only considers adversaries restricted to the so-called *Algebraic Isogeny Model* (AIM), where every elliptic curve output by the adversary is required to also include an *explanation*, i.e., a description of an isogeny from that curve to a public (or previously seen) curve. This model is an isogeny analogue of the Algebraic Group Model and is shown to generalize the Algebraic Group Action Model [DHK+23], which had previously been used to model the security of isogeny-based constructions in the oriented setting.

PRISM. The signature scheme PRISM [BBC+25] applies the hash-and-sign paradigm to the following 2-move identification protocol: to prove knowledge of an isogeny $\varphi_{\mathrm{sk}} \colon E_0 \to E_{\mathrm{pk}}$, the prover is challenged by the verifier with a random a-bit prime q. The prover then computes and outputs a response isogeny φ_{rsp} with domain E_{pk} and degree $q(2^a - q)$. Choosing this degree allows to efficiently represent φ_{rsp} in a two-dimensional representation. The security of the identification protocol relies on a one-more type assumption that gives the adversary access to an oracle called a SPEDIO (special degree isogeny oracle), which takes as input an a-bit prime q and outputs an efficient 2D-representation of a uniformly random isogeny of degree $q(2^a - q)$. The goal of the adversary is to output a 2D-isogeny of degree $q'(2^a - q')$, where q has not been queried to the SPEDIO before.

In order to obtain a signature scheme from this protocol, the authors define a hash function $\mathsf{H}_{\mathrm{prime}}$ which has as output space the primes of exactly a bits. Using this, any input message is hashed together with the public key to obtain a prime q that defines the response degree. In the more recent, *salted* version of PRISM [BBC+26], the signer also adds a random salt to the message before hashing and publishes this salt along with the signature. This simple modification allows to reduce the parameter a, resulting in smaller signatures, as well as a speedup in both the signing and verification times.

The EUF-CMA security of the PRISM signature is proven in two different flavors, depending on how this hash function is modeled.

- The first proof is in the standard model and EUF-CMA relies on the same one-more type assumption as the identification protocol plus the assumption that the hash function $\mathsf{H}_{\mathrm{prime}}$ is collision-resistant.
- In a second version, the hash function is modeled as a random oracle, which allows the security to rely on a strictly weaker assumption than the identification protocol. In this version, the adversary has no direct access to a SPEDIO, but is simply given polynomially many signatures of random degrees $q_i(2^a - q_i)$ and is asked to provide a 2D-representation of degree $q(2^a - q)$ for a random given prime q not among the collection $\{q_i\}$.

Given its simpler design, PRISM appears to be more flexible than SQIsign – i.e., easier to use as a building block for more advanced functionalities. As PRISM is a recent proposal, this direction has not yet been thoroughly explored, but we expect that it will be pursued in the near future.

C Computation of j-invariants

If instead of $\mathbf{x} = (A_0, C_0, A_e, C_e)$, we want the public statement to just consist of the j-invariants of the start and end curves, we can additionally compute these in the polynomial P. We start with the relation

$$j_i = 256 \frac{(3C_i - A_i^2)^3}{C_i^2(4C_i - A_i^2)}.$$

Since this is a rational function with a polynomial of degree 6 as its numerator, embedding this directly in the sumcheck argument would increase the degree of P. Thus the prover would be required to send a larger polynomial in the sumcheck argument, which also means more mask elements are needed to mask this polynomial. We would like to change this into a polynomial by multiplying both sides by $C_i^2(4C_i - A_i^2)$, but here we need to be careful to not add solutions, i.e. we require that $C_i^2(4C_i - A_i^2) \neq 0$.

