SmallWood: Hash-Based Polynomial Commitments and Zero-Knowledge Arguments for Relatively Small Instances

Thibauld Feneuil and Matthieu Rivain

CryptoExperts, Paris, France
thibauld.feneuil@cryptoexperts.com
matthieu.rivain@cryptoexperts.com

Abstract. Zero-knowledge proofs (ZKPs) are a fundamental building block in cryptography, enabling powerful privacy-preserving and verifiable computations. In the post-quantum era, hash-based ZKPs have emerged as a promising direction due to their conjectured resistance to quantum attacks, along with their simplicity and efficiency.
In this work, we introduce SmallWood, a hash-based polynomial commitment scheme (PCS) and zero-knowledge argument system optimized for relatively small instances. Building on the recent degree-enforcing commitment scheme (DECS) from the Threshold-Computation-in-the-Head (TCitH) framework, we refine its formalization and combine it with techniques from Brakedown. This results in a new hash-based PCS that is particularly efficient for polynomials of relatively small degree –typically up to 2^{16}– outperforming existing approaches in this range.
Leveraging this new PCS, we design a hash-based zero-knowledge argument system that outperforms the state-of-the-art in terms of proof sizes for witness sizes ranging from 2^6 to 2^{16}. Additionally, we present exact zero-knowledge arguments for lattice-based problems using SmallWood, demonstrating highly competitive performance: our scheme yields proof sizes under 25 KB across a wide range of lattice parameters, including Kyber and Dilithium instances.

1 Introduction

Zero-knowledge proofs (ZKPs) or arguments are fundamental cryptographic tools that allow a prover to convince a verifier of the validity of a statement without revealing any additional information beyond its truth. They serve as essential building blocks in applications ranging from privacy-preserving authentication to blockchain scalability and secure multiparty computation (MPC). As the cryptographic community prepares for a post-quantum era, the development of post-quantum secure ZKPs has become increasingly important. Among the most conservative and robust approaches to post-quantum proof systems are those based solely on symmetric cryptographic primitives, which offers strong security guarantees under minimal assumptions and thus provide a reliable fallback in post-quantum cryptography proposals. These include the so-called "MPC-in-the-Head" (MPCitH) paradigm [IKOS07] which has been efficiently instantiated and improved over the recent years, see e.g. [KKW18, BN20, AGH+23, BBD+23, FR25b], and hash-based SNARKs or STARKs combining error-correcting codes and Merkle trees, see e.g. [BBHR18a, BBHR19, BCR+19, GLS+23, ACFY24, ZCF24].

MPCitH-based zero-knowledge arguments using GGM seed trees, and in particular the recently proposed *VOLE-in-the-Head* (VOLEitH) [BBD+23] and *Threshold-Computation-in-the-Head* (TCitH) [FR25b] frameworks, provides the smallest proofs for statements involving small witnesses. These protocols are particularly well-suited for proving knowledge of preimages of one-way functions, an archetypal example of a compact witness scenario encountered in the design of post-quantum signature schemes (see, e.g., [BBB+25, BBFR25]). Beyond these small instances, larger –yet still relatively small– instances arise in several important cryptographic contexts, including privacy-enhancing technologies, actively secure MPC. While these applications may not require asymptotically succinct proofs as in SNARKs, they nonetheless call for efficient schemes that yield compact proofs in this intermediate regime.

Most existing hash-based polynomial commitment schemes and zero-knowledge argument systems focus on achieving asymptotic succinctness, see e.g. [BBHR18a, BBHR19, BCR+19, GLS+23, ACFY24, ZCF24].

In contrast, only a few proposals provide compact proofs in intermediate regime where the witness size typically ranges from 2^6 to 2^{16}. Notable examples include Ligero [AHIV17, AHIV23] and the Merkle-tree variant of the TCitH framework [FR25b].

In the present work, we revisit the challenge of constructing efficient hash-based polynomial commitment schemes (PCS) and zero-knowledge arguments tailored to "relatively small instances". By this, we mean that the size of the *extended witness* (or committed polynomial) typically ranges from 2^6 to 2^{16} field elements.[1] We introduce SmallWood,[2] a new PCS and zero-knowledge argument scheme inspired by techniques from Ligero [AHIV17, AHIV23], Brakedown [GLS+23], and Threshold-Computation-in-the-Head (TCitH) [FR25b]. At the core of our approach is the recent notion of a *degree-enforcing commitment scheme* (DECS), introduced in [FR25b], which we formalize as a (binding and zero-knowledge) *small-domain PCS*. Building on this foundation and adapting techniques from Brakedown, we develop a full polynomial commitment scheme, SmallWood-PCS, and a zero-knowledge argument system, SmallWood-ARK. For the latter, we introduce the notion of *parallel and aggregated constraint systems* (PACS) –a generalization of of LPPC systems[3] from TCitH and Ligero-style statements– and design an efficient *polynomial interactive oracle proof* (PIOP) tailored to such statements. We formally establish the soundness and zero-knowledge properties of both SmallWood-PCS and SmallWood-ARK in the random oracle model. The resulting schemes set a new benchmark among hash-based zero-knowledge proof systems, achieving smaller proof sizes for witness lengths ranging from 2^6 to 2^{16}.

We showcase SmallWood-ARK in two core application domains: arithmetic circuits and lattice-based statements. Compared to existing schemes such as Ligero, TCitH-Π_{Ligero}, and VOLE-in-the-Head, SmallWood-ARK achieves smaller proof sizes, especially over large fields and for high-degree constraints. For lattice-based statements –including ISIS, LWE, and subset-sum as a special case– we introduce a novel compression technique for lattice witnesses that enables more compact proofs in some settings (specifically low infinity norm). On benchmark lattice instances from prior literature, SmallWood-ARK outperforms all known MPC-in-the-Head/hash-based schemes [BN20, Beu20, FMRV22, FR25b] and provides proof sizes competitive with specialized lattice-based protocols [BLS19, LNS21, LNP22]. Applied to standard cryptographic primitives like Kyber [ABD+21] and Dilithium [BDK+21], SmallWood-ARK yields proof sizes between 14–16 KB and 17–23 KB, respectively, which are, to the best of our knowledge, the smallest sizes reported in the state of the art for these statements. This notably surpasses the recent results of [BCGZ24], for which we additionally identify several technical flaws.

SmallWood has potential for a wide range of further applications involving zero-knowledge proofs for relatively small instances, *e.g.*, in privacy-enhancing technologies. In particular, it underpins the CAPSS framework [FR25a], a follow-up work that provides SNARK-friendly post-quantum signatures, aggregated signatures, and anonymous credentials.

2 Preliminaries

2.1 Polynomials

Along this paper, we shall call $\boldsymbol{P} = (P_1, \ldots, P_n) \in (\mathbb{F}[X])^n$ a *vector polynomial*. Its degree is defined as $\deg \boldsymbol{P} = (\deg P_1, \ldots, \deg P_n) \in \mathbb{N}^n$ and is said to be lower than or equal to $\boldsymbol{d} = (d_1, \ldots, d_n) \in \mathbb{N}^n$ if $\deg P_i \leq d_i$ for every $i \in [1, n]$. The set of polynomials with coefficient from $\mathbb{F}$ and degree at most d shall be denoted $\mathbb{F}[X]^{(\leq d)}$.

[1] Here, the term *extended witness* refers to the full variable assignment of the computation being proven (e.g. inputs and outputs of multiplication gates in an arithmetic circuit). The structure of the extended witness depend on the specific constraint system that is used by the proof system (e.g. R1CS, or PACS in our context) and its size is proportional to the complexity of the computation being proven.

[2] Wood because the scheme is based on hashes and trees, and Small because it targets smaller witness sizes compared to asymptotically succinct hash-based proof systems (e.g. SNARKs or STARKs).

[3] The LPPC terminology stands for *(global) linear and parallel polynomial constraints*, introduced in [FR25b] as a generalization of Ligero's statements.

The evaluation of a vector polynomial $\boldsymbol{P}$ in a point $e \in \mathbb{F}$ is defined as the vector $\boldsymbol{P}(e) = (P_1(e), \ldots, P_n(e)) \in \mathbb{F}^n$. For any set $E \subseteq \mathbb{F}$, we denote $\boldsymbol{P}|_E$ the set of evaluations of $\boldsymbol{P}$ over E, that is $\boldsymbol{P}|_E := \{\boldsymbol{P}(e)\}_{e \in E}$. We further denote, $\boldsymbol{P} = \mathsf{Interpol}(E, \boldsymbol{P}|_E)$ the interpolation which returns the lowest-degree vector polynomial $\boldsymbol{P}$ matching $\boldsymbol{P}|_E$.

2.2 Interactive Commitment Schemes

In this paper, we consider transparent interactive commitment schemes. The transparency feature means that no (trusted) setup is required by those schemes. The scheme parameters which vary according to the security level are left implicit in the exposition. The interactive feature means that those schemes are composed of *public-coin verifier* (PCV) interactive protocol, namely interactive protocols between a stateful (PPT) prover and a stateful public-coin (PPT) verifier. In such a protocol Π, the prover $\Pi.\mathcal{P}$ and the verifier $\Pi.\mathcal{V}$ exchange messages. The protocol might terminate with either $\Pi.\mathcal{P}$ or $\Pi.\mathcal{V}$ producing the final message. All the messages from $\Pi.\mathcal{V}$ are fresh uniform random values (over some space depending on the definition of the protocol) except for the final message of the protocol which is a string from $\{\textsc{Accept}, \textsc{Reject}\}$ (when coming from $\Pi.\mathcal{V}$). We denote

$$(out_{\mathcal{P}}, out_{\mathcal{V}}, \pi) \leftarrow \Pi(in_{\mathcal{P}}, in_{\mathcal{V}})$$

for running Π with $in_{\mathcal{P}}$ and $in_{\mathcal{V}}$ as input of $\Pi.\mathcal{P}$ and $\Pi.\mathcal{V}$ respectively, getting $out_{\mathcal{P}}$ and $out_{\mathcal{V}}$ as output of $\Pi.\mathcal{P}$ and $\Pi.\mathcal{V}$ respectively, and π as protocol transcript. In addition, we shall consider PCV protocols in the *random oracle model* (ROM) where the prover and verifier have access to a random oracle $\mathcal{H}$. In that case, we denote $\Pi^{\mathcal{H}}$ a protocol where both parties have access to $\mathcal{H}$.

We introduce below the notion of a Polynomial Commitment Scheme (PCS) used throughout this work. A PCS allows a prover to commit to one or more polynomials and later open selected evaluations in a verifiable way, while keeping all other evaluations hidden.

Definition 1 (Polynomial Commitment Scheme). *Let $\mathcal{H}$ be a random oracle. Let $n \in \mathbb{N}$ and $\boldsymbol{d} = (d_1, \ldots, d_n) \in \mathbb{N}^n$. Let $\mathbb{F}$ a finite field. A* polynomial commitment scheme *with parameters $(\mathbb{F}, n, \boldsymbol{d})$ in the ROM is a pair of PCV protocols $(\mathsf{Commit}^{\mathcal{H}}, \mathsf{Eval}^{\mathcal{H}})$:*

- $\mathsf{Commit}^{\mathcal{H}}$*: In this protocol, $\mathsf{Commit}.\mathcal{P}$ takes a vector polynomial $\boldsymbol{P} \in \mathbb{F}[X]^{(\leq d_1)} \times \cdots \times \mathbb{F}[X]^{(\leq d_n)}$ as input and produces an opening key key as output while $\mathsf{Commit}.\mathcal{V}$ takes no input and produces no output. The transcript of the protocol is the commitment denoted com. We abuse notations and denote*

$$(\mathsf{com}, \mathsf{key}) \leftarrow \mathsf{Commit}^{\mathcal{H}}(\boldsymbol{P})$$

  to mean $(\mathsf{key}, \bot, \mathsf{com}) := (out_{\mathcal{P}}, out_{\mathcal{V}}, \pi) \leftarrow \mathsf{Commit}^{\mathcal{H}}(in_{\mathcal{P}} := \boldsymbol{P}, in_{\mathcal{V}} := \bot)$.
- $\mathsf{Eval}^{\mathcal{H}}$*: In this protocol, $\mathsf{Eval}.\mathcal{P}$ takes an opening key key as input and $\mathsf{Eval}.\mathcal{V}$ takes a commitment com as input. Both parties additionally take a set of evaluation points $E \subseteq \mathbb{F}$ and a set of evaluations $\boldsymbol{P}|_E := \{\boldsymbol{P}(e)\}_{e \in E}$. The protocol ends with the verifier returning $out_{\mathcal{V}} \in \{\textsc{Accept}, \textsc{Reject}\}$. We abuse notations and denote*

$$(out_{\mathcal{V}}, \pi) \leftarrow \mathsf{Eval}^{\mathcal{H}}(\mathsf{key}, \mathsf{com}, E, \boldsymbol{P}|_E)$$

  to mean $(\bot, out_{\mathcal{V}}, \pi) \leftarrow \mathsf{Eval}^{\mathcal{H}(\cdot)}\big(in_{\mathcal{P}} := (\mathsf{key}, E, \boldsymbol{P}|_E), in_{\mathcal{V}} := (\mathsf{com}, E, \boldsymbol{P}|_E)\big)$, sometimes skipping π from the output if only $out_{\mathcal{V}}$ is relevant.

The scheme is

- correct: *for any vector polynomial $\boldsymbol{P} = (P_1, \ldots, P_n) \in \mathbb{F}[X]^{(\leq d_1)} \times \cdots \times \mathbb{F}[X]^{(\leq d_n)}$ and any set of evaluation points $E \subseteq \mathbb{F}$, we have:*

$$\Pr\left[out = \textsc{Accept} \;\middle|\; \begin{array}{l} (\mathsf{com}, \mathsf{key}) \leftarrow \mathsf{Commit}^{\mathcal{H}}(\boldsymbol{P}) \\ out \leftarrow \mathsf{Eval}^{\mathcal{H}}(\mathsf{key}, \mathsf{com}, E, \boldsymbol{P}|_E) \end{array}\right] = 1 \ ,$$

  where $\boldsymbol{P}|_E := \{\boldsymbol{P}(e)\}_{e \in E}$.

– (t, ε)-extractable binding*: there exists a polynomial-time algorithm $\mathcal{E}$ (the* extractor*) which given a set $\mathcal{Q}$ of query-response pairs to $\mathcal{H}$ and a commitment outputs a degree-$\boldsymbol{d}$ polynomial $\boldsymbol{P}$ such that, for any stateful adversary $\mathcal{A}$ running in time t and playing the role of the prover in the above protocols, we have:*

$$\Pr\left[\begin{array}{l|l} out = \textsc{Accept} \wedge & (\mathsf{com}, *) \leftarrow \mathsf{Commit}^{\mathcal{H},\mathcal{A}^{\mathcal{H}}}(*) \\ \exists e \in E,\ \boldsymbol{P}(e) \neq \boldsymbol{p}_e & \boldsymbol{P} \leftarrow \mathcal{E}(\mathcal{Q}, \mathsf{com}) \\ & \big(E, \{\boldsymbol{p}_e\}_{e\in E}\big) \leftarrow \mathcal{A}^{\mathcal{H}}() \\ & out \leftarrow \mathsf{Eval}^{\mathcal{H},\mathcal{A}^{\mathcal{H}}}\big(*, \mathsf{com}, E, \{\boldsymbol{p}_e\}_{e\in E}\big) \end{array}\right] \leq \varepsilon\ ,$$

where $\mathsf{Commit}^{\mathcal{H},\mathcal{A}^{\mathcal{H}}}$ *and* $\mathsf{Eval}^{\mathcal{H},\mathcal{A}^{\mathcal{H}}}$ *denote the protocols where the prover is played by $\mathcal{A}$ with access to the random oracle $\mathcal{H}$, $*$ denote arbitrary inputs/outputs of the adversary which do not affect the definition, and $\mathcal{Q}$ denotes the set of query-response pairs to $\mathcal{H}$ during the execution of the* $\mathsf{Commit}^{\mathcal{H},\mathcal{A}^{\mathcal{H}}}$ *protocol.*

– (t, ξ)-honest-verifier zero knowledge ((t, ξ)-HVZK)*: there exists a PPT algorithm* Sim *(the* simulator*), with ability to program $\mathcal{H}$, such that for any stateful adversary $\mathcal{A}$ running in time t, for any vector polynomial $\boldsymbol{P} = (P_1, \ldots, P_n) \in \mathbb{F}[X]^{(\leq d_1)} \times \cdots \times \mathbb{F}[X]^{(\leq d_n)}$ and for any set of evaluation points $E \subseteq \mathbb{F}$, we have:*

$$\Pr\left[\hat{b} = b \;\middle|\; \begin{array}{l} (\mathsf{com}^{(0)}, \mathsf{key}) \leftarrow \mathsf{Commit}^{\mathcal{H}}(\boldsymbol{P}) \\ (out, \pi^{(0)}) \leftarrow \mathsf{Eval}^{\mathcal{H}}(\mathsf{key}, \mathsf{com}^{(0)}, E, \boldsymbol{P}|_E) \\ (\mathsf{com}^{(1)}, \pi^{(1)}) \leftarrow \mathsf{Sim}(E, \boldsymbol{P}|_E) \\ b \leftarrow \{0, 1\} \\ \hat{b} \leftarrow \mathcal{A}^{\mathcal{H}}(\mathsf{com}^{(b)}, \pi^{(b)}) \end{array}\right] \leq \frac{1}{2} + \xi\ .$$

We now introduce the notion of a Linear-Map Vector Commitment Scheme (LVCS) that we use in our work. While the notion of PCS is well-established, the notion of LVCS was recently introduced in [CNR+22]. The definition provided below adapts the formalization from that work to the transparent and interactive setting with extractability in the random oracle model.

Definition 2 (Linear-Map Vector Commitment Scheme). *Let $\mathcal{H}$ be a random oracle. Let n_{rows}, n_{cols} be some integers. Let $\mathbb{F}$ a finite field. A* linear-map vector commitment scheme *with parameters $(\mathbb{F}, n_{rows}, n_{cols})$ in the ROM is a pair of PCV protocols $(\mathsf{Commit}^{\mathcal{H}}, \mathsf{Eval}^{\mathcal{H}})$:*

– $\mathsf{Commit}^{\mathcal{H}}$*: In this protocol,* $\mathsf{Commit}.\mathcal{P}$ *takes n_{rows} vectors $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{n_{rows}} \in \mathbb{F}^{n_{cols}}$ as input and produces an opening key* key *as output while* $\mathsf{Commit}.\mathcal{V}$ *takes no input and produces no output. The transcript of the protocol is the commitment denoted* com*. We abuse notations and denote*

$$(\mathsf{com}, \mathsf{key}) \leftarrow \mathsf{Commit}^{\mathcal{H}}(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{n_{rows}})$$

to mean $(\mathsf{key}, \perp, \mathsf{com}) := (out_{\mathcal{P}}, out_{\mathcal{V}}, \pi) \leftarrow \mathsf{Commit}^{\mathcal{H}}(in_{\mathcal{P}} := (\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{n_{rows}}), in_{\mathcal{V}} := \perp)$.

– $\mathsf{Eval}^{\mathcal{H}}$*: In this protocol,* $\mathsf{Eval}.\mathcal{P}$ *takes an opening key* key *as input and* $\mathsf{Eval}.\mathcal{V}$ *takes a commitment* com *as input. For some $m \in \mathbb{N}$, both parties additionally take a set of coefficients $\mathcal{C} := \{c_{k,j}\}$ with $1 \leq k \leq m$ and $1 \leq j \leq n_{rows}$, and a set of vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m \in \mathbb{F}^{n_{cols}}$. The protocol ends with the verifier returning $out_{\mathcal{V}} \in \{\textsc{Accept}, \textsc{Reject}\}$. We abuse notations and denote*

$$(out_{\mathcal{V}}, \pi) \leftarrow \mathsf{Eval}^{\mathcal{H}}(\mathsf{key}, \mathsf{com}, \mathcal{C}, (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m))$$

to mean $(\perp, out_{\mathcal{V}}, \pi) \leftarrow \mathsf{Eval}\big(in_{\mathcal{P}} := (\mathsf{key}, \mathcal{C}, (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)), in_{\mathcal{V}} := (\mathsf{com}, \mathcal{C}, (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m))\big)$*, sometimes skipping π from the output if only $out_{\mathcal{V}}$ is relevant.*

The scheme is

– correct: *for any vectors* $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{n_{rows}} \in \mathbb{F}^{n_{cols}}$ *and any set of coefficients* $\mathcal{C} := \{c_{k,j}\}$*, we have:*

$$\Pr\left[out = \textsc{Accept} \;\middle|\; \begin{array}{l} (\mathsf{com}, \mathsf{key}) \leftarrow \mathsf{Commit}^{\mathcal{H}}(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{n_{rows}}) \\ out \leftarrow \mathsf{Eval}^{\mathcal{H}}(\mathsf{key}, \mathsf{com}, \mathcal{C}, (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)) \end{array}\right] = 1 \ ,$$

where $\boldsymbol{v}_k := \sum_{j=1}^{n_{rows}} c_{k,j} \cdot \boldsymbol{r}_j$ *for every* $k \in [1, m]$*.*

– (t, ε)-extractable binding*: there exists a polynomial-time algorithm* $\mathcal{E}$ *(the* extractor*) which, given a set* $\mathcal{Q}$ *of query-response pairs to* $\mathcal{H}$ *and a commitment, outputs* n_{rows} *vectors* $(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{n_{rows}})$ *such that, for any stateful adversary* $\mathcal{A}$ *running in time* t *and playing the role of the prover in the above protocols, we have:*

$$\Pr\left[\begin{array}{c} out = \textsc{Accept} \wedge \\ \exists k,\ \sum_j c_{k,j} \cdot \boldsymbol{r}_j \neq \boldsymbol{v}_k \end{array} \;\middle|\; \begin{array}{l} (\mathsf{com}, *) \leftarrow \mathsf{Commit}^{\mathcal{H}, \mathcal{A}^{\mathcal{H}}}(*) \\ (\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{n_{rows}}) \leftarrow \mathcal{E}(\mathcal{Q}, \mathsf{com}) \\ \big(\mathcal{C} := \{c_{k,j}\}, (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)\big) \leftarrow \mathcal{A}^{\mathcal{H}}() \\ out \leftarrow \mathsf{Eval}^{\mathcal{H}, \mathcal{A}^{\mathcal{H}}}\big(*, \mathsf{com}, \mathcal{C}, (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)\big) \end{array}\right] \leq \ \varepsilon \ ,$$

where $\mathsf{Commit}^{\mathcal{H}, \mathcal{A}^{\mathcal{H}}}$ *and* $\mathsf{Eval}^{\mathcal{H}, \mathcal{A}^{\mathcal{H}}}$ *denote the protocols where the prover is played by* $\mathcal{A}$ *with access to the random oracle* $\mathcal{H}$*,* $*$ *denote arbitrary inputs/outputs of the adversary which do not affect the definition, and* $\mathcal{Q}$ *denotes the set of query-response pairs to* $\mathcal{H}$ *during the execution of the* $\mathsf{Commit}^{\mathcal{H}, \mathcal{A}^{\mathcal{H}}}$ *protocol.*

– (t, ξ)-honest-verifier zero knowledge ((t, ξ)-HVZK)*: there exists a PPT algorithm* Sim *(the simulator) such that for any stateful adversary* $\mathcal{A}$ *running in time* t*, for any vectors* $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{n_{rows}} \in \mathbb{F}^{n_{cols}}$ *and any set of coefficients* $\mathcal{C} := \{c_{k,j}\}$ *and corresponding linear combination vectors* $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$*, we have:*

$$\Pr\left[\hat{b} = b \;\middle|\; \begin{array}{l} (\mathsf{com}^{(0)}, \mathsf{key}) \leftarrow \mathsf{Commit}(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{n_{rows}}) \\ (out, \pi^{(0)}) \leftarrow \mathsf{Eval}(\mathsf{key}, \mathsf{com}^{(0)}, \mathcal{C}, (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)) \\ (\mathsf{com}^{(1)}, \pi^{(1)}) \leftarrow \mathsf{Sim}(\mathcal{C}, (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)) \\ b \leftarrow \{0, 1\} \\ \hat{b} \leftarrow \mathcal{A}(\mathsf{com}^{(b)}, \pi^{(b)}) \end{array}\right] \leq \ \frac{1}{2} + \xi \ .$$

2.3 Non-Interactive Zero-Knowledge Arguments

We formally define *non-interactive zero-knowledge arguments of knowledge* (NIZK-ARK) in the next definition. Our definition restricts to transparent NIZK-ARK schemes which do not require a trusted setup and we keep the setup implicit of the sake of simplicity. Moreover, we focus on NIZK-ARK achieving straightline-extractability in the random oracle model.

Definition 3 (NIZK-ARK). *Let* $\mathcal{H}$ *be a random oracle. A* non-interactive zero-knowledge argument of knowledge *for a family of relations* $\mathcal{R}$ *in the ROM is defined by the following algorithms:*

– $\mathsf{Prove}^{\mathcal{H}} : (w \mid R, x) \mapsto \pi$*. Given a witness* w*, a relation* $R \in \mathcal{R}$ *and a statement* x*, such that* $(x, w) \in R$*, outputs a proof* π*.*

– $\mathsf{Verify}^{\mathcal{H}} : (\pi \mid R, x) \mapsto 1/0$*. Given a proof* π*, a relation* $R \in \mathcal{R}$ *and a statement* x*, outputs* 1 *("accept") or* 0 *("reject").*

The algorithm Prove *is probabilistic while the algorithm* Verify *is deterministic. The scheme is:*

– correct*: for any relation* $R \in \mathcal{R}$ *and statement-witness pair* $(x, w) \in R$*, we have:*

$$\Pr\Big[\mathsf{Verify}^{\mathcal{H}}(\pi \mid R, x) = 1 \;\Big|\; \pi \leftarrow \mathsf{Prove}^{\mathcal{H}}(w \mid R, x)\Big] = 1 \ .$$

– (t, ε)-straightline-extractable knowledge sound*: there exists a PPT algorithm* Ext *which for any relation* $R \in \mathcal{R}$*, statement-witness pair* $(x, w) \in R$*, and adversary* $\mathcal{A}$ *running in time* t*, verifies:*

$$\Pr\left[\begin{array}{l|l} \mathsf{Verify}^{\mathcal{H}}(\pi \mid R, x) = 1 & \pi \leftarrow \mathcal{A}^{\mathcal{H}}(R, x) \\ \wedge\ (x, w) \notin R & w \leftarrow \mathsf{Ext}(\mathcal{Q}, \pi) \end{array}\right] \leq \varepsilon\ ,$$

where $\mathcal{Q}$ *denotes the set of query-response pairs made by* $\mathcal{A}$ *to* $\mathcal{H}$*.*

– (t, ξ)-zero knowledge*: there exists a PPT algorithm* Sim *which outputs a simulated proof indistinguishable from a genuine proof. Specifically, for any relation* $R \in \mathcal{R}$*, statement-witness pair* $(x, w) \in R$*, and adversary* $\mathcal{A}$ *running in time* t*, we have:*

$$\Pr\left[\hat{b} = b \;\middle|\; \begin{array}{l} \pi^{(0)} \leftarrow \mathsf{Prove}^{\mathcal{H}}(w \mid R, x) \\ \pi^{(1)} \leftarrow \mathsf{Sim}() \\ b \leftarrow \{0, 1\} \\ \hat{b} \leftarrow \mathcal{A}^{\mathcal{H}}(\pi^{(b)}) \end{array}\right] \leq \xi\ .$$

3 Degree Enforcing Commitment Scheme

A common approach to defining a *somewhat* polynomial commitment scheme from hash functions is to rely on Merkle trees. The idea is to use a Merkle tree-based vector commitment to commit to the evaluations of a (vector) polynomial $\boldsymbol{P}$ on an evaluation domain $\mathbb{E}$. However, such a somewhat PCS has two limitations. First, it does not allow proving the evaluation of $\boldsymbol{P}$ for any point $e \in \mathbb{F}$, but for $e \in \mathbb{E}$ only. Second, a malicious prover could commit to evaluations that do not correspond to a polynomial of the right degree. This second limitation can be addressed by equipping the commitment with a protocol that statistically proves the low degree of the committed polynomial or its proximity to a low-degree polynomial. Several protocols have been proposed to target asymptotic efficiency, such as Ligero [AHIV17, AHIV23] or FRI [BBHR18a, BGKS20]. A more recent work has proposed the notion of *degree-enforcing commitment scheme* (DECS) [FR25b] which achieves promising results for the "relatively small" regime. This scheme guaranties to the verifier that the committed evaluations are of degree at most d.

