# Revisiting Fiat-Shamir NIZKPs and Signatures from Seed-Friendly Sigma Protocols

Andrea Flamini[1], Federico Pintore[2], and Edoardo Signorini[3]

[1] Hasso Plattner Institute, University of Potsdam, Potsdam, Germany
[2] Università di Trento, Trento, Italy
[3] European Commission, Joint Research Centre (JRC), Ispra, Italy

**Abstract.** Seed-friendly sigma protocols have a small challenge space, and when compiled into NIZKP, they require parallel repetition to reduce extraction error. Fiat–Shamir proofs obtained from seed-friendly Sigma protocols can be made more compact with fixed-weight parallel repetitions, and seed-tree compression. So far, these have been presented as compression techniques for the proof size, but standard analysis of the extraction property of NIZKP still relies on the Forking Lemma and rewinding.

In this work, we show that not only fixed-weight parallel repetition and seed-tree compression improve the compactness of the proofs, but also improve their security. Interestingly, when the prover randomness is generated from a salted seed tree, interacting with a random oracle, the standard Fiat–Shamir compilation is *straight-line extractable* in the classical random-oracle model, without any additional proof field or proof-of-work.

We further show that the Fiat-Shamir NIZKP from seed-friendly Sigma protocols satisfies multi-proof zero knowledge, and that its straight-line extractability does not require the underlying seed-friendly Sigma protocol to have unique responses.

## 1 Introduction

*NIZKPs.* Non-interactive zero-knowledge proofs of knowledge (in the following simply NIZKPs) allow a prover to prove knowledge of a secret without revealing it or interacting with the verifier. Knowledge is formalized through an extractor that can recover the secret from a successful prover. A standard way to build NIZKPs is to compile Sigma protocols using general-purpose transforms [30,52,50,57,31], with the Fiat–Shamir transform being the most common. Its popularity stems from its compactness and computational efficiency: when the underlying challenge space is sufficiently large, a proof consists of a single Sigma-protocol transcript [54,14,48].

*Seed-Friendly Sigma Protocols and the Fixed-Weight Framework.* Seed-friendly Sigma protocols [34] have small challenge spaces, are first-message retrievable, and admit a special challenge $\tilde{\mathsf{ch}}$ whose response can be computed from the

prover's random tape without a witness. Deriving this tape from a short seed allows the special response to be encoded compactly. To limit proof-size growth under parallel repetition, the *fixed-weight framework* [34, Section 2] repeats the underlying protocol $L$ times and restricts challenge vectors to weight $\rho$[4]. Thus, only $\rho$ repetitions require full responses, while the remaining $L - \rho$ use compact special responses. In *seeded fixed-weight parallel repetitions*, seed trees further reduce the number of seeds revealed to reconstruct these compact responses [13].

These techniques are particularly useful for post-quantum signatures from seed-friendly Sigma protocols, including code-based schemes (LESS [12], MEDS [20]), isogeny-based schemes (CSI-FiSh [11], CSI-SharK [3], PEGASIS [23]), lattice-based schemes [27], and schemes based on trilinear forms (ALTEQ [56]).

*Straight-Line Extractable NIZKPs.* Standard Fiat–Shamir extractors use rewindable access to the prover [52]. Straight-line extractability [31] instead requires recovery of the secret from only the successful prover's random-oracle queries and its generated proof. In the random oracle model, this property addresses three limitations of rewinding-based extraction. First, rewinding is generally incompatible with composable security frameworks such as the UC model [15], motivating straight-line extractable NIZKPs for protocols proven secure in these settings [47,19,18]. Applications include threshold signatures [46,24,25] and anonymous credential systems [39]. Straight-line extractability also arises in security proofs for blind signatures [44,43]and their applications to anonymous tokens [32], and anonymous credentials supporting blind issuance of attributes [17,33]. Second, in multi-party protocols, reductions may need to rewind several corrupted actors controlled by an adversary. Nested rewinding can require super-polynomial reduction time, obstructing proofs of concurrent security [55,51]. Third, standard forking-lemma analyses of Fiat–Shamir NIZKPs and signatures in the classical random oracle model typically incur a quadratic tightness loss [8,52]. This loss also affects the usual analysis of fixed-weight repetitions of seed-friendly Sigma protocols, and can leave the resulting primitives with fewer provable bits of security than claimed [21]. Straight-line extraction offers a route to NIZKPs and signatures with tight reductions to the underlying assumption [9,35].

*Existing Solutions.* One route to straight-line extraction strengthens the model used in the analysis. For group-based proofs, the algebraic group model (AGM) supplies representations of the adversary's group elements, making straight-line extraction easier to achieve. To give an example, Orrù's analysis of keyed-verification anonymous credentials [49] uses the AGM to establish straight-line extractability for some Fiat–Shamir NIZKPs.

Within the random oracle model, Pass [50] and Unruh [57] proposed general-purpose transforms from Sigma protocols to straight-line extractable NIZKPs; another approach, which produces more compact proofs than these, is the Fischlin transform and its variants [31,19,45].

[4] In this work, we identify the challenge space $\mathsf{ChSpace}$ of a seed-friendly Sigma protocol with $\{0, 1, \ldots, |\mathsf{ChSpace}| - 1\}$, and $\tilde{\mathsf{ch}}$ with 0. The weight of a challenge vector is the number of components $\mathsf{ch}_i \neq \tilde{\mathsf{ch}}$.

Other transforms, tailored to specific Sigma protocols, additionally use homomorphic commitments [42] or additively homomorphic encryption [18], introducing additional computational assumptions.

Other work focuses on particular classes of interactive protocols: commit-and-open Sigma protocols [26], the Schnorr Sigma protocol [19,45], or multi-round protocols [53]. In this paper, we focus on seed-friendly Sigma protocols, the class formalized by Flamini, Pintore, Signorini, and Tognolini in [34] for which the authors propose a tailored straight-line extractable transform, to later enable the optimizations within the fixed-weight framework. Analyzing transforms for seed-friendly sigma protocols can impact the design and the efficiency of constructions based on cryptographic group actions, including advanced signatures, leading to improvements in the design or security analysis of the existing solutions [28,5,37,38,10]. It also provides useful guidance on extending the analysis of the transform for multi-round identification schemes [6].

### 1.1 Our Contribution

In this paper, we revisit the security of NIZKPs from seed-friendly Sigma protocols in the way they are usually compiled, namely, using the Fiat–Shamir transform and some optimizations within the fixed-weight framework. The most relevant optimization is the use of seed trees to generate the randomness used in each parallel repetition of the seed-friendly Sigma protocol, which reduces the size of the resulting signature or NIZKP. The security of this optimization is usually argued separately, through a hiding property of the seed tree used only in the zero-knowledge proof [10,13]. Soundness is proven with the Forking Lemma [8], independently of how the prover's randomness is generated. In contrast, we use the seeds in the extraction argument as well. We show that generating the randomness from seeds (or seed trees to improve compactness) enables an alternative to the Forking Lemma analysis: the derived NIZKPs are straight-line extractable in the classical random oracle model, without enlarging the proof or requiring a proof of work.

Interestingly, the extraction error is governed by the number of challenge supports (sets of $\rho$ non-special positions), $\binom{L}{\rho}$. In contrast, the usual rewinding-based analysis uses the full challenge-space size $\binom{L}{\rho}(q-1)^\rho$, where $q = |\mathsf{ChSpace}|$. For binary challenges, these combinatorial quantities coincide, and our analysis simply avoids the quadratic tightness loss of the Forking Lemma.

We prove adaptive multi-proof zero knowledge for the derived NIZKP as well as straight-line extraction without requiring unique responses. The latter is an interesting distinction from the Fischlin transform [31] and the GAO transform [34], which require unique responses. For the Sigma protocol from cryptographic group actions, unique responses follow from the hardness of the Stabilizer Computation Problem. Hence, our extraction result does not rely on this problem.

### 1.2 Our Techniques

The approach we use to analyze our NIZKP is the following: from the seed-friendly Sigma protocol $\Sigma$, we define the Sigma protocol $\Sigma_{\mathsf{tree}}$ that is its $(L, \rho)$-fixed-weight parallel repetition, where the randomness used to generate the first message of $\Sigma$ in each of the $L$ repetitions is derived using a seed tree. This is the main difference compared to standard analysis that plugs the use of seed trees on top of the Fiat–Shamir NIZKP. We prove that $\Sigma_{\mathsf{tree}}$ is complete, HVZK, and special sound. This paves the way to our analysis of the compilation of $\Sigma_{\mathsf{tree}}$ into an NIZKP via the standard Fiat–Shamir transform.

*Seed Trees for Randomness Generation.* More in detail, in Section 3 we define the notion of *seed tree* we use to generate the random coins for the $(L, \rho)$-fixed-weight parallel repetition of $\Sigma$. Seed trees are useful because when the prover receives a fixed weight challenge with the indices in $I, |I| = \rho$ corresponding to challenges $\neq 0$, it must disclose all the seeds (the labels of the leaves) in $[L] \setminus I$, while keeping hidden the ones in $I$. But if seeds are generated using a seed tree, this requires revealing only the labels of the smallest set of nodes $\mathsf{Cov}_L(I)$ whose subtrees contain exactly the leaves outside $I$, improving the compactness of the response. To be able to use seed trees, however, more specifically to prove the HVZK of $\Sigma_{\mathsf{tree}}$, we need to ensure that we can simulate the responses generated from seed trees. With this purpose, we present as an intermediate result the seed tree hiding lemma (Lemma 1), that states that the distribution of $\mathsf{Cov}_L(I)$, with their labels, and of the labels of the leaves in $I$, is indistinguishable from uniformly random labels for the same nodes of the tree. This analysis is done considering a programmable random oracle, whose hash table is empty. In practice, the hash $\mathsf{H}$ used to generate seed trees will be fixed; however, every time a prover generates a seed tree, it samples a random $\mathsf{salt} \leftarrow\!\!\$\ \{0,1\}^{2\lambda}$ that is used as a prefix to every random oracle query used for its generation. With overwhelming probability, this salt is unique, and no previous queries have been performed using this salt, so the table associated with $\mathsf{H}(\mathsf{salt}||\cdot)$ is empty.

*$\Sigma_{\mathsf{tree}}$: the Seeded Fixed-Weight Repetition of $\Sigma$.* In Section 4 we define the $(L, \rho)$-fixed-weight repetition of the seed-friendly Sigma protocol $\Sigma$ as the Sigma protocol that explicitly generates its randomness using seed trees. The prover samples a root seed and a fresh salt $\mathsf{salt}$, then uses a random oracle to expand the root into a tree with $L$ leaf seeds. A second, independent oracle maps each leaf to the random tape for its repetition, as $r_i = \mathsf{H}_{\mathsf{rnd}}(\mathsf{salt}, i, \mathsf{seed}_i)$. Including the repetition index makes these inputs distinct, so fresh oracle answers supply independent tapes with the base protocol's randomness distribution. The response reveals a cover of the leaves needed for the special-challenge positions, allowing the verifier to reconstruct their tapes, while keeping the other leaves hidden. The seed tree hiding lemma, together with the HVZK of $\Sigma$, enables us to prove that $\Sigma_{\mathsf{tree}}$ satisfies HVZK. In fact, we can simulate the revealed seed-tree labels and use the base simulator for the remaining repetitions; seed-tree hiding bounds the change in the distinguisher view. The special soundness property is a direct

consequence of the special soundness of $\Sigma$. We also show that for $\Sigma_{\mathsf{tree}}$, first-message retrievability and high first-message min-entropy carry over from the base protocol $\Sigma$.

*$\Pi_{\mathtt{FS}}$: a Straight-Line Extractable NIZKP.* We compile $\Sigma_{\mathsf{tree}}$ with Fiat–Shamir obtaining $\Pi_{\mathtt{FS}}$, and since we have already proven that it satisfies completeness, HVZK, special soundness, and has high first-message min-entropy, the way is paved to prove, in Section 5, that the derived NIZKP satisfies the multi-proof zero-knowledge property (also referred to as unbounded non-interactive zero-knowledge [29]). To prove straight-line extractability, instead, we present an extractor, generalizing that of Beullens et al. [9], that pairs an accepting transcript for $\tilde{\mathsf{ch}}$, reconstructed from a randomness query, with a transcript in the proof having the same first message and a different challenge. Special soundness of $\Sigma$ then yields a witness. The key is to fix which positions have such a randomness query *before* the Fiat–Shamir answer is sampled (Section 5.2). For true statements, high first-message min-entropy bounds the probability that later randomness queries match a fixed first message. Outside this event, acceptance and extraction failure force the challenge support to equal the positions lacking an earlier matching query. This event has probability at most $\binom{L}{\rho}^{-1}$ per fresh Fiat–Shamir query, irrespective of the non-special challenge values. For false statements, special soundness permits at most one accepting challenge vector per first-message vector.

It is interesting to note that, since we do not rely on rewinding for the analysis of the extraction property of this Fiat–Shamir NIZKP, the special soundness of $\Sigma_{\mathsf{tree}}$ is not leveraged for the definition of the extractor.

To conclude, in Section 5.3 we discuss the implications of our bounds for parameter selection, compare our approach with proof-of-work transforms, and list open problems.

## 2 Preliminaries

Throughout the paper we use standard cryptographic notation. We denote by $\mathbb{N}^*$ the set of non-zero natural numbers, by $[N]$ the set $\{1, \ldots, N\}$, and by $[M, N]$ the set $\{M, M+1, \ldots, N\}$. For two functions $a, b : \mathbb{N} \to \mathbb{R}$, we write $a(\lambda) \approx b(\lambda)$ when $|a(\lambda) - b(\lambda)|$ is negligible in $\lambda$.

Where not otherwise specified, each algorithm is probabilistic polynomial-time (PPT). We write $y \leftarrow \mathsf{A}(x)$ for a deterministic assignment, $y \leftarrow\!\!\$\ \mathsf{A}(x)$ for a probabilistic one, and $\mathsf{A}(x; r)$ to make the use of random coins $r$ explicit; in pseudocode, $=$ is reserved for equality testing. For a set $S$, $s \leftarrow\!\!\$\ S$ denotes uniform sampling, and $\perp$ denotes a failure. We write $\mathsf{A}^{\mathsf{F}}$ for the execution of $\mathsf{A}$ with access to a function $\mathsf{F}$; when a construction uses several random oracles, $\mathsf{H}$ denotes their collection, and $A^{\mathsf{H}}$ denotes access to all of them.

We use $\mathcal{R} \subseteq X \times W$ to denote both a family $\{\mathcal{R}_\lambda \subseteq X_\lambda \times W_\lambda\}_{\lambda \in \mathbb{N}}$ of NP binary relations, where $X_\lambda$ and $W_\lambda$ are finite subsets of $\{0,1\}^*$, and any of its elements. Elements of $X$ are called statements; if $(x, w) \in \mathcal{R}$, then $w$ is said to be a witness for $x$.