We will now present a method that increases the proof size by two field elements, corresponding exactly to the (masked) openings of A_i and C_i. Additionally, it makes use of five unused indices of the mask, for which we define $D_c(x) = \tilde{B}(1, x, \overrightarrow{6n+3+c})$ for $c \in [5]$. Furthermore, for ease of exposition we let $A_0(x) = \tilde{B}(x, 0, \vec{0})$ and $C_0(x) = \tilde{B}(x, 1, \vec{0})$. We then impose the following constraints

$$
\begin{aligned}
0 &= (1-x)(D_1(x) - A_0^2(x)) \\
0 &= (1-x)(D_2(x) - C_0^2(x)) \\
0 &= (1-x)(D_3(x) - D_1^2(x)) \\
0 &= (1-x)(j_0 D_2(x)(4C_0(x) - D_1(x))) \\
&\quad + 256(D_1(x)D_3(x) - 3C_0(x)D_2(x) - 9C_0(x)D_3(x) + 27D_2(x)D_1(x)) \\
0 &= (1-x)(C_0(x)D_4(x) - 1) \\
0 &= (1-x)(D_5(x)(4C_0(x) - D_1(x)) - 1),
\end{aligned}
$$

letting the prover select $D_c(x)$, $c \in [5]$, to satisfy these constraints. This is integrated into the PIOP as follows: after $\mathcal{P}$ has committed to $\tilde{B}$, we ask the $\mathcal{V}$ to supply a random $h_2 \in \mathbb{F}$, using

$$
\begin{aligned}
&P(h_0, h_1, \vec{k}, \vec{i}, x, \vec{j}) \\
&+ h_2(1-x)(D_1(x) - A_0^2(x)) \\
&+ h_2^2(1-x)(D_2(x) - C_0^2(x)) \\
&+ h_2^3(1-x)(D_3(x) - D_1^2(x)) \\
&+ h_2^4(1-x)(j_0 D_2(x)(4C_0(x) - D_1(x))) \\
&\quad + 256(D_1(x)D_3(x) - 3C_0(x)D_2(x) - 9C_0(x)D_3(x) + 27D_2(x)D_1(x)) \\
&+ h_2^5(1-x)(C_0(x)D_4(x) - 1) \\
&+ h_2^6(1-x)(D_5(x)(4C_0(x) - D_1(x)) - 1)
\end{aligned}
$$

as the new sumcheck polynomial. This does not increase the degree of the polynomial. Indeed, it only adds to the prover's work in the round where x is set, where $\mathcal{P}$ needs to compute two evaluations of this polynomial ($x = 0$ is known to be 0 and $x = 1$ can be derived from the sumcheck itself).

Efficiency. Since the degree of the polynomial is not increased, the verifier's work during sumcheck does not change. The prover needs to evaluate this additional polynomial twice, for $x \in \{2, 3\}$ in the sumcheck round where x is the free variable; note that the additional polynomial is still 0 at $x \in \{0, 1\}$. Evaluating the polynomial requires 23 finite field multiplications, plus 7 for the evaluations of $A_0, C_0, \{D_c\}_{c\in[5]}$. Likewise, the verifier needs to evaluate this polynomial once, using the evaluations $(A_0(r_x), C_0(r_x))$ and $\{D_c(x)\}_{c\in[5]}$.

We implement the j-invariant version of DeepSPRINT in our framework, with a performance overhead of 1%-2.4% across all benchmarks. The additional openings of $\tilde{B}$ give a proof overhead of 0.9kB, 1.3kB and 1.8kB for the three benchmarked parameter sets, respectively.

Round by round knowledge soundness. The terms $(1-x)$ ensure that the sumcheck argument tests for the situation where $x = 0$. We thus have $D_1(0) = A_0^2$, $D_2(0) = C_0^2$, $D_3(0) = A_0^4$. The fourth constraint tests that $C_0^2(4C_0 - A_0^2)j_0 = 256(3C_0 - A_0)^3$, and the last two constraints test that the denominator is not zero. Since h_2 is picked uniformly at random, the probability of picking a root even when the constraints are not satisfied is $6/|\mathbb{F}|$, which is therefore added to the first round's knowledge soundness error. The other rounds' errors are unchanged since the sumcheck argument is over a polynomial of the same degree.