While the notion of DECS introduced in [FR25b] keeps the binding property implicit for the committed evaluations, we note here that combining the notion of degree enforcing from [FR25b] together with the binding of committed evaluations yields a standard PCS (as per Definition 1) with main difference that here only evaluations on a small domain $\mathbb{E}$ can be opened. We formalize this notion as a *small-domain PCS* in the next definition.[4]

Definition 4 (Small-Domain Polynomial Commitment Scheme). *Let* $N, n \in \mathbb{N}$*,* $\boldsymbol{d} = (d_1, \ldots, d_n) \in \mathbb{N}^n$ *and* $\varepsilon \in [0, 1)$*. Let* $\mathbb{F}$ *a finite field and* $\mathbb{E} = \{e_1, \ldots, e_N\} \subseteq \mathbb{F}$ *(the* evaluation domain*). An interactive* ε-small-domain polynomial commitment scheme *(SD-PCS) with parameters* $(\mathbb{F}, \mathbb{E}, n, \boldsymbol{d})$ *is a polynomial commitment scheme as defined in Definition 1, except that the set of evaluation points* E *in input of the* Eval *protocol must be a subset of the evaluation domain* $\mathbb{E}$*.*

We recall the degree-enforcing commitment scheme from [FR25b] in Figure 1, while additionally incorporating masking techniques to achieve zero-knowledge (the original construction was not zero-knowledge). This scheme restricts the vector degree parameter to be of the form $\boldsymbol{d} = (d, \ldots, d)$ for some $d \in \mathbb{N}$. The parameter $\eta \in \mathbb{N}$ is chosen to achieve the desired level of binding soundness. Our description further relies on a general distribution $\mathcal{D}_\Gamma$ for the challenge which may differ from the uniform distribution.

The following theorem establishes the binding soundness of the DECS construction in the random oracle model. Its proof is a direct adaptation of the security analysis from [FR25b]. Recall that in the original work,

[4] Our definition can be thought of as a more general and more formal definition of the degree-enforcing commitment scheme from [FR25b] which further integrates the binding of evaluations (with extractability in the ROM), considers general Commit and Eval protocols (whereas the original definition assumed fixed number of rounds), and further includes the zero-knowledge property.

DECS.Commit:

On input a vector polynomial $\boldsymbol{P} \in \left(\mathbb{F}[X]^{(\leq d_{\mathsf{decs}})}\right)^{n_{\mathsf{decs}}}$ for the prover, DECS.Commit consists of the following 3-pass protocol:

1. $\mathcal{P}$ samples uniformly at random η degree-d_{decs} polynomials $\boldsymbol{M} = (M_1, \ldots, M_\eta) \leftarrow (\mathbb{F}[X]^{(\leq d_{\mathsf{decs}})})^\eta$ and N random hash commitment tapes $(\rho_1, \ldots, \rho_N) \leftarrow (\{0,1\}^\lambda)^N$. $\mathcal{P}$ computes $u_j = \mathsf{Hash}(\boldsymbol{P}(e_j), \boldsymbol{M}(e_j), j, \rho_j)$ for every $e_j \in \mathbb{E}$ and $\mathsf{root} = \mathsf{MerkleRoot}(u_1, \ldots, u_N)$. $\mathcal{P}$ sends root to $\mathcal{V}$.
2. $\mathcal{V}$ samples a random matrix $\Gamma = (\gamma_{k,i})_{k,i} \leftarrow \mathcal{D}_\Gamma$ and sends it to $\mathcal{P}$.
3. $\mathcal{P}$ computes the polynomials

$$R_k(X) = M_k(X) + \sum_{i=1}^{n_{\mathsf{decs}}} \gamma_{k,i} \cdot P_i(X) \quad \text{for all } k \in [1, \eta]$$

   and sends the vector polynomial $\boldsymbol{R} := (R_1, \ldots R_\eta)$ to $\boldsymbol{V}$.

$\mathcal{P}$ returns $\mathsf{key} := (\boldsymbol{P}, \boldsymbol{M})$ while the commitment (i.e., the protocol transcript) is defined as $\mathsf{com} := (\mathsf{root}, \Gamma, \boldsymbol{R})$.

DECS.Eval:

The prover takes as input an opening key $\mathsf{key} = (\boldsymbol{P}, \boldsymbol{M})$, and the verifier takes as input a commitment $\mathsf{com} = (\mathsf{root}, \Gamma, \boldsymbol{R})$. Both parties additionally take a set of evaluation points $E \subseteq \mathbb{E}$ and a set of evaluations $\boldsymbol{P}|_E := \{\boldsymbol{P}(e)\}_{e \in E}$. The evaluation protocol works as follows:

1. $\mathcal{P}$ sends the evaluations $\boldsymbol{M}|_E$, the random tapes $\{\rho_j \; ; \; e_j \in E\}$ and the proof π_{MT} made of the authentication paths from the leaves $\{u_j\}_{e_j \in E}$ to the Merkle root root.
2. $\mathcal{V}$ performs the following checks:
   - For all j s.t. $e_j \in E$, verify the authentication path for $u_j = \mathsf{Hash}(\boldsymbol{P}(e_j), \boldsymbol{M}(e_j), j, \rho_j)$ in π_{MT} w.r.t. root.
   - For all $e \in E$ and all $k \in [1, \eta]$, verify the equality $R_k(e) = M_k(e) + \sum_{i=1}^{n_{\mathsf{decs}}} \gamma_{k,i} \cdot P_i(e)$.
   - For all $k \in [1, \eta]$, verify the degree $\deg R_k \leq d$.

Fig. 1: Degree-enforcing commitment scheme (DECS) from [FR25b]. This is a zero-knowledge small-domain PCS with parameters $(\mathbb{F}, \mathbb{E}, n_{\mathsf{decs}}, \boldsymbol{d})$ with $\boldsymbol{d} = (d_{\mathsf{decs}}, \ldots, d_{\mathsf{decs}})$ for $d_{\mathsf{decs}} \in \mathbb{N}$. Additional parameters: the number of degree-enforcing test repetitions $\eta \in \mathbb{N}$ and the challenge distribution $\mathcal{D}_\Gamma$ (a probability distribution over $\mathbb{F}^{\eta \times n_{\mathsf{decs}}}$).

the DECS scheme was only shown to satisfy an *ad hoc* security notion referred to as *degree-enforcing*. The theorem below shows that this notion can be naturally reformulated as a standard extractable binding notion, thereby formalizing DECS as a small-domain polynomial commitment scheme under Definitions 1 and 4. Moreover, for any bounded number Q of random oracle queries, this soundness guarantee is information-theoretic: regardless of the running time of the adversary $\mathcal{A}$, any violation of the binding property can only occur with a bounded probability.

Theorem 1. *In the ROM, modelling* Hash *as a random oracle, the DECS protocol depicted in Figure 1 is a* $(t_{\mathcal{A}}, \varepsilon_{\mathsf{decs}})$*-binding small-domain PCS with*

$$\varepsilon_{\mathsf{decs}} = \binom{N}{d_{\mathsf{decs}} + 2} \cdot \varepsilon_{\mathcal{D}} + \frac{(Q+1)^2}{2^{2\lambda+1}} \quad \textit{with} \quad \varepsilon_{\mathcal{D}} := \max_{v \in \mathbb{F}^n \setminus \{0\},\, u \in \mathbb{F}^\eta} \Pr_{\Gamma \leftarrow \mathcal{D}_\Gamma}[\Gamma \cdot v + u = 0] \; ,$$

where Q is the number of random oracle queries made by the adversary. When $\mathcal{D}_\Gamma$ is the uniform distribution over $\mathbb{F}^{\eta \times n}$, we have $\varepsilon_{\mathcal{D}} = 1/|\mathbb{F}|^\eta$ implying:

$$\varepsilon_{\mathsf{decs}} = \frac{\binom{N}{d_{\mathsf{decs}}+2}}{|\mathbb{F}|^\eta} + \frac{(Q+1)^2}{2^{2\lambda+1}} \; .$$

Proof. Let us first build the extractor involved in the security proof, then we will upper bound the adversary's advantage in breaking the PCS binding property.

Extraction. Let us build the extractor $\mathcal{E}_{\mathsf{decs}}$. $\mathcal{E}_{\mathsf{decs}}$ takes as input the commitment $\mathsf{com} := (\mathsf{root}, \Gamma, \boldsymbol{R})$, together with the set $\mathcal{Q}$ of query-response to the hash function (modeled as a random oracle). It proceeds as follows:

1. $\mathcal{E}_{\mathsf{decs}}$ parses $\mathcal{Q}$ to find the Merkle tree leaves $u_1, \ldots, u_N$ by iteratively searching the pre-images of the node digests. If a node has no preimage, we set the child nodes as $\perp$. The child nodes of a parent node which is $\perp$ are also $\perp$. If a node has *several* preimages under the right format, then $\mathcal{E}_{\mathsf{decs}}$ outputs $\perp$.
2. For each u_j that is not $\perp$, $\mathcal{E}_{\mathsf{decs}}$ parses $\mathcal{Q}$ to find the pre-image $(\boldsymbol{p}_j, \boldsymbol{m}_j, j, \rho_j)$ of u_j. If this does not exist under this format, then we set $(\boldsymbol{p}_j, \boldsymbol{m}_j) = (\perp, \perp)$. If this is defined multiple times, then $\mathcal{E}_{\mathsf{decs}}$ outputs $\perp$. For all $u_j = \perp$, we also set $(\boldsymbol{p}_j, \boldsymbol{m}_j) = (\perp, \perp)$.
3. $\mathcal{E}_{\mathsf{decs}}$ computes the set
$$E_{\text{openable}} := \{e_j \text{ s.t. } (\boldsymbol{p}_j, \boldsymbol{m}_j) \neq (\perp, \perp) \text{ and } \boldsymbol{R}(e_j) = \boldsymbol{m}_j + \Gamma \cdot \boldsymbol{p}_j\}.$$
4. $\mathcal{E}_{\mathsf{decs}}$ now interpolates the polynomial $\boldsymbol{P}$ such that $\boldsymbol{P}(e_j) = \boldsymbol{p}_j$ for all $e_j \in E_{\text{openable}}$. If $\boldsymbol{P}$ is of degree strictly larger than d, then $\mathcal{E}_{\mathsf{decs}}$ outputs $\perp$. Otherwise, it outputs $\boldsymbol{P}$.

Upper bound. We want to upper bound the probability that an adversary succeeds in forging a valid opening while the opened evaluations are not consistent with the polynomial $\boldsymbol{P}$ extracted by the extractor $\mathcal{E}_{\mathsf{decs}}$.

First, let us bound the probability that the extraction fails (i.e. outputs $\perp$) while an adversary succeeds in forging a valid opening. Multiple preimages in the two first extraction steps would consists in a collision for the hash function. Since the later is modelled as a random oracle, the probability $\mathcal{E}_{\mathsf{decs}}$ outputs $\perp$ at those steps is then bound by $Q^2/2^{2\lambda+1}$, where $Q = |\mathcal{Q}|$. If $\mathcal{E}_{\mathsf{decs}}$ outputs $\perp$ at the last step, it means that the candidate polynomial $\boldsymbol{P}$ defined by interpolation over E_{openable} is of degree strictly larger than d. It implies that E_{openable} is of size at least $d+2$. We thus know that there exists a subset $E' \subset E_{\text{openable}}$ of size $d+2$ such that the polynomial $\boldsymbol{P}^{(E')}$ built by interpolation $e_j \in E' \mapsto \boldsymbol{p}_j$ is of degree $d+1$. Let us denote $\boldsymbol{M}^{(E')}$ the polynomial obtained by interpolation $e_j \in E' \mapsto \boldsymbol{m}_j$. We have

$$\boldsymbol{R} = \boldsymbol{M}^{(E')} + \Gamma \cdot \boldsymbol{P}^{(E')}$$

this relation is of degree $d+1$ and verifies in $d+2$ points as $E' \subset E_{\text{openable}}$. As $\boldsymbol{R}$ is of degree d, the degree-$(d+1)$ coefficient of $\boldsymbol{M}^{(E')} + \Gamma \cdot \boldsymbol{P}^{(E')}$ is zero. To summarize, if $\mathcal{E}_{\mathsf{decs}}$ outputs $\perp$ is the last step, then there exists a subset E of size $d+2$ such that the degree-$(d+1)$ coefficient of $\boldsymbol{M}^{(E')} + \Gamma \cdot \boldsymbol{P}^{(E')}$ is zero, while $\boldsymbol{P}^{(E')}$ is of degree $d+1$. The event only depends on the committed evaluations and on the challenge, i.e. does not depend on the choice of $\boldsymbol{R}$. We have

$$\begin{aligned}
&\Pr[\mathcal{E}_{\mathsf{decs}} \text{ outputs } \perp \text{ at the last step}] \\
&\quad \leq \Pr_{\Gamma}\left[\exists E' : |E'| = d+2 \wedge \mathsf{coeff}_{d+1}\left(\boldsymbol{P}^{(E')}\right) \neq 0 \wedge \mathsf{coeff}_{d+1}\left(\boldsymbol{M}^{(E')} + \Gamma \cdot \boldsymbol{P}^{(E')}\right) = 0\right] \\
&\quad \leq \sum_{E' : |E'| = d+2} \Pr\left[\mathsf{coeff}_{d+1}\left(\boldsymbol{P}^{(E')}\right) \neq 0 \wedge \mathsf{coeff}_{d+1}\left(\boldsymbol{M}^{(E')} + \Gamma \cdot \boldsymbol{P}^{(E')}\right) = 0\right] \qquad (1) \\
&\quad \leq \sum_{E' : |E'| = d+2} \Pr\left[\mathsf{coeff}_{d+1}\left(\boldsymbol{M}^{(E')}\right) + \Gamma \cdot \mathsf{coeff}_{d+1}\left(\boldsymbol{P}^{(e)}\right) = 0 \mid \mathsf{coeff}_{d+1}\left(\boldsymbol{P}^{(E')}\right) \neq 0\right] \\
&\quad \leq \sum_{E' : |E'| = d+2} \varepsilon_{\mathcal{D}} \leq \binom{N}{d+2} \cdot \varepsilon_{\mathcal{D}} \qquad (2)
\end{aligned}$$

where (1) holds from the union bound and (2) holds by definition of the distribution $\mathcal{D}_\Gamma$ using the fact that $\mathsf{coeff}_{d+1}\left(\boldsymbol{P}^{(e)}\right) \neq 0$. To sum up, the probability that the extractor fails while the adversary produces a valid opening is upper bounded by

$$\binom{N}{d+2} \cdot \varepsilon_{\mathcal{D}} + \frac{Q^2}{2^{2\lambda+1}} \ .$$

We can observe that the probability that the adversary produces a valid opening while the extractor returns a polynomial is negligible. Let us assume that an opened evaluation point $e_j \in E$ is not in E_{openable}:

– either it is not in E_{openable} because $(\boldsymbol{p}_j, \boldsymbol{m}_j)$ was $(\bot, \bot)$, thus having a valid opening for e_j means that the adversary found the missing hash pre-image of some tree node or leaf digest.
– or it is not in E_{openable} because $\boldsymbol{R}(e_j) \neq \boldsymbol{m}_j + \Gamma \cdot \boldsymbol{p}_j$, thus having a valid opening for e_j means that the adversary found a second hash pre-image that leads to a different opening (so the adversary found a hash collision).

So we get that $E \subset E_{\text{openable}}$ with overwhelming probability. And the evaluations are necessarily the extracted ones, as otherwise there would be a hash collision. As a result, the probability that the adversary produces a valid opening while the extractor returns a polynomial is

$$\frac{Q}{2^{2\lambda}} + \frac{Q^2}{2^{2\lambda+1}} \leq \frac{(Q+1)^2}{2^{2\lambda+1}} \ ,$$

which concludes the proof. □

The following theorem establishes the zero-knowledge property of the DECS protocol. We emphasize that the original DECS construction was not introduced with a zero-knowledge guarantee; this property is achieved here through the additional masking mechanisms integrated into our variant.

Theorem 2. *In the ROM, modelling* Hash *as a random oracle, the DECS protocol depicted in Figure 1 is a* (t, ξ)*-HVZK small-domain PCS, with*

$$\xi \leq \frac{Q}{2^{\lambda}}$$

where Q *is the number of random oracle queries made by the adversary.*

Proof. We first describe the HVZK simulator Sim taking $(E, \boldsymbol{P}|_E)$ as input. If $|E| \geq d_{\mathsf{decs}} + 1$, then Sim reconstructs $\boldsymbol{P}$ from $\boldsymbol{P}|_E$ and runs a genuine execution:

$$(\mathsf{com}^{(1)}, \mathsf{key}) \leftarrow \mathsf{Commit}(\boldsymbol{P})$$
$$(out, \pi^{(1)}) \leftarrow \mathsf{Eval}(\mathsf{key}, \mathsf{com}^{(1)}, E, \boldsymbol{P}|_E)$$

and returns $(\mathsf{com}^{(1)}, \pi^{(1)})$. We then have that $(\mathsf{com}^{(1)}, \pi^{(1)})$ is identically distributed to $(\mathsf{com}^{(0)}, \pi^{(0)})$ and hence $\xi = 0$. Now let us assume that $|E| \leq d_{\mathsf{decs}}$. The simulator runs as follows:

1. Sample a random matrix $\Gamma = (\gamma_{k,i})_{k,i} \leftarrow \mathcal{D}(\mathbb{F}^{\eta \times n_{\mathsf{decs}}})$,
2. Sample a random vector polynomial $\boldsymbol{R} = (R_1, \ldots, R_\eta) \leftarrow (\mathbb{F}[X]^{(\leq d_{\mathsf{decs}})})^\eta$,
3. Sample N random hash commitment tapes $(\rho_1, \ldots, \rho_N) \leftarrow (\{0,1\}^\lambda)^N$,
4. For all $e \in \mathbb{E}$, sample a random vector $\boldsymbol{P}(e) \leftarrow \mathbb{F}^{n_{\mathsf{decs}}}$,
5. For all $e \in E$, and all $k \in [1, \eta]$ defines

$$M_k(e) = R_k(e) - \sum_{i=1}^{n_{\mathsf{decs}}} \gamma_{k,i} \cdot P_i(e) \ .$$

   For all $e \in \mathbb{E} \setminus E$, sample a random vector $\boldsymbol{M}(e) \leftarrow \mathbb{F}^\eta$.
6. Compute $u_j = \mathsf{Hash}(\boldsymbol{P}(e_j), \boldsymbol{M}(e_j), j, \rho_j)$ for every $e_j \in \mathbb{E}$ and $\mathsf{root} = \mathsf{MerkleRoot}(u_1, \ldots, u_N)$.
7. Let $\mathsf{com}^{(1)} = (\mathsf{root}, \Gamma, \boldsymbol{R})$. Let $\pi^{(1)}$ the evaluation protocol transcript made of the evaluations $\boldsymbol{M}|_E$, the random tapes $\{\rho_j \ ; \ e_j \in E\}$ and the proof π_{MT} (i.e. the authentication paths from the leaves $\{u_j\}_{e_j \in E}$ to the Merkle root root). Return $(\mathsf{com}^{(1)}, \pi^{(1)})$.

We now use a hybrid argument to bound the adversary's advantage in distinguishing a proof $(\mathsf{com}^{(1)}, \pi^{(1)})$ returned by the above simulator from a genuine proof generated using the full polynomial $\boldsymbol{P}$. Let us denote Game_0, the standard HVZK game depicted in Definition 4 and let Game_1 the game which is similar to Game_0

but stopping whenever the adversary makes a RO query of the form $\mathsf{Hash}(\boldsymbol{p}, \boldsymbol{m}, j, \rho_j)$ for any $\boldsymbol{p} \in \mathbb{F}^{n_{\mathsf{decs}}}$ and $\boldsymbol{m} \in \mathbb{F}^{\eta}$, for j such that $e_j \notin E$ and for ρ_j the random tape generated by the simulator. We have:

$$\Pr[\mathsf{Game}_1 \text{ stops}] \leq \frac{Q}{2^{\lambda}} .$$

Indeed, for each RO query of the right format, the adversary has probability $1/2^{\lambda}$ of using the right ρ_j. Then, conditioned to the event that Game_1 does not stop, we have that $(\mathsf{com}^{(1)}, \pi^{(1)})$ is identically distributed to a genuine proof $(\mathsf{com}^{(0)}, \pi^{(0)})$ computed from $\boldsymbol{P}$. Specifically,

- Γ is freshly random from $\mathcal{D}(\mathbb{F}^{\eta \times n_{\mathsf{decs}}})$,
- $\boldsymbol{M}|_E$ and $\boldsymbol{R}$ are uniformly random from $(\mathbb{F}^{\eta})^{|E|}$ and $(\mathbb{F}[X])^{\eta}$ satisfying the degree-enforcing test,
- $\{\rho_j \,;\, e_j \in E\}$ are uniformly random from $\{0,1\}^{\lambda}$,
- root is the Merkle root corresponding to leaves $u_j = \mathsf{Hash}(\boldsymbol{P}(e_j), \boldsymbol{M}(e_j), j, \rho_j)$ for every j such that $e_j \in E$ and for uniformly random leaves u_j for every j such that $e_j \notin E$,
- π_{MT} is made of correct authentication paths from $\{u_j \,;\, e_j \in E\}$ to root.

We deduce:

$$\Pr[\mathcal{A} \text{ wins } \mathsf{Game}_0] \leq \underbrace{\Pr[\mathcal{A} \text{ wins } \mathsf{Game}_1 \mid \mathsf{Game}_1 \text{ does not stop}]}_{\leq 1/2} + \underbrace{\Pr[\mathsf{Game}_1 \text{ stops}]}_{\leq Q/2^{\lambda}}$$

which concludes the proof. □

4 SmallWood Polynomial Commitment Scheme

This section introduces further commitment schemes constructed from a small-domain PCS. Although our concrete schemes are based on the DECS protocol pictured in Figure 1, we stress that our constructions and analysis can rely on any small-domain PCS and could benefit future constructions of small-domain PCS. We first introduce SmallWood-LVCS a linear-map vector commitment scheme (LVCS) and then introduce SmallWood-PCS, a polynomial commitment scheme (PCS) constructed from SmallWood-LVCS. The proposed constructions are based on techniques from [BCG20, Lee21, GLS+23]. Our main contribution is to formally describe and analyze the LVCS and PCS obtained by applying these techniques with the DECS protocol and to extend these schemes to inherently integrate the zero-knowledge property.

4.1 SmallWood LVCS

Let $\Omega = \{\omega_1, \ldots, \omega_{n_{\mathsf{cols}}}\} \subseteq \mathbb{F}$. The high-level principle of the LVCS is to commit the rows $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{n_{\mathsf{rows}}} \in \mathbb{F}^{n_{\mathsf{cols}}}$ as polynomials $P_1, \ldots, P_{n_{\mathsf{rows}}}$ using the DECS protocol. Specifically, for all j, we define P_j by interpolation as a polynomial satisfying $(P_j(\omega_1), \ldots, P_j(\omega_{n_{\mathsf{cols}}})) = \boldsymbol{r}_j$. Later, to prove $\boldsymbol{v}_k = \sum_j c_{k,j} \cdot \boldsymbol{r}_j$ for some vector $\boldsymbol{v}_k \in \mathbb{F}^{n_{\mathsf{cols}}}$ and coefficients $c_{k,1}, \ldots, c_{k,n_{\mathsf{rows}}} \in \mathbb{F}$, the prover sends the polynomial $Q_k(X) = \sum_j c_{k,j} \cdot P_j(X)$ to the verifier who checks $(Q_k(\omega_1), \ldots, Q_k(\omega_{n_{\mathsf{cols}}})) = \boldsymbol{v}_k$. The verifier then challenges the prover to open some evaluations $\boldsymbol{P}|_E$ for $\boldsymbol{P} = (P_1, \ldots, P_{n_{\mathsf{cols}}})$ and $E \subseteq \mathbb{E}$ a subset of the DECS evaluation domain, and checks $Q_k(e) = \sum_j c_{j,k} \cdot P_j(e)$ for all $e \in E$ (thus verifying that Q_k has been correctly constructed by the prover).

To achieve the zero-knowledge property, the set Ω must be disjoint from $\mathbb{E}$. In addition, we must ensure that opening ℓ evaluations of $P_1, \ldots, P_{n_{\mathsf{rows}}}$ leaks no information about the rows' coefficients, where $\ell := |E|$. To this purpose, we randomly pick vectors $\bar{\boldsymbol{r}}_j \in \mathbb{F}^{\ell}$ and build the P_j as the polynomial of degree $\leq d := n_{\mathsf{cols}} + \ell - 1$ satisfying both $(P_j(\omega_1), \ldots, P_j(\omega_{n_{\mathsf{cols}}})) = \boldsymbol{r}_j$ and $(P_j(\omega'_1), \ldots, P_j(\omega'_{\ell})) = \bar{\boldsymbol{r}}_j$ for some set $\Omega' = \{\omega'_1, \ldots, \omega'_{\ell}\} \subseteq \mathbb{F}$ disjoint from Ω.

The opening proof is sound thanks to the Schwartz-Zippel lemma: if there exists i such that $(\boldsymbol{v}_k)_i \neq \sum_j c_{k,j} \cdot (\boldsymbol{r}_j)_i$, while $(Q_k(\omega_1), \ldots, Q_k(\omega_{n_{\mathsf{cols}}})) = \boldsymbol{v}_k$ (which is necessary for the verifier to accept), then the polynomial Q_k sent by the prover does not equal $\sum_j c_{k,j} \cdot P_j$ (since it differs at least in point ω_i). Then, the probability that the verifier accepts the proof is at most $\binom{d}{\ell} / \binom{|\mathbb{E}|}{\ell}$, where $d = \max\{\deg P_j\} \leq n_{\mathsf{cols}} + \ell - 1$.

LVCS.Commit:

On input vectors $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{n_{\mathsf{rows}}}$ from $\mathbb{F}^{n_{\mathsf{cols}}}$ for the prover, run the following protocol:

1. $\mathcal{P}$ samples uniformly at random $\bar{\boldsymbol{r}}_1, \ldots, \bar{\boldsymbol{r}}_{n_{\mathsf{rows}}} \leftarrow \mathbb{F}^{\ell}$ and, for all $j \in [1, n_{\mathsf{rows}}]$, builds by interpolation the degree-$(n_{\mathsf{cols}} + \ell - 1)$ polynomial $P_j(X)$ such that
$$(P_j(\omega_1), \ldots, P_j(\omega_{n_{\mathrm{cols}}})) = \boldsymbol{r}_j \quad \text{and} \quad (P_j(\omega'_1), \ldots, P_j(\omega'_\ell)) = \bar{\boldsymbol{r}}_j \ .$$
Let $\boldsymbol{P} = (P_1, \ldots, P_{n_{\mathsf{rows}}})$ the associated vector polynomial.
2. $\mathcal{P}$ and $\mathcal{V}$ run the protocol $\mathsf{DECS.Commit}(\boldsymbol{P})$ resulting in the commitment $\mathsf{com}_{\mathsf{decs}}$ and the opening key $\mathsf{key}_{\mathsf{decs}}$.

$\mathcal{P}$ returns $\mathsf{key} := (\mathsf{key}_{\mathsf{decs}}, (\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{n_{\mathsf{rows}}}), (\bar{\boldsymbol{r}}_1, \ldots, \bar{\boldsymbol{r}}_{n_{\mathsf{rows}}}))$ while the commitment (i.e., the protocol transcript) is defined as $\mathsf{com} := \mathsf{com}_{\mathsf{decs}}$.