***Game 1**: Zero-Knowledge*

$\mathsf{Exp}^{\text{ZK-REAL}}_{(\mathsf{Prove},\mathsf{Verify}),\mathsf{H}}(\mathcal{D})$:

1: $(x, w, \mathsf{st}_{\mathcal{D}}) \leftarrow\!\$\ \mathcal{D}_0^{\mathsf{H}}(\lambda)$
2: ***if*** $(x, w) \in \mathcal{R}$ ***then***
3: $\quad \pi \leftarrow\!\$\ \mathsf{Prove}^{\mathsf{H}}(x, w)$
4: ***else***
5: $\quad \pi \leftarrow \bot$
6: ***return*** $\mathcal{D}_1^{\mathsf{H}}(\pi, \mathsf{st}_{\mathcal{D}})$

$\mathsf{Exp}^{\text{ZK-IDEAL}}_{(\mathsf{Prove},\mathsf{Verify}),\mathsf{Sim}}(\mathcal{D})$:

1: $(\mathsf{H}^{(0)}, \mathsf{st}_S) \leftarrow\!\$\ \mathsf{Sim}_0(\lambda)$
2: $(x, w, \mathsf{st}_{\mathcal{D}}) \leftarrow\!\$\ \mathcal{D}_0^{\mathsf{H}^{(0)}}(\lambda)$
3: ***if*** $(x, w) \in \mathcal{R}$ ***then***
4: $\quad (\mathsf{H}^{(1)}, \pi) \leftarrow\!\$\ \mathsf{Sim}_1(\mathsf{st}_S, x, 1)$
5: ***else***
6: $\quad (\mathsf{H}^{(1)}, \pi = \bot) \leftarrow\!\$\ \mathsf{Sim}_1(\mathsf{st}_S, x, 0)$
7: ***return*** $\mathcal{D}_1^{\mathsf{H}^{(1)}}(\pi, \mathsf{st}_{\mathcal{D}})$

**Straight-Line Extractable NIZKPs** In this work, we use the definition of straight-line extractable NIZKPs from [31], also adopted in [34].

**Definition 1** (NIZKP with straight-line extractor [31])**.** *A straight-line extractable NIZKP for a binary relation $\mathcal{R}$ is a pair of oracle-calling PPT algorithms $(\mathsf{Prove}, \mathsf{Verify})$ satisfying the following properties.*

1. Completeness*: for a random oracle $\mathsf{H}$, any $(x, w) \in \mathcal{R}$ and $\pi \leftarrow\!\$\ \mathsf{Prove}^{\mathsf{H}}(x, w)$,*

$$\Pr\left[\mathsf{Verify}^{\mathsf{H}}(x, \pi) = \mathsf{reject}\right] \leq \epsilon_c^{\mathsf{Prove}},$$

   *where $\epsilon_c^{\mathsf{Prove}}$ is negligible in the security parameter $\lambda$.*
2. Zero-Knowledge*: Let $\mathsf{H}$ be a random oracle. There exists a simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ such that, for any PPT distinguisher $\mathcal{D} = (\mathcal{D}_0, \mathcal{D}_1)$, the quantity*

$$\left|\Pr\left[\mathsf{Exp}^{\text{ZK-REAL}}_{(\mathsf{Prove},\mathsf{Verify}),\mathsf{H}}(\mathcal{D}) = 1\right] - \Pr\left[\mathsf{Exp}^{\text{ZK-IDEAL}}_{(\mathsf{Prove},\mathsf{Verify}),\mathsf{Sim}}(\mathcal{D}) = 1\right]\right|$$

   *is negligible in $\lambda$, where $\mathsf{Exp}^{\text{ZK-REAL}}_{(\mathsf{Prove},\mathsf{Verify}),\mathsf{H}}(\mathcal{D})$ and $\mathsf{Exp}^{\text{ZK-IDEAL}}_{(\mathsf{Prove},\mathsf{Verify}),\mathsf{Sim}}(\mathcal{D})$ are defined in Game 1.*
3. Straight-Line Extraction*: there exists a PPT algorithm $\mathsf{Ext}$ such that, for every adversary $\mathcal{A}$, if $\mathsf{H}$ is a random oracle, $(x, \pi) \leftarrow\!\$\ \mathcal{A}^{\mathsf{H}}(\lambda)$, and $Q_{\mathsf{H}}(\mathcal{A})$ is the ordered list of queries made by $\mathcal{A}$ together with their answers, then*

$$\Pr\left[\mathsf{Verify}^{\mathsf{H}}(x, \pi) = \mathsf{accept} \ \wedge\ \big(x, \mathsf{Ext}(x, \pi, Q_{\mathsf{H}}(\mathcal{A}))\big) \notin \mathcal{R}\right] \leq \epsilon_{\mathsf{sle}}, \qquad (1)$$

   *where $\epsilon_{\mathsf{sle}}$ is negligible in $\lambda$.*

   Note that the probability in (1) is not conditioned on acceptance.

**Seed-Friendly Sigma Protocols** A Sigma protocol $\Sigma = (\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2))$ for a binary relation $\mathcal{R}$ is a three-message public-coin interactive proof system: on input $(x, w) \in \mathcal{R}$, the prover sends a first message $f \leftarrow\!\$\ \mathsf{P}_1(x, w)$, the verifier replies with a uniformly random challenge $\mathsf{ch}$ from a challenge space $\mathsf{ChSpace}$, and the prover responds with $z \leftarrow\!\$\ \mathsf{P}_2(x, w, f, \mathsf{ch})$. The tuple $(f, \mathsf{ch}, z)$ is called a *transcript*. For a Sigma protocol, we require the standard properties (completeness, special soundness, honest-verifier zero-knowledge) from [22].

*First-Message Min-Entropy.* For every fixed $(x, w) \in \mathcal{R}$, the min-entropy of the first message is

$$H_\infty\big(\mathsf{P}_1(x, w)\big) := -\log_2 \max_f \Pr[\mathsf{P}_1(x, w) = f], \tag{2}$$

where the probability is over the honest prover's randomness. We say that $\Sigma$ has first-message min-entropy at least $h = h(\lambda)$ if the quantity in (2) is at least $h$ for every $(x, w) \in \mathcal{R}$, and that it has *high first-message min-entropy* if $h = \omega(\log \lambda)$ [31,1]. Throughout the security bounds, $h$ denotes this uniform lower bound.

**Definition 2** (First-Message-Retrievable Sigma Protocol)**.** *A Sigma protocol $\Sigma$ is* first-message retrievable *if there is a deterministic PPT algorithm $F$ such that, for every statement $x$ and every accepting transcript $(f, \mathsf{ch}, z)$ for $x$,*

$$F(x, \mathsf{ch}, z) = f.$$

*Thus, the challenge and response uniquely determine the first message of every accepting transcript. On inputs that do not correspond to an accepting transcript, the output of $F$ may be arbitrary.*

**Definition 3** (Unique Responses)**.** *A Sigma protocol $\Sigma$ for a relation $\mathcal{R}$ has* unique responses *if, for every PPT adversary $\mathcal{A}$, the probability*

$$\mathsf{Adv}^{\mathrm{ur}}_{\Sigma,\mathcal{A}}(\lambda) := \Pr\left[\begin{matrix}(x, f, \mathsf{ch}, z, z') \leftarrow\!\$\ \mathcal{A}(1^\lambda) :\ z \neq z' \wedge \\ \mathsf{V}_2(x, f, \mathsf{ch}, z) = \mathsf{V}_2(x, f, \mathsf{ch}, z') = \mathsf{accept}\end{matrix}\right]$$

*is negligible in $\lambda$.*

Note that this property is not restricted to true statements in the relation. We next define a special class of Sigma protocols that we call *seed-friendly*.

**Definition 4** (Seed-Friendly Sigma Protocol)**.** *Consider a Sigma protocol $\Sigma = (\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2))$ for a binary relation $\mathcal{R}$, with $\mathsf{ChSpace}$ being its challenge space and $\mathcal{D}_{\mathsf{rnd}}$ the distribution of the random tape of $\mathsf{P}_1$. We say that $\Sigma$ is seed-friendly if the following properties are satisfied:*

1. *it is first-message retrievable through a deterministic algorithm $F$;*
2. *$|\mathsf{ChSpace}|$ is polynomially bounded;*
3. *there is a special challenge $\tilde{\mathsf{ch}} \in \mathsf{ChSpace}$ and a deterministic public polynomial-time function $Z$ such that, for every $(x, w) \in \mathcal{R}$ and $r$ in the support of $\mathcal{D}_{\mathsf{rnd}}$, if $f = \mathsf{P}_1(x, w; r)$ then $(f, \tilde{\mathsf{ch}}, Z(x, r))$ is accepting;*
4. *it has high first-message min-entropy.*

*Remark 1.* Throughout this work, we will identify the challenge space $\mathsf{ChSpace}$ of a seed-friendly Sigma protocol with the set $\{0, 1, \ldots, |\mathsf{ChSpace}| - 1\}$, and assume that the special challenge $\tilde{\mathsf{ch}}$ is 0.

*Remark 2.* Unlike in [34], the unique-responses property is not required to satisfy seed friendliness, and none of our results relies on it.

For a statement $x$ and a random tape $r$, we call $T_x(r) := F(x, \tilde{\mathsf{ch}}, Z(x, r))$ the *traced first message*, which is publicly computable without a witness. Properties 1 and 3 give $T_x(r) = \mathsf{P}_1(x, w; r)$ for every $(x, w) \in \mathcal{R}$ and supported random tape $r$. Thus, for true statements, $T_x(r)$ has exactly the honest first-message distribution, independently of the witness.

*Remark 3.* Property 3 provides the public response function $Z$; it does not by itself provide a short seed to derive $r$. In our ROM construction, a seed is mapped to $r$ by the randomness oracle $\mathsf{H}_{\mathsf{rnd}}(\mathsf{salt}, i, \mathsf{seed})$, whose value at every fresh input is sampled from $\mathcal{D}_{\mathsf{rnd}}$. Replacing this oracle with a concrete seed expander requires a separate pseudorandomness argument for the resulting transcripts; a short deterministic seed need not reproduce $\mathcal{D}_{\mathsf{rnd}}$ in statistical distance.

Seed-friendly Sigma protocols arise naturally across diverse settings (e.g., the Schnorr Sigma protocol with a restricted challenge space). A classic example is the Sigma protocol for graph isomorphisms [36], restricted to graphs with at least $2^h$ distinct relabelings.

In the post-quantum setting, the Sigma protocol $\Sigma_\star$ from cryptographic group actions (Appendix B) is seed-friendly when all permitted orbits have size at least $2^h$, and has unique responses under the Stabilizer Computation Problem assumption.

The seed-friendly structure extends to more advanced protocols built on top of $\Sigma_\star$, such as the ring and group signature schemes in [10,9].

**Parallel and Fixed-Weight Repetition** The $L$-fold parallel repetition $\Sigma^L$ executes $L$ independent copies of $\Sigma$ for the same relation [22]. Its transcripts are $\big((f_i)_{i\in[L]}, (\mathsf{ch}_i)_{i\in[L]}, (z_i)_{i\in[L]}\big)$, with each $(f_i, \mathsf{ch}_i, z_i)$ a base-protocol transcript and challenge space $\mathsf{ChSpace}^L$. Special soundness bounds its soundness error by $|\mathsf{ChSpace}|^{-L}$. Fixed-weight repetition restricts the number of non-special challenges to control proof size.

**Definition 5** (Fixed-Weight Repetition)**.** *Let $\Sigma = (\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2))$ be a seed-friendly Sigma protocol for a binary relation $\mathcal{R}$ with challenge space $\mathit{ChSpace}$. Given $L, \rho \in \mathbb{N}^*$ with $\rho \le L$, the $(L, \rho)$-*fixed-weight repetition *of $\Sigma$, denoted by $\Sigma^{L,\rho}$, is the $L$-fold parallel repetition of $\Sigma$ with challenge space restricted to*

$$\mathit{ChSpace}^{L,\rho} = \{\mathsf{ch} \in \mathit{ChSpace}^L \mid \mathsf{wt}(\mathsf{ch}) = \rho\},$$

*where $\mathsf{wt}(\mathsf{ch})$ denotes the number of components $\mathsf{ch}_i \neq \tilde{\mathsf{ch}}$.*

A transcript for $\Sigma^{L,\rho}$ contains exactly $L - \rho$ transcripts of $\Sigma$ for the special challenge $\tilde{\mathsf{ch}}$, whose responses can be encoded as $\lambda$-bit seeds, and $\rho$ transcripts with full-size responses. The special soundness of $\Sigma$ makes the soundness error of $\Sigma^{L,\rho}$ equal to

$$\frac{1}{|\mathsf{ChSpace}^{L,\rho}|} = \binom{L}{\rho}^{-1} \frac{1}{\big(|\mathsf{ChSpace}| - 1\big)^{\rho}},$$

and the parameters $(L, \rho)$ are chosen so that $|\mathsf{ChSpace}^{L,\rho}| \ge 2^\lambda$.

**Random Oracles and Salting** Our constructions and security proofs are in the ROM. We use three independent oracles with the following query form and output distributions:

- $\mathsf{H}_{\mathsf{tree}}(\mathsf{salt}, \mathsf{ad}, \mathsf{seed})$ returns a uniform $2\lambda$-bit string, interpreted as two $\lambda$-bit child seeds.
- $\mathsf{H}_{\mathsf{rnd}}(\mathsf{salt}, i, \mathsf{seed})$ returns a sample from $\mathcal{D}_{\mathsf{rnd}}$, the distribution of the prover random tape for a given seed-friendly Sigma protocol $\Sigma$.
- $\mathsf{H}_{\mathsf{FS}}(x, \mathbf{f})$ returns a uniform challenge in the challenge space $\mathsf{ChSpace}^{L,\rho}$ of $\Sigma^{L,\rho}$.

Every fresh input receives an independent sample from the specified output distribution; repeated queries return the same answer. In particular, $\mathsf{H}_{\mathsf{rnd}}$ is an ideal random function with output distribution $\mathcal{D}_{\mathsf{rnd}}$, which need not be uniform. The underlying Sigma protocol $\Sigma$ and parameters $L, \rho$ are fixed and implicit. Here $\mathsf{seed} \in \{0,1\}^\lambda$, $\mathsf{ad}$ is a node address, $i \in [L]$, and $\mathbf{f}$ is a vector of $L$ first messages for $\Sigma$.

Oracle inputs are encoded canonically and injectively, with distinct domain-separation tags $\mathsf{tree}$, $\mathsf{rnd}$, and $\mathsf{FS}$. We write $\mathsf{H}(q)$ for the answer to a tagged query $q$. For an adversary $\mathcal{A}$, the ordered transcript $Q_{\mathsf{H}}(\mathcal{A})$ records tagged input–answer pairs, such as $\big((\mathsf{rnd}, \mathsf{salt}, i, \mathsf{seed}), r\big)$. An implementation can use one domain-separated XOF followed by appropriate sampling for $\mathcal{D}_{\mathsf{rnd}}$ and $\mathsf{ChSpace}^{L,\rho}$; the analysis uses the ideal output distributions above. Digest-encoded variants with an additional commitment oracle are discussed in Appendix C.