Zero-knowledge. No additional leakage is added to the sumcheck phase. We should therefore make sure that the additional evaluations are distributed uniformly at random. The analysis of the standard protocol uses the indices $\tilde{B}(1, y, \vec{c'})$ with $y \in \{0,1\}$ and $c' \in \{0, \ldots, 6n+3\}$ as well $\tilde{B}(1, \vec{y'}, \vec{1})$ with $\vec{y'} \in \{0,1\}^2$. Since we want to mask the openings of A_0, A_e, C_0 and C_e, we want to shift the openings of the mask to $\tilde{B}(1, y, \vec{c'})$ with $c' \in [6n+4]$ and the analysis of the random openings to $\tilde{B}(1, \vec{y'}, \overrightarrow{(2^{n-1}-2)})$ for $\vec{y'} \in \{0,1\}^2$. This does not otherwise affect the original analysis of zero-knowledge. This masks the openings $A_0(r_x)$, $C_0(r_x)$, $A_e(r_x)$ and $C_e(r_x)$. The indices $\tilde{B}(1, x, \vec{c})$, $x \in \{0,1\}$, are unused as long as $6n+4 < c < 2^{n-1}-2$. In particular, the indices $6n+4+[10]$ are unused, provided $n \geq 7$. Since these entries are distributed uniformly at random when $x = 1$, the revealed entries at r_x are also all distributed uniformly at random as desired.

D Arbitrary Path Lengths through $\tilde{Q}$-Puncturing

To enable the prover to provide a valid walk only up to the desired length, allowing any value beyond it (such as 0), we would like $\tilde{Q}(\vec{i}, \vec{j}, \vec{l})$ to be zero at every index $i > e'$, with $e' \in [e]$. We can express this as a polynomial factor in just $\vec{l}$, keeping $\tilde{Q}$ multilinear. Indeed, let $\mathsf{zeros}(e')$ be the indices v such that $\vec{e'}_v = 0$. We can then multiply each term in $\tilde{Q}$ by the factor

$$
\begin{aligned}
f_{e'}(\vec{l}) &= 1 - \sum_{v \in \mathsf{zeros}(e')} l_v \, \tilde{\mathrm{eq}}(\overrightarrow{(e'_{v+1\ldots n)}}, \overrightarrow{(l_{v+1\ldots n)}}) \\
&= 1 - \left[\prod_{v=\mathsf{zeros}(e')_k+1}^{n} \tilde{\mathrm{eq}}(e'_v, l_v) \right] \\
&\quad \cdot \Big(l_{\mathsf{zeros}(e')_k} + \left[\prod_{v=\mathsf{zeros}(e')_{k-1}+1}^{\mathsf{zeros}(e')_k} \tilde{\mathrm{eq}}(e'_v, l_v) \right] \\
&\qquad \cdot \Big(l_{\mathsf{zeros}(e')_{k-1}} + \left[\prod_{v=\mathsf{zeros}(e')_{k-2}+1}^{\mathsf{zeros}(e')_{k-1}} \tilde{\mathrm{eq}}(e'_v, l_v) \right] \\
&\qquad \ldots \cdot l_{\mathsf{zeros}(e')_1} \Big) \cdots \Big)
\end{aligned}
$$

to subtract such higher contributions from the sum. This formula follows from the observation that the integers greater than e' are precisely those for which there is a location v where $l_v = 1$ and $e'_v = 0$, and are equal to e' for the more significant digits $w > v$. This may be seen more easily as follows: a number not equal to e' must differ in a digit, and the most significant digit where this is the case must change a 0 to a 1 instead of vice versa. For example, any number higher than `0b10010` has the form `0b11...`, `0b101...` or `0b10011`. Instead of a factor $\tilde{Q}$, we will now use the factor

$$
\tilde{Q}_{e'}(\vec{i}, \vec{j}, \vec{k}) = \sum_{l \in \{0,1\}^n \setminus \{1\ldots1\}} f_{e'}(\vec{l}) \, \tilde{\mathrm{eq}}(\overrightarrow{(l+1)}, \vec{j}) \, \tilde{\mathrm{eq}}(\vec{l}, \vec{i}) \, \tilde{\mathrm{eq}}(\vec{l}, \vec{k}).
$$

Like before, we have to ensure that this adapted $\tilde{Q}_{e'}$ is still efficiently computable over $\mathbb{F}^{3n}$:

Lemma 10. The value of $\tilde{Q}_{e'}(\vec{i}, \vec{j}, \vec{k})$ with $\vec{i}, \vec{j}, \vec{k} \in \mathbb{F}^n$ is computable in $8n-6$ field multiplications and $13n-7$ field additions.