LVCS.Eval:

In this protocol, the prover takes as input an LVCS opening key $\mathsf{key} = (\mathsf{key}_{\mathsf{decs}}, (\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{n_{\mathsf{rows}}}), (\bar{\boldsymbol{r}}_1, \ldots, \bar{\boldsymbol{r}}_{n_{\mathsf{rows}}}))$, and the verifier takes as input a commitment com. Both parties additionally take a set of coefficients $\mathcal{C} := \{c_{k,j}\}$ with $1 \le k \le m$ and $1 \le j \le n_{\mathsf{rows}}$, and a set of vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m \in \mathbb{F}^{n_{\mathsf{cols}}}$. The protocol runs as follows:

1. $\mathcal{P}$ computes $\bar{\boldsymbol{v}}_k = \sum_{j=1}^{n_{\mathsf{rows}}} c_{k,j} \cdot \bar{\boldsymbol{r}}_j$ for every $k \in [1, m]$ and sends $\bar{\boldsymbol{v}}_1, \ldots, \bar{\boldsymbol{v}}_m$ to $\mathcal{V}$.
2. $\mathcal{V}$ samples $E \leftarrow \binom{\mathbb{E}}{\ell}$ (random subset $E \subseteq \mathbb{E}$ of size $|E| = \ell$) and send it to $\mathcal{P}$.
3. $\mathcal{P}$ sends $\{\boldsymbol{p}_e\}_{e \in E} := \{\boldsymbol{P}(e)\ ;\ e \in E\}$ to $\mathcal{V}$.
4. $\mathcal{P}$ and $\mathcal{V}$ run the protocol $\mathsf{DECS.Eval}(\mathsf{key}_{\mathsf{decs}}, \mathsf{com}_{\mathsf{decs}}, E, \{\boldsymbol{p}_e\}_{e \in E})$. If the output is Reject, $\mathcal{V}$ stops and outputs Reject. Otherwise, the protcol continues.
5. For every $k \in [1, m]$, $\mathcal{V}$ computes $Q_k(X)$ such that
$$(Q_k(\omega_1), \ldots, Q_k(\omega_{n_{\mathrm{cols}}})) = \boldsymbol{v}_k \quad \text{and} \quad (Q_k(\omega'_1), \ldots, Q_k(\omega'_\ell)) = \bar{\boldsymbol{v}}_k \ .$$
$\mathcal{V}$ checks $Q_k(e) = \sum_{j=1}^{n_{\mathsf{rows}}} c_{k,j} \cdot (\boldsymbol{p}_e)_j$ for all $k \in [1, m]$ and $e \in E$. If all the equalities are verified, then $\mathcal{V}$ outputs Accept, otherwise $\mathcal{V}$ outputs Reject.

Fig. 2: SmallWood-LVCS: Linear-map vector commitment scheme from the DECS protocol.

In practice, the prover does not need to send the polynomials $\{Q_k\}_k$ in full but only the vectors $\bar{\boldsymbol{v}}_k = \sum_{j=1}^{n_{\mathsf{rows}}} c_{k,j} \cdot \bar{\boldsymbol{r}}_j$ which, together with the vectors $\boldsymbol{v}_k$'s, enable the verifier to recover $\{Q_k\}_k$ by interpolation.

The obtained LVCS is formally described in Figure 2 and its security is given by the following theorem.

Theorem 3. *The LVCS depicted in Figure 2 is (t, ε)-extractable binding with*

$$\varepsilon = \varepsilon_{\mathsf{decs}} + \frac{\binom{n_{\mathsf{cols}} + \ell - 1}{\ell}}{\binom{N}{\ell}} \ ,$$

where $\varepsilon_{\mathsf{decs}}$ is the advantage of breaking the extractable binding property of the underlying DECS protocol in time t.

Proof. Let us first build the extractor involved in the security proof, then we will upper bound the adversary's advantage in breaking the LVCS binding property.

Extraction. Let us build the extractor $\mathcal{E}_{\mathsf{lvcs}}$. $\mathcal{E}_{\mathsf{lvcs}}$ takes as input the commitment com, together with the set $\mathcal{Q}$ of query-response pairs to the hash function (modeled as a random oracle). It proceeds as follows:

- $\mathcal{E}_{\mathsf{lvcs}}$ calls $\mathcal{E}_{\mathsf{decs}}$ over $\mathcal{Q}$ and com to get a polynomial $\boldsymbol{P}$ of degree $(n_{\mathsf{cols}} + \ell - 1)$. If it fails, $\mathcal{E}_{\mathsf{lvcs}}$ directly outputs $\perp$.
- For all j, $\mathcal{E}_{\mathsf{lvcs}}$ computes $\boldsymbol{r}_j$ as $(P_j(\omega_1), \ldots, P_j(\omega_{n_{\mathrm{cols}}}))$. Then it returns $(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{n_{\mathsf{rows}}})$.

Upper bound. We want to upper bound the probability that an adversary succeeds in forging a valid opening while the opened linear combinations $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)$ are not consistent with the vectors $(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{n_{\mathsf{rows}}})$ extracted by the extractor $\mathcal{E}_{\mathsf{lvcs}}$.

Denoting $\boldsymbol{Q}$ the polynomial reconstructed by the verifier in the opening phase and $\boldsymbol{P}$ the polynomial extracted by the DECS extractor, we have

$$
\begin{aligned}
&\Pr[\text{adversary wins the LVCS extractable binding game}] \\
&\quad \leq \Pr[\text{valid opening and } \mathcal{E}_{\mathsf{lvcs}} \text{ fails to extract}] \\
&\qquad + \Pr[\text{valid opening and } \mathcal{E}_{\mathsf{lvcs}} = (\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{n_{\mathsf{rows}}}) \text{ and } \exists k,\ \sum_j c_{k,j} \cdot \boldsymbol{r}_j \neq \boldsymbol{v}_k] \qquad (3) \\
&\quad \leq \Pr[\text{valid opening and } \mathcal{E}_{\mathsf{decs}} \text{ fails to extract}] \\
&\qquad + \Pr[\text{valid opening and } \mathcal{E}_{\mathsf{decs}} = \boldsymbol{P} \text{ and } \boldsymbol{P}|_E \neq \{\boldsymbol{p}_e\}_{e \in E}] \\
&\qquad + \Pr[\text{valid opening and } \mathcal{E}_{\mathsf{decs}} = \boldsymbol{P} \text{ and } \exists k,\ \sum_j c_{k,j} \cdot P_j \neq Q_k \text{ and } \boldsymbol{P}|_E = \{\boldsymbol{p}_e\}_{e \in E}] \qquad (4) \\
&\quad \leq \Pr[\text{adversary wins the DECS extractable binding game}] \\
&\qquad + \Pr[\text{valid opening and } \mathcal{E}_{\mathsf{decs}} = \boldsymbol{P} \text{ and } \exists k,\ \sum_j c_{k,j} \cdot P_j \neq Q_k \text{ and } \boldsymbol{P}|_E = \{\boldsymbol{p}_e\}_{e \in E}] \qquad (5)
\end{aligned}
$$

where (3) just splits the event depending on the LVCS extractor behavior, (4) splits the second probability depending on whether $\boldsymbol{P}|_E = \{\boldsymbol{p}_e\}_{e \in E}$ or not, and (5) merges the two first event as it correspond to the success event of the DECS extractable inding game. Let us stress that the existence of an index k such that $\sum_j c_{k,j} \cdot P_j \neq Q_k$ implies that $\sum_j c_{k,j} \cdot \boldsymbol{r}_j \neq \boldsymbol{v}_k$, as it corresponds to the evaluations into Ω.

Let us upper bound the probability that the adversary succeeds in producing a valid opening knowing that there exists some k' such that $Q_{k'} \neq \sum_j c_{k',j} \cdot P_j$ while having $\boldsymbol{P}|_E = \{\boldsymbol{p}_e\}_{e \in E}$. Since the opening is valid, we knows that $Q_k(e) = \sum_{j=1}^{n_{\mathsf{rows}}} c_{k,j} \cdot (\boldsymbol{p}_e)_j$ for all $k \in [1, m]$ and $e \in E$ as it corresponds to a verifier check. So we get that $Q_{k'}|_E = \sum_j c_{k',j} \cdot P_j|_E$ using $\boldsymbol{P}|_E = \{\boldsymbol{p}_e\}_{e \in E}$. Over the randomness on the choice of E, we have

$$
\Pr_E[Q_{k'}|_E = \sum_j c_{k',j} \cdot P_j|_E \mid \sum_j c_{k',j} \cdot P_j \neq Q_{k'}] \leq \frac{\binom{n_{\mathsf{cols}} + \ell - 1}{\ell}}{\binom{N}{\ell}}
$$

thanks to the Schwartz-Zippel Lemma and because the tested polynomial relation is of degree $n_{\mathsf{cols}} + \ell - 1$.

To sum up, the probability that an adversary wins the LVCS extractable binding game is upper bound by

$$
\varepsilon_{\mathsf{decs}} + \frac{\binom{n_{\mathsf{cols}} + \ell - 1}{\ell}}{\binom{N}{\ell}} \ .
$$

□

Theorem 4. *The LVCS depicted in Figure 2 is (t, ξ)-HVZK provided that the underlying DECS protocol is (t, ξ)-HVZK.*

Proof. By definition, the vectors $\bar{\boldsymbol{v}}_1, \ldots, \bar{\boldsymbol{v}}_m$ and the evaluation points $\boldsymbol{P}|_E := \{\boldsymbol{P}(e)\,;\, e \in E\}$ are uniformly random and independent from the input vectors $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{n_{\mathsf{rows}}}$. The HVZK simulator simply samples these values randomly (for a random evaluation set E of size ℓ) and call the HVZK simulator of the DECS for the rest. □

4.2 SmallWood PCS

We now describe how to construct a PCS from the LVCS. The idea is to arrange the coefficients of the committed polynomial as rows of the LVCS commitment. Then a polynomial evaluation can be obtained

from a linear combination of the rows with the appropriate powers of the evaluation point as coefficients. Additional care must be given so that the opened LVCS evaluations do not leak information in order to ensure the zero-knowledge property.

We explain the principle with a simple example. For a polynomial $P(X) = \sum_{i=0}^{d} a_i \cdot X^i$ and some parameters $\mu, \nu \in \mathbb{N}$ such that $\nu\mu = d+1$, consider the matrix

$$\boldsymbol{A} = \begin{bmatrix} a_0 & a_\mu & \cdots & a_{(\nu-1)\mu} \\ a_1 & a_{\mu+1} & \cdots & a_{(\nu-1)\mu+1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{\mu-1} & a_{2\mu-1} & \cdots & a_{\nu\mu-1} \end{bmatrix} .$$

We can express an evaluation $P(e)$ from a linear combination $\boldsymbol{v}$ of the rows of the matrix:

$$P(e) = \langle \boldsymbol{v}, (1, e^\mu, e^{2\mu}, \ldots, e^{(\nu-1)\mu}) \rangle \quad \text{with} \quad \boldsymbol{v} := (1, e, e^2, \ldots, e^{\mu-1}) \cdot \boldsymbol{A} .$$

This way, a linear-map vector commitment of the rows of $\boldsymbol{A}$ yields a polynomial commitment of P. However, this is not zero-knowledge since opening the evaluation $P(e)$ implies opening the vector $\boldsymbol{v}$ which reveals additional information on the coefficients of P, specifically ν degrees of freedom (i.e. linear combinations of the coefficients) instead of 1. To make the construction zero-knowledge, we shall mask these $\nu - 1$ revealed degrees of freedom using randomness. For this, the matrix $\boldsymbol{A}$ is replaced by the following $(\mu+1) \times \nu$ matrix:

$$\boldsymbol{A'} = \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{0} \end{bmatrix} + \begin{bmatrix} 0 & -r_1 & -r_2 & \cdots & -r_{\nu-2} & -r_{\nu-1} \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ r_1 & r_2 & r_3 & \cdots & r_{\nu-1} & 0 \end{bmatrix} .$$

In words, we append a $\boldsymbol{0}$ (all-0 row vector) to $\boldsymbol{A}$, then we add $(0, -r_1, \ldots, -r_{\nu-1})$ to the first row and $(r_1, \ldots, r_{\nu-1}, 0)$ to the last row. We now have:

$$P(e) = \langle \boldsymbol{v'}, (1, e^\mu, e^{2\mu}, \ldots, e^{(\nu-1)\mu}) \rangle \quad \text{with} \quad \boldsymbol{v'} := (1, e, e^2, \ldots, e^{\mu-1}, e^\mu) \cdot \boldsymbol{A'} .$$

Moreover, it can be checked that $\boldsymbol{v'}$ is uniformly random among the vector of $\mathbb{F}^\nu$ satisfying the relation $P(e) = \sum_i v'_i \cdot e^{i \cdot \mu}$ which implies that $\boldsymbol{v'}$ reveal no more information on P than the evaluation $P(e)$. To open ℓ' evaluations $\{P(e)\}$, one shall add ℓ' such rows of randomness to the above matrix.

Our PCS follows the above approach which we now describe for the general setting of a vector polynomial $\boldsymbol{P} = (P_1, \ldots, P_{n_{\mathsf{pcs}}}) \in \mathbb{F}[X]^{(\leq d_1)} \times \cdots \times \mathbb{F}[X]^{(\leq d_{n_{\mathsf{pcs}}})}$ which aims to be opened in ℓ' points $E' = \{e_1, \ldots, e_{\ell'}\}$ (here we use the notation E' to avoid confusion with the set E of the DECS evaluation points arising in the LVCS description). Each polynomial $P_j(X) = \sum_i a_{j,i} \cdot X^i$ gives rise to a matrix $\boldsymbol{A}_j$ with $\mu + \ell'$ rows and ν_j columns such that

$$\nu_j := \lceil (d_j + 1 - \ell')/\mu \rceil .$$

For the sake of simplicity, we shall assume that $d_j + 1 - \ell'$ is a multiple of μ, so that $\nu_j = (d_j + 1 - \ell')/\mu$. We also require $\ell' \leq \mu$. The matrix $\boldsymbol{A}_j$ is then defined as:

$$\boldsymbol{A}_j := \left[\begin{array}{ccc|c} a_{j,0} & \cdots & a_{j,(\nu_j-2)\mu} & a_{j,(\nu_j-1)\mu} \\ \vdots & \ddots & \vdots & \vdots \\ a_{j,\mu-1} & \cdots & a_{j,(\nu_j-1)\mu-1} & a_{j,\nu_j\mu-1} \\ \hline 0 & \cdots & 0 & a_{j,\nu_j\mu} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & a_{j,\nu_j\mu+\ell'-1} \end{array}\right] + \left[\begin{array}{cccc|c} 0 & -r_{j,1,1} & \cdots & -r_{j,1,\nu_j-2} & -r_{j,1,\nu_j-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & -r_{j,\ell',1} & \cdots & -r_{j,\ell',\nu_j-2} & -r_{j,\ell',\nu_j-1} \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \\ \hline r_{j,1,1} & r_{j,1,2} & \cdots & r_{j,1,\nu_j-1} & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ r_{j,\ell',1} & r_{j,\ell',2} & \cdots & r_{j,\ell',\nu_j-1} & 0 \end{array}\right] .$$

In the above equation, the horizontal line marks the frontier between the first μ rows and the last ℓ' rows (which are added for the zero-knowledge property). We optimize the mapping by overflowing the high-degree coefficients of P_j on the ℓ' extra rows of the last column. Using this tweak, one can verify that the $\boldsymbol{A}_j$ matrix satisfies:

$$P_j(e) = \langle \boldsymbol{v}^{(e)}, (1, e^\mu, e^{2\mu}, \ldots, e^{(\nu_j-1)\mu}) \rangle \quad \text{with} \quad \boldsymbol{v}^{(e)} := (1, e, e^2, \ldots, e^{\mu+\ell'-1}) \cdot \boldsymbol{A}_j$$

for every evaluation point $e \in \mathbb{F}$. Moreover, one can check that for any ℓ' evaluation points $E' = \{e_1, \ldots, e_{\ell'}\}$, the vectors $\{\boldsymbol{v}^{(e)}\}_{e \in E'}$ do not reveal any more information than the evaluations $\{P_j(e)\}_{e \in E'}$.

Let us now consider the case where $d_j + 1 - \ell'$ is not a multiple of μ. Since ν_j is defined as $\nu_j := \lceil (d_j + 1 - \ell')/\mu \rceil$, it implies that

$$\mu \cdot \nu_j + \ell' > d_j + 1 \ .$$

We might consider building $\boldsymbol{A}_j$ as previously by writing $P_j(X)$ as $\sum_{i=0}^{d'_j} a_{j,i} \cdot X^i$, with $d'_j = \mu \cdot \nu_j + \ell' - 1$ and $(a_{j,d_j+1}, \ldots, a_{j,d'_j}) = (0, \ldots, 0)$. However, in doing so, the verifier would only be guaranteed that the prover committed to a degree-d'_j polynomial, rather than a degree-d_j polynomial, since there would be no assurance that the coefficients $(a_{j,d+1}, \ldots, a_{j,d'})$ are actually zero. To avoid this issue, we can define $\boldsymbol{A}_j$ as

$$\boldsymbol{A}_j := \left[\begin{array}{ccc|c} a_{j,0} & \cdots & a_{j,(\nu_j-2)\mu} & \left.\begin{matrix}0\\ \vdots \\ 0\end{matrix}\right\} \begin{matrix}\delta_j \\ \text{times}\end{matrix} \\ \vdots & \ddots & \vdots & \\ \vdots & \ddots & \vdots & \\ \vdots & \ddots & \vdots & \\ \hline a_{j,\mu-1} & \cdots & a_{j,(\nu_j-1)\mu-1} & a_{j,(\nu_j-1)\mu} \\ \hline 0 & \cdots & 0 & a_{j,d_j-\ell'+1} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & a_{j,d_j} \end{array}\right] + \left[\begin{array}{ccc|c} 0 & -r_{j,1,1} & \cdots & -r_{j,1,\nu_j-2} & \left.\begin{matrix}0\\ \vdots \\ 0\end{matrix}\right\} \begin{matrix}\delta_j \\ \text{times}\end{matrix} \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & -r_{j,\ell',1} & \cdots & -r_{j,\ell',\nu_j-2} & \\ 0 & 0 & \cdots & 0 & -r_{j,1,\nu_j-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & -r_{j,\ell',\nu_j-1} \\ \hline r_{j,1,1} & r_{j,1,2} & \cdots & r_{j,1,\nu_j-1} & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ r_{j,\ell',1} & r_{j,\ell',2} & \cdots & r_{j,\ell',\nu_j-1} & 0 \end{array}\right] \tag{6}$$

with $\delta_j := (\mu \cdot \nu_j + \ell') - (d_j + 1)$, namely we shift the last column of $\boldsymbol{A}_j$ to the bottom by δ_j coefficients. Then one can verify that this new $\boldsymbol{A}_j$ matrix satisfies

$$P_j(e) = \langle \boldsymbol{v}^{(e)}, (1, e^\mu, e^{2\mu}, \ldots, e^{(\nu_j-2)\mu}, e^{(\nu_j-1)\mu-\delta_j}) \rangle \quad \text{with} \quad \boldsymbol{v}^{(e)} := (1, e, e^2, \ldots, e^{\mu+\ell'-1}) \cdot \boldsymbol{A}_j \ .$$

Let us now define the global matrix $\boldsymbol{A}$ of the LVCS which embeds all the matrices $\boldsymbol{A}_j$. Remind that all the $\boldsymbol{A}_j$'s have the same number of rows $\mu + \ell'$ but different number of columns ν_j (depending on the degree d_j of P_j). We introduce an additional parameter β for the number of $(\mu + \ell')$-row layers in the matrix $\boldsymbol{A}$. The global matrix $\boldsymbol{A}$ is then defined as:

$$\underbrace{\boldsymbol{A}}_{\beta(\mu+\ell') \times \lceil \nu/\beta \rceil} := \mathsf{Stack}_\beta\big(\underbrace{\boldsymbol{A}_1 | \ldots | \boldsymbol{A}_{n_{\mathsf{pcs}}}}_{(\mu+\ell') \times \nu} \big)$$

where $\nu = \sum_j \nu_j$ is the total number of columns of the $\boldsymbol{A}_j$ matrices and Stack_β is the mapping which consists in splitting the columns of a matrix in β subsequent groups and stacking them so that the resulting matrix has $\beta(\mu + \ell')$ rows (instead of $\mu + \ell'$) and $\lceil \nu/\beta \rceil$ columns (instead of ν), possibly padded with all-0 columns.

To open the evaluations $P_1(e), \ldots, P_{n_{\mathsf{pcs}}}(e)$ for a point $e \in \mathbb{F}$, one opens the LVCS evaluations with the coefficient vectors:

$$\boldsymbol{u}_1 \otimes (1, e, \ldots, e^{\mu+\ell'}), \ldots, \boldsymbol{u}_\beta \otimes (1, e, \ldots, e^{\mu+\ell'}) \ ,$$

where $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_\beta$ is the canonical basis of $\mathbb{F}^\beta$ and $\otimes$ is the tensor product. Denoting $\boldsymbol{v}_1^{(e)}, \ldots, \boldsymbol{v}_\beta^{(e)}$ the LVCS openings associated to these coefficients, i.e. $\boldsymbol{v}_k^{(e)} := \big(\boldsymbol{u}_k \otimes (1, e, \ldots, e^{\mu+\ell'})\big) \cdot \boldsymbol{A}$ for every $k \in [1, \beta]$, one can check that we get

$$(\boldsymbol{v}_1^{(e)} \mid \ldots \mid \boldsymbol{v}_\beta^{(e)}) = (\hat{\boldsymbol{v}}_1^{(e)} \mid \ldots \mid \hat{\boldsymbol{v}}_{n_{\mathsf{pcs}}}^{(e)})$$

where $\hat{\boldsymbol{v}}_1^{(e)} \ldots, \hat{\boldsymbol{v}}_{n_{\mathsf{pcs}}}^{(e)}$ are vectors of size $\nu_1, \ldots, \nu_{n_{\mathsf{pcs}}}$ such that

$$P_j(e) = \langle \hat{\boldsymbol{v}}_j^{(e)}, (1, e^{\mu}, \ldots, e^{(\nu_j-2)\mu}, e^{(\nu_j-1)\mu-\delta_j}) \rangle$$

for every $j \in [1, n_{\mathsf{pcs}}]$. The obtained PCS is formally described in Figure 3 and its security is stated in the following theorem.

PCS.Commit:

On input a vector polynomial $\boldsymbol{P} = (P_1, \ldots, P_{n_{\mathsf{pcs}}}) \in \mathbb{F}[X]^{(\leq d_1)} \times \cdots \times \mathbb{F}[X]^{(\leq d_n)}$ for the prover, PCS.Commit consists in the following interactive protocol:

1. For all $1 \leq j \leq n_{\mathsf{pcs}}$, $\mathcal{P}$ picks $\ell'(\nu_j - 1)$ values $r_{j,1,1}, \ldots, r_{j,\nu_j-1,\ell'}$ uniformly at random from $\mathbb{F}$ and defines the matrix $\boldsymbol{A}_j$ following Equation (6) from the those random values and the coefficients $\{a_{j,i}\}_{j,i}$ of the polynomials $P_j(X) = \sum_{i=0}^{d_j} a_{j,i} X^i$.
2. $\mathcal{P}$ and $\mathcal{V}$ run the protocol $\mathsf{LVCS.Commit}(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{n_{\mathsf{rows}}})$ with

$$\begin{bmatrix} \boldsymbol{r}_1^\top \\ \boldsymbol{r}_2^\top \\ \vdots \\ \boldsymbol{r}_{n_{\mathsf{rows}}}^\top \end{bmatrix} = \mathsf{Stack}_\beta\big(\boldsymbol{A}_1 | \ldots | \boldsymbol{A}_{n_{\mathsf{pcs}}}\big) \ ,$$

   resulting in the commitment $\mathsf{com}_{\mathsf{LVCS}}$ and opening key $\mathsf{key}_{\mathsf{LVCS}}$.

$\mathcal{P}$ returns $\mathsf{key} := (\mathsf{key}_{\mathsf{LVCS}}, \boldsymbol{A}_1, \ldots \boldsymbol{A}_{n_{\mathsf{pcs}}})$ while the commitment (i.e., the protocol transcript) is defined as $\mathsf{com} := \mathsf{com}_{\mathsf{LVCS}}$.

PCS.Eval:

In this protocol, the prover takes as input a PCS opening key $\mathsf{key} = (\mathsf{key}_{\mathsf{LVCS}}, \boldsymbol{A}_1, \ldots \boldsymbol{A}_{n_{\mathsf{pcs}}})$, and the verifier takes as input a commitment com. Both parties additionally take a set of evaluation points $E' \subseteq \mathbb{F}$ and a set of evaluations $\boldsymbol{P}|_{E'} := \{\boldsymbol{P}(e)\}_{e \in E'}$. The protocol runs as follows:

1. $\mathcal{P}$ computes

$$\boldsymbol{v}_k^{(e)} := \big(\boldsymbol{u}_k \otimes (1, e, \ldots, e^{\mu+\ell'-1})\big) \cdot \boldsymbol{A}$$

   for every $k \in [1, \beta]$ and $e \in E'$, where $\boldsymbol{A} = \mathsf{Stack}_\beta\big(\boldsymbol{A}_1 | \ldots | \boldsymbol{A}_{n_{\mathsf{pcs}}}\big)$ and sends $\{\boldsymbol{v}_1^{(e)}, \ldots, \boldsymbol{v}_\beta^{(e)}\}_{e \in E'}$ to $\mathcal{V}$.
2. For every $j \in [1, n_{\mathsf{pcs}}]$ and $e \in E'$, $\mathcal{V}$ checks that the following equality correctly holds:

$$P_j(e) = \langle \hat{\boldsymbol{v}}_j^{(e)}, (1, e^{\mu}, \ldots, e^{(\nu_j-2)\mu}, e^{(\nu_j-1)\mu-\delta_j}) \rangle$$

   where $\hat{\boldsymbol{v}}_1^{(e)} \ldots, \hat{\boldsymbol{v}}_{n_{\mathsf{pcs}}}^{(e)}$ are vectors of size $\nu_1, \ldots, \nu_{n_{\mathsf{pcs}}}$ defined as $(\hat{\boldsymbol{v}}_1^{(e)} \mid \ldots \mid \hat{\boldsymbol{v}}_{n_{\mathsf{pcs}}}^{(e)}) = (\boldsymbol{v}_1^{(e)} \mid \ldots \mid \boldsymbol{v}_\beta^{(e)})$ and where $\delta_j := (\mu \cdot \nu_j + \ell') - (d_j + 1)$. If one of those check fails, then $\mathcal{V}$ stops and returns Reject. Otherwise, the protocol continues.
3. $\mathcal{P}$ and $\mathcal{V}$ run the protocol $\mathsf{LVCS.Eval}(\mathsf{key}_{\mathsf{LVCS}}, \mathsf{com}, \mathcal{C}, \{\boldsymbol{v}_1^{(e)}, \ldots, \boldsymbol{v}_\beta^{(e)}\}_{e \in E'})$ with coefficients

$$\mathcal{C} = \{\boldsymbol{u}_k \otimes (1, e, \ldots, e^{\mu+\ell'-1})\}_{k \in [1,\beta], e \in E'} \ .$$

   If the output is Reject, $\mathcal{V}$ outputs Reject. Otherwise (the output is Accept), $\mathcal{V}$ outputs Accept.

Fig. 3: SmallWood-PCS: Polynomial Commitment Scheme from SmallWood-LVCS.

Theorem 5. *The PCS depicted in Figure 3 is (t, ε)-extractable binding assuming that the underlying LVCS is (t, ε)-extractable binding.*

Proof. Let us first build the extractor involved in the security proof, then we will upper bound the adversary's advantage in breaking the PCS binding property.