The salt $\mathsf{salt}$ is public and sampled uniformly from $\{0,1\}^{2\lambda}$, independently for each honest execution.

## 3 Salted Seed Trees

Our construction and our security analysis strongly relies on the property of *salted seed-trees* (just seed tree in the following) that enable the generation of multiple seeds (leaves) starting from a single seed root. The way a seed tree is generated is through queries to a random oracle $\mathsf{H}_{\mathsf{tree}}$, always queried with a prefix that is a salt, in our constructions a $2\lambda$-bit string, which is specific for each seed tree. The use of a random salt in the generation of a seed tree is crucial to ensure that the derived random oracle $\mathsf{H}_{\mathsf{tree}}(\mathsf{salt}||\cdot)$ will be non-programmed in the inputs queried during the seed tree generation, unless with negligible probability.

We adopt the notation of [10, Section 2.6]. Fix $L$ and recursively split each interval of leaf indices into a left interval of size $\lceil L/2 \rceil$ and a right interval of size $\lfloor L/2 \rfloor$, stopping at single leaves. This determines the binary tree $\mathcal{T}_L$ with leaves indexed from left to right by $[L]$. The root has address $\varepsilon$; the left and right children of a node at address $\mathsf{ad}$ have addresses $\mathsf{ad}\|0$ and $\mathsf{ad}\|1$, respectively. A seed tree is specified by $(L, \mathsf{salt}, \mathsf{seed}_{\mathsf{root}})$, with $\mathsf{seed}_{\mathsf{root}} \leftarrow\!\!\$\ \{0,1\}^\lambda$. If a node at address $\mathsf{ad}$ has label $\mathsf{seed}$, its children are labelled by the two $\lambda$-bit blocks of $\mathsf{H}_{\mathsf{tree}}(\mathsf{salt}, \mathsf{ad}, \mathsf{seed})$. Distinct nodes have distinct addresses, so their oracle inputs differ even when their labels agree.

**Definition 6** (Seed Tree and Cover)**.** *For $I \subseteq [L]$, let $\mathsf{Cov}_L(I)$ be the smallest set of nodes of $\mathcal{T}_L$ whose leaf sets partition the leaves with indices in $[L] \setminus I$ (equivalently, the maximal nodes with no descendant leaf indexed by $I$). A seed tree is provided with the following algorithms.*

- $(\mathsf{seed}_1, \ldots, \mathsf{seed}_L) \leftarrow \mathsf{SeedTree}^{\mathsf{H}_{\mathsf{tree}}}(\mathsf{salt}, \mathsf{seed}_{\mathsf{root}}, L)$ *computes all labels and returns the leaf labels.*
- $\mathsf{seeds}_{\mathsf{int}} \leftarrow \mathsf{ReleaseSeeds}^{\mathsf{H}_{\mathsf{tree}}}(\mathsf{salt}, \mathsf{seed}_{\mathsf{root}}, L, I)$ *returns the address–label pairs of the nodes in $\mathsf{Cov}_L(I)$, in* canonical order*, i.e. by lexicographic address (left to right in $\mathcal{T}_L$).*
- $\{\mathsf{seed}_i\}_{i \in [L] \setminus I} \leftarrow \mathsf{RecoverLeaves}^{\mathsf{H}_{\mathsf{tree}}}(\mathsf{salt}, \mathsf{seeds}_{\mathsf{int}}, L, I)$ *returns $\perp$ unless the addresses in $\mathsf{seeds}_{\mathsf{int}}$ are exactly those of $\mathsf{Cov}_L(I)$, in canonical order; otherwise it expands the supplied labels and returns the leaf labels with indices in $[L] \setminus I$.*

We now show that the distribution of $\mathsf{seeds}_{\mathsf{int}}$ and the hidden seeds $\{\mathsf{seed}_i\}_{i \in I}$ is indistinguishable from random seeds if we consider an adversary who can query the random oracle after seeing a real or simulated sample of seeds.

For the ideal distribution used below, we introduce the algorithm $\mathsf{seeds}_{\mathsf{int}} \leftarrow\!\!\$\ \mathsf{SimulateSeeds}(L, I)$ that assigns independent uniform $\lambda$-bit labels to the nodes of $\mathsf{Cov}_L(I)$ and returns their canonical address–label list. We write $N_{\mathsf{seed}}$ for the maximum size of $\mathsf{Cov}_L(I)$ over sets $I \subseteq [L]$ of size $\rho$; standard encodings give $N_{\mathsf{seed}} = O(\rho \log(L/\rho) + \log L)$ [13, Proposition 2].

The following lemma, whose proof is given in Appendix A.1, adapts [9, Lemma A.3]: in this experiment we expose both the released cover and the hidden leaf labels and include node addresses in oracle inputs. In fact, this is the setting needed when the hidden leaf labels are replaced by independently generated Sigma-protocol transcripts in the HVZK proof (Theorem 3).

**Lemma 1** (Seed-Tree Hiding)**.** *Fix $L \in \mathbb{N}^*$, $I \subseteq [L]$, and a salt $\mathsf{salt}$. Consider the following two experiments, both giving their output and oracle access to a computationally unbounded distinguisher $\mathcal{A}$*

$$
\begin{aligned}
\mathsf{Real}_I:\ & \mathsf{seed}_{\mathsf{root}} \leftarrow\!\!\$\ \{0,1\}^\lambda;\\
& (\mathsf{seed}_1, \ldots, \mathsf{seed}_L) \leftarrow \mathsf{SeedTree}^{\mathsf{H}_{\mathsf{tree}}}(\mathsf{salt}, \mathsf{seed}_{\mathsf{root}}, L);\\
& \mathsf{seeds}_{\mathsf{int}} \leftarrow \mathsf{ReleaseSeeds}^{\mathsf{H}_{\mathsf{tree}}}(\mathsf{salt}, \mathsf{seed}_{\mathsf{root}}, L, I);\\
& \textit{return } (\mathsf{seeds}_{\mathsf{int}}, (\mathsf{seed}_i)_{i \in I}),\\
\mathsf{Ideal}_I:\ & \mathsf{seeds}_{\mathsf{int}} \leftarrow\!\!\$\ \mathsf{SimulateSeeds}(L, I);\\
& \mathsf{seed}_i \leftarrow\!\!\$\ \{0,1\}^\lambda \textit{ independently for } i \in I;\\
& \textit{return } (\mathsf{seeds}_{\mathsf{int}}, (\mathsf{seed}_i)_{i \in I}).
\end{aligned}
$$

*In each experiment, the oracle table starts empty. The experiment generates its output before invoking $\mathcal{A}$, which then queries that same oracle table. If $\mathcal{A}(\mathsf{seeds}_{\mathsf{int}}, (\mathsf{seed}_i)_{i \in I})$ makes at most $Q_{\mathsf{tree}}$ queries to $\mathsf{H}_{\mathsf{tree}}(\mathsf{salt}, \cdot, \cdot)$, then,*

$$
|\Pr[\mathcal{A}(\mathsf{Real}_I) = 1] - \Pr[\mathcal{A}(\mathsf{Ideal}_I) = 1]| \le \frac{Q_{\mathsf{tree}}}{2^\lambda}.
$$

In particular, up to some negligible additive term, the hidden leaf lables are jointly uniform and independent of the released labels of the cover. The claim remains true after applying $\mathsf{RecoverLeaves}$ to the cover and its labels, by deterministic post-processing.

So far we have only described seed-trees and in Lemma 1 we have characterized the distribution of the released seeds, together with those which remain hidden (with indices in $I$). As seed trees will be used in our construction to derive the prover's random tape in S

igma protocol executions, we state the following corollary describing how these random tapes are distributed when generated honestly.

**Corollary 1.** *Fix $L \in \mathbb{N}^*$ and a salt $\mathsf{salt} \in \{0,1\}^{2\lambda}$. Let $\mathsf{H}_{\mathsf{tree}}$ and $\mathsf{H}_{\mathsf{rnd}}$ be the independent random oracles defined in Section 2. Sample $\mathsf{seed}_{\mathsf{root}} \leftarrow\!\!\$\ \{0,1\}^{\lambda}$ and compute*

$$\begin{aligned}(\mathsf{seed}_1, \ldots, \mathsf{seed}_L) &\leftarrow \mathsf{SeedTree}^{\mathsf{H}_{\mathsf{tree}}}(\mathsf{salt}, \mathsf{seed}_{\mathsf{root}}, L),\\ r_i &\leftarrow \mathsf{H}_{\mathsf{rnd}}(\mathsf{salt}, i, \mathsf{seed}_i) \qquad (i \in [L]).\end{aligned}$$

*Then $r_1, \ldots, r_L$ are independent samples from $\mathcal{D}_{\mathsf{rnd}}$. Equivalently, their joint distribution is $\mathcal{D}_{\mathsf{rnd}}^L$, where the probability is over the root seed and both random oracles.*

*Proof sketch.* The leaf labels are independent of $\mathsf{H}_{\mathsf{rnd}}$, whose inputs are distinct by index even when labels coincide. Conditioning on the labels therefore gives independent $\mathcal{D}_{\mathsf{rnd}}$ samples; the full argument is in Appendix A.2. □

*Remark 4.* In the multi-proof zero-knowledge hybrid (Theorem 4), a newly sampled $2\lambda$-bit salt matches a salt in the prior tree and randomness queries with probability at most $Q_{\mathsf{zk}} 2^{-2\lambda}$. Conditioned on no such match, the seed-tree oracle entries for the new salt are fresh, as assumed in Lemma 1.

## 4 A Seeded Fixed-Weight Sigma Protocol

In this section, we describe the generic interactive protocol whose transform via Fiat-Shamir will be analysed in the following sections. The protocol, which we denote by $\Sigma_{\mathsf{tree}}$, is the $(L, \rho)$-fixed-weight repetition of a seed-friendly Sigma protocol $\Sigma$ (Definition 5) in which the prover's randomness is generated from a salted seed tree. As we prove in Section 5, the placement inside the protocol of this randomness-generation procedure, via random oracle queries, is what makes its Fiat–Shamir compilation straight-line extractable. The first message remains a vector of $L$ first messages for $\Sigma$; its retrievability will allow the non-interactive proof to omit this vector[5].

[5] Variants where the vector of first messages is encoded as a random oracle digest before computing the fixed-weight challenge, used by concrete protocols such as LESS [12], are discussed in Appendix C.

**Transformation 1**: Seeded fixed-weight repetition from $\Sigma$ to $\Sigma_{\mathsf{tree}}$

We consider the (implicit) parameterization of $\Sigma_{\mathsf{tree}}$ with $\rho, L \in \mathbb{N}*,\ \rho \leq L$.

**Round 1:** $\mathsf{P}_1^{\mathsf{tree},\mathsf{H}}(x, w)$:
1: $\mathsf{salt} \leftarrow\$ \{0,1\}^{2\lambda}$
2: $\mathsf{seed}_{\mathsf{root}} \leftarrow\$ \{0,1\}^{\lambda}$
3: $(\mathsf{seed}_1, \ldots, \mathsf{seed}_L) \leftarrow \mathsf{SeedTree}^{\mathsf{H}_{\mathsf{tree}}}(\mathsf{salt}, \mathsf{seed}_{\mathsf{root}}, L)$
4: **for** $i$ **from** 1 **to** $L$ **do**
5: $\quad r_i \leftarrow \mathsf{H}_{\mathsf{rnd}}(\mathsf{salt}, i, \mathsf{seed}_i)$
6: $\quad f_i \leftarrow \mathsf{P}_1(x, w; r_i)$
7: $\mathbf{f} \leftarrow (f_1, \ldots, f_L)$
8: $st \leftarrow (\mathsf{salt}, \mathsf{seed}_{\mathsf{root}}, \mathbf{f})$
9: Prover returns $\mathbf{f}$ to Verifier and retains $st$.

**Round 2:** $\mathsf{V}_1^{\mathsf{tree}}(\mathbf{f})$:
1: $\mathsf{ch} = (\mathsf{ch}_1, \ldots, \mathsf{ch}_L) \leftarrow\$ \mathsf{ChSpace}^{L,\rho}$
2: Verifier returns $\mathsf{ch}$ to Prover.

**Round 3:** $\mathsf{P}_2^{\mathsf{tree},\mathsf{H}}(x, w, st, \mathsf{ch})$:
1: $(\mathsf{salt}, \mathsf{seed}_{\mathsf{root}}, \mathbf{f}) \leftarrow st$
2: $I \leftarrow \{i \in [L] \mid \mathsf{ch}_i \neq 0\}$
3: **for** $i \in I$ **do**
4: $\quad z_i \leftarrow\$ \mathsf{P}_2(x, w, f_i, \mathsf{ch}_i)$
5: $\mathsf{seeds}_{\mathsf{int}} \leftarrow \mathsf{ReleaseSeeds}^{\mathsf{H}_{\mathsf{tree}}}(\mathsf{salt}, \mathsf{seed}_{\mathsf{root}}, L, I)$
6: Prover sends $\mathsf{resp} \leftarrow (\mathsf{salt}, \mathsf{seeds}_{\mathsf{int}}, \{z_i\}_{i \in I})$ to Verifier.

**Verification:** $\mathsf{V}_2^{\mathsf{tree},\mathsf{H}}(x, \mathbf{f}, \mathsf{ch}, \mathsf{resp})$:
1: Output reject unless $\mathsf{ch} \in \mathsf{ChSpace}^{L,\rho}$ and $\mathsf{resp}$ is of the form $(\mathsf{salt}, \mathsf{seeds}_{\mathsf{int}}, (z_i)_{i \in I})$, where $I = \{i \mid \mathsf{ch}_i \neq 0\}$
2: $\{\mathsf{seed}_i\}_{i \in [L] \setminus I} \leftarrow \mathsf{RecoverLeaves}^{\mathsf{H}_{\mathsf{tree}}}(\mathsf{salt}, \mathsf{seeds}_{\mathsf{int}}, L, I)$
3: Output reject if $\mathsf{RecoverLeaves}$ returns $\perp$
4: **for** $i \in [L] \setminus I$ **do**
5: $\quad \widetilde{r}_i \leftarrow \mathsf{H}_{\mathsf{rnd}}(\mathsf{salt}, i, \mathsf{seed}_i)$
6: $\quad \widetilde{z}_i \leftarrow Z(x, \widetilde{r}_i)$,
7: $\quad$ Output reject if $\mathsf{V}_2(x, f_i, 0, \widetilde{z}_i) = \mathsf{reject}$
8: **for** $i \in I$ **do**
9: $\quad$ Output reject if $\mathsf{V}_2(x, f_i, \mathsf{ch}_i, z_i) = \mathsf{reject}$
10: Output accept

We overview below the distinguishing features of $\Sigma_{\mathsf{tree}}$, that we describe formally in the next section.