Proof. We adopt a recursive approach, like in Lemma 3. First, we define

$$\begin{aligned}
g_{e',m}(\vec{l}) &= \left[\prod_{v=\mathsf{zeros}(e')_k+1}^{m} \tilde{\mathrm{eq}}(e'_v, l_v)\right] \\
&\quad \cdot \left(l_{\mathsf{zeros}(e')_k} + \left[\prod_{v=\mathsf{zeros}(e')_{k-1}+1}^{\mathsf{zeros}(e')_k} \tilde{\mathrm{eq}}(e'_v, l_v)\right]\right. \\
&\quad \left. \cdot \left(l_{\mathsf{zeros}(e')_{k-1}} + \left[\prod_{v=\mathsf{zeros}(e')_{k-2}+1}^{\mathsf{zeros}(e')_{k-1}} \tilde{\mathrm{eq}}(e'_v, l_v)\right] \dots \cdot l_{\mathsf{zeros}(e')_1}\right) \dots\right) \\
&= \begin{cases} \tilde{\mathrm{eq}}(e'_m, l_m) \cdot g_{e',m-1}(\overrightarrow{l_{m-1\dots1}}) & \text{if } \vec{e'}_m = 1 \\ l_m + \tilde{\mathrm{eq}}(e'_m, l_m) \cdot g_{e',m-1}(\overrightarrow{l_{m-1\dots1}}) & \text{if } \vec{e'}_m = 0 \end{cases} \\
&= (1 - e'_m) l_m + \tilde{\mathrm{eq}}(e'_m, l_m) g_{e',m-1}(\overrightarrow{l_{m-1\dots1}})
\end{aligned}$$

such that $f_{e'}(\vec{l}) = 1 - g_{e',n}(\vec{l})$, i.e. $g_{e',m}(\vec{l})$ are the consecutive partial products of $f_{e'}(\vec{l})$. The base case for the recursion relation is $g_{e',0}(\vec{l}) = 0$, since we don't want to exclude any terms *a priori*.

We recall that

$$\tilde{Q}(\vec{i}, \vec{j}, \vec{k}) = \sum_{l \in \{0,1\}^n \setminus \{1\dots1\}} \tilde{\mathrm{eq}}(\overrightarrow{(l+1)}, \vec{j})\, \tilde{\mathrm{eq}}(\vec{l}, \vec{i})\, \tilde{\mathrm{eq}}(\vec{l}, \vec{k}).$$