Extraction. Let us build the extractor $\mathcal{E}_{\mathsf{pcs}}$. $\mathcal{E}_{\mathsf{pcs}}$ takes as input the commitment com, together with the set $\mathcal{Q}$ of query-response to the hash function. It proceeds as follows:

- $\mathcal{E}_{\mathsf{pcs}}$ calls the extractor $\mathcal{E}_{\mathsf{lvcs}}$ over $\mathcal{Q}$ and com to get vectors $(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{n_{\mathsf{rows}}})$. If the extraction fails, $\mathcal{E}_{\mathsf{pcs}}$ directly outputs $\perp$.
- $\mathcal{E}_{\mathsf{pcs}}$ re-arranges the extracted values into a set $\{\boldsymbol{A}_j\}_j$ of matrices such that

$$\begin{bmatrix} \boldsymbol{r}_1^\top \\ \boldsymbol{r}_2^\top \\ \vdots \\ \boldsymbol{r}_{n_{\mathsf{rows}}}^\top \end{bmatrix} = \mathsf{Stack}_\beta\big(\boldsymbol{A}_1 | \ldots | \boldsymbol{A}_{n_{\mathsf{pcs}}}\big) \ .$$

- For all (i, j), $\mathcal{E}_{\mathsf{pcs}}$ computes the coefficient $a_{j,i} := \sum_{(k,k') \in S_{j,i}} (\boldsymbol{A}_j)_{k,k'}$, where the set $S_{j,i}$ is defined such as

$$\begin{aligned} S_{j,i} := \{(k, k') \text{ s.t. } i = (k-1) + \mu(k'-1), \text{ with } 1 \le k \le \mu + \ell' \text{ and } 1 \le k' < \nu_j\} \\ \cup \{(k, \nu_j) \text{ s.t. } i = (k-1) + \mu(\nu_j - 1) - \delta_j, \text{ with } 1 \le k \le \mu + \ell'\} \, . \end{aligned}$$

  Then $\mathcal{E}_{\mathsf{pcs}}$ computes the polynomials $P_j(X) := \sum_{i=0}^{d_j} a_{j,i} X^i$ for all j, and it returns $\boldsymbol{P} := (P_1, \ldots, P_{n_{\mathsf{pcs}}})$.

Let us stress that the output polynomials are necessarily of the right degrees. Fix j. The maximal value i such that $S_{j,i} \neq \emptyset$ is $\max\{(\mu + \ell' - 1) + \mu(\nu_j - 1 - 1), (\mu + \ell' - 1) + \mu(\nu_j - 1) - \delta_j\}$. As

$$\nu_j := \left\lceil \frac{d_j + 1 - \ell'}{\mu} \right\rceil \qquad \text{and} \qquad \delta_j := (\mu \cdot \nu_j + \ell') - (d_j + 1) \ ,$$

we get that $0 \le \delta_j < \mu$. We then have that the maximal achievable degree is

$$(\mu + \ell' - 1) + \mu(\nu_j - 1) - \delta_j = (\mu \cdot \nu_j + \ell') + (\mu - \mu) - 1 - \delta_j = d_j \ .$$

Upper bound. We now show that the adversary cannot win the PCS binding game if they do not win the LVCS binding game. As a result, the advantage of the adversary in breaking the PCS binding property is upper bounded by their advantage in breaking the LVCS binding property.

We have that for all $e \in E'$ and $k \in [1, \beta]$,

$$\boldsymbol{v}_k^{(e)} = \sum_{i=0}^{\mu+\ell'-1} e^i \cdot \boldsymbol{r}_{(\mu+\ell')\cdot(k-1)+1+i}$$

by definition of the LVCS extractable binding game, where the vectors $(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{n_{\mathsf{rows}}})$ are those extracted by the LVCS extractor. By reverting the stacking operation, we have that for all $e \in E'$,

$$\begin{aligned} (\hat{\boldsymbol{v}}_1^{(e)} \mid \ldots \mid \hat{\boldsymbol{v}}_{n_{\mathsf{pcs}}}^{(e)}) &= \sum_{i=0}^{\mu+\ell'-1} e^i \cdot (\boldsymbol{r}_{(\mu+\ell')\cdot 0+1+i} \mid \ldots \mid \boldsymbol{r}_{(\mu+\ell')\cdot(\beta-1)+1+i}) \\ &= \sum_{i=0}^{\mu+\ell'-1} e^i \cdot \mathsf{row}_{i+1}(\boldsymbol{A}_1 | \ldots | \boldsymbol{A}_{n_{\mathsf{pcs}}}) \ , \end{aligned}$$

implying that, for all $e \in E'$ and $j \in [1, n_{\mathsf{pcs}}]$, we have $\hat{\boldsymbol{v}}_j^{(e)} = \sum_{i=0}^{\mu+\ell'-1} e^i \cdot \mathsf{row}_{i+1}(\boldsymbol{A}_j)$.

The verifier then checks that the opened evaluations $\{\boldsymbol{p}_e\}_{e\in E'}$ satisfy

$$(\boldsymbol{p}_e)_j = \langle \hat{\boldsymbol{v}}_j^{(e)}, (1, e^{\mu}, \ldots, e^{(\nu_j-2)\mu}, e^{(\nu_j-1)\mu-\delta_j})\rangle$$

for all j and e. The right term of the above expression can be written as

$$\begin{aligned}
&\langle \hat{\boldsymbol{v}}_j^{(e)}, (1, e^{\mu}, \ldots, e^{(\nu_j-1)\mu}, e^{(\nu_j-1)\mu-\delta_j})\rangle \\
&\quad = e^{(\nu_j-1)\mu-\delta_j} \cdot (\hat{\boldsymbol{v}}_j^{(e)})_{\nu_j} + \sum_{k'=1}^{\nu_j-1} e^{\mu(k'-1)} \cdot (\hat{\boldsymbol{v}}_j^{(e)})_{k'} \\
&\quad = e^{(k-1)+(\nu_j-1)\mu-\delta_j} \cdot \left(\sum_{k=1}^{\mu+\ell'} e^{k-1} \cdot (\boldsymbol{A}_j)_{k,\nu_j}\right) + \sum_{k'=1}^{\nu_j-1} e^{\mu(k'-1)} \cdot \left(\sum_{k=1}^{\mu+\ell'} e^{k-1} \cdot (\boldsymbol{A}_j)_{k,k'}\right) \qquad (7) \\
&\quad = \sum_{k=1}^{\mu+\ell'} e^{(k-1)+(\nu_j-1)\mu-\delta_j} \cdot (\boldsymbol{A}_j)_{k,\nu_j} + \sum_{k'=1}^{\nu_j-1}\sum_{k=1}^{\mu+\ell'} e^{(k-1)+\mu(k'-1)} \cdot (\boldsymbol{A}_j)_{k,k'} \\
&\quad = \sum_i e^i \cdot \sum_{(k,k')\in S_{j,i}} (\boldsymbol{A}_j)_{k,k'} = P_j(e) \qquad (8)
\end{aligned}$$

where (7) is obtained using the definition of $\hat{\boldsymbol{v}}_j^{(e)}$ and (8) is obtained using the definition of $S_{j,i}$ used in the PCS extractor. So the verifier checks that $(\boldsymbol{p}_e)_j = P_j(e)$, with $\boldsymbol{P}$ extracted by $\mathcal{E}_{\mathsf{pcs}}$.

To sum up, if the adversary does not break the LVCS binding property and produces a valid PCS opening, the opened evaluations are ensured to be consistent with the extracted polynomial. Therefore, breaking the PCS binding property requires breaking the LVCS binding property. □

Theorem 6. *The PCS depicted in Figure 3 is (t,ξ)-HVZK provided that the underlying LVCS is (t,ξ)-HVZK.*

Proof. By definition, the $\hat{\boldsymbol{v}}_k^{(e)}$ are uniformly distributed vectors satisfying

$$P_j(e) = \langle \hat{\boldsymbol{v}}_j^{(e)}, (1, e^{\mu}, \ldots, e^{(\nu_j-2)\mu}, e^{(\nu_j-1)\mu-\delta_j})\rangle$$

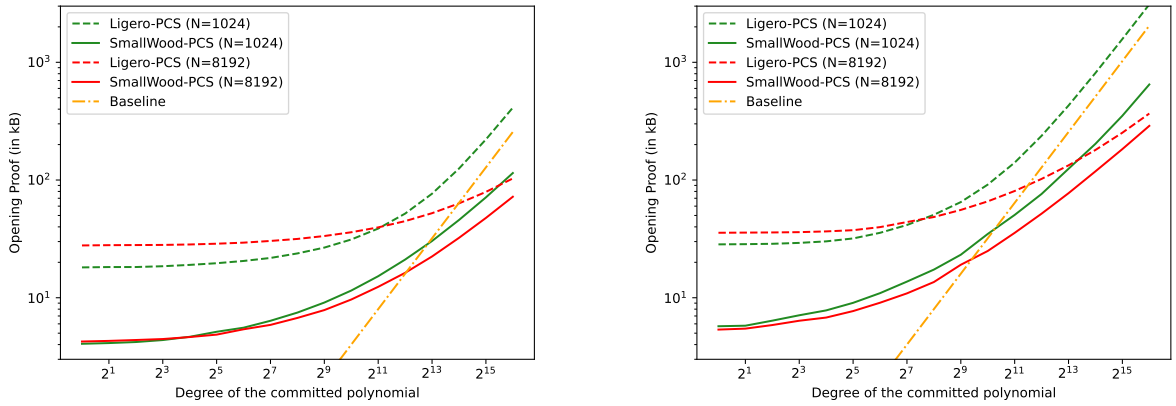
for every $j \in [1, n_{\mathsf{pcs}}]$ and $e \in E'$. The HVZK simulator simply samples these vectors randomly and call the HVZK simulator of the LVCS for the rest. □

4.3 Comparison with the State of the Art

In this section, we compare our polynomial commitment scheme with prior works on hash-based polynomial commitments. Our scheme aims to be used with for "small" polynomials, typically of degree less than 2^{16}. This parameter regime differs from the target of succint polynomial commitments with applications to SNARKs/STARKs. For instance, the authors of Brakedown [GLS+23] compare the state of the art for polynomials of degrees ranging from 2^{13} to 2^{29}. This difference of regime has two implications:

- While focusing on small polynomials, the resulting commitment is often not "succinct" in the sense that it is larger than the committed polynomial. This is not a concern in our context, as our primary goal is to achieve zero-knowledge arguments with small concrete sizes rather than asymptotic succinctness.
- To fairly compare different PCS based on Merkle trees, we need to consider a given size for the evaluation domain $N = |\mathbb{E}|$ (i.e., the number of leaves in the Merkle tree). The policy commonly used in the prior works is to fix the rate of the underlying linear code, i.e., to fix the ratio between the number of committed coefficients per row and the size of the evaluation domain. For example, the authors of Brakedown [GLS+23] use rates equal to 1/4, 1/2 and 38/39. While such a policy makes sense asymptotically, it is irrelevant while focusing on small polynomials, for which it would lead to small Merkle trees and hence larger commitments (because more evaluations must then be opened). For this reason, we consider a fixed size of the evaluation domain $N = |\mathbb{E}|$.

We compare our scheme to Ligero-PCS, the polynomial commitment scheme described in [GLS+23] based on Ligero [AHIV17, AHIV23]. Our scheme is similar to this PCS but relies on the DECS protocol instead of Ligero's proximity test with further adaptation to achieve the zero-knowledge property. For a fair comparison, we also integrate our tweaks to make Ligero-PCS zero-knowledge (these tweaks are summarized in Appendix A). Figure 4 provides the obtained sizes for the commitment plus one evaluation opening with a 128-bit soundness for 32-bit and 256-bit fields. Let us mention that n_{rows}, the number of rows in the underlying LVCS, is a flexible parameter which we select to minimize the proof size.



(a) Over a 32-bit field.

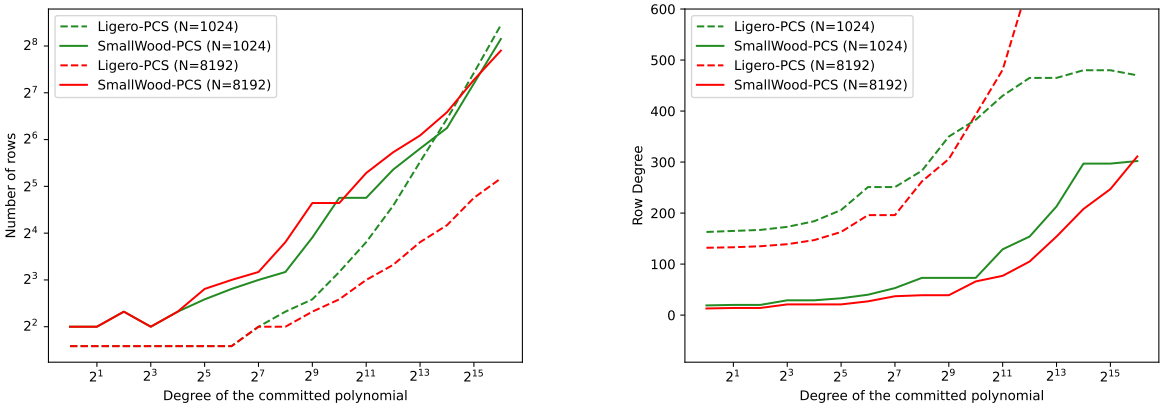
(b) Over a 256-bit field.

Fig. 4: Comparison of hash-based zero-knowledge PCS with 128-bit soundness. The baseline curve gives the cost of sending the entire committed polynomial.

We observe that SmallWood-PCS leads to smaller sizes than Ligero-PCS in this parameter regime. This comes from the fact that the DECS outperforms the Ligero's proximity test for small parameters [FR25b]. We also observe that the smallest sizes achievable by Ligero-PCS are around 20 KB and around 30 KB (for very small polynomials) for respectively 32-bit and 256-bit fields, while SmallWood-PCS can go as low as 4–6 KB. We also observe that SmallWood-PCS crosses the "succinctness threshold" (size for which is becomes more compact than the committed polynomial) around 2^{12} and 2^{13} for the selected parameters (i.e., $N \in [2^{10}, 2^{13}]$) for 32-bit fields and around 2^9 and 2^{10} for 256-bit fields.

We can explain why Ligero-PCS do not perform well for (very) small polynomials. This is because the number of opened leaves in the Merkle tree is at least 128 when targeting a 128-bit soundness. Indeed, opening ℓ leaves leads to a soundness error of $(\frac{1+\rho}{2})^\ell$, where ρ is the rate of the underlying linear code. In our context, the rate is $\rho = \frac{n_{\mathsf{cols}}+\ell}{N}$ where n_{cols} is the row-size parameter of the LVCS and ℓ the number of opened evaluation in the Merkle tree.

While SmallWood-PCS leads to smaller size than Ligero-PCS, let us stress that the comparison made in Figure 4 is yet in favor of Ligero-PCS. Indeed, while for a given N the computation cost of the Merkle tree is the same for both PCS, we can argue that building the data committed as leaves in the Merkle tree is more expensive in Ligero-PCS than in the DECS. Indeed, as illustrated in Figure 5, while optimizing the internal parameters of the LVCS to minimize the size, Ligero-PCS selects fewer rows leading to row polynomials of higher degrees. Evaluating these larger polynomial is hence computationally more intensive. For example, to commit a degree-64 polynomial over a 256-bit field with $N = 8192$, SmallWood-PCS requires evaluating 8 degree-27 polynomials while Ligero-PCS requires evaluating 3 degree-196 polynomials.



(a) Number of rows in the LVCS.

(b) Degree of the row polynomials in the LVCS.

Fig. 5: Comparison of the optimal LVCS parameters for SmallWood-PCS and Ligero-PCS over a 256-bit field. Those are the parameters minimizing the sizes which are used for the results depicted in Figure 4b.

Comparison with further schemes. In [GLS+23], the authors propose Brakedown-PCS. This polynomial commitment scheme is a variant of Ligero-PCS in which the underlying Reed-Solomon code is replaced by a linear code with linear-time encoding. This tweak does not enable them to achieve smaller sizes (in fact, it tends to produce larger sizes in our regime since their code is not a maximum distance separable (MDS) code), but it leads to a scheme for which the prover/committer runs in time linear in the size of the committed polynomial. Therefore, the committer's running time is asymptotically better than Ligero-PCS and SmallWood-PCS, but in the regime of committing to small polynomials, it presents no advantage.

In [VP19], Vlasov and Panarin propose a polynomial commitment scheme based on the FRI protocol [BBHR18a]. However, their scheme is not zero-knowledge. As explained in their article, zero-knowledge is often "achieved at the application level" in FRI-based protocols. We can thus not compare our scheme with this PCS, because our main goal is the zero-knowledge property since we are focusing on small polynomials, and making their scheme zero-knowledge would require a non-trivial analysis.

5 SmallWood Zero-Knowledge Arguments

In this section, we present zero-knowledge arguments constructed using the standard methodology of combining a *polynomial interactive oracle proof* (PIOP) with a polynomial commitment scheme, and then applying the Fiat–Shamir transform to make the protocol non-interactive. To this end, we introduce *parallel and aggregated constraint systems* (PACS), a generalization of the LPPC systems introduced in [FR25b].[5] We then define the PACS PIOP, a generalized and optimized abstraction of Ligero's protocol for verifying arithmetic circuits [AHIV17]. By combining the PACS PIOP with SmallWood-PCS, and applying the Fiat-Shamir transform, we obtain SmallWood-ARK, a hash-based zero-knowledge argument for which we provide a detailed performance and security analysis.

We begin by presenting the syntax of PACS statements and formally introducing the PACS PIOP. Next, we discuss its integration with various hash-based PCS, highlighting the relevance of SmallWood-PCS in this setting. We then describe how the Fiat-Shamir transform is applied to the combination of the PACS PIOP and SmallWood-PCS to construct SmallWood-ARK, along with a summary of its parameters and the resulting proof sizes. Finally, we establish the (straightline-extractable) knowledge soundness and zero-knowledge properties of SmallWood-ARK.

[5] LPPC stands for *(global) linear and parallel polynomial constraints.*

5.1 PACS Statements

In the PACS PIOP, the prover aims to prove knowledge of a witness w arranged as a matrix:

$$w = \begin{bmatrix} w_{1,1} & \dots & w_{1,s} \\ \vdots & \ddots & \vdots \\ w_{n,1} & \dots & w_{n,s} \end{bmatrix}$$

which satisfies two types of constraints:

1. *Parallel polynomial constraints:* for $f_1, \dots, f_{m_1}$ some polynomials from $\mathbb{F}[X_1, \dots, X_n, Y_1, \dots, Y_{n_c}]$ of maximal degree d and $\{\theta_{j,i,k}\}$ constants from $\mathbb{F}$, the witness matrix satisfies:

$$\forall j \in [1, m_1],\ \forall k \in [1, s] \ : \quad f_j(w_{1,k}, \dots, w_{n,k}, \theta_{j,1,k}, \dots, \theta_{j,n_c,k}) = 0 \ .$$

2. *Aggregated parallel polynomial constraints:* for $f'_1, \dots, f'_{m_2}$ some polynomials from $\mathbb{F}[X_1, \dots, X_n, Y_1, \dots, Y_{n_c}]$ of maximal degree d' and $\{\theta'_{j,i,k}\}$ constants from $\mathbb{F}$, the witness matrix satisfies:

$$\forall j \in [1, m_2] \ : \quad \sum_{k=1}^{s} f'_j(w_{1,k}, \dots, w_{n,k}, \theta'_{j,1,k}, \dots, \theta'_{j,n_c,k}) = 0 \ .$$

The PACS statement is hence defined by the m_1 polynomials $\{f_j\}$ and their associated constants $\{\theta_{j,i,k}\}$ and the m_2 polynomials $\{f'_j\}$ and their associated constants $\{\theta'_{j,i,k}\}$. From this definition, we see that each polynomial constraint f_j is verified in parallel on each column of the witness matrix (possibly with different constants) while, for a constraint of the second type, the sum of f'_j applied to each column is constrained to equal 0.

Remark 1. PACS generalizes the LPPC systems introduced in [FR25b]. In the latter, the second type of constraints –referred to as *global linear constraints*– are of the form $\sum_{i,k} a_{i,k} w_{i,k} = t$ for some constants $\{a_{i,k}\}$ and t. Notably, such constraints can be captured as aggregated parallel polynomial constraints of degree 2, with the constants embedded into the θ' arguments of the f'_j function. This generalization enhances the expressiveness of the system and can lead to more compact argument sizes in certain settings, as demonstrated in our application to lattice-based problems in Section 6.

5.2 The PACS Polynomial IOP

We now describe the PACS PIOP for verifying such statements. This protocol is the PIOP version of the LPPC MPC protocol from [FR25b], extended to support aggregated parallel polynomial constraints and further optimized by batching both types of constraints to reduce communication overhead. Let us first recall that a PIOP is an interactive proof in which the prover can send a *polynomial oracle* $[P_1, \dots, P_n]$ to the verifier for polynomials $P_1, \dots, P_n \in \mathbb{F}[X]$. From such a polynomial oracle, the verifier can then query some evaluations. Namely, for any point $e \in \mathbb{F}$, a query e to the oracle provides the verifier with the polynomial evaluations $P_1(e), \dots, P_n(e)$. The number of queries made by the verifier is fixed by the definition of the PIOP protocol.

Let $\Omega = \{\omega_1, \dots, \omega_s\}$ some fixed points of $\mathbb{F}$. Let $\mathbb{S} \subseteq \mathbb{F} \setminus \Omega$, the set of evaluation points that can be queried by the verifier (in practice the definition of $\mathbb{S}$ depends on the underlying PCS; $\mathbb{S}$ must exclude Ω for zero-knowledge to be achieved). Let $\{\Theta_{j,i}(X)\}$ and $\{\Theta'_{j,i}(X)\}$ be the degree-$(s-1)$ polynomials defined by interpolation such that $\Theta_{j,i}(\omega_k) = \theta_{j,i,k}$ and $\Theta'_{j,i}(\omega_k) = \theta'_{j,i,k}$ for all j, i, k. Let $\ell \in \mathbb{N}$ be the number of queries from the verifier to the polynomial oracle and let $\rho \in \mathbb{N}$ some security parameter. Finally, we let d_Q be defined as:

$$d_Q = \max\left(d \cdot (\ell + s - 1) + s - 1 \ , \ d' \cdot (\ell + s - 1)\right) \ , \tag{9}$$

where, as introduced above, $d = \max_j \deg(f_j)$ and $d' = \max_j \deg(f'_j)$.

The PACS PIOP is described in detail in Protocol 6. Let us outline its high-level structure. The prover begins by interpolating each row $(w_{i,1}, \ldots, w_{i,s})$ of the witness matrix into a polynomial $P_i \in \mathbb{F}[X]$ over an extended evaluation domain. Specifically, the domain includes the set $\Omega = \{\omega_1, \ldots, \omega_s\}$ used to encode the witness values, as well as ℓ additional points sampled at random from $\mathbb{F}$. These extra points introduce blinding randomness and ensure zero-knowledge properties while revealing ℓ evaluations to the verifier. The prover hence constructs P_i such that:

$$P_i(\omega_1) = w_{i,1}, \quad \ldots, \quad P_i(\omega_s) = w_{i,s} \ ,$$

and $P_i(r_1), \ldots, P_i(r_\ell)$ are random values at distinct points $r_1, \ldots, r_\ell \in \mathbb{F} \setminus \Omega$. Consequently, each polynomial P_i has degree at most $s + \ell - 1$. The prover additionally samples random polynomials $M_1, \ldots, M_\rho$ of degree d_Q (as defined in Equation (9)), subject to the constraint:

$$\sum_{\omega \in \Omega} M_i(\omega) = 0 \quad \text{for all } i \in [1, \rho] \ .$$

The prover then sends a polynomial oracle $[\boldsymbol{P}, \boldsymbol{M}]$ to the verifier, where $\boldsymbol{P} = (P_1, \ldots, P_n)$ and $\boldsymbol{M} = (M_1, \ldots, M_\rho)$.

Next, the verifier samples random polynomials $\Gamma'_{i,j}(X) \in \mathbb{F}[X]^{(\leq s-1)}$, for all $i \in [1, \rho]$ and $j \in [1, m_1]$, as well as random scalars $\gamma'_{i,j} \in \mathbb{F}$ for all $i \in [1, \rho]$ and $j \in [1, m_2]$. These are sent to the prover under the form of strings $\Gamma'_1, \ldots, \Gamma'_\rho \in \mathbb{F}^{m_1 \cdot s + m_2}$. Using these random values, the prover computes the vector of polynomials $\boldsymbol{Q} = (Q_1, \ldots, Q_\rho)$ where:

$$Q_i(X) = M_i(X) + \sum_{j=1}^{m_1} \Gamma'_{i,j}(X) \cdot F_j(X) + \sum_{j=1}^{m_2} \gamma'_{i,j} \cdot F'_j(X) \ , \tag{10}$$

with

$$F_j(X) := f_j(P_1(X), \ldots, P_n(X), \Theta_{j,1}(X), \ldots, \Theta_{j,n_c}(X)) \ , \tag{11}$$

and

$$F'_j(X) := f'_j(P_1(X), \ldots, P_n(X), \Theta'_{j,1}(X), \ldots, \Theta'_{j,n_c}(X)) \ . \tag{12}$$

The prover sends $\boldsymbol{Q}$ to the verifier, who checks two conditions: (1) that the vector polynomial $\boldsymbol{Q}$ is consistent with the oracle $[\boldsymbol{P}, \boldsymbol{M}]$ on randomly chosen evaluation points, and (2) that the following constraint holds:

$$\forall i \in [1, \rho] \ : \ \sum_{\omega \in \Omega} Q_i(\omega) = 0 \ . \tag{13}$$

If the prover behaves honestly and the constraints are satisfied, Equation (13) always holds, regardless of the verifier's random choices $\{\Gamma_{i,j}(X)\}$ and $\{\gamma'_{i,j}\}$. Indeed, by construction, $F_j(\omega) = 0$ for every $j \in [1, m_1]$ and $\omega \in \Omega$, which implies that $\sum_{\omega \in \Omega} \Gamma_{i,j}(X) \cdot F_j(X) = 0$. Similarly, the polynomials M_i and F'_j are defined so that their sum over Ω vanishes. Therefore, the verifier's check in Equation (13) succeeds.

Soundness. The soundness of the PACS PIOP holds for the following reasons:

1. If at least one parallel polynomial constraint is not satisfied, then the term $\sum_{j=1}^{m_1} \Gamma'_{i,j}(X) \cdot F_j(X)$ in the expression for Q_i does *not* vanish when summed over Ω, except with probability at most $1/|\mathbb{F}|$, where the probability is over the random choice of the polynomials $\{\Gamma'_{i,j}(X)\}$ (assuming uniform distribution). Similarly, if at least one batched parallel polynomial constraint is not satisfied, then the term $\sum_{j=1}^{m_2} \gamma'_{i,j} \cdot F'_j(X)$ in the expression for Q_i also does *not* vanish when summed over Ω, except with probability at most $1/|\mathbb{F}|$, where the probability is over the random choice of the $\{\gamma'_{i,j}\}$. To conclude, since the checking is performed ρ times in parallel using independent randomness, if at least one constraint is not satisfied, then the probability that all the Q_i's vanish when summed over Ω is at most $1/|\mathbb{F}|^\rho$, assuming that $\mathcal{D}_{\Gamma'}$ is the uniform distribution over $\mathbb{F}^{m_1 \cdot s + m_2}$.