- *Seeded randomness.* The per-repetition randomness is derived as $r_i \leftarrow \mathsf{H}_{\mathsf{rnd}}(\mathsf{salt}, i, \mathsf{seed}_i)$, where the leaf seeds $\mathsf{seed}_1, \ldots, \mathsf{seed}_L$ are generated with a seed tree (Section 3) rooted in $\mathsf{seed}_{\mathsf{root}}$. The prover independently samples a $2\lambda$-bit salt $\mathsf{salt}$, included in the response, and uses it in every query to $\mathsf{H}_{\mathsf{tree}}$ and $\mathsf{H}_{\mathsf{rnd}}$ during a protocol execution.
- *Fixed-weight challenges.* Challenges are sampled from $\mathsf{ChSpace}^{L,\rho}$; the $L - \rho$ components with $\mathsf{ch}_i = 0$ are answered by releasing seed-tree labels via $\mathsf{ReleaseSeeds}$.

### 4.1 The Protocol $\Sigma_{\mathsf{tree}}$

Let $\Sigma = (\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2))$ be a seed-friendly Sigma protocol for a binary relation $\mathcal{R}$ with challenge space $\mathsf{ChSpace}$, and let $L, \rho \in \mathbb{N}^*$, $\rho \leq L$. Applying Transformation 1 to $\Sigma$ yields the protocol $\Sigma_{\mathsf{tree}} = (\mathsf{P}^{\mathsf{tree}} = (\mathsf{P}_1^{\mathsf{tree}}, \mathsf{P}_2^{\mathsf{tree}}), \mathsf{V}^{\mathsf{tree}} = (\mathsf{V}_1^{\mathsf{tree}}, \mathsf{V}_2^{\mathsf{tree}}))$ for $\mathcal{R}$. The number $L$ of parallel repetitions is implicit in the algorithms $\mathsf{P}^{\mathsf{tree}}, \mathsf{V}^{\mathsf{tree}}$.

We now show that the Sigma protocol $\Sigma_{\mathsf{tree}}$ is first message retrievable.

**Proposition 1.** *If $\Sigma$ is seed-friendly, with first-message-retrieval algorithm $F$ and response-computing algorithm $Z$, then also $\Sigma_{\mathsf{tree}}$ is first message retrievable.*

**Algorithm 1**: First message retrieval algorithm $F^{\mathsf{tree},\mathsf{H}}$ for $\Sigma_{\mathsf{tree}}$

$F^{\mathsf{tree},\mathsf{H}}(x, \mathsf{ch}, \mathsf{resp})$:
1: $I \leftarrow \{i \in [L] \mid \mathsf{ch}_i \neq 0\}$
2: $(\mathsf{salt}, \mathsf{seeds}_{\mathsf{int}}, \{z_i\}_{i \in I}) \leftarrow \mathsf{resp}$
3: $\{\mathsf{seed}_i\}_{i \in [L] \setminus I} \leftarrow \mathsf{RecoverLeaves}^{\mathsf{H}_{\mathsf{tree}}}(\mathsf{salt}, \mathsf{seeds}_{\mathsf{int}}, L, I)$
4: **return** $\perp$ if RecoverLeaves returns $\perp$
5: **for** $i \in [L] \setminus I$ **do**
6: $\quad \widetilde{r}_i \leftarrow \mathsf{H}_{\mathsf{rnd}}(\mathsf{salt}, i, \mathsf{seed}_i)$
7: $\quad f_i \leftarrow T_x(\widetilde{r}_i)$
8: **for** $i \in I$ **do**
9: $\quad f_i \leftarrow F(x, \mathsf{ch}_i, z_i)$
10: $\mathbf{f} \leftarrow (f_1, \ldots, f_L)$
11: **return** $\mathbf{f}$

*Proof.* In Algorithm 1, we define a deterministic algorithm $F^{\mathsf{tree},\mathsf{H}}(x, \mathsf{ch}, \mathsf{resp})$[6] such that, given a challenge $\mathsf{ch}$ and a response $\mathsf{resp}$ from a transcript of the protocol $\Sigma_{\mathsf{tree}}$ obtained via Transformation 1, together with the proven statement $x$, returns the unique first message $\mathbf{f}$ for which $(\mathbf{f}, \mathsf{ch}, \mathsf{resp})$ is an accepting transcript.

We notice that every step is deterministic and enables the computation of a single first message, as long as $\mathsf{ch}$ and $\mathsf{resp}$ are well formed.

□

*Tracing from a Seed.* Using the public map $T_x$ introduced in Section 2, the *first message traced from a seed* $\mathsf{seed}$ at salt $\mathsf{salt}$ and index $i$ is simply

$$T_x\big(\mathsf{H}_{\mathsf{rnd}}(\mathsf{salt}, i, \mathsf{seed})\big).$$

This is a first message reconstructed at a zero-challenge position in Algorithm 1.

*Remark 5.* It is obvious that if (1) the random tape is sampled from $\mathcal{D}_{\mathsf{rnd}}$ (the distribution used to sample outputs of $\mathsf{H}_{\mathsf{rnd}}$), (2) the min-entropy of the first message for a fixed $(w, x) \in \mathcal{R}$ is $\geq h$, and (3) the first message is deterministically determined by the random tape of the prover, then also the min-entropy of $\mathcal{D}_{\mathsf{rnd}}$ is $\geq h$. Otherwise, if a tape was sampled with probability $< 2^{-h}$, also $T_x(r)$ would appear with the same probability.

### 4.2 Security of $\Sigma_{\mathsf{tree}}$

We establish completeness, first-message min-entropy, and special soundness for $\Sigma_{\mathsf{tree}}$. A peculiarity of this Sigma protocol is that it provides random oracle access to both the prover and the verifier, even if it is an interactive protocol, and it is proven secure in the ROM model. This might seem a weakness of the protocol; however, since we target the non-interactive version of it, compiled with Fiat-Shamir, we would need to rely on the ROM anyway.

[6] We use a superscript to distinguish it from the algorithm $F$ for the base Sigma protocol $\Sigma$.

*Min-Entropy and HVZK with RO access.* We stress that the entropy calculation below is over the prover's coins and the independent oracles. The HVZK experiments likewise start with fresh oracle tables. In a shared-oracle execution, a newly sampled $2\lambda$-bit salt makes this execution's tree and randomness inputs fresh relative to earlier queries, except for salt conflicts charged separately in the Fiat–Shamir analysis.

**Theorem 1** (Completeness and Min-Entropy)**.** *If $\Sigma$ is complete, then $\Sigma_{\mathsf{tree}}$ is complete. Moreover, for every fixed $(x, w) \in \mathcal{R}$, if $\Sigma$ has first-message min-entropy at least $h$, the first message $\mathbf{f}$ of a single honest execution of $\Sigma_{\mathsf{tree}}$ has at least $Lh$ bits of min-entropy over the prover's coins and the independent oracles.*

*Proof.* Within an honest execution, $\mathsf{RecoverLeaves}$ reconstructs the leaf labels $\mathsf{seed}_i$ at every index $i$ outside $I$, and the algorithm $Z(x, \widetilde{r}_i)$ returns the response associated to the first message traced by $\mathsf{seed}_i$, namely $f_i$. At indices $i \in I$, the completeness of $\Sigma$ guarantees that $\mathsf{V}_2(x, f_i, \mathsf{ch}_i, z_i) = \mathsf{accept}$. Thus, every component check is successful.

By Corollary 1, the random tapes $r_1, \ldots, r_L$ in an honest execution are independent samples from $\mathcal{D}_{\mathsf{rnd}}$. Hence, for every fixed vector $\mathbf{g} = (g_1, \ldots, g_L)$,

$$\Pr[\mathbf{f} = \mathbf{g}] = \prod_{i=1}^{L} \Pr[\mathsf{P}_1(x, w; r_i) = g_i] \leq 2^{-Lh}$$

by (2). □

*Remark 6.* The above proves the stated min-entropy bound over the prover's coins and oracle choice, *not conditioned on a fixed oracle*. To account for the salt in a shared-oracle execution, fix a prior oracle transcript with at most $Q$ tree and randomness oracle inputs. A newly sampled $2\lambda$-bit salt matches a salt in that transcript with probability at most $Q2^{-2\lambda}$. When it does not match, all salt-dependent inputs of this execution are fresh relative to the transcript, so the same product bound holds conditioned on that transcript and no salt match. The exceptional event is charged separately in Corollary 3 and Theorem 4.

**Theorem 2** (Special Soundness)**.** *Let $\Sigma$ be a seed-friendly Sigma protocol for $\mathcal{R}$ satisfying special soundness. Then also $\Sigma_{\mathsf{tree}}$ is special sound.*

*Proof.* We show that there exists a deterministic polynomial-time algorithm $\mathsf{Ext}^{\mathsf{tree}}$ that, on input $x$ and two accepting transcripts

$$\big(\mathbf{f}, \mathsf{ch}, \mathsf{resp}\big) \quad \text{and} \quad \big(\mathbf{f}, \mathsf{ch}', \mathsf{resp}'\big), \qquad \mathsf{ch} \neq \mathsf{ch}',$$

outputs a witness $w$ with $(x, w) \in \mathcal{R}$. Choose any $j$ such that $\mathsf{ch}_j \neq \mathsf{ch}'_j$. From each response, reconstruct the corresponding base-protocol response: use the included $z_j$ (resp. $z'_j$) when the challenge is nonzero, and use $Z(x, \mathsf{H}_{\mathsf{rnd}}(\mathsf{salt}, j, \mathsf{seed}_j))$ (resp. $Z(x, \mathsf{H}_{\mathsf{rnd}}(\mathsf{salt}', j, \mathsf{seed}'_j))$) when $\mathsf{ch}_j = 0$ (resp. $\mathsf{ch}'_j = 0$). Since both transcripts of $\Sigma_{\mathsf{tree}}$ are accepting, the two base transcripts of $\Sigma$ at component $j$ are accepting and correspond to the same first message $f_j$. Then the special-soundness extractor of $\Sigma$ returns $w$ with $(x, w) \in \mathcal{R}$. □

*Remark 7 (Unique responses are not needed).* Neither Theorem 2 nor the straight-line extraction theorem (Theorem 5) uses unique responses. Thus, for $\Sigma_\star$, our results do not rely on the hardness of the Stabilizer Computation Problem.

We prove HVZK for $\Sigma_{\mathsf{tree}}$ with post-transcript access to the tree and randomness oracles. Because the prover uses these oracles, the distinguisher must see oracle answers consistent with the transcript. The salt is sampled during transcript generation and appears in the response.

**Theorem 3** (HVZK)**.** *Let $\Sigma$ be a seed-friendly Sigma protocol for $\mathcal{R}$ satisfying HVZK. There exists a PPT simulator $\mathsf{Sim}^{\mathsf{tree}}$ with access to $\mathsf{H}$ such that, for any $(x, w) \in \mathcal{R}$, any $\mathsf{ch} \in \mathit{ChSpace}^{L,\rho}$, and any computationally-unbounded distinguisher $\mathcal{A}$ making at most $Q$ queries to $\mathsf{H}$,*

$$\left|\Pr\left[\mathcal{A}^{\mathsf{H}}\left(\mathsf{P}^{\mathsf{tree},\mathsf{H}}(x, w, \mathsf{ch})\right) = 1\right] - \Pr\left[\mathcal{A}^{\mathsf{H}}\left(\mathsf{Sim}^{\mathsf{tree},\mathsf{H}}(x, \mathsf{ch})\right) = 1\right]\right| \leq \frac{Q}{2^\lambda},$$

*where $\mathsf{P}^{\mathsf{tree},\mathsf{H}}(x, w, \mathsf{ch})$ denotes a full transcript generated by the honest prover conditioned on the verifier sending the challenge $\mathsf{ch}$. Both experiments start with fresh oracle tables. The distinguisher is invoked only after the full transcript has been generated, receives access to the oracle used during its generation, and has no prior oracle-dependent state.*

*Proof sketch.* On input $(x, \mathsf{ch})$, let $I = \{i \mid \mathsf{ch}_i \neq 0\}$. The simulator samples $\mathsf{salt} \leftarrow\!\!\$\ \{0,1\}^{2\lambda}$ and $\mathsf{seeds}_{\mathsf{int}} \leftarrow\!\!\$\ \mathsf{SimulateSeeds}(L, I)$. It obtains the leaves outside $I$ as

$$\{\mathsf{seed}_i\}_{i \in [L] \setminus I} \leftarrow \mathsf{RecoverLeaves}^{\mathsf{H}_{\mathsf{tree}}}(\mathsf{salt}, \mathsf{seeds}_{\mathsf{int}}, L, I).$$

For $i \notin I$, it computes the traced first message $f_i = T_x\big(\mathsf{H}_{\mathsf{rnd}}(\mathsf{salt}, i, \mathsf{seed}_i)\big)$. For every $i \in I$, it invokes the HVZK simulator of $\Sigma$ on $(x, \mathsf{ch}_i)$ to obtain $(f_i, z_i)$. It outputs the transcript $(\mathbf{f}, \mathsf{ch}, (\mathsf{salt}, \mathsf{seeds}_{\mathsf{int}}, (z_i)_{i \in I}))$.