Now, let

$$\begin{aligned}
J_m(\vec{i}, \vec{j}, \vec{k}) &= \sum_{\vec{l} \in \{0,1\}^m \setminus \{1\dots1\}} g_{e',m}(\vec{l})\, \tilde{\mathrm{eq}}(\overrightarrow{(l+1)}, \vec{j})\, \tilde{\mathrm{eq}}(\vec{l}, \vec{i})\, \tilde{\mathrm{eq}}(\vec{l}, \vec{k}) \\
&= (1 - e'_m) g_{e',m-1}(\vec{1})(1 - i_m)(1 - k_m) j_m \prod_{w<m} i_w k_w (1 - j_w) \\
&\quad + \sum_{l_m \in \{0,1\}} \sum_{\vec{l}_{m-1\dots1} \in \{0,1\}^{m-1} \setminus \{1\dots1\}} \left((1 - e'_m) l_m + \tilde{\mathrm{eq}}(e'_m, l_m) g_{e',m-1}(\vec{l}_{m-1\dots1})\right) \\
&\qquad \cdot \tilde{\mathrm{eq}}(l_m, i_m)\, \tilde{\mathrm{eq}}(l_m, k_m)\, \tilde{\mathrm{eq}}(\overrightarrow{(l+1)}_m, j_m) \\
&\qquad \cdot \tilde{\mathrm{eq}}(\vec{l}_{m-1\dots1}, \vec{i}_{m-1\dots1})\, \tilde{\mathrm{eq}}(\vec{l}_{m-1\dots1}, \vec{i}_{m-1\dots1})\, \tilde{\mathrm{eq}}(\overrightarrow{(l+1)}_{m-1\dots1}, \vec{i}_{m-1\dots1}) \\
&= (1 - e'_m) g_{e',m-1}(\vec{1})(1 - i_m)(1 - k_m) j_m \prod_{w<m} i_w k_w (1 - j_w) \\
&\quad + \sum_{\vec{l}_{m-1\dots1} \in \{0,1\}^{m-1} \setminus \{1\dots1\}} (1 - e'_m) g_{e',m-1}(\vec{l}_{m-1\dots1})(1 - i_m)(1 - k_m)(1 - j_m) \\
&\qquad \cdot \tilde{\mathrm{eq}}(\vec{l}_{m-1\dots1}, \vec{i}_{m-1\dots1})\, \tilde{\mathrm{eq}}(\vec{l}_{m-1\dots1}, \vec{i}_{m-1\dots1})\, \tilde{\mathrm{eq}}(\overrightarrow{(l+1)}_{m-1\dots1}, \vec{i}_{m-1\dots1}) \\
&\quad + \sum_{\vec{l}_{m-1\dots1} \in \{0,1\}^{m-1} \setminus \{1\dots1\}} \left((1 - e'_m) + e'_m g_{e',m-1}(\vec{l}_{m-1\dots1})\right) i_m k_m j_m \\
&\qquad \cdot \tilde{\mathrm{eq}}(\vec{l}_{m-1\dots1}, \vec{i}_{m-1\dots1})\, \tilde{\mathrm{eq}}(\vec{l}_{m-1\dots1}, \vec{i}_{m-1\dots1})\, \tilde{\mathrm{eq}}(\overrightarrow{(l+1)}_{m-1\dots1}, \vec{i}_{m-1\dots1}) \\
&= (1 - e'_m)(1 - i_m)(1 - k_m) j_m \cdot g_{e',m-1}(\vec{1}) \prod_{w<m} i_w k_w (1 - j_w) \\
&\quad + e'_m i_m j_m k_m \cdot \tilde{Q}_{m-1}(\vec{i}_{m-1\dots1}, \vec{j}_{m-1\dots1}, \vec{k}_{m-1\dots1}) \\
&\quad + \left[e'_m i_m j_m k_m + (1 - e'_m)(1 - i_m)(1 - j_m)(1 - k_m)\right] \\
&\qquad \cdot J_{m-1}(\vec{i}_{m-1\dots1}, \vec{j}_{m-1\dots1}, \vec{k}_{m-1\dots1}).
\end{aligned}$$

Here we have decomposed the sum over $\vec{l}$ into three cases: either $\vec{l} = 01\dots1$, or it starts with a 0 or a 1 with at least one 0 in $\vec{l}_{m-1\dots1}$. The base case is $J_1(\vec{i}, \vec{j}, \vec{k}) = 0$. From this formula, an iterative algorithm follows: for increasing m, we compute $g_{e',m-1}(\vec{1})$, $\prod_{w<m} i_w k_w (1 - j_w)$, $\tilde{Q}_m$ and finally $\tilde{J}_m$. When $m = n$ we can compute

$$\tilde{Q}_{e'}(\vec{i}, \vec{j}, \vec{k}) = \tilde{Q}(\vec{i}, \vec{j}, \vec{k}) - \tilde{J}_m(\vec{i}, \vec{j}, \vec{k}).$$

Note that $g_{e',m}(\vec{l}) = 1 - \tilde{\text{eq}}(\vec{e'}_{m\ldots 1}, \vec{1})$. Using the same strategy as in Lemma 3, the number of field operations follows. □