1. $\mathcal{P}$ builds random polynomials $P_1, \ldots, P_n \in \mathbb{F}[X]^{(\leq \ell' + s - 1)}$ defined such that
$$\forall\, i \in [1, n],\ P_i(\omega_1) = w_{i,1}, \ldots, P_i(\omega_s) = w_{i,s}\ .$$
2. $\mathcal{P}$ samples ρ random polynomials $M_1, \ldots, M_\rho \in \mathbb{F}[X]^{(\leq d_Q)}$ such that $\sum_{\omega \in \Omega} M_j(\omega) = 0$ for all j.
3. $\mathcal{P}$ sends a polynomial oracle $[\boldsymbol{P}, \boldsymbol{M}]$ to $\mathcal{V}$, where $\boldsymbol{P} = (P_1, \ldots, P_n)$, $\boldsymbol{M} = (M_1, \ldots, M_\rho)$.
4. $\mathcal{V}$ samples $\Gamma'_1, \ldots, \Gamma'_\rho \leftarrow \mathcal{D}_{\Gamma'}$, for $\mathcal{D}_{\Gamma'}$ a probability distribution over $\mathbb{F}^{m_1 \cdot s + m_2}$. $\mathcal{V}$ sends $\Gamma'_1, \ldots, \Gamma'_\rho$ to $\mathcal{P}$.
5. For all $i \in [1, \rho]$, $\mathcal{V}$ and $\mathcal{P}$ parse $\Gamma'_i \in \mathbb{F}^{m_1 \cdot s + m_2}$ as m_1 polynomials $\Gamma'_{i,j}(X) \in \mathbb{F}[X]^{(\leq s-1)}$, for $j \in [1, m_1]$, and m_2 field elements $\gamma'_{i,j} \in \mathbb{F}$ for $j \in [1, m_2]$.
6. $\mathcal{P}$ computes the polynomial Q_i defined by Equation (10), for all $1 \leq i \leq \rho$. $\mathcal{P}$ sends $\boldsymbol{Q} = (Q_1, \ldots, Q_\rho)$ to $\mathcal{V}$.
7. $\mathcal{V}$ samples $E' \leftarrow \binom{\mathbb{S}}{\ell'}$ (random subset $E' \subseteq \mathbb{S}$ of size $|E'| = \ell'$) and queries the oracle $[\boldsymbol{P}, \boldsymbol{M}]$ for evaluations of the polynomials at the points in E'.
8. $\mathcal{V}$ checks that
   (a) $\boldsymbol{Q}$ verifies Equation (10) for all $i \in [1, \rho]$ for the opened evaluations at the points in E',
   (b) $\boldsymbol{Q}$ verifies $\sum_{\omega \in \Omega} Q_i(\omega) = 0$, for every $i \in [1, \rho]$.

Fig. 6: The PACS Polynomial IOP.

2. The relations are checked on ℓ' random points of $\mathbb{S}$. By the Schwartz-Zippel lemma, if one of the relations does not hold then the check passes with probability at most $\binom{d_Q}{\ell'} / \binom{|\mathbb{S}|}{\ell'}$.

Formally, we have the following result:

Theorem 7 (Soundness of the PACS PIOP). *The PACS PIOP described in Figure 6 has soundness error:*
$$\varepsilon \leq \varepsilon_{\mathcal{D}}^{\rho} + \frac{\binom{d_Q}{\ell}}{\binom{|\mathbb{S}|}{\ell}} \quad \textit{with} \quad \varepsilon_{\mathcal{D}} := \max_{v \neq 0} \Pr_{\Gamma' \leftarrow \mathcal{D}_{\Gamma'}}[\langle \Gamma', v \rangle = 0]\ .$$
Namely, for any polynomial oracle $[\boldsymbol{P}, \boldsymbol{M}]$ with underlying witness $(w_{i,k}) = (P_i(\omega_k))$ that fails to satisfy at least one of the PACS constraints, the verifier accepts with probability at most ε. When $\mathcal{D}_{\Gamma'}$ is the uniform distribution over $\mathbb{F}^{m_1 \cdot s + m_2}$, we have $\varepsilon_{\mathcal{D}} = 1/|\mathbb{F}|$ implying:
$$\varepsilon \leq \frac{1}{|\mathbb{F}|^\rho} + \frac{\binom{d_Q}{\ell}}{\binom{|\mathbb{S}|}{\ell}}\ .$$

Proof. For the sake of simplicity, we first address the case of $\mathcal{D}_{\Gamma'}$ being the uniform distribution over $\mathbb{F}^{m_1 \cdot s + m_2}$. We then generalize to any distribution.

The verification process of the PACS PIOP consists in verifying that the vector polynomial $\boldsymbol{Q}$ sent by the prover satisfy the vanishing constraint, that is $\sum_{\omega \in \Omega} Q_i(\omega) = 0$, for every $i \in [1, \rho]$, and that it has been properly constructed from the committed polynomials, following the definition equation (10).

Let us first evaluate the probability that a proper construction of $\boldsymbol{Q}$ satisfy the batched constraint $\sum_{\omega \in \Omega} Q_i(\omega) = 0$ for all $i \in [1, \rho]$ over the random choice of Γ' when the witness does not satisfy the PACS statement.

– If (at least) one parallel polynomial constraint is not satisfied, then there exists (j^*, k^*) such that $f_{j^*}(w_{1,k^*}, \ldots, w_{n,k^*}, \theta_{j^*,1,k^*}, \ldots) \neq 0$ or, equivalently, $F_{j^*}(\omega^*) \neq 0$, where $\omega^* = \omega_{k^*}$. In that case, for all $i \in [1, \rho]$, we have:
$$\sum_{\omega \in \Omega} Q_i(\omega) = \Gamma'_{i,j^*}(\omega^*) \cdot F_{j^*}(\omega^*) + \sum_{\omega \in \Omega \setminus \{\omega^*\}} \Gamma'_{i,j^*}(\omega) \cdot F_{j^*}(\omega) + \sum_{\omega \in \Omega} \Big[M_i(\omega) + \sum_{j \neq j^*} \Gamma'_{i,j}(\omega) \cdot F_j(\omega) + \sum_j \gamma'_{i,j} \cdot F'_j(\omega) \Big]\ .$$

Since $F_{j^*}(\omega^*) \neq 0$, the first term in the right-hand side of the above equation is uniformly random and independent of the other terms by the randomness of Γ'_{i,j^*}. We deduce that for every $i \in [1,\rho]$, the sum $\sum_{\omega\in\Omega} Q_i(\omega)$ vanishes with probability $1/|\mathbb{F}|$.

– If (at least) one batched parallel polynomial constraint is not satisfied, there exists j^* such that

$$\sum_{h\in[1,s],i\in[1,n]} f'_{j^*}(w_{1,k^*},\ldots,w_{n,k^*},\theta'_{j^*,1,k^*},\ldots) \neq 0$$

or, equivalently, $\sum_{\omega\in\Omega} F'_{j^*}(\omega) \neq 0$. In that case, for all $i \in [1,\rho]$, we have:

$$\sum_{\omega\in\Omega} Q_i(\omega) \;=\; \gamma'_{i,j^*} \cdot \sum_{\omega\in\Omega} F'_{j^*}(\omega) \;+\; \sum_{\omega\in\Omega}\Big[M_i(\omega) \;+\; \sum_j \Gamma'_{i,j}(\omega) \cdot F_j(\omega) \;+\; \sum_{j\neq j^*} \gamma'_{i,j} \cdot F'_j(\omega)\Big] \;.$$

Since $\sum_{\omega\in\Omega} F'_{j^*}(\omega) \neq 0$, the first term in the right-hand side of the above equation is uniformly random and independent of the other terms by the randomness of γ'_{i,j^*}. We deduce that for every $i \in [1,\rho]$, the sum $\sum_{\omega\in\Omega} Q_i(\omega)$ vanishes with probability $1/|\mathbb{F}|$.

To sum up, when the witness does not satisfy the PACS statement, the probability to have $\sum_{\omega\in\Omega} Q_i(\omega) = 0$ for all $i \in [1,\rho]$ for properly constructed $Q_1,\ldots,Q_\rho$ is at most $1/|\mathbb{F}|^\rho$.

Let us now evaluate the probability that a wrongly constructed $\boldsymbol{Q}$ passes the verification on the open evaluations. The Schwartz-Zippel lemma states that if a degree-d_Q polynomial relation is not satisfied, then the probability that it holds for ℓ' random distinct evaluation points is at most $\binom{d_Q}{\ell'}/\binom{|\mathbb{S}|}{\ell'}$. We thus obtain a global soundess error of $1/|\mathbb{F}|^\rho + \binom{d_Q}{\ell'}/\binom{|\mathbb{S}|}{\ell'}$ which concludes the proof for $\mathcal{D}_{\Gamma'}$ being the uniform distribution.

When $\mathcal{D}_{\Gamma'}$ is not the uniform distribution, one can verify that satisfying $\sum_{\omega\in\Omega} Q_i(\omega) = 0$ is equivalent to ensuring $\langle \Gamma'_i, v\rangle = 0$ for $\Gamma'_i \leftarrow \mathcal{D}_{\Gamma'}$ and some vector $v \in \mathbb{F}^{m_1\cdot s+m_2}$, which is non-zero whenever a constraint is violated. This completes the proof of the general statement. □

Zero Knowledge. The zero-knowledge property stands from the masking of the vector polynomial $\boldsymbol{Q}$ (with $\boldsymbol{M}$) and the randomness of the vector polynomial $\boldsymbol{P}$ which makes the revealed evaluations independent of the witness. Formally we have:

Theorem 8 (Zero-Knowledge of the PACS PIOP). *The PACS PIOP described in Figure 6 is perfectly honest-verifier zero-knowledge. Namely, there exists a simulator which perfectly simulates a transcript of the protocol together with the answers to the polynomial oracle queries without knowledge of the witness w.*

Proof. We build a simulator $\mathcal{S}$ which outputs a perfect distribution of the protocol transcript and associated answers to the polynomial oracle queries for any challenged batching randomness Γ' and evaluation points E' (given to the simulator as input):

1. $\mathcal{S}$ samples random degree-$(\ell'+s-1)$ polynomials $P_1,\ldots,P_n$. Let $\boldsymbol{P} = (P_1,\ldots,P_n)$.
2. $\mathcal{S}$ samples random degree-d_Q polynomials $Q_1,\ldots,Q_\rho$ such that $\sum_{\omega\in\Omega} Q_i(\omega) = 0$. Let $\boldsymbol{Q} = (Q_1,\ldots,Q_\rho)$.
3. $\mathcal{S}$ parses Γ' as polynomials $\Gamma'_{i,j}(X) \in \mathbb{F}[X]^{(\leq s-1)}$, for $i \in [1,\rho]$ and $j \in [1,m_1]$, and field elements $\gamma'_{i,j} \in \mathbb{F}$ for $i \in [1,\rho]$ and $j \in [1,m_2]$.
4. $\mathcal{S}$ computes $M_1,\ldots,M_\rho$ as:

$$M_i(X) = Q_i(X) - \sum_{j=1}^{m_1} \Gamma'_{i,j}(X)\cdot F_j(X) + \sum_{j=1}^{m_2} \gamma'_{i,j}\cdot F'_j(X) \;,$$

where F_j and F'_j are as defined in (11) and (12). Let $\boldsymbol{M} = (M_1,\ldots,M_\rho)$.

5. $\mathcal{S}$ returns the transcript made of

$$\boldsymbol{Q}, \{\boldsymbol{P}(e), \boldsymbol{M}(e)\}_{e\in E'} \;.$$

First, we can observe that, by construction of $\boldsymbol{Q}$ and $\boldsymbol{M}$, the transcript produced by this simulator is valid, i.e. it passes the verification. Now, let us briefly explained why it is perfectly indistinguishable from a real transcript. In both cases (real transcript and simulated transcript):

- the opened evaluations of $\boldsymbol{P}$ are uniformly random since these polynomials are interpolated from ℓ' random evaluations (in addition to witness coordinates);
- the coordinate polynomials of $\boldsymbol{Q}$ are uniformly random such that $\sum_{\omega\in\Omega} Q_i(\omega) = 0$. While it is direct in the simulated transcript, it comes from the randomness of M_i in the real transcripts.
- Given the polynomial $\boldsymbol{Q}$ and the opened evaluations of $\boldsymbol{P}$, the opened evaluations of $\boldsymbol{M}$ are totally determined by the verification equation.

□

Underlying PCS. The performance of the proof system obtained by combining the PACS PIOP with a concrete PCS highly depends on the latter. This composition involves committing to the vector polynomials $\boldsymbol{P}$ and $\boldsymbol{M}$ using the PCS, in place of the polynomial oracle. Subsequently, the evaluation protocol of the PCS is executed, allowing the prover to reveal the evaluations of these polynomials at random points $e_1, \ldots, e_{\ell'}$ while proving to the verifier that these evaluations are consistent with the commitment. This process replaces the verifier's direct queries to the polynomial oracle. To ensure the zero-knowledge property of the PIOP, the underlying PCS must be zero-knowledge (a.k.a. hiding), such that the verifier learns only the revealed evaluations.

In Ligero [AHIV17], to commit to a vector polynomial $\boldsymbol{P}(X)$, the prover commits to N evaluations $\boldsymbol{P}(e_1), \ldots, \boldsymbol{P}(e_N)$ into a Merkle tree of N leaves. Opening an evaluation then consists in revealing a leaf of the tree along with its authentication path with respect to the committed root. However, the prover should prove that the committed evaluations correspond to the evaluations of a low-degree polynomial. Indeed, without further check, nothing would prevent the prover to commit a polynomial of degree up to $N-1$ which would make the protocol unsound. To proceed, in Ligero the prover runs a proximity test in parallel of the proof system. This approach is pushed further in TCitH-MT [FR25b] which relies on the degree-enforcing commitment scheme (DECS) as recalled in Section 3. This scheme ensures that the committed evaluations correspond to a polynomial of the right degree which improves the performance of the PACS proof system for "relatively small" instances (while asymptotically Ligero remains better).

In both Ligero and TCitH, only the N committed evaluations of the polynomials (as leaves of the Merkle tree) can be opened. This implies that the set $\mathbb{S}$ in the PACS PIOP has a cardinality of N. In contrast, SmallWood-PCS supports the opening of committed polynomials at any point within the field. When the field is large enough, this approach offers three significant advantages:

- *Reduced verification cost.* Using the DECS directly as a PCS restricts openings to $|\mathbb{S}| = N$ evaluation points, with N being the size of the Merkle tree. To achieve λ-bit security, at least ℓ' evaluations must be opened, such that $\binom{N}{\ell'} > 2^\lambda$. Employing a standard PCS, however, enables the prover to open committed polynomials at any point in $\mathbb{F}$. Since $\mathbb{F}$ is substantially larger than N in our context, fewer evaluations are required to maintain the same security level. This reduction translates into lower computational costs for the verifier when checking the polynomial relations (Equation (10)) for the opened evaluations.
- *Lower polynomial degrees.* Using the DECS directly requires increasing the number of evaluations, i.e., using a larger value of ℓ', which leads to a higher degree $s-1+\ell'$ for the committed polynomials. In contrast, employing our PCS reduces the degrees of the committed polynomials which in turns reduces the degree d_Q of $\boldsymbol{Q}$. This degree reduction lowers the communication overhead associated to these polynomials as well as the computational costs for the verifier; it further improves the soundness.
- *Lighter DECS opening.* Although SmallWood-PCS is built uppon the DECS, the latter is not used to directly commit the witness polynomials of the PACS PIOP. Note that for PIOP checking relations of degree d, employing the DECS direcly to commit polynomials results in a soundness error $\binom{d\cdot\ell}{\ell}/\binom{N}{\ell}$. Instead, SmallWood-PCS employs the DECS within an LVCS checking linear relations, i.e., relations of degree $d=1$. This way, fewer evaluations need to be opened in the DECS to reach a given soundness.

5.3 Non-Interactive Zero-Knowledge Argument Scheme

SmallWood-ARK is a zero-knowledge argument scheme for PACS statements obtained by compiling the PACS PIOP with SmallWood-PCS and applying the Fiat-Shamir transform. Figure 7 illustrates the obtained nine-round interactive proof prior to applying the Fiat-Shamir transform. The first three rounds constitute the PCS commitment (equivalently, the LVCS or DECS commitment). The next three rounds correspond to the core PIOP protocol. The final three rounds comprise the PCS opening phase (equivalently, the LVCS opening), with the very last round specifically being the DECS opening. To avoid confusion in the following, we adopt the notations $\boldsymbol{P}_{\mathsf{piop}}$ and $\boldsymbol{M}_{\mathsf{piop}}$ to denote the vector polynomials $\boldsymbol{P}$ and $\boldsymbol{M}$ of the PACS PIOP. Similarly, we use $\boldsymbol{P}_{\mathsf{decs}}$ and $\boldsymbol{M}_{\mathsf{decs}}$ to refer to the vector polynomials $\boldsymbol{P}$ and $\boldsymbol{M}$ of the DECS.

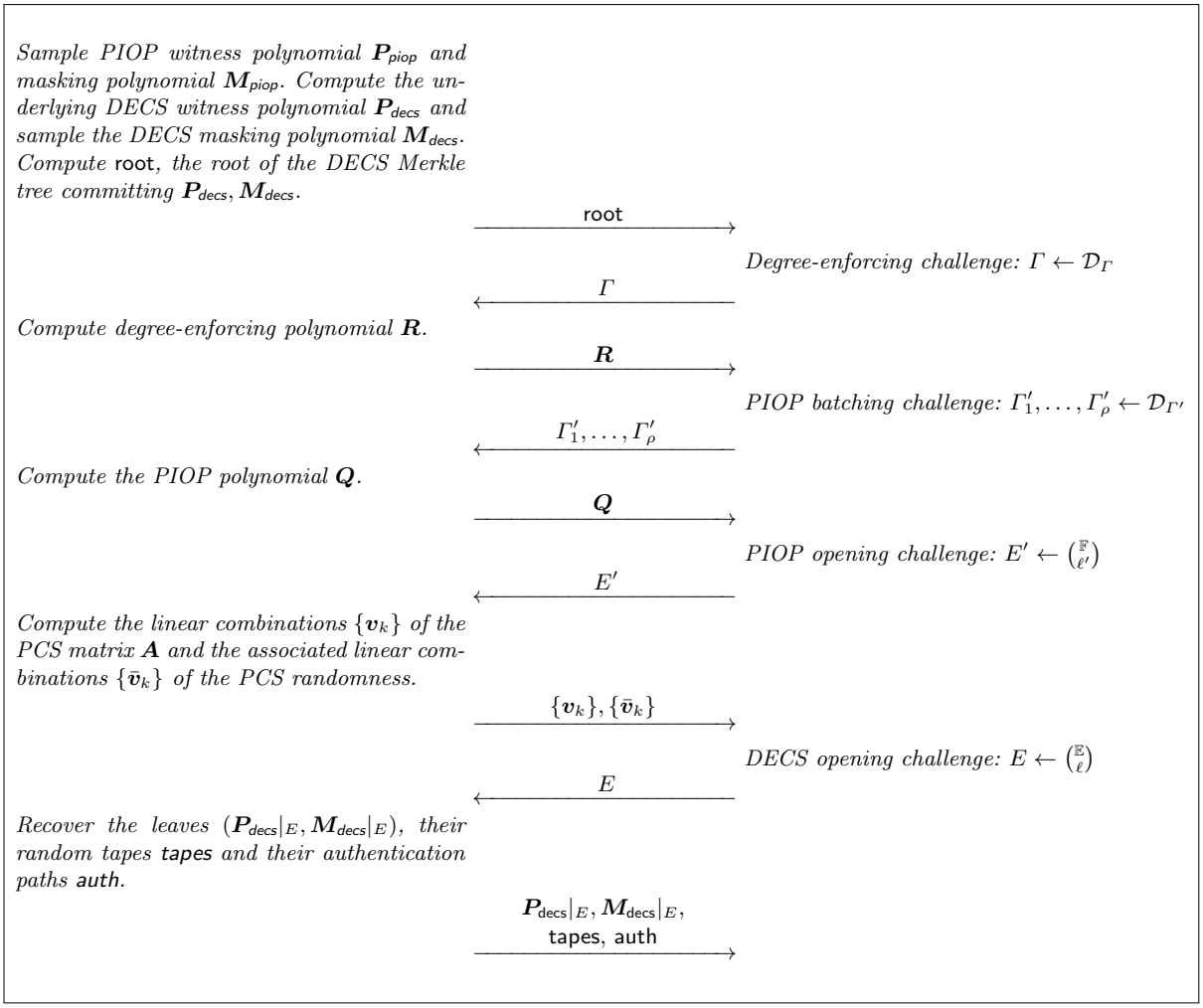


Fig. 7: Nine-round interactive SmallWood proof system.

Assuming that $\mathcal{D}_\Gamma$ and $\mathcal{D}_{\Gamma'}$ are the uniform distributions, this interactive proof achieves round-by-round soundness with the following soundness errors:

$$\varepsilon_1 = \frac{\binom{N}{d_{\mathsf{decs}}+2}}{|\mathbb{F}|^\eta}, \quad \varepsilon_2 = \frac{1}{|\mathbb{F}|^\rho}, \quad \varepsilon_3 = \frac{\binom{d_Q}{\ell'}}{\binom{|\mathbb{F}|}{\ell'}}, \quad \varepsilon_4 = \frac{\binom{n_{\mathsf{cols}}+\ell-1}{\ell}}{\binom{N}{\ell}}, \tag{14}$$

where ε_1 corresponds to degree-enforcing soundness error (arising in the DECS commitment), ε_2 corresponds to the first-round soundness error of the PIOP (batching of constraints), ε_3 corresponds to the second-round soundness error of the PIOP (evaluation queries), and ε_4 corresponds to the PCS/LVCS opening soundness error. To achieve a λ-bit security, one shall constrain the chosen parameters to ensure that these soundness errors are at most $2^{-\lambda}$ (see Table 1 below for an exhaustive list of the SmallWood parameters).To further reduce the proof size, we employ the technique of *grinding* [Sta21]. Concretely, for parameters $\kappa_1, \kappa_2, \kappa_3, \kappa_4 \in \mathbb{N}$, we embed a κ_i-bit proof-of-work into the i-th Fiat-Shamir hash computation, for each $i \in \{1, 2, 3, 4\}$. This relaxation allows each individual soundness error ε_i to satisfy the bound $\varepsilon_i \leq 2^{-\lambda+\kappa_i}$, rather than the stricter original constraint.

To achieve strong concrete security, SmallWood-ARK employs a random salt. During proof generation, the prover begins by selecting a random salt $\mathsf{salt} \in \{0,1\}^{2\lambda}$. This salt is then used as auxiliary input both when hashing the leaves of the Merkle tree in the PCS and when deriving the first Fiat-Shamir hash.

Application of the Fiat-Shamir Transform. Our concrete application of the Fiat-Shamir transform with grinding is illustrated in Figure 8. In each challenge round, the challenge is derived by first hashing the previous Fiat-Shamir output h_{i-1} together with the prover's latest message msg_i, collectively denoted $h_i^{(\mathrm{in})}$. Grinding is then applied as follows: a counter ctr_i is initialized to zero, and one computes

$$\mathsf{XOF}_i : (h_i^{(\mathrm{in})}, \mathsf{ctr}_i) \mapsto (h_i, v) \in \{0,1\}^{2\lambda} \times \{0,1\}^{\kappa_i} \ .$$

If the extracted value v equals the all-zero string 0^{κ_i}, then h_i is accepted as the output hash. Otherwise, the counter ctr_i is incremented and the process is repeated. Once grinding succeeds, the resulting h_i is used to derive the i-th challenge chal_i via a XOF, and it also serves as input to the next Fiat-Shamir round. In this iterative definition, we define $h_0 := (\mathsf{salt}, \mathsf{x}_{\mathrm{PACS}})$, where $\mathsf{x}_{\mathrm{PACS}}$ is a string encoding the underlying PACS statement. When SmallWood is used as the basis of a signature scheme, the final hash may additionally absorb the message to be signed.

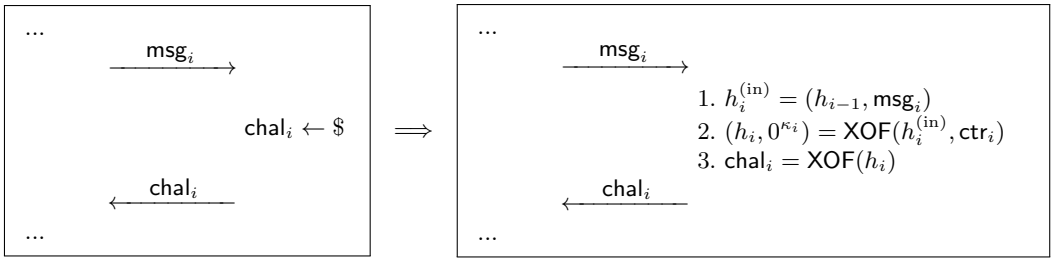


Fig. 8: Illustration of our concrete Fiat-Shamir transformation with grinding.

The following diagram summarizes the four sequential applications of the Fiat-Shamir transform to the SmallWood interactive proof (Figure 7):

$$\begin{array}{lcccc}
h_1^{(\mathrm{in})} = (h_0, \mathsf{root}) & \to & h_1 & \to & \Gamma \\
h_2^{(\mathrm{in})} = (h_1, \boldsymbol{R}) & \to & h_2 & \to & \Gamma' \\
h_3^{(\mathrm{in})} = (h_2, \boldsymbol{Q}_1, \boldsymbol{Q}_2) & \to & h_3 & \to & E' \\
h_4^{(\mathrm{in})} = (h_3, \{\boldsymbol{v}_k\}, \{\bar{\boldsymbol{v}}_k\}, \mathsf{msg}) & \to & h_4 & \to & E
\end{array}$$

The arrows in the leftmost column represent the hash derivation with grinding (through $\mathsf{XOF}_1, \ldots, \mathsf{XOF}_4$), while those in the rightmost column correspond to the derivation of the challenges (through $\mathsf{XOF}'_1, \ldots,$

XOF'_4). Here, msg is an optional input message (when SmallWood-ARK is used as a signature scheme) which may be set to the empty string.

We note that the four counters $\mathsf{ctr}_1, \mathsf{ctr}_2, \mathsf{ctr}_3, \mathsf{ctr}_4$ are included in the proof to enable efficient verification. Rather than recomputing the proof-of-work, the verifier simply checks that the corresponding hash output contains the required all-zero prefix.

Summary of Parameters. The different parameters of the scheme are summarized in Table 1. Besides the parameters of the PACS statement, which are given as input of the scheme, the parameters are either chosen (to satisfy the target soundness while, e.g., reducing the proof size) or defined from the other parameters. This is depicted in the last column of the table. For the basic version of SmallWood-ARK, we use the uniform distribution over $\mathcal{D}(\mathbb{F}^{\eta \times n})$ for the DECS and the number of evaluation set for the PACS PIOP has cardinality $|\mathbb{S}| = |\mathbb{F}| - s$.

Table 1: Parameters of SmallWood-ARK.