By Lemma 1 and the base protocol's HVZK, the coupled views agree unless a tree query guesses a hidden ancestor or a randomness query guesses a hidden leaf. These events cost at most $(Q_{\mathsf{tree}} + Q_{\mathsf{rnd}})2^{-\lambda} \leq Q2^{-\lambda}$. The full distribution comparison is in Appendix A.3. □

**Corollary 2.** *$\Sigma_{\mathsf{tree}}$ as defined above is Sigma protocol with high first-message min-entropy.*

*Proof.* The statement is a consequence of Theorem 1, Theorem 2 and Theorem 3. □

## 5 Compiling $\Sigma_{\mathsf{tree}}$ with Fiat-Shamir

In this section we prove our main result: the Fiat–Shamir compilation of $\Sigma_{\mathsf{tree}}$ is a straight-line extractable NIZKP for the original relation $\mathcal{R}$, with no change to the prover computation of the plain transform. The proof strategy generalizes the one used by Beullens et al. [9, Section 6]. The extraction error is governed by

**Transformation 2**: $\Pi_{\mathsf{FS}}$: Fiat-Shamir applied to $\Sigma_{\mathsf{tree}}$

Let $\Sigma_{\mathsf{tree}}$ be the seeded fixed-weight protocol obtained by applying Transformation 1 to a seed-friendly Sigma protocol $\Sigma$ for a binary relation $\mathcal{R}$. Let $L, \rho$ be the implicit parameters determining the number of parallel repetitions of $\Sigma$, hence the number of leaves of the tree, and the fixed weight.
We define $\Pi_{\mathsf{FS}} = (\mathsf{Prove}^{\mathsf{H}}_{\mathsf{FS}}, \mathsf{Verify}^{\mathsf{H}}_{\mathsf{FS}})$

$\mathsf{Prove}^{\mathsf{H}}_{\mathsf{FS}}(x, w)$:
1: $(\mathbf{f}, st) \leftarrow\!\$\ \mathsf{P}_1^{\mathsf{tree},\mathsf{H}}(x, w)$
2: $\mathsf{ch} \leftarrow \mathsf{H}_{\mathsf{FS}}(x, \mathbf{f})$
3: $\mathsf{resp} \leftarrow\!\$\ \mathsf{P}_2^{\mathsf{tree},\mathsf{H}}(x, w, st, \mathsf{ch})$
4: **return** $\pi \leftarrow (\mathsf{ch}, \mathsf{resp})$

$\mathsf{Verify}^{\mathsf{H}}_{\mathsf{FS}}(x, \pi = (\mathsf{ch}, \mathsf{resp}))$:
1: $\mathbf{f} \leftarrow F^{\mathsf{tree},\mathsf{H}}(x, \mathsf{ch}, \mathsf{resp})$
2: **return** accept iff $\mathbf{f} \neq \perp$, $\mathsf{V}_2^{\mathsf{tree},\mathsf{H}}(x, \mathbf{f}, \mathsf{ch}, \mathsf{resp}) = \mathsf{accept}$, and $\mathsf{ch} = \mathsf{H}_{\mathsf{FS}}(x, \mathbf{f})$

the *weight entropy* $\binom{L}{\rho}$, rather than the full cardinality $|\mathsf{ChSpace}^{L,\rho}|$ (Section 5.3).

The NIZKP $\Pi_{\mathsf{FS}} = (\mathsf{Prove}_{\mathsf{FS}}, \mathsf{Verify}_{\mathsf{FS}})$ is obtained by applying the standard Fiat–Shamir transform [30,52] for first-message-retrievable Sigma protocols to $\Sigma_{\mathsf{tree}}$. Its challenge is sampled uniformly as $\mathsf{ch} \leftarrow \mathsf{H}_{\mathsf{FS}}(x, \mathbf{f})$. The verifier reconstructs $\mathbf{f}$, checks the base protocol's acceptance conditions, and tests that the hash output equals the challenge in the proof.

*Remark 8 (No transform overhead).* A proof consists of a fixed-weight challenge, a salt, at most $N_{\mathsf{seed}}$ seed-tree labels, and $\rho$ full responses. The prover computes exactly $L$ first messages, and the verifier reconstructs them from the transmitted challenge and response. Thus, the direct Fiat–Shamir input $\mathbf{f}$ adds neither a proof field nor an additional hash computation. Concrete schemes that transmit a digest from which the challenge is derived are treated in Appendix C.

We now show that the NIZKP satisfies the completeness, the zero-knowledge, and the straight-line extraction properties formally defined in Definition 1.

Completeness is a direct consequence of the completeness of $\Sigma_{\mathsf{tree}}$, so we focus on the remaining properties: zero-knowledge and straight-line extraction.

### 5.1 Zero-Knowledge

In this section, we actually prove a stronger notion of zero-knowledge for NIZKP that enables the distinguisher to receive multiple real or simulated proofs, the *multi-proof zero knowledge* (also referred to as unbounded non-interactive zero knowledge in [29]).

**Definition 7** (Multi-Proof Zero-Knowledge)**.** *In the real experiment, a distinguisher has access to a random oracle* $\mathsf{H}$ *and to a prove oracle that, on* $(x, w) \in \mathcal{R}$*, returns* $\mathsf{Prove}^{\mathsf{H}}(x, w)$*. In the ideal experiment, it has access to the consistent simulated oracle produced by* $\mathsf{Sim}_0$ *and to an oracle returning* $\mathsf{Sim}_1(x)$*. We call* $\Pi$ multi-proof zero-knowledge *if the two experiments are computationally indistinguishable for every PPT distinguisher making polynomially many oracle and prove queries (*$Q_{\mathsf{H}}$ *and* $Q_P$*, respectively).*

The following simulator establishes multi-proof zero knowledge for $\Pi_{\mathsf{FS}}$. The simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ maintains lazy-sampled tables for the three oracles. On a proof request for $x$, it samples $\mathsf{ch} \leftarrow\!\$\ \mathsf{ChSpace}^{L,\rho}$ and runs the HVZK $\mathsf{Sim}^{\mathsf{tree}}(x, \mathsf{ch})$ from Theorem 3, on a tentative copy of the oracle tables, obtaining $(\mathbf{f}, \mathsf{ch}, \mathsf{resp})$. It reads $\mathsf{salt}$ from $\mathsf{resp}$. If that salt occurs in a prior tree- or randomness-oracle input, or if $(x, \mathbf{f})$ already has a Fiat–Shamir table entry, it discards all entries made during this attempt and retries with fresh $\mathsf{ch}$ and simulator randomness. Otherwise it commits the tentative entries, sets $\mathsf{H}_{\mathsf{FS}}(x, \mathbf{f}) = \mathsf{ch}$, and returns the compact proof $(\mathsf{ch}, \mathsf{resp})$.

This simulator is a retrying variant of the *canonical* simulator of [29, Theorem 1], with an additional salt-freshness check.

**Theorem 4** (Multi-Proof Zero-Knowledge)**.** *Let $\Sigma$ be a seed-friendly Sigma protocol that is complete and HVZK, and let a distinguisher make at most $Q_H$ hash queries and $Q_P$ adaptive proof queries. Put $Q_{\mathsf{zk}} = Q_H + (2L+1)Q_P$. The simulator above satisfies Definition 7 with distinguishing advantage at most*

$$\epsilon_{\mathsf{zk}}^{(m)} \le Q_P Q_{\mathsf{zk}} \left( \frac{1}{2^{\lambda}} + \frac{1}{2^{2\lambda}} + \frac{1}{2^{Lh}} \right). \tag{3}$$

*Its expected running time is polynomial.*

*Proof.* We follow the canonical-simulator argument of [29, Theorem 1] for $\Sigma_{\mathsf{tree}}$. Since $\Sigma_{\mathsf{tree}}$ itself uses random oracles, we additionally account for salt conflicts. When a new salt is absent from prior tree and randomness inputs, the HVZK bound of Theorem 3 applies after prior oracle interaction, and the conditional first-message min-entropy is at least $Lh$ by Theorem 1. First-message retrievability allows the simulator to return the compact proof $(\mathsf{ch}, \mathsf{resp})$. The detailed argument establishing (3) and the expected polynomial running time is given in Appendix A.4. □

Then the following corollary holds.

**Corollary 3** (Zero-Knowledge)**.** *Let $\Sigma$ be a seed-friendly Sigma protocol that is complete and HVZK, and let a distinguisher make at most $Q_H$ random oracle queries. Put $Q_{\mathsf{zk}} = Q_H + 2L + 1$. The simulator satisfies the zero-knowledge requirement of Definition 1 with distinguishing advantage at most*

$$\epsilon_{\mathsf{zk}}^{(1)} \le Q_{\mathsf{zk}} \left( \frac{1}{2^{\lambda}} + \frac{1}{2^{2\lambda}} + \frac{1}{2^{Lh}} \right). \tag{4}$$

*Its expected running time is polynomial.*

*Proof.* Apply Theorem 4 with $Q_P = 1$. □

### 5.2 Straight-Line Extraction

The extractor $\mathsf{Ext}_{\mathsf{FS}}$ receives a statement $x$, a proof $\pi$, and the adversary's ordered oracle transcript $Q_{\mathsf{H}}(\mathcal{A})$ (Definition 1). We will assume, without loss of generality,

**Algorithm 2**: Straight-line extractor $\mathsf{Ext}_{\mathsf{FS}}$ for $\Pi_{\mathsf{FS}}$

$\mathsf{Ext}_{\mathsf{FS}}(x, \pi = (\mathsf{ch}, \mathsf{resp}), Q_{\mathsf{H}}(\mathcal{A}))$:
1: **if** $\mathsf{Verify}^{\mathsf{H}}_{\mathsf{FS}}(x, \pi) = \mathsf{reject}$ **return** $\perp$
2: $I \leftarrow \{i \in [L] \mid \mathsf{ch}_i \neq 0\}$;
3: $\mathbf{f} \leftarrow F^{\mathsf{tree},\mathsf{H}}(x, \mathsf{ch}, \mathsf{resp})$
4: **for** $i \in I$ **do**
5: | **for each** entry $((\mathsf{rnd}, \mathsf{salt}', i, \mathsf{seed}), r) \in Q_{\mathsf{H}}(\mathcal{A})$, for any $\mathsf{salt}'$, **do**
6: | | $\widetilde{z} \leftarrow Z(x, r)$
7: | | **if** $F(x, 0, \widetilde{z}) = f_i$ and $\mathsf{V}_2(x, f_i, 0, \widetilde{z}) = \mathsf{accept}$ **then**
8: | | | **return** $w \leftarrow \mathsf{Ext}_{\Sigma}\big(x, (f_i, 0, \widetilde{z}), (f_i, \mathsf{ch}_i, z_i)\big)$
9: **return** $\perp$

that $\mathcal{A}$ performs all the random oracle queries needed to verify the proof, so these inputs and associated digests are included in the oracle transcript $Q_{\mathsf{H}}(\mathcal{A})$. Apart from the verifier's oracle evaluations in its initial acceptance check, $F$, $Z$, and the special-soundness extractor of $\Sigma$ are plain algorithms, and every randomness-oracle value used for extraction is read from the transcript.

*Extraction procedure overview.* The proofs of $\Pi_{\mathsf{FS}}$ include $L$ encoding of transcripts of $\Sigma$, of which $|I|$ are for non-zero challenges, while the other $L - |I|$ are for zero-challenges, and the responses must be encoded as seeds. The seeds included in the response are the same used to compute the associated first message components $f_i$. What we prove in our straight-line extraction theorem is that if a prover generates a valid proof, with overwhelming probability it has sent a query to $\mathsf{H}_{\mathsf{rnd}}$ with input $(\mathsf{salt}, i, \mathsf{seed}_i)$, which outputs the randomness $r_i$ used to produce the $i$-th component of $\mathbf{f}$. And importantly, these queries are performed before the prover sends the random oracle query $(x, \mathbf{f})$ to $\mathsf{H}_{\mathsf{FS}}$ to compute the Fiat-Shamir challenge, where $x$ is the proven statement. This guarantees that the extractor will be able to extract a witness from one of the transcripts of $\Sigma$ associated with the support of the challenge of $\Sigma_{\mathsf{tree}}$, and from the query to $\mathsf{H}_{\mathsf{rnd}}$ used to derive the randomness to compute it.

*Explained components of* $\mathbf{f}$ *at time* $t$. To formally present extraction strategy, the proof crucially requires a time-indexed notion of explanation. Consider the $t$-th distinct Fiat–Shamir query $(\mathsf{FS}, x, \mathbf{f})$, where $\mathbf{f} = (f_1, \ldots, f_L)$. A component $i$ of $\mathbf{f}$ is *explained at time* $t$ if at that time the list of queries made by $\mathcal{A}$ contains a randomness query $\mathsf{H}_{\mathsf{rnd}}(\mathsf{salt}', i, \mathsf{seed}) = r$, for any salt $\mathsf{salt}'$, such that $T_x(r) = f_i$ and $(f_i, 0, Z(x, r))$ is accepting. We call $\mathsf{H}_{\mathsf{rnd}}(\mathsf{salt}', i, \mathsf{seed}) = r$ an *explaining query* for the $i$-th component of $\mathbf{f}$: its returned random tape gives the extractor an accepting challenge-zero response for $f_i$, regardless of how the proof's seed tree was generated. The extractor and the explanation definition accept any salt; only the returned $r$ and its accepting zero response matter.

Let $E^{\mathbf{f}}_t$ be the explained components of $\mathbf{f}$ at time $t$ and $U^{\mathbf{f}}_t = [L] \setminus E^{\mathbf{f}}_t$. Since the vector is part of the Fiat–Shamir query, both sets are fixed before its answer is sampled. This timing is essential: after seeing the challenge, the adversary can issue randomness queries specifically for its zero positions, so a definition based on the final query list could make $U^{\mathbf{f}}_t$ depend on the challenge support, thereby

invalidating the support-matching argument we will make. A later explaining query at a nonzero position lets the extractor succeed; for a true statement, one at a zero position is charged to $\mathsf{bad}_{\mathsf{tr}}$ below.

**Theorem 5** (Straight-Line Extraction)**.** *Let $\Sigma$ be a seed-friendly Sigma protocol for $\mathcal{R}$ satisfying special soundness, and let $\rho \leq L$. For every adversary making at most $Q_H$ oracle queries, the extractor of Algorithm 2 satisfies*

$$\epsilon_{\mathsf{sle}} \leq 2\frac{Q_H + 1}{\binom{L}{\rho}} + \frac{(Q_H + 2L + 1)^2}{2^h}. \tag{5}$$

*Proof.* Whenever the extractor returns a value other than $\perp$ on an accepting proof, both base transcripts passed to $\mathsf{Ext}_\Sigma$ are accepting and have distinct challenges. Special soundness therefore guarantees that the returned value is a witness for $\mathcal{R}$. It suffices to bound the probability that verification accepts and the extractor returns $\perp$.

Let $Q_{\mathsf{FS}} \leq Q_H$ be the number of distinct Fiat–Shamir queries made by $\mathcal{A}$, and set $\widehat{Q} = Q_H + 2L + 1$ to account for all distinct oracle inputs first introduced while verifying its output: at most $L$ seed-tree expansions, $L$ randomness derivations, and one final Fiat–Shamir evaluation. Repeated evaluations during reconstruction and base verification reuse the same table entries.

Without loss of generality, we assume that before outputting a proof $\pi$ for $x$, the algorithm $\mathcal{A}$ performs the associated Fiat-Shamir random oracle query to $\mathsf{H}_{\mathsf{FS}}$. In fact, if this is not the case, we can define a wrapper algorithm $\mathcal{A}'$ that does it, and can perform $Q_{\mathsf{FS}} + 1$ queries to $\mathsf{H}_{\mathsf{FS}}$, as reported in the theorem's bound. We consider two cases: when the proof is for a false statement and when it is for a true statement.

**False statement** $x$**.** In this case, special soundness implies that each fixed base first message $f_i$ admits an accepting response for at most one challenge: two distinct accepting challenges would yield a witness for $x$. Consequently, a fixed vector $\mathbf{f}$ admits at most one accepting challenge vector, regardless of later oracle queries. A fresh Fiat–Shamir answer equals that vector with probability at most $|\mathsf{ChSpace}^{L,\rho}|^{-1} \leq \binom{L}{\rho}^{-1}$. A union bound on the queries to $\mathsf{H}_{\mathsf{FS}}$ yields that this happens with probability $\leq \frac{Q_{\mathsf{FS}}+1}{\binom{L}{\rho}}$.

**True statement** $x$**.** We expose the random oracles by lazy sampling. Each candidate Fiat–Shamir query fixes its statement and first-message vector before its answer is sampled. Define the event $\mathsf{bad}_{\mathsf{tr}}$ as follows. After $\mathcal{A}$ sends a candidate query $(\mathsf{FS}, x, \mathbf{f})$ for a true statement at time $t$, some new query $(\mathsf{rnd}, \mathsf{salt}, i, \mathsf{seed})$ has a fresh answer $r \leftarrow\!\!\$\ \mathcal{D}_{\mathsf{rnd}}$ such that $T_x(r) = f_i$ and $(f_i, 0, Z(x, r))$ is accepting for an unexplained position $i \in U_t^{\mathbf{f}}$. We consider the cases where $\mathsf{bad}_{\mathsf{tr}}$ happens and where it does not.

– $\underline{\mathsf{bad}_{\mathsf{tr}}\text{ happens}}$. An input queried before time $t$ cannot newly explain $i \in U_t^{\mathbf{f}}$: if its answer explained $f_i$, then $i \in E_t^{\mathbf{f}}$ already. A completely fresh input $(\mathsf{rnd}, \mathsf{salt}, i, \mathsf{seed})$ returns an independent sample from $\mathcal{D}_{\mathsf{rnd}}$, even if its salt appeared in earlier queries.

For true statements, $T_x(r)$ has the honest first-message distribution. Thus, for each ordered pair of a true candidate Fiat–Shamir query and a later randomness query, (2) bounds the chance that $T_x(r)$ equals the fixed $f_i$ by $2^{-h}$. Including verifier queries, there are at most $\widehat{Q}^2$ such pairs, and therefore

$$\Pr[\mathsf{bad}_{\mathsf{tr}}] \leq \widehat{Q}^2 2^{-h}.$$

Note that $\mathsf{bad}_{\mathsf{tr}}$ includes the case where the query refers to seeds actually generated from a seed tree and the salt $\mathsf{salt}$ used in $\pi$, but it is not limited to that. This informally states that a prover can explain a first message with a seed, after its generation, only with negligible probability. Therefore, if the proofs allows the verifier to produce a seed that explains the first message, the prover with overwhelming probability has sent a query to $\mathsf{H}_{\mathsf{rnd}}$ with input that seed.

– $\underline{\mathsf{bad}_{\mathsf{tr}} \text{ does not happen}}$. Suppose an accepting output proof $\pi$ for a true statement $x$ uses the $t$-th candidate Fiat–Shamir query $(x, \mathbf{f})$, whose answer is the challenge $\mathsf{ch}$ contained in the proof. Let $I = \{i : \mathsf{ch}_i \neq 0\}$. Outside $\mathsf{bad}_{\mathsf{tr}}$, we consider the following two cases:
  - $\underline{I \cap E_t^{\mathbf{f}} \neq \varnothing}$. In this case extraction succeeds. In fact, the corresponding randomness query belongs to $Q_{\mathsf{H}}(\mathcal{A})$, hence $\mathsf{Ext}_{\mathsf{FS}}$ can compute a witness for $x$ because it has both the nonzero-challenge transcript contained in $\pi$ and the challenge-zero transcript determined by the randomness query.
  - $\underline{I \cap E_t^{\mathbf{f}} = \varnothing}$. This implies $I \subseteq U_t^{\mathbf{f}}$. Suppose $i \in U_t^{\mathbf{f}} \setminus I$: acceptance supplies a revealed seed for position $i$ and verification obtains $r = \mathsf{H}_{\mathsf{rnd}}(\mathsf{salt}, i, \mathsf{seed})$ with $T_x(r) = f_i$. If this randomness query preceded time $t$, then $i \in E_t$, a contradiction. If it occurred later, $\mathsf{bad}_{\mathsf{tr}}$ occurs (another contradiction). Therefore, $U_t \subseteq I$, forcing $I = U_t$. Hence, outside $\mathsf{bad}_{\mathsf{tr}}$, acceptance and extraction failure imply $I = U_t$. Given the oracle transcript preceding the candidate query, $U_t$ is fixed and the challenge support is uniform, so the event $I = U_t$ has probability at most $\binom{L}{\rho}^{-1}$. A union bound over the $Q_{\mathsf{FS}} + 1$ queried candidates, tells us that whenever $\mathsf{bad}_{\mathsf{tr}}$ does not happen, acceptance of the proof and extraction failure happens with probability $\leq \frac{Q_{\mathsf{FS}}+1}{\binom{L}{\rho}}$.

Combining the false-statement bound, the probability of $\mathsf{bad}_{\mathsf{tr}}$, and the support bound for true statements gives

$$\epsilon_{\mathsf{sle}} \leq 2\frac{Q_{\mathsf{FS}} + 1}{\binom{L}{\rho}} + \frac{\widehat{Q}^2}{2^h}.$$

Substituting $Q_{\mathsf{FS}} \leq Q_H$ and $\widehat{Q} = Q_H + 2L + 1$ proves (5).

□

*Remark 9.* The support-matching step is where our analysis diverges from the binary-challenge setting. In [9] the base OR Sigma protocol has $\mathsf{ChSpace} = \{0, 1\}$, so $|\mathsf{ChSpace}^{L,\rho}| = \binom{L}{\rho}$ and this bound matches the full Fiat–Shamir

soundness error. For a $q$-ary seed-friendly Sigma protocol ($q = |\mathsf{ChSpace}| > 2$), an adversary that *knows a witness* can compute, for each component it plans to leave unexplained, a first message answerable at every nonzero challenge while remaining unable to answer 0, which requires exhibiting a seed. Since straight-line extraction must succeed even against provers that know a witness, the term $Q/\binom{L}{\rho}$ is unavoidable for this extractor: the $(q-1)^\rho$ factor of $|\mathsf{ChSpace}^{L,\rho}|$ contributes to soundness against witness-*less* provers, but not to extractability.

### 5.3 Discussion

*Parameter Implications.* Let $\kappa$ be the target extraction-security level and $\widehat{Q} = Q_H + 2L + 1$. The bound (5) is at most $2^{-\kappa}$ if

$$\log_2 \binom{L}{\rho} \geq \kappa + 2 + \log_2(Q_H + 1), \qquad h \geq \kappa + 1 + 2\log_2 \widehat{Q}.$$

Hence $\binom{L}{\rho} \geq 2^\lambda$ alone does not give extraction error $2^{-\lambda}$: the bound also depends on the number of queries and on the first-message min-entropy. Since the extractor is straight-line, this error enters a reduction additively, whereas forking-lemma analyses lose a square root in the advantage [8,52].

Plain Fiat–Shamir soundness depends on the number of challenges $\binom{L}{\rho}(q-1)^\rho$, whereas our extraction bound depends only on the number of supports $\binom{L}{\rho}$. For binary challenges the two numbers are equal, and our analysis removes the quadratic loss of the forking lemma. For $q > 2$, the nonzero challenge values do not improve our bound, so parameters chosen only for plain soundness may give a larger extraction error, and $L$ may need to be increased.

*Comparison with Proof-of-Work Transforms.* The Fischlin transform and its descendants (including GAO and SC-Coll-GAO [31,45,19,34]) achieve straight-line extraction by forcing the prover to evaluate the random oracle on multiple transcripts per first message. The prover searches for transcripts that satisfy a hash predicate. As a result, its running time varies between executions, and it fails with a completeness error that is a parameter of the transform, typically $2^{-40}$ [19,34]. These transforms also change the proof and the verification algorithm, so existing Fiat–Shamir schemes must be modified to use them. Our transform instead extracts from the queries already used to derive the prover's randomness. It adds neither a proof of work nor a new proof field, and leaves verification unchanged. In exchange, we require an explicit seeded-randomness interface, sufficient first-message min-entropy, and the condition on $\binom{L}{\rho}$ above. Moreover, our extraction result does not need unique responses, which Fischlin's transform and GAO require [31,34]. On the other hand, the extraction error of proof-of-work transforms improves with the number of nonzero challenge values, whereas ours does not. The two approaches can be combined, as discussed below.

*Open Problems.* Our analysis is classical: the extractor reads the adversary's query list. In the QROM, queries cannot be recorded in this way. Compressed-oracle results for commit-and-open protocols [26] may apply, but checking that their assumptions and bounds cover our extractor requires a separate proof. Since QROM analyses of Fiat–Shamir proofs of knowledge have larger losses than classical ones, a straight-line argument in the QROM would be especially useful for post-quantum schemes.

Unlike ours, the rewinding-based extractor for Fiat–Shamir [29] and the extractors for Fischlin [31] and the GAO family [34] improve when the challenge space contains several challenges besides $\tilde{\mathsf{ch}}$. It would be interesting to close this gap. Following the SC-Coll-GAO transform [34], one could add a proof of work on the $\rho$ transcripts in the support of the Fiat–Shamir challenge. This forces the prover to query the random oracle on additional transcripts, which increases the probability of extracting a witness through the special soundness of $\Sigma$. This change can be added on top of our analysis, and would allow a smaller $L$ at the cost of more prover work.

Our extraction notion is single-proof: the adversary does not receive honestly generated proofs, whose random-oracle queries would be hidden from the extractor. Extending the analysis to this multi-proof setting, as needed, for instance, for tight EUF-CMA reductions of the associated Fiat–Shamir signatures, is left to future work.

MPC-in-the-head and VOLE-in-the-head proofs [40,7] typically derive the randomness of each emulated party from salted seed trees. It is open whether our extractor extends to these multi-round protocols [53]. Finally, we leave to future work the application of our bounds to concrete schemes such as LESS [12], including the choice of $L$, $\rho$, and the seed lengths.

**Acknowledgments.** The first author is partially funded by SPRIND, the Federal Agency for Breakthrough Innovation, and Common Codes. The authors are solely responsible for the content of this publication; the positions presented here do not reflect the views of SPRIND and Common Codes. The second author acknowledges support from Ripple's University Blockchain Research Initiative and from the INDAM group GNSAGA.

## References

1. Abdalla, M., An, J.H., Bellare, M., Namprempre, C.: From identification to signatures via the fiat-shamir transform: Minimizing assumptions for security and forward-security. In: Advances in Cryptology—EUROCRYPT 2002: International Conference on the Theory and Applications of Cryptographic Techniques Amsterdam, The Netherlands, April 28–May 2, 2002 Proceedings 21. pp. 418–433. Springer (2002)
2. Alamati, N., De Feo, L., Montgomery, H., Patranabis, S.: Cryptographic group actions and applications. In: Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part II 26. pp. 411–439. Springer (2020)

3. Atapoor, S., Baghery, K., Cozzo, D., Pedersen, R.: CSI-SharK: CSI-FiSh with Sharing-friendly Keys, p. 471–502. Springer Nature Switzerland (2023). https://doi.org/10.1007/978-3-031-35486-1_21, http://dx.doi.org/10.1007/978-3-031-35486-1_21
4. Baldi, M., Barenghi, A., Beckwith, L., Biasse, J., Chou, T., Esser, A., Gaj, K., Karl, P., Mohajerani, K., Pelosi, G., Persichetti, E., Saarinen, M.O., Santini, P., Wallace, R., Zweydinger, F.: LESS — Linear Equivalence Signature Scheme. Tech. rep., National Institute of Standards and Technology (2025), round 2 specification, version 2.0, available at https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-2/spec-files/less-spec-round2-web.pdf
5. Battagliola, M., Borin, G., Di Crescenzo, G., Meneghetti, A., Persichetti, E.: Enhancing threshold group action signature schemes: Adaptive security and scalability improvements. In: International Conference on Post-Quantum Cryptography. pp. 129–161. Springer (2025)
6. Battagliola, M., Longo, R., Pintore, F., Signorini, E., Tognolini, G.: A revision of cross security: Proofs and attacks for multi-round fiat–shamir signatures. Mediterranean Journal of Mathematics **22**(5) (Jul 2025). https://doi.org/10.1007/s00009-025-02882-7
7. Baum, C., Braun, L., Delpech de Saint Guilhem, C., Klooß, M., Orsini, E., Roy, L., Scholl, P.: Publicly verifiable zero-knowledge and post-quantum signatures from VOLE-in-the-head. In: Advances in Cryptology – CRYPTO 2023. LNCS, vol. 14085, pp. 581–615. Springer (2023). https://doi.org/10.1007/978-3-031-38554-4_19
8. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: Proceedings of the 13th ACM conference on Computer and communications security. pp. 390–399 (2006)
9. Beullens, W., Dobson, S., Katsumata, S., Lai, Y.F., Pintore, F.: Group signatures and more from isogenies and lattices: Generic, simple, and efficient. In: Designs, Codes and Cryptography,. pp. 91(6), 2141–2200. Springer (2023)
10. Beullens, W., Katsumata, S., Pintore, F.: Calamari and falafl: logarithmic (linkable) ring signatures from isogenies and lattices. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 464–492. Springer (2020)
11. Beullens, W., Kleinjung, T., Vercauteren, F.: Csi-fish: efficient isogeny based signatures through class group computations. In: International conference on the theory and application of cryptology and information security. pp. 227–247. Springer (2019)
12. Biasse, J.F., Micheli, G., Persichetti, E., Santini, P.: Less is more: code-based signatures without syndromes. In: Progress in Cryptology-AFRICACRYPT 2020: 12th International Conference on Cryptology in Africa, Cairo, Egypt, July 20–22, 2020, Proceedings 12. pp. 45–65. Springer (2020)
13. Borin, G., Persichetti, E., Pintore, F., Reijnders, K., Santini, P.: A guide to the design of digital signatures based on cryptographic group actions: A guide to the design of digital signatures. J. Cryptol. **38**(3) (Jun 2025). https://doi.org/10.1007/s00145-025-09542-9
14. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: International Conference on Security in Communication Networks. pp. 268–289. Springer (2002)
15. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: Proceedings 42nd IEEE Symposium on Foundations of Computer Science. pp. 136–145. IEEE (2001)

16. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: Csidh: an efficient post-quantum commutative group action. In: Advances in Cryptology–ASIACRYPT 2018: 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2–6, 2018, Proceedings, Part III 24. pp. 395–427. Springer (2018)
17. Chairattana-Apirom, R., Harding, F., Lysyanskaya, A., Tessaro, S.: Server-aided anonymous credentials. In: Annual International Cryptology Conference. pp. 291–324. Springer (2025)
18. Chen, M., Dey, P., Ganesh, C., Mukherjee, P., Sarkar, P., Sasmal, S.: Universally composable non-interactive zero-knowledge from sigma protocols via a new straight-line compiler. In: IACR International Conference on Public-Key Cryptography. pp. 381–417. Springer (2025)
19. Chen, Y.H., Lindell, Y.: Optimizing and implementing fischlin's transform for uc-secure zero knowledge. IACR Communications in Cryptology (Jul 2024). https://doi.org/10.62056/a66chey6b
20. Chou, T., Niederhagen, R., Persichetti, E., Randrianarisoa, T.H., Reijnders, K., Samardjiska, S., Trimoska, M.: Take your meds: digital signatures from matrix code equivalence. In: International conference on cryptology in Africa. pp. 28–52. Springer (2023)
21. Dagdelen, Ö., Venturi, D.: A second look at fischlin's transformation. In: Progress in Cryptology–AFRICACRYPT 2014: 7th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 28-30, 2014. Proceedings 7. pp. 356–376. Springer (2014)
22. Damgård, I.: On $\sigma$-protocols. Lecture Notes, University of Aarhus, Department for Computer Science **84** (2002)
23. Dartois, P., Eriksen, J.K., Fouotsa, T.B., Herlédan Le Merdy, A., Invernizzi, R., Robert, D., Rueger, R., Vercauteren, F., Wesolowski, B.: PEGASIS: Practical Effective Class Group Action using 4-Dimensional Isogenies, p. 67–99. Springer Nature Switzerland (2025). https://doi.org/10.1007/978-3-032-01855-7_3, http://dx.doi.org/10.1007/978-3-032-01855-7_3
24. Doerner, J., Kondi, Y., Lee, E., Shelat, A.: Secure two-party threshold ecdsa from ecdsa assumptions. In: 2018 IEEE Symposium on Security and Privacy (SP). pp. 980–997. IEEE (2018)
25. Doerner, J., Kondi, Y., Lee, E., Shelat, A., Tyner, L.: Threshold bbs+ signatures for distributed anonymous credential issuance. In: 2023 IEEE Symposium on Security and Privacy (SP). pp. 773–789. IEEE (2023)
26. Don, J., Fehr, S., Majenz, C., Schaffner, C.: Efficient nizks and signatures from commit-and-open protocols in the qrom. In: Annual International Cryptology Conference. pp. 729–757. Springer (2022)
27. Ducas, L., van Woerden, W.: On the lattice isomorphism problem, quadratic forms, remarkable lattices, and cryptography. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 643–673. Springer (2022)
28. D'Alconzo, G., Flamini, A., Meneghetti, A., Signorini, E.: A framework for group action-based multi-signatures and applications to less, meds, and alteq. In: Public-Key Cryptography – PKC 2025. p. 99–133. Springer Nature Switzerland (2025). https://doi.org/10.1007/978-3-031-91823-0_4
29. Faust, S., Kohlweiss, M., Marson, G.A., Venturi, D.: On the non-malleability of the fiat-shamir transform. In: International Conference on Cryptology in India. pp. 60–79. Springer (2012)

30. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Conference on the theory and application of cryptographic techniques. pp. 186–194. Springer (1986)
31. Fischlin, M.: Communication-efficient non-interactive proofs of knowledge with online extractors. In: Annual International Cryptology Conference. pp. 152–168. Springer (2005)
32. Flamini, A., Friedrichs, K., Katz, J., Ladd, W., Lehmann, A., Sefranek, M.: Issuer-hiding anonymous tokens and credentials from key-randomizable signatures. Cryptology ePrint Archive, Paper 2026/870 (2026), https://eprint.iacr.org/2026/870
33. Flamini, A., Lee, E., Lysyanskaya, A.: Multi-holder anonymous credentials from bbs signatures. In: Advances in Cryptology – CRYPTO 2025. p. 325–357. Springer Nature Switzerland (2025). https://doi.org/10.1007/978-3-032-01887-8_11
34. Flamini, A., Pintore, F., Signorini, E., Tognolini, G.: New straight-line extractable NIZKPs for cryptographic group actions. In: Advances in Cryptology – CRYPTO 2026. Springer (2026)
35. Fujisaki, E., Suzuki, K.: Traceable ring signature. In: International Workshop on Public Key Cryptography. pp. 181–200. Springer (2007)
36. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. Journal of the ACM (JACM) **38**(3), 690–728 (1991)
37. Hanzlik, L., Lai, Y.F., Mula, M., Paracucchi, E., Slamanig, D., Tang, G.: Tanuki: New frameworks for (concurrently secure) blind signatures from post-quantum groups actions. Cryptology ePrint Archive, Paper 2025/1100 (2025), https://eprint.iacr.org/2025/1100
38. Hanzlik, L., Lai, Y.F., Paracucchi, E., Persichetti, E.: Wombat: Post-quantum blind signature from standard group action assumptions and more. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 277–307. Springer (2026)
39. Hesse, J., Singh, N., Sorniotti, A.: How to bind anonymous credentials to humans. In: 32nd USENIX Security Symposium (USENIX Security 23). pp. 3047–3064 (2023)
40. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC). pp. 21–30. ACM (2007). https://doi.org/10.1145/1250790.1250794
41. Ji, Z., Qiao, Y., Song, F., Yun, A.: General linear group action on tensors: A candidate for post-quantum cryptography. In: Theory of cryptography conference. pp. 251–281. Springer (2019)
42. Katsumata, S.: A new simple technique to bootstrap various lattice zero-knowledge proofs to qrom secure nizks. In: Annual International Cryptology Conference. pp. 580–610. Springer (2021)
43. Katsumata, S., Reichle, M., Sakai, Y.: Practical round-optimal blind signatures in the rom from standard assumptions. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 383–417. Springer (2023)
44. Klooß, M., Reichle, M., Wagner, B.: Practical blind signatures in pairing-free groups. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 363–395. Springer (2024)
45. Kondi, Y., Shelat, A.: Improved straight-line extraction in the random oracle model with applications to signature aggregation. In: International Conference on the

Theory and Application of Cryptology and Information Security. pp. 279–309. Springer (2022)
46. Lindell, Y.: Simple three-round multiparty schnorr signing with full simulatability. IACR Communications in Cryptology (Apr 2024). https://doi.org/10.62056/a36c0l5vt
47. Lysyanskaya, A., Rosenbloom, L.N.: Universally composable $\sigma$-protocols in the global random-oracle model. In: Theory of Cryptography Conference. pp. 203–233. Springer (2022)
48. Lyubashevsky, V.: Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In: International conference on the theory and application of cryptology and information security. pp. 598–616. Springer (2009)
49. Orrù, M.: Revisiting keyed-verification anonymous credentials. In: Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security. pp. 1188–1199 (2025)
50. Pass, R.: On deniability in the common reference string and random oracle model. In: Advances in Cryptology-CRYPTO 2003: 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003. Proceedings 23. pp. 316–337. Springer (2003)
51. Pointcheval, D., Stern, J.: Provably secure blind signature schemes. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 252–265. Springer (1996)
52. Pointcheval, D., Stern, J.: Security proofs for signature schemes. In: International conference on the theory and applications of cryptographic techniques. pp. 387–398. Springer (1996)
53. Rotem, L., Tessaro, S.: Straight-line knowledge extraction for multi-round protocols. In: Annual International Cryptology Conference. pp. 95–127. Springer (2025)
54. Schnorr, C.P.: Efficient signature generation by smart cards. Journal of cryptology **4**, 161–174 (1991)
55. Shoup, V., Gennaro, R.: Securing threshold cryptosystems against chosen ciphertext attack. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 1–16. Springer (1998)
56. Tang, G., Duong, D.H., Joux, A., Plantard, T., Qiao, Y., Susilo, W.: Practical post-quantum signature schemes from isomorphism problems of trilinear forms. In: Annual international conference on the theory and applications of cryptographic techniques. pp. 582–612. Springer (2022)
57. Unruh, D.: Non-interactive zero-knowledge proofs in the quantum random oracle model. In: Advances in Cryptology-EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II 34. pp. 755–784. Springer (2015)

# A Deferred Proofs

## A.1 Proof of the Seed-Tree Hiding Lemma

*Proof of Lemma 1.* Expansion queries use $\mathsf{H}_{\mathsf{tree}}(\mathsf{salt}, \mathsf{ad}, \mathsf{seed})$ with the fixed salt of Lemma 1.

As a proof artifact, we define the *border of* $I$ $\mathsf{Frontier}(I)$ as the set of nodes whose addresses are the ones in $\mathsf{seeds}_{\mathsf{int}}(I)$, together with the nodes in $I$.

Consider the following game hops:

- $\mathsf{Game}_0 = \mathsf{Real}_I$.
- $\mathsf{Game}_1$: sample a random root $\mathsf{seed}_{\mathsf{root}} \leftarrow\!\!\$\ \{0,1\}^\lambda$, but instead of building the whole seed tree, expand only the labels of the nodes above $\mathsf{Frontier}(I)$ (their ancestors). Then return $(\mathsf{seeds}_{\mathsf{int}}, (\mathsf{seed}_i)_{i \in I})$. This game is indistinguishable from $\mathsf{Game}_0$ as later adversarial queries, including expansions at frontier nodes $\mathsf{Frontier}(I)$ are answered lazily.
- $\mathsf{Game}_2$: In this game, instead of computing the subtree whose nodes are the ancestors of the nodes in $\mathsf{Frontier}(I)$ starting from a root $\mathsf{seed}_{\mathsf{root}}$, we sample uniformly at random all the nodes belonging to this subtree, and we program the random oracle so that the random oracle relation between descendants and ancestors holds. Note that here it is crucial that the random oracle queries' inputs, together with the uniformly sampled label, contain the address of the node in question, whose descendants' labels we are determining. This guarantees that all the random oracle queries that the experiment must program will have distinct inputs, and therefore the outputs are uniformly distributed and independent.
  The oracle table starts empty, and the experiment programs these inputs before invoking the distinguisher. Since the addresses make the inputs distinct, the programmed answers have the same joint distribution as the honest expansion answers in $\mathsf{Game}_1$. Later queries are answered consistently from the same table, so these two games have the same distribution.
- $\mathsf{Game}_3$: We modify the game by non-programming the random oracle on the inputs specifying the relation among the labels we have sampled uniformly at random. The distribution of the nodes of the subtree is the same as they are still chosen uniformly at random, but since the inputs associated to the subtree internal nodes are not programmed anymore, if the distinguisher sends a query corresponding to a specific address and the right label, the experiment will output a random digest in $\{0,1\}^{2\lambda}$ instead of the two children of the internal node. Define $\mathsf{bad}$ in $\mathsf{Game}_3$ as a query that specifies the address and label of a hidden ancestor. With those ancestor entries unprogrammed, their labels are uniform and independent of the distinguisher's view and query sequence. Each adaptive query fixes one address and candidate label, so an unconditional union bound over at most $Q_{\mathsf{tree}}$ queries gives $\Pr[\mathsf{bad}] \leq Q_{\mathsf{tree}} 2^{-\lambda}$.
- $\mathsf{Game}_4 = \mathsf{Ideal}_I$. The difference from $\mathsf{Game}_3$ is that the experiment does not sample the internal nodes of the subtree, only the leaves. The distinguisher has no way to distinguish the two experiments, as the distribution of the elements given to the distinguisher is the same, and the RO is not programmed.

Combining the above game hops, the fundamental lemma of game playing now bounds the distinguishing advantage by

$$|\Pr[\mathcal{A}(\mathsf{Real}_I) = 1] - \Pr[\mathcal{A}(\mathsf{Ideal}_I) = 1]| \leq \Pr[\mathsf{bad}] \leq \frac{Q_{\mathsf{tree}}}{2^\lambda}.$$

Therefore, the advantage of the distinguisher in distinguishing $\mathsf{Real}_I$ from $\mathsf{Ideal}_I$ is at most $\frac{Q_{\mathsf{tree}}}{2^\lambda}$.

□

### A.2 Random-Tape Independence

*Proof of Corollary 1.* The leaf seeds depend only on $\mathsf{seed}_{\mathsf{root}}$ and $\mathsf{H}_{\mathsf{tree}}$, so they are independent of $\mathsf{H}_{\mathsf{rnd}}$. Condition on any leaf vector $(\mathsf{seed}_1, \ldots, \mathsf{seed}_L) = (s_1, \ldots, s_L)$ having positive probability. Under this conditioning, $\mathsf{H}_{\mathsf{rnd}}$ remains a random oracle. Its inputs $(\mathsf{salt}, i, s_i)$ are pairwise distinct because they contain different indices, even if some leaf seeds coincide. Hence, for every vector $(t_1, \ldots, t_L)$ in the support of $\mathcal{D}_{\mathsf{rnd}}^L$,

$$\begin{aligned}&\Pr[(r_1, \ldots, r_L) = (t_1, \ldots, t_L) \mid (\mathsf{seed}_1, \ldots, \mathsf{seed}_L) = (s_1, \ldots, s_L)] \\ &\quad = \prod_{i=1}^{L} \Pr_{r \leftarrow\$ \mathcal{D}_{\mathsf{rnd}}}[r = t_i].\end{aligned}$$

The right-hand side does not depend on the conditioned leaf vector. Averaging over that vector proves the claim. □

### A.3 HVZK of $\Sigma_{\mathsf{tree}}$

*Proof of Theorem 3.* Use the simulator described in the main-body proof sketch and write $I = \{i : \mathsf{ch}_i \neq 0\}$. It remains to compare distributions. Couple the honest seed tree to the ideal cover and hidden leaves of Lemma 1, using lazy-sampled oracle tables. In the real experiment, the tree-oracle answers at strict ancestors of the released cover and hidden leaves are fixed by the honest prover. In the ideal coupling, those hidden-ancestor relations are omitted, and expansions from the released cover remain consistent in both experiments.

Conditioned on the seed tree, the honest random tapes at $(\mathsf{salt}, i, \mathsf{seed}_i)$ are independent $\mathcal{D}_{\mathsf{rnd}}$ samples, since the randomness oracle $\mathsf{H}_{\mathsf{rnd}}$ is independent of the tree oracle $\mathsf{H}_{\mathsf{tree}}$ and the inputs are distinct by position. At a zero position, let $f_i = \mathsf{P}_1(x, w; r_i)$. Property 3 makes $(f_i, 0, Z(x, r_i))$ accepting, and first-message retrievability gives

$$f_i = F(x, 0, Z(x, r_i)) = T_x(r_i).$$

Thus, the honest prover and simulator compute the same first message from the recovered cover at zero positions. At nonzero positions, exact HVZK of $\Sigma$ lets us replace each honest pair $(f_i, z_i)$ by an independently simulated pair with the same distribution. The simulator leaves the randomness-oracle entry at each hidden leaf unprogrammed (as it does not know the associated randomness); this difference is visible only if the distinguisher queries that input.

Define $\mathsf{bad}$ in the ideal coupled view:

- a tree query hits the label of a hidden ancestor at its specified address (i.e. the substitution of the real seed tree generation is distinguished by the ideal generation according to Lemma 1, which happens with probability $\leq \frac{Q_{\mathsf{tree}}}{2^\lambda}$ by Lemma 1);
- a randomness query hits the label of a hidden leaf at its specified position, which happens with probability at most $\frac{Q_{\mathsf{rnd}}}{2^\lambda}$.