Parameters of the PACS Statement:		
$\mathbb{F}$	Base field	
s	Number of columns in the witness matrix (packing factor)	
n	Number of rows in the witness matrix	
d	Maximal degree of parallel polynomial constraints	
m_1	Number of parallel polynomial constraints	
m_2	Number of aggregated parallel polynomial constraints	
Parameters of the PACS PIOP:		
ℓ'	Number of oracle queries (polynomial evaluations)	*chosen*
ρ	Number of parallel repetitions	*chosen*
d_Q	Degree of $\boldsymbol{Q}$	(9)
Parameters of the PCS:		
n_{pcs}	Number of committed polynomials	$n_{\mathsf{pcs}} = n + \rho$
$\{d_j\}_{1 \le j \le n}$	Degrees of the PIOP witness polynomials $P_1, \ldots, P_n$	$d_j = \ell' + s - 1$
$\{d_j\}_{n < j \le n+\rho}$	Degrees of the PIOP masking polynomials $M_1, \ldots, M_\rho$	$d_j = d_Q$
μ	Number of coefficient rows in the matrices $\{\boldsymbol{A}_j\}$	*chosen*
$\{\nu_j\}$	Number of columns of the matrices $\{\boldsymbol{A}_j\}$	$\nu_j = \lceil (d_j + 1 - \ell')/\mu \rceil$
β	Stacking factor (number of $(\mu + \ell')$-row layers in $\boldsymbol{A}$)	*chosen*
Parameters of the LVCS:		
n_{rows}	Number of committed row vectors	$n_{\mathsf{rows}} = \beta \cdot (\mu + \ell')$
n_{cols}	Size of committed row vectors	$n_{\mathsf{cols}} = \lceil (\sum_j \nu_j)/\beta \rceil$
m	Number of LVCS queries	$m = \beta \cdot \ell'$
ℓ	Number of DECS evaluation queries	*chosen*
Parameters of the DECS:		
n_{decs}	Number of committed polynomials	$n_{\mathsf{decs}} = n_{\mathsf{rows}}$
d_{decs}	Degree of committed polynomials	$d_{\mathsf{decs}} = n_{\mathsf{cols}} + \ell - 1$
N	Size of the evaluation domain (number of Merkle leaves)	*chosen*
η	Number of parallel repetitions of the degree-enforcing round	*chosen*
$\kappa_1, \kappa_2, \kappa_3, \kappa_4$	Grinding parameters	*chosen*

Proof Composition and Size. A naive SmallWood-ARK proof is composed of the salt (salt), the grinding counters (ctr_1, ctr_2, ctr_3, ctr_4) and the prover-to-verifier messages (see Figure 7). We obtain the following

(naive) proof size:

$$|\pi_{\text{naive}}| = \underbrace{128}_{\mathsf{ctr}_1,\ldots,\mathsf{ctr}_4} + \underbrace{4\lambda}_{\mathsf{salt},\,\mathsf{root}} + \underbrace{\eta\cdot(d_{\mathsf{decs}}+1)\cdot\log_2|\mathbb{F}|}_{\boldsymbol{R}} + \underbrace{\rho\cdot d_Q\cdot\log_2|\mathbb{F}|}_{\boldsymbol{Q}} + \underbrace{\ell'\cdot\left(\sum_{i=1}^{n_{\mathsf{pcs}}}\nu_i\right)\cdot\log_2|\mathbb{F}|}_{\{\boldsymbol{v}_1^{(e)},\ldots,\boldsymbol{v}_\beta^{(e)}\}_{e\in E'}}$$
$$+ \underbrace{m\cdot\ell\cdot\log_2|\mathbb{F}|}_{\bar{\boldsymbol{v}}_1,\ldots,\bar{\boldsymbol{v}}_m} + \underbrace{\ell\cdot n_{\mathsf{rows}}\cdot\log_2|\mathbb{F}|}_{\boldsymbol{P}_{\mathsf{decs}}|_E} + \underbrace{\ell\cdot\eta\cdot\log_2|\mathbb{F}|}_{\boldsymbol{M}_{\mathsf{decs}}|_E} + \underbrace{2\lambda\cdot\mathsf{nodes}(\ell,N)}_{\mathsf{auth}} + \underbrace{\ell\cdot\lambda}_{\mathsf{tapes}},$$

where $\mathsf{nodes}(\ell, N)$ denotes the average number of nodes in the authentication paths for the opening of ℓ leaves in a Merkle tree of size N (omitting the root). This can be optimized, as some elements of the proof are redundant. Specifically:

(1) The ℓ evaluations of $\boldsymbol{R}$ (specifically, $\boldsymbol{R}|_E$) can be inferred from the evaluations of $\boldsymbol{P}_{\mathsf{decs}}|_E$ and $\boldsymbol{M}_{\mathsf{decs}}|_E$. Therefore, only $d_{\mathsf{decs}}+1-\ell$ additional evaluations of $\boldsymbol{R}$ are needed in the proof to fully reconstruct $\boldsymbol{R}$.
(2) The ℓ' evaluations of $\boldsymbol{Q}$ (specifically, $\boldsymbol{Q}|_{E'}$) can be inferred from the evaluations of $\boldsymbol{P}_{\mathsf{piop}}|_{E'}$ and $\boldsymbol{M}_{\mathsf{piop}}|_{E'}$ (which are derived from the $\{\boldsymbol{v}_k^{(e)}\}$ per the PCS definition). Therefore, only $d_Q-\ell'$ additional evaluations of $\boldsymbol{Q}$ are needed in the proof to fully reconstruct $\boldsymbol{Q}$.
(3) The vector polynomial $\boldsymbol{Q}=(Q_1,\ldots,Q_\rho)$ must satisfy $\sum_{\omega\in\Omega}Q_i(\omega)=0$ for all $i\in[1,\rho]$ for the proof to be accepted. We can thus skip one additional coefficient (or evaluation) per Q_i and reconstruct them from this constraint.
(4) For each evaluation point $e\in E$, the LVCS linear combinations $\boldsymbol{v}_1^{(e)},\ldots,\boldsymbol{v}_\beta^{(e)}$ and the DECS evaluations $\boldsymbol{P}_{\mathsf{decs}}(e)$ and $\boldsymbol{M}_{\mathsf{decs}}(e)$ must satisfy m linear relations for the proof to be accepted (see the LVCS verification). Consequently, m coefficients from $\boldsymbol{P}_{\mathsf{decs}}(e)$ and $\boldsymbol{M}_{\mathsf{decs}}(e)$ can be omitted and later reconstructed using the remaining entries and the vectors $\boldsymbol{v}_1^{(e)},\ldots,\boldsymbol{v}_\beta^{(e)}$.

The redundancy described above can be eliminated from the proof, resulting in a more compact proof π. In doing so, an additional Fiat-Shamir hash (specifically, h_3) is required to enable the verifier to derive the set E' (which is necessary for reconstructing $\boldsymbol{Q}$). The resulting proof has the following size:

$$|\pi| = |\pi_{\text{naive}}| + 2\lambda - \Big(\underbrace{\eta\cdot\ell}_{(1)} + \underbrace{\rho\cdot(\ell'+1)}_{(2),\,(3)} + \underbrace{m\cdot\ell}_{(4)}\Big)\cdot\log_2|\mathbb{F}|\ .$$

Security. The following theorems state the straightline-extractable knowledge soundness and zero-knowledge security of SmallWood-ARK according to Definition 3.

Theorem 9. *Let $\mathcal{D}_\Gamma$ and $\mathcal{D}_{\Gamma'}$ be the uniform distributions over $\mathbb{F}^{\eta\times n_{\mathsf{decs}}}$ and $\mathbb{F}^{m_1\cdot s+m_2}$. In the Random Oracle Model (ROM), where* Hash, XOF_i *and* XOF'_i *for $i\in\{1,2,3,4\}$ are modeled as independent random oracles, SmallWood-ARK is (t,ε)-straightline-extractable knowledge sound with*

$$\varepsilon \le \frac{Q_0^2+Q_1^2+Q_2^2+Q_3^2+Q_4^2}{2^{2\lambda}} + \frac{Q_1\cdot\varepsilon_1}{2^{\kappa_1}} + \frac{Q_2\cdot\varepsilon_2}{2^{\kappa_2}} + \frac{Q_3\cdot\varepsilon_3}{2^{\kappa_3}} + \frac{Q_4\cdot\varepsilon_4}{2^{\kappa_4}}\ ,$$

for $\varepsilon_1,\varepsilon_2,\varepsilon_3,\varepsilon_4$ given in Equation (14), and $t\ge Q_0+Q_1+Q_2+Q_3+Q_4$, for every $Q_0,\ldots,Q_4\in\mathbb{N}$ (representing the numbers of queries made by the adversary to the random oracles Hash, XOF_1, …, XOF_4*).*

Proof. We show that if an adversary $\mathcal{A}$ can produce a valid SmallWood-ARK proof π for a statement x, then one can extract a valid witness w from the random oracle queries $\mathcal{Q}$ made by $\mathcal{A}$, unless a failure event Fail occurs, implying $\varepsilon\le\Pr[\mathsf{Fail}]$. This event shall be the union of five events Fail_0, …, Fail_4 such that

$$\Pr[\mathsf{Fail}_0] \le \frac{Q_0^2+Q_1^2+Q_2^2+Q_3^2+Q_4^2}{2^{2\lambda}} \tag{15}$$

and

$$\Pr[\mathsf{Fail}_i \mid (\neg\,\mathsf{Fail}_0)\wedge\ldots\wedge(\neg\,\mathsf{Fail}_{i-1})] \le \frac{Q_i\cdot\varepsilon_i}{2^{\kappa_i}}$$

for every $i \in \{1,2,3,4\}$, thus implying

$$\Pr[\mathsf{Fail}] = \Pr[\mathsf{Fail}_0 \vee \ldots \vee \mathsf{Fail}_4] \leq \frac{Q_0^2 + Q_1^2 + Q_2^2 + Q_3^2 + Q_4^2}{2^{2\lambda}} + \frac{Q_1 \cdot \varepsilon_1}{2^{\kappa_1}} + \frac{Q_2 \cdot \varepsilon_2}{2^{\kappa_2}} + \frac{Q_3 \cdot \varepsilon_3}{2^{\kappa_3}} + \frac{Q_4 \cdot \varepsilon_4}{2^{\kappa_4}} \ .$$

We will successively introduce each of the five failure events, showing that one of them should be satisfied for the proof to be valid in case of extraction failure, and proving the above inequalities.

Without loss of generality, all the random oracle queries involved in the verification of the forged proof π are in $\mathcal{Q}$. We denote $\mathcal{Q}_0 \subseteq \mathcal{Q}$ the set of queries made to Hash (the hash function involved in the DECS) and $\mathcal{Q}_i \subseteq \mathcal{Q}$ the set of queries made to XOF_i (the XOF for the i-th Fiat-Shamir challenge) such that $Q_0 = |\mathcal{Q}_0|$, $\ldots$, $Q_4 = |\mathcal{Q}_4|$.

Failure event 0. We first denote Fail_0 the event that a collision occurs in $\mathcal{Q}$, namely two different queries to a random oracle leads to the same answer. This occurs with probability:

$$\Pr[\mathsf{Fail}_0] \leq \frac{Q_0^2 + Q_1^2 + Q_2^2 + Q_3^2 + Q_4^2}{2^{2\lambda}} \ .$$

Failure event 1. Each valid query $q_1 = (\mathsf{salt}, \mathsf{x}_{\text{PACS}}, \mathsf{root}) \in \mathcal{Q}_1$ defines a partial DECS commitment. A valid query q_1 is one for which the output (h_1, v) of XOF_1 satisfies $v = 0$ (i.e., it verifies the proof of work) and such that root is the root of a (partial) Merkle tree defined by queries from $\mathcal{Q}_0$. We only assume the Merkle tree to be partial, meaning that at least one leaf of the right format is among the queries in $\mathcal{Q}_0$ as well as all the nodes from this leaf to the root. We then define $(\boldsymbol{P}_{\mathsf{decs}}^{(q_1)}, \boldsymbol{M}_{\mathsf{decs}}^{(q_1)})$ as the polynomials interpolated from all the Merkle leaves of the right format in $\mathcal{Q}_0$ which are correctly incoming the root in q_1. The answer of the query q_1 further determines a random DECS challenge Γ (through the application of XOF_1' to the answer h_1 to q_1). From this challenge and the polynomials $(\boldsymbol{P}_{\mathsf{decs}}^{(q_1)}, \boldsymbol{M}_{\mathsf{decs}}^{(q_1)})$, one can derive the DECS commitment answer $\boldsymbol{R}^{(q_1)}$.

Informally, the failure event Fail_1 occurs if there exists a valid query $q_1 \in \mathcal{Q}_1$ such that one can produce a valid DECS commitment which opens to evaluations that interpolate to a polynomial of degree strictly greater than the prescribed bound d_{decs}. Specifically, this happens when there exists an evaluation set $E \subseteq \mathbb{E}$ of size $|E| \geq d_{\mathsf{decs}} + 2$ for which:

- the pair of evaluation vectors $(\boldsymbol{P}_{\mathsf{decs}}^{(q_1)}|_E, \boldsymbol{M}_{\mathsf{decs}}^{(q_1)}|_E)$ interpolates to a vector polynomial of degree strictly greater than d_{decs}, yet
- the corresponding committed values $\boldsymbol{R}^{(q_1)}|_E$ interpolate to a vector polynomial of degree at most d_{decs}.

This constitutes a soundness violation of the DECS. By the soundness guarantee of the DECS (see Theorem 1), and assuming no hash collisions occurred (i.e., $\neg\mathsf{Fail}_0$), this event occurs with probability at most ε_1 for any individual valid query q_1. Furthermore, due to the proof-of-work mechanism, each query q_1 is only valid with probability at most $2^{-\kappa_1}$. Therefore, the overall probability of Fail_1 satisfies:

$$\Pr[\mathsf{Fail}_1 \mid \neg\mathsf{Fail}_0] \leq \frac{Q_1 \cdot \varepsilon_1}{2^{\kappa_1}} \ .$$

Failure event 2. A query $q_2 \in \mathcal{Q}_2$ is said to be valid if the output (h_2, v) of XOF_2 satisfies $v = 0$ (i.e., it meets the proof-of-work condition), and the input to XOF_2 is of the correct format $(h_1, \boldsymbol{R})$, where h_1 is the output of a valid query from $\mathcal{Q}_1$. Each such valid query defines a complete DECS commitment.

Assuming that neither Fail_0 nor Fail_1 occur, it follows that, for each valid q_2, the DECS opening corresponds to a pair of vector polynomials $(\boldsymbol{P}_{\mathsf{decs}}^{(q_2)}, \boldsymbol{M}_{\mathsf{decs}}^{(q_2)})$ of degree at most d_{decs}. The polynomial $\boldsymbol{P}_{\mathsf{decs}}^{(q_2)}$ further defines a witness polynomial $\boldsymbol{P}_{\mathsf{piop}}^{(q_2)}$ for the PIOP protocol, along with its associated masking polynomial $\boldsymbol{M}_{\mathsf{piop}}^{(q_2)}$.

We now describe the extractor. It proceeds by scanning $\mathcal{Q}$ for valid queries q_2. For each such query, it checks whether the associated witness polynomial $\boldsymbol{P}_{\mathsf{piop}}^{(q_2)}$ satisfies all the PACS constraints. If it does, the extractor outputs a valid witness by evaluating $\boldsymbol{P}_{\mathsf{piop}}^{(q_2)}$ on the designated witness support Ω.

Back to our failure event: by assumption, the extraction fails, implying that for all valid queries q_2, the polynomial $\boldsymbol{P}_{\mathsf{piop}}^{(q_2)}$ does not satisfy the full set of PACS constraints. The response of query q_2 determines a random batching challenge Γ' (through the application of XOF_2' to the answer h_2 to q_2), as used in the PIOP protocol. The event Fail_2 occurs if, despite the PACS constraints being violated, there exist a vector polynomial $\boldsymbol{Q}$ (of prescribed degrees d_Q) that satisfies the PACS verification equations (see Equation (10)) with respect to $\boldsymbol{P}_{\mathsf{piop}}^{(q_2)}$, $\boldsymbol{M}_{\mathsf{piop}}^{(q_2)}$.

By the soundness of the PACS protocol, and under the assumption that no earlier failure events occurred, this happens with probability at most ε_2 for any valid query q_2, over the randomness of Γ'. Since each such query is only valid with probability at most $2^{-\kappa_2}$ due to the proof-of-work, we conclude:

$$\Pr[\mathsf{Fail}_2 \mid (\neg\mathsf{Fail}_0) \wedge (\neg\mathsf{Fail}_1)] \leq \frac{Q_2 \cdot \varepsilon_2}{2^{\kappa_2}} \ .$$

Failure event 3. A query $q_3 \in \mathcal{Q}_3$ is considered valid if the output (h_3, v) of XOF_3 satisfies $v = 0$ (i.e., it passes the proof-of-work check), and the input to XOF_3 is of the expected format $(h_2, \boldsymbol{Q})$, where h_2 is the output of a valid query $q_2 \in \mathcal{Q}_2$. Let us denote by $\boldsymbol{Q}^{(q_3)}$ the vector polynomial associated with the query q_3.

Assuming that Fail_0, Fail_1, and Fail_2 do not occur, it follows that for each valid query $q_3 \in \mathcal{Q}_3$ (with underlying $q_2 \in \mathcal{Q}_2$), the polynomial $\boldsymbol{Q}^{(q_3)}$, together with the committed polynomials $\boldsymbol{P}_{\mathsf{piop}}^{(q_2)}$ and $\boldsymbol{M}_{\mathsf{piop}}^{(q_2)}$, do *not* satisfy the PACS verification equations globally.

Let E' be the random evaluation set determined by q_3 (through the application of XOF_3' to the answer h_3 to q_3) which corresponds to the second challenge of the PIOP. The failure event Fail_3 occurs if, despite the above polynomials not satisfying all the PACS constraints, their evaluations on the specific challenge set E' *do* satisfy the PACS verification equations (see Equation (10)).

According to the soundness of the PACS protocol, and conditioned on the non-occurrence of previous failure events, the probability of this happening for a single valid query q_3 is at most ε_3, over the randomness of E'. Taking into account the proof-of-work condition (which restricts valid queries to probability at most $2^{-\kappa_3}$), we obtain:

$$\Pr[\mathsf{Fail}_3 \mid (\neg\mathsf{Fail}_0) \wedge (\neg\mathsf{Fail}_1) \wedge (\neg\mathsf{Fail}_2)] \leq \frac{Q_3 \cdot \varepsilon_3}{2^{\kappa_3}} \ .$$

Failure event 4. A query $q_4 \in \mathcal{Q}_4$ is considered valid if the output (h_4, v) of XOF_4 satisfies $v = 0$ (i.e., it passes the proof-of-work condition), and the input to XOF_4 is of the expected format $(h_3, \{\boldsymbol{v}_k\}, \{\bar{\boldsymbol{v}}_k\}, \mathsf{msg})$, where h_3 is the output of a valid query $q_3 \in \mathcal{Q}_3$. We denote by $\{\boldsymbol{v}_k^{(q_4)}\}$ and $\{\bar{\boldsymbol{v}}_k^{(q_4)}\}$ the two sets of vectors associated with the query q_4.

Assuming that Fail_0, Fail_1, Fail_2, and Fail_3 do not occur, it follows that for each valid q_4, the vector pairs $\{\boldsymbol{v}_k^{(q_4)}\}, \{\bar{\boldsymbol{v}}_k^{(q_4)}\}$ do not form a valid LVCS statement. In particular, there exists at least one evaluation point $e \in E'$ at which the corresponding opening of the PCS must be incorrect.

The query q_4 further determines the random evaluation set E (through the application of XOF_4' to the answer h_4 to q_4), which constitutes the final Fiat-Shamir challenge of the LVCS protocol and is used to verify the DECS opening. The failure event Fail_4 occurs if, despite the underlying vectors being invalid, the opening nevertheless passes the LVCS verification on the randomly chosen evaluation set E.

By the soundness of the LVCS scheme, and conditioned on the non-occurrence of the prior failure events, the probability of this happening for a single valid query q_4 is at most ε_4, over the randomness of E. Accounting for the proof-of-work filter, which ensures that a query is only valid with probability at most $2^{-\kappa_4}$, we conclude:

$$\Pr[\mathsf{Fail}_4 \mid (\neg\mathsf{Fail}_0) \wedge (\neg\mathsf{Fail}_1) \wedge (\neg\mathsf{Fail}_2) \wedge (\neg\mathsf{Fail}_3)] \leq \frac{Q_4 \cdot \varepsilon_4}{2^{\kappa_4}} \ .$$

We have shown that if no valid witness can be extracted from the set of random oracle queries $\mathcal{Q}$ (as described above), then one of the five failure events $\mathsf{Fail}_0, \ldots, \mathsf{Fail}_4$ must occur in order for the forged proof π to pass the verification. It follows that the knowledge soundness error satisfies $\varepsilon \leq \Pr[\mathsf{Fail}]$, which concludes the proof. □

Theorem 10. *In the Random Oracle Model (ROM), where* Hash, XOF_i *and* XOF'_i *for* $i \in \{1,2,3,4\}$ *are modeled as independent random oracles,* SmallWood-ARK *is* (t,ξ)*-zero-knowledge, with*

$$\xi \leq Q \cdot \left(\frac{1}{2^\lambda} + \frac{1}{2^{2\lambda}}\right)$$

and $t \geq Q$*, for every* $Q \in \mathbb{N}$ *(representing the number of RO queries made by the adversary).*

Proof. The zero-knowledge property of SmallWood-ARK follows naturally in the ROM from the zero-knowledge properties of the PACS PIOP and SmallWood-PCS, together with the ability to program the random oracles used in the Fiat–Shamir transform.

In detail, the zero-knowledge simulator for SmallWood-ARK proceeds as follows:

1. Invoke the zero-knowledge simulator of the PACS PIOP (Theorem 8) to obtain a perfect simulation of the transcript:
$$\Gamma', \boldsymbol{Q}_1, \boldsymbol{Q}_2, E', \boldsymbol{P}_{\mathsf{piop}|_{E'}}, \boldsymbol{M}_{\mathsf{piop}|_{E'}} \ .$$

2. Run the zero-knowledge simulator of SmallWood-PCS on input $E', \boldsymbol{P}_{\mathsf{piop}|_{E'}}, \boldsymbol{M}_{\mathsf{piop}|_{E'}}$ to obtain a simulated output:
$$\{\boldsymbol{v}_k\}, \{\bar{\boldsymbol{v}}_k\}, E, \boldsymbol{P}_{\mathsf{decs}}|_E, \mathsf{com}_{\mathsf{decs}}, \pi_{\mathsf{decs}} \ ,$$
where:
$$\mathsf{com}_{\mathsf{decs}} = (\mathsf{root}, \Gamma, \boldsymbol{R}) \ , \quad \pi_{\mathsf{decs}} = (\boldsymbol{M}_{\mathsf{decs}}|_E, \mathsf{auth}, \mathsf{tapes}) \ .$$
By the zero-knowledge property of SmallWood-PCS (Theorems 2, 4 and 6), this simulation is ξ_{pcs}-indistinguishable from a real transcript, with
$$\xi_{\mathsf{pcs}} = \xi_{\mathsf{lvcs}} = \xi_{\mathsf{decs}} \leq \frac{Q}{2^\lambda} \ .$$

3. Using the simulated values, program the random oracles as follows:
   - Sample a random salt: $\mathsf{salt} \leftarrow \{0,1\}^{2\lambda}$.
   - Sample $h_1 \leftarrow \{0,1\}^{2\lambda}$ and program the oracles $\mathsf{XOF}_1(\mathsf{salt}, \mathsf{x}_{\mathrm{PACS}}, \mathsf{root}) := h_1$, and $\mathsf{XOF}'_1(h_1) := \Gamma$.
   - Sample $h_2 \leftarrow \{0,1\}^{2\lambda}$ and program the oracles $\mathsf{XOF}_2(h_1, \boldsymbol{R}) := h_2$, and $\mathsf{XOF}'_2(h_2) := \Gamma'$.
   - Sample $h_3 \leftarrow \{0,1\}^{2\lambda}$ and program the oracles $\mathsf{XOF}_3(h_2, \boldsymbol{Q}_1, \boldsymbol{Q}_2) := h_3$, and $\mathsf{XOF}'_3(h_3) := E'$.
   - Sample $h_4 \leftarrow \{0,1\}^{2\lambda}$ and program the oracles $\mathsf{XOF}_4(h_3, \{\boldsymbol{v}_k\}, \{\bar{\boldsymbol{v}}_k\}, \mathsf{msg}) := h_4$ and $\mathsf{XOF}'_4(h_4) := E$.
4. At each step, if the simulator must program the random oracle on a query that has already been made, it aborts. Since each programming step involves fresh randomness over $\{0,1\}^{2\lambda}$, the probability of a collision with any previous query is at most $Q/2^{2\lambda}$. We note that the above description omits the proof-of-work filtering for simplicity; handling it in the simulation is straightforward.
5. If no oracle programming conflict occurs, the simulator returns a valid SmallWood-ARK proof by assembling the simulated components into the expected format.

The resulting simulated proof is ξ-indistinguishable from a real SmallWood-ARK proof, with total distinguishing advantage bounded by:

$$\xi \leq \xi_{\mathsf{pcs}} + \frac{Q}{2^{2\lambda}} = Q \cdot \left(\frac{1}{2^\lambda} + \frac{1}{2^{2\lambda}}\right) \ .$$

□

5.4 SmallWood for Small Fields

It can be observed from the soundness analysis and the scheme parameters that, in its basic form, SmallWood-ARK is particularly well-suited for large fields, typically those of size significantly greater than 2^λ (e.g. fields of size $2^{2\lambda}$). In such settings, the soundness error ε_3 can be made negligible by opening just a single polynomial evaluation from the PCS, i.e., by setting the parameter $\ell' = 1$. This is desirable, as ℓ' has a substantial impact on the overall proof size. However, for smaller fields, achieving negligible ε_3 may require a larger value of ℓ', resulting in a heavier proof. To address this, we introduce below a variant of SmallWood-ARK specifically tailored for a "small" base field $\mathbb{F}$.

In this variant, the PACS statement remains unchanged from the original formulation, with the underlying witness matrix still defined over the base field $\mathbb{F}$ (see Section 5.1). However, the witness vector polynomial $\boldsymbol{P}$ and the masking vector polynomial $\boldsymbol{M}$ are now defined over an extension field $\mathbb{K}$ of $\mathbb{F}$. Let θ denote the extension degree of $\mathbb{K}$ over $\mathbb{F}$. The size of $\mathbb{K}$ is chosen such that only a single polynomial evaluation needs to be opened, i.e., $\ell' = 1$. Such a choice also implies that the dimension ρ of the vector $\boldsymbol{Q}$ can be set to 1 while ensuring a negligible soundness error $\varepsilon_2 = 1/|\mathbb{K}|^\rho$. In consequence, only a single masking polynomial is needed. We therefore simplify the notation and write Q and M instead of $\boldsymbol{Q}$ and $\boldsymbol{M}$. In the first step of this adapted PACS PIOP, the prover constructs random polynomials $P_1, \ldots, P_n \in \mathbb{K}[X]^{(\leq \ell'+s-1)}$ such that their evaluations $P_i(\omega_k) = w_{i,k}$ belong to the small field $\mathbb{F} \subseteq \mathbb{K}$ for all $i \in [1, n]$ and $k \in [1, s]$. The remainder of the PACS PIOP follows the original protocol (see Figure 6), with the base field $\mathbb{F}$ replaced by $\mathbb{K}$, and with the evaluation set $\mathbb{S}$ redefined as $\mathbb{K} \setminus \Omega$.

The SmallWood-PCS scheme is then adapted to leverage the fact that, although the committed polynomials $P_1, \ldots, P_n$ are defined over $\mathbb{K}[X]$, they admit s evaluations (on the witness support Ω) in the small field $\mathbb{F} \subseteq \mathbb{K}$. To exploit this, we commit these $\mathbb{K}[X]$ polynomials using SmallWood-LVCS over the small field $\mathbb{F}$. Specifically, we adapt the construction of the matrices $\boldsymbol{A}_j$ to commit $\mathbb{K}[X]$ polynomials while defining their entries over $\mathbb{F}$. Let ϕ be the $\mathbb{F}$-linear field embedding isomorphism from $\mathbb{F}^\theta$ to $\mathbb{K}$. Recall that the witness support is given by $\Omega = \{\omega_1, \ldots, \omega_s\} \subseteq \mathbb{K}$, and choose an additional distinct evaluation point $\omega_{s+1} \in \mathbb{K} \setminus \Omega$ for the randomness support. To commit a witness polynomial P_j, we adapt the matrix $\boldsymbol{A}_j$ from SmallWood-PCS (see Section 4.2) as a single column of dimension $s + \theta$, defined as $\boldsymbol{A}_j = (a_1, \ldots, a_{s+\theta})^T$ with

$$(a_1, \ldots, a_s) = (P_j(\omega_1), \ldots, P_j(\omega_s)) \quad \text{and} \quad (a_{s+1}, \ldots, a_{s+\theta}) = \phi^{-1}(P_j(\omega_{s+1})) \ .$$

For any evaluation point $e \in \mathbb{K}$, there exists a vector $\boldsymbol{v}_e \in \mathbb{K}^{s+\theta}$ such that $\boldsymbol{v}_e \cdot \boldsymbol{A}_j = P_j(e)$. Since each P_j is encoded as a single-column matrix, no extra randomness is required to mask the result of the linear combination $\boldsymbol{v}_e \cdot \boldsymbol{A}_j$, which directly yields the desired evaluation. In this version of SmallWood-PCS, the number of columns n_{cols} in the global LVCS matrix is a free parameter from which the remaining parameters are derived. Specifically, we obtain $\lceil n/n_{\mathsf{cols}} \rceil$ layers of $s+\theta$ rows to commit $P_1, \ldots, P_n$. The first layer consists of $(\boldsymbol{A}_1 \mid \cdots \mid \boldsymbol{A}_{n_{\mathsf{cols}}})$, the second of $(\boldsymbol{A}_{n_{\mathsf{cols}}+1} \mid \cdots \mid \boldsymbol{A}_{2n_{\mathsf{cols}}})$, and so forth.