Let $Q_{\mathsf{tree}}$ and $Q_{\mathsf{rnd}}$ be the numbers of queries of each type. A union bound gives

$$\Pr[\mathsf{bad}] \leq 2^{-\lambda} \left(Q_{\mathsf{tree}} + Q_{\mathsf{rnd}}\right) \leq Q 2^{-\lambda}.$$

Until $\mathsf{bad}$ occurs, the coupled transcript and oracle views agree. This proves the claim.

□

### A.4 Proof of Multi-Proof Zero-Knowledge

*Proof of Theorem 4.* Order the proof requests chronologically. For $0 \leq k \leq Q_P$, let $\mathsf{Game}_k$ answer the first $k$ proof requests with the HVZK transcript simulator followed by Fiat–Shamir programming as specified above and all remaining requests honestly, using one shared lazy-sampled oracle. Thus $\mathsf{Game}_0$ is the real experiment and $\mathsf{Game}_{Q_P}$ is the ideal one. To compare $\mathsf{Game}_{k-1}$ and $\mathsf{Game}_k$, expose the Fiat–Shamir value $\mathsf{ch}$ at the $k$-th proof's eventual oracle input. If that input is fresh, sampling $\mathsf{ch}$ before generating an honest $\Sigma_{\mathsf{tree}}$ transcript conditioned on $\mathsf{ch}$, and programming it afterward, is identical to lazy sampling it during honest proof generation.

Tree and randomness queries explicitly contain $\mathsf{salt}$; direct Fiat–Shamir queries do not.

**Salt-dependent queries** The salt in every attempted transcript is uniform on $\{0,1\}^{2\lambda}$. Before the $k$-th request, there are at most $Q_{\mathsf{zk}}$ prior tree and randomness inputs, including the queries made while generating earlier proofs. Irrespective of their remaining fields and adaptive choice, the new salt equals the salt in one of these inputs with probability at most $Q_{\mathsf{zk}}/2^{2\lambda}$.

**Fiat-Shamir queries** Condition on the prior oracle transcript and on the new salt being absent from its tree- and randomness-oracle inputs. All salt-dependent inputs in this attempt are then fresh relative to that transcript. Each simulated component first message has the honest marginal distribution: the base HVZK simulator has exactly the honest transcript distribution at nonzero positions, and the randomness oracle supplies independent $\mathcal{D}_{\mathsf{rnd}}$ samples at zero positions. Thus $\mathbf{f}$ has the product distribution bounded in Theorem 1, independently of the prior Fiat–Shamir table. A union bound over its at most $Q_{\mathsf{zk}}$ entries gives programming-conflict probability at most $Q_{\mathsf{zk}} 2^{-Lh}$.

Each tentative attempt is therefore rejected with probability at most

$$\delta := Q_{\mathsf{zk}} \left( \frac{1}{2^{2\lambda}} + \frac{1}{2^{Lh}} \right).$$

A rejected attempt is not a simulation failure: its tentative entries are discarded, so they do not enlarge the shared tables. Every retry starts from the same tables with fresh randomness. Since $\delta$ is negligible for polynomial query bounds, the expected number of attempts per request is at most $(1-\delta)^{-1}$, giving expected polynomial running time.

For the distinguishing argument, couple each simulated request to a version that aborts on its first conflicting attempt. The views agree unless a first attempt is rejected, whose probability over all $Q_P$ requests is at most

$$Q_P\delta = Q_P Q_{\mathsf{zk}} \left( \frac{1}{2^{2\lambda}} + \frac{1}{2^{Lh}} \right).$$

Conditioned on no conflict, all prior tree and randomness entries have salts different from the salt of the $k$-th proof and are therefore independent auxiliary information for the single-proof hybrid. The transition from $\mathsf{Game}_{k-1}$ to $\mathsf{Game}_k$ only replaces the $k$-th honest $\Sigma_{\mathsf{tree}}$ transcript conditioned on $\mathsf{ch}$ by $\mathsf{Sim}^{\mathsf{tree}}(x, \mathsf{ch})$. By Theorem 3, this changes the view by at most $Q_{\mathsf{zk}} 2^{-\lambda}$. All later adaptive interaction is part of the distinguishing environment, and the query budget includes both its external queries and the oracle evaluations made while generating the other proofs. The bound, therefore, applies to the complete shared-oracle experiment. Summing the $Q_P$ adjacent-hybrid bounds and charging the first-attempt conflict events gives (3). □

# B Cryptographic Group Actions

## B.1 Group Actions and Hardness Assumptions

**Definition 8** (Group Action)**.** *A group $(G, \cdot, e)$, with $e$ the identity element, is said to act on a set $X$ if there exists a map*

$$\begin{aligned} \star : G \times X &\longrightarrow X \\ (g, x) &\longmapsto g \star x \end{aligned}$$

*such that $e \star x = x$ for every $x \in X$ and $g \star (h \star x) = (g \cdot h) \star x$ for every $x \in X$ and $g, h \in G$. In this case, we say that the triple $(G, X, \star)$ is a group action.*

To build cryptographic schemes, only group actions $(G, X, \star)$ for which $G$ and $X$ are both finite are considered. In addition, it must be possible to efficiently perform some operations, such as: (1) computing the group operation and the inverse of a group element; (2) performing (uniform) random sampling in $X$ and $G$, (3) deciding equality and validity of a representation of elements in $X$; (4) computing the group action $\star$. Group actions for which these operations are efficient are called *effective group actions* [2].

Besides efficiency requirements, group actions for cryptographic applications must satisfy security properties. In particular, a group action must be based on one-way functions, or equivalently-hard problems. An effective group action that satisfies this further requirement is called *cryptographic group action*. A (typically hard) problem for group actions is the Group Action Inversion Problem (GAIP), defined below.

**Definition 9** (Group Action Inversion Problem (GAIP))**.** *The GAIP on a group action $(G, X, \star)$ is defined as follows. Let $x_0 \in X$. Given as input a uniformly-random element $x_1 \in \{g \star x_0 | g \in G\}$, output a group element $g \in G$ such that $x_1 = g \star x_0$.*

Another well-known problem is the Stabilizer Computation Problem (SCP), defined as follows.

**Definition 10** (Stabilizer Computation Problem (SCP))**.** *On input a uniformly-random $x_0 \in X$, the SCP over a group action $(G, X, \star)$ consists in finding an element $g \in G \setminus \{e\}$ such that $g \star x_0 = x_0$.*

Examples of effective group actions for which these problems are believed to be hard are based on isogenies [16][7], tensors [41,56] and linear codes [20,12].

### B.2 The Sigma Protocol $\Sigma_\star$

The Sigma protocol from a cryptographic group action, which we denote by $\Sigma_\star$, is defined as follows.

**Definition 11** (The Sigma Protocol $\Sigma_\star$)**.** *Let $(G, X, \star)$ be a cryptographic group action and let*

$$\mathcal{R}_\star = \left\{ \big((x_0, x_1), g\big) \in X^2 \times G \;\middle|\; x_1 = g \star x_0 \right\}$$

*be the binary relation whose hardness is the GAIP on the orbit of $x_0$. The Sigma protocol $\Sigma_\star$ for $\mathcal{R}_\star$ has challenge space $\mathsf{ChSpace} = \{0, 1\}$ and the following three-move flow. On input a statement $(x_0, x_1)$ and a witness $g$, the prover runs $\mathsf{P}_1$ to sample $\tilde{g} \leftarrow\!\!\$\ G$ and sends the first message $f \leftarrow \tilde{g} \star x_0$. The verifier runs $\mathsf{V}_1$ to sample a challenge $\mathsf{ch} \leftarrow\!\!\$\ \{0, 1\}$. The prover then runs $\mathsf{P}_2$ to compute the response $z \leftarrow \tilde{g}\, g^{-\mathsf{ch}}$, and the verifier accepts if and only if $\mathsf{V}_2$ finds $f = z \star x_{\mathsf{ch}}$.*

The protocol is first-message retrievable through $F\big((x_0, x_1), \mathsf{ch}, z\big) = z \star x_{\mathsf{ch}}$, and the random tape of $\mathsf{P}_1$ is the sampled group element itself, so $r = \tilde{g}$ and the response to the challenge $\mathsf{ch} = 0$ is the public function $Z\big((x_0, x_1), r\big) = r$.

*Seed-Friendliness of $\Sigma_\star$.* It can be easily proved that $\Sigma_\star$ is complete, special sound and honest-verifier zero knowledge, as per Section 2. The simulator $\mathsf{Sim}$, used to prove the HVZK property, takes in input a statement $(x_0, x_1)$ and a challenge $\mathsf{ch} \in \{0, 1\}$, samples uniformly at random a group element $z \in G$, computes the first message $f = z \star x_{\mathsf{ch}}$ and returns the transcript $(f, \mathsf{ch}, z)$. The witness extractor $\mathsf{Ext}$ considered to prove the special-soundness property takes two transcripts $(f, 0, z_0)$ and $(f, 1, z_1)$ for the same statement $(x_0, x_1)$, and outputs a witness $w = z_1^{-1} z_0$.

When we instantiate the Sigma protocol $\Sigma_\star$ described in Definition 11 using a group action $(G, X, \star)$ for which SCP is hard, then $\Sigma_\star$ has unique responses. In fact, given a first message $f$ and a challenge $\mathsf{ch}$, it is hard for any prover to compute two distinct valid responses.

For a fixed valid statement $(x_0, x_1)$, a uniform group element maps $x_0$ to a uniform first message in the orbit $G \star x_0$. Its min-entropy is therefore $\log_2 |G \star x_0|$.

[7] For the original CSIDH group action, only the action of group elements with a specific form can be effectively computed.

If every permitted orbit has size at least $2^h$, for a common $h = \omega(\log \lambda)$, then $\Sigma_\star$ has high first-message min-entropy. Under this condition, it is seed-friendly because it is first-message retrievable, has constant challenge space and $\widetilde{\mathsf{ch}} = 0$ is the special challenge with public response $Z((x_0, x_1), r) = r$ (see Definition 4).

## C Digest-Encoded Variants

The generic construction in Sections 4 and 5 feeds the complete recovered first-message vector directly to the Fiat–Shamir oracle. This is the cleanest formulation when first messages are retrievable: the proof transmits the challenge, and no separate digest is needed. Some concrete protocols instead expose a fixed-length digest. We record here how this encoding changes the construction and its analysis.

### C.1 A Separate Digest Layer

Let $\mathsf{H}_{\mathsf{com}} : \{0,1\}^* \to \{0,1\}^{2\lambda}$ be an independent random oracle, using its own domain-separation tag, and define

$$\mathsf{com} \leftarrow \mathsf{H}_{\mathsf{com}}(\mathsf{salt}, \mathbf{f}), \qquad \mathsf{ch} \leftarrow \mathsf{H}_{\mathsf{FS}}(x, \mathsf{salt}, \mathsf{com}).$$

In this variant, the Fiat–Shamir oracle takes $(x, \mathsf{salt}, \mathsf{com})$ as input instead of the direct construction's $(x, \mathbf{f})$; its output remains uniform in $\mathsf{ChSpace}^{L,\rho}$. The interactive first message is $(\mathsf{salt}, \mathsf{com})$, and verification reconstructs $\widetilde{\mathbf{f}}$ from $(\mathsf{ch}, \mathsf{resp})$ and checks $\mathsf{com} = \mathsf{H}_{\mathsf{com}}(\mathsf{salt}, \widetilde{\mathbf{f}})$. The Fiat–Shamir proof may contain either $\mathsf{com}$ or $\mathsf{ch}$ together with $\mathsf{salt}$ and $\mathsf{resp}$.

This encoding is no longer exactly special sound. Given two accepting transcripts with the same $(\mathsf{salt}, \mathsf{com})$ and distinct challenges, the extractor reconstructs $\mathbf{f}$ and $\mathbf{f}'$. If they agree, the proof of Theorem 2 returns a witness; otherwise, the two vectors give a collision

$$\mathsf{H}_{\mathsf{com}}(\mathsf{salt}, \mathbf{f}) = \mathsf{H}_{\mathsf{com}}(\mathsf{salt}, \mathbf{f}').$$

Thus special soundness holds up to commitment collisions.

The straight-line proofs transfer by conditioning on the absence of the following global event: either two distinct commitment inputs receive the same digest, or a Fiat–Shamir query containing $\mathsf{com}$ occurs before any commitment query producing $\mathsf{com}$ and a later commitment query does produce it. If at most $\widehat{Q}_{\mathsf{com}}$ relevant oracle evaluations occur, lazy sampling and a union bound give

$$\Pr[\mathsf{Bad}_{\mathsf{com}}] \le \frac{\widehat{Q}_{\mathsf{com}}^2}{2^{2\lambda-1}}.$$

Outside this event, every accepting Fiat–Shamir query determines a unique vector $\mathbf{f}$ before its answer is sampled, so the proof of Theorem 5 applies unchanged. With $\widehat{Q}_{\mathsf{com}} = Q_H + 3L + 2$, the single-proof extraction bound becomes

$$\epsilon^{(1)}_{\mathsf{oe},\mathsf{com}} \le \frac{Q_H + 1}{\binom{L}{\rho}} + \frac{(Q_H + 3L + 2)^2}{2^h} + \frac{(Q_H + 3L + 2)^2}{2^{2\lambda-1}}.$$

The HVZK bound is unchanged. Multi-proof zero knowledge uses $Q_{\mathsf{zk}}^{\mathsf{com}} = Q_H + (3L+2)Q_P$ in (3); its programming-conflict term remains unchanged.

### C.2 LESS-Style Recovered Digests

The digest layer above is a useful generic adapter, but it is not the literal computation specified by LESS. In the Round-2 specification [4], the signer first computes the recoverable round values $B^{(1)}, \ldots, B^{(t)}$ and then sets

$$\mathsf{cmt} \leftarrow \mathsf{Hash}(B^{(1)} \| \cdots \| B^{(t)} \| \mathsf{salt} \| \mathsf{msg}), \qquad \mathsf{ch} \leftarrow \mathsf{SampleChallenge}(\mathsf{cmt}).$$

The signature contains $\mathsf{cmt}$. The verifier derives $\mathsf{ch}$ from it, reconstructs the $B^{(i)}$, and checks the same hash equation. Hence $\mathsf{cmt}$ is the encoded Fiat–Shamir output, not an intermediate digest that is fed to a second independent Fiat–Shamir hash. A concrete instantiation should therefore analyze the composite hash-and-sampling operation, including the exact distribution of $\mathsf{SampleChallenge}$, rather than identify LESS with the two-oracle adapter above. The direct construction in the main text captures its recoverability structure; this paragraph records the serialization difference needed for an exact LESS instantiation.