Next, we commit the masking polynomial M. Unlike the witness polynomials, M consists purely of random coefficients (or evaluations) over the large field $\mathbb{K}$, and has $d_Q + 1$ coefficients (with $d_Q \gg s$). The polynomial M is committed as in the original SmallWood-PCS, except that each row over $\mathbb{K}$ is split into θ rows over $\mathbb{F}$ using ϕ^{-1}. Let $\boldsymbol{A}^{(\mathbb{K})}_{n+1}$ be the matrix that SmallWood-PCS would produce to commit M, with entries in $\mathbb{K}$. This matrix has $\mu + 1$ rows, where $\mu = \lceil d_Q/n_{\mathsf{cols}} \rceil$, and each row lies in $\mathbb{K}^{n_{\mathsf{cols}}}$. We define $\boldsymbol{A}_{n+1}$ as the matrix obtained by applying ϕ^{-1} to each row of $\boldsymbol{A}^{(\mathbb{K})}_{n+1}$, thereby yielding θ rows over $\mathbb{F}^{n_{\mathsf{cols}}}$. As a result, $\boldsymbol{A}_{n+1}$ has $\lceil 1 + d_Q/n_{\mathsf{cols}} \rceil \cdot \theta$ rows and n_{cols} columns. Evaluations of M are obtained via linear combinations as in the original scheme, with the application of ϕ embedded into the coefficients of the linear combinations.

The global matrix $\boldsymbol{A}$ is thus composed of the witness matrices $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_n$ organized into $\lceil n/n_{\mathsf{cols}} \rceil$ layers of $s + \theta$ rows, and the matrix $\boldsymbol{A}_{n+1}$ forming an additional layer of $\lceil 1 + d_Q/n_{\mathsf{cols}} \rceil \cdot \theta$ rows. The total number of rows is therefore

$$n_{\mathsf{rows}} = \left\lceil \frac{n}{n_{\mathsf{cols}}} \right\rceil \cdot (s + \theta) + \left\lceil \frac{d_Q}{n_{\mathsf{cols}}} + 1 \right\rceil \cdot \theta \ . \tag{16}$$

To perform a single linear combination over the rows of $\boldsymbol{A}$ with coefficients in $\mathbb{K}$, we require θ LVCS queries, one for each coordinate in the ϕ^{-1}-representation. Hence, the total number of LVCS queries equals the number of layers in $\boldsymbol{A}$ multiplied by θ, i.e., $m = \left\lceil \frac{n}{n_{\mathsf{cols}}} + 1 \right\rceil \cdot \theta$.

Table 2: Parameters of SmallWood-ARK in the small-field setting.

Parameters of the PACS Statement:		
$\mathbb{F}$	Base field	
$\mathbb{K}$	Extension field	
θ	Extension degree of $\mathbb{K}$ over $\mathbb{F}$	
s	Number of columns in the witness matrix (packing factor)	
n	Number of rows in the witness matrix	
d	Maximal degree of parallel polynomial constraints	
m_1	Number of parallel polynomial constraints	
m_2	Number of aggregated parallel polynomial constraints	
Parameters of the PACS PIOP:		
ℓ'	Number of oracle queries (polynomial evaluations)	$\ell' = 1$
ρ	Number of parallel repetitions	$\rho = 1$
d_Q	Degree of Q	(9)
Parameters of the PCS / LVCS:		
n_{cols}	Size of committed row vectors	*chosen*
n_{rows}	Number of committed row vectors	(16)
m	Number of LVCS queries	$m = \lceil 1 + n/n_{\mathsf{cols}} \rceil \cdot \theta$
ℓ	Number of DECS evaluation queries	*chosen*
Parameters of the DECS:		
n_{decs}	Number of committed polynomials	$n_{\mathsf{decs}} = n_{\mathsf{rows}}$
d_{decs}	Degree of committed polynomials	$d_{\mathsf{decs}} = n_{\mathsf{cols}} + \ell - 1$
N	Size of the evaluation domain (number of Merkle leaves)	*chosen*
η	Number of parallel repetitions of the degree-enforcing round	*chosen*
$\kappa_1, \kappa_2, \kappa_3, \kappa_4$	Grinding parameters	*chosen*

Table 2 summarizes the parameters of SmallWood-ARK in the small-field setting. We also provide the corresponding proof size formula below. For the naive proof size, we obtain:

$$|\pi_{\text{naive}}| = \underbrace{128}_{\mathsf{ctr}_1,\ldots,\mathsf{ctr}_4} + \underbrace{4\lambda}_{\mathsf{salt},\,\mathsf{root}} + \underbrace{\eta \cdot (d_{\mathsf{decs}} + 1) \cdot \log_2 |\mathbb{F}|}_{\boldsymbol{R}} + \underbrace{d_Q \cdot \theta \cdot \log_2 |\mathbb{F}|}_{Q} + \underbrace{(n + n_{\mathsf{cols}}) \cdot \theta \cdot \log_2 |\mathbb{F}|}_{\boldsymbol{v}_1^{(e)},\ldots,\boldsymbol{v}_m^{(e)}}$$
$$+ \underbrace{m \cdot \ell \cdot \log_2 |\mathbb{F}|}_{\bar{\boldsymbol{v}}_1,\ldots,\bar{\boldsymbol{v}}_m} + \underbrace{\ell \cdot n_{\mathsf{rows}} \cdot \log_2 |\mathbb{F}|}_{\boldsymbol{P}_{\mathsf{decs}}|_E} + \underbrace{\ell \cdot \eta \cdot \log_2 |\mathbb{F}|}_{\boldsymbol{M}_{\mathsf{decs}}|_E} + \underbrace{2\lambda \cdot \mathsf{nodes}(\ell, N)}_{\mathsf{auth}} + \underbrace{\ell \cdot \lambda}_{\mathsf{tapes}},$$

while for the optimized proof size, we get:

$$|\pi| = |\pi_{\text{naive}}| + 2\lambda - \Big(\underbrace{\eta \cdot \ell}_{(1)} + \underbrace{2\theta}_{(2),(3)} + \underbrace{m \cdot \ell}_{(4)} \Big) \cdot \log_2 |\mathbb{F}| \,.$$

6 Application

In what follows, we showcase SmallWood-ARK for two different applications. Section 6.1 focuses on zero-knowledge arguments for arithmetic circuits, while Section 6.2 addresses zero-knowledge arguments for lattice-based problems. In both cases, we compare the obtained proof sizes to other schemes from the state.

Let us further note that the SmallWood proof system has potential in a broad range of other applications, whenever one needs to prove statements with a "relatively small" witnesses in zero knowledge. This setting frequently arises in privacy-enhancing technologies. For instance, SmallWood-ARK has been used in the recent CAPSS framework [FR25a], a follow-up work building efficient SNARK-friendly post-quantum signature schemes, enabling practical post-quantum aggregated signatures and anonymous credentials.

6.1 Application to Arithmetic Circuits

Similarly to Ligero [AHIV17, AHIV23] and TCitH-MT [FR25b], our scheme provide sublinear arguments for arithmetic circuits. For this purpose, an arithmetic circuit statement $C(x) = y$ can be expressed as a PACS statement as follows: the witness w is made of all the pairs (a_j, b_j) with a_j and b_j the two operands of the j^{th} multiplication gate of C on input x. Then for each operand a_j or b_j, we rely on an aggregated constraint to check that it is correctly computed from the outputs of previous multiplication gates or from the public input x. Given some j, the left operand a_j is built as linear combination between some public scalars $\alpha_{j,k}$ and the outputs of previous multiplication gates (most of the scalars are usually zero):

$$a_j = \sum_{k<j} \alpha_{j,k} \cdot (a_k \cdot b_k) \ . \tag{17}$$

We have the same relation for the right operand b_j for some scalars $\beta_{j,k}$:

$$b_j = \sum_{k<j} \beta_{j,k} \cdot (a_k \cdot b_k) \ . \tag{18}$$

We can observe that those constraints perfectly match the format of the aggregated constraints, assuming that a_j and b_j are stacked in the same column of the witness matrix, for every gate j. Denoting $|C|$ the number of multiplication gates in C and assuming that $|C|$ is a multiple of s, the witness (of size $|w| = 2|C|$) is arranged so that we have no parallel constraints ($m_1 = 0$) and $m_2 = 2 \cdot |C| + |y|$ aggregated constraints, where $|y|$ aggregated constraints correspond to the checking that the circuit output is indeed the public value y.[6]

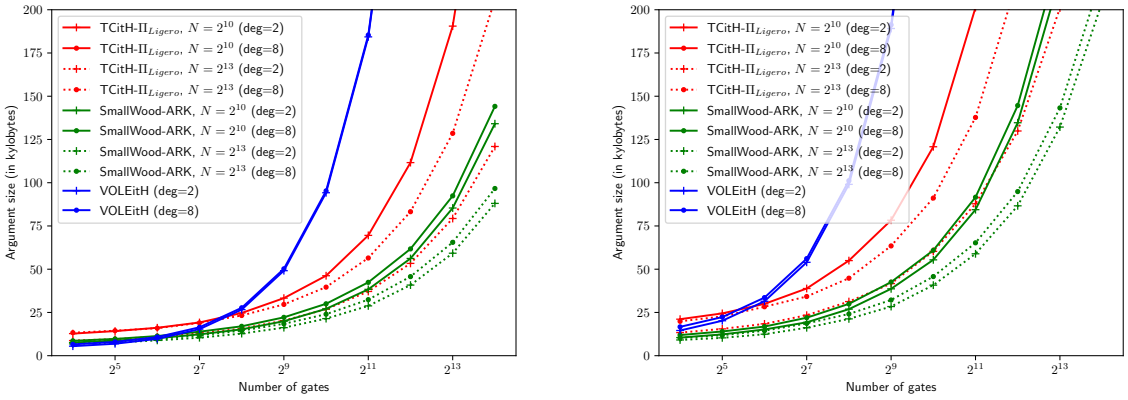
We provide hereafter the proof sizes of SmallWood-ARK applied to arithmetic circuits of sizes up to 2^{14} over 64-bit and 256-bit fields. For 256-bit fields we use the standard version of SmallWood-ARK, while for 64-bit fields, we use the small-field version depicted in Section 5.4. Since the number of Merkle leaves N is the parameter that most significantly affects the prover's running time (most of the computation being devoted to the evaluation and hashing of the leaves of the Merkle tree), we use two different values of N to illustrate the impact of this parameters: $N = 1024$ and $N = 8192$. For the other parameters of SmallWood-ARK, we fix μ to $d_j + 1 - \ell' = s$, for $d_j = \ell' + s - 1$ the degree of the PIOP witness polynomials. This ensures that each witness polynomial fits into exactly one column of the LVCS (i.e., $\nu_j = 1$ for witness polynomials). This choice eliminates the need for additional randomness while committing the witness polynomials in the LVCS. We note that the masking polynomials of the PIOP are still committed through several columns (since they have larger degrees) and still require additional randomness. The parameters β and s are then chosen such that the proof size is minimized.

We compare these results to TCitH-MT-Π_{Ligero}, the Merkle tree variant of the Threshold-Computation-in-the-Head framework applied to the Ligero protocol [FR25b]. This argument system is structurally close to SmallWood-ARK as it also makes use of the DECS, but it makes direct use of it as polynomial commitment scheme for the underlying PIOP. We further compare SmallWood-ARK to recent zero-knowledge arguments based on the MPC-in-the-Head paradigm and GGM trees. The most prominent examples include the VOLE-in-the-Head framework [BBD$^+$23] and its variant TCitH-GGM [FR25b]. While these schemes achieve highly compact proofs for small circuits – such as those used in post-quantum signature designs (e.g., [FR25b, BFG$^+$24, OTX24, HJ25]) – their linear proof size quickly renders them non-competitive compared to SmallWood-ARK as the witness size increases. We compare our scheme with the VOLE-in-the-Head framework which provides proofs that are usually smaller than TCitH-GGM. Since VOLE-in-the-Head is structurally different from our scheme and TCitH-MT, selecting parameters that allow for a fair comparison

[6] In this description, we omit the handling of the input x for the sake of simplicity. It can be handled by adding a "virtual" pair (a_j, b_j) such that $a_j \cdot b_j = x_j$ for every secret coordinate of the input x (while public part of the input can be directly hardcoded into the definition of the aggregated constraints). Alternatively, the secret input can be embedded in the first row (or first few rows) of the witness matrix which then enter as a linear term in (17) and (18).

with these schemes is challenging. In the following, we consider VOLE-in-the-Head instantiated with 11 GGM trees of 2^{11} leaves each, as these parameters are commonly used in the literature for this framework.

Figures 9a and 9b provide the proof sizes of the three schemes for circuits of size $\leq 2^{14}$ (in terms of multiplication gates) for 64-bit fields and 256-bit fields respectively. To illustrate the ability of SmallWood-ARK to handle higher-degree gates, we also provide results for circuits made of degree-8 gates (instead of degree-2 multiplication gates). Table 3 further provides some concrete parameters and sizes for circuits of size $|C| = 2^{12}$ (in terms of non-linear gates).



(a) Proof sizes for 64-bit fields

(b) Proof sizes for 256-bit fields

Fig. 9: Comparison of VOLE-in-the-Head, TCitH-Π_{Ligero} and SmallWood-ARK for arithmetic circuits (with input $x \in \mathbb{F}^{16}$).

SmallWood-ARK vs. TCitH-MT. While SmallWood-ARK is only marginally better than TCitH-MT over small fields (e.g., 64 bits or less), the improvement becomes significantly more pronounced over larger fields. For instance, for circuits of size 2^{12} over a 256-bit field and using $N = 1024$, SmallWood-ARK achieves a 62% reduction in proof size. The more modest gain over small fields can be explained by the reduced advantage of using a full-domain PCS compared to a direct use of the DECS as a small-domain PCS within the PIOP (as done in TCitH-MT). Indeed, over small fields, $N = |\mathbb{E}|$ is closer to – or may even be equal to – the size of the field $|\mathbb{F}|$, and using a full-domain PCS induces additional communication costs. We also observe that increasing N tends to reduce the gap between TCitH-MT and SmallWood-ARK, which again follows from the fact that N approaches $|\mathbb{F}|$. Nevertheless, even with $N = 8192$, the gain remains substantial, reaching 33% over 256-bit fields. Since N directly impacts the prover's running time, this shows that SmallWood-ARK can achieve highly competitive proof sizes without requiring heavy prover computation.

Beyond its improved performance over large fields, SmallWood-ARK also exhibits a key advantage over TCitH-MT when the degree d of the PACS constraints exceeds 2. While this situation does not arise for standard arithmetic circuits, it is relevant in many other settings where higher-degree constraints enable to optimize the proof size (e.g., arithmetic-oriented hash functions in the CAPSS framework [FR25a]). In TCitH-MT, the degree d of the polynomial constraints has a significant impact on the proof size, whereas its effect on SmallWood-ARK is minimal. This allows SmallWood-ARK to handle higher-degree constraints without a substantial increase in proof size. To illustrate this point, we consider arithmetic circuits with non-linear gates of degree 8 (with fan-in 2) instead of standard degree-2 multiplication gates. In this setting, SmallWood-ARK achieves significantly smaller proofs than TCitH-MT. For example, with $|C| = 2^{12}$ and $|\mathbb{F}| = 2^{64}$, the proof size of SmallWood-ARK with $N = 1024$ increases from 56 KB to 62 KB (a 10% increase), whereas the proof size of TCitH-MT grows from 112 KB to 358 KB (a factor of 3). Over a 256-bit field, the

Table 3: Parameters and proof sizes for VOLE-in-the-Head [BBD+23], TCitH-Π_{Ligero} [FR25b] and SmallWood-ARK for arithmetic circuits of size $|C| = 2^{12}$ (with $|x| = 100$).

Scheme	$\|\mathbb{F}\|$	Degree	N	ℓ	s	η	ℓ'	ρ	Size
VOLEitH	2^{64}	2	11×2^{11}	1	1	–	–	–	364.5 KB
TCitH-Π_{Ligero}			1024	69	86	12	–	2	111.6 KB
			8192	24	84	15	–	2	53.4 KB
SmallWood-ARK			1024	40	46	11	2	1	56.2 KB
			8192	19	41	13	2	1	40.8 KB
VOLEitH	2^{256}	2	11×2^{11}	1	1	–	–	–	1 450.7 KB
TCitH-Π_{Ligero}			1024	69	86	3	–	1	358.1 KB
			8192	27	133	5	–	1	130.0 KB
SmallWood-ARK			1024	56	48	4	1	1	134.7 KB
			8192	22	64	5	1	1	86.6 KB
VOLEitH	2^{64}	8	11×2^{11}	1	1	–	–	–	365.6 KB
TCitH-Π_{Ligero}			1024	–	–	–	–	–	–
			8192	42	85	17	–	2	83.2 KB
SmallWood-ARK			1024	41	30	11	2	1	61.7 KB
			8192	19	31	13	2	1	45.7 KB
VOLEitH	2^{256}	8	11×2^{11}	1	1	–	–	–	1452.8 KB
TCitH-Π_{Ligero}			1024	–	–	–	–	–	–
			8192	55	81	6	–	1	217.8 KB
SmallWood-ARK			1024	56	24	4	1	1	144.6 KB
			8192	22	32	5	1	1	94.8 KB

contrast is even sharper: no choice of parameters for TCitH-MT with $N = 1024$ suffices to meet the desired security level (requiring larger N), while the proof size of SmallWood-ARK remains nearly unchanged, with only a 7% increase.

SmallWood-ARK vs. VOLE-in-the-Head. As expected, VOLE-in-the-Head is competitive for very small circuits but quickly becomes non-competitive as the circuit size increases. More precisely, SmallWood-ARK outperforms VOLE-in-the-Head for circuits of size larger than 2^7 over 64-bit fields and larger than 2^4 over 256-bit fields. We note that the circuit arithmetization considered here is particularly favorable to VOLE-in-the-Head, since each multiplication gate contributes a single element to the VOLE-in-the-Head witness, compared to two elements per gate in SmallWood-ARK.[7] In scenarios involving non-linear gates with fan-in 1 (or other scenarios well suited for PACS arithmetization), the crossover point would likely occur for even smaller circuits. Similarly to SmallWood-ARK, we observe that the impact of the constraint degree remains limited in the VOLE-in-the-Head framework.

SmallWood-ARK vs. hash-based zk-SNARKs. We finally compare SmallWood-ARK with hash-based zk-SNARKs, i.e., hash-based zero-knowledge proof systems targeting asymptotic succinctness. We focus on a comparison with Ligero [AHIV17, AHIV23] and Aurora [BCR+19]. Figure 10 reports the corresponding proof sizes for arithmetic circuits over a 256-bit field. To ensure a fair comparison between Ligero, SmallWood-ARK, and TCitH – which share similar Merkle-tree-based commitments – we use the same number N of Merkle-tree leaves for all three schemes, following Ligero's formula for selecting N as a function of the circuit size $|C|$. We observe that SmallWood-ARK clearly outperforms Ligero.

To obtain proof sizes for Aurora, we rely on the open-source implementation provided in `libiop`,[8] configured to achieve provable security. Aurora has asymptotic proof complexity $O(\log |C|)$, compared to $O(\sqrt{|C|})$

[7] Since VOLE-in-the-Head does not rely on packing, its witness matrix consists of a single column containing the outputs of the non-linear gates, without redundancy.

[8] `https://github.com/scipr-lab/libiop`

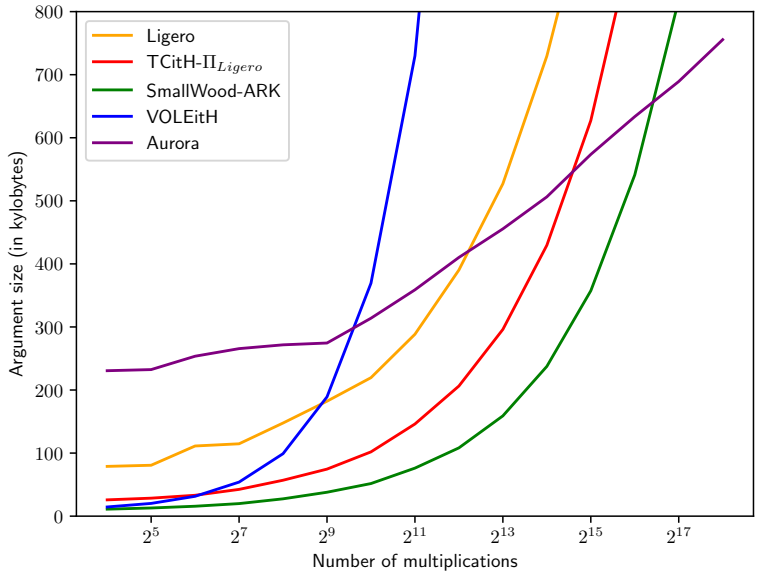


Fig. 10: Comparison of Ligero, TCitH, Aurora and SmallWood-ARK for zero-knowledge proof for arithmetic circuits over a 256-bit field.

for Ligero. As a result, Aurora is asymptotically better than Ligero and SmallWood-ARK, but SmallWood-ARK achieves smaller proof sizes when the circuit is sufficiently small. In the setting of Figure 10, the crossover point occurs for circuits of size between 2^{16} and 2^{17} (in number of multiplication gates). We also note that Aurora is not field-agnostic: it requires a field whose additive or multiplicative group has size equal to a large power of two, whereas SmallWood-ARK is compatible with arbitrary finite fields.

We discuss other hash-based (zk-)SNARKs that could be compared with SmallWood-ARK, including STARK [BBHR18b], Ligero++ [BFH+20], and Brakedown [GLS+23]. The STARK proof system based on the FRI protocol [BBHR18a] was introduced in [BBHR18b]. According to the Aurora paper, Aurora achieves smaller proof sizes than STARK for arithmetic circuit statements, which is consistent with the fact that the STARK language does not natively support arithmetic circuits. Ligero++ [BFH+20] combines ideas from Ligero and Aurora to improve proof sizes; however, the lack of detailed parameter choices and the absence of a public implementation make a fair comparison difficult. In any case, the reported proof sizes of Ligero++ in our regime are not competitive with SmallWood-ARK: the paper reports sizes exceeding 200 KB for relatively small circuits such as LowMC or AES. More recently, Brakedown [GLS+23] proposes linear-time field-agnostic SNARKs for R1CS, from which we drew inspiration for the design of SmallWood-PCS. However, Brakedown is not zero-knowledge by design, and adapting it to a zero-knowledge setting would require a non-trivial analysis to estimate the resulting proof sizes. Moreover, our analysis in Section 4.3 shows that SmallWood-PCS outperforms Brakedown's polynomial commitment scheme for polynomials of relatively small degrees, suggesting that SmallWood-ARK would also outperform a zero-knowledge variant of Brakedown in this regime.

Asymptotic behavior of SmallWood-ARK. Hash-based SNARKs relying on a Ligero-like proximity test have proof sizes that scale as $\Theta(|C|^{1/2})$. For argument systems based on the DECS, like TCitH-MT and SmallWood-ARK, the proof size scales as $\Theta(|C|^{2/3})$. This behavior stems from the cost of revealing the degree-enforcing polynomial $\boldsymbol{R}$. Indeed, if the witness matrix is arranged with $n = \Theta(|C|^{1-\alpha})$ rows and $s = \Theta(|C|^{\alpha})$ columns, the degree-enforcing polynomial $\boldsymbol{R}$ has degree $d = \Theta(|C|^{\alpha})$ and dimension $\eta = \Theta(|C|^{\alpha})$ to achieve negligible

soundness error. On the other hand, the opened DECS evaluations are of size $\Theta(n) = \Theta(|C|^{1-\alpha})$. The optimal choice of α that minimizes the proof size is $\alpha = 1/3$ for which the proof size scales as $\Theta(|C|^{2/3})$. In this setting, the prover must evaluate $\Theta(|C|^{2/3})$ polynomials of degree $\Theta(|C|^{1/3})$ at $\Theta(|C|^{1/3})$ evaluation points. Assuming that no evaluation speed-ups (e.g., NTT-based methods) can be applied, the cost of this step – which constitutes the computational bottleneck – scales as $\Theta(|C|^{4/3})$. The verification algorithm essentially consists of checking a constant number of Merkle tree leaves, resulting in a cost of $\Theta(|C|^{2/3})$. We stress that SmallWood-ARK does not aim at asymptotic optimality, but rather at achieving practical proof sizes for circuits of moderate size, while remaining field-agnostic and compatible with high-degree constraints.

6.2 Application to Lattices

We now present an application of the SmallWood-ARK proof system to obtain short and exact zero-knowledge arguments for lattice problems. The SmallWood-ARK proof system outperforms the prior schemes when proving knowledge of "relatively small" witnesses with polynomial constraints of degree possibly greater than two, and proving lattice-based statements perfectly fits this context.

There exist two approaches to achieve short and efficient proofs for lattice instances: specialized lattice-based zero-knowledge techniques [LNP22, BS23] and MPC-in-the-Head/hash-based proof systems [BN20, FMRV22, FR25b]. The first approach relies on the algebraic structure of the lattice instances to provide exact lattice-based proofs for these statements. The initial work in this area is presented in [BLS19], which proposes a zero-knowledge proof of knowledge for the Short Integer Solution (SIS) problem, resulting in proofs of several hundred kilobytes. Several subsequent works aimed to improve the original scheme [LNS21, LNP22]. Notably, [LNP22] achieves proofs of less than twenty kilobytes and eliminates the need for an NTT-friendly field. The second approach (MPC-in-the-Head / hash-based proof systems) relies solely on symmetric cryptography, without leveraging the algebraic structure of lattice problems. As a result, early applications produced large proofs, though this approach offers flexibility and is less sensitive to the form of the statement. Combinatorial techniques for lattice proofs began with [LNSW13], yielding multi-megabyte proofs. Subsequent works based on MPC-in-the-Head and Cut-and-Choose [BN20, Beu20, FMRV22] reduced this to hundreds of kilobytes. The more recent TCitH framework [FR25b] succeeds to achieve proof sizes under 20 KB for some lattice statements, comparable to specialized methods. As we show hereafter, using SmallWood-ARK allows us to outperform the prior MPC-in-the-Head/hash-based proof systems to prove lattice statements.

In what follows, we show how to apply SmallWood-ARK to check an instance of a Learning With Error (LWE) problem. Such a scheme proves the knowledge of two vectors $s \in \mathbb{F}_q^n$ and $e \in \mathbb{F}_q^m$ such that $\|s\|_\infty, \|e\|_\infty \leq \beta$ and $t = A \cdot s + e$, where $A \in \mathbb{F}_q^{m \times n}$ and $t \in \mathbb{F}_q^m$ are publicly known. To prove that the vector $v := (s \parallel e) \in \mathbb{F}_q^{n+m}$ is of infinity norm at most β, the prior hash-based schemes rely on a binary decomposition. They show that v can be written as

$$v = \sum_{i=0}^{k-2} 2^i v^{(i)} + (2\beta - 2^{k-1} + 1)v^{(k-1)} - \boldsymbol{\beta} \ , \tag{19}$$

where $\boldsymbol{\beta} = (\beta, \ldots, \beta) \in \mathbb{F}_q^{n+m}$, $k = \lceil \log_2(2\beta + 1) \rceil$ and $v^{(0)}, \ldots, v^{(k-1)} \in \{0,1\}^{n+m}$. By proving that Equation (19) holds and that the $v^{(i)}$ vectors belong to $\{0,1\}^{n+m}$, we can deduce that v indeed satisfies the norm constraint. This can be readily verified by observing that $v = -\boldsymbol{\beta}$ when each $v^{(i)}$ is set to the all-zero vector, and $v = \boldsymbol{\beta}$ when each $v^{(i)}$ is set to the all-one vector. Moreover, to prove that a vector is binary, one can show that each coordinate is a root of the polynomial $X \cdot (X - 1)$, which is a degree-2. The main drawback with this method is that the underlying witness which contains the binary decomposition is large, leading to large proof sizes. Since the communication cost of the proof systems scales with the witness size, the goal would be to minimize the size of the witness. The above procedure only relies on degree-2 polynomial constraints. However, SmallWood-ARK can handle efficiently larger degrees (as illustrated in Section 6.1). Therefore, to prove that $\|v\| \leq \beta$, one can show that each coordinate is a root of the polynomial

$$(X + \beta) \cdot (X + \beta - 1) \cdot \ldots \cdot (X - \beta + 1) \cdot (X - \beta) \ ,$$

which is a degree-$(2\beta+1)$ constraint. If β is too large, one can still rely on decomposition as above, but using a larger basis. Unfortunately, constraints of large degree cannot be effectively exploited when β is small. For instance, if one needs to prove that v is a binary vector, it is not possible to directly use a high-degree constraint to reduce the witness size. To address this limitation, we introduce hereafter a compression technique for lattice witnesses specially suited for constraints systems such as PACS.

Compression Technique for Lattice Witnesses. Assume we wish to prove that p values $v_1, \ldots, v_p$ belong to the set $\{0, \ldots, \alpha-1\}$. Let $\mathsf{Decompress} : S \subseteq \mathbb{F} \to \{0, \ldots, \alpha-1\}^p \subseteq \mathbb{F}^p$ be a fixed bijection, where S is a set of size $|S| = \alpha^p$. To prove that $(v_1, \ldots, v_p) \in \{0, \ldots, \alpha-1\}^p$, it suffices to show the existence of $\tilde{v} \in S$ such that $\mathsf{Decompress}(\tilde{v}) = (v_1, \ldots, v_p)$. This claim reduces to satisfying two types of constraints on $\tilde{v}$:

- The membership constraint $\tilde{v} \in S$, which can be enforced by checking that $\tilde{v}$ is a root of the degree-α^p polynomial $(X - s_1) \cdots (X - s_{\alpha^p})$, where $S = \{s_1, \ldots, s_{\alpha^p}\}$.
- The decompression constraint $\mathsf{Decompress}(\tilde{v}) = (v_1, \ldots, v_p)$, which can be realized by expressing $\mathsf{Decompress}$ as a vector of univariate polynomials, each of degree at most $\alpha^p - 1$.

For example, let $p = 2$, $\alpha = 2$, and $S = \{0, 1, 2, 3\}$ with $\mathsf{Decompress}$ defined as:

$$\begin{array}{ll} \mathsf{Decompress}(0) = (0,0) & \\ \mathsf{Decompress}(1) = (0,1) & \text{and} \end{array} \qquad \begin{array}{l} \mathsf{Decompress}(2) = (1,0) \\ \mathsf{Decompress}(3) = (1,1) \end{array}$$

In this case, $\mathsf{Decompress}$ can be written as:

$$\mathsf{Decompress}(X) = \left(\underbrace{X(X-1)\left(\frac{X-3}{-2} + \frac{X-2}{6}\right)}_{P_1}, \underbrace{X(X-2)\left(\frac{X-3}{2} + \frac{X-1}{6}\right)}_{P_2} \right) .$$

Then the membership constraint consists in verifying that $\tilde{v}$ satisfies $\tilde{v} \cdot (\tilde{v}-1) \cdot (\tilde{v}-2) \cdot (\tilde{v}-3) = 0$, while the decompression constraint consists in verifying that $v_1 = P_1(\tilde{v})$ and $v_2 = P_2(\tilde{v})$. This compression technique allows p values to be packed into a single element, with a deterministic method to unpack them via polynomial relations of degree at most α^p.

In practice, the values $v_1, \ldots, v_p$ are not part of the explicit witness; instead, they are derived on the fly using $\mathsf{Decompress}$ to verify other constraints (e.g., linear constraints in the LWE setting). While a natural choice for S may be the set $\{0, \ldots, \alpha^p - 1\}$, we emphasize that S can be chosen arbitrarily, allowing $\mathsf{Decompress}$ to be instantiated with lower-degree polynomials. Although this does not reduce the degree of the membership constraint (which remains $\deg(S)$), a good choice of S might reduce computational overhead.

Application of SmallWood and Comparison. We apply SmallWood-ARK on toy benchmark instances with $q \approx 2^{32}$ and $q \approx 2^{61}$ from the lattice zero-knowledge literature as well as on Kyber [ABD+21] and Dilithium [BDK+21] instances. We additionally apply SmallWood-ARK on an instance of the subset-sum problem, which can be seen extreme form of the LWE/ISIS problem where one has only one linear equation ($m = 1$). The results are depicted in Table 4. For Kyber instances, we obtain proof sizes ranging from 14 to 16 KB across all three security levels. For Dilithium instances, the proof sizes range from 17 to 23 KB. To the best of our knowledge, these are the smallest proof sizes reported in the literature for such statements. Additionally, we achieve a short proof size of 12.5 KB for a 128-bit security subset-sum instance of dimension $n = 256$ and modulus $q \approx 2^{256}$.

We compare our proof sizes with those of TCitH-MT [FR25b] and VOLE-in-the-Head [BBD+23]. The sizes achieved by the VOLE-in-the-Head framework are computed assuming that it uses 11 seed trees of size 2^{11}, with the optimizations of [BBM+24]. We give more details on how the VOLEitH-based proof sizes are derived below in a dedicated paragraph. We can observe that SmallWood-ARK clearly outperforms TCitH-MT, reducing sizes from 21–58 KB to 14–23 KB. SmallWood-ARK also outperforms the VOLEitH-based

Table 4: Proof sizes when applied on some SIS/LWE instances. "$\beta = 1/2$" means that the secret is a binary vector.

LWE Instance	LWE Parameters				SmallWood Parameters									SmallWood-ARK	Comparison	
	q	n	m	β	N	ℓ	p	s	η	ℓ'	ρ	κ_{decs}	κ_{piop}		TCitH-MT	VOLEitH
Toy Example 1	$\approx 2^{32}$	2048	512	1	4096	18	2	11	16	4	-	8	7	**16 977 B**	38 081 B	39 072 B*
Toy Example 2	$\approx 2^{61}$	4096	1024	1/2	4096	19	3	7	10	2	-	9	12	**21 356 B**	57 409 B	51 470 B*
Kyber512	3329	2×256	2×256	3	2048	20	1	10	36	11	-	7	7	14 115 B	21 185 B	**11 754 B**
Kyber768	3329	3×256	3×256	2	2048	21	1	14	40	11	-	7	7	**15 004 B**	24 938 B	15 519 B
Kyber1024	3329	4×256	4×256	2	2048	21	1	18	41	11	-	8	7	**16 455 B**	28 241 B	19 637 B
Dilithium2	8380417	4×256	4×256	2	4096	18	1	16	23	6	-	8	0	**17 514 B**	40 100 B	36 241 B
Dilithium3	8380417	5×256	6×256	4	4096	18	1	18	23	6	-	8	0	**22 076 B**	57 526 B	45 727 B
Dilithium5	8380417	7×256	8×256	2	4096	19	1	25	25	6	-	8	0	**22 700 B**	58 088 B	60 527 B
Subset-Sum	$\approx 2^{256}$	256	1	1/2	4096	18	3	2	2	1	1	5	0	**12 557 B**	-	13 484 B*

* A straightforward application of VOLEitH to the instances "Toy Example 1", "Toy Example 2" and "Subset-Sum" would lead to proof sizes of respectively 51 392 B, 102 308 B and 96 580 B. The sizes given the table rely on the sharing-over-integers technique from [FMRV22], respectively with parameters $(A, p) = (2^{23}, 2)$, $(A, p) = (2^{24}, 4)$, and $(A, p) = (2^{18}, 1)$, where p is the packing value as defined above in the compression technique and A is the range size in which the additive shares are sampled using [FMRV22].

scheme, except for Kyber instances and the subset-sum instance. For Kyber512, the VOLEitH-based scheme remains competitive with SmallWood-ARK due to the relatively small LWE parameters. As a result, although the proof size of the VOLEitH-based scheme grows linearly with the witness size, it does not reach the regime where the proof size becomes prohibitively large. For the subset-sum instance, we combine VOLEitH with the sharing-over-integers technique from [FMRV22], resulting in competitive proof sizes. This improvement stems from the fact that the method of [FMRV22] is particularly well suited to settings involving small secret values over large fields. Without this technique, the VOLEitH proof size would be approximately 100 KB.

We note that the recent work of [BCGZ24] proposes zero-knowledge arguments for lattice-based statements built on the original version of the TCitH framework [FR23]. The authors report proof sizes of 83 KB (with 152 KB of pre-processing) for Kyber512. However, as detailed in Appendix B, we identify several security flaws in their scheme, affecting both its soundness and zero-knowledge property. For this reason, we exclude this work from our comparison.

We further compare our results to previous works on the toy benchmark instances in Table 5. As explained at the beginning of the application section, there are mainly two approaches to get efficient proofs for lattice, specialized lattice-based zero-knowledge techniques and hash-based proof systems. As demonstrated above, SmallWood-ARK outperforms prior MPC-in-the-Head/hash-based schemes. One can observe that SmallWood-ARK also achieves competitive sizes compared to those obtained by lattice-based techniques. For example, while the best lattice-based scheme [LNP22] obtains sizes around 15 KB on the first LWE instance, SmallWood-ARK has sizes around 17 KB. Let us stress that [LNP22] prove a L_2 norm, where all the other works prove a L_∞. In a context where one strictly focuses on L_∞, SmallWood-ARK seems to be the proof system leading to the smallest proofs.

Applying VOLEitH to lattice-based statement. We explain hereafter how we apply the VOLE-in-the-Head framework to an LWE statement. This framework can prove that a witness satisfies a public set of polynomial constraints. In the context of LWE statement, the witness is just the secret vector s, possibly compressed with the above technique (using Decompress). We use the optimizations from [BBM+24]. The VOLEitH parameters are the following: the constraint degree d, the size $|w|$ of the witness $w \in \mathbb{F}_q^{|w|}$, the number τ of seed trees, the number N of leaves for each seed tree, an upper bound T_{open} of the number of revealed seeds, a bit size κ for a proof-of-work nonce, the number ρ of checked relations in the consistency check, the degree

Table 5: Comparison of existing exact ZKP for ISIS instances. ($\star$): All the schemes prove the infinity norm, except [LNP22] that proves the L_2-norm.

Scheme	Year	Technique	Any q	Toy Example 1		Toy Example 2	
				Proof Size	Rej. Rate	Proof Size	Rej. Rate
[LNSW13]	2013	Comb.	✓	3600 KB	0	8988 KB	0
Ligero [AHIV17]	2017	SNARK	q prime + NTT	157 KB‡	0	200 KB‡	0
Aurora [BCR^{+}19]	2019	SNARK	q prime + NTT	71 KB‡	0	71 KB‡	0
[BLS19]	2019	Alg.	q prime + NTT	384 KB	0.92	-	-
[BN20]	2020	Comb.	q prime	-	-	4077 KB	0
[Beu20]	2020	Comb.	q prime	233 KB	0	444 KB	0
[ENS20]	2020	Alg.	q prime + NTT	47 KB	0.95	-	-
[LNS21]	2021	Alg.	q prime + NTT	33.3 KB	0.85	-	-
[FMRV22] (batching)	2022	Comb.	✓	291 KB	0.04	291 KB	0.04
[FMRV22] (C&C)	2022	Comb.	✓	184 KB	0.05	184 KB	0.05
[LNP22]*	2022	Alg.	q prime	14.7 KB	0.86	-	-
[FR25b]	2023	Comb.	q prime	38 KB	0	57 KB	0
VOLEitH [BBD^{+}23]†	2025	Comb.	q prime	51 KB	0	102 KB	0
VOLEitH [BBD^{+}23] + [FMRV22]†	2025	Comb.	q prime	39 KB	0.01	51 KB	0.01
SmallWood-ARK	2025	Comb.	q prime	17 KB	0	21 KB	0

* All the schemes prove the infinity norm, except [LNP22] that proves the L_2-norm.
† These results follow from the application of VOLEitH [BBD^{+}23] (with sharing over the integers [FMRV22]) to LWE instances as presented below.
‡ These results come from [Beu20].

ρ' of the used extension field. In that setting, the bit size of the produced proof is

$$|\pi_{\mathsf{voleith}}| = (3\lambda+\kappa) + \underbrace{T_{\text{open}}\cdot\lambda + \tau\cdot 2\lambda}_{\text{AVC}} + \underbrace{\tau\cdot|w|\cdot\log_2 q}_{\text{Witness}} + \underbrace{(d-1)\rho'\cdot\log_2 q}_{\text{Broadcast}} + \underbrace{(\tau-1)\cdot(d-1)\cdot\rho'\cdot\log_2 q}_{\text{Masking}} + \underbrace{(\tau-1)\cdot\rho\cdot\log_2 q + \rho\cdot\log_2 q}_{\text{Consistency test}} . \quad (20)$$

In practive, we select the VOLEitH parameters as follows. We take $\tau = 11$ and $N = 2048$, which are typical parameters for VOLEitH-based schemes. Then, we take $T_{\text{open}} = 115$, $\kappa = 32$, ρ such that $\rho\cdot\log_2 q \leq (\lambda+16)$, and $\rho' := \tau$.

When applying the VOLE-in-the-Head framework to LWE statements, one can observe that the obtained proof system builds size-N additive sharings of small values $s_1,\ldots,s_n \in \{0,\ldots,B-1\}$ modulo q. It means that the auxiliary values cost $\log_2 q$ bits per coefficients, even though the shared value is very small ($B \ll q$). We can tweak the proof system to rely on additive sharings over integers, instead of additive sharings modulo q, as proposed by [FMRV22]. Each additive share will be sampled in $\{0,\ldots,A-1\}$ for some parameter $A < q$ and the auxiliary values will be computed over integers. However, it is well-known that an additive sharing over integers leaks information about the shared values. To handle this issue, [FMRV22] proposes to rely on rejection sampling: the prover will restarts the proof procedure if responding to the verifier challenge would leak secret information. The moment where the prover should abort the protocol is properly defined in [FMRV22]. We just here recall the rejection rate: the probability to abort is

$$p_{\text{rej}} = 1 - \left(1 - \frac{B-1}{A}\right)^{|w|\cdot\tau} \approx |w|\cdot\tau\cdot\frac{B-1}{A} .$$

Therefore, the choice of A is a trade-off between communication cost and rejection rate. A small value for A would lead to shorter proofs, but with a high rejection rate. In our work, we use the rejection strat-

egy of [FMRV22] such that the rejection rate is around 0.01. In that case, the proof size is also given by Equation (20), except that the term $\tau \cdot |w| \cdot \log_2 q$ is replaced by $\tau \cdot |w| \cdot \log_2 A$.

Acknowledgements

This research was supported by the Ethereum Foundation under the 2024 Academic Grants Round.

References

ABD+21. Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crypstals-kyber – algorithm specifications and supporting documentation. Version 3.02 – August 4, 2021, 2021. `https://pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf`.

ACFY24. Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. STIR: Reed-solomon proximity testing with fewer queries. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 380–413. Springer, Cham, August 2024.

AGH+23. Carlos Aguilar-Melchor, Nicolas Gama, James Howe, Andreas Hülsing, David Joseph, and Dongze Yue. The return of the SDitH. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 564–596. Springer, Cham, April 2023.

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

AHIV23. Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: lightweight sublinear arguments without a trusted setup. *DCC*, 91(11):3379–3424, 2023.

BBB+25. Carsten Baum, Ward Beullens, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Christian Majenz, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger, Lawrence Roy, and Peter Scholl. FAEST: Algorithm Specifications – Version 2.0, 2025. `https://faest.info/faest-spec-v2.0.pdf`.

BBD+23. Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Emmanuela Orsini, Lawrence Roy, and Peter Scholl. Publicly verifiable zero-knowledge and post-quantum signatures from VOLE-in-the-head. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 581–615. Springer, Cham, August 2023.

BBFR25. Ryad Benadjila, Charles Bouillaguet, Thibauld Feneuil, and Matthieu Rivain. MQOM: MQ on my Mind – Algorithm Specifications and Supporting Documentation. Version 2.0, 2025. `https://mqom.org/docs/mqom-v2.0.pdf`.

BBHR18a. Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast reed-solomon interactive oracle proofs of proximity. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *ICALP 2018*, volume 107 of *LIPIcs*, pages 14:1–14:17. Schloss Dagstuhl, July 2018.

BBHR18b. Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046, 2018.

BBHR19. Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable zero knowledge with no trusted setup. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 701–732. Springer, Cham, August 2019.

BBM+24. Carsten Baum, Ward Beullens, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger, Lawrence Roy, and Peter Scholl. One tree to rule them all: Optimizing GGM trees and OWFs for post-quantum signatures. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part I*, volume 15484 of *LNCS*, pages 463–493. Springer, Singapore, December 2024.

BCG20. Jonathan Bootle, Alessandro Chiesa, and Jens Groth. Linear-time arguments with sublinear verification from tensor codes. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 19–46. Springer, Cham, November 2020.

BCGZ24. Weihao Bai, Long Chen, Qianwen Gao, and Zhenfeng Zhang. MPC-in-the-head framework without repetition and its applications to the lattice-based cryptography. In *2024 IEEE Symposium on Security and Privacy*, pages 578–596. IEEE Computer Society Press, May 2024.

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

BDJ+23. Lennart Braun, Cyprien Delpech de Saint Guilhem, Robin Jadoul, Emmanuela Orsini, Nigel P. Smart, and Titouan Tanguy. ZK-for-Z2K: MPC-in-the-head zero-knowledge proofs for $\mathbb{Z}_{2^k}$. In Elizabeth A. Quaglia, editor, *19th IMA International Conference on Cryptography and Coding*, volume 14421 of *LNCS*, pages 137–157. Springer, Cham, December 2023.

BDK+21. Shi Bai, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crypstals-dilithium – algorithm specifications and supporting documentation. Version 3.1 – February 8, 2021, 2021. `https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf`.

Beu20. Ward Beullens. Sigma protocols for MQ, PKP and SIS, and Fishy signature schemes. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 183–211. Springer, Cham, May 2020.

BFG+24. Loïc Bidoux, Thibauld Feneuil, Philippe Gaborit, Romaric Neveu, and Matthieu Rivain. Dual support decomposition in the head: Shorter signatures from rank SD and MinRank. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part II*, volume 15485 of *LNCS*, pages 38–69. Springer, Singapore, December 2024.

BFH+20. Rishabh Bhadauria, Zhiyong Fang, Carmit Hazay, Muthuramakrishnan Venkitasubramaniam, Tiancheng Xie, and Yupeng Zhang. Ligero++: A new optimized sublinear IOP. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 2025–2038. ACM Press, November 2020.

BGKS20. Eli Ben-Sasson, Lior Goldberg, Swastik Kopparty, and Shubhangi Saraf. DEEP-FRI: Sampling outside the box improves soundness. In Thomas Vidick, editor, *ITCS 2020*, volume 151, pages 5:1–5:32. LIPIcs, January 2020.

BLS19. Jonathan Bootle, Vadim Lyubashevsky, and Gregor Seiler. Algebraic techniques for short(er) exact lattice-based zero-knowledge proofs. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 176–202. Springer, Cham, August 2019.

BN20. Carsten Baum and Ariel Nof. Concretely-efficient zero-knowledge arguments for arithmetic circuits and their application to lattice-based cryptography. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 495–526. Springer, Cham, May 2020.

BS23. Ward Beullens and Gregor Seiler. LaBRADOR: Compact proofs for R1CS from module-SIS. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 518–548. Springer, Cham, August 2023.

CNR+22. Matteo Campanelli, Anca Nitulescu, Carla Ràfols, Alexandros Zacharakis, and Arantxa Zapico. Linear-map vector commitments and their practical applications. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part IV*, volume 13794 of *LNCS*, pages 189–219. Springer, Cham, December 2022.

ENS20. Muhammed F. Esgin, Ngoc Khanh Nguyen, and Gregor Seiler. Practical exact proofs from lattices: New techniques to exploit fully-splitting rings. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 259–288. Springer, Cham, December 2020.

FMRV22. Thibauld Feneuil, Jules Maire, Matthieu Rivain, and Damien Vergnaud. Zero-knowledge protocols for the subset sum problem from MPC-in-the-head with rejection. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part II*, volume 13792 of *LNCS*, pages 371–402. Springer, Cham, December 2022.

FR23. Thibauld Feneuil and Matthieu Rivain. Threshold linear secret sharing to the rescue of MPC-in-the-head. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part I*, volume 14438 of *LNCS*, pages 441–473. Springer, Singapore, December 2023.

FR25a. Thibauld Feneuil and Matthieu Rivain. CAPSS: A framework for SNARK-friendly post-quantum signatures. Cryptology ePrint Archive, Report 2025/061, 2025.

FR25b. Thibauld Feneuil and Matthieu Rivain. Threshold computation in the head: Improved framework for post-quantum signatures and zero-knowledge arguments. *Journal of Cryptology*, 38(3):28, July 2025.

GLS+23. Alexander Golovnev, Jonathan Lee, Srinath T. V. Setty, Justin Thaler, and Riad S. Wahby. Brakedown: Linear-time and field-agnostic SNARKs for R1CS. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part II*, volume 14082 of *LNCS*, pages 193–226. Springer, Cham, August 2023.

HJ25. Janik Huth and Antoine Joux. VOLE-in-the-head signatures from subfield bilinear collisions. In Goichiro Hanaoka and Bo-Yin Yang, editors, *ASIACRYPT 2025, Part IV*, volume 16248 of *LNCS*, pages 3–34. Springer, Singapore, December 2025.

IKOS07. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 21–30. ACM Press, June 2007.

KKW18. Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 525–537. ACM Press, October 2018.

Lee21. Jonathan Lee. Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part II*, volume 13043 of *LNCS*, pages 1–34. Springer, Cham, November 2021.

LNP22. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 71–101. Springer, Cham, August 2022.

LNS21. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Shorter lattice-based zero-knowledge proofs via one-time commitments. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 215–241. Springer, Cham, May 2021.

LNSW13. San Ling, Khoa Nguyen, Damien Stehlé, and Huaxiong Wang. Improved zero-knowledge proofs of knowledge for the ISIS problem, and applications. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 107–124. Springer, Berlin, Heidelberg, February / March 2013.

OTX24. Ying Ouyang, Deng Tang, and Yanhong Xu. Code-based zero-knowledge from VOLE-in-the-head and their applications: Simpler, faster, and smaller. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part V*, volume 15488 of *LNCS*, pages 436–470. Springer, Singapore, December 2024.

Sta21. StarkWare. ethSTARK documentation. Cryptology ePrint Archive, Report 2021/582, 2021.

VP19. Alexander Vlasov and Konstantin Panarin. Transparent polynomial commitment scheme with polylogarithmic communication complexity. Cryptology ePrint Archive, Report 2019/1020, 2019.

ZCF24. Hadas Zeilberger, Binyi Chen, and Ben Fisch. BaseFold: Efficient field-agnostic polynomial commitment schemes from foldable codes. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 138–169. Springer, Cham, August 2024.

A Summary of the Zero-Knowledge Tweaks

Three leakage sources are tackled by our tweaks:

- *In the PCS construction.* Opening a PCS into an evaluation point e consists of opening the underlying LVCS for the coefficient tuple $(1, e, \ldots, e^{n_{\text{rows}}})$. By denoting $\boldsymbol{v}^{(e)}$ the underlying linear combination opended through the LVCS, the polynomial evaluation is deduced as $P(e) = \langle \boldsymbol{v}, (1, e^{\mu}, \ldots, e^{(\nu-1)\mu}) \rangle$. However, $\boldsymbol{v}$ should not leak information about P and to proceed, one needs to commit few additional rows in the LVCS that contains randomness (see the coefficients $\{r_{j,i,k}\}_{j,i,k}$ in Figure 3).
- *In the LVCS construction.* Each row is interpolated as a polynomial which is committed using the DECS protocol (or Merkle trees with Ligero's proximity test for Ligero-PCS). Then the LVCS opens those polynomials into ℓ random points. To avoid leakage, the polynomials should be interpolated with enough randomness (see the coefficients $\bar{\boldsymbol{r}}_1, \ldots, \bar{\boldsymbol{r}}_{n_{\text{rows}}}$ in Figure 2).
- *In the DECS (or Merkle trees with Ligero's proximity test for Ligero-PCS).* To ensure that the committed polynomials are of the right degree, one reveals random linear combination of those polynomials. We mask the result of these linear combinations by additional random polynomials (see the polynomials $M_1, \ldots, M_\eta$ in Figure 1). Moreover, we add random tapes in hash computation of the Merkle leaves (see the tapes $\rho_1, \ldots, \rho_N$ in Figure 1) and, for each revealed leaf, we need to reveal the corresponding tape.

B Technical Flaws in [BCGZ24]

The recent work of [BCGZ24] also proposes exact MPCitH-based arguments for lattices. However, this work suffers the following technical flaws.

1. First of all, the [BCGZ24]'s underlying MPC protocol uses non-linear operations, but the [BCGZ24]'s soundness proof is inspired to the soundness proof of [FR23] which only supports linear operations. The authors do not explain how to adapt the proof for non-linear operations, therefore the soundness proof does not hold as is because there is no guarantee that the committed sharings are valid Shamir's secret sharings. To be precise, the issue in their security proof comes in the extract

   > "Since $succ_{P^*}$ occurs, we have $C(v_i) = 1$. Then for each set $J \in \mathcal{J}$ (w^J is a bad witness) such that $J \subset H$ (the parties in J are honest), we have $V_i = \hat{V}_i$ for every $i \in J$, which implies $C(\hat{v}_i) = C(v_i) = 1$. Namely, a bad witness ω^J gives rise to a wrong result (for a bad witness ω outputs $C(x, \omega) = 1$)."

   Unfortunately, some quantities are not properly defined in [BCGZ24] such as $C(v_i)$ and $C(\hat{v}_i)$. v_i probably refers to the broadcast message of the i^{th} party (assuming v_i and V_i refer to the same quantity), but C does not seem to be defined. Taking back the original [FR23]'s proof, we could guess that C refers to the verification function performed by the verifier on the broadcast message ("g" using the notation of [FR23]), however this function should be executed over the broadcast values, and not to its shares. We would need to have $C(v) = 1$ instead of $C(v_i) = 1$, where v is the reconstructed value from $(v_1, \ldots, v_N)$. This extract should be the place where the fact that the MPC protocol perform only linear operations is important. In the [FR23]'s proof, it uses the fact that two degree-d sharings are consistent on $d+1$ shares, so the shared values are the same. Since [BCGZ24] uses quadratic operations in the MPC protocol, an adaptation of the [FR23] would lead to a degree-d sharing and a degree-$2d$ sharing that match over $d+1$ shares, but it does not enable us to conclude something about the two shared values.

   Supporting non-linear operations in the MPC protocol in MPCitH-based schemes is possible, but it requires to use additional techniques such as proximity tests [AHIV17] or degree-enforcing commitments [FR25b].

2. Then, the [BCGZ24] proof system does not seem zero-knowledge (even if it is supposed to be). To compute the multiplication gate $z = x \cdot y$, the prover computes the product of two degree-d Shamir's secret sharings that correspond to the two operands, obtaining a degree-$2d$ sharing $[\![z]\!]_{(2d)} := [\![x]\!]_{(d)} \cdot [\![y]\!]_{(d)}$ of the multiplication output, then the prover subtracts the resulting sharing with another degree-d sharing of the multiplicaton output $[\![z]\!]_{(d)}$. The resulting substracted sharing

   $$[\![u]\!]_{(2d)} := [\![z]\!]_{(2d)} - [\![z]\!]_{(d)}$$

   is revealed to the verifier, which will check that it is indeed a degree-$2d$ sharing of zero. This operation aims to compute a degree-d sharing of the multiplication z while ensuring that it is consistent with the sharings of x and y. However, we can observe that revealing $[\![u]\!]_{(2d)}$ to the verifier leaks information about the secret values x, y and z. The high-degree terms of $[\![u]\!]_{(2d)}$, which are the same as those of $[\![x]\!]_{(d)} \cdot [\![y]\!]_{(d)}$, depends on the shared values x and y. It is of course possible to mitigate this information leakage using masks, but it will increase the produced proof sizes.

3. Finally, the authors use the proof system to prove knowledge of some private key for FrodoKEM. However, the FrodoKEM arithmetic is not over a field ($\mathbb{Z}_{2^{15}}$ or $\mathbb{Z}_{2^{16}}$), and the [BCGZ24]'s security proofs implicitly assume that the underlying arithmetic is a field (for example, in the Schwartz-Zippel lemma). One can support $\mathbb{Z}_{q^m}$ rings in such proof systems, but it requires to use non-trivial techniques such as those in [BDJ+